Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Elisabeth Enerhaug

Algorithms For Solving The Learning With Errors Problem

Master's thesis in Mathematical Sciences
Supervisor: Professor Kristian Gjøsteen
June 2021

NTNU
Norwegian University of
Science and Technology

Elisabeth Enerhaug

Algorithms For Solving The Learning With Errors Problem

Master's thesis in Mathematical Sciences
Supervisor: Professor Kristian Gjøsteen
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Abstract

This thesis looks at four different algorithms for solving the Learning with Errors (LWE) problem.

The first algorithm, BKW, is comparable to Gaussian elimination. Then, we show that LWE can be reduced to a Closest Vector Problem (CVP) in a lattice. Consequently, we look at two algorithms for lattice basis reduction: LLL and BKZ. For all these three algorithms we investigate how they work, their complexity and some properties that come as a result.

Lastly, we look at how we can use lattices and CVP to solve a problem in the ring version of LWE. We use the round-off algorithm for CVP to find a short generator of a principal ideal, given a long generator.

Samandrag

Denne avhandlinga ser på fire ulike algoritmar for å løyse "Learning with Errors" (LWE, Læring med Feil) problemet.

Den første, BKW-algoritmen, kan samanliknas med gaussisk eliminasjon. Deretter viser vi at LWE kan reduserast til eit "Closest Vector Problem" (CVP, Nærmaste Vektor Problem) i eit gitter. Derfor ser vi på to basisreduksjonsalgoritmar for gitter: LLL og BKZ. For alle desse tre algoritmane ser vi på korleis dei fungerer, kompleksiteten og eventuelle eigenskapar som skulle følgje.

Til sist ser vi korleis vi kan bruke gitter til å løyse eit problem i ringversjonen av LWE. Vi bruker avrundingsalgoritmen for CVP for å finne ein kort generator for eit hovudideal, gitt ein lang generator.

Acknowledgements

Writing a thesis in the middle of a pandemic is certainly not an easy undertaking. And the fact that it has been such a challenge means there is a lot of people I owe a lot of gratitude to, and who deserve the (albeit small) honour of being mentioned at the beginning of this thesis.

First and foremost, thank you to my supervisor, Professor Kristian Gjøsteen, for suggesting this topic, and for providing me with guidance and advice every step of the way.

Next, I want to thank the madhouse that was study room 393c. Thank you for the laughter, the coffee breaks and the support to Endre, Christina, Katrine, Johannes, Kristoffer, and Ole.

Thesis writing can be disheartening at times. Therefore, I would like to thank all of my friends who doubled as armchair psychologists and private cheerleading squad this year. Especially thanks to Johanna Magdalena Husebye for the phone calls, to Anna Bakkebø for the walks, and to Ailsa Robertson for the rants. It has kept me sane.

Lastly, I owe the biggest of thanks to my parents, Jakob and Møyfrid, and sisters, Hanna and Solveig, for simply being my family. Although little of what follows will make sense to you, I could not have done it without all of your unwavering support. I love you.

Contents

1 Introduction

As quantum computers are threatening to break more classical public key encryption schemes, like RSA and Diffie-Hellman, we are looking to find cryptosystems that are harder to break for such machines. Over the past several years, more and more cryptosystems are relying on lattices. One reason for this is that lattices are thought to be robust against quantum computers. In 2005, Regev introduced the Learning with Errors (LWE) problem and a public key encryption scheme based on it ([Reg05]). Regev proved that solving the LWE problem, and hence breaking his cryptosystem, is as hard as certain worst-case lattice problems, which implies it is difficult for quantum computers to break.

The idea behind LWE can be explained using a matrix equation. Put simply, let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{s} \in \mathbb{Z}_q^n$ be randomly sampled where q is a prime and $n \leq m$. From some error distribution χ, sample $\boldsymbol{\nu} = (\nu_1, ..., \nu_n) \leftarrow \chi$. Then, given $(\mathbf{A}, \boldsymbol{t} = \mathbf{A}^T\mathbf{s} + \boldsymbol{\nu} \pmod q)$, can we find $\mathbf{s}$? This is called the *search-LWE problem.* In Regev's public key encryption scheme, $(\mathbf{A}, \boldsymbol{t})$ is the public key and $\mathbf{s}$ is the secret key.

In this thesis we will look at four different algorithms that aims to solve the LWE problem. The standard way of trying to solve the LWE problem is by using lattices. However, we begin by analysing a non-lattice algorithm, BKW, to show that it is possible to solve LWE using a non-lattice based algorithm. The BKW algorithm uses an approach similar to Gaussian elimination in order to eliminate blocks in the rows of $\mathbf{A}$.

In Chapter 5, we introduce the concept of lattices and show that the LWE problem for $\mathbf{s}$ can be reduced to a Closest Vector Problem (CVP). There are several algorithms for solving a CVP in a lattice, but they work best on lattice bases that have short vectors, ordered roughly according to length and that are as close to orthogonal as possible. The LLL algorithm (Chapter 6) and BKZ algorithm (Chapter 7) are both algorithm that

aims to reduce a basis to achieve this. The LLL algorithm is incredibly versatile, and have applications far beyond basis reduction. The BKZ algorithm is in many ways a generalisation for the LLL algorithm and gives a better search bound. However, while the LLL algorithm can be proven to terminate in polynomial time, this can not be proved for BKZ. For all of these three algorithms we look at how and why they work, study their complexity and potential trade-offs to be considered. For the lattice algorithms, we also look at some nice properties of the reduced basis they return.

Up until this point, we will have looked at lattices with very little assumed properties. In Chapter 8, we will look at lattices with a lot more algebraic structure; principal ideal lattices. These are used in the ring version of LWE, called Ring-LWE og RLWE. We will not go into detail about Ring-LWE, but instead look at a common problem within Ring-LWE, called the *Short Generator Principal Ideal Problem*, or SG-PIP for short. The SG-PIP, put simply is: For a ring R and a principal ideal (g), where g is a short generator. If we are given a long generator g', can we find a generator that is short enough? This might not look like an LWE problem, but we will show how it can be transformed into a CVP for lattice and how we can apply the round-off algorithm to solve it.

2 Notation

This thesis follows the standard convention of denoting vectors as bold lower case letters, e.g. $\boldsymbol{a}$, and matrices as bold upper case letters, e.g. $\mathbf{A}$. Vectors are treated as column vectors.

The rounding function $\lfloor a \rceil$ for $a \in \mathbb{R}$ refers to the closest integer to a. I.e. $\lfloor a \rceil = \lfloor a + \frac{1}{2} \rfloor$. For a vector $\boldsymbol{a} = (a_1, ..., a_k) \in \mathbb{R}^k$, $\lfloor \boldsymbol{a} \rceil = (\lfloor a_1 \rceil, ..., \lfloor a_k \rceil)$.

We will also use standard notation for inner products and norm. That is, $\langle \boldsymbol{a}, \mathbf{b} \rangle = \sum_{i=1}^{k} a_i \overline{b_i}$. Note that for real vectors, this is equivalent to the dot product. The norm $\|\boldsymbol{a}\| = \langle \boldsymbol{a}, \boldsymbol{a} \rangle^{\frac{1}{2}}$ will denote the standard Euclidean norm. Any other norm will be denoted by a subscript, e.g. $\|\boldsymbol{a}\|_1$.

3 Learning With Errors and Regev's Cryptosystem

3.1 Cryptography Basics

This project assumes the reader has a good understanding of the cryptography basics. Still, in order to quickly grasp the ideas, a short summary of public key encryption is given here. It is based on Gjøsteen's lecture notes in Cryptography ([Gjø19a] and [Gjø19b]).

3.1.1 Alice, Bob and Eve - The Basic Idea

The fundamental idea behind cryptography is this: Two people, Alice and Bob, wants to exchange messages without a third person, Eve - the adversary, being able to read them. In order to do this, they encrypt their messages using a *cryptosystem*. A cryptosystem *encrypts* a message, m, creating a cipher, c, before sending it. It also *decrypts* c back to m. The goal is that only Alice and Bob can encrypt and decrypt messages, and make it as hard as possible for Eve to do the same.

Definition 3.1. A *symmetric cryptosystem* consist of

- a set $\mathcal{K}$ of *keys*,
- a set $\mathcal{P}$ of *plaintexts*,
- a set $\mathcal{C}$ of *ciphertexts*,
- an *encryption algorithm* $\mathcal{E}(k, m)$ that takes a key $k \leftarrow \mathcal{K}$ and a message $m \leftarrow \mathcal{P}$ as input and outputs a ciphertext c,
- a *decryption algorithm* $\mathcal{D}(k, c)$ that takes a key $k \leftarrow \mathcal{K}$ and a ciphertext c and outputs either a plaintext or $\perp$.

For any key k and any plaintext m we have that $\mathcal{D}(k, \mathcal{E}(k, m)) = m$.

For a cryptosystem to work, it is essential that the key used for encryption

and decryption is kept secret from Eve. As long as a secret key is established, Alice and Bob can send encrypted messages to each other with a low probability of Eve being able to read them.

3.1.2 Public Key Encryption

Establishing a shared secret between two people takes time and space. For various reasons it might not be practical, or even possible. Consider a situation where Bob wants to receive messages from multiple people, not just Alice. It is unpractical to keep track of all the individual keys. Instead of a key exchange protocol, he can use *public key encryption*.

Definition 3.2. A public key encryption scheme consists of three algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$:

- The key generation algorithm $\mathcal{K}$ takes no input and outputs an encryption key ek and a decryption key, dk. To each encryption key ek there is an associated message set, M_{ek}.
- The encryption algorithm $\mathcal{E}$ takes an encryption key ek and a message $m \in M_{ek}$ as input and outputs a ciphertext c.
- The decryption algorithm $\mathcal{D}$ takes the decryption key dk and cipher text c as input and outputs either the message m or $\perp$, which indicates encryption failure.

We require for any key pair $(ek, dk) \leftarrow \mathcal{K}$ and for any message $m \in M_{ek}$ that $\mathcal{D}(dk, \mathcal{E}(ek, m)) = m$.

Put simply, Bob generates a key set $(ek, dk) \leftarrow \mathcal{K}$ and makes the encryption key ek public. Alice use the public key to encrypt her message $c = \mathcal{E}(ek, m)$ and send it to Bob. Bob then decrypts the message using his secret key $m = \mathcal{D}(dk, c)$. See Figure 1.

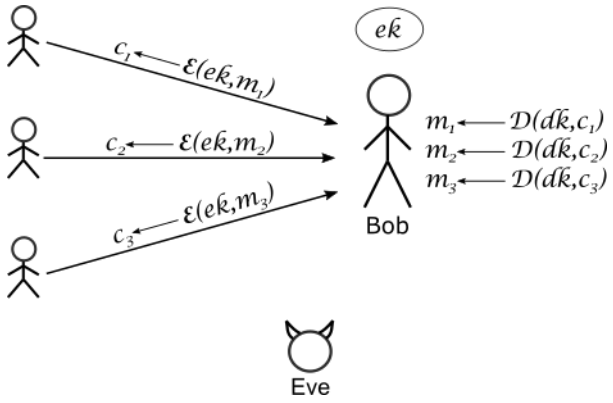


Figure 1: Public Key Encryption.

3.2 The Learning With Errors Problem

The Learning with Errors (LWE) problem was first introduced by Regev in 2005 ([Reg05]). It is a natural extension of the Learning Parity with Noise(LPN) problem that has existed in Computer Science and Cryptography for a long time. The LPN problem is simply LWE for $q = 2$ (we will see what this means below). In fact, algorithms like BKW (see Chapter 4) were first invented for LPN and then extended to LWE. This section is loosely based on [Gjø19b, Sec 7.2] and [LP11, Section 2].

For a large prime q, sample uniformly randomly $\boldsymbol{a}_i$'s from $\mathbb{Z}_q^n$ for $i = 1, ..., m$. Then, for a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$, calculate $t_i = \langle \boldsymbol{a}_i, \mathbf{s} \rangle \pmod q$ for $1 \leq i \leq m$. This system can be expressed in terms of the matrix equation $\boldsymbol{t} = \mathbf{A}^T\mathbf{s} \pmod q$, where $\mathbf{A} = (\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_m) \in \mathbb{Z}_q^{n \times m}$ and $\boldsymbol{t} = (t_1, ...t_m) \in \mathbb{Z}_q^m$. As long as $m \geq n$ this system is clearly solvable for $\mathbf{s}$ by simply using Gaussian elimination.

To make this system hard to solve, we add an error ν_i to each calculation:

$$\boldsymbol{a}_i \leftarrow \mathbb{Z}_q^n, \qquad\qquad t_i = \langle \mathbf{s}, \boldsymbol{a}_i \rangle + \nu_i \ \in \mathbb{Z}_q$$

The error is sampled randomly from some error distribution χ, e.g. the discrete Gaussian distribution. The updated matrix equation is now

$$\boldsymbol{t} = \mathbf{A}^T\mathbf{s} + \boldsymbol{\nu},$$

where $\boldsymbol{\nu}^T = (\nu_1, ..., \nu_m) \leftarrow \chi$. Then the two types of the LWE problem reads as follows:

Definition 3.3. Let q be a large prime. Sample $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$ and $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly randomly, where $m \geq n$. From some error distribution χ, sample $\boldsymbol{\nu} \leftarrow \chi$. Calculate $\boldsymbol{t} = \mathbf{A}^T\mathbf{s} + \boldsymbol{\nu} \pmod q$. Then the two different LWE problems are:

- *The search-LWE problem*: Given $(\mathbf{A}, \boldsymbol{t})$, can we find $\mathbf{s}$?

- *The decision-LWE problem*: Given $(\mathbf{A}, \boldsymbol{t})$ constructed as described, can we distinguish between this pair and a uniformly random $(\tilde{\mathbf{A}}, \tilde{\boldsymbol{t}})$?

In this thesis we will be focusing on the search version and simply refer to it as the LWE problem throughout this thesis. Regev showed in his paper that the two problems are equivalent as long as q is bounded above by some polynomial in n. He also showed that for appropriately chosen q and χ, solving the LWE was as hard as solving a worst-case lattice problems*. Based on the LWE problem, Regev also constructed a public key encryption scheme, which we will discuss in the next section. Note that part of what we mean by choosing an appropriate error distribution is that the error $\boldsymbol{\nu}$ should not be "too large". We will explain what we mean by this later. In his paper, Regev used the wrapped rounded Gaussian centred at zero to show his results.

Lastly, before we move on to the next section, we will note that in this thesis we will be working over $\mathbb{Z}_q$, for a prime q which is standard. But it can be generalised to work on $\mathbb{F}_q$, a finite field of size q, since $\mathbb{F}_q \cong \mathbb{Z}_q$ [See e.g. BJN94, Theorem 16.4.2].

*Specifically the quantum hardness of worst case lattice problems such as GapSVP and SIVP.

3.3 Regev's Cryptosystem

In order to construct Regev's cryptosystem, we assume we have unrestricted access to an LWE oracle:

Definition 3.4. Let n, q be positive integers where q is a prime. A *Learning with Errors oracle* $\Pi_{\mathbf{s},\chi}$ for a secret vector $\mathbf{s} \in \mathbb{Z}_q^n$ and a probability distribution χ is an oracle returning

$$\left(\boldsymbol{a} \xleftarrow{U} \mathbb{Z}_q^n, \langle \boldsymbol{a}, \mathbf{s} \rangle + \nu\right),$$

where $\nu \leftarrow \chi$.

Now we describe Regev's public key encryption scheme.

Key generation algorithm

A public key encryption scheme requires two keys, ek and dk. First, start by sampling a random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$. This will be Bob's secret key, that is $dk = \mathbf{s}$. We use this to construct the public key. Since we have unrestricted access to an LWE oracle $\Pi_{\mathbf{s},\chi}$, the key generation algorithm samples m samples $(\boldsymbol{a}, t) \leftarrow \Pi_{\mathbf{s},\chi}$ such that there are n linearly independent $\boldsymbol{a}$'s. As described earlier, these samples can be combined into a matrix equation

$$\boldsymbol{t} = \mathbf{A}^T\mathbf{s} + \boldsymbol{\nu} \pmod{q}.$$

where Bob then outputs the public key $ek = (\mathbf{A}, \boldsymbol{t})$.

Encryption algorithm

Alice now wants to send a message to Bob. The message τ is limited to $\tau \in \{0, 1\}$. To encrypt her message, she first samples a short, random vector $\boldsymbol{r} \leftarrow \mathbb{Z}_q^m$, then she uses the public key to calculate her ciphertext thus:

$$\boldsymbol{x} = \mathbf{A}\boldsymbol{r}, \quad w = \boldsymbol{r}^T\boldsymbol{t} + \tau \left\lfloor \frac{q}{2} \right\rfloor$$

Encryption outputs the ciphertext $c = (\boldsymbol{x}, w)$.

Decryption algorithm

Bob receives the ciphertext $c = (\boldsymbol{x}, w)$. He uses his secret key $\mathbf{s}$ to calculate:

$$\begin{aligned}\eta = w - \boldsymbol{x} \cdot \mathbf{s} &= \boldsymbol{r}^T \boldsymbol{t} + \tau \left\lfloor \frac{q}{2} \right\rfloor - (\mathbf{A}\boldsymbol{r})^T \mathbf{s} \\ &= \boldsymbol{r}^T (\mathbf{A}^T \mathbf{s} + \boldsymbol{\nu}) + \tau \left\lfloor \frac{q}{2} \right\rfloor - \boldsymbol{r}^T \mathbf{A}^T \mathbf{s} \\ &= \boldsymbol{r}^T \boldsymbol{\nu} + \tau \left\lfloor \frac{q}{2} \right\rfloor\end{aligned}$$

Bob does not know $\boldsymbol{r}$, but he does know that both $\boldsymbol{r}$ and $\boldsymbol{\nu}$ are short vectors, while q is a large prime. We assume $\langle \boldsymbol{r}, \boldsymbol{\nu} \rangle < \left\lfloor \frac{q}{2} \right\rfloor$. This means that if $|\eta| > \left\lfloor \frac{q}{2} \right\rfloor$, $\tau = 1$ and if $|\eta| < \left\lfloor \frac{q}{2} \right\rfloor$, $\tau = 0$.

Alice and Bob have now established a way to encrypt and decrypt messages, but what does Eve see? Obviously, Eve sees the public key $(\mathbf{A}, \boldsymbol{t})$, she also sees Alice's ciphertext $(\boldsymbol{x}, w)$. Is there a way for her to break this system by finding $\mathbf{s}$?

From the adversary's perspective this is clearly a LWE problem. So the security of Regev's system relies on the hardness of LWE, i.e. quantum-hard. It is also a nice cryptosystem in other ways, like being much more efficient than previous lattice based cryptosystems. Before, public keys were of size $O(n^4)$ and encrypting messages increased its size with a factor of $O(n^2)$. For Regev's cryptosystem it is $O(n^2)$ and $O(n)$ respectively, which is a lot less Lindner and Peikert have later described a modified LWE cryptosystem that requires even smaller keys ([LP11]).

Now that we have an understanding of the problem and cryptosystem at the core of this thesis, we can start to look at algorithms that aims to solve the LWE problem. The standard approach is to use lattices. This makes sense since lattices are thought to be robust against quantum computers and the hardness of LWE is linked closely to worst-case lattice problems. Therefore, most of this thesis will be centred around lattices and lattice algorithms. However, we start of with the BKW algorithm, which is a

non-lattice algorithm that uses Gaussian elimination. This shows that it is possible to take a non-lattice approach.

4 BKW - Gaussian elimination for LWE

Named, like so many other algorithms, after its creators, the Blum-Kalai-Wasserman (BKW) algorithm was first developed as a tool to solve the LPN problem but has since been expanded as a method for solving LWE.

This chapter we will follow Duc, Tramèr, and Vaudenay's article ([DTV15]), and also use some material from Albrecht et al. ([Alb+15]).

Note that several times in this chapter, we will refer to only a part, or block, of a vector. That is for $\boldsymbol{a} = (a_1, ..., a_k)$, $\boldsymbol{a}_{[i,j]} = (a_i, ..., a_j)$ for $1 \leq i < j \leq k$.

We will start with going through the basic idea behind the algorithm.

Assume $\mathbf{A}$ is an $n \times n$ invertible matrix. $\boldsymbol{t} = \mathbf{A}^T\mathbf{s} + \boldsymbol{\nu}$ is the system of equations:

$$\begin{pmatrix} t_1 \\ \vdots \\ t_n \end{pmatrix} = \begin{pmatrix} * & \dots & * \\ \vdots & \ddots & \vdots \\ * & \dots & * \end{pmatrix} \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_n \end{pmatrix}. \tag{4.1}$$

If $\boldsymbol{\nu} = \mathbf{0}$, we could solve this as a system of linear equations by using Gaussian elimination. That is, (1.) use row operations on $\mathbf{A}^T$ to reduce the coefficient matrix into row echelon form, (2.) recover one unknown value in $\mathbf{s}$, and lastly (3.) substitute this back up the system.

The idea is similar for for BKW. Imagine, for $\boldsymbol{\nu} \neq \mathbf{0}$, we used row reductions to get the system (4.1) in the form:

$$\begin{pmatrix} t'_1 \\ \vdots \\ t'_n \end{pmatrix} = \begin{pmatrix} * & \dots & * \\ \vdots & \ddots & \vdots \\ 0 & \dots & * \end{pmatrix} \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} \nu'_1 \\ \vdots \\ \nu'_n \end{pmatrix},$$

where $\mathbf{A}^T$ is now in row echelon form. *If* ν_n is sufficiently small, it is possible to have a reasonably good guess at what s_n is. This guess could be made better by having more systems of equations of this form for the same $\mathbf{s}$ (since we have unlimited access to the LWE oracle). This way we

could check that our guess for s_n is gives the desired result in more than one instance. However, it is not a given that we will be able to find a good guess for s_n. Each row operation, say $t_1 + t_2 = \langle \boldsymbol{a}_1 + \boldsymbol{a}_2, \mathbf{s} \rangle + (\nu_1 + \nu_2)$, has an error term potentially twice as big as those previously. If we multiplied any of rows with a scalar this would increase the error even more. Seeing as the n^{th} row is most likely what requires most row operations, each reduction significantly decreases the likelihood of distinguishing s_n, if not rendering it completely impossible.

The BKW algorithm aims to minimize the number of row operations performed (and hence minimize the noise increase) by taking advantage of our unlimited access to the LWE oracle $\Pi_{\mathbf{s},\chi}$ (see Definition 3.4). For some integer $1 \leq \beta < n$, the idea is to find samples $(\boldsymbol{a}, t), (\boldsymbol{a}^*, t^*) \leftarrow \Pi_{\mathbf{s},\chi}$ where $\boldsymbol{a}_{[1,\beta]} = \pm \boldsymbol{a}^*_{[1,\beta]}$, which means $\boldsymbol{a}_{[1,\beta]} \mp \boldsymbol{a}^*_{[1,\beta]} = \mathbf{0}$. This way we have eliminated β elements, by only one "row operation", but the new error term is only the sum of two terms: $\nu \mp \nu^*$. Repeating this for the next block of non-zero elements, and so on until we only have a small number of non-zero elements left, say $d \leq \beta$ elements. On these elements we can perform an exhaustive search over $\mathbb{Z}_q$ to find the last d elements of $\mathbf{s}$, granted the noise is sufficiently small. Again, creating several systems like this on which we can test our hypothesis for the last elements of $\mathbf{s}$ will increase the probability of finding it.

There is clear trade-off to be considered in this algorithm. In order to reduce the increase of noise, we want to make as few block reductions as possible, meaning we want large blocks and a small number of reductions. This, however, requires a lot more computational power. Seeing as the worst case scenario for getting two samples $\boldsymbol{a}_1, \boldsymbol{a}_2$ where the first β block of elements match, is sampling $\frac{q^\beta - 1}{2}$ elements from $\Pi_{\mathbf{s},\chi}$ (this is shown in Lemma 4.2). Clearly, for large q, a large β will increase the number of searches and samples considerably. On the other hand, having small blocks and lots of repetition, will reduce the number of samples required

from the oracle, but will significantly increase our error term rendering our exhaustive search at the end of the reduction a lot more difficult, if not impossible.

4.1 The Algorithm

Now that we have an idea how what the algorithm aims to do, we will go through the different stages in more detail. We will also give an estimate of how many samples r that is required for a "reasonably good" guess of $\mathbf{s}$.

The three stages of BKW are:

1. Sample reduction
2. Hypothesis testing
3. Back substitution

The algorithm takes two positive integers as input, α, β, where $\alpha\beta \leq n$. Where β is the size of the blocks of the elements we want reduce for each round of row operations and α is the number of times we do this reduction. Let $d = n - (\alpha - 1)\beta \leq \beta$ be the size of the very last block. Now we are going to look at the three stages of the BKW algorithm.

4.1.1 Sample Reduction

The first stage of the BKW algorithm is called *sample reduction*. Sample reduction is comparable to row reduction in solving a system of linear equations.

The sample reduction algorithm is set up as a system of BKW-oracles $\mathcal{A}_{\mathbf{s},\chi,l}$, $0 < l < \alpha$. $\mathcal{A}_{\mathbf{s},\chi,l}$ outputs samples $(\boldsymbol{a}, t = \langle \boldsymbol{a}, \mathbf{s} \rangle + \nu)$ where $\boldsymbol{a} \in \mathbb{Z}_q^n$, and the first $l\beta$ elements of $\boldsymbol{a}$ is zero and $\nu \leftarrow 2^l\chi$. We will now look at how we construct such oracles.

Clearly, $\mathcal{A}_{\mathbf{s},\chi,0} = \Pi_{\mathbf{s},\chi}$. We exploit this and our unlimited access to $\Pi_{\mathbf{s},\chi}$ to create $\mathcal{A}_{\mathbf{s},\chi,1}$ (and consequently the other $\mathcal{A}_{\mathbf{s},\chi,l}$'s).

Because we have unlimited access to samples from $\Pi_{\mathbf{s},\chi}$, the aim is to obtain two samples $(\boldsymbol{a}, t), (\boldsymbol{a}^*, t^*) \leftarrow \mathcal{A}_{\mathbf{s},\chi,0} = \Pi_{\mathbf{s},\chi}$ such that $\boldsymbol{a}_{[1,\beta]} = \pm\boldsymbol{a}^*_{[1,\beta]}$. In order to do this, we construct a table, T_1, that stores samples from $\mathcal{A}_{\mathbf{s},\chi,0}$. When we call $\mathcal{A}_{\mathbf{s},\chi,1}$, the oracle samples $(\boldsymbol{a}, t) \leftarrow \mathcal{A}_{\mathbf{s},\chi,0}$ and checks if there exist a match $(\boldsymbol{a}^*, c^*) \in T_1$ such that $\boldsymbol{a}_{[1,\beta]} = \pm\boldsymbol{a}^*_{[1,\beta]}$. If such a match exist in T_1, $\mathcal{A}_{\mathbf{s},\chi,1}$ outputs $(\boldsymbol{a} \mp \boldsymbol{a}^*, t \mp t^*)$. Clearly, $t \mp t^* = \langle \boldsymbol{a} \mp \boldsymbol{a}^*, \mathbf{s} \rangle + (\nu \mp \nu^*)$ where $(\nu \mp \nu^*) \leftarrow 2^1\chi$. If not, the oracle puts $(\boldsymbol{a}, t)$ into the table and obtains a new sample from $\mathcal{A}_{\mathbf{s},\chi,0}$ to check for a new match. It repeats this until a match is found.

The procedure is the same for a general l, $1 \leq l \leq \alpha - 1$. Each oracle $\mathcal{A}_{\mathbf{s},\chi,l}$ has a corresponding table T_l in which it stores samples from $\mathcal{A}_{\mathbf{s},\chi,l-1}$. Then, when we want to obtain a sample from $\mathcal{A}_{\mathbf{s},\chi,l}$, $\mathcal{A}_{\mathbf{s},\chi,l}$ recursively calls the oracles $\mathcal{A}_{\mathbf{s},\chi,i}$, for $i = l-1, ..., 1, 0$, to obtain a sample $(\boldsymbol{a}, t)$ where the first $\beta(l-1)$ entries of $\boldsymbol{a}$ is zero. Then we check for a match $(\boldsymbol{a}^*, t^*) \in T_l$ such that $\boldsymbol{a}_{[(l-1)\beta+1,l\beta]} = \pm\boldsymbol{a}^*_{[(l-1)\beta+1,l\beta]}$. If this match exist, $\mathcal{A}_{\mathbf{s},\chi,l}$ outputs $(\boldsymbol{a} \mp \boldsymbol{a}^*, t \mp t^*)$, with an error from $2^l\chi$. If not, $(\boldsymbol{a}, t)$ is stored in T_l and $\mathcal{A}_{\mathbf{s},\chi,l}$ recursively calls $\mathcal{A}_{\mathbf{s},\chi,i}$ again.

An easier way to understand this might be the more mechanical description: We start by sampling a sample $(\boldsymbol{a}, t)$ from $\mathcal{A}_{\mathbf{s},\chi,0} = \Pi_{\mathbf{s},\chi}$. Then we check for a match in T_1 and calculate the new value, then then the same for T_2, T_3 all the way up to T_{l-1}. If, at any point, we do not find a match in one of the tables, we add that sample to the table and start over again with a new sample from $\Pi_{\mathbf{s},\chi}$. Note that if one of the β-blocks are only zeros, we do not need a match for it, it goes straight to "the next level."

$\mathcal{A}_{\mathbf{s},\chi,\alpha-1}$ outputs samples $(\boldsymbol{a}, t)$, where $\boldsymbol{a} = (0, \ldots, 0, a_{(\alpha-1)\beta+1}, \ldots, a_n)$. The final block has size $n - (\alpha-1)\beta \leq \beta$. For how we approach the final oracle $\mathcal{A}_{\mathbf{s},\chi,\alpha}$ of the sample reductions the approaches differ.

- In the original BKW algorithm, designed for LPN, the final oracle $\mathcal{A}_{\mathbf{s},\chi,a}$ would sample from $\mathcal{A}_{\mathbf{s},\chi,a-1}$ reducing so that only one element, a_n, is non-zero.
- In the updated version for the BKW algorithm for LWE, [Alb+15], generalises this by choosing a parameter $d \leq n - (\alpha - 1)\beta$. Then sample from $\mathcal{A}_{\mathbf{s},\chi,a-1}$ such that only the d last elements of $\boldsymbol{a}$ are non-zero. They note that the best results are obtained for $d = 1$ or 2 ($d = 1$ being the same as the original BKW).
- In [DTV15], they completely skip the last oracle, and perform the hypothesis test on the entire last block, putting $d = n - (\alpha - 1)\beta$. This, they argue, will decrease run-time as they only make $2^{\alpha-1}$ recursive calls, not 2^{α}, which will reduce the error summation by a half.

For our description, we will follow [DTV15], although we will note some differences to [Alb+15]. In the rest of this section, we will go through an analysis of the cost of constructing each table T_l. Since there is a lot of variables to be considered, we will define them all in the following definition and refer back to them in the consequent lemmas.

Definition 4.1. Let n, q be positive integers and $\Pi_{\mathbf{s},\chi}$ be an LWE oracle, where $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret vector and χ the error distribution on $\mathbb{Z}_q$. Let the integers α and β be such that $1 \leq \alpha \leq n$ and $\alpha\beta \leq n$. For $1 \leq l \leq \alpha - 1$, $\mathcal{A}_{\mathbf{s},\chi,l}$ is the BTW-oracle outputting samples $(\boldsymbol{a}, t)$ where the first $l\beta$ elements of $\boldsymbol{a}$ is zero. For each $\mathcal{A}_{\mathbf{s},\chi,l}$ we have a corresponding table T_l that stores samples from $\mathcal{A}_{\mathbf{s},\chi,l-1}$. The details for $\mathcal{A}_{\mathbf{s},\chi,l}$ and T_l are discussed above. Lastly, define $d = n - (\alpha - 1)\beta \leq \beta$ to be size of the last block.

We begin by show the maximum number of samples each table T_l must store.

Lemma 4.2. *Let n, q, α, β, l and T_l be as in Definition 4.1. Then the*

maximum number of samples each table T_l need to store is

$$\left(\frac{q^\beta - 1}{2}\right).$$

Proof. There are q^β possible combinations of β elements from $\mathbb{Z}_q$. Next, we note that we do not need to store the case where the whole block is zero, as this will be directly outputted by the oracle $\mathcal{A}_{\mathbf{s},\chi,l}$, therefore we are down to $q^\beta - 1$ possible combinations.

Lastly, for each LWE sample, there are two possible matches, the positive *and* negative. By the symmetry of $\mathbb{Z}_q$, this means we only need half of the possible combinations. Hence we are left with the required

$$\left(\frac{q^\beta - 1}{2}\right).$$

□

This means that in order to get r samples from $\mathcal{A}_{\mathbf{s},\chi,l}$, we make at most $\frac{q^\beta - 1}{2} + r$ calls to $\mathcal{A}_{\mathbf{s},\chi,l-1}$.

It is also worth noting that when storing samples in T_l, we do not need to store the first $(l-1)\beta$ elements as these are zero by construction. Hence the length of each entry is $n-(l-1)\beta+1$ (the $+1$ value is for the t value).

Now we prove an upper bound on the number of ring operations required to fill T_l.

Lemma 4.3. *Let n, q, α, β, l and T_l be as in Definition 4.1. If all tables T_j for $1 \leq j < l$ have been filled with $\left(\frac{q^\beta - 1}{2}\right)$ samples. Then the number of ring operations in $\mathbb{Z}_q$ required to fill T_l is upper bounded by*

$$\left(\frac{q^\beta - 1}{2}\right)(l-1)\left((n+1) - \frac{l}{2}\beta\right).$$

Proof. T_1 has no blocks that needs to be cancelled, so this requires zero ring operations.

To construct T_2, we first draw a sample from the oracle $\Pi_{\mathbf{s},\chi}$ and match it to a vector in T_1, cancelling out the first β elements, meaning we have to perform $(n+1-\beta)$ additions on the remaining elements. Hence, to fill the table, we perform $\left(\frac{q^\beta-1}{2}\right)(n+1-\beta)$ additions in $\mathbb{Z}_q$.

The same argument goes for T_3: Draw a sample from $\Pi_{\mathbf{s},\chi}$ and perform $(n+1-\beta)$ ring operations with a vector from T_1. Then we match this new vector with a vector from T_2, making a further β elements zero, and performing $n+1-2\beta$ ring operations on the remaining elements. Meaning that, in total, we perform $\left(\frac{q^\beta-1}{2}\right)((n+1-\beta)+(n+1-2\beta))$ additions in $\mathbb{Z}_q$ to fill T_3.

By now, a pattern is emerging. For a general $1 \leq l \leq \alpha-1$, assume we have filled all the tables for indexes less than l. Then, to fill T_l, we have to perform

$$\begin{aligned}\left(\frac{q^\beta-1}{2}\right)\sum_{j=1}^{l-1}(n+1-j\beta) &= \left(\frac{q^\beta-1}{2}\right)\left((l-1)(n+1)-\frac{l}{2}(l-1)\beta\right), \\ &= \left(\frac{q^\beta-1}{2}\right)(l-1)\left(n+1-\frac{l}{2}\beta\right),\end{aligned}$$

operations, as required. □

Lastly, we look at the cost of obtaining r different samples from a BKW-oracle $\mathcal{A}_{\mathbf{s},\chi,l}$.

Lemma 4.4. *Let n, q, α, β, l and $\mathcal{A}_{\boldsymbol{s},\chi,l}$ be as in Definition 4.1. The worst case cost of obtaining r samples from $\mathcal{A}_{\boldsymbol{s},\chi,\alpha-1}$ is upper bounded by*

$$\left(\frac{q^\beta-1}{2}\right)\left(\frac{(\alpha-1)(\alpha-2)}{2}(n+1)-\frac{\alpha\beta(\alpha-1)(\alpha-2)}{6}\right)+r\left(\frac{\alpha-1}{2}(n+1)\right), \tag{4.2}$$

additions in $\mathbb{Z}_q$ and $(\alpha-1)\left(\frac{q^\beta-1}{2}\right)+r$ calls to $\Pi_{\boldsymbol{s},\chi}$.

Proof. Worst case scenario for obtaining r samples from $\mathcal{A}_{\mathbf{s},\chi,\alpha-1}$ occurs when we have to fill each table T_l, $1 \leq l \leq \alpha-1$, with $\frac{q^\beta-1}{2}$ samples for all

$l = 2, \ldots, \alpha - 1$. The number of operations it takes to fill one table T_l is given in Lemma 4.3. Hence, we sum up these operations for $l = 2, ..., \alpha - 1$ to get the total number of additions in $\mathbb{Z}_q$ required to fill all the tables:

$$\left(\frac{q^\beta - 1}{2}\right) \sum_{l=2}^{\alpha-1} \left((l-1)(n+1) - \frac{\beta l}{2}(l-1)\right).$$

Note that

$$\sum_{l=2}^{\alpha-1}(l-1) = \frac{(\alpha-1)(\alpha-2)}{2} \quad \text{and} \quad \sum_{l=2}^{\alpha-1} l(l-1) = \frac{2\alpha(\alpha-1)(\alpha-2)}{6}.$$

Hence, it takes

$$\left(\frac{q^\beta - 1}{2}\right)\left(\frac{(\alpha-1)(\alpha-2)}{2}(n+1) - \frac{\alpha\beta(\alpha-1)(\alpha-2)}{6}\right) \tag{4.3}$$

operations to fill all $\alpha - 1$ tables.

To obtain a sample from $\mathcal{A}_{\mathbf{s},\chi,\alpha-1}$, where only the last d elements can be non-zero, we sample a random tuple $(\boldsymbol{a}, t)$ from $\Pi_{\mathbf{s},\chi}$ and match it with vectors in $T_1, T_2, \ldots, T_{\alpha-1}$ in turn. The number of additions done in $\mathbb{Z}_q$ is upper bounded by

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

Therefore, to sample r independent samples from $\mathcal{A}_{\mathbf{s},\chi,\alpha-1}$, the number of additions in $\mathbb{Z}_q$ is upper bounded by

$$r\left(\frac{\alpha-1}{2}\right)(n+2). \tag{4.4}$$

Adding this with Equation (4.3) completes the first part of the proof. The second part follows from the fact that it takes $(\alpha-1)\left(\frac{q^\beta-1}{2}\right)$ calls to $\Pi_{\mathbf{s},\chi}$ to fill all $(\alpha-1)$ tables, and then a further r calls to obtain the r samples. □

4.1.2 Hypothesis Testing

After the reduction step, we are left with r samples of the form $(\boldsymbol{a}_i, t_i)$, $1 \leq i \leq r$, where only the d last elements of $\boldsymbol{a}_i$ are non-zero. Let $\mathbf{s}' = \mathbf{s}_{[n-d,n]} \in \mathbb{Z}_q^d$ be the last d entries of the secret vector $\mathbf{s}$.

Note that

$$t_i = \langle \boldsymbol{a}_i, \mathbf{s} \rangle + \nu_i = \langle \boldsymbol{a}_{i,[n-d,n]}, \mathbf{s}' \rangle + \nu_i \quad \Rightarrow \quad \nu_i = t_i - \langle \boldsymbol{a}_{i,[n-d,n]}, \mathbf{s}' \rangle$$

Since the first $n-d$ elements of $\boldsymbol{a}$ are zero, we will drop the block notation $\boldsymbol{a}_{i,[n-d,n]}$ for ease of writing, and simply treat the $\boldsymbol{a}_i$'s as vectors in $\mathbb{Z}_q^d$ for the remainder of this section.

Now we want to find $\mathbf{s}'$. We know what $\boldsymbol{a}_i$ and t_i are, and we also know that ν_i is the sum of $2^{\alpha-1}$ error samples uniformly and independently sampled from χ. The procedure of hypothesis testing is then an exhaustive search over $\mathbb{Z}_q^d$ where for each $\mathbf{v} \in \mathbb{Z}_q^d$ we set the hypothesis $\mathbf{v} = \mathbf{s}'$. The exhaustive search is done in the following way: Define the function $f : \mathbb{Z}_q^d \mapsto \mathbb{C}$ by

$$f(\boldsymbol{x}) = \sum_{j=1}^{r} \pi_j(\boldsymbol{x}) e^{\frac{2\pi}{q} i t_j}, \text{ where } \pi_j(\boldsymbol{x}) = \begin{cases} 1 \text{ if } \boldsymbol{x} = \boldsymbol{a}_j \\ 0 \text{ otherwise} \end{cases},$$

where r is the number of samples. For ease of writing, put $\xi = \frac{2\pi i}{q}$. The Discrete Fourier Transform (DFT) of f is,

$$\begin{aligned} \hat{f}(\mathbf{v}) &= \sum_{\boldsymbol{x} \in \mathbb{Z}_q^d} f(\boldsymbol{x}) e^{-\xi \langle \boldsymbol{x}, \mathbf{v} \rangle}, \\ &= \sum_{\boldsymbol{x} \in \mathbb{Z}_q^d} \sum_{j=1}^{r} \pi_j(\boldsymbol{x}) e^{\xi t_j} e^{-\xi \langle \boldsymbol{x}, \mathbf{v} \rangle}, \\ &= \sum_{j=1}^{r} e^{-\xi(\langle \boldsymbol{a}_j, \mathbf{v} \rangle - t_j)}. \end{aligned}$$

Note here that $\hat{f}(\mathbf{s}') = \sum_{j=1}^{r} e^{-\xi(\langle \boldsymbol{a}_j, \mathbf{s} \rangle - t_j)} = \sum_{j=1}^{r} e^{-\xi \nu_j}$ where ν_j is the sum of $2^{\alpha-1}$ independent samples from χ; $\nu_j = \nu_{j,1} \pm \cdots \pm \nu_{j,2^{\alpha-1}}$. Meaning

that

$$\hat{f}(\mathbf{s}') = \sum_{j=1}^{r} e^{\xi(\nu_{j,1} \pm \cdots \pm \nu_{j,2^\alpha - 1})}. \tag{4.5}$$

In Section 4.1.4, we will show that as long as we choose the values α and the number of samples r appropriately, there is a high probability that $\arg\max_{\mathbf{v} \in \mathbb{Z}_q^d} \mathrm{Re}(\hat{f}(\mathbf{v})) = \mathbf{s}'$. We also show how many independent samples r of this form we should obtain in order to get the correct result. This is an important, but lengthy result, so for now we simply claim that this is the case and focus on the algorithm. Hence, the hypothesis testing algorithm computes the real part of $\hat{f}(\mathbf{v})$ for all $\mathbf{v} \in \mathbb{Z}_q^d$ and return the $\mathbf{v}$ for which the real value has a maximum.

4.1.3 Back Substitution

Back substitution was not part of the original BKW algorithm, but was added on as a last step later on. It is similar to the back substitution we do when solving a system of linear equation with Gaussian elimination.

After Hypthesis testing, we have hopefully recovered $\mathbf{s}'$ with high probability. Then we go back to the tables T_l, $l = 1, \ldots, \alpha - 1$. For every $(\boldsymbol{a}, t) \in T_l$, let $\boldsymbol{a}' = \boldsymbol{a}_{[n-d,n]}$. Then update the values

$$\boldsymbol{a} \leftarrow \boldsymbol{a} - (0, \ldots, 0, \boldsymbol{a}')$$
$$t \leftarrow t - \langle \boldsymbol{a}', \mathbf{s}' \rangle.$$

For each pair $(\boldsymbol{a}_i, t_i)$ it takes $2d$ operations to update each row, and since there is in total $(\alpha - 1)\left(\frac{q^\beta - 1}{2}\right)$ rows in all the tables T_l, back substitution requires $2d(\alpha - 1)\left(\frac{q^\beta - 1}{2}\right)$ operations.

When this is done, we repeat the algorithm on the next block of values until all elements of $\mathbf{s}$ are found.

4.1.4 Proof That $\arg\max_{\mathbf{v}\in\mathbb{Z}_q^d} \mathbf{Re}(\hat{f}(\mathbf{v})) = \mathbf{s}'$ Is A Reasonable Assumption

Returning to Equation (4.5) and using Euler's formula, $e^{ix} = \cos x + i \sin x$, we see that we get a sum of multiplications of $2^{\alpha-1}$ factors of the form $\left(\cos\left(\frac{2\pi}{q}\nu\right) + i\sin\left(\frac{2\pi}{q}\nu\right)\right)$ where $\nu \leftarrow \chi$. We want to calculate what we expect the values of random variables of this form to be, when χ is the *rounded Gaussian distribution* $\bar{\Psi}_{\sigma,q}$ or the *discrete Gaussian distributions* $D_{\sigma,q}$. Both of these are widely used for the LWE problem. The aim is to prove that that the argmax of $\hat{f}(\mathbf{v})$ is $\mathbf{s}'$ with a high probability. We also want to use this value to find a lower bound on how big the sample size r needs to be.

This will be quite a lengthy result. We start of, in Lemmas 4.5 and 4.7, by finding expected values for $\cos\left(\frac{2\pi}{q}\chi\right)$ and $\sin\left(\frac{2\pi}{q}\chi\right)$. Then, in Lemma 4.8, we use this to find a lower bound for the real part of $\hat{f}(\mathbf{s}')$. Lastly, in Lemma 4.11 we show that the probability that $\arg\max_{\mathbf{v}\in\mathbb{Z}_q^d} \mathrm{Re}(\hat{f}(\mathbf{v})) \neq \mathbf{s}'$ gets smaller the higher number of samples we have, and use this to find a lower bound on what r should be.

Before we start, we briefly explain the rounded Gaussian distribution $\bar{\Psi}_{\sigma,q}$. Let $N(0,\sigma^2)$ denote the continuous Gaussian distribution with mean 0 and standard deviation σ. If we wrap this distribution around a circle with circumference $q \geq 0$, we obtain the *wrapped* Gaussian distribution $\Psi_{\sigma,q}$. The *rounded* Gaussian distribution can be obtained by sampling from $\Psi_{\sigma,q}$ and rounding the result to the nearest integer in the interval $(-\frac{q}{2}, \frac{q}{2}]$. This is the distribution used in Regev's original result ([Reg05]).

Lemma 4.5. *For an odd integer q, let $X = \bar{\Psi}_{\sigma,q}$ or $D_{\sigma,q}$ and let $Y = \frac{2\pi}{q}X$. Then*

$$\mathbb{E}[\cos(Y)] \geq \begin{cases} \frac{q}{\pi}\sin\left(\frac{\pi}{q}\right) e^{-\frac{2\pi^2\sigma^2}{q^2}} & \text{if } X = \bar{\Psi}_{\sigma,q} \\ 1 - \frac{2\pi^2\sigma^2}{q^2} & \text{if } X = D_{\sigma,q} \end{cases} \tag{4.6}$$

Proof. **Case 1**: $X = D_{\sigma,q}$: This is Lemma 11 in [DTV15], the proof if which is pretty straight forward, but it uses material slightly beyond the scope of this thesis.

Case 2: $X = \bar{\Psi}_{\sigma,q}$.

Let S_l be the set of integers in $(lq - \frac{q}{2}, lq + \frac{q}{2}]$. We start of with some basic statistics formulas and properties needed for this proof.

The *expected value* $\mathbb{E}$ for a random variable X with a finite number of possible outcomes $x_1, \ldots, x_n$, with the associated probabilities $p_1, \ldots, p_n$ respectively, is given by

$$\mathbb{E}[X] = \sum_{i=1}^{n} x_i p_i.$$

In our case, the possible outcomes of $\cos Y$ are $\cos\left(\frac{2\pi}{q}x\right)$ for $x \in S_0$.

The probability density function for $N(0, \sigma^2)$ is given by

$$p(\theta; \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{\theta^2}{2\sigma^2}}.$$

And the probability density function for the wrapped Gaussian $\Psi_{\sigma,q}$ is given by:

$$g(\theta; \sigma, q) = \sum_{l=-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(\theta+lq)^2}{2\sigma^2}},$$

for $\theta \in \left(-\frac{q}{2}, \frac{q}{2}\right]$. Note that

$$g(\theta; \sigma, q) = \sum_{l=-\infty}^{\infty} p(\theta + lq; \sigma). \tag{4.7}$$

The probability mass function of the rounded wrapped Gaussian distribution $\bar{\Psi}_{\sigma,q}$ is given by

$$\Pr(x \leftarrow \bar{\Psi}_{\sigma,q}) = \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} g(\theta; \sigma, q) \ \mathrm{d}\theta. \tag{4.8}$$

for the integer x in the interval $\left(-\frac{q}{2}, \frac{q}{2}\right]$.

Returning to our expected value for $\cos(Y)$, we note that the probability for $\cos\left(\frac{2\pi}{q}x\right)$ is simply equal to the probability of x, which is given by Equations (4.7) and (4.8). Hence, the expected value of $\cos(Y)$ is

$$\begin{aligned}\mathbb{E}[\cos(Y)] &= \sum_{x\in S_0} \cos\left(\frac{2\pi}{q}x\right) \sum_{l=-\infty}^{\infty} \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} p(\theta + lq; \sigma)\ \mathrm{d}\theta, \\ &= \sum_{l=-\infty}^{\infty} \sum_{x\in S_0} \cos\left(\frac{2\pi}{q}x + 2\pi l\right) \int_{x-\frac{1}{2}}^{x+\frac{1}{2}} p(\theta + lq; \sigma)\ \mathrm{d}\theta.\end{aligned}$$

Use the substitution $u(\theta) = \theta + lq$ in the integration and put $x' = x + lq$, then we get:

$$\begin{aligned}\mathbb{E}[\cos(Y)] &= \sum_{l=-\infty}^{\infty} \sum_{x'\in S_l} \cos\left(\frac{2\pi}{q}x'\right) \int_{x'-\frac{1}{2}}^{x'+\frac{1}{2}} p(\theta; \sigma)\ \mathrm{d}\theta, \\ &= \sum_{x'=-\infty}^{\infty} \cos\left(\frac{2\pi}{q}x'\right) \int_{x'-\frac{1}{2}}^{x'+\frac{1}{2}} p(\theta; \sigma)\ \mathrm{d}\theta.\end{aligned}$$

At this point we state the *Poisson summation formula*, which is Lemma 25 in [DTV15].

Lemma 4.6 (Poisson summation formula)**.** *Let $f : \mathbb{R} \to \mathbb{C}$ be a function of the Schwart space and $\mathcal{F}(f)$ its continuous Fourier transform, then*

$$\sum_{l=-\infty}^{\infty} f(l) = \sum_{k=-\infty}^{\infty} \mathcal{F}(f)(k).$$

Recall that the Continuous Fourier transform (CFT) of a function $f : \mathbb{R} \to \mathbb{C}$ is given by $\mathcal{F}(f)(k) = \int_{-\infty}^{\infty} f(x)e^{-2\pi kxi}\mathrm{d}x$. Applying Poisson summation formula to our expression for $\mathbb{E}[\cos(Y)]$ gives:

$$\begin{aligned}\mathbb{E}[\cos(Y)] &= \sum_{k=-\infty}^{\infty} \mathcal{F}\left(\cos\left(\frac{2\pi}{q}x'\right) \int_{x'-\frac{1}{2}}^{x'+\frac{1}{2}} p(\theta; \sigma)\ \mathrm{d}\theta\right)(k), \\ &= \sum_{k=-\infty}^{\infty} \left(\mathcal{F}\left(\cos\left(\frac{2\pi}{q}x'\right)\right) * \mathcal{F}\left(\int_{x'-\frac{1}{2}}^{x'+\frac{1}{2}} p(\theta; \sigma)\ \mathrm{d}\theta\right)\right)(k),\end{aligned} \tag{4.9}$$

where $*$ denotes the convolution operator [†]. We want to find the two Fourier transform of the convolution. Straight forward calculations gives $\mathcal{F}\left(\cos\left(\frac{2\pi}{q}x\right)\right)(k) = \frac{1}{2}\left(\delta\left(k-\frac{1}{q}\right)+\delta\left(k+\frac{1}{q}\right)\right)$, where δ is the Dirac delta function. The second transform requires a little more computations (we omit to write (k) after each Fourier transform for ease of reading):

$$\begin{aligned}\mathcal{F}\left(\int_{x'-\frac{1}{2}}^{x'+\frac{1}{2}} p(\theta;\sigma)\ \mathrm{d}\theta\right) &= \mathcal{F}\left(\int_{-\infty}^{x+\frac{1}{2}} p(\theta;\sigma)\ \mathrm{d}\theta\right) - \mathcal{F}\left(\int_{-\infty}^{x-\frac{1}{2}} p(\theta;\sigma)\ \mathrm{d}\theta\right),\\ &= \mathcal{F}\left(\int_{-\infty}^{x} p(\theta-\frac{1}{2};\sigma)\ \mathrm{d}\theta\right) - \mathcal{F}\left(\int_{-\infty}^{x} p(\theta+\frac{1}{2};\sigma)\ \mathrm{d}\theta\right),\\ &= \left(e^{\pi k i} - e^{-\pi k i}\right)\mathcal{F}\left(\int_{-\infty}^{x} p(\theta;\sigma)\ \mathrm{d}\theta\right), \qquad (4.10)\end{aligned}$$

by the translation property of CFT[‡]. Now we have the integration property:

$$\mathcal{F}\left(\int_{-\infty}^{x} f(y)\ \mathrm{d}y\right)(k) = \frac{1}{2i\pi k}\mathcal{F}(f)(k) + \frac{1}{2}\mathcal{F}(f)(0)\delta(k)$$

Note also, from straight forward calculations, that $\mathcal{F}\left(p(\theta;\sigma)\right)(k) = e^{-2\pi^2\sigma^2k^2}$. That means Equation (4.10) becomes

$$\begin{aligned}&2i\sin(\pi k)\left(\frac{1}{2i\pi k}\mathcal{F}(p(\theta;\sigma))(k) + \frac{1}{2}\mathcal{F}(p(\theta;\sigma))(0)\delta(k)\right),\\ &= 2i\sin(\pi k)\left(\frac{1}{2i\pi k}e^{-2\pi^2\sigma^2k^2} + \frac{1}{2}\delta(k)\right).\end{aligned}$$

Returning to Equation (4.9), we get

$$\begin{aligned}&\mathcal{F}\left(\cos\left(\frac{2\pi}{q}x'\right)\right) * \mathcal{F}\left(\int_{x'-\frac{1}{2}}^{x'+\frac{1}{2}} p(\theta;\sigma)\ \mathrm{d}\theta\right)\\ &= \frac{1}{2}\left(\left(\delta\left(k-\frac{1}{q}\right)+\delta\left(k+\frac{1}{q}\right)\right) * \left(2i\sin(\pi k)\left(\frac{1}{2i\pi k}e^{-2\pi^2\sigma^2k^2} + \frac{1}{2}\delta(k)\right)\right)\right).\end{aligned}$$

Note that $\delta(k) * \delta(k \pm \frac{1}{q}) = 0$ for $k \in \mathbb{Z}$ (which is the case here). This means we get the convolution

$$\frac{1}{2}\left(\delta\left(k-\frac{1}{q}\right)+\delta\left(k+\frac{1}{q}\right)\right) * \left(\sin(\pi k)\left(\frac{1}{\pi k}e^{-2\pi^2\sigma^2k^2}\right)\right). \qquad (4.11)$$

[†]Convolution: $(u*v)(x) := \int_{-\infty}^{\infty} u(y)v(x-y)\mathrm{d}y$

[‡]Translation: $\mathcal{F}(f(x-y))(k) = e^{-2\pi y k i}\mathcal{F}(f)(k)$

Recall that convolution is distributive, put $h(k) = \sin(\pi k)\left(\frac{1}{\pi k}e^{-2\pi^2\sigma^2k^2}\right)$, and calculate

$$\begin{aligned}\delta\left(k \pm \frac{1}{q}\right) * h(k) &= \int_{-\infty}^{\infty} \delta\left(y \pm \frac{1}{q}\right) h(k-y)\ \mathrm{d}y, \\ &= h\left(k \mp \frac{1}{q}\right).\end{aligned}$$

Note also that $\sin\left(\pi k \pm \frac{\pi}{q}\right) = \pm(-1)^k \sin\left(\frac{\pi}{q}\right)$. Equation (4.11) then becomes

$$\frac{q}{2\pi}\sin\left(\frac{\pi}{q}\right)(-1)^k\left(\frac{e^{-2\pi^2\sigma^2\frac{(qk+1)^2}{q}}}{qk+1} - \frac{e^{-2\pi^2\sigma^2\frac{(qk-1)^2}{q}}}{qk-1}\right).$$

It is easy to check that this is an even function. Hence, going all the way back to Equation (4.9) we get that $\mathbb{E}[\cos(Y)]$ is equal to

$$\frac{q}{\pi}\sin\left(\frac{\pi}{q}\right)e^{\frac{2\pi^2\sigma^2}{q^2}} + \sum_{k=1}^{\infty}\frac{q}{\pi}\sin\left(\frac{\pi}{q}\right)(-1)^k\left(\frac{e^{-2\pi^2\sigma^2\frac{(qk+1)^2}{q}}}{qk+1} - \frac{e^{-2\pi^2\sigma^2\frac{(qk-1)^2}{q}}}{qk-1}\right). \tag{4.12}$$

Note that the first term in the summation is positive and that the absolute value is clearly decreasing as k increases. Hence deduce

$$\mathbb{E}[\cos(Y)] \geq \frac{q}{\pi}\sin\left(\frac{\pi}{q}\right)e^{-\frac{2\pi^2\sigma^2}{q^2}}.$$

□

Next, we find the expected value for $\sin\left(\frac{2\pi}{q}\chi\right)$. It might come as a relief that this result is a lot more straight forward.

Lemma 4.7. *For an odd integer q, let $X =\sim \bar{\Psi}_{\sigma,q}$ or $D_{\sigma,q}$ and let $Y = \frac{2\pi}{q}X$. Then*

$$\mathbb{E}[\sin(Y)] = 0.$$

Proof. For both distributions, note that for odd q, the distributions are perfectly symmetric around 0. The result follows trivially from the symmetry of the sine function.

□

Here we define the variable

$$R_{\sigma,q,\chi} := \begin{cases} \frac{q}{\pi} \sin\left(\frac{\pi}{q}\right) e^{-\frac{2\pi^2\sigma^2}{q^2}} & \text{if } \chi = \bar{\Psi}_{\sigma,q}, \\ \\ 1 - \frac{2\pi^2\sigma^2}{q^2} & \text{if } \chi = D_{\sigma,q}. \end{cases} \tag{4.13}$$

Now we derive a lower bound on $\mathbb{E}[\mathrm{Re}(\hat{f}(\mathbf{s}')]$ that we will use when calculating the probability of failure in the next lemma.

Lemma 4.8. $\mathbb{E}[Re(\hat{f}(\boldsymbol{s}')] \geq r \cdot (R_{\sigma,q,\chi})^{2^{\alpha-1}}$.

Proof. Recall from Equation (4.5) that $\hat{f}(\mathbf{s}') = \sum_{j=1}^{r} e^{\xi(\nu_{j,1} \pm \cdots \pm \nu_{j,2^{\alpha-1}})}$. Using Lemmas 4.5 and 4.7, and the independence of the samples from χ we get that

$$\begin{aligned} \mathbb{E}\left[\mathrm{Re}\left(\hat{f}(\mathbf{s})\right)\right] &= \mathrm{Re}\left(\sum_{j=1}^{r} \mathbb{E}\left[e^{\xi(\nu_{j,1} \pm \cdots \pm \nu_{j,2^{\alpha-1}})}\right]\right), \\ &= \mathrm{Re}\left(\sum_{j=1}^{r} \mathbb{E}\left[e^{-\xi\nu_{j,1}}\right] \ldots \mathbb{E}\left[e^{-\xi\nu_{j,2^{\alpha-1}}}\right]\right), \\ &= \mathrm{Re}\left(\sum_{j=1}^{r} \mathbb{E}\left[\cos\left(\frac{2\pi}{q}\nu_{j,1}\right)\right] \ldots \mathbb{E}\left[\cos\left(\frac{2\pi}{q}\nu_{j,2^{\alpha-1}}\right)\right]\right), \\ &\geq \sum_{j=1}^{r} R_{\sigma,q,\chi}^{2^{\alpha-1}}, \\ &= r \cdot R_{\sigma,q,\chi}^{2^{\alpha-1}}. \end{aligned}$$

□

Next, we are going to quote one result from [DTV15] and one result from statistics that we will use further on, but not prove in this thesis.

Lemma 4.9 ([DTV15, Lemma 14])**.** *Let $G \subset \mathbb{Z}_q$ be a subgroup of $\mathbb{Z}_q$, let $X \xleftarrow{U} G$ and let $z \in \mathbb{Z}_q$ be independent from X. Then $\mathbb{E}\left[e^{\frac{2\pi}{q}(X+z)s}\right] = 0$.*

Theorem 4.10 (Hoeffding's Inequality, [Hoe63])**.** *Let $X_1, \ldots, X_n$ be n independent random variables such that $\Pr(X_j \in [\alpha_j, \beta_j]) = 1$ for $1 \leq j \leq n$. Define $X = \sum_{j=1}^{n} X_n$, then*

$$\Pr(X - \mathbb{E}[X] \geq R) \leq \exp\left(\frac{-2R^2}{\sum_{j=1}^{n}(\beta_j - \alpha_j)}\right) \tag{4.14}$$

and

$$\Pr(X - \mathbb{E}[X] \leq -R) \leq \exp\left(\frac{-2R^2}{\sum_{j=1}^{n}(\beta_j - \alpha_j)}\right) \tag{4.15}$$

for any $R > 0$.

In the next lemma, we prove an upper bound on the probability that $\arg\max_{\mathbf{v}\in\mathbb{Z}_q^d} \mathrm{Re}(\hat{f}(\mathbf{v}))$ is *not* $\mathbf{s}'$. We will use this upper bound in Theorem 4.12 to bound the number of samples required to find $\mathbf{s}'$ with a certain probability.

Lemma 4.11. *Let $\hat{f}$ be as defined in Equation (4.5). Let ϵ denote the probability that $\arg\max_{\boldsymbol{v}\in\mathbb{Z}_q^d} Re(\hat{f}(\boldsymbol{v})) \neq \boldsymbol{s}'$. Then*

$$\epsilon \leq q^d e^{-\frac{\tau}{8}(R_{\sigma,q,\chi})^{2^\alpha}} \tag{4.16}$$

Proof. Define the two cases:

- A: $\exists\ \mathbf{v} \in \mathbb{Z}_q^d \backslash \{\mathbf{s}'\}$ such that $\mathrm{Re}(\hat{f}(\mathbf{s}')) \leq \mathrm{Re}(\hat{f}(\mathbf{v}))$.
- B: $\mathrm{Re}(\hat{f}(\mathbf{s}')) \leq \mathrm{Re}(\hat{f}(\mathbf{v}))$ for some fixed $\mathbf{v} \in \mathbb{Z}_q^d \backslash \{\mathbf{s}'\}$.

Note that $\epsilon = \Pr(\mathrm{A})$. We know there is q^d unique vectors in $\mathbb{Z}_q^d$. So the probability that there exist one vector $\mathbf{v} \in \mathbb{Z}_q^d$ such that $\mathrm{Re}(\hat{f}(\mathbf{s}')) \leq \mathrm{Re}(\hat{f}(\mathbf{v}))$ is bounded above by all the possible vectors in $\mathbb{Z}_q^d$ times the probability that one of them might satisfy $\mathrm{Re}(\hat{f}(\mathbf{s}')) \leq \mathrm{Re}(\hat{f}(\mathbf{v}))$. Hence $\Pr(A) \leq q^d \Pr(B)$.

Next we observe that $\Pr(B) = \Pr(\mathrm{Re}(\hat{f}(\mathbf{s}')) - \mathrm{Re}(\hat{f}(\mathbf{v})) \leq 0)$. Which is the probability that $\sum_{j=1}^{r} \left(\mathrm{Re}\left(e^{\xi(\langle \boldsymbol{a}_j, \mathbf{s}'\rangle - t_j)}\right) - \mathrm{Re}\left(e^{\xi(\langle \boldsymbol{a}_j, \mathbf{v}\rangle - t_j)}\right)\right) \leq 0$. Define

$\boldsymbol{x} = \mathbf{v} - \mathbf{s}'$ and note that $\langle \boldsymbol{a}_j, \mathbf{v} \rangle - t_j = \langle \boldsymbol{a}_j, \boldsymbol{x} \rangle + \nu_j$. Thus, the summation becomes

$$X = \sum_{j=1}^{r} \left(\mathrm{Re}\left(e^{\xi(\langle \boldsymbol{a}_j, \mathbf{s}' \rangle - t_j)} \right) - \mathrm{Re}\left(e^{\xi(\langle \boldsymbol{a}_j, \boldsymbol{x} \rangle + \nu_j)} \right) \right) \leq 0.$$

Define $X_j = u_j + v_j$ where $u_j = \mathrm{Re}\left(e^{\xi(\langle \boldsymbol{a}_j, \mathbf{s}' \rangle - t_j)} \right)$ and $v_j = \mathrm{Re}\left(e^{\xi(\langle \boldsymbol{a}_j, \mathbf{v} \rangle - t_j)} \right)$. Since $\boldsymbol{a}_j$ and ν_j are sampled uniformly and independently from each other, and because $\boldsymbol{x}$ is fixed and non-zero, $\langle \boldsymbol{a}_j, \boldsymbol{x} \rangle$ is uniformly distributed in a subgroup of $\mathbb{Z}_q$ and so is $\langle \boldsymbol{a}_j, \mathbf{v} \rangle - t_j$. Hence, by Lemma 4.9, $\mathbb{E}[v_j] = 0$. Therefore we can find a lower bound on $\mathbb{E}[X]$ thus:

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{j=1}^{r} \mathbb{E}[X_j] = \sum_{j=1}^{r} \mathbb{E}[u_j], \\
&= \sum_{j=1}^{r} \mathbb{E}\left[\mathrm{Re}\left(e^{\xi(\langle \boldsymbol{a}_j, \mathbf{s}' \rangle - t_j)} \right) \right], \\
&\geq r \cdot (R_{\sigma,q,\chi})^{2^{\alpha-1}},
\end{aligned}$$

by Lemma 4.8. Now, using Equation (4.15) from Theorem 4.10, putting $R = \mathbb{E}[X]$ and observing that $-2 \leq X_j \leq 2$, we get

$$\begin{aligned}
\Pr(B) = \Pr(X \leq 0) &= \Pr(X - \mathbb{E}[X] \leq -\mathbb{E}[X]), \\
&\leq \exp\left(\frac{-2\mathbb{E}[X]^2}{\sum_{j=1}^{n} (2 - (-2))} \right), \\
&\leq \exp\left(\frac{-2r^2 (R_{\sigma,q,\chi})^{2^\alpha}}{r \cdot 4^2} \right), \\
&= \exp\left(\frac{-r}{8} (R_{\sigma,q,\chi})^{2^\alpha} \right).
\end{aligned}$$

Putting this all together gives

$$\epsilon \leq q^d e^{\frac{r}{8} \cdot (R_{\sigma,q,\chi})^{2^\alpha}}$$

□

Note that $\Pr\left(\arg\max_{\mathbf{v}}\left(\mathrm{Re}(\hat{f}(\mathbf{v}))\right) = \mathbf{s}'\right) = 1 - \epsilon \geq 1 - q^d e^{\frac{r}{8}\cdot(R_{\sigma,q,\chi})^{2^\alpha}}$ is the probability of success.

From this, we can derive the number of samples we need in order to derive the correct value for $\mathbf{s}'$ with high probability.

Theorem 4.12. *Let n, q, l, $\boldsymbol{s}$, χ, $\Pi_{\boldsymbol{s},\chi}$, α, β, $\mathcal{A}_{\boldsymbol{s},\chi,\alpha-1}$ and d be as described in Definition 4.1. Let $\boldsymbol{s}' \in \mathbb{Z}_q^d$ be equal to the last d elements of $\boldsymbol{s}$. Let $\epsilon \in (0,1)$ denote the probability that we fail to recover $\boldsymbol{s}'$.*

Then the number of independent samples r^* required from $\mathcal{A}_{\boldsymbol{s},\chi,\alpha-1}$ to recover $\boldsymbol{s}'$ is

$$r^* \geq 8d \log\left(\frac{q}{\epsilon}\right) (R_{\sigma,q,\chi})^{-2^\alpha} \tag{4.17}$$

where $R_{\sigma,q,\chi}$ is given by Lemma 4.5.

Proof. Using Lemma 4.11, considering ϵ and solving for r yields the required result. □

4.2 Analysis

Now we want to calculate the complexity of BKW with multidimensional DFT. This result follows Theorem 17 in [DTV15]. We also follow their choice of choosing $\alpha \cdot \beta = n$ for ease of notation, but note that the general case for $n = (\alpha - 1)\beta + d$ follows similarly.

When calculating the time complexity of the BKW algorithm we need to add together:

- γ_1 := The number of additions in $\mathbb{Z}_q$ to produce all $\alpha - 1$ T_l tables.
- γ_2 := The number of additions in $\mathbb{Z}_q$ to produce the samples required to recover all blocks of $\mathbf{s}$ with probability ϵ.
- γ_3 := The number of operations in $\mathbb{C}$ to prepare and compute the DFT's.
- γ_4 := The number of operations in $\mathbb{Z}_q$ for the back substitution

We will go through them in turn:

Producing the tables: γ_1

We showed this in the proof of Lemma 4.4, in Equation (4.3), i.e.

$$\gamma_1 = \left(\frac{q^\beta - 1}{2}\right)\left(\frac{(\alpha-1)(\alpha-2)}{2}(n+1) - \frac{\alpha\beta(\alpha-1)(\alpha-2)}{6}\right).$$

Recovering the blocks: γ_2

Next we look at how many additions in $\mathbb{Z}_q$ it takes to recover all $\alpha - 1$ blocks of $\mathbf{s}$ assuming all tables T_l, $l = 1, \ldots, \alpha - 1$ have been filled. For this we must first calculate the number of operations required to reduce *one* sample such that only the last few elements are non-zero, and then multiply this with the number of samples required (similar to what we did in Equation (4.4)). Note that when we want to obtain a sample from $\mathcal{A}_{\mathbf{s},\chi,\alpha-2}$, we do not sample anything from $\mathcal{A}_{\mathbf{s},\chi,\alpha-1}$ and hence we don not make use of the table $T_{\alpha-1}$ at all. Hence, to obtain one sample from $\mathcal{A}_{\mathbf{s},\chi,\alpha-1-j}$ we must perform

$$\begin{aligned}\sum_{i=1}^{\alpha-1-j}(n+1-i\beta) &= (\alpha-1-j)(n+1) - \beta\frac{(\alpha-1-j)(\alpha-j)}{2}, \\ &= (\alpha-1-j)\left((n+1) - \frac{\beta}{2}(\alpha-j)\right), \\ &= (\alpha-1-j)\left((n+1) - \frac{n}{2} + \frac{\beta j}{2}\right),\end{aligned}$$

additions in $\mathbb{Z}_q$. Let

$$r^*_{j,\epsilon} = 8\beta\log\left(\frac{q}{\epsilon}\right)(R_{\sigma,q,\chi})^{-2^{\alpha-j}}$$

and define $\epsilon' = \frac{\epsilon}{\alpha}$. Then

$$\gamma_2 = \sum_{j=0}^{\alpha-1} r^*_{j,\epsilon'}\frac{(\alpha-1-j)}{2}(n+2+\beta j).$$

Prepare and compute the DFT: γ_3

As DFT and Fast Fourier Transform (FFT) is not the focus of this thesis, we will simply state the result from Theorem 17 [DTV15]. The number of operations in $\mathbb{C}$ to prepare and compute the DFT's is

$$\gamma_3 = 2\sum_{j=0}^{\alpha-1} r^*_{j,\epsilon'} + C_{FFT} \cdot n \cdot q^\beta \cdot \log(q),$$

where C_{FFT} is a small constant in the complexity of DFT.

Back substitution: γ_4

Lastly, we look at the number of operations in $\mathbb{Z}_q$ for back substitution. Note that for the first block, $d \leq \beta$. And for each block we look at after $\alpha - 1$, i.e. $\alpha - 2, \alpha - 3, \ldots, 1$ we back substitute blocks of a size upper bounded by β. Hence, by the same argument as in Section 4.1.3, the number of operations done per row is upper bounded by 2β. For each table, there are $\left(\frac{q^\beta - 1}{2}\right)$ rows and for the $(\alpha - 1 - j)^{\text{th}}$ block, we have to back substitute in $\alpha - 2 - j$ tables. Hence, in total, back substitution requires

$$\alpha_4 = 2\beta\left(\frac{q^\beta - 1}{2}\right)\sum_{j=1}^{\alpha-2} j = \beta\left(\frac{q^\beta - 1}{2}\right)(\alpha - 1)(\alpha - 2).$$

operations in $\mathbb{Z}_q$. All of this can be combined into a theorem:

Theorem 4.13. *Let n, q be positive integers, where $\boldsymbol{s} \in \mathbb{Z}_q^n$. Let α, β be positive integers such that $\alpha\beta = n$. Let C_{FFT} be the small constant in the complexity of the Fast Fourier Transform computation. Let $0 < \epsilon < 1$ be the probability of failure define $\epsilon' = \frac{\epsilon}{\alpha}$. For $0 \leq j \leq \alpha - 1$, let $r^*_{j,\epsilon} = 8\beta \log\left(\frac{q}{\epsilon}\right)(R_{\sigma,q,\chi})^{-2^{\alpha-j}}$ where $R_{\sigma,q,\chi}$ is given in Equation*(4.13)*. Assuming that all the samples after reduction are independent, the time complexity of the BKW algorithm to recover $\boldsymbol{s}$ with probability at least $1-\epsilon$*

is $\gamma_1 + \gamma_2 + \gamma_3 + \gamma_4$ *where*

$$\gamma_1 = \left(\frac{q^\beta - 1}{2}\right)\left(\frac{(\alpha-1)(\alpha-2)}{2}(n+1) - \frac{\alpha\beta(\alpha-1)(\alpha-2)}{6}\right)$$

$$\gamma_2 = \sum_{j=0}^{\alpha-1} r^*_{j,\epsilon}\frac{(\alpha-1-j)}{2}(n+2+\beta j)$$

$$\gamma_3 = 2\sum_{j=0}^{\alpha-1} r^*_{j,\epsilon'} + C_{FFT}\cdot n\cdot q^\beta\cdot\log(q)$$

$$\gamma_4 = \beta\left(\frac{q^\beta-1}{2}\right)(\alpha-1)(\alpha-2)$$

where γ_1 *is the number of additions in* $\mathbb{Z}_q$ *to produce all tables* T_j*,* γ_2 *is the number of additions in* $\mathbb{Z}_q$ *to produce the samples required to recover all blocks of* $\boldsymbol{s}$ *with probaility* $1-\epsilon$*,* γ_3 *is the number of operations in* $\mathbb{C}$ *to prepare and compute the DFT's, and* γ_4 *is the number of operations in* $\mathbb{Z}_q$ *for back substitution.*

Proof. Follows from discussion above. □

5 Lattices and The Closest Vector Problem

Now we turn our attention to lattices. Lattices are becoming more and more prevalent in cryptography. This is because research suggest that lattice problems are robust against quantum computation. Regev showed that breaking his cryptosystem was as hard as a worst case lattice problem. Hence, it makes sense for us to look at how lattices can be used to solve LWE.

First, we will look at some basic lattice properties, before showing how the Learning with Errors problem can be reduced to a Closest Vector Problem (CVP) in a lattice. There exists several CVP algorithms, like enumeration, nearest plane and the round-off algorithms. Common for all of them is that they work best for a lattice with a "nice" basis. By "nice" we mean that the basis consist of short vectors ordered roughly according to length and that they are as orthogonal as possible. This will not automatically be the case. However, there has been a lot of research done on creating algorithms that reduces a lattice basis to a nice basis, we will look at two of them. In Chapter 6, we will look at the LLL algorithm, and and in Chapter 7, we will look at a generalisation for LLL, the BKZ algorithm.

For the LLL and BKZ algorithm we do not make many assumption about the lattices we use. However, in the last chapter, Chapter 8, we will look at lattices with a lot more algebraic structure, called principal ideal lattices. These are lattices generated by principal ideals in rings, and which provide us with some nice properties. Specifically, we will use algebraic number theory to "build" a lattice based on the *cyclotomic units*. Then we will use this structure to show how we can apply the round-off algorithm on a common problem in the ring version of LWE (Ring-LWE), called the Shortest Generator Principal Ideal Problem (SG-PIP).

First, however, we will review some lattice basics.

5.1 Lattice Basics

Here we will only go through some basic properties of lattices that are essential for this thesis. Most of the material here is based on the lecture notes in Cryptography on Public Key Encryption by Gjøsteen ([Gjø19b]).

Definition 5.1. A *lattice* Λ is a subgroup of $\mathbb{R}^m$

$$\Lambda = \left\{ \sum_{i=1}^{n} a_i \mathbf{b}_i \;\middle|\; a_i \in \mathbb{Z} \right\},$$

where $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{R}^m$ are linearly independent vectors.

The vectors $\mathbf{b}_i$ form the *basis* of the lattice. We can construct a $n \times m$ basis matrix $\mathbf{B} = (\mathbf{b}_1, ..., \mathbf{b}_n)$. Then

$$\Lambda = \Lambda(\mathbf{B}) = \{\mathbf{B}\boldsymbol{a} \mid \boldsymbol{a} \in \mathbb{Z}^n\}.$$

If $n = m$, Λ is *full rank*.

The *dual basis* $\mathbf{B}^\vee = \{\mathbf{b}_1^\vee, ..., \mathbf{b}_n^\vee\} \subset \text{ span}(\mathbf{B})$ and dual lattice $\Lambda(\mathbf{B}^\vee)$ of $\Lambda(\mathbf{B})$, are defined to satisfy $\langle \mathbf{b}_i^\vee, \mathbf{b}_j \rangle = \delta_{ij}$ where δ is the Kronecker delta. Hence, $\mathbf{B}^T \cdot \mathbf{B}^\vee = I$.

Note that from here until Chapter 8, Λ will be a lattice of rank n with a basis $\mathbf{B} = (\mathbf{b}_1, ..., \mathbf{b}_n) \in \mathbb{R}^{m \times n}$.

Definition 5.2. The *determinant of a lattice* Λ with basis matrix $\mathbf{B}$ is defined by

$$\det(\Lambda) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}.$$

Note that the determinant does not depend on the choice of basis.

For our reduction algorithms, we will work with particularly with one specific type of lattice:

Definition 5.3. The lattice Λ is *q-ary* if $q\mathbb{Z}^n \subseteq \Lambda \subseteq \mathbb{Z}^n$

5.1.1 The Closest Vector Problem

Here we will give some basic definitions that are essential for understanding how LWE and lattices relate.

Definition 5.4. Let Λ be a lattice. The i^{th}-*succesive minimum* $\lambda_i(\Lambda)$ of Λ is the minimal real number such that there are i linearly independent vectors of length $\lambda_i(\Lambda)$ in Λ.

Note that as long as it is clear what lattice we are talking about, we will not write the brackets, that is, $\lambda_i = \lambda_i(\Lambda)$. For the most part, we will be concerned with the shortest vector in the lattice. Therefore we let $\lambda(\Lambda)$ denote the length of the shortest vector in the lattice. That is,

$$\lambda = \lambda(\Lambda) = \lambda_1(\Lambda) = \min_{\boldsymbol{x} \in \Lambda} \|\boldsymbol{x}\| .$$

The only exception to this is in the last chapter, where λ will denote the eigenvalues of a matrix. We chose to use the same notation since we will not use the shortest vector in Chapter 8 and λ is standard notation for both the shortest vector in a lattice and the eigenvalues of a matrix. It will be very clear from context what is meant, and should not cause any confusion.

Now we can define two problems, where the latter is very relevant in this thesis:

Definition 5.5 (Shortest Vector Problem - SVP)**.** The *shortest vector problem* for a lattice $\Lambda \subseteq \mathbb{R}^m$ is to find a vector $\boldsymbol{x} \in \Lambda$ such that $\|\boldsymbol{x}\| = \lambda(\Lambda)$.

Definition 5.6 (Closest Vector Problem - CVP)**.** The *closest vector problem* for a lattice $\Lambda \subseteq \mathbb{R}^m$ and a point $\boldsymbol{z} \in \mathbb{R}^m$, is to find $\boldsymbol{x} \in \Lambda$ such that for all $\boldsymbol{y} \in \Lambda$, $\|\boldsymbol{z} - \boldsymbol{x}\| \leq \|\boldsymbol{z} - \boldsymbol{y}\|$.

That is, given a point in $\mathbb{R}^m$, we want to find the closest possible point in the lattice. If we can assume the point $\boldsymbol{z}$ is less than $\frac{\lambda(\Lambda)}{2}$ distance away from $\boldsymbol{x}$, the closest vector problem is also called a Bounded Distance

Decoding problem, or BDD. Although we will make this assumption, we will mostly refer to this as a CVP.

As mentioned in the introduction, there are several algorithms developed to solve CVP in lattices. In this chapter, we will look closer at enumeration (see Section 5.3), which will motivate our look at the LLL and BKZ algorithms in the following chapters. In Chapter 8 we will look at the round-off algorithm, which will be central for solving the SG-PIP. Common for these algorithms, however, are that they work best for a lattice basis that consist of short vectors, ordered according to length and that are as orthogonal as possible.

5.1.2 Gram-Schmidt

The Gram-Schmidt algorithm is a natural response when we state we want an orthogonal basis. However, it is not a given that the Gram-Schmidt basis itself lies in the lattice. Still, the Gram-Schmidt basis of a lattice can still be very useful when trying to find a nice basis in a lattice, and we will use it extensively in the reduction algorithms.

Here we define The Gram Schmidt basis $\mathbf{b}_1^*, ..., \mathbf{b}_n^*$ of $\mathbf{b}_1, ..., \mathbf{b}_n$ as follows:

$\mathbf{b}_1^* = \mathbf{b}_1$, then for $1 \leq j < i < n$;

$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \mathbf{b}_j^* \qquad \text{where} \qquad \mu_{ij} = \frac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}. \tag{5.1}$$

In matrix form this can be written

$$\begin{pmatrix} \mathbf{b}_1^T \\ \vdots \\ \mathbf{b}_n^T \end{pmatrix} = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ \mu_{ij} & & 1 \end{pmatrix} \begin{pmatrix} \mathbf{b}_1^{*T} \\ \vdots \\ \mathbf{b}_n^{*T} \end{pmatrix}, \tag{5.2}$$

or $\mathbf{B}^T = \mathbf{Q}\mathbf{B}^{*T}$, where $\mathbf{Q}$ is the lower triangular matrix with 1 in the diagonal and μ_{ij} elsewhere.

From the orthogonality of the Gram-Schmidt basis and the fact that the determinant of a lattice does not depend on choice of basis, it follows that

$$\det(\Lambda) = \prod_{i=1}^{n} \|\mathbf{b}_i^*\| .$$

5.2 The Lattice Attack

Recall, the LWE problem is, given $(\mathbf{A}, \boldsymbol{t} = \mathbf{A}^T\mathbf{s} + \boldsymbol{\nu} \pmod q)$, can we find $\mathbf{s}$? Here we will show that the LWE problem can be reduced to CVP in a q-ary lattice.

Define the lattice:

$$\Lambda_q(\mathbf{A}^T) = \left\{ \boldsymbol{x} \in \mathbb{Z}^m \mid \exists\ \boldsymbol{y} \in \mathbb{Z}^n \text{ s.t. } \mathbf{A}^T\boldsymbol{y} \equiv \boldsymbol{x} \pmod q \right\}$$

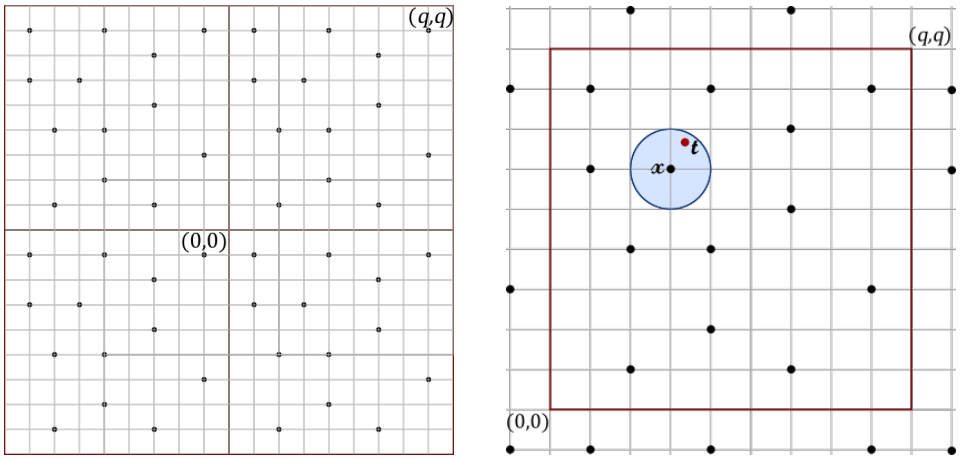
It is easy to see that this is a q-ary lattice: For any $q\boldsymbol{x} \in q\mathbb{Z}^n$, $\mathbf{0} \in \mathbb{Z}^m$ is such that $\mathbf{A}^T\mathbf{0} \equiv q\boldsymbol{x} \equiv \mathbf{0} \pmod q$, hence $q\boldsymbol{x} \in \Lambda_q(\mathbf{A}^T)$. So $q\mathbb{Z} \subseteq \Lambda_q(\mathbf{A}^T)$. A nice property of a q-ary lattice is that it is periodic for q. In Figure 2a we have given a simple example of what we mean by this in 2 dimensions.

Obviously, $\mathbf{A}^T\mathbf{s} \in \Lambda_q(\mathbf{A}^T)$. We assume $\|\boldsymbol{\nu}\| < \frac{\lambda}{2}$, then the closest lattice point to $\boldsymbol{t}$ in $\Lambda_q(\mathbf{A}^T)$ is $\mathbf{A}^T\mathbf{s}$, see Figure 2b. Hence, if we can find $\boldsymbol{x} \in \Lambda_q(\mathbf{A}^T)$ such that $\boldsymbol{x}$ is the closest vector to $\boldsymbol{t}$ in $\Lambda_q(\mathbf{A}^T)$. Then $\boldsymbol{x} - \boldsymbol{t} = \boldsymbol{\nu} \pmod q$, and we can easily find $\mathbf{s}$. Hence, we are left with a CVP.

Next we will be looking at the enumeration algorithm, which is both a useful tool in finding the closest vector, but is also used in lattice reduction algorithms such as BKZ.

5.3 Enumeration

One approach to finding short vectors is to set some upper bound $R > 0$ and go through all linear combinations of the basis vectors to find the ones with a norm less than or equal to R. That is, we want to find $\boldsymbol{a} = (a_1, ..., a_n) \in \mathbb{Z}$ such that $\|\mathbf{B}\boldsymbol{a}\| \leq R$. This is called *enumeration*.



(a) q-ary lattice **(b)** Closest vector problem

Figure 2: **(a)** A q-ary lattice is repetitive around q. That means we can divide the lattice into q-"boxes" and limit our search to one box. **(b)** We assume $\boldsymbol{t}$ lies within a $\lambda_1/2$ radius to a lattice point (illustrated by the circle). The aim is to find $\boldsymbol{x} \in \Lambda_q(\mathbf{A}^T)$.

We can also use enumeration for solving a CVP. If $\boldsymbol{x}$ is the closest vector to the point $\boldsymbol{t}$, then, if we have an estimate for what the closest vector is and an upper bound R for how far $\boldsymbol{x}$ is from the estimate, we can use enumeration of short vectors to find $\boldsymbol{x}$. We also use enumeration in the BKZ method in order to find short vectors in smaller "blocks" of the input lattice.

Let $\mathbf{b}_1, ..., \mathbf{b}_n$ be the basis of the lattice $\Lambda \subset \mathbb{R}^m$ and let $\mathbf{b}_1^*, ..., \mathbf{b}_n^*$ be the corresponding Gram-Schmidt basis. Every vector $\boldsymbol{x} \in \Lambda$ can be written as a linear combination of the bases:

$$\boldsymbol{x} = \sum_{i=1}^{n} a_i \mathbf{b}_i = \sum_{i=1}^{n} \alpha_i \mathbf{b}_i^*, \tag{5.3}$$

where $a_i \in \mathbb{Z}$ and $\alpha_i \in \mathbb{R}$ for all $i = 1, ..., n$.

Writing each $\mathbf{b}_i$ in terms of the Gram-Schmidt vectors, and rearranging

in terms of $\mathbf{b}_i^*$ gives the expression

$$\boldsymbol{x} = \sum_{i=1}^{n} \left(a_i + \sum_{j=i+1}^{n} a_j \mu_{ji} \right) \mathbf{b}_i^* = \sum_{i=1}^{n} \alpha_i \mathbf{b}_i^*. \tag{5.4}$$

Clearly, $a_n = \alpha_n \in \mathbb{Z}$. Hence, we start by enumerating all vectors where $\|\alpha_n \mathbf{b}_n^*\| \leq R$, which is equivalent to all the $a_n \in \mathbb{Z}$ that satisfies $|a_n| \leq R/\|\mathbf{b}_n^*\|$. For each of these a_n, we move on to find a_{n-1} that satisfy

$$\left\|\alpha_{n-1}\mathbf{b}_{n-1}^* + \alpha_n \mathbf{b}_n^*\right\|^2 \leq |\alpha_{n-1}|^2 \left\|\mathbf{b}_{n-1}^*\right\|^2 + |\alpha_n|^2 \|\mathbf{b}_n^*\|^2 \leq R^2.$$

Using Equation (5.4), this is equivalent to

$$|a_{n-1} + a_n \mu_{n,n-1}|^2 \left\|\mathbf{b}_{n-1}^*\right\|^2 + |a_n|^2 \|\mathbf{b}_n^*\|^2 \leq R^2.$$

Since $a_{n-1} \in \mathbb{Z}$ is the only unknown value, we can find all possible options that satisfies Equation (5.4). And in this manner, the algorithm continues until we have found all $a_1, ..., a_n$ such that $\|a_1\mathbf{b}_1 + ... + a_n\mathbf{b}_n\| \leq R$.

We see how this algorithm is equivalent to a tree search were the i^{th} level searches for an integer a_i's that satisfy

$$\left(a_i + \sum_{j=i+1}^{n} a_j \mu_{ji} \right)^2 \|\mathbf{b}_i^*\|^2 + \sum_{j=i+1}^{n} \left(a_j + \sum_{k=j+1}^{n} a_s \mu_{kj} \right)^2 \left\|\mathbf{b}_j^*\right\|^2 \leq R^2, \tag{5.5}$$

for given $a_{i+1}, ..., a_n$.

Note that, if at some point there does not exist an a_i such that Equation (5.5) hold, we simply put $a_i = 0$ and move on to a_{i-1}.

The possible options for the i^{th} coordinate, that is, the number of branches at the i^{th} level in our search, is upper bounded by

$$|a_i| \leq \frac{R}{\|\mathbf{b}_i^*\|}.$$

Which means that the total number of coordinates that are being enumerated is bounded above by

$$\prod_{i=1}^{n} \frac{R}{\|\mathbf{b}_i^*\|} = \frac{R^n}{\det(\Lambda)}. \tag{5.6}$$

To enumerate all points in a lattice is time consuming and requires a large amount of space, especially for large lattices (which is usually the case). There are several ways of improving this search. For example, we could update R to be the length of the shortest vector as we go along. In Computer Science terms, this is called *pruning*. It is clear, however, that the enumeration algorithm works best when it has a good upper bound and basis vectors that are short and do not grow too rapidly in size. Now we will look at two basis reduction algorithms that aims to find this for a given lattice.

6 LLL - A More Orthogonal Basis

The first lattice basis reduction algorithm we will look at is the LLL algorithm. It was introduced in 1982 and is named after its inventors; Arjen Lenstra, Hendrik Lenstra and László Lovász ([LLL82]). We will be focusing on its application to SVP and CVP. However, it a very versatile algorithm and its use is not limited to lattice problems. For example, it can also be used for factoring polynomials over the integers or rational numbers, or, given a good enough approximation, finding the minimal polynomial of an algebraic number. Both because of its wide applicability and its apparent simplicity, it has been recognised as one of the most important algorithmic achievements of the twentieth century ([NV10, p. v]).

This chapter is based on Lenstra, Lenstra, and Lovász's original article ([LLL82]), with supplementary material from [Gal12, Chapter 17], [Gjø19b] and [Reg04].

As stated in the previous chapter, the aim of a basis reduction algorithm is to find a basis $\mathbf{b}_1, ..., \mathbf{b}_n$ for the lattice $\Lambda \subset \mathbb{R}^m$ that consist of short vectors ordered roughly according to size. We also want them to be as orthogonal as possible. This is what we hope to achieve with a basis that is δ-LLL-reduced:

Definition 6.1. A lattice basis $\mathbf{b}_1, ...\mathbf{b}_n$ is *δ-LLL reduced* if its corresponding Gram-Schmidt basis $\mathbf{b}_1^*, ..., \mathbf{b}_n^*$ satisfies

- (Size-reduced)
$$|\mu_{ij}| \leq \frac{1}{2} \text{ for } 1 \leq j < i \leq n, \tag{6.1}$$
and
- (Lovász condition)
$$\delta \left\|\mathbf{b}_{i-1}^*\right\|^2 \leq \left\|\mathbf{b}_i^* + \mu_{i,i-1}\mathbf{b}_{i-1}^*\right\|^2 \text{ for } 1 < i \leq n, \tag{6.2}$$

where $\frac{1}{4} < \delta < 1$.

Why does this define a "nice" lattice as we described above? To answer this, first note that a lattice basis is "close to orthogonal" if the Gram-Schmidt vectors do not grow too rapidly [Gal12, Chapter 17.2]. We can write the length of the Gram-Schmidt vectors thus:

$$(\|\mathbf{b}_1\|, ..., \|\mathbf{b}_n\|) \leq \begin{pmatrix} \|\mathbf{b}_1^*\| & |\mu_{21}| \|\mathbf{b}_2^*\| & \dots & |\mu_{n1}| \|\mathbf{b}_1^*\| \\ 0 & \|\mathbf{b}_2^*\| & \dots & |\mu_{n2}| \|\mathbf{b}_2^*\| \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \|\mathbf{b}_n^*\| \end{pmatrix}.$$

For a size reduced basis, the Euclidean norm of any off-diagonal element is at most half the value of the element in the diagonal of the same row.

Next, we note that the Lovász condition can be written $(\delta - |\mu_{i,i-1}^2|) \left\|\mathbf{b}_{i-1}^*\right\|^2 \leq \|\mathbf{b}_i^*\|^2$. From this, we gather that the Lovász condition implies that $\mathbf{b}_i$ is not much longer than $\mathbf{b}_{i-1}$, and hence implies basis vectors of an δ-LLL-reduced lattice do not grow too rapidly and are ordered roughly according to length with the shortest one in the beginning.

Another way to look at the Lovász condition is by considering the 2×2 submatrix from the diagonal:

$$\begin{pmatrix} \left\|\mathbf{b}_{i-1}^*\right\| & |\mu_{i,i-1}| \left\|\mathbf{b}_{i-1}^*\right\| \\ 0 & \|\mathbf{b}_i^*\| \end{pmatrix}.$$

The Lovász condition requires that the first column of this matrix is shorter than its second column by a factor of δ. In our next chapter, we will see how Schnorr's BKZ algorithm is a generalisation of this for $\beta \times \beta$ blocks.

In Example (6.2) we see how both conditions are important to get a nice basis. Note that a common choice for δ is $\frac{3}{4}$.

In this chapter we will start in Section 6.1, where we go through the algorithm in detail, showing how and why it works. Then, in Section 6.2, we will prove that it does, in fact, terminate, and what its runtime is. And

lastly, in Section 6.3, we will prove that an $\frac{3}{4}$-LLL-reduced lattice basis has some nice properties for our enumeration algorithm.

6.1 The algorithm

Here we will go through the LLL algorithm in detail. First, we outline the steps of the algorithm, then we look at an example in $\mathbb{R}^2$ to get an idea of how it works. After this example, we will go through each step in detail.

6.1.1 Outline

Input: A lattice basis $\mathbf{b}_1, ..., \mathbf{b}_n$, $\frac{1}{4} < \delta < 1$

1. Compute the Gram-Schmidt basis and coefficients. Put $k := 1$.
2. If $k = 1$, $\mathbf{b}_1 = \mathbf{b}_1^*$, hence, put $k := k+1$ and continue (on this Step). If $2 \leq k \leq n$, use row operations to ensure $|\mu_{k,k-1}| \leq \frac{1}{2}$.
3. If the Lovász condition is not satisfied for k, swap $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$, and calculate the new Gram-Schmidt vectors and other affected coefficients. Put $k := k - 1$ and go back to Step 2.
4. Use row operations to ensure that $|\mu_{kj}| \leq \frac{1}{2}$ for $1 \leq j < k$. Put k:=k+1.
5. If $k = n + 1$, the algorithm terminates. Else go to Step 2.

Output: A δ-LLL-reduced basis.

Now, we demonstrate the algorithm for a simple example in $\mathbb{R}^2$.

Example 6.2. Define the full rank lattice $\Lambda \subset \mathbb{R}^2$ by the basis vectors:

$$\mathbf{b}_1 = \begin{pmatrix} 3 \\ 5 \end{pmatrix} \quad \text{and} \quad \mathbf{b}_2 = \begin{pmatrix} -2 \\ -1 \end{pmatrix}.$$

As we see in Figure 3a, the vectors are not particularly orthogonal. Still, calculating $|\mu_{21}| = \left|-\frac{11}{34}\right| \leq \frac{1}{2}$ shows that the basis is size reduced. How-

ever,

$$\|\mathbf{b}_2^* + \mu_{21}\mathbf{b}_1^*\|^2 = \|\mathbf{b}_2\|^2 = 5 < \delta \cdot 34 = \delta \|\mathbf{b}_1^*\|$$

for all $\frac{1}{4} < \delta < 1$. Hence, the Lovász condition is not satisfied and the basis is therefore not LLL-reduced.

To get an LLL-reduced basis we swap the basis vectors:

$$\mathbf{b}_1 = \begin{pmatrix} -1 \\ -2 \end{pmatrix} \quad \text{and} \quad \mathbf{b}_2 = \begin{pmatrix} 3 \\ 5 \end{pmatrix}.$$

For this new basis, $|\mu_{21}| = \left|-\frac{13}{5}\right| = |-2.2| > \frac{1}{2}$, so we reduce the basis: $\mathbf{b}_2 := \mathbf{b}_2 - \lfloor \mu_{21} \rceil \mathbf{b}_1 = \mathbf{b}_2 + 2\mathbf{b}_1 = (-1, 3)^T$. The new $|\mu_{21}|$ is equal to $\left|\frac{2-3}{5}\right| = 0.2$, which is size reduced.

Then we check that the Lovász condition is satisfied:

$$\|\mathbf{b}_2^* + \mu_{21}\mathbf{b}_1^*\|^2 = \|\mathbf{b}_2\|^2 = 10 \geq \delta \cdot 5 = \delta \|\mathbf{b}_1^*\|,$$

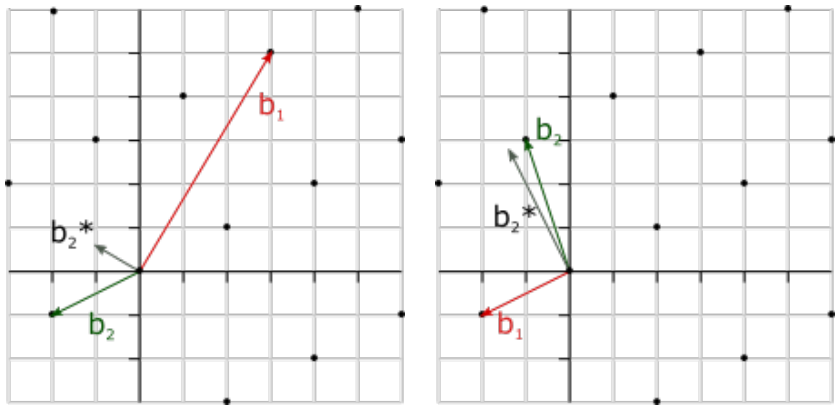
for all $\frac{1}{4} < \delta < 1$. Hence, the basis $\{(-1, -2)^T, (-1, 3)^T\}$ is a LLL-reduced basis for the lattice Λ and any δ. From Figure 3b, it is clearly more orthogonal than the original basis. Observe also that the vectors are ordered according to length and are shorter than their original basis (at least for $\mathbf{b}_2$). It is a very basic example (we could have found the same result by simple observation), but it is still worth noting that we achieved all we hoped for with the algorithm.

6.1.2 The Algorithm Steps

Here we will go through the steps of the algorithm, outlined above, in detail. As stated earlier, the input parameters are the basis vectors of the lattice, $\mathbf{b}_1, ..., \mathbf{b}_n$, and a number $\frac{1}{4} < \delta < 1$ for how much we want it reduced.

Step 1: Calculate the Gram-Schmidt basis

We start by finding the Gram Schmidt basis for the lattice as described in Section 5.1.2. We now have two bases for the lattice Λ. The relationship



(a) Size-reduced **(b)** LLL-reduced

Figure 3: Two different basis for the same lattice $\Lambda \subset \mathbb{R}^2$. The figure illustrates the importance of both conditions for the LLL algorithm.

between the two bases can be represented by the matrix system $\mathbf{B}^{*T} = \mathbf{Q}\mathbf{B}^T$ shown in Equation (5.2). It might be helpful to think of the steps of this algorithm as row operations on the augmented matrix:

$$\left(\begin{array}{ccc|c} 1 & & 0 & \mathbf{b}_1^T \\ & \ddots & & \vdots \\ \mu_{ij} & & 1 & \mathbf{b}_n^T \end{array}\right). \tag{6.3}$$

Note here that the algorithm does not need to keep track of all the vectors, but simply $\|\mathbf{b}_i^*\|$ and μ_{ij}, for $1 \leq j < i \leq n$.

When the set-up is done, the algorithm starts to iterate over a variable $1 \leq k \leq n$. Initially $k := 1$. For each k, we want to ensure that the conditions for a δ-LLL-reduced basis is satisfied. That is, for each k we want the following conditions to hold:

$$|\mu_{kj}| \leq \frac{1}{2} \text{ for } 1 \leq j < k, \tag{6.4}$$

and

$$\delta \left\|\mathbf{b}_{k-1}^*\right\|^2 \leq \left\|\mathbf{b}_k^* + \mu_{k,k-1}\mathbf{b}_{k-1}^*\right\|^2. \tag{6.5}$$

Step 2: Ensuring $|\mu_{k,k-1}| \leq \frac{1}{2}$

If $k = 1$; $\mathbf{b}_1 = \mathbf{b}_1^*$, so the conditions holds trivially. Hence we put $k := k+1$ and continue (on this step).

If $2 \leq k \leq n$; the algorithm check if $|\mu_{k,k-1}| \leq \frac{1}{2}$ is satisfied. If not, we assign the new value $\mathbf{b}_k := \mathbf{b}_k - \lfloor \mu_{k,k-1} \rceil \, \mathbf{b}_{k-1}$. This is equivalent to making the row operation $r_k \mapsto r_k - \lfloor \mu_{k,k-1} \rceil \, r_{k-1}$ to the matrix in Equation (6.3). Clearly, $|\mu_{k,k-1}| \leq \frac{1}{2}$ is now satisfied. So we move on to Step 3.

Note that we check that the Lovász condition hold, before ensuring that the rest of the Gram-Schmidt basis is size reduced for the current k. This is because if the Lovász condition is not satisfied, the algorithm manipulates the basis in a way that changes the Gram-Schmidt coefficients. Therefore, we check the Lovász condition first to avoid making any computations twice.

Step 3: The Lovász Condition

If Equation (6.5) is satisfied for $k \geq 2$ we move on to Step 4. However, if

$$\delta \left\| \mathbf{b}_{k-1}^* \right\|^2 > \left\| \mathbf{b}_k^* + \mu_{k,k-1} \mathbf{b}_{k-1}^* \right\|^2$$

the Lovász condition does not hold and we need to manipulate the basis. We do this by swapping $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$, while leaving the other $\mathbf{b}_i$'s unchanged. This swap means that everything connected with $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$ also changes. Hence, we have to calculate new values for what was $\mathbf{b}_{k-1}^*$, $\mathbf{b}_k^*$, $\mu_{k,k-1}$, $\mu_{k-1,j}$, μ_{kj}, $\mu_{i,k-1}$ and $\mu_{i,k}$ for $j < k-1$ and $i > k$. In Section 6.1.3, we show how these calculations are done and why swapping the basis vectors means the Lovász condition is satisfied for the new $\mathbf{b}_{k-1}^*$ and $\mathbf{b}_k^*$. But for now, we will focus on the algorithm. Since $\mathbf{b}_{k-1}^*$ has a new value, we are no longer sure that $\mathbf{b}_{k-2}^*$ and $\mathbf{b}_{k-1}^*$ satisfies the Lovász condition, nor are we sure that the new $\mu_{k,k-1}$ is size reduced. Therefore we put $k := k-1$ and go to Step 2.

Step 4: Size reduction

At this point in the algorithm we can assume Equation (6.5) holds for all positive integers less than or equal to k. Hence we move on to ensure the basis is size reduced, i.e. that Equation (6.4) holds for the current k. This is done in a similar way to what we did for $\mu_{k,k-1}$. Iterate over μ_{kj} for j in reverse order from $k-1$ down to 1. If $|\mu_{kj}| > \frac{1}{2}$, calculate and assign the new value $\mathbf{b}_k := \mathbf{b}_k - \lfloor \mu_{kj} \rceil \, \mathbf{b}_j$. This is equivalent to the row operation $r_k \mapsto r_k - \lfloor \mu_{kj} \rceil \, r_j$ to the augmented matrix in Equation (6.3) (meaning only μ_{kl} with $l < j$ gets modified). At the end of this step, both Equation (6.4) and (6.5) holds for the current k. Lastly, increment $k := k + 1$ continue to Step 5.

Step 5: Check for termination

If $k \leq n$, go to Step 2. Else, if $k > n$, the algorithm terminates and returns an δ-LLL-reduced basis. In Section 6.2.1, we prove that the algorithm always terminates.

6.1.3 Proof That Swapping The Basis Vectors Satisfies The Lovász Condition

Here we will spend some time on the finer details of Step 3 in the LLL algorithm. We will look at why swapping $\mathbf{b}_{k-1}$ and $\mathbf{b}_k$ leads to the Lovász condition being satisfied, and also what the updated values for $\mathbf{b}^*_{k-1}$, $\mathbf{b}^*_k$, $\mu_{k,k-1}$, $\mu_{k-1,j}$, μ_{kj}, $\mu_{i,k-1}$ and $\mu_{i,k}$, for $j < k-1$ and $i > k$, are. Recall that for Step 3, we know $2 \leq k \leq n$.

The algorithm works as follows; define a new set of basis vectors $\boldsymbol{c}_i$ where $\boldsymbol{c}_i = \mathbf{b}_i$ for $i \neq k-1, k$ and

$$\boldsymbol{c}_{k-1} = \mathbf{b}_k \quad \text{and} \quad \boldsymbol{c}_k = \mathbf{b}_{k-1}$$

Also let

$$\eta_{ij} = \frac{\langle \boldsymbol{c}_i, \mathbf{c}^*_j \rangle}{\langle \mathbf{c}^*_j, \mathbf{c}^*_j \rangle}$$

denote the new Gram-Schmidt coefficients.

We begin by calculating new Gram-Schmidt vectors $\mathbf{c}_{k-1}^*$ and $\mathbf{c}_k^*$ in order to check that the Lovász condition is satisfied:

$$\begin{aligned}\mathbf{c}_{k-1}^* &= \boldsymbol{c}_{k-1} - \sum_{j=1}^{k-2} \eta_{k-1,j}\mathbf{c}_j^*, \\ &= \mathbf{b}_k - \sum_{j=1}^{k-2} \mu_{kj}\mathbf{b}_j^*. \end{aligned} \tag{6.6}$$

Because, for $1 \leq j < k-1$, $\eta_{k-1,j} = \dfrac{\langle \boldsymbol{c}_{k-1}, \mathbf{c}_j^* \rangle}{\langle \mathbf{c}_j^*, \mathbf{c}_j^* \rangle} = \dfrac{\langle \mathbf{b}_k, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle} = \mu_{kj}$. Similarly, $\eta_{k,j} = \mu_{k-1,j}$. Then, note that

$$\mathbf{b}_k^* = \mathbf{b}_k - \mu_{k,k-1}\mathbf{b}_{k-1}^* - \sum_{j=1}^{k-2} \mu_{kj}\mathbf{b}_j^*. \tag{6.7}$$

Using Equation (6.7) to substitute for the summation in Equation (6.6) we get

$$\mathbf{c}_{k-1}^* = \mathbf{b}_k^* + \mu_{k,k-1}\mathbf{b}_{k-1}^*. \tag{6.8}$$

In a similar way, we find that

$$\mathbf{c}_k^* = \mathbf{b}_{k-1}^* - \eta_{k,k-1}\mathbf{c}_{k-1}^*. \tag{6.9}$$

With these new Gram-Schmidt basis vectors we can check that the Lovász condition hold for k:

$$\begin{aligned} \left\|\mathbf{b}_k^* + \mu_{k,k-1}\mathbf{b}_{k-1}^*\right\|^2 &< \delta \left\|\mathbf{b}_{k-1}^*\right\|^2, \\ \left\|\mathbf{c}_{k-1}^*\right\|^2 &< \delta \left\|\mathbf{c}_k^* + \eta_{k,k-1}\mathbf{c}_{k-1}^*\right\|^2, \\ \delta \left\|\mathbf{c}_{k-1}^*\right\|^2 &< \left\|\mathbf{c}_k^* + \eta_{k,k-1}\mathbf{c}_{k-1}^*\right\|^2, \end{aligned} \tag{6.10}$$

for $\frac{1}{4} < \delta < 1$, as required. Since we now know that the Lovász condition hold for k, we want to calculate the new Gram-Schmidt coefficients. Obviously, $\eta_{ij} = \mu_{ij}$ as long as neither i and j are equal to k or $k-1$. Above,

we showed that $\eta_{k,j} = \mu_{k-1,j}$ and $\eta_{k-1,j} = \mu_{k,j}$ for $1 \leq j < k-1$. Next,

$$\begin{aligned}
\eta_{k,k-1} &= \frac{\langle \boldsymbol{c}_k, \mathbf{c}^*_{k-1} \rangle}{\langle \mathbf{c}^*_{k-1}, \mathbf{c}^*_{k-1} \rangle}, \\
&= \frac{\langle \mathbf{b}_{k-1}, \mathbf{b}^*_k + \mu_{k,k-1}\mathbf{b}^*_{k-1} \rangle}{\langle \mathbf{c}^*_{k-1}, \mathbf{c}^*_{k-1} \rangle}, \\
&= \mu_{k,k-1} \frac{\left\| \mathbf{b}^*_{k-1} \right\|^2}{\left\| \mathbf{c}^*_{k-1} \right\|^2}.
\end{aligned}$$

For $i > k$ we calculate $\eta_{i,k-1}$ and η_{ik};

$$\begin{aligned}
\eta_{i,k-1} &= \frac{\langle \boldsymbol{c}_i, \mathbf{c}^*_{k-1} \rangle}{\langle \mathbf{c}^*_{k-1}, \mathbf{c}^*_{k-1} \rangle}, \\
&= \frac{\langle \mathbf{b}_i, \mathbf{b}^*_k \rangle}{\langle \mathbf{c}^*_{k-1}, \mathbf{c}^*_{k-1} \rangle} + \mu_{k,k-1} \frac{\langle \mathbf{b}_i, \mathbf{b}^*_{k-1} \rangle}{\langle \mathbf{c}^*_{k-1}, \mathbf{c}^*_{k-1} \rangle}, \\
&= \frac{\langle \mathbf{b}_i, \mathbf{b}^*_k \rangle}{\langle \mathbf{b}^*_k, \mathbf{b}^*_k \rangle} \frac{\langle \mathbf{b}^*_k, \mathbf{b}^*_k \rangle}{\langle \mathbf{c}^*_{k-1}, \mathbf{c}^*_{k-1} \rangle} + \mu_{k,k-1} \frac{\langle \mathbf{b}_i, \mathbf{b}^*_{k-1} \rangle}{\langle \mathbf{b}^*_{k-1}, \mathbf{b}^*_{k-1} \rangle} \frac{\langle \mathbf{b}^*_{k-1}, \mathbf{b}^*_{k-1} \rangle}{\langle \mathbf{c}^*_{k-1}, \mathbf{c}^*_{k-1} \rangle}, \\
&= \mu_{ik} \frac{\left\| \mathbf{b}^*_k \right\|^2}{\left\| \mathbf{c}^*_{k-1} \right\|^2} + \mu_{i,k-1}\eta_{k,k-1}.
\end{aligned}$$

To calculate η_{ik}, we note that we can use Equations (6.8) in (6.9) to get an expression for $\mathbf{c}^*_k$ in terms of $\mathbf{b}^*_{k-1}$ and $\mathbf{b}^*_k$:

$$\mathbf{c}^*_k = (1 - \mu_{k,k-1}\eta_{k,k-1})\mathbf{b}^*_{k-1} - \eta_{k,k-1}\mathbf{b}^*_k = \frac{\left\| \mathbf{b}^*_k \right\|^2}{\left\| \mathbf{c}^*_{k-1} \right\|^2}\mathbf{b}^*_{k-1} - \eta_{k,k-1}\mathbf{b}^*_k$$

An immediate consequence of this is that

$$\left\| \mathbf{c}^*_k \right\|^2 = \frac{\left\| \mathbf{b}^*_k \right\|^2 \left\| \mathbf{b}^*_{k-1} \right\|^2}{\left\| \mathbf{c}^*_{k-1} \right\|^4} (\left\| \mathbf{b}^*_k \right\|^2 + \mu^2_{k,k-1} \left\| \mathbf{b}^*_{k-1} \right\|^2) = \frac{\left\| \mathbf{b}^*_k \right\|^2 \left\| \mathbf{b}^*_{k-1} \right\|^2}{\left\| \mathbf{c}^*_{k-1} \right\|^2}. \tag{6.11}$$

Use this to calculate η_{ik};

$$\begin{aligned}
\eta_{ik} &= \frac{\langle \boldsymbol{c}_i, \mathbf{c}_k^* \rangle}{\langle \mathbf{c}_k^*, \mathbf{c}_k^* \rangle}, \\
&= \frac{\|\mathbf{b}_k^*\|^2}{\left\|\mathbf{c}_{k-1}^*\right\|^2} \frac{\langle \mathbf{b}_i, \mathbf{b}_{k-1}^* \rangle}{\langle \mathbf{c}_k^*, \mathbf{c}_k^* \rangle} - \eta_{k,k-1} \frac{\langle \mathbf{b}_i, \mathbf{b}_k^* \rangle}{\langle \mathbf{c}_k^*, \mathbf{c}_k^* \rangle}, \\
&= \frac{\|\mathbf{b}_k^*\|^2}{\left\|\mathbf{c}_{k-1}^*\right\|^2} \frac{\langle \mathbf{b}_i, \mathbf{b}_{k-1}^* \rangle}{\langle \mathbf{b}_{k-1}^*, \mathbf{b}_{k-1}^* \rangle} \frac{\langle \mathbf{b}_{k-1}^*, \mathbf{b}_{k-1}^* \rangle}{\langle \mathbf{c}_k^*, \mathbf{c}_k^* \rangle} - \eta_{k,k-1} \frac{\langle \mathbf{b}_i, \mathbf{b}_k^* \rangle}{\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle} \frac{\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle}{\langle \mathbf{c}_k^*, \mathbf{c}_k^* \rangle}, \\
&= \frac{\left\|\mathbf{b}_{k-1}^*\right\|^2 \|\mathbf{b}_k^*\|^2}{\left\|\mathbf{c}_{k-1}^*\right\|^2 \|\mathbf{c}_k^*\|^2} \left(\mu_{i,k-1} - \mu_{k,k-1}\mu_{ik}\right), \\
&= \mu_{i,k-1} - \mu_{k,k-1}\mu_{ik}.
\end{aligned}$$

Now we have a new system of equations:

$$\begin{pmatrix} \boldsymbol{c}_1^T \\ \vdots \\ \boldsymbol{c}_n^T \end{pmatrix} = \begin{pmatrix} 1 & & \\ & \ddots & \\ \eta_{ij} & & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{c}_1^{*T} \\ \vdots \\ \boldsymbol{c}_n^{*T} \end{pmatrix}$$

Or, in terms of the old basis vectors, we see where we keep the old values and where we have calculated new values:

$$\begin{pmatrix} \mathbf{b}_1^T \\ \vdots \\ \mathbf{b}_{k-2}^T \\ \mathbf{b}_k^T \\ \mathbf{b}_{k-1}^T \\ \mathbf{b}_{k+1}^T \\ \mathbf{b}_{k+2}^T \\ \vdots \\ \mathbf{b}_n \end{pmatrix} = \begin{pmatrix} 1 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ \mu_{k-2,1} & \dots & 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ \mu_{k,1} & \dots & \mu_{k,k-2} & 1 & 0 & 0 & 0 & \dots & 0 \\ \mu_{k-1,1} & \dots & \mu_{k-1,k-2} & \eta_{k,k-1} & 1 & 0 & 0 & \dots & 0 \\ \mu_{k+1,1} & \dots & \mu_{k+1,k-2} & \eta_{k+1,k-1} & \eta_{k+1,k} & 1 & 0 & \dots & 0 \\ \mu_{k+2,1} & \dots & \mu_{k+2,k-2} & \eta_{k+2,k-1} & \eta_{k+2,k} & \mu_{k+2,k+1} & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu_{n1} & \dots & \mu_{n,k-2} & \eta_{n,k-1} & \eta_{nk} & \mu_{n,k+1} & \mu_{n,k+2} & \dots & 1 \end{pmatrix} \begin{pmatrix} \mathbf{b}_1^{*T} \\ \vdots \\ \mathbf{b}_{k-2}^{*T} \\ \boldsymbol{c}_{k-1}^{*T} \\ \boldsymbol{c}_k^{*T} \\ \mathbf{b}_{k+1}^{*T} \\ \mathbf{b}_{k+2}^{*T} \\ \vdots \\ \boldsymbol{c}_n^{*T} \end{pmatrix}$$

6.2 Analysis

In this section we will prove that the algorithm terminates and what its runtime is.

6.2.1 Termination

Theorem 6.3. *Let $\Lambda \subseteq \mathbb{Z}^m$ be a lattice with basis $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$, where $n \leq m$. Let $\frac{1}{4} < \delta < 1$. Then, when inputting $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ and δ, the LLL algorithm will terminate.*

This proof follows the arguments used in [LLL82] and in the proof of Theorem 17.5.1 in [Gal12]. Note here that Lenstra, Lenstra, and Lovász proved that the LLL algorithm terminates for any lattice $\Lambda \subseteq \mathbb{R}^m$. The proof here is equivalent to that, except for in the last argument, where we use the fact that $\mathbf{b}_i \in \mathbb{Z}^m$. Since, for the LWE problem, we are only concerned with integer lattices, we choose to just prove the special case.

Proof. Step 4 of the algorithm run a maximum of $k-1$ times more than Step 3. Hence it is enough to show that Step 3 will only happen a finite number of times. We introduce the quantities

$$d_i = \det(\mathbf{b}_1, ..., \mathbf{b}_i) = \prod_{j=1}^{i} \left\|\mathbf{b}_j^*\right\|$$

and

$$D = \prod_{i=1}^{n-1} d_i = \prod_{i=1}^{n-1} \|\mathbf{b}_i^*\|^{n-i}.$$

D is called the *potential* of Λ and maps the length of the lattice basis to a positive number. The lower D is, the shorter our Gram-Schmidt basis is. Note how it gives more weight to the first basis vectors than to the last, which means having shorter Gram-Schmidt vectors at the beginning means more than at the end of the basis. Also note that D is bounded above for $\rho \geq \max_i \|\mathbf{b}_i\|$ by

$$D = \prod_{i=1}^{n-1} \|\mathbf{b}_i^*\|^{n-i} \leq \prod_{i=1}^{n-1} \rho^{n-i} = \rho^{\frac{n(n-1)}{2}}.$$

D only changes if the Gram-Schmidt basis of the lattice changes, which only happens in Step 3. Let $\tilde{d}_i$ denote the value of d_i after swapping basis

vectors as described in Step 3. Clearly, $\tilde{d}_i = d_i$ for all $i < k-1$, and recall, from Equation (6.10), that $\left\|\mathbf{c}_{k-1}^*\right\| \|\mathbf{c}_k^*\| = \left\|\mathbf{b}_{k-1}^*\right\| \|\mathbf{b}_k^*\|$. This implies that $\tilde{d}_i = d_i$ for all $i \neq k-1$. Next calculate

$$\tilde{d}_{k-1} = \prod_{j=1}^{k-1} \|\mathbf{c}_j^*\| = \left(\prod_{j=1}^{k-1} \|\mathbf{b}_j^*\|\right) \left\|\mathbf{c}_{k-1}^*\right\| \leq \sqrt{\delta} d_{k-1}$$

by the Lovász condition (see Equation (6.11)). This implies that for each swap, D decreases with a factor of $\sqrt{\delta}$, that is $\tilde{D} \leq \sqrt{\delta} D$. Since D only changes when we swap the basis, we have proved that D is bounded above and is strictly decreasing throughout the algorithm. If we can prove D bounded below by a positive value, we prove that the LLL algorithm terminates.

Since $\Lambda \in \mathbb{Z}^m$, it follows that $d_i \in \mathbb{Z}$ and also that $d_i \geq 1$. This again implies that $D \geq 1$ and hence we have a lower bound on D and the algorithm terminates. □

6.2.2 Runtime

We have now showed how the algorithm works and that it does, in fact, terminate. Lastly in this analysis section, we will look at its running time.

Proposition 6.4. *Let $\Lambda \subset \mathbb{Z}^n$ be a lattice with basis , $\boldsymbol{b}_1, .., \boldsymbol{b}_n$ and let $\rho \in \mathbb{R}$, be such that $\rho \geq \max\left(2,\ \max_i \|\boldsymbol{b}_i\|\right)$. Then the number of arithmetic operations needed by the LLL algorithm is $O(n^4 \log \rho)$.*

Proof. The initialisation of the algorithm, that is Step 1 (finding the Gram-Schmidt basis vectors), requires $O(n^3)$ arithmetic operations, but only occurs once throughout the run of the algorithm.

We will now look at how many times we run through Step 3 and 4 and how many arithmetic operation each instance require. From the fact that $D \leq \rho^{\frac{n(n-1)}{2}}$, it follows that $0 \leq \log D \leq \frac{n(n-1)}{2} \log \rho$. Hence, the number of times we run through Step 3 is upper bounded by $O(n^2 \log \rho)$. Also,

since we run through Step 4 a maximum of $k-1$ times more than Step 3, the number of times we run through Step 4 is also bounded above by $O(n^2 \log \rho)$.

Now we look at how many arithmetic operation each step requires. Both Step 2 and 3 are bounded above by $O(n)$ operations. For Step 4, each time we reduce μ_{kj} for $1 \leq j \leq k-1$, we do a maximum of n operations, and we have to perform a maximum of n of these reduction, hence Step 2 requires $O(n^2)$ operations.

Putting it all together, we get that the algorithm is $O(n^4 \log \rho)$. □

6.3 Bounds

A good measure of how good our basis reduction is, is how close the length of our shortest basis vector $\mathbf{b}_1$ is to the shortest lattice length λ. This is because the closer $\mathbf{b}_1$ is to λ, the better bound R we have on our enumeration process that aims to find short vectors around a point in the lattice. This entire section is devoted to prove the following bound on a $\frac{3}{4}$-LLL reduced basis:

Theorem 6.5. *Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ be a $\frac{3}{4}$-LLL-reduced basis for the lattice $\Lambda \subset \mathbb{R}^m$. λ_i is the i^{th} successive minimum of Λ. Then*

$$2^{\frac{1-i}{2}} \lambda_i \leq \|\boldsymbol{b}_i\| \leq 2^{\frac{n-1}{2}} \lambda_i,$$

for $1 \leq i \leq n$.

We begin by finding a bound on the basis vectors of an LLL-reduced basis:

Proposition 6.6. *Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ be an $\frac{3}{4}$-LLL-reduced basis for a lattice $\Lambda \subset \mathbb{R}^m$ and let $\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^*$ be the corresponding Gram-Schmidt basis. Then*

$$\|\boldsymbol{b}_j\|^2 \leq 2^{i-1} \|\boldsymbol{b}_i^*\|^2$$

for $1 \leq j \leq i \leq n$.

Proof. By definition, a $\frac{3}{4}$-LLL-reduced basis will satisfy

$$\begin{aligned}
\left\|\mathbf{b}_{i-1}^*\right\|^2 &\leq 2\left\|\mathbf{b}_i^*\right\|^2 \\
\left\|\mathbf{b}_{i-2}^*\right\|^2 &\leq 2^2\left\|\mathbf{b}_i^*\right\|^2 \\
&\vdots \\
\left\|\mathbf{b}_j^*\right\|^2 &\leq 2^{i-j}\left\|\mathbf{b}_i^*\right\|^2
\end{aligned} \tag{6.12}$$

for $1 \leq j \leq i \leq n$. By the fact that the Gram-Schmidt vectors are orthogonal to each other we know

$$\begin{aligned}
\left\|\mathbf{b}_j\right\|^2 &\leq \left\|\mathbf{b}_j^*\right\|^2 + \sum_{k=1}^{j-1} \mu_{jk}^2 \left\|\mathbf{b}_k^*\right\|^2, \\
&\leq \left\|\mathbf{b}_j^*\right\|^2 + \sum_{k=1}^{j-1} \frac{1}{4} 2^{j-k} \left\|\mathbf{b}_j^*\right\|^2, \\
&= \left\|\mathbf{b}_j^*\right\|^2 \left(1 + \left(\frac{2^j}{4}\right) \sum_{k=1}^{j-1} 2^{-k}\right), \\
&= \left\|\mathbf{b}_j^*\right\|^2 \left(1 + \frac{2^j}{4}\left(\frac{1-2^{-j}}{1-2^{-1}} - 1\right)\right), \\
&= \left\|\mathbf{b}_j^*\right\|^2 \left(1 + \frac{1}{4}\left(2^j - 2\right)\right), \\
&\leq 2^{j-1} \left\|\mathbf{b}_j^*\right\|^2.
\end{aligned}$$

Now, using Equation (6.12), we get

$$\left\|\mathbf{b}_j\right\| \leq 2^{j-1}2^{i-j} \left\|\mathbf{b}_i^*\right\|^2 = 2^{i-1} \left\|\mathbf{b}_i^*\right\|^2.$$

□

Here we will briefly remark on a result that follows immediately from the proposition above and the fact that $\|\mathbf{b}_i^*\| \leq \|\mathbf{b}_i\|$.

Corollary 6.7. *Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ be an $\frac{3}{4}$-LLL-reduced basis for a lattice $\Lambda \subset \mathbb{R}^m$ and let $\boldsymbol{b}_1^*, ..., \boldsymbol{b}_n^*$ be the corresponding Gram-Schmidt basis. Then we*

have

$$|\det(\Lambda)| \leq \prod_{i=1}^{n} \|\boldsymbol{b}_i\| \leq 2^{\frac{n(n-1)}{4}} |\det(\Lambda)|, \tag{6.13}$$

$$\|\boldsymbol{b}_1\| \leq 2^{\frac{n-1}{4}} |\det(\Lambda)|^{\frac{1}{rb}}. \tag{6.14}$$

Since we assume $\mathbf{b}_1$ is roughly the shortest vector in the basis it makes sense to put the enumeration bound R to be $\|\mathbf{b}_1\|$. Recall that the number of enumerations we have to do for an upper bound R is $R^n/\det(\Lambda)$. For a $\frac{3}{4}$-LLL-reduced basis, we can use use Equation (6.14) to give a new estimate, that only depend on n, on the upper bound on the number of enumerations:

$$\frac{R^n}{\det(\Lambda)} = \frac{\|\mathbf{b}_1\|^n}{\det(\Lambda)} \leq 2^{\frac{n(n-1)}{4}}.$$

Returning to Theorem 6.5, we will see in the proof of the theorem that Proposition 6.6 is enough to prove the lower bound. We therefore turn our attention to proving the upper bound in the theorem. By definition, there exist some linearly independent set of vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_i \in \Lambda$ such that $\lambda_i = \max(\|\boldsymbol{x}_1\|, ..., \|\boldsymbol{x}_i\|)$. Hence we aim to bound the basis vectors $\mathbf{b}_1, ..., \mathbf{b}_n$ above by any linearly independent set of vectors in the lattice, and use this to prove the upper bound in Theorem 6.5.

First, we prove a bound on $\mathbf{b}_1$:

Proposition 6.8. *Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ be the $\frac{3}{4}$-LLL-reduced basis for the lattice $\Lambda \subset \mathbb{R}^m$. Then*

$$\|\boldsymbol{b}_1\|^2 \leq 2^{n-1} \|\boldsymbol{x}\|^2,$$

for every non-zero $\boldsymbol{x} \in \Lambda$.

Proof. Every $\boldsymbol{x} \in \Lambda$ can be written as a linear combination of the basis vectors $\mathbf{b}_1, ..., \mathbf{b}_n$. Let $\mathbf{b}_1^*, ..., \mathbf{b}_n^*$ be the corresponding Gram-Schmidt basis. Then for all $\boldsymbol{x} \in \Lambda$ there exists $a_i \in \mathbb{Z}$ and $\alpha_i \in \mathbb{R}$ such that

$$\boldsymbol{x} = \sum_{i=1}^{n} a_i \mathbf{b}_i = \sum_{i=1}^{n} \alpha_i \mathbf{b}_i^*.$$

Let k be the largest index for which $a_k \neq 0$. Then $a_k = \alpha_k$. This implies that

$$\begin{aligned} \|\boldsymbol{x}\|^2 &\geq \alpha_k^2 \|\mathbf{b}_k^*\|^2 \geq \|\mathbf{b}_k^*\|^2, \qquad (6.15) \\ 2^{k-1} \|\boldsymbol{x}\|^2 &\geq 2^{k-1} \|\mathbf{b}_k^*\|^2, \\ 2^{n-1} \|\boldsymbol{x}\|^2 &\geq \|\mathbf{b}_1\|^2, \end{aligned}$$

by Proposition 6.6. □

Proposition 6.9. *Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ be a $\frac{3}{4}$-LLL-reduced basis for the lattice $\Lambda \subset \mathbb{R}^m$. Let $\boldsymbol{x}_1, ..., \boldsymbol{x}_t \in \Lambda$ be linearly independent vectors. Then*

$$\|\boldsymbol{b}_j\|^2 \leq 2^{n-1} \max\{\|\boldsymbol{x}_1\|^2, ..., \|\boldsymbol{x}_t\|^2\},$$

for $j = 1, ..., t$.

Proof. Let $\boldsymbol{x}_1, ..., \boldsymbol{x}_t$ be $t \leq n$ linearly independent vectors in the lattice. For each $\boldsymbol{x}_j$, we can write

$$\boldsymbol{x}_i = \sum_{j=1}^{n} a_{ij} \mathbf{b}_j$$

for some coefficients $a_{ij} \in \mathbb{Z}$ for $1 \leq i \leq t$.

For fixed i, denote $k(i)$ as the index of the largest non-zero coefficient such that $a_{i,k(i)} \neq 0$. Renumber the $\boldsymbol{x}_i$'s such that $k(1) \leq k(2) \leq ... \leq k(t)$.

Observe that for all $1 \leq i \leq t$, $i \leq k(i)$. This is because if $i > k(i)$, then $\boldsymbol{x}_1, ..., \boldsymbol{x}_j \in \text{span}(\mathbf{b}_1, ..., \mathbf{b}_{j-1})$, which contradicts the linear independence of $\boldsymbol{x}_i$. From this and Proposition 6.6, it follows that

$$\begin{aligned} \|\mathbf{b}_i\|^2 &\leq 2^{k(i)-1} \left\|\mathbf{b}_{k(i)}^*\right\|^2, \\ &\leq 2^{n-1} \left\|\mathbf{b}_{k(i)}^*\right\|^2, \\ &\leq 2^{n-1} \|\boldsymbol{x}_i\|^2. \end{aligned}$$

by Equation (6.15). □

Now we have all the bounds we need to put a bound on $\mathbf{b}_i$ in terms of the successive minima λ_i.

Proof of Theorem 6.5. We know $\lambda_i = \max(\|\boldsymbol{x}_1\|, ..., \|\boldsymbol{x}_i\|)$ for some set of linearly independent vectors $\boldsymbol{x}_1, ..., \boldsymbol{x}_i \in \Lambda$, the upper bound follows immediately from Proposition 6.9.

For the lower bound, note that

$$\lambda_i \leq \max_{1 \leq j \leq i} \|\mathbf{b}_i\| \leq 2^{\frac{i-1}{2}} \|\mathbf{b}_i^*\| \leq 2^{\frac{i-1}{2}} \|\mathbf{b}_i\|,$$

by Proposition 6.6. The result follows. □

It follows that $\lambda \leq \|\mathbf{b}_1\| \leq 2^{n-1}\lambda$.

7 Block Korkin-Zolotarev - Expanding LLL

In the LLL algorithm we looked at 2×2-blocks of the lattice Λ and made sure the first basis vector was (within a factor of δ) shorter than the next vector. The BKZ, or Block Korkin-Zolotarev, algorithm, generalises this idea to blocks of size $\beta \geq 2$.

The concept of a Korkin-Zolotarev reduced basis was coined as early as 1873 by the Russian mathematicians Aleksandr Korkin and Yegor Ivanovich Zolotarev ([KZ73]). A Korkin-Zolotarev reduced basis has better properties than an LLL-reduced basis, such as better bounds on $\|\mathbf{b}_1\|$, but there is no known polynomial time algorithm to compute it. In 1987 Schnorr introduced an algorithm for the block version of a Korkin-Zolotarev reduced basis, the BKZ algorithm.

The material in this chapter is mainly based on [Sch87], [SE94] and [CN11].

7.1 Korkin-Zolotarev Reduced Basis

Before we define the block version, we will look at what we mean by a Korkin-Zolotarev reduced basis. In order to to this, we need to introduce some notation. First, let the map $\pi_i : \mathbb{R}^n \mapsto \text{span}(\mathbf{b}_1, ..., \mathbf{b}_{i-1})^\perp$ denote the orthogonal projection such that that $\mathbf{b} - \pi_i(\mathbf{b}) \in \text{span}(\mathbf{b}_1, ..., \mathbf{b}_{i-1})$. That is, $\pi_i(\mathbf{b})$ "removes" the parts of $\mathbf{b}$ that lies in $\text{span}(\mathbf{b}_1, ..., \mathbf{b}_{i-1})$, leaving only the orthogonal parts. Note that

$$\begin{aligned}
\pi_i(\mathbf{b}_i) &= \mathbf{b}_i^*, \\
\pi_i(\mathbf{b}_{i+1}) &= \mu_{i+1,i}\mathbf{b}_i^* + \mathbf{b}_{i+1}^*, \\
\pi_i(\mathbf{b}_{i+2}) &= \mu_{i+2,i}\mathbf{b}_i^* + \mu_{i+2,i+1}\mathbf{b}_{i+1}^* + \mathbf{b}_{i+2}^*. \\
&\vdots
\end{aligned}$$

In general,

$$\pi_i(\mathbf{b}_k) = \begin{cases} \sum_{j=i}^{k-1} \mu_{k,j}\mathbf{b}_j^* + \mathbf{b}_k^* & \text{if } i \leq k \leq n \\ \\ \mathbf{0} \quad \text{if } 1 \leq k < i \end{cases}.$$

Note that π_1 of the basis vectors of a lattice will give the vector in terms of the Gram-Schmidt basis.

Define $\Lambda_i = \pi_i(\Lambda)$ to be the lattice orthogonal to $\text{span}(\mathbf{b}_1, ..., \mathbf{b}_{i-1})$. Or, in other words, Λ_i is the lattice spanned by $\{\pi_i(\mathbf{b}_i), ..., \pi_i(\mathbf{b}_n)\}$. Note that $\Lambda_1 = \Lambda$. In Figure 4 we have illustrated the different spans for Λ_i for a lattice of rank 5. As we see there, this is simply a decreasing blocks of the Gram-Schmidt basis.

$$\begin{pmatrix} \mathbf{b}_1^* & \mu_{21}\mathbf{b}_1^* & \mu_{31}\mathbf{b}_1^* & \mu_{41}\mathbf{b}_1^* & \mu_{51}\mathbf{b}_1^* \\ 0 & \mathbf{b}_2^* & \mu_{32}\mathbf{b}_2^* & \mu_{42}\mathbf{b}_2^* & \mu_{52}\mathbf{b}_2^* \\ 0 & 0 & \mathbf{b}_3^* & \mu_{43}\mathbf{b}_3^* & \mu_{52}\mathbf{b}_3^* \\ 0 & 0 & 0 & \mathbf{b}_4^* & \mu_{54}\mathbf{b}_4^* \\ 0 & 0 & 0 & 0 & \mathbf{b}_5^* \end{pmatrix} \begin{matrix} \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \\ \Lambda_5 \end{matrix}$$

Figure 4: Illustration indicating which columns that span each Λ_i for a lattice of rank $n = 5$.

Now we are ready to state what is meant by a Korkin-Zolotarev reduced basis.

Definition 7.1. An ordered basis $\mathbf{b}_1, ..., \mathbf{b}_n$ is *Korkin-Zolotarev reduced* if it is size-reduced and if

$$\|\mathbf{b}_i^*\| = \lambda(\Lambda_i)$$

for all $i = 1, ..., n$.

The idea is to put the shortest vector of each lattice Λ_i as the first basis vector. Clearly, this implies that $\|\mathbf{b}_1\| = \lambda(\Lambda)$. In general, for a Korkin-Zolotarev reduced basis we have this bound:

Theorem 7.2 ([LLS90, Theorem 2.1])**.** *Every Korkin-Zolotarev reduced basis* $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ *satisfies*

$$\frac{4}{(i+3)} \leq \frac{\|\boldsymbol{b}_i\|^2}{\lambda_i^2} \leq \frac{(i+3)}{4}$$

for $i = 1, ..., n$.

This bound is a lot stronger than the bound we found for the LLL algorithm in Theorem 6.5. However, reducing a basis to a Korkin-Zolotarev basis is the same as searching through the whole lattice for the shortest vector, which is what we are trying to avoid by basis reduction.

According to [SE94, Section 5], the fastest known algorithm for Korkin-Zolotarev reduction of a basis $\mathbf{b}_1, ..., \mathbf{b}_n \in \mathbb{Z}^m$ with $\rho = \max_i(\|\mathbf{b}_i\|^2)$ has a theoretic worst case time bound of $m^{\frac{m+o(m)}{2}} + O(m^4 \log \rho)$ arithmetic steps on $O(m \log \rho)$-bit integers. Hence, trying to break the LWE problem by Korkin-Zolotarev reducing the basis will not be very effective. Instead, we will look at Schnorr's block version of a Korkin-Zolotarev reduced basis.

7.2 BKZ - Block Korkin-Zolotarev

Schnorr introduced a block version of a Korkin-Zolotarev reduced basis in 1987 and an algorithm to achieve it. A BKZ reduction can be thought of as a generalisation of the LLL algorithm. Where, for the LLL algorithm, we compared blocks of size 2, while for the BKZ algorithm we look at blocks of size $\beta \geq 2$.

However, before we can look at what we mean by block reduced basis, we must define what we mean by a block.

Definition 7.3. Let $\mathbf{b}_1, ..., \mathbf{b}_n$ be the basis of a lattice $\Lambda \subset \mathbb{R}^m$ and let π_i denote the orthogonal projection on $\text{span}(\mathbf{b}_1, ..., \mathbf{b}_{i-1})$. Then, for $\beta \geq 2$ and $k = \min(j + \beta - 1, n)$, a block $B_{[j,k]}$ is defined to be

$$\mathbf{B}_{[j,k]} = (\pi_j(\mathbf{b}_j), ..., \pi_j(\mathbf{b}_k))$$

for $j = 1, ..., n$, and $\Lambda_{[j,k]}$ is the lattice spanned by $\mathbf{B}_{[j,k]}$.

A more intuitive way of thinking about this might be to look at the Gram-Schmidt basis of $\mathbf{b}_1, ..., \mathbf{b}_n$ and draw up the blocks $B_{[j,k]}$ that span the lattices $\Lambda_{[j,k]}$, like we have done in Figure 5.

$$\begin{pmatrix} \mathbf{b}_1^* & \mu_{2,1}\mathbf{b}_1^* & \dots & \mu_{\beta,1}\mathbf{b}_1^* & \mu_{\beta+2,1}\mathbf{b}_1^* & \dots & \mu_{n,1}\mathbf{b}_1^* \\ 0 & \mathbf{b}_2^* & \dots & \mu_{\beta,2}\mathbf{b}_2^* & \mu_{\beta+1,2}\mathbf{b}_2^* & \dots & \mu_{n,2}\mathbf{b}_2^* \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & \mathbf{b}_\beta^* & \mu_{\beta+1,\beta}\mathbf{b}_\beta^* & \dots & \mu_{n,\beta} \\ 0 & 0 & \dots & 0 & \mathbf{b}_{\beta+1}^* & \dots & \mu_{n,\beta+1}\mathbf{b}_\beta^* \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \dots & \dots & \mathbf{b}_n^* \end{pmatrix}$$

Figure 5: Illustration of the blocks of a basis

Now that we have an idea of what a block is, we can define a Block Korkin-Zolotarev reduced basis:

Definition 7.4. A basis $\mathbf{b}_1, ..., \mathbf{b}_n$ is *β-reduced* for $\beta \geq 2$ where $k = \min(j + \beta - 1, n)$ if it is size reduced and if

$$\left\|\mathbf{b}_j^*\right\| = \lambda(\Lambda_{[j,k]}), \tag{7.1}$$

for $j = 1, ..., n$.

That is, a lattice Λ is BKZ-reduced for $\beta \geq 2$ if $\mathbf{b}_j^*$ is the shortest vector of $\Lambda_{[j,k]}$. In a block reduction, we look at β-blocks around the diagonal of the Gram-Schmidt basis, like in Figure 5. If $\beta = n$ we get a Korkin-Zolotarev reduced basis, and if $\beta = 2$ the BKZ is equivalent to an LLL-reduction. From Theorem 7.2, we observe that a larger block size will lead to a more accurate result. However, a larger β also requires more computational power and longer running time.

In the next section, we will look at Schnorr's algorithm for achieving a BKZ reduced basis of a lattice. He was, however, unable to prove that it

terminates.

7.3 The Algorithm

In this section we will look Schnorr's algorithm for achieving a BKZ-reduced basis.

For this algorithm, we extend the LLL algorithm to also remove any linear dependency of its input vectors. That is, if we input a set of vectors that are not linearly independent, the LLL algorithm will still return a linearly independent, LLL-reduced set of basis vectors that span the same space. We will not discuss the details of this extension here, but it is well-defined (see e.g. [SE94, Section 4]).

There are two counters, j and z in this algorithm. j is the iteration constant and increases cyclically in the interval $j = 1, ..., n$. z keeps track of how many consecutive blocks that are β-reduced. If the algorithm encounters a block where Equation (7.1) is not satisfied, it fixes this, but puts $z := 0$. As opposed to the LLL algorithm, this algorithm does not terminate when $j = n + 1$, but rather when $z = n$. That is, when all the blocks are β-reduced.

7.3.1 Outline

Input: The LLL reduction factor δ, the lattice basis $\mathbf{b}_1, ..., \mathbf{b}_n$, and the size of the blocks $\beta \geq 2$.

1. Set $j := 0$, $z := 0$ and δ-LLL reduce the basis $\mathbf{b}_1, ..., \mathbf{b}_n$.
2. Put $j := j \pmod{n-1} + 1$, $k := \min(j + \beta - 1, n)$ and $h := \min(k + 1, n)$. Enumerate to find $\mathbf{b}^{new} = \sum_{j=i}^{k} v_i \mathbf{b}_i$, $v_i \in \mathbb{Z}$ not all zero, such that $\|\pi_j(\mathbf{b}^{new})\| = \lambda(\Lambda_{[j,k]})$.
   (a) If $\lambda(\Lambda_{[j,k]}) = \|\pi_j(\mathbf{b}^{new})\| < \left\|\mathbf{b}_j^*\right\|$, then a new shortest vector is found. Set $z := 0$ and insert the new vector at the beginning of the current block; $(\mathbf{b}_1, ..., \mathbf{b}_{j-1}, \mathbf{b}^{new}, \mathbf{b}_j, ..., \mathbf{b}_h)$. LLL reduce

this set of vectors to remove dependency, update the Gram-Schmidt basis, and get new LLL-reduced, linearly independent basis $\mathbf{b}_1, ..., \mathbf{b}_h$.

(b) Else, if $\lambda(\Lambda_{[j,k]}) = \left\|\mathbf{b}_j^*\right\|$, no new shortest vector is found, set $z := z + 1$ and LLL reduce the truncated basis $(\mathbf{b}_1, ..., \mathbf{b}_h)$.

3. If $z = n$ output $\mathbf{b}_1, ..., \mathbf{b}_n$, else if $z < n$ go to Step 2.

Output: A β-reduced basis for Λ.

7.3.2 Block reduction

Both Step 1 and 3 in the algorithm are pretty straight forward, so we will only go into slightly more detail about Step 2.

Step 2 focuses on a block $\mathbf{B}_{[j,k]}$ for $j \in \{1, ..., n-1\}$ and $k = \min(j + \beta - 1, n)$. The aim is to ensure that the shortest vector of the lattice $\Lambda_{[j,k]}$ is at the beginning of the block. To do this we also define the variable $h = \min(k + 1, n)$ to keep track of the end of the block.

First, we search to find the shortest vector in $\Lambda_{[j,k]}$. Practically, this is done by enumeration, as described in Section 5.3, with $\left\|\mathbf{b}_j^*\right\|$ as an initial upper bound and updating this if it finds a shorter vector. At the end, we have a non-zero $\mathbf{v} = (v_j, ..., v_{j+k}) \in \mathbb{Z}^k$ such that

$$\left\| \pi_j \left(\sum_{i=j}^{k} v_i \mathbf{b}_i \right) \right\| = \lambda(\Lambda_{[j,k]})$$

Otherwise, if $\left\|\mathbf{b}_j^*\right\| = \lambda(\Lambda_{[j,k]})$, LLL is called on the truncated basis $\mathbf{b}_1, ..., \mathbf{b}_h$ and we increment z by 1.

At the end of each iteration of j, the basis $\mathbf{B} = \mathbf{b}_1, ..., \mathbf{b}_n$ is such that $\mathbf{b}_1, ..., \mathbf{b}_h$ is LLL reduced. When $z = n$, we have gone through all the blocks without having to alter the basis once, hence the basis is β-reduced, the algorithm terminates and returns $\mathbf{B}$.

7.4 Analysis

Schnorr was not able to prove that his algorithm terminates, and in fact, there no good upper bound known for the complexity of BKZ. In practice, it has proved itself to work well for $\beta = 20$ while becoming problematic for $\beta = 25$. The method uses enumeration, which for $\beta \geq 30$ starts to dominate the algorithm and is too expensive for $\beta \geq 40$ ([SE94],[CN11]).

In recent years there has been a lot of work done on improving this algorithm, one of them being the BKZ 2.0 algorithm by Chen and Nguyen ([CN11]). They propose four modification to BKZ, rendering it more applicable to block sizes larger than 30. In short, these improvements are: (1.) Early-abort, that is, simply adding a parameter specifying how many iterations that should be performed. (2.) Pruning when enumerating[§]. (3.) Preprocessing the block so that the projected basis is more reduced than LLL. And lastly, (4.) optimizing choice of initial lower bound on the enumeration radius. According to [CN11], this is the best lattice reduction in practice.

7.5 Bounds

Now we want to investigate some properties of a BKZ reduced basis:

At this stage we define the *Korkin-Zolotarev constant* to be

$$\alpha_\beta = \max_{1 \leq j \leq \beta} \frac{\|\mathbf{b}_1\|^2}{\left\|\mathbf{b}_j^*\right\|^2}$$

For a Korkin-Zolotarev reduced basis $\mathbf{b}_1, ..., \mathbf{b}_\beta$.

Theorem 7.5 ([Sch87, Theorem 2.3])**.** *Every β-reduced basis $\boldsymbol{b}_1, ..., \boldsymbol{b}_n$ satisfies*

$$\|\boldsymbol{b}_1\|^2 \leq \alpha_\beta^{\frac{n-1}{\beta-1}} \lambda(\Lambda)^2,$$

provided $\beta - 1$ divides $n - 1$.

[§]Specifically Gama-Nguyen-Regev pruning introduced in [GNR10].

Proof. Let $\mathbf{v}$ be a vector in the lattice where $\|\mathbf{v}\| = \lambda(\Lambda)$. Similar to earlier, we know we can write $\mathbf{v}$ in terms of the basis vectors:

$$\mathbf{v} = \sum_{i=1}^{n} v_i \mathbf{b}_i = \sum_{i=1}^{n} \tilde{v}_i \mathbf{b}_i^*$$

where $v_i \in \mathbb{Z}$ and $\tilde{v}_i \in \mathbb{R}$. Let k be the largest integer for which $v_i \neq 0$. Then $v_k = \tilde{v}_k$ and $\lambda(\Lambda)^2 = \|\mathbf{v}\|^2 \geq \|\mathbf{b}_k^*\|^2$.

Every β-reduced basis satisfies:

$$\|\mathbf{b}_i^*\| \leq \alpha_\beta \left\|\mathbf{b}_{i+j}^*\right\|^2 \quad \text{for } j \leq \beta - 1,\ i + j \leq n.$$

By the same argument, if $r = i+j$, then $\|\mathbf{b}_r^*\| \leq \alpha_\beta \left\|\mathbf{b}_{r+s}^*\right\|$ where $s \leq \beta-1$ and $r + s \leq n$. Then $\|\mathbf{b}_i^*\|^2 \leq \alpha_\beta \left\|\mathbf{b}_{i+j}^*\right\| \leq \alpha_\beta^2 \left\|\mathbf{b}_{i+j+s}^*\right\|$ where $j + s \leq 2(\beta - 1)$ and $i + j + s \leq n$. That is, $\|\mathbf{b}_i^*\| \leq \alpha_\beta^2 \left\|\mathbf{b}_{i+j}^*\right\|$ for $j \leq 2(\beta - 1)$ and $i + j \leq n$. Repeated applications of this argument gives:

$$\|\mathbf{b}_i^*\| \leq \alpha_\beta^\tau \left\|\mathbf{b}_{i+j}^*\right\|^2 \quad \text{for } j \leq \tau(\beta - 1),\ i + j \leq n. \tag{7.2}$$

where τ is an integer. Note that $j \leq \tau(\beta - 1) \leq (n - 1)$, which implies $\tau \leq \frac{n-1}{\beta-1}$. Also, for $\mathbf{v}$, $\mathbf{b}_k^*$ will be the longest Gram-Schmidt basis vector in the basis since the basis is β-reduced. Hence we get

$$\|\mathbf{b}_1\|^2 = \|\mathbf{b}_1^*\|^2 \leq \alpha_\beta^{\frac{n-1}{\beta-1}} \|\mathbf{b}_k^*\|^2 \leq \alpha_\beta^{\frac{n-1}{\beta-1}} \lambda(\Lambda)^2,$$

as required. □

Note that $\alpha_2 = \frac{4}{3}$, which implies that for an 2-reduced basis $\|\mathbf{b}_1\| \leq \left(\frac{2}{\sqrt{3}}\right)^{n-1} \lambda(\Lambda)$, which is a better bound than what we found in Theorem 6.5.

However, α_β is not something that will be readily available. In his article, Schnorr shows how this can be bounded above by the *Hermite constant*.

Note also that if $\beta = n$, we get $\alpha_n = 1$ and we return to our our upper bound for a Korkin-Zolotarev reduced basis for $\|\mathbf{b}_1\|$.

In any case, the BKZ algorithm gives a better basis for enumeration than LLL, however, it does take a lot more computation and for $\beta \geq 30$ it is not sure to terminate in polynomial time.

8 LWE in Rings

As we have seen already, lattices are an attractive tool for cryptography. So far in this thesis we have tried to solve the LWE problem using algorithms with Gaussian elimination or q-ary lattices without any assumed structure. However, the most efficient lattice-based cryptosystems are related to ideal lattices, which correspond to ideals in certain families of rings (e.g. $\mathbb{Z}[X]/(X^{2^k}+1)$). In this chapter, we will look at how we can use the algebraic structure of a principal ideal lattice to solve a common problem in Ring-LWE.

Several recent cryptosystems rely on principal ideals that have "relatively short" generators that act as secret keys. Then the problem, put simply, is; given the ring R and a principal ideal $(g) = gR$, can we find the secret g ([Ber14])? Or, in more formal terms: Given some $\mathbb{Z}$-basis of an ideal that is guaranteed to have a "short" generator g, find a sufficiently short generator. An attack consist of two main parts:

1. Given a $\mathbb{Z}$-basis, find some arbitrary generator g' for the ideal generated by the short generator g; g' does not need to be short.
2. From g' find a sufficiently short generator g.

The first part is known as the *Principal Ideal Problem* (PIP) and will not be discussed in this project, [Cra+16] mentions several efficient algorithms to this purpose. Instead, we will focus on the second part of the attack. This is called the *Short Generator of a Principal Ideal Problem*, or SG-PIP for short. The SG-PIP might not look like the LWE problem we are familiar with, but it will become apparent, as we start constructing our lattice, that it is in fact such a system we are solving.

In this last chapter, a lot more algebra is required than in the previous chapters. This is in order to both define the lattice and understand why the algorithm works. Because of this though, this chapter will have somewhat of a different structure compared to the preceding ones. We will not

spend time on run-time or computations, but focus solely on understanding the lattice and why the algorithm works. This chapter is based on and follows closely [Cra+16].

We begin, in Section 8.1, by presenting the round-off algorithm. We have mentioned this earlier as a potential algorithm for solving a CVP, and it is the algorithm we will base our main result on. In Section 8.2 we will introduce the group notation used in this chapter. Then, in Sections 8.3 and 8.4, we go through some material on circulant matrices and characters. Understanding these concepts lays the foundation for proving that our lattice basis has the necessary properties for decoding. Following this, in Section 8.5, we will use a primitive n^{th} root of unity ζ to the define the cyclotomic number field $\mathbb{Q}(\zeta)$ and its ring of integers $R = \mathbb{Z}[\zeta]$. For this ring we define our full rank lattice; the *log-unit-lattice*. Our main results, however, will focus on a sublattice of the log-unit-lattice, using the log embedding on the cyclotomic units. Having introduced our sublattice and its basis, we devote all of Section 8.6 to proving an upper bound on the dual basis vectors of the sublattice, which is crucial for constructing and proving the main algorithm in Section 8.7. In the last section, Section 8.8, we prove that the algorithm works well for a continuous Gaussian distribution.

8.1 The Round-Off Algorithm

The round-off algorithm is a standard approach to solving BDD (CVP with an assumption that the error is bounded).

Theorem 8.1 ([Cra+16, Claim in Section 2.1])**.** *Let $\Lambda \subseteq \mathbb{R}^m$ be a lattice with basis $\boldsymbol{B} = (\boldsymbol{b}_1, ..., \boldsymbol{b}_n)$ where $n \leq m$. Let $\boldsymbol{t} = \boldsymbol{x} + \boldsymbol{\nu} \in \mathbb{R}^m$ for some $\boldsymbol{x} \in \Lambda$ and $\boldsymbol{\nu} \in \mathbb{R}^m$. If $\langle \boldsymbol{b}_j^\vee, \boldsymbol{\nu} \rangle \in [-\frac{1}{2}, \frac{1}{2})$ for all $j = 1, ..., n$. Then, on input $\boldsymbol{t}$ and basis $\boldsymbol{B}$, the round-off algorithm outputs $\boldsymbol{x}$.*

Recall from Section 5.1 that $\mathbf{B}^\vee$ denotes the dual of a lattice basis $\mathbf{B}$.

Proof. $\boldsymbol{x} \in \Lambda$ implies there exists $\boldsymbol{z} \in \mathbb{Z}^n$ such that $\boldsymbol{x} = \mathbf{B}\boldsymbol{z}$. Then

$$\begin{aligned}(\mathbf{B}^\vee)^T\boldsymbol{t} &= (\mathbf{B}^\vee)^T\boldsymbol{x} + (\mathbf{B}^\vee)^T\boldsymbol{\nu} \\ &= \boldsymbol{z} + (\mathbf{B}^\vee)^T\boldsymbol{\nu}.\end{aligned}$$

Then $\left\lfloor (\mathbf{B}^\vee)^T\boldsymbol{t} \right\rceil = \left\lfloor \boldsymbol{z} + (\mathbf{B}^\vee)^T\boldsymbol{\nu} \right\rceil = \boldsymbol{z}$ because $\boldsymbol{z} \in \mathbb{Z}^m$ and because $\langle \mathbf{b}_j^\vee, \boldsymbol{\nu} \rangle \in [-\frac{1}{2}, \frac{1}{2})$ for all j (so it will not affect the round-off algorithm). So $\mathbf{B}\left\lfloor (\mathbf{B}^\vee)^T\boldsymbol{t} \right\rceil = \mathbf{B}\boldsymbol{z} = \boldsymbol{x}$ as desired.

□

8.2 Group Notation

Following Cramer et al. [Cra+16], we will use group notation for indexing. That is, instead of $i = 1, ..., n$, we have $i \in G$ for some group G. That means, if we refer to $i \in G$, i is the group element. However, we will at times also refer to $i_\alpha \in G$ for $\alpha = 1, ..., |G|$, which refers to the α^{th} group element in G. We break with the convention only when deemed necessary to make proofs as clear as possible and we will endeavour to make it clear when we do so. Note however, that the order of the elements is not important and the enumeration is arbitrary.

In this project, G will always be a finite abelian group of integers, and the relevant groups for our results will be $\mathbb{Z}_n^*$ and $\mathbb{Z}_n^*/\{\pm 1\}$, where * denotes the unit group.

8.3 G-Circulant Matrices

For this section, we assume $G = (G, *)$ is a finite abelian group. We start of by defining a G-circulant matrix and looking at a few examples, before associating it to characters and character groups in Section 8.4.

Definition 8.2 (Circulant Matrix)**.** For a vector $\boldsymbol{a} = (a_i)_{i \in G}$ indexed by G, the *G-circulant matrix* associated with $\boldsymbol{a}$ is the G-by-G matrix whose $(i, j)^{\text{th}}$ entry is $a_{i*j^{-1}}$ (where $i, j \in G$).

In order to get a better understanding of the group notation and G-circulant matrices, we will look at a few examples.

Example 8.3. Let $G = (\mathbb{Z}_m, +)$. Then $\boldsymbol{a} = (a_0, a_1, ..., a_{m-1})$ and j^{-1} is $-j \pmod m$. The $(i,j)^{\text{th}}$ entry is therefore $a_{i*j^{-1}} = a_{i-j \pmod n}$. The G-circulant matrix associated with $\boldsymbol{a}$ is:

$$\begin{pmatrix} a_0 & a_{m-1} & \dots & a_2 & a_1 \\ a_1 & a_0 & \dots & a_3 & a_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m-1} & a_{m-2} & \dots & a_1 & a_0 \end{pmatrix}.$$

This first example is perhaps the definition of circulant matrices most of us are familiar with. However, when dealing with unit groups, which we will be later on, the indexing will not necessarily be as intuitive, as we will see in the following example.

Example 8.4. Let $G = (\mathbb{Z}_5^*, \cdot)$. Then $\boldsymbol{a} = (a_1, a_2, a_3, a_4)$ and $1^{-1} = 1,\ 2^{-1} = 3,\ 3^{-1} = 2,\ 4^{-1} = 4$. Then the G-circulant matrix of $\boldsymbol{a}$ is

$$\begin{pmatrix} a_1 & a_3 & a_2 & a_4 \\ a_2 & a_1 & a_4 & a_3 \\ a_3 & a_4 & a_1 & a_2 \\ a_4 & a_2 & a_3 & a_1 \end{pmatrix}.$$

For example, the $(2,3)$ entry of the matrix is $a_{2.3^{-1}} = a_{2.2} = a_4$.

Example 8.5. Because 5 is prime, there are no gaps in the indexing. If n is not a prime number, say $G = \mathbb{Z}_9^*$, then $\boldsymbol{a} = (a_1, a_2, a_4, a_5, a_7, a_8)$ and the G-circulant matrix associated with $\boldsymbol{a}$ is

$$\begin{pmatrix} a_1 & a_5 & a_7 & a_2 & a_4 & a_8 \\ a_2 & a_1 & a_5 & a_4 & a_8 & a_7 \\ a_4 & a_2 & a_1 & a_8 & a_7 & a_5 \\ a_5 & a_7 & a_8 & a_1 & a_2 & a_4 \\ a_7 & a_8 & a_4 & a_5 & a_1 & a_2 \\ a_8 & a_4 & a_2 & a_4 & a_5 & a_1 \end{pmatrix}.$$

Notice how, when using group notation, we do not make any rows or columns zero when an index is not in G. For example, the element in the 3^{rd} row and 4^{th} column is not 0 because $3 \notin G$. Instead, it will be the 3^{rd} element in G times the inverse of the 4^{th} element in G, i.e. $4{\cdot}5^{-1} = 4{\cdot}2 = 8$.

8.4 Characters

We will now go through some basic properties of characters and character groups. The proofs will be given briefly or not at all. The reader is referred to Chapter 6 in [Apo76] for details.

Definition 8.6 (Character)**.** A character κ is a group morphism $\kappa : G \mapsto \{u \in \mathbb{C} \ : \ |u| = 1\}$. i.e. $\kappa(ij) = \kappa(i)\kappa(j)\ \forall i, j \in G$.

Definition 8.7 (Character group)**.** The *character group* $(\hat{G}, \cdot)$ is the set of characters of G, with the group operation being the usual multiplication of functions. i.e. $(\kappa \cdot \eta)(i) = \kappa(i) \cdot \eta(i)$ for $i \in G$ and $\kappa, \eta \in \hat{G}$.

The *principal character* of a character group is the character $\kappa \in \hat{G}$ such that $\kappa(i) = 1$ for all $i \in G$. We will denote this to be the first element in $\hat{G}$. That means $\hat{G}\backslash\{1\}$ is the group of characters except the principal character.

Now note that $|\hat{G}| = |G| = m$ ([Apo76, Theorem 6.8]). Also note that for all $\kappa \in \hat{G}$, $\kappa(1_G) = 1$ ([Apo76, Theorem 6.7]), hence $\overline{\kappa(i)} = \kappa(i)^{-1} = \kappa(i^{-1})$.

Now identify the character $\kappa \in \hat{G}$ with the vector $\boldsymbol{\kappa} = (\kappa(i))_{i \in G} \in \mathbb{C}^m$. From the fact that $\langle \boldsymbol{\kappa}, \boldsymbol{\kappa} \rangle = \sum_{i \in G} \kappa(i) \cdot \overline{\kappa(i)} = \sum_{i \in G} 1 = |G|$, it is clear that all characters have equal Euclidean norm; $\|\boldsymbol{\kappa}\| = \sqrt{|G|}$. In fact, distinct characters $\kappa, \eta \in \hat{G}$ are orthogonal:

$$\langle \boldsymbol{\kappa}, \boldsymbol{\eta} \rangle = \sum_{i \in G} \kappa(i) \cdot \overline{\eta(i)} = \sum_{i \in G} (\kappa\eta^{-1})(i) = 0. \tag{8.1}$$

This follows from [Apo76, Theorem 6.10], where, for all $\kappa \in \hat{G}\backslash\{1\}$, $\sum_{i \in G} \kappa(i) = 0$. Combining this with a similar result: $\sum_{\kappa \in \hat{G}} \kappa(i) = 0$

for all $i \in G\backslash\{1\}$ ([Apo76, Theorem 6.13]) implies that the the complex G-by-$\hat{G}$ matrix $\mathbf{P}_G = |G|^{-\frac{1}{2}}(\kappa(i))_{i\in G,\kappa\in\hat{G}}$ is unitary. Let $\kappa_\alpha(i_\beta)$ denote the α^{th} character in $\hat{G}$ on the β^{th} group element in G. Then for $|G| = m$, $\mathbf{P}_G$ looks like:

$$\mathbf{P}_G = |G|^{-\frac{1}{2}} \begin{pmatrix} \kappa_1(i_1) & \kappa_2(i_1) & \dots & \kappa_m(i_1) \\ \kappa_1(i_2) & \kappa_2(i_2) & \dots & \kappa_m(i_2) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_1(i_m) & \kappa_2(i_m) & \dots & \kappa_m(i_m) \end{pmatrix}. \tag{8.2}$$

Next, we prove that for a finite abelian group G, the concepts of a G-circulant matrix $\mathbf{A}$ and the character matrix $\mathbf{P}_G$ are very much connected, and the eigenvalues of $\mathbf{A}$ are defined by the characters of G.

Lemma 8.8 ([Cra+16, Lemma 1])**.** *A complex matrix $\boldsymbol{A}$ is G-circulant if and only if the $\hat{G} \times \hat{G}$-matrix $\boldsymbol{P}_G^{-1}\boldsymbol{A}\boldsymbol{P}_G$ is diagonal; equivalently, the columns of $\boldsymbol{P}_G$ are the eigenvectors of $\boldsymbol{A}$. If $\boldsymbol{A}$ is the G-circulant matrix associated with $\boldsymbol{a} = (a_i)_{i\in G}$, its eigenvalue corresponding to $\kappa \in \hat{G}$ is $\lambda_\kappa = \langle \boldsymbol{a}, \boldsymbol{\kappa}\rangle = \sum_{i\in G} a_i \cdot \overline{\kappa(i)}$.*

Proof. ($\Leftarrow$) Assume $\mathbf{A}$ is G-circulant associated with the vector $\boldsymbol{a}$.

It is a well known result in linear algebra that a matrix is diagonalisable if there exists a non-singular square matrix $\boldsymbol{P}$ such that $\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{D}$, where $\mathbf{D}$ is a diagonal matrix consisting of the eigenvalues of $\mathbf{A}$ in its diagonal and the corresponding eigenvectors are the columns of $\mathbf{P}$. Hence it suffices to prove that $\boldsymbol{\kappa}$, for $\kappa \in \hat{G}$, are the eigenvectors of $\mathbf{A}$. For $i \in G$ the i^{th} element in $\mathbf{A}\boldsymbol{\kappa}$ is

$$(\mathbf{A}\boldsymbol{\kappa})_i = \sum_{j\in G} a_{ij^{-1}}\kappa(j) = \left(\sum_{k\in G} a_k\overline{\kappa(k)}\right)\kappa(i) = \langle \boldsymbol{a}, \boldsymbol{\kappa}\rangle\kappa(i) = \lambda_\kappa\kappa(i),$$

where $k = ij^{-1}$. Hence, $\mathbf{A}\boldsymbol{\kappa} = \lambda_\kappa\boldsymbol{\kappa}$. As there are there are $|G|$ distinct $\boldsymbol{\kappa}$'s, these are all the eigenvectors. That means that $\mathbf{A}$ is diagonalisable with eigenvectors $\boldsymbol{\kappa}$ as the columns in $\mathbf{P}$, which is equal to $\mathbf{P}_G$, and hence we have our result.

($\Rightarrow$) Now we assume $\mathbf{P}_G^{-1}\mathbf{A}\mathbf{P}_G$ is diagonal and we want to show $\mathbf{A}$ is G-circulant. That is, $\mathbf{P}_G^{-1}\mathbf{A}\mathbf{P}_G = \mathbf{D}$. We can write $\mathbf{D} = \sum_{\kappa\in\hat{G}} \lambda_\kappa \mathbf{D}_\kappa$ where $\mathbf{D}_\kappa$ is the zero matrix except for 1 in the diagonal entry in the κ^{th} row. The sum of circulant matrices is circulant. It is therefore enough to prove that $\mathbf{P}_G\mathbf{D}_\kappa\mathbf{P}_G^{-1}$ is G-circulant.

$\mathbf{D}_\kappa\mathbf{P}_G^{-1}$ leaves the κ^{th} row of $\mathbf{P}_G^{-1}$ unchanged, and all other entries are zero. Hence the κ^{th} row of $\mathbf{D}_\kappa\mathbf{P}_G^{-1}$ consists of elements of the form $\overline{\kappa(i)}$ for $i \in G$. To show this explicitly, let $\kappa_{\alpha\beta} = \kappa_\alpha(i_\beta)$, where κ_α is the α^{th} element in $\hat{G}$ and i_β the β^{th} element in G. Then

$$\begin{aligned}
\mathbf{P}_G\mathbf{D}_{\kappa_\alpha}\mathbf{P}_G^{-1} &= |G|^{-1}\begin{pmatrix} \kappa_{11} & \kappa_{21} & \dots & \kappa_{m1} \\ \kappa_{12} & \kappa_{22} & \dots & \kappa_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_{1m} & \kappa_{2m} & \dots & \kappa_{mm} \end{pmatrix}\begin{pmatrix} \ddots & & 0 \\ & 1_{\alpha\alpha} & \\ 0 & & \ddots \end{pmatrix}\begin{pmatrix} \overline{\kappa_{11}} & \overline{\kappa_{12}} & \dots & \overline{\kappa_{1m}} \\ \overline{\kappa_{21}} & \overline{\kappa_{22}} & \dots & \overline{\kappa_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\kappa_{m1}} & \overline{\kappa_{m2}} & \dots & \overline{\kappa_{mm}} \end{pmatrix}, \\
&= |G|^{-1}\begin{pmatrix} \kappa_{11} & \kappa_{21} & \dots & \kappa_{m1} \\ \kappa_{12} & \kappa_{22} & \dots & \kappa_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_{1m} & \kappa_{2m} & \dots & \kappa_{mm} \end{pmatrix}\begin{pmatrix} 0 & 0 & \dots & 0 \\ \overline{\kappa_{\alpha1}} & \overline{\kappa_{\alpha2}} & \dots & \overline{\kappa_{\alpha m}} \\ & & & \\ 0 & 0 & \dots & 0 \end{pmatrix}, \\
&= |G|^{-1}\begin{pmatrix} \kappa_{\alpha1}\overline{\kappa_{\alpha1}} & \kappa_{\alpha1}\overline{\kappa_{\alpha2}} & \dots & \kappa_{\alpha1}\overline{\kappa_{\alpha m}} \\ \kappa_{\alpha2}\overline{\kappa_{\alpha1}} & \kappa_{\alpha2}\overline{\kappa_{\alpha2}} & \dots & \kappa_{\alpha2}\overline{\kappa_{\alpha m}} \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_{\alpha m}\overline{\kappa_{\alpha1}} & \kappa_{\alpha m}\overline{\kappa_{\alpha2}} & \dots & \kappa_{\alpha m}\overline{\kappa_{\alpha m}} \end{pmatrix}, \\
&= |G|^{-1}\begin{pmatrix} \kappa_\alpha(i_1 i_1^{-1}) & \kappa_\alpha(i_1 i_2^{-1}) & \dots & \kappa_\alpha(i_1 i_m^{-1}) \\ \kappa_\alpha(i_2 i_1^{-1}) & \kappa_\alpha(i_2 i_2^{-1}) & \dots & \kappa_\alpha(i_2 i_m^{-1}) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa_\alpha(i_m i_1^{-1}) & \kappa_\alpha(i_m i_2^{-1}) & \dots & \kappa_\alpha(i_m i_m^{-1}) \end{pmatrix}.
\end{aligned}$$

So the $(\alpha,\beta)^{\text{th}}$ entry depends only on $i_\alpha i_\beta^{-1}$ as required. Hence $\mathbf{P}_G\mathbf{D}_{\kappa_\alpha}\mathbf{P}_G^{-1}$ is a G-circulant matrix, and therefore, by linearity, so is $\mathbf{P}_G\mathbf{D}\mathbf{P}_G^{-1}$. $\square$

It follows that every row and column of $\mathbf{A}$ has squared Euclidean norm:

$$\|\boldsymbol{a}\|^2 = \|\mathbf{P}_G^* \boldsymbol{a}\|^2 = |G|^{-1} \sum_{\kappa \in \hat{G}} |\lambda_\kappa|^2. \tag{8.3}$$

Where the first equality follows from the fact that $\mathbf{P}_G$ is unitary and therefore does not change the Euclidean norm of a vector when multiplied with it. The second equality follows by simple calculation:

$$\mathbf{P}_G^* \boldsymbol{a} = |G|^{-\frac{1}{2}} \begin{pmatrix} \sum_j a_j \overline{\kappa_{1j}} \\ \sum_j a_j \overline{\kappa_{2j}} \\ \vdots \\ \sum_j a_j \overline{\kappa_{mj}} \end{pmatrix} = |G|^{-\frac{1}{2}} \begin{pmatrix} \langle \boldsymbol{a}, \boldsymbol{\kappa}_1 \rangle \\ \langle \boldsymbol{a}, \boldsymbol{\kappa}_2 \rangle \\ \vdots \\ \langle \boldsymbol{a}, \boldsymbol{\kappa}_m \rangle \end{pmatrix} = |G|^{-\frac{1}{2}} \begin{pmatrix} \lambda_{\kappa_1} \\ \lambda_{\kappa_2} \\ \vdots \\ \lambda_{\kappa_m} \end{pmatrix}.$$

8.4.1 Dirichlet Characters and L-series

Now we will look at a special type of characters, called *Dirichlet characters*. Again, the properties and statements will be given in brief. The reader is encouraged to look at Chapter 3 in [Was97] for further details. Instead, we will look at some helpful examples to aid our understanding of characters further on.

Definition 8.9 (Dirichlet character)**.** A Dirichlet character κ is a character of the unit group $\mathbb{Z}_k^*$ for some positive integer k. That is, a multiplicative homomorphism $\kappa : \mathbb{Z}_k^* \to \{u \in \mathbb{C} : |u| = 1\}$.

We say a character κ of the group $\mathbb{Z}_l^*$ is *defined* modulo l. If $l|k$, κ induces a character $\mathbb{Z}_k^* \to \{u \in \mathbb{C} : |u| = 1\}$ by composition of the natural homomorphism $\mathbb{Z}_k^* \to \mathbb{Z}_l^*$, $z \mapsto z \pmod k$ and κ. In which case, κ is defined both modulo l and modulo k, since they are essentially the same map. See Example 8.10 below to get a better understanding of what we mean by this.

If κ is defined modulo k, and there does not exist any other positive integer $l|k$ such that κ is also defined modulo l, then k is minimal and is called the *conductor* of κ. We denote the conductor of κ as f_κ, or just f. When a character is defined modulo its conductor, it is called *primitive*.

We can extend κ to a map $\kappa : \mathbb{Z} \to \mathbb{C}$ where $\kappa(a) = 0$ if $\gcd(a, f_\kappa) \neq 1$. We shall only consider primitive characters as this makes $\kappa(a) = 0$ happen as little as possible.

To get a better understanding of Dirichlet characters, and characters in general, we will now look at some basic examples.

Example 8.10 ([Was97, Chapter 3, Example 1])**.** Define the Dirichlet character $\kappa : \mathbb{Z}_8^* \to \{u \in \mathbb{C} : |u| = 1\}$ by

$$\kappa(1) = 1, \qquad \kappa(5) = 1,$$
$$\kappa(3) = -1, \qquad \kappa(7) = -1.$$

Note that $4|8$, that $5 \equiv 1 \pmod 4$ and $7 \equiv 3 \pmod 4$, and that

$$\kappa(5) = 1 = \kappa(1),$$
$$\kappa(7) = -1 = \kappa(3).$$

Therefore, κ is also defined modulo 4, but is 4 minimal? The only positive integer less than 4 that divides 8 is 2 (and 1, but 1 obviously does not define κ). However, since $3 \equiv 1 \pmod 2$ but $\kappa(1) \neq \kappa(3)$, 2 does not define κ. Therefore 4 is minimal and the conductor of κ is $f_\kappa = 4$.

A character is called *even* if $\kappa(-1) = 1$ and *odd* if $\kappa(-1) = -1$. The example above is an *odd* character. Note that the even characters of $\mathbb{Z}_n^*$ correspond to the characters of $\mathbb{Z}_n^*/\{\pm 1\}$. To understand this, we will take a closer at $\mathbb{Z}_n^*/\{\pm 1\}$ and the characters of $\mathbb{Z}_n^*$:

Example 8.11. Some examples of $\mathbb{Z}_n^*/\{\pm 1\}$:

n	$\varphi(n)$	$\mathbb{Z}_n^*$	$\mathbb{Z}_n^*/\{\pm 1\}$
3	2	$\{1, 2\}$	$\{1\}$
4	2	$\{1, 3\}$	$\{1\}$
5	4	$\{1, 2, 3, 4\}$	$\{1, 2\}$
9	6	$\{1, 2, 4, 5, 7, 8\}$	$\{1, 2, 4\}$

For example, for $n = 5$, $3 \in \mathbb{Z}_5^*$ is equal to $3 = -2 = 2$ in $\mathbb{Z}_5^*/\{\pm 1\}$ as $\mathbb{Z}_n^*/\{\pm 1\}$ is essentially the group where positive and negative values are equal. We also see that the order of $\mathbb{Z}_n^*/\{\pm 1\}$ is equal to $|\mathbb{Z}_n^*|/2 = \frac{\varphi(n)}{2}$.

Now, we look at another example to get a better idea of what we mean when we say that the even characters of $\mathbb{Z}_n^*$ correspond with the characters of $\mathbb{Z}_n^*/\{\pm 1\}$?.

Example 8.12. The Dirichlet characters of $\mathbb{Z}_5^*$ is:

$\kappa \backslash a$	1	2	3	4
$\kappa_1(a)$	1	1	1	1
$\kappa_2(a)$	1	i	$-i$	-1
$\kappa_3(a)$	1	-1	-1	1
$\kappa_4(a)$	1	$-i$	i	-1

(Table from [Apo76, p.139].) We see that κ_1 and κ_3 are even characters. In $\mathbb{Z}_n^*/\{\pm 1\}$, $3 = 2$ and $4 = 1$, and note that $\kappa_{1/3}(3) = \kappa_{1/3}(2)$ and $\kappa_{1/3}(4) = \kappa_{1/3}(1)$. So the even characters could be defined as

$\kappa \backslash a$	1	2
$\kappa_1(a)$	1	1
$\kappa_3(a)$	1	-1

These are all the characters of $\mathbb{Z}_5^* \backslash \{\pm 1\}$. And from this table we can gather that $\kappa_3(3) = \kappa_3(2) = -1$, which corresponds with the first table.

So any character of $\mathbb{Z}_n^*/\{\pm 1\}$ will be even when mapped from $\mathbb{Z}_n^*$, and

since there are at most $|\mathbb{Z}_n^*|/2$ even characters in $\mathbb{Z}_n^*$ and $|\mathbb{Z}_n^*|/2$ characters in $\mathbb{Z}_n^*/\{\pm 1\}$, these will have a $1-1$ correspondence.

We say that a Dirichlet character is *quadratic* if all its values are real and it is not the principal character. For example, κ_3, in Example 8.11, is quadratic, while κ_2 and κ_4 are non-quadratic.

Definition 8.13 (Dirichlet L-series)**.** For a Dirichlet character κ, the *Dirichlet L-function* is defined as the formal series

$$L(s,\kappa) = \sum_{n=1}^{\infty} \frac{\kappa(n)}{n^s}$$

For any Dirichlet character κ, $L(s,\kappa)$ is absolutely convergent for all $s \in \mathbb{C}$ with $\mathrm{Re}(s) > 1$. It is also known that $L(s,\kappa)$ converges and is non-zero for any non-principal Dirichlet character ($\kappa \neq 1$ for all $a \in \mathbb{Z}_n$). For our purposes, we will only need $L(1,\kappa)$, and for that we have the following bound, which we will not prove in this thesis:

Theorem 8.14 ([Cra+16, Theorem 1])**.** *There exist a constant $c > 0$ such that, for any non-quadratic character κ of conductor $f_\kappa > 1$,*

$$\frac{1}{\ell(f_\kappa)} \leq L(1,\kappa) \leq \ell(f_\kappa), \tag{8.4}$$

where $\ell(f_\kappa) = c\ln(f_\kappa)$. Moreover, for any quadratic character κ,

$$|L(1,\kappa)| \geq \frac{1}{c\sqrt{f_\kappa}}. \tag{8.5}$$

8.5 Primitive Roots and Cyclotomic Number Fields

In this section, we begin by building up the basic theory and define the field and ring that we will use in the remainder of this chapter. Then, in Section 8.5.1, we define the logarithmic embedding that, when working on our ring, will define a full rank lattice; the *log-unit-lattice*. In Section 8.5.2 we introduce the cyclotomic units that define a sublattice of the log-unit lattice, this sublattice is what we will use in our main algorithm. At

the end of this section, we will have built up enough theory to relate the SG-PIP to a CVP and we do this in Section 8.5.3.

Definition 8.15. An *algebraic number field* $\mathbb{F}$ is an extension field of $\mathbb{Q}$ such that its dimension $[\mathbb{F} : \mathbb{Q}]$ is finite.

An algebraic number field is *Galois* if the order of its automorphism group equal its dimension.

Definition 8.16. Let $\mathbb{F}$ be a field. $\zeta \in \mathbb{F}$ is a n^{th} *root of unity* if $\zeta^n = 1$. If, in addition, $\zeta^k \neq 1$ for any integer $k < n$, then n is the order of ζ and ζ is called a *primitive* n^{th} *root of unity.*

If $\zeta \in \mathbb{F}$ is a n^{th} a primitive root of unity, then the complete set of primitive n^{th} roots of unity in $\mathbb{F}$ consists of powers ζ^j for $j \in \mathbb{Z}_n^*$.

Definition 8.17. A number field $\mathbb{F}$ is *cyclotomic* if $\mathbb{F} = \mathbb{Q}(\zeta)$ for some n^{th} root of unity $\zeta \in \mathbb{F}$. Its degree is $\varphi(n)$ where n is the order of ζ and φ is the Euler totient function.

For this project we define our field to be the cyclotomic number field $\mathbb{Q}(\zeta)$ for some primitive n^{th} root of unity $\zeta \in \mathbb{Q}(\zeta)$. Then $[\mathbb{Q}(\zeta) : \mathbb{Q}] = \varphi(n)$ where ζ's minimal polynomial is $\prod_{j \in \mathbb{Z}_n^*}(x - \zeta^j) \in \mathbb{Z}[x]$. That means a typical element $x \in \mathbb{Q}(\zeta)$ looks like

$$x = \sum_{i=0}^{\varphi(n)-1} a_i \zeta^j,$$

where $a_i \in \mathbb{Q}$. In this project we will work with the ring $R = \mathbb{Z}[\zeta]$[¶]. Let U denote the cyclic group of n^{th} roots of unity with multiplication as its group operation, that is $U = \langle\zeta\rangle$. Define the automorphism $\sigma_j : \mathbb{Q}(\zeta) \to$

[¶]R is the ring of integers of $\mathbb{Q}(\zeta)$, that means the ring of roots of monic, irreducible polynomials with integer coefficients $\mathbb{Q}(\zeta)[x]$.

$\mathbb{Q}(\zeta)$ as the identity on $\mathbb{Q}$ and $\sigma_j(\zeta) = \zeta^j$ where $j \in \mathbb{Z}_n^*$[‖]. That is,

$$\sigma_j(x) = \sigma_j\left(\sum_{i=0}^{\varphi(n)-1} a_i\zeta^i\right) = \sum_{i=0}^{\varphi(n)-1} a_i\sigma_j(\zeta^i) \sum_{i=0}^{\varphi(n)-1} a_i\zeta^{ij}.$$

The concatenation of these embeddings is known as the *canonical embeddings* and is defined by $\boldsymbol{\sigma}(x) = (\sigma_j(x))_{j\in\mathbb{Z}_n^*}$. This gives the norm $\|x\| = \|\boldsymbol{\sigma}(x)\|$ for all $x \in \mathbb{Q}(\zeta)$.

We will also define the algebraic norm $N : R \to \mathbb{Z}$ of R by

$$N(a) = \prod_{j\in Z_n^*} \sigma_j(a).$$

It is easy to verify that N satisfies the definition of a norm. It follows directly that

$$N(u) = \pm 1. \tag{8.6}$$

for any unit $u \in R$. This is because if u is a unit in R, then there exist a $v \in R$ such that $uv = 1$, which implies that $1 = N(1) = N(uv) = N(u)N(v)$.

A concrete example of such a cyclotomic field is $\mathbb{Q}(\omega)$ where ω is the n^{th} primitive root of unity in the complex numbers $e^{\frac{2\pi}{n}i}$.

8.5.1 The Logarithmic Embedding

The mapping $\sigma_j(x)$ comes in conjugate pairs, that is, $\sigma_j(x) = \overline{\sigma_{-j}(x)}$ where $|\sigma_j(x)| = |\sigma_{-j}(x)|$. Since we, for the most part, will be concerned with the magnitudes, we look at the multiplicative quotient group $G = \mathbb{Z}_n^*/\{\pm 1\}$. Note that $|\mathbb{Z}_n^*| = \varphi(n)$ and $|\mathbb{Z}_n^*/\{\pm 1\}| = \frac{\varphi(n)}{2}$.

Next, for $G = \mathbb{Z}_n^*/\{\pm 1\}$, define the logarithmic embedding:

$$\mathbf{Log} : \mathbb{Q}(\zeta) \to \mathbb{R}^{\frac{\varphi(n)}{2}}$$

$$\mathbf{Log}(a) = (\log|\sigma_j(a)|)_{j\in G}.$$

[‖]In fact $G(\mathbb{Q}(\zeta)|\mathbb{Q}) = \{\sigma_j\}_{j\in\mathbb{Z}_n^*}$.

The **Log** embedding defines a group morphism, mapping the multiplicative group $\mathbb{Q}(\zeta)^*$ to an additive subgroup of $\mathbb{R}^{\varphi(n)/2}$. That is $\mathbf{Log}(a \cdot b) = \mathbf{Log}(a) + \mathbf{Log}(b)$.

When restricting **Log** to R^*, the Dirichlet Unit Theorem ([See e.g. Sam72, Chapter 4.4, Theorem 1]) implies that $\pm U = \pm\langle\zeta\rangle$ defines the kernel of **Log** and that $\mathbf{Log}\, R^*$ is a lattice of rank $\frac{\varphi(n)}{2} - 1$. Ideally we want our lattice to be full rank, so next we show that $\mathbf{Log}\, R^*$ is orthogonal to the all-one vector $\mathbf{1}$:

$$\langle \mathbf{Log}(u), \mathbf{1} \rangle = \sum_{j \in G} \log |\sigma_j(u)| = \log \left(|N(u)| \right) = 0.$$

This follows from the fact that $N(u) = \pm 1$ for all $u \in R^*$ as shown in Equation (8.6). That means $\mathbf{Log}\, R^*$ is a full rank lattice in the linear subspace of $\mathbb{R}^{\frac{\varphi(n)}{2}}$ orthogonal to $\mathbf{1}$. We refer to it as the *log-unit lattice*.

8.5.2 Cyclotomic Units

Next, we will define a subgroup of R^* called the cyclotomic units. This will define a sublattice to $\mathbf{Log}\, R^*$ with some nice basis properties, as we will see.

Define

$$z_j := \zeta^j - 1 \tag{8.7}$$

for $j \in \mathbb{Z}_n \backslash \{0\}$. Note that

$$\begin{aligned} z_{-j} &= \zeta^{-j} - 1, \\ \zeta^j z_{-j} &= 1 - \zeta^j, \\ z_j &= -\zeta^j z_{-j}, \end{aligned}$$

which means $\mathbf{Log}(z_j) = \mathbf{Log}(z_{-j})$. Define A to be a multiplicative subgroup of $\mathbb{Q}(\zeta)^*$ generated by ζ and z_j for $j = 1, ..., n-1$. Then the group of *cyclotomic units*, C, is defined by

$$C = A \cap R^*.$$

We want to find a generating set for C, which is given by a Lemma in [Was97] and which we will only state here:

Lemma 8.18 ([Was97, Lemma 8.1])**.** *Let n be a prime power, and define*

$$b_j := \frac{z_j}{z_1} = \frac{\zeta^j - 1}{\zeta - 1}. \tag{8.8}$$

The group C of cyclotomic units is generated by $\pm\zeta$ and b_j for $j = G\backslash\{1\}$ ($G = \mathbb{Z}_n^*/\{\pm 1\}$)

Note that $\mathbf{Log}\, C$ is a sublattice of $\mathbf{Log}\, R^*$. Discussion following Theorem 2 in [Cra+16] shows that it is reasonable to assume $[\mathbf{Log}\, R^* : \mathbf{Log}\, C]$ is quite small. Here, $[\mathbf{Log}\, C : \mathbf{Log}\, R^*]$ is the index of the subgroup $\mathbf{Log}\, C$ over $\mathbf{Log}\, R^*$. Recall that this is equal to the number of left (or right) cosets of $\mathbf{Log}\, C$ in $\mathbf{Log}\, R^*$. So when $[\mathbf{Log}\, R^* : \mathbf{Log}\, C]$ is small, there are very few cosets of $\mathbf{Log}\, C$ in $\mathbf{Log}\, R^*$, and the two groups are not too different.

8.5.3 The Lattice Problem

We have now established enough algebraic theory to relate the SG-PIP to a CVP in a lattice. Recall that the SG-PIP states: If we know that the principal ideal (g) in a ring is generated by a short generator g, and we are given a long generator g' for the same ideal, can we find a sufficiently short generator?

We will assume the long generator is of the form $g' = ug$ where g is the short lattice and u is a cyclotomic unit. Then we can apply the logarithmic embedding to g' and get:

$$\mathbf{Log}(g') = \mathbf{Log}(ug) = \mathbf{Log}(u) + \mathbf{Log}(g).$$

Since $\mathbf{Log}(u)$ is in the lattice $\mathbf{Log}\, C$, we see how this translates to a CVP of the form $\boldsymbol{t} = \boldsymbol{x} + \boldsymbol{\nu}$ where $\boldsymbol{x} = \mathbf{Log}(u)$ is the lattice point closest to the target $\boldsymbol{t} = \mathbf{Log}(g')$ and $\boldsymbol{\nu} = \mathbf{Log}(g)$ is the noise term. Since we also assume g is short, this is, in fact, a BDD problem.

As stated earlier, we wish to apply the round-off algorithm on this lattice (to find $\mathbf{Log}(u)$), but in order to do this, we need to show that the basis of $\mathbf{Log}\, C$ is well suited for the round-off algorithm. Hence, we will use the entire next section to study the basis of $\mathbf{Log}\, C$ and find an upper bound on the dual basis. After that, in Section 8.7, we will finally be ready to show how we can solve the SG-PIP using the log-unit lattice and the round-off algorithm.

8.6 Bounds On The Dual Basis

From now on we will define n, the cyclotomic index, to be a prime power and also $G = \mathbb{Z}_n^*/\{\pm 1\}$. For concreteness, we will situate the cyclotomic number field in the complex numbers and define the primitive n^{th} root of unity to be $\omega = \omega_n = e^{\frac{2\pi i}{n}}$. That means $\mathbb{Q}(\omega)$ is our cyclotomic number field and $\sigma_j(\omega) = \omega^j$. This follows the convention of [Cra+16]. Although they tend to switch between ω and ζ, to keep things as general as possible, we choose to keep to ω to avoid any confusion. Still, it is worth noting that many of the following results holds for the general primitive n^{th} root of unity ζ.

The aim of this section is to show that the canonical generators b_j of C are geometrically well suited for bounded distance decoding. That means we want to find an upper bound on the dual basis of the lattice $\mathbf{Log}\, C$.

In Lemma 8.18, we defined the canonical generators of C to be $b_j = (\omega^j - 1)/(\omega - 1)$. Associate b_j for $j \in G\backslash\{1\}$ with the log-embeddings:

$$\mathbf{b}_j = \mathbf{Log}(b_j).$$

By Lemma 8.18 we know b_j together with $\pm\omega$ generate the cyclotomic units C. Since $\mathbf{Log}\, C$ is a sublattice of $\mathbf{Log}\, R^*$, then $\mathbf{b}_j$ for $j \in G\backslash\{1\}$ forms a basis for $\mathbf{Log}\, C$, because $\pm\omega$ defines the kernel of $\mathbf{Log}$, as discussed in Section 8.5.1.

This is the basis we want to apply the round-off algorithm to, hence we

need to bound the dual basis. This whole section will lead up to proving that $\left\|\mathbf{b}_j^\vee\right\|^2 \leq O(n^{-1}\log^3 n)$ for all $j \in G\backslash\{1\}$. We do this by first proving that the dual basis vectors all have equal Euclidean norm, and then find an upper bound for them in terms of the Dirichlet L-series $L(1,\kappa)$ where κ is the even characters of $\mathbb{Z}_n^*$.

First, we prove that the Euclidean norm of $\mathbf{b}_j^\vee$ are all equal to each other.

Lemma 8.19 ([Cra+16, Lemma 3])**.** *For all $j \in G\backslash\{1\}$ we have*

$$\left\|\boldsymbol{b}_j^\vee\right\|^2 = |G|^{-1} \sum_{\kappa\in\hat{G}\backslash\{1\}} |\lambda_\kappa|^{-2}.$$

Proof. Since we know the rows and columns of a G-circulant matrix have the same Euclidean norm, we want to try to relate the $\mathbf{b}_j$ to a G-circulant matrix. Recall that $z_j = \omega^j - 1$ and define

$$\boldsymbol{z}_j := \mathbf{Log}(z_j).$$

Since $z_j = b_j z_1$, it follows that $\boldsymbol{z}_j = \mathbf{b}_j + \boldsymbol{z}_1$. Now define the matrix $\mathbf{Z}$ to be the matrix with $\boldsymbol{z}_{j^{-1}}$ in its j^{th} column. Then the $(i,j)^{\text{th}}$ entry of $\mathbf{Z}$ (for $i,j \in G$) is defined by $\log|\sigma_{j^{-1}}(z_i)| = \log|\omega^{ij^{-1}} - 1|$. So the $(i,j)^{\text{th}}$ entry of $\mathbf{Z}$ is only dependent on ij^{-1}, hence $\mathbf{Z}$ is a G-circulant matrix associated with $\boldsymbol{z}_1 = (\log|\omega^i - 1|)_{i\in G}$.

Let $\boldsymbol{z}_j^\vee$ denote the dual to $\boldsymbol{z}_j$. Since $\langle \boldsymbol{z}_i^\vee, \boldsymbol{z}_j\rangle = \delta_{ij}$, $\boldsymbol{z}_j^\vee$ make out the columns of $\mathbf{Z}^{-T}$. Note that $\mathbf{Z}^{-1}$ is well defined because the eigenvalues of $\mathbf{Z}$ are $\lambda_\kappa = \langle \boldsymbol{z}_1, \boldsymbol{\kappa}\rangle$ (as shown in Lemma 8.8).

Now we claim that $\mathbf{b}_j^\vee = \boldsymbol{z}_j^\vee - |G|^{-1}\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle\mathbf{1}$ (i.e. the projection of $\boldsymbol{z}_j^\vee$ orthogonal to $\mathbf{1}$). For this to be the dual basis, we need to prove two things:

1. That $\mathbf{b}_j^\vee \in \operatorname{span}(\mathbf{b}_i)_{i\in G\backslash\{1\}}$.
2. That $\langle \mathbf{b}_j^\vee, \mathbf{b}_i\rangle = \delta_{ji}$ for all $i,j \in G\backslash\{1\}$.

We start by proving $\mathbf{b}_j^\vee \in \text{span}(\mathbf{b}_i)_{i\in G\setminus\{1\}}$. Since $\text{span}(\mathbf{b}_i)_{i\in G\setminus\{1\}}$ defines the space orthogonal to the all-one vector $\mathbf{1}$, $\mathbf{b}_j^\vee \in \text{span}(\mathbf{b}_i)_{i\in G\setminus\{1\}}$ is equivalent to proving that $\mathbf{b}_j^\vee$ is orthogonal to $\mathbf{1}$ for all $j \in G\setminus\{1\}$:

$$\langle \mathbf{b}_j^\vee, \mathbf{1}\rangle = \langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle - |G|^{-1}\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle \underbrace{\langle \mathbf{1}, \mathbf{1}\rangle}_{|G|} = 0,$$

as desired. Next we check the second condition:

$$\langle \mathbf{b}_j^\vee, \mathbf{b}_i\rangle = \langle \boldsymbol{z}_j^\vee - |G|^{-1}\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle\mathbf{1}, \mathbf{b}_i\rangle = \langle \boldsymbol{z}_j^\vee, \mathbf{b}_i\rangle = \langle \boldsymbol{z}_j^\vee, \boldsymbol{z}_i - \boldsymbol{z}_1\rangle = \delta_{ji},$$

where the second equality holds because $\langle \mathbf{b}_j, \mathbf{1}\rangle = 0$ for all $j \in G\setminus\{1\}$, and the third follows from the fact that $\mathbf{b}_i = \boldsymbol{z}_i - \boldsymbol{z}_1$. Hence $\mathbf{b}_j^\vee = \boldsymbol{z}_j^\vee - |G|^{-1}\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle\mathbf{1}$.

Now we want to calculate the Euclidean norm of $\mathbf{b}_j^\vee$:

$$\begin{aligned}
\left\|\mathbf{b}_j^\vee\right\|^2 &= \langle \mathbf{b}_j^\vee, \mathbf{b}_j^\vee\rangle, \\
&= \langle \boldsymbol{z}_j^\vee + |G|^{-1}\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle\mathbf{1},\ \boldsymbol{z}_j^\vee + |G|^{-1}\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle\mathbf{1}\rangle, \\
&= \left\|\boldsymbol{z}_j^\vee\right\|^2 - |G|^{-1}\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle^2. \qquad (8.9)
\end{aligned}$$

Therefore, we want to calculate $\left\|\boldsymbol{z}_j^\vee\right\|^2$ and $\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle$. We begin by finding $\left\|\boldsymbol{z}_j^\vee\right\|$. By Lemma 8.8, the eigenvalues of $\mathbf{Z}$ are defined to be $\lambda_\kappa = \langle \boldsymbol{z}_1, \boldsymbol{\kappa}\rangle = \sum_{j\in G} z_j \cdot \overline{\kappa(j)}$, for the eigenvectors $\boldsymbol{\kappa} \in \hat{G}$. Hence, the eigenvalues of $\mathbf{Z}^{-T}$ are $\langle \boldsymbol{z}_1^\vee, \boldsymbol{\kappa}\rangle = \lambda_\kappa^{-1}$. Since all rows and columns of a G-circulant matrix have the same Euclidean norm, and by Equation (8.3), we know

$$\left\|\boldsymbol{z}_j^\vee\right\|^2 = \|\boldsymbol{z}_1^\vee\|^2 = |G|^{-1}\sum_{\kappa\in\hat{G}} |\lambda_\kappa|^{-2}. \qquad (8.10)$$

Lastly, we want to find the value of $\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle$. From the fact that $\boldsymbol{z}_j = \mathbf{b}_j + \boldsymbol{z}_1$, we get

$$\langle \boldsymbol{z}_j^\vee, \mathbf{1}\rangle = \langle \boldsymbol{z}_1^\vee, \mathbf{1}\rangle = \langle \boldsymbol{z}_j^\vee, \boldsymbol{\kappa}_1\rangle = \lambda_{\kappa_1}^{-1}, \qquad (8.11)$$

where κ_1 is the principal character in $\hat{G}$. Substituting Equations (8.10) and (8.11) into Equation (8.9) gives the desired result

$$\left\|\mathbf{b}_j^\vee\right\|^2 = |G|^{-1} \sum_{\kappa \in \hat{G}\setminus\{1\}} |\lambda_\kappa|^{-2}.$$

□

Now we have proved that all dual basis vectors $\mathbf{b}_j^\vee$ have the same Euclidean norm. Clearly, finding an upper bound for the eigenvectors λ_κ of $\mathbf{Z}$ gives us an upper bound on the dual basis vectors $\mathbf{b}_j^\vee$. Recall $G = \mathbb{Z}_n^*/\{\pm 1\}$ and that the characters in $\hat{G}$ correspond to the even characters of $\mathbb{Z}_n^*$. That means the eigenvectors of $\mathbf{Z}$ are

$$\begin{aligned}\lambda_\kappa = \langle \boldsymbol{z}_1, \boldsymbol{\kappa} \rangle &= \sum_{a \in G} \overline{\kappa(a)} \log|1 - \omega_n^a|, \\ &= \frac{1}{2} \sum_{a \in \mathbb{Z}_n^*} \overline{\kappa(a)} \log|1 - \omega_n^a|, \end{aligned} \tag{8.12}$$

because $|1 - \omega_n^a| = |1 - \omega_n^{-a}|$ (since $(|\sigma_j(x)| = |\sigma_{-j}(x)|)$.

Theorem 4 in [Cra+16], which is a combination of Lemma 4.8 and Theorem 4.9 in [Was97], states that:

Theorem 8.20 ([Cra+16, Theorem 4])**.** *Let κ be an even Dirichlet character of conductor $f > 1$. and let $\omega_f = e^{\frac{2\pi}{f} i} \in \mathbb{C}$. Then*

$$\left| \sum_{a \in \mathbb{Z}_f^*} \overline{\kappa(a)} \cdot \log|1 - \omega_f^a| \right| = \sqrt{f} |L(1, \kappa)|. \tag{8.13}$$

We will not prove this theorem here. We note, however, that if we can get the summation in Equation (8.12) on the form of the left hand side of Equation (8.13), we can consequently use Theorem 8.14 to get an upper bound for the $\mathbf{b}_j^\vee$'s.

Since f is the conductor of an even character $\kappa \in \mathbb{Z}_n^*$, f divides n. Let $\psi : \mathbb{Z}_n^* \to \mathbb{Z}_f^*$ be given by reduction modulo f. Hence, for $a \in \mathbb{Z}_f^*$, we

have $\kappa(b_i) = \kappa(a)$ for all $b_i \in \mathbb{Z}_n^*$ that are such that $b_i \equiv a \pmod f$. That means we can put $\overline{\kappa(a)}$ in Equation (8.12) outside a parenthesis and get:

$$\sum_{a\in\mathbb{Z}_n^*} \overline{\kappa(a)} \cdot \log|1-\omega_n^a| = \sum_{a\in\mathbb{Z}_f^*} \overline{\kappa(a)} \left(\sum_{\substack{b\in\mathbb{Z}_n^* \\ \psi(b)=a}} \log|1-\omega_n^b| \right),$$

$$= \sum_{a\in\mathbb{Z}_f^*} \overline{\kappa(a)} \cdot \log \left(\prod_{\substack{b\in\mathbb{Z}_n^* \\ \psi(b)=a}} |1-\omega_n^b| \right). \tag{8.14}$$

(We disregard the $\frac{1}{2}$ term for now.) For a primitive k^{th} root of unity ω of a field $\mathbb{F}$, the polynomial $x^k - 1 \in \mathbb{F}[x]$ can be factorised as $x^k - 1 = \prod_{j=0}^{k-1}(x - \omega^j)$. Divide both sides by x^k and let $y = x^{-1}$, then we get the equality $1 - y^k = \prod_{j=0}^{k-1}(1 - y\omega^j)$. Now, for each $a \in \mathbb{Z}_f^*$ there are n/f b's in $\mathbb{Z}_n^*$ such that $\psi(b) = a$. That means

$$\prod_{\substack{b\in\mathbb{Z}_n^* \\ \psi(b)=a}} |1-\omega_n^b| = \prod_{r=0}^{\frac{n}{f}-1} |1-(\omega_n^a)^r| = |1-\omega_n^{\frac{n}{f}a}| = |1-\omega_f^a|.$$

Note that ω_n^a is a primitive root of unity as a is coprime to n by definition. Put this into Equation (8.14) and use Theorem 8.20 to get

$$\sum_{a\in\mathbb{Z}_n^*} \overline{\kappa(a)} \cdot \log|1-\omega_n^a| = \sum_{a\in\mathbb{Z}_f^*} \overline{\kappa(a)} \cdot \log|1-\omega_f^a|,$$

$$= \sqrt{f}\,|L(1,\kappa)|,$$

Hence, we have proven the Corollary to Theorem 8.20:

Corollary 8.21 ([Cra+16, Cororllary 1])**.** *Suppose $f > 1$ divides a prime power n. For any even Dirichlet character κ of conductor f.*

$$\left| \sum_{a\in\mathbb{Z}_n^*} \overline{\kappa(a)} \cdot \log|1-\omega_n^a| \right| = \sqrt{f}\,|L(1,\kappa)|$$

Next, we state the clear result of relating this back to the norm of the dual basis:

Lemma 8.22. *Let $n = p^k$ for a prime p, and let $\{\boldsymbol{b}_j^\vee\}_{j\in G\setminus\{1\}}$ denote the basis dual to $\{\boldsymbol{b}_j\}_{j\in G\setminus\{1\}}$. Then the Euclidean norms $\left\|\boldsymbol{b}_j^\vee\right\|$ are all equal, and*

$$\left\|\boldsymbol{b}_j^\vee\right\|^2 = 4|G|^{-1} \sum_{\kappa\in\hat{G}\setminus\{1\}} f_\kappa^{-1}|L(1,\kappa)|^{-2}.$$

Proof. From Equation (8.12) and Corollary 8.21 we deduce $|\lambda_\kappa| = \left|\frac{1}{2}\sqrt{f_\kappa}\, L(1,\kappa)\right|$. By Lemma 8.19 the result follows:

$$\left\|\mathbf{b}_j^\vee\right\|^2 = |G|^{-1} \sum_{\kappa\in\hat{G}\setminus\{1\}} |\lambda_\kappa|^{-2} = 4|G|^{-1} \sum_{\kappa\in\hat{G}\setminus\{1\}} f_\kappa^{-1}|L(1,\kappa)|^{-2}. \tag{8.15}$$

□

Finally, we can find an upper bound on the $\mathbf{b}_j^\vee$'s in terms of n.

Theorem 8.23 ([Cra+16, Theorem 3])**.** *Let $n = p^k$ for a prime p, and let $\{\boldsymbol{b}_j^\vee\}_{j\in G\setminus\{1\}}$ denote the basis dual to $\{\boldsymbol{b}_j\}_{j\in G\setminus\{1\}}$. Then*

$$\left\|\boldsymbol{b}_j^\vee\right\|^2 \le 2k|G|^{-1}\cdot(c+\ell(n)^2) \le O(n^{-1}\cdot\log^3 n),$$

for some constant $c > 0$.

Proof. By Lemma 8.22 we have

$$\left\|\mathbf{b}_j^\vee\right\|^2 = 4|G|^{-1} \sum_{\kappa\in\hat{G}\setminus\{1\}} f_\kappa^{-1}|L(1,\kappa)|^{-2}. \tag{8.16}$$

To find an upper bound on $\left\|\mathbf{b}_j^\vee\right\|^2$ we will use Theorem 8.14.

First we consider contributions coming from quadratic characters. For all primes p, there are no more than three quadratic characters (with finite conductor) [MV06, Chapter 9.3]. Hence, by Equation (8.5) from Theorem 8.14 there exist some constant $c_\kappa > 0$ such that $f_\kappa^{-1}|L(1,\kappa)|^{-2} \le c_\kappa^2$.

$$\sum_{\substack{\kappa\in\hat{G}\setminus\{1\}\\ \text{quad }\kappa}} f_\kappa^{-1}|L(1,\kappa)|^{-2} \le \sum_{\substack{\kappa\in\hat{G}\setminus\{1\}\\ \text{quad }\kappa}} c_\kappa^2 = c.$$

Where c is some positive constant.

Now we move on to the non-quadratic characters. Here we use Equation (8.4) to first bound $|L(1,\kappa)|^{-2} \leq \ell(f_\kappa)^2 \leq \ell(n)^2$. Then we want to find an upper bound for $\sum_{\kappa \in \hat{G}\setminus\{1\}} f_\kappa$.

The fact that $f_\kappa | n$ means $f_\kappa = p^l$ for some $l = 1, .., k$ (since $f_\kappa > 1$). The maximum number of Dirichlet characters with conductor $f = p^l$ is p^l. At most half of these are even (and we are only looking at even characters when working with $G = \mathbb{Z}_n^*/\{\pm 1\}$). So we can bound each f_κ above by the number of possible even characters times its conductor:

$$\sum_{\kappa \in \hat{G}\setminus\{1\}} f_\kappa^{-1} \leq \sum_{\kappa \in \hat{G}\setminus\{1\}} \frac{p^l}{2}\frac{1}{p^l} = \frac{k}{2}.$$

This means we can bound the sum of non-quadratic character thus:

$$\sum_{\substack{\kappa \in \hat{G}\setminus\{1\} \\ \text{non-quad } \kappa}} f_\kappa^{-1} |L(1,\kappa)|^{-2} \leq \ell(n)^2 \frac{k}{2}.$$

Now we plug this into Equation (8.15) to prove our theorem:

$$\begin{aligned}
\left\|\mathbf{b}_j^\vee\right\|^2 &= 4|G|^{-1} \sum_{\kappa \in \hat{G}\setminus\{1\}} f_\kappa^{-1} |L(1,\kappa)|^{-2}, \\
&\leq 4|G|^{-1} \left(c_1 + \frac{k}{2}\ell(n)^2\right), \\
&= 4|G|^{-1} \left(c_1 + c_2 \frac{k}{2} \ln^2(n)\right), \\
&\leq 2k|G|^{-1}(c_3 + c_1 \ln^2(n)), \\
&\leq 2|G|^{-1} \ln(n)(c_3 + c_2 \ln^2(n)),
\end{aligned}$$

where $c_i > 0$ are constants. Lastly, we recall that $|G| = \frac{\varphi(n)}{2}$. In the words of Hardy and Wright ([HW54, Chapter 18.4]), the order of $\varphi(n)$ is almost always "nearly n". Hence we obtain our desired result:

$$\left\|\mathbf{b}_j^\vee\right\|^2 \leq O(n^{-1} \log^3(n)).$$

□

8.7 The Algorithm

Now that we have a bound on the dual basis of $\mathbf{Log}\, C$, we are ready to prove our main result in this chapter; the round-off algorithm applied to the log-unit-lattice in order to solve the SG-PIP problem.

Theorem 8.24 ([Cra+16, Theorem])**.** *Let χ be a distribution over $\mathbb{Q}(\omega)$ with the property that for any tuple of vectors $\boldsymbol{v}_1, ..., \boldsymbol{v}_{\varphi(n)/2-1} \in \mathbb{R}^{\varphi(n)/2}$ of Euclidean norm 1 that are orhtogonal to the all-1 vector $\mathbf{1}$, the probability that $|\langle \boldsymbol{Log}(g), \boldsymbol{v}_i \rangle| < cn^{\frac{1}{2}} \cdot (\log n)^{-\frac{3}{2}}$ holds for all $i = 1, ..., \frac{\varphi(n)}{2} - 1$ is at least some $P > 0$. Then there is an efficient algorithm that given $g' = ug$ where g is chosen from χ and $u \in C$ is a cyclotomic unit, outputs an element of the form $\pm\omega^j g$ with probability at least P.*

Proof. The input is g', a generator. We want to prove that the output of the algorithm is the sufficiently short generator $\pm\omega^j g$.

As described in Section 8.5.3 we can apply the logarithmic embedding to the long generator $g' = ug$ to get a closest vector problem:

$$\mathbf{Log}(g') = \mathbf{Log}(u) + \mathbf{Log}(g).$$

Since $u \in C$, $\mathbf{Log}(u) \in \mathbf{Log}\, C$ which is in our lattice with basis $\mathbf{b}_j$ discussed in the previous section. We can apply the round-off algorithm from the beginning of this chapter (see Section 8.1) by putting $\boldsymbol{x} = \mathbf{Log}(u)$ and $\boldsymbol{\nu} = \mathbf{Log}(g)$, but only if $\langle \mathbf{Log}(g), \mathbf{b}_j^\vee \rangle \in [-\frac{1}{2}, \frac{1}{2})$ for all $j \in G \backslash \{1\}$.

With probability P, we can assume $|\langle \mathbf{Log}(g), \mathbf{v}_i \rangle| < cn^{\frac{1}{2}} \cdot (\log n)^{-\frac{3}{2}}$ for any tupple of vectors $\mathbf{v}_1, ..., \mathbf{v}_{\varphi(n)/2-1} \in \mathbb{R}^{\varphi(n)/2}$ where $\|\mathbf{v}_i\| = 1$ and $\langle \mathbf{v}_i, \mathbf{1} \rangle = 0$ for all $i = 1, ..., \frac{\varphi(n)}{2} - 1$. The $\mathbf{b}_j^\vee$'s are already orthogonal to $\mathbf{1}$ and they all have equal Euclidean norm, albeit not necessarily 1. However, in Lemma 8.22 we found that $\left\|\mathbf{b}_j^\vee\right\|^2 = \alpha^2$, where $\alpha^2 = 4|G|^{-1} \sum_{\kappa \in \hat{G} \backslash \{1\}} f_\kappa^{-1} |L(1, \kappa)|^{-2}$, which means we can scale the norm of the $\mathbf{b}_j^\vee$'s thus:

$$\alpha^{-1} \left\|\mathbf{b}_j^\vee\right\| = 1$$

(we can safely assume $\alpha > 0$). Note that $\alpha^{-1}\mathbf{b}_j^\vee$ is still orthogonal to $\mathbf{1}$. By Theorem 8.23 $c'n^{\frac{1}{2}}(\log n)^{-\frac{3}{2}} < \alpha^{-1}$ for some constant $c' > 0$. By assumption, with probability P, we get;

$$\begin{aligned}
|\langle \mathbf{Log}(g), \alpha^{-1}\mathbf{b}_j^\vee \rangle| &< cn^{\frac{1}{2}}(\log n)^{-\frac{3}{2}} \\
\alpha^{-1}|\langle \mathbf{Log}(g), \mathbf{b}_j^\vee \rangle| &< cn^{\frac{1}{2}}(\log n)^{-\frac{3}{2}} \\
c'n^{\frac{1}{2}}(\log n)^{-\frac{3}{2}}|\langle \mathbf{Log}(g), \mathbf{b}_j^\vee \rangle| &< cn^{\frac{1}{2}}(\log n)^{-\frac{3}{2}} \\
|\langle \mathbf{Log}(g), \mathbf{b}_j^\vee \rangle| &< \frac{c}{c'}
\end{aligned}$$

for all j's. So if $c \leq \frac{c'}{2}$ we have our desired upper bound on the dual basis of $\mathbf{Log}\, C$, with a probability P. Hence, with probability P, when we input $\mathbf{B}^\vee$ and $\mathbf{Log}(g')$ into the round-off algorithm, it outputs $\mathbf{Log}(u) \in \mathbf{Log}\, C$.

Then we can find integer coefficients a_j such that $\mathbf{Log}(u) = \sum_{j \in G\setminus\{1\}} a_j \mathbf{b}_j$ and compute $v = \prod_{j \in G\setminus\{1\}} b_j^{a_j}$. Then

$$\mathbf{Log}(v) = \sum_{j \in G\setminus\{1\}} a_j \mathbf{Log}(b_j) = \sum_{j \in G\setminus\{1\}} a_j \mathbf{b}_j = \mathbf{Log}(u),$$

which means $v = \pm\omega^k u$ for some integer k. Hence $g'/v = \pm\omega^j g$ as desired.

□

8.8 Distributions

We end our chapter, and in fact, this thesis, by showing how choosing χ to be the continuous Gaussian distribution gives a good bound on P in Theorem 8.24.

For χ to be a good distribution, we need to show that the probability that $|\langle \mathbf{Log}(g), \boldsymbol{a} \rangle| < cn^{\frac{1}{2}}(\log\, n)^{-\frac{3}{2}}$, for any $\boldsymbol{a} = (a_j)_{j \in G} \in \mathbb{R}^{\varphi(n)/2}$ that is orthogonal to $\mathbf{1}$ and has Euclidean norm 1, is high.

Note that for any $b \in \mathbb{Q}(\omega)$, $\sigma(b) = x + iy$ for some $x, y \in \mathbb{R}$. Hence, if g is sampled by χ , and we say $|\sigma_j(g)| = (X_j + Y_j^2)^{\frac{1}{2}}$, $X_j, Y_j \in \mathbb{R}$, then

X_j and Y_j is also sampled by χ. Hence, in this chapter, we will consider χ to be a distribution on the real numbers. We want to show that for $X_1, ..., X_n$ and $Y_1, ..., Y_n$ sampled by χ, and for any collection of vectors $\boldsymbol{a}_1, ..., \boldsymbol{a}_l \in \mathbb{R}^n$, where $\|\boldsymbol{a}_i\| = 1$ and $\langle \boldsymbol{a}, \mathbf{1} \rangle = 0$, the probability that

$$|\langle \mathbf{Log}(g), \boldsymbol{a}_j \rangle| = \left| \sum_{i=1}^{n} \log((X_i^2 + Y_i^2)^{\frac{1}{2}} a_{ji} \right| \geq t$$

for just one j, where t is some tail bound, is very small.

We will show in Lemma 8.28 that this is true for $\chi = N(0, r)$, the continuous Gaussian distribution. However, the results should be possible to extend to other distributions.

Before we begin to prove anything, we introduce a concept that will be used in both the proofs of this section:

Definition 8.25. For $\alpha, \beta > 0$, a random variable X is (α, β)-*subexponential* if

$$\mathbb{E}[\cosh(\alpha X)] \leq \beta.$$

We begin by proving a tail bound for the sum of some independently sampled (α, β)-subexponential variables multiplied by any real number a_i. Note that this result is independent of distribution.

Lemma 8.26 ([Cra+16, Lemma 4])**.** *Let $X_1, ..., X_k$ be independent centred (i.e. $\mathbb{E}[X_i] = 0$) (α, β)-subexponential random variables. Then, for any $\boldsymbol{a} = (a_1, ..., a_k) \in \mathbb{R}^k$ and every $t \geq 0$,*

$$\Pr\left[\left| \sum_{i=0}^{k} a_i X_i \right| \geq t \right] \leq 2e^{-\eta},$$

where

$$\eta = \min\left(\frac{\alpha^2 t^2}{8\beta \|\boldsymbol{a}\|^2}, \frac{\alpha t}{2 \|\boldsymbol{a}\|_\infty} \right).$$

Proof. We claim that $f(x) = e^{-\delta x} + \delta x - 1 \geq 0$ for all $x, \delta \in \mathbb{R}$. (This can easily be verified by checking that of $f(x)$ has a minima for $x = 0$ for

both positive and negative values of δ). Hence we have the inequality

$$\begin{aligned} e^{\delta x} - \delta x - 1 &\leq (e^{\delta x} - \delta x - 1) + (e^{-\delta x} - \delta x - 1), \\ &= e^{\delta x} + e^{-\delta x} - 2, \\ &= 2\cosh(\delta x) - 2, \\ &= 2(\cosh(\delta x) - 1), \\ &\leq 2\delta^2(\cosh(x) - 1), \end{aligned} \tag{8.17}$$

where the last inequality holds for $|\delta| \leq 1$. By scaling, we can assume without loss of generality that $\alpha = 1$. Let X be $(1, \beta)$-subexponential variable. Then applying the inequality in Equation (8.17) to $\mathbb{E}[f(X)]$ gives

$$\begin{aligned} \mathbb{E}[e^{\delta X}] - \delta\mathbb{E}[X] - 1 &\leq +2\delta^2\mathbb{E}[\cosh(\delta X) - 1], \\ \mathbb{E}[e^{\delta X}] &\leq 1 + 2\delta^2(\beta - 1), \\ &\leq 1 + 2\delta^2\beta, \\ &\leq e^{2\delta^2\beta}, \end{aligned} \tag{8.18}$$

where the last inequality follows from the Taylor series. First note that for $\mu > 0$

$$\Pr\left[\sum a_i X_i \geq t\right] = \Pr\left[e^{\mu \sum a_i X_i} \geq e^{\mu t}\right]. \tag{8.19}$$

where X_i are independently sampled $(1, \beta)$-subexponential variables. Here we state Markov's inequality:

Theorem 8.27 (Markov's Inequality)**.** *If X is a nonnegative random variable and $a > 0$, then*

$$\Pr(X \geq a) \leq \frac{\mathbb{E}[X]}{a}.$$

Applying this to Equation (8.19) gives

$$\begin{aligned}\Pr\left[e^{\mu\sum a_i X_i} \geq e^{\mu t}\right] &\leq \frac{\mathbb{E}\left[e^{\mu\sum a_i X_i}\right]}{e^{\mu t}}, \\ &= e^{-\mu t}\prod \mathbb{E}\left[e^{\mu a_i X_i}\right].\end{aligned}$$

Now, put $\delta = \mu a_i$ and assume $\mu||\boldsymbol{a}||_\infty < 1$. Then apply Equation (8.18) to each i, we get

$$\begin{aligned}\Pr\left[\sum a_i X_i \geq t\right] &\leq e^{-\mu t}e^{2\mu^2\beta\sum a_i^2}, \\ &= e^{-\mu t+2\mu^2\beta||\boldsymbol{a}||_2^2}.\end{aligned}$$

Putting $\mu = \min\left(\frac{t}{4\beta\|\boldsymbol{a}\|^2}, \frac{1}{\|\boldsymbol{a}\|_\infty}\right)$ makes the above bound

$$\Pr\left[\sum a_i X_i \geq t\right] \leq e^{-\frac{\mu t}{2}} = e^{-\eta}.$$

The theorem deals with the absolute value, so we note that the same argument can be applied to $-\boldsymbol{a}$ and hence the proof is completed. □

This theorem tells us that for any $\boldsymbol{a} \in \mathbb{R}^k$ and independently sampled $(1,\beta)$-subexponential variables X_i, the probability that the $|\sum_i a_i X_i|$ does not lie inside a tail bound, i.e. that the absolute value is greater than t, is in fact quite small.

If we can prove that for X_j, Y_j independently sampled from χ, that $\log((X_j^2+Y_j^2)^{\frac{1}{2}})$ are centred $(1,\beta)$-subexponential variables. Then we can apply this theorem to our assumption that $|\langle \mathbf{Log}(g), \boldsymbol{a}\rangle| < cn^{\frac{1}{2}}(\log(n)^{-\frac{3}{2}}$ for one $\boldsymbol{a}$ (where $\|\boldsymbol{a}\| = 1$ and $\langle \boldsymbol{a}, \mathbf{1}\rangle = 0$). If we can also prove that for several sums of this form, the probability that just one of them is greater than the tail bound is still low, then we have proved that χ is a good distribution for Theorem 8.24.

As stated at the beginning, we will prove this for the continuous Gaussian distribution $\chi = N(0, r)$.

Lemma 8.28 ([Cra+16, Lemma 5])**.** *Let $X_1, ..., X_k, Y_1, ..., Y_k$ be independent and identically distributed $N(0, r)$ variables for some $r > 0$, and let $\hat{X}_i = (X_i^2 + Y_i^2)^{\frac{1}{2}}$ Then, for any vectors $\boldsymbol{a}^{(1)}, ..., \boldsymbol{a}^{(l)} \in \mathbb{R}^k$ of Euclidean norm 1 that are orthogonal to the all-1 vector, and every $t \geq c$ for some universal constant c,*

$$\Pr\left[\exists j, \left|\sum_{i=0}^{k} a_i^{(j)} \log(\hat{X}_i)\right| \geq t\right] \leq 2le^{-\frac{t}{2}}.$$

Proof. The union bound states that for a countable set of events $A_1, A_2, ...$, $\Pr[\cup_i A_i] \leq \sum_i \Pr[A_i]$. Therefore, we define the event

$$A_j = \left|\sum_{i=0}^{k} a_i^{(j)} \log \hat{X}_i\right| \geq t.$$

Then $\Pr\left[\exists j, \left|\sum_{i=0}^{k} a_i^{(j)} \log \hat{X}_i\right| \geq t\right] = \Pr\left[\cup_j A_j\right]$, i.e the probability that at least one A_j occurs. Assume that $\Pr(A_j)$ is the same for all $j = 1, .., l$. Then, by the union bound:

$$\begin{aligned}\Pr\left[\exists j\,;\, \left|\sum_{i=0}^{k} a_i^{(j)} \log \hat{X}_i\right| \geq t\right] &= \Pr\left[\cup_j A_j\right], \\ &\leq \sum_{i=1}^{l} \Pr(A_i), \\ &= l \Pr(A_1). \end{aligned} \tag{8.20}$$

Hence we only need to find an upper bound for $\Pr(A_1)$. For simplicity, we denote $\boldsymbol{a}^{(1)} = \boldsymbol{a}$ and recall that, by assumption, $\sum_{i=0}^{k} a_i = \langle \boldsymbol{a}, \mathbf{1} \rangle = 0$ and $\|\boldsymbol{a}\| = 1$.

We can assume, without loss of generality, that $r = 1$. The $\hat{X}_i$'s in the theorem is a Chi distribution with 2 degrees of freedom, also known as a *Rayleigh distribution*. The probability density function is given by $p(\theta, 2) = \theta e^{-\frac{\theta^2}{2}}$ and

$$\mathbb{E}[\hat{X}_i] = \int_{-\infty}^{\infty} \theta p(\theta, 2)\, \mathrm{d}\theta = \int_{-\infty}^{\infty} \theta^2 e^{-\frac{\theta^2}{2}}\, \mathrm{d}\theta = \sqrt{\frac{\pi}{2}}.$$

In fact, $\mathbb{E}[\hat{X}_i^{-1}] = \mathbb{E}[\hat{X}_i]$, so both are finite. This implies that

$$\mathbb{E}[\cosh(\log \hat{X}_i)] = \frac{\mathbb{E}\left[e^{\log \hat{X}_i}\right] + \mathbb{E}\left[e^{-\log \hat{X}_i}\right]}{2} = \frac{1}{2}(\mathbb{E}[\hat{X}_i + \mathbb{E}[\hat{X}_i^{-1}]) \leq \beta',$$

for some $\beta' > 0$. Hence, $\log \hat{X}_i$ is a $(1, \beta')$-subexponential random variable. From this it follows that the variables $Z_i = \log \hat{X}_i - \mathbb{E}[\log \hat{X}_i]$ are centred $(1, \beta)$-subexponential random variables for some constant $\beta > 0$. Hence, we can apply Lemma 8.26 to the Z_i's and get:

$$\Pr\left[\left|\sum_{i=0}^{k} a_i Z_i\right| \geq t\right] = \Pr\left[\left|\sum_{i=0}^{k} a_i \log \hat{X}_i - \underbrace{\sum_{i=0}^{k} a_i \mathbb{E}[\log \hat{X}_i]}_{=0}\right| \geq t\right],$$

$$\leq 2e^{-\eta}.$$

Where $\sum_{i=0}^{k} a_i \mathbb{E}[\log \hat{X}_i] = 0$ follows from the fact that because $\log(\hat{X})$ are independently and identically distributed, they all have the same expected value. Hence we can move the expected value outside the summation, and we know $\sum_i a_i = 0$.

Note that $\|\boldsymbol{a}\|^2 = 1$ and this also implies that $\|\boldsymbol{a}\|_\infty \leq 1$. So, here, $\eta = \min\left(\frac{t^2}{8\beta}, \frac{t}{2\|\boldsymbol{a}\|_\infty}\right) \geq \frac{t}{2}$ for $t \geq 4\beta$. We therefore get

$$\Pr\left[\left|\sum_{i=0}^{k} a_i \log \hat{X}_i\right| \geq t\right] \leq 2e^{-\frac{t}{2}}.$$

Putting this into Equation (8.20), proves the Lemma. □

Hence, if we sample the short generator g from the continuous Gaussian distribution $N(0, r)$, the probability that for all i, $|\langle \mathbf{Log}(g), \mathbf{v}_i \rangle| \leq \hat{c}$, for all tuples of vectors $\mathbf{v}_1, ..., \mathbf{v}_{\varphi(n)/2-1} \in \mathbb{R}^{\varphi(n)/2}$ that are such that $\|\mathbf{v}_i\| = 1$ and $\langle \mathbf{v}_i, \mathbf{1} \rangle = 0$, and where $\hat{c}$ is the bound given in Theorem 8.24, is bounded above by $2e^{-\frac{\hat{c}}{2}}$. That is, the probability of the algorithm returning the correct result for this distribution is high.

Cramer et al. shows in [Cra+16, Lemma 5] that small perturbations in the continuous Gaussian distribution still satisfy the conditions of Theorem

8.24. They also show they can apply this Lemma to show that a discrete Gaussian distribution with greater than a certain bound, see discussion at the end of Section 5 in [Cra+16].

Abbreviations

BDD Bounded Distance Decoding. 40

BKW Blum-Kalai-Wasserman algorithm. 15

BKZ Block Korkin-Zolotarev. 66

CFT Continuous Fourier Transform. 27

CVP Closest Vector Problem. 39

DFT Discrete Fourier Transform. 23

LLL Lenstra-Lenstra-Lovász algorithm. 45

LPN Learning Parity with Noise. 9

LWE Learning With Errors. 9

PIP Principal Ideal Problem. 73

SG-PIP Shortest Generator Principal Ideal Problem. 73

SVP Shortest Vector Problem. 39

References

[Alb+15] Martin Albrecht et al. "On the complexity of the BKW algorithm on LWE". In: *Designs, Codes and Cryptography* 74 (Feb. 2015). DOI: 10.1007/s10623-013-9864-x.

[Apo76] Tom M. Apostol. *Introduction to Analytic Number Theory*. New York, NY: Springer New York, 1976, pp. 129–145. ISBN: 978-1-4757-5579-4. DOI: 10.1007/978-1-4757-5579-4_7. URL: https://doi.org/10.1007/978-1-4757-5579-4_7.

[Ber14] Daniel J. Bernstein. *A subfield-logarithm attack against ideal lattices*. [retrieved 12.06.2021]. 2014. URL: https://blog.cr.yp.to/20140213-ideal.html.

[BJN94] Phani Bhushan Bhattacharya, Surender Kumar Jain, and SR Nagpaul. *Basic abstract algebra*. Cambridge University Press, 1994.

[CN11] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *Advances in Cryptology – ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1–20. ISBN: 978-3-642-25385-0.

[Cra+16] Ronald Cramer et al. "Recovering Short Generators of Principal Ideals in Cyclotomic Rings". In: *Advances in Cryptology – EUROCRYPT 2016*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 559–585. ISBN: 978-3-662-49896-5.

[DTV15] Alexandre Duc, Florian Tramèr, and Serge Vaudenay. "Better Algorithms for LWE and LWR". In: *Advances in Cryptology – EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 173–202. ISBN: 978-3-662-46800-5.

[Gal12] Stephen Galbraith. *Mathematics of Pulic Key Encryption*. ch17. Cambridge University Press, 2012.

[Gjø19a] Kristian Gjøsteen. *A Brief Introduction to Symmetric Cryptography - Lecture Notes*. NTNU, Oct. 2019.

[Gjø19b] Kristian Gjøsteen. *Pulic Key encryption - Lecture Notes*. NTNU, Oct. 2019.

[GNR10] Nicolas Gama, Phong Q Nguyen, and Oded Regev. "Lattice enumeration using extreme pruning". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2010, pp. 257–278.

[Hoe63] Wassily Hoeffding. "Probability Inequalities for Sums of Bounded Random Variables". In: *Journal of the American Statistical Association* 58.301 (1963), pp. 13–30. ISSN: 01621459. URL: http://www.jstor.org/stable/2282952.

[HW54] Godfrey Harold Hardy and Edward Maitland Wright. *An introduction to the theory of numbers*. 3rd. Oxford university press, 1954.

[KZ73] Aleksandr Korkine and G Zolotareff. "Sur les formes quadratiques". In: *Mathematische Annalen* 6.3 (1873), pp. 366–389.

[LLL82] A.K Lenstra, H.W. Lenstra, and L Lovász. "Factoring polynomials with rational coefficients". In: *Mathematische Annalen* 261.4 (1982), pp. 515–534. DOI: 10.1007/BF01457454. URL: https://doi.org/10.1007/BF01457454.

[LLS90] J. C. Lagarias, H. W. Lenstra, and C. P. Schnorr. "Korkin-Zolotarev bases and successive minima of a lattice and its reciprocal lattice". In: *Combinatorica* 10 (1990), pp. 333–338. DOI: 10.1007/BF02128669. URL: https://doi.org/10.1007/BF02128669.

[LP11] Richard Lindner and Chris Peikert. "Better Key Sizes (and Attacks) for LWE-Based Encryption". In: *Topics in Cryptology – CT-RSA 2011*. Ed. by Aggelos Kiayias. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 319–339. ISBN: 978-3-642-19074-2.

[MV06] Hugh L. Montgomery and Robert C. Vaughan. *Multiplicative Number Theory I : Classical Theory*. Cambridge Studies in Advanced Mathematics Vol. 97. Cambridge University Press, 2006. ISBN: 9780521849036. URL: http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=178884&site=ehost-live.

[NV10] Phong Q Nguyen and Brigitte Vallée. *The LLL algorithm*. Springer, 2010.

[Reg04] Oded Regev. *Lattices in Computer Science: Lecture 2 - LLL Algorithm*. [Tel Aviv University, retrieved 27.06.21]. 2004. URL: https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/lll.pdf.

[Reg05] Oded Regev. "On Lattices, Learning with Errors, Random Linear Codes, and Cryptography". In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93. ISBN: 1581139608. DOI: 10.1145/1060590.1060603. URL: https://doi.org/10.1145/1060590.1060603.

[Sam72] Pierre Samuel. *Algebraic Theory of Numbers*. Trans. by Allan J. Sillberger. London: Kershaw Publishing Company LTD, 1972.

[Sch87] C.P. Schnorr. "A hierarchy of polynomial time lattice basis reduction algorithms". In: *Theoretical Computer Science* 53

(1987), pp. 201–224. URL: https://doi.org/10.1016/0304-3975(87)90064-8.

[SE94] C.P. Schnorr and M. Euchner. "Lattice baisis reduction: Improved practical algorithms and solving subset sum problems". In: *Mathematical Programming* 66 (1994), pp. 181–199. DOI: 10.1007/BF01581144. URL: https://doi.org/10.1007/BF01581144.

[Was97] Lawrence C. Washington. *Introduction to Cyclotomic Fields*. New York, NY: Springer New York, 1997. ISBN: 978-1-4612-1934-7. DOI: 10.1007/978-1-4612-1934-7_3. URL: https://doi.org/10.1007/978-1-4612-1934-7_3.

Elisabeth Enerhaug

Algorithms For Solving The Learning With Errors Problem

NTNU
Norwegian University of
Science and Technology