2024

Ring Learning with Errors

Sarah Days-Merrill
University of Vermont

Ring Learning with Errors

A Dissertation Presented

by

Sarah Days-Merrill

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Mathematical Sciences

May, 2024

Defense Date: March 8, 2024
Dissertation Examination Committee:

Christelle Vincent, Ph.D., Advisor
Carmen Petrick Smith, Ph.D., Chairperson
Taylor Dupuy, Ph.D.
Mike Miller Eismeier, Ph.D.
Holger Hoock, DPhil, Dean of the Graduate College

Abstract

Over the last twenty years, lattice-based cryptosystems have gained interest due to their level of security against attacks from quantum computers. The main such systems are based on the hardness of Ring Learning with Errors (RLWE).

The Learning with Errors (LWE) problems were first introduced in 2005 by Regev [Reg09] and in 2010, [LPR10] developed the Ring Learning with Errors (RLWE) problems as candidates for safe encryption against quantum computers. Let K be a number field with ring of integers $\mathcal{O}_K$. For a prime q, the RLWE problems rely on samples of the form $(a, b) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K/q\mathcal{O}_K$ where a is drawn uniformly at random and $b = as + e$ where $s \in \mathcal{O}_K/q\mathcal{O}_K$ is called the secret and e is a small error term drawn from a Gaussian distribution.

Since the introduction of RLWE, attacks for solving the search and decision problems have been developed to exploit vulnerabilities of underlying number fields (mostly 2-power cyclotomic fields). An area which has been studied less rigorously is on the bounds of the security parameters to ensure safety from attack. We aim to study the security parameters for the Chi-Squared Attack from [CLS17a, CLS17b] which attempt to solve the non-dual RLWE decision problem. After testing the parameters suggested in the current literature ([CLS17b, Pei16b]), we suggest that we can find tighter bounds that rely on the norm of the prime $\mathfrak{q} \subset \mathcal{O}_K$ and the standard deviation σ of the Gaussian distribution from which the error terms are drawn.

Another area of interest involves the ways in which the error terms are sampled and how to control its growth in size. This is of particular interest for the Polynomial Ring Learning with Errors (PLWE) problems because sampling the error coefficients require that we sample from a monogenic ring. In short, the process of sampling the error terms amounts to choosing a "small" vector in $\mathcal{O}_K$ and then reducing it modulo a prime q. The PLWE problem is not typically defined for non-monogenic number fields. However, for $f(x) \in \mathbb{Z}[x]$ a monic irreducible polynomial of degree n, the Dedekind-Kummer Theorem tells us that while $P = \mathbb{Z}[x]/\langle f(x)\rangle$ is not isomorphic to $\mathcal{O}_K$ in most cases, $\mathcal{O}_K/(q) \cong \mathbb{F}_q[x]/\langle f(x)\rangle$ when q does not divide the index of P in $\mathcal{O}_K$.

Our work studies, first, the possibility of sampling "small" error vectors in the ring $\mathbb{F}_q[x]/\langle f(x)\rangle$ directly. This approach was not promising. Our second approach uses coset representatives. Suppose $[\mathcal{O}_K : \mathbb{Z}[\alpha]] = m > 1$ for a root $\alpha \in K$ of a minimal polynomial f. Let $\beta_1, \ldots, \beta_m$ be coset representatives of $\mathcal{O}_K/\mathbb{Z}[\alpha]$. Suppose we sample $p(\alpha) \in P$ according to PLWE and sample β_i uniformly at random. Output $p(\alpha) + \beta_i \in \mathcal{O}_K$. Calculating the statistical distance between this PLWE sampling algorithm and the RLWE sampling algorithm, we hope to expand the PLWE sampling to a wider class of rings of integers.

To my teachers, past and present

Acknowledgements

A huge thank you to my family for their continued support throughout my life and entire educational journey. My parents encourage me to always try new things and to have a curious mind. Thank you for your patience, kindness, and support. My brothers' positive energies always lift my spirits and I am grateful for all the memories and conversations we share. I also want to thank my soon-to-be husband. Thank you for bringing laughter into each and every day. You are my best friend and I am so grateful for all that you do for me. I look forward to all our future adventures.

A big thank you to all my teachers, both past and present. With such great role models and mentors, I am excited to follow in their foot steps as an educator. Specifically, I want to thank my PhD advisor, Dr Christelle Vincent, for her valuable guidance throughout my graduate school experience. She welcomed me into this project and provided me with a handful of opportunities to further my development as a professional. Her advice at each stage in my journey has been indispensable and I will put it to great use for years to come.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cryptography is the study of secure communication in the presence of an adverse third party. Public key cryptography is a system that uses corresponding public and private keys. A public key is used to encrypt a message, while a private (secret) key is needed to then decrypt the message. For example, if Person A wants to send a message (plaintext) to Person B, Person A would use Person B's public key to encrypt it. Then Person B would use their private key to decrypt the encrypted message (ciphertext) back to plaintext. The security of public key cryptography is based on "hard problems." When we say a problem is "hard," we mean that the number of steps or resources needed to solve the problem grows fast when the size of the input grows, causing our algorithm to be incredibly slow. Cryptographers consider problems that take even the fastest known algorithms an exponential, or at least sub-exponential, amount of time or space to solve.

RSA is one of the earliest public key cryptosystems, and it is based on the hardness of factoring. While it is "easy" to multiply two prime numbers, such as 1489×701, it is "hard" to do the reverse, which in this case would be to express 1043789 as its prime factors. However, with the possibility of quantum computers becoming prevalent, "hard" problems could be solved in a reasonable amount of time, threatening our current cryptosystems. Shor's Algorithm [Sho97] can find the prime factors of an

integer using a quantum computer in polynomial time. The publication of Shor's Algorithm sparked the study of post-quantum cryptography. Not to be confused with quantum cryptography, post-quantum cryptography means that encryption and decryption take place on a classical computer but is safe against attacks performed on both a classical and quantum computer. (In quantum cryptography, encryption and decryption take place on a quantum computer.)

As a result, cryptographers are studying problems that are still believed to be "hard" for both classical and quantum computers. Over the last 20 years, lattice-based cryptosystems have started attracting interest due to their level of security against polynomial-time attacks. In 2016, the National Institute of Standards and Technology (NIST) opened a competition to test and standardize post-quantum algorithms. As of August 2023, there are three schemes which are candidates for standardization; two of these schemes are lattice-based while the third is a hash-based cryptosystem. The lattice-based cryptosystems are based on the hardness of a variation of the Ring Learning with Errors (RLWE) problem.

In 2005, Regev introduced the Learning with Errors (LWE) problems: the search and decision problems [Reg09]. The drawback on the LWE problems is that the public and private key sizes that are needed are quite large. A few years later, in 2010, [LPR10] introduced the Ring Learning with Errors (RLWE) search and decision problems. Let K be a number field, O_K its ring of integers, and q a prime unramified in K. Given a secret element $s \in \mathcal{O}_K/q\mathcal{O}_K$, an **RLWE sample** is of the form $(a, b = a \cdot s + e) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K/q\mathcal{O}_K$, where a is chosen uniformly at random and e is a "small" error term. Here $\cdot$ denotes multiplication.

The **search problem** is to find the value of $s \in \mathcal{O}_K/q\mathcal{O}_K$ given independent RLWE samples. The **decision problem** is to determine if the given samples $(a, b) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K/q\mathcal{O}_K$ are RLWE samples or uniformly random. These problems are sometimes referred to as the **non-dual RLWE problems**.

Since the introduction of RLWE, other variations of the problems have been developed. The **dual RLWE problems** utilize the dual of the ring of integers,

$$\mathcal{O}_K^\vee = \{x \in K : \mathrm{Tr}(x\mathcal{O}_K) \subseteq \mathbb{Z}\},$$

and the **Polynomial Ring Learning with Errors (PLWE) problems** draw samples from a polynomial ring. As we will see, the main difference between the variations of the problems is the way in which the error terms are sampled. In particular, the PLWE problems are not typically defined for non-monogenic number fields, which is an avenue we explore in Chapter 8.

Attacks which attempt to solve the RLWE problems have been formulated to exploit the vulnerabilities of underlying number fields (mostly 2-power cyclotomic fields). In [CLS17a] and [CLS17b], the authors describe a Chi-Squared Attack which attempts to solve the non-dual RLWE decision problem. Theorem 3 and Corollary 1 in [CLS17b] state bounds on the security parameters to ensure a Gaussian distribution is indistinguishable from a uniform distribution for 2-power cyclotomic number fields. At the same time, Theorem 5.2 in [Pei16b] states bounds on the security parameters which ensure safety for dual RLWE for arbitrary number fields.

One of our contributions to this area of research is on the security parameter bounds that predict when instances are safe against attacks. We converted the parameter from [Pei16b] for the dual RLWE problem to an equivalent parameter for the non-dual RLWE problem. Using this converted parameter, along with the conditions from [CLS17b], we present a conjecture that there is a trade-off between the norm of the prime and the standard deviation of the Gaussian distribution. In doing this, we can predict that a tighter bound can be found in comparison to the bound in [Pei16b].

Our other contribution concerns the Polynomial Ring Learning with Errors

problems. Let $f(x) \in \mathbb{Z}[x]$ be a monic irreducible polynomial of degree n and q be a prime such that $f(x) \pmod q$ factors completely into linear terms. Consider the polynomial ring $P = \mathbb{Z}[x]/\langle f(x)\rangle$, then $P/qP = \mathbb{F}_q[x]/\langle f(x)\rangle$. Given a secret polynomial $s(x) \in P/qP$, **PLWE samples** are of the form $(a(x), b(x)) \in P/qP \times P/qP$ where $a(x)$ is drawn uniformly at random and $b(x) = a(x)s(x) + e(x)$ for a "small" error term $e(x) \in P/qP$. To draw a small error, we draw the coefficients of $\tilde{e}(x) \in P$ to be "small" in the power basis. When the number field K defined by $f(x)$ is monogenic, the polynomial ring P is isomorphic to the ring of integers $\mathcal{O}_K$ of K. So the process of sampling the error terms amounts to choosing a "small" vector in $\mathcal{O}_K$ and then reducing it modulo q. The PLWE problem is not typically defined for non-monogenic number fields. We examine a couple of possible strategies to allow for non-monogenic number fields to be considered.

1.1 Organization of Thesis

In Chapter 2, we define the ring of integers $\mathcal{O}_K$ of a number field K and prove that $\mathcal{O}_K$ is a Dedekind domain. We also discuss the various canonical embeddings and use them to define a lattice. Lastly, we introduce Gaussian distributions for sampling lattice points.

We continue to discuss Gaussian distributions in the following chapter, Chapter 3. Here we define a subgaussian and a variety of sampling algorithms to sample from a Gaussian distribution over the integers as well as to sample over a lattice. We also discuss how we can discretize a continuous Gaussian distribution, which will be useful when we compare security parameter bounds from both a continuous and discrete Gaussian distribution.

In Chapter 4, we introduce the Learning with Errors (LWE) problems from [Reg09] as an introduction into the Ring Learning with Errors problems, starting in Chapter

5. We also discuss the variations of this problem (non-dual RLWE, dual RLWE, and Polynomial Ring Learning with Errors (PLWE)).

In Chapter 6, we mention a few algorithms that attempt to solve the LWE and RLWE problems. Here we introduce the Chi-Squared Attack from [CLS17a, CLS17b], which will be of main interest for the next chapter. (This will be the attack that we use to test the parameter bounds that we find in Chapter 7.)

In Chapter 7, we consider the known security bounds mentioned in [CLS17b, Pei16b] for the non-dual RLWE and dual RLWE problems, respectively. Using the relationships between the variations of the problems from Chapter 5, we convert the security parameter from [Pei16b] to get an equivalent bound for the non-dual RLWE problem (equivalently the PLWE problem when K is monogenic). Using the Chi-Squared Attack from [CLS17a, CLS17b] and SageMath code from [ELOS15], we test the security bounds to show that the results in [Pei16b] are not sharp.

In Chapter 8, we consider a variety of ways to generalize the Polynomial Ring Learning with Errors (PLWE) sampling for non-monogenic rings of integers. To start, we explore the Dedekind-Kummer Theorem as our inspiration for these new sampling algorithms. Our first attempt considers reducing modulo a prime q first and then sampling directly from $\mathcal{O}_K/q\mathcal{O}_K$. Due to some vulnerabilities in this sampling algorithm, this approach is not promising. Our next attempt uses coset representatives. Upon calculating the statistical distance between the RLWE sampling distribution on $\mathcal{O}_K$ and our proposed PLWE sampling distribution on $\mathcal{O}_K$, we show that our sampling algorithm is an interesting way to expand the PLWE sampling to non-monogenic number fields that should warrant further study.

Future work to still be explored will be mentioned in Chapter 9.

Chapter 2

Background

2.1 Preliminaries

Definition 2.1.1. A **number field** K is a finite field extension of $\mathbb{Q}$.

We can construct a number field as

$$K = \mathbb{Q}(\alpha) \cong \mathbb{Q}[x]/(f(x))$$

where α is a root of some monic irreducible polynomial $f(x) \in \mathbb{Q}[x]$. We call $f(x)$ the minimal polynomial of α over $\mathbb{Q}$.

When viewing K as a vector space over $\mathbb{Q}$, we can find a basis for K.

Definition 2.1.2. A set B is a **basis** for a vector space V if every element of V can be written uniquely as a linear combination of elements in B.

A vector space can have multiple bases, and each of them has a dual. Before defining the dual of a basis, we first define the Kronecker delta function.

Definition 2.1.3. The **Kronecker delta function** is a mapping of two variables,

which outputs 1 when they agree and 0 otherwise:

$$\delta_{jk} = \begin{cases} 1 & j = k, \\ 0 & j \neq k. \end{cases}$$

Definition 2.1.4. If $B = \{b_1, \ldots, b_n\}$ is a basis of $\mathbb{C}^n$, its **dual basis** $B^\vee = \{d_1, \ldots, d_n\}$ is characterized by $\langle b_j, d_k \rangle = \delta_{jk}$, where δ_{jk} is the Kronecker delta function and $\langle \cdot, \cdot \rangle$ is the usual complex dot product.

2.2 Ring of Integers

Considering the minimal polynomial of $\alpha \in K$ over $\mathbb{Q}$, we will define the set of all α such that its minimal polynomial has integer coefficients. (Recall that we require the minimal polynomial to be monic.)

Definition 2.2.1. For a number field K, the set of algebraic integers, elements whose minimal polynomial over $\mathbb{Q}$ has integer coefficients, in K is denoted $\mathcal{O}_K$, and called the **ring of integers of** K.

Example 2.2.2.

i. For the field $K = \mathbb{Q}$, its ring of integers is $\mathcal{O}_\mathbb{Q} = \mathbb{Z}$, which is the smallest possible ring of integers.

ii. The ring of Gaussian integers $\mathbb{Z}[i]$ is the ring of integers for the Gaussian rationals $\mathbb{Q}(i)$.

iii. Let $K = \mathbb{Q}(\sqrt{5}) = \{a + b\sqrt{5} : a, b \in \mathbb{Q}\}$. To compute $\mathcal{O}_K$, let $x = a + b\sqrt{5} \in K$. Then $x^2 = a^2 + 5b^2 + 2ab\sqrt{5}$ and $2ax = 2a^2 + 2ab\sqrt{5}$. Then

$$x^2 - 2ax = a^2 + 5b^2 + 2ab\sqrt{5} - 2a^2 - 2ab\sqrt{5} = 5b^2 - a^2.$$

Moving terms to one side, we get the form of the minimal polynomial of elements of the form $a + b\sqrt{5}$ as

$$x^2 - 2ax + a^2 - 5b^2 = 0.$$

The coefficients are integers when $2a \in \mathbb{Z}$ and $a^2 - 5b^2 \in \mathbb{Z}$. So either $a \in \mathbb{Z} \implies b \in \mathbb{Z}$ or $a \in \frac{1}{2}\mathbb{Z} \implies b \in \frac{1}{2}\mathbb{Z}$. Thus

$$\mathcal{O}_K = \mathbb{Z}\left[\frac{1+\sqrt{5}}{2}\right] = \left\{c + d\left(\frac{1+\sqrt{5}}{2}\right) : c, d \in \mathbb{Z}\right\}.$$

Notice that in each of these examples, $\mathbb{Z}$ is a subset of the ring of integers. This is true for all $\mathcal{O}_K$ for any number field K, that $\mathbb{Z} \subseteq \mathcal{O}_K$.

Our next goal is to show that the ring of integers of a number field is a Dedekind domain. We will start with some useful definitions.

Definition 2.2.3. A commutative ring R with multiplicative identity 1 is called an **integral domain** if R has no zero divisors.

Definition 2.2.4. An element $r \in R$ integral over S if there exists a monic polynomial $p(x) \in S[x]$ such that $p(r) = 0$.

Definition 2.2.5. The **integral closure of S in R** is the ring of all $r \in R$ which are integral over S.

Example 2.2.6. If $S = \mathbb{Z}$ and $R = K$ for some number field K, then the integral closure of S in R is $\mathcal{O}_K$ by definition of the ring of integers of K.

Definition 2.2.7. An ideal I is **prime** if whenever $ab \in I$, then $a \in I$ or $b \in I$.

Definition 2.2.8. A commutative ring R with multiplicative identity 1 is called **Noetherian** if every ideal of R is finitely generated.

Definition 2.2.9. An integral domain R is a **Dedekind domain** if all three conditions hold:

i. R is integrally closed in its field of fractions,

ii. every non-zero prime ideal is a maximal ideal,

iii. R is Noetherian.

The ring of integers $\mathcal{O}_K$ of a number field K is a Dedekind domain, which is a fact we will recall in Chapter 6, when discussing the Chi-Squared Attack.

Theorem 2.2.10. *Let K be a number field, $\mathcal{O}_K$ its ring of integers. Then $\mathcal{O}_K$ is a Dedekind domain.*

Proof. i. Suppose $\alpha \in K$ is integral over $\mathcal{O}_K$. Then α is a root of a monic polynomial $f(x) = a_0 + \ldots + a_{n-1}x^{n-1} + x^n$, $a_i \in \mathcal{O}_K$. Since $\mathbb{Z}[a_0, \ldots, a_{n-1}]$ is finitely generated, then $\mathbb{Z}[a_0, \ldots, a_{n-1}, \alpha]$ is finitely generated. (Note: α^m for $m \geq n$, we can rewrite it using smaller powers of α.) This implies that α is integral over $\mathbb{Z}$. Thus $\alpha \in K \cap \overline{\mathbb{Z}} = \mathcal{O}_K$, and $\mathcal{O}_K$ is integrally closed.

ii. Let I be a non-zero prime ideal of $\mathcal{O}_K$. Then $I \cap \mathbb{Z}$ is a non-zero prime ideal. Suppose $I \cap \mathbb{Z} = (p)$ for some prime p. If we can show that $\mathcal{O}_K/I$ is a field, we win. The ring $\mathcal{O}_K/I$ is finite over $\mathbb{Z}/(p) = \mathbb{Z}/p\mathbb{Z} = \mathbb{F}_p$. We also have that $\mathcal{O}_K/I$ is an integral domain. A finite integral domain is a field, and so $\mathcal{O}_K/I$ is a field. Thus I is a maximal ideal.

iii. Let $\{b_1, \ldots, b_n\}$ be a basis of K over $\mathbb{Q}$. Then there exists a nonzero integer d such $db_i \in \mathcal{O}_K$ for all i, and $\{db_1, \ldots, db_n\}$ is also a basis for K as a vector space over $\mathbb{Q}$. Therefore, without loss of generality, we may assume $b_i \in \mathcal{O}_K$ for all i. Let $\{b_1^\vee, \ldots, b_n^\vee\}$ be its dual basis. We want to show that

$$b_1\mathbb{Z} + b_2\mathbb{Z} + \ldots + b_n\mathbb{Z} \subset \mathcal{O}_K \subset b_1^\vee\mathbb{Z} + b_2^\vee\mathbb{Z} + \ldots + b_n^\vee\mathbb{Z}.$$

Since $b_i \in \mathcal{O}_K$ for all i and $\mathbb{Z} \subset \mathcal{O}_K$, we have the first inclusion. To show the second inclusion, let α be an element of $\mathcal{O}_K$. Then α is a unique linear

combination of the $b_j^\vee$,

$$\alpha = \sum_j c_j b_j^\vee$$

for $c_j \in K$. For each i, b_i and α are in $\mathcal{O}_K$, then their product $b_i\alpha$ is also in $\mathcal{O}_K$. The trace of an element of $\mathcal{O}_K$ is an integer, and therefore $Tr(b_i\alpha) \in \mathbb{Z}$, but we have

$$\begin{aligned} Tr(b_i\alpha) &= Tr(b_i \sum_j c_j b_j^\vee) \\ &= \sum_j c_j Tr(b_i b_j^\vee) \\ &= \sum_j c_j \delta_{ij} \\ &= c_i. \end{aligned}$$

Therefore $\alpha \in b_1^\vee\mathbb{Z} + b_2^\vee\mathbb{Z} + \ldots + b_n^\vee\mathbb{Z}$, and so $\mathcal{O}_K$ is a finitely generated $\mathbb{Z}$-module. Since $\mathbb{Z}$ is Noetherian, any $\mathbb{Z}$-submodule of $\mathcal{O}_K$ must also be a finitely generated $\mathbb{Z}$-module, and thus $\mathcal{O}_K$ is Noetherian.

□

Another fact about the ring of integers $\mathcal{O}_K$ for a number field K is that it is a fractional ideal.

Definition 2.2.11. Let R be an integral domain, and K its fraction field. A **fractional ideal** of R is an R-submodule I of K such that $rI \subseteq R$ for some $r \in R$.

2.3 Canonical Embedding

The canonical embedding of a fractional ideal describes a lattice in $\mathbb{R}^n$. To define this, we first talk about the embeddings of a number field K.

Let K be a number field of degree n. Then K must have exactly n **embeddings**

into the complex numbers, denoted $\sigma_i : K \longrightarrow \mathbb{C}$. These embeddings fix the prime field $\mathbb{Q}$ and permute the roots of irreducible polynomials. An embedding whose image is contained in $\mathbb{R}$ is called a **real embedding**; the rest are called **complex embeddings**. Let s_1 denote the number of real embeddings and s_2 denote the number of pairs of complex embeddings of K. Then $n = s_1 + 2s_2$.

Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be the real and complex embeddings of K. We order the embeddings so that $\sigma_1, \ldots, \sigma_{s_1}$ are the real embeddings and $\sigma_{s_1+1}, \ldots, \sigma_{s_1+2s_2}$ are the complex embeddings where $\sigma_{s_1+s_2+k} = \overline{\sigma_{s_1+k}}$ for $k = 1, 2, \ldots, s_2$. Let

$$H = \{(x_1, \ldots, x_n) \in \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2} : x_{s_1+s_2+k} = \overline{x_{s_1+k}} \; \forall k = 1, 2, \ldots, s_2\}.$$

Then we define the canonical embedding of K in the following way.

Definition 2.3.1. The **canonical embedding** $\theta : K \longrightarrow H \subseteq \mathbb{R}^{s_1} \times \mathbb{C}^{2s_2}$ is defined as

$$\theta(x) = (\sigma_1(x), \ldots, \sigma_n(x)).$$

Some authors ([CLS17b, CLS17a, ELOS15, EHL14]) define similar canonical embeddings to describe the lattice of a fractional ideal to be used for the RLWE problems, which we will refer to as the alternative canonical embedding and the adjusted canonical embedding.

Definition 2.3.2. The **alternative canonical embedding** $\theta_{ALT} : K \to \mathbb{R}^n$ is

defined as

$$\theta_{ALT}(\alpha) = \begin{bmatrix} \sigma_1(\alpha) \\ \dots \\ \sigma_{s_1}(\alpha) \\ Re(\sigma_{s_1+1}(\alpha)) \\ \dots \\ Re(\sigma_{s_1+s_2}(\alpha)) \\ Im(\sigma_{s_1+1}(\alpha)) \\ \dots \\ Im(\sigma_{s_1+s_2}(\alpha)) \end{bmatrix}.$$

Definition 2.3.3. The **adjusted canonical embedding** $\theta_{ADJ} : K \longrightarrow \mathbb{C}^n$ is defined as

$$\theta_{ADJ}(\alpha) = \begin{bmatrix} \sigma_1(\alpha) \\ \dots \\ \sigma_{s_1}(\alpha) \\ \sqrt{2}Re(\sigma_{s_1+1}(\alpha)) \\ \sqrt{2}Im(\sigma_{s_1+1}(\alpha)) \\ \dots \\ \sqrt{2}Re(\sigma_{s_1+s_2}(\alpha)) \\ \sqrt{2}Im(\sigma_{s_1+s_2}(\alpha)) \end{bmatrix}.$$

We note that under a canonical embedding, we can endow K with a norm by identifying an element with its canonical embedding and using the Euclidean norm on $\mathbb{C}^n$

$$||x|| := ||\theta(x)||.$$

2.4 Lattices

Now, we use the canonical embedding of a fractional ideal $I \subset K$, $\theta(I)$, to define a lattice, Λ_I in $H \cong \mathbb{R}^n$ where n is the degree of K.

Definition 2.4.1. A **lattice** in $\mathbb{R}^n$ is the $\mathbb{Z}$-span of a $\mathbb{R}$-basis of $\mathbb{R}^n$.

Under the canonical embedding, $\Lambda_{\mathcal{O}_K} := \theta(\mathcal{O}_K)$ is a lattice in $\mathbb{R}^n$.

Example 2.4.2. Let $K = \mathbb{Q}(i)$ and $\mathcal{O}_K = \mathbb{Z}[i]$. Notice that $[K : \mathbb{Q}] = 2$, so we have the pair of complex embeddings

$$\sigma_1(i) = i \qquad \sigma_2(i) = -i.$$

A basis of $\mathcal{O}_K$ is $1, i$. Evaluating the alternative canonical embedding at these basis elements of $\mathcal{O}_K$ will give the basis of the lattice $\theta_{ALT}(\mathcal{O}_K)$. So

$$\theta_{ALT}(1) = (Re(\sigma_1(1)), Im(\sigma_1(1))) = (1, 0)$$

and

$$\theta_{ALT}(i) = (Re(\sigma_1(i)), Im(\sigma_1(i))) = (0, 1)$$

gives the basis of the lattice $\theta_{ALT}(\mathcal{O}_K)$ as $(1, 0)$ and $(0, 1)$. Figure 2.1 shows $\theta_{ALT}(\mathcal{O}_K)$ in $\mathbb{R}^2$.

Now, we do another example involving the dual basis. This example illustrates that, in the cases that interest us, the dual lattice $\Lambda_{B^\vee}$ is much denser than the lattice Λ_B.

Example 2.4.3. Consider the lattice Λ_B in $\mathbb{R}^2$ generated by the basis $B = \{(1, 1), (0, 2)\}$, shown in Figure 2.2.

We will now find the dual basis of B, $B^\vee$. Given the definition of a dual basis, we need to see when $\langle b_j, d_k \rangle$ equals 1 for $j = k$, and 0 for $j \neq k$. So if $B^\vee = \{d_1, d_2\}$

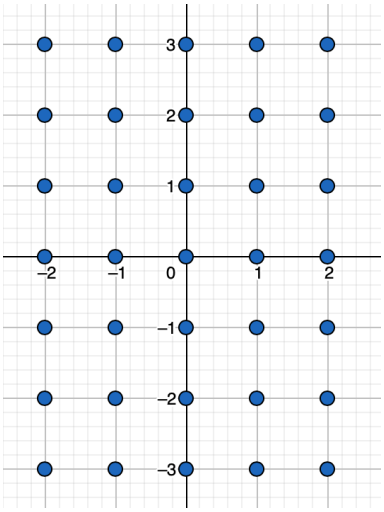

Figure 2.1: $\theta_{ALT}(\mathbb{Z}[i])$.

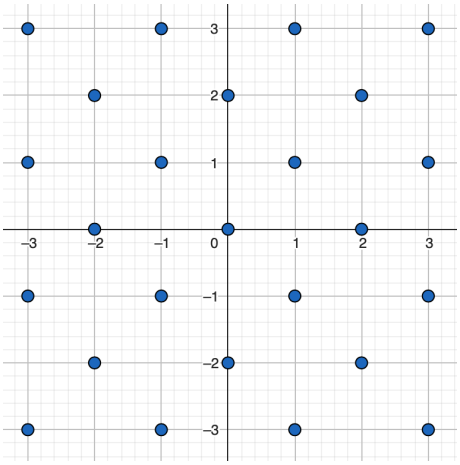

Figure 2.2: The lattice Λ_B *with basis* B.

with $d_1 = (a_1, a_2)$ and $d_2 = (a_3, a_4)$, we get the following equations

$$\langle b_1, d_1 \rangle = 1a_1 + 1a_2 = 1$$

$$\langle b_2, d_1 \rangle = 0a_1 + 2a_2 = 0$$

$$\langle b_1, d_2 \rangle = 1a_3 + 1a_4 = 0$$

$$\langle b_2, d_2 \rangle = 0a_3 + 2a_4 = 1$$

Solving for the a_i's, we get that $d_1 = (1, 0)$ and $d_2 = \left(-\frac{1}{2}, \frac{1}{2}\right)$. Thus the dual basis of the basis $B = \{(1, 1), (0, 2)\}$ is $B^\vee = \left\{(1, 0), \left(-\frac{1}{2}, \frac{1}{2}\right)\right\}$. The lattice $\Lambda_{B^\vee}$ is shown

in Figure 2.3.

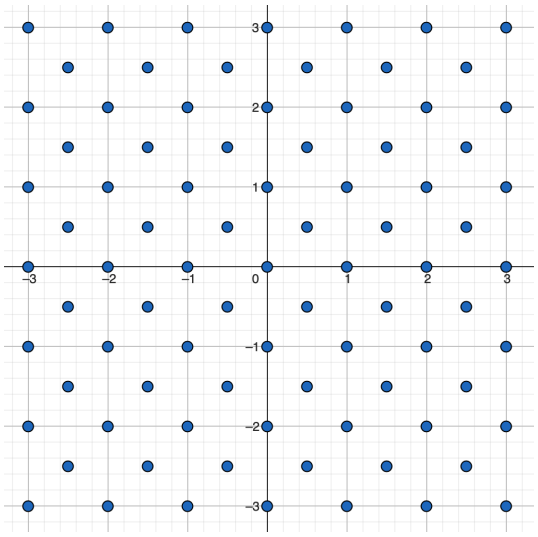

Figure 2.3: The lattice $\Lambda_{B^\vee}$ corresponding to the dual basis $B^\vee$.

The lattice generated by the dual basis is, in fact, the dual lattice of the lattice generated by B, i.e $\Lambda_{B^\vee} = \Lambda_B^\vee$. We will recall this example when we compare the non-dual and dual Ring Learning with Errors problems in Chapter 5.

2.5 Gaussian Distributions

To sample lattice points, we use a Gaussian distribution.

Definition 2.5.1. For $\sigma > 0$, the **continuous Gaussian function** $\rho_\sigma : \mathbb{R}^n \to [0, 1)$ is

$$\rho_\sigma(x) = e^{-||x||^2/2\sigma^2}$$

where $||\cdot||$ is the Euclidean norm.

Using the Gaussian function, we define the discrete Gaussian distributions over a lattice.

Definition 2.5.2. Given a lattice $\Lambda \subseteq \mathbb{R}^n$ and real number $\sigma > 0$, the **discrete Gaussian distribution** on Λ with standard deviation σ is given by the probability

distribution function

$$D_{\Lambda,\sigma}(x) = \frac{\rho_\sigma(x)}{\sum_{y\in\Lambda}\rho_\sigma(y)}$$

for all $x \in \Lambda$.

Definition 2.5.3. The width r of a Gaussian distribution is defined to be $r = \sqrt{2\pi}\sigma$.

2.6 Subgaussians

Next, we introduce subgaussians, distributions which are dominated by centered Gaussian distributions and have strong tail decay.

Definition 2.6.1. A **(standard normal) subgaussian random variable** X is a random variable that satisfies any one of the following three properties:

- $\mathbb{P}\{|X| > t\} = \frac{2}{\sqrt{2\pi}}\int_t^\infty e^{-x^2/2}dx \leq 2e^{-t^2/2}$ for $t \geq 1$,
- $(\mathbb{E}[|X|^p])^{1/p} = \sqrt{2}\left[\frac{\Gamma((1+p)/2)}{\Gamma(1/2)}\right]^{1/p} = O(\sqrt{p})$ for $p \geq 1$,
- $\mathbb{E}[\exp(tX)] = e^{t^2/2}$ for $t \in \mathbb{R}$.

Note: These three properties are equivalent (see [Ver12] Lemma 5.5 for the proof).

Definition 2.6.2. Let $t \in \mathbb{R}$. We call $\mathbb{E}[\exp(tX)]$ the **moment-generating function** of X. When X, a subgaussian random variable, has variance $\sigma^2 \neq 1$, the moment-generating function is then

$$\mathbb{E}[\exp(tX)] = \exp(\sigma^2 t^2/2).$$

Definition 2.6.3 ([LPR13])**.** Let $\delta > 0$. A random variable X is said to be a δ**-subgaussian** with standard deviation $\sigma > 0$ if for all $t \in \mathbb{R}$, the (scaled)

moment-generating function of X satisfies

$$\mathbb{E}[\exp(2\pi tX)] \leq \exp(\delta) \cdot \exp(\pi(\sqrt{2\pi}\sigma)^2 t^2).$$

The significance of this distribution is that in the next chapter, when we discretize a continuous Gaussian distribution, the result will be a subgaussian.

Chapter 3

Sampling Algorithms

Before we introduce the Learning with Errors problems, we will continue to discuss Gaussian distributions and ways we can sample from Gaussian distributions over the integers as well as over a lattice. We end this chapter with some insight as to how to discretize a continuous Gaussian on $\mathbb{R}^n$ to a Gaussian on a given translate of a lattice in $\mathbb{R}^n$ using the coordinate-wise randomized rounding sampling technique. The ideas in this chapter will be recalled time and time again throughout the rest of this work. In Chapter 5, we highlight that the variations of the RLWE problems differ mainly in the way that the error terms are sampled. In Chapter 7, we compare the security bounds from a continuous Gaussian distribution with a discrete distribution over a lattice. Finally, in Chapter 8, we explore a new means of sampling inspired by Polynomial Ring Learning with Errors (PLWE) and compare the probability of sampling an element under these sampling algorithms and our proposed scheme.

3.1 Sampling Over The Integers

We begin by exploring ways in which we can sample from a discrete Gaussian distribution over the integers. There are two ways to do this:

- rejection sampling,

- the inversion method.

3.1.1 Rejection Sampling

[GPV08] describes this sampling algorithm as a way of sampling integers from a discrete Gaussian distribution.

Suppose that one wishes to draw an integer from a discrete Gaussian distribution with center c and width $r = \sqrt{2\pi}\sigma$, implicitly choosing n. Let $f(n)$ be a fixed function such that $f(n) \geq \omega\sqrt{\log n}$. (Here ω denotes an estimate of the order of growth.) Then choose some integer $x \in \mathbb{Z} \cap [c - rf(n), c + rf(n)]$ uniformly at random, and with probability $\rho_\sigma(x - c)$, output x. Otherwise, repeat the process of choosing an integer $x \in \mathbb{Z} \cap [c - rf(n), c + rf(n)]$ uniformly at random until one is chosen. (Recall that ρ_σ is the continuous Gaussian function.)

Example 3.1.1. Let $f(n) = \log n$, $(c, r) = (1, 3)$, and $n = 3$. Then

$$\begin{aligned} x &\in \mathbb{Z} \cap [c - rf(n), c + rf(n)] \\ &= \mathbb{Z} \cap [1 - 3\log 3, 1 + 3\log 3] \\ &= \mathbb{Z} \cap [-0.43, 2.43] \\ &= \{0, 1, 2\}. \end{aligned}$$

Suppose we chose $x = 0$ uniformly at random. Then

$$\begin{aligned} \rho_\sigma(x - c) = \rho_\sigma(-1) &= \exp\left(-\frac{||-1||^2}{2\left(\frac{3}{\sqrt{2\pi}}\right)^2}\right) \\ &= \exp\left(-\frac{\pi\|-1\|^2}{3^2}\right) \\ &= 0.71. \end{aligned}$$

So, 71% of the time, we would output $x = 0$, if chosen.

Suppose we chose $x = 1$ uniformly at random. Then

$$\rho_\sigma(0) = e^0 = 1$$

and so $x = 1$ is outputted 100% of the time that it is chosen.

Similarly to when $x = 0$, if we chose $x = 2$ uniformly at random, then it is outputted 71% of the time.

3.1.2 The Inversion Method

The inversion method sampling algorithm is the following:

1. Take samples $t \in [0, 1]$ uniformly at random.
2. Return the smallest number $x \in \mathbb{Z}$ such that $F(x) \geq t$ for the cumulative distribution function F of a random variable.

In other words, we are choosing a percentage uniformly at random and in step 2, returning the smallest number x in the domain such that $P(X \leq x) \geq t$.

Example 3.1.2. Suppose we have the discrete probability distribution shown in Table 3.1.

Table 3.1: Discrete Probability Distribution

x_i	P(X= x_i)
0	0.2
1	0.1
2	0.4
3	0.3

The **cumulative distribution function (cdf)** is given by

$$F(x_i) = P(X \leq x_i).$$

In other words, it is the sum all P(X= x_j) where $x_j \leq x_i$. Table 3.2 shows the cdf corresponding to the discrete probability distribution from Table 3.1.

Table 3.2: Cumulative Distribution Function

x_i	$F(x_i)$
0	0.2
1	0.3
2	0.7
3	1.0

To sample using the inversion method, we would choose $t \in [0, 1]$, say $t = 0.64$. Then we return the smallest number x_i such $F(x_i) \geq t$. Then we would return $x_i = 2$ since $F(2) = 0.7 \geq 0.64$ and $x_i = 2$ is the smallest number for which that is true.

3.2 Sampling from a Lattice

Given a lattice Λ, a basis $B = \{b_1, \ldots, b_k\}$ for this lattice, and points $x, c \in H$, we can sample a short vector $f \in \Lambda + (c - x)$ using the following algorithms:

- rejection sampling,
- coordinate-wise randomized rounding.

3.2.1 Rejection Sampling

[GPV08] has an algorithm to sample from a discrete Gaussian $D_{\Lambda,\sigma,c}$ centered at c with standard deviation σ which follows closely to rejection sampling over the integers. The algorithm is as follows:

1. Define $\pi_k : \mathbb{R}^n \to \text{span}(b_1, \ldots, b_k)$ for $k \in \{1, \ldots, n\}$ and define $\tilde{b_k} = \pi_k(b_k)$. Let $v_n = 0$ and $c_n = c$. For $i = n, \ldots, 1$,
   (a) Let $c'_i = \langle c_i, \tilde{b_i} \rangle / \langle \tilde{b_i}, \tilde{b_i} \rangle \in \mathbb{R}$ and $s'_i = \sigma / \|\tilde{b_i}\| > 0$.

(b) Choose z_i from $D_{\mathbb{Z},s_i',c_i'}$.

(c) Let $c_{i-1} = c_i - z_i b_i$ and let $v_{i-1} = v_i + z_i b_i$.

2. Output v_0.

3.2.2 Coordinate-Wise Randomized Rounding

Using cosets, [LPR13] describes an algorithm called coordinate-wise randomized rounding in which we can sample a short vector from a lattice coset.

Given a coset $\Lambda + (c - x)$, we represent $c - x$ in the basis $B = \{b_1, \ldots, b_n\}$ as

$$c - x \equiv \sum_i a_i b_i \pmod{\Lambda}$$

for $a_i \in [0, 1)$. Then we will randomly and independently choose $f_i \in \{a_i - 1, a_i\}$ to have expectation zero, and output $f = \sum_i f_i b_i \in \Lambda + (c - x)$.

A note on expectation zero: Suppose the probability of choosing $f_i = a_i - 1$ is p_i, i.e. $\mathrm{P}(f_i = a_i - 1) = p_i$. Then $\mathrm{P}(f_i = a_i) = 1 - p_i$. The expected value of f_i is

$$\begin{aligned}\mathbb{E}[f_i] &= (a_i - 1)p_i + a_i(1 - p_i) \\ &= a_i p_i - p_i + a_i - a_i p_i \\ &= a_i - p_i.\end{aligned}$$

So if f_i is predicted to have expectation zero, then $a_i = p_i$. Thus the possible values of f_i closest to zero have the highest probability of being chosen.

Example 3.2.1. Let $\Lambda = \mathbb{Z}^2, B = \{(1, 0), (0, 1)\}, x = \left(\frac{5}{4}, -\frac{1}{3}\right)$, and $c = \left(\frac{1}{2}, \frac{5}{3}\right)$. Then the lattice coset is $\Lambda + \left(-\frac{3}{4}, 2\right)$. We want to represent $\left(-\frac{3}{4}, 2\right)$ as $\sum_i a_i b_i \pmod{\Lambda}$:

$$\begin{aligned}\left(-\frac{3}{4}, 2\right) &\equiv a_1(1,0) + a_2(0,1) \pmod{\Lambda} \\ &\equiv (a_1, 0) + (0, a_2) \pmod{\Lambda} \\ &\equiv (a_1, a_2) \pmod{\Lambda} \\ &\equiv \left(\frac{1}{4}, 0\right) \pmod{\Lambda}\end{aligned}$$

And so, $a_1 = \frac{1}{4}$ and $a_2 = 0$. Then $f_1 \in \left\{-\frac{3}{4}, \frac{1}{4}\right\}$ and $f_2 \in \{-1, 0\}$. Taking into account that f_i has expectation zero, then $f_2 = 0$ with 100% probability, so there are 2 possibilities for f. We sample

$$f = \left(-\frac{3}{4}, 0\right) \text{ with probability } 25\%$$

and

$$f = \left(\frac{1}{4}, 0\right) \text{ with probability } 75\%.$$

Another algorithm, the deterministic approach which is also mentioned in [LPR13], follows closely. For this approach, $a_i \in [-\frac{1}{2}, \frac{1}{2})$ instead and we output $f = \sum_i a_i b_i$.

3.3 Discretizing to a Lattice Point

These sampling algorithms can be used to discretize a sample drawn from a continuous distribution to a lattice point drawn from a discrete Gaussian over some lattice in $\mathbb{R}^n$. With the coordinate-wise randomized rounding algorithm as motivation, [LPR13] describes how we can sample a short vector using the randomized rounding scheme to output a lattice point.

Given a lattice Λ, a basis B, and points $x, c \in H$, we can discretize a vector $x \in H$

drawn from a continuous Gaussian distribution to a point $y \in \Lambda + c$ where $\|x - y\|$ is small by sampling a short vector $f \in \Lambda + (c - x)$, and outputting the lattice point $y = x + f$.

Example 3.3.1. Continuing from Example 3.2.1, we sample $y = x + f \in \Lambda + c$ as either $y_1 = \left(\frac{1}{2}, -\frac{1}{3}\right)$ (with probability 25%) or $y_2 = \left(\frac{3}{2}, -\frac{1}{3}\right)$ (with probability 75%). Notice the lattice point y_2, which is sampled with higher probability, is closer to $x = \left(\frac{5}{4}, -\frac{1}{3}\right)$, meaning $\|x - y_2\| \leq \|x - y_1\|$.

To further support our calculations in Chapter 7, we draw attention to some facts from [LPR13]. It is first noted that f is a 0-subgaussian with standard deviation $\sigma = s_1(B) = \frac{\sqrt{2}^{\ell-2}}{2^{\ell}-1}$ for $K = \mathbb{Q}(\zeta_{2^\ell})$ where s_1 is the spectral norm. [LPR13] also states that a Gaussian distribution is itself a subgaussian and the sum of subgaussian distributions is also a subgaussian distribution. So if y is a $(\sum \delta_i)$ - subgaussian, then it has width $(\sum r_i^2)^{1/2}$ where r_i is the width of the δ_i-subgaussian. Hence, since x and f are subgaussians, y is a subgaussian. We will utilize this fact when converting the security parameter bounds between variations of the RLWE problems in Chapter 7.

Chapter 4

Learning with Errors

The Learning with Errors (LWE) problems were first introduced by Regev in 2005 [Reg09]. We will explore this problem as an introduction into the Ring Learning with Errors problems, which are the focus of the next chapter.

4.1 The Problems

The search and decision LWE problems both require samples of a special form.

Definition 4.1.1. Let q be a prime integer and n a positive integer. Given $s \in \mathbb{F}_q^n$ chosen uniformly at random and e a "small" element of $\mathbb{F}_q$, an **LWE sample** $(a, b) \in \mathbb{F}_q^n \times \mathbb{F}_q$ is formed such that a is chosen uniformly at random from $\mathbb{F}_q^n$ and b is computed as $b = a \cdot s + e$, where $\cdot$ is the dot product. We define the notion of "small" in Section 4.2.2.

Definition 4.1.2. The **LWE search problem** takes m LWE samples (a_i, b_i) and attempts to find the secret $s \in \mathbb{F}_q^n$.

Definition 4.1.3. The **LWE decision problem** takes m samples $(a_i, b_i) \in \mathbb{F}_q^n \times \mathbb{F}_q$ and determines whether they are LWE samples or uniformly random samples.

4.2 Error Term

For each LWE sample, a "small" element of $\mathbb{F}_q$ is drawn. This "small" element is referred to as the error term because it adds some noise to the dot product $a \cdot s$.

4.2.1 Why is noise needed?

Noise is needed to ensure that one cannot solve for s by simply using Gaussian elimination. Notice that if $e_i = 0$ for all i, then the LWE samples are $(a_i, b_i = a_i \cdot s)$. So we would have m many equations for the b_i's of the form:

$$a_{i1}s_1 + a_{i2}s_2 + \ldots + a_{in}s_n = b_i$$

in n variables and so we can solve for $s = (s_1, s_2, \ldots, s_n)$.

The security of these problems come down to the bounds on the standard deviation on the Gaussian distribution from which the error terms are drawn:

- If the error terms are too "big" (standard deviation is very large), then they might wrap around when we reduce modulo q (this is an issue for decryption, which we do not cover in detail).
- If the standard deviation is too small, then the values e_i will be near zero with high probability. In that case, it becomes probable that Gaussian elimination can be used to solve for s successfully, as the error terms will not grow too fast during the algorithm.

Other vulnerable instances exist which make the LWE problems "easy" to solve. Another such case, shown in [Pei16b], is when the error term comes from a continuous Gaussian distribution in which e_i is almost always in an interval $z + [-\frac{1}{2}, \frac{1}{2})$ for some $z \in \mathbb{Z}$. In this case, we use integer rounding and subtract z to remove the error term

from all samples, which brings us back to the errorless case. We will discuss these ideas further in Chapter 6.

4.2.2 How is the error term drawn?

When the standard deviation σ of a Gaussian distribution is small, we have a high probability of picking a small number. Since we want to pick a "small" element from $\mathbb{F}_q$, we draw from a truncated discrete Gaussian distribution.

Definition 4.2.1. Let $\sigma > 0$, $C \in \mathbb{Z}$. The **truncated discrete Gaussian distribution** on $\mathbb{Z}$ with variance σ^2 and constraint C is given by

$$P(X = x) = \begin{cases} \dfrac{\exp\left(\frac{-x^2}{2\sigma^2}\right)}{\sum_{-C \leq y \leq C} \exp\left(\frac{-y^2}{2\sigma^2}\right)} & \text{if } x \in \mathbb{Z} \text{ and } -C \leq x \leq C \\ 0 & \text{otherwise.} \end{cases}$$

The error terms for LWE samples are drawn from this discrete Gaussian for some σ where the constraint is $\lfloor \frac{q}{2} \rfloor$ for q an odd prime.

4.3 Drawback

The drawback of cryptography based on the LWE problems is that the public and private key sizes needed are quite large, which is not the case for the Ring Learning with Errors (RLWE) problems. [Reg10] states that key sizes need to be on the order of n^2 using the LWE set-up. This is due to needing at least n many vectors $a_1, \ldots, a_n \in \mathbb{F}_q^n$. By introducing a ring structure, we are able to reduce the key sizes to almost linear size.

Chapter 5

Ring Learning with Errors

The Ring Learning with Errors (RLWE) problems were introduced in 2010 by [LPR10]. These problems are believed to be "hard" for both classical and quantum computers due to reductions to the worst-case Approx-SIVP, a version of the shortest vector problem, on ideal lattices on certain error distributions.

Due to the presumed difficulty of the RLWE problems, current candidates for safe encryption against quantum computers are based on these "hard" problems. As of August 2023, two out of the three proposed schemes for standardization from the NIST PQC competition are based on variations of the RLWE problems.

5.1 Non-Dual RLWE

The RLWE problems follow closely to the LWE set-up with the exception that the b_i coordinates now have dimension n. It is this property that makes RLWE more efficient.

Definition 5.1.1. A **RLWE instance** is $\mathcal{R} = (K, q, \sigma, s)$ where K is a number field, q a prime integer, $\sigma > 0$ is a real number, and $s \in \mathcal{O}_K/q\mathcal{O}_K$ where $\mathcal{O}_K$ is the ring of integers of K. We call s the **secret**.

Definition 5.1.2. A **(non-dual) RLWE sample** for an instance $\mathcal{R} = (K, q, \sigma, s)$ is of the form

$$(a, b = a \cdot s + e \pmod{q\mathcal{O}_K}) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K/q\mathcal{O}_K$$

where $a \in \mathcal{O}_K/q\mathcal{O}_K$ is chosen uniformly at random and e is drawn from a Gaussian distribution with standard deviation σ.

In analogy to the LWE problems, we have search and decision RLWE problems.

Definition 5.1.3. The **search (non-dual) RLWE problem** is to find the secret $s \in \mathcal{O}_K/q\mathcal{O}_K$ given m RLWE samples $(a_i, b_i) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K/q\mathcal{O}_K$.

Definition 5.1.4. The **decision (non-dual) RLWE problem** is to determine if the m pairs of the form $(a_i, b_i) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K/q\mathcal{O}_K$ are RLWE samples or uniformly random samples.

5.2 Dual RLWE

Before defining the dual RLWE problems, we define the dual of the ring of integers, denoted $\mathcal{O}_K^\vee$, of a number field K.

Definition 5.2.1. Let K be a number field. For a fractional ideal $I \subset K$, the **dual ideal** $I^\vee$ is $I^\vee = \{x \in K : Tr(xI) \subseteq \mathbb{Z}\}$.

Similar to the non-dual RLWE problem, the dual RLWE has two variations focusing on samples (a_i, b_i) of a special form.

Definition 5.2.2. Let $s \in \mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$. A **dual RLWE sample** is of the form

$$(a, b = a \cdot s + e) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$$

where $a \in \mathcal{O}_K/q\mathcal{O}_K$ is chosen uniformly at random and e is drawn from a Gaussian distribution with standard deviation σ.

Definition 5.2.3. The **search dual RLWE problem** is to find the secret $s \in \mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$ given m independent dual RLWE samples $(a, b) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$.

Definition 5.2.4. The **decision dual RLWE problem** is to determine if m samples of the form $(a_i, b_i) \in \mathcal{O}_K/q\mathcal{O}_K \times \mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$ are dual RLWE samples or uniformly random samples.

Notice that $a_i \in \mathcal{O}_K/q\mathcal{O}_K$, but here the b_i coordinate is in $\mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$. In [LPR10] Section 3.3, the authors go into great detail to explain why this is the correct definition for the dual RLWE problems. In short, having $b \in \mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$ gives the tightest connection to worst-case ideal lattice problems. The secret s corresponds to the closest vector in a Bounded Distance Decoding (BDD) problem. Along with a spherical Guassian distribution, this condition on the b_i also provides as much security as possible while preserving the encryption and decryption results for cryptographic applications.

It has been suggested in [Pei16b] that one should strongly consider a dual RLWE instance when attempting to solve these problems, due to the lattice of $\mathcal{O}_K^\vee$ being denser than the lattice of $\mathcal{O}_K$. Recall Example 2.4.3 where we found the dual basis of the basis $B = \{(1, 1), (0, 2)\}$. Upon examining the lattices generated by B and its dual, we saw that the points of the dual lattice were less spaced out. Due to the lattice's density, [Pei16b] highlights that non-dual RLWE instances may seem safe from attack in certain situations, but are vulnerable when converted to the dual RLWE setting.

5.3 PLWE

The last version of the RLWE problems is the Polynomial Ring Learning with Errors (PLWE) problems.

Let $f(x)$ be a monic irreducible polynomial in $\mathbb{Z}[x]$ of degree n. Let q be a prime such that $f(x) \pmod q$ factors completely into linear terms. Let $P = \mathbb{Z}[x]/\langle f(x)\rangle$; its elements are all polynomials of degree less than n with integer coefficients. The power basis of P is $1, x, x^2, \ldots, x^{n-1}$. Then $P/qP = (\mathbb{Z}/q\mathbb{Z})/\langle f(x)\rangle = \mathbb{F}_q[x]/\langle f(x)\rangle$.

Definition 5.3.1. Given a secret polynomial $s(x) \in P/qP$ and error polynomial $e(x)$ drawn from a discrete Gaussian distribution over P, a **PLWE sample** is of the form $(a(x), b(x)) \in P/qP \times P/qP$ where $a(x)$ is chosen uniformly at random and $b(x) = a(x)s(x) + e(x) \pmod{qP}$.

We want to bring attention to the fact that the errors are drawn from a discrete Gaussian distribution over P. By this, we mean that we are sampling the coefficients of the error polynomial from a discrete Gaussian over the integers. Recall from Chapter 3, where we showed how this sampling can be performed.

Similarly to the RLWE versions mentioned above, we can define search and decision PLWE problems as follows.

Definition 5.3.2. The **search PLWE problem** is to discover the secret $s(x) \in P/qP$ given m PLWE samples $(a_i(x), b_i(x))$.

Definition 5.3.3. The **decision PLWE problem** is to determine if m pairs of the form $(a_i(x), b_i(x)) \in P/qP \times P/qP$ are PLWE samples or uniformly random samples.

5.4 Sampling the Error Terms

In this section, we emphasize the main difference among the variations of the problem, which is how we sample the error terms.

- Non-dual RLWE and PLWE: $a_i, s \in \mathcal{O}_K/q\mathcal{O}_K$.
  - Non-dual RLWE: the error coefficients are sampled from a discrete Gaussian distribution over the lattice $\Lambda_{\mathcal{O}_K} = \theta(\mathcal{O}_K)$.
  - PLWE (when $\mathcal{O}_K$ is monogenic): error coefficients are sampled to be "small" in the power basis.
- Dual RLWE: $a_i \in \mathcal{O}_K$, $s \in \mathcal{O}_K^\vee/q\mathcal{O}_K^\vee$.
  - error coefficients are sampled from a Gaussian distribution over $\Lambda_{\mathcal{O}_K^\vee} = \theta(\mathcal{O}_K^\vee)$.

5.5 Converting between the Problems

As mentioned in Chapter 4, an important security parameter for RLWE is the standard deviation σ of the error distribution. In our study of this parameter, it is essential to find relationships between the variations of the RLWE problem. To begin, we compare the non-dual RLWE and dual RLWE set-ups.

5.5.1 Non-dual and dual RLWE Relationship

To start, we will first define the different ideal of a number field K. This discussion follows [Con09].

Definition 5.5.1. The **different ideal** of a number field K is

$$\mathcal{D}_K = (\mathcal{O}_K^\vee)^{-1} = \{x \in K : x\mathcal{O}_K^\vee \subset \mathcal{O}_K\}.$$

If K is monogenic, then the different ideal is principal. [Con09] states that

$$\mathcal{O}_K^\vee = \frac{1}{f'(\alpha)}\mathcal{O}_K,$$

when K is monogenic for f a minimal polynomial of $\alpha \in K$. In other words, $\mathcal{D}_K = (f'(\alpha))$, the ideal generated by the derivative of the minimal polynomial of α evaluated at α.Let γ be the generator of the different ideal $\mathcal{D}_K$. Then, when K is monogenic, $\mathcal{O}_K = \gamma\mathcal{O}_K^\vee$. If a power basis for $\mathcal{O}_K$ is $B = \{1, \alpha, \alpha^2, \ldots, \alpha^{n-1}\}$, then a power basis for $\mathcal{O}_K^\vee$ is $B^\vee = \{\gamma^{-1}, \gamma^{-1}\alpha, \gamma^{-1}\alpha^2, \ldots, \gamma^{-1}\alpha^{n-1}\}$.

Example 5.5.2. Let $K = \mathbb{Q}(\zeta_{2^3})$ and $\mathcal{O}_K = \mathbb{Z}[\zeta_{2^3}]$. Then $\mathcal{O}_K$ has integral basis $1, \zeta_{2^3}, \zeta_{2^3}^2, \zeta_{2^3}^3$. Thus the basis of the lattice $\theta(\mathcal{O}_K)$ is $\theta(1), \theta(\zeta_{2^3}), \theta(\zeta_{2^3}^2), \theta(\zeta_{2^3}^3)$. Then the dual ring of integers is $\mathcal{O}_K^\vee = \frac{1}{2^2}\mathcal{O}_K = \frac{1}{4}\mathbb{Z}[\zeta_{2^3}]$. The integral basis of $\mathcal{O}_K^\vee$ is $\frac{1}{4}, \frac{1}{4}\zeta_{2^3}, \frac{1}{4}\zeta_{2^3}^2, \frac{1}{4}\zeta_{2^3}^3$. Thus the basis of the lattice of $\mathcal{O}_K^\vee$ is

$$\theta\left(\frac{1}{4}\right), \theta\left(\frac{1}{4}\zeta_{2^3}\right), \theta\left(\frac{1}{4}\zeta_{2^3}^2\right), \theta\left(\frac{1}{4}\zeta_{2^3}^3\right).$$

Example 5.5.3. For general 2-power cyclotomics, $K = \mathbb{Q}(\zeta_{2^\ell})$, the ring of integers is $\mathcal{O}_K = \mathbb{Z}[\zeta_{2^\ell}]$ and its dual is $\mathcal{O}_K^\vee = \frac{1}{2^{\ell-1}}\mathcal{O}_K = \frac{1}{2^{\ell-1}}\mathbb{Z}[\zeta_{2^\ell}]$.

When K is not monogenic and $\alpha \in \mathcal{O}_K$, we have that $\mathbb{Z}[\alpha]$ is a fractional ideal of K (because $\mathbb{Z} \subset \mathcal{O}_K$). Since we also have the inclusion $\mathcal{O}_K \subset \mathcal{O}_K^\vee$, then we get that $\mathcal{O}_K^\vee \subset \mathbb{Z}[\alpha]^\vee$.

5.5.2 PLWE and RLWE Relationship

Now, we discuss the relationship between the PLWE and RLWE problems. As mentioned, the PLWE problems are defined only for monogenic fields. Throughout this section, suppose K is a monogenic number field, $\mathcal{O}_K$ its ring of integers, and $\mathcal{O}_K^\vee$ its dual. Let $P = \mathbb{Z}[x]/\langle f(x)\rangle$ be a polynomial ring for $f(x)$ a monic irreducible polynomial in $\mathbb{Z}[x]$.

Suppose that $\alpha \in \mathcal{O}_K$ is a root of $f(x)$, then $P \cong \mathcal{O}_K$ via the map $x \mapsto \alpha$. By the previous subsection, when K is monogenic, $\mathcal{O}_K = \gamma\mathcal{O}_K^\vee$ for γ the generator of the different ideal. So $P \cong \mathcal{O}_K = \gamma\mathcal{O}_K^\vee$. In Chapter 8, we will explore ways in which we can attempt to define PLWE-like sampling for non-monogenic rings of integers.

In PLWE, we represent elements of $\mathcal{O}_K$ as polynomials in some α, and in RLWE, we represent elements of $\mathbb{C}^n$ via the canonical embedding. To relate the two sampling techniques, we must find the change of basis matrix to go from the non-dual/dual RLWE setting to the PLWE setting.

Definition 5.5.4. Let $1, \alpha, \alpha^2, \ldots, \alpha^{n-1}$ be an integral basis for $\mathcal{O}_K$. The matrix M_α is the **change of basis matrix** with columns $\{\theta(\alpha^i)\}$ that maps P to $\theta(\mathcal{O}_K)$.

In other words, for $a_0 + a_1x + \ldots + a_{n-1}x^{n-1} \in P = \mathbb{Z}[x]/\langle f(x)\rangle$, then

$$M_\alpha \begin{bmatrix} a_0 \\ a_1 \\ \cdots \\ a_{n-1} \end{bmatrix} \in \theta(\mathcal{O}_K).$$

Similarly, we have a change of basis matrix that corresponds to P and $\mathcal{O}_K^\vee$:

Definition 5.5.5. The matrix $M_\alpha^\vee$ is the **change of basis matrix** that maps P to $\theta(\mathcal{O}_K^\vee)$. Let $\gamma^{-1}, \gamma^{-1}\alpha, \gamma^{-1}\alpha^2, \ldots, \gamma^{-1}\alpha^{n-1}$ be an integral basis of $\mathcal{O}_K^\vee$, then the

columns of $M_\alpha^\vee$ are $\{\theta(\gamma^{-1}\alpha^i)\}$.

We wish to highlight that the PLWE problems require discrete Gaussian distributions, since we are sampling coefficients of the error polynomial over the integers. However, the RLWE problems can be defined using both discrete and continuous Gaussians. Thus to fully convert between PLWE and RLWE, we also need to either be considering only discrete Gaussian distributions or discretize the continuous Gaussian distribution from the RLWE problems. We refer the reader to Chapter 3 for ways in which we can discretize a continuous Gaussian.

Chapter 6

Known Attacks

There are a handful of attacks that attempt to solve the Ring Learning with Errors problems [Pei09, ACPS09, AG11, MP12, EHL14, ELOS15, Pei16b, CIV16, CLS17a, CLS17b]. While these problems are rely on the hardness of variations of the shortest vector problem, specific instances might be unsafe due to the vulnerabilities of the underlying number field.

To begin this chapter, we start with an algorithm which attempts to solve the LWE problems. The decision problem attacks rely on the ability to distinguish between a Gaussian distribution and a uniform distribution. The search problem attacks attempt to reduce the problem to errorless samples, so that one can solve for the secret using Gaussian elimination.

6.1 Attacking the LWE problems

Recall the LWE set-up from Chapter 4: Let q and n be positive integers. Given a secret $s \in \mathbb{F}_q^n$ chosen uniformly at random and $e \in \mathbb{F}_q$ a "small" error term, an LWE sample has the form

$$(a, b = a \cdot s + e) \in \mathbb{F}_q^n \times \mathbb{F}_q$$

where a is chosen uniformly at random from $\mathbb{F}_q^n$.

To begin discussing the attacks to these problems, we will first make note of some essential components to keep in mind. In Chapter 4, we mentioned that error terms are essential for these problems because otherwise, one can easily solve for the secret using techniques from linear algebra. The following attacks rely on the ability to reduce to errorless samples or distinguish between an error distribution and a uniform distribution.

6.1.1 Linearization

Another algorithm which attempts to solve the LWE problems uses linearization. [AG11] provides an attack that uses a change of variable technique to obtain a system of linear equations so that we can find the secret of an LWE search problem. The authors call this technique a "reduction to Gaussian elimination with polynomial coefficients" because this method of reducing to Gaussian elimination involves equations whose coefficients are low-degree polynomials.

Before stating the algorithm, we introduce the definition of linearization from [AG11].

Definition 6.1.1. Let d, n be integers and $[n] = \{1, 2, \ldots, n\}$. Let

$$g(z) = \sum_{S \subseteq [n], |S| \leq d} c_S \prod_{i \in S} z_i$$

be a multilinear polynomial of degree d in n variables where $c_S \in \mathbb{F}_2$. The **linearization** of g, denoted $L(g)$, is a linear function over the variables y_S, where S ranges over subsets of $[n]$ of size at most d

$$L(g) = \sum_{S \subseteq [n], |S| \leq d} c_S y_S.$$

We define $y_0 = 1$, so that the number of new variables is $\sum_{i=0}^{d} \binom{n}{i}$.

The following are assumption for this attack:

- $d \in \mathbb{Z}$ is a bound for the error terms (i.e. $|e_i| \leq d$),
- $2d + 1 < q$.

So, by assumption, $d < \frac{q-1}{2}$ and $e \in (-\frac{q-1}{2}, \frac{q-1}{2})$.

Linearization Algorithm from [AG11]:

1. Write a one variable polynomial of degree $D = 2d + 1$ such that $P(e) = 0$ for e an error term drawn according to an LWE instance (truncated discrete Gaussian - see Chapter 4 for reference):

$$P(e) = e \prod_{j=1}^{d} (e + j)(e - j).$$

2. Let $z = (z_j)$ be a vector of variables with dimension n. Write a system of equations that allows us to recover the secret s, with high probability, by substituting $e = a \cdot z + b$ in the polynomial $P(e)$. This gives a polynomial of degree D of the variables z_j:

$$(a \cdot z + b) \prod_{j=1}^{d} (a \cdot z + b + j)(a \cdot z + b - j). \tag{6.1}$$

3. Linearize Equation (6.1) using a variable vector y, which is indexed by vectors $v \in \mathbb{Z}^n$ such that $1 \leq \sum_{i=1}^{n} v_i \leq D$. Then y_v corresponds to $\prod_{i=1}^{n} z_j^{v_j}$. We denote $y_0 = 1$ and define y^k to be the vector of all variables with total degree k, i.e. containing all y_v with $\sum v_i = k$. Hence

$$y = (1, y^1, y^2, \ldots, y^D).$$

4. Let L be the linearization operator in Definition 6.1.1, which replaces each $\prod_{i=1}^{n} z_j^{v_j}$ in the polynomial $P(a \cdot z + b)$ with the corresponding y_v variable. The linearized equation will be a linear constraint on the y variables,

$$L(P(a \cdot z + b)) = 0.$$

[AG11] states that the algorithm queries an LWE oracle $\mathrm{O}(N\alpha q^2 \log q)$ times, generating a system of linear equations over the y variables. Using Gaussian elimination, one can then find the secret s, solving the LWE search problem.

6.2 Attacking the RLWE problems

The articles [EHL14, ELOS15, CLS17b, CIV16, CLS17a] introduce an attack, which we refer to as the Chi-Squared Attack, and that attempts to solve the decision RLWE problem. Following its algorithm, we mention how one can use the general set-up of the Chi-Squared Attack along with the Chi-Squared Test to find the secret given RLWE samples, solving the search problem.

We will state this attack in the non-dual RLWE setting.

6.2.1 Chi-Squared Attack

Let K be a number field and $\mathcal{O}_K$ its ring of integers. Let q be an unramified prime in K. The essential component of this attack is the ring homomorphism

$$\phi : \mathcal{O}_K/q\mathcal{O}_K \to \mathbb{F}_{q^f},$$

where f is a residue degree of a prime $\mathfrak{q}$ dividing q.

This map is due to the Chinese Remainder Theorem, modding out by $\mathfrak{q}_i|q$,

$$\mathcal{O}_K/q\mathcal{O}_K \cong \mathcal{O}_K/\mathfrak{q}_1 \times \mathcal{O}_K/\mathfrak{q}_2 \times \ldots \times \mathcal{O}_K/\mathfrak{q}_k,$$

where $k|n$. Now, since $\mathcal{O}_K$ is a Dedekind Domain, all the prime ideals $\mathfrak{q}_i$ for $1 \leq i \leq k$ are maximal ideals, therefore $\mathcal{O}_K/\mathfrak{q}_i \cong \mathbb{F}_{q^{f_i}}$ for $1 \leq i \leq k$. Thus

$$\mathcal{O}_K/q\mathcal{O}_K \cong \mathbb{F}_{q^{f_1}} \times \ldots \times \mathbb{F}_{q^{f_k}}.$$

Hence ϕ is the map which records only one of the $\mathbb{F}_{q^{f_i}}$ coordinates in the isomorphism above.

For an RLWE instance $\mathcal{R} = (K, q, \sigma, s)$, the Chi-Squared Attack is as follows:

1. Transport samples to $(\phi(a_i), \phi(b_i)) \in \mathbb{F}_{q^f} \times \mathbb{F}_{q^f}$.

2. Consider each possible guess $g \in \mathbb{F}_{q^f}$, in turn, as the image of the secret $\phi(s)$.

3. Calculate the image of the errors as

$$\phi(b_i) - \phi(a_i)g = \phi(e_i).$$

4. If the values $\phi(b_i) - \phi(a_i)g$ follow a non-uniform distribution, then we have guessed correctly ($g = \phi(s)$) and the samples are RLWE samples. Otherwise, if they follow a uniform distribution, then either the guess is incorrect or the uniform and error distributions are indistinguishable. We continue until all possible guesses g have been made.

The Chi-Squared Attack is successful, with high probability, when the Gaussian distribution and uniform distribution are distinguishable and q and f are small enough that we can loop through the possible guesses in $\mathbb{F}_{q^f}$. The distributions have a higher

probability of being distinguishable when σ is small, n is small, and q is large. It is also helpful to have a large number of samples (a_i, b_i). Therefore, for the sake of security, we must choose RLWE instances where the reduction modulo a prime $\mathfrak{q}$ of the error distribution is indistinguishable from uniform. In the next chapter, we discuss the security parameter bounds for which RLWE instances are better protected from this attack.

Chapter 7

Security Conditions

In Chapter 6, we discussed two of attacks that attempt to solve the LWE and RLWE problems. These attacks utilize two vulnerabilities of these problems - reduction to errorless samples and distinguishable distributions. We saw that the choice of the standard deviation σ for our error distribution is an essential piece of security for RLWE instances. When σ is too small, $e = 0$ with high probability, allowing for errorless samples. However, when σ is too large, the error terms run the risk of wrapping around modulo q, which is an issue for decryption. So while we do not want a small σ, we also need to ensure that our σ is not too large. Here, we will discuss the current security parameter bounds in the literature for the Chi-Squared Attack and convert between them to suggest tighter conditions.

7.1 Known Security Bounds

To start, we define the statistical distance between two distributions.

Definition 7.1.1. If P, Q are two probability distributions on the set S, then the **statistical distance** between P and Q is defined as

$$d(P, Q) = \frac{1}{2} \sum_{t \in S} |P(t) - Q(t)|.$$

This value tells us how similar the two distributions are: the smaller $d(P, Q)$ is the harder it is to distinguish P from Q. Since the decision problems rely on being able to differentiate between a uniform and non-uniform distribution, security bounds from [Pei16b] and [CLS17b] tell us when instances are safe.

In Theorem 5.2 of [Pei16b], conditions are given for which a dual RLWE instance with a continuous error distribution is safe against the Chi-Squared Attack (the error distribution is indistinguishable from uniform).

Theorem 7.1.2 (Theorem 5.2 [Pei16b]). *Let $\mathfrak{q} \subset \mathcal{O}_K$ be any ideal of norm $N(\mathfrak{q}) \leq 2^n$, where n is the degree of the number field K. Let the width of the Gaussian distribution $r = \sqrt{2\pi}\sigma \geq 2$. Then the reduced error distribution mod $\mathfrak{q}\mathcal{O}_K^\vee$ is within statistical distance 2^{-2n} of a uniform distribution.*

Interestingly, the bound on the standard deviation of the continuous error distribution is fixed; it does not depend on K or q.

At the same time, for 2-power cyclotomic fields, Theorem 3 and Corollary 1 in [CLS17b] mention conditions in which the reduction under the map $\phi : \mathcal{O}_K/q\mathcal{O}_K \to \mathbb{F}_{q^f}$ of the error distribution will be indistinguishable from uniform for the non-dual RLWE instance (and equivalently for PLWE, since K is monogenic) using a discrete Gaussian distribution.

Theorem 7.1.3 (Theorem 3 and Corollary 1 [CLS17b]). *Let $K = \mathbb{Q}(\zeta_{2^\ell})$ be a 2-power cyclotomic field. If $q \equiv 1 \pmod{2^\ell}$ and $q < (2^\ell)^2$, then the statistical distance between the reduction modulo $\mathfrak{q}$ of the error distribution, where $\mathfrak{q}$ divides q, and a uniform distribution is bounded above by*

$$\frac{q-1}{2}\left(\frac{2^\ell + \sqrt{q}}{2^{\ell+1}}\right)^{2^\ell \sigma^2}. \tag{7.1}$$

Unlike the bound from [Pei16b], this one depends on the prime q and the standard deviation.

We note that since $q \equiv 1 \pmod{2^\ell}$, the norm of $\mathfrak{q}$ in K dividing q is just q. Also, since $q < (2^\ell)^2$, the quotient $\frac{2^\ell+\sqrt{q}}{2^{\ell+1}} < 1$. For fixed ℓ, one can keep the expression (7.1) small by decreasing q and increasing σ. Similarly, for fixed ℓ, if q is large and σ is small, then (7.1) is big, meaning it should be easy to distinguish between the error distribution and a uniform distribution, making the instance vulnerable to attack. Since the Chi-Squared Attack depends on the ability to distinguish the reduced error distribution from a uniform distribution, this theorem suggests that there might be a "trade-off" between the norm of the prime and the standard deviation of the Gaussian distribution: a smaller prime can be safe with a wider distribution and a larger prime can be safe with a narrower distribution. [Pei16b] does not appear to show this trade-off.

Notice that the condition from [CLS17b] is for the PLWE problems, while the condition from [Pei16b] is for the dual RLWE problems. Another note is that [Pei16b] uses a continuous Gaussian distribution, but [CLS17b] considers a discrete distribution. Hence, to compare these results, we want to use the relationships from Chapter 5 to convert the parameter bound $\sigma \geq \frac{2}{\sqrt{2\pi}}$ from [Pei16b] to get its equivalence for the PLWE setting.

For the remainder of this chapter, we convert the bounds using the change of basis matrix and properties of subgaussian distributions. We then take our converted bounds and test them under the Chi-Squared Attack. The results from these tests suggest that the bound $\sigma \geq \frac{2}{\sqrt{2\pi}}$ from [Pei16b] can be made tighter.

7.2 Converting Bounds

For ease, we convert the bounds from a continuous distribution to a discrete distribution using the width parameter and then revert back to the standard deviation. (The bound $r \geq 2$ is cleaner than $\sigma \geq \frac{2}{\sqrt{2\pi}}$.)

To start, recall discretizing a continuous distribution to a lattice coset $\Lambda + c$ from Chapter 3. The idea was to sample a short vector $f \in \Lambda + (c - x)$ and output $y = x + f$ where $y \in \Lambda + c$.

Let $K = \mathbb{Q}(\zeta_{2^\ell})$. In the continuous dual RLWE setting, the condition $r \geq 2$ implies that the width of x as a subgaussian is $r_1 \geq 2$. As a consequence, the width of f as a subgaussian is $\sqrt{2\pi}s_1(B) = \sqrt{2\pi}\frac{\sqrt{2}^{\ell-2}}{2^{\ell-1}} = r_2$ where $s_1(B)$ is the spectral norm of B, a basis of the lattice Λ, viewed as a matrix. After discretizing, we find that the bound on the width r of y is

$$\begin{aligned} r &\geq (2^2 + (\sqrt{2\pi}\frac{\sqrt{2}^{\ell-2}}{2^{\ell-1}})^2)^{1/2} \\ &= (4 + 2\pi\frac{2^{\ell-2}}{2^{2(\ell-1)}})^{1/2} \\ &= (4 + \pi\frac{2^{\ell-1}}{2^{2(\ell-1)}})^{1/2} \\ &= (4 + \frac{\pi}{2^{\ell-1}})^{1/2} \\ &= \sqrt{4 + \frac{\pi}{2^{\ell-1}}}. \end{aligned}$$

Thus a continuous error distribution which has width $r \geq 2$ corresponds to a discrete error distribution with width $r \geq \sqrt{4 + \frac{\pi}{2^{\ell-1}}}$. Notice that as ℓ increases, the discrete error distribution width approaches 2, which is what we would expect.

Converting this back to the standard deviation gives us

$$\begin{aligned} \sigma' &\geq \frac{\sqrt{4 + \frac{\pi}{2^{\ell-1}}}}{\sqrt{2\pi}} \\ &= \sqrt{\frac{4 + \frac{\pi}{2^{\ell-1}}}{2\pi}} \\ &= \sqrt{\frac{2}{\pi} + \frac{1}{2^\ell}}, \end{aligned}$$

as a secure bound for a dual RLWE instance with a discrete Gaussian distribution according to [Pei16b].

Now that we have the corresponding discrete parameter bound for the dual RLWE setting, we can use the change of basis matrix to find the PLWE parameter bound for a discrete error distribution.

Using the change of basis matrix code (see Appendix A), one can show that

$$\sigma_{can} \geq \sqrt{\frac{2^{2\ell}}{\pi} + 2^{\ell-1}}$$

when using the standard canonical embedding. Using the adjusted canonical embedding (see Appendix A.2), one would find that

$$\sigma_{adj} \geq \sqrt{\frac{2^{2\ell-1}}{\pi} + 2^{\ell-2}},$$

which is expected (the adjusted canoncial embedding has an extra factor of $\sqrt{2}$, which is why the two bounds only differ by that factor).

7.3 Testing Bounds

Using the bounds on σ_{can} and σ_{adj} and the code for the Chi-Squared Attack (see Appendix B) originally from [ELOS15], we tested the predicted security on 2-power cyclotomic fields.

Table 7.1 shows the success rate of the Chi-Squared Attack solving the PLWE problem from our experiments. The table shows the various primes q used, the corresponding $\frac{1}{2}+\frac{q}{2^{\ell+1}}$ value (from Theorem 3 and Corollary 1 in [CLS17b]), the largest standard deviation σ for which the PLWE problem was solved, and the results of five separate trials of 100 attempts. All these results used the number field $K = \mathbb{Q}(\zeta_{2^4})$.

Primes were chosen to satisfy the conditions from [CLS17b]. Here the predicted safe is $\sigma_{can} \geq 9.46$ (standard canonical) or $\sigma_{adj} \geq 6.69$ (adjusted canonical) from previous calculations.

Table 7.1: Largest σ for which the PLWE instance was solved for $K = \mathbb{Q}(\zeta_{2^4})$.

Testing Bounds							
q	$\frac{1}{2} + \frac{\sqrt{q}}{2^{\ell+1}}$	σ	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
17	0.6288	13	**1/100**	0/100	**3/100**	**1/100**	**1/100**
19	0.8078	9	**1/100**	0/100	0/100	0/100	0/100
113	0.8322	12	0/100	0/100	**1/100**	0/100	0/100
193	0.9341	5	**1/100**	0/100	0/100	0/100	0/100
241	0.9851	1	0/100	0/100	0/100	**1/100**	0/100

While the first table showed the largest σ for which the PLWE problem was solved, Table 7.2 and Table 7.3 look closer at the trials run on the smallest prime $q = 17$ and the largest prime $q = 241$, respectively.

Table 7.2: Successes for varying σ with $q = 17$ and $K = \mathbb{Q}(\zeta_{2^4})$.

Testing with small prime					
σ	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
14	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
13	**1 / 100**	0 / 100	**3 / 100**	**1 / 100**	**1 / 100**
12	0 / 100	**3 / 100**	0 / 100	**1 / 100**	0 / 100
11	0 / 100	**1 / 100**	0 / 100	0 / 100	0 / 100
10	**1 / 100**	0 / 100	**1 / 100**	0 / 100	0 / 100
9	0 / 100	**3 / 100**	**1 / 100**	0 / 100	0 / 100
8	**1 / 100**	0 / 100	0 / 100	**1 / 100**	**1 / 100**
7	0 / 100	**1 / 100**	**1 / 100**	**1 / 100**	0 / 100
6	**1 / 100**	**2 / 100**	**1 / 100**	**1 / 100**	**1 / 100**
5	0 / 100	**3 / 100**	**1 / 100**	**1 / 100**	0 / 100

From these tables, we predict that there must be a "trade-off" between the prime modulus and standard deviation of the Gaussian distribution that controls the security of these problems. [Pei16b] states that $N(\mathfrak{q}) \leq 2^n$ and $\sigma \geq \frac{2}{\sqrt{2\pi}}$ ensures security for a dual RLWE instance with a continuous Gaussian distribution. Our results show that there are PLWE instances with a discrete Gaussian distribution which are secure against the Chi-Squared Attack, even if they are outside of the range predicted by [Pei16a], showing that Peikert's results are not sharp. Fix $q = 241$.

Table 7.3: Successes for varying σ with $q = 241$ with $K = \mathbb{Q}(\zeta_{24})$.

Testing with large prime					
σ	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
10	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
9	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
8	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
7	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
6	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
5	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
4	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
3	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
2	0 / 100	0 / 100	0 / 100	0 / 100	0 / 100
1	0 / 100	0 / 100	0 / 100	**1 / 100**	0 / 100

Then $\sigma_{can} \geq 9.46$ and $\sigma_{adj} \geq 6.69$, implying that the PLWE instance should be safe. From Table 7.3, we see that this PLWE instance is still secure for smaller standard deviation, when $\sigma \geq 2$.

We hypothesize the following relationship between the prime modulus and the standard deviation of the Gaussian distribution:

- For small primes, a large standard deviation is needed for security.
- As the prime grows larger, the standard deviation can grow smaller and still provide security.
- There is a limit to how large the standard deviation can be due to possible issues with decryption.
- There is a limit to how small the standard deviation can be before the instance becomes vulnerable (reduces to an errorless RLWE).

Figure 7.1 illustrates these predictions.

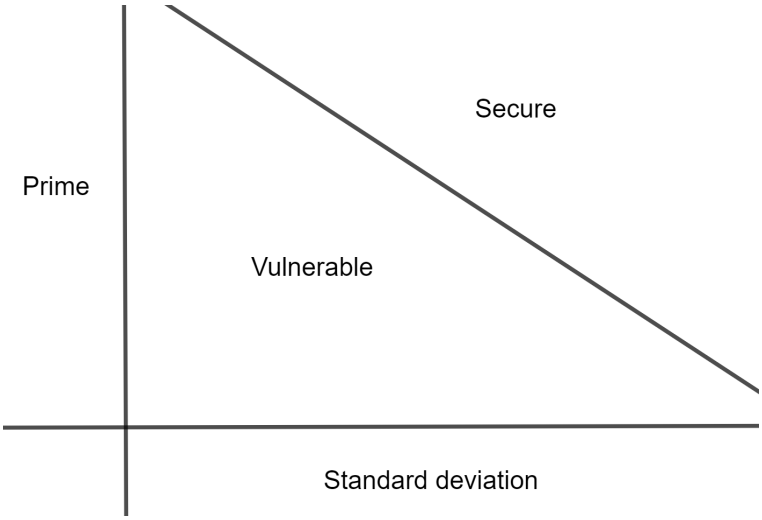

Figure 7.1: Predictions for bounds of security.

Chapter 8

PLWE for Non-Monogenic Rings of Integers

As defined, the Polynomial Ring Learning with Errors sampling algorithm requires that the underlying number field K be monogenic. The reason for this requirement is that in this case, there is a monic irreducible polynomial $f(x) \in \mathbb{Z}[x]$ of degree n such that

$$\mathcal{O}_K \cong P = \mathbb{Z}[x]/\langle f(x) \rangle.$$

As a consequence the elements of $\mathcal{O}_K$ can be represented as polynomials, which can be drawn by sampling their coefficients. As motivation for this chapter, we use the Dedekind-Kummer Theorem to extend this type of sampling to the non-monogenic case.

8.1 Dedekind-Kummer Theorem

We begin by stating the Dedekind-Kummer Theorem:

Theorem 8.1.1 (Dedekind-Kummer Theorem [Ded78])**.** *Let K be a number field and let $\alpha \in \mathcal{O}_K$ be such that $K = \mathbb{Q}(\alpha)$. Let $f(x) \in \mathbb{Z}[x]$ be the minimal polynomial of α*

over $\mathbb{Q}$ *and* $P = \mathbb{Z}[\alpha]/\langle f(x)\rangle$. *For any prime* q *with* $q \nmid [\mathcal{O}_K : P]$, *write*

$$f(x) \equiv g_1(x)^{e_1} \cdots g_r(x)^{e_r} \pmod{q}$$

where $g_i(x)$*'s are distinct monic irreducibles in* $(\mathbb{Z}/q\mathbb{Z})[x]$. *Then*

$$q\mathcal{O}_K = \mathfrak{q}_1^{e_1} \cdots \mathfrak{q}_r^{e_r}$$

where each $\mathfrak{q}_i$ *is a prime ideal of* $\mathcal{O}_K$ *and* $e_i > 0$. *Futhermore, there is a bijection between the* $\mathfrak{q}_i$*'s and the* g_i*'s such that* $N(\mathfrak{q}_i) = q^{\deg g_i}$.

The following corollary says that this is also true if $q \nmid disc(P)$.

Corollary 8.1.2. *Let* $K = \mathbb{Q}(\alpha)$ *be a number field and* $\alpha \in \mathcal{O}_K$ *have minimal polynomial* $f(x) \in \mathbb{Z}[x]$. *For any* $q \nmid disc(P)$ *where* $P = \mathbb{Z}[\alpha]/\langle f(x)\rangle$, *the shape of the factorizations of* $q\mathcal{O}_K \in \mathcal{O}_K$ *and* $f(x) \in (\mathbb{Z}/q\mathbb{Z})[x]$ *agree.*

Proof. Since $disc(P) = disc(\mathcal{O}_K)[\mathcal{O}_K : P]^2$ and $q \nmid disc(P)$, then $q \nmid [\mathcal{O}_K : P]$ and so it follows by the Dedekind-Kummer Theorem. □

The following example illustrates that all but finitely many primes are at our disposal to choose which satisfy the Dedekind-Kummer Theorem. We also explore the shape of certain factorizations.

Example 8.1.3. Let $K = \mathbb{Q}(\sqrt[4]{24})$. Certainly $P = \mathbb{Z}[\sqrt[4]{24}] \subset \mathcal{O}_K$. Then

$$disc(\mathbb{Z}[\sqrt[4]{24}]) = 2^{17} \cdot 3^3.$$

So any prime $q \neq 2, 3$ does not divide $[\mathcal{O}_K : \mathbb{Z}[\sqrt[4]{24}]]$. Then for any prime $q \neq 2, 3$, we can show how $q\mathcal{O}_K$ factors by computing how $x^4 - 24$, which is the minimal polynomial of $\sqrt[4]{24}$, factors in $(\mathbb{Z}/q\mathbb{Z})[x]$.

Consider $q = 5$, then we have

$$x^4 - 24 \equiv (x^2 + 2)(x^2 + 3) \pmod 5.$$

So $q\mathcal{O}_K = \mathfrak{q}_{25}\mathfrak{q}'_{25}$. We use the notation $\mathfrak{q}_{25}$ and $\mathfrak{q}'_{25}$ since each of these are prime ideals of norm 25.

Proposition 8.1.4. *If a prime q does not divide $[\mathcal{O}_K : P]$, then*

$$\mathcal{O}_K/(q) \cong P/qP = \mathbb{F}_q[x]/\langle f(x)\rangle.$$

Proof. Let $d = [\mathcal{O}_K : \mathbb{Z}[\alpha]]$. Suppose q is a prime such that $q \nmid d$. Notice that

$$d\mathcal{O}_K \subset \mathbb{Z}[\alpha] \subset \mathcal{O}_K.$$

There exists a natural ring homomorphism $\mathbb{Z}[\alpha]/(q) \hookrightarrow \mathcal{O}_K/(q)$, which is onto. Since $\mathbb{Z}[\alpha]$ and $\mathcal{O}_K$ are free rank n $\mathbb{Z}$-modules, then both $\mathbb{Z}[\alpha]/(q)$ and $\mathcal{O}_K/(q)$ have size q^n (there are q possibilities for each of the coefficients). So the ring homomorphism is an isomorphism, $\mathbb{Z}[\alpha]/(q) \cong \mathcal{O}_K/(q)$.

Now, we show that $\mathbb{Z}[\alpha]/(q) \cong \mathbb{F}_q[x]/\langle f(x)\rangle$. Let $f(x) \in \mathbb{Z}[x]$ be the minimal polynomial of α. We have that $\mathbb{Z}[\alpha] \cong \mathbb{Z}[x]/\langle f(x)\rangle$ as rings. Then reducing modulo q gives

$$\mathbb{Z}[\alpha]/(q) \cong \mathbb{Z}[x]/(f(x), q) \cong (\mathbb{Z}/q\mathbb{Z})[x]/\langle f(x)\rangle \cong \mathbb{F}_q[x]/\langle f(x)\rangle.$$

So since both $\mathcal{O}_K/(q)$ and $\mathbb{F}_q[x]/\langle f(x)\rangle$ are each isomorphic to $\mathbb{Z}[\alpha]/(q)$, then

$$\mathcal{O}_K/(q) \cong \mathbb{F}_q[x]/\langle f(x)\rangle.$$

□

For the full proof of the Dedekind-Kummer Theorem, see [Con12].

Example 8.1.5. Let $K = \mathbb{Q}(\alpha) \cong \mathbb{Q}[x]/(x^3 - x - 8)$, which is a non-monogenic number field. The ring of integers of K is $\mathcal{O}_K = \mathbb{Z} + \mathbb{Z}\alpha + \mathbb{Z}(\frac{1}{2}\alpha^2 + \frac{1}{2}\alpha)$. Consider $q = 17$. We want to show that $\mathcal{O}_K/(17) \cong \mathbb{F}_{17}[\overline{\alpha}]$, where $\overline{\alpha} \equiv \alpha \pmod{q}$. First, notice that the elements of $\mathbb{F}_{17}[\overline{\alpha}]$ are of the form $a_1 + a_2\overline{\alpha} + a_3\overline{\alpha}^2$ where $a_i \in \mathbb{Z}/17\mathbb{Z}$. The elements of $\mathcal{O}_K/(17)$ are of the form $b_1 + b_2\overline{\alpha} + b_3(\frac{1}{2}\overline{\alpha}^2 - \frac{1}{2}\overline{\alpha}) = b_1 + (b_2 - \frac{1}{2}b_3)\overline{\alpha} + \frac{1}{2}b_3\overline{\alpha}^2$ where $b_i \in \mathbb{Z}/17\mathbb{Z}$. So we have the following maps

$$\begin{aligned} a_1 &\mapsto b_1, \\ a_2 &\mapsto b_2 - \frac{1}{2}b_3 \equiv b_2 + 9b_3 \pmod{17}, \\ a_3 &\mapsto \frac{1}{2}b_3 \equiv 9b_3 \pmod{17}. \end{aligned}$$

(Notice that $2^{-1} \equiv 9 \pmod{17}$.) We want to show that these maps are surjective. For this, it suffices to show that multiplication by 9 is surjective. We will enumerate the possibilities since our prime is small.

$$\begin{array}{lll} 0 \equiv 9 \cdot 0 \pmod{17} & 1 \equiv 9 \cdot 2 \pmod{17} & 2 \equiv 9 \cdot 4 \pmod{17} \\ 3 \equiv 9 \cdot 6 \pmod{17} & 4 \equiv 9 \cdot 8 \pmod{17} & 5 \equiv 9 \cdot 10 \pmod{17} \\ 6 \equiv 9 \cdot 12 \pmod{17} & 7 \equiv 9 \cdot 14 \pmod{17} & 8 \equiv 9 \cdot 16 \pmod{17} \\ 9 \equiv 9 \cdot 1 \pmod{17} & 10 \equiv 9 \cdot 3 \pmod{17} & 11 \equiv 9 \cdot 5 \pmod{17} \\ 12 \equiv 9 \cdot 7 \pmod{17} & 13 \equiv 9 \cdot 9 \pmod{17} & 14 \equiv 9 \cdot 11 \pmod{17} \\ 15 \equiv 9 \cdot 13 \pmod{17} & 16 \equiv 9 \cdot 15 \pmod{17} & \end{array}$$

We also see that $\mathbb{F}_{17}[\alpha]$ and $\mathcal{O}_K/(17)$ are both of size 17^3 (since there are 17 possibilities for each of the coefficients, which is true when we have a basis). Thus we have that $\mathcal{O}_K/(17) \cong \mathbb{F}_{17}[\alpha]$.

With the Dedekind-Kummer Theorem as motivation, we explore two ways in which we attempt to expand the PLWE sampling:

- sample directly from $\mathcal{O}_K/q\mathcal{O}_K$,
- use coset representatives to sample.

8.2 Sample Directly from $\mathcal{O}_K/q\mathcal{O}_K$

Let $K = \mathbb{Q}(\alpha)$ be a number field. Let $P = \mathbb{Z}[x]/\langle f(x)\rangle$ be a polynomial ring where $f(x) \in \mathbb{Z}[x]$ is the monic minimal polynomial of $\alpha \in K$. Suppose q be a prime such that $q \nmid [\mathcal{O}_K : P]$.

Our algorithm is to first reduce the ring of integers $\mathcal{O}_K$ modulo q so that

$$\mathcal{O}_K/(q) \cong P/qP = (\mathbb{Z}/q\mathbb{Z}[x])/\langle f(x)\rangle \cong \mathbb{F}_q[\overline{\alpha}].$$

Then P/qP is monogenic and we can sample the error terms from here, using a truncated discrete Gaussian distribution, which we recall from Chapter 4:

Definition 8.2.1. Let $\sigma > 0$, $C \in \mathbb{Z}$. The **truncated discrete Gaussian distribution** on $\mathbb{F}_q$ with variance σ^2 and constraint C is given by

$$P(X = x) = \begin{cases} \dfrac{\exp\left(\frac{-x^2}{2\sigma^2}\right)}{\sum_{-C\leq y\leq C}\exp\left(\frac{-y^2}{2\sigma^2}\right)} & \text{if } x \in \mathbb{F}_q \text{ and } -C \leq x \leq C \\ 0 & \text{otherwise.} \end{cases}$$

8.2.1 Vulnerabilities

With this proposed sampling algorithm, there are a few concerns about making the PLWE instances more vulnerable to attack. The first concern is that the error terms

now come from a much smaller space. Thus there are fewer values that they can take. Algorithm 1 in [ELOS15] is an attack on the PLWE decision problem for which we know that the error terms come from a small set of possible values. Additionally, the Linearization Attack from [AG11], mentioned in Chapter 6, is an algorithm works when the error terms are bounded. With these concerns and a known attack that makes this set-up vulnerable, this sampling approach is not promising.

8.3 Coset Representatives Sampling

Let $\mathcal{O}_K \neq \mathbb{Z}[\alpha]$ be a ring of integers for a number field $K = \mathbb{Q}(\alpha)$ and $\alpha \in \mathcal{O}_K$. Then $\mathbb{Z}(\alpha) \subset \mathcal{O}_K$. Suppose $[\mathcal{O}_K : \mathbb{Z}(\alpha)] = m$ for some $m > 1$, so the cardinality of $\mathcal{O}_K/\mathbb{Z}(\alpha)$ equals m. Let $\beta_1, \beta_2, \ldots, \beta_m$ be coset representatives of $\mathcal{O}_K/\mathbb{Z}[\alpha]$. Every element of $\mathcal{O}_K$ can be written uniquely as $p(\alpha) + \beta_i$. Let $P = \mathbb{Z}[x]/\langle f(x)\rangle$ be a polynomial ring where $f(x)$ is the minimal polynomial of α. Our proposed sampling algorithm on $\mathcal{O}_K$ using coset representatives is shown in Algorithm 1.

Algorithm 1 Coset Representative Sampling on $\mathcal{O}_K$

Require: $\mathcal{O}_K$, a ring of integers of a number field $K = \mathbb{Q}(\alpha)$ with $\alpha \in \mathcal{O}_K$; $[\mathcal{O}_K : \mathbb{Z}(\alpha)] = m$ for some $m > 1$; $\beta_1, \beta_2, \ldots, \beta_m$ be coset representatives of $\mathcal{O}_K/\mathbb{Z}[\alpha]$; $P = \mathbb{Z}[x]/\langle f(x)\rangle$ be a polynomial ring where $f(x)$ is the minimal polynomial of α
Ensure: $p(\alpha) + \beta_i \in \mathcal{O}_K$
sample $p(\alpha) \in P$ according to PLWE sampling
sample β_i uniformly at random
return $p(\alpha) + \beta_i$

8.3.1 Statistical Distance

We wish to see how our sampling algorithm on $\mathcal{O}_K$ using coset representatives compares to RLWE sampling on $\mathcal{O}_K$. To do this, we calculate the statistical distance between these two probability distributions. Recall the definition for the statistical distance:

Definition 8.3.1. If P, Q are two probability distributions on the set S, then the **statistical distance** between P and Q is defined as

$$d(P,Q) = \frac{1}{2}\sum_{t\in S}|P(t) - Q(t)|.$$

Our goal is to bound $|D_{\Lambda_{\mathcal{O}_K}}(p(\alpha)+\beta) - D_{new}(p(\alpha)+\beta)|$, where $D_{\Lambda_{\mathcal{O}_K}}(p(\alpha)+\beta)$ is the probability of sampling $p(\alpha)+\beta$ according to RLWE sampling and $D_{new}(p(\alpha)+\beta)$ is the probability of sampling $p(\alpha)+\beta$ according to our proposed sampling. We will recall the various distributions that we have:

1. RLWE on $\mathcal{O}_K$: $D_{\Lambda_{\mathcal{O}_K}}(\theta(r)) = \dfrac{\exp\left(\frac{-||\theta(r)||^2}{2\sigma^2}\right)}{\sum_{y\in\Lambda_{\mathcal{O}_K}}\exp\left(\frac{-||y||^2}{2\sigma^2}\right)}$ for $r \in \mathcal{O}_K$,

2. RLWE on $\mathbb{Z}[\alpha]$: $D_{\Lambda_P}(\theta(p(\alpha))) = \dfrac{\exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right)}{\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2}{2\sigma^2}\right)}$ for $p(\alpha) \in \mathbb{Z}[\alpha]$,

3. PLWE on $\mathbb{Z}[\alpha]$: $D_{\mathbb{Z}}(p) = \dfrac{\exp\left(\frac{-||p||^2}{2\sigma^2}\right)}{\sum_{y\in\mathbb{Z}^n}\exp\left(\frac{-||y||^2}{2\sigma^2}\right)}$ for $p \in \mathbb{Z}[\alpha]$ viewed as a vector of its coefficients,

4. Our Sampling on $\mathcal{O}_K$: $D_{new}(p(\alpha)+\beta) = \dfrac{1}{m}\dfrac{\exp\left(\frac{-||p||^2}{2\sigma^2}\right)}{\sum_{y\in\mathbb{Z}^n}\exp\left(\frac{-||y||^2}{2\sigma^2}\right)}$ for $p(\alpha)+\beta \in \mathcal{O}_K$.

We first compare the RLWE distribution on $\mathcal{O}_K$ to $\frac{1}{m}$ times the RLWE distribution on $\mathbb{Z}[\alpha]$. Then we compare the RLWE distribution on $\mathbb{Z}[\alpha]$ with the PLWE distribution on $\mathbb{Z}[\alpha]$. Finally, we compare $\frac{1}{m}$ times the PLWE distribution on $\mathbb{Z}[\alpha]$ with our proposed sampling distribution on $\mathcal{O}_K$.

8.3.2 Compare RLWE on $\mathcal{O}_K$ and $\mathbb{Z}[\alpha]$

Theorem 8.3.2. *The statistical distance between the RLWE distribution with standard deviation σ on $\mathcal{O}_K$ and the RLWE distribution with standard deviation σ on*

$\mathbb{Z}[\alpha]$ *is bounded above by*

$$\frac{2^{2n+3}\sigma^{2n}\pi^{n/2}n!C(\beta)}{C_{\Lambda_P}C_\beta}$$

where n *is the degree of the number field* K, $C(\beta) = \max_i||\theta(\beta_i)||$, $C_{\Lambda_P} = \sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2(1+\gamma)}{2\sigma^2}\right)$, *and* $C_\beta = \sum_{i=1}^m \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)$.

Proof. First, notice that

$$\begin{aligned}\sum_{y\in\Lambda_{\mathcal{O}_K}}\exp\left(\frac{-||y||^2}{2\sigma^2}\right) &= \sum_{i=1}^m\sum_{z\in\Lambda_P}\exp\left(\frac{-||z+\theta(\beta_i)||^2}{2\sigma^2}\right)\\ &= \sum_{i=1}^m\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2-||\theta(\beta_i)||^2-2\sum_{j=1}^n z_j\theta(\beta_i)_j}{2\sigma^2}\right).\end{aligned} \tag{8.1}$$

By the Cauchy-Swartz and Young's inequality, we have

$$\begin{aligned}2\sum_{j=1}^n z_j\theta(\beta_i)_j = 2(z\cdot\theta(\beta_i)) \qquad &\text{where } \cdot \text{ is the dot product}\\ \leq \gamma||z||^2 + \frac{1}{\gamma}||\theta(\beta_i)||^2&\end{aligned}$$

for some $\gamma > 0$. Thus Equation 8.1 is bounded below by

$$\sum_{i=1}^m\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2-||\theta(\beta_i)||^2-(\gamma||z||^2+\frac{1}{\gamma}||\theta(\beta_i)||^2)}{2\sigma^2}\right).$$

By collecting common terms and rearranging, we get

$$\begin{aligned}&\sum_{i=1}^m\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2-||\theta(\beta_i)||^2-(\gamma||z||^2+\frac{1}{\gamma}||\theta(\beta_i)||^2)}{2\sigma^2}\right)\\ &\qquad= \sum_{i=1}^m\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2(1+\gamma)-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)\\ &\qquad= \sum_{i=1}^m\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2(1+\gamma)}{2\sigma^2}\right)\exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)\\ &\qquad= \sum_{i=1}^m\exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2(1+\gamma)}{2\sigma^2}\right).\end{aligned}$$

Now, we are ready to compare the RLWE distribution on $\mathcal{O}_K$ with $\frac{1}{m}$ times the RLWE distribution on $\mathbb{Z}[\alpha]$:

$$\begin{aligned}
&\left| D_{\Lambda_{\mathcal{O}_K}}(\theta(p(\alpha)+\beta)) - \frac{1}{m} D_{\Lambda_P}(\theta(p(\alpha))) \right| \\
&= \left| \frac{\exp\left(\frac{-||\theta(p(\alpha)+\beta)||^2}{2\sigma^2}\right)}{\sum_{y\in\Lambda_{\mathcal{O}_K}} \exp\left(\frac{-||y||^2}{2\sigma^2}\right)} - \frac{1}{m}\frac{\exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right)}{\sum_{z\in\Lambda_P} \exp\left(\frac{-||z||^2}{2\sigma^2}\right)} \right| \\
&\leq \left| \frac{\exp\left(\frac{-||\theta(p(\alpha)+\beta)||^2}{2\sigma^2}\right)}{\sum_{i=1}^{m} \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right) \sum_{z\in\Lambda_P} \exp\left(\frac{-||z||^2(1+\gamma)}{2\sigma^2}\right)} - \frac{1}{m}\frac{\exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right)}{\sum_{z\in\Lambda_P} \exp\left(\frac{-||z||^2}{2\sigma^2}\right)} \right| \\
&\leq \left| \frac{\exp\left(\frac{-||\theta(p(\alpha)+\beta)||^2}{2\sigma^2}\right)}{\sum_{i=1}^{m} \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right) \sum_{z\in\Lambda_P} \exp\left(\frac{-||z||^2(1+\gamma)}{2\sigma^2}\right)} - \frac{1}{m}\frac{\exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right)}{\sum_{z\in\Lambda_P} \exp\left(\frac{-||z||^2(1+\gamma)}{2\sigma^2}\right)} \right|
\end{aligned} \tag{8.2}$$

Let $C_{\Lambda_P} = \sum_{z\in\Lambda_P} \exp\left(\frac{-||z||^2(1+\gamma)}{2\sigma^2}\right)$. Also, notice that

$$\exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right) < 1$$

for $1 \leq i \leq m$ and so,

$$\sum_{i=1}^{m} \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right) < \sum_{i=1}^{m} 1 = m.$$

Let $C_\beta = \sum_{i=1}^{m} \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)$. Then we can rewrite and further bound Equation 8.2 as

$$\begin{aligned}
&\left| D_{\Lambda_{\mathcal{O}_K}}(\theta(p(\alpha)+\beta)) - \frac{1}{m} D_{\Lambda_P}(\theta(p(\alpha))) \right| \\
&\leq \frac{1}{C_{\Lambda_P}} \left| \frac{\exp\left(\frac{-||\theta(p(\alpha)+\beta)||^2}{2\sigma^2}\right)}{\sum_{i=1}^{m} \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)} - \frac{1}{m}\exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right) \right|
\end{aligned}$$

$$\leq \frac{1}{C_{\Lambda_P}} \left| \frac{\exp\left(\frac{-||\theta(p(\alpha)+\beta)||^2}{2\sigma^2}\right)}{\sum_{i=1}^{m} \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)} - \frac{\exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right)}{\sum_{i=1}^{m} \exp\left(\frac{-||\theta(\beta_i)||^2(1+\frac{1}{\gamma})}{2\sigma^2}\right)} \right|$$

$$= \frac{1}{C_{\Lambda_P} C_\beta} \left| \exp\left(\frac{-||\theta(p(\alpha)+\beta)||^2}{2\sigma^2}\right) - \exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right) \right|.$$

Let $a = \min\{||\theta(p(\alpha)+\beta)||^2, ||\theta(p(\alpha))||^2\}$ and
$b = \max\{||\theta(p(\alpha)+\beta)||^2, ||\theta(p(\alpha))||^2\}$. Using the Mean Value Theorem, we get the bound

$$\exp\left(\frac{-||\theta(p(\alpha)+\beta)||^2}{2\sigma^2}\right) - \exp\left(\frac{-||\theta(p(\alpha))||^2}{2\sigma^2}\right)$$
$$\leq \exp\left(\frac{-a}{2\sigma^2}\right) \frac{||\theta(p(\alpha)+\beta_i)||^2 - ||\theta(p(\alpha)||^2}{2\sigma^2}.$$

Using the difference of squares and triangle inequality, we also get the bound on the norms

$$|||\theta(p(\alpha)+\beta)||^2 - ||\theta(p(\alpha))||^2| \leq 2b|||\theta(p(\alpha)+\beta)|| - ||\theta(p(\alpha))|||$$
$$\leq 2b||\theta(\beta)||.$$

Let $r = \beta_r + p_r(\alpha) \in \beta_r + \mathbb{Z}[\alpha]$. Thus the statistical distance between the RLWE distribution on $\mathcal{O}_K$ and $\frac{1}{m}$ times the RLWE distribution on $\mathbb{Z}[\alpha]$ is bounded above by

$$\frac{1}{2C_{\Lambda_P} C_\beta} \sum_{r \in \mathcal{O}_K} \left| \exp\left(\frac{-a_r}{2\sigma^2}\right) \frac{b_r}{\sigma^2} ||\theta(\beta_r)|| \right| \tag{8.3}$$

where $a_r = \min\{||\theta(r)||^2, ||\theta(p_r(\alpha))||^2\}$ and $b = \max\{||\theta(r)||^2, ||\theta(p_r(\alpha))||^2\}$. To further bound the statistical distance, let $C(\beta) = \max_i ||\theta(\beta_i)||$. Then

$$||\theta(r) - \theta(p_r(\alpha))|| = ||\theta(\beta_r)|| \leq C(\beta).$$

So $a_r \geq ||\theta(r)|| - C(\beta)$ and $b_r \leq ||\theta(r)|| + C(\beta)$. For large enough $||\theta(r)||$, say $||\theta(r)|| \geq 2C(\beta)$, then $a_r \geq \frac{||\theta(r)||}{2}$ and $b_r \leq 2||\theta(r)||$. Then Equation 8.3 is bounded above by

$$\begin{aligned}
&\frac{1}{2C_{\Lambda_P}C_\beta}\sum_{r\in\mathcal{O}_K}\left|\exp\left(\frac{-||\theta(r)||}{4\sigma^2}\right)\frac{2||\theta(r)||}{\sigma^2}C(\beta)\right| \\
&\qquad = \frac{C(\beta)}{C_{\Lambda_P}C_\beta}\sum_{r\in\mathcal{O}_K}\left|\exp\left(\frac{-||\theta(r)||}{4\sigma^2}\right)\frac{||\theta(r)||}{\sigma^2}\right| \\
&\qquad \approx \frac{C(\beta)}{C_{\Lambda_P}C_\beta}\int_{\mathbb{R}^n}\exp\left(\frac{-||\theta(x)||}{4\sigma^2}\right)\frac{||\theta(x)||}{\sigma^2}dV \\
&\qquad = \frac{C(\beta)}{C_{\Lambda_P}C_\beta}Vol(S^{n-1})\int_0^\infty r^n\exp\left(\frac{-r}{4\sigma^2}\right)\frac{1}{\sigma^2}dr \\
&\qquad = \frac{C(\beta)Vol(S^{n-1})}{C_{\Lambda_P}C_\beta}\int_0^\infty (4\sigma^2u)^n\exp(-u)4du \qquad (r=4\sigma^2u) \\
&\qquad = \frac{C(\beta)}{C_{\Lambda_P}C_\beta}Vol(S^{n-1})4^{n+1}\sigma^{2n}\int_0^\infty u^n\exp(-u)du \\
&\qquad = \frac{C(\beta)}{C_{\Lambda_P}C_\beta}Vol(S^{n-1})4^{n+1}\sigma^{2n}\Gamma(n+1) \\
&\qquad = \frac{C(\beta)}{C_{\Lambda_P}C_\beta}Vol(S^{n-1})4^{n+1}\sigma^{2n}n! \\
&\qquad = \frac{2^{2n+3}\sigma^{2n}\pi^{n/2}n!C(\beta)}{C_{\Lambda_P}C_\beta}.
\end{aligned}$$

□

Remark 8.3.3. The statistical distance between the RLWE distribution on $\mathcal{O}_K$ and the RLWE distribution on $\mathbb{Z}[\alpha]$ is bounded above by a quantity that depends on the degree of the number field K. In practice, the degree n can be quite large, and so $n!$ is a huge constant. Thus the statistical distance bound above may suggest that the distributions on $\mathcal{O}_K$ and on $\mathbb{Z}[\alpha]$ are vastly different. However, our intuition suggests

otherwise. Notice the lattice $\theta(\mathcal{O}_K) = \cup_{i=1}^{m} \theta(\mathbb{Z}[\alpha]) + \theta(\beta_i)$ is a union of m translates of $\theta(\mathbb{Z}[\alpha])$. While we do expect the distance to be bounded by a function depending on $C(\beta)$, we do not expect the distance to increase with n. Therefore, we expect that one can find better estimates than the ones stated in the proof to show that these two distributions are statistically close.

8.3.3 Compare RLWE and PLWE on $\mathbb{Z}[\alpha]$

We now compare the RLWE distribution on $\mathbb{Z}[\alpha]$ with the PLWE distribution on $\mathbb{Z}[\alpha]$.

Theorem 8.3.4. *Let $M : P \to \theta(\mathcal{O}_K)$ be a change of basis matrix where P is a polynomial ring and $\mathcal{O}_K$ a ring of integers of a number field K. The statistical distance between the RLWE distribution with standard deviation $||M||\sigma$ on $\mathbb{Z}[\alpha]$ and $\frac{1}{m}$ times the PLWE distribution with standard deviation σ on $\mathbb{Z}[\alpha]$ where $m = [\mathcal{O}_K : \mathbb{Z}[\alpha]]$ is zero, showing that these two distributions are equivalent.*

Proof. First, we notice that $\theta(p) = M \cdot p$ where M is the change of basis matrix from PLWE to RLWE and $p \in \mathbb{Z}[\alpha]$ is a vector of its coefficients. Also, note that

$$\begin{aligned}
\sum_{y \in \mathbb{Z}^n} \exp\left(\frac{-||y||^2}{2\sigma^2}\right) &= \sum_{y \in \mathbb{Z}^n} \exp\left(\frac{-||M||^2||y||^2}{2(||M||\sigma)^2}\right) \\
&\geq \sum_{y \in \mathbb{Z}^n} \exp\left(\frac{-||M \cdot y||^2}{2(||M||\sigma)^2}\right) \\
&= \sum_{y \in \mathbb{Z}^n} \exp\left(\frac{-||\theta(y)||^2}{2(||M||\sigma)^2}\right) \\
&= \sum_{z \in \Lambda_P} \exp\left(\frac{-||z||^2}{2(||M||\sigma)^2}\right).
\end{aligned}$$

The spectral norm $||M|| = t$ is the largest singular value of M. If M is a unitary matrix, then $||M^{-1}|| = ||M||^{-1} = \frac{1}{t}$.

Then the statistical distance between the RLWE distribution on $\mathbb{Z}[\alpha]$ with

standard deviation $t\sigma$ and the PLWE distribution on $\mathbb{Z}[\alpha]$ with standard deviation σ is

$$\begin{aligned}
&\frac{1}{2}\sum_{p\in\mathbb{Z}[\alpha]}\left|\frac{\exp\left(\frac{-||\theta(p)||^2}{2(t\sigma)^2}\right)}{\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2}{2(t\sigma)^2}\right)}-\frac{\exp\left(\frac{-||p||^2}{2\sigma^2}\right)}{\sum_{y\in\mathbb{Z}^n}\exp\left(\frac{-||y||^2}{2\sigma^2}\right)}\right| \\
&\quad=\frac{1}{2}\sum_{p\in\mathbb{Z}[\alpha]}\left|\frac{\exp\left(\frac{-||\theta(p)||^2}{2(t\sigma)^2}\right)}{\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2}{2(t\sigma)^2}\right)}-\frac{\exp\left(\frac{-||M^{-1}\cdot\theta(p)||^2}{2\sigma^2}\right)}{\sum_{y\in\mathbb{Z}^n}\exp\left(\frac{-||y||^2}{2\sigma^2}\right)}\right| \\
&\quad\leq\frac{1}{2}\sum_{p\in\mathbb{Z}[\alpha]}\left|\frac{\exp\left(\frac{-||\theta(p)||^2}{2(t\sigma)^2}\right)}{\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2}{2(t\sigma)^2}\right)}-\frac{\exp\left(\frac{-||M^{-1}||^2||\theta(p)||^2}{2\sigma^2}\right)}{\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2}{2(||M||\sigma)^2}\right)}\right| \\
&\quad=\frac{1}{2}\sum_{p\in\mathbb{Z}[\alpha]}\left|\frac{\exp\left(\frac{-||\theta(p)||^2}{2(t\sigma)^2}\right)}{\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2}{2(t\sigma)^2}\right)}-\frac{\exp\left(\frac{-||\theta(p)||^2}{2(t\sigma)^2}\right)}{\sum_{z\in\Lambda_P}\exp\left(\frac{-||z||^2}{2(t\sigma)^2}\right)}\right| \\
&\quad=\frac{1}{2}\sum_{p\in\mathbb{Z}[\alpha]}|0|=0,
\end{aligned}$$

showing that these two distributions are equivalent.

□

8.3.4 Compare PLWE on $\mathbb{Z}[\alpha]$ and Our Sampling

Finally, we compare $\frac{1}{m}$ times the probability of sampling with PLWE on $\mathbb{Z}[\alpha]$ with our proposed sampling algorithm on $\mathcal{O}_K$ using coset representatives.

Theorem 8.3.5. *The statistical distance between $\frac{1}{m}$ times the PLWE distribution on $\mathbb{Z}[\alpha]$ and our proposed sampling algorithm on $\mathcal{O}_K$ is zero.*

Proof. Let $r=\beta_r+p_r(\alpha)\in\beta_r+\mathbb{Z}[\alpha]=\mathcal{O}_K$ for some coset representative β_r. The statistical distance between the two probability distributions is

$$\frac{1}{2}\sum_{r\in\mathcal{O}_K}\left|\frac{1}{m}D_{\mathbb{Z}}(p_r(\alpha))-D_{new}(p_r(\alpha)+\beta_r)\right|$$

$$= \frac{1}{2} \sum_{r \in \mathcal{O}_K} \left| \frac{1}{m} D_{\mathbb{Z}}(p_r(\alpha)) - \frac{1}{m} D_{\mathbb{Z}}(p_r(\alpha)) \right| = 0.$$

This shows that the distributions are equivalent.

□

8.4 Conclusions

Combining the results from Theorem 8.3.2, Theorem 8.3.4, and Theorem 8.3.5, we have proved that the statistical distance between the RLWE distribution on $\mathcal{O}_K$ and our sampling algorithm on $\mathcal{O}_K$ using coset representatives is bounded above by a quantity that depends on the degree n of a number field K. While this quantity is showing to be large, we suggest in Remark 8.3.3 that better estimates can be made to reduce this bound. By reducing this bound, one could show that our proposed PLWE sampling algorithm is statistically close to the RLWE sampling distribution. Thus Algorithm 1 is an interesting PLWE sampling algorithm which could be used to expand sampling for non-monogenic number fields and warrants further study.

Chapter 9

Future Work

With post-quantum cryptography attracting much interest, there are a variety of directions this area of research can go from here. We want to continue studying the security parameter bounds for more accuracy in predicting vulnerable instances of the RLWE problems under the Chi-Squared Attack. Knowing that there is a trade-off between the prime and standard deviation, we want to find a bound for security in terms of these two variables. We also want to find better estimates to bound the statistical distance between the RLWE distribution on $\mathcal{O}_K$ and the RLWE distribution on $\mathbb{Z}[\alpha]$. Upon completing these calculations, we want to implement the coset representative sampling algorithm to expand the PLWE sampling to non-monogenic number fields.

Bibliography

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. *Annual International Cryptology Conference*, 2009.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. *International Colloquium on Automata, Languages and Programming*, 2011.

[CIV16] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of Ring-LWE revisited. In *Advances in cryptology—EUROCRYPT 2016. Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 147–167. Springer, Berlin, 2016.

[CLS17a] Hao Chen, Kristin Lauter, and Katherine Stange. Attacks on search RLWE problem with small errors. *SIAM Journal on Applied Algebra and Geometry*, 1(1), 2017.

[CLS17b] Hao Chen, Kristin Lauter, and Katherine Stange. Security considerations for Galois non-dual RLWE families. In *Selected areas in cryptography—SAC 2016*, volume 10532 of *Lecture Notes in Computer Science*, pages 443–462. Springer, Cham, 2017.

[Con09] K. Conrad. The different ideal. 2009. https://kconrad.math.uconn.edu/blurbs/gradnumthy/different.pdf.

[Con12] K. Conrad. Factoring after Dedekind, 2012. https://kconrad.math.uconn.edu/blurbs/gradnumthy/dedekindf.pdf.

[Ded78] R. Dedekind. Ueber den Zusammenhang zwischen der Theorie der ideale und der Theorie der höheren Congruenzen. *Abhandlungen der Königlichen Gesellschaft der Wissenschaften in Göttingen*, 23:3–38, 1878.

[EHL14] Kirsten Eisenträger, Sean Hallgren, and Kristin Lauter. Weak instances of PLWE. In Antoine Joux and Amr Youssef, editors, *Selected Areas in Cryptography – SAC 2014*, pages 183–194, Cham, 2014. Springer International Publishing.

[ELOS15] Yara Elias, Kristin Lauter, Ekin Ozman, and Katherine Stange. Provably weak instances of Ring-LWE. In *Advances in cryptology—CRYPTO 2015. Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 63–92. Springer, Heidelberg, 2015.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 197–206, New York, NY, USA, 2008. Association for Computing Machinery.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Advances in cryptology—EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Comput. Sci.*, pages 1–23. Springer, Berlin, 2010.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 35–54, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 700–718, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. *Symposium on the Theory of Computing*, 2009.

[Pei16a] Chris Peikert. A decade of lattice cryptography. *Foundations and Trends in Theoretical Computer Science*, 2016.

[Pei16b] Chris Peikert. How (not) to instantiate Ring-LWE. In *Security and cryptography for networks*, volume 9841 of *Lecture Notes in Comput. Sci.*, pages 411–430. Springer, Cham, 2016.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), sep 2009.

[Reg10] Oded Regev. The learning with errors problem (invited survey). *2010 IEEE 25th Annual Conference on Computational Complexity*, 2010.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, oct 1997.

[Ver12] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *Compressed Sensing*, 2012.

Appendix A

Change of Basis Matrix

We adapted the change of basis matrix code within the Chi-Squared Attack (see Appendix B) from [ELOS15] to go from PLWE to dual RLWE. This first block of code, Listing A.1, uses the standard canonical embedding. This is followed by the change of basis matrix, using the adjusted canonical embedding (see Listing A.2).

Listing A.1: Change of Basis Matrix

```
1
2  def change_of_basis_matrix_PLWE_to_dual(f):
3      """
4      Returns a matrix M_alpha^\vee to go from PLWE to dual RLWE whose columns
5      are {theta(gamma^(-1)alpha^i)} where gamma = generator of the different
6      ideal and alpha = generator of basis of ring of integers of the number
7      field with defining polynomial f
8
9      INPUT:
10     f - a polynomial with coefficients in QQ
11
12     OUTPUT:
13     M - a matrix with entries in the real field RealField(prec)
14     where prec = 300
15     key - a tuple containing labels determining what kind of embeddings:
16     0 = real embeddings,
17     1 = complex embeddings,
18     2 = pair of complex embedding (its conjugate)
19
20     NOTE:
21     This depends on Sage outputting the complex embeddings in such a way
22     that the conjugate of a strictly complex embedding follows the embedding
23
24     """
25
```

```
    # N is a number field defined by minimal polynomial f.
    #fdeg is the degree of the min polynomial
    N.<a> = NumberField(f)
    fdeg = f.degree()
    f_ = derivative(f)
    ff = fast_callable(f_, vars=[x])
    gen= ff(a)
    key = [0 for i in range(fdeg)]  #initiate the key array
    embs = N.embeddings(CC) # same as N.complex_embeddings()
    M = matrix(RealField(300),fdeg,fdeg)
    #initiate the nxn matrix where n = degree of polynomial f
    #(as well as degree of number field)

    #Reminder: for range loop starts at index 0 and will end with fdeg-1
    apows = [a^j for j in range(fdeg)]
    #a is the generator of the Number Field

    # this loop will now fill in the matrix where the i-th column is
    #theta(gamma^(-1)a^i) where theta is the canonical embedding and
    #gamma is the generator of the different ideal of N
    i = 0
    while i < fdeg:
        em = embs[i]

        # output to assure the user that the program is not stuck in an
        #infinite loop
        #if Mod(i,20)==Mod(0,20) or Mod(i,20)==Mod(1,20):
            #print("Embedding matrix: ", i, " rows out of ", fdeg,
            #" complete.")

        # test whether the embedding is real or complex.

        # if block - real embedding: imag() returns the imaginary part
        #of the embedding of a
        if em(a).imag() == 0:
            key[i] = 0  #denotes a real embedding
            for j in range(fdeg):
                M[i,j] = em((1/gen)*apows[j]).real()
            i = i + 1
        # else block - complex embedding
        else:
            key[i] = 1
            key[i+1] = 2 #the following embedding is the complex conjugate
            for j in range(fdeg):
                M[i,j] = (em((1/gen)*apows[j])).real()
                M[i+1,j] = ((em((1/gen)*apows[j]))*CC.gen()).real()
            i = i + 2
    return M, key, embs
M, key, embs = change_of_basis_matrix_PLWE_to_dual(f)
M_inv = M.inverse()  #this gives us the matrix to go from dual to PLWE
M_inv
```

Listing A.2: Adjusted Change of Basis Matrix

```
1
2  def adjusted_change_of_basis_matrix_PLWE_to_dual(f):
3      """
4      Returns a matrix M_alpha^\vee that goes from PLWE to dual RLWE whose
5      columns are {theta(gamma^(-1)alpha^i)} where gamma = generator of the
6      different ideal and alpha = generator of basis of ring of integers of the
7      number field with defining polynomial f
8
9      INPUT:
10     f - a polynomial with coefficients in QQ
11
12     OUTPUT:
13     M - a matrix with entries in the real field RealField(prec)
14     where prec = 300
15     key - a tuple containing labels determining what kind of embeddings:
16     0 = real embeddings,
17     1 = complex embeddings,
18     2 = pair of complex embedding (its conjugate)
19
20     NOTE:
21     This depends on Sage outputting the complex embeddings in such a way
22     that the conjugate of a strictly complex embedding follows the embedding
23
24     """
25
26     # N is a number field defined by minimal polynomial f.
27     #fdeg is the degree of the min polynomial
28     N.<a> = NumberField(f)
29     fdeg = f.degree()
30     f_ = derivative(f)
31     ff = fast_callable(f_, vars=[x])
32     gen= ff(a)
33     key = [0 for i in range(fdeg)]  #initiate the key array
34     embs = N.embeddings(CC) # same as N.complex_embeddings()
35     M = matrix(RealField(300),fdeg,fdeg)
36     #initiate the nxn matrix
37     #where n = degree of polynomial f
38
39     #Reminder: for range loop starts at index 0 and will end with fdeg-1
40     apows = [a^j for j in range(fdeg)]
41     #a is the generator of the Number Field
42
43     # this loop will now fill in the matrix where the i-th column is
44     #theta(gamma^(-1)a^i) where theta is the canonical embedding and gamma is
45     #the generator of the different ideal of N
46     i = 0
47     while i < fdeg:
48         em = embs[i]
49
50         # output to assure the user that the program is not stuck in
51         #an infinite loop
52         #if Mod(i,20)==Mod(0,20) or Mod(i,20)==Mod(1,20):
53             #print("Embedding matrix: ", i, " rows out of ", fdeg,
```

```
            # " complete.")

        # test whether the embedding is real or complex.

        # if block - real embedding: imag() returns the imaginary part
        # of the embedding of a
        if em(a).imag() == 0:
            key[i] = 0  #denotes a real embedding
            for j in range(fdeg):
                M[i,j] = em((1/gen)*apows[j]).real()
            i = i + 1
        # else block - complex embedding
        else:
            key[i] = 1
            key[i+1] = 2 #the following embedding is the complex conjugate
            for j in range(fdeg):
                M[i,j] = sqrt(2)*(em((1/gen)*apows[j])).real()
                M[i+1,j] = sqrt(2)*((em((1/gen)*apows[j]))*CC.gen()).real()
            i = i + 2
    return M, key, embs
AD_M, key, embs = adjusted_change_of_basis_matrix_PLWE_to_dual(f)
AD_M_inv = AD_M.inverse()  #this gives us the matrix to go from dual to PLWE
AD_M_inv
```

Appendix B

Chi-Squared Attack

This code is a slight update and adaptation of [ELOS15] done by Alec Critten and myself.

Listing B.1: Chi-Squared Attack

```
1
2
3  def change_of_basis_matrix(f):
4      """
5      Returns a matrix, columns basis 1, x, x^2, x^3, ... given in the
6      canonical embedding (aka the Minkowski embedding) - this matrix is the
7      change of basis matrix to go from PLWE to non-dual RLWE
8
9      INPUT:
10     f - a polynomial with coefficients in QQ
11
12     OUTPUT:
13     M - a matrix with entries in the real field RealField(300) where 300 is
14     the precision key - a tuple containing the signature of real embeddings
15     and pairs of complex embeddings
16
17     NOTE:
18     This depends on Sage outputting the complex embeddings in such a way
19     that the conjugate of a strictly complex embedding follows the embedding
20
21     """
22
23     N.<a> = NumberField(f)
24     n = f.degree()
25
26     # construct a square n*n matrix M, where n is the degree of f
27     # for key - a 0 entry indicates a real embedding; 1 and 2 respectively
28     indicate the real and complex parts of an imaginary embedding
29     key = [0 for i in range(n)]
```

```
    embs = N.embeddings(CC) # same as N.complex_embeddings()
    M = matrix(RealField(300),n,n)

    #print("Preparing an embedding matrix:  computing powers of the root.")
    apows = [ a^j for j in range(n) ] #a is the generator of the number field
    #print("Finished computing the powers of the root.")
    i = 0

    while i < n:
        em = embs[i]

        # test whether the embedding is real or complex.
        # if block - real embedding
        if em(a).imag() == 0:
            key[i] = 0
            for j in range(n):
                M[i,j] = em(apows[j]).real()
            i = i + 1
        # else block - complex embedding
        else:
            key[i] = 1
            key[i+1] = 2 #assumes the next embedding is the complex conjugate
            for j in range(n):
                M[i,j] = em(apows[j]).real()
                M[i+1,j] = (em(apows[j])*CC.gen()).real()
                #gives different result using .imag() because of the ordering
                #sage uses on the embeddings
            i = i + 2
    return M, key


def random_vec(q, n):
    """
    Produces a random vector from (Z/qZ)^n.

    INPUT:
    q - an integer modulus
    n - the dimension of (Z/qZ)^n

    OUTPUT:
    - a random vector from (Z/qZ)^n
    """
    return vector([ZZ.random_element(0,q) for i in range(n)])


def modq(r,q):
    """
    Computes the result of r modulo q, for r a real number and q an integer.

    INPUT:
    r - a real number
    q - an integer modulus
```

```
84
85      OUTPUT:
86      t*q - the real number in [0,q) such that (r - t*q) = k*q for k an integer
87      """
88      t = r/q - floor(r/q)
89      return t*q
90
91
92
93  def call_sampler(f, sig):
94      """
95      Sets up the discrete Gaussian sampler.
96
97      INPUT:
98      f - the polynomial
99      sig - sigma for the distribution
100
101     OUTPUT:
102     sample - function which will return vectors in the discrete gaussian
103     distribution over change of basis matrix with entries in QQ and sigma
104     """
105
106     cm,key = change_of_basis_matrix(f)
107     # entries of matrix are in RealField with precision 10
108     cm53 = cm.change_ring(RealField(10))
109     cmqq = cm53.change_ring(QQ)
110     sampler = DiscreteGaussianDistributionLatticeSampler(cmqq.transpose(),
111     sig)
112     # transpose is due the Sage code convention of assuming lattice
113     #vectors are defined by rows
114     return sampler()
115
116
117
118
119 def get_sample(cm, secret, q, key, f, sig, S):
120     """
121     Creates a sample [a,b] using a Gaussian sampler on a given lattice,
122     secret, and prime.
123
124     INPUT:
125     cm - a given lattice as a matrix
126     secret - the secret
127     q - the prime modulus q
128     key - the "key" as before in change_of_basis() - signature of embeddings
129     f - polynomial function
130     sig - sigma
131     S - polynomial ring
132
133     OUTPUT:
134     [a,b] - a sample pair
135     """
136
137
```

```
    cm_inv = cm.inverse()
    # e is an error in the real vector space R^n
    e = call_sampler(f, sig).change_ring(RealField(300))
    dim = cm.dimensions()[0] # detect dimension of lattice
    pre_a = random_vec(q, dim) #a random vector from (Z/qZ)^n
    a = cm*pre_a # create a, in R^n
    b = vecmul_poly(a,secret,cm, S) + e # create b, in R^n
    pre_b = cm_inv*b # move to basis in cm in order to reduce mod q
    pre_b_red = vector([modq(c,q) for c in pre_b]) #modding q
    b = cm*pre_b_red
    return [a, b]


def setup_rings(f,q):

    """
    Sets up the parameters for attack.

    INPUT:
    f- the polynomial function
    q - the prime modulus q

    OUTPUT:
    R - polynomial ring with variable x
    S - polynomial ring with variable z
    """

    R.<x> = PolynomialRing(IntegerModRing(q)) #this is P in Elias 2015
    S.<z> = R.quotient(f)  #this is P_q in Elias 2015
    #print("Setting up parameters, polynomial = ", f, " and prime = ",
    #q, " and sigma = ", sig)
    #print("Verifying properties:  ")
    #print("Prime?", q.is_prime())
    #print("Irreducible? ", f.is_irreducible())
    #print("Value at 1 modulo q?", Mod(f.subs(y=1),q))
    return R, S


def prepare_matrices(polyonly, f):
    """
    Computes the lattices in the Minkowski space (determined by the
    change_of_basis_matrix function)

    INPUT:
    polyonly - a Boolean value that determines whether the attack is a
    Poly-LWE attack (true) or a Ring-LWE attack (false)
    f - polynomial

    OUTPUT:
    cmqq - the change of basis matrix with rational approximations
    """

```

```
192     #print("Preparing matrices.")
193
194     # if the key is changed in this function, we'll set this value to true
195     returnKey = False
196
197     # If the attack is Poly-LWE
198     if polyonly:
199         # already thinking in terms of polynomial coefficients
200         cm = matrix.identity(f.degree())
201         key = 0
202     # If the attack is Ring-LWE
203     else:
204         #change of basis matrix goes from polynomial coefficients
205         #to lattice entries
206         cm,key = change_of_basis_matrix(f)
207         returnKey = True
208     #print("Embedding matrix prepared.")
209     #print("Inverse matrix found.")
210
211     if polyonly:
212         cmqq = cm.change_ring(ZZ)
213     else:
214         # in RLWE case, cm has real entries. provide rational approximation
215         #as in previous function
216         cm53 = cm.change_ring(RealField(10))
217         cmqq = cm53.change_ring(QQ)
218     #print("All matrices prepared.")
219
220     if returnKey == True:
221         return cm, cmqq, key
222     else:
223         return cm, cmqq, key
224
225
226
227
228 def make_poly(u,cm,S):
229     """
230     Constructs a polynomial directly from a vector in the real vector
231     space R^n, given a change of basis matrix and a variable from polynomial
232     ring.
233
234     INPUT:
235     u - the vector in R^n
236     cm - a change of basis matrix
237     S - polynomial ring with variable to use to make polynomial
238
239     OUTPUT:
240     pol - a polynomial with integer coefficients and variable determined by S
241     """
242
243     #coefficients of the polynomial are given by the change of basis matrix
244     coeffs = cm*u
245     pol = 0 #initialize a polynomial
```

```
246     for i in range(len(coeffs)):
247         pol = pol + ZZ(round(coeffs[i]))*S.gen()^i
248         # S.gen() determines what poly ring the polynomial lives in and
249         #its variable
250     return pol
251
252
253
254 def make_vec(f,cm):
255     """
256     Constructs a vector in the Minkowski space directly from a polynomial
257
258     INPUT:
259     f - polynomial function in polynomial ring S
260     cm - change of basis matrix
261
262     OUTPUT:
263     - the vector
264     """
265
266     if f == 0:
267         coeffs = [0 for i in range(cm.ncols())]
268         #function is identically zero so all coefficients are zero
269     else:
270         coeffs = [0 for i in range(cm.ncols())]
271         #initiates vector of coefficients as all zeros
272         colist = lift(f).coefficients() #lifts f into polynomial ring S
273         for i in range(len(colist)):
274             coeffs[i] = ZZ(colist[i])
275     return cm*vector(coeffs)
276
277
278
279 def vecmul_poly(u,v,cm, S):
280     """
281     Multiplies two vectors in the Minkowski space using their forms in
282     the polynomial ring.
283
284     INPUT:
285     u & v - the vectors to be multiplied together
286     cm - change of basis matrix
287
288     OUTPUT:
289     - the result of the multiplication (passed from the function which
290     constructs the resultant vector)
291     """
292
293     # Convert vectors into polynomials
294     # z is the variable of the polynomial ring
295     # S.<z> = R.quotient(f) = P_q in Elias 2015
296     poly_u = make_poly(u,cm.inverse(),S)
297     poly_v = make_poly(v,cm.inverse(),S)
298
299     # Multiply the polynomials and return the result in vector form
```

```
300     poly_prod = poly_u*poly_v
301     return make_vec(poly_prod,cm)
302
303
304
305 def initiate_sampler(sig, cmqq):
306     """
307     Creates the sampler on the lattice embedded in R^n or ZZ^n.
308
309     INPUT:
310     sig - the sigma of the distribution
311     cmqq - canonical matrix with rational entries
312
313     OUTPUT:
314     sampler - discrete gaussian distribution
315     """
316
317     #print("Initiating Sampler.")
318     sampler =
319     DiscreteGaussianDistributionLatticeSampler(cmqq.transpose(), sig)
320     #print("Sampler initiated with sigma", RDF(sig))
321     return sampler
322
323
324
325 def error_test(num, sampler, n):
326     """
327     This function produces some sample error vectors, so that the
328     user can verify that the sampler is working properly.
329
330     INPUT:
331     num - the number of error vectors to be produced
332     sampler
333     n - degree of f
334
335     OUTPUT:
336     none
337     """
338     #print("Testing the error vector production by producing ", num,
339     #" errors.")
340
341     # Construct a tuple of errors
342     errorlist = [sampler().norm().n() for i in range(num)]
343
344     # Compute and print avg. and max. norms
345     meannorm = mean(errorlist) # average norm
346     maxnorm = max(errorlist) # maximum norm
347     #print("The average error norm is ",
348     #RDF(meannorm/( sqrt(n)*sampler.sigma*sqrt(2*pi) )), " times sqrt(n)*s.")
349     maxratio = RDF(maxnorm/( sqrt(n)*sampler.sigma*sqrt(2*pi) ))
350     #print("The maximum error norm is ", maxratio, " times sqrt(n)*s.")
351
352     #if maxratio > 1:
353         #print("~~~~~~~~~~~~~~~~~~~~~~~ ERROR ~~~~~~~~~~~~~~~~~~~~~~~~")
```

```
354         #print("The errors do not satisfy a proven upper bound in norm.")
355     return
356
357
358
359 def create_secret(cm,q, n):
360     """
361     Creates the secret.
362
363     INPUT:
364     cm - change of basis matrix
365     q - prime
366     n - degree of polynomial f
367
368     OUTPUT:
369     secret - the secret
370     """
371
372     secret = cm*random_vec(q,n)
373     return secret
374
375
376
377 def create_samples(num_of_samples,cm, secret, q, key, f, sig, S):
378     """
379     Creates a list of samples from the sample distribution.
380
381     INPUT:
382     num_of_samples - the number of samples to be created
383     cm - canonical matrix
384     secret - the secret in the RLWE pronlem
385     q - prime
386     key - the signature of the embeddings
387     f - the polynomial function
388     sig - sigma
389     S - polynomial ring
390
391     OUTPUT:
392     samps - the samples
393     """
394
395     # We store the samples in a list
396     samps = []
397     #print("Creating samples")
398
399     # Create the given number of samples
400     for i in range(num_of_samples):
401         #print("Creating sample number ", i)
402         samp = get_sample(cm, secret, q, key, f, sig, S)
403         samps.append(samp)
404
405
406     #print("Done creating ", num_of_samples, "samples.")
407     return samps
```

```
408
409
410
411 def go_to_q(u,cm,q, R):
412     """
413     Converts a vector into an integer modulo q.
414
415     INPUT:
416     u - vector in the Minkowski space (R^n)
417     cm - change of basis matrix
418     q - modulus of the polynomial ring
419     R - the polynomial ring
420
421     OUTPUT:
422     the integer value of pol_eval modded by q
423     """
424
425     pol = make_poly(u,cm,R)
426     #print(cm)
427     #print("pol: ", pol)
428     pol_eval = pol.subs(1)
429     #print("pol_eval: ", pol_eval)
430     #print("q: ", q)
431     return IntegerModRing(q)(pol_eval)
432
433
434
435
436
437
438
439 #-----------------------------------------------#
440 #ATTACK-SPECIFIC FUNCTIONS                      #
441 #-----------------------------------------------#
442
443
444 # Check to make sure moving to q preserves product - the last two lines equal
445 def sanity_check(f,q, S, R):  #do we care about the result of this??
446     cm, key = change_of_basis_matrix(f)
447     #print("Initiating sanity check")
448     cm_inv = cm.inverse()
449     pvec1 = random_vec(q,f.degree())
450     vec1 = cm*pvec1
451     pvec2 = random_vec(q,f.degree())
452     vec2 = cm*pvec2
453     #print("vec1: ", vec1)
454     #print("vec2: ", vec2)
455     #print("q:", q)
456     vprod2 = vecmul_poly(vec1,vec2,cm, S)
457     first_thing = go_to_q(vprod2,cm_inv, q, R)
458     second_thing = go_to_q(vec1,cm_inv, q, R)*go_to_q(vec2,cm_inv,q, R)
459     #if first_thing == second_thing:
460      #   print("Sanity confirmed.")
461     #else:
```

```
462      #   print("~~~~~~~~~~~~~~~~~~~~~~ ERROR ~~~~~~~~~~~~~~~~~~~~~~~~")
463          #print("Sanity problem:", first_thing, " is not equal to ",
464          #second_thing, ".")
465         #print("Are you sure your ring has root 1 mod q?")
466     return True
467
468
469
470 # Algorithm 2
471 # reportrate controls how often it updates the status of the loop;
472 #larger = less frequently
473 # quickflag = True will run only the secret and
474 #a few other values to give a quick idea if it works
475 def alg2(samps, reportrate, cm_inv, q, lift_secret, S, quickflag = False):
476     #print("Beginning new algorithm.", type(cminv))
477
478     T = RealDistribution('chisquared', q-1)
479     alpha = 1- 1/(10*q)
480     delta = T.cum_distribution_function_inv(alpha)
481
482     num_of_samps = len(samps)
483     a = [ 0 for i in range(num_of_samps)]
484     b = [ 0 for i in range(num_of_samps)]
485     #print("Moving samples to F_q.")
486     for i in range(num_of_samps):
487         sample = samps[i]
488         a[i] = go_to_q(sample[0],cm_inv,q, S)
489         b[i] = go_to_q(sample[1],cm_inv,q, S)
490     possibles = []
491     winner = [[],0]
492     #print("Samples have been moved to F_q.")
493     #for i in range(2):
494         #if i == 0:
495             #print("!!!!! ROUND 1: !!!!! First, checking how many samples
496             #the secret survives (peeking ahead).")
497             #iterat = [lift_secret]
498         #if i == 1:
499             #print("!!!!! ROUND 2: !!!!! Now, running the attack naively.")
500     possibles = []
501     if quickflag:
502         #print("We are doing it quickly (not a full test).")
503         iterat = range(1000)
504     else:
505         iterat = range(q)
506     import collections
507     for g in iterat:
508         #if Mod(g,reportrate) == Mod(0,reportrate):
509             #print("Currently checking residue ", g)
510         g = IntegerModRing(q)(g)
511         error_list = [] #curly E from paper
512         for i in range(num_of_samps):
513             error_list.append(b[i]-a[i]*g)
514         counts = collections.Counter(error_list)
515         chi_squared = sum([(counts[j]-num_of_samps/q)^2*q/num_of_samps
```

```
        for j in range(q)])
        if chi_squared > delta:
            possibles.append(g)
        #potential = True
        #ctr = 0
        #while ctr < num_of_samps and potential:
        #    e = abs(lift(IntegerModRing(q)(b[ctr]-g*a[ctr])))
        #    if e > q/4 and e < 3*q/4: #change condition to a chi-square test
                #potential = False
        #        if ctr == winner[1]:
        #            winner[0].append(g)
        #            print("We have a new tie for longest chain:", g,
        #           " has survived ", ctr, " rounds.")
        #        if ctr > winner[1]:
        #            winner = [[g],ctr]
        #            print("We have a new longest chain of samples
        #           survived:", g, " has survived ", ctr, " rounds.")
        #    ctr = ctr + 1
        #if potential == True:
        #    print("We found a potential secret: ", g)
        #    possibles.append(g)
        #if g == lift_secret:
        #    if i == 0:
        #        print("The real secret survived ", ctr, "samples.")
                #break
    #print("Full list of survivors of the ", numsamps, " samples:",
    #possibles)
    #print("The real secret mod q was: ", lift_secret)
    #print("possibles:", len(possibles))
    if len(possibles) == 1 and possibles[0] == lift_secret:
        #print("Success! You found the secret.")
        return True
    else:
        #print("Failure! You did not find the secret.")
        return False

# Run a simulation.
def shebang(f,q,sig,num_of_samples,num_of_trials,quickflag=False,
            polyonly=False):

    n = f.degree()
    N.<a> = NumberField(f)
    #if polyonly:
        #print("Welcome to the Poly-LWE Attack.")
    #else:
        #print("Welcome to the Ring-LWE Attack.")
        #print("The attack should theoretically work if the following
        #quantity is greater than 1.")
        #print("Quantity: ",
        #RDF( q/( 2*sqrt(2)*sqrt(2*pi)*sig*n*(q-1)^((n-1)/2/n))))
    #timer = Timer()
    #timer2 = Timer()
    #timer.start()
    #print("********** PHASE 1: SETTING UP SYSTEM ")
```

```
    R, S = setup_rings(f,q)
    #removed calculating sig here and added it as an input of function
    cm, cmqq, key = prepare_matrices(polyonly, f)
    #for polyonly = true the two matrices are just identity matrices;
    #for polyonly = false we use the change_of_basis_matrix function
    if not polyonly:
        #print("Computing the adjustment factor for s.")
        cembs = (n - len(N.embeddings(RR)))/2
        detscale =
            RealField(300)((2^(-cembs)*sqrt(abs(f.discriminant())))^(1/n))
        # adjust the sigma and sval by this value
        #sval = sval*detscale  #sval is the width of the Gaussian
        sig = sig*detscale   #WHY are we scaling this?
        #print("Adjusted s for use with this embedding, result is ", sval)
    sampler = initiate_sampler(sig, cmqq)
    #print("The sampler has been created with sigma = ", RDF(sampler.sigma))
    #print("Sampled vectors will have expected norm ",
    #RDF(sqrt(n)*sampler.sigma))
    error_test(5, sampler, f.degree())
    #print("Time for Phase 1: ", timer.stop())
    #timer.start()
    count_successes = 0
    #timer2.start()
    for trialnum in range(num_of_trials):
        #print("*~*~*~*~*~*~*~*~* TRIAL NUMBER ", trialnum, "*~*~*~*~*~*~*~*~*")
        #print("********** PHASE 2: CREATE SECRET AND SAMPLES")
        secret = create_secret(cm,q, n)
        samps = create_samples(num_of_samples,cm, secret, q, key, f, sig, S)
        sanity_check(f,q, S, R)
        #print("Time for Phase 2: ", timer.stop())
        #timer.start()
        #print("********** PHASE 3: HISTOGRAMS")
        #histogram_of_errors(sampler,cm,q)
        #print("The histogram of errors (above) should be clustered at
        #edges for success.")
        #histogram_of_as(samps,cm,q)
        #print("The histogram of a's (above) should be fairly uniform.")
        #histogram_of_errors_2(samps,cm,q) <- this line doesn't work
        #print("The histogram of sample errors (above) should be clustered at
        #edges for success.")
        #print("Time for Phase 3: ", timer.stop())
        #timer.start()
        #print("********** PHASE 4: ATTACK ALGORITHM")
        lift_secret = go_to_q(secret,cm.inverse(),q, R)
        #modding the secret by q
        # = secret_mod_q(secret,cm,q, R)
        result = alg2(samps, 10,cm.inverse(), q, lift_secret, R, quickflag)
        #print("Result of Algorithm 2:", result)
        #print("Time for Phase 4: ", timer.stop())
        if result == True:
            count_successes = count_successes + 1
        #print(count_successes, " out of ", trialnum+1, " successes so far.")
    #totaltime = timer2.stop()
    #print("Total time for ", trialnum+1, "trials was ", totaltime)
```

```
624     print("Total number of successes in finding the secret:",
625         count_successes, "out of", trialnum+1)
626     return count_successes
627
628
629
630
631
632
633
634
635 #-----------------------------------------------#
636 #HISTOGRAMS                                     #
637 #-----------------------------------------------#
638
639
640 def histoq(data):
641     """
642     Given a list of elements of Z/qZ, make a histogram and zero count.
643
644     INPUT:
645     data - elements in Z/qZ
646
647     OUTPUT:
648     [hist, zeroct] - tuple with the histogram and zero count
649     """
650     hist = [0 for i in range(10)] # empty histogram
651     zeroct=0 # count of zeroes mod q
652
653     # iterate over each data element, add to the histogram or zero count
654     for datum in data:
655         e = datum
656         if e == 0:
657             zeroct = zeroct+1
658         histbit = floor(ZZ(e)*10/q)
659         hist[histbit]=hist[histbit]+1
660     return [hist, zeroct]
661
662
663
664 def histo(data,cm,q):
665     """
666     Given a list of vectors in R^n, create a histogram of their values
667     in Z/qZ under make_poly, together with a zero count.
668
669     INPUT:
670     data - vectors in R^n
671     cm.inverse() - change of basis matrix
672
673     OUTPUT:
674     - histogram of the values in Z/qZ with zero count
675     """
676     return histoq([go_to_q(datum,cm.inverse(),q) for datum in data])
677
```

```
678
679
680 def histogram_of_errors(sampler,cm,q):
681     """
682     Create a histogram of error vectors, transported to polynomial ring.
683
684     INPUT:
685     sampler
686
687     OUTPUT:
688     none (output is printed externally)
689     """
690
691     print("Creating a histogram of errors mod q.")
692
693     # Create the list of errors
694     errs = []
695     for i in range(80):
696         errs.append(sampler())
697
698     # Construct a histogram from the error list and print it to the screen
699     hist = histo(errs,cm.inverse(),q)
700     print("The number of error vectors that are zero:", hist[1])
701     bar_chart(hist[0], width=1).show(figsize=2)
702     return
703
704
705
706 def histogram_of_as(samps,cm,q):
707     """
708     Create a histogram of the a's in the samples,
709     transported to polynomial ring.
710
711     INPUT:
712     none
713
714     OUTPUT:
715     none (output is printed externally)
716     """
717
718     print("Creating a histogram of sample a's mod q.")
719     a_vals = [samp[0] for samp in samps]
720     hist = histo(a_vals,cm.inverse(),q)
721     print("The number of a's that are zero:", hist[1])
722     bar_chart(hist[0], width=1).show(figsize=2)
723     return
724
725
726
727
728 def histogram_of_errors_2(samps,cm,q):
729     """
730     Create a histogram of errors by correct guess.
731
```

```
732     INPUT:
733     none
734
735     OUTPUT:
736     none (output is printed externally)
737     """
738
739     print("Creating a histogram of supposed errors if sample is guessed,
740         mod q.")
741     print(q)
742
743     hist = histoq([(IntegerModRing(q)(go_to_q(sample[1],cm.inverse(),q) -
744         go_to_q(sample[0],cm.inverse(),q)*
745             go_to_q(secret,cm.inverse(),q))).lift() for sample in samps])
746     # hist = histoq([ lift(Zq(go_to_q(sample[1],cm.inverse(),q) -
747     #go_to_q(sample[0],cm.inverse(),q)*go_to_q(secret,cm.inverse(),q)))
748     #for sample in samps])
749     print("The number of such that are zero:", hist[1])
750     bar_chart(hist[0], width=1).show(figsize=2)
751     return
752
753
754
755
756 from sage.stats.distributions.discrete_gaussian_lattice
757 import DiscreteGaussianDistributionLatticeSampler
```