KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Security of LWE-based cryptosystems

Lauren De Meyer

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: wiskundige ingenieurstechnieken

Promotor:
Prof. dr. ir. Bart Preneel

Assessoren:
Prof. dr. ir. D. Huybrechs
Prof. dr. ir K. Meerbergen

Begeleider:
Dr. ir. F. Vercauteren

Academiejaar 2014 – 2015

Preface

Enormous gratitude goes out to Professor Bart Preneel for introducing me into the world of cryptography, giving me opportunities to explore my interests and supporting me in my endeavours. I would especially like to thank Fre Vercauteren for devoting so much of his time to familiarizing me with this subject and guiding me through my thesis. This thesis would not have come together without his experience and knowledge.

Special thanks is owed to Danny De Cock and Pieter Maene for the significant part they played in the deployment of my website. I want to mention my friends Eline, Koen, Lien, my parents and sisters, who frequently had to (pretend to) listen to my frustrations about website development.

Last but not least, I express gratitude to my mother, who bravely took on the challenge of proofreading a text full of unfamiliar mathematical and technical writing.

Lauren De Meyer

Contents

Abstract

The Learning with Errors (LWE) problem is an important mathematical problem with conjectured security against quantum computers. At present, the best attacks require exponential complexity in the LWE dimension. LWE opens doors to efficient cryptographic constructions with known theoretical proofs of security, even for a world with quantum computers. Moreover, it allows for extremely various applications such as fully-homomorphic and identity-based encryption.

The practical security of LWE can be measured by the complexity of a successful attack against it. A lot of uncertainty surrounds this topic and consistent results are hard to find in literature. Currently, a designer must comb through countless papers to find a security estimate. Moreover, as most authors [LP11, DTV15, LN13] only provide results for a limited number of parameter sets, it is difficult to determine the security of a new one. In order to get rid of uncertainty, complexity and need to compromise, we need a consistent method to figure out the security of LWE for any choice of parameters.

In this thesis, we collect the most important attacks against the LWE problem and investigate their complexity as a function of the parameters. Because of LWE's connection with the shortest vector problem, lattice basis reduction is an important component of most attacks. A good approximation of the runtime of such algorithms is therefore essential in our analysis of LWE's security. In this cryptographic context however, we are dealing with lattice dimensions and computational costs that go far beyond the scope of experiments. This difficulty is the source of a lot of confusion. Various estimates have been proposed in earlier literature, some quite simple and some very elaborate. We need a runtime prediction that is both easy enough to compute and complicated enough to capture the influence of all relevant variables. In this thesis, we propose a new way of predicting lattice reduction execution times based on ideas available in existing literature.

Our results lay the basis for a web application that allows people to query the security of any set of parameters. Unlike most publications that focus on just one kind of attack against LWE, our website considers the impact of multiple competing methods on the security. Moreover, based on these formulas and security estimates, we are able to calculate the value of certain parameters in order to achieve a particular security level. The aim of this application is to simplify the process of choosing parameters, inspire more trust in the security of LWE and boost the conception of new cryptographic schemes for a post-quantum world.

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

LWE	Learning with Errors
IND-CPA	Indistinguishable under Chosen Plaintext Attack
SVP	Shortest Vector Problem
BDD	Bounded Distance Decoding
SIVP	Shortest Independent Vectors Problem
SIS	Short Integer Solution
GSA	Geometric Series Assumption
R-LWE	Ring Learning with Errors
FFT	Fast Fourier Transform
BKW	Blum-Kalai-Wasserman

Symbols

e	Euler's number = exp(1)
π	The number pi
sec	Number of bits security
n	LWE dimension
q	LWE modulus
s	Gaussian width parameter
m	Number of LWE samples
δ	Root-Hermite factor
$\mathbb{E}$	Expected Value

Vectors are printed in bold

CHAPTER 1

Introduction

1.1 Context

Mathematical problems such as integer factorization and the discrete logarithm problem have formed the foundation of public-key cryptography for many years. Today, the security of various prevailing encryption schemes is endangered by the prospect of quantum computing. For instance, the widely used RSA cryptosystem can easily be attacked by a quantum computer. As a result, we see a renewed interest in lattice-based cryptography, which appears to be resistant to quantum attacks. In particular, the Learning with Errors (LWE) problem, a generalization of the Learning Parity with Noise problem, has been proven to be equally hard to solve as worst-case lattice problems. It has therefore become an important building block in modern cryptographic systems and a popular topic in present-day research. In addition to its significance in post-quantum cryptography, the LWE problem also has promising applications, such as fully-homomorphic and identity-based encryption. With a fully homomorphic encryption scheme it is possible to perform calculations on encrypted data, which opens up the opportunity to outsource private computations to third parties.

1.2 Research goals and strategy

Progress in the development of new LWE-based cryptographic algorithms for real-world applications has somewhat been thwarted by the lack of concrete security estimates. Theoretical and asymptotic statements are not enough to inspire confidence in this relatively young group of cryptosystems. Furthermore, designers must understand how a choice of parameters influences the security of their cryptosystems. The primary objective of this thesis is to develop an accessible and fast web application that allows cryptosystem designers to query the security level of the LWE problem for an arbitrary set of parameters. In addition, we have succeeded to include the functionality of proposing new parameters to users.

A substantial part of this thesis will be devoted to an in-depth and critical analysis of existing literature. Despite a growing interest in the usefulness of the LWE problem, a comprehensive overview is not yet available. Apart from establishing a background for the design of our intended web application, this survey constitutes an extensive reference including all knowledge necessary to understand the subject. Where proofs or elaborations are incomplete or lacking, we complement the existing literature with our own developments.

In the literature, we examine the existing attacks against LWE and we search for formulas that express their complexity as a function of the LWE parameters. These formulas form the foundation for our web application. As speed is important for a website, particular effort was also devoted to the optimization of the implementations.

1.3 Structure of this thesis

The structure of the thesis reflects the four parts of our analysis.

In the **second chapter**, we present the required background on cryptography, lattices and discrete Gaussian distributions. This chapter introduces basic definitions and formulas that are used in the rest of this work.

Chapter three elaborates on the main issue of this work, the learning with errors problem itself. We explain in detail the structure of existing attacks againts LWE and demonstrate the derivations of mathematical formulas that were used in the web application.

In **chapter four** we provide an example of an LWE-based cryptosystem with the aim to discuss the limitations for LWE parameters that such a design may entail. This chapter emphasizes that security is not the only concern when choosing LWE parameters.

Chapter five contains a description of our own experiments and implementations. During our work, we came upon an additional objective of finding a decent runtime estimate for the BKZ algorithm. This compelled us to temporarily shift our main focus to lattice basis reduction and implement Chen and Nguyen's simulation algorithm for experiments. This chapter zooms in on our query for a new BKZ runtime estimate and completes the description of our website's functionalities. In addition to constructing the web application with our implementations, we use them to enhance insight into the LWE problem by showing the sensitivity of security estimates to the parameters or investigating how the choice of certain parameters affects key sizes.

Finally, in **chapter six**, we recapitulate the features of LWE and its attacks and summarize our most important results and observations.

CHAPTER 2

Background

In this chapter, we situate the LWE problem within the world of cryptography and lay the mathematical foundation for the attacks against it. LWE-based encryption schemes belong to the field of lattice-based cryptography, which in turn is classified under post-quantum cryptography. In section 2.1, we describe the basics of public-key cryptography and security. Because of the close connection between the LWE problem and lattice problems, section 2.2 thereafter presents an in-depth background on lattices and lattice basis reduction algorithms. Finally, LWE samples are inner products with noise, drawn from a discrete Gaussian distribution. In section 2.3, we describe the notion of continuous and discrete Gaussians over a lattice and a lattice's smoothing parameter.

2.1 Public-Key Cryptography

Cryptographic algorithms can generally be divided in two groups: Asymmetric (or public-key) and symmetric cryptosystems. The main difference is that public-key cryptography requires two keys: a public key pk and a secret key sk, whereas symmetric cryptography schemes use one shared secret key. The idea of public-key encryption is that the encryption of some message with a public key can only be decrypted with the corresponding secret key:

$$\mathrm{Dec}_{sk}\left(\mathrm{Enc}_{pk}(m)\right) = m \tag{2.1}$$

When Alice wants to receive a confidential message from Bob, she first sends her public key to Bob, therein allowing him to encrypt his message (or plaintext). An adversary Eve can also get a hold of this public key, but she will not be able to use it to decrypt the ciphertext that Bob sends to Alice. Both keys are thus mathematically related. However, it must be computationally impossible for Eve to derive the secret key from the public key.

2.1.1 Lattice-based cryptography

Most encryption schemes base their security on the hardness of some mathematical problem. The RSA cryptosystem for example relies on the difficulty of factoring a product of two large primes. In this thesis, we consider a relatively young group of cryptographic algorithms, whose security is based on a particular lattice problem. The general interest in lattice-based cryptography has increased because of the prospects of quantum computers. Many of the well-known cryptosystems we use today are easily attacked by a quantum computer. For example, Shor's quantum algorithms for integer factorization and the discrete logarithm problem have been known since 1994 [Sho99] and could be used to attack RSA or ElGamal. Lattice-based cryptosystems appear to be resistant both to classical and quantum adversaries.

2.1.2 Evaluating Security

There are many different levels of security for a cryptographic system. First and foremost, it should be hard for an adversary to decrypt a ciphertext without the secret key. In another stronger degree of security, the adversary should not even be able to obtain partial information on the plaintext. This notion was first introduced by Goldwasser and Micali [GM82] and is known today as semantic security. They showed that semantic security is actually equivalent to ciphertext indistinguishability. Consider the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- $\mathcal{C}$ generates a public key pk and a secret key sk, shares pk with the adversary and keeps sk secret.
- $\mathcal{A}$ sends two distinct chosen messages m_0 and m_1 to $\mathcal{C}$
- $\mathcal{C}$ flips a bit $b \in \{0, 1\}$ and encrypts the message m_b. The ciphertext $c = \text{Enc}_{pk}(m_b)$ is sent to $\mathcal{A}$ and b is kept secret.
- $\mathcal{A}$ tries to deduce the bit b.

The adversary succeeds in distinguishing the ciphertexts of chosen messages if he can guess the bit b correctly. The game is therefore called the IND-CPA (Indistinguishable under chosen-plaintext attack) game. The adversary's success is measured by his advantage over random guessing:

$$\text{Adv}(\mathcal{A}) = \big|\Pr[\text{Success}|b = 0] - \Pr[\text{Success}|b = 1]\big|$$

When $\mathcal{A}$'s strategy is to randomly pick a bit b, his success probability is always $1/2$ so his advantage $\text{Adv}(\mathcal{A})$ is zero. A cryptosystem is considered IND-CPA secure if any polynomially bounded adversary has only a negligible advantage in the IND-CPA game. Another version of the game replaces m_1 with a by $\mathcal{C}$ randomly generated message. In that case, the adversary tries to distinguish the encryption of a known plaintext from something uniformly random. It can be proven that these games are equivalent. Again, if any adversary has only a negligible advantage, then the cryptosystem is semantically secure.

In practice, the security of a cryptosystem is measured by the effort required to mount a successful attack. Such an attack can attempt to recover a secret key or break

semantic security, depending on the adversary's objective. Suppose he has access to a method that requires T operations to obtain an advantage ϵ. The adversary can then mount a successful attack (obtain overall advantage $\mathcal{O}(1)$) by repeating this method ϵ^{-1} times, provided that subsequent executions of this method independently result in the same advantage. In this case, the total complexity is $T(\epsilon) \times \epsilon^{-1}$ operations so one may assume that the cryptosystem has sec $= \log_2(\min_\epsilon\{T(\epsilon) \times \epsilon^{-1}\})$ bits of security. Sometimes, it is more efficient to repeat a simple, unsuccessful attack many times than to put all effort in one successful attempt. A cryptosystem with sec bits of security can resist all adversaries, who can perform at most 2^{sec} operations.

2.2 Lattices

An m-dimensional lattice is defined as a discrete additive subgroup of $(\mathbb{R}^m, +)$, generated from a basis by forming linear combinations with integer coefficients. Suppose $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ is a full rank matrix with $m \geq n$. In this work, we consider two lattices of dimension m:

$$\Lambda_q(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}^n \text{ s.t. } \mathbf{z} = \mathbf{A}^T\mathbf{s} \bmod q\} \tag{2.2}$$

$$\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = 0 \pmod q\} \tag{2.3}$$

The first lattice $\Lambda_q(\mathbf{A})$ is formed by linear integer combinations of the rows of $\mathbf{A}$ (mod q). A vector $\mathbf{z}$ is considered an element or member of the lattice $\Lambda(\mathbf{A})$ if and only if there exists an integer vector $\mathbf{s} \in \mathbb{Z}^n$ such that $\mathbf{z} = \mathbf{A}^T\mathbf{s} \bmod q$. Vectors in the second lattice $\Lambda_q^\perp(\mathbf{A})$ are all orthogonal (mod q) to the rows of $\mathbf{A}$. In Figure 2.1, we show an example of a two-dimensional lattice. Both lattices in (2.2) and (2.3) are q-ary lattices because any vector $\mathbf{z}$ is a member of Λ_q if and only if $\mathbf{z} \bmod q$ is a member. As a lattice by definition contains the zero vector, this means that $q\mathbb{Z}^m \subset \Lambda_q \subset \mathbb{Z}^m$.

The matrix $\mathbf{A}$ was defined as a non-square matrix. For any full rank lattice Λ of dimension m, a square matrix $\mathbf{B} \in \mathbb{Z}^{m\times m}$ can be found such that $\Lambda = \Lambda(\mathbf{B})$. We call that matrix $\mathbf{B}$ the lattice's basis. This lattice basis is not unique because the product of a basis with a unimodular matrix will not change its lattice. In other words, performing unimodular operations (like permuting and adding columns/rows) on a matrix $\mathbf{B}$ doesn't modify the lattice $\Lambda(\mathbf{B})$ that was generated by $\mathbf{B}$.

Lemma 1 ([Eis09])**.** *Let $\boldsymbol{B}$, $\boldsymbol{B}' \in \mathbb{R}^{m\times m}$ be two rational non-singular matrices. One has $\Lambda(\boldsymbol{B}) = \Lambda(\boldsymbol{B}')$ if and only if there exists a unimodular matrix $\boldsymbol{U} \in \mathbb{Z}^{m\times m}$ with $\boldsymbol{B}' = \boldsymbol{U} \cdot \boldsymbol{B}$.*

Proof. ($\Rightarrow$): We first start from the assumption that $\Lambda(\mathbf{B}) = \Lambda(\mathbf{B}')$. On the one hand, this means that $\Lambda(\mathbf{B}') \subseteq \Lambda(\mathbf{B}) \Leftrightarrow$ for all elements $\mathbf{z} = \mathbf{B}'^T\mathbf{s}$ with $\mathbf{s} \in \mathbb{Z}^m$, $\mathbf{z} \in \Lambda(\mathbf{B}') \subseteq \Lambda(\mathbf{B}) \Leftrightarrow \exists \mathbf{x} \in \mathbb{Z}^m$ such that $\mathbf{z} = \mathbf{B}^T\mathbf{x} \Leftrightarrow \mathbf{z}^T = \mathbf{s}^T\mathbf{B}' = \mathbf{x}^T\mathbf{B}$. If we let $\mathbf{s}$ be unit vectors, then $\exists \mathbf{U} \in \mathbb{Z}^{m\times m}$ such that $\mathbf{I} \cdot \mathbf{B}' = \mathbf{U} \cdot \mathbf{B}$.
On the other hand, $\Lambda(\mathbf{B}) \subseteq \Lambda(\mathbf{B}')$ so we obtain $\exists \mathbf{V} \in \mathbb{Z}^{m\times m}$ s.t. $\mathbf{B} = \mathbf{V}\mathbf{B}'$ with a similar reasoning. It follows that $\mathbf{B} = \mathbf{V} \cdot \mathbf{U} \cdot \mathbf{B}$ and since $\mathbf{B}$ is non-singular,

$\mathbf{V}\cdot\mathbf{U} = \mathbf{I}_m \Leftrightarrow \det(\mathbf{V}\cdot\mathbf{U}) = 1$. Finally, with $\mathbf{U}$ an integer matrix, we have $\det(\mathbf{U}) \in \mathbb{Z}$ so $\det(\mathbf{U}) = \pm 1$.

($\Leftarrow$): We assume now that $\mathbf{B}' = \mathbf{U}\cdot\mathbf{B}$ for some $\mathbf{U} \in \mathbb{Z}^{m\times m}$ with $|\det(\mathbf{U})| = 1$. For each element $\mathbf{z} \in \Lambda(\mathbf{B}') = \Lambda(\mathbf{U}\cdot\mathbf{B})$, we know by definition that $\exists \mathbf{s} \in \mathbb{Z}^m$ s.t. $\mathbf{z} = \mathbf{B}'^T\mathbf{s} = \mathbf{B}^T\mathbf{U}^T\mathbf{s} \Leftrightarrow \exists \mathbf{x} = \mathbf{U}^T\mathbf{s} \in \mathbb{Z}^m$ s.t. $\mathbf{z} = \mathbf{B}^T\mathbf{x} \Leftrightarrow \mathbf{z} \in \Lambda(\mathbf{B})$. Therefore $\Lambda(\mathbf{B}') \subseteq \Lambda(\mathbf{B})$. Furthermore, since $\mathbf{U}$ is unimodular, $\mathbf{B} = \mathbf{U}^{-1}\cdot\mathbf{B}'$ and the same argument leads to $\Lambda(\mathbf{B}) \subseteq \Lambda(\mathbf{B}')$. □

For every lattice Λ, there is a dual or reciprocal lattice Λ^*.

$$\Lambda^* = \{\mathbf{y} \in \mathbb{R}^m \text{ such that } \langle \mathbf{y}, \mathbf{z}\rangle \in \mathbb{Z}\ \forall \mathbf{z} \in \Lambda\} \tag{2.4}$$

In the case of q-ary lattices, the two lattices from (2.2) and (2.3) are dual up to a factor q:

$$q\cdot\Lambda_q(\mathbf{A})^* = \Lambda_q^\perp(\mathbf{A}) \tag{2.5}$$

$$q\cdot\Lambda_q^\perp(\mathbf{A})^* = \Lambda_q(\mathbf{A}) \tag{2.6}$$

We only demonstrate (2.5) as the proof for (2.6) is similar.

Proof [HW11]. ($\subseteq$): Suppose $\tilde{\mathbf{y}} \in q\cdot\Lambda_q(\mathbf{A})^* \Leftrightarrow \exists \mathbf{y} \in \Lambda_q(\mathbf{A})^*$ such that $\tilde{\mathbf{y}} = q\cdot\mathbf{y}$. From the definition of a dual lattice, we know that $\mathbf{y} \in \mathbb{R}^m$ and $\forall \mathbf{z} \in \Lambda_q(\mathbf{A}) : \langle \mathbf{z}, \mathbf{y}\rangle \in \mathbb{Z}$. Recall the definition of $\Lambda_q^\perp(\mathbf{A})$ in (2.3): we have $\mathbf{A}\tilde{\mathbf{y}} = q\cdot\mathbf{A}\mathbf{y}$. Hence, if $\mathbf{A}\mathbf{y} \in \mathbb{Z}$, then $\mathbf{A}\tilde{\mathbf{y}} \equiv 0 \pmod q$. Therefore, all that remains to prove that $\tilde{\mathbf{y}} \in \Lambda_q^\perp(\mathbf{A})$, is to show that $\tilde{\mathbf{y}}$ and $\mathbf{A}\mathbf{y}$ are integral:
Firstly, suppose $\mathbf{z} = q\cdot\mathbf{e}_i$ with $\mathbf{e}_i$ a unit vector $(0,\dots,0,1,0,\dots,0)$. $\mathbf{z} \in \Lambda_q(\mathbf{A})$ since $q\mathbb{Z}^m \subset \Lambda_q(\mathbf{A})$ so by construction $\langle \mathbf{z}, \mathbf{y}\rangle \in \mathbb{Z}$. As a result, $\tilde{\mathbf{y}}$ is an integral vector:

$$\begin{aligned}\langle \mathbf{e}_i, \tilde{\mathbf{y}}\rangle &= q\cdot\langle \mathbf{e}_i, \mathbf{y}\rangle \\ &= \langle \mathbf{z}, \mathbf{y}\rangle \in \mathbb{Z}\end{aligned}$$

Secondly, for $\mathbf{z} = \mathbf{A}^T\mathbf{e}_i \in \Lambda_q(\mathbf{A})$: $\langle \mathbf{z}, \mathbf{y}\rangle \in \mathbb{Z}$ which means $\mathbf{A}\mathbf{y}$ is integral:

$$\begin{aligned}\langle \mathbf{e}_i, \mathbf{A}\mathbf{y}\rangle &= \langle \mathbf{A}^T\mathbf{e}_i, \mathbf{y}\rangle \\ &= \langle \mathbf{z}, \mathbf{y}\rangle \in \mathbb{Z}\end{aligned}$$

Therefore $\tilde{\mathbf{y}} \in \Lambda_q^\perp(\mathbf{A})$ so $q\cdot\Lambda_q(\mathbf{A})^* \subseteq \Lambda_q^\perp(\mathbf{A})$.

($\supseteq$): Suppose that $\tilde{\mathbf{y}} \in \Lambda_q^\perp(\mathbf{A}) \Leftrightarrow$ by definition $\tilde{\mathbf{y}} \in \mathbb{Z}^m$ and $\mathbf{A}\tilde{\mathbf{y}} \equiv 0 \pmod q \Leftrightarrow \exists \mathbf{x} \in \mathbb{Z}^n$ such that $\mathbf{A}\tilde{\mathbf{y}} = q\cdot\mathbf{x}$. Now, with $\mathbf{y} = q^{-1}\cdot\tilde{\mathbf{y}}$, we must prove that $\forall \mathbf{z} \in \Lambda_q(\mathbf{A}) : \langle \mathbf{z}, \mathbf{y}\rangle \in \mathbb{Z}$.
Firstly, for $\mathbf{z} = \mathbf{A}^T\mathbf{s}$ with $\mathbf{s} \in \mathbb{Z}^n$:

$$\begin{aligned}\langle \mathbf{z}, \mathbf{y}\rangle &= \langle \mathbf{A}^T\mathbf{s}, q^{-1}\cdot\tilde{\mathbf{y}}\rangle \\ &= \langle \mathbf{s}, q^{-1}\cdot\mathbf{A}\tilde{\mathbf{y}}\rangle \\ &= \langle \mathbf{s}, q^{-1}\cdot q\cdot\mathbf{x}\rangle \\ &= \langle \mathbf{s}, \mathbf{x}\rangle \in \mathbb{Z}\end{aligned}$$

Secondly, for $\mathbf{z} = q \cdot \mathbf{r}$ with $\mathbf{r} \in \mathbb{Z}^m$:

$$\begin{aligned}\langle \mathbf{z}, \mathbf{y} \rangle &= \langle q \cdot \mathbf{r}, q^{-1} \cdot \tilde{\mathbf{y}} \rangle \\ &= \langle \mathbf{r}, \tilde{\mathbf{y}} \rangle \in \mathbb{Z}\end{aligned}$$

Therefore $\mathbf{y} \in \Lambda_q(\mathbf{A})^*$ so $\Lambda_q^\perp(\mathbf{A}) \subseteq q \cdot \Lambda_q(\mathbf{A})^*$. □

The volume of a lattice $\mathrm{vol}(\Lambda) = \det(\Lambda)$ is defined by $|\det(\mathbf{B})|$ for any basis $\mathbf{B}$ of Λ. This volume is equal to the volume of the fundamental parallelepiped $\mathcal{P}_{1/2}(\mathbf{B})$ of that basis. For $\mathbf{B} = \langle \mathbf{b}_1, \ldots, \mathbf{b}_m \rangle$:

$$\mathcal{P}_{1/2}(\mathbf{B}) = \{\sum_{i=1}^{m} x_i \mathbf{b}_i : -\frac{1}{2} \leq x_i < \frac{1}{2}\} \tag{2.7}$$

This parallelepiped is the unit cell of the lattice, as can be seen in Figure 2.1. To investigate the volume of q-ary lattices $\Lambda_q(\mathbf{A})$ with q prime and $\mathbf{A}$ a full rank matrix, we use the rank-nullity theorem $\mathrm{rnk}(\mathbf{A}) + \mathrm{nul}(\mathbf{A}) = m$ and the assumption that $\mathrm{rnk}(\mathbf{A}) = n$ to obtain $\mathrm{nul}(\mathbf{A}) = m - n$. We then use Lagrange's theorem [Rot01] to determine the volume of $\Lambda_q^\perp \subset \mathbb{Z}_q^m$:

$$\mathrm{vol}(\Lambda_q^\perp(\mathbf{A})) = \frac{\mathrm{vol}(\mathbb{Z}_q^m)}{[\mathbb{Z}_q^m : \Lambda_q^\perp(\mathbf{A})]} = \frac{q^m}{q^{m-n}} = q^n \tag{2.8}$$

To obtain the volume of Λ_q, we use equation (2.6) and the property $\mathrm{vol}(\Lambda_q^*) = \mathrm{vol}(\Lambda_q)^{-1}$.

$$\mathrm{vol}(\Lambda_q(\mathbf{A})) = \mathrm{vol}(q \cdot \Lambda_q^\perp(\mathbf{A})^*) = q^m \cdot \mathrm{vol}(\Lambda_q^\perp(\mathbf{A}))^{-1} = q^{m-n} \tag{2.9}$$

Alternatively, we can again use Lagrange's theorem and the assumption $\mathrm{rnk}(\mathbf{A}) = n$ to obtain the same result:

$$\mathrm{vol}(\Lambda_q(\mathbf{A})) = \frac{\mathrm{vol}(\mathbb{Z}_q^m)}{[\mathbb{Z}_q^m : \Lambda_q(\mathbf{A})]} = \frac{q^m}{q^n} = q^{m-n} \tag{2.10}$$

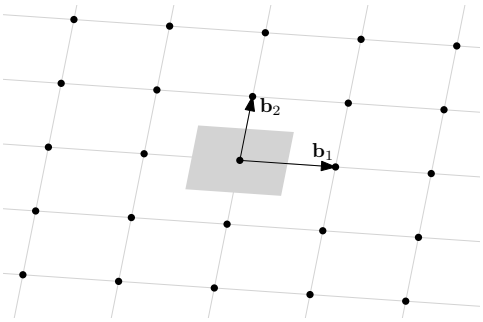


FIGURE 2.1. A fundamental parallelepiped of a 2-dimensional lattice

As a discrete subgroup of $\mathbb{R}^m$, a non-trivial lattice must contain a nonzero vector $\mathbf{v}$ of minimal length. The length of this shortest vector is called the lattice's *first*

minimum $\lambda_1(\Lambda) = \min\{\|\mathbf{v}\| : \mathbf{v} \in \Lambda, \mathbf{v} \neq 0\}$. It should be noted that the shortest vector is not unique as $\|\mathbf{v}\| = \|-\mathbf{v}\|$. Finding such a vector is considered hard and is probably the best known among lattice problems. However, proving its NP-hardness has been a problem for the last two decades [Mic98].

Definition 1 (Shortest Vector Problem (SVP))**.** *Given a lattice* Λ*, find a vector* $\boldsymbol{v} \in \Lambda$ *such that* $\|\boldsymbol{v}\| = \lambda_1(\Lambda)$*.*

A well-known rough estimate for $\lambda_1(\Lambda)$ is provided by the following heuristic.

Theorem 1 (Gaussian Heuristic [Che13])**.** *Let* Λ *be a* m *dimensional lattice in* $\mathbb{R}^m$*, and* S *a measurable subset of* $\mathbb{R}^m$ *with finite volume. The Gaussian Heuristic predicts that the number of lattice points in* S*:*

$$\#\{S \cap \Lambda\} \approx \frac{\mathrm{vol}(S)}{\mathrm{vol}(\Lambda)} \tag{2.11}$$

Therefore, for any lattice Λ, if we let v_m be the volume of an m-dimensional unit ball, the ball of radius $(\mathrm{vol}(\Lambda)/v_m)^{1/m}$ is expected to contain one lattice point. This radius is often used as an upper bound for the shortest vector length:

$$\mathbb{E}[\lambda_1(\Lambda)] \approx \mathrm{GH}(\Lambda) = \Big(\frac{\mathrm{vol}(\Lambda)}{v_m}\Big)^{\frac{1}{m}} \tag{2.12}$$

Many other lattice problems are known. We mention the following three:

Definition 2 (Bounded Distance Decoding (BDD) Problem [LN13])**.** *Given a lattice* Λ *and a point* $\boldsymbol{t}$ *"close" to* Λ*, find* $\boldsymbol{z} \in \Lambda$ *such that* $\|\boldsymbol{z} - \boldsymbol{t}\|$ *is minimized. In this work, we regard* $\boldsymbol{t}$ *"close" if there exists a unique* $\boldsymbol{z}$ *such that* $\|\boldsymbol{z} - \boldsymbol{t}\| \leq \gamma \,\mathrm{vol}(\Lambda)^{1/m}$ *for some small* γ*.*

Definition 3 (Shortest Independent Vectors Problem (SIVP))**.** *Given a lattice* Λ *of dimension* m*, find* m *linearly independent vectors* $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots \boldsymbol{v}_m$ *such that* $\max \|\boldsymbol{v}_i\| < \max_{\boldsymbol{B}} \|\boldsymbol{b}_i\|$ *for any basis* $\boldsymbol{B} = \langle \boldsymbol{b}_1, \ldots, \boldsymbol{b}_m \rangle$*.*

Definition 4 (GapSVP$_\beta$)**.** *Given a lattice* Λ *and a value* r*, determine whether* $\lambda_1(\Lambda) \leq r$ *or* $\lambda_1(\Lambda) > \beta \cdot r$*.*

This last problem is also called the Decision version of the SVP.

2.2.1 Lattice Enumeration

Lattice enumeration is a method that can be used to approximate the SVP as well as the BDD problem. Given a lattice Λ, a BDD Radius R and a target point $\mathbf{t}$, the algorithm enumerates all lattice vectors $\mathbf{z} \in \Lambda$ such that $\|\mathbf{z} - \mathbf{t}\| \leq R$. Among these vectors, the one with the smallest norm $\|\mathbf{z} - \mathbf{t}\|$ solves the BDD Problem.

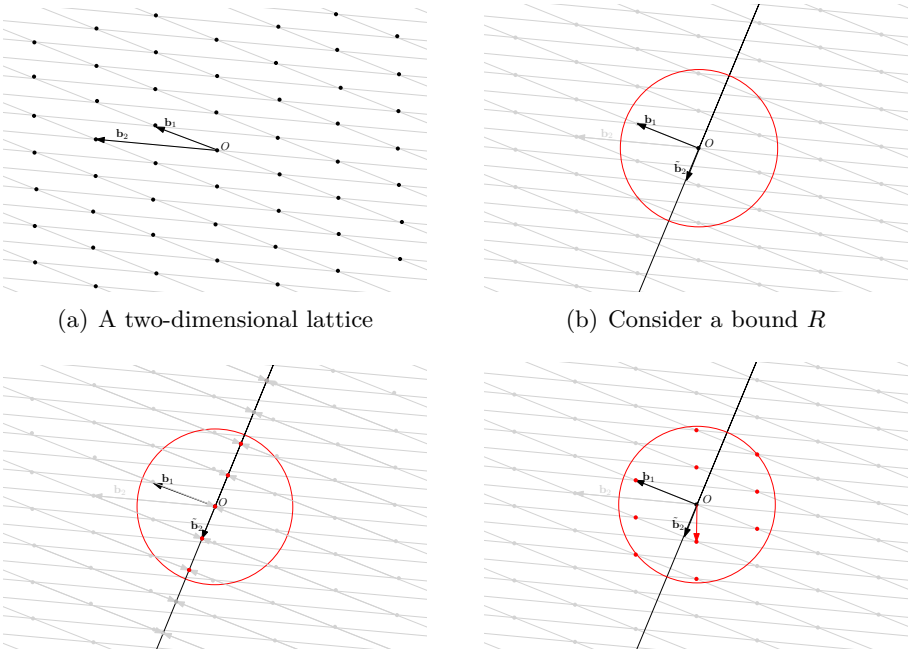
Enumeration of an m-dimensional lattice $\Lambda(\mathbf{B})$ bears resemblance to a search through a tree of m levels: Consider $\pi_i(\mathbf{x})$ the orthogonal projection of $\mathbf{x}$ onto $\langle \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1} \rangle^\perp$. Level k of the search tree contains all vectors $\mathbf{z} \in \Lambda$ with

$\|\pi_{m-k+1}(\mathbf{t}-\mathbf{z})\| \leq R$. The root of the tree is level $k=0$ and the leaves are at level m. They are all vectors $\mathbf{z}$ with $\|\mathbf{t}-\mathbf{z}\| \leq R$.

The enumeration procedure for the SVP is equivalent to the one just described except for it being centered around the origin instead of $\mathbf{t}$. This procedure often serves as subroutine in basis reduction algorithms

We illustrate how to find a shortest vector in a 2-dimensional lattice in Figure 2.2.

Pruning the enumeration tree to decrease time complexity at the cost of the success probability was proposed by Schnorr and Euchner [SE94] and later analysed by Gama, Nguyen and Regev [GNR10]. The latter authors proposed to search a subtree by using a different radius R_k for each level k with $R_1 \leq R_2 \leq \ldots \leq R_m = R$.



(a) A two-dimensional lattice

(b) Consider a bound R

(c) Project lattice points on $\mathbf{b}_1^{\perp}$ and keep only projections within the bound

(d) Enumerate all lattice points from the projected lattice and choose the shortest

FIGURE 2.2. Illustration of lattice enumeration to find the shortest vector

2.2.2 Basis Reduction

Among the multiple bases of a lattice, some prove to be better than others. A basis is considered "good" if its vectors are close to the Gram-Schmidt vectors. An ideal basis would be one with $\|\mathbf{b}_1\| = \lambda_1(\Lambda)$. These bases are called *Korkine-Zolotarev* reduced.

Definition 5 ((Hermite-)Korkine-Zolotarev (HKZ) reduced bases [LLS90])**.** *Let Λ be an m-dimensional lattice with basis vectors $\langle \boldsymbol{b}_1, \ldots, \boldsymbol{b}_m \rangle$. This basis is Korkine-Zolotarev reduced*[1] *if*

[1]Some authors prefer to name them Hermite-Korkine-Zolotarev (HKZ) reduced

1. $\boldsymbol{b}_1$ *is a nonzero shortest vector in* $\Lambda \Leftrightarrow \lambda_1(\Lambda) = \|\boldsymbol{b}_1\|$
2. $|\mu_{i,1}| < \frac{1}{2}$ *for* $2 \leq i \leq m$ *with* $\mu_{i,j} = \langle \boldsymbol{b}_i, \tilde{\boldsymbol{b}}_j \rangle / \langle \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{b}}_j \rangle$
3. *The orthogonal projection of the basis on* $\boldsymbol{b}_1^{\perp}$ *is Korkine-Zolotorev reduced: the basis* $\langle \boldsymbol{b}_2 - \mu_{2,1}\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m - \mu_{m,1}\boldsymbol{b}_1 \rangle$ *is a HKZ basis.*

Unfortunately, no polynomial-time algorithm exists to compute such a basis for an arbitrarily large lattice. Enumeration of short vectors as in Figure 2.2 is only practical for small dimensions. In general, finding a HKZ basis would imply solving the hard problem described in Definition 1. Instead, basis reduction algorithms try to make the basis vectors as short and orthogonal as possible. The most common algorithms today are the LLL-algorithm [LLL82] and the BKZ algorithm [Sch03, CN11]. At the end of these methods, the first basis vector $\|\mathbf{b}_1\|$ is used as an approximation of the lattice's shortest vector.

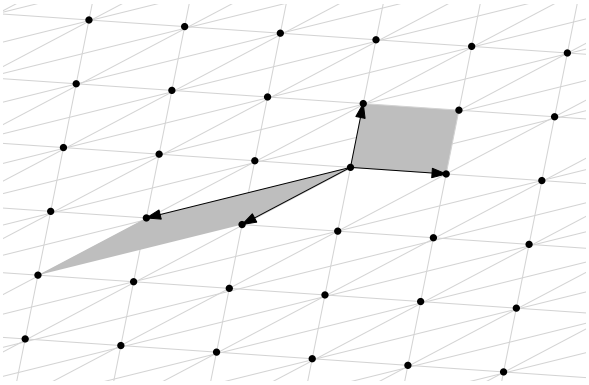


Figure 2.3. Comparison of a bad and good basis

To illustrate the difference between a good and bad basis, Figure 2.3 shows two different bases for the same 2-dimensional lattice. The unit cell of the bad basis is a long and thin parallelepiped. Identifying a shortest vector in an arbitrary lattice of higher dimension with a similar basis is clearly not straightforward. The other basis' unit cell is almost square and one of the basis vectors is even a shortest lattice vector.

The quality of a lattice basis is typically described by a characteristic called the *root-Hermite factor* δ. Gama and Nguyen [GN08] observed that the shortest vector after reduction of an m-dimensional lattice basis $\mathbf{B}$ has the following form

$$\|\mathbf{b}_1\| = \delta^m \det(\Lambda)^{1/m} \tag{2.13}$$

with parameter δ exclusively depending on the reduction algorithm used (and thus independent from the lattice itself), except if the lattice has an exceptional structure. The number δ^m is designated the *Hermite factor* of the lattice. Clearly, a small root-Hermite factor δ indicates a short basis vector $\mathbf{b}_1$ and thus a basis of good quality. For q-ary lattices, $q\mathbb{Z}^m \subset \Lambda_q$, so there is always a vector of length q. Therefore, the length of a short vector $\mathbf{v}$, found with a basis reduction algorithm is approximately

$$\|\mathbf{v}\| = \min\{q, \delta^m \det(\Lambda)^{1/m}\} \tag{2.14}$$

BKZ Basis Reduction

The *block Korkine-Zolotarev* (BKZ) lattice basis reduction algorithm was introduced by Schnorr and Euchner [SE94]. As the name implies, it computes a Korkine-Zolotarev basis for small dimensional projections of the original lattice. The dimension of these "blocks" is indicated by input parameter β. The choice for this blocksize influences both the runtime of the algorithm and the quality of the resulting basis. In the original version of the algorithm, runtimes for $\beta > 30$ are not feasible [LP11]. With recent improvements in lattice enumeration techniques, Chen and Nguyen [CN11] have designed a new version of the algorithm: BKZ 2.0, parametrized by the blocksize β and a maximum number of rounds R. Their optimization allows for blocksizes $\beta \geq 50$.

ALGORITHM 1 BKZ 2.0 [CN11]

Input: m-dimensional LLL-reduced basis $\mathbf{B} = \langle \mathbf{b}_1, \ldots, \mathbf{b}_m \rangle$
Output: m-dimensional BKZ-reduced basis $\mathbf{B} = \langle \mathbf{b}_1, \ldots, \mathbf{b}_m \rangle$

1: **for** R Rounds **do**
2: **for** $i = 1$ to $m - \beta$ **do**
3: project β-dimensional Lattice $\langle \mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1} \rangle \perp \langle \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1} \rangle$
4: Find a short vector $\mathbf{v}$ in the projected Lattice
5: $\mathbf{b}_i \leftarrow \mathbf{v}$
6: **end for**
7: **end for**

Finding a short vector in step 4 is done using Gama-Nguyen-Regev's enumeration with extreme pruning algorithm [GNR10] that was mentioned in section 2.2.1.

Apart from their improvements to the algorithm, Chen and Nguyen [CN11] also came up with a simulation algorithm that, given β, R and the Gram-Schmidt norms of an LLL-reduced basis, can predict the Gram-Schmidt norms of the BKZ 2.0-reduced basis after R rounds. This is an important tool in the analysis of the security of LWE, as the actual BKZ algorithm is computationally too costly to experiment with.

An important concept for BKZ-reduced bases is the *Geometric Series Assumption* (GSA). According to Schnorr [Sch03], the lengths $\|\tilde{\mathbf{b}}_i\|$ of the Gram-Schmidt vectors of a BKZ-reduced basis $\mathbf{B}$ decay as follows:

$$\|\tilde{\mathbf{b}}_i\| = \|\mathbf{b}_1\| \cdot \alpha^{i-1} \tag{2.15}$$

for some $0 < \alpha < 1$. Lindner and Peikert's experiments [LP11] have confirmed this. The factor α can be computed using the Hermite-root factor (2.13):

$$\det(\Lambda) = \prod_{i=1}^{m} \|\tilde{\mathbf{b}}_i\| = \|\mathbf{b}_1\|^m \prod_{i=1}^{m} \alpha^{i-1} = \|\mathbf{b}_1\|^m \cdot \alpha^{\sum_{i=1}^{m} i-1} = \delta^{m^2} \cdot \det(\Lambda) \cdot \alpha^{m(m-1)/2}$$
$$\Leftrightarrow \alpha = \delta^{-2m/(m-1)} \tag{2.16}$$

The GSA thus allows us to compute the lengths of all Gram-Schmidt vectors of a BKZ-reduced basis.

2.2.3 Ideal Lattices

Suppose $f(x) \in \mathbb{Z}[x]$ is a monic polynomial of degree n and $R = \mathbb{Z}[x]/f(x)$ is the ring of polynomials of degree at most $n-1$. As the elements of R are polynomials with n coefficients $\in \mathbb{Z}$, the ring R is isomorphic with the integer lattice $\mathbb{Z}^n$. In other words, there exists an isomorphism ψ, mapping polynomials from R to vectors in $\mathbb{Z}^n$. The most common and straightforward mapping is the *coefficient embedding*, which maps a polynomial to a vector of its coefficients [LPR10]:

$$\psi : R \to \mathbb{Z}^n : v(x) = \sum_{i=1}^{n} v_i x^{i-1} \to \psi(v) = \mathbf{v} = [v_0, v_1, \dots v_{n-1}]^T \tag{2.17}$$

An ideal I of a ring R is defined as a subset of R that is closed under addition and multiplication by arbitrary elements from R.

Definition 6 (Ring Ideal)**.** *For any ring $(R, +, \cdot)$, equipped with addition and multiplication, we define $(R, +)$ its additive group. A subset I is called an ideal of R if it satisfies the following conditions:*

1. *$(I, +)$ is a subgroup of $(R, +)$*
2. *$\forall r \in R : r \cdot I \subset I$*
3. *$\forall r \in R : I \cdot r \subset I$*

An ideal lattice $\psi(I)$ is defined as the set of polynomials from I, mapped to $\mathbb{Z}^n$. In fact, for any $v \in R$, the polynomials $v \cdot x^i \bmod f(x)$ for $i \in \{0, \dots, n-1\}$ form a basis for the ideal lattice $\psi(I)$, $I = \langle v \rangle$ [Sch13]. Let $\text{rot}(v) = v \cdot x \bmod f(x)$ and its coefficient embedding $\text{rot}(\mathbf{v})$ be rotations mod $f(x)$ of v. A basis of $\psi(I)$ is a collection of rotations of a vector $\text{rot}(\mathbf{v}) \in \psi(R)$. This means that we can store the basis matrix of an ideal lattice using n elements instead of n^2: We only have to store the vector $\mathbf{v}$. This is an important advantage of ideal lattices over normal ones. Moreover, for $f(x)$ a monic and irreducible polynomial of degree n, the vectors $\{\psi(v \cdot x^i \bmod f(x))\}_{i=0}^{n-1}$ are linearly independent [Sch13] so the ideal's basis has full rank.

This thesis only considers the $2n$-th cyclotomic polynomials $f(x) = x^n + 1$ with n a power of 2. The roots of f are $\{\zeta^k | k \in \mathbb{Z}_{2n}^*\}$ with ζ a primitive $2n$-th root of unity. In such case, we can easily show that a rotation of $\mathbf{v} = [v_0, v_1, \dots, v_{n-1}]^T$ has the form $[-v_{n-1}, v_0, v_1, \dots, v_{n-2}]^T$:

$$\begin{aligned}
\text{rot}(v(x)) &= v(x) \cdot x \bmod (x^n + 1) \\
&= (v_0 + v_1 \cdot x + \dots + v_{n-1} \cdot x^{n-1}) \cdot x \bmod (x^n + 1) \\
&= (v_0 \cdot x + v_1 \cdot x^2 + \dots + v_{n-1} \cdot x^n) \bmod (x^n + 1) \\
&= -v_{n-1} + v_0 \cdot x + \dots + v_{n-2} \cdot x^{n-1} \\
\Leftrightarrow \text{rot}(\mathbf{v}) &= [-v_{n-1}, v_0, \dots, v_{n-2}]^T
\end{aligned} \tag{2.18}$$

The equality on the fourth line follows from the fact that $x^n \equiv -1 \pmod{x^n + 1}$. The ideals in R with the coefficient embedding and $f(x) = x^n + 1$ are therefore named anti-cyclic lattices.

For cyclotomic polynomials, it is possible to define an alternative way to map polynomials to vectors. The *canonical embedding* builds an n-dimensional vector from the evaluations of a polynomial in the n roots of $f(x)$:

$$\tilde{\psi} : R \mapsto \mathbb{C}^n : v(x) \mapsto \tilde{\psi}(v) = [v(\zeta), v(\zeta^3), \ldots, v(\zeta^{2n-1})]^T \tag{2.19}$$

Contrarily to the coefficient embedding, not only addition but also multiplication can be conducted coordinate-wise.

2.3 Gaussian measures

Let us consider the Gaussian function $\rho_s(\mathbf{x}) = e^{(-\pi\|\mathbf{x}\|^2/s^2)}$ for $\mathbf{x} \in \mathbb{R}^m$, with Gaussian width parameter $s \in \mathbb{R}$. A normal distribution with mean 0 and standard deviation $\sigma = \frac{s}{\sqrt{2\pi}}$ has density function $\nu_s(\mathbf{x}) = s^{-m} \cdot \rho_s(\mathbf{x})$. Steinfeld defines the density function of a continuous Gaussian reduced modulo a lattice $\Lambda \subset \mathbb{Z}^m$ with basis $\mathbf{B}$ [Ste14] for $\mathbf{x} \in \mathcal{P}_{1/2}(\mathbf{B})$:

$$\nu'_s(\mathbf{x}) \stackrel{\text{def}}{=} (\nu_s \bmod \Lambda)(\mathbf{x}) = \sum_{\mathbf{v}\in\Lambda} \nu_s(\mathbf{x}+\mathbf{v}) = s^{-m} \cdot \sum_{\mathbf{v}\in\Lambda} \rho_s(\mathbf{x}+\mathbf{v}) \tag{2.20}$$

We use the following notation to evaluate the function ρ in a set $\mathbf{X}$: $\rho_s(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} \rho_s(\mathbf{x})$. This way, we have that $\nu'_s(\mathbf{x}) = s^{-m}\rho_s(\mathbf{x}+\Lambda)$.

Micciancio and Regev [MR07] use Fourier analysis to show that this density function (2.20) approximates a uniform distribution on the fundamental parallelepiped $\mathcal{P}_{1/2}(\mathbf{B})$ when the width parameter is large. Recall the Fourier transform of a function $h : \mathbb{R}^m \mapsto \mathbb{R}$:

$$\hat{h}(\mathbf{w}) = \int_{\mathbb{R}^m} h(\mathbf{x}) e^{-2\pi i\langle \mathbf{x},\mathbf{w}\rangle} d\mathbf{x} \tag{2.21}$$

and the following two translation properties:

$$h(\mathbf{x}) = g(\mathbf{x}+\mathbf{v}) \Leftrightarrow \hat{h}(\mathbf{w}) = e^{2\pi i\langle \mathbf{v},\mathbf{w}\rangle}\hat{g}(\mathbf{w}) \tag{2.22}$$

$$h(\mathbf{x}) = e^{2\pi i\langle \mathbf{x},\mathbf{v}\rangle} g(\mathbf{x}) \Leftrightarrow \hat{h}(\mathbf{w}) = \hat{g}(\mathbf{w}-\mathbf{v}) \tag{2.23}$$

An important characteristic of a Gaussian is that its Fourier transform is a Gaussian as well. More specifically: $\hat{\rho}_s(\mathbf{w}) = s^m \cdot \rho_{1/s}(\mathbf{w})$ is the Fourier transform of $\rho_s(\mathbf{x})$.

Finally, Poisson's summation formula is defined as follows:

Lemma 2 (Poisson's summation formula [Reg09])**.** *For any lattice Λ and any function $f : \mathbb{R}^m \mapsto \mathbb{C}$:*

$$f(\Lambda) = \det(\Lambda^*) \cdot \hat{f}(\Lambda^*)$$

where $\hat{f}$ denotes the Fourier transform of f.

The domain of ν'_s can easily be extended from $\mathcal{P}_{1/2}(\mathbf{B})$ to $\mathbb{R}^m$ as ρ_s is well defined for all $\mathbf{x} \in \mathbb{R}^m$. In addition, a lattice Λ is a closed group under addition so this

extension is periodic (i.e. $\rho_s(\mathbf{x}+\Lambda+\mathbf{v}) = \rho_s(\mathbf{x}+\Lambda)$, $\forall \mathbf{x} \in \mathbb{R}^m$ and $\forall \mathbf{v} \in \Lambda$). One can thus use a Fourier series representation for ν'_s [Ste14]:

$$\nu'_s(\mathbf{x}) = s^{-m} \cdot \rho_s(\mathbf{x}+\Lambda) = s^{-m} \cdot f(\Lambda) = s^{-m} \cdot \det(\Lambda^*) \cdot \hat{f}(\Lambda^*) \tag{2.24}$$

The last step of (2.24) uses Lemma 2 with $f(\mathbf{v}) = \rho_s(\mathbf{x}+\mathbf{v})$. Due to property (2.22), we know that $\hat{f}(\mathbf{w}) = \hat{\rho}_s(\mathbf{w})e^{2\pi i\langle \mathbf{x},\mathbf{w}\rangle}$ and we obtain the following:

$$\nu'_s(\mathbf{x}) = s^{-m} \cdot \det(\Lambda^*) \cdot \sum_{\mathbf{w}\in\Lambda^*} \hat{\rho}_s(\mathbf{w})e^{2\pi i\langle \mathbf{x},\mathbf{w}\rangle} = \det(\Lambda^*) \cdot \sum_{\mathbf{w}\in\Lambda^*} \rho_{1/s}(\mathbf{w})e^{2\pi i\langle \mathbf{x},\mathbf{w}\rangle} \tag{2.25}$$

We consider the uniform (constant) and non-uniform terms of (2.25) separately. Firstly, for $\mathbf{w} = \mathbf{0}$, the constant term $\det(\Lambda^*) \cdot \rho_{1/s}(\mathbf{0}) = \det(\Lambda^*) = 1/\det(\Lambda)$ is identical to the uniform distribution on $\mathcal{P}_{1/2}(\mathbf{B})$: $\nu_0(\mathbf{x}) = 1/\det(\Lambda)$ for $\mathbf{x} \in \mathcal{P}_{1/2}(\mathbf{B})$. From the non-uniform terms (with $\mathbf{w} \in \Lambda^* \setminus \{\mathbf{0}\}$), the sum of the coefficients is used as measure for ν'_s's non-uniformity [Ste14]:

$$S_s(\Lambda) \overset{\text{def}}{=} \sum_{\mathbf{w}\in\Lambda^*\setminus\{\mathbf{0}\}} \rho_{1/s}(\mathbf{w}) = \rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \tag{2.26}$$

In other words,

$$\nu'_s(\mathbf{0}) = \frac{1}{\det(\Lambda)} \sum_{\mathbf{w}\in\Lambda^*} \rho_{1/s}(\mathbf{w}) = \frac{1}{\det(\Lambda)}(1 + S_s(\Lambda))$$

The statistical distance between two continuous distributions ν_0 and ν_1 is defined as follows:

$$\Delta(\nu_0, \nu_1) = \frac{1}{2}\int_{\mathbb{R}^m} |\nu_0(\mathbf{x}) - \nu_1(\mathbf{x})| d\mathbf{x} \tag{2.27}$$

We now look for a bound on the statistical distance between a continuous Gaussian reduced modulo a lattice $\nu'_s(\mathbf{x})$ and the uniform distribution on $\mathcal{P}_{1/2}(\mathbf{B})$ $\nu_0(\mathbf{x})$.

$$\begin{aligned}
\Delta(\nu_0, \nu'_s) &= \frac{1}{2}\int_{\mathcal{P}_{1/2}(\mathbf{B})} |\frac{1}{\det(\Lambda)} - \frac{1}{\det(\Lambda)} \sum_{\mathbf{w}\in\Lambda^*} \rho_{1/s}(\mathbf{w})e^{2\pi i\langle \mathbf{x},\mathbf{w}\rangle}| d\mathbf{x} \\
&= \frac{1}{2 \cdot |\det(\Lambda)|}\int_{\mathcal{P}_{1/2}(\mathbf{B})} |1 - (1 + \sum_{\mathbf{w}\in\Lambda^*\setminus\{\mathbf{0}\}} \rho_{1/s}(\mathbf{w})e^{2\pi i\langle \mathbf{x},\mathbf{w}\rangle})| d\mathbf{x} \\
&= \frac{1}{2 \cdot |\det(\Lambda)|}\int_{\mathcal{P}_{1/2}(\mathbf{B})} \underbrace{|\sum_{\mathbf{w}\in\Lambda^*\setminus\{\mathbf{0}\}} \rho_{1/s}(\mathbf{w})e^{2\pi i\langle \mathbf{x},\mathbf{w}\rangle}|}_{\leq S_s(\Lambda)} d\mathbf{x} \\
&\leq \frac{1}{2}S_s(\Lambda) \cdot \frac{1}{|\det(\Lambda)|}\int_{\mathcal{P}_{1/2}(\mathbf{B})} d\mathbf{x} = \frac{1}{2}S_s(\Lambda)
\end{aligned} \tag{2.28}$$

This distance decreases as the parameter s increases. Indeed, as most of the mass of $\rho_{1/s}$ is situated within a width of $\frac{1}{s}$ (see Figure 2.4), decreasing this width also decreases the number of "large" terms in the sum of (2.26).

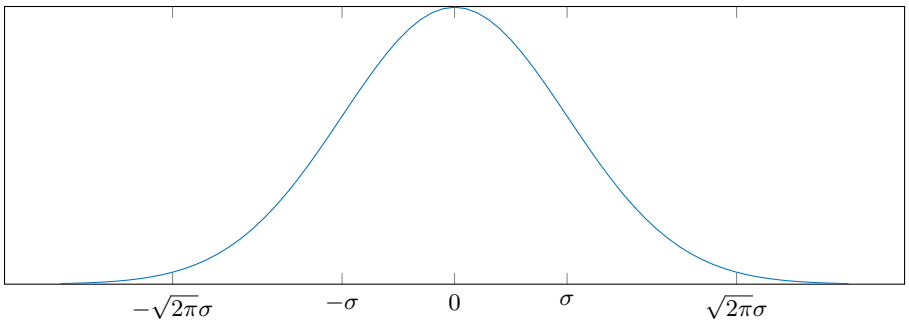


FIGURE 2.4. One dimensional Gaussian function $\rho_s(x)$ with standard deviation $\sigma = \frac{s}{\sqrt{2\pi}}$. Most of its mass is within a width $\sqrt{2\pi}\sigma = s$ of the origin

When $\frac{1}{s}$ drops below the shortest vector length of Λ^* (i.e. when $s > \frac{1}{\lambda_1(\Lambda^*)}$), all the terms of non-uniformity $S_s(\Lambda)$ are small. The statistical distance between ν'_s and ν_0 disappears and ν'_s becomes *smoothed*. Micciancio and Regev [MR07] were the first to introduce the notion of a lattice's smoothing parameter:

Definition 7 (The Smoothing Parameter [MR07])**.** *For an m-dimensional lattice Λ and a real $\epsilon > 0$, we define its smoothing parameter $\eta_\epsilon(\Lambda)$ to be the smallest s such that $S_s(\Lambda) \leq \epsilon$.*

$$s \geq \eta_\epsilon(\Lambda) \Leftrightarrow S_s(\Lambda) \leq \epsilon$$

To estimate this parameter, we need the following Lemma:

Lemma 3 ([MR07] Lemma 3.3)**.** *For any m-dimensional lattice Λ and positive real $\epsilon > 0$,*

$$\eta_\epsilon(\Lambda) \leq \sqrt{\frac{\ln(2m(1+1/\epsilon))}{\pi}} \cdot \lambda_m(\Lambda)$$

In the LWE problem we are dealing with discrete Gaussians. A discrete Gaussian distribution over a lattice Λ, with mean 0 and standard deviation $\sigma = \frac{s}{\sqrt{2\pi}}$ is defined as follows:

$$\forall \mathbf{x} \in \Lambda : D_{\Lambda,s}(\mathbf{x}) \stackrel{\text{def}}{=} \frac{\rho_s(\mathbf{x})}{\rho_s(\Lambda)} \tag{2.29}$$

The discrete Gaussian shows the same *smoothing* behaviour as a continuous Gaussian. Gentry, Peikert and Vaikuntanathan [GPV08] proved that a discrete Gaussian $D_{\Lambda,s}$ modulo a sublattice $\Lambda' \subset \Lambda$ is distributed almost uniformly as well when $s \geq \eta_\epsilon(\Lambda')$.

Lemma 4 ([GPV08])**.** *Let $\Lambda' \subset \Lambda \subset \mathbb{R}^m$ be lattices. For any $\epsilon \in (0, \frac{1}{2})$ and $s \geq \eta_\epsilon(\Lambda')$, we have $\Delta(D_{\Lambda,s} \bmod \Lambda', D_0) \leq 2\epsilon$ with D_0 the uniform distribution over the quotient group Λ/Λ'.*

Furthermore, Micciancio and Regev [MR07] demonstrated that the statistical properties of this discrete Gaussian $D_{\Lambda,s}$ are very close to those of the continuous Gaussian ν_s in the smoothing region. In particular, they obtain the following Lemma for the first and second moments:

Lemma 5 ([MR07]). *For any m-dimensional lattice Λ, unit vector $\boldsymbol{e}_i$ and real $0 < \epsilon < 1, s \geq 2\eta_\epsilon(\Lambda)$,*

$$\Big|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim D_{\Lambda,s}}[\langle \boldsymbol{x}, \boldsymbol{e}_i\rangle]\Big| \leq \frac{\epsilon s}{1-\epsilon}$$

$$\Big|\mathop{\mathbb{E}}_{\boldsymbol{x}\sim D_{\Lambda,s}}[\langle \boldsymbol{x}, \boldsymbol{e}_i\rangle^2] - \frac{s^2}{2\pi}\Big| \leq \frac{\epsilon s^2}{1-\epsilon}$$

Alternatively, we can investigate the density functions of the continuous and discrete Gaussian distributions in the smoothing region. We start by using Poisson's summation formula (Lemma 2) for $\rho_s(\Lambda)$.

$$\begin{aligned}
\rho_s(\Lambda) &= \det(\Lambda^*)\cdot\hat{\rho}_s(\Lambda^*) = \det(\Lambda^*)\cdot s^m\cdot\rho_{1/s}(\Lambda^*)\\
&= \det(\Lambda^*)\cdot s^m\sum_{\mathbf{w}\in\Lambda^*}\rho_{1/s}(\mathbf{w})\\
&= \det(\Lambda^*)\cdot s^m\Big(\rho_{1/s}(\mathbf{0}) + \sum_{\mathbf{w}\in\Lambda^*\ \{\mathbf{0}\}}\rho_{1/s}(\mathbf{w})\Big)\\
&= \frac{s^m}{\det(\Lambda)}(1+S_s(\Lambda))
\end{aligned} \tag{2.30}$$

In the smoothing region, when $S_s(\Lambda) = \epsilon << 1$, we may thus assume that $\rho_s(\Lambda) \approx s^m/\det(\Lambda)$, leading to the conclusion that the continuous Gaussian indeed approximates the discrete Gaussian well in the smoothing region:

$$D_{\Lambda,s}(\mathbf{x}) = \frac{\rho_s(\mathbf{x})}{\rho_s(\Lambda)} \approx \det(\Lambda)s^{-m}\rho_s(\mathbf{x}) = \det(\Lambda)\nu_s(\mathbf{x}) \text{ if } s \geq \eta_\epsilon(\Lambda) \tag{2.31}$$

For the integer lattice $\Lambda = \mathbb{Z}$, we know that $\det(\mathbb{Z}) = 1$ and $\lambda_1(\mathbb{Z}) = 1$. Lemma 3 helps us to find a suitable lower bound for s:

$$D_{\mathbb{Z},s}(\mathbf{x}) \approx \nu_s(\mathbf{x}) \text{ for } s \geq \sqrt{\frac{\ln(2(1+1/\epsilon))}{\pi}} \geq \eta_\epsilon(\mathbb{Z}) \tag{2.32}$$

Table 2.1. The upper bound of $\eta_\epsilon(\mathbb{Z})$ for some values ϵ

ϵ	$s \geq$
2^{-100}	4.7
2^{-200}	6.6
2^{-500}	10.5

To conclude this section, we recall some properties of the 1-dimensional continuous normal distribution $Z \sim \mathcal{N}(0,1)$. The cumulative normal distribution $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}\exp(-\frac{t^2}{2})dt$ represents the probability $\Pr[Z \leq x]$ for $x > 0$. For negative

values, we have $\Phi(-x) = 1 - \Phi(x)$. A relation between the cumulative normal distribution and the error function $\text{erf}(x) = \frac{1}{\sqrt{\pi}} \int_{-x}^{x} \exp(-t^2)dt$ is derived as follows:

$$\begin{aligned}
\Phi(x) &= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{x/\sqrt{2}} \exp\big(-(\frac{t}{\sqrt{2}})^2\big) d(\frac{t}{\sqrt{2}}) \\
\text{erf}(\frac{x}{\sqrt{2}}) &= \frac{1}{\sqrt{\pi}} \int_{-x/\sqrt{2}}^{x/\sqrt{2}} \exp\big(-(\frac{t}{\sqrt{2}})^2\big) d(\frac{t}{\sqrt{2}}) \\
&= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{x/\sqrt{2}} \exp\big(-(\frac{t}{\sqrt{2}})^2\big) d(\frac{t}{\sqrt{2}}) - \frac{1}{\sqrt{\pi}} \int_{-\infty}^{-x/\sqrt{2}} \exp\big(-(\frac{t}{\sqrt{2}})^2\big) d(\frac{t}{\sqrt{2}}) \\
&= \Phi(x) - \Phi(-x) = \Phi(x) - (1 - \Phi(x)) \\
&= 2 \cdot \Phi(x) - 1 \\
\Leftrightarrow \Phi(x) &= \frac{1}{2}(1 + \text{erf}(\frac{x}{\sqrt{2}}))
\end{aligned} \tag{2.33}$$

For Gaussian variables X with mean μ and standard deviation σ ($X \sim \mathcal{N}(\mu, \sigma^2)$), we have

$$\Pr[X \leq x] = \Phi(\frac{x-\mu}{\sigma}) = \frac{1}{2}(1 + \text{erf}(\frac{x-\mu}{\sigma\sqrt{2}})) \tag{2.34}$$

2.4 Conclusion

This chapter situated our work within the world of cryptography and introduced theoretical concepts that we will need for our analysis. The fundamental properties of lattices and ideal lattices were explained and we provided examples of well-known lattice problems. The basis reduction algorithm BKZ 2.0 is an essential component of most LWE-solving methods and will be studied further in depth.

CHAPTER 3

Security of the LWE-problem

In this chapter, we present the main parameters of Learning with Errors and the problem itself. We describe a variant of LWE in section 3.2: the Ring-Learning with Errors problem, which is supposedly easier to exploit, but not easier to break. After briefly reviewing the theoretical security proofs that made LWE so popular, three strategies for solving LWE in practice are thoroughly examined. We derive formulas for their computational costs and success probabilities in order to determine a security measure as a function of the main parameters.

3.1 The LWE-problem

The Learning with Errors (LWE) problem is characterized by a dimension $n \geq 1$, integer modulus $q \geq 2$ and an error distribution χ over $\mathbb{Z}$. This error distribution is typically a discrete Gaussian $D_{\mathbb{Z},s}$ with standard deviation $\sigma = \frac{s}{\sqrt{2\pi}}$.

Let $\mathbf{s} \in \mathbb{Z}_q^n$ be a secret vector of dimension n. An LWE-sample is constructed by choosing a uniformly random vector $\mathbf{a} \in \mathbb{Z}_q^n$ and an error term $e \leftarrow \chi$ and producing the pair $(\mathbf{a}, t = \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. The distribution consisting of these samples is called the LWE-distribution $A_{\mathbf{s},\chi} \subset \mathbb{Z}_q^n \times \mathbb{Z}_q$. For the sake of convenience, the LWE-problem is often expressed in matrix notation. For m LWE-samples $(\mathbf{a}_i, t_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \bmod q)$, let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be the matrix with columns $\mathbf{a}_i$ and let the values $e_i \leftarrow \chi$ and $t_i \in \mathbb{Z}_q$ be the entries of respectively $\mathbf{e} \in \mathbb{Z}^m$ and $\mathbf{t} \in \mathbb{Z}_q^m$. With these notations, two LWE problems are defined:

Definition 8 (LWE Decision Problem)**.** *Given* $(\boldsymbol{A}, \boldsymbol{t})$ *with* $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ *and* $\boldsymbol{t} \in \mathbb{Z}_q^m$*, determine whether* $\boldsymbol{t}$ *is chosen uniformly at random from* $\mathbb{Z}_q^m$ *or* $\boldsymbol{t} = \boldsymbol{A}^T \boldsymbol{s} + \boldsymbol{e} \bmod q$*.*

Definition 9 (LWE Search Problem)**.** *Given* $(\boldsymbol{A}, \boldsymbol{t})$ *with* $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ *and* $\boldsymbol{t} \in \mathbb{Z}_q^m$*, find* $\boldsymbol{s} \in \mathbb{Z}_q^n$ *such that* $\boldsymbol{t} = \boldsymbol{A}^T \boldsymbol{s} + \boldsymbol{e} \bmod q$*.*

Consider $\mathbf{z} = \mathbf{A}^T \mathbf{s} \bmod q \in \Lambda_q(\mathbf{A})$. The LWE Search-problem is related to a Bounded Distance Decoding (BDD) problem in $\Lambda_q(\mathbf{A})$ with $\mathbf{t} = \mathbf{z} + \mathbf{e} \bmod q$. When one can

solve the BDD problem and recover $\mathbf{z}$, finding $\mathbf{s}$ and thus solving the LWE Search problem is trivial.

3.1.1 The Hardness of LWE

The LWE problem is used in public cryptosystems because it is considered a hard problem. When he introduced the problem, Regev [Reg09] observed that it is as hard to solve as some worst-case lattice problems.

Theorem 2 (Theorem 1.1 from [Reg09]). *Consider integers n, q and a Gaussian parameter s such that $\frac{s}{q} \in (0,1)$ and $s > 2\sqrt{n}$. If there exists an efficient algorithm that solves the LWE Search problem with modulus q and error distribution $\chi = D_{\mathbb{Z},s}$, then there is an efficient quantum algorithm that approximates the GapSVP and the SIVP to within $O(\frac{nq}{s})$ in the worst case.*

As no such algorithms to solve GapSVP or SIVP exist, we may indeed assume that the LWE problem is hard and can even resist to quantum adversaries. Moreover, Peikert [Pei09] has shown that for large moduli $q \geq 2^{n/2}$, there is not only a quantum reduction from solving worst-case GapSVP to solving LWE, but also a classical probabilistic polynomial-time reduction, therefore assuring that we can also base the hardness results on weaker assumptions. Furthermore, both Regev and Peikert have demonstrated that the LWE Decision problem is at least as hard as the LWE Search problem. Lemma 6 show the reduction from Decision to Search LWE.

Lemma 6 (Lemma 4.2 from [Reg09]). *Let $n \geq 1$ be some integer, $2 \leq q \leq poly(n)$ be a prime and χ some distribution on $\mathbb{Z}_q$. Assume that we have access to a procedure W that for all $\boldsymbol{s}$, accepts with probability exponentially close to 1 on inputs from $A_{\boldsymbol{s},\chi}$ and rejects with probability exponentially close to 1 on inputs from $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$. Then, there exists an efficient algorithm W' that, given samples from $A_{\boldsymbol{s},\chi}$ for some $\boldsymbol{s}$, outputs $\boldsymbol{s}$ with probability exponentially close to 1.*

Proof [APS15]. We demonstrate how W' can recover the first component $\mathbf{s}_0$ of $\mathbf{s}$. The procedure for the other coordinates is similar. Consider some fixed $k \in \mathbb{Z}_q$ and a uniformly random $l \in \mathbb{Z}_q$. Given a sample $(\mathbf{a}, t)$, W' creates a different pair $(\mathbf{a}', t') = (\mathbf{a} + (l, 0, \ldots, 0), t + l \cdot k)$. If the first sample $(\mathbf{a}, t)$ is uniformly random, then so is the second. If $(\mathbf{a}, t)$ is drawn from $A_{\mathbf{s},\chi}$ and if $k = \mathbf{s}_0$, then $(\mathbf{a}', t')$ also follows $A_{\mathbf{s},\chi}$. On the other hand, if $k \neq \mathbf{s}_0$, then the transformation maps $A_{\mathbf{s},\chi}$ to the uniform distribution (thanks to the primality of q). Since there are only $|\mathbb{Z}_q| = q$ possibilities for k, we can try them all and feed $(\mathbf{a} + (l, 0, \ldots, 0), t + l \cdot k)$ to W. When W accepts $(\mathbf{a}', t')$ as a sample from $A_{\mathbf{s},\chi}$, we know that $k = \mathbf{s}_0$. □

The reduction from Search to Decision LWE is trivial. If we have access to an oracle that given samples $(\mathbf{a}, t)$ from $A_{\mathbf{s},\chi}$, returns the secret $\mathbf{s}$, we can use it to compute the errors $e = t - \langle \mathbf{a}, \mathbf{s} \rangle$. This result together with that of Lemma 6 shows the equivalence of Decision-LWE and Search-LWE.

The most interesting feature regarding LWE's security is the fact that solving the decision version of LWE becomes no easier when the secret $\mathbf{s}$ is chosen from the

error distribution χ instead of a uniformly random vector [ACPS09]. The following Lemma shows a reduction from small-secret LWE to normal LWE.

Lemma 7. *Let χ^n be an n-dimensional extension of χ, where each component is sampled according to χ. Given an LWE distribution $A_{\boldsymbol{s},\chi}$ with samples $(\boldsymbol{a}, t) = (\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ with $\boldsymbol{a}$ uniform $\in \mathbb{Z}_q^n$, $e \leftarrow \chi$ and $\boldsymbol{s} \in \mathbb{Z}_q^n$, there exists a deterministic polynomial transformation, mapping $A_{\boldsymbol{s},\chi}$ to another LWE distribution $A_{\boldsymbol{x},\chi}$ with $\boldsymbol{x} \leftarrow \chi^n$ and mapping the uniform distribution $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ onto itself.*

Proof [LP11]. Given access to $A_{\mathbf{s},\chi}$, one can draw samples $(\mathbf{a}_i, t_i)$ and form a square invertible matrix $\mathbf{A}_1 \in \mathbb{Z}_q^{n \times n}$ with columns $\mathbf{a}_i$ by discarding those samples that would make $\mathbf{A}_1$ singular. When $\mathbf{A}_1$ is complete, we draw more samples to form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} \mathbf{t}_1 \\ \mathbf{t}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \end{bmatrix} \mathbf{s} + \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} = \mathbf{A}^T \mathbf{s} + \mathbf{e} \bmod q \tag{3.1}$$

By construction, the matrix $\mathbf{A}_2$ is uniform, the entries from $\mathbf{e}$ are drawn from χ and $\mathbf{A}_1$ is square and invertible. Now we use the following transformation to convert the LWE samples $(\mathbf{A}, \mathbf{t})$ from $A_{\mathbf{s},\chi}$ to samples $(\bar{\mathbf{A}}, \bar{\mathbf{t}})$ from another LWE-distribution:

$$\bar{\mathbf{A}} \leftarrow -\mathbf{A}_1^{-1} \cdot \mathbf{A}_2 \bmod q \tag{3.2}$$

$$\begin{aligned} \bar{\mathbf{t}} \leftarrow \bar{\mathbf{A}}^T \mathbf{t}_1 + \mathbf{t}_2 &= -\mathbf{A}_2^T \cdot \mathbf{A}_1^{-T} (\mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1) + \mathbf{A}_2^T \mathbf{s} + \mathbf{e}_2 \\ &= -\mathbf{A}_2^T \mathbf{s} + \bar{\mathbf{A}}^T \mathbf{e}_1 + \mathbf{A}_2^T \mathbf{s} + \mathbf{e}_2 = \bar{\mathbf{A}}^T \mathbf{e}_1 + \mathbf{e}_2 \bmod q \end{aligned} \tag{3.3}$$

By construction, $\bar{\mathbf{A}}$ is uniform when $\mathbf{A}_2$ is uniform. We therefore have valid samples $(\bar{\mathbf{A}}, \bar{\mathbf{t}} = \bar{\mathbf{A}}^T \mathbf{e}_1 + \mathbf{e}_2)$ from a distribution $A_{\mathbf{x},\chi}$ with $\mathbf{x} \leftarrow \chi^n$ since we have successfully replaced $\mathbf{s}$ with $\mathbf{e}_1$. When we consider the same transformation on a uniformly random $(\mathbf{A}, \mathbf{t}) \in U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$, $(\bar{\mathbf{A}}, \bar{\mathbf{t}})$ is uniform as well. □

Lemma 7 shows that we can solve the LWE decision problem if we can solve the small-secret variant. Hence, under the hardness assumption of normal LWE, $(\bar{\mathbf{A}}, \bar{\mathbf{A}}^T \mathbf{e}_1 + \mathbf{e}_2 \bmod q)$ is indistinguishable from uniform. This result has beneficial consequences for the key sizes in LWE-based cryptosystems.

3.2 The Ring-LWE problem

The Ring-LWE problem (R-LWE) is a variant of the LWE-problem which exploits algebraic structure. It was first introduced by Lyubashevsky et al. [LPR10]. For cryptographic applications, using R-LWE instead of LWE leads to smaller key sizes and more efficient operations, while maintaining the cryptographic strength. A more detailed comparison to LWE follows later.

Using an irreducible polynomial $f(x) = x^n + 1 \in \mathbb{Z}[x]$ of degree $n = 2^k$, define the ring of integer polynomials modulo $f(x)$: $R = \mathbb{Z}[x]/f(x)$. Furthermore, consider the ring $R_q = R/qR = \mathbb{Z}_q[x]/f(x)$: Every element (or polynomial) in R_q has degree

at most $n-1$ and coefficients $\in \mathbb{Z}_q = \{0, \ldots, q-1\}$. The secret $s = s(x)$ is an element of R_q and the error distribution χ is once more a centered Discrete Gaussian. An R-LWE sample is a pair $(a(x), a(x)s(x) + e(x) \bmod f(x))$ with $a(x)$ uniformly random $\in R_q$ and $e(x) \leftarrow \chi$. The distribution consisting of such samples is the R-LWE distribution $A_{s(x),\chi} \subset R_q \times R_q$.

Definition 10 (R-LWE Decision Problem)**.** *Given independent samples in $R_q \times R_q$, determine whether they were drawn from $A_{s(x),\chi}$ or from the uniform distribution over $R_q \times R_q$.*

Definition 11 (R-LWE Search Problem)**.** *Given independent samples from the R-LWE distribution $A_{s(x),\chi}$, find $s(x) \in R_q$.*

As with other LWE problems, the error distribution χ is a centered Gaussian. In this case however, the distribution is defined over the embedding of R_q in $\mathbb{Z}^n$ and is thus no longer 1-dimensional. This detail leads to extra complications since in general, an n-dimensional Gaussian requires an $n \times n$ covariance matrix. Yet when $f(x) = x^n + 1$ with n a power of 2, the coefficients of an error $e(x) \leftarrow \chi$ are independent and χ can be considered the product distribution of n 1-dimensional Gaussians.[1]

Note that 1 sample from the R-LWE distribution replaces n samples from the LWE distribution. To demonstrate this, we examine the polynomial product $a(x) \cdot s(x) \bmod f(x)$ using the notion of rotations (2.18):

$$
\begin{aligned}
a(x) \cdot s(x) \bmod f(x) &= a(x) \cdot (s_0 + s_1 \cdot x + \ldots + s_{n-1} \cdot x^{n-1}) \bmod f(x) \\
&= a(x) \cdot s_0 + a(x) \cdot x \cdot s_1 + \ldots a(x) \cdot x^{n-1} \cdot s_{n-1} \bmod f(x) \\
&= \sum_{i=0}^{n-1} a(x) \cdot x^i \cdot s_i \bmod f(x) \\
&= \sum_{i=0}^{n-1} \mathrm{rot}^i(a(x)) \cdot s_i \bmod q \\
&= \begin{bmatrix} \mathbf{a} & \mathrm{rot}(\mathbf{a}) & \ldots & \mathrm{rot}^{n-1}(\mathbf{a}) \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix} \bmod q \\
&= \begin{bmatrix} a_0 & -a_{n-1} & \ldots & -a_1 \\ a_1 & a_0 & \ldots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \ldots & a_0 \end{bmatrix} \cdot \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix} \bmod q
\end{aligned}
$$

[1] To be more precise, only a canonical embedding from $R_q \to \mathbb{Z}_q^n$ (2.19) can lead to diagonal covariance matrixes. In general, with the coefficient embedding (2.17), the error distribution is a multivariate Gaussian with dependent coordinates. However, for the cyclotomic polynomials $f(x) = x^n + 1$ with n a power of 2, the canonical embedding and coefficient embedding are isometric [LPR10], so the error terms are independent after all.

$$= \mathbf{A}^T\mathbf{s} \bmod q \tag{3.4}$$

We define $\mathbf{a} = [a_0, a_1, \ldots, a_{n-1}]^T$, $\mathbf{s} = [s_0, s_1, \ldots, s_{n-1}]^T$ and $\mathbf{e} = [e_0, e_1, \ldots, e_{n-1}]^T$ the respective coefficient embeddings of $a(x)$, $s(x)$ and $e(x)$ and the matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}^T \\ \mathrm{rot}(\mathbf{a})^T \\ \vdots \\ \mathrm{rot}^{n-1}(\mathbf{a})^T \end{bmatrix} \in \mathbb{Z}^{n\times n}$$

Recalling from section 2.2.3 that $\mathbf{A}$ forms a full rank basis for an ideal lattice, we may conclude from (3.4) that one R-LWE sample $(a, t) = (a, as + e \bmod f(x))$ is equivalent to n LWE samples $(\mathbf{A}, \mathbf{t}) = (\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q)$. When extending (3.4) to multiple samples, the number of samples needed for R-LWE is significantly lower than for LWE. From l R-LWE samples, one obtains a q-ary lattice of dimension $m = l \cdot n$ (*i.e.* $l << m$).

The above enables us to explain why R-LWE is more efficient than LWE. Firstly, the storage of the lattice basis $\mathbf{A}$ is more efficient because we are dealing with an ideal lattice and we only have to store $\mathbf{a}$. Furthermore, each sample from R-LWE gives n pseudorandom values $\in \mathbb{Z}_q$ instead of just one scalar, while the cost of generating each instance is still small: The Fast Fourier Transform (FFT) allows multiplications in R_q to be evaluated with only $\mathcal{O}(n \log n)$ operations in $\mathbb{F}_q$. Finally, in cryptographic applications, where a public key is constructed from multiple LWE-samples, the size of the public key with R-LWE is thus reduced with a factor n.

Even with these advantages, Lyubashevsky et al. have proven that R-LWE shows hardness properties similar to those of LWE.

Theorem 3 ([LPR10])**.** *Suppose it is hard for polynomial-time quantum algorithms to approximate (the search version of) the shortest vector problem (SVP) in the worst case in ideal lattices in R to within a fixed poly(n) factor. Then, any poly(n) number of samples drawn from the R-LWE distribution $A_{s,\chi}$ are pseudorandom to any polynomial-time (possibly quantum) attacker.*

Furthermore, they showed that the R-LWE Search problem reduces to the R-LWE Decision problem.

3.3 Known attacks against LWE

We have shown two distinct LWE-problems and correspondingly, one can differentiate two kinds of attacks: a *distinguishing* attack (against the LWE Decision problem) and a *decoding* attack (against the LWE Search problem). Because of the equivalence of the problems, any method that solves Decision-LWE or Search-LWE, solves LWE in general.

3.3.1 The Distinguishing attack

The following attack was first described by Micciancio and Regev [MR09]. Suppose we have a short nonzero integral vector $\mathbf{v} \in \Lambda_q^\perp(\mathbf{A})$. To distinguish an LWE instance $(\mathbf{A}, \mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q)$ from uniformly random, one should test whether the inner product $\langle \mathbf{v}, \mathbf{t} \rangle$ is "close" to 0 mod q (*i.e.* $|\langle \mathbf{v}, \mathbf{t} \rangle| < \frac{q}{4}$). More specifically, when $\mathbf{t}$ is an LWE sample, we know by definition of $\Lambda_q^\perp(\mathbf{A})$ that $\mathbf{v}^T\mathbf{A}^T = (\mathbf{A}\mathbf{v})^T = 0 \pmod q$ so $\langle \mathbf{v}, \mathbf{t} \rangle = \mathbf{v}^T\mathbf{A}^T\mathbf{s} + \mathbf{v}^T\mathbf{e} \bmod q = \langle \mathbf{v}, \mathbf{e} \rangle \bmod q$. With every entry of $\mathbf{e}$ drawn from the error distribution and $\mathbf{v}$ a short vector, one can expect $\langle \mathbf{v}, \mathbf{e} \rangle$ to be small and therefore $\mathbf{t}$ to pass the test. A uniform $\mathbf{t}$ on the other hand, only passes the test with probability $\frac{1}{2}$.

This method is often associated with the Short Integer Solution (SIS) Problem.

Definition 12 (Short Integer Solution (SIS) Problem)**.** *Given m vectors $\boldsymbol{a}_i \in \mathbb{Z}_q^n$, find a non-trivial short solution $\boldsymbol{z} \in \mathbb{Z}^m$ such that $\sum_{i=1}^m \boldsymbol{a}_i z_i = 0 \in \mathbb{Z}_q^n$.*

One can immediately observe that a solution to the SIS problem is a short vector in $\Lambda_q^\perp(\mathbf{A})$ and thus a suitable candidate for $\mathbf{v}$.

Lemma 8 (The Advantage of the SIS-based distinguishing attack)**.** *Given LWE samples $(\boldsymbol{A}, \boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e} \bmod q)$ with $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and a short nonzero integral vector $\boldsymbol{v} \in \Lambda_q^\perp(\boldsymbol{A})$. The distribution $\langle \boldsymbol{v}, \boldsymbol{e} \rangle \bmod q$ can be distinguished from uniform with advantage close to $\exp\left(-\pi(\frac{\|\boldsymbol{v}\|s}{q})^2\right)$ when $\|\boldsymbol{v}\|s \leq q$.*

Proof. We investigate the advantage of this method by calculating the statistical distance between $\langle \mathbf{v}, \mathbf{e} \rangle \bmod q$ and the uniform distribution. We know that each e_i follows a discrete Gaussian distribution:

$$\langle \mathbf{v}, \mathbf{e} \rangle = \sum_i v_i e_i \text{ with } \forall i : e_i \sim D_{\mathbb{Z},s} \tag{3.5}$$

As a linear combination of identically distributed Gaussians, the inner product $\langle \mathbf{v}, \mathbf{e} \rangle$ is Gaussian distributed itself. Each e_i has mean 0 and standard deviation σ. The inner product distribution is thus centered around zero as well and the standard deviation follows from an easy computation:

$$\sqrt{\sum_i v_i^2 \sigma^2} = \sigma \sqrt{\sum_i v_i^2} = \sigma \|\mathbf{v}\| \tag{3.6}$$

We therefore have a discrete Gaussian with width parameter $\|\mathbf{v}\|s$ reduced modulo q when $\mathbf{t}$ comes from an LWE-sample:

$$\langle \mathbf{v}, \mathbf{e} \rangle \bmod q \sim D_{\mathbb{Z}, \|\mathbf{v}\|s} \bmod q\mathbb{Z} \tag{3.7}$$

As was explained in section 2.3, a Gaussian modulo a lattice Λ' is *smoothed* when the Gaussian parameter exceeds the inverse of the shortest vector length of the dual lattice $\frac{1}{\lambda_1(\Lambda'^*)}$. Thanks to Lemma 4, we may expect the same behaviour for discrete

Gaussians. In particular, we consider here the distribution $D_{\Lambda,\|\mathbf{v}\|s}$ mod Λ' with $\Lambda = \mathbb{Z}$ and $\Lambda' = q\mathbb{Z}$, so we know that $\langle \mathbf{v}, \mathbf{e} \rangle$ mod q is smoothed for $\|\mathbf{v}\|s > \frac{1}{\lambda_1((q\mathbb{Z})^*)} = q$:

$$(\Lambda')^* = (q\mathbb{Z})^* = \frac{1}{q}\mathbb{Z} \text{ and } \lambda_1((\Lambda')^*) = \lambda_1(\frac{1}{q}\mathbb{Z}) = \frac{1}{q} \tag{3.8}$$

From (2.28), we learned that the statistical distance between $\langle \mathbf{v}, \mathbf{e} \rangle$ mod q and the uniform distribution is bounded by $\frac{1}{2}S_{\|\mathbf{v}\|s}(\Lambda')$ with

$$S_{\|\mathbf{v}\|s}(\Lambda') = S_{\|\mathbf{v}\|s}(q\mathbb{Z}) = \sum_{\mathbf{w}\in(q\mathbb{Z})^*\setminus\{\mathbf{0}\}} \rho_{1/(\|\mathbf{v}\|s)}(\mathbf{w}) = \sum_{\mathbf{w}\in\frac{1}{q}\mathbb{Z}\setminus\{\mathbf{0}\}} \exp\big(-\pi(\|\mathbf{v}\|s\|\mathbf{w}\|)^2\big) \tag{3.9}$$

Knowing that $\forall \mathbf{w} \in \frac{1}{q}\mathbb{Z} \setminus \{\mathbf{0}\} : \|\mathbf{w}\| \geq \lambda_1(\frac{1}{q}\mathbb{Z}) = \frac{1}{q}$, every term in the above sum is bounded from above by $\exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big)$. Furthermore, let us consider the vector $\mathbf{w}' \in \frac{1}{q}\mathbb{Z}$ with length $\|\mathbf{w}'\| = \lambda_1(\frac{1}{q}\mathbb{Z}) = \frac{1}{q}$. The following result then follows from the fact that each summand in (3.9) is non-negative:

$$\sum_{\mathbf{w}\in\frac{1}{q}\mathbb{Z}\setminus\{\mathbf{0}\}} \rho_{1/(\|\mathbf{v}\|s)}(\mathbf{w}) \geq \rho_{1/(\|\mathbf{v}\|s)}(\mathbf{w}') = \exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big) \tag{3.10}$$

Hence, there exists a positive constant $C < \infty$ such that

$$\exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big) \leq S_{\|\mathbf{v}\|s}(q\mathbb{Z}) \leq C \cdot \exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big) \tag{3.11}$$

The above allows us to conclude that when the Gaussian parameter $\|\mathbf{v}\|s$ is smaller than q, the distribution $\langle \mathbf{v}, \mathbf{e} \rangle$ mod q is not smoothed and can be distinguished from uniform with advantage close to $\exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big)$. □

This advantage depends strongly on the quality of the short vector $\mathbf{v}$, so a lattice basis reduction is the first step in the attack. It follows from equations (2.8) and (2.14) that the shortest length we can achieve for $\mathbf{v} \in \Lambda_q^\perp(\mathbf{A})$ using basis reduction is approximately

$$\|\mathbf{v}\| = \min\{q, \delta^m \det(\Lambda_q^\perp(\mathbf{A}))^{1/m}\} = \min\{q, \delta^m q^{n/m}\} \tag{3.12}$$

with δ depending strongly on the reduction algorithm used. By computing the derivative of $\delta^m q^{n/m}$ with respect to m, Micciancio and Regev [MR09] have deduced an optimal subdimension $m = \sqrt{n\frac{\log_2 q}{\log_2 \delta}}$ for which a minimal length of $\|\mathbf{v}\|$ can be found.

$$\left(\delta^m q^{n/m}\right)' = \delta^m q^{n/m}(\log_2 \delta - \frac{n}{m^2}\log_2 q) = 0 \Leftrightarrow m = \sqrt{n\frac{\log_2 q}{\log_2 \delta}} \tag{3.13}$$

A higher dimension would prevent the reduction algorithm from finding short vectors whereas a lower dimension would lead to a too sparse lattice with too few short

vectors. By substituting the optimal subdimension in (3.12) and using $\delta = 2^{\log_2 \delta}$ and $q = 2^{\log_2 q}$, one obtains a shortest vector of length at least

$$\min\{q, 2^{2\sqrt{n \log_2 q \log_2 \delta}}\} \tag{3.14}$$

We recall that m is the number of columns in $\mathbf{A}$ and thus the number of LWE-samples used. If the number of available samples is higher than the optimal subdimension, some of these samples can simply be omitted. On the other hand, lacking sufficient samples, the optimal subdimension cannot be used.

3.3.2 The Bounded Distance Decoding attack

For the *decoding* attack, we explore existing solutions to the BDD problem, as solving the latter implies finding a solution to the LWE Search problem. Decoding attacks involve a basis reduction in the first step and a post-reduction step to find the solution. For the first step, the BKZ 2.0 algorithm is used as it is currently deemed the most practical alternative. For the second step, Lindner and Peikert [LP11] have adapted the NearestPlane algorithm of Babai [Bab86]. Their version has been further randomized by Liu and Nguyen [LN13].

The NearestPlane algorithm of Babai [Bab86] This algorithm is a post-reduction effort to obtain the solution of a BDD problem. Given the basis $\mathbf{B} = \langle \mathbf{b}_1, \ldots, \mathbf{b}_m \rangle$ of a lattice and a target point $\mathbf{t}$, it returns the unique vector $\mathbf{z}$ such that $\mathbf{z} - \mathbf{t} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ with $\tilde{\mathbf{B}} = \langle \tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_m \rangle$ the Gram-Schmidt orthogonalization of $\mathbf{B}$.

Algorithm 2 Babai's NearestPlane algorithm [LN13]

Input: m-dimensional lattice basis $\mathbf{B}$ and target point $\mathbf{t}$
Output: $\mathbf{z} \in \Lambda(\mathbf{B})$ s.t. $\mathbf{z} - \mathbf{t} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}})$

1: $\mathbf{z} \leftarrow 0$
2: **for** $i = m$ to 1 **do**
3: Compute $c \in \mathbb{Z}$ closest to $\frac{\langle \tilde{\mathbf{b}}_i, \mathbf{t} \rangle}{\langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{b}}_i \rangle}$
4: $\mathbf{t} \leftarrow \mathbf{t} - c\mathbf{b}_i$
5: $\mathbf{z} \leftarrow \mathbf{z} + c\mathbf{b}_i$
6: **end for**

The problem with this algorithm is that it only solves the BDD problem (and in our case the LWE problem) if $\mathbf{t} - \mathbf{z} = \mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}})$, which is not often the case. Since it is not yet known how to efficiently compute the success probability, Lindner-Peikert [LP11] have approximated it by replacing the distribution χ with a continuous Gaussian of mean 0 and standard deviation $\sigma = \frac{s}{\sqrt{2\pi}}$. By (2.32), we may assume the same results for a discrete Gaussian when s is sufficiently large. We can refer to Table 2.1 for a custom choice of bound on s. However, Lindner and Peikert suggest to have s not smaller than 8 to avoid an attack proposed by Arora and Ge in [AG11], which uses non-linear polynomials of a degree depending on the parameter s.

If we may indeed assume that each e_j is normal distributed with the same standard deviation σ, then $\langle \mathbf{e}, \tilde{\mathbf{b}}_i \rangle$ is a Gaussian with standard deviation $\|\tilde{\mathbf{b}}_i\|\sigma$.

$$\Pr[\mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}})] = \prod_{i=1}^{m} \Pr[|\langle \mathbf{e}, \tilde{\mathbf{b}}_i \rangle| < \frac{\|\tilde{\mathbf{b}}_i\|^2}{2}] = \prod_{i=1}^{m} \Big[\Phi\Big(\frac{\|\tilde{\mathbf{b}}_i\|^2/2}{\|\tilde{\mathbf{b}}_i\|\sigma}\Big) - \Phi\Big(\frac{-\|\tilde{\mathbf{b}}_i\|^2/2}{\|\tilde{\mathbf{b}}_i\|\sigma}\Big)\Big] \tag{3.15}$$

Thanks to the orthogonality of the Gram-Schmidt vectors $\tilde{\mathbf{b}}_i$, the values $\langle \mathbf{e}, \tilde{\mathbf{b}}_i \rangle$ can be considered independently. Recall from (2.33) that that $\Phi(x) - \Phi(-x) = \mathrm{erf}(\frac{x}{\sqrt{2}})$. Therefore,

$$\Pr[\mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}})] = \prod_{i=1}^{m} \Big[\Phi\Big(\frac{\|\tilde{\mathbf{b}}_i\|\sqrt{2\pi}}{2s}\Big) - \Phi\Big(-\frac{\|\tilde{\mathbf{b}}_i\|\sqrt{2\pi}}{2s}\Big)\Big] = \prod_{i=1}^{m} \mathrm{erf}\Big(\frac{\|\tilde{\mathbf{b}}_i\|\sqrt{\pi}}{2s}\Big) \tag{3.16}$$

With an intuitive interpretation, it can be shown why this probability is usually very small. In a reduced basis, the first Gram-Schmidt vectors are relatively long, whereas the last few are quite short. This causes the parallelepiped to be "long" and "skinny" which makes it unlikely to contain $\mathbf{e}$. For this reason, Lindner and Peikert have adapted the algorithm.

The Lindner-Peikert NearestPlanes algorithm [LP11] Lindner and Peikert have enhanced the success probability of the algorithm at the expense of running time by making the parallelepiped $\mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ wider in some directions. Their algorithm is parametrized by a vector $\mathbf{d} \in \mathbb{Z}^m$ that assigns a multiplication factor d_i to each direction $\tilde{\mathbf{b}}_i$. The entries of the vector are chosen such as to maximize $\min_i(d_i \cdot \|\tilde{\mathbf{b}}_i\|)$.

Algorithm 3 NearestPlanes algorithm [LN13]

Input: m-dimensional lattice basis $\mathbf{B}$, vector $\mathbf{d} \in \mathbb{Z}^m$ and target point $\mathbf{t}$
Output: A set of $\prod_{i=1}^{m} d_i$ distinct lattice vectors $\in \Lambda(\mathbf{B})$ close to $\mathbf{t}$

1: **if** $m = 0$ **then**
2: **return** 0
3: **else**
4: Compute the d_m integers $c_1, c_2, \ldots, c_{d_m} \in \mathbb{Z}$ closest to $\langle \tilde{\mathbf{b}}_m, \mathbf{t} \rangle / \langle \tilde{\mathbf{b}}_m, \tilde{\mathbf{b}}_m \rangle$
5: **return** $\bigcup_{i=1}^{d_m} \Big(c_i \mathbf{b}_m + \text{NearestPlanes}\{\mathbf{b}_1, \ldots, \mathbf{b}_{m-1}, (d_1, \ldots, d_{m-1}), \mathbf{t} - c_i \mathbf{b}_m\}\Big)$
6: **end if**

It should be noted that not only this algorithm is recursive, but that the recursive calls can be run entirely in parallel. Therefore, the following lemma was given for the runtime.

Lemma 9 ([LP11]). *For $\boldsymbol{t} \in \mathrm{span}(\boldsymbol{B})$, $\mathrm{NearestPlanes}(\boldsymbol{B}, \boldsymbol{d}, \boldsymbol{t})$ returns the set of all $\boldsymbol{z} \in \Lambda(\boldsymbol{B})$ such that $\boldsymbol{t} \in \boldsymbol{z} + \mathcal{P}_{1/2}(\tilde{\boldsymbol{B}} \cdot \boldsymbol{D})$ where $\boldsymbol{D} = \mathrm{diag}(\boldsymbol{d})$. The running time is essentially $\prod_{i=1}^{m} d_i$ times as large as that of $\mathrm{NearestPlane}(\boldsymbol{B}, \boldsymbol{t})$.*

For the success probability, Lindner and Peikert again replace the error distribution with a continuous Gaussian. If $\mathbf{t} = \mathbf{z} + \mathbf{e}$, the probability that $\mathbf{z}$ is in the output set

of NearestPlanes$(\mathbf{B}, \mathbf{d}, \mathbf{t})$ is

$$\Pr[\mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}} \cdot \mathrm{diag}(\mathbf{d}))] = \prod_{i=1}^{m} \Pr[|\langle \mathbf{e}, \tilde{\mathbf{b}}_i \rangle| < d_i \cdot \langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{b}}_i \rangle / 2] = \prod_{i=1}^{m} \mathrm{erf}\left(\frac{d_i \cdot \|\tilde{\mathbf{b}}_i\|\sqrt{\pi}}{2s}\right) \tag{3.17}$$

This expression shows the adversary's trade-off between the time he spends on basis reduction on the one hand and on nearest planes executions on the other hand. The factors d_i make sure that the success probability is reasonable, even for a basis of bad quality. On the other hand, when the basis quality is good and the basis vectors are almost orthogonal, there is no need for large factors d_i. The more time the attacker spends on basis reduction, the less effort is needed for the Nearest Planes part.

According to Liu and Nguyen [LN13], this NearestPlanes algorithm is essentially the same as pruned lattice enumeration, because it enumerates all $\mathbf{z}$ within the parallelepiped $\mathcal{P}_{1/2}(\mathrm{diag}(\mathbf{d}) \cdot \tilde{\mathbf{B}})$ centered around $\mathbf{t}$, using radius $R = \frac{1}{2}\sqrt{\sum_{i=1}^{m} d_i^2 \|\tilde{\mathbf{b}}_i\|^2}$. If the i-th coordinate of a vector $\mathbf{x}$ in a normalized Gram-Schmidt basis is defined as

$$\zeta_i(\mathbf{x}) = \frac{\langle \mathbf{x}, \tilde{\mathbf{b}}_i \rangle}{\|\tilde{\mathbf{b}}_i\|^2} \tag{3.18}$$

then at each tree level k, the algorithm only considers those vectors $\mathbf{z} \in \Lambda$ such that $|\zeta_i(\mathbf{z}-\mathbf{t})| \leq d_i \|\tilde{\mathbf{b}}_i\|/2$ for all $i \geq m+1-k$. Therefore, the NearestPlanes enumeration tree is a subset of a Gamma-Nguyen-Regev enumeration tree.

Randomizing the NearestPlanes Algorithm [LN13] Based on the similarities to enumeration, Liu and Nguyen have proposed two optimizations for the algorithm. Firstly, they suggest omitting the parallelepiped $\mathcal{P}_{1/2}(\mathrm{diag}(\mathbf{d}) \cdot \tilde{\mathbf{B}})$ as restriction and using arbitrary bounds R_i for $|\zeta_i(\mathbf{z} - \mathbf{t})|$ instead:

$$|\zeta_i(\mathbf{z} - \mathbf{t})| \leq R_i \text{ instead of } |\zeta_i(\mathbf{z} - \mathbf{t})| \leq d_i \|\tilde{\mathbf{b}}_i\|/2 \tag{3.19}$$

It is indeed possible to choose a constant bound R for all directions such that $d_i = \lceil \frac{R}{\|\tilde{\mathbf{b}}_i\|} \rceil$ in order for the search space to be as square as possible. The success probability then becomes

$$\Pr[\mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}} \cdot \mathrm{diag}(\mathbf{d}))] = \prod_{i=1}^{m} \mathrm{erf}\left(\lceil \frac{R}{\|\tilde{\mathbf{b}}_i\|} \rceil \frac{\|\tilde{\mathbf{b}}_i\|\sqrt{\pi}}{2s}\right) \geq \mathrm{erf}\left(\frac{R\sqrt{\pi}}{2s}\right)^m \tag{3.20}$$

with the number of Nearest Plane runs $= \prod_{i=1}^{m} d_i = \prod_{i=1}^{m} \lceil \frac{R}{\|\tilde{\mathbf{b}}_i\|} \rceil$. Since the last Gram Schmidt vectors are shorter than the first ones, the vector $\mathbf{d}$ looks like

$$[1, 1, \ldots, 1, 2, \ldots, 2, 3, \ldots]$$

Secondly, Liu and Nguyen repeat the algorithm multiple times in order to augment the success probability. They believe that using l different randomized bases causes a multiplication by l for both the running time and success probability.

This extra freedom allows for better trade-offs between BKZ2.0 reduction and NearestPlane running times. Consider a constant $c < 1$ such that $R_i = c \cdot d_i \|\tilde{\mathbf{b}}_i\|/2$.

Using $c \cdot d_i$ instead of d_i makes the algorithm $1/c$ times faster at the cost of the success probability. However, the decrease in success probability is less than the speed-up $1/c$. Therefore, repeating the faster algorithm a sufficient number of times to regain a good success probability actually leads to a smaller total cost. Mathematically, we see that for $c < 1$:

$$c \cdot \mathrm{erf}(a) < \mathrm{erf}(c \cdot a) < \mathrm{erf}(a)$$

so the new success probability is more than c times the original one. In other words, with the smaller search space, the algorithm is $\frac{1}{c}$ times faster while at the same time $\frac{1}{c}$ times the new success probability exceeds the old one.

$$\mathrm{erf}(a) < \frac{\mathrm{erf}(c \cdot a)}{c} \tag{3.21}$$

The problem with this approach is the uncertainty about whether or not subsequent executions independently result in the same success probability ϵ. From the GSA, we know the following:

$$\|\mathbf{b}_1\| = \delta^m \det(\mathbf{B})^{1/m} \tag{2.13}$$
$$\|\tilde{\mathbf{b}}_i\| = \|\mathbf{b}_1\| \cdot \alpha^{i-1} \tag{2.15}$$

with $\alpha = \delta^{-2m/(m-1)}$ (2.16). This means that the Nearest Planes search space is almost completely determined by δ. Does this mean the adversary's search spaces in different executions are not independent?

Let's assume these search spaces (fundamental parallelepipeds for different lattice bases) are m-dimensional cubes with side-length r around the origin. Such a cube has 2^m vertices and if $r = 1$, the distance from the origin to such a vertex is $\sqrt{m}/2$.

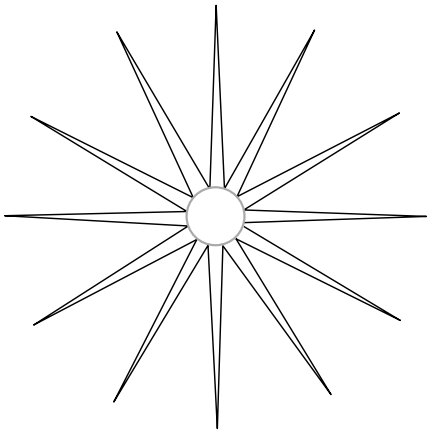


FIGURE 3.1. The 2^m spikes of a unit hypercube [HN89]. Each spike has length $\frac{\sqrt{m}}{2}$

We investigate what these cubes have in common: the inscribed sphere of radius $R = \frac{r}{2}$. Its volume in dimension m is given by

$$V_m(R) = \frac{2\pi^{m/2} R^m}{m\Gamma(m/2)}$$

whereas that of the cubes is simply r^m. In Figure 3.2, we demonstrate the evolution of these volumes for growing dimension m. Furthermore, we plot the ratio of the cube's volume that is located outside the sphere $(r^m - V_m)/(r^m)$.

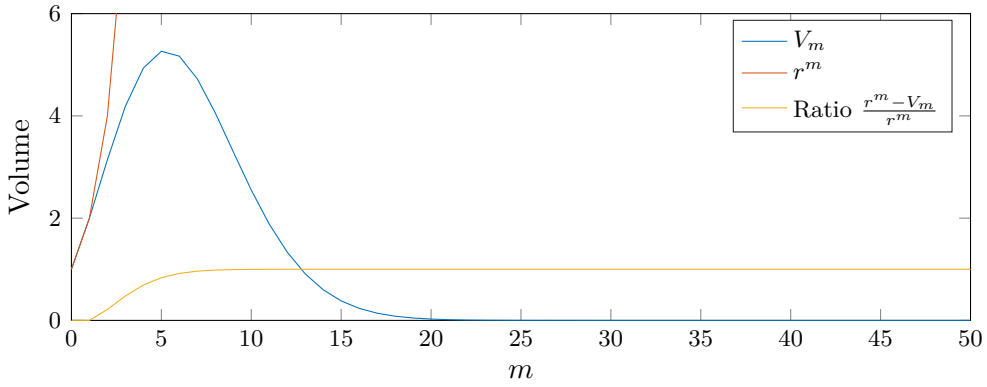


Figure 3.2. Evolution of the volumes of m-dimensional cubes and inscribed spheres when $R = 1$ ($\Leftrightarrow r = 2$)

The volume of m-dimensional spheres goes to zero very quickly for $m \to \infty$ and as a result, the ratio $\frac{r^m - V_m}{r^m}$ becomes 1. The volume of the cubes is thus located almost completely outside the inscribed sphere. This doesn't necessarily imply that the m-dimensional search spaces of bounded distance decoding are completely independent, but we assume that this is approximately the case and that the attacker can obtain an overall success probability $\mathcal{O}(1)$ by performing the attack ϵ^{-1} times with $\epsilon = \Pr[\mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}} \cdot \mathrm{diag}(\mathbf{d}))]$ in each execution.

If, on the other hand, the number of available LWE samples is not limited, the attack can be repeated with different lattices. Randomizing the lattice basis is in that case no longer necessary. Each subset of m LWE instances $(\mathbf{A}_i, \mathbf{A}_i^T\mathbf{s} + \mathbf{e}_i \bmod q)$ can form a different lattice for the same secret $\mathbf{s}$. With an unbounded number of samples, an adversary can create as many such lattices as necessary.

3.3.3 The BKW Algorithm

Another attack strategy worth mentioning is that of the Blum-Kalai-Wasserman (BKW) algorithm [BKW03], which was originally conceived for the Learning Parity with Noise (LPN) Problem. Since LWE is basically a generalization of LPN for larger moduli, Albrecht et al. studied the application of this algorithm to LWE in [ACF+15] and for some parameter sets, the algorithm performs better than BDD. With BKW, the world of lattices is abandoned and the problem $\mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q$ is considered as a noisy linear system of equations with unknown secret $\mathbf{s}$. Without the noise, one could use Gaussian elimination to transform the system to a triangular one and solve it using backward substitution. Albrecht [ACF+15] identifies three similar stages in his algorithm:

1. Sample reduction: a blocked version of Gaussian elimination

2. Hypothesis testing: testing candidate sub-solutions to recover a component of the secret $\mathbf{s}$
3. Back substitution: use the partial result to reduce the problem to a smaller one and repeat

For the remainder of this section, it is assumed that the adversary has unlimited access to LWE samples.

Sample Reduction As with Gaussian elimination, the first stage of the algorithm attempts to transform the system to triangular form. Instead of working component-wise, one chooses a parameter $b \leq n$, indicating a block-size of components to convert to zeros in each iteration. Given an oracle $A_{\mathbf{s},\chi}$ that produces samples from the LWE distribution, one can build oracles $B_{\mathbf{s},\chi,i}$, generating new samples $(\mathbf{a}, t)$ in which the first i blocks (thus the first $b \cdot i$ coordinates) of $\mathbf{a}$ are zero. Consider the following division of n by b, that is slightly different from the Euclidean division:

$$n = (d-1) \cdot b + r \text{ with } d = \lceil \frac{n}{b} \rceil \text{ and } 0 < r \leq b \tag{3.22}$$

The parameter d is the addition depth, signifying that the oracles must be constructed up to $B_{\mathbf{s},\chi,d-1}$ to obtain samples with $0 < r \leq b$ nonzero components in $\mathbf{a}$. Albrecht's procedure to construct these oracles goes as follows [ACF+15]:

> The first oracle $B_{\mathbf{s},\chi,0}$ is identical to the LWE oracle $A_{\mathbf{s},\chi}$.
>
> For $1 \leq i < d$: To create a sample of $B_{\mathbf{s},\chi,i}$, query the previous oracle $B_{\mathbf{s},\chi,i-1}$ repeatedly and fill a table T^i with the samples $(\mathbf{a}, t)$, indexed on the elements in the i^{th} block: $(\mathbf{a}_{(i-1)\cdot b}, \mathbf{a}_{(i-1)\cdot b+1}, \ldots, \mathbf{a}_{i\cdot b-1})$. By construction, the previous elements of $\mathbf{a}$ are already zero. When you obtain a sample $(\mathbf{a}', t')$, whose i^{th} block agrees with that of a sample $(\mathbf{a}, t)$ already in T^i, a collision is found. Use these samples to construct a new sample $(\mathbf{a} \pm \mathbf{a}', t \pm t')$ of oracle $B_{\mathbf{s},\chi,i}$. Thanks to the symmetry of the LWE problem, sign changes may be ignored.

Lemma 10 considers the complexity of this procedure.

Lemma 10 (Complexity of Sample Reduction [DTV15])**.** *Let n, q be positive integers and $A_{\boldsymbol{s},\chi}$ an LWE oracle, where $\boldsymbol{s} \in \mathbb{Z}_q^n$. Let $d \in \mathbb{Z}$ with $1 \leq d \leq n$, let b be such that $(d-1)b < n \leq db$ [2] and let $r = n - (d-1)b$. The worst case cost of obtaining m samples $(\boldsymbol{a}_j, t_j)$ from oracle $B_{\boldsymbol{s},\chi,d-1}$, where the $\boldsymbol{a}_j$ are zero for all but the first r elements, is upper bounded by*

$$\Big(\frac{q^b-1}{2}\Big)\Big(\frac{(d-1)(d-2)}{2}(n+1) - \frac{db(d-1)(d-2)}{6}\Big) + m\Big(\frac{d-1}{2}(n+2)\Big)$$

additions in $\mathbb{Z}_q$ and $(d-1)\frac{q^b-1}{2} + m$ calls to $A_{\boldsymbol{s},\chi}$.

The memory required in the worst case to store the set of tables T^1 through T^{d-1}, expressed in elements in $\mathbb{Z}_q$ is upper bounded by

$$\Big(\frac{q^b-1}{2}(d-1)\Big(n+1-\frac{d-2}{2}b\Big)\Big).$$

[2] The original Lemma says $db \leq n$, but this is probably a mistake as it doesn't comply with (3.22)

Proof [DTV15]. We first investigate the cost of constructing the tables. Because of the symmetry of LWE, we need at most $\frac{q^b-1}{2}$ samples from $B_{\mathbf{s},\chi,i-1}$ to fill T^i. Querying $B_{\mathbf{s},\chi,i-1}$ once requires the construction of T^{i-1} and the addition of two samples with $(i-1)\cdot b$ components in common, *i.e.* $n+1-(i-1)b$ additions in $\mathbb{Z}_q$. We prove by induction that the construction of table T^i requires

$$\left(\frac{q^b-1}{2}\right)\cdot(i-1)\cdot\left((n+1)-\frac{i}{2}b\right) \tag{3.23}$$

additions in $\mathbb{Z}_q$:

The construction of table T^1 demands up to $\frac{q^b-1}{2}$ samples from $B_{\mathbf{s},\chi,0}$ and thus no additions. Constructing T^2 requires at most $\frac{q^b-1}{2}$ queries from $B_{\mathbf{s},\chi,1}$. Each output from $B_{\mathbf{s},\chi,1}$ is the addition of $n+1-b$ elements in $\mathbb{Z}_q$. The cost of composing T^2 is thus $\frac{q^b-1}{2}\cdot(n+1-b)$ additions in $\mathbb{Z}_q$, equivalent to the expression in (3.23) for $i=2$.

We now assume that (3.23) holds the cost of creating T^i and derive the number of additions required for constructing T^{i+1}: We need up to $\frac{q^b-1}{2}$ samples from $B_{\mathbf{s},\chi,i}$ to fill this table, each of which requires table T^i and the addition of $n+1-ib$ elements in $\mathbb{Z}_q$. We add these $\frac{q^b-1}{2}(n+1-ib)$ additions to the cost of composing T^i:

$$\begin{aligned}
&\left(\frac{q^b-1}{2}\right)\cdot\left((i-1)\cdot\left((n+1)-\frac{i}{2}b\right)+(n+1)-ib\right)\\
=&\left(\frac{q^b-1}{2}\right)\cdot\left(i(n+1)-ib(\frac{i-1}{2}+1)\right)\\
=&\left(\frac{q^b-1}{2}\right)\cdot i\cdot\left((n+1)-\frac{i+1}{2}b\right)
\end{aligned}$$

thus proving (3.23). The total number of additions for constructing tables T^1 through T^{d-1} is then

$$\begin{aligned}
&\left(\frac{q^b-1}{2}\right)\cdot\sum_{i=1}^{d-1}(i-1)\cdot(n+1-\frac{i}{2}b)\\
=&\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{(d-1)(d-2)}{2}(n+1)-b\sum_{i=1}^{d-2}\frac{j(j+1)}{2}\right)\\
=&\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{(d-1)(d-2)}{2}(n+1)-\frac{bd(d-1)(d-2)}{6}\right)
\end{aligned}$$

Furthermore, when all tables are complete, each query to $B_{\mathbf{s},\chi,i}$ requires $n+1-ib$ additions and a query to $B_{\mathbf{s},\chi,i-1}$. The complexity of querying $B_{\mathbf{s},\chi,d-1}$ m times thus results recursively in $m\cdot\sum_{j=1}^{d-1}(n+1-jb) < m(d-1)(n+1-\frac{d}{2}) = m\frac{d-1}{2}(n+2)$ additions in $\mathbb{Z}_q$ and the creation of tables T^1 through T^{d-1}.

Proving the maximum number of calls to $A_{\mathbf{s},\chi}$ for m queries to $B_{\mathbf{s},\chi,d-1}$ is trivial as each table requires up to $\frac{q^b-1}{2}$ calls.

Finally, table T^i has up to $\frac{q^b-1}{2}$ rows of $n+1-(i-1)b$ nonzero elements in $\mathbb{Z}_q$. In total, we need to store

$$\begin{aligned}&\Big(\frac{q^b-1}{2}\Big)\cdot\sum_{i=1}^{d-1}\big(n+1-(i-1)b\big)\\ =&\Big(\frac{q^b-1}{2}\Big)\cdot(d-1)\cdot\Big(n+1-\frac{d-2}{2}b\Big)\end{aligned}$$

$\mathbb{Z}_q$ elements. □

After the sample reduction, we are left with m samples $(\mathbf{a}_j, t_j)$ from oracle $B_{\mathbf{s},\chi,d-1}$. The vector $\mathbf{a}_j$ has only $r = n-(d-1)b$ nonzero components. We assume that $\mathbf{s}'$ is the corresponding block of the secret such that $t_j = \langle \mathbf{a}_j, \mathbf{s}'\rangle + \nu_j$ with ν_j the sum of 2^{d-1} original error terms. The next step in the algorithm is to try and recover $\mathbf{s}' \in \mathbb{Z}_q^r$.

Hypothesis Testing In order to evaluate a candidate solution $\mathbf{v} \in \mathbb{Z}_q^r$, Albrecht et al. [ACF+15] use a log-likelihood ratio to test which distribution the errors $t_j - \langle \mathbf{a}_j, \mathbf{v}\rangle$ follow. For $\mathbf{v} = \mathbf{s}'$, these errors are expected to be the sum of 2^{d-1} independent samples from χ, multiplied by ± 1. Duc et al. [DTV15] propose an alternative approach that is not only faster, but also easier to analyse. They define the following function:

$$f(\mathbf{x}) \stackrel{\text{def}}{=} \sum_{j=1}^{m} 1_{\mathbf{a}_j=\mathbf{x}} \cdot e^{2\pi i t_j/q} \quad ,\forall \mathbf{x} \in \mathbb{Z}_q^r \tag{3.24}$$

where $1_{\pi(\mathbf{x})}$ is 1 when $\pi(\mathbf{x})$ is true and 0 otherwise. The discrete Fourier transform of f is

$$\begin{aligned}\hat{f}(\mathbf{v}) &= \sum_{\mathbf{x}\in\mathbb{Z}_q^r} f(\mathbf{x})\cdot e^{-2\pi i\langle \mathbf{x},\mathbf{v}\rangle/q}\\ &= \sum_{\mathbf{x}\in\mathbb{Z}_q^r}\sum_{j=1}^{m} 1_{\mathbf{a}_j=\mathbf{x}}\cdot e^{2\pi i(t_j-\langle \mathbf{x},\mathbf{v}\rangle)/q}\\ &= \sum_{j=1}^{m} e^{-2\pi i(\langle \mathbf{a}_j,\mathbf{v}\rangle - t_j)/q}\end{aligned} \tag{3.25}$$

This Fourier transform is all that that is needed to obtain a quick and easy hypothesis test as the following Lemma shows:

Lemma 11 (Lemma 15 from [DTV15])**.** $\operatorname{argmax}_{\boldsymbol{v}} \operatorname{Re}\big(\hat{f}(\boldsymbol{v})\big) = \boldsymbol{s}'$ *with probability greater than*

$$1 - q^r \cdot \exp\big(-\frac{m}{8}\cdot \mathbb{E}[\cos(2\pi\chi/q)]^{2^d}\big)$$

Before proving this result, we need to mention some other Lemmas and Theorems.

Lemma 12 (Lemma 11 from [DTV15]). *For q an odd integer, let $X \sim \chi$ and $Y \sim 2\pi X/q$. Then*

$$\mathbb{E}[\cos(Y)] \geq R_{\sigma,q} = 1 - 2(\frac{\pi\sigma}{q})^2 \text{ and } \mathbb{E}[\sin(Y)] = 0.$$

Lemma 13 (Lemma 13 from [DTV15]). $\mathbb{E}\big[\operatorname{Re}(\hat{f}(\boldsymbol{s}'))\big] \geq m \cdot \left(R_{\sigma,q}\right)^{2^{d-1}}$.

Proof [DTV15]. From (3.25), we get

$$\hat{f}(\mathbf{s}') = \sum_{j=1}^{m} e^{-2\pi i(\langle \mathbf{a}_j, \mathbf{s}' \rangle - t_j)/q} = \sum_{j=1}^{m} e^{-2\pi i(\nu_{j,1} \pm \nu_{j,2} \pm \ldots \pm \nu_{j,2^{d-1}})/q}.$$

with $\nu_{j,l}$ independent samples from χ.

$$\begin{aligned}\mathbb{E}\big[\operatorname{Re}(\hat{f}(\mathbf{s}'))\big] &= \operatorname{Re}\Big(\sum_{j=1}^{m} \mathbb{E}[e^{-2\pi i(\nu_{j,1} \pm \nu_{j,2} \pm \ldots \pm \nu_{j,2^{d-1}})/q}]\Big) \\ &= \operatorname{Re}\Big(\sum_{j=1}^{m} \mathbb{E}\big[\cos\big(\frac{2\pi}{q}\nu_{j,1}\big)\big]^{2^{d-1}}\Big)\end{aligned}$$

because the noise samples $\nu_{j,l}$ are independent. From Lemma 12, we know that $\mathbb{E}[\cos(2\pi\nu_{j,l}/q)] \geq R_{\sigma,q} = 1 - 2(\pi\sigma/q)^2$, so it follows that

$$\mathbb{E}\big[\operatorname{Re}(\hat{f}(\mathbf{s}'))\big] \geq \sum_{j=1}^{m} (R_{\sigma,q})^{2^{d-1}} = m \cdot (R_{\sigma,q})^{2^{d-1}}.$$

□

Lemma 14 (Lemma 14 from [DTV15]). *Let $G \subset \mathbb{Z}_q$, let $X \xleftarrow{U} G$ and let $e \in \mathbb{Z}_q$ be independent from X. Then, $\mathbb{E}\big[\exp(\frac{2\pi i}{q}(X+e))\big] = 0$.*

Theorem 4 (Hoeffding's Inequality [DTV15]). *Let $X_1, X_2, \ldots, X_m$ be m independent random variables such that $\Pr[X_j \in [\alpha_j, \beta_j]] = 1$ for $1 \leq j \leq m$. We define $X = X_1 + X_2 + \ldots + X_m$. We have that*

$$\Pr[X - \mathbb{E}[X] \geq T] \leq \exp\Big(\frac{-2T^2}{\sum_{j=1}^{m}(\beta_j - \alpha_j)^2}\Big)$$

and

$$\Pr[X - \mathbb{E}[X] \leq -T] \leq \exp\Big(\frac{-2T^2}{\sum_{j=1}^{m}(\beta_j - \alpha_j)^2}\Big)$$

for any $T > 0$.

We are now ready to demonstrate the proof of Lemma 11.

Proof of Lemma 11 [DTV15]. Specifically, we show that

$$\Pr[\mathrm{Re}(\hat{f}(\mathbf{v})) \geq \mathrm{Re}(\hat{f}(\mathbf{s}'))] \leq \exp\Big(-\frac{m}{8}(R_{\sigma,q})^{2^d}\Big) \text{ for some fixed } \mathbf{v} \neq \mathbf{s}' \tag{3.26}$$

The result in Lemma 11 follows from the fact that we can upper bound the probability that there exists some $\mathbf{v}$ such that $\mathrm{Re}(\hat{f}(\mathbf{v})) \geq \mathrm{Re}(\hat{f}(\mathbf{s}'))$ by q^r times (3.26), using a union bound.

Let $u_j = \mathrm{Re}\left(e^{-2\pi i(\langle \mathbf{a}_j, \mathbf{s}'\rangle) - t_j)/q}\right)$ and $w_j = \mathrm{Re}\left(e^{-2\pi i(\langle \mathbf{a}_j, \mathbf{v}\rangle - t_j)/q}\right)$. Then, using Equation (3.25) one gets:

$$\Pr[\mathrm{Re}(\hat{f}(\mathbf{v})) \geq \mathrm{Re}(\hat{f}(\mathbf{s}'))] = \Pr\Big[\sum_{j=1}^{m}\Big(u_j - w_j\Big) \leq 0\Big]$$

Define the errors $e_j = \langle \mathbf{a}_j, \mathbf{s}'\rangle - t_j$ for $1 \leq j \leq m$ and $\mathbf{z} = \mathbf{v} - \mathbf{s}'$ such that

$$\langle \mathbf{a}_j, \mathbf{v}\rangle - t_j = \langle \mathbf{a}_j, \mathbf{z}\rangle + e_j \tag{3.27}$$

By definition, $\mathbf{a}_j$ is uniformly distributed in $\mathbb{Z}_q^r$ and independent from e_j. Since $\mathbf{z}$ is fixed and nonzero, the right hand side of (3.27) is uniformly distributed in a subset of $\mathbb{Z}_q$ and we can thus apply Lemma 14. As a result, $\mathbb{E}[e^{-2\pi i(\langle \mathbf{a}_j, \mathbf{v}\rangle - t_j)/q}] = 0 \Leftrightarrow \mathbb{E}[w_j] = 0$.

Furthermore, using Lemma 13, one finds that $\mathbb{E}\Big[\mathrm{Re}(\hat{f}(\mathbf{s}'))\Big] = \mathbb{E}\Big[\sum_{j=1}^{m} u_j\Big] \geq m \cdot (R_{\sigma,q})^{2^{d-1}}$.

Finally, to apply Hoeffding's Inequality, let $X = \sum_{j=1}^{m} X_j$ with $X_j = u_j - w_j$ and note that $\forall j,\ X_j \in [-2, 2]$. The previous results give us that $\mathbb{E}[X] \geq m \cdot (R_{\sigma,q})^{2^{d-1}}$. With $T = \mathbb{E}[X] > 0$, Theorem 4 leads us to the sought probability:

$$\begin{aligned}\Pr[X \leq 0] &= \Pr[X - \mathbb{E}[X] \leq -\mathbb{E}[X]] \\ &\leq \exp\Big(\frac{-2\mathbb{E}[X]^2}{\sum_{j=1}^{m} 16}\Big) \\ &\leq \exp\Big(-\frac{m}{8} \cdot (R_{\sigma,q})^{2^d}\Big)\end{aligned}$$

□

In conclusion, all that needs to be done in stage two of the BKW algorithm, is computing the fast Fourier Transform $\hat{f}(\mathbf{v})$ of (3.24) and searching the argument $\mathbf{v} \in \mathbb{Z}_q^r$ that maximizes it. We found that calculating $\hat{f}(\mathbf{v})$ directly is less efficient than computing $f(\mathbf{x})$ and performing the Fast Fourier Transform, because $f(\mathbf{x})$ is only nonzero for $\mathbf{x} = \mathbf{a}_j$.

The question that remains is the number of samples from $B_{\mathbf{s},\chi,d-1}$ needed to ensure a large success probability. This is easily derived from the result in Lemma (11).

Theorem 5 (Theorem 16 from [DTV15]). *Let n, q be positive integers and $A_{\boldsymbol{s},\chi}$ an LWE oracle, where $\boldsymbol{s} \in \mathbb{Z}_q^n$. Let $d \in \mathbb{Z}$ with $1 \leq d \leq n$, let b be such that $(d-1)b < n \leq db$, and let $r = n - (d-1)b$. Let $B_{\boldsymbol{s},\chi,d-1}$ be the oracle returning*

samples $(\boldsymbol{a}_j, t_j)$ *where the* $\boldsymbol{a}_j$ *are zero for all but the first* r *elements. Denote the vector consisting of the first* r *elements of* $\boldsymbol{s}$ *as* $\boldsymbol{s}'$*. Fix an* $\epsilon \in (0,1)$*. Then, the number of independent samples* m *from* $B_{\boldsymbol{s},\chi,d-1}$*, which are required such that we fail to recover the secret block* $\boldsymbol{s}'$ *with probability at most* ϵ *satisfies*

$$m \geq 8 \cdot r \cdot \ln(\frac{q}{\epsilon}) \cdot \left(1 - 2(\frac{\pi\sigma}{q})^2\right)^{-2^d}$$

Furthermore, the hypothesis testing phase that recovers $\boldsymbol{s}'$ *requires* $2m + C_{FFT} \cdot r \cdot q^r \cdot \log q$ *operations in* $\mathbb{C}$ *and requires storage for* q^r *complex numbers, where* C_{FFT} *is the small constant in the complexity of the FFT.*

Proof [DTV15]. For a fixed m, we need

$$\Pr\left[\exists \mathbf{v} \neq \mathbf{s}' : \mathrm{Re}(\hat{f}(\mathbf{v})) \geq \mathrm{Re}(\hat{f}(\mathbf{s}'))\right] \leq q^r \cdot \exp\Big(-\frac{m}{8} \cdot (R_{\sigma,q})^{2^d}\Big) \leq \epsilon$$

When we solve this for m with $R_{\sigma,q} = 1 - 2(\frac{\pi\sigma}{q})^2$, we find the intended result.

The memory complexity comes from storing $f(\mathbf{x})$ as q^r elements from $\mathbb{C}$. An in-place FFT requires no additional storage.

To compute $f(\mathbf{x})$, we need an exponentiation and addition in $\mathbb{C}$ for each sample ($=2m$ operations). Finally, the discrete Fourier transform involves $C_{\mathrm{FFT}} \cdot r \cdot q^r \cdot \log q$ complex operations. □

Back Substitution The final stage uses the recovered candidate for $\mathbf{s}'$ to reduce the tables T^i, similarly to solving a triangular system. Table T^{d-1} can be dropped after this phase, since it would become completely zero. The cost of back substitution is $2b$ operations per row. Afterwards, the algorithm starts over with sample reduction for the next block of $\mathbf{s}$.

We complete this section with a summary of the complexity of the BKW algorithm. For simplicity, we only consider the case where $db = n$ such that $r = b$.

Theorem 6 (Complexity of the BKW algorithm from [DTV15])**.** *Let* n, q *be positive integers and* $A_{\boldsymbol{s},\chi}$ *be an LWE oracle, where* $\boldsymbol{s} \in \mathbb{Z}_q^n$*. Let* $d, b \in \mathbb{N}$ *be such that* $db = n$*. Let* C_{FFT} *be the small constant in the complexity of the fast Fourier transform computation. Let* $0 < \epsilon < 1$ *be a targeted success rate and define* $\epsilon' = (1-\epsilon)/d$*. For* $0 \leq j \leq d-1$*, let*

$$m_{j,\epsilon} \overset{def}{=} 8 \cdot b \cdot \ln(\frac{q}{\epsilon}) \cdot \left(1 - 2(\frac{\pi\sigma}{q})^2\right)^{-2^{d-j}} \tag{3.28}$$

Under the standard heuristic that all the samples after reduction are independent, the time complexity of BKW to recover the secret $\boldsymbol{s}$ *with probability at least* ϵ *is* $c_1 + c_2 + c_3 + c_4$*, where*

$$c_1 = \Big(\frac{q^b - 1}{2}\Big) \cdot \Big(\frac{(d-1)\cdot(d-2)}{2}(n+1) - \frac{b}{6}(d \cdot (d-1) \cdot (d-2))\Big) \tag{3.29}$$

is the number of additions in $\mathbb{Z}_q$ *to produce all tables* T^i, $0 \leq i \leq d-1$,

$$c_2 = \sum_{j=0}^{d-1} m_{j,\epsilon'} \cdot \frac{d-1-j}{2} \cdot (n+2) \tag{3.30}$$

is the number of additions in $\mathbb{Z}_q$ *to produce the samples required to recover all blocks of* $\boldsymbol{s}$ *with probability* ϵ,

$$c_3 = 2\Big(\sum_{j=0}^{d-1} m_{j,\epsilon'}\Big) + C_{FFT} \cdot n \cdot q^b \cdot \log q \tag{3.31}$$

is the number of operations in $\mathbb{C}$ *to prepare and compute the DFT's, and*

$$c_4 = (d-1) \cdot (d-2) \cdot b \cdot \frac{q^b - 1}{2} \tag{3.32}$$

is the number of operations in $\mathbb{Z}_q$ *for back substitution.*

The number of calls to $A_{\boldsymbol{s},\chi}$ *is* $(d-1) \cdot \frac{q^b-1}{2} + m_{0,\epsilon'}$.

Finally, the memory complexity in number of elements from $\mathbb{Z}_q$ *and* $\mathbb{C}$ *are respectively*

$$\Big(\frac{q^b-1}{2} \cdot (d-1) \cdot \big(n+1-b\frac{d-2}{2}\big)\Big) + m_{0,\epsilon} \text{ and } q^b$$

Proof [DTV15]. To recover all blocks of $\mathbf{s}$, the three stages of the algorithm are repeated d times. The set of tables is created only once and modified with back substitution in each iteration. During back substitution, the last table can always be dropped. By using a failure probability $\epsilon' = (1-\epsilon)/d$ for each of the iterations, the overall success probability according to Boole's inequality is at least ϵ.

Lemma 10 proves (3.29) and tells us that obtaining m samples from $B_{\mathbf{s},\chi,d-1}$ costs at most $m((d-1)(n+2)/2)$ operations. The result in (3.30) follows from the fact that each round uses one table less and thus the cost of obtaining $m_{j,\epsilon'}$ samples from $B_{\mathbf{s},\chi,j}$ is upper bounded by

$$m_{j,\epsilon'} \cdot \frac{d-1-j}{2} \cdot (n+2)$$

During hypothesis testing (Theorem 5), we have 2 operations per sample to construct $f(\mathbf{x})$, explaining the first part of (3.31). The Discrete Fourier transform is applied d times, which brings the total cost to

$$\sum_{j=0}^{d-1} C_{\text{FFT}} \cdot b \cdot q^b \cdot \log q = C_{\text{FFT}} \cdot n \cdot q^b \cdot \log q$$

operations (3.31).

Finally, back substitution is applied to $d-2$ tables the first time, each of which has $(q^b-1)/2$ rows. The cost per row is $2b$ operations and the number of tables

decrements in each iteration. The total cost is then

$$\begin{aligned}&\sum_{j=1}^{d-2} 2b \cdot (j \cdot \frac{q^b - 1}{2})\\ =&2b \cdot \frac{q^b - 1}{2} \cdot \sum_{j=1}^{d-2} j\\ =&2b \cdot \frac{q^b - 1}{2} \cdot \frac{(d-1)(d-2)}{2}\end{aligned}$$

which is the result in (3.32).

The number of calls to the LWE oracle follows directly from Lemma 10. The first $m_{0,\epsilon'}$ can be stored and reused for subsequent blocks of $\mathbf{s}$, since $m_{0,\epsilon'} > m_{j,\epsilon'}$ for $j > 0$.

The storage of elements in $\mathbb{Z}_q$ consists of these $m_{0,\epsilon'}$ samples and the cost for the tables from Lemma 10. The memory complexity in $\mathbb{C}$ elements refers to the in-place FFT computation as in Theorem 5. □

3.4 Conclusion

The LWE problem is theoretically a hard problem, but in practice, approximating solutions exist. Since the decision version and search version of the problem are proven to be equivalent, we can solve LWE by one of three strategies: solve the LWE decision problem via the SIS problem, solve the LWE search problem by the BDD method or use the BKW algorithm.

The first two approaches are dominated by lattice basis reduction. We defined their computational costs and success probability or advantage ϵ. Under the assumption of unlimited access to LWE samples, it is worthwhile to accept $\epsilon << 1$ and repeat these attacks multiple times. Multiplying the complexity of the attack with the inverse of ϵ then yields the overall security level. For the runtime of lattice basis reduction, further research is needed.

The BKW algorithm on the other hand has an efficient closed form expression for its complexity. This combinatorial algorithm treats the problem as a noisy linear system and involves no lattice reduction. It is however only usable when there is no limit on the number of available LWE samples.

From a designer's point of view, it is important to minimize the number of samples that are exposed to an adversary. It makes the BDD attack more difficult and it can prevent the BKW algorithm. Finally, we saw that the LWE problem is not easier to solve, when the elements of the secret vector $\mathbf{s}$ are chosen from the error distribution.

CHAPTER 4

LWE-Based Encryption

Using the structure of the LWE problem, it is possible to define a cryptosystem that is equally hard to break as the LWE problem is to solve. For its security, the LWE-cryptosystem relies on the assumption that the function $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q$ is hard to invert, which according to Regev is the case when the SIVP is hard to approximate, even for quantum computers [Reg09].

We present an example of an LWE-based cryptosystem in section 4.1 and its theoretical security proof in section 4.2. In the next chapter, we will see that the effective security of such a system depends on the choice of LWE parameters. However, security is not the only determinant to consider when choosing parameters. The particular design of a cryptosystem can enforce additional constraints. In section 4.3, we discuss the criteria of correct decryption.

4.1 A public-key cryptosystem

In this section, we describe an LWE-based public-key cryptosystem as defined by Lindner and Peikert [LP11], which they based on Micciancio's [Mic10] generalization of several earlier schemes. The system's parameters are a message length l, the integer modulus $q \geq 2$, two integer dimensions $n_1, n_2 \geq 1$ and two Gaussian parameters: s_k for key generation and s_e for the error distribution.

Generally, when encrypting messages from an alphabet Σ, an encoding between the message space and $\mathbb{Z}_q$ is needed. Define two operations

$$\text{encode} : \Sigma \to \mathbb{Z}_q \text{ and } \text{decode} : \mathbb{Z}_q \to \Sigma \tag{4.1}$$

such that for some threshold $t \geq 1$: $\text{decode}(\text{encode}(m) + e \bmod q) = m$ if $|e| < t$. The decoder thus tolerates small perturbations, because as in the LWE-problem, the cryptosystem will use an error term to hide information.

Key Generation $\text{Gen}(1^l)$: Generate a uniformly random public matrix $\mathbf{A} \in \mathbb{Z}_q^{n_1 \times n_2}$. With $\mathbf{R} \leftarrow D_{\mathbb{Z},s_k}^{n_1 \times l}$ and a secret key $\mathbf{S} \leftarrow D_{\mathbb{Z},s_k}^{n_2 \times l}$, let $\begin{bmatrix}\mathbf{A} & \mathbf{P}\end{bmatrix} \in \mathbb{Z}_q^{n_1 \times (n_2+l)}$ be

the public key with $\mathbf{P} = \mathbf{R} - \mathbf{A} \cdot \mathbf{S} \in \mathbb{Z}_q^{n_1 \times l}$. In matrix form we have

$$\begin{bmatrix} \mathbf{A} & \mathbf{P} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S} \\ \mathbf{I} \end{bmatrix} = \mathbf{R} \bmod q \tag{4.2}$$

Encryption $\text{Enc}(\mathbf{A}, \mathbf{P}, \mathbf{m} \in \Sigma^l)$: Choose $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \times \mathbb{Z}^l$ with $\mathbf{e}_1 \leftarrow D_{\mathbb{Z},s_e}^{n_1}$, $\mathbf{e}_2 \leftarrow D_{\mathbb{Z},s_e}^{n_2}$ and $\mathbf{e}_3 \leftarrow D_{\mathbb{Z},s_e}^{l}$. Let $\bar{\mathbf{m}} = \text{encode}(\mathbf{m}) \in \mathbb{Z}_q^l$ and compute the ciphertext

$$\mathbf{c} = \begin{bmatrix} \mathbf{A}^T & \mathbf{I} & \mathbf{O} \\ \mathbf{P}^T & \mathbf{O} & \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \\ \mathbf{e}_3 + \bar{\mathbf{m}} \end{bmatrix} = \begin{bmatrix} \mathbf{A}^T\mathbf{e}_1 + \mathbf{e}_2 \\ \mathbf{P}^T\mathbf{e}_1 + \mathbf{e}_3 + \bar{\mathbf{m}} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} \in \mathbb{Z}_q^{(n_2+l)} \tag{4.3}$$

Decryption $\text{Dec}(\mathbf{c}, \mathbf{S})$: Output $\text{decode}(\mathbf{c}_1^T \cdot \mathbf{S} + \mathbf{c}_2^T)^T$. This expression is equivalent to $\text{decode}(\mathbf{e}^T \cdot \bar{\mathbf{R}} + \bar{\mathbf{m}}^T)^T \in \Sigma^l$ with the matrix $\bar{\mathbf{R}}$ defined as $\begin{bmatrix} \mathbf{R} \\ \mathbf{S} \\ \mathbf{I} \end{bmatrix}$.

$$\begin{aligned} \begin{bmatrix} \mathbf{c}_1^T & \mathbf{c}_2^T \end{bmatrix} \cdot \begin{bmatrix} \mathbf{S} \\ \mathbf{I} \end{bmatrix} &= \mathbf{e}_1^T\mathbf{A}\mathbf{S} + \mathbf{e}_2^T\mathbf{S} + \mathbf{e}_1^T\mathbf{P} + \mathbf{e}_3^T + \bar{\mathbf{m}}^T \\ &= \mathbf{e}_1^T\mathbf{R} + \mathbf{e}_2^T\mathbf{S} + \mathbf{e}_3^T + \bar{\mathbf{m}}^T = \mathbf{e}^T \cdot \bar{\mathbf{R}} + \bar{\mathbf{m}}^T \end{aligned} \tag{4.4}$$

The output will thus be equal to $\mathbf{m}$ if $\forall j \in [1, l]$: $|\langle \mathbf{e}, \bar{\mathbf{r}}_j \rangle| < t$, with $\bar{\mathbf{r}}_j$ the columns of $\bar{\mathbf{R}}$.

It should be noted that the decryption operation in (4.4) can be found in (4.2) as well. In fact, each row of the public key $\begin{bmatrix} \mathbf{A} & \mathbf{P} \end{bmatrix}$ is therefore an encryption of zero.

To demonstrate the LWE-samples in this cryptosystem, we will consider $l = 1$ for simplicity. Firstly, in the public key, we have n_1 LWE-samples $(\mathbf{A}^T, \mathbf{P})$ hiding a secret $-\mathbf{S} \in \mathbb{Z}^{n_2}$, since $\mathbf{P} = (\mathbf{A}^T)^T \cdot (-\mathbf{S}) + \mathbf{R} \in \mathbb{Z}_q^{n_1}$ with $\mathbf{A}^T$ uniformly random $\in \mathbb{Z}_q^{n_2 \times n_1}$ and the elements from $\mathbf{R}$ drawn from $D_{\mathbb{Z},s_k}$. The LWE dimension in this case is n_2 and the noise distribution uses Gaussian width parameter s_k.

Secondly, let's define $\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{P}^* \end{bmatrix} \in \mathbb{Z}_1^{n_1 \times (n_2+1)}$ with $\mathbf{P}^*$ uniformly random $\in \mathbb{Z}_q^{n_1}$ and $\mathbf{c} \leftarrow \text{Enc}(\bar{\mathbf{A}}, \mathbf{m})$. Then $\mathbf{c} = \bar{\mathbf{A}}^T\mathbf{e}_1 + \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ \bar{\mathbf{m}} \end{bmatrix} \in \mathbb{Z}_q^{n_2+1}$, in which we recognize $n_2 + 1$ LWE-samples of a secret $\mathbf{e}_1$ with LWE dimension n_1 and error terms drawn from $D_{\mathbb{Z},s_e}$. Therefore, if we can solve the LWE Distinguishing problem, we can break the semantic security of this cryptosystem. Alternatively, to recover the secret key $\mathbf{S}$ from the public key, one must solve the LWE Search problem.

We remark that this cryptosystem uses Lemma 7 and draws its LWE secrets from the error distributions $D_{\mathbb{Z},s_k}$ and $D_{\mathbb{Z},s_e}$.

4.2 Security Proof

It can be proven that, if indeed solving the LWE Distinguishing problem is hard, the above cryptosystem can be considered CPA-secure, meaning that an adversary

cannot with non-neglible advantage distinguish a ciphertext from a uniformly random element $\in \mathbb{Z}_q^{(n_2+l)}$.

Theorem 7 ([LP11]). *The given LWE-based cryptosystem is CPA-secure if the decision-LWE problem with modulus q is hard for*
(i) dimension n_2 with error distribution $D_{\mathbb{Z},s_k}$ and for
(ii) dimension n_1 with error distribution $D_{\mathbb{Z},s_e}$.

Proof [LP11]. We prove the CPA-security by showing that an adversary's view $(\mathbf{A}, \mathbf{P}, \mathbf{c})$ in the IND-CPA game is computationally indistinguishable from uniformly random. Firstly, $\mathbf{A}^T$ is uniformly random $\in \mathbb{Z}_q^{n_2 \times n_1}$ by construction and

$$\mathbf{P} = (\mathbf{A}^T)^T \cdot (-\mathbf{S}) + \mathbf{R} \in \mathbb{Z}_q^{n_1 \times l}$$

with $\mathbf{R} \leftarrow D_{\mathbb{Z},s_k}^{n_1 \times l}$. Every column $\mathbf{p}_i$ of $\mathbf{P}$ corresponds to LWE samples

$$(\mathbf{A}^T, (\mathbf{A}^T)^T(-\mathbf{s}_i) + \mathbf{r}_i \bmod q)$$

with $\mathbf{s}_i \in \mathbb{Z}_q^{n_2}$ the columns of $\mathbf{S}$ and $\mathbf{r}_i \leftarrow D_{\mathbb{Z},s_k}^{n_1}$ the columns of $\mathbf{R}$. Therefore, if the LWE Distinguishing problem is hard for modulus q, dimension n_2 and error distribution $D_{\mathbb{Z},s_k}$ (assumption (i)), the adversary cannot distinguish $(\mathbf{A}, \mathbf{P}, \mathbf{c})$ from $(\mathbf{A}, \mathbf{P}^*, \mathbf{c})$ with $\mathbf{P}^*$ uniformly random $\in \mathbb{Z}^{n_1 \times l}$.

Secondly, if we define $\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{P}^* \end{bmatrix} \in \mathbb{Z}^{n_1 \times (n_2+l)}$ and $\mathbf{c} \leftarrow \mathrm{Enc}(\bar{\mathbf{A}}, \mathbf{m})$, we have $n_2 + l$ LWE-samples

$$\mathbf{c} = \bar{\mathbf{A}}^T \mathbf{e}_1 + \begin{bmatrix} \mathbf{e}_2 \\ \mathbf{e}_3 \end{bmatrix} + \begin{bmatrix} 0 \\ \bar{\mathbf{m}} \end{bmatrix} \in \mathbb{Z}_q^{n_2+l}$$

of a secret $\mathbf{e}_1 \leftarrow D_{\mathbb{Z},s_e}^{n_1}$. Under the hardness assumption of LWE with modulus q, dimension n_1 and $\chi = D_{\mathbb{Z},s_e}$ (assumption (ii)), $(\bar{\mathbf{A}}, \mathbf{c})$ is indistinguishable from a uniformly random $(\bar{\mathbf{A}}, \mathbf{c}^*)$. The system is thus semantically secure. □

4.3 Parameters for correctness

Depending on how a particular LWE-based cryptosystem is designed, one will need specific conditions to ensure correct decryption. For instance, the cryptosystem introduced in section 4.1 doesn't guarantee that $\mathrm{Dec}(\mathrm{Enc}(\mathbf{A}, \mathbf{P}, \mathbf{m}), \mathbf{S}) = \mathbf{m}$ for any choice of parameters. Lindner and Peikert [LP11] have derived a condition in order for decryption to be correct. We recall from equation (4.4) that $\mathrm{decode}\left(\mathbf{c}_1^T \cdot \mathbf{S} + \mathbf{c}_2^T\right)^T = \mathbf{m}$ if and only if $\forall j \in [1, l]$: $|\langle \mathbf{e}, \bar{\mathbf{r}}_j \rangle| < t$ with $\bar{\mathbf{r}}_j$ the columns of $\bar{\mathbf{R}} = \begin{bmatrix} \mathbf{R} \\ \mathbf{S} \\ \mathbf{I} \end{bmatrix}$, $\mathbf{R} \leftarrow D_{\mathbb{Z},s_k}^{n_1 \times l}$ and $\mathbf{S} \leftarrow D_{\mathbb{Z},s_k}^{n_2 \times l}$. They provide us with the following lemma for correctness:

Lemma 15 ([LP11]). *In the cryptosystem from section 4.1, the error probability per symbol (over the choice of secret key) is bounded from above by any desired $p > 0$, as long as*

$$s_k \cdot s_e \leq \frac{\sqrt{2\pi}}{c} \cdot \frac{t}{\sqrt{(n_1 + n_2) \cdot \ln(2/p)}}$$

with $c \geq 1$ a value that depends (essentially) only on $n_1 + n_2$ and t the threshold value for decoding (4.1).

Before proceeding with proofs, it is worth mentioning two lemma's from Banaszczyk about the Discrete Gaussian distribution.

Lemma 16 ([LP11]). *Let $c \geq 1$ and $C = c \cdot \exp(\frac{1-c^2}{2}) < 1$. Then for any real $s > 0$ and any integer $n \geq 1$, we have*

$$\Pr\left[\|D_{\mathbb{Z}^n,s}\| \geq c \cdot \frac{1}{\sqrt{2\pi}} \cdot s\sqrt{n}\right] \leq C^n$$

Lemma 17 ([LP11]). *For any real $s > 0$ and $T > 0$, and any $\boldsymbol{x} \in \mathbb{R}^n$, we have*

$$\Pr\left[|\langle \boldsymbol{x}, D_{\mathbb{Z}^n,s}\rangle| \geq T \cdot s\|\boldsymbol{x}\|\right] < 2\exp(-\pi \cdot T^2)$$

Proof of Lemma 15 [LP11]. For correct decryption, we need $|\langle \mathbf{e}, \bar{\mathbf{r}}_j\rangle| < t$ with , $\bar{\mathbf{r}}_j \in \mathbb{Z}^{n_1+n_2+l}$ and $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \times \mathbb{Z}^l$. The entries from $\bar{\mathbf{r}}_j$ are drawn from $D_{\mathbb{Z},s_k}$ and the entries from $\mathbf{e}$ are drawn from $D_{\mathbb{Z},s_e}$.

First, we only consider $\bar{\mathbf{e}} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{Z}^{n_1+n_2}$. According to Lemma 16: $\Pr\left[\|\bar{\mathbf{e}}\| \geq c \cdot \frac{1}{\sqrt{2\pi}} \cdot s_e\sqrt{n_1+n_2}\right] \leq C^{n_1+n_2}$. We assume this probability very small ($C \ll 1$) and thus

$$\|\bar{\mathbf{e}}\| < c \cdot \frac{1}{\sqrt{2\pi}} \cdot s_e\sqrt{n_1+n_2} \tag{4.5}$$

for some $c \geq 1$.

With the entries of $\mathbf{R}$ and $\mathbf{S}$ drawn independently from $D_{\mathbb{Z},s_k}$, each $\langle \mathbf{e}, \bar{\mathbf{r}}_j\rangle$ is independent and distributed like $\langle \bar{\mathbf{e}}, D^{n_1+n_2}_{\mathbb{Z},s_k}\rangle$ [1]. From Lemma 17, we know the probability that this value exceeds the threshold t: $\Pr\left[|\langle \bar{\mathbf{e}}, D^{n_1+n_2}_{\mathbb{Z},s_k}\rangle| \geq t\right] < 2\exp\left(-\pi(\frac{t}{s_k\|\bar{\mathbf{e}}\|})^2\right)$. (We replace parameter T from Lemma 17 with $\frac{t}{s_k\|\bar{\mathbf{e}}\|}$). This probability concerns the error probability per symbol so we bound it by p.

$$2\exp(-\pi T^2) \leq p \Leftrightarrow T^2 \geq \frac{\ln(2/p)}{\pi} \Leftrightarrow \frac{s_k\|\bar{\mathbf{e}}\|}{t} \leq \sqrt{\frac{\pi}{\ln(2/p)}} \tag{4.6}$$

Finally, we apply the assumption of equation (4.5) and obtain

$$\frac{s_k}{t} \cdot \|\bar{\mathbf{e}}\| < \frac{s_k}{t} \cdot c \cdot \frac{1}{\sqrt{2\pi}} \cdot s_e\sqrt{n_1+n_2} \leq \sqrt{\frac{\pi}{\ln(2/p)}} \tag{4.7}$$

which is equivalent to the statement in Lemma 15. □

[1] Lindner and Peikert ignore the last term $\mathbf{e}_3$ and compensate with some slack in their choice of parameters

The value of c depends on the probability of choosing a "bad" encryption vector $\mathbf{e}$ (the probability that $\|\bar{\mathbf{e}}\| \geq c \cdot \frac{1}{\sqrt{2\pi}} \cdot s_e\sqrt{n_1+n_2}$). By setting a sufficiently low bound for $C^{n_1+n_2}$ (for example $C^{n_1+n_2} \leq 2^{-60}$), we can calculate c numerically using the equation $C = c \cdot \exp(\frac{1-c^2}{2})$. Examples are shown in Table 4.1. It is also possible for the encryption procedure to simply discard bad encryption vectors.

There is no need for the error probability per symbol p to be extremely low. In Table 4.1 for example, we have $p = 0.01$. When p is only reasonably small, the redundancy in an error-correcting code can ensure that a receiver is able to recover the entire message correctly.

TABLE 4.1. Examples of bounds for c and the resulting trade-off between s and t when the probability of choosing a bad encryption vector $C^{n_1+n_2}$ is bounded by 2^{-60} and the decryption error probability per symbol $p = 0.01$

(n_1+n_2)	$c \geq$	$\frac{s_k \cdot s_e}{t} \leq$
256	1.43	0.0845
512	1.30	0.0658
640	1.26	0.0603
1024	1.21	0.0499
2048	1.15	0.0372

Keeping in mind that the decoding threshold t is proportional to q, we find a trade-off between s and q in Lemma 15. The Gaussian width is bounded from above by q, while at the same time enforcing a minimum value for the modulus. Other cryptosystems show similar limitations and we will see that accordingly, lower values for q correspond to more security. At the same time, recall that s is bounded from below as well for two reasons. Firstly, the discrete Gaussian distribution over $\mathbb{Z}$ only approximates the continuous distribution well if s exceeds the smoothing parameter of the integer lattice (see Table 2.1). Secondly, Lindner and Peikert advise that s should be at least 8 in order to avoid the Arora-Ge attack [AG11], which attempts to recover the secret $\mathbf{s}$ as the root from a polynomial with degree $\sim s$.

In homomorphic encryption schemes, it is possible to perform calculations on encrypted data. The decryption of the outcome matches the result of these operations, when carried out on the plaintext. A cryptosystem that would allow any number of operations (*i.e.* the execution of any functionality) is known as a fully homomorphic encryption scheme. In cloud computing, fully homomorphic encryption would enable remote computations on data while completely preserving the owner's privacy. Ensuring correctness in homomorphic schemes is however more complex, since multiplicative operations on ciphertexts result in a significant noise growth. As a consequence, these cryptosystems often require much larger values for q. For instance, Table 4.2 specifies lower bounds of the LWE modulus for the correctness of fully homomorphic encryption schemes FV (Fan-Vercauteren) [FV12] and YASHE [BLLN13].

Table 4.2. Table 3 from [LN14]: Minimal value of $\log_2(q)$ to ensure correctness of YASHE and FV, with overwhelming probability, using $s = 8\sqrt{2\pi}$. L is number of homomorphic multiplications

(a) YASHE

n	1024	2048	4096	8192	16384
$L = 0$	20	21	22	23	24
$L = 1$	62	64	66	68	70
$L = 10$	265	286	306	326	346
$L = 50$	1150	1250	1350	1450	1550

(b) FV

n	1024	2048	4096	8192	16384
$L = 0$	19	20	21	22	23
$L = 1$	40	43	46	49	52
$L = 10$	229	250	271	292	313
$L = 50$	1069	1170	1271	1372	1473

4.4 Conclusion

Creating new encryption schemes and choosing LWE parameters for post-quantum cryptography is a challenging task. Apart from ensuring that key sizes are not too large and that encryption and decryption operations with (n, q, s) are feasible while attacks against LWE are not, the cryptographer must also deal with the correctness issues of his design. The Gaussian width is bounded from below, both because of the Arore-Ge attack [AG11] and because the discrete distribution must approximate the continuous one sufficiently well. Meanwhile, to achieve correct decryption, the LWE modulus is restricting s from above, wile also lower bounded itself. In the next chapter, we will see that unfortunately, a "high" s and "low" q are preferable for security.

CHAPTER 5

LWE Parameters

So far, we have assumed that the LWE problem is hard, yet we were able to describe attacks against it in section 3.3. We noted that the complexity and success probability of such an attack and thus the concrete hardness of LWE depends on the choice of LWE parameters. The LWE problem is characterized by a dimension $n \geq 1$, an integer modulus $q \geq 2$ and a Gaussian parameter s for the error distribution $\chi = D_{\mathbb{Z},s}$. The following chapter seeks to answer two questions:

- Given a set of parameters (n, q, s), how can we evaluate the concrete security?
- How should we choose these parameters in order to obtain a particular security level?

In section 5.1, we first investigate the cost of performing lattice reduction with BKZ 2.0, as no consensus can be found in literature at the moment. After proposing a new way to estimate this runtime, we use it to predict the complexities of the attacks that were introduced in section 3.3. The results lead to a security level according to each method, as a function of the parameters Security(n, q, s). We describe how we implemented the calculations and we investigate the sensitivity of Security(n, q, s) to each parameter. Finally, section 5.3 demonstrates how we use all this to compute the modulus q that one would need to achieve a desired degree of security.

5.1 The runtime of BKZ

The execution time of BKZ reduction is an important element in our search for secure parameters. Currently, there is no consistent upper bound for its complexity and in practice, the quality of reduced bases often exceeds theoretical bounds. Moreover, the high lattice dimensions in a cryptographic context make experimenting very difficult. According to Chen and Nguyen [CN12], BKZ with a blocksize $\beta \geq 40$ is too expensive. Also, a root-Hermite factor $\delta = 1.005$ appears to be just feasible in practice, while $\delta = 1.001$ is not realistic. Nonetheless, in order to get concrete security

estimates for LWE parameters, we need an approximation of the BKZ runtime for dimensions that go beyond the scope of experiments.

Gama and Nguyen [GN08] showed that the basis reduction runtime needed to achieve a certain root-Hermite factor δ for random lattices in large dimensions mainly depends on δ alone. Lindner and Peikert [LP11] have confirmed this behaviour for random q-ary lattices and derived a lower bound estimate using least-squares regression. With a fixed set of LWE parameters and the optimal subdimension $m = \sqrt{n \log_2 q / \log_2 \delta}$, they found the best linear fit $t_{BKZ}(\delta) = \log_2(T_{BKZ}(\delta)) = \frac{1.806}{\log_2 \delta} - 91$. Thinking of future advances in algorithms and hardware, they also proposed a more conservative lower bound $t_{BKZ}(\delta) = \frac{1.8}{\log_2 \delta} - 110$. This approximation is based on the BKZ implementation of the 5.5.2 version of Shoup's NTL Library [Sho][1]. Moreover, Lindner and Peikert ran the BKZ algorithm with successively increasing blocksizes β. Schneider and Buchmann showed in [SB10] that on average, reduction finishes faster when BKZ starts immediately with a high blocksize β. For these reasons, we expect that this approach is no longer suitable. A more recent runtime estimate was introduced by Albrecht et al. [ACF+15], based on some BKZ 2.0 data points from Liu and Nguyen [LN13].

Table 5.1. Data points from Table 2 in [LN13], with T_{BKZ} in seconds

δ	1.006	1.007	1.008	1.009	1.01	1.012
$\log_2(T_{BKZ})$	95.3	61.8	42	28	18.4	8.2

Abandoning Lindner and Peikert's assumption that $\log_2(T_{BKZ}) = \mathcal{O}(1/\log_2(\delta))$, Albrecht et al. interpolated the data points from Table 5.1 and obtained the function $t_{BKZ}(\delta) = \frac{0.009}{\log_2^2(\delta)} - 27$. The data in [LN13] was however based on simulation results from Chen-Nguyen [CN11], which they updated in a later work [CN12].

Table 5.2 shows results from Lepoint and Naehrig [LN14], based on both data from Chen-Nguyen [CN12] and results from the BKZ2.0 simulation algorithm. The table shows clearly that the lattice dimension m does have an influence. Sadly, there is not enough data for each dimension to extrapolate a runtime approximation.

Table 5.2. Data from [LN14]: minimal root-Hermite factor δ achievable with a given number of enumeration nodes

$\log_2$(# enumeration nodes)	1000	5000	10000	15000	20000
64	1.00851	1.00896	1.00918	1.00931	1.00940
80	1.00763	1.00799	1.00811	1.00826	1.00833
128	1.00592	1.00609	1.00619	1.00624	1.00628

All these measures are shown in Figure 5.1. For a fair comparison, we convert them all to the number of clock cycles needed for reduction. For Lindner-Peikert and Albrecht, we take into account the fact that they considered computers running at

[1] That is, this BKZ implementation does not yet have the optimizations that Chen and Nguyen used to create BKZ 2.0. The newest library currently available is NTL 9.1.0

2.3GHz. The data in Table 5.2 gives the number of enumeration nodes in BKZ 2.0 reduction. Chen and Nguyen mention in [CN12] that one enumeration node requires about 200 clock cycles.

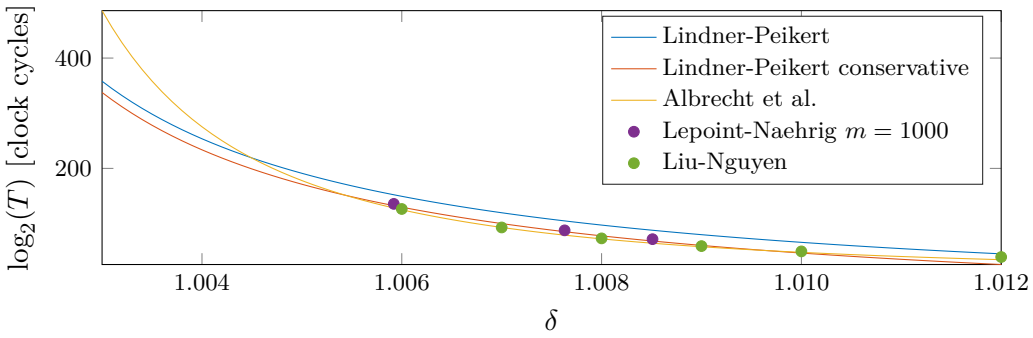


FIGURE 5.1. Comparing BKZ runtime estimates

It is clear from Figure 5.1 that we have insufficient data for small root-Hermite factors to determine the best runtime estimate.

We apply the ideas used in previous works and search for our own extrapolation, starting from Chen and Nguyen's most recent data [CN12]. Table 5.3 shows the blocksizes β needed to obtain a certain root-Hermite factor δ.

TABLE 5.3. Table 3 from [CN12]: Approximate required blocksize for high-dimensional BKZ, as predicted by the simulation algorithm

Target root-Hermite factor δ	1.005	1.006	1.007	1.008	1.009	1.01
Approximate blocksize β	286	216	168	133	106	85

In reality, the root-Hermite factor also depends on the lattice dimension m, but in her thesis [Che13], Chen provided evidence for the following limit:

$$\lim_{m\to\infty} \delta(\beta, m) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}} \tag{5.1}$$

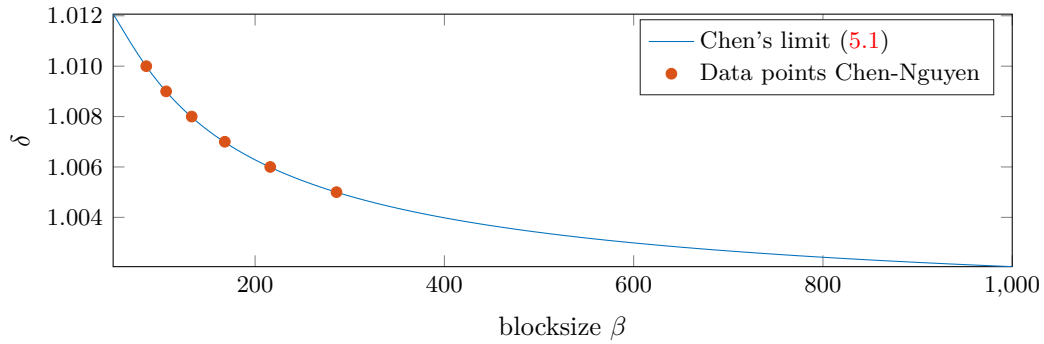


FIGURE 5.2. Illustration of Chen's $\lim_{m\to\infty} \delta(\beta, m)$ with data points from Table 5.3 [CN12]

No expression for $\delta(\beta, m)$ itself can be found in Chen's thesis, but we will use her results to prove that for BKZ 2.0 reduction with block size β:

$$\delta(\beta, m) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{1/\beta}\right)^{(1+\beta^2/(6m^2))\frac{m-1}{2m(\beta-1)}} \tag{5.2}$$

Proof of (5.2) (partly from [Che13]). Let $\{\mathbf{b}_1, \ldots, \mathbf{b}_m\}$ be the output basis of the BKZ 2.0 algorithm, and $\{\tilde{\mathbf{b}}_1, \ldots, \tilde{\mathbf{b}}_m\}$ its Gram-Schmidt orthogonalized basis. Let $l_i = \ln(\|\tilde{\mathbf{b}}_i\|)$ and $\Lambda(\mathbf{B}_{[i,j]})$ denote a sublattice of Λ with basis vectors $\{\mathbf{b}_i, \ldots, \mathbf{b}_j\}$. For convenience of notation, we also let $L_{[i,j]} = \ln(\mathrm{vol}(\Lambda(\mathbf{B}_{[i,j]}))$.

Define the *Local Hermite factor* h_i for $1 \leq i \leq m$ as the ln of the Hermite factor for each block $\mathbf{B}_{[i,j]}$:

$$h_i = l_i - \frac{L_{[i,\min(i+\beta-1,m)]}}{\min(\beta, m-i+1)} \tag{5.3}$$

This expression can be derived from the definition of the Hermite factor (2.13): $\|\tilde{\mathbf{b}}_i\| = \exp(h_i) \cdot \mathrm{vol}(\Lambda(\mathrm{Block}_i))^{1/\dim(\mathrm{Block}_i)}$ where Block_i is either $\mathbf{B}_{[i,i+\beta-1]}$ or $\mathbf{B}_{[i,m]}$ and $\dim(\mathrm{Block}_i)$ is respectively β or $m-i+1$.

Next, consider the *Global Hermite factor* g_i for $1 \leq i < m$, which is the ln of the root-Hermite factor for each block $\mathbf{B}_{[i,m]}$:

$$g_i = \frac{\beta-1}{m-i}\left(l_i - \frac{L_{[i,m]}}{m-i+1}\right) \tag{5.4}$$

This concept is defined such that the actual root-Hermite factor is as follows:

$$\delta(\beta, m) = \left(\|\mathbf{b}_1\| / \mathrm{vol}(\Lambda)^{1/m}\right)^{1/m} = \exp(g_1)^{\frac{m-1}{m(\beta-1)}} \tag{5.5}$$

The next Lemma demonstrates that g_1 can be expressed as a linear combination of h_i's.

Lemma 18 (Lemma 4.1.1. from [Che13])**.** *We calculate g_i from h_i by induction with the following relationship:*

$$g_i = \begin{cases} \frac{\beta-1}{m-i}h_i & m-\beta+1 \leq i < m \\ \frac{\beta}{m-i+1}h_i + \frac{m-i-\beta+1}{m-i+1} \cdot \frac{\sum_{j=1}^{\beta-1} g_{i+j}}{\beta-1} & 1 \leq i \leq m-\beta \end{cases}$$

We may thus write $g_1 = \sum_{i=1}^{m-1} k_i h_i$ for some coefficients k_i. Now, let $K = \sum_{i=1}^{m-1} k_i$.

Lemma 19 (Lemma 4.1.2. from [Che13])**.** *We have the following property for k_i:*

1. $k_1 = \frac{\beta}{m}$
2. $\forall i,\ k_i < e \cdot \frac{\max(\beta, m-\beta)}{m(m-1)} \leq \frac{e}{m}$
3. $1 \leq K < 1 + \frac{\beta^2}{4m^2}$. *The lower bound is reached when $m = \beta$. In addition, there exists a constant $c \approx 5.96$ such that $\lim_{m\to\infty} \frac{K-1}{1/n^2} \to c$.*

Corollary 1 (Corollary 4.1.3. from [Che13]). *If for $1 < i < m-1$, h_i's are i.i.d. variables with $\mathbb{E}[h_i] = \bar{h}$, then $\mathbb{E}[g_1] = \frac{\beta}{m} \cdot h_1 + (K - \frac{\beta}{m})\bar{h}$.*

According to the Gaussian Heuristic (2.12), $\exp(h_i) \approx \mathrm{GH}(\Lambda(\mathbf{B}_{[i,i+\beta-1]})) = \left(\frac{\mathrm{vol}(\Lambda(\mathbf{B}_{[i,i+\beta-1]}))}{v_\beta}\right)^{1/\beta}$, so we approximate $\exp(\bar{h}) \approx v_\beta^{-1/\beta}$. For the unit ball volume, we apply Stirling's estimate:

$$v_\beta = \frac{1}{\sqrt{\pi\beta}}\left(\frac{2\pi e}{\beta}\right)^{\beta/2}$$

and we obtain the following expression for $\exp(\bar{h})$:

$$\exp(\bar{h}) \approx \sqrt{\frac{\beta}{2\pi e}} \cdot (\pi\beta)^{\frac{1}{2\beta}}$$

We then use Corollary 1 to find

$$\mathbb{E}[g_1] = \frac{\beta}{m} \cdot \mathbb{E}[h_1] + (K - \frac{\beta}{m})\bar{h} = K\bar{h} \approx (1 + \frac{\beta^2}{6m^2})\bar{h}$$

Finally, expression (5.5) leads us to the intended result (5.2):

$$\begin{aligned}\delta(\beta, m) &= \exp(g_1)^{\frac{m-1}{m(\beta-1)}} \\ &= \exp(\bar{h})^{(1+\frac{\beta^2}{6m^2})\frac{m-1}{m(\beta-1)}}\end{aligned}$$

□

We thus use expression (5.2) to relate a basis quality to a BKZ blocksize β. Using this blocksize, we now try to find the BKZ 2.0 runtime. Table 5.4 shows Chen-Nguyen's upper bound for the complexity of one enumeration subroutine as a function of β.

TABLE 5.4. Table 4 from [CN12]: Upper bound on the cost of the enumeration subroutine

Blocksize β	100	110	120	130	140	150	160	170
$\log_2$(# enumeration nodes)	39	44	49	54	60	66	72	78
Blocksize β	180	190	200	210	220	230	240	250
$\log_2$(# enumeration nodes)	84	96	99	105	111	120	127	134

Lepoint and Naehrig performed a least-squares regression of these points and obtained $\log_2(\#\text{ Enumeration nodes}) = 0.64\beta - 28$ for $\beta \in \{100, \ldots, 250\}$ (see Figure 5.3). We convert this figure to the number of clock cycles as before, assuming 200 clock cycles per enumeration node:

$$\log_2(\#\text{ clock cycles for enumeration subroutine}) = 0.64\beta - 20.36 \tag{5.6}$$

In [APS15], we find two alternatives for the enumeration subroutine. Their Table 1 shows estimates of the complexity of finding a short vector with the `fplll` library [Ste] or a sieving algorithm [AKS01] instead of using lattice enumeration. The comparison in Figure 5.3 shows that sieving is an important competitor when β becomes large. This method is however also more demanding for memory, so we assume for now that BKZ 2.0 uses enumeration, which is known to have only polynomial memory complexity [LMvdP14].

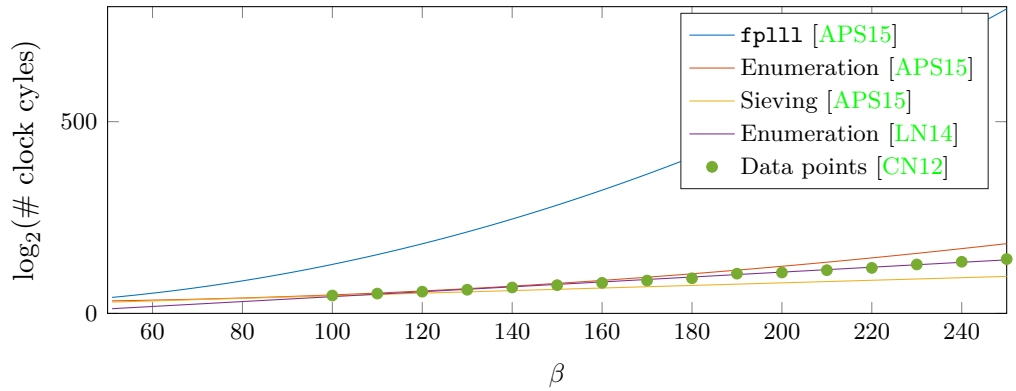


Figure 5.3. Complexity of 1 SVP subroutine as a function of β with data points from [CN12], the least squares regression of [LN14] and estimates from [APS15]

One round of BKZ 2.0 reduction performs $\mathcal{O}(m)$ enumeration subroutines, with m the dimension of the lattice. Hanrot et al. show in [HPS11] that a polynomial number of calls to the enumeration subroutine is sufficient to obtain a basis quality close to that of a full reduction:

$$\text{Number of calls} = \Omega\Big(\frac{m^3}{\beta^2}\big(\log_2(m) + \log_2(\log_2(\max_i \|\mathbf{b}_i\|))\big)\Big)$$

We use the simulation algorithm from [CN12] to confirm that the root-Hermite factor indeed decreases mainly in the early rounds of BKZ 2.0 (see Figure 5.4).

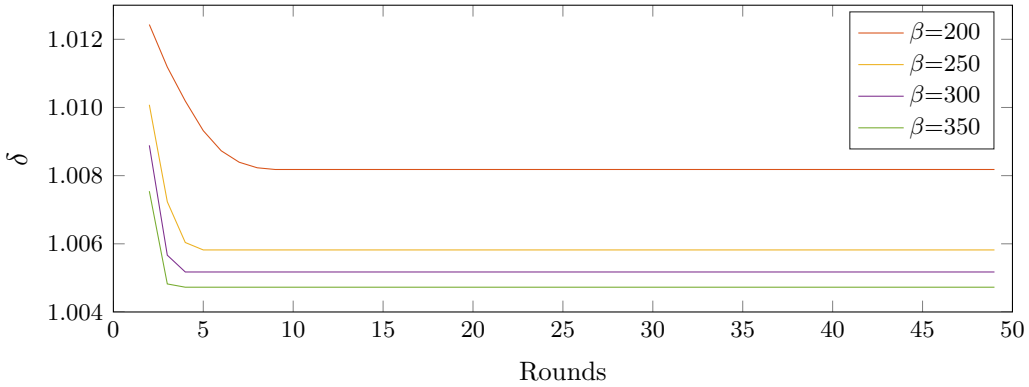


Figure 5.4. Simulation results, showing the evolution of the root-Hermite factor with the number of BKZ 2.0 rounds

We therefore assume a fixed number of rounds $R = \frac{m^2}{\beta^2}\log_2(m)$ and we conclude that when performing BKZ 2.0 reduction on an m-dimensional basis using blocksize β, one achieves a root-Hermite factor $\delta(\beta, m)$, requiring around $C_{BKZ}(\beta, m)$ clock cycles.

$$\delta(\beta, m) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{1/\beta}\right)^{(1+\beta^2/(6m^2))\frac{m-1}{2m(\beta-1)}} \tag{5.7}$$

$$\log_2(C_{BKZ}(\beta, m)) = \log_2\left(\frac{m^3}{\beta^2}\log_2(m)\right) + 0.64\beta - 20.36 \tag{5.8}$$

Figure 5.5 compares this result for $m = 1000$ with those of Lindner-Peikert [LP11] and Albrecht [ACF+15]. Lindner and Peikert's estimation is often deemed too conservative. Albrecht's function on the other hand, increases very rapidly for decreasing δ. Our estimate thus offers a compromise: it is more cautious than that of Albrecht, but not as conservative as Lindner-Peikert's. An additional advantage of this new measure, is that it varies with the lattice dimension m.

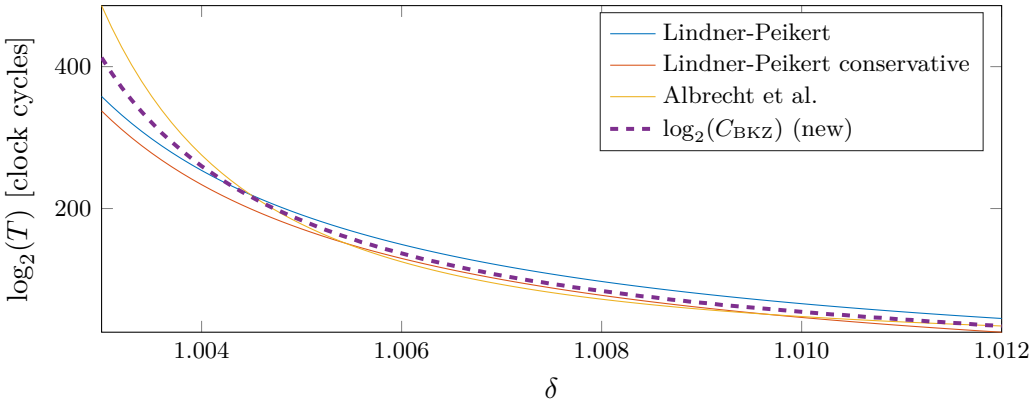


FIGURE 5.5. Comparing BKZ runtime estimates

For the implementation of $C_{\mathrm{BKZ}}(\delta, m)$, we have to take into account the minimum root-Hermite factor that is feasible in an m-dimensional lattice. We approximate this minimum value with $\delta_{\min} = \delta(m, m)$. When the input δ to C_{BKZ} is smaller than $\delta_{\min}$, we assume the BKZ runtime to be infinite.

5.1.1 Sensitivity analysis

In an attempt to evaluate the trustworthiness of our runtime estimate, we investigate its sensitivity with respect to the variables δ, β and m. Figure 5.5 shows that $\log_2(C_{\mathrm{BKZ}})$ is most sensitive around low values of δ, corresponding to the region where experimental data are scarce due to lack of feasibility. The large deviations between estimates of different authors confirm the uncertainty that surrounds this region. We inspect the effect of a variation $\Delta\delta = 0.0001$ when $m = 1000$:

$$|\log_2(C_{\mathrm{BKZ}})(1.01 + \Delta\delta, m) - \log_2(C_{\mathrm{BKZ}})(1.01, m)| = 1.2113$$
$$|\log_2(C_{\mathrm{BKZ}})(1.003 + \Delta\delta, m) - \log_2(C_{\mathrm{BKZ}})(1.003, m)| = 21.6060$$

Indeed, the resulting change of $\log_2(C_{\text{BKZ}})$ in the neighbourhood of $\delta = 1.01$ is relatively small compared to that around $\delta = 1.003$. At this point, it is difficult to determine how much of this deviation is the result of actual variations in complexity.

In [SB10], Schneider and Buchmann state that the runtime of BKZ is polynomial in the lattice dimension m and exponential in the blocksize parameter β. Figure 5.6 shows the evolution of (5.8) with each of these parameters when one is fixed and confirms the statements of Schneider and Buchmann. The very small increase with m also confirms Gama and Nguyen's claim that basis reduction runtime mainly depends on δ alone.

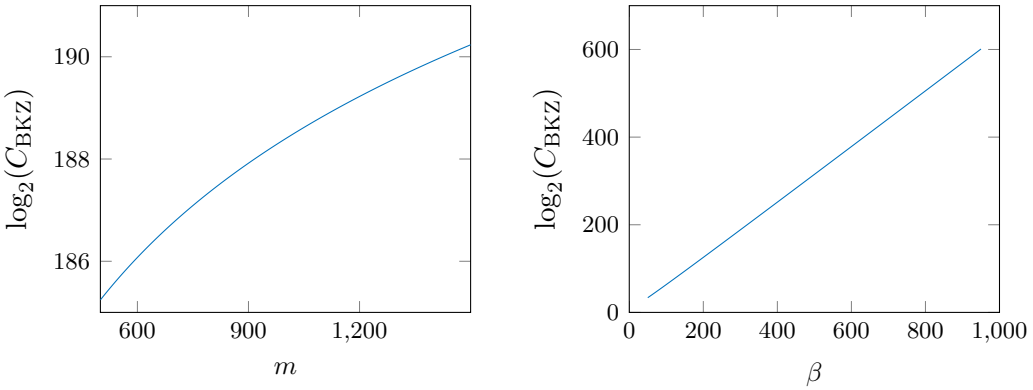


FIGURE 5.6. $\log_2(C_{\text{BKZ}}(m, \beta))$ in the neighbourhood of $(m, \beta) = (1000, 300)$

5.2 Estimating Security

We now seek to measure the security of an LWE-based cryptosystem starting from fixed parameters n, q and s. We investigate the number of clock cycles C needed to mount an attack with a particular advantage ϵ. It should be kept in mind that the total complexity of an attack is then in the order of $C \times \epsilon^{-1}$ clock cycles, under the condition that repeated executions are independent. We also assume that the number of clock cycles is equivalent to the number of bit-operations.

5.2.1 The distinguishing attack (SIS)

We first examine the *distinguishing* attack, for which the advantage is close to $\exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big)$ with $\mathbf{v}$ a short vector $\in \Lambda_q^{\perp}(\mathbf{A})$. Assuming an adversary can mount this attack with advantage ϵ, the length of vector $\mathbf{v}$ is bounded as follows: [LP11]

$$\exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big) \geq \epsilon \Leftrightarrow -\pi(\frac{\|\mathbf{v}\|s}{q})^2 \geq \ln(\epsilon) \Leftrightarrow \|\mathbf{v}\| \leq \frac{q}{s}\sqrt{-\frac{\ln(\epsilon)}{\pi}} \tag{5.9}$$

This result is the maximum allowed length of the vector $\mathbf{v}$ to ensure that an adversary obtains the desired advantage ϵ. We use equation (3.14) to figure out the basis quality

that is required to obtain a vector with such length.

$$\|\mathbf{v}\| = 2^{2\sqrt{n \log_2 q \log_2 \delta}} \Leftrightarrow \log_2 \|\mathbf{v}\| = 2\sqrt{n \log_2 q \log_2 \delta}$$
$$\Leftrightarrow \log_2 \delta = \frac{(\log_2 \|\mathbf{v}\|)^2}{4n \log_2 q} \Leftrightarrow \delta = 2^{(\log_2 \|\mathbf{v}\|)^2/(4n \log_2 q)} \tag{5.10}$$

In this way, given a set of parameters n, q, s, one can determine the root-Hermite factor δ that an adversary needs to mount an attack with advantage ϵ. Note that equation (5.10) only applies when we may use the optimal subdimension. This is the case if the number of available samples is at least $\sqrt{n \log_2 q / \log_2 \delta}$. In other cases, m is equal to the total number of available samples and δ must be calculated from $\|\mathbf{v}\| = \delta^m q^{n/m}$ (3.12). Once the adversary's minimal root-Hermite factor is found, we use (5.7) and (5.8) to estimate the running time of the distinguishing attack. To obtain the blocksize, we solve the scalar equation $\delta(\beta, m) = \delta$ for β with m the optimal subdimension. We then use this β to compute $\log_2(C_{BKZ})$.

Figure 5.7 shows results for the set of parameters $(n, q, s) = (320, 4093, 8)$. For each advantage ϵ, the root-Hermite factor δ needed by an adversary and the BKZ reduction time to achieve it were determined. The total complexity of the attack is measured by the number of security bits $\text{Security}_{\text{SIS}}(n, q, s) = \min\{\log_2(C_{BKZ}(\delta, m) \times \epsilon^{-1})\} = \min\{c_{BKZ}(\delta, m) - \log_2(\epsilon)\}$. This particular set of parameters leads to a security level of 221 bits according to the distinguishing attack, which means that it resists all adversaries who can perform up to 2^{221} operations.

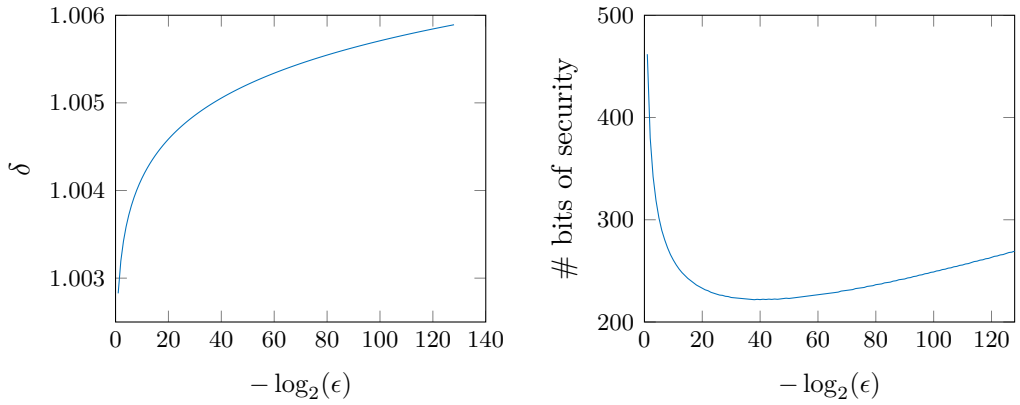


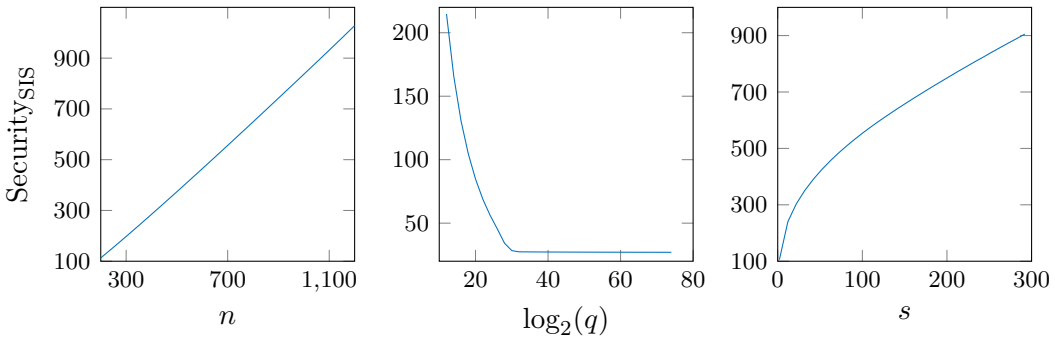
FIGURE 5.7. Results of the security analysis for the distinguishing attack with $n = 320, q = 4093, s = 8$. Left: The root-Hermite factor δ needed to obtain some advantage. Right: The number of bits of security in function of the used advantage

Sensitivity analysis

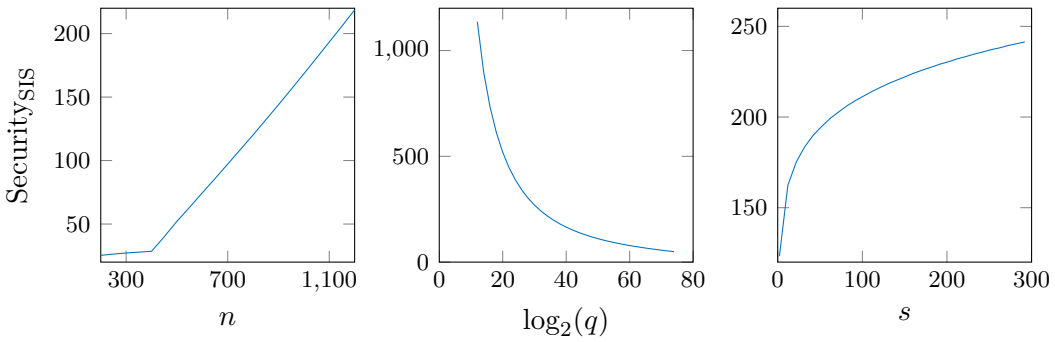
We now have a procedure that, given a parameter set (n, q, s), computes the number of bits security according to the SIS attack. We use this function $\text{Security}_{\text{SIS}}(n, q, s)$ to measure the influence of each parameter on the security. Figure 5.8 shows the evolution of $\text{Security}_{\text{SIS}}$ as a function of one parameter, when the other two are

fixed. As in most cryptographic systems, the dimension n has a linear impact on the security. Parameters q and s are best made respectively as small and large as possible. This is usually prohibited by the design of the cryptosystem itself. For example, Lemma 15 shows a trade-off to be made between q and s if we want the system of chapter 4 to decrypt correctly.

When comparing the parameter sets in Figure 5.8, we also remark that the influence of s decreases when the other parameters are larger. On the other hand, increasing n also seems to increase the sensitivity of $\text{Security}_{\text{SIS}}$ to q.



(a) $(n, q, s) = (320, 4093, 8.0)$



(b) $(n, q, s) = (1024, 4093478590233419, 20.05)$

FIGURE 5.8. Sensitivity analysis of $\text{Security}_{\text{SIS}}$ for a fixed set (n, q, s)

5.2.2 The Bounded Distance Decoding attack

For the *decoding* attack, an adversary's success probability is given by equation (3.20).

$$\Pr[\mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}} \cdot \text{diag}(\mathbf{d}))] = \prod_{i=1}^{m} \text{erf}\left(d_i \frac{\|\tilde{\mathbf{b}}_i\|\sqrt{\pi}}{2s}\right) \tag{3.20}$$

with $\|\tilde{\mathbf{b}}_i\| = \|\mathbf{b}_1\|\alpha^{i-1}$ (2.15) and $d_i = \lceil \frac{R}{\|\tilde{\mathbf{b}}_i\|} \rceil$. Recall that the optimal subdimension

$$m = \sqrt{n \log_2 q / \log_2 \delta}$$

minimizes the length of the first basis vector of $\Lambda_q^\perp(\mathbf{A})$. By duality, the same dimension maximizes the length of the first basis vector of $\Lambda_q(\mathbf{A})$. Suppose $\mathbf{c}_1$ is the shortest basis vector of $\Lambda_q^\perp(\mathbf{A})$, obtained by BKZ reduction with the optimal subdimension. From (3.12) we know that

$$\|\mathbf{c}_1\| = \delta^m q^{n/m} \tag{5.11}$$

By duality, we have $\|\tilde{\mathbf{b}}_m\| = \frac{q}{\|\mathbf{c}_1\|}$, with $\tilde{\mathbf{b}}_m$ the last Gram Schmidt vector of the basis of $\Lambda_q(\mathbf{A})$.

$$\|\tilde{\mathbf{b}}_m\| = \frac{q}{\|\mathbf{c}_1\|} = \frac{q}{\delta^m q^{n/m}} = \delta^{-m} q^{1-n/m} \tag{5.12}$$

Furthermore, by the GSA (2.15), we know that $\|\tilde{\mathbf{b}}_m\| = \|\mathbf{b}_1\|\alpha^{m-1}$ with $\alpha^{m-1} = \delta^{-2m}$. The length of $\mathbf{b}_1$ is thus as follows:

$$\|\mathbf{b}_1\| = \frac{\|\tilde{\mathbf{b}}_m\|}{\alpha^{m-1}} = \delta^{2m}\|\tilde{\mathbf{b}}_m\| = \delta^m q^{1-n/m} \tag{5.13}$$

Note that this corresponds to $\|\mathbf{b}_1\| = \delta^m \det(\Lambda_q(\mathbf{A}))^{1/m}$ (2.13) with $\det(\Lambda_q(\mathbf{A})) = q^{m-n}$ (2.9).

To determine the security of LWE based on Bounded Distance Decoding, we must again determine the runtime of the attack for each possible success probability ϵ and minimize the resulting overall complexity. We assume first that the goal success probability ϵ is known. The attack consists of two parts (lattice basis reduction and the NearestPlanes algorithm) and it is most efficient when the computational cost of these components are balanced. We must thus find a root-Hermite factor δ for which the BKZ 2.0 runtime needed to achieve it is approximately equal to the NearestPlanes complexity that results from it.

$$c_{\text{BKZ}} \approx c_{\text{NP}} \tag{5.14}$$

For the latter complexity, we use Lemma 9. Lindner-Peikert's algorithm executes approximately $\prod_{i=1}^m d_i$ Babai's NearestPlane nodes. In [LP11], they estimate that around 2^{16} executions can be performed per second on a 2.3 GHz computer. We therefore assume $\frac{\#\text{ clock cycles}}{\text{node}} \approx \frac{2.3\cdot 10^9}{2^{16}}$. We then obtain the following result for the runtime in number of clock cycles:

$$c_{\text{NP}} = \log_2(C_{\text{NP}}) = \log_2(\frac{\#\text{ clock cycles}}{\text{node}}) + \log_2(\prod_{i=1}^m d_i) = 15 + \log_2(\prod_{i=1}^m d_i) \tag{5.15}$$

For a given root-Hermite factor δ, we compute the optimal subdimension m and a value R such that the Nearest Planes algorithm with $d_i = \lceil R/\|\tilde{\mathbf{b}}_i\| \rceil$ reaches a particular success probability ϵ. Then, with $\mathbf{d}$, we can calculate c_{NP} and using δ and m, we compute c_{BKZ}. We repeat this routine until we obtain $c_{\text{NP}} \approx c_{\text{BKZ}}$.

Figure 5.9 shows the results for the same set of parameters as before. If we assume that the adversary has access to an unlimited number of samples, he can perform the attack multiple times. We then have a security level $\text{Security}_{\text{BDD}} = \min\{\log_2((C_{\text{BKZ}}+$

$C_{\mathrm{NP}}) \times \epsilon^{-1})\} \approx \min\{\log_2(2C_{\mathrm{BKZ}}) - \log_2(\epsilon)\} = \min\{1 + c_{\mathrm{BKZ}}(\delta, m) - \log_2(\epsilon)\} = 197$ bits according to the bounded distance decoding method.

We remark that the use of the optimal subdimension for m is again conditional on the availability of a sufficient number of samples.

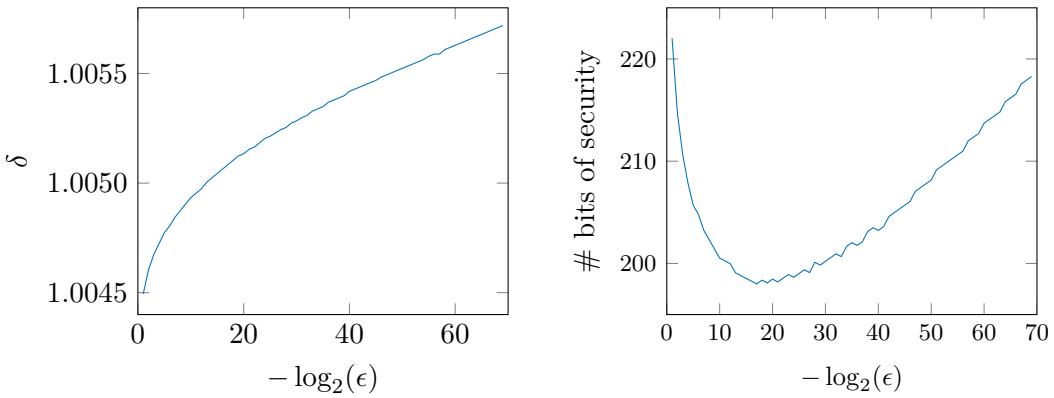


Figure 5.9. Results of the security analysis for the decoding attack with $n = 320$, $q = 4093$, $s = 8$. Left: The root-Hermite factor δ needed to obtain some advantage. Right: The number of bits of security in function of the used advantage

Like Lindner and Peikert [LP11], we see that this decoding attack is faster than the distinguishing attack. This also corresponds to the theoretical result from Lemma 6, which says that the LWE Decision problem is at least as hard as the LWE Search problem.

Implementation

The implementation of this security analysis is not straightforward. Apart from some numerical issues, a naive search for a suitable δ can be very slow. We therefore mention some important optimizations.

Firstly, obtaining a value R such that the success probability (3.20) is equal to a given ϵ is computationally demanding. Therefore, we created a heuristic algorithm to compute the vector d instead.

Algorithm 4 uses Lindner and Peikert's philosophy that the vector $\mathbf{d}$ should maximize $\min_i(d_i \cdot \|\tilde{\mathbf{b}}_i\|)$. We will show that this indeed leads to the most efficient way to enhance the success probability. Furthermore, with this algorithm, we obtain the same vector $\mathbf{d}$ as we would by solving a non-linear equation for R and letting $d_i = \lceil R/\|\tilde{\mathbf{b}}_i\| \rceil$, but we get the result in significantly less time.

ALGORITHM 4 Find the optimal vector $\mathbf{d}$

Input: Parameters n, q, s, root-Hermite factor δ and an expected success probability ϵ

Output: m-dimensional vector $\mathbf{d}$ with m the optimal subdimension

1: Compute m (optimal subdimension), $\|\mathbf{b}_1\|$ and α (GSA) as a function of δ, n and q
2: Initialize $\mathbf{d} = (1, 1, 1, ..., 1)$
3: Compute Pr[success]
4: **while** Pr[success] $< \epsilon$ **do**
5: Find $i_0 = \text{argmin}_i(d_i \|\mathbf{b}_1\| \alpha^{i-1})$
6: Increment d_{i_0}
7: Update Pr[success]
8: **end while**

To improve the success probability in (3.20), we must choose an index i_0 and increment d_{i_0}. In order to compare the merit of our alternatives, we define the potential gain of an index i as the ratio of the increase in success probability to the increase in complexity c_{NP} (5.15)), that would result from increasing d_i:

$$\text{Gain}_i = \frac{\Delta_i \Pr[\text{succes}]}{\Delta_i c_{NP}} \tag{5.16}$$

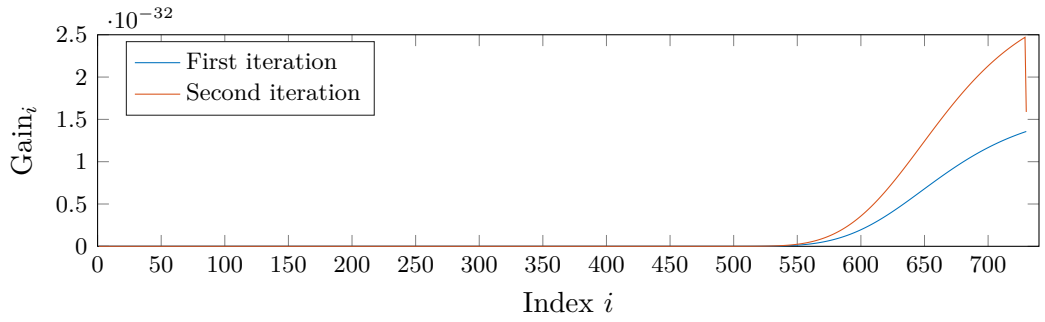
with

$$\Delta_i \Pr[\text{success}] = \Pr[\text{success}]\Big(\frac{\text{erf}((d_i+1)\|\mathbf{b}_1\|\alpha^{i-1}\sqrt{\pi}/(2s))}{\text{erf}(d_i\|\mathbf{b}_1\|\alpha^{i-1}\sqrt{\pi}/(2s))} - 1\Big)$$

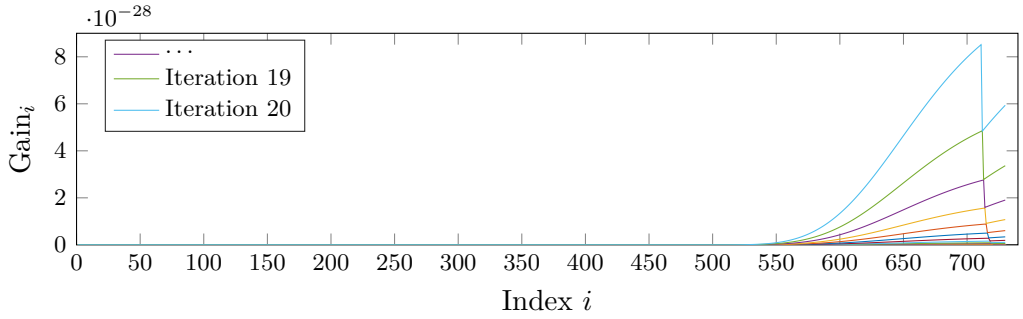
and

$$\Delta_i c_{NP} = \log_2(d_i + 1) - \log_2(d_i) = \log_2(1 + 1/d_i)$$

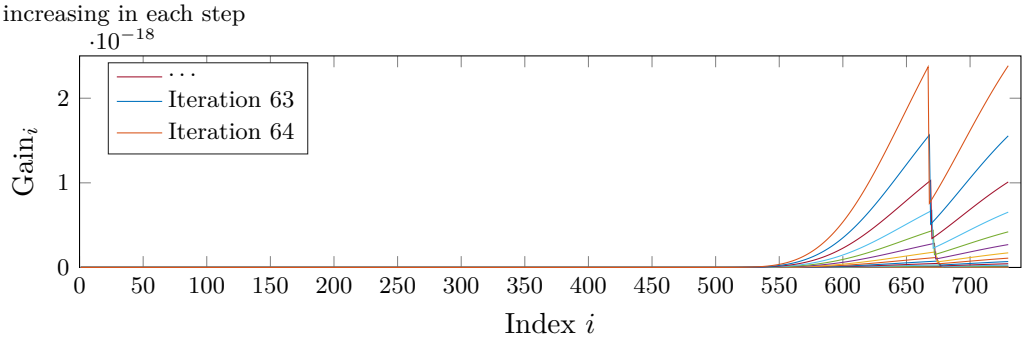
In Figure 5.10, we show this gain for all i in a few iterations of Algorithm 4.



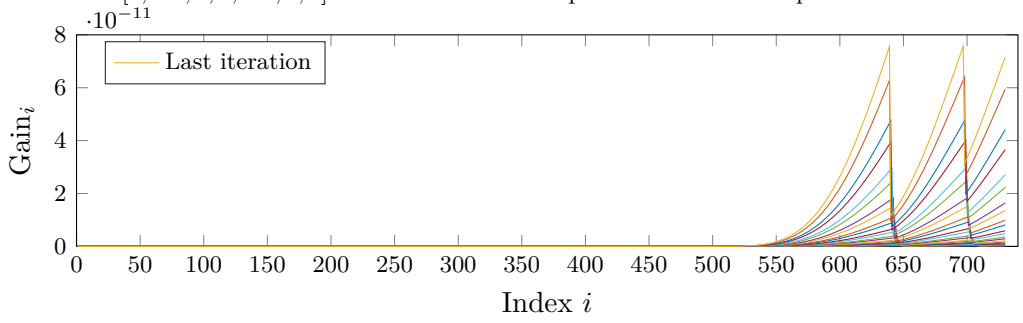
(a) In the first iteration of Algorithm 4 (blue), when $\mathbf{d} = [1, 1, \ldots, 1]$, the choice of index has no influence on the change in complexity, but the success probability for the last index increases most because the last Gram Schmidt vector is the shortest one. The last entry of $\mathbf{d}$ therefore shows the highest gain. It is incremented and $\mathbf{d}$ becomes $[1, 1, \ldots, 1, 2]$. In the following iteration (orange), the effect of having changed d_m is clearly visible as a drop in the gain. We then increment the second to last entry of $\mathbf{d}$ as its gain is then the largest. $\mathbf{d}$ becomes $[1, \ldots, 1, 2, 2]$



(b) During the first iterations, $\mathbf{d}$ has the form $[1, \ldots, 1, 2, \ldots, 2]$ with the number of 2's at the end increasing in each step



(c) At a certain moment (orange), the gain of the last entry of $\mathbf{d}$ is the largest again. At this point, $\mathbf{d}$ becomes $[1, \ldots, 1, 2, \ldots, 2, 3]$ and the behaviour of previous iterations repeats



(d) The algorithm continues in this way until the success probability reaches ϵ. $\mathbf{d}$ then has the form $[1, \ldots, 1, 2, \ldots, 2, 3, \ldots, 3]$

FIGURE 5.10. Illustration of why a vector $\mathbf{d}$ of the form $[1, \ldots, 1, 2 \ldots, 2, 3, \ldots]$ is the most efficient way to reach a certain success probability ϵ with Nearest Planes. In this example, we use $n = 320, q = 4093, s = 8, \delta = 1.005$

In an efficient implementation of Algorithm 4, the success probability is not recomputed from scratch in each iteration. Instead, we update it as follows after d_{i_0} is incremented:

$$\Pr[\text{success}] \leftarrow \Pr[\text{success}] \cdot \frac{\operatorname{erf}\left(d_{i_0}\|\mathbf{b}_1\|\alpha^{i_0-1}\sqrt{\pi}/(2s)\right)}{\operatorname{erf}\left((d_{i_0}-1)\|\mathbf{b}_1\|\alpha^{i_0-1}\sqrt{\pi}/(2s)\right)}$$

However, this optimization causes trouble when the initial success probability is too small for the computer's precision. When Pr[success] starts as zero, the update rule

will not have any effect and the `while`-loop in Algorithm 4 will never end. For this reason, we initialize $\mathbf{d}$ in way that it can't result in an initial Pr[success] smaller than 10^{-300}.

The biggest issue in the BDD security analysis is finding the most efficient way to determine δ such that $c_{\text{BKZ}} \approx c_{\text{NP}}$. Increasing δ causes an increase of c_{NP} while decreasing c_{BKZ}. We can thus increment δ with a fixed step $\Delta\delta$ until $c_{\text{NP}} > c_{\text{BKZ}}$, but finding a suitable step length is nearly impossible. When $\Delta\delta$ is too large, the resulting c_{NP} can be $\gg c_{\text{BKZ}}$. The BDD complexity is wrong when c_{NP} and c_{BKZ} are not balanced. A small $\Delta\delta$ on the other hand, ensures a correct result, but significantly slows down the process because it increases the number of executions of Algorithm 4. There is no consistent answer to this trade-off between accuracy and speed, as the ideal step length varies enormously amongst different parameter sets. With a naive implementation, it can take 20 minutes to obtain balanced complexities and thus a proper BDD runtime estimate for just one ϵ.

We use a double `while`-loop for a dynamically varying step length to get accurate results as fast as possible. We start with a large $\Delta\delta$ in order to rapidly get to a neighbourhood where c_{NP} and c_{BKZ} are of the same magnitude. When $c_{\text{NP}} > c_{\text{BKZ}}$, we go back one step by decrementing δ and we reduce the step length. This process is then repeated until c_{NP} and c_{BKZ} are close enough. Algorithm 5 demonstrates the search procedure.

Algorithm 5 Get the BDD complexity to obtain a certain success probability

Input: Parameters n, q, s, initial root-Hermite factor δ_0 and an expected success probability ϵ
Output: c_{BKZ}, c_{NP} and the root-Hermite factor δ that ensures $c_{\text{BKZ}} \approx c_{\text{NP}}$

Initialize $\Delta\delta = 10^{-4}$ and $\delta = \delta_0$
while $|c_{\text{NP}} - c_{\text{BKZ}}| > 1$ and $\Delta\delta > 10^{-6}$ **do**
  Reset $c_{\text{NP}} = 0$
  while $c_{\text{NP}} < c_{\text{BKZ}}$ and $|c_{\text{NP}} - c_{\text{BKZ}}| > 1$ **do**
    $\delta = \delta + \Delta\delta$
    $m \leftarrow$ optimal subdimension (n, q, δ)
    Find the optimal vector $\mathbf{d}$ for $(n, q, s, \delta, \epsilon)$ (Algorithm 4)
    Calculate $c_{\text{NP}}(\mathbf{d})$
    Calculate $c_{\text{BKZ}}(\delta, m)$
  end while
  $\delta = \delta - \Delta\delta$
  $\Delta\delta = \Delta\delta/2$
end while
$\delta = \delta + 2\Delta\delta$

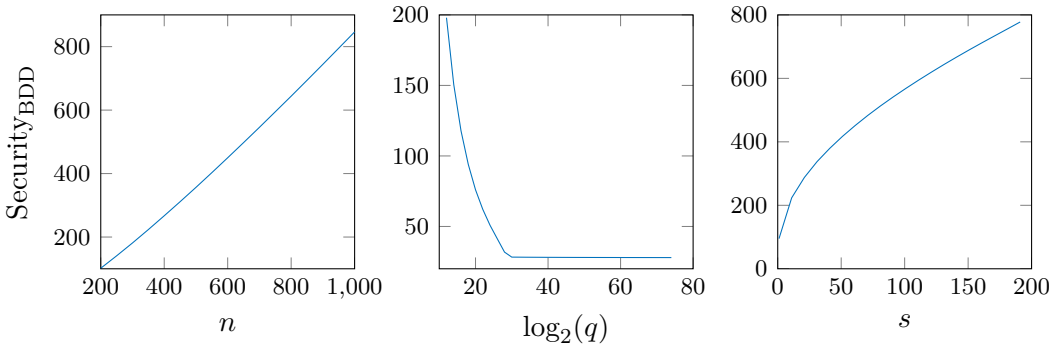
We repeat Algorithm 5 for $\epsilon = 2^{-i}$ with increasing i and choose the minimal overall complexity $(c_{\text{BKZ}} + 1 + i)$ amongst the results. A good choice for the initial root-Hermite factor δ_0 as input for Algorithm 5 can speed up the process even further.

In Figure 5.9, we saw that the root-Hermite factor only increases as ϵ decreases. We therefore use the resulting δ of each iteration as a "warm start" for the next.

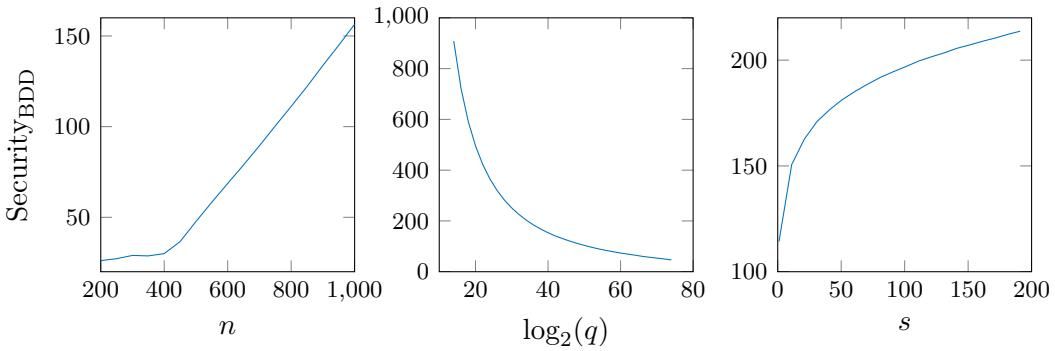
It only remains to specify a good δ_0 for the first iteration, when $\epsilon = \frac{1}{2}$. We exploit the fact that the NearestPlanes complexity (5.15) is constant as long as $\mathbf{d} = [1, 1, \ldots, 1]$. With a lattice basis of very good quality (small δ), the original NearestPlane algorithm of Babai (Algorithm 2) suffices to obtain a good success probability. While δ is very small, the vector $\mathbf{d}$ thus remains filled with ones and c_{NP} remains equal to 15. For this reason, we calculate the very first δ_0 from the success probability $\Pr[\text{Success}] = \epsilon = 0.5$ by fixing $\mathbf{d} = [1, 1, \ldots, 2]$. This is the first root-Hermite factor that results in $c_{\text{NP}} > 15$.

Sensitivity analysis

Similarly to our analysis of $\text{Security}_{\text{SIS}}(n, q, s)$, we now investigate the sensitivity of $\text{Security}_{\text{BDD}}(n, q, s)$. The results in Figure 5.11 and Figure 5.8 are very much alike and the same conclusions may be drawn from it. This resemblance is a natural consequence of the fact that both the SIS-based attack and the BDD attack are associated with lattice reduction. The BKZ 2.0 runtime is the most important ingredient in the complexities of both techniques.



(a) $(n, q, s) = (320, 4093, 8.0)$



(b) $(n, q, s) = (1024, 4093478590233419, 20.05)$

FIGURE 5.11. Sensitivity analysis of $\text{Security}_{\text{BDD}}$ around a fixed set (n, q, s)

5.2.3 The BKW attack

Thanks to Theorem 6, determining the complexity of a BKW attack for a certain parameter set is very straightforward. Given n, q and s and a desired success probability ϵ, we compute the required number of samples for each iteration and use these to compute $c_1 + c_2 + c_3 + c_4$ with equations (3.29) to (3.32). As in [DTV15], we assume $C_{\text{FFT}} = 1$. The result of Theorem 6 is the number of operations in $\mathbb{Z}_q$ to execute the BKW algorithm. We convert this estimate to the number of bit operations by multiplication with $\log_2(q)$ and we assume that this is equivalent to the number of clock cycles.

$$C_{\text{BKW}} = (c_1 + c_2 + c_3 + c_4) \cdot \log_2(q) \tag{5.17}$$

$$c_{\text{BKW}} = \log_2(C_{\text{BKW}}) = \log_2(c_1 + c_2 + c_3 + c_4) + \log_2(\log_2(q)) \tag{5.18}$$

The addition depth d is chosen by calculating c_{BKW} for all possible d and choosing the one that results in the lowest complexity for $\epsilon = 0.99$. Table 5.5 demonstrates that BKW can be more efficient than Bounded Distance Decoding for some parameter sets. However, one should always keep in mind that BKW has a large memory complexity and that it assumes unlimited access to the LWE oracle.

TABLE 5.5. Comparison of the complexity of the BKW attack and the BDD attack for two parameter sets. The storage of BKW is given in number of elements in $\mathbb{Z}_q$

n	q	s	d	$\text{Security}_{\text{BKW}}$	$\log_2(\text{Memory}_{\text{BKW}})$	$\text{Security}_{\text{BDD}}$
256	4093	8.3	22	157.7	150.2	147.8
320	4093	8.0	23	185.5	177.9	197.8

Implementation

Consider formula (3.28) for calculating the number of samples. We rewrite $m_{0,\epsilon}$ as follows:

$$m_{0,\epsilon} = 8 \cdot b \cdot \ln\left(\frac{q}{\epsilon}\right) \cdot f(x)$$

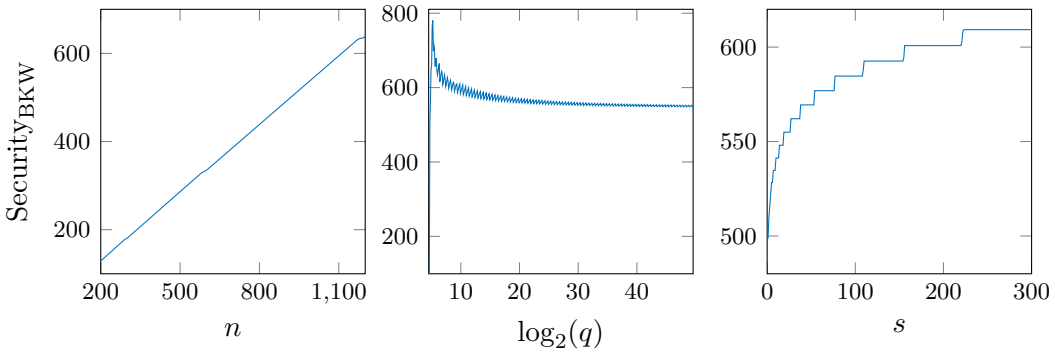
with $f(x) = (1-x)^{-2^d}$ and $x = 2(\pi\sigma/q)^2 = \pi(s/q)^2$. We don't implement this formula for $m_{0,\epsilon}$ exactly because when q is large, x can become so small that $f(x) = 1$ when it actually should be very large. For example, when $d = 60$ and $x = 10^{-16}$, $f(x) = \mathcal{O}(10^{55})$. However, the same d with $x = 10^{-17}$ gives us $f(x) = 1$. To avoid this precision problem, we calculate $f(x)$ in the logarithmic domain and apply the first-order Taylor approximation $\ln(1-x) \approx -x$ when $x \to 0$:

$$\begin{aligned} f(x) &= \exp\big(\ln f(x)\big) = \exp\big(-2^d \cdot \ln(1-x)\big) \\ &\approx \exp\big(-2^d \cdot -x\big) = \exp\big(2^d \cdot x\big) \end{aligned}$$

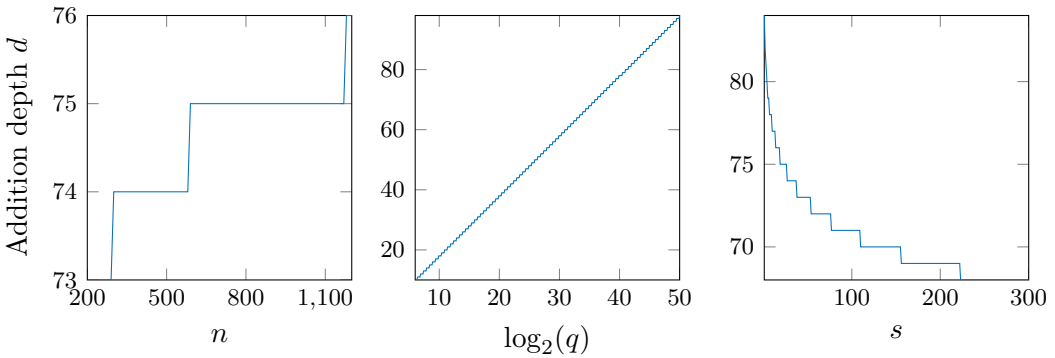
Sensitivity analysis

The evolution of $\text{Security}_{\text{BKW}}$ is quite different from that of the previous two estimates. In Figure 5.12(a), we see many discontinuities in the complexity of BKW. Using

Theorem 6, we calculated the complexity for various values of the addition depth parameter d and chose the smallest one as $\text{Security}_{\text{BKW}}$. Figure 5.12(b) shows the values of d that lead to the smallest BKW complexity and clarifies the blocked behaviour in $\text{Security}_{\text{BKW}}$.



(a) $\text{Security}_{\text{BKW}}$



(b) Addition depth d

FIGURE 5.12. Sensitivity analysis of $\text{Security}_{\text{BKW}}(n, q, s)$ and the corresponding addition depth d around a fixed set $(n, q, s) = (1024, 2^{51.86}, 20.05)$

If we look passed the blocked behaviour, we notice again a linear influence of n and polynomial for s. For large values of q, Albrecht's claim that BKW complexity is determined almost completely by n and s is confirmed. However, $\text{Security}_{\text{BKW}}$ increases significantly when q becomes small. This is due entirely to the number of samples needed for the attack (Equation (3.28)).

5.3 Estimating Parameters

In this section we attempt to select parameters for an LWE-based cryptosystem such that it resists adversaries who can perform 2^{sec} operations. We first assume the LWE dimension n and Gaussian width parameter s are known. In order to find our integer modulus q, we must pick a desired security parameter sec. Van de

Pol-Smart [vdPS13] and Lepoint-Naehrig [LN14] already described how to estimate the modulus q for the distinguishing attack. We make some corrections to this approach and extend it by including the BDD and BKW methods.

For an adversary to mount an attack in 2^{sec} clock cycles, there is a parameter λ such that the adversary spends 2^{λ} clock cycles on an attack with advantage ϵ. This leads to the relation $2^{\text{sec}} = 2^{\lambda} \cdot \epsilon^{-1} \Leftrightarrow \text{sec} = \lambda - \log_2(\epsilon)$. Lepoint and Naehrig made the mistake of using $\lambda = \text{sec}$ and $\epsilon = 2^{-\text{sec}}$ to estimate parameters with sec bits of security. We know that a complexity $2^{\lambda} = 2^{\text{sec}}$ and a success probability $\epsilon = 2^{-\text{sec}}$ actually result in overall cost of $2^{2\text{sec}}$. We will however see that estimating q in this way doesn't result in sec nor 2sec bits of security.

Let us assume for now that λ is known and that $\epsilon = 2^{\lambda - \text{sec}}$. Later, we will extend the analyses from the literature by looking for the best choice of λ such that the resulting parameters effectively result in sec bits of security.

First, we must find out the best lattice basis quality the adversary can obtain with his 2^{λ} clock cycles. We therefore consider the cost of BKZ-2.0 reduction on an m-dimensional lattice with blocksize β (5.8). Using the formula for $c_{\text{BKZ}}(\beta, m)$, we compute for each value of m the adversary's advised blocksize β by demanding $C_{BKZ} \leq 2^{\lambda} \Leftrightarrow c_{BKZ} \leq \lambda$. We then use Chen's limit (5.1) to obtain the adversary's basis quality δ.

In case of the *distinguishing* attack, the adversary's advantage $\exp\big(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\big)$ is bounded by ϵ with $\|\mathbf{v}\| = \delta^m \cdot q^{n/m}$ (3.12). This leads to the following [LN14]:

$$\exp\big(-\pi(\frac{\delta^m q^{n/m} s}{q})^2\big) \leq \epsilon \Leftrightarrow \delta^m q^{(n/m)-1} s \geq \sqrt{-\frac{\ln(\epsilon)}{\pi}} \tag{5.19}$$

Define the right hand side $\alpha = \sqrt{-\frac{\ln(\epsilon)}{\pi}}$ and one obtains the following upper limit for q:

$$m \log_2(\delta) + (\frac{n-m}{m}) \log_2(q) \geq \log_2(\frac{\alpha}{s}) \Leftrightarrow \log_2(q) \leq \frac{m^2 \log_2(\delta) + m \log_2(s/\alpha)}{m-n} \tag{5.20}$$

Since the value for m is not yet known, the upper limit for q must be computed for each $m > n$ up to a reasonable value. Lepoint and Naehrig for example obtain the smallest possible δ and the upper limit for q for each m between 1000 and 65000. The m with the smallest upper bound for q is chosen [LN14]:

$$\log_2(q) \leq \min_{m>n} \frac{m^2 \log_2(\delta) + m \log_2(s/\alpha)}{m-n} \tag{5.21}$$

According to van de Pol and Smart [vdPS13], the resulting value for m is close to the optimal subdimension (3.13) for the resulting q and δ. Indeed, if we choose for example sec$= 128$, $\lambda = 64$, $n = 320$ and $s = 8$, we find the minimum bound $\log_2(q) \leq 18.85$ for $m = 681$ and $\delta = 1.0091$. The optimal subdimension for the same δ and q is 679.

This is where the previous methods stop, but it still remains to determine the best value of λ such that the resulting security level of these parameters is

effectively equal to sec bits of security. This is not very straightforward as Figure 5.13 shows. If we compute $\text{Security}_{\text{SIS}}$ for the parameters from the example above ($n = 320, s = 8, q = 2^{18.85}$), we obtain only 96 bits of security instead of 128. Indeed, the adversary doesn't necessarily use the same value for λ in his attack. He can choose different trade-offs for λ and ϵ which may result in complexity less than 2^{128}. In order to determine a set of parameters that truly results in our intended security level, various values for λ should be put to trial by performing the parameter estimation for each one and checking the resulting number of security bits afterwards. As in section 5.2, the security is analysed by looking for the advantage ϵ that results in the smallest overall complexity. In the case of $n = 320, s = 8$ with sec$= 128$, we obtain $\text{Security}_{\text{SIS}} = 128$ bits of security when $\log_2(q) \leq 16.29$, $m = 717$ and $\delta = 1.0068$, estimated for $\lambda = 110$. This value for $\log_2(q)$ is the largest for which a successful distinguishing attack requires a complexity of at least 2^{sec}.

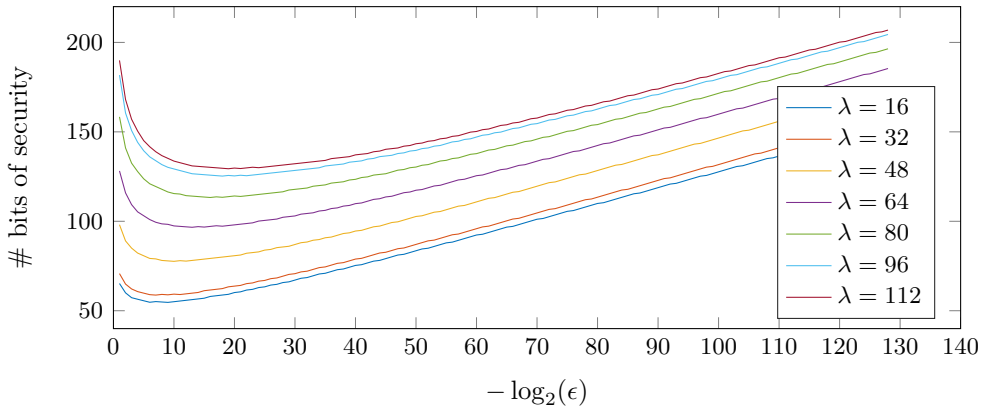
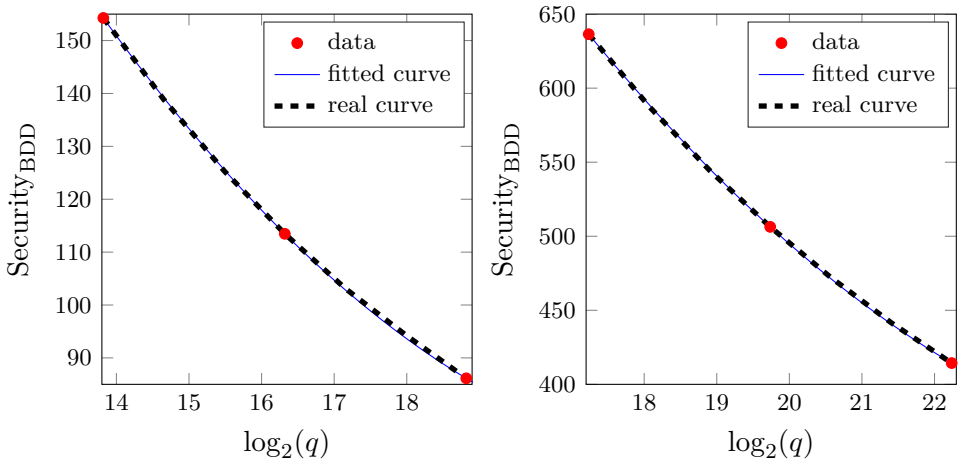


Figure 5.13. Results of security analyses for $n = 320$ and $s = 8$: the number of bits security in function of an adversary's used advantage ϵ, when q is estimated for sec$= 128$ with various λ

The previous procedure estimates the modulus q that prevents the distinguishing attack. However, as the SIS-based attack is the least menacing of the three, 16.18 is still merely an upper bound and doesn't actually result in 128 bits of security. The LWE modulus must be chosen such that both $\text{Security}_{\text{BDD}}$ and $\text{Security}_{\text{BKW}}$ are at least 128. Still, we can use (5.21) to calculate an initial upper bound UB_{SIS} for our search. The dependency of $\text{Security}_{\text{BDD}}$ and $\text{Security}_{\text{BKW}}$ on q is too complex to derive a formula such as (5.21). Instead, we choose to decrease $\log_2(q)$ from UB_{SIS} until the expected security level is achieved. There are two possible scenario's. Either $\text{Security}_{\text{BDD}} < \text{Security}_{\text{BKW}}$ or the other way around. In both cases, q must be decreased until $\min(\text{Security}_{\text{BDD}}, \text{Security}_{\text{BKW}}) = \text{sec}$.

To estimate q such that $\text{Security}_{\text{BDD}}$ is equal to sec bits of security, we must calculate $\text{Security}_{\text{BDD}}(n, q, s)$ for many candidate moduli q. As this evaluation is quite expensive, we want to avoid a linear search. The evolution of $\text{Security}_{\text{BDD}}$ with $\log_2(q)$ in Figure 5.11 is thankfully very smooth and we can easily interpolate it with a quadratic polynomial. For this, we only need to compute the BDD complexity

in three points. We choose the upper bound $\mathrm{UB}_{\mathrm{SIS}}$ as the largest node and two equidistant points $< \mathrm{UB}_{\mathrm{SIS}}$. The coefficients of the interpolated polynomial are then used to calculate the $\log_2(q)$ for which $\mathrm{Security}_{\mathrm{BDD}} = \mathrm{sec}$. Figure 5.14 demonstrates how well the interpolated polynomial corresponds to the real curve.



(a) $n = 320$, $s = 8.0$ (b) $n = 1024$, $s = 8\sqrt{2\pi}$

FIGURE 5.14. Comparison of the interpolated polynomial with the exact $\mathrm{Security}_{\mathrm{BDD}}$ curve as a function of $\log_2(q)$

It is also possible that q should be estimated based on BKW complexity. This is especially the case for small $\log_2(q)$ values (see Figure 5.15), which are required when a very high security level is sought. This also means that we might have to decrease $\log_2(q)$ further after the estimation through interpolation for $\mathrm{Security}_{\mathrm{BDD}}$ because $\mathrm{Security}_{\mathrm{BKW}}$ has become the smallest.

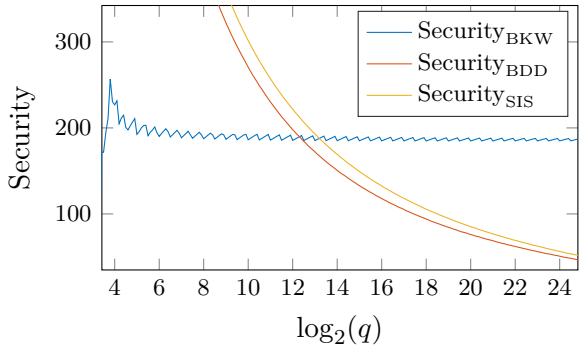


FIGURE 5.15. Evolution of all security levels for small $\log_2(q)$

The evolution of $\mathrm{Security}_{\mathrm{BKW}}$ as shown in Figure 5.15 is not at all smooth as a function of $\log_2(q)$. The interpolation approach is of no use here. Instead, we decrease $\log_2(q)$ multiplicatively until $\mathrm{Security}_{\mathrm{BKW}} >$ the required sec. If the result is on a decreasing slope (further decreasing $\log_2(q)$, decreases $\mathrm{Security}_{\mathrm{BKW}}$), we check

if a smaller $\log_2(q)$ results in a BKW complexity closer to sec. We illustrate this procedure with an example below, both for the BDD search and the BKW search.

Taking again $n = 320$, $s = 8$ and sec $= 128$, we estimate the initial upper bound $\mathrm{UB}_{\mathrm{SIS}}$ for $\lambda = 0.99\mathrm{sec}$. The security at the result $\log_2(q) = 18.82$ is 86.14 bits according to the BDD attack and 187.72 according to the BKW algorithm. We interpolate $\mathrm{Security}_{\mathrm{BDD}}$ in $\mathrm{UB}_{\mathrm{SIS}}$, $\mathrm{UB}_{\mathrm{SIS}} - 2.5$ and $\mathrm{UB}_{\mathrm{SIS}} - 5$ and obtain a polynomial $p(x) = p_0x^2 + p_1x + p_2$. We set

$$\log_2(q) = \frac{-p_1 - \sqrt{p_1^2 - 4p_0(p_2 - \mathrm{sec})}}{2p_0} = 15.27$$

which leads to an overall security of 128 bits.

Now, suppose $n = 512$, $s = 8\sqrt{2\pi}$ and sec $= 400$. A $\log_2(q)$ bound $\mathrm{UB}_{\mathrm{SIS}} = 16.47$ results in $\mathrm{Security}_{\mathrm{BDD}} = 282.5$ and $\mathrm{Security}_{\mathrm{BKW}} = 301.48$. In order to get at least 400 bits of security for the BDD attack, $\log_2(q)$ must drop further to 13.68 according to our interpolating polynomial. However, at this value, $\mathrm{Security}_{\mathrm{BKW}} < \mathrm{Security}_{\mathrm{BDD}}$, so the search continues. The value $\log_2(q) = 5.44$ is the first for which we find $\mathrm{Security}_{\mathrm{BKW}} > 400$. Further decreasing $\log_2(q)$ to 5.28 results in $\mathrm{Security}_{\mathrm{BKW}} = 401.63$. Table 5.6 summarizes the results.

Table 5.6. Results of the search for a suitable modulus q, given (n, s) and an expected security sec

n	s	sec	$\log_2(q) \leq$	BDD	BKW	Resulting security
320	8.0	128	18.82	86.14	187.72	86.14
			15.27	128.72	188.0	128.72
512	20.05	400	16.47	282.5	301.48	282.5
			13.68	400.12	311.28	311.28
			5.44	∞	411.99	411.99
			5.28	∞	401.63	401.63

We note that if one wants to design a cryptosystem such as that of chapter 4, a criterion for correctness (such as that of Lemma 15) should also be checked. If the condition is not fulfilled, the process of parameter estimation might have to be repeated with a lower s. Such criteria depend entirely on the design of a specific cryptosystem and are thus not incorporated in the web application.

In [LN14], Lepoint and Naehrig present maximal values of $\log_2(q)$ to ensure 80 bits of security, using formula (5.21) with $\lambda = 80$ and $\epsilon = 2^{-80}$. It is clear now, that the resulting parameter set will have more than 80 bits of security. For example, the parameter set $(n, \log_2(q), s) = (1024, 47.5, 8\sqrt{2\pi})$ has a security level of 113 bits. Apart from forgetting to find the optimal λ for estimation as we demonstrated in Figure 5.13, they only consider security based on the distinguishing attack. Below, we present a correction of Table 2 from [LN14], which in combination with Table 4.2 can lead to an appropriate choice of q for the fully homomorphic encryption schemes FV and YASHE.

TABLE 5.7. Maximal values for $\log_2(q)$ to ensure a certain security level, with $s = 8\sqrt{2\pi}$

n	1024	2048	4096	8192	16384
Maximal $\log_2(q)$ for 80 bits security	57.0	108.9	215.3	436.1	881.1
Maximal $\log_2(q)$ for 160 bits security	38.9	72.3	139.6	275.7	551.6

Finally, we end this section with a short investigation of how one best chooses the LWE dimension n. We assume that cryptographers already know their Gaussian width s and the security level sec that they want to achieve and that they want to choose the dimension n such that the key sizes in the cryptosystem are as small as possible. For various values of n, we estimate the modulus q required to achieve the desired security level sec and we approximate the size of the keys with $n \log_2(q)$ (the size of n elements in $\mathbb{Z}_q$). Figure 5.16 shows that this key size grows significantly with increasing n. Designers of LWE-based cryptosystem are thus advised to choose the smallest dimension n that can still achieve the intended security sec with a modulus q that is not too close to the Gaussian width parameter in order to ensure correct decryption.

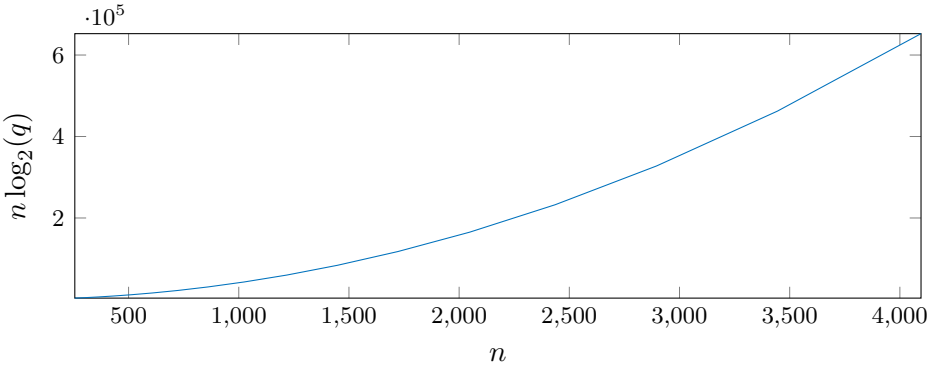


FIGURE 5.16. Illustration of the evolution of key sizes $n \log_2(q)$ with the choice of dimension n for $s = 8$ and sec=128

5.4 Conclusion

We have shown how to evaluate the security of the LWE problem for a parameter set (n, q, s) according to the three most important attacks. To do this, we proposed a new estimate of BKZ 2.0 reduction's runtime. A sensitivity analysis indicated that one should be cautious with the results. At present, it is impossible to verify them with actual experiments. The security evaluation according to the BKW algorithm follows a clear closed-form expression and doesn't have the disadvantage of uncertainty. However, this method is only usable when the number of LWE samples is unlimited. Moreover, its memory requirements are much larger than those of BDD or SIS.

The SIS-based method was analysed mainly to show that the decision LWE problem is at least as hard as the search LWE problem, as was predicted in theory.

For the actual security of learning with errors, BDD and BKW appear to be equally important as each can outperform the other for some parameters. For Lindner and Peikert's NearestPlanes algorithm, it was shown that a vector $\mathbf{d}$ of the form $[1, \ldots, 1, 2, \ldots, 2, 3, \ldots]$ obtains the best success probability for the lowest cost.

Additionally, we proposed a method to determine the LWE modulus q that satisfies some security requirement. This allowed us to find that cryptographers are advised to choose their LWE dimension n as small as possible. These functionalities were implemented in a web application, allowing designers to get security results for arbitrary parameters without the need to delve into the mathematics.

CHAPTER 6

Conclusion

The LWE Problem is a simple yet strong tool for cryptographic purposes. Two versions of the problem exist, but it can be proven that they are equivalent. The LWE problem can therefore be solved via either Decision-LWE or Search-LWE. LWE-based cryptosystems are robust to weak keys because the problem remains equally hard when the secret is chosen from a "smaller" distribution. This allows designers to reduce secret key sizes without losing security.

Three methods to attack LWE were described, of which BDD and BKW are the most important. Each of them can outperform the other for some parameter sets and each has its own benefits. The BKW algorithm has the advantage of simplicity. In addition, its security analysis isn't affected by the uncertainty that surrounds the complexity of lattice basis reduction. On the other hand, in contrast to BKW, bounded distance decoding can still be applied when the number of available LWE samples is limited, albeit with more difficulty. Moreover, the memory requirements of the BKW method far exceed those of BDD.

The complexity of attacking LWE is not the only matter to be considered when designing a public-key cryptosystem. The dimension and modulus must be such that key sizes are reasonable and operations on ciphertexts are feasible. Furthermore, the need for correct decryption enforces an upper bound on the Gaussian width parameter s and a lower bound on the LWE modulus q. As high values for s and small moduli q result in more difficult attacks, the design of the cryptosystem also bounds the maximal obtainable security. We found that, if one wants to minimize the key sizes ($\mathcal{O}(n \log_2(q))$) in a cryptographic system, it is best to choose the smallest dimension n that can achieve the desired security level with a modulus q that is not too close to s.

The connection of LWE to the shortest vector problem makes lattice basis reduction an important component of security analyses. We proposed a new runtime estimate that is straightforward to calculate as well as complicated enough to capture the effect of multiple variables. Nonetheless, a lot of confusion still surrounds this concept as the complexity of verifying these predictions for very high dimensions is

beyond our abilities. Further research by evaluation and comparison of the existing BKZ runtime estimates with the actual algorithm could alleviate doubt about this matter, but would require abundant computational resources. For now, our new estimate has the advantage over the others that it also takes into account the impact of the lattice dimension, while still having a closed-form expression.

As a result, we succeeded to construct a fast and practical tool for measuring LWE security based on the three studied attacks. The efficient formulas also allowed us to investigate how particular parameters influence the concrete security. Furthermore, the web application can calculate the upper bound for the modulus q in order to obtain a minimal security level. This tool should facilitate the design of new LWE-based cryptosystems, inspire confidence in the security of LWE and expedite the use of LWE for real-world applications.

APPENDIX A

Paper

Veiligheid van LWE cryptosystemen

Lauren De Meyer, Fre Vercauteren*, Bart Preneel*
KU Leuven ESAT/COSIC
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
*voornaam.achternaam@esat.kuleuven.be

***Samenvatting*—Learning with errors (LWE) is een wiskundig probleem dat de laatste tijd vaak gebruik wordt in publieke sleutel encryptie omdat men de moeilijkheid (hardness) ervan heeft bewezen. Theoretische bewijzen zeggen echter niets over de concrete complexiteit van LWE. We ontwerpen daarom een web applicatie, die in functie van de gebruikte parameters, de effectieve veiligheid van LWE-gebaseerde cryptosystemen kan berekenen.**

I. Inleiding

De asymmetrische cryptografie steunt al geruime tijd op wiskundige problemen zoals het ontbinden in priemfactoren en het discreet logaritme probleem. Echter, de opkomst van quantum computers zal de veiligheid van encryptie-algoritmen, die op deze problemen gebaseerd zijn, in het gedrang brengen. Rooster-gebaseerde cryptosystemen daarentegen blijken quantum aanvallen wel te kunnen weerstaan, waardoor hun populariteit de laatste jaren beduidend is toegenomen. Een van de belangrijke bouwblokken in moderne cryptosystemen is het "Learning with Errors" probleem, dat niet alleen veelbelovend is op het vlak van veiligheid, maar ook interessante toepassingsmogelijkheden heeft, zoals die van homomorfe of identiteit-gebaseerde encryptie.

De effectieve veiligheid van een cryptosysteem kan geëvalueerd worden aan de hand van de complexiteit van een succesvolle aanval. Hierover bestaat echter nog veel onzekerheid en de resultaten in bestaande literatuur zijn niet altijd samenhangend. Bovendien worden deze resultaten doorgaans slechts voor een beperkte groep parameters gegeven. Het is dus nog niet vanzelfsprekend voor ontwerpers om de veiligheid van een arbitraire set parameters te achterhalen. Om de keuze van LWE parameters te vergemakkelijken, ontwikkelen wij een web applicatie die voor elke willekeurige keuze van parameters de veiligheid berekent. Daarvoor onderzoeken we de complexiteit van bestaande aanvallen tegen LWE als functie van de parameters.

II. Achtergrond

A. Learning with errors

Het "Learning with errors" (LWE) probleem wordt gekenmerkt door drie parameters: een dimensie n, een modulus q en een Gauss parameter s. Die laatste parameter bepaalt een foutendistributie χ, waarvoor men meestal een discrete Gaussverdeling met standaarddeviatie $\sigma = s/\sqrt{2\pi}$ gebruikt.

Gegeven een geheime vector $\mathbf{s} \in \mathbb{Z}_q^n$, wordt een LWE distributie $A_{\mathbf{s},\chi}$ gemaakt, waarin elk element het geheim $\mathbf{s}$ "verstopt" in een inwendig product met ruis:

$$(\mathbf{a}, t) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \bmod q) \leftarrow A_{\mathbf{s},\chi}$$

met $\mathbf{a}$ een willekeurig element van $\mathbb{Z}_q^n$ en e een fout uit de ruisdistributie χ. Men heeft bewezen dat het recupereren van de geheime vector uit dergelijke samples een moeilijk probleem is. Er bestaan twee varianten van LWE:

Definitie 1 (LWE Beslissingsprobleem)**.** *Gegeven $(\boldsymbol{A}, \boldsymbol{t})$ met $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ en $\boldsymbol{t} \in \mathbb{Z}_q^m$, bepaal of $\boldsymbol{t}$ uniform willekeurig gekozen is uit $\mathbb{Z}_q^m$ of $\boldsymbol{t} = \boldsymbol{A}^T \boldsymbol{s} + \boldsymbol{e} \bmod q$.*

Definitie 2 (LWE Zoekprobleem)**.** *Gegeven $(\boldsymbol{A}, \boldsymbol{t})$ met $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ en $\boldsymbol{t} \in \mathbb{Z}_q^m$, vind de vector $\boldsymbol{s} \in \mathbb{Z}_q^n$ waarvoor $\boldsymbol{t} = \boldsymbol{A}^T \boldsymbol{s} + \boldsymbol{e} \bmod q$.*

Hierbij is $(\mathbf{A}, \mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q)$ met $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{t} \in \mathbb{Z}_q^m$ een verzameling van m LWE samples $\Leftrightarrow$ de kolommen van $\mathbf{A}$ zijn willekeurig $\in \mathbb{Z}_q^n$ en de elementen van $\mathbf{e}$ komen uit de ruisdistributie χ.

Toen hij het LWE probleem introduceerde, bewees Regev [Reg09] dat het even moeilijk is als bepaalde roosterproblemen, zelfs voor quantum aanvallers. Daarnaast heeft men bewezen dat het zoek- en beslissingsprobleem equivalent zijn, wat betekent dat men LWE kan oplossen, door een van beiden uit te zoeken.

Er is echter nog veel onzekerheid rond de effectieve veiligheid van LWE. Theoretische bewijzen zijn niet voldoende om vertrouwen te scheppen in de veiligheid van deze relatief jonge groep cryptosystemen. Het is ook belangrijk dat ontwerpers begrijpen hoe de keuze van LWE parameters n, q en s de veiligheid van een bepaald systeem beïnvloedt.

Om dit te onderzoeken, kijken we naar de bestaande methoden die het LWE probleem benaderend oplossen en berekenen we de complexiteit van dergelijke succesvolle aanvallen in functie van de gebruikte LWE parameters.

B. Roosters

Gegeven een matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ met $m \geq n$, maken we gebruik van twee m-dimensionele roosters:

$$\Lambda_q(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \exists \mathbf{s} \in \mathbb{Z}^n \text{ s.t. } \mathbf{z} = \mathbf{A}^T\mathbf{s} \bmod q\} \quad (1a)$$

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = 0 \pmod q\} \quad (1b)$$

Het eerste rooster (1a) wordt gevormd door gehele lineaire combinaties van de rijen van $\mathbf{A}$ $(\bmod q)$. De vectoren in het tweede rooster staan loodrecht $(\bmod q)$ op de rijen van $\mathbf{A}$.

Een basis $\mathbf{B} = (\mathbf{b}_1, \ldots, \mathbf{b}_m)$ van een m-dimensioneel rooster met volle rank is een $m \times m$ matrix van basisvectoren. Het fundamenteel parallellepipedum van een rooster wordt gevormd door deze basisvectoren rond de oorsprong:

$$\mathcal{P}_{1/2}(\mathbf{B}) = \{\sum_{i=1}^{m} x_i \mathbf{b}_i : -\frac{1}{2} \leq x_i < \frac{1}{2}\} \tag{2}$$

Een roosterbasis is niet uniek en sommigen zijn beter dan anderen. De kwaliteit van een basis $\mathbf{B}$ wordt aangeduid met de "root-Hermite factor" δ, waarvoor geldt

$$\|\mathbf{b}_1\| = \delta^m \operatorname{vol}(\Lambda)^{1/m}. \tag{3}$$

Een kleine root-Hermite factor is equivalent met een korte basisvector $\mathbf{b}_1$ en duidt dus op een roosterbasis van goede kwaliteit. Basis reductie algoritmen proberen de kwaliteit van een rooster te verbeteren door de basisvectoren zo kort en orthogonaal mogelijk (*i.e.* zo dicht mogelijk bij Gram-Schmidt vectoren) te maken. Het populairste basis reductie algoritme vandaag is het BKZ algoritme van Schnorr en Euchner [SE94]. Hierbij wordt een rooster in kleinere blokken van dimensie $\beta < m$ verdeeld en kiest men een korte vector uit die sub-roosters als nieuwe basisvector. Het algoritme werd onlangs geoptimaliseerd door Chen en Nguyen [CN11] en wordt nu BKZ 2.0 genoemd.

Een interessante eigenschap van BKZ-gereduceerde basen is dat ze een veronderstelling van meetkundige reeksen (Geometric Series assumption: GSA) volgen. Schnorr [SE94] toont aan dat de lengtes van de Gram-Schmidt vectoren $\tilde{\mathbf{b}}_i$ van een gereduceerde basis $\mathbf{B} = \langle \mathbf{b}_i \rangle$ als volgt afnemen:

$$\|\tilde{\mathbf{b}}_i\| = \|\mathbf{b}_1\| \cdot \alpha^{i-1} \tag{4}$$

met $\alpha = \delta^{-2m/(m-1)}$.

III. Aanvallen tegen LWE

A. Het beslissingsprobleem aanvallen (Distinguishing attack)

De eerste aanval werd beschreven door Micciancio en Regev [MR09] en is een methode om het LWE beslissingsprobleem op te lossen. Om uit te maken of een tweetal $(\mathbf{A}, \mathbf{t})$ van de LWE verdeling komt of willekeurig is, zoekt men een korte vector $\mathbf{v} \in \mathbb{Z}_q^n$ waarvoor $\mathbf{A}\mathbf{v} = 0 \pmod q \Leftrightarrow v \in \Lambda_q^{\perp}(\mathbf{A})$. Indien het inwendig product $\langle \mathbf{v}, \mathbf{t} \rangle$ "klein" is $\pmod q$ (*i.e.* $|\langle \mathbf{v}, \mathbf{t} \rangle| < \frac{q}{4}$), besluit men dat $(\mathbf{A}, \mathbf{t})$ een LWE sample is. Immers, wanneer $(\mathbf{A}, \mathbf{t}) \leftarrow A_{\mathbf{s},\chi}$, dan geldt $\mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q \Leftrightarrow \langle \mathbf{v}, \mathbf{t} \rangle = \mathbf{v}^T\mathbf{A}^T\mathbf{s} + \mathbf{v}^T\mathbf{e} \bmod q = \langle \mathbf{v}, \mathbf{e} \rangle \bmod q$. De elementen van $\mathbf{e}$ volgen de ruisverdeling χ en $\mathbf{v}$ is een korte vector dus we verwachten dat $\langle \mathbf{v}, \mathbf{e} \rangle \bmod q$ dan inderdaad klein is. Lemma 1 geeft het voordeel (advantage) van deze methode ten opzichte van willekeurig gissen. Hoe korter de vector $\mathbf{v}$ is, hoe groter het voordeel.

Lemma 1. *Gegeven LWE samples $(\boldsymbol{A}, \boldsymbol{t} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e})$ met $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ en een korte vector $\boldsymbol{v} \in \mathbb{Z}_q^n$ waarvoor $\boldsymbol{A}\boldsymbol{v} = 0 \pmod q$, kan de verdeling $\langle \boldsymbol{v}, \boldsymbol{e} \rangle \bmod q$ onderscheiden worden van een uniforme verdeling met voordeel $\epsilon = \exp\big(-\pi(\frac{\|\boldsymbol{v}\| s}{q})^2\big)$ als $\|\boldsymbol{v}\| s \leq q$.*

De rekentijd van deze methode hangt volledig af van de tijd die nodig is om een korte vector $\mathbf{v}$ te vinden. Hiervoor past men doorgaans het BKZ 2.0 algoritme toe om de kwaliteit van het rooster te verbeteren. De eerste basisvector is de kortste en wordt dan gebruikt als $\mathbf{v}$. Volgens (3) hebben we dan

$$\|\mathbf{v}\| = \delta^m \operatorname{vol}(\Lambda_q^{\perp}(\mathbf{A}))^{1/m} = \delta^m q^{n/m}. \tag{5}$$

Hoe kort deze vector is, hangt vooral af van de blokgrootte β, die ook de rekentijd van BKZ 2.0 bepaalt. De aanvaller moet een trade-off maken tussen de complexiteit van basis reductie C_{BKZ} en het voordeel van de methode ϵ (zie Lemma 1). Om een voordeel $\mathcal{O}(1)$ te verkrijgen, moet de procedure ϵ^{-1} maal herhaald worden. Men kiest de blokgrootte β dus best zodat de totale kost van de aanval $C_{\text{BKZ}} \times \epsilon^{-1}$ minimaal is.

B. Bounded distance decoding

De tweede soort aanval gaat op zoek naar de geheime vector $\mathbf{s}$ door het Bounded Distance Decoding probleem op te lossen, dat gedefinieerd is als volgt:

Definitie 3 (Bounded Distance Decoding (BDD) Probleem [LN13])**.** *Gegeven een rooster Λ en een punt $\boldsymbol{t}$ "dichtbij" Λ, zoek het roosterpunt $\boldsymbol{z} \in \Lambda$ waarvoor $\|\boldsymbol{z} - \boldsymbol{t}\|$ minimaal is.*

Wanneer men het BDD probleem kan oplossen en het roosterpunt $\mathbf{z} \in \Lambda_q(\mathbf{A})$ dat het dichtste bij $\mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q$ ligt, kan vinden, is het oplossen van het LWE zoekprobleem triviaal.

Het BDD probleem in een rooster $\Lambda(\mathbf{B})$ wordt opgelost met het Nearest Plane algoritme van Babai [Bab86], dat de unieke vector $\mathbf{z}$ zoekt waarvoor $\mathbf{t} - \mathbf{z} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}})$ (met $\tilde{\mathbf{B}}$ de Gram-Schmidt orthogonalisatie van $\mathbf{B}$). Om de kwaliteit van de roosterbasis te verbeteren en de kans op succes

$$\Pr[\text{Succes}] = \Pr[\mathbf{t} - \mathbf{z} = \mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}})]$$

te vergroten, is BKZ 2.0 basis reductie opnieuw nodig als eerste stap. Echter, dan nog is de kans op succes met deze methode zeer klein. Gram-Schmidt orthogonalisatie resulteert doorgaans in grote verschillen tussen de lengte van de eerste en laatste vectoren $\tilde{\mathbf{b}}_1$ en $\tilde{\mathbf{b}}_m$, waardoor het fundamenteel parallellepipedum "lang" en "dun" is. Om een meer vierkante zoekruimte te bekomen, hebben Lindner en Peiker [LP11] elke richting $\tilde{\mathbf{b}}_i$ van het parallellepipedum uitgebreid met een factor d_i. Het nieuwe algoritme produceert een set van alle roostervectoren $\mathbf{z} \in \Lambda(\mathbf{B})$ waarvoor $\mathbf{t} \in \mathbf{z} + \mathcal{P}_{1/2}(\tilde{\mathbf{B}} \cdot \operatorname{diag}(\mathbf{d}))$. De vector $\mathbf{d} \in \mathbb{Z}^m$ wordt gekozen zodat $\min_i(d_i \cdot \|\tilde{\mathbf{b}}_i\|)$ maximaal is.

Naast de rekentijd voor BKZ 2.0 reductie, vergt deze methode $\prod_{i=1}^{m} d_i$ uitvoeringen van het originele algoritme van Babai. De kans dat de methode slaagt is de kans dat de outputset de juiste vector $\mathbf{z}$ bevat, *i.e.* de kans dat $\mathbf{t} - \mathbf{z} = \mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}} \cdot \operatorname{diag}(\mathbf{d}))$. Als de elementen van $\mathbf{e}$ normaal verdeeld zijn, is dit

$$\begin{aligned} \Pr[\text{Succes}] &= \Pr[\mathbf{e} \in \mathcal{P}_{1/2}(\tilde{\mathbf{B}} \cdot \operatorname{diag}(\mathbf{d}))] \\ &= \prod_{i=1}^{m} \operatorname{erf}\Big(\frac{d_i \cdot \|\tilde{\mathbf{b}}_i\| \sqrt{\pi}}{2s}\Big). \end{aligned} \tag{6}$$

Hoe meer een aanvaller op basis reductie inzet, hoe kleiner de factoren d_i mogen zijn om een bepaalde $\Pr[\text{Succes}] = \epsilon$

te halen en dus hoe minder inspanning het Nearest Planes algoritme vergt. Vergelijking (6) toont dus een trade-off tussen de complexiteiten van de twee onderdelen C_{BKZ} en C_{NP}. In een efficiënte aanval, zijn die gebalanceerd. Liu en Nguyen [LN13] merken op dat de totale kost $(C_{\text{BKZ}} + C_{\text{NP}}) \times \epsilon^{-1}$ opnieuw geoptimaliseerd kan worden met een goede keuze van ϵ. Indien het aantal beschikbare LWE samples niet beperkt is, is het immers voordeliger om een lagere kans op succes na te streven en de aanval meerdere keren uit te voeren.

C. BKW algoritme

De derde strategie om LWE op te lossen heeft niets met roosters of basis reductie te maken, maar beschouwt het probleem $\mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q$ met onbekende $\mathbf{s}$ als een linear systeem met ruis. Het BKW algoritme (vernoemd naar bedenkers Blum,Kalai en Wasserman [BKW03]) kan beschouwd worden als een geblokte versie van Gauss-eliminatie en terugsubstitutie. Elke iteratie bestaat uit drie stadia: [ACF+15]

Eerst worden de koppels $(\mathbf{a}, t)$ gereduceerd tot samples waarin slechts b elementen van $\mathbf{a}$ niet nul zijn door herhaaldelijk paren te combineren, waarvoor een blok van b coördinaten in $\mathbf{a}$ overeenkomt. Alle samples en combinaties van samples worden bijgehouden in een reeks tabellen T^i, $i = 1 \ldots \lceil \frac{n}{b} \rceil$ waarbij tabel T^i de paren bevat met $i-1$ blokken van $\mathbf{a}$ gelijk aan nul. In de laatste tabel zitten dan een aantal kleinere LWE samples $(\mathbf{a}', \mathbf{t}' = \langle \mathbf{a}', \mathbf{s}' \rangle + \mathbf{e}' \bmod q)$ waarbij $\mathbf{s}' \in \mathbb{Z}_q^b$ een blok van b componenten van $\mathbf{s}$ is.

In het tweede deel van het algoritme worden kandidaat oplossingen voor deze kleinere geheime vector $\mathbf{s}'$ vergeleken en geëvalueerd. Men weet dat de elementen van de nieuwe ruisvector $\mathbf{e}' = \mathbf{t}' - \langle \mathbf{a}', \mathbf{s}' \rangle \bmod q$ elk een som van onafhankelijke fouten uit χ zijn. De kennis van deze foutenverdeling laat toe het volgende te bewijzen:

Lemma 2 (Lemma 15 van [DTV15])**.** *Gegeven m samples $(\boldsymbol{a}_j, t_j = \langle \boldsymbol{a}_j, \boldsymbol{s} \rangle + e_j \bmod q)$. Als $\hat{f}$ de Fourier transformatie van de functie*

$$f(\boldsymbol{x}) \stackrel{def}{=} \sum_{j=1}^{m} 1_{\boldsymbol{a}_j = \boldsymbol{x}} \cdot e^{2\pi i t_j / q} \quad , \forall \boldsymbol{x} \in \mathbb{Z}_q^r$$

is, dan geldt dat $\operatorname{argmax}_{\boldsymbol{v}} \operatorname{Re}\big(\hat{f}(\boldsymbol{v})\big) = \boldsymbol{s}'$ *met een waarschijnlijkheid groter dan*

$$1 - q^r \cdot \exp\big(-\frac{m}{8} \cdot \mathbb{E}[\cos(2\pi\chi/q)]^{2^d}\big)$$

Men berekent dus $\hat{f}(\mathbf{k})$ voor elke mogelijke $\mathbf{k} \in \mathbb{Z}_q^b$ en kiest de kandidaat oplossing met het grootste reële deel als $\mathbf{s}'$. Vervolgens worden de tabellen gereduceerd met deze oplossing (terugsubstitutie) en wordt de procedure herhaald met één tabel minder voor een volgende blok van componenten.

De totale complexiteit van deze methode werd in detail onderzocht door Duc et al [DTV15].

Theorema 1 (Complexiteit van het BKW algoritme [DTV15])**.** *Beschouw n, q positieve gehele getallen en $A_{\boldsymbol{s},\chi}$ een LWE verdeling, met $\boldsymbol{s} \in \mathbb{Z}_q^n$. Kies $d, b \in \mathbb{N}$ zodat $db = n$. Kies een gewenste succeswaarschijnlijkheid $0 < \epsilon < 1$. Voor elke iteratie $0 \leq j \leq d-1$ is het aantal benodigde samples als volgt:*

$$m_{j,\epsilon'} \stackrel{def}{=} 8 \cdot b \cdot \ln(\frac{q}{\epsilon'}) \cdot \Big(1 - 2(\frac{\pi\sigma}{q})^2\Big)^{-2^{d-j}} \tag{7}$$

met $\epsilon' = (1-\epsilon)/d$. In de veronderstelling dat gereduceerde samples onafhankelijk zijn, bedraagt de tijdscomplexiteit van BKW om, met een waarschijnlijkheid tenminste gelijk aan ϵ, de geheime vector $\boldsymbol{s}$ te vinden, $c_1 + c_2 + c_3 + c_4$ waarbij

$$c_1 = \Big(\frac{q^b - 1}{2}\Big) \cdot \Big(\frac{(d-1)\cdot(d-2)}{2}(n+1) - \ldots \\ \frac{b}{6}(d \cdot (d-1) \cdot (d-2))\Big) \tag{8}$$

het aantal optellingen in $\mathbb{Z}_q$ is om alle tabellen te produceren,

$$c_2 = \sum_{j=0}^{d-1} m_{j,\epsilon'} \cdot \frac{d-1-j}{2} \cdot (n+2) \tag{9}$$

het aantal optellingen in $\mathbb{Z}_q$ is om de benodigde samples te creëren en alle blokken van $\boldsymbol{s}$ te recupereren met waarschijnlijkheid ϵ,

$$c_3 = 2\Big(\sum_{j=0}^{d-1} m_{j,\epsilon'}\Big) + C_{FFT} \cdot n \cdot q^b \cdot \log q \tag{10}$$

het aantal operaties in $\mathbb{C}$ is voor de Fourier transformatie van f en

$$c_4 = (d-1) \cdot (d-2) \cdot b \cdot \frac{q^b - 1}{2} \tag{11}$$

het aantal operaties in $\mathbb{Z}_q$ is voor terugsubstitutie van de tabellen.

De geheugencomplexiteit in het aantal elementen van $\mathbb{Z}_q$ en $\mathbb{C}$ bedraagt respectievelijk

$$\Big(\frac{q^b - 1}{2} \cdot (d-1) \cdot \big(n + 1 - b\frac{d-2}{2}\big)\Big) + m_{0,\epsilon'} \text{ en } q^b$$

Men moet er wel rekening mee houden dat deze methode alleen te gebruiken is als de aanvaller toegang heeft tot een onbeperkt aantal LWE samples.

IV. Complexiteit van BKZ 2.0

Voor twee van de besproken strategieën is BKZ 2.0 reductie een belangrijk instrument. Er bestaat echter nog geen eensgezindheid over de precieze complexiteit van dit algoritme. De kwaliteit van een gereduceerde basis is in de praktijk vaak beter dan de theorie voorspelt. Daarbij hebben we in een cryptografische context te maken met roosterdimensies, waarvoor praktische experimenten nog niet haalbaar zijn. Toch moeten we voor dergelijke parameters in staat zijn om de complexiteit van BKZ 2.0 te schatten als we een realistisch idee van de toekomstige veiligheid van LWE willen verkrijgen.

Volgens Gama en Nguyen [GN08] hangt de rekentijd om een bepaalde roosterbasiskwaliteit δ te bereiken vooral af van de root-Hermite factor δ zelf. Bijgevolg hebben zowel Lindner en Peikert [LP11] als Albrecht et al. [ACF+15] een functie van

alleen δ ontworpen door middel van interpolatie. De schatting van Lindner-Peikert

$$\log_2(T_{\text{BKZ}}(\delta)) = \frac{1.8}{\log_2(\delta)} - 110$$

is gebaseerd op eigen experimenten met het originele BKZ algoritme. Omdat BKZ intussen werd geoptimaliseerd door Chen en Nguyen [CN11], wordt deze benadering niet meer gebruikt. Albrecht et al. interpoleerden data uit het werk van Chen en Nguyen [CN11].

$$\log_2(T_{\text{BKZ}}(\delta)) = \frac{0.009}{\log_2^2(\delta)} - 27$$

Die resultaten werden echter aangepast in een tweede versie van het werk [CN12].

We combineren de resultaten van verschillende auteurs en ontwerpen onze eigen schatting van de BKZ complexiteit als functie van zowel de gewenste basiskwaliteit δ als de roosterdimensie m.

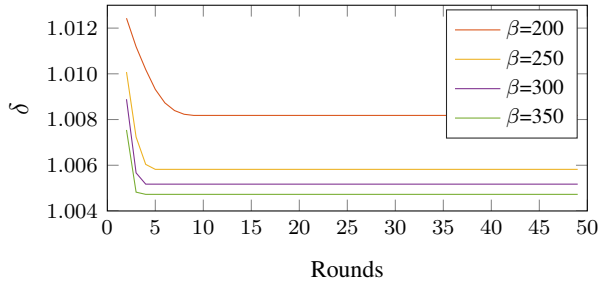
Uit de thesis van Chen, leiden we de functie $\delta(\beta, m)$ af, die de kwaliteit van een m-dimensionele roosterbasis geeft na reductie met het BKZ 2.0 algoritme met blokgrootte β.

$$\delta(\beta, m) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{1/\beta}\right)^{(1+\beta^2/(6m^2))\frac{m-1}{2m(\beta-1)}} \tag{12}$$

Het BZK 2.0 algoritme is de herhaalde uitvoering van een procedure, waarbij elke basisvector $\mathbf{b}_i$ voor $i = 1 \ldots m$ vervangen wordt door een korte vector uit een β-dimensioneel subrooster. In [CN12] vinden we een tabel die de kost van de korte vector subroutine voor bepaalde blokgroottes β geeft. Op basis van deze data produceerden Lepoint en Naehrig [LN14] met een kleinste-kwadratenbenadering de volgende functie:

$$\log_2(\text{\# klokcycli/subroutine}) = 0.64\beta - 20.36$$

De subroutine wordt in elke BKZ 2.0 iteratie m keer uitgevoerd en uit [HPS11] leiden we af dat $\frac{m^2}{\beta^2}\log_2(m)$ iteraties voldoende zijn om een basiskwaliteit te bekomen, die voldoende dicht bij de beste kwaliteit ligt. De root-Hermite factor daalt immers vooral tijdens de eerste rondes (zie Figure 1).



FIGUUR 1. De evolutie van de root-Hermite factor δ met het aantal herhalingen van de BKZ 2.0 procedure

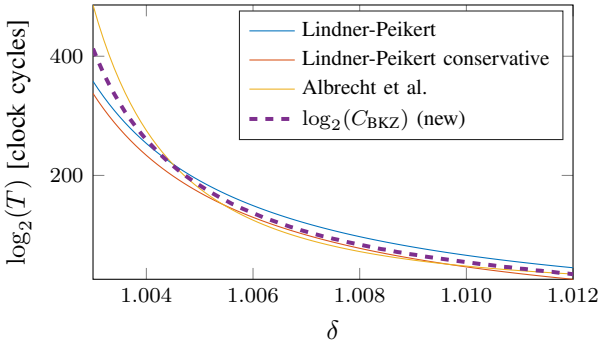
We besluiten dat BKZ 2.0 basis reductie met een blokgrootte β op een m-dimensioneel rooster tot een basiskwaliteit $\delta(\beta, m)$ leidt en $C_{\text{BKZ}}(\beta, m)$ klokcycli vergt waarbij

$$\delta(\beta, m) = \left(\frac{\beta}{2\pi e}(\pi\beta)^{1/\beta}\right)^{(1+\beta^2/(6m^2))\frac{m-1}{2m(\beta-1)}} \tag{13}$$

en

$$\log_2(C_{BKZ}(\beta, m)) = \log_2\left(\frac{m^3}{\beta^2}\log_2(m)\right) + 0.64\beta - 20.36 \tag{14}$$

We zetten de bestaande benaderingen van BKZ rekencomplexiteit om in # klokcycli en vergelijken ze met onze nieuwe schatting in Figuur 2. Velen vinden Lindner en Peikert's benadering te conservatief. Albrecht's functie daarentegen, stijgt zo snel voor dalende δ, dat ze een overschatting dreigt te worden. Bij het kiezen van parameters is een onderschatting van de aanvalcomplexiteit steeds veiliger. Onze benadering reikt daarom een goed compromis aan: voorzichtiger dan Albrecht's functie, maar minder conservatief als die van Lindner en Peikert. Onze formule heeft daarbij het voordeel dat ze kan variëren met de roosterdimensie m.



FIGUUR 2. Vergelijking van BKZ complexiteit benaderingen

V. RESULTATEN

De bekomen formules van complexiteit C en slaagkans of voordeel ϵ kunnen gebruikt worden om de minimale totale complexiteit $C \times \epsilon^{-1}$ te berekenen van een succesvolle aanval tegen LWE.

Voor de aanval tegen het *beslissingsprobleem*, maken we gebruiken van Lemma 1 om de roosterkwaliteit te berekenen, waarmee deze aanval slaagt met voordeel tenminste gelijk aan ϵ:

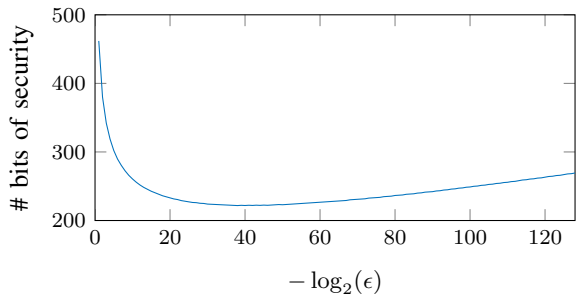
$$\exp\left(-\pi(\frac{\|\mathbf{v}\|s}{q})^2\right) \geq \epsilon \Leftrightarrow -\pi(\frac{\|\mathbf{v}\|s}{q})^2 \geq \ln(\epsilon)$$
$$\Leftrightarrow \|\mathbf{v}\| \leq \frac{q}{s}\sqrt{-\frac{\ln(\epsilon)}{\pi}}$$

De lengte van vector $\mathbf{v}$ (5) is minimaal gelijk aan $2^{2\sqrt{n\log_2 q \log_2 \delta}}$ wanneer de roosterdimensie $m = \sqrt{n\log_2 q/\log_2 \delta}$ [MR09].

$$\|\mathbf{v}\| = 2^{2\sqrt{n\log_2 q\log_2\delta}} \Leftrightarrow \log_2\|\mathbf{v}\| = 2\sqrt{n\log_2 q\log_2\delta}$$
$$\Leftrightarrow \log_2\delta = \frac{(\log_2\|\mathbf{v}\|)^2}{4n\log_2 q} \Leftrightarrow \delta = 2^{(\log_2\|\mathbf{v}\|)^2/(4n\log_2 q)}$$

Voor elke ϵ, kunnen we deze root-Hermite factor δ berekenen en samen met de optimale subdimensie $m = \sqrt{n\log_2 q/\log_2\delta}$ gebruiken om de complexiteit te beoordelen aan de hand van $C_{\text{BKZ}}(\delta, m)$. Figuur 3 toont het resultaat voor een parameter set $(n, q, s) = (320, 4093, 8)$. Hieruit leiden we

af dat het veiligheidsniveau van deze parameters gelijk is aan 221 bits. Dit betekent dat ze weerstaan aan aanvallers die tot 2^{221} operaties kunnen uitvoeren.



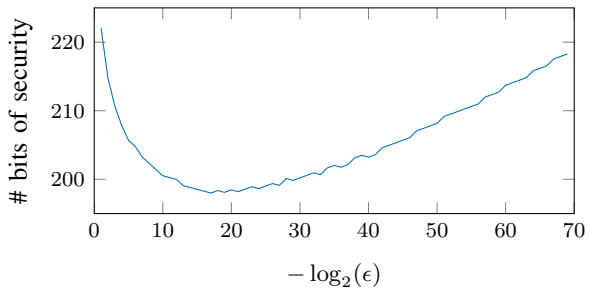
FIGUUR 3. $n = 320, q = 4093, s = 8$. Het aantal bits veiligheid in functie van de door een aanvaller gebruikte ϵ

Voor de *bounded distance decoding* aanval, zoeken we voor elke mogelijk ϵ de roosterkwaliteit δ waarvoor de rekentijden van de twee methode onderdelen (BKZ 2.0 reductie en Nearest Planes) ongeveer gelijk zijn: $\log_2(C_{\text{BKZ}}) \approx \log_2(C_{\text{NP}})$. Voor die laatste complexiteit maken we gebruik van Lindner en Peikert's veronderstelling dat Babai's algoritme ongeveer 2^{16} maal per seconde kan worden uitgevoerd op een 2.3 GHz computer [LP11].

$$\begin{aligned}\log_2(C_{\text{NP}}) &= \log_2(\frac{\text{\# clock cycles}}{\text{Babai}}) + \log_2(\#\text{Babai}) \\ &= \log_2(\frac{2.3 \cdot 10^9}{2^{16}}) + \log_2(\prod_{i=1}^{m} d_i) \\ &= 15 + \log_2(\prod_{i=1}^{m} d_i) \end{aligned} \tag{15}$$

De factoren d_i kiezen we zodanig dat het fundamenteel parallellepipedum zo vierkant mogelijk is en dat $\Pr[\text{Succes}]$ zoals gedefinieerd in (6) gelijk is aan ϵ. De lengtes van de Gram-Schmidt vectoren kunnen berekend worden als functie van δ met behulp van de GSA (4).

In Figuur 4 tonen we opnieuw de resultaten voor $(n, q, s) = (320, 4093, 8)$. De veiligheid van deze parameters volgens bounded distance decoding is gelijk aan 197 bits.



FIGUUR 4. $n = 320$, $q = 4093$, $s = 8$. Het aantal bits veiligheid in functie van de door een aanvaller gebruikte ϵ

Tenslotte laat Theorema 1 toe om de complexiteit van het *BKW algoritme* te berekenen. We moeten alleen het aantal operaties in $\mathbb{Z}_q$ omzetten naar het aantal bit-operaties door vermenigvuldiging met $\log_2(q)$. Voor parameter d experimenteren we met verschillende waarden en opteren we uiteindelijk voor diegene met de laagste complexiteit voor $\epsilon = 0.99$. Tabel I toont enkele resultaten en laat ook zien dat het BKW algoritme voor sommige parameter sets efficienter is dan de bounded distance decoding aanval. Men moet er wel mee rekening houden dat deze methode veeleisend is wat betreft de geheugencomplexiteit.

Tabel I: Vergelijking van de complexiteit van een BKW aanval en een BDD aanval voor twee parameter sets. De geheugencomplexiteit van BKW wordt uitgedrukt in het aantal elementen $\in \mathbb{Z}_q$

n	q	s	d	Sec_{BKW}	$\log_2(\text{Mem}_{\text{BKW}})$	Sec_{BDD}
256	4093	8.3	22	157.7	150.2	146.8
320	4093	8.0	23	185.5	177.9	196.8

VI. CONCLUSIE

Het LWE probleem is een eenvoudig maar krachtig hulpmiddel voor het ontwerp van nieuwe cryptosystemen. We beschreven drie methoden om het probleem aan te vallen, waarvan BDD en BKW de belangrijkste zijn. Elk van beide oplossingsstrategieën kan de andere overtreffen voor bepaalde parameters.

We introduceerden een nieuwe manier om de complexiteit van het BKZ 2.0 algoritme te schatten. Deze nieuwe benadering heeft als voordeel over de bestaande functies dat ze kan variëren met roosterdimensie.

De veiligheidsanalyses op basis van de drie aanvalsmethoden werden geïmplementeerd als web app, die kan gebruikt worden om de veiligheid van elke set LWE parameters op te vragen en zo vertrouwen in LWE gebaseerde cryptosystemen op te bouwen.

REFERENTIES

[ACF+15] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugére, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74(2):325–354, 2015.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, July 2003.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In DongHoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin Heidelberg, 2011.

[CN12] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. full version. http://www.di.ens.fr/~ychen/research/Full_BKZ.pdf, 2012.

[DTV15] Alexandre Duc, Florian Tramér, and Serge Vaudenay. Better algorithms for LWE and LWR. Cryptology ePrint Archive, Report 2015/056 http://eprint.iacr.org/, 2015.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer Berlin Heidelberg, 2008.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *Advances in Cryptology CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 447–464. Springer Berlin Heidelberg, 2011.

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *Topics in Cryptology CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer Berlin Heidelberg, 2013.

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. Cryptology ePrint Archive, Report 2014/062, 2014.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer Berlin Heidelberg, 2011.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 147–191. Springer Berlin Heidelberg, 2009.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, September 2009.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1-3):181–199, 1994.

APPENDIX B

Poster

The Security of LWE-based Cryptosystems

KATHOLIEKE UNIVERSITEIT LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Master Wiskundige ingenieurstechnieken

Masterproef
Lauren De Meyer

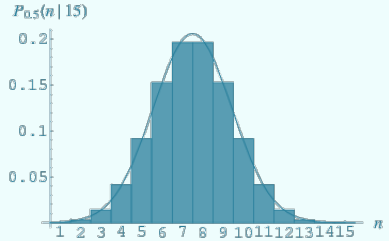
Promotor:
Prof. dr. ir. Bart Preneel

Academiejaar 2014-2015

The LWE Problem

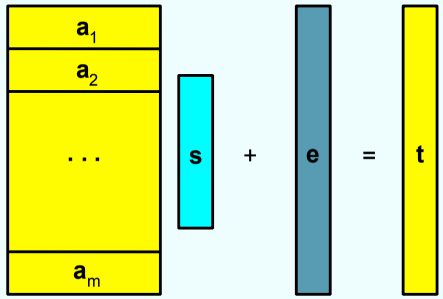
Parameters:

- Dimension n
- Integer modulus q (all elements are in $\mathbb{Z}_q$)
- Error distribution χ (discrete Gaussian with standard deviation σ)



A secret vector **s**

LWE samples: draw $e_i \leftarrow \chi$ and vectors $\mathbf{a}_i$ uniform: $(\mathbf{A},\mathbf{t}) = (\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e} \bmod q)$



Solving LWE = finding **s**

Estimating Security

Investigate complexity of attacks

n	q	σ	SIS	BDD	BKW
256	7681	4,51	159	141	162
320	4093	3,19	215	192	185
512	12289	4,86	363	334	298

Online tool:

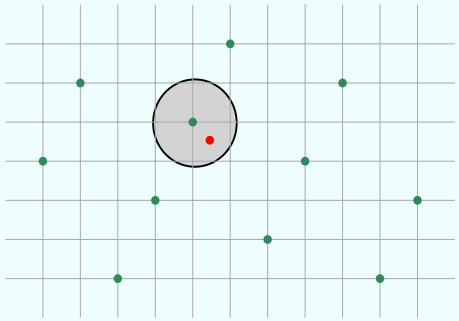
- Compute security of (n,q,σ)
- Compute modulus q for (n,σ) & given security level

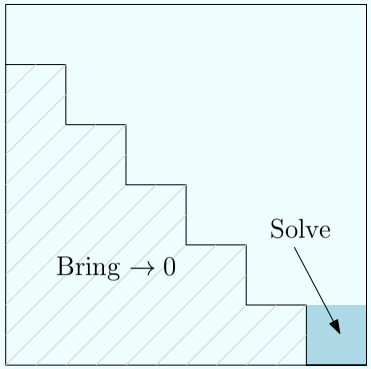
Attacks against LWE

BDD (Bounded Distance Decoding):

$\mathbf{z} = \mathbf{A}^T\mathbf{s} \bmod q \in$ Lattice $\Lambda(\mathbf{A})$

$\rightarrow$ Find the lattice point closest to **t**





BKW:

Solve a noisy linear system

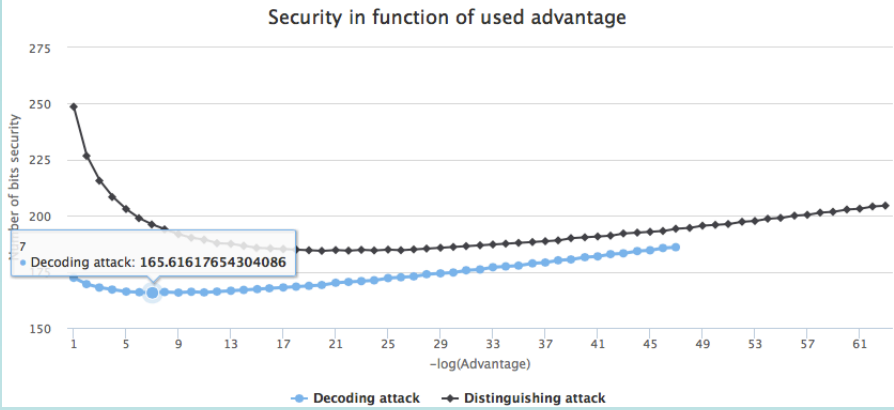
(~ Gaussian elimination & Back Substitution)

Result: Online Security Estimation Tool

n	500
q	497126
s	8.0

This parameter set has **165** bits of security.

SIS	184.33 bits
BDD	165.62 bits
BKW	276.87 bits



APPENDIX C

Web application

The main goal of this work was to create an efficient web app that estimates the security level of any set of LWE parameters (n, q, s). Furthermore, given parameters n, s and a desired security level sec, the tool can calculate the modulus q that is needed to obtain sec bits of security. We now demonstrate how to use this web app and how to interpret its outputs. The tool can be found at http://www.cosic.esat.kuleuven.be/LWESecurity.

C.1 Estimate Security

Figure C.1 shows the homepage, which holds the first functionality: estimating security. As in the examples of section 5.2, we query a security analysis for $(n, q, s) = (320, 4093, 8)$.

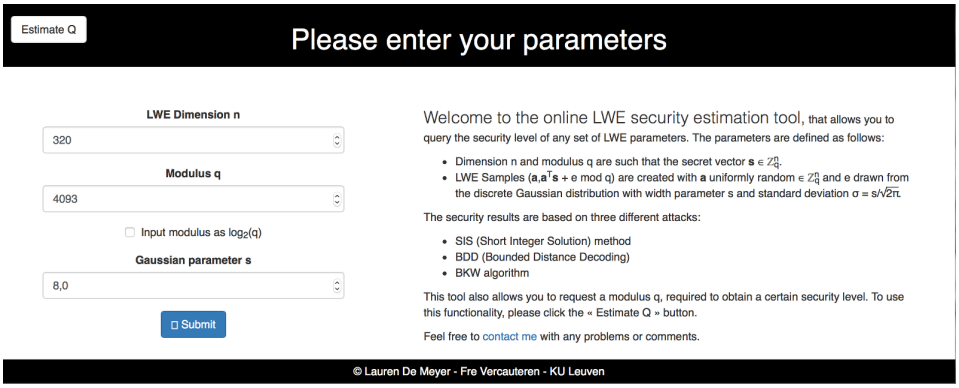


FIGURE C.1. The website's homepage

The result is shown in Figure C.2. At the top of the page, an overview of the total complexity of each attack method is shown and the minimum value is given as the overall security of the parameter set (185 bits). For the SIS- and BDD-based

attack, the graph shows the security level as a function of the used advantage or success probability ϵ. The minimum of both graphs corresponds to the result in the upper right table. The BKW algorithm is always performed for $\epsilon = 0.99$ and therefore doesn't require a graph. Information about the data points is shown when the cursor moves over them and a graph can be made (in)visible by clicking on its legend entry.

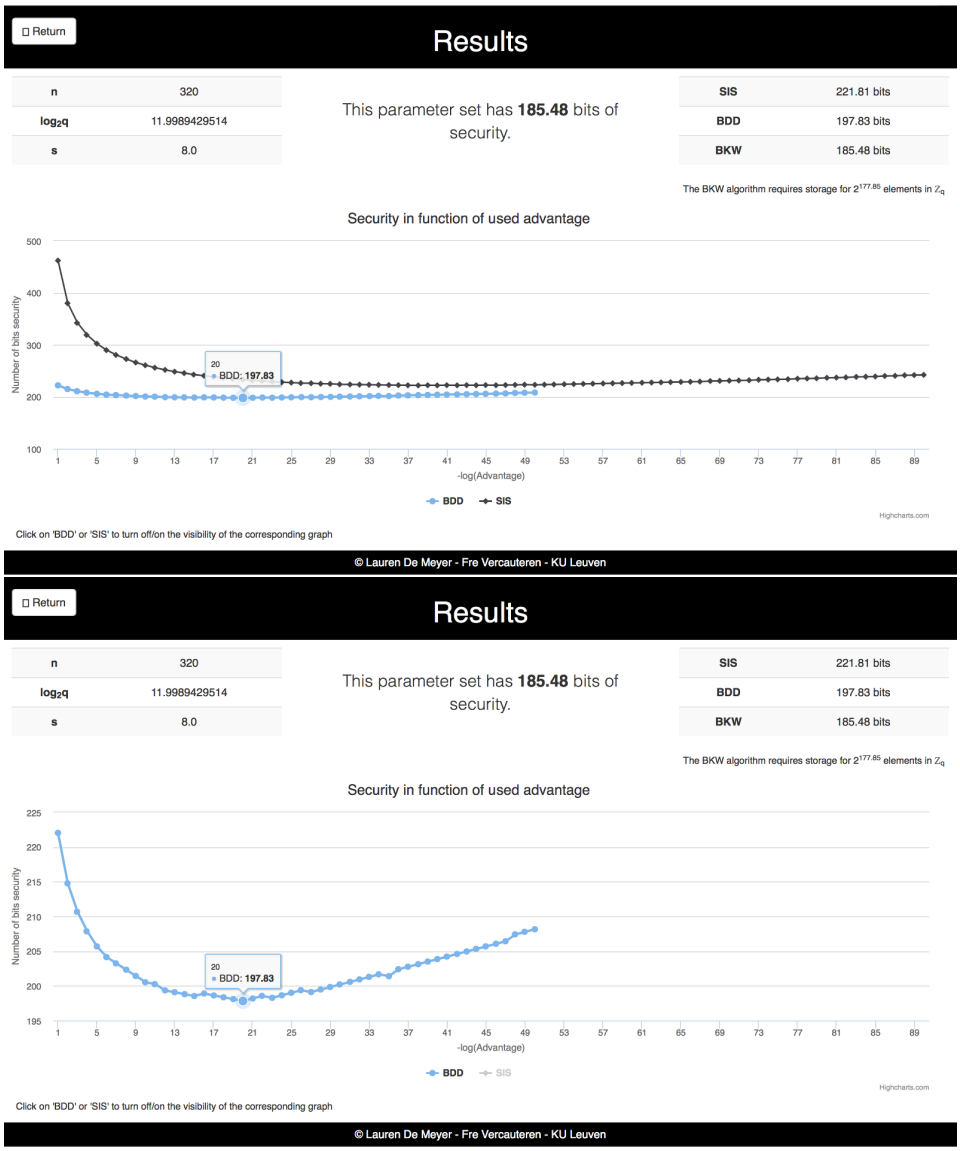


Figure C.2. Result of the security analysis for $(n, q, s) = (320, 4093, 8)$

C.2 Estimate q

We now `Return` to the homepage and click on `Estimate Q` to demonstrate the second purpose. Like the example in Table 5.6, we choose $(n, s) = (320, 8)$ and a security level sec=128 bits (see Figure C.3).

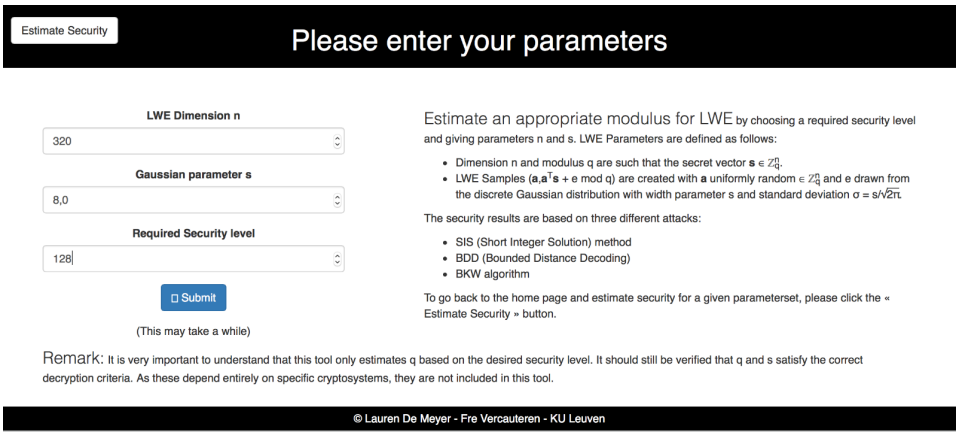


FIGURE C.3. Form to estimate q

Figure C.4 shows the result. At the top of the page, we present the complete parameter set with the newly estimated q and its security analysis. The security level is indeed 128 bits. Below, we show a graph of the approximate security level as a function of $\log_2(q)$, giving the user an indication of the security's sensitivity to changes in the parameter.

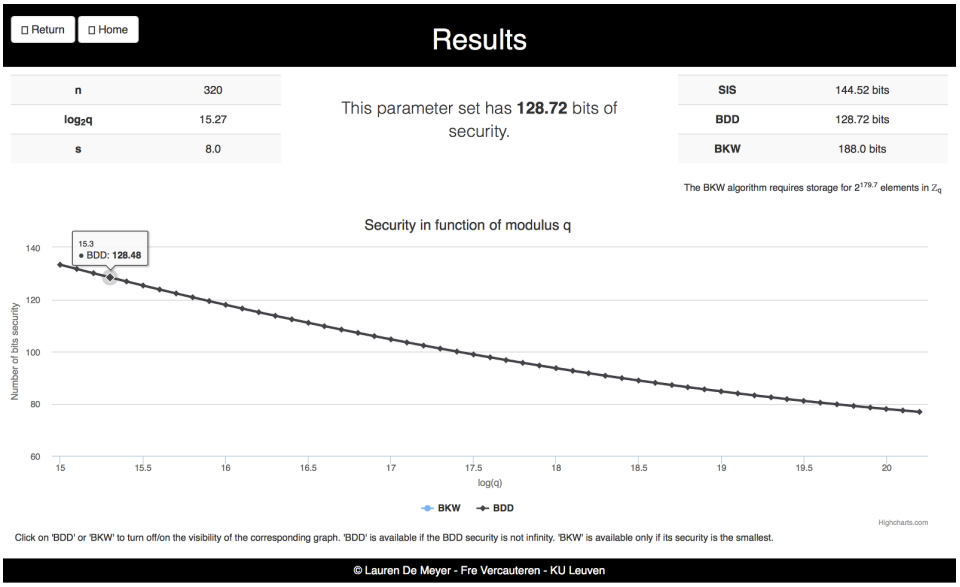


FIGURE C.4. Result of estimating q for $(n, s, sec) = (320, 8, 128)$

Since the SIS-based complexity is always worse than that of BDD or BKW, we

don't include a graph for it. In this case, no graph for the BKW algorithm is shown either, because the attack is not better than BDD for these parameters.

We `Return` to demonstrate a case where BKW does compete with BDD. We let $(n, s) = (512, 8\sqrt{2\pi})$ and request a security level of 400 bits. The result is shown in Figure C.5. For small values of q, the BKW method is typically more efficient (see Figure 5.15). Nonetheless, we also display the BDD graph. With BKW's high memory requirements, users might want to estimate q based on the BDD attack.

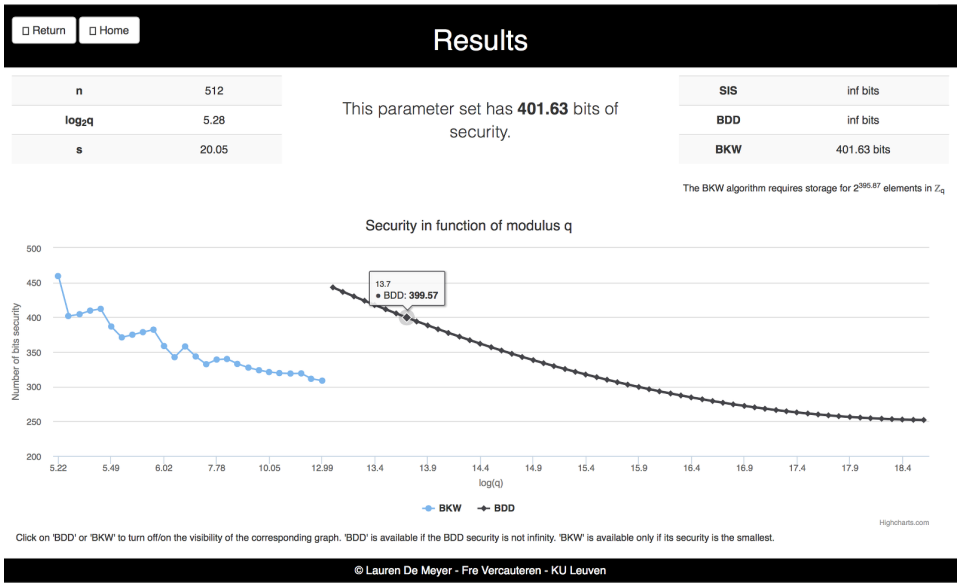


Figure C.5. Result of estimating q for $(n, s, sec) = (512, 8\sqrt{2\pi}, 400)$

C.3 Comparison to Albrecht's implementation

The functionality of calculating the concrete hardness of LWE instances was in the meantime also implemented by Albrecht in a Sage module [Alb15]. We will compare our website with his and explain the differences using Albrecht's code on Bitbucket.

Albrecht's tool includes two more attacks against LWE. We may ignore these as their complexity is too high to compete with that of BKW or BDD. We show the results of our and Albrecht's tool for the parameter set $(n, q, s) = (256, 65\,537, 64)$ in Table C.1

Table C.1. Comparison of Albrecht's results to ours for $(n, q, s) = (256, 65\,537, 64)$

	Us	Albrecht
SIS	188.94	208.7
BDD	168.5	180.9
BKW	182.74	182.8

The security result for the BKW algorithm is identical because we both use the approach of [DTV15]. The divergence for the lattice-based attacks can largely be explained by the choice of our BKZ 2.0 runtime. Albrecht uses Chen's limit $\lim_{m\to\infty} \delta(\beta, m)$ (5.1) to figure out the blocksize that is required to obtain a particular basis quality, whereas we also include the dependency on m: $\delta(\beta, m)$ (5.2). Furthermore, for the shortest vector subroutine in BKZ, we assume

$$\log_2(\#\text{ clock cycles for subroutine}) = 0.64\beta - 20.36 \tag{5.6}$$

and Albrecht uses

$$\log_2(\#\text{ clock cycles for subroutine}) = 0.00290\beta^2 - 0.12266\beta + 31.47497 \tag{C.1}$$

These two costs were compared in Figure 5.3. The data point from [CN12] appear to agree more with Lepoint and Naehrig's function (5.6). However, if we apply these changes to our tool, we obtain the results shown in Table C.2

TABLE C.2. Comparison of Albrecht's results to ours when we use his BKZ 2.0 runtime estimate for $(n, q, s) = (256, 65537, 64)$

	Us	Albrecht
SIS	208.24	208.7
BDD	180.39	180.9
BKW	182.74	182.8

The dissimilarities have almost completely disappeared. The remaining difference in the SIS-based estimate can be attributed to the fact that Albrecht's calculations stop as soon as the security level starts to rise, at $\epsilon = 2^{-51}$. Our estimate is found for $\epsilon = 2^{-54}$ and is slightly lower. The small contrast in $\text{Security}_{\text{BDD}}$ is due to our distinct strategies in searching for a suitable root-Hermite factor δ (see Algorithm 5). Albrecht's approach is somewhat less optimized and therefore results in a longer waiting time.

Bibliography

[ACF+15] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the BKW algorithm on LWE. *Designs, Codes and Cryptography*, 74(2):325–354, 2015.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer Berlin Heidelberg, 2009.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *Proceedings of the 38th International Colloquim Conference on Automata, Languages and Programming - Volume Part I*, ICALP'11, pages 403–415, Berlin, Heidelberg, 2011. Springer-Verlag.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pages 601–610, New York, NY, USA, 2001. ACM.

[Alb15] Martin R. Albrecht. Sage module for solving concrete lwe instances, 2015.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015.

[Bab86] László Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, July 2003.

[BLLN13] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. Cryptology ePrint Archive, Report 2013/075, 2013. http://eprint.iacr.org/.

[Che13] Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, l'Université Paris Diderot, 2013.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In DongHoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology ,Äì ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer Berlin Heidelberg, 2011.

[CN12] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. full version. http://www.di.ens.fr/~ychen/research/Full_BKZ.pdf, 2012.

[DTV15] Alexandre Duc, Florian Tramér, and Serge Vaudenay. Better algorithms for LWE and LWR. Cryptology ePrint Archive, Report 2015/056 http://eprint.iacr.org/, 2015.

[Eis09] Friedrich Eisenbrand. Integer programming and algorithmic geometry of numbers. In *50 Years of Integer Programming 1958-2008. The Early years and State-of-the-Art Surveys.*, chapter 14. Springer-Verlag, 2009.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, pages 365–377, New York, NY, USA, 1982. ACM.

[GN08] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel Smart, editor, *Advances in Cryptology EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer Berlin Heidelberg, 2008.

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer Berlin Heidelberg, 2010.

[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. ACM.

[HN89] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *Advances in Cryptology CRYPTO 2011*, volume 6841 of *Lecture*

Notes in Computer Science, pages 447–464. Springer Berlin Heidelberg, 2011.

[HW11] Jesko Hüttenhain and Lars Wallenborn. Topics in post-quantum cryptography: Lattice based methods. Seminar at the Mathematical Institute of the University Bonn, January 2011.

[LLL82] Arjen Lenstra, Jr. Lenstra, Hendrik, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LLS90] Jeffrey C. Lagarias, Hendrik Lenstra, and Claus-Peter Schnorr. Korkin-zolotarev bases and successive minima of a lattice and its reciprocal lattice. *Combinatorica*, 10(4):333–348, 1990.

[LMvdP14] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Finding shortest lattice vectors faster using quantum search. Cryptology ePrint Archive, Report 2014/907, 2014. http://eprint.iacr.org/.

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *Topics in Cryptology CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer Berlin Heidelberg, 2013.

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes FV and YASHE. Cryptology ePrint Archive, Report 2014/062, 2014.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer Berlin Heidelberg, 2011.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2010.

[Mic98] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. In *in Proc. 39th Symposium on Foundations of Computer Science*, pages 92–98, 1998.

[Mic10] Daniele Micciancio. Duality in lattice cryptography. In *Public Key Cryptography (Paris)*, 2010. Invited talk.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, April 2007.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors,

Post-Quantum Cryptography, pages 147–191. Springer Berlin Heidelberg, 2009.

[Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 333–342, New York, NY, USA, 2009. ACM.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, September 2009.

[Rot01] Richard L. Roth. A history of lagrange's theorem on groups. *Mathematics Magazine*, 74(2):99–108, 2001.

[RS10] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. *IACR Cryptology ePrint Archive*, 2010:137, 2010.

[SB10] Michael Schneider and Johannes Buchmann. Extended lattice reduction experiments using the BKZ algorithm. *Sicherheit*, 170:241–252, 2010.

[Sch03] Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In *In Proc. STACS 2003, Eds. H. Alt and M. Habib, LNCS 2607*, pages 145–156. Springer, 2003.

[Sch13] Michael Schneider. Sieving for shortest vectors in ideal lattices. In Amr Youssef, Abderrahmane Nitaj, and AboulElla Hassanien, editors, *Progress in Cryptology AFRICACRYPT 2013*, volume 7918 of *Lecture Notes in Computer Science*, pages 375–391. Springer Berlin Heidelberg, 2013.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1-3):181–199, 1994.

[Sho] Victor Shoup. Number theory library (NTL) for C++. http://www.shoup.net./ntl/.

[Sho99] Peter Williston Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41:303–332, January 1999.

[Ste] Damien Stehlé. Floating point LLL library. https://github.com/dstehle/fplll.

[Ste14] Ron Steinfeld. NTRU cryptosystem: Recent developments and emerging mathematical problems in finite polynomial rings. In H. Niederreiter, A. Ostafe, D. Panario, and A. Winterhof, editors, *Algebraic Curves and Finite Fields: Cryptography and Other Applications*, volume 16 of *Radon Series on Computational and Applied Mathematics*. De Gruyter, 2014.

[Vau06] Serge Vaudenay. *A Classical introduction to Cryptography: Applications for Communication Security*. Springer US, 2006.

[vdPS13] Joop van de Pol and Nigel P. Smart. Estimating key sizes for high dimensional lattice-based systems. In Martijn Stam, editor, *Cryptography and Coding*, volume 8308 of *Lecture Notes in Computer Science*, pages 290–303. Springer Berlin Heidelberg, 2013.

Fiche masterproef

Student: Lauren De Meyer

Titel: Security of LWE-based cryptosystems

Nederlandse titel: Veiligheid van LWE cryptosystemen

UDC: 51-7

Keywords: lattice-based cryptography, learning with errors, concrete security, lattices, basis reduction, BKZ, bounded distance decoding, BDD, BKW

Korte inhoud:
Mathematical problems such as integer factorization and the discrete logarithm problem have formed the foundation of public-key cryptosystems for many years. Today, the security of various prevailing encryption schemes is endangered by the prospect of quantum computing. As a result, we see a renewed interest in lattice-based cryptography, which appears to be resistant to quantum attacks. In particular, the Learning with Errors (LWE) problem has been proven to be equally hard to solve as worst-case lattice problems. It has therefore become an important building block in modern cryptographic systems and a popular topic in present-day research. Its significance also stems from having extremely various applications, such as fully-homomorphic encryption and identity-based cryptography. However, progress in the development of new cryptographic algorithms for real-world applications has somewhat been thwarted by the lack of concrete security estimates. Theoretical and asymptotic statements are not enough to inspire confidence in this relatively young group of cryptosystems. Furthermore, designers must understand how a choice of parameters influences the security of their cryptosystems.
In this thesis, we collect the most important attacks against the LWE problem and investigate their complexity as a function of the parameters. Our results lay the basis for a web application that allows people to query the security of any set of parameters. Unlike most publications that focus on just one kind of attack against LWE, our website considers the impact of multiple competing methods on the security. Moreover, based on these formulas and security estimates, we are able to calculate the value of certain parameters in order to achieve a particular security level. The aim of this application is to simplify the process of choosing parameters, inspire more trust in the security of LWE and boost the conception of new cryptographic schemes for a post-quantum world.

Thesis voorgedragen tot het behalen van de graad van Master of Science in de ingenieurswetenschappen: wiskundige ingenieurstechnieken

Promotor: Prof. dr. ir. Bart Preneel

Assessoren: Prof. dr. ir. D. Huybrechs
Prof. dr. ir K. Meerbergen

Begeleider: Dr. ir. F. Vercauteren