ALGORITHMS IN LATTICE-BASED CRYPTANALYSIS

by

Shaun Miller

A Dissertation Submitted to the Faculty of

The Charles E. Schmidt College of Science

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

Florida Atlantic University

Boca Raton, FL

August 2020

ALGORITHMS IN LATTICE-BASED CRYPTANALYSIS

by

Shaun Miller

This dissertation was prepared under the direction of the candidate's dissertation advisor, Dr. Shi Bai, Department of Mathematical Sciences, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the Charles E. Schmidt College of Science and was accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

SUPERVISORY COMMITTEE:

Shi Bai

Shi Bai, Ph.D.
Dissertation Advisor

Rainer Steinwandt, Ph.D.
Dissertation Co-Advisor

Koray Karabina, Ph.D.

Spyros Magliveras, Ph.D.

Edoardo Persichetti
Edoardo Persichetti (Jul 22, 2020 13:29 EDT)

Edoardo Persichetti, Ph.D.

Rainer Steinwandt, Ph.D.
Chair, Department of Mathematical Sciences

Teresa Wilcox
Teresa Wilcox (Jul 22, 2020 14:07 EDT)

Teresa Wilcox, Ph.D.
Interim Dean, The Charles E. Schmidt College of Science

Robert W. Stackman Jr., Ph.D.
Dean, Graduate College

July 22, 2020

Date

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my advisors, Dr. Shi Bai and Dr. Rainer Steinwandt. This dissertation would not have been possible without your support. Under your supervision, my graduate studies were an incredibly rewarding experience. I am fortunate to have studied amongst my helpful peers and professors at Florida Atlantic University. In particular, I would like to thank my committee members for your thoughtful comments and enthusiasm throughout my studies. Many resources have been provided to me through grants supported by the National Institute of Standards and Technology (NIST) and the NATO Science for Peace and Security Programme. I would like to thank these organizations for the financial support which allowed me to focus on my research in a comfortable environment with financial security.

My friends and family have consistently been present during my graduate studies. I would like to personally thank Dr. Hakim Mohamoud and Matthew Meinholz for your powerful friendship during these many years. Last but not least, I would like to express my thanks to my mother and stepfather, Jennifer and Gary Johnston, who have supported me during the entirety of my education.

ABSTRACT

Author:	Shaun Miller
Title:	Algorithms in Lattice-Based Cryptanalysis
Institution:	Florida Atlantic University
Dissertation Advisor:	Dr. Shi Bai
Degree:	Doctor of Philosophy
Year:	2020

An adversary armed with a quantum computer has algorithms[66, 33, 34] at their disposal, which are capable of breaking our current methods of encryption. Even with the birth of post-quantum cryptography[52, 62, 61], some of best cryptanalytic algorithms are still quantum [45, 8]. This thesis contains several experiments on the efficacy of lattice reduction algorithms, BKZ and LLL. In particular, the difficulty of solving Learning With Errors is assessed by reducing the problem to an instance of the Unique Shortest Vector Problem. The results are used to predict the behavior these algorithms may have on actual cryptographic schemes with security based on hard lattice problems. Lattice reduction algorithms require several floating-point operations including multiplication. In this thesis, I consider the resource requirements of a quantum circuit designed to simulate floating-point multiplication with high precision.

NOTATION

The following notations are used in this dissertation.

$0^{\otimes n}$ The all zeros vector of length n

$\mathbf{b}_i$ A basis element of the lattice $\mathfrak{L}$

χ A normal distribution with standard deviation σ

ℓ The length of the exponent of a floating-point number

$\mathfrak{L}$ An integer lattice of dimension d

$\mathfrak{L}(\mathfrak{B})$ The integer lattice spanned by the rows of $\mathfrak{B}$

$\mathfrak{L}[i:j]$ The projected lattice spanned by $\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), \ldots, \pi_i(\mathbf{b}_{j-1})$

$\mathfrak{L}_q(\mathbf{A})$ The integer lattice $\{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} \equiv \mathbf{A}\mathbf{x} \pmod{q}, \forall \mathbf{x} \in \mathbb{Z}^n\}$

$\mathbf{0}_{n \times m}$ The $n \times m$ all zeros matrix

$\mathbf{A}^T$ The transpose of a matrix $\mathbf{A}$

$\mathbf{I}_n$ The $n \times n$ identity matrix

$\Phi_n(x)$ The *cyclotomic polynomial* whose roots are the primitive n^{th} roots of unity

$\pi_i(\mathfrak{v})$ The projection of the lattice vector $\mathfrak{v}$ orthogonal to $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}$

$\mathbb{Q}$ The field of rational numbers

$\mathbb{R}$ The field of real numbers

$\mathbb{R}^d$	The d-dimensional vector space over the real numbers
σ	The standard deviation of normal distribution χ
$\mathfrak{v}$	An element of the lattice $\mathfrak{L}$
$\mathbf{x}$	A vector in $\mathbb{R}^d$
$\mathbb{Z}$	The ring of rational integers
$\mathbb{Z}^d$	The d-dimensional module space over the integers
$\mathbb{Z}_q$	The ring of rational integers modulo q
k	The precision of a floating-point number
$R[x]$	The set of polynomials in x with coefficients in R
$R^{r\times c}$	The matrix space with r rows and c columns with coefficients from R
$width$	The total bits needed to store a floating-point number

To my lovely mother, Jennifer

ALGORITHMS IN LATTICE-BASED CRYPTANALYSIS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In this chapter we review background material needed for analyzing the performance of lattice reduction algorithms. Each succeeding section is written with the intent of direct application to algorithms used in lattice-based cryptography. Theorems and problems in the succeeding sections are instantiated with the intent of narrowing the focus of the material. Though, it is often the case that more general versions of the problems exist.

1.1 CRYPTOGRAPHY

The pursuit of secure communication has been in constant disarray due to breakthroughs in technology, evolving political climate, and the adaptation of different cultures. Use of the Internet, becoming greater by the day, requires carefully constructed channels of communication to ensure user information is transferred securely. This secure form of communication is the objective of cryptography. In modern day cryptography, two users encrypt messages using a symmetric-key (also known as secret-key) encryption scheme such as the Advanced Encryption Standard (AES) [26], originally titled Rijndael. Suppose these two users are the popular Alice and Bob. Both agree on a key, Alice encrypts a message with the key and sends the encrypted message or ciphertext to Bob. Bob decrypts the ciphertext with the same key as Alice, revealing her message. An attack is any attempt by an adversary to break the scheme by revealing some substantial information about the message.

Establishing Alice and Bob's key over an insecure channel presents another cryp-

tographic challenge. As it turns out, Alice and Bob are able to transmit information viewable by everyone and construct a key known only to Alice and Bob. Public-key cryptography provides secure communication with the added benefit of allowing users to generate their own different keys. Many authors opt to call this model asymmetric-key cryptography for that reason. RSA [63, 23] has been one of the most frequently used public-key cryptosystems since being invented in 1977. Though, it should be noted elliptic curves offer more security with smaller key sizes. It is of little surprise that many companies made the switch from RSA to elliptic curve cryptography. We refer the reader to [43] for more on elliptic curves and choose to focus on RSA in the section due to the historical significance and the simplicity. Textbook RSA, as presented by Stinson [69], requires the generation of two prime numbers $p, q \in \mathbb{Z}$. Let $a, b \in \mathbb{Z}_{\phi(pq)}$ be such that $ab \equiv 1 \bmod \phi(pq)$ where ϕ is Euler's totient function. Encryption of a message $x \in \mathbb{Z}_{pq}$ in the ciphertext $y \in \mathbb{Z}_{pq}$ is done as follows:

$$y \equiv x^b \bmod pq.$$

A given ciphertext, y, can be decrypted with a.

$$x \equiv y^a \bmod pq$$

We will use Alice and Bob to paint the picture. Say Alice wants to receive an encrypted message from Bob. Alice generates p, q, a, and b as above. The values (pq, b) form the public-key. Anyone, including Bob and a malicious eavesdropper, is able to view the public-key. Bob encrypts x and sends the corresponding ciphertext, y, to Alice. Alice calculates $y^a \equiv x^{ab} \equiv x \bmod pq$, revealing Bob's message. A private key can be established through RSA. If Bob generates the key, he will be able to securely transfer it to Alice. Just replace the key with x in the above discussion. In practice, public-key cryptography is used specifically for exchanging keys and authentication. AES is much faster than the discussed public-key alternatives. Thus, encryption of messages is done by means of AES with a key established by a public-key encryption

scheme.

Security will be necessarily discussed extensively throughout the thesis. Any of the discussed encryption schemes can be broken by an adversary with an exponential amount of time. However, no such adversary exists. If an attack takes a few trillion years and requires a trillion times the computational power of the world, we should be confident in the encryption scheme's security against the attack. A secure cryptographic scheme suffers from no attack that can be mounted in polynomial time. Only through years of analyzing the best attacks, called cryptanalysis, can we gain confidence in the security of cryptography. Measuring security reduces to understanding the complexity of the best known attack. There has been over 40 years of cryptanalysis concentraiting on RSA. In fact, a team has just factored an 829-bit RSA public-key [20]. Though a major breakthrough, this is a small public-key for RSA and was not considered to be secure. These breakthroughs in attacks offer insight into attacks of larger parameter sets. A 2048-bit RSA key is believed to be secure against all foreseeable attacks using a classical computer. If there are more breakthroughs in factoring large numbers, the key size used in implementations of RSA will need to be reconsidered. The impending threat of future attacks is the primary concern of this thesis. For if an adversary should get their hands on an operational quantum computer, RSA will be obsolete [66].

1.2 LATTICES

Integer combinations of linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m \in \mathbb{R}^n$ form a lattice $\mathfrak{L} = \{\sum_{i=1}^{m} x_i \mathbf{b}_i | x_i \in \mathbb{Z}\}$. The set of vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m$ are called the *basis* of the lattice. Put simply, a lattice is a discrete subspace of $\mathbb{R}^n$. We will assume lattices are of full rank, i.e., $m = n$. The determinant or volume of a lattice, denoted $\mathrm{Vol}(\mathfrak{L})$, is the determinant of the $n \times n$ matrix with rows $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$. The volume of a lattice is independent of the basis used. Geometrically, $\mathrm{Vol}(\mathfrak{L})$ refers to the

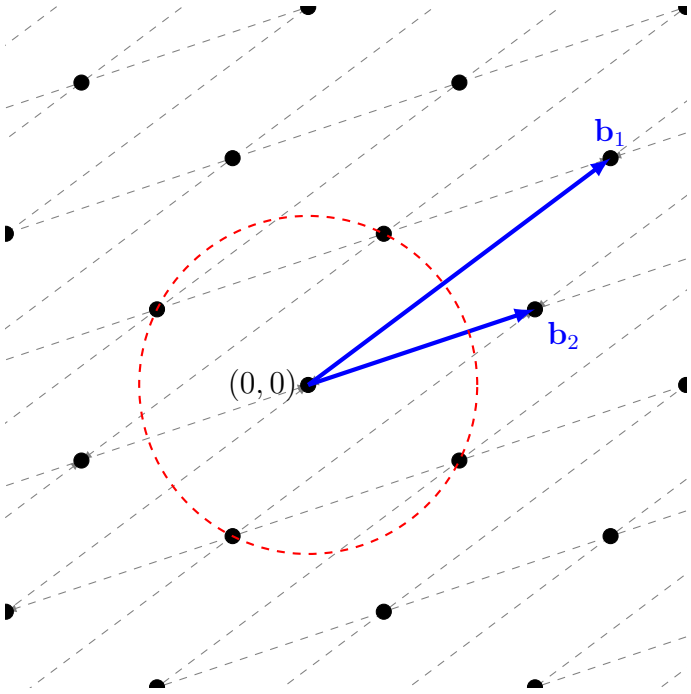

Figure 1: Lattice Generated by $\mathbf{b}_1 = (4,3)$ and $\mathbf{b}_2 = (3,1)$ with $\lambda_1(\mathfrak{L}) = \sqrt{5}$.

n-dimensional volume of the parallelipiped in $\mathbb{R}^n$ whose edges are the basis vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$. Let D_r be the n-dimensional ball in $\mathbb{R}^n$ of Euclidean radius r centered at the origin. Denote the volume of D_1 as V_n. Often we refer to the ℓ_2 norm of the shortest vector of the lattice denoted by $\lambda_1(\mathfrak{L})$. This can be extended successively:

Definition 1.2.1 (Successive minima)**.** For any lattice $\mathfrak{L}$, the i-th minimum $\lambda_i(\mathfrak{L})$ is the radius of the smallest ball with center the origin and containing i linearly independent lattice vectors:

$$\lambda_i(\mathfrak{L}) = \inf\{r : \ \dim(\mathrm{span}(\mathfrak{L} \cap \mathcal{D}_r)) \geq i\}.$$

Minkowski's convex body theorem [55] states that $\lambda_1(\mathfrak{L}) \leq 2V_n^{-1/n}\mathrm{Vol}(\mathfrak{L})^{1/n}$. The average version of Minkowski's theorem is often referred to as the Gaussian Heuristic: the λ_1 of a random n-dimensional lattice is asymptotically

$$GH(\mathfrak{L}) = V_n^{-1/n}\mathrm{Vol}(\mathfrak{L})^{1/n}$$

Lattice algorithms often consider a basis of *mutually orthogonal* vectors that also spans $\mathbb{R}^n$. Given any basis of $\mathbb{R}^n$, one can derive mutually orthogonal vectors spanning the same space by calculating the Gram-Schmidt Orthogonalization of the input.

Definition 1.2.2. The **Gram-Schmidt Orthogonalization** of a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ of $\mathbb{R}^n$ is the basis $\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*$ where

$$\mathbf{b}_1^* = \mathbf{b}_1$$

$$\mathbf{b}_k^* = \mathbf{b}_k - \sum_{j=1}^{k-1} \mu_{k,j} \mathbf{b}_j^* \text{ where } \mu_{k,j} = \frac{\mathbf{b}_k \cdot \mathbf{b}_j^*}{\mathbf{b}_j^* \cdot \mathbf{b}_j^*}$$

The resulting mutually orthogonal basis can be made orthonormal by replacing $\mathbf{b}_i$ with $\mathbf{b}_i * \frac{1}{\|\mathbf{b}_i\|}$. Standard texts [70, 38] vary in deciding whether or not the resulting basis from Gram-Schmidt Orthogonalization be normalized. We do not take this extra step in normalizing the vectors while discussing a Gram-Schmidt Orthogonalized basis. It is worth noting that the software packages such as fpylll [28, 27] used in section 2.3 make use of a normalized Gram-Schmidt Orthogonalized basis but store the unnormalized lengths $\|\mathbf{b}_i^*\|$.

Definition 1.2.3. Vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ are **mutually orthogonal** if $\mathbf{b}_j \cdot \mathbf{b}_i = 0$ for $i \neq j$. If the extra condition $\mathbf{b}_i \cdot \mathbf{b}_i = 1$ is satisfied, the vectors are said to be **orthonormal**.

The reader should note that the Gram-Schmidt Orthogonalization (GSO) of a basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ will almost always generate a different lattice. Lattice vectors are restricted to integer combinations of basis vectors often excluding $\mu_{k,j} \in \mathbb{Q}$. An equivalent definition of volume emerges from defining GSO, namely $\text{Vol}(\mathfrak{L}) = \prod_{i \leq n} \|\mathbf{b}_i^*\|$. Take into consideration that with all implementations of the Gram-Schmidt process, the values $\mu_{k,j}$ are calculated to some precision. Therefore, resulting vectors are only an approximations of truly orthogonal vectors.

Algorithm 1: Gram-Schmidt Algorithm of [40]

Input: linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{R}^n$

Output: mutually orthogonal basis vectors $\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_n^*$ and transformation matrix $\mathbf{M}$

for *i from 1 to n* **do**

  $\mathbf{b}_i^* \leftarrow \mathbf{b}_i$;

  for *j from* 1 *to* $i-1$ **do**

    $\mu_{k,j} \leftarrow \frac{\mathbf{b}_k \cdot \mathbf{b}_j^*}{\mathbf{b}_j^* \cdot \mathbf{b}_j^*}$;

    $\mathbf{b}_k^* \leftarrow \mathbf{b}_k^* - \mu_{k,j}\mathbf{b}_j^*$;

  end

end

Denote the projection of $\mathfrak{v}$ orthogonal to the linear subspace $(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})$ as $\pi_k(\mathfrak{v})$. When $k = 1$, π_k acts as the identity transformation. A lattice vector $\mathfrak{v} \in \mathfrak{L}$ can be decomposed into $\mathfrak{v} = \pi_k(\mathfrak{v}) + \sum_{j=1}^{k-1} \mu_{k,j}\mathbf{b}_j^*$ where $\mu_{k,j} = \frac{\mathfrak{v} \cdot \mathbf{b}_j^*}{\mathbf{b}_j^* \cdot \mathbf{b}_j^*}$. Notice $\pi_k(\mathfrak{v})$ is the component of $\mathfrak{v}$ in $(\mathbf{b}_1, \ldots, \mathbf{b}_{k-1})^\perp \subseteq (\mathbf{b}_k, \mathbf{b}_{k+1}, \ldots, \mathbf{b}_n)$ and $\sum_{j=1}^{k-1} \mu_{k,j}\mathbf{b}_j^*$ is the component of $\mathfrak{v}$ in $(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{k-1})$. It should be clear that $\pi_k(\mathbf{b}_k) = \mathbf{b}_k^*$ and $\|\pi_k(\mathfrak{v})\| \leq \|\mathfrak{v}\|$. Let $\mathfrak{L}_{[i,j]}$ denote the sublattice spanned by the basis $(\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}) \ldots, \pi_i(\mathbf{b}_j))$.

Length will always be discussed in the ℓ_2 norm in this thesis. The shortest vector problem (SVP) is stated simply: given a lattice $\mathfrak{L}$ find a nonzero $\mathfrak{a} \in \mathfrak{L}$ such that $\|\mathfrak{a}\| \leq \|\mathfrak{b}\|$ for all $\mathfrak{b} \in \mathfrak{L}$. Van Emde Boas conjectured SVP was NP-hard in 1981 [74] and this was proven in the affirmative by Ajtai in 1997 [3]. In other words, Ajtai showed there is a probabilistic Turing-machine which reduces any problem in NP to instances of the shortest vector problem in polynomial time. The first minimum (and sometimes even the second (Section 2.1)) is known using the Gaussian Heuristic, providing approximation of the length of the shortest vector of $\mathfrak{L}$. However, no known polynomial time algorithm exists which is able to check whether or not $\mathfrak{a}$ is

indeed the shortest vector in $\mathfrak{L}$. This fact excludes SVP from the list of problems known to be NP-Complete. The exact SVP problem is too hard to use in practice. In cryptography, one needs a more relaxed version of the problem. We will often concern ourselves with the unique shortest vector problem (uSVP) instead of the standard SVP.

Definition 1.2.4 (Unique Shortest Vector Problem: uSVP$_\gamma$)**.** Let $\gamma \geq 1$. Given as input a basis of a lattice $\mathfrak{L}$ such that $\lambda_2(\mathfrak{L}) \geq \gamma \cdot \lambda_1(\mathfrak{L})$, the goal is to find a non-zero vector $\mathfrak{v} \in \mathfrak{L}$ of norm $\lambda_1(\mathfrak{L})$. We will denote the γ as the *gap* of the uSVP$_\gamma$ problem.

Enumeration and sieving algorithms are more practical than combinatorial methods [17, 1] when considering lattice problems. The complexity of sieving is $2^{O(d)}$ which is faster than enumeration's $2^{O(d\log(d))}$. This advantage in complexity comes at the cost of exponential space requirements [45]. Sieving would likely be more efficient than enumeration when considering a lattice of such large dimesion. Laarhoven, Mosca, and van de Pol [45] claim sieving requires storing a list with approximately $2^{0.2075d}$ entries; d being the dimension of the input lattice. See the selection of block size in section 2.3 for more information on "blocks" of lattices in cryptanalysis. We consider the a sublattice of dimension 396. Using sieving as an SVP oracle would then be required to store a list of length $2^{0.2075(386)} \approx 2^{80}$. That's around 1,200,000,000,000 trillion list entries! Storage concerns leave much room for debate between the two methods. [7] suggests sieving is faster for lattices larger than 250. All experiments done in the thesis will use enumeration to circumvent the storage requirements of sieving. Sieving may be faster asymptotically, but enumeration seems to be faster in practice while considering smaller dimensional lattices. Due to the difficulty of the problem, regardless of choice between enumeration and sieving, small parameters are usually used in experiments. We select parameters in experiments that restrict SVP oracles to lattices of dimension at most 75 (sublattices of much larger lattices). Small experiments only require enumeration while larger dimensional lattices will probably

enjoy more efficiency from a sieving algorithm. Likely, a combination of the two will be considered in an actual cryptanalytic attack. For more work on sieving algorithms we direct the reader to [58, 75, 44, 16, 45].

Given a basis of the lattice $\mathfrak{L}$ and specified radius r, an enumeration algorithm outputs $\mathfrak{L} \cap D_r$. Enumeration algorithms consider calculations on the GSO basis rather than the originally entered basis. Since $\|\pi_k(\mathfrak{v})\| \leq \|\mathfrak{v}\|$ for all k, enumeration finds partial coefficients by finding all $\mathfrak{a} \in \mathfrak{L}$ such that $\|\pi_d(\mathfrak{a})\| \leq r$. Then from those vectors, the vectors $\mathfrak{a}$ which satisfy $\|\pi_{d-1}(\mathfrak{a})\| \leq r$ are found. This process of finding $\mathfrak{a}$ satisfying $\|\pi_{d-k+1}(\mathfrak{a})\| \leq r$ continues for ascending $k \in \{3, 4, \ldots, d\}$. The result is the desired intersection $\mathfrak{L} \cap D_r$.

Improvements to enumeration have been well studied. A depth-first enumeration tree is formed with vectors in $\pi_{d-k+1}(\mathfrak{L}) \cap D_r$ as nodes of depth k. Several quantum tree algorithms are available [11, 8, 56] allowing for faster enumeration with the aid of a quantum computer. Of course, exhausting the entire tree for the intersection $\mathfrak{L} \cap D_r$ is expensive even with the quantum speed ups. Enumeration focused on a smaller set of lattice points by a process called pruning will provide additional gains in efficiency. A pruning set $\mathfrak{P}$ is chosen and pruned enumeration outputs $\mathfrak{L} \cap D_r \cap \mathfrak{P}$. This will be faster than enumerating all of $\mathfrak{L} \cap D_r$ for an appropriately chosen pruning set. Of course, limiting enumeration with pruning may exclude the desired solution from the pruning set. A thorough process for choosing parameters guaranteeing a solution for $\mathfrak{L} \cap D_r \cap \mathfrak{P}$ with a high probability can be found in [32].

Most often when discussing pruned enumeration, the process of cylindric pruning is what authors are referring to. Through cylindric pruning, $P \cap \mathfrak{L} \cap D_r$ is formed by requiring different radii at different nodes. Not much differs from typical enumeration besides restricting intermediate steps to smaller radii. That is, the nodes at depth $n+1-k$ are the vectors in $\pi_k(\mathfrak{L}) \cap D_{r_k}$ for $r_k \leq r$ chosen before execution. Discrete pruning [10] provides a clever alternative that can be more simple and more parallelizable than

its cylindric counterpart. Performance of both methods of pruning will need to be further studied in order to definitively say one is more advantageous than the other. The pruning set is constructed through the use of tags $t \in \mathbb{N}^n$ and cells $C(t) \in \mathbb{R}^n$. Cells are chosen with the requirements that $\mathbb{R}^n = \cup_{t \in T} C(t)$ where T is a countable set, $C(t) \cap C(t') = \emptyset$ when $t \neq t'$, and $\mathfrak{L} \cap C(t)$ contains exactly one point in $\mathfrak{L}$ that can be easily computed. The natural partition, Fig. 2, provides cells and tags with these requirements.

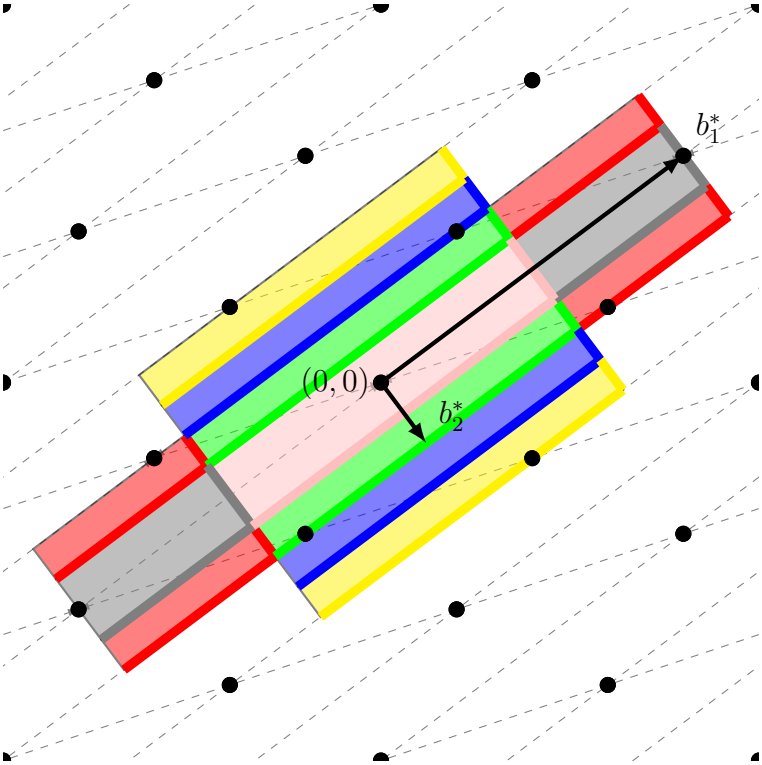

Figure 2: Natural partition $C(t) = \{\sum_{i=1}^{n} x_i \mathbf{b}_i^* | -(t_i+1)/2 < x_i \leq -t_i/2$ **or** $t_i/2 < x_i \leq (t_i+1)/2\}$ **with** $\mathbf{b}_1 = (4,3), \mathbf{b}_2 = (3,1)$**. The tags are** $(0,0)$**,** $(0,1)$**,** $(0,2)$**,**$(0,3)$**,** $(1,0)$**,** $(1,1)$**.**

1.2.1 Lattice-Based Cryptography

The most pupular public-key encryption schemes used today [63, 54] are vulnerable to algorithms performed on a quantum computer [66]. Lattice-based cryptography provides an attractive alternative believed to be secure against an adversary with

such capabilities. Believing in security may require some trust by the reader. As of today, an adversary with a access to quantum algorithms does not hold a significant advantage over classical algorithms when solving certain lattice problems. That is not to say classical and quantum algorithms are equally efficient. In fact, it has been shown quantum algorithms are generally better [45]. One can rest assured these algorithms, such as the quantum sieve and enumeration using a quantum tree searching algorithm, are still exponential (or worse) in the dimension of the lattice [45, 11, 58].

The attraction not only lies in the quantum security, but the hardness guarantees, as well. Cryptographic constructions with security requirements reducing to certain lattice problems enjoy worst case hardness security guarantees. This means breaking the cryptographic construction implies an efficient algorithm for solving any instance of some underlying lattice problem [53]. Ajtai, perhaps, gave birth to lattice-based cryptography in 1996 with his result in [2] showing the Short Integer Solution (SIS) problem is at least as hard as approximating several worst case lattice problems. That is, if there exists an oracle solving a random instantiation of SIS there exists a polynomial reduction to an oracle solving any instance of the underlying lattice problem. Problems such as LWE [62] and its ring variant [50] soon followed SIS with classical [61] and quantum reductions to such lattice problems. We return to Regev's LWE problem and the ring variant in section 1.4.

1.3 REDUCTION ALGORITHMS

A lattice reduction algorithm takes as input a lattice basis, and outputs a shorter, more orthogonal basis. Reduction algorithms are no stranger to cryptanalytic applications. LLL [46] is perhaps the most famous reduction algorithm and it was designed by Lenstra, Lenstra, and Lovász in 1982 with the intention of factoring polynomials. In 1996, coincidently the same year Ajtai published his famous result, Coppersmith

published an attack on RSA implementations with small exponent values [24].

The quality of a lattice basis is considered "good" if the basis consists of short, nearly orthogonal vectors. Suppose $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ is the current basis of $\mathfrak{L}$. Quantitatively, one measure of quality is the so-called Hermite factor $\mathrm{HF}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \|\mathbf{b}_1\|/\mathrm{Vol}^{1/n}(\mathfrak{L})$. Lattice reduction algorithms output reduced lattice bases with $\mathrm{HF}(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n) = \delta^n$. The Root Hermite Factor, δ, is a function of the input parameter to the reduction algorithm. An example of an input parameter would be the block size used in the Block-Korkine-Zolotarev (BKZ) [65] algorithm.

The best lattice reduction algorithms used today are improvements and modifications to Schnorr and Euchner's BKZ algorithm [65, 22, 37]. During a BKZ tour, an SVP oracle is called on the lattice of basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_\beta$ for some specified β, and $\mathbf{b}_1$ is updated to the result of the oracle. An SVP oracle is then called on the projected lattice spanned by the basis $\pi_2(\mathbf{b}_2), \pi_2(\mathbf{b}_3), \ldots, \pi_2(\mathbf{b}_{\beta+1})$; $\mathbf{b}_2$ is updated, and so on. Considering all the sublattices generated by $\pi_k(\mathbf{b}_k), \ldots, \pi_k(\mathbf{b}_{k+\ell-1})$ where $\ell = \min(\beta, d-k+1)$ while updating accordingly completes the BKZ tour. Denote the projected lattice of $\mathfrak{L}$ with basis $\pi_i(\mathbf{b}_i), \pi_i(\mathbf{b}_{i+1}), ..., \pi_i(\mathbf{b}_j)$ as $\mathfrak{L}[i:j]$. A basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ of $\mathfrak{L}$ is BKZ-β reduced for blocksize $\beta \geq 2$ if it is size-reduced and satisfies:

$$\|\mathbf{b}_i^*\| = \lambda_1(\mathfrak{L}_{[i,\min(i+\beta-1,n)]}),\ \forall i \leq n.$$

The work [21] shows that a BKZ-β reduced basis, $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$, satisfies $\|\mathbf{b}_1\| = \delta^n \mathrm{Vol}(\mathfrak{L}))$ where

$$\delta(\beta) \approx \left(\frac{\beta}{2\pi e} \cdot (\pi\beta)^{1/\beta}\right)^{\frac{1}{2(\beta-1)}}.$$

Notice the BKZ algorithm limits input of the SVP oracle to smaller dimensional projected lattices instead of $\mathfrak{L}$ in its entirety. Calls to the SVP oracle dominates overall cost of BKZ. Therefore, the complexity of BKZ is determined by the complexity of the SVP oracle being used.

Another useful heuristic is the so-called *Geometric Series Assumption* (GSA) in-

troduced in [64], which states that the Gram-Schmidt norms $\{\|\mathbf{b}_i^*\|\}_{i\leq n}$ of a BKZ-reduced basis behave as a geometric series, i.e., there is a constant $r > 1$ such that $\|\mathbf{b}_i^*\|/\|\mathbf{b}_{i+1}^*\| \approx r$ for all $i < n$. The root Hermite Factor suggests $\|\mathbf{b}_1\| \approx \delta_0^n \mathrm{Vol}^{1/n}(\mathfrak{L})$. Since $\mathrm{Vol}(\mathfrak{L}) = \prod_{i=1}^n \|\mathbf{b}_i\|$, $r = \delta_0^{-2n/(n-1)} \approx \delta_0^{-2}$ in the GSA assumption.

1.4 LEARNING WITH ERRORS

LWE allows for secure, efficient, and flexible public-key cryptosystems [19, 18, 7]. With all the attention, proper analysis of this problem's difficulty is imperative, especially if we expect to see these schemes used in industy. The learning with errors problem (LWE) [62] consists of finding a solution to linear equations that have been slightly altered with noise. Each "noisy" linear equation is called a sample of LWE. Noise is selected from a discrete Gaussian distribution, χ, with standard deviation σ. We assume χ has some finite support, i.e., χ only contains values in $\mathbb{Z}_q = \{\lceil -q/2 \rceil, \ldots, \lfloor (q-1)/2 \rfloor\}$. Values $x \in \chi$ are returned with probability proportional to $\exp(-\|x\|/2\sigma^2)$. For convenience, we present the LWE problem in its matrix form $(\mathbf{A}, \mathbf{b} \equiv \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$ where $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \in \chi^n$, and $\mathbf{e} \in \chi^m$. In search LWE (sLWE), $(\mathbf{A}, \mathbf{b})$ is given and the task is to recover $\mathbf{s}$. The task of decision LWE (dLWE) is to distinguish LWE samples $(\mathbf{A}, \mathbf{b})$ from a random distribution with some non negligible advantage. An oracle that solves sLWE trivially reduces to one that also solves dLWE. Astoundingly, an oracle that solves dLWE also provides a solution for sLWE. Regev gives a proof in [62] of this fact. Most cryptographic implementations consider LWE in its normal form. This means entries of $\mathbf{s}$ are selected from the same discrete Gaussian as the error entries as opposed to the entries being selected uniformly at random from $\mathbb{Z}_q$. Throughout this thesis, we will assume LWE refers to normal form LWE.

Definition 1.4.1 (Search $\mathrm{LWE}_{m,n,q,\chi}$)**.** With input parameters $n \geq 1$, modulus $q \geq 2$ and distribution χ, the search version of $\mathrm{LWE}_{m,n,q,\chi}$ problem consists of m samples

of the form $(\mathbf{a}, \mathbf{b}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, with $\mathbf{a} \hookleftarrow U(\mathbb{Z}_q^n)$, $\mathbf{s} \hookleftarrow \chi^n$, $\mathbf{b} = \langle \mathbf{a}, \mathbf{s} \rangle + e \pmod{q}$, and $e \hookleftarrow \chi$. Typically $m \geq n$. We say that an algorithm solves the search $\mathrm{LWE}_{n,q,\chi}^m$ if it outputs $\mathbf{s}$ with probability $\mathrm{poly}(1/(n \log q))$ in time $\mathrm{poly}(n \log q)$.

1.4.1 Learning with Errors over Rings

To save on the cost of key sizes and computation, often the security requirements of previously mentioned cryptographic schemes reduce to a ring-based variant of LWE (RingLWE). RingLWE was introduced in 2010 and proven to enjoy similar hardness guarantees as LWE by Lyubashevsky, Peikert, and Regev [50]. Additional algebraic structure raises some concern over the security against quantum adversaries. However, it should be noted that no significant progress has been made on a quantum algorithm that solves RingLWE with more success than LWE. The same methods for solving LWE are able to be applied to RingLWE. Denote $\Phi_n(x)$ as the *cyclotomic polynomial* whose roots are the primitive n^{th} roots of unity. $\Phi_n(x)$ is an irreducible monic polynomial in $\mathbb{Z}[x]$ of degree $\phi(n)$. The previous authors released a companion paper explaining in detail the benefits of using a cyclotomic ring, $R = \mathbb{Z}_q[x]/\Phi_n(x)$ [51]. Multiplication and inversion can be performed in $O(n \log n)$ scalar operations in cyclotomic number fields. When $n = 2^k$, $\phi_{2^k}(x) = x^{2^{k-1}} + 1$. For simplicity, we will assume that the ring R is always $\mathbb{Z}_q[x]/\Phi_n(x)$ where n is a power of 2. That is, R is an $n/2$ dimensional polynomial ring where elements have coefficients from $\mathbb{Z}_q$.

In the search version of RingLWE, $(a(x), b(x) = a(x)s(x) + e(x) + (\Phi_n(x)) \bmod q)$ is given where the coefficients of $e(x)$ and $s(x)$ are sampled independently from χ, and $a(x) \in R$. The task is to recover $s(x)$. Exactly n LWE instances can be constructed from a RingLWE sample $(a(x), b(x))$. Given $(a(x), b(x))$, an LWE in matrix representation $(\mathbf{A}, \mathbf{b})$ is derived where $A \in \mathbb{Z}_q^{n \times n}$, $\mathbf{b} \in \mathbb{Z}_q^n$. The operation of polynomial multiplication by a polynomial in R can be represented by matrix multiplication of coefficients. Suppose $a(x) = a_{n/2-1}x^{n/2-1} + \cdots + a_1 x + a_0$, $b(x) = b_{n/2-1}x^{n/2-1} + \cdots + b_1 x + b_0$,

and $s(x) = s_{n/2-1}x^{n/2-1} + \cdots + s_1 x + s_0$. Let $\mathbf{a}, \mathbf{b}, \mathbf{s}$ be the vectors with the coefficients of $a(x)$, $b(x)$, and $s(x)$, respectfully, as entries. The matrix

$$\mathbf{A} = \begin{bmatrix} a_0 & -a_{(n/2)-1} & -a_{(n/2)-2} & \dots & -a_1 \\ a_1 & a_0 & -a_{(n/2)-1} & \dots & -a_2 \\ a_2 & a_1 & a_0 & \dots & -a_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{(n/2)-1} & a_{(n/2)-2} & a_{(n/2)-3} & \dots & a_0 \end{bmatrix}$$

provides the LWE sample $(\mathbf{A}, \mathbf{b})$. All the algorithms and strategies discussed for solving LWE also can also be used for solving RingLWE.

1.5 QUANTUM CIRCUITS

Unlike their classical counterparts, quantum gates are restricted to those able to be specified as *unitary* matrices. A unitary matrix, U, has the property $U^\dagger U = I$ where $U^\dagger$ denotes the complex conjugate transpose of U. The inverse of a unitary matrix is again unitary so quantum gates are inherently reversible. Many classical gates, such as the NAND gate, are irreversible and unable to be simulated directly by a quantum computer. Since every quantum gate is reversible, the uncomputing circuits can be constructed from the original forward gates. Two simple reversible gates are presented in Fig 3. As amazing as it may seem, the Toffoli gate allows for any classical gate to be replaced by an equivalent quantum circuit. The Toffoli gate takes three qubits as input; two control qubits and one other qubit. If both control qubits are high, a NOT gate is applied to the other qubit. The circuit denoted $\pm$ takes three qubits as input with one qubit initially zero. If the nonzero qubits are the same, the zero qubit will remain zero. Otherwise, the zero qubit will be set to $|1\rangle$. Fredkin gates take three qubits as input. Two of the qubit values are swapped if the control qubit is high. The swapping of two qubits can be done with three CNOT gates. For more introductory information concerning quantum gates and circuits we refer the reader

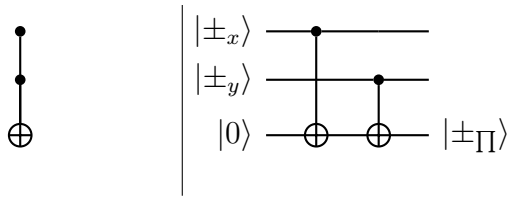

Figure 3: Toffoli gate (left) and sign gate (right) denoted $\pm$.

$$T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}$$

Figure 4: T-gate

to [60].

Resources for quantum computers are expensive. A thorough study of the resources required by a specific circuit helps us understand the practicality of implementation on an actual quantum computer. T-count is typically used to measure the resources due to the fact T-gates are complicated to implement when compared to other single qubit operators [9, 30]. The T-gate is presented in Figure 4. Toffoli and Fredkin (CSWAP) gates can be constructed using 7 T-gates [9]. We opt to stay at the higher level and do not consider possible savings at a lower lever. We also include another measurement of the efficiency of a quantum circuit by considering the number of qubits required to perform the computation. More complicated algorithms can be simulated using quantum circuits with the implemented arithmetic components. We give a novel circuit for computing the dot product of two vectors in $\mathbb{R}^k$ in Chapter 4. Figure 5 displays the gates used in Chapter 4.

Qubit registers representing integers in binary representation have qubits ordered by ascending significance. A nonnegative integer $x \in \mathbb{Z}$ less than 2^n can be held in an n-qubit register denoted $|x\rangle = |x_0 x_1, ..., x_{n-1}\rangle$ where $|x_0\rangle$ is the least significant qubit of the binary representation. Recall the register $|x_i x_{i+1} \dots x_{j-1}\rangle$ is denoted as $|x_{[i:j]}\rangle$ and the most significant qubit determines the sign of a value in 2's comple-

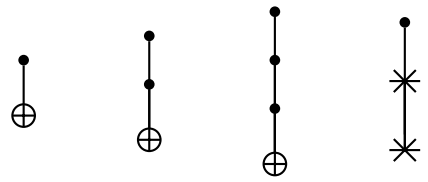

Figure 5: Elementary gates from left to right: CNOT, CCNOT (Toffoli), CCCNOT, CSWAP (Fredkin)

ment representation. We implement many circuits, including those for addition and multiplication, in the open-source software framework for quantum computing, ProjectQ [68, 36]. The software allows us to produce accurate resource estimations of the implemented circuits. Software implementations also allow us to test for correctness before taking resource counts. Instead of outputting T-count, the resource counting engine of ProjectQ outputs the specific gates used in the circuits.

Some superfluous output of a circuit, such as the carry bit in our circuit for 2's complement, does not serve any useful purpose at the end of computation. We denote these qubits as $|g\rangle$. Many authors refer to them as garbage qubits. Garbage qubits are still able to be used in computation, but forfeit holding any meaningful value unless uncomputed.

CHAPTER 2

ANALYSIS OF PRIMAL ATTACK

2.1 LWE TO USVP

In this chapter we consider the block size needed by the BKZ algorithm to solve certain instances of learning with erros. LWE can be considered as an average version of the Bounded Distance Decoding problem (BDD). A dual problem of BDD is the previously discussed Unique Shortest Vector Problem ($uSVP$).

Definition 2.1.1 (Bounded Distance Decoding: BDD_α)**.** Let $0 < \alpha < \frac{1}{2}$. Given a basis $\mathbf{B}$ of the lattice $\mathfrak{L}$ and a vector $\mathbf{t}$ such that $\text{dist}(\mathbf{t}, \mathfrak{L}) \leq \alpha \cdot \lambda_1(\mathfrak{L})$, find a lattice vector $\mathbf{x} \in \mathfrak{L}$ closest to $\mathbf{t}$. We will denote the α as the *gap* of the BDD_α problem.

Kannan's embedding [41] will be used to solve BDD in this analysis. Other methods of solving BDD are Babai's nearest plane algorithm [12] or Lindner-Peikert's randomized nearest plane algorithm [47]. These algorithms have been further investigated by Liu and Nguyen [48] who show they can be considered as cases of pruned enumeration algorithms.

We assume $\mathbf{B}$ has rows of basis elements. Given a BDD instance $(\mathbf{B}, \mathbf{t})$ where $\mathfrak{L}$ has rank d and $\mathbf{e}$ is the "shift", we consider the following basis matrix.

$$\mathbf{B}' = \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{t} & 1 \end{bmatrix}$$

This is a lattice of rank $d+1$ and volume $\text{Vol}(\mathfrak{L})$. Since $\mathbf{x} \in \mathfrak{L}$ there exists $\mathbf{y} \in \mathbb{Z}^d$ such that $\mathbf{B}^T * \mathbf{y} = \mathbf{x}$. The lattice generated by the rows of $\mathbf{B}'$ contains a short vector

related to the potential solution of the BDD problem.

$$\begin{bmatrix} \mathbf{B^T} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ -1 \end{bmatrix} = \begin{bmatrix} \mathbf{e} \\ -1 \end{bmatrix}$$

Lyubashevsky and Micciancio [49] provide a reduction, which can reduce any $BDD_{1/\gamma}$ instance $(\mathbf{B}, \mathbf{t})$ to an $uSVP_{\gamma/2}$ instance with basis

$$\mathbf{B}' = \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{t} & \mu \end{bmatrix}$$

with μ set to be the distance $\text{dist}(\mathbf{t}, \mathfrak{L}) \leq \lambda_1(\mathfrak{L})/(2\gamma)$, where $\mathfrak{L}$ is the lattice spanned by $\mathbf{B}$. In more detail, if $\mathbf{x}$ denotes a closest vector to $\mathbf{t}$ in $\mathfrak{L}$ then it is shown that the vector $\mathbf{s}' = ((\mathbf{x} - \mathbf{t}), -\text{dist}(\mathbf{t}, \mathfrak{L}))$ is a shortest non-zero vector of the lattice generated by the basis $\mathbf{B}'$.

Later, Bai et al. [14] propose to preprocess the lattice $\mathfrak{L}$ using Khot's sparsification technique [42] before resorting to Kannan's embedding: the component μ is decreased to be $O(d/n)$, and the losing factor in the reduction is improved from 2 to $\sqrt{2}$. In practice [5, 7, 6], one usually sets $\mu = 1$. Bai, Miller, and Wen [13] show that the gap in the $uSVP$ instance is somewhat close to the upper-bound γ in practice, even though this is not guaranteed in the worst-case. With $\mu = 1$, it seems that $BDD_{1/\gamma}$ already reduces to $uSVP_{0.9\gamma}$ compared to $uSVP_{0.7\gamma}$ with $\mu = \|\mathbf{e}\|$. This also explains why it is preferable to use $\mu = 1$ in practice.

2.2 PRIMAL ATTACK ON LIMITED SAMPLES

Given the matrix LWE instance $(\mathbf{A}, \mathbf{b} \equiv \mathbf{As} + \mathbf{e} \pmod{q})$, the lattice $\mathfrak{L} = \{\mathbf{x} \in \mathbb{Z}^{m+n+1} \mid (\mathbf{A} \mid \mathbf{I}_m \mid \mathbf{b}) \cdot \mathbf{x} \equiv \mathbf{0} \bmod q\}$ can be constructed. This is a lattice of rank $d = m + n + 1$ and volume q^m. An adversary typically only has access to n samples of LWE when considering several public-key encryption proposals [7, 18]. With a restriction that $m \leq n$, it is almost always optimal to choose $m = n$. One can check

numerically using Equation (3.3). The vector $\mathfrak{v} = (\mathbf{s}^T|\mathbf{e}^T|1) \in \mathfrak{L}$ will be considerably shorter than the rest of the lattice vectors if $\mathbf{s}$ and $\mathbf{e}$ have normally distributed entries. We expect the length of $\|\mathfrak{v}\| \approx \sigma\sqrt{2n}$ when the standard deviation of the distribution is σ. Actual values $\|\mathfrak{v}\|$ will be slightly larger due to the appended 1. As we shall see later in the section, larger length should only make the the value $\mathfrak{v}$ more difficult to find. Projections of $\mathfrak{v}$ should have length $\|\pi_k(\mathfrak{v})\| \approx \sigma\sqrt{d-k+1}$ since a projection of a random vector $\mathbf{b} \in \mathbb{R}^d$ over a basis of size k should be of length approximately $\sqrt{k/d}\|\mathbf{b}\|$. The lattice $\mathfrak{L}$ of rank $d = m + n + 1$ is constructed from a matrix LWE instance $(\mathbf{A}, \mathbf{b} \equiv \mathbf{As} + \mathbf{e} \pmod q)$ as follows:

$$\mathfrak{L} = \begin{bmatrix} \mathbf{I}_n & \mathbf{A}^T & 0 \\ 0_{m\times n} & q\mathbf{I}_m & \vdots \\ 0^{\otimes n} & \mathbf{b} & 1 \end{bmatrix}$$

In the New Hope key exchange paper [7], a method of estimating the cost for solving LWE is given. Their method considers the evolution of the Gram-Schmidt coefficients of the unique shortest vector, i.e., the values $\pi_k(\mathfrak{v})$, within the BKZ tours. More precisely, it compares the expected length of the projected shortest vector, $\mathfrak{v}$, with the Gram-Schmidt lengths estimated by the GSA assumption. The key observation is that partial information of shortest vector will be recovered in the last block, when the orthogonal projection $\pi_{d-\beta+1}(\mathfrak{v})$ is shorter than $\mathbf{b}^*_{d-\beta+1}$. Once the SVP oracle reaches the last sublattice of dimension β, $\mathfrak{L}[d-\beta+1, d]$, the projection $\pi_{d-\beta+1}(\mathfrak{v})$ is recovered since $\|\pi_{d-b+1}(\mathfrak{v})\| \leq \|b^*_{d-\beta+1}\|$. Recovering the projection $\pi_{d-\beta+1}(\mathfrak{v})$ reveals β coefficients of $\mathfrak{v}$. GSA predicts $\|\mathbf{b}^*_i\| \approx \delta_0^{d-2i+2}\mathrm{Vol}(\mathfrak{L})^{1/d}$, so given $\mathfrak{L}$ above, $\|\mathbf{b}^*_{d-\beta+1}\|$ is assumed equal to $\delta_0^{2\beta-d}q^{m/n+m+1}$. Thus the success condition for recovering $\mathfrak{v}$ can be formulated as follows.

$$\sqrt{\beta}\sigma \leq \delta^{2\beta-(n+m+1)}q^{(m)/m+n+1} \tag{2.1}$$

where δ depends on β. Martin et al. [6] mentions $\mathfrak{v}$ is typically recovered from a size reduction algorithm on the tour the projection $\pi_{d-\beta+1}(\mathfrak{v})$ was found. Regardless if

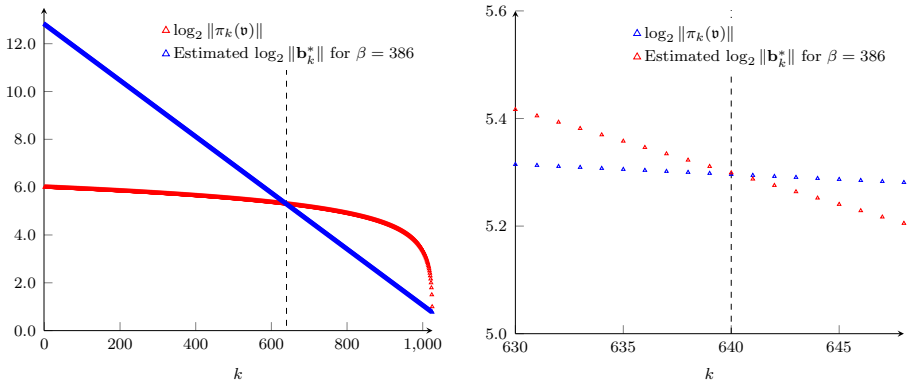

Figure 6: Comparison between $\log_2 \|\pi_k(v)\|$ and GSA for block size 386 using LWE parameters for NewHope-512 [18](April 10th 2020 update): $n = 512$, $\sigma = 2$, $q = 12289$. A vertical line is placed at $d - \beta + 1 = 640$.

a BKZ tour calls for a size-reduction algorithm, we can expect BKZ to recover all the coefficients of $\mathfrak{v}$ within $\lceil d/\beta \rceil$ tours. The next tour will call an SVP oracle on the preceding updated sublattice $\mathfrak{L}[d - 2\beta + 2, d - \beta + 1]$ of dimension β, in which $\pi_{d-2\beta+2}(\mathfrak{v})$ is likely to be the shortest vector. This tour of BKZ recovers β more coefficients. Repeating the process, BKZ recovers all the coefficients of $\mathfrak{v}$.

Figure 6 displays theoretical $\log_2$ lengths of $\pi_k(\mathfrak{v})$ and $\|\mathbf{b}_k^*\|$ for an actual candidate of the NIST Post-Quantum Standard known as NewHope [7] (Updated April 10, 2020). Using methods mentioned above, the LWE sample extracted from NewHope-512 can be reduced to a lattice of dimension $d = 2n + 1 = 1025$. The projection lengths are approximated by $\|\pi_k(\mathfrak{v})\| \approx \sigma\sqrt{d - k + 1}$ while $\|\mathbf{b}_k^*\|$ is approximated by GSA for block size $\beta = 386$. Since $\|\pi_{1025-386+1}(\mathfrak{v})\| \leq \|\mathbf{b}^*_{1025-386+1}\|$ calling an SVP oracle to the sublattice $\mathfrak{L}_{[640:1025]}$ of dimension 386 should return a projection of $\mathfrak{v}$. We can conclude that NewHope-512 would be vulnerable to a primal attack using BKZ-386. The resources required to launch such an attack are massive as discussed in Chapter 1.

2.3 BLOCK SIZE REQUIREMENTS TO SOLVE LWE

Often [6, 13] smaller block sizes are reported to solve LWE than those estimated by Equation (3.3). In this section, we consider experiments that explore the behavior of the GSO vectors and the projections $\pi_k(\mathfrak{v})$. We attempt to explain the success of BKZ for small block sizes using the same theory to arrive at (3.3). Specifically, we compare the lengths of the projections $\pi_k(\mathfrak{v})$ to $\|\mathbf{b}_k^*\|$. Before the lengths are considered, the lattice is reduced as much as possible without recovering $\mathfrak{v}$.

2.3.1 Caveats in Finding Actual $\pi_i(\mathfrak{v})$

Recording of such projections must be done with care. There are cases when $\pi_i(\mathfrak{v}) = b_i^*$ resulting in $\pi_j(\mathfrak{v}) = 0$ for $j > i$. As unavoidable as this phenomenon may be, we are still able to get a decent understanding of the $\|\pi_k(\mathfrak{v})\|$ for early values of k. When a projection $\pi_k(\mathfrak{v})$ is found for $k < d - \beta + 10$ the value is omitted from the calculation of average $\|\pi_k(\mathfrak{v})\|$ to circumvent the issue of an average value distorted by some $\pi_j(\mathfrak{v})$ for $j > k$. Let β be the block size necessary to solve the current instance. The value $d - \beta + 10$ is chosen to ensure an early projection is not considered close to $d - \beta_{2017} + 1$, i.e., the index where $\|\pi_k(\mathfrak{v})\|$ is estimated to be greater than or equal to $\|b_k^*\|$.

Early projections will almost always be found when k is close to d. An understanding of the average behavior of these projections is nearly impossible due to the inconsistency and the frequent value of zero. Unfortunately due to these inconsistencies, we limit our experiment and omit these last indices from the results. The last few indices of the GSO predictions for $\|\mathbf{b}_k^*\|$ are also highly inaccurate. A simulator [15] is used in the following chapter to better predict $\|\mathbf{b}_k^*\|$ for values of k close to d.

2.3.2 Experimental Results

We consider 500 lattices for four different parameter sets of LWE. Let α equal the quotient σ/q. For consistency and concreteness, we set q equal to the first prime greater than $((100-n)10+500)$ with parameters maintaining the relation

$$\log q/\log\alpha * \log(n\log q/\log^2\alpha) = c$$

for some constant c as in [13]. This roughly means the running-time for solving LWE is asymptotically single-exponential in n. We opt for a smaller modulus for smaller dimensional lattices to circumvent using larger precision floating-point arithmetic. Parameter choices are displayed in table 1. All lattice reduction implementations use the fpylll [28, 27] software package. A progressive BKZ algorithm takes a lattice basis as input and makes use of pruned enumeration. The block size is initially set to 20 and incremented by one after 20 tours without recovering $\mathfrak{v}$. Let β be the block size of the current tour of the progressive BKZ algorithm. We make use of the preprocessing strategy available in the fpylll implementation of BKZ. During preprocessing, for β' significantly smaller than β, BKZ-β' is called on a β dimensional sublattice before enumeration. Strategies such as preprocessing and enumeration allow for faster lattice reduction without impacting our analysis. Basis vectors are recorded before each tour of BKZ allowing us to analyze the behavior of $\|b_i^*\|$ right before the algorithm recovers $\mathfrak{v}$. From the the basis before recovery, GSO lengths, $\|b_i^*\|$, are recorded and the average of the 500 instances is calculated for each i. We record the block size used to successfully recover $\mathfrak{v}$ and compare to the theoretical block size requirement by Equation (3.3). Theoretical required block size is computed numerically using (3.3). The left side of the equation is replaced by $\sqrt{\beta/d}\|\mathfrak{v}\|$ since we actually know the size of $\mathfrak{v}$.

As noted, when sampling $\mathfrak{v}$ the vector should be slightly longer than the expected $\sigma\sqrt{2n}$ because of the explicit appended 1. A longer vector should only make the

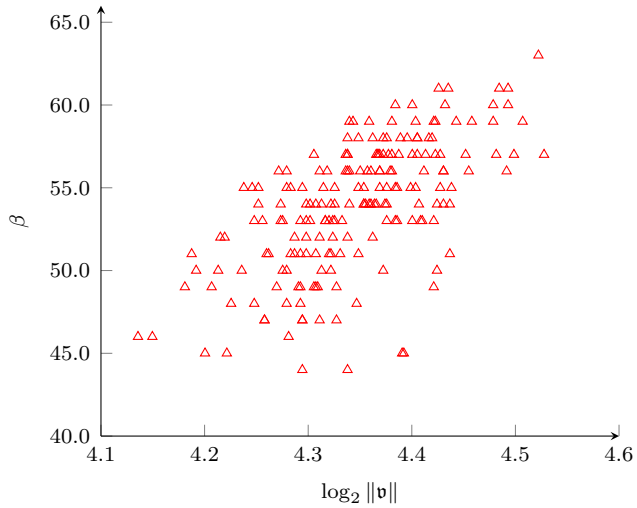

Figure 7: LWE parameters $n = 95$, $d = 191$, $\sigma = 1.45$, $q = 557$. Horizontal axis is $\|\mathfrak{v}\|$ for the $\mathfrak{v}$ sampled during the expermint. Vertical access represents the experimental block size, β, used to recover $\mathfrak{v}$.

problem more difficult, i.e., a larger block size should be necessary for longer $\mathfrak{v}$. Figure 7 displays the correlation between the length of $\mathfrak{v}$ and the block size necessary to recover the vector. To offer more transparency within the experiment, we note the average length of $\log_2 \|\mathfrak{v}\|$ for the instances used. Given the LWE parameters, β_{2017} is found numerically, as well, using Equation (3.3) exactly as it appears.

It should be of no surprise that the theoretical distribution of the block size counts are centered around the theoretical mean. Experimental block sizes seem to exhibit similar behavior (besides maybe when $n = 70$) with a significantly smaller center. The odd distribution of block sizes of the example when $n = 70$ is typical in a BKZ algorithm reducing lattices of small rank. For larger rank, the block sizes seem to consistently center around a certain value, as is the case when $n = 80, 90, 95$. We attempt to explain why smaller block sizes are effective using the same theory that developed Equation (3.3) by considering the Gram-Schmidt orthogonalized basis vectors and the projections of the short vector $\mathfrak{v}$. As explained early in the section, we consider these values on the tour before $\mathfrak{v}$ is recovered. This is a key difference

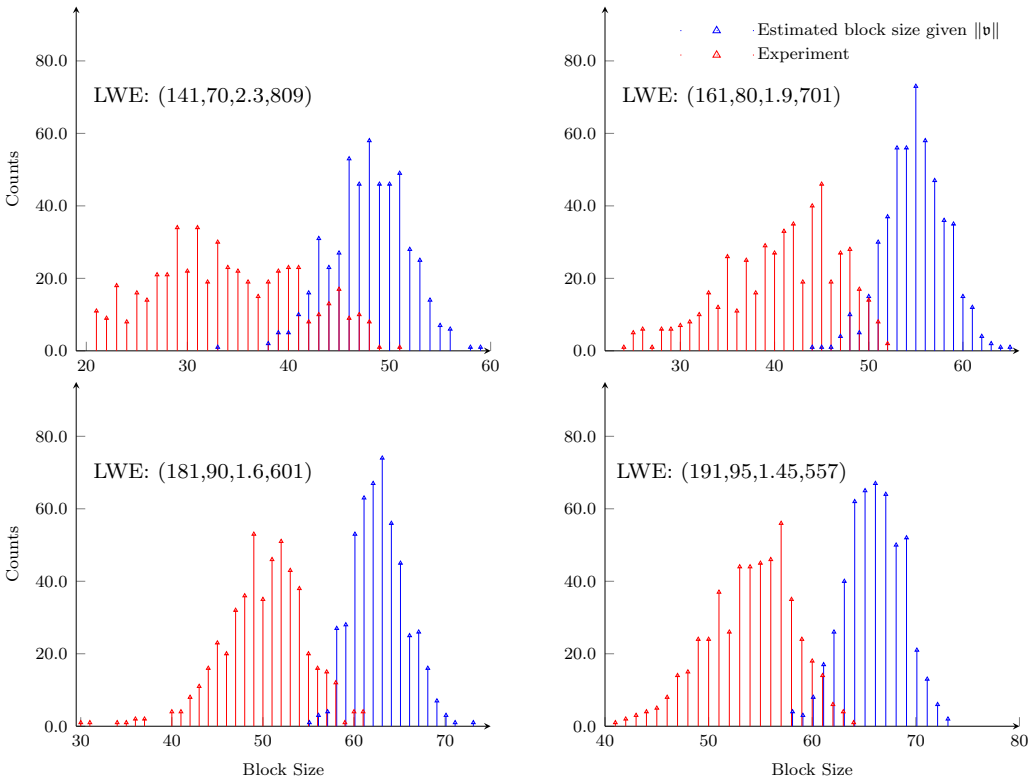

Figure 8: 500 LWE instances of each parameter set, denoted LWE:(d, n, σ, q) where d, n, σ, and q are the dimension of the lattice, dimension of LWE secret, standard deviation of distribution χ, and the prime modulus, respectively. In each instance, we record the block size required to recover υ and also the block size estimated by Equation (3.3). Counts of both theoretical and experimental block sizes are displayed.

d	n	σ	q	avg $\log_2 \|\mathfrak{v}\|$	β_{avg}	β_{2017}
191	95	1.45	557	4.347	54.07	65
181	90	1.6	601	4.442	50.132	62
161	80	1.9	701	4.590	40.88	55
141	70	2.3	809	4.775	33.914	48

Table 1: LWE parameters used in experiments with $c = 0.44$. The value β_{avg} is the average block size needed to solve 500 LWE instances of the specified parameters where β_{2017} denotes the block size estimated by Equation (3.3).

when comparing this study to those previous. As can be seen in Figure 9, a later intersection between the actual GSOs and the projections may indicate that a smaller block size is sufficient. The delayed intersection suggests a projection of the lattice will be recovered in the last block $\mathfrak{L}_{[d-\beta'+1,d]}$ using block size $\beta' < \beta$. Any intersection between the experimental projections and GSOs could prove interesting, however. The impact of finding early projections (at index close to d) was first explored in [6] and later explored in greater depth by [13]. Effects of these multiple intersections will be further reported in Section 3.3. Graphs in Figure 9 are the zoomed version of Figure 6 but for much smaller parameter sets.

The intersection between the projections and the GSOs happens sooner in the experiments rather than later. Results of the experiments seem to suggest that Equation (3.3) cannot be used to explain why these early projections are found. Another observation shows the second intersection occurs earlier than predicted in thoery. This results in early projection being found which could potentially alter the data collected by such experiments [6]. We explain how we circumvented these erroneous calculations in the following section.

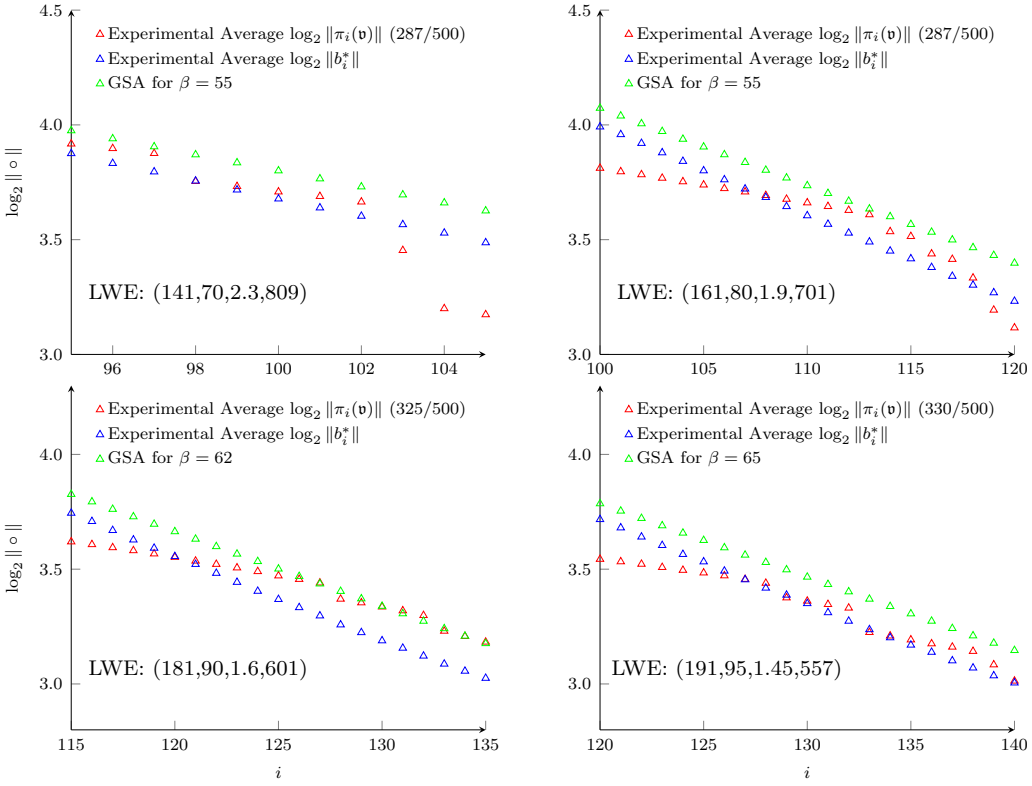

Figure 9: 500 LWE instances of each parameter set, denoted LWE:(d, n, σ, q) where d, n, σ, and q are the dimension of the lattice, dimension of LWE secret, standard deviation of distribution χ, and the prime modulus, respectively. The actual values for $\|\mathbf{b}_k^*\|$ and $\|\pi_k(\mathfrak{v})\|$ are recorded after running the described progressive BKZ algorithm.

CHAPTER 3
A REFINED ANALYSIS OF THE COST FOR SOLVING LWE VIA USVP

3.1 REVISITING THE COST OF SOLVING USVP

In this section, we first revisit the two approaches of [31, 7] for estimating the cost of solving uSVP and the analysis in [6]. Then we expand the comparison in [6] of the two estimates with a larger set of LWE parameters. Furthermore, we verify the accuracy of the second estimate on the smaller dimension regime, where the first estimate could lead to a smaller blocksize.

3.1.1 Two Estimates

Recall that we can view the LWE problem as a BDD problem. For simplicity, we will use the lattice $\mathfrak{L}_q(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m : \mathbf{y} \equiv \mathbf{A}\mathbf{x} \pmod q, \forall \mathbf{x} \in \mathbb{Z}^n\}$. The lattice $\mathfrak{L}_q(\mathbf{A})$ with the target point $\mathbf{t}$ defines a BDD instance: note this is a $\mathrm{BDD}_{1/\gamma}$ instance with $\gamma = \lambda_1(\mathfrak{L}_q(\mathbf{A}))/\|\mathbf{e}\|$. The lattice $\mathfrak{L}_q(\mathbf{A})$ has rank m and volume q^{m-n}. By Gaussian Heuristic, we have $\lambda_1(\mathfrak{L}_q(\mathbf{A})) \approx \sqrt{\frac{m}{2\pi e}}\, q^{(m-n)/m}$. On the other hand, the LWE error $\mathbf{e}$ has length about $\sqrt{m}\alpha q$. Thus we obtain a $\mathrm{BDD}_{1/\gamma}$ instance where

$$\gamma \approx \frac{\min\left(q, \sqrt{\frac{m}{2\pi e}}\, q^{(m-n)/m}\right)}{\sqrt{m}\alpha q}. \tag{3.1}$$

For convenience, we assume that q is not too small and hence $\gamma \approx q^{-n/m}/\alpha$.

We first recall the estimate for solving uSVP by Gama and Nguyen [31] (we will refer to it as the *first estimate* or the *2008 estimate*). First, one assumes that the above $\mathrm{BDD}_{1/\gamma}$ reduces to uSVP_γ, where $\gamma \approx q^{-n/m}/\alpha$. Then Gama and Nguyen [31]

show that the shortest vector in the uSVP_γ problem can be recovered as soon as $\gamma \geq \tau \cdot \delta^m$ where δ is root Hermite factor of the algorithm used. Here $\tau < 1$ is an empirical constant determined by experiments: it has been investigated that τ lies in between 0.3 and 0.4 when using the BKZ algorithm [5, 6]. For simplicity, we will omit the constant τ in the asymptotic analysis (but set it to be 0.3 in actual experiments). As noted in Equation (1.3), the $\delta(\beta)$ is a decreasing function of β and therefore we want to maximize δ. The optimal m is asymptotically $\frac{2n \log q}{\log(1/\alpha)}$ which leads to maximum $\delta \approx \alpha^{\log \alpha/(4n \log q)}$. The running time of BKZ-β is $2^{O(\beta)}$ using the core-SVP model. In terms of LWE parameters this is asymptotically

$$\exp\left(c_t \cdot \frac{n \log q}{\log^2 \alpha} \cdot \log\left(\frac{n \log q}{\log^2 \alpha}\right)\right) \tag{3.2}$$

for some constant c_t.

In the New Hope key exchange paper [7], another method for estimating the cost for solving LWE is given. We will refer to it as the *second estimate* or the *2016 estimate*. Instead of looking at the gap of the uSVP directly, it considers the evolution of the Gram-Schmidt coefficients of the unique shortest vector in the BKZ tours. More precisely, it compares the expected length of the projected (expected) shortest vector $\mathfrak{v} = (\mathbf{e}, -1)$ with the Gram-Schmidt lengths estimated by the GSA assumption. The key observation is that partial information of shortest vector $\mathfrak{v}$ will be recovered in the last block, when the orthogonal projection of $\mathfrak{v}$ to the first $d - \beta$ Gram-Schmidt vectors is shorter than the expected $\mathbf{b}^*_{d-\beta+1}$ predicated by the GSA assumption. Thus the success condition for recovering $(\mathbf{e}, -1)$ can be formulated as follows.

$$\sqrt{\beta}\alpha q \leq \delta^{2\beta - m} q^{(m-n)/m} \tag{3.3}$$

where δ depends on β. Here we simply take the rank of the lattice to be $m \approx d$.

These two estimates have been investigated extensively by Albrecht et al. in work [6]. They show that the lattice reduction experiments largely follow the behaviour expected from the second estimate. Furthermore, they also present a sound

analysis to show that, after the last β Gram-Schmidt coefficients of the shortest vector is recovered, a further size reduction is often sufficient to recover the complete secret immediately. In fact, this can happen at indices smaller than the $d - \beta + 1$. As noted in [6], they observe an interesting phenomenon that in several cases the lattice reduction even behaves better than the second estimate for some parameters: the BKZ algorithm recovers a projection $\pi_i(\mathfrak{v})$ at index following a distribution with a center smaller than $d - \beta + 1$. It is outlined in [6] that this may be caused by the occurrence of a second intersection of the projected vector with the Gram-Schmidt vectors.

3.1.2 Comparison of Estimates with Various (n, q, α)

In this subsection, we expand the comparison in [6] on the two estimates with a larger set of LWE parameters. Note that a numerical comparison of two estimates is already given in the work [6]. Here we expand the range of the LWE parameters to the single-exponential regime: observe that the comparison in the Figure 1 of [6] fixes q, α and increases n. This compares the two estimates for LWE parameters in the super-exponential regime because of the estimate in Equation (3.2). Here we assumed that the optimal m in the 2006 estimate is asymptotically the same as the 2008 estimate. Note that the 2008 estimate (e.g. Equation (3.3)) can be re-formulated as

$$\beta^{1/(2\beta)} \leq \left(\frac{q^{-n/m}}{\alpha}\right)^{1/m} \beta^{1/(2m)}.$$

This can be compared to the uSVP gap argument in the 2008 estimate [31] where we have $\beta^{1/(2\beta)} \leq (q^{-n/m}/\alpha)^{1/m}$ instead. We want to minimize the β in Equation (3.3). This is a constraint optimization problem which seems tedious. Instead we find the optimal m and β numerically. In setting the LWE parameters (n, q, α), we maintain

the relation that

$$\log q / \log^2 \alpha \cdot \log(n \log q / (\log^2 \alpha)) \tag{3.4}$$

being a constant c. Note that this corresponds to the multiplier in front of n in the Equation (3.2). This roughly means the running-time for solving LWE is asymptotically single-exponential.

We describe the parameters we used in the comparison. We set $c = 0.25$ and 0.35 respectively. For each c, we take $q = n^2$ and $q = n^4$ (thus four sets of parameters). Such parameters simulate commonly used conservative parameters (e.g. q not too large). Then we compute the corresponding α. For each set of parameters (n, q, α), we find the optimal m that leads to the smallest β using the 2016 estimate and the 2008 estimate (we set the empirical constant $\tau = 0.3$) respectively. We denote the smallest blocksize required from the two estimates as β_{2008} and β_{2016}. For each set of parameters, we plot the blocksize β required as an (increasing) function of n; we also plot the normalised blocksize difference which records $(\beta_{2008} - \beta_{2016})/\beta_{2008}$: this roughly illustrates the "improvement percentage". If this value is negative, we simply denote it by 0 but we will further consider these cases later. We plot the comparison on the four sets of parameters in Figures 10-13.

It can be observed that the impacts of (the difference of) the two methods increases with the decrement of q. Similarly, the difference of the two methods increases with the decrement of α. This also confirms the comparison of the two methods in [6] in the single-exponential region.

3.1.3 Smaller Dimension

Note that in the small dimension (in terms of LWE n) regime (some of which might be still relevant to practical schemes), the first estimate leads to a smaller blocksize. This is due to the empirical constant τ set to be 0.3. There might be a tendency to use the first estimate as it produces more conservative estimates. We further confirm

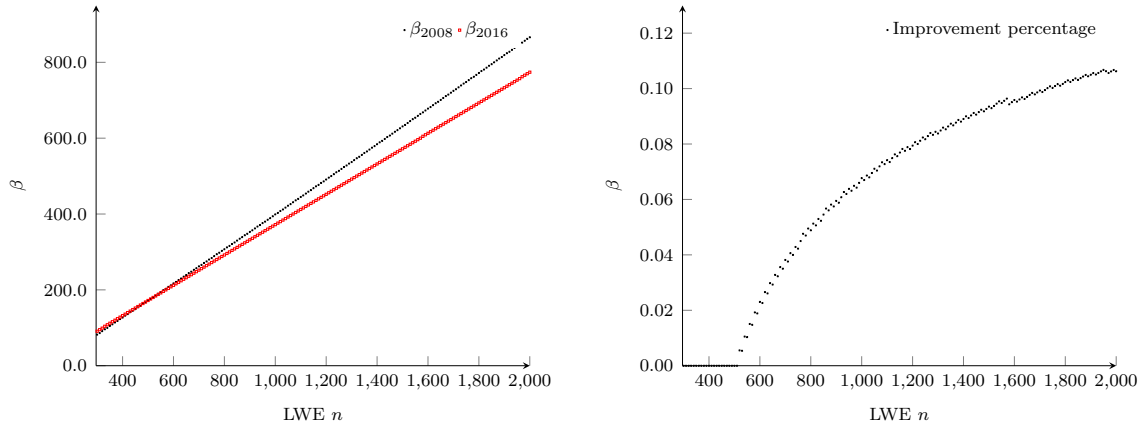

Figure 10: Comparison of blocksize β of two estimates when $c = 0.25$ and $q = n^2$. The right is the same as the left hand side, but compares the improvement percentage of the blocksize.

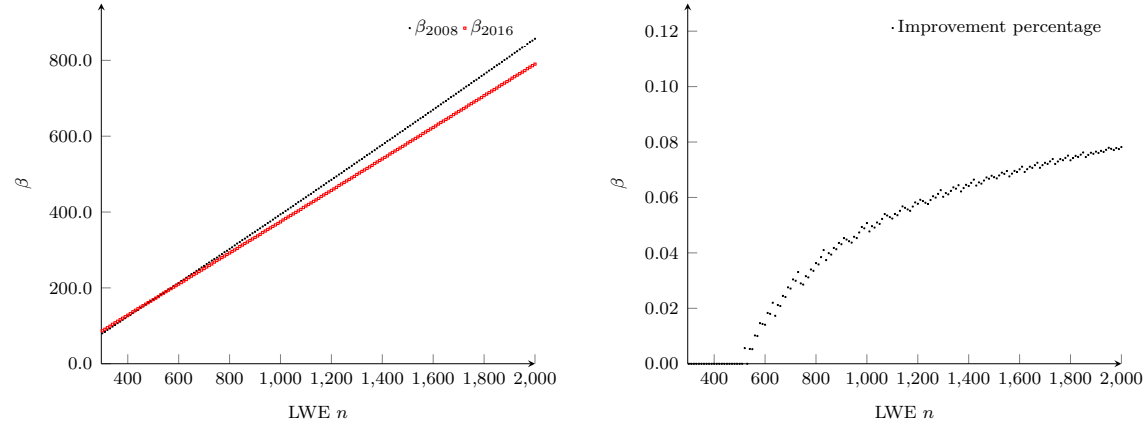

Figure 11: Comparison of blocksize β of two estimates when $c = 0.25$ and $q = n^4$. The right is the same as the left hand side, but compares the improvement percentage of the blocksize.

the accuracy of the second estimate for these smaller dimensions. Note that for tiny blocksizes (e.g. $\beta \leq 30$), it has been observed in [15] that the Gaussian heuristic in local blocks is not accurate in BKZ; nor such blocksize matter the running-time of BKZ too much. Thus we do not consider these tiny blocksizes. We choose parameters n, q, α such that the blocksizes are ≥ 40 and compare the two methods in such region. Using the same approach as the last subsection, we set $c = 0.5$ and $q = n^2$. Then we find the corresponding α for the error rate. For each (n, q, α), we find the optimal m that leads to the smallest β using the 2016 estimate and the 2008 estimate respectively.

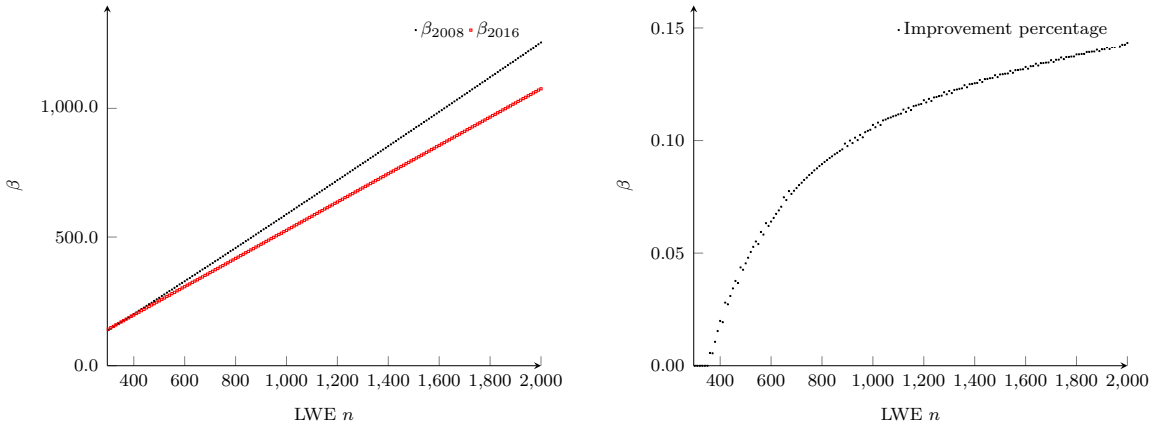

Figure 12: Comparison of blocksize β of two estimates when $c = 0.35$ and $q = n^2$. The right is the same as the left hand side, but compares the improvement percentage of the blocksize.

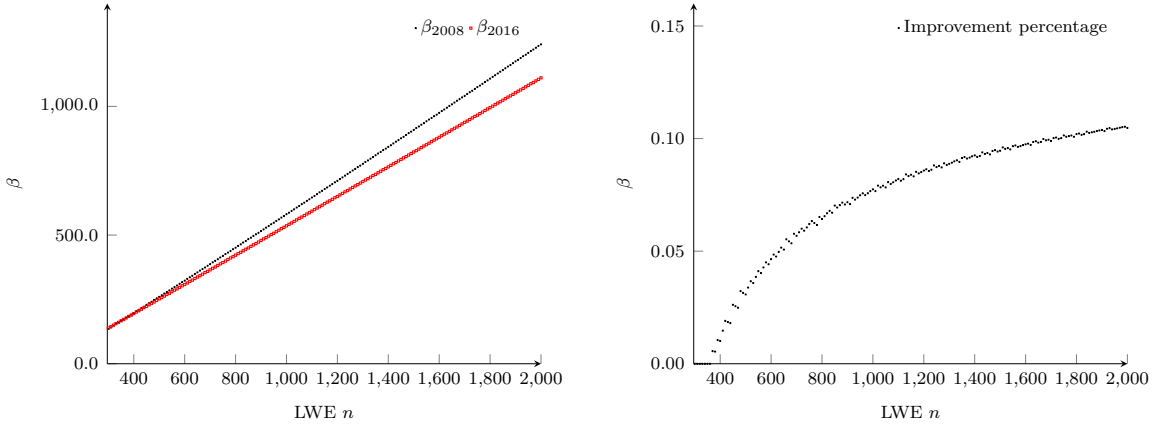

Figure 13: Comparison of blocksize β of two estimates when $c = 0.35$ and $q = n^4$. The right is the same as the left hand side, but compares the improvement percentage of the blocksize.

For the 2008 estimate, we set the empirical constant $\tau = 0.3$: approximately we are comparing the two estimates in terms of $\delta^d \approx q^{-n/m}/(0.3\alpha)$ with $\delta^m \approx q^{-n/m}/\alpha\sqrt{\beta}$.

In Figure 14 we can observe, for small LWE dimension n, the first estimate gives a smaller blocksize due to the empirical constant 0.3. Then we look at the concrete experiments with LWE parameters $n = 110$, $q = 12101$, $\sigma = \alpha q = 7.2$ of 100 instances. Using the 2008 estimate, the optimal $m = 277$ which leads to the $\beta = 39$. Using the 2016 estimate, the optimal $m = 294$ which leads to the $\beta = 66$. In Figure 2, the experiments using BKZ of various blocksize as well as different number of samples

are tabulated. It can be seen that the 2016 estimate indeed provides a more accurate estimate: all BKZ instances using $\beta = 66$ succeed with $m = 294$ as predicated by the 2016 estimate. We note that many instances even succeeded with smaller blocksize $\beta = 60$. This is perhaps due to the second intersection phenomenon as observed in [6]. We will look at this phenomenon later.

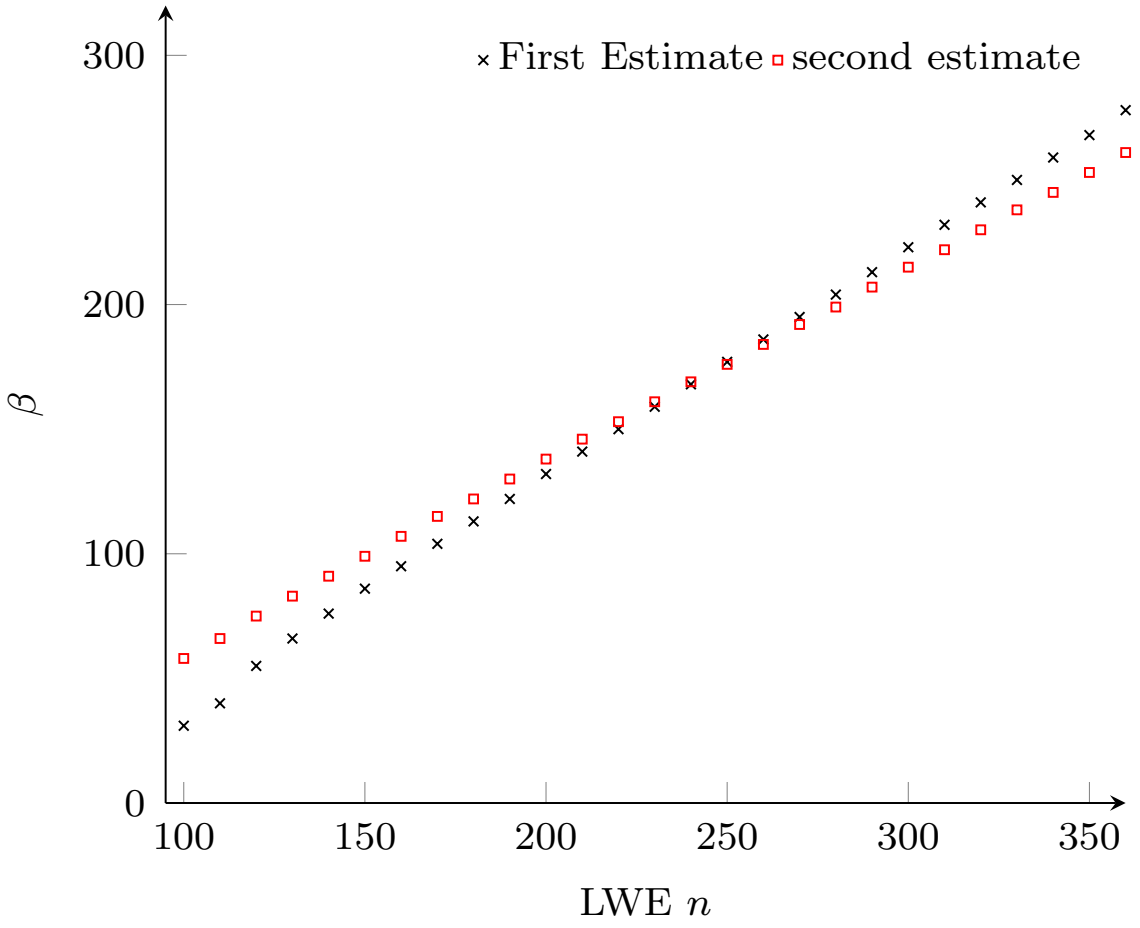

Figure 14: Comparison of blocksize β of two estimates when $c = 0.5$ and $q = n^2$ for small n region.

3.1.4 Further Experiments on the Projection Length

The success condition for recovering the shortest vector in Equation (3.3) depends mainly on two heuristics: first, the norm of the Gram-Schmidt vectors in a BKZ reduced basis follows from the GSA assumption; second, the norm of the projection

LWE parameters: $n = 110$, $q = 12101$, $\sigma = 7.21$			
2008 estimate	Sample m	Blocksize β	Succ. prob.
(smallest $\beta = 39$	277	40	0%
with $m = 277$)	277	50	0%
2016 estimate	optimal m	blocksize β	Succ. prob.
(smallest $\beta = 66$	294	50	0%
with $m = 294$)	294	60	52%
	294	66	100%

Table 2: Experimental comparison of two estimate for small n region.

of the shortest vector onto the vector space spanned by the last β Gram-Schmidt vector is about $\alpha q\sqrt{\beta}$.

In practice, it is known [31, 15] that the GSA assumption does not quite fit the BKZ experiments. However, the GSA assumption is optimistic from an attacker's point of view, which leads to a more conservative estimate. Hence we will assume this is the case. We will look at the second heuristic on the projection length. Denote the shortest vector to be $\mathfrak{v}$. The heuristic on the project length essentially requires that $\mathfrak{v}$, when expanded in terms of Gram-Schmidt vectors, have similar length on all components. This follows true if the Heuristic 2 described in work [32] is true: The distribution of the coordinates of the target vector $\mathfrak{v}$, when written in the normalized Gram-Schmidt basis $(\mathbf{b}_1^*/\|\mathbf{b}_1^*\|, \mathbf{b}_2^*/\|\mathbf{b}_2^*\|, \cdots, \mathbf{b}_m^*/\|\mathbf{b}_m^*\|)$ of the input basis, looks like a uniformly distributed vector of norm $\|\mathfrak{v}\|$. Observe that the heuristic depends on the shape of the input basis. For example, when the input basis is strongly reduced, the shortest vector $\mathfrak{v}$ may already appear in the basis and hence the heuristic will not be true.

An experimental study has been presented in Figure 2 of [6] using 16 LLL reduced bases. We conduct further experiments on the length of projected shortest vector on

BKZ reduced bases of various blocksizes. We use the same parameters as Figure 2 of [6]: we generate 200 LWE instances of $n = 65$, $m = 182$, $q = 521$ and $\sigma = 8/\sqrt{2\pi}$ (the results are averaged over these instances). We reduce the embedded bases using LLL and BKZ-β for $\beta = 10, 20, 30, 40, 45$. Note here we choose the largest blocksize to be 45 since this prevents the shortest vector from being recovered with high probability. Similarly, in the reduced bases, we do not consider those where the shortest vector has already been found. The experimental results are illustrated in Figure 15. It can be seen that the projection norms of the shortest vector indeed follow a similar shape in all LLL/BKZ-reduced bases. When the lattice is more reduced, the projected norm seems to follow more closely to the theoretical estimate except the last few indices. As a conclusion, it seems even plausible to use the theoretical estimate $\sqrt{m-i+1}\,\alpha q$ except for the last several indices. This might cause a problem for estimating the γ for the second intersection. We will consider such problem in a later section.

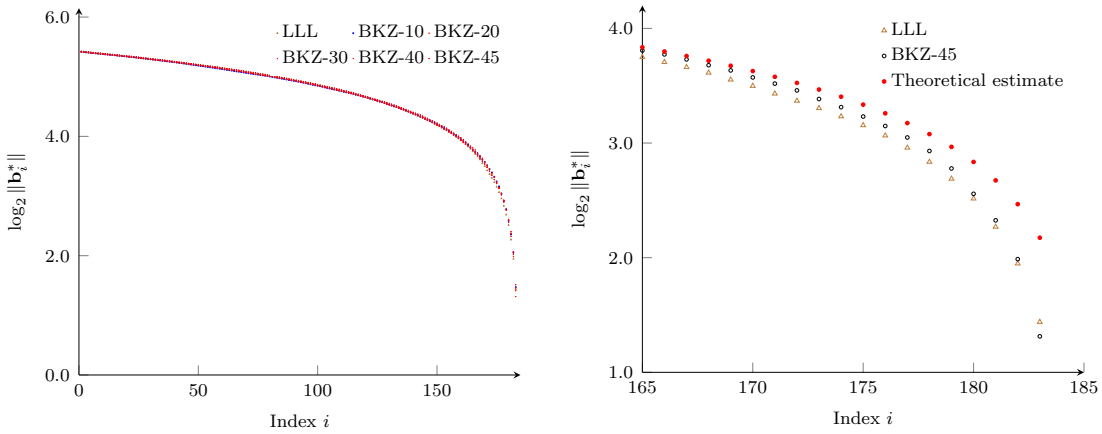

Figure 15: Logarithmic norm of the projection of $\mathfrak{v}$ on BKZ-β reduced bases for $\beta = 10, 20, 30, 40, 45$. The right is the same as the left hand side, but zoomed-in for only LLL and BKZ-45. Furthermore, theoretical estimate $\log_2(\sqrt{m-i+1}\,\alpha q)$ is plotted.

3.2 GAP IN uSVP FROM LWE

In this section, we study the practical behavior of the reduction from the BDD problem to the uSVP problem. Note that in practice, we usually use the Kannan's embedding with $\mu = 1$. However, in theory, it is not known that whether the gap γ of the embedded uSVP lattice in this case is optimal. But this seems to be the preferable setting in practice. As shown experimentally in [76], decreasing the embedding height is advantageous in solving LWE via the embedding technique. In this section we further explain why $\mu = 1$ is preferable by investigating the concrete gaps in the uSVP problem.

Let the BDD problem arise from LWE be $\mathrm{BDD}_{1/\gamma}$. We recall the reduction from $\mathrm{BDD}_{1/\gamma}$ to $\mathrm{uSVP}_{\gamma/2}$ by Lyubashevsky and Micciancio [49]. Given the $\mathrm{BDD}_{1/\gamma}$ instance $(\mathbf{B}, \mathbf{t})$, the following embedded lattice is contructed

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0} & \mu \end{pmatrix} \in \mathbb{Q}^{n+1 \times n+1},$$

where μ is set to be the distance $d = \mathrm{dist}(\mathbf{t}, \mathfrak{L}(\mathfrak{B}))$. Since this is a $\mathrm{BDD}_{1/\gamma}$ instance, we know that $d \leq \lambda_1(\mathcal{L})/\gamma$. Let $\in \mathfrak{L}(\mathfrak{B})$ denote a closest vector to the target point $\mathbf{t}$. Lyubashevsky and Micciancio [49] show that the vector $\mathbf{s}' = ((-\mathbf{t})^{\mathrm{T}}, -\mu)^{\mathrm{T}}$ is a shortest non-zero vector in the lattice $\mathfrak{L}(\mathfrak{B}')$ and other independent vectors are at least γ times larger than this. The reduction cares about the worst-case behaviors. In practice, it may be quite possible that all other independent vectors are more than $\gamma/2$ times larger and hence leads to a uSVP problem with larger gap. In fact, we will show that this is indeed the case in practice and investigate to what extent it is better than the $\gamma/2$-gap. Note that there is a natural upper-bound for the reduction. Precisely, the gap in the uSVP problem cannot be larger than $\sqrt{2}\gamma/2$ since a shortest vector in the BDD lattice also resides in the embedded lattice and $\mathbf{s}'$ has length $\sqrt{2}d/2$. On the other hand, in practice, we just take $\mu = 1$ in the embedded lattice. We assume

that the vector $((-\mathbf{t})^{\mathrm{T}}, -1)^{\mathrm{T}}$ is a shortest non-zero vector in the lattice $\mathfrak{L}(\mathfrak{B}')$ and such that there is a sufficiently large gap between all other independent vectors and this shortest non-zero vector. In the 2008 estimate, this is equivalently assumed to be that the uSVP problem derived from $\mu = 1$ has a gap of γ (although this is not supported theoretically in the worst case). In fact, such γ-gap already implies the reduction has reached its natural upper bound – note that the shortest vector in the given BDD lattice $\mathfrak{L}(\mathbf{B})$ is about γ times larger than d as defined.

In this section, we investigate concretely the gap in the uSVP problem in experiments. Perhaps surprisingly, we show that the gap in the uSVP instance are somewhat close to the upper-bound γ in practice, even though this is not guaranteed in the worst-case. This also explains that why it is preferable to use $\mu = 1$ in practice. We set up the following experiments to investigate the gap in the resulted uSVP instance in practice. For each set of parameters, we generate 100 LWE instances. For each instance, we construct the embedded lattices in two ways, with $\mu = 1$ and $\mu = d$ where $d = \lfloor \|\mathbf{e}\| \rceil$. In experiments, we compute and compare the gaps in the resulted uSVP instances.

n	m	q	BDD lattice		uSVP lattice $\mu = 1$			uSVP lattice $\mu = \|\mathbf{e}\|$		
			Theory	Experiment	Theoretical upper	Experiment	Ratio	Theoretical upper	Experiment	Ratio
16	32	1031	2.71	2.78	2.78	$\lesssim 2.55$	0.92	1.97	$\lesssim 1.96$	0.71
16	48	1031	8.40	8.49	8.49	$\lesssim 7.81$	0.92	6.00	$\lesssim 5.99$	0.71
32	48	8101	1.65	1.68	1.68	$\lesssim 1.58$	0.94	1.19	$\lesssim 1.19$	0.71
32	64	8101	7.23	7.33	7.33	$\lesssim 6.95$	0.94	5.18	$\lesssim 5.16$	0.70

Table 3: Experimental comparison on the gap of uSVP derived from two embeddings.

We explain the notations in Table 3. For each parameter n, m, q in LWE, we use error deviation $\sigma = 3.1925 \approx \frac{8}{\sqrt{2\pi}}$. For each LWE/BDD instance, we calculate the theoretical gap in the BDD problem from $\min\left(q, (\Gamma(1+m/2)^{1/m})/\sqrt{\pi}\cdot q^{(m-n)/m}\right)/(\sigma\sqrt{m})$. Note that we can measure in a better way: since we know the errors, we use the average norm of the errors in the denominator (instead of the estimate $\sigma\sqrt{m}$). This is

tabulated in the **"Theory"** sub-column under "BDD". Then we use BKZ_m to find the $\lambda_1(\mathfrak{L}(\mathfrak{B}))$ and divide that by the norm of error in LWE. This is recorded in the **"Experiment"** sub-column under "BDD". Note that the experimental values obtained is slightly larger than the theory; this is perhaps due to the solver only finding the approximate shortest vector in practice. Then we construct the embedded uSVP lattices with $\mu = 1$ and $\mu = d$, respectively. The sub-columns **"Theoretical upper"** under "uSVP lattice" denote the upper bound of the gap in the uSVP instances one can achieve using the values in the **"Experiment"** (not "Theory") sub-column under "BDD", for each type of embedding, respectively. For example, the experiment value 2.78 under $n = 16, m = 32, q = 1031$ implies that the corresponding uSVP instances with $\mu = \|\mathbf{e}\|$ can at most have a gap of 1.97. The sub-column **"Experiment"** under "uSVP lattice" gives the experimental values for the gaps between the norm of a second shortest vector and $\|(\mathbf{e}^{\mathrm{T}}, -\mu)^{\mathrm{T}}\|$. Note that here we approximate the norm of a second shortest vector by considering the second shortest vector in a reduced basis using BKZ of blocksize m. This is not necessarily the λ_2 but hopefully a close approximation. Thus we denote "$\lesssim$" in the table. For the lattice reduction, we use BKZ in `FPLLL` until exhaustion with full enumeration for $m = 32$ and pruned enumeration for other m. The sub-column **"Ratio"** under "uSVP lattice" computes the ratio between the uSVP gap and the BDD gap. That is, it reflects the practical behavior of the reduction from $\text{BDD}_{1/x}$ to uSVP_y where the sub-column "Ratio" is computed as y/x. The larger the ratio, the better (larger gap) the uSVP instance is. All the figures in the table are averaged over 100 instances.

From a theoretical perspective, it is perhaps surprising to see that the BDD-uSVP reduction works pretty well in practice with both μ. In particular, with $\mu = 1$, it seems that $\text{BDD}_{1/\gamma}$ already reduces to $\text{uSVP}_{0.9\gamma}$ in practice. In theory for such case ($\mu = 1$), it is possible that there exists a lattice point $' \in \mathfrak{L}(\mathbf{B})$ that is closer to $k \cdot \mathbf{t}$ for some multiple k, and therefore $('-k\cdot\mathbf{t}, -k)$ decreases the desirable gap. However,

experiments in Table 3 seems to imply that such bad points are rare in practice. Note that such cases can be provably eliminated by setting a larger $\mu = \|\mathbf{e}\|$ as shown in [49]. Specifically for such μ, it is guaranteed that the uSVP gap is $\gamma/2$ (from $\mathrm{BDD}_{1/\gamma}$) in the worst case. Similarly, the practical/average behavior seems to be much better: with $\mu = \|\mathbf{e}\|$, the $\mathrm{BDD}_{1/\gamma}$ problem reduces to $\mathrm{uSVP}_{0.7\gamma}$ in practice.

We do not know how to explain such average behavior in theory. It may be related to the difference on the natural upper-bounds in two embeddings: with $\mu = 1$, the natural upper-bound of the gap in the uSVP problem is γ. This is larger than that (e.g. $\gamma/\sqrt{2}$) derived from the lattice using $\mu = \|\mathbf{e}\|$. Thus it may be due to a larger upper-bound providing larger "room" for the reduction, together with annoying "extremely close" lattice points (to multiple of target vector $\mathbf{t}$) being rare in practice. It may be interesting to further investigate this, e.g. by trying more μ between 1 and $\|\mathbf{e}\|$ and observe the impacts to the uSVP gap. We leave more investigations on this for future work.

So far, we've only discussed the gap appeared in the embedded uSVP instance under different embedding parameters. We further look at the impacts on the cost estimate under different embedding heights. In the 2008 estimate, it is assumed that given as input a $\mathrm{BDD}_{1/\gamma}$ problem, one can reduce to a uSVP_{γ} problem. Then the root Hermite factor δ can be derived from the gap γ and hence the blocksize & running-time. It is also natural to see that when using the 2008 estimate, it is preferable to use $\mu = 1$ since it leads to a larger gap in the uSVP problem. In the 2016 estimate, the gap of the uSVP problem is not used explicitly. But one can see that the estimate is asymptotically equivalent to $\delta^m \leq \sqrt{\beta}\, \frac{\sqrt{m} q^{(m-n)/m}}{\|(\mathbf{e}|\mu)}\|$. The fractional part of the equation corresponds to the gap in the uSVP problem. Note that the difference on the gap using $\mu = 1$ and $\mu = \|\mathbf{e}\|$ is at most a scaling factor of $\sqrt{2}$. It seems to be a small factor however it may affect the concrete security level of schemes with moderate size.

3.3 SECOND INTERSECTION

An interesting phenomenon observed in [6] shows that in several cases the lattice reduction behaves even better than the 2016 estimate for some parameters. First, the BKZ algorithm recovers a projection $\pi_i(\mathfrak{v})$ at index following a distribution with a center below $d-\beta+1$. After that, a size reduction usually immediately recovers the full secret. It is outlined in [6] that this may be caused by the occurrence of a second intersection of the projected vector with the Gram-Schmidt vectors. For example, to solve LWE parameter $n=65, m=182, q=521$ and $\alpha q = 8/\sqrt{2\pi}$, we need to execute BKZ with blocksize $\beta = 56$ according to Equation (3.3). Since $\beta = 56$ satisfies Equation (3.3), a projection of our error should be found at index $d-\beta+1=128$, recovering the last 56 coefficients of the error which leads to size reduction recovering the rest. In experiments the projection is found earlier (at index ≈ 124.76) and the coefficients of the error are found after one more call to size reduction. Second, the blocksize required to recover the secret (on average) is actually smaller than that estimated from Equation (3.3). For the LWE parameter mentioned above, it requires to run BKZ using blocksize 56 according to Equation (3.3). However, as noted in [6], using blocksize 51 is sufficient to recover more than half of the instances. Some justification has been outlined in Subsection 4.3 of [6] while we provide a more theoretical analysis.

We first recall the phenomenon in more detail as well as a brief explanation given in [6]. According to Equation (3.3), the projection of the shortest vector should be recovered at position $d-\beta+1$ when running the BKZ with blocksize β on the $uSVP$ instance over a d-dimensional lattice (recall that in our description, the $d=m+1$). However, it is observed that the existence of a second intersection on the expected projection length of the shortest vector and the Gram-Schmidt norms under GSA assumption may speed-up the recovery of $\mathfrak{v}$. For example, Figure 16 compares the (logarithmic) Gram-Schmidt norms of BKZ_{56} reduced basis under GSA assumption

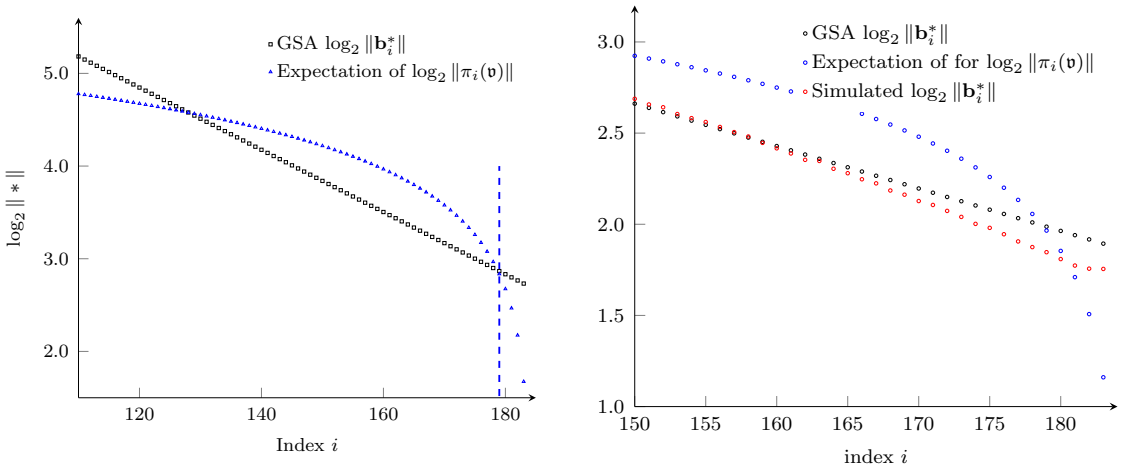

Figure 16: Comparison between Gram-Schmidt norms of BKZ_{56} under GSA and the expected length of $\pi_i(\mathfrak{v})$.

and the expected length of $\pi_i(\mathfrak{v})$. Note there are 5 indexes in which $\|\pi_i(\mathfrak{v})\|$ is smaller than the Gram-Schmidt norms, thus in this case, we denote $\kappa = 5$. In particular, after the second intersection, the expected length of $\pi_i(\mathfrak{v})$ will be less than the $\|\mathbf{b}_i^*\|$ for κ indexes in the end. Hence the projection is likely to be the smallest vector of the projected lattice $\mathfrak{L}(\pi_{d-\kappa+1}(\mathbf{b}_{d-\kappa+1}), \ldots, \pi_{d-\kappa+1}(\mathbf{b}_d))$ of dimension κ. The SVP oracle will find this projection and the BKZ algorithm will then insert it at index $d-\kappa+1$. As a result, $\mathbf{b}_{d-\kappa+1}$ is updated to be (the lifted vector of) the projection of the vector $\mathfrak{v}$ over the last κ Gram-Schmidt vectors. Further, it is likely that $\pi_{d-\beta-\kappa+1}(\mathfrak{v})$ is the shortest vector of the projected lattice $\mathfrak{L}(\pi_{d-\beta-\kappa+1}(\mathbf{b}_{d-\beta-\kappa+1}), \ldots, \pi_{d-\beta-\kappa+1}(\mathbf{b}_{d-\kappa+1}))$ of size β after which $\mathfrak{v}$ can be recovered by a size reduction according to [6].

Therefore, assuming a projection of our vector $\pi_{d-\kappa+1}(\mathfrak{v})$ has already been found, an SVP oracle will find $\pi_{d-\beta-\kappa+1}(\mathfrak{v})$ in the lattice
$\mathfrak{L}(\pi_{d-\beta-\kappa+1}(\mathbf{b}_{d-\beta-\kappa+1}), \ldots, \pi_{d-\beta-\kappa+1}(\mathbf{b}_{d-\kappa+1}))$.

3.3.1 On Smaller Blocksize

A related interesting phenomenon is that often a smaller blocksize may be already sufficient to solve the $uSVP$ problem. This has been observed in [6] where a blocksize

of $\beta' = \beta - \kappa$ is sufficient to recover the secret with high probability. We give a heuristic justification of this based on the second intersection. Suppose now β is the smallest blocksize that satisfies Equation (3.3) with a nonzero κ depending on β.

Denote $\beta' = \beta - \kappa$. Suppose $\mathrm{BKZ}_{\beta'}$ is run (instead of BKZ_{β}). For convenience, let δ_β denote the value of δ given blocksize β. Let κ' be the amount of indixes where the projection of $\mathfrak{v}$ is smaller than the GSA predicated Gram-Schmidt norm. Due to the second intersection, a projection of $\mathfrak{v}$ is likely to be found at index $d - \kappa' + 1$ so after SVP the vector $\mathbf{b}_{d-\kappa'+1}$ will contain the last κ' coefficients of $\mathfrak{v}$. Therefore the norm of $\mathfrak{v}$, if decomposed in terms of the Gram-Schmidt vectors $\mathbf{b}_i$, will concentrate on the first $d - \kappa' + 1$ components. More precisely, $\|\mathfrak{v}\|^2 = \sum_{i=1}^{d-\kappa'+1} c_i^2 \|\mathbf{b}_i^*\|^2$ where c_i are the coefficients in the decomposition. Following the same reasoning as in [6], we look at the β'-dimensional lattice $\mathfrak{L}(\pi_{d-\beta'-\kappa'+1}(\mathbf{b}_{d-\beta'-\kappa'+1}), ..., \pi_{d-\beta'-\kappa'+1}(\mathbf{b}_{d-\kappa'+1}))$. If the projected shortest vector has a smaller norm than the GSA predicated norm of blocksize β', then we would be able to recover the last $\beta' + \kappa$ coefficients. The success condition can be phrased as

$$\sqrt{\beta' + \kappa'}\alpha q \leq \delta_{\beta'}^{2\beta'-d+2\kappa'} \operatorname{Vol}(\mathfrak{L})^{1/d}. \tag{3.5}$$

Equation (3.5) is sometimes satisfied, but not always, depending on the relation between κ and κ'. It seems plausible to assume that $\kappa' \approx \kappa$ for the analysis, albeit this may not be true in practice. (This can be seen from experiments in that the newly found β' will not recover as many error vectors as the original β. For example, $\beta' = 51$ in the aforementioned LWE parameters can only recover half of the instances.) Note that if $\kappa \approx \kappa'$, the left-hand side of Equation (3.5) is the same as $\sqrt{\beta}\alpha q$ and the right-hand side is larger, hence $\pi_{d-\beta'-\kappa'+1}(\mathfrak{v})$ is the shortest vector in the local lattice. By recovering $\beta' + \kappa$ coefficients of $\mathfrak{v}$, a following size reduction will find the rest with a high probability.

3.3.2 Experiments on κ

In the experiments to follow, we consider the last projection of our vector $\mathfrak{v}$ that was found before it is completely recovered in the next tour by size reduction. This confirms the existence of κ in practice. With LWE parameters $n = 65, m = 182, q = 521$ in both parameter sets, we consider two different choices of αq that produces different κ. The first is $\alpha q = 3.192$ and requires $\beta = 56$ while the second is $\alpha q = 2.469$ and requires $\beta = 42$. We run 800 instances in total and take the average for both parameter sets. The distribution of κ found are plotted in Figures 17 and 18. The y-axis represents the counts over 800 where a projection of $\mathfrak{v}$ was found at index $d-\kappa+1$ before the tour it was completely recovered and the x-axis is the value κ. In both cases, we did not consider projections of $\mathfrak{v}$ that were found at an index less than or equal to $d - \beta + 1$ as this will probably be where $\mathfrak{v}$ is recovered by size reduction. The experiment that required $\beta = 56$ was allowed to run for at most 20 tours while the experiment requiring $\beta = 42$ is allowed 60 tours.

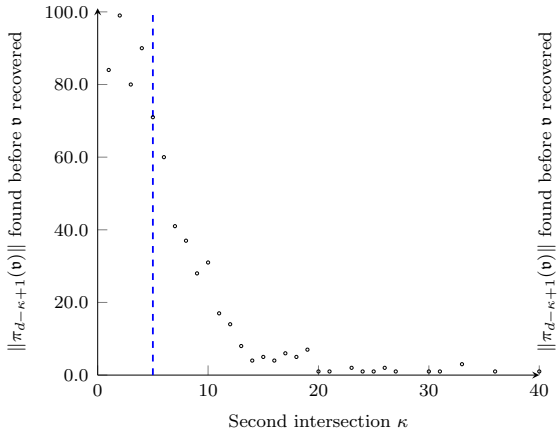

Figure 17: Blocksize $\beta = 56$ required in Equation (3.3) and $\kappa = 5$.

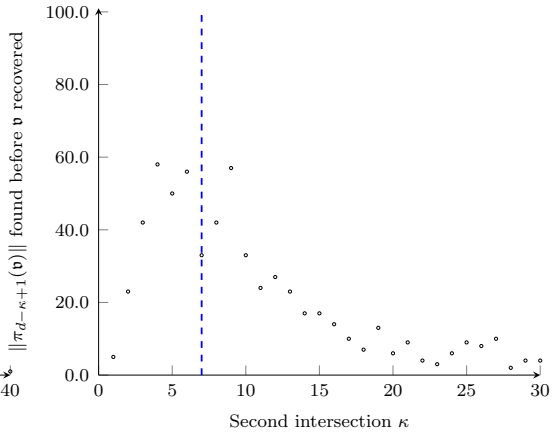

Figure 18: Blocksize $\beta = 42$ required in Equation (3.3) and $\kappa = 7$.

We notice that the experimental values for κ indeed follow approximately from the theoretical predicate from Equation (3.3). However, we also notice that the experimental value for κ seems to be slightly less than the predicted value. This

could be due to the inaccuracy of GSA when predicting the length of the last few projections. It is known that the simulator-based approach [31, 15] provides a better estimation for the behavior of the lengths $\|\mathbf{b}_i^*\|$. We considered the average simulated $\|\mathbf{b}_i^*\|$ over 1000 instances with blocksize 56 and 200 tours. By comparing the simulator to the expected length of our projection (see Figure 16), we see that fewer projections of $\mathfrak{v}$ are below the simulator after the second intersection: There are 3 (resp. 5) indexes in which $\|\pi_i(\mathfrak{v})\|$ is smaller than the simulator's (resp. GSA's) value for $\|\mathbf{b}_i^*\|$ (comparing Figure 17 with Figure 16).

3.3.3 Convergence of κ

It has been conjectured [6] that the second intersection will not happen for cryptographic meaningful parameters. We first show that the position of the second intersection approaches 0 as $\beta \to \infty$. We will also provide a numerical analysis for the index of the second intersection using both GSA assumption and simulator. We first take the logarithm of both the Gram-Schmidt norm at index x and the norm of $\pi_x(\mathfrak{v})$:

$$\log(\pi_x(\|\mathfrak{v}\|)) \approx \log(\sqrt{d-x+1} \cdot \alpha q),$$
$$\log(\|\mathbf{b}_x^*\|) \approx (x-1)\log(\alpha) + \log(\|\mathbf{b}_1\|)$$

where $\alpha \approx \delta^{-2}$ is the constant ratio in GSA. Note that $\|\mathbf{b}_1\| \approx \delta^d Vol(\mathfrak{L})^{1/d}$. Assuming Equation (3.3) is satisfied so that $\alpha q \approx \delta^{2\beta-d}\mathrm{Vol}(\mathfrak{L})^{1/d}/\beta^{1/2}$, the inequality can be represented as

$$\log(\frac{\kappa}{\beta}) \leq -4\log(\delta)(-\kappa+\beta) \tag{3.6}$$

where $\kappa = d-x+1$. If there is a nontrivial second intersection, the above relation has to be true for at least $\kappa = 1$. Using $\delta \approx v_\beta^{-1/(\beta(\beta-1))}$, one could see that for large enough blocksize, this relation can not be satisfied and hence the second intersection will not happen for large blocksize. This shows that the second intersection approaches 0 as

$\beta \to \infty$. Further, we numerically investigate the evolution of κ in terms of β using Relation (3.6). Figure 19 considers the values of κ given by Relation (3.6) for different values of β. Notice that Figure 19 shows that $\beta = 278$ is the smallest blocksize where κ already becomes 0. This suggests there is no second intersection when $\beta \geq 278$ is needed to satisfy equation Equation (3.3). However, this could be an over-estimate from the attacker's point of view since the GSA assumption is used here.

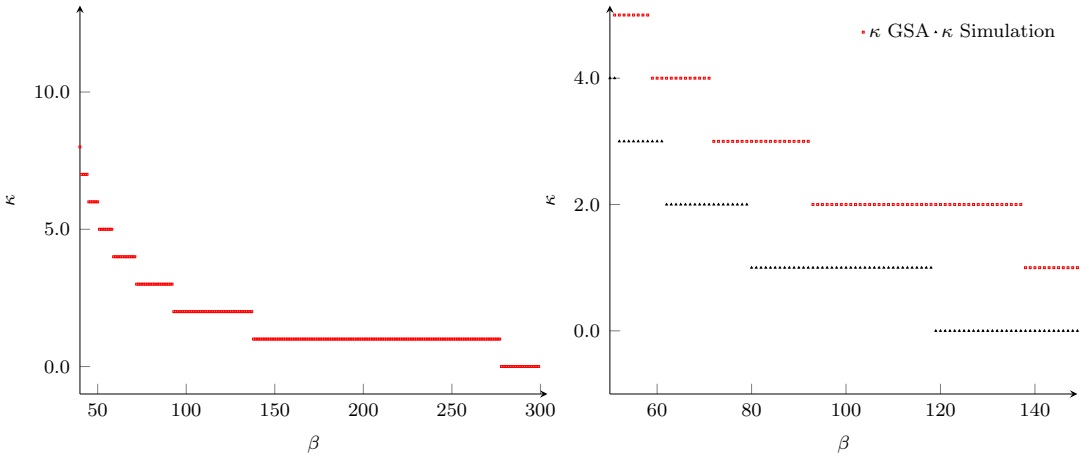

Figure 19: Maximal κ satisfying Equation (3.3) given β.

To get a more accurate estimation of the value κ, we further compare that with the BKZ simulator. The next figure considered several different parameter sets ($n = 65, m = 182, q = 521$) only varying in αq and necessary β (averaged over α and LWE instances). We simulate 200 tours of BKZ-β using the BKZ simulator and averaged 1000 instances of each parameter set. Figure 19 shows that the value of κ derived by comparing the simulated $\|\mathbf{b}_i^*\|$ to $\|\pi_i(\mathfrak{v})\|$ suggests there is no second intersection for blocksizes larger than 120. One can also see this produces slightly smaller κ for a given β than the comparison assuming GSA. This seems reasonable since the GSA assumption is known to be optimistic from an attacker's point of view. In conclusion, this further suggests that a second intersection will only affect the results of running BKZ-β on smaller parameter sets.

CHAPTER 4
RESOURCE ESTIMATES FOR A FLOATING-POINT MULTIPLICATION QUANTUM CIRCUIT

Both LLL and discrete pruning [10] must handle fractions produced by Gram-Schmidt orthogonlization, which have huge numerators and denominators. High precision floating-point arithmetic is used to handle such fractions in order to preserve resources while representing more values than fixed-point representations. A quantum implementation of LLL, such as that in [73], would still need to handle the floating-point arithmetic necessary to perform Gram-Schmidt orthogonalization. A floating-point adaptation of LLL [57] by Nguyen and Stehlé requires a precision of $\log_2(3)d$ where d is the dimension of the lattice. This floating-point adaptation is referred to as L^2. We consider the quantum cost of quantum floating-point circuits of this precision for lattices used in the cryptanalysis of a few lattice-based proposals. Security estimates of lattice-based proposals are calculated in [4] as well as the block-size requirements for successful primal attacks. Enumeration with discrete pruning require several calls to a multiplication and division algorithm since the process requires Gram-Schmidt orthogonalization. Though, LLL is called on the entire d-dimensional lattice while running BKZ, we limit our focus to the sublattice on which an enumeration algorithm is called. One is able to copy-paste the smaller local "block" and apply enumeration on this sublattice during BKZ process. To estimate the floating-point requirements, we consider the precision required by L^2 when called on this sublattice of dimension β.

Floating-point values are tuples $(x_\pm, x_M, x_E)$ where $x \approx (-1)^{x_\pm} x_M 2^{x_E}$. Analogous

Scheme	β for Primal Attack [4]	$\log_2(3) * \beta$	ℓ	width
NewHope-512	386	612	23	636
Frodo-640	485	769	25	795

Table 4: Precision Requirements for L^2 algorithm on sublattice of dimension β

to fixed-point operation width, a floating-point value's *width* is the number of qubits to store the tuple representing the floating-point value. The value $x_M \in [1, 2)$ is called the *mantissa* and the length of x_M's binary representation is the *precision*. Exponents of the floating-point tuple are held in ℓ-qubit registers. Denote the precision as k. We maintain the following relationship between width and precision in accordance with the IEEE standard for floating-point arithmetic [67]:

$$k = width - \lceil (4 * \log_2(width)) \rfloor + 13.$$

The 2^0 entry is always high and stored as the most significant digit for mantissa representations.

4.1 ADDITION AND MULTIPLICATION CIRCUITS

We decide to construct the addition circuit designed by Takahashi et al. [71] and implement the circuit in the open-source software framework for quantum computing, ProjectQ [68, 36]. Other addition circuits include [29, 25], however, these come with the cost of greater resource requirements than the implemented addition circuit. To construct the addition circuit we need $5n - 5$ CNOT gates and $2n - 1$ Toffoli gates where n is the width of the operation. An actual addition circuit for 6-bit addition is presented in Fig. 20.

Perhaps it is of little surprise the addition circuit also works when the input integers are in 2's complement representation with the sign as the most significant

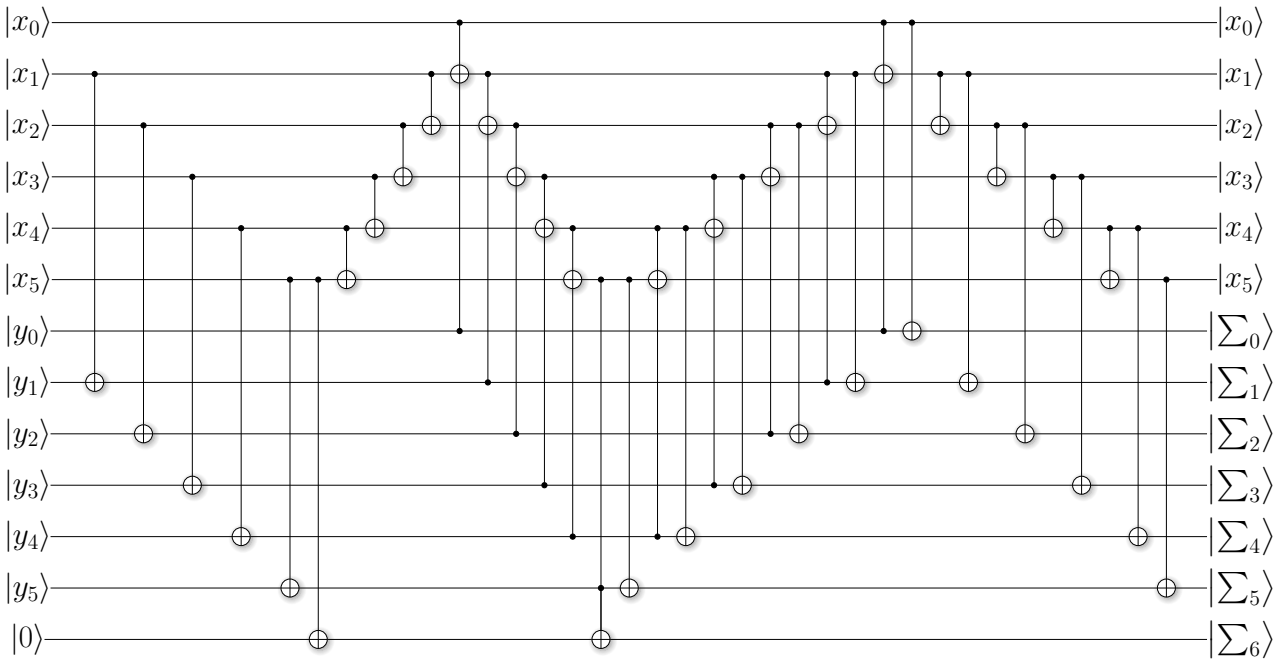

Figure 20: 6-bit Quantum Circuit for Addition from [71]

bit. The carry bit contains no meaningful data in the output when adding numbers of this form. In future sections we assume values are in 2's complement representation and control for overflows and underflows. Integer values x and y of n qubits are embedded in $(n+1)$-qubit registers $|\overline{x}\rangle$ and $|\overline{y}\rangle$, respectively. Overflow/underflow occurs when $|\overline{x}_{n-1}\rangle \neq |\overline{x}_n\rangle$. This means we have different signs for the n-qubit and $n+1$-qubit representations. We make use of the previous sign circuit to detect these errors. Some other quantum circuit designs for addition are [25, 29] and each can be used interchangeably with our choice of circuit. These two options require additional qubits and have a higher T-count than the circuit designed by Takahashi et al.

Multiplication follows naturally from an addition circuit through a process of controlled, shifted addition. The product of two n-qubit values $x, y \in \mathbb{Z}$ is held in a $2n$-bit register. This $2n$-qubit register is initially all zeros. If the least significant qubit of $|y\rangle$ is high, $|x\rangle$ is added to the first n-qubits of the all zeros register. Then, if the next qubit of $|y\rangle$ is high, $|x\rangle$ is added to the n-qubits in positions 1 to $n+1$. This process continues until all the qubits of $|y\rangle$ have been exhausted. The circuits

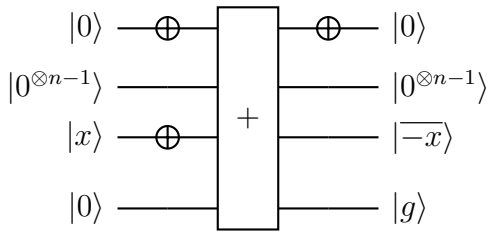

Figure 21: Circuit for $2's$ Complement

for addition and multiplication are denoted + and ×, repsectively.

$$+ : |x, y, 0\rangle \rightarrow |x, x + y\rangle$$

$$\times : |x, y, 0^{\otimes 2n}\rangle \rightarrow |x, y, x \times y\rangle$$

Resource counts of these quantum arithmetic operations are calculated using the resource counter in ProjectQ and displayed in Table 5.

Table 5: Resource counts of arithmetic circuits

Op_{bit}	Qbts	CNOT	CCNOT	CCCNOT
$+_{32}$	65	155	63	0
$+_{64}$	129	315	127	0
$\times_{32}$	128	0	4960	2016
$\times_{64}$	256	0	20160	8128

Signed multiplication can be performed by making use of the previously discussed sign circuit in conjunction with a circuit for multiplication. To compute 2's complement, a NOT gate is applied to every entry in the register. The input register is then added to $|1\rangle$ using the addition circuit. We denote the circuit simulating 2's complement in Fig. 21 as 2_c. In Fig 21, the value $\overline{-x}$ is the n-qubit representation of $-x$ in 2's complement representation. The register holding $|1\rangle$ is formed by applying

a NOT gate to the first entry of $|0^{\otimes n}\rangle$.

Subtraction is done by applying 2's compliment, adding, then uncomputing 2's compliment to circumvent the production of an extra carry bit. Say we wish to find the difference $x-y$. The n-qubit register holding $-y$ in 2's complement is found using the circuit 2_c. Values x and $-y$ are then added using the implemented addition circuit. Any carry bit produced while adding two values in 2's complement representation is discarded as useless output.

4.2 FLOATING-POINT MULTIPLICATION

To perform floating-point multiplication we turn to the hand-optimized proposistion by Häner et al. [35]. The authors report that their hand-optimized circuit enjoys huge reductions in required qubits and gates when compared to an automatically generated approach. Circuits which are automatically generated are much easier to analyze since the width directly determines the resources necessary for the circuit. For example: the floating-point addition circuit designed by [59] will always require $4(width)-7$ qubits. Hand-optimized implementations require different component gates with input that doesn't scale with the operation width and precision. To multiply two floating-point values x and y, the mantissa are multiplied and the exponents added. If the mantissa product is greater than 2, the binary representation of the product is shifted left by one, and one is added to the resulting exponent. The sign of the resulting product is found in the obvious way using the previously discussed sign circuit.

4.2.1 Improved Circuit Design

We improve on the circuit by Häner et al. by replacing the shift circuits used in the original implementation with a simpler, less resource consuming, controlled shift circuit. Another modification includes the test for an overflow or underflow of exponents. To test for overflow/underflow we embed the ℓ qubit 2's complement representation

of exponents in $\ell + 1$ qubit 2's complement representations. We denote y' as the $\ell + 1$-qubit embedding of the ℓ-qubit exponent y. When the sign qubit of one representation does not match the other, an overflow/underflow has occurred. We present the modified circuit in Figure 22.

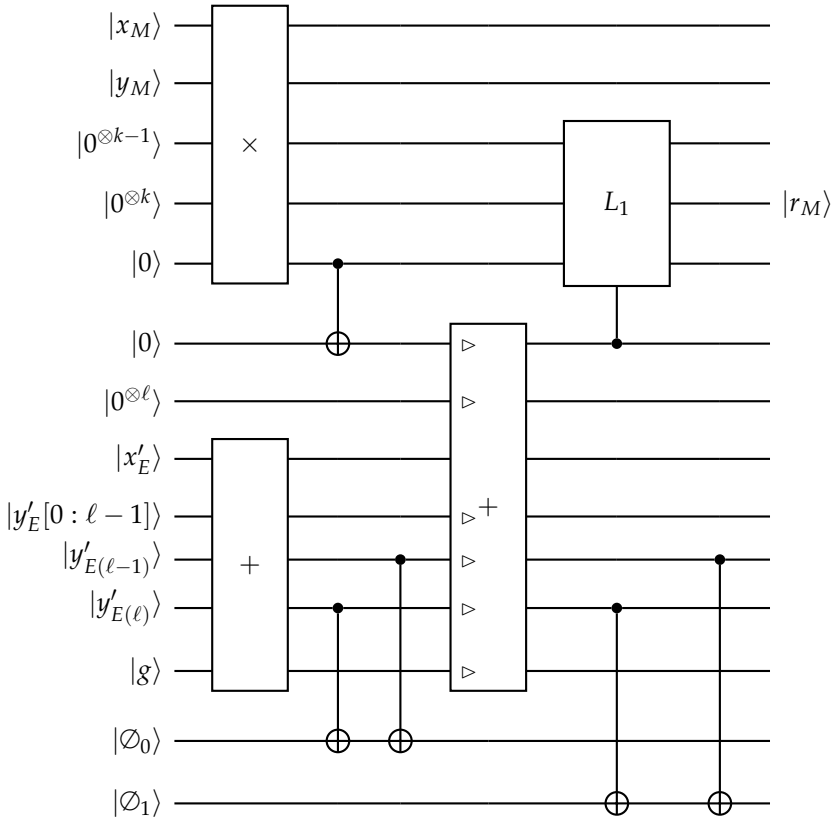

Figure 22: Floating-Point Multiplication Circuit Following the Design of [35]. High $|\emptyset_i\rangle$ signals an underflow or overflow has occurred.

The original proposition uses a circuit for shifting a qubit register by a value specified by a separate input in the circuit. However, in multiplication we only have to shift by one instead of a specified value. We compare these two shifting circuits in Figure 23. Due to the resource savings, we choose to implement the circuit which only shifts by one in order to save on resources. This circuit is denoted $L1$ while the circuit in [35] is denoted Cshift.

If the product of the two mantissa is greater than 2, the most significant bit of the

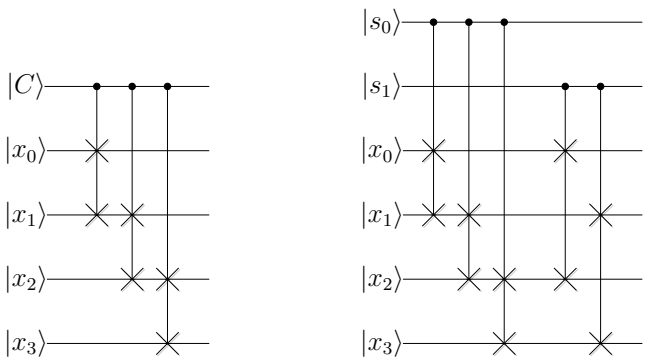

Figure 23: Two left shift circuits. Circuit on the left shifts by only 1 where the circuit on the right shifts by a specified value held in the register $|s\rangle$.

$2n$-bit qubit register will be high. A controlled NOT with this qubit as the controlled will determine if the $\ell + 1$-qubit representation contains the binary representation of 1 or zero. When the control qubit is high, the value of one in the $\ell + 1$ register will be added to the exponent. Exponents are of ℓ-qubits but are held in an $\ell + 1$-qubit register to allow for overflow/underflow testing. Denote the embedding of $|x_E\rangle$ as $|x'_E\rangle$. The least significant digit of the originally zero $\ell + 1$ register will be 1 or 0 depending on the mantissa product. A controlled left shift is applied to the mantissa product if this qubit is high. The resulting exponent will be contained in the register originally holding y_E after the circuit is computed. m qubits of the register holding the product of the mantissa will contain the resulting mantissa, denoted r_M. We choose to omit the qubits and gates required for computing the resulting sign in Figure 22. Resources needed for computing the sign will be considered in our analysis of the resource requirements for the floating-point multiplication circuit.

4.3 RESOURCE ESTIMATES FOR HIGH PRECISION

Applications of floating-point operations will likely be those that require high precision. When considering lattices of large dimensions, precision requirements for the L^2 algorithm are much higher than the precision used in the circuits considered in [35].

A primal attack calls enumeration on sublattices of dimension β. In this section, we consider the precision required by the L^2 algorithm for lattices of dimension equal to the required block-size for a successful primal attack on a few NIST proposals. The precision requirements are given as the $\log_2(3)\beta$ column of Table 4. We consider the quantum cost of each componenet circuit of the floating-point multiplier and then give the entire circuit's cost.

Multiplication of the mantissa x_M and y_M requires the most resources in the floating-point cirucit. Both mantissa are held in k-bit registers and the k-qubit multiplication requires an intial all zeros qubit register of length $2k$.

Table 6: Resource counts for multiplication circuits

Op	width	Qbts	CCNOT	CCCNOT
$\times$	769	3076	1181953	2952960
$\times$	612	2448	1869660	748476

Two $(\ell+1)$-qubit addition circuits are required for floating-point multiplication. The first adds the two ℓ-qubit exponents embedded in qubit registers of length $\ell+1$. A second addition circuit adds either 0 or 1 to the resulting exponent if the mantissas' product is greater than or equal to 2. After each addition, an overflow/underflow test is executed. If the most significant qubit of the $(\ell+1)$-qubit representation differs from the most significant qubit of the ℓ-qubit representation, an overflow/underflow flag occurs. This means the sign of the exponent in the ℓ-qubit representation is different than the sign of the $(\ell+1)$-qubit representation. Qubits $|\emptyset_0\rangle$ and $|\emptyset_1\rangle$ are initialized to $|0\rangle$. A non-zero result for these qubits at the end of the computation signifies an overflow/underflow has occured. Exponents are assumed to be in 2's complement representation so the carry qubit of the addition circuit holds no meaningful value.

The controlled left shift is applied to a register of $2k$ qubits. We compare both shift circuits in Figure 23. However, in the final estimates for the resources of floating-point

Table 7: Resource counts for addition circuits

Op	width	Qbts	CNOT	CCNOT
+	23	47	110	45
+	26	53	125	51

multiplication, only the left circuit is considered.

Table 8: Resource counts for two shifting circuits

Op	width	Qbts	CSWAP
$L1$	1224	1225	1223
CShift	1224	1235	11417
$L1$	1538	1539	1537
CShift	1538	1549	14871

Table 9: Resource counts for floating-point multiplication circuit

Op	k	ℓ	Qbts	CNOT	CCNOT	CCCNOT	CSWAP
$\times_{fp}$	612	23	2526	226	1869750	748476	1223
$\times_{fp}$	769	25	3160	256	1182055	2952960	1537

CHAPTER 5
CONCLUSIONS AND FURTHER WORK

5.1 CONCLUSIONS

The thesis discusses the inconsistencies in lattice-basis reduction for smaller lattices used to solve LWE. Many instances of LWE for small parameters seem to be easier to solve in practice than estimate given by Equation (3.3). It is tough to say whether or not the behavior of lattice-reduction for larger parameter sets will be similar. We are unable to analyze actual basis reduction on larger parameter sets due to the time and resource restrictions of the reduction. By studying easier instances of LWE, we hope to make accurate predictions about more difficult instances. Higher dimensional lattices seem to behave much more consistently than the lower dimensional ones, e.g., the differences in Figure 9 for $n = 70$ and $n = 95$. As shown in Chapter 3.3, some of the subtleties that exist for the experiments should not be present when considering lattices used in the cryptanalysis of the NIST proposals. These subtleties lie in the analysis of the projections of $\mathfrak{v}$. The fact early projections of the vector are typically found during lattice reduction make for a difficult analysis. For more difficult instances of reducing LWE to uSVP, the projection lengths may behave more consistently. Nonetheless, the work in this thesis provides usefull insight to the efficacy of one of the best attacks on lattice-based cryptosystems to date.

Floating-point multiplication still remains expensive in terms of quantum resources. The huge resource expense becomes even more obvious when considering applications involving high precision. Results in Chapter 4 confirm we have a long way to go before a practical implementation of floating-point multiplication in crypt-

analysis is possible.

5.2 FUTURE WORK

Considering the primal attack for larger, more difficult instances of LWE is imperative in the determination of security estimates. We would like to consider these difficult instances, though, the floating-point calculations required for lattice reduction are quite restricting. A more thorough investigation of the projection lengths could help link the theoretical estimates to the results we are seeing in practice.

Floating-point division quantum circuits do exist [39]. However, the resource requirements of the mentioned divider are far more than when considering integer dividers [72]. We would like to be able to constuct a hand-optimized floating-point divider like the circuit in Chapter 4. A hand-optimized design could lead to far more efficient floating-point dividers.

APPENDICES

APPENDIX A

PROJECTQ IMPLEMENTATIONS

Here are some implementations of the component circuits disscussed in 4. The circuits are implemented in the circuit in the open-source software framework for quantum computing, ProjectQ [68, 36]

```
from projectq.ops import CNOT, Toffoli, C, Swap, Control
from projectq import MainEngine
from projectq.types import Qubit, Qureg


def ADD(x = Qureg(), y = Qureg(), z = Qureg()):
    #### [ x | y | 0 > —> [ x | x+y >
    n = len(x)
    for i in range(1,n): CNOT | (x[i],y[i])#1
    for i in range(n-1,0,-1):
        if i+1 == n : CNOT | (x[i], z)#2
        else: CNOT | (x[i], x[i+1])
    for i in range(n):
        if i == n-1: Toffoli | (y[i], x[i], z)#3
        else: Toffoli | (y[i], x[i], x[i+1])
    for i in range(n-1,0,-1): #4
        CNOT | (x[i],y[i])
        Toffoli | (y[i-1], x[i-1], x[i])
    for i in range(1, n-1): CNOT | (x[i], x[i+1])#5
    for i in range(n): CNOT | (x[i], y[i])#6
```

```
def SHIFT(s = Qureg(), x = Qureg()):
    #shift x to the left by s bits
```

```
    j = len(s)
    for k in range(j):
        i = 0
        while i + 2**k <= 2**j-1:
            if i+2**k < len(x):
                C(Swap,1) | (s[k], x[i], x[i + (2**k)])
            i += 1
```

```
def fixed_MUL(b = Qureg() ,a = Qureg(), z = Qureg(),
        eng = MainEngine()):
    #### [ b | a | z > --> [ b | a | a*b >
    n = len(b)
    for _ in range(n):
        with Control(eng, b[_]): ADD( a, z[_:n+_], z[n+_])
```

BIBLIOGRAPHY

[1] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. Solving the shortest vector problem in 2^n time using discrete gaussian sampling: Extended abstract. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 733–742. ACM, 2015.

[2] Miklós Ajtai. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.

[3] Miklós Ajtai. The shortest vector problem in l_2 is np-hard for randomized reductions. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(47), 1997.

[4] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In Dario Catalano and Roberto De Prisco, editors, *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, volume 11035 of *Lecture Notes in Computer Science*, pages 351–367. Springer, 2018.

[5] Martin R. Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-svp. In Hyang-Sook Lee and Dong-Guk Han, editors, *Information Security and Cryptology - ICISC 2013 - 16th International Conference, Seoul, Korea, November 27-29, 2013, Revised Selected Papers*, volume 8565 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2013.

[6] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to LWE. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 297–322. Springer, 2017.

[7] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, pages 327–343. USENIX Association, 2016.

[8] Andris Ambainis and Martins Kokainis. Quantum algorithm for tree size estimation, with applications to backtracking and 2-player games. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 989–1002, 2017.

[9] Matthew Amy, Dmitri Maslov, Michele Mosca, and Martin Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[10] Yoshinori Aono and Phong Q. Nguyen. Random sampling revisited: Lattice enumeration with discrete pruning. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 65–102, 2017.

[11] Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen. Quantum lattice enumeration and tweaking discrete pruning. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 405–434. Springer, 2018.

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

[13] Shi Bai, Shaun Miller, and Weiqiang Wen. A refined analysis of the cost for solving LWE via usvp. In Johannes Buchmann, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings*, volume 11627 of *Lecture Notes in Computer Science*, pages 181–205. Springer, 2019.

[14] Shi Bai, Damien Stehlé, and Weiqiang Wen. Improved reduction from the bounded distance decoding problem to the unique shortest vector problem in lattices. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 76:1–76:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[15] Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in BKZ. In Thomas Peyrin and Steven D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part I*,

volume 11272 of *Lecture Notes in Computer Science*, pages 369–404. Springer, 2018.

[16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 10–24. SIAM, 2016.

[17] Anja Becker, Nicolas Gama, and Antoine Joux. A sieve algorithm based on overlattices. *LMS Journal of Computation and Mathematics*, 17(A):49–70, 2014.

[18] Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1006–1018, 2016.

[19] Joppe W Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, G Seiler, and D Stehlé. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. *Cryptology ePrint Archive*, (20180716: 135545), 2017.

[20] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann. Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment. *CoRR*, abs/2006.06197, 2020.

[21] Yuanmi Chen. *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*. PhD thesis, Paris 7, 2013.

[22] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2011.

[23] Clifford C Cocks. A note on non-secret encryption. *CESG Memo*, 1973.

[24] Don Coppersmith. Finding a small root of a univariate modular equation. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165. Springer, 1996.

[25] Steven A Cuccaro, Thomas G Draper, Samuel A Kutin, and David Petrie Moulton. A new quantum ripple-carry addition circuit. *arXiv preprint quant-ph/0410184*, 2004.

[26] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

[27] The FPLLL development team. fplll, a lattice reduction library. Available at `https://github.com/fplll/fplll`, 2019.

[28] The FPYLLL development team. fpylll, a lattice reduction library. Available at `https://github.com/fplll/fplll`, 2019.

[29] TG DRAPER. Addition on a quantum computer. *quantph/0008033*, 2000.

[30] Austin G Fowler, Ashley M Stephens, and Peter Groszkowski. High-threshold universal quantum computation on the surface code. *Physical Review A*, 80(5):052312, 2009.

[31] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.

[32] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 257–278. Springer, 2010.

[33] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[34] Lov K Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325, 1997.

[35] Thomas Häner, Mathias Soeken, Martin Roetteler, and Krysta M. Svore. Quantum circuits for floating-point arithmetic. In Jarkko Kari and Irek Ulidowski, editors, *Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 162–174. Springer, 2018.

[36] Thomas Häner, Damian S. Steiger, Krysta M. Svore, and Matthias Troyer. A software methodology for compiling quantum programs. *Quantum Science and Technology*, 3(2):020501, 2018.

[37] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 447–464. Springer, 2011.

[38] Kenneth Hoffman and Ray Kunze. Linear algebra. 1971. *Englewood Cliffs, New Jersey*.

[39] Lafifa Jamal and Hafiz Md Hasan Babu. Efficient approaches to design a reversible floating point divider. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 3004–3007. IEEE, 2013.

[40] Antoine Joux. *Algorithmic cryptanalysis*. CRC press, 2009.

[41] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[42] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM (JACM)*, 52(5):789–808, 2005.

[43] Neal Koblitz, Alfred Menezes, and Scott A. Vanstone. The state of elliptic curve cryptography. *Des. Codes Cryptogr.*, 19(2/3):173–193, 2000.

[44] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2015.

[45] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Finding shortest lattice vectors faster using quantum search. *Des. Codes Cryptogr.*, 77(2-3):375–400, 2015.

[46] Hendrik Willem Lenstra, Arjen K Lenstra, L Lovfiasz, et al. Factoring polynomials with rational coeficients. 1982.

[47] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.

[48] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309. Springer, 2013.

[49] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2009.

[50] 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, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[51] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

[52] Robert J McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.

[53] Daniele Micciancio. Lattice based cryptography. In Henk C. A. van Tilborg, editor, *Encyclopedia of Cryptography and Security*. Springer, 2005.

[54] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer, 1985.

[55] H Minkowski. and (1910). *Geometric der Zahlen, Parts I and II. Teubner, Leipzig*, 1896.

[56] Ashley Montanaro. Quantum-walk speedup of backtracking algorithms. *Theory Comput.*, 14(1):1–24, 2018.

[57] Phong Q Nguyen and Damien Stehlé. An LLL algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.

[58] Phong Q Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008.

[59] Trung Duc Nguyen and Rodney Van Meter. A resource-efficient design for a reversible floating point adder in quantum computing. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 11(2):1–18, 2014.

[60] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[61] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342. ACM, 2009.

[62] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[63] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[64] Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*, volume 2607 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2003.

[65] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.

[66] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[67] IEEE std.754 2019. *IEEE Standard for Floating-Point Arithmetic*. 2019.

[68] Damian S. Steiger, Thomas Häner, and Matthias Troyer. ProjectQ: an open source software framework for quantum computing. *Quantum*, 2:49, 2018.

[69] Douglas Robert Stinson and Maura Paterson. *Cryptography: theory and practice*. CRC press, 2018.

[70] Gilbert Strang. Linear algebra and its applications, (1988). *Hartcourt Brace Jovanovich College Publishers*, 1988.

[71] Yasuhiro Takahashi, Seiichiro Tani, and Noboru Kunihiro. Quantum addition circuits and unbounded fan-out. *Quantum Inf. Comput.*, 10(9&10):872–890, 2010.

[72] Himanshu Thapliyal, Edgard Munoz-Coreas, T.S.S. Varun, and Travis S. Humble. Quantum circuit designs of integer division optimizing T-count and T-depth. *IEEE Transactions on Emerging Topics in Computing*, 2019.

[73] Marcel Tiepelt and Alan Szepieniec. Quantum LLL with an application to mersenne number cryptosystems. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2019.

[74] Peter van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. *Tecnical Report, Department of Mathmatics, University of Amsterdam*, 1981.

[75] Panagiotis Voulgaris and Daniele Micciancio. Faster exponential time algorithms for the shortest vector problem. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:65, 2009.

[76] Yuntao Wang, Yoshinori Aono, and Tsuyoshi Takagi. Hardness evaluation for search LWE problem using progressive BKZ simulator. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 101(12):2162–2170, 2018.