On the Security of Lattice-Based Cryptography Against Lattice Reduction and Hybrid Attacks

Vom Fachbereich Informatik der

Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades

Doktor rerum naturalium (Dr. rer. nat.)

von

Dipl.-Ing. Thomas Wunderer

geboren in Augsburg.

Referenten:	Prof. Dr. Johannes Buchmann
	Dr. Martin Albrecht
Tag der Einreichung:	08. 08. 2018
Tag der mündlichen Prüfung:	20. 09. 2018
Hochschulkennziffer:	D 17

Abstract

Over the past decade, lattice-based cryptography has emerged as one of the most promising candidates for post-quantum public-key cryptography. For most current lattice-based schemes, one can recover the secret key by solving a corresponding instance of the unique Shortest Vector Problem (uSVP), the problem of finding a shortest non-zero vector in a lattice which is unusually short.

This work is concerned with the concrete hardness of the uSVP. In particular, we study the uSVP in general as well as instances of the problem with particularly small or sparse short vectors, which are used in cryptographic constructions to increase their efficiency.

We study solving the uSVP in general via lattice reduction, more precisely, the Block-wise Korkine-Zolotarev (BKZ) algorithm. In order to solve an instance of the uSVP via BKZ, the applied block size, which specifies the BKZ algorithm, needs to be sufficiently large. However, a larger block size results in higher runtimes of the algorithm. It is therefore of utmost interest to determine the minimal block size that guarantees the success of solving the uSVP via BKZ. In this thesis, we provide a theoretical and experimental validation of a success condition for BKZ when solving the uSVP which can be used to determine the minimal required block size. We further study the practical implications of using so-called sparsification techniques in combination with the above approach.

With respect to uSVP instances with particularly small or sparse short vectors, we investigate so-called hybrid attacks. We first adapt the "hybrid lattice reduction and meet-in-the-middle attack" (or short: the hybrid attack) by Howgrave-Graham on the NTRU encryption scheme to the uSVP. Due to this adaption, the attack can be applied to a larger class of lattice-based cryptosystems. In addition, we enhance the runtime analysis of the attack, e.g., by an explicit calculation of the involved success probabilities. As a next step, we improve the hybrid attack in two directions as described in the following.

To reflect the potential of a modern attacker on classical computers, we show how to parallelize the attack. We show that our parallel version of the hybrid attack scales well within realistic parameter ranges. Our theoretical analysis is supported by practical experiments, using our implementation of the parallel hybrid attack which employs Open Multi-Processing and the Message Passing Interface.

To reflect the power of a potential future attacker who has access to a large-scale quantum computer, we develop a quantum version of the hybrid attack which replaces the classical meet-in-the-middle search by a quantum search. Not only is the quantum hybrid attack faster than its classical counterpart, but also applicable to a wider range of uSVP instances (and hence to a larger number of lattice-based schemes) as it uses a quantum search which is sensitive to the distribution on the search space.

Finally, we demonstrate the practical relevance of our results by using the techniques developed in this thesis to evaluate the concrete security levels of the lattice-based schemes submitted to the US National Institute of Standards and Technology's process of standardizing post-quantum public-key cryptography.

Publications

Publications used in this thesis

[1] Johannes A. Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the Hardness of LWE with Binary Error: Revisiting the Hybrid Lattice-Reduction and Meet-in-the-Middle Attack. In: Progress in Cryptology - AFRICACRYPT 2016 - 8th International Conference on Cryptology in Africa, Fes, Morocco, April 13-15, 2016, Proceedings. 2016, pp. 24-43.

[2] Thomas Wunderer. A Detailed Analysis of the Hybrid Lattice-Reduction and Meet-in-the-Middle Attack. In: Journal of Mathematical Cryptology, to appear.

[3] Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. A Hybrid Lattice Basis Reduction and Quantum Search Attack on LWE. In: Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings. 2017, pp. 184-202.

[4] Martin R. Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the Expected Cost of Solving uSVP and Applications to LWE. In: 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. 2017, pp. 297-322.

[5] 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: Security and Cryptography for Networks – 11th International Conference, SCN 2018, Amalfi, Italy, September 5 - September 7, 2018, Proceedings. Lecture Notes in Computer Science, Springer 2018, to appear.

[6] Yuntao Wang and Thomas Wunderer. Revisiting the Sparsification Technique in Kannan's Embedding Attack on LWE. In: Information Security Practice and Experience – 14th International Conference, ISPEC 2018, Tokyo, Japan,

September 25-27, 2018, Proceedings. Lecture Notes in Computer Science, Springer 2018, to appear.

[7] Martin R. Albrecht, Benjamin R. Curtis, and Thomas Wunderer. An Exploration of the Hybrid Attack on Small-secret LWE. Work in progress.

[8] Thomas Wunderer, Michael Burger, and Giang Nam Nguyen. Parallelizing the Hybrid Lattice Reduction and Meet-in-the-Middle Attack. In: CSE-2018 – 21st IEEE International Conference on Computational Science and Engineering, Bucharest, Romania, October 29 - 31, 2018, to appear.

Other publications

[9] Patrick Holzer, Thomas Wunderer, and Johannes A. Buchmann. Recovering Short Generators of Principal Fractional Ideals in Cyclotomic Fields of Conductor $p^\alpha q^\beta$. In: Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings. 2017, pp. 346-368.

[10] Michael Burger, Christian Bischof, Alexandru Calotoiu, Thomas Wunderer, and Felix Wolf. Exploring the Performance Envelope of the LLL Algorithm. In: CSE-2018 – 21st IEEE International Conference on Computational Science and Engineering, Bucharest, Romania, October 29 - 31, 2018, to appear.

Contents

1 Introduction

Public-key cryptography. In our modern world, billions of internet connections are protected by Public-Key Cryptography (PKC) every day. To guarantee the effectiveness of this protection, PKC is required to be secure against attacks. Currently, the security of virtually all PKC algorithms that are used in practice today is based on number-theoretic problems such as the integer factorization problem or the discrete logarithm problem. However, as shown by Peter Shor [Sho97], the integer factorization problem and the discrete logarithm problem can be solved in polynomial time on quantum computers, rendering virtually all of today's PKC algorithms insecure in a world where large-scale quantum computers exist. While currently only small-scale quantum computers exist, recent advances in technology and engineering suggest that it is not implausible that a large-scale quantum computer which can break current PKC algorithms can be built within the next one or two decades [Mos15].

Post-quantum and lattice-based cryptography. This threat has resulted in a search for alternative PKC algorithms that withstand quantum attacks, called post-quantum cryptography [BBD09, JF11]. The urgency of developing and deploying post-quantum PKC has been recognized by the US National Institute of Standards and Technology (NIST) in 2015, when they inidiated the process of standardizing post-quantum public-key encryption schemes, key encapsulation mechanisms, and digital signature algorithms, resulting in a call for proposals in 2016 [Nat16]. The received submissions can be categorized into different classes, including lattice-based, hash-based, code-based, isogeny-based, and multivariate cryptography. With roughly a third of the submissions, lattice-based cryptography is the largest of the above categories. The history of lattice-based cryptography [Pei16a] started over a decade ago and since then, it has developed into one of the most promising candidates for post-quantum cryptography due to its high efficiency and wealth of applications, ranging from basic PKC algorithms such as [HPS98, Reg09, LP11, ADPS16, BG14a] to cryptographic primitives with enhanced functionality such as fully homomorphic encryption [BV11, GSW13] or obfuscation of some families of circuits [BVWW16].

Cryptanalysis of lattice-based cryptography. The security of lattice-based cryptosystems is based on the presumed hardness of lattice problems such as the Learning

With Errors (LWE) problem, the Short Integer Solution (SIS) problem, their corresponding ring or module variants, or the NTRU problem. In more detail, if a lattice-based scheme is provided with a security reduction, being able to break the scheme (e.g., recover the secret key) implies that one can efficiently solve the underlying lattice problem. To analyze the security of lattice-based schemes, more concretely to determine their security levels, it is therefore important to analyze the hardness of the above-mentioned lattice problems arising in cryptography. Solving such lattice problems, and hence breaking lattice-based schemes, can typically be reduced to solving an instance of the unique Shortest Vector Problem (uSVP), the problem of finding an unusually short shortest non-zero vector in a lattice. For instance, in the case of LWE this can be done via Kannan's [Kan87] or Bai and Galbraith's [BG14b] embedding, which is often referred to as the primal (lattice) attack. One of the most common and efficient general approaches to solve uSVP is via lattice reduction [LLL82, Sch87, GN08a, HPS11, CN11, MW16]. In addition to studying this general approach, it is also important to consider specific attacks for special instantiations of the uSVP, as argued in the following. In order to increase the efficiency of lattice-based PKC, in particular in the context of fully homomorphic encryption, variants of lattice problems with small and/or sparse short vectors have been introduced. Using such instances in cryptographic constructions can reduce the execution time (e.g., due to faster arithmetic or sampling algorithms) and key sizes. These instances, however, might be vulnerable to specialized attacks. For instance, if the shortest non-zero vector of a uSVP instance is particularly small and/or sparse, one can combine lattice reduction with combinatorial techniques in so-called hybrid attacks.

1.1 Contribution and Organization

In this work, we answer the following research questions. What is the cost of solving the uSVP using lattice reduction? How can one decrease this cost for special instances of the uSVP by combining lattice reduction with combinatorial techniques? Can one further improve such algorithms by using parallel or quantum computing? And last but not least, how do the developed techniques influence security estimates for cryptographic schemes?

We focus on solving the uSVP, as most cryptographic lattice problems can be transformed into a uSVP instance, and apply our results to various LWE- and NTRU-based cryptosystems. We consider algorithms to solve uSVP instances in general as well as hybrid algorithms that are designed to perform better on uSVP instances with small and/or sparse short vectors.

To study the uSVP in general, we examine the cost of the Block-wise Korkine-Zolotarev (BKZ) [Sch87] or BKZ 2.0 [CN11] lattice reduction algorithms for solving the uSVP. In more detail, the BKZ algorithm is specified by a block size, which is the main factor in determining the algorithm's runtime. To be more precise, applying a bigger block size results in a higher runtime of the algorithm and current research suggests that the runtime increases exponentially with the block size. It is therefore desirable to apply a block size which is as small as possible. However, using a block size which is too small, BKZ is not expected to be successful in solving the uSVP. In order to solve the uSVP as efficiently as possible, it is therefore essential to determine the minimal block size that guarantees success. In the current literature, there exist two different estimates to determine the minimal block size, which we call the 2008 estimate [GN08b] and the 2016 estimate [ADPS16], predicting vastly different results. The 2008 estimate has been used for years to estimate the security of many lattice-based cryptosystems (e.g., [BG14a, CHK$^+$17, CKLS16a, CLP17, ABB$^+$17]), but its validity is based on experiments in rather small dimensions, which may not be representative for cryptographic applications. The recently introduced 2016 estimate on the other hand has not yet been examined at all. In this work, we provide a detailed theoretical and experimental analysis of the 2016 estimate. Under standard lattice assumptions, we show that if the block size satisfies the 2016 estimate, BKZ recovers a projection of the uSVP solution from which the so-called size reduction subroutine recovers the entire solution. We further provide practical experiments performed in medium to large block sizes. Our results validate the 2016 estimate, answering the important question about the minimal block size required to solve the uSVP via BKZ. In addition, we apply our results to show that several security estimates in the literature based on the old estimate need to be revised.

Using our above-mentioned results, we investigate the practical implications of using sparsification techniques [Kho03, Kho04, DK13, DRS14, SD16] when embedding lattice problems into uSVP instances. The use of sparsification techniques has been proposed in the context of theoretical reductions from lattice problems to the uSVP [BSW16], but has not yet been studied from a practical, cryptanalytic point of view. We show that, while these techniques yield improved theoretical reductions, in general they do not lead to better attacks in practice. To draw this conclusion, we show that for reasonable parameters the expected speedup gained by sparsification techniques under the 2016 estimate is not sufficient to compensate for the small success probability introduced by these techniques.

After having considered these general approaches to solve the uSVP, we focus on hybrid attacks designed to perform better on small and/or sparse instances of the uSVP. We first adapt the "hybrid lattice reduction and meet-in-the-middle attack" [HG07] (short: the hybrid attack) on the NTRU encryption scheme [HPS98] to a more general framework which applies to solving the uSVP, and hence most lattice-based cryptosystems. The hybrid attack provides a trade-off between lattice techniques such as lattice reduction and combinatorial techniques, i.e., a meet-in-the-middle search, and is currently considered the best attack on NTRU [HG07, HHGP$^+$07, HHHGW09, HPS$^+$17, Sch15]. Besides adapting the attack to a uSVP framework, which enables to apply the attack to a broader class of cryptosystems, our main contribution is to provide an improved analysis of the hybrid attack. While previous analyses suffer from using unnecessary and oversimplifying assumptions, such as ignoring or simplifying success probabilities, our analysis is based on reasonable assumptions. One of the most important of our improvements is an explicit calculation of the collision-finding probabilities in the meet-in-the-middle search. Furthermore, we apply our improved analysis to reevaluate the security levels of several lattice-based cryptosystems against the hybrid attack. We compare our results to the 2016 estimate to showcase the improvement of the hybrid attack over a generic lattice attack in the case of particularly small and/or sparse short vectors.

As a next step, in order to reflect the full potential of a powerful attacker on classical computers, we show how to parallelize the hybrid attack. We introduce parallelization to the attack in three different ways. First, we run multiple randomized attacks in parallel to reduce the runtime of the entire attack. Second, we perform the meet-in-the-middle search in parallel to speed up the search phase of the attack. Third, the BKZ precomputation can potentially be run in parallel if a parallel implementation of BKZ is available. Our theoretical analysis shows that our parallel hybrid attack scales well withing realistic parameter ranges. We support our theoretical considerations with practical experiments, employing OpenMP and the Message Passing Interface in our implementation. Our experiments confirm that running multiple instances of the attacks in parallel significantly reduces the overall runtime and show that our parallel meet-in-the-middle search scales very well.

Next, we develop a quantum version of the hybrid attack, using a generalization of Grover's quantum search algorithm [Gro96] by Brassard et al. [BHMT02]. Our quantum hybrid attack is not only faster and more versatile (i.e., applicable to a wider range of lattice-based cryptosystems) than its classical counterpart, but also eliminates the problems of large memory requirements and low collision-finding probabilities in the classical meet-in-the-middle search. We show how to minimize the runtime of the quantum hybrid attack by optimizing the quantum search and the attack parameters. In addition, we discuss techniques that can be used to further improve the attack. We demonstrate our improvements by applying the quantum hybrid attack to various uSVP instances. We compare our results to the classical hybrid attack and the general approach of solving the uSVP using lattice reduction under the 2016 estimate, highlighting the improvements of the quantum hybrid attack for small and/or sparse instances of the uSVP.

Finally, we analyze the security of the lattice-based schemes accepted to NIST's standardization process, highlighting the importance of this work. In their submissions, the authors were asked to estimate the security of their schemes. However, the applied methods among the different submissions are not uniform, making it hard to compare the security levels of different schemes. We provide security estimates for all LWE- or NTRU-based NIST candidates against the primal attack under the 2016 estimate, using all proposed cost models for lattice reduction. This enables a fair comparison of the security levels of the different schemes. In addition, we analyze selected schemes with respect to the quantum hybrid attack, which, depending on the applied cost model for lattice reduction, yields significantly lower costs.

Organization. This thesis is structured as follows.

Chapter 2: this chapter presents all the necessary notation and mathematical background on lattices, lattice problems, and lattice algorithms and summarizes some related work.

Chapter 3: this chapter provides theoretical and experimental evidence for the validity of a recently proposed [ADPS16] (but not yet studied) success condition for solving the uSVP using the BKZ lattice reduction algorithm. This success condition determines the security level of most lattice-based cryptosystems.

Chapter 4: this chapter studies the practical influence of using sparsification techniques when embedding lattice problems into an instance of uSVP as suggested in the context of a theoretical reduction in [BSW16].

Chapter 5: this chapter provides a uSVP framework for the hybrid lattice reduction and meet-in-the-middle attack [HG07] and an improved runtime analysis of the attack which can be used to derive security estimates for several lattice-based cryptosystems.

Chapter 6: this chapter shows how the hybrid attack can be parallelized and examines the obtained speedup both theoretically and experimentally.

Chapter 7: this chapter develops an improved quantum version of the hybrid attack which compared to its classical counterpart is faster and applicable to a wider class of uSVP instances.

Chapter 8: this chapter analyzes the security of lattice-based schemes accepted to NIST's standardization process [Nat16] with respect to the primal attack under the 2016 estimate and the quantum hybrid attack.

Chapter 9: this chapter concludes this work and states some research questions that remain open for future work.

2 Background

In this chapter, we provide the background necessary for this work, following and unifying the preliminaries of the author's publications used in this thesis.

2.1 Notation

Throughout this work, vectors are denoted in bold lowercase letters, e.g., $\mathbf{a}$, and matrices in bold uppercase letters, e.g., $\mathbf{A}$. Polynomials are written in normal lower case letters, e.g., a. We frequently identify polynomials $a = \sum_{i=0}^{n} a_i x^i$ with their coefficient vectors $\mathbf{a} = (a_0, \ldots, a_n)$, indicated by using the corresponding bold letter. We use the notation $\mathbb{Z}_q$ for the quotient ring $\mathbb{Z}/q\mathbb{Z}$. By $\mathbf{a} \bmod q$ we indicate that each component of the vector is reduced modulo q to lie in the interval $[-\left\lceil \frac{q}{2} \right\rceil, \frac{q}{2})$. Let $n, q \in \mathbb{N}$, $f \in \mathbb{Z}[x]$ be a polynomial of degree n, and $R_q = \mathbb{Z}_q[x]/(f)$. We define the *rotation matrix* of a polynomial $a \in R_q$ as $\mathsf{rot}(a) = (\mathbf{a}, \mathbf{ax}, \mathbf{ax}^2, \ldots, \mathbf{ax}^{n-1}) \in \mathbb{Z}_q^{n \times n}$, where $\mathbf{ax}^i$ denotes the coefficient vector of the polynomial ax^i. Then for $a, b \in R_q$, the matrix-vector product $\mathsf{rot}(a) \cdot \mathbf{b} \mod q$ corresponds to the product of polynomials $ab \in R_q$.

We write $\langle \cdot, \cdot \rangle$ for the inner products and $\cdot$ for matrix-vector products. By abuse of notation we consider vectors to be row resp. column vectors depending on context, such that $\mathbf{v} \cdot \mathbf{A}$ and may $\mathbf{A} \cdot \mathbf{v}$ are meaningful, and omit indicating that vectors are transposed. We write $\mathbf{I}_m$ for the $m \times m$ identity matrix over whichever base ring is implied from context. We write $\mathbf{0}_{m \times n}$ for the $m \times n$ all zero matrix. If the dimensions are clear from the context, we may omit the subscripts. We use the abbreviation $\log(\cdot)$ for $\log_2(\cdot)$. We further write $\|\cdot\|$ instead of $\|\cdot\|_2$ for the Euclidean norm. For a vector $\mathbf{v}$, its *Hamming weight* is defined as the number of non-zero entries. For $N \in \mathbb{N}_0$ and $m_1, \ldots, m_k \in \mathbb{N}_0$ with $m_1 + \ldots + m_k = N$ the multinomial coefficient is defined as

$$\binom{N}{m_1, \ldots, m_k} = \frac{N!}{m_1! \cdot \ldots \cdot m_k!}.$$

For a probability distribution X, we write $x \xleftarrow{\$} X$ if an element x is sampled according to X. For every element a in the support of X, we write $x_a := \Pr[a = b | b \xleftarrow{\$} X]$. We will specifically refer to the discrete Gaussian distribution D_σ as the distribution such that

$$\forall y \in \mathbb{Z} : \Pr[x = y | x \xleftarrow{\$} D_\sigma] \sim \exp\left(-\frac{y^2}{2\sigma^2}\right).$$

For a probabilistic algorithm $\mathcal{A}$, $x \xleftarrow{\$} \mathcal{A}$ assigns the outcome of one (random) run of $\mathcal{A}$ to x.

2.2 Lattices and Lattice Bases

In this work, we use the following definition of lattices. A discrete additive subgroup of $\mathbb{R}^d$ for some $d \in \mathbb{N}$ is called a *lattice*. In this case, d is called the *dimension* of the lattice. Let d be a positive integer. For a set of vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \subset \mathbb{R}^d$, the lattice spanned by $\mathbf{B}$ is defined as

$$\Lambda(\mathbf{B}) = \left\{ \mathbf{x} \in \mathbb{R}^d \mid \mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \text{ for } \alpha_i \in \mathbb{Z} \right\}.$$

Let $\Lambda \subset \mathbb{R}^d$ be a lattice. A set of vectors $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\} \subset \mathbb{R}^d$ is called a basis of Λ if $\mathbf{B}$ is $\mathbb{R}$-linearly independent and $\Lambda = \Lambda(\mathbf{B})$. Abusing notation, we identify lattice bases with matrices and vice versa by taking the basis vectors as the columns of the matrix. The number of vectors in a basis of a lattice is called the *rank* of the lattice. A lattice $\Lambda \subset \mathbb{R}^d$ is called a *full-rank* lattice if its rank is equal to the dimension d. In this case, every basis matrix of Λ is a square $d \times d$ matrix. For a point $\mathbf{t} \in \mathbb{R}^d$ and a lattice $\Lambda \subset \mathbb{R}^d$ we define the *distance* from $\mathbf{t}$ to the lattice as $\mathsf{dist}(\mathbf{t}, \Lambda) = \min_{\mathbf{x} \in \Lambda} \|\mathbf{t} - \mathbf{x}\|$. Note that the minimum exists as a lattice is a discrete set. For a lattice basis $\mathbf{B} = \{\mathbf{b}_1, ..., \mathbf{b}_n\}$ the corresponding *Gram-Schmidt basis* $\mathbf{B}^* = \{\mathbf{b}_1^*, ..., \mathbf{b}_n^*\}$ is defined as follows.

- Set $\mathbf{b}_1^* = \mathbf{b}_1$.
- For $j = 2, \ldots, n$, iteratively set

$$\mathbf{b}_j^* = \pi_j(\mathbf{b}_j) = \mathbf{b}_j - \sum_{k=1}^{j-1} \frac{\langle \mathbf{b}_j, \mathbf{b}_k^* \rangle}{\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle} \cdot \mathbf{b}_k^*.$$

Let q be a positive integer. An integer lattice $\Lambda \subset \mathbb{Z}^d$ that contains $q\mathbb{Z}^d$ is called a *q-ary* lattice. Note that every q-ary lattice is full-rank as it contains the full-rank lattice $q\mathbb{Z}^d$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{d \times n}$, we define the q-ary lattice spanned by $\mathbf{A}$ as

$$\Lambda_q(\mathbf{A}) := \{\mathbf{v} \in \mathbb{Z}^d \mid \exists \mathbf{w} \in \mathbb{Z}^n : \mathbf{A}\mathbf{w} = \mathbf{v} \mod q\}.$$

For a lattice basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\} \subset \mathbb{R}^d$ of a rank-n lattice its (centered) *fundamental parallelepiped* is defined as

$$\mathcal{P}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \mid -1/2 \leq \alpha_i < 1/2 \text{ for all } i \in \{1, \ldots, n\} \right\}.$$

The *determinant* $\det(\Lambda)$ of a lattice $\Lambda \subset \mathbb{R}^d$ of rank n, also called its *(co-)volume*, is defined as the n-dimensional volume of the fundamental parallelepiped of a basis of Λ, i.e., $\det(\Lambda) = \sqrt{\det(\mathbf{B}^T\mathbf{B})}$. Note that the determinant of the lattice is well defined, i.e., it is independent of the basis. For a full-rank lattice Λ of rank d, the determinant of the lattice $\det(\Lambda)$ is the absolute value of the determinant of any basis $\mathbf{B}$ and it holds that $\det(\Lambda) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$. For two full-rank lattices $\Lambda' \subset \Lambda$ it holds that $[\Lambda : \Lambda'] = \det(\Lambda') / \det(\Lambda)$. In particular, if $\Lambda' \subset \Lambda \subset \mathbb{Z}^d$ are full-rank integer lattices it holds that $\det(\Lambda) \mid \det(\Lambda')$. We write $\lambda_i(\Lambda)$ for *Minkowski's successive minima*, i.e., the radius of the smallest ball centered around zero containing i linearly independent lattice vectors. In particular, the length of the shortest non-zero vectors of a lattice Λ is denoted by $\lambda_1(\Lambda)$. For a full-rank lattice $\Lambda \subset \mathbb{R}^d$ the *Gaussian Heuristic* predicts

$$\lambda_1(\Lambda) \approx \sqrt{\frac{d}{2\pi e}} \det(\Lambda)^{1/d}.$$

For a lattice basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ and for $i \in \{1, \ldots, n\}$ let $\pi_{\mathbf{B},i}(\mathbf{v})$ denote the orthogonal projection of $\mathbf{v}$ onto $\{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\}$, where $\pi_{\mathbf{B},1}$ is the identity. We extend the notation to sets of vectors in the natural way. Since usually the basis $\mathbf{B}$ is clear from the context, we omit it in the notation and simply write π_i instead of $\pi_{\mathbf{B},i}$. A basis is called *size reduced* if it satisfies the following definition. An algorithm that size reduced a basis is recalled in Algorithm 1.

Definition 2.1. *Let* $\mathbf{B}$ *be a basis,* $\mathbf{b}_i^*$ *its Gram-Schmidt vectors and*

$$\mu_{i,j} = \left\langle \mathbf{b}_i, \mathbf{b}_j^* \right\rangle / \left\langle \mathbf{b}_j^*, \mathbf{b}_j^* \right\rangle.$$

Then the basis $\mathbf{B}$ *is called* size reduced *if* $|\mu_{i,j}| \leq 1/2$ *for* $1 \leq j \leq i \leq n$.

Algorithm 1: Size reduction

```
Input : lattice basis $\mathbf{B}$, top index $i$, start index $1 \leq s < i$
1 for $j$ from $i-1$ to $s$ do
2     $\mu_{ij} \leftarrow \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle$;
3     $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor \mu_{ij} \rceil \mathbf{b}_j$;
```

2.3 Lattice Problems

Lattice-based cryptography is based on the presumed hardness of computational problems in lattices. In the following we describe the important lattice problems relevant for this work.

2.3.1 Shortest Vector Problems

One of the most fundamental and most studied lattice problems is the *Shortest Vector Problem (SVP).*

Definition 2.2. *(SVP) Given a lattice basis* $\mathbf{B}$*, the task is to find a shortest non-zero vector in the lattice* $\Lambda(\mathbf{B})$.

An important variant of the SVP in the context of lattice-based cryptography is the *unique Shortest Vector Problem (uSVP)*, where one is given the promise that the shortest non-zero vector is uniquely short.

Definition 2.3. *(uSVP*$_\gamma$*) Given a gap* $\gamma \geq 1$ *and a lattice* Λ *with* $\lambda_2(\Lambda) \geq \gamma\lambda_1(\Lambda)$*, find a shortest non-zero lattice vector in* Λ.

2.3.2 Closest Vector Problems

Besides finding short vectors in lattices, an important computational problem is to find lattice vectors that are close to some target vectors in space. This is called the *Closest Vector Problem (CVP).*

Definition 2.4. *(CVP) Given a full-rank lattice* $\Lambda \subset \mathbb{R}^d$ *and a target point* $\mathbf{t} \in \mathbb{R}^d$*, find a lattice vector* $\mathbf{x} \in \Lambda$ *with* $\|\mathbf{t} - \mathbf{x}\| = \mathsf{dist}(\mathbf{t}, \Lambda)$.

A variant of the closest vector problem relevant in lattice-based cryptography it the *Bounded Distance Decoding (BDD)* problem.

Definition 2.5. *(BDD*$_\alpha$*) Given* $0 < \alpha \leq 1/2$*, a full-rank lattice* $\Lambda \subset \mathbb{R}^d$*, and a target point* $\mathbf{t} \in \mathbb{R}^d$ *with* $\mathsf{dist}(\mathbf{t}, \Lambda) < \alpha\lambda_1(\Lambda)$*, find the unique lattice vector* $\mathbf{v} \in \Lambda$ *such that* $\|\mathbf{t} - \mathbf{v}\| < \alpha\lambda_1(\Lambda)$.

2.3.3 Learning with Errors

The *Learning With Errors* (LWE) problem is defined as follows.

Definition 2.6 (LWE [Reg09]). *Let* n, q *be positive integers,* χ *be a probability distribution on* $\mathbb{Z}$ *and* $\mathbf{s}$ *be a secret vector in* $\mathbb{Z}_q^n$*. We denote by* $L_{\mathbf{s},\chi}$ *the probability distribution on* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *obtained by choosing* $\mathbf{a} \in \mathbb{Z}_q^n$ *uniformly at random, choosing*

$e \in \mathbb{Z}$ according to χ and considering it in $\mathbb{Z}_q$, and returning $(\mathbf{a}, b) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.
Decision-LWE *is the problem of, given (arbitrarily many) pairs $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ that are either all sampled independently according to $L_{\mathbf{s},\chi}$ or the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$, deciding which is the case.*
Search-LWE *is the problem of recovering* $\mathbf{s}$ *from (arbitrarily many) independent samples $(\mathbf{a}_i, b_i) = (\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\chi}$.*

We may write LWE instances in matrix form $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$, where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{b} \in \mathbb{Z}_q^m$ and rows correspond to samples $(\mathbf{a}_i, b_i)$ for some number of samples m. In many instantiations, χ is a discrete Gaussian distribution with standard deviation σ. In the discrete Gaussian case with standard deviation σ, we expect the error vector $\mathbf{e}$ to have length approximately $\|\mathbf{e}\| \approx \sqrt{m}\sigma$. Note that the attacker can choose a number of samples that is optimal for the applied attack. In typical cryptographic settings, however, the number of provided samples is not unlimited but bounded, e.g., by the secret dimension n or by $2n$. In this case, the bound needs to be respected when an attacker chooses their number of samples.

Related problems. Based on the concept of LWE, related problems with additional algebraic structure have been proposed. In particular, in the *Ring-LWE* [SSTX09, LPR10] (RLWE) problem polynomials s, a_i and e_i (where s and e_i are “short”) are drawn from a ring of the form $\mathcal{R}_q = \mathbb{Z}_q[x]/(\phi)$ for some polynomial ϕ of degree n. Then, given a list of Ring-LWE samples $\{(a_i, a_i \cdot s + e_i)\}_{i=1}^m$, the Search-RLWE problem is to recover s and the Decision-RLWE problem is to distinguish the list of samples from a list uniformly sampled from $\mathcal{R}_q \times \mathcal{R}_q$. More generally, in the *Module-LWE* [LS15] (MLWE) problem vectors (of polynomials) $\mathbf{a}_i$, $\mathbf{s}$ and polynomials e_i are drawn from $\mathcal{R}_q^k$ and $\mathcal{R}_q$ respectively. Search-MLWE is the problem of recovering $\mathbf{s}$ from a set $\{(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_{i=1}^m$, Decision-MLWE is the problem of distinguishing such a set from a set uniformly sampled from $\mathcal{R}_q^k \times \mathcal{R}_q$.

One can view RLWE and MLWE instances as LWE instances by interpreting the coefficients of elements in $\mathcal{R}_q$ as vectors in $\mathbb{Z}_q^n$ and ignoring the algebraic structure of $\mathcal{R}_q$. This identification with LWE is the standard approach for estimating the concrete hardness of solving RLWE and MLWE due to the absence of known cryptanalytic techniques exploiting algebraic structure.

One can also define LWE-like problems by replacing the addition of the error term by a deterministic rounding process. For instance, the *Learning With Rounding (LWR)* problem is of the form $\left(\mathbf{a}, b := \left\lfloor \frac{p}{q} \langle \mathbf{a}, \mathbf{s} \rangle \right\rceil\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$ for some moduli p and q. We can interpret such an instance as an LWE instance by multiplying the second component by q/p and assuming that $q/p \cdot b = \langle \mathbf{a}, \mathbf{s} \rangle + e$, where e is uniformly chosen from the interval $(-q/2p, q/2p]$ [BPR12]. The resulting variance of this error term can then be calculated as $\frac{(q/p)^2-1}{12}$, following [Ngu18]. Analogously, the same applies to RLWR- and MLWR-like instances that use deterministic rounding instead of adding an error term.

2.3.4 NTRU

The *NTRU* problem is the foundation of the NTRU encryption scheme [HPS96] and following encryption (e.g., [SHRS17, BCLvV17a]) and signature (e.g., [ZCHW17b, PFH$^+$17]) schemes.

Definition 2.7 (NTRU [HPS96])**.** *Let n, q be positive integers, $\phi \in \mathbb{Z}[x]$ be a monic polynomial of degree n, and $\mathcal{R}_q = \mathbb{Z}_q[x]/(\phi)$. Let $f \in \mathcal{R}_q^\times, g \in \mathcal{R}_q$ be small polynomials (i.e., having small coefficients) and $h = g \cdot f^{-1} \bmod q$.*

Search-NTRU *is the problem of recovering f or g given h.*

Remark 2.1. *One can exchange the roles of f and g (in the case that g is invertible) by replacing h with $h^{-1} = f \cdot g^{-1} \bmod q$, if this leads to a better attack.*

The most common ways to choose the polynomials f (or g) are the following. The first is to choose f to have small coefficients (e.g., ternary). The second is to choose F to have small coefficients (e.g., ternary) and to set $f = pF$ for some (small) prime p. The third is to choose F to have small coefficients (e.g., ternary) and to set $f = pF + 1$ for some (small) prime p.

The NTRU problem can be reduced to solving (a variant[1] of) the uSVP in the NTRU lattice $\Lambda(\mathbf{B})$ generated by the columns of

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_n & \mathbf{H} \\ \mathbf{0} & \mathbf{I}_n \end{pmatrix},$$

where $\mathbf{H}$ is the rotation matrix of h, see for example [CS97, HPS98]. Indeed, $\Lambda(\mathbf{B})$ contains the short vector $(\mathbf{f} \,|\, \mathbf{g})$, since $hf = g \bmod q$ and hence $(\mathbf{f} \,|\, \mathbf{g}) = \mathbf{B}(\mathbf{w} \,|\, \mathbf{g})$ for some $\mathbf{w} \in \mathbb{Z}^n$. Furthermore, it can be assumed that the vector $(\mathbf{f} \,|\, \mathbf{g})^t$ and its rotations (and theirs additive inverses) are uniquely short vectors in $\Lambda(\mathbf{B})$. In addition, if $f = pF$ or $f = pF + 1$ for some small polynomial F one can construct a similar uSVP lattice that contains $(\mathbf{F} \,|\, \mathbf{g})$, see for example [Sch15]. Similar to LWE, in order to improve this attack, rescaling (see Section 3.3.1 for more details) and dimension reducing techniques can be applied [MS01]. Dimension reducing techniques resemble choosing the number of samples in LWE. Note that the dimension of the lattice must be between n and $2n$ by construction.

[1] Note that the NTRU lattice contains $(\mathbf{f} \,|\, \mathbf{g})^t$ and all its rotations $(\mathbf{f}\mathbf{X}^i \,|\, \mathbf{g}\mathbf{X}^i)^t$, hence possibly n linearly independent unusually short vectors, which is not the case in the standard definition of uSVP and can possibly be exploited.

2.4 Lattice Algorithms

In this section, we summarize the lattice algorithms that are relevant for this work and their behavior. We start by giving a short exposition about heuristic runtime estimates and their relevance in lattice-based cryptography compared to mathematically rigorous statements.

2.4.1 Runtime Estimates

This work is concerned with the security of lattice-based schemes and for that matter with the concrete hardness of lattice problems, in particular (variants of) the uSVP. To that end, we aim at determining the runtime or cost of lattice algorithms to solve such problems and we are particularly interested in the average-case or expected behavior of those algorithms. There are two kinds of results that can be derived, namely mathematically rigorous or heuristic statements. Often, mathematically rigorous statements can be used to derive (upper) bounds on the runtime of lattice algorithms, while heuristic statements are used to predict the average-case behavior. The latter is arguably of greater interest in a cryptanalytic setting as it can be used to estimate concrete security levels of cryptographic schemes. In this spirit, many of our results are based on common heuristics which are standard assumptions in lattice-based cryptography, for example about the lengths of shortest non-zero vectors in random lattices, the shape of reduced lattice bases, or the lengths of orthogonal projections of vectors. Such heuristics are typically supported by theoretical and/or experimental evidence indicating that under plausible assumptions they constitute reliable predictors. Many of our results are therefore also of a heuristic nature and provide good estimates for the practical behavior of lattice algorithms. One could attempt to formulate these results as mathematically rigorous theorems by stating that all of the heuristics hold exactly in the theorem requirements. However, we refrain from doing so as, in our opinion, it deceives the reader.

2.4.2 Lattice Reduction

Informally, *lattice reduction* (also called *lattice basis reduction* or *basis reduction*) is the process of improving the quality of a lattice basis. To express the output quality of a lattice reduction, we may relate the shortest vector in the output basis to the determinant of the lattice in the *Hermite-factor regime* or to the shortest vector in the lattice, in the *approximation-factor regime.* Note that any algorithm finding a vector with approximation-factor α in some lattice Λ, i.e., a vector of length at most $\alpha\lambda_1(\Lambda)$, can be used to solve the uSVP with a gap $\lambda_2(\Lambda)/\lambda_1(\Lambda) > \alpha$.

The best known theoretical bound for lattice reduction is attained by Slide reduction [GN08a]. In this work, however, we consider the *Block-wise Korkine-Zolotarev (BKZ)* [SE94] algorithm, more precisely BKZ 2.0 [CN11, Che13], which performs better in practice. We may simply use the term BKZ to refer to BKZ and BKZ 2.0. BKZ is specified by a block size β, which is upper-bounded by the rank of the lattice. The BKZ-β algorithm repeatedly calls an SVP oracle for finding (approximate) shortest non-zero vectors in projected lattices (also called local blocks) of dimension β. A pseudocode for the BKZ 2.0 algorithm is provided in Chapter 3 in Algorithm 3. It has been shown that after polynomially many calls to the SVP oracle, the basis does not change much more [HPS11].

For the rest of this subsection, let $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_d\} \subset \mathbb{R}^d$ be a basis of some lattice Λ. After BKZ-β reduction, we call the basis *BKZ-β reduced* and in the Hermite-factor regime assume [Che13] that this basis contains a vector of length $\|\mathbf{b}_1\| = \delta^d \cdot \det(\Lambda)^{1/d}$, where

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

is called the *root Hermite factor*. Throughout this work, we implicitly assume that this relation between β and δ holds without explicitly mentioning it. Furthermore, we generally assume that for a BKZ-β reduced basis the *Geometric Series Assumption (GSA)* holds.

Definition 2.8 (Geometric Series Assumption [Sch03])**.** *The norms of the Gram-Schmidt vectors after lattice reduction satisfy*

$$\|\mathbf{b}_i^*\| = \alpha^{i-1} \cdot \|\mathbf{b}_1\| \text{ for some } 0 < \alpha < 1.$$

Combining the GSA with the root Hermite factor $\|\mathbf{b}_1\| = \delta^d \cdot \det(\Lambda)^{1/d}$ and $\det(\Lambda) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|$, we get $\alpha = \delta^{-2d/(d-1)} \approx \delta^{-2}$ for the GSA. While the GSA is widely relied upon in lattice-based cryptography (see, e.g., [APS15, ADPS16, AWHT16, CN11, MW16, HG07]), we emphasize that it does not offer precise estimates, in particular for the last indices of highly reduced bases, see, e.g., [Che13].

Runtime estimates for BKZ. In the following, we summarize the most common ways to estimate the cost of BKZ. Note that currently there is no consensus in the cryptographic community as to which approach to use. BKZ proceeds in several tours (also called rounds). Let d be the lattice dimension, β be the applied block size, and k be the required number of tours in BKZ. Each tour of BKZ consists of d SVP calls, $d - \beta + 1$ of which are in dimension β and $\beta - 1$ of which are in smaller dimensions. One typically estimates the cost $T_{\mathsf{BKZ}}(d, \beta, k)$ of BKZ by predicting the number of SVP oracle calls and multiplying this number by the estimated cost $T_{\mathsf{SVP}}(\beta)$ for one SVP oracle call in dimension β. This can for instance be done via

$$T_{\mathsf{BKZ}}(d, \beta, k) = dk \cdot T_{\mathsf{SVP}}(\beta)$$

or

$$T_{\mathsf{BKZ}}(d, \beta, k) = (d - \beta + 1)k \cdot T_{\mathsf{SVP}}(\beta).$$

The first estimate assumes that all of the SVP calls of one tour are in dimension β, while the latter estimate accounts for the fact that the last SVP calls in each tour are performed in dimension smaller than β and ignores their cost. An alternative (conservative) estimate, commonly referred to as the *core-SVP* estimate [ADPS16], is to estimate the cost of BKZ to be the cost of one SVP call, i.e.,

$$T_{\mathsf{BKZ}}(d, \beta, k) = T_{\mathsf{SVP}}(\beta).$$

How to estimate $T_{\mathsf{SVP}}(\beta)$ is discussed in Section 2.4.3. It remains to estimate the number of tours k required by BKZ. The most common approaches are to either use the BKZ 2.0 simulator of [Che13, CN11] to determine k or to heuristically set $k = 8$, see, e.g., [APS15].

2.4.3 SVP Algorithms

As mentioned above, lattice reduction algorithms make heavy use of SVP solvers. The two most commonly used types of such SVP algorithms for security estimates are *enumeration* algorithms [Kan83, FP85, MW15] and *sieving* algorithms [AKS01, LMvdP15, BDGL16]. Sieving algorithms offer a better asymptotic runtime complexity than enumeration algorithms, but the exact cross-over point is unknown (see e.g. the discussion in [Laa15b]). However, sieving algorithms require access to exponentially large memory, while enumeration only requires polynomial memory, which may render sieving algorithms less practical in high dimensions. Both sieving and enumeration algorithms benifit from quantum speedups [LMvdP15, ANS18]. For more details on those algorithms, we refer to the respective works. In this work, we are mainly concerned with runtime estimates for those algorithms in order to estimate the runtime of lattice reduction algorithms. Unfortunately, different estimates exist throughout the literature. The most common ones are the following. A list of more estimates (for SVP and BKZ) that exist in the literature can be found in Section 8.3.

For enumeration algorithms in dimension β, the most common cost estimate is given by an interpolation by Albrecht et al. [APS15] based on experiments of Chen and Nguyen [CN11]:

$$T_{\mathsf{SVP}_\beta} \approx 2^{0.187\beta \log_2(\beta) - 1.019\beta + 16.1} \approx 2^{0.270\beta \ln(\beta) - 1.019\beta + 16.1}.$$

Classical sieving algorithms in dimension β are often assumed [BDGL16, Alb17] to require a cost of

$$T_{\mathsf{SVP}_\beta} \approx 2^{0.292\,\beta + 16.4},$$

while quantum sieving [LMvdP15] algorithms are assumed to cost

$$T_{\mathsf{SVP}_\beta} \approx 2^{0.265\,\beta + 16.4}.$$

We note that the different cost models diverge on the unit of operations they are using. In the enumeration models, the unit is “number of nodes visited during enumeration”. It is typically assumed that processing one node costs about 100 CPU cycles [CN11]. For classical sieving algorithms the elementary operation is typically an operation on integers or floating point numbers, costing about one CPU cycle. For quantum SVP algorithms the unit is typically the number of Grover iterations required. It is not clear how this translates to traditional CPU cycles. Of course, for models which suppress lower order terms, the unit of computation considered is immaterial.

More details on various methods to cost SVP and BKZ are provided in Section 8.3, where we discuss the cost models applied in the submissions to NIST’s standardization process [Nat16].

2.4.4 Kannan’s Embedding Technique

One of the most common approaches to solve LWE is *Kannan’s embedding approach* [Kan87], which views LWE as a BDD problem and then embeds it into a uSVP instance. It can be described as follows. Let

$$L_{(\mathbf{A},q)} = \{\mathbf{v} \in \mathbb{Z}_q^m \mid \mathbf{v} \equiv \mathbf{A}\mathbf{x} \pmod{q} \text{ for some } \mathbf{x} \in \mathbb{Z}^n\}$$

be the q-ary lattice generated by $\mathbf{A}$ and $\mathbf{B}$ be some basis of $L_{(\mathbf{A},q)}$. Then it holds that $\mathbf{b} \in L_{(\mathbf{A},q)} + \mathbf{e}$, since $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q$. Hence $\mathbf{e}$ can be recovered by solving a BDD problem in $L_{(\mathbf{A},q)}$ with target vector $\mathbf{b}$. In order to solve this BDD problem, it is embedded into a uSVP instance

$$\begin{pmatrix}\mathbf{e}\\ M\end{pmatrix} \in \Lambda(\mathbf{B}') \text{ with } \mathbf{B}' = \begin{pmatrix}\mathbf{B} & \mathbf{b}\\ \mathbf{0} & M\end{pmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)},$$

where M is the so-called embedding factor. Typical choices of M are discussed in, e.g., [LM09, AFG14, APS15], and include $M = 1$ or $M = \|\mathbf{e}\|$. As pointed out in [APS15], $M = 1$ is typically more efficient and therefore often used in practice, including this work, see also [WAT18]. The dimension of the obtained uSVP lattice is $m + 1$ and with high probability, its determinant is $M \cdot q^{m-n}$, see for example [AFG14]. This uSVP instance is then solved by running lattice reduction on the basis $\mathbf{B}'$. Embedding LWE into uSVP and solving it via lattice reduction is also referred to as the *primal attack*. A simplified pseudocode of Kannan’s embedding approach is given in Algorithm 2.

2.4.5 Babai’s Nearest Plane

Babai’s Nearest Plane algorithm [Bab86] (denoted by NP in the following) is a BDD algorithm and an important building block of several attacks or algorithms. For more

Algorithm 2: Kannan's embedding approach

```
Input: An LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \mod q) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, embedding
       factor $M$
1 Construct a lattice basis $\mathbf{B} \in \mathbb{Z}^{m\times m}$ of the lattice
  $L_{(\mathbf{A},q)} = \{\mathbf{v} \in \mathbb{Z}_q^m \mid \mathbf{v} \equiv \mathbf{A}\mathbf{x} \pmod q \text{ for some } \mathbf{x} \in \mathbb{Z}^n\}$ ;
2 Set $\mathbf{B}' = \begin{pmatrix} \mathbf{B} & \mathbf{b} \\ \mathbf{0} & M \end{pmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)}$;
3 Recover $\pm \begin{pmatrix} \mathbf{e} \\ M \end{pmatrix}$ by solving uSVP in $\Lambda(\mathbf{B}')$ using lattice reduction;
4 return $\mathbf{e}$;
```

details on the algorithm we refer to Babai's original work [Bab86] or Lindner and Peikert's work [LP11]. We use the Nearest Plane algorithm in a black box manner and the following is sufficient to know. The input for the Nearest Plane algorithm is a lattice basis $\mathbf{B} \subset \mathbb{Z}^d$ of a full-rank lattice and a target vector $\mathbf{t} \in \mathbb{R}^d$ and the corresponding output is a vector $\mathbf{e} \in \mathbb{R}^d$ such that $\mathbf{t} - \mathbf{e} \in \Lambda(\mathbf{B})$. We denote the output by $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}) = \mathbf{e}$. If there is no risk of confusion, we may omit the basis in the notation, writing $\mathrm{NP}(\mathbf{t})$ instead of $\mathrm{NP}_{\mathbf{B}}(\mathbf{t})$. The output of the Nearest Plane algorithm satisfies the following condition, as shown in [Bab86].

Lemma 2.1. *Let $\mathbf{B} \subset \mathbb{Z}^d$ be a basis of a full-rank lattice and $\mathbf{t} \in \mathbb{R}^d$ be a target vector. Then $\mathrm{NP}_{\mathbf{B}}(\mathbf{t})$ is the unique vector $\mathbf{e} \in \mathcal{P}(\mathbf{B}^*)$ that satisfies $\mathbf{t} - \mathbf{e} \in \Lambda(\mathbf{B})$, where $\mathbf{B}^*$ is the Gram-Schmidt basis of $\mathbf{B}$.*

In [HHHGW09], Hirschhorn et al. experimentally verify the number of bit operations (defined as in [LV01]) of one Nearest Plane call in dimension d to be approximately $d^2/2^{1.06}$. Furthermore, they conservatively assume that using precomputation the number of operations might possibly be decreased to $d/2^{1.06}$. However, this speedup has not yet been confirmed in practice.

2.4.6 Other Lattice Algorithms and Attacks

Besides the algorithms to solve lattice problems discussed in this work, there also exist other algorithms or attacks. We briefly discuss the most common ones in the following.

The dual attack on LWE solves the Decision-LWE problem by reducing it to the short integer solution problem [Ajt96]. This problem is then solved by finding short vectors in the lattice $\{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{x}^t\mathbf{A} \equiv \mathbf{0} \mod q\}$, where $\mathbf{A}$ is the LWE matrix and q the LWE modulus. In the case of small or sparse secret distributions, the dual attack can further be improved [Alb17]. Note that there is a computational overhead if one wants to convert this attack into an attack on Search-LWE.

The decoding attack [LP11] on LWE solves the Search-LWE problem by viewing it as a BDD problem. This BDD problem can then for instance be solved by Babai's Nearest Plane algorithm, see Section 2.4.5. In the case of small or sparse secret vectors, the hybrid attack as discussed in Chapters 5, 6, and 7 can also be seen as an improvement of the decoding attack.

The BKW attack [BKW00] and its improvements [AFFP14, GJS15, KF15, GJMS17] are combinatorial approaches to solve the Search-LWE problem. The main practical downside of these attacks is that they require access to exponentially many LWE sample and exponentially large memory. However, the first problem can be circumvented by producing more samples.

There also exist algebraic attacks on LWE [AG11, ACF$^+$15]. However, similar to the BKZ-style attacks, these attacks require a large number of LWE samples (or are less efficient in the case of few samples), which is typically not provided in a cryptographic context.

In addition to algorithms that solve lattice problems for standard lattices, there also exists a line of work which aims at solving the ring-variants of lattice problems more efficiently. For instance, these works include the discovery of polynomial-time quantum algorithms that recover short vectors in principal ideal lattices over cyclotomic number fields of prime-power degree [CDPR16, BS16]. These results can be used to obtain better approximation fectors for approximate SVP in general ideal lattices over certain number fields, e.g., [CDW17, Bia17]. In addition, there have been recent discoveries of some alleged weak instances of Ring-LWE, e.g., [EHL14, ELOS16, ELOS15] which, however, may be explained by an unfortunate choice in the LWE error distribution as detailed in [CIV16, Pei16b]. In the case of NTRU, subfield and other attacks on overstretched NTRU assumptions [ABD16, CJL16, KF17] have been presented, which have consequences for instance on NTRU-based fully homomorphic encryption. The author of this thesis contributed to this line of work with the joint publication [9] by extending the results of [CDPR16] to cyclotomic number fields whose conductor is a product of two prime-powers, which is not part of this thesis.

3 On the Expected Cost of Solving uSVP via Lattice Reduction

One of the currently most common and efficient approaches to solve lattice problems such as LWE or the NTRU problem is to embed them into a uSVP instance and then solve the resulting uSVP instance using the BKZ [SE94] (or BKZ 2.0 [CN11, Che13]) lattice reduction algorithm. It is therefore an important cryptanalytic task to predict the cost of solving uSVP using BKZ. This cost is mainly determined by the applied block size, which size specifies the BKZ algorithm, where a bigger block size yields a higher cost. However, if the block size is not sufficiently large, BKZ will not succeed in solving uSVP, begging the question about the minimal block size that guarantees success. In the current literature there exist two different estimates for this minimal block size: the *2008 estimate* introduced in [GN08b], developed in [AFG14, APS15, Göp16, HKM17], and applied in, e.g., [BG14a, CHK$^+$17, CKLS16a, CLP17, ABB$^+$17], and the recently introduced [ADPS16] *2016 estimate* applied in, e.g., [BCD$^+$16, BDK$^+$18]. However, the two estimates predict vastly different costs. For example, considering an LWE instance with $n = 1024$, $q \approx 2^{15}$, and a discrete Gaussian LWE error distribution with standard deviation $\sigma = 3.2$, the former predicts a cost of roughly 2^{355} operations, whereas the latter predicts a cost of roughly 2^{287} operations to solve the problem.[2] This begs the question whether the 2016 estimate should replace the 2008 estimate. So far, the 2008 estimate has been experimentally studied only for small parameters and block sizes, while the 2016 estimate has not been subject to a theoretical or experimental analysis, thus the question remains open.

Contribution. In this chapter, we provide the first theoretical and experimental validation of the 2016 estimate. Our theoretical analysis is based on standard lattice assumptions such as the Geometric Series Assumption (GSA) and the assumption that the unique shortest non-zero vector is distributed in a random direction relative to the rest of the basis. Under these assumptions we show that, using a block size satisfying the 2016 estimate, BKZ eventually recovers a projection of the unique shortest

[2] Assuming the same cost model for BKZ with block size β, where an SVP oracle call in dimension β costs $2^{0.292\,\beta+16.4}$ [BDGL16, APS15, Laa15b].

non-zero vector and with high probability the so-called size reduction subroutine immediately recovers the uSVP solution from its projection. For our experiments we employ the widely-used `fplll` 5.1.0 [FPL17] and `fpylll` 0.2.4dev [FPY17] libraries and use medium to larger block sizes. Our results confirm that the behavior of BKZ largely follows the 2016 estimates. Finally, we demonstrate the cryptographic relevance of our work by giving reduced attack costs for some lattice-based schemes. In particular, we give reduced costs for solving the LWE instances underlying TESLA [ABB$^+$17] and the somewhat homomorphic encryption scheme in [BCIV17]. We also show that under the revised, corrected estimate, the primal attack performs about as well on SEAL v2.1 parameter sets as the dual attack from [Alb17].

Organization. In Section 3.1, we recall the two competing estimates from the literature. Our analysis of the 2016 estimate is presented in Section 3.2. The theoretical aspects are presented in Sections 3.2.1 and 3.2.3. In Section 3.2.2, we provide our experimental setup and results. Both theory and practice confirm the 2016 estimate. Finally, using the 2016 estimate, in Section 3.3 we show that some proposed parameters from the literature need to be updated to maintain the currently claimed level of security.

Publications. This chapter is based on the publication [4] presented at ASIACRYPT 2017.

3.1 Estimates

As highlighted above, two competing estimates, the 2008 and the 2016 estimate, exist in the literature for when block-wise lattice reduction succeeds in solving uSVP instances. However, the predicted costs under these two estimates differ greatly as illustrated in Figure 3.1.

3.1.1 2008 Estimate

A first systematic experimental investigation into the behavior of the lattice reduction algorithms LLL, DEEP and BKZ was provided in [GN08b]. In particular, [GN08b] investigates the behavior of these algorithms for solving uSVP for families of lattices arising in cryptography.

For uSVP, the authors performed experiments in small block sizes on two classes of semi-orthogonal lattices and on Lagarias-Odlyzko lattices [LO83], which permit to estimate the gap $\lambda_2(\Lambda)/\lambda_1(\Lambda)$ between the first and second minimum of the lattice. The authors of [GN08b] observed that LLL and BKZ seem to recover a unique shortest non-zero vector with high probability whenever $\lambda_2(\Lambda)/\lambda_1(\Lambda) \geq \tau\delta^d$, where δ

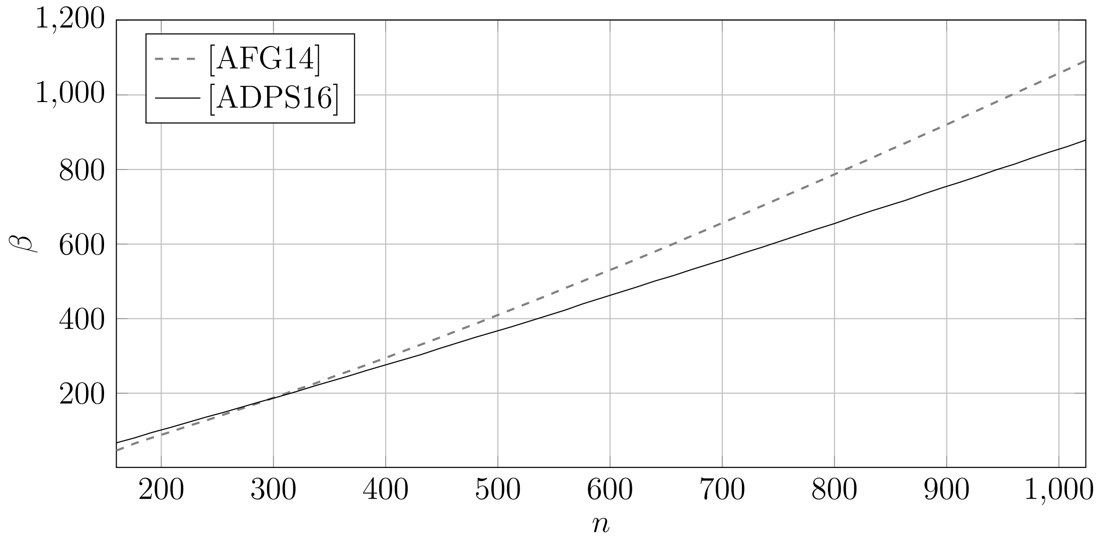

Figure 3.1: Required block size β according to the estimates given in [AFG14] and [ADPS16] for solving LWE with modulus $q = 2^{15}$, an error distribution with standard deviation $\sigma = 3.2$ and increasing secret dimension n. For [AFG14] we set $\tau = 0.3$ and use the embedding factor 1. Lattice reduction runs in time $2^{\Omega(\beta)}$.

is the root Hermite factor of the reduced basis and $\tau < 1$ is an empirically determined constant that depends on the lattice family and algorithm used.

In [AFG14] an experimental analysis of solving an LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ based on the same estimate was carried out for lattices using Kannan's embedding (see Section 2.4.4). The embedding lattice contains an unusually short vector $\mathbf{v} = (\mathbf{e} \mid M)$ of squared norm $\lambda_1(\Lambda)^2 = \|\mathbf{v}\|^2 = \|\mathbf{e}\|^2 + M^2$. Thus, when $M = \|\mathbf{e}\|$ resp. $M = 1$ this implies $\lambda_1(\Lambda) \approx \sqrt{2m}\sigma$ resp. $\lambda_1(\Lambda) \approx \sqrt{m}\sigma$, where σ is the standard deviation of the LWE error distribution χ, i.e., $\mathbf{e}_i \leftarrow_{\$} \chi$. The second minimum $\lambda_2(\Lambda)$ is assumed to correspond to the Gaussian Heuristic for the lattice. Experiments in [AFG14] using LLL and BKZ (with block sizes 5 and 10) confirmed the 2008 estimate, providing constant values for τ for such lattices, depending on the chosen algorithm, for a 10% success rate. Overall, τ was found to lie between 0.3 and 0.4 when using BKZ.

Still focusing on LWE, in [APS15] a closed formula for δ is given as a function of n, σ, q, and τ, which implicitly assumes $M = \|\mathbf{e}\|$. In [Göp16], a bound for δ in the [GN08b] model for the case of $M = 1$, which is mainly used in practice, is given. In [HKM17], a related closed formula is given, directly expressing the asymptotic running time for solving LWE using this approach.

3.1.2 2016 Estimate

In [ADPS16], an alternative estimate is outlined. Let $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$ be an LWE instance, σ be the standard deviation of the LWE error distribution, $\mathbf{B}$ be a basis of the corresponding uSVP lattice using Kannan's embedding, and $d = m + 1$. The 2016 estimate predicts that $\mathbf{e}$ can be found if[3]

$$\sqrt{\beta/d}\;\|(\mathbf{e} \mid 1)\| \approx \sqrt{\beta}\sigma \leq \delta^{2\beta-d}\det(\Lambda(\mathbf{B}))^{1/d}, \tag{3.1}$$

under the assumption that the Geometric Series Assumption holds (until a projection of the unusually short vector is found). In the general case of uSVP in some full-rank lattice of dimension d with unique shortest non-zero vector $\mathbf{v}$, this can be generalized to

$$\sqrt{\beta/d}\,\lambda_1(\Lambda) = \sqrt{\beta/d}\;\|\mathbf{v}\| \leq \delta^{2\beta-d}\det(\Lambda)^{1/d}. \tag{3.2}$$

The brief justification for this estimate given in [ADPS16] notes that this condition ensures that the projection of $\mathbf{e}$ orthogonally to the first $d - \beta$ (Gram-Schmidt) vectors is shorter than the expectation for $\mathbf{b}^*_{d-\beta+1}$ under the GSA. This brief note can be extended as follows. As the projection of $\mathbf{e}$ is shorter than the expectation for $\mathbf{b}^*_{d-\beta+1}$, it would be found by the SVP oracle when called on the last block of size β. Hence, for any β satisfying (3.1), the actual behavior would deviate from that predicted by the GSA. Finally, the argument can be completed by appealing to the intuition that a deviation from expected behavior on random instances — such as the GSA — leads to a revelation of the underlying structural, secret information.[4]

3.2 Solving uSVP

Given the significant differences in expected solving time under the two estimates, cf. Figure 3.1, and recent progress in publicly available lattice reduction libraries enabling experiments in larger block sizes [FPL17, FPY17], we conduct a more detailed examination of BKZ's behavior on uSVP instances. For this, we first explicate the outline from [ADPS16] to establish the expected behavior, which we then experimentally investigate in Section 3.2.2. Overall, our experiments confirm the expectation of the 2016 estimate. However, the algorithm behaves somewhat better than expected, which we then explain in Section 3.2.3.

For the rest of this chapter, let $\mathbf{v}$ be a shortest non-zero vector in some d-dimensional full-rank uSVP lattice Λ. Furthermore, in the case of solving LWE via Kannan's embedding, let $d = m + 1$ and $\mathbf{v} = (\mathbf{e} \mid 1) \in \mathbb{Z}_q^d$, where m is the number of LWE samples, q the modulus, and $\mathbf{e}$ the LWE error vector.

[3][ADPS16] has $2\beta - d - 1$ in the exponent, which seems to be an error.

[4]We note that observing such a deviation implies solving Decision-LWE.

3.2.1 Prediction

Projected norm.

In what follows, we assume the unique shortest non-zero vector $\mathbf{v}$ is drawn from a spherical distribution or is at least “not too skewed” with respect to the current basis. As a consequence, following [ADPS16], we assume that all orthogonal projections of $\mathbf{v}$ onto a k-dimensional subspace of $\mathbb{R}^d$ have expected norm $(\sqrt{k}/\sqrt{d})\,\|\mathbf{v}\|$. Note that this assumption can be dropped by adapting (3.2) to $\|\mathbf{v}\| \leq \delta^{2\beta-d}\det(\Lambda)^{\frac{1}{d}}$ since $\|\pi_{d-\beta+1}(\mathbf{v})\| \leq \|\mathbf{v}\|$.

Finding a projection of the short vector.

Assume that β is chosen minimally such that (3.2) holds. When running BKZ, the length of the Gram-Schmidt basis vectors of the current basis converge to the lengths predicted by the GSA. Therefore, at some point BKZ will find a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ of Λ for which we can assume that the GSA holds with root Hermite factor δ. Now, consider the stage of BKZ where the SVP oracle is called on the last full projected block of size β with respect to this basis $\mathbf{B}$. Note that the projection $\pi_{d-\beta+1}(\mathbf{v})$ of the shortest non-zero vector is contained in the lattice

$$\Lambda_{d-\beta+1} := \Lambda\left(\pi_{d-\beta+1}(\mathbf{b}_{d-\beta+1}), \ldots, \pi_{d-\beta+1}(\mathbf{b}_d)\right),$$

since

$$\pi_{d-\beta+1}(\mathbf{v}) = \sum_{i=d-\beta+1}^{d} \nu_i \pi_{d-\beta+1}(\mathbf{b}_i) \in \Lambda_{d-\beta+1}, \text{ where } \nu_i \in \mathbb{Z} \text{ with } \mathbf{v} = \sum_{i=1}^{d} \nu_i \mathbf{b}_i.$$

By (3.2), the projection $\pi_{d-\beta+1}(\mathbf{v})$ is in fact expected to be the shortest non-zero vector in $\Lambda_{d-\beta+1}$, since it is shorter than the GSA’s estimate for $\lambda_1(\Lambda_{d-\beta+1})$, i.e.

$$\|\pi_{d-\beta+1}(\mathbf{v})\| \approx \frac{\sqrt{\beta}}{\sqrt{d}}\,\|\mathbf{v}\| \leq \delta^{-2(d-\beta)+d}\det(\Lambda)^{\frac{1}{d}}.$$

Hence the SVP oracle will find $\pm\pi_{d-\beta+1}(\mathbf{v})$ and BKZ inserts

$$\mathbf{b}_{d-\beta+1}^{\mathsf{new}} = \pm \sum_{i=d-\beta+1}^{d} \nu_i \mathbf{b}_i$$

into the basis $\mathbf{B}$ at position $d-\beta+1$. In other words, by finding $\pm\pi_{d-\beta+1}(\mathbf{v})$, BKZ recovers the last β coefficients $\nu_{d-\beta+1}, \ldots, \nu_d$ of $\mathbf{v}$ with respect to the basis $\mathbf{B}$.

Finding the short vector.

The above argument can be extended to an argument for the full recovery of $\mathbf{v}$. Consider the case that in some tour of BKZ-β, a projection of $\mathbf{v}$ was found at index $d-\beta+1$. Then in the following tour, by arguments analogous to the ones above, a projection of $\mathbf{v}$ will likely be found at index $d-2\beta+2$, since now it holds that

$$\pi_{d-2\beta+2}(\mathbf{v}) \in \Lambda_{d-2\beta+2} := \Lambda\left(\pi_{d-2\beta+2}(\mathbf{b}_{d-2\beta+2}), \ldots, \pi_{d-2\beta+2}(\mathbf{b}^{\mathsf{new}}_{d-\beta+1})\right).$$

Repeating this argument for smaller indices shows that after a few tours $\mathbf{v}$ will be recovered. Furthermore, noting that BKZ calls LLL which in turn calls size reduction, i.e., Babai's Nearest Plane [Bab86], at some index $i > 1$ size reduction will recover $\mathbf{v}$ from $\pi_i(\mathbf{v})$. In particular, it is well-known that size reduction (Algorithm 1) will succeed in recovering $\mathbf{v}$ whenever

$$\mathbf{v} \in \mathbf{b}^{\mathsf{new}}_{d-\beta+1} + \left\{ \sum_{i=1}^{d-\beta} c_i \cdot \mathbf{b}_i^* : c_i \in \left[-\frac{1}{2}, \frac{1}{2}\right] \right\}. \tag{3.3}$$

3.2.2 Observation

The above discussion naturally suggests a strategy to verify the expected behavior. We have to verify that the projected norms $\|\pi_i(\mathbf{v})\| = \|\pi_i(\mathbf{e} \mid 1)\|$ do indeed behave as expected and that $\pi_{d-\beta+1}(\mathbf{v})$ is recovered by BKZ-β for the minimal $\beta \in \mathbb{N}$ satisfying (3.1). Finally, we have to measure when and how $\mathbf{v} = (\mathbf{e} \mid 1)$ is eventually recovered.

Thus, we ran lattice reduction on many lattices constructed from LWE instances $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$ using Kannan's embedding. In more detail, we picked the entries of $\mathbf{s}$ and $\mathbf{A}$ uniformly at random from $\mathbb{Z}_q$, the entries of $\mathbf{e}$ from a discrete Gaussian distribution with standard deviation $\sigma = 8/\sqrt{2\pi}$, and we constructed our basis as in Section 2.4.4 with embedding factor $M = 1$. For parameters (n, q, σ), we then estimated the minimal pair (in lexicographical order) (β, m) to satisfy (3.1).

Implementation.

To perform our experiments, we used SageMath 7.5.1 [S+17] in combination with the `fplll` 5.1.0 [FPL17] and `fpylll` 0.2.4dev [FPY17] libraries. All experiments were run on a machine with Intel(R) Xeon(R) CPU E5-2667 v2 @ 3.30GHz cores ("strombenzin") resp. Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz ("atomkohle"). Each instance was reduced on a single core, with no parallelization.

Our BKZ implementation inherits from the implementation in `fplll` and `fpylll` of BKZ 2.0 [Che13, CN11] algorithm. As in BKZ 2.0, we restricted the enumeration radius to be approximately the size of the Gaussian Heuristic for the projected sublattice, apply recursive BKZ-β' preprocessing with a block size $\beta' < \beta$, make use of extreme pruning [GNR10] and terminate the algorithm when it stops making significant progress. We give simplified pseudo-code of our BKZ implementation in Algorithm 3. We ran BKZ for at most 20 tours using `fplll`'s default pruning and preprocessing strategies and, using `fplll`'s default auto abort strategy, terminated the algorithm whenever the slope of the Gram Schmidt vectors did not improve for five consecutive tours. Additionally, we aborted if a vector of length $\approx \|\mathbf{v}\|$ was found in the basis (in line 14 of Algorithm 3).

Algorithm 3: Simplified BKZ 2.0 Algorithm

```
   Input: LLL-reduced lattice basis $\mathbf{B}$, block size $\beta$, preprocessing block size $\beta'$
 1 repeat                                                          // tour
 2     for $\kappa \leftarrow 1$ to $d$ do                             // step$_\kappa$
 3         size reduction from index 1 to $\kappa$ (inclusive);
 4         $\ell \leftarrow \|\mathbf{b}_\kappa^*\|$;
           // extreme pruning + recursive preprocessing
 5         repeat until termination condition met
 6             rerandomize $\pi_\kappa(\mathbf{b}_{\kappa+1}, \ldots, \mathbf{b}_{\kappa+\beta-1})$;
 7             LLL on $\pi_\kappa(\mathbf{b}_\kappa, \ldots, \mathbf{b}_{\kappa+\beta-1})$;
 8             BKZ-$\beta'$ on $\pi_\kappa(\mathbf{b}_\kappa, \ldots, \mathbf{b}_{\kappa+\beta-1})$;
 9             $\mathbf{v} \leftarrow$ SVP on $\pi_\kappa(\mathbf{b}_\kappa, \ldots, \mathbf{b}_{\kappa+\beta-1})$;
10             if $\mathbf{v} \neq \perp$ then
11                 extend $\mathbf{B}$ by inserting $\mathbf{v}$ into $\mathbf{B}$ at index $\kappa + \beta$;
12                 LLL on $\pi_\kappa(\mathbf{b}_\kappa, \ldots, \mathbf{b}_{\kappa+\beta})$ to remove linear dependencies;
13                 drop row with all zero entries;
14         size reduction from index 1 to $\kappa$ (inclusive);
15         if $\ell = \|\mathbf{b}_\kappa^*\|$ then
16             yield $\top$;
17         else
18             yield $\perp$;
19     if $\top$ for all $\kappa$ then
20         return;
```

Implementations of block-wise lattice reduction algorithms such as BKZ make heavy use of LLL [LLL82] and size reduction. This is to remove linear dependencies introduced during the algorithm, to avoid numerical stability issues and to improve the performance of the algorithm by moving short vectors to the front earlier. The main modification in our implementation is that calls to LLL during preprocessing and postprocessing are restricted to the current block, not touching any other vector, to aid analysis. That is, in Algorithm 3, LLL is called in lines 7 and 12 and we modified these LLL calls not to touch any row with index smaller than κ, not even to perform size reduction.

As a consequence, we only make use of vectors with index smaller than κ in lines 3 and 14. Following the implementations in [FPL17, FPY17], we call size reduction from index 1 to κ before (line 3) and after (line 14) the innermost loop with calls to the SVP oracle. These calls do not appear in the original description of BKZ. However, since the innermost loop re-randomizes the basis when using extreme pruning, the success condition of the original BKZ algorithm needs to be altered. That is, the algorithm cannot break the outer loop once it makes no more changes as originally specified. Instead, the algorithm terminates if it does not find a shorter vector at any index κ. Now, the calls to size reduction ensure that the comparison at the beginning and end of each step κ is meaningful even when the Gram-Schmidt vectors are only updated lazily in the underlying implementation. That is, the call to size reduction triggers an internal update of the underlying Gram-Schmidt vectors and are hence implementation artifacts. The reader may think of these size reduction calls as explicating calls otherwise hidden behind calls to LLL and we stress that our analysis applies to BKZ as commonly implemented, our changes merely enable us to more easily predict and experimentally verify the behavior.

We note that the break condition for the innermost loop at line 5 depends on the pruning parameters chosen, which control the success probability of enumeration. Since it does not play a material role in our analysis, we simply state that some condition will lead to a termination of the innermost loop.

Finally, we recorded the following information. At the end of each step κ during lattice reduction, we recorded the minimal index i such that $\pi_i(\mathbf{v})$ is in span$(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ and whether $\pm\mathbf{v}$ itself is in the basis. In particular, to find the index i in the basis $\mathbf{B}$ of $\pi_i(\mathbf{v})$ given $\mathbf{v}$, we compute the coefficients of $\mathbf{v}$ in basis $\mathbf{B}$ (at the current step) and pick the first index i such that all coefficients with larger indices are zero. Then, we have $\pi_i(\mathbf{b}_i) = c \cdot \pi_i(\mathbf{v})$ for some $c \in \mathbb{R}$. From the algorithm, we expect to have found $\pm\pi_i(\mathbf{b}_i) = \pi_i(\mathbf{v})$ and call i the index of the projection of $\mathbf{v}$.

Results.

In Figure 3.2, we plot the average norms of $\pi_i(\mathbf{v})$ and the expectation $\sqrt{d-i+1}\,\sigma \approx \sqrt{\frac{d-i+1}{d}}\sqrt{m \cdot \sigma^2 + 1}$, indicating that $\sqrt{d-i+1}\,\sigma$ is a close approximation of the expected lengths except perhaps for the last few indices.

Recall that, as illustrated in Figure 3.3, we expect to find the projection $\pi_{d-\beta+1}(\mathbf{v})$ when (β, d) satisfy (3.1), eventually leading to a recovery of $\mathbf{v}$, say, by an extension of the argument for the recovery of $\pi_{d-\beta+1}(\mathbf{v})$. Our experiments, summarized in Table 3.1, show a related, albeit not identical behavior. Defining a cut-off index $c = d - 0.9\beta + 1$ and considering $\pi_\kappa(\mathbf{v})$ for $\kappa < c$, we observe that the BKZ algorithm

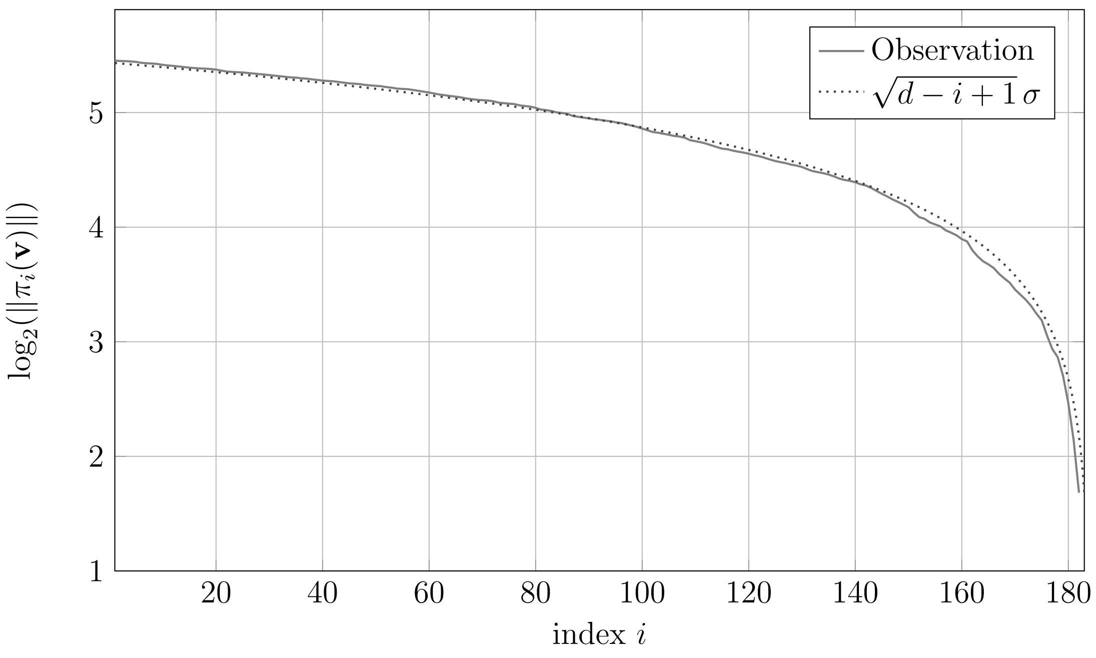

Figure 3.2: Expected and average observed norms $\|\pi_i(\mathbf{v})\|$ for 16 bases (LLL-reduced) and vectors $\mathbf{v}$ of dimension $d = m + 1$ and determinant q^{m-n} with LWE parameters $n = 65, m = 182, q = 521$ and standard deviation $\sigma = 8/\sqrt{2\pi}$.

typically first recovers $\pi_\kappa(\mathbf{v})$ which is immediately followed by the recovery of $\mathbf{v}$ in the same step. In more detail, in Figure 3.4 we show the measured probability distribution of the index κ such that $\mathbf{v}$ is recovered from $\pi_\kappa(\mathbf{v})$ in the same step. Note that the mean of this distribution is smaller than $d - \beta + 1$. We explain this bias in Section 3.2.3.

The recovery of $\mathbf{v}$ from $\pi_\kappa(\mathbf{v})$ can be effected by one of three subroutines: either by a call to LLL, by a call to size reduction, or by a call to enumeration that recovers $\mathbf{v}$ directly. Since LLL itself contains many calls to size reduction, and enumeration being lucky is rather unlikely, size reduction is a good place to start the investigation. Indeed, restricting the LLL calls in Algorithm 3 as outlined in Section 2.4.2, identifies that size reduction suffices. That is, to measure the success rate of size reduction recovering $\mathbf{v}$ from $\pi_\kappa(\mathbf{v})$, we observe size reduction acting on $\pi_\kappa(\mathbf{v})$. Here, we consider size reduction to fail in recovering $\mathbf{v}$ if it does not recover $\mathbf{v}$ given $\pi_\kappa(\mathbf{v})$ for $\kappa < c$ with $c = d - 0.9\beta + 1$, regardless of whether $\mathbf{v}$ is finally recovered at a later point either by size reduction on a new projection, or by some other call in the algorithm such as an SVP oracle call at a smaller index. As shown in Table 3.1, size reduction's success rate is close to 1. Note that the cut-off index c serves to limit underestimating the success rate: intuitively we do not expect size reduction to succeed when starting from a projection with larger index, such as $\pi_{d-\gamma+1}(\mathbf{v})$ with $\gamma < 10$. We discuss this in Section 3.2.3.

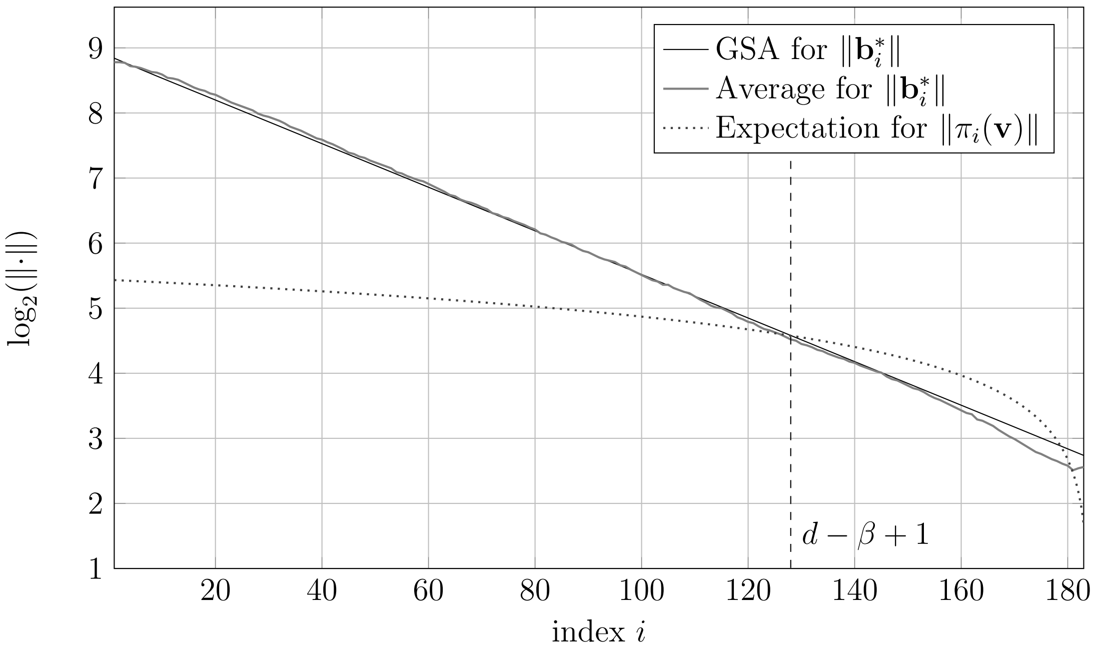

Figure 3.3: Expected and observed norms for lattices of dimension $d = m + 1 = 183$ and determinant q^{m-n} after BKZ-β reduction for LWE parameters $n = 65, m = 182, q = 521$ and standard deviation $\sigma = 8/\sqrt{2\pi}$ and $\beta = 56$ (minimal (β, m) such that (3.1) holds). Average of Gram-Schmidt lengths is taken over 16 BKZ-β reduced bases of random q-ary lattices, i.e. *without* an unusually short vector.

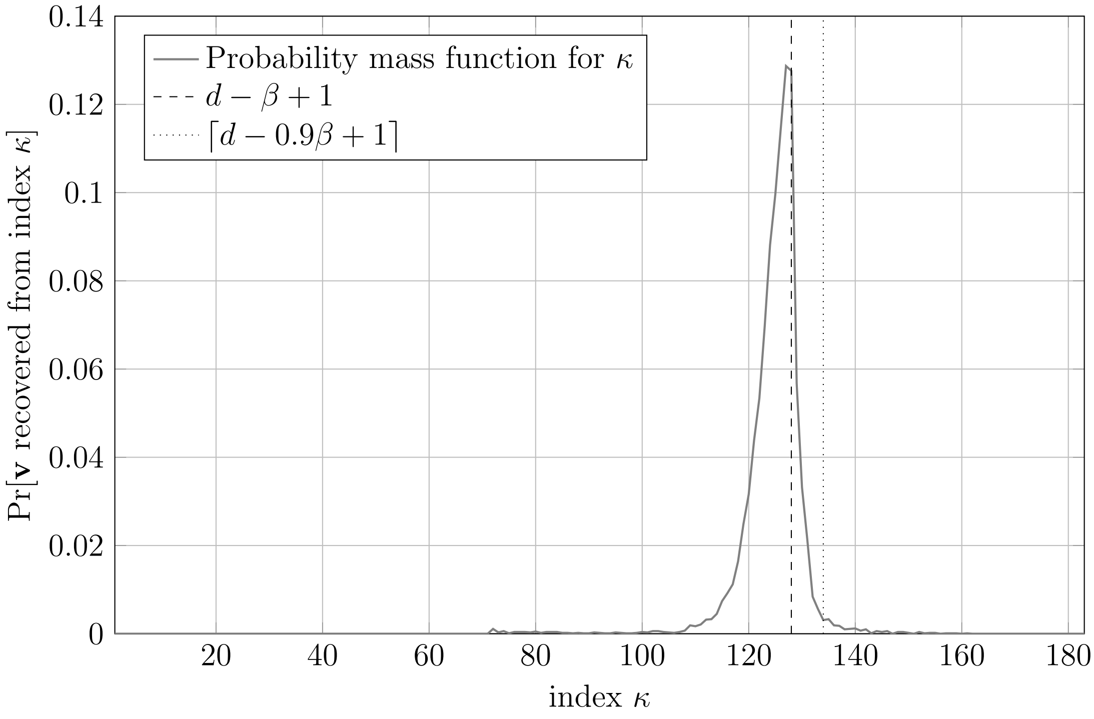

Figure 3.4: Probability mass function of the index κ from which size reduction recovers $\mathbf{v}$, calculated over 10,000 lattice instances with LWE parameters $n = 65, m = 182, q = 521$ and standard deviation $\sigma = 8/\sqrt{2\pi}$, reduced using $\beta = 56$. The mean of the distribution is ≈ 124.76 while $d - \beta + 1 = 128$.

n	q	β_{2016}	m_{2016}	β	#	$\mathbf{v}$	same step		time
							$\kappa < c$	$\kappa = d - \beta + 1$	
65	521	56	182	56	10000	93.3%	99.7%	99.7%	1,131.4
				51		52.8%	98.8%	97.3%	1,359.3
				46		4.8%	96.4%	85.7%	1,541.2
100	2053	67	243	67	500	88.8%	99.8%	100.0%	28,803.7
				62		39.6%	99.5%	100.0%	19,341.9
				57		5.8%	100.0%	100.0%	7,882.2
				52		0.2%	0.0%	—	3,227.0
108	2053	77	261	77	5	100.0%	100.0%	100.0%	351,094.2

Table 3.1: Overall success rate ("$\mathbf{v}$") and success rate of size reduction ("same step") for solving LWE instances characterised by n, σ, q with m samples, standard deviation $\sigma = 8/\sqrt{2\pi}$, minimal (β_{2016}, m_{2016}) such that $\sqrt{b_{2016}}\,\sigma \leq \delta_0^{2\beta_{2016}-(m_{2016}+1)} q^{(m_{2016}-n)/(m_{2016}+1)}$ with δ_0 in function of β_{2016}. The column "β" gives the actual block size used in experiments. The "same step" rate is calculated over all successful instances where $\mathbf{v}$ is found before the cut-off point c and for the instances where exactly $\pi_{d-b+1}(\mathbf{v})$ is found (if no such instance is found, we do not report a value). In the second case, the sample size is smaller, since not all instances recover $\mathbf{v}$ from exactly $\kappa = d - \beta + 1$. The column "time" lists average solving CPU time for one instance, in seconds. Note that our changes to the algorithm and our extensive record keeping lead to an increased running time of the BKZ algorithm compared to [FPL17, FPY17]. Furthermore, the occasional longer running time for smaller block sizes is explained by the absence of early termination when $\mathbf{v}$ is found.

Overall, Table 3.1 confirms the prediction from [ADPS16]: picking $\beta = \beta_{2016}$ to be the block size predicted by the 2016 estimate leads to a successful recovery of $\mathbf{v}$ with high probability. Note that the observed success probability may even be increased by increasing the success probability of the enumeration routine from 0.5 (default) to a value close to 1.

3.2.3 Explaining Observation

As noted above, our experiments indicate that the algorithm behaves better than expected by (3.2). Firstly, the BKZ algorithm does not necessarily recover a projection of $\mathbf{v}$ at index $d - \beta + 1$. Instead, the index κ at which we recover a projection $\pi_\kappa(\mathbf{v})$ follows a distribution with a center below $d - \beta + 1$, cf. Figure 3.4. Secondly, size reduction usually immediately recovers $\mathbf{v}$ from its projection $\pi_\kappa(\mathbf{v})$ at that index. This is somewhat unexpected, since we do not have the guarantee that $|c_i| \leq 1/2$ as required in the success condition of size reduction given in (3.3).

Finding the projection.

To explain the bias towards a recovery of $\pi_\kappa(\mathbf{v})$ for some $\kappa < d - \beta + 1$, note that if (3.2) holds then for the parameter sets *in our experiments* the lines for $\|\pi_i(\mathbf{v})\|$ and $\|\mathbf{b}_i^*\|$ intersect twice (cf. Figure 3.3). Let $d - \gamma + 1$ be the index of the second intersection. Thus, there is a good chance that $\|\pi_{d-\gamma+1}(\mathbf{v})\|$ is a shortest vector in the lattice spanned by the last projected block of some small rank γ and will be placed at index $d - \gamma + 1$. As a consequence, all projections $\pi_i(\mathbf{v})$ with $i > d - \gamma + 1$ will be zero and $\pi_{d-\beta-\gamma+1}(\mathbf{v})$ will be contained in the β-dimensional lattice

$$\Lambda_{d-\beta-\gamma+1} := \Lambda\left(\pi_{d-\beta-\gamma+1}(\mathbf{b}_{d-\beta-\gamma+1}), \ldots, \pi_{d-\beta-\gamma+1}(\mathbf{b}_{d-\gamma+1})\right),$$

enabling it to be recovered by BKZ-β at an index $d - \beta - \gamma + 1 < d - \beta + 1$. Thus, BKZ in our experiments behaves better than predicted by (3.2). We note that another effect of this second intersection is that, for very few instances, it directly leads to a recovery of $\mathbf{v}$ from $\pi_{d-\beta-\gamma+1}(\mathbf{v})$.

Giving a closed formula incorporating this effect akin to (3.2) would entail to predict the index γ and then replace β with $\beta + \gamma$ in (3.2). However, as illustrated in Figure 3.3, neither does the GSA hold for the last 50 or so indices of the basis [Che13] nor does the prediction $\sqrt{d - i + 1}\,\sigma$ for $\|\pi_{d-1+1}(\mathbf{v})\|$.

We stress that while the second intersection often occurs for parameter sets within reach of practical experiments, it does not always occur for all parameter sets. That is, for many large parameter sets, e.g. those in [ADPS16], a choice of β satisfy (3.2) does *not* lead to a predicted second intersection at some larger index. Thus, this effect may highlight the pitfalls of extrapolating experimental lattice reduction data from small instances to large instances.

Finding the short vector.

In what follows, we assume that the projected norm $\|\pi_{d-k}(\mathbf{v})\|$ is indeed equal to the expected norm (cf. Figure 3.2). We further assume that $\pi_i(\mathbf{v})$ is distributed in a random direction with respect to the rest of the basis. This assumption holds for LWE where the vector $\mathbf{e}$ is sampled from a (near) spherical distribution. We also note that we can rerandomize the basis and thus the relative directions. Under this assumption, we show that size reduction recovers the short vector $\mathbf{v}$ with high probability. More precisely, we show:

Heuristic 3.1. *Let $\mathbf{v} \in \Lambda \subset \mathbb{R}^d$ be a shortest non-zero vector as assumed in this section and $\beta \in \mathbb{N}$ be a block size. Assume that* (3.2) *holds, the current basis $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ is such that $\mathbf{b}_\kappa^* = \pi_\kappa(\mathbf{v})$ for $\kappa = d - \beta + 1$ and*

$$\mathbf{v} = \mathbf{b}_k + \sum_{i=1}^{k-1} \nu_i \mathbf{b}_i$$

for some $\nu_i \in \mathbb{Z}$, and the GSA holds for $\mathbf{B}$ until index κ. If the size reduction step of BKZ-β is called on $\mathbf{b}_\kappa$, it recovers $\mathbf{v}$ with high probability over the randomness of the basis.

Note that if BKZ has just found a projection of $\mathbf{v}$ at index κ, the current basis is as required by Heuristic 3.1. Now, let $\nu_i \in \mathbb{Z}$ denote the coefficients of $\mathbf{v}$ with respect to the basis $\mathbf{B}$, i.e.,

$$\mathbf{v} = \mathbf{b}_{d-\beta+1} + \sum_{i=1}^{d-\beta} \nu_i \mathbf{b}_i.$$

Let $\mathbf{b}_{d-\beta+1}^{(d-\beta+1)} = \mathbf{b}_{d-\beta+1}$, where the superscript denotes a step during size reduction. For $i = d-\beta, d-\beta-1, \dots, 1$ size reduction successively finds $\mu_i \in \mathbb{Z}$ such that

$$\mathbf{w}_i = \mu_i \pi_i(\mathbf{b}_i) + \pi_i(\mathbf{b}_{d-\beta+1}^{(i+1)}) = \mu_i \mathbf{b}_i^* + \pi_i(\mathbf{b}_{d-\beta+1}^{(i+1)})$$

is the shortest element in the coset

$$L_i := \{\mu \mathbf{b}_i^* + \pi_i(\mathbf{b}_{d-\beta+1}^{(i+1)}) | \mu \in \mathbb{Z}\}$$

and sets

$$\mathbf{b}_{d-\beta+1}^{(i)} := \mu_i \mathbf{b}_i + \mathbf{b}_{d-\beta+1}^{(i+1)}.$$

Note that if $\mathbf{b}_{d-\beta+1}^{(i+1)} = \mathbf{b}_{d-\beta+1} + \sum_{j=i+1}^{d-\beta} \nu_j \mathbf{b}_j$, as in the first step $i = d - \beta$, then we have that

$$\pi_i(\mathbf{v}) = \nu_i \mathbf{b}_i^* + \pi_i(\mathbf{b}_{d-\beta+1}^{(i+1)}) \in L_i$$

is contained in L_i and hence

$$L_i = \pi_i(\mathbf{v}) + \mathbb{Z}\mathbf{b}_i^*.$$

If the projection $\pi_i(\mathbf{v})$ is in fact the shortest element in L_i, for the newly defined vector $\mathbf{b}_{d-\beta+1}^{(i)}$ it also holds that

$$\mathbf{b}_{d-\beta+1}^{(i)} = \nu_i \mathbf{b}_i + \mathbf{b}_{d-\beta+1}^{(i+1)} = \mathbf{b}_{d-\beta+1} + \sum_{j=i}^{d-\beta} \nu_j \mathbf{b}_j.$$

Hence, if $\pi_i(\mathbf{v})$ is the shortest element in L_i for all i, size reduction finds the shortest vector

$$\mathbf{v} = \mathbf{b}_{d-\beta+1}^{(1)}$$

and inserts it into the basis at position $d - \beta + 1$, replacing $\mathbf{b}_{d-\beta+1}$.

It remains to argue that with high probability p for every i we have that the projection $\pi_i(\mathbf{v})$ is the shortest element in L_i. Assuming independence, the success probability p is given by

$$p = \prod_{i=1}^{d-\beta} p_i,$$

where the probabilities p_i are defined as

$$p_i = \Pr\left[\pi_i(\mathbf{v}) \text{ is the shortest element in } \pi_i(\mathbf{v}) + \mathbb{Z}\mathbf{b}_i^*\right].$$

For each i the probability p_i is equal to the probability that

$$\|\pi_i(\mathbf{v})\| < \min\{\|\pi_i(\mathbf{v}) + \mathbf{b}_i^*\|, \|\pi_i(\mathbf{v}) - \mathbf{b}_i^*\|\}$$

as illustrated in Figure 3.5. To approximate the probabilities p_i, we model them as

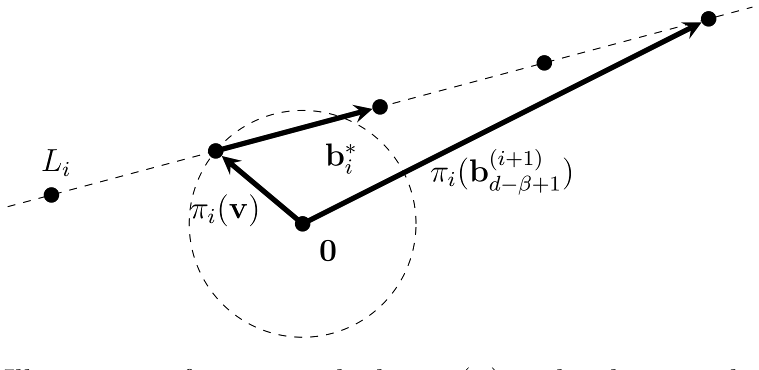

Figure 3.5: Illustration of a case such that $\pi_i(\mathbf{v})$ is the shortest element on L_i.

follows. By assumption, we have

$$r_i := \|\pi_i(\mathbf{v})\| = (\sqrt{d-i+1}/\sqrt{d})\,\|\mathbf{v}\| \text{ and } R_i := \|\mathbf{b}_i^*\| = \delta^{-2(i-1)+d}\det(\Lambda)^{\frac{1}{d}},$$

and that $\pi_i(\mathbf{v})$ is uniformly distributed with norm r_i. We can therefore model p_i as described in the following and illustrated in Figure 3.6.

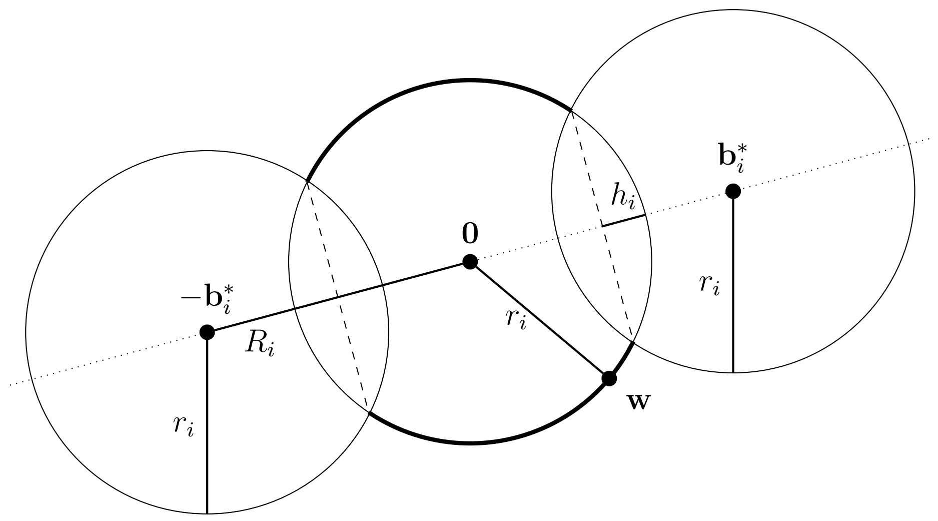

Figure 3.6: Illustration of the success probability p_i in $\mathbb{R}^2$. If $\mathbf{w}$ is on the thick part of the circle, step i of size reduction is successful.

Pick a point $\mathbf{w}$ with norm r_i uniformly at random. Then the probability p_i is approximately the probability that $\mathbf{w}$ is closer to $\mathbf{0}$ than it is to $\mathbf{b}_i^*$ and to $-\mathbf{b}_i^*$, i.e.

$$r_i < \min\{\|\mathbf{w}-\mathbf{b}_i^*\|, \|\mathbf{w}+\mathbf{b}_i^*\|\}.$$

Calculating this probability leads to the following approximation of p_i

$$p_i \approx \begin{cases} 1 - \frac{2A_{d-i+1}(r_i,h_i)}{A_{d-i+1}(r_i)} & \text{if } R_i < 2r_i \\ 1 & \text{if } R_i \geq 2r_i \end{cases},$$

where $A_{d-i+1}(r_i)$ is the surface area of the sphere in $\mathbb{R}^{d-i+1}$ with radius r_i and $A_{d-i+1}(r_i,h_i)$ is the surface area of the hyperspherical cap of the sphere in $\mathbb{R}^{d-i+1}$ with radius r_i of height h_i with $h_i = r_i - R_i/2$. Using the formulas provided in [Li11], an easy calculation leads to

$$p_i \approx \begin{cases} 1 - \frac{\int_0^{2\frac{h_i}{r_i}-\left(\frac{h_i}{r_i}\right)^2} t^{((d-i)/2)-1}(1-t)^{-1/2}dt}{B(\frac{d-i}{2},\frac{1}{2})} & \text{if } R_i < 2r_i \\ 1 & \text{if } R_i \geq 2r_i \end{cases},$$

where $B(\cdot,\cdot)$ denotes the Euler beta function. Note that $R_i \geq 2r_i$ corresponds to (3.3).

Estimated success probabilities p for different block sizes β are plotted in Figure 3.7. Note that if we assume equality holds in (3.2), the success probability p only depends on the block size β and not on the specific lattice dimension, determinant of the lattice, or the length of the unique short vector, since then the ratios between the predicted norms $\|\pi_{d-\beta+1-k}(\mathbf{v})\|$ and $\left\|\mathbf{b}^*_{d-\beta+1-k}\right\|$ only depend on β for all $k = 1, 2, \ldots$, since

$$\frac{\|\pi_{d-\beta+1-k}(\mathbf{v})\|}{\left\|\mathbf{b}^*_{d-\beta+1-k}\right\|} = \frac{\frac{\sqrt{\beta}\sqrt{\beta+k}}{\sqrt{\beta}\sqrt{d}}\,\|\mathbf{v}\|}{\delta^{2(\beta+k)-d}\det(\Lambda)^{\frac{1}{d}}} = \frac{\frac{\sqrt{\beta+k}}{\sqrt{\beta}}\delta^{2\beta-d}\det(\Lambda)^{\frac{1}{d}}}{\delta^{2(\beta+k)-d}\det(\Lambda)^{\frac{1}{d}}} = \frac{\sqrt{\beta+k}}{\sqrt{\beta}}\delta^{-2k}$$

and the estimated success probability only depends on these ratios.

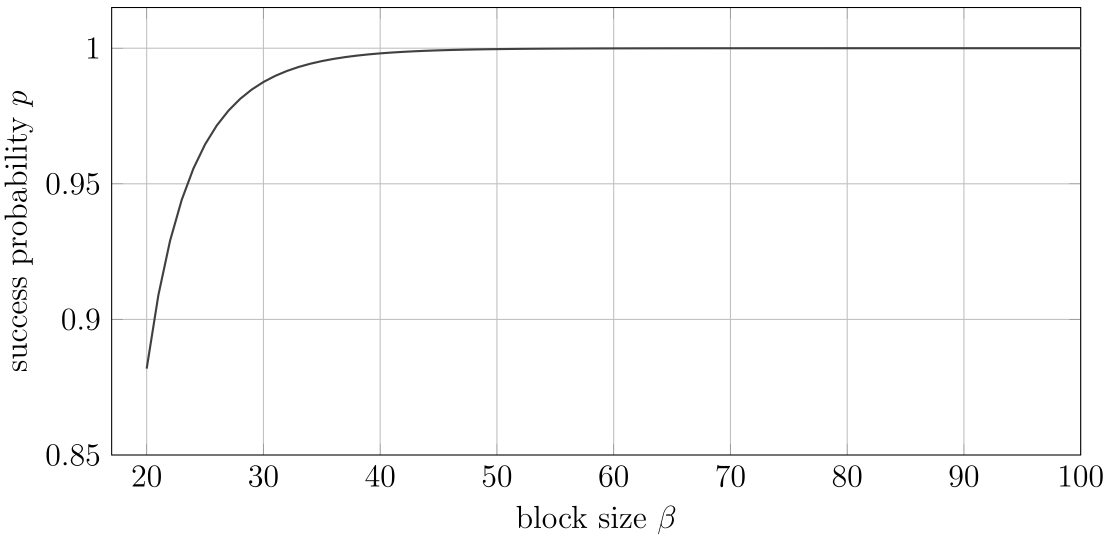

Figure 3.7: Estimated success probability p for varying block sizes β, assuming β is chosen minimal such that (3.2) holds.

The prediction given in Figure 3.7 is in line with the measured probability of finding $\mathbf{v}$ in the same step when its projection $\pi_{d-\beta+1}(\mathbf{v})$ is found as reported in Table 3.1 for $\beta = \beta_{2016}$ and $m = m_{2016}$. Finally, note that by the above analysis we do not expect to recover $\mathbf{v}$ from a projection $\pi_{d-\gamma+1}(\mathbf{v})$ for some small $\gamma \ll \beta$ except with small probability.

3.3 Applications

Section 3.2 indicates that (3.2) is a reliable condition for when lattice reduction will succeed in solving uSVP. Furthermore, as illustrated in Figure 3.1, applying (3.2) lowers the required block sizes compared to the 2008 model which is heavily relied upon in the literature. Thus, in this section we evaluate the impact of applying the revised 2016 estimate to various parameter sets from the literature. Indeed, for many schemes we find that their parameters need to be adapted to maintain the currently claimed level of security.

Many of the schemes considered below feature an unusually short LWE secret vector $\mathbf{s}$, where $s_i \leftarrow_{\$} \{-B, \dots, B\}$ for some small $B \in \mathbb{Z}_q$. Furthermore, some schemes pick the secret to also be sparse such that most components of $\mathbf{s}$ are zero. Thus, before we apply the revised 2016 estimate, we briefly recall the alternative embedding due to Bai and Galbraith [BG14b] which takes these small (and sparse) secrets into account.

3.3.1 Bai and Galbraith's embedding

Consider an LWE instance in matrix form $(\mathbf{A}, \mathbf{b}) \equiv (\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$. It holds that the vector $(\nu\,\mathbf{s} \mid \mathbf{e} \mid 1)$, for some $\nu \neq 0$, is contained in the lattice Λ

$$\Lambda = \left\{ \mathbf{x} \in (\nu\mathbb{Z})^n \times \mathbb{Z}^{m+1} \;\middle|\; \left(\frac{1}{\nu}\mathbf{A} \mid \mathbf{I}_m \mid -\mathbf{b}\right) \cdot \mathbf{x} \equiv \mathbf{0} \bmod q \right\}, \tag{3.4}$$

where ν allows to balance the size of the secret and the noise by rescaling the secret. An $(n + m + 1) \times (n + m + 1)$ basis $\mathbf{M}$ for Λ can be constructed as

$$\mathbf{M} = \begin{pmatrix} \nu\mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ -\mathbf{A} & q\mathbf{I}_m & \mathbf{b} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix}.$$

Indeed, $\mathbf{M}$ is full-rank, $|\det(\mathbf{M})| = \det(\Lambda)$, and the integer span of $\mathbf{M}$ is contained in Λ, as can be seen by

$$\left(\frac{1}{\nu}\mathbf{A} \mid \mathbf{I}_m \mid -\mathbf{b}\right) \cdot \begin{pmatrix} \nu\mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ -\mathbf{A} & q\mathbf{I}_m & \mathbf{b} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix} = (\mathbf{A} - \mathbf{A} \mid q\mathbf{I}_m \mid \mathbf{b} - \mathbf{b}) \equiv \mathbf{0} \bmod q.$$

Finally, note that $\mathbf{M} \cdot (\mathbf{s} \mid \mathbf{x} \mid 1) = (\nu\,\mathbf{s} \mid \mathbf{e} \mid 1)$ for some vector of $\mathbf{x}$. If $\mathbf{s}$ is small and/or sparse, choosing $\nu = 1$, the vector $(\mathbf{s} \mid \mathbf{e} \mid 1)$ is unbalanced, i.e., $\frac{\|\mathbf{s}\|}{\sqrt{n}} \ll \frac{\|\mathbf{e}\|}{\sqrt{m}} \approx \sigma$, where σ is the standard deviation of the LWE error distribution. We may then want to rebalance it by choosing an appropriate value of ν such that $\|(\nu\,\mathbf{s} \mid \mathbf{e} \mid 1)\| \approx \sigma\sqrt{n+m}$. Rebalancing preserves $(\nu\,\mathbf{s} \mid \mathbf{e} \mid 1)$ as the unique shortest non-zero vector in the lattice, while at the same time increasing the determinant of the lattice being reduced, reducing the block size required by (3.2).

If $\mathbf{s} \xleftarrow{\$} \{-1, 0, 1\}^n$ we expect $\|\nu\,\mathbf{s}\|^2 \approx \frac{2}{3}\nu^2 n$. Therefore, we can chose $\nu = \sqrt{\frac{3}{2}}\sigma$ to obtain $\|\nu\,\mathbf{s}\| \approx \sigma\sqrt{n}$, so that $\|(\mathbf{s} \mid \mathbf{e} \mid 1)\| \approx \sigma\sqrt{n+m}$. Similarly, if exactly $w < n$ entries of $\mathbf{s}$ are non-zero and chosen from $\{-1, 1\}$, we have $\|\nu\,\mathbf{s}\|^2 = w\,\nu^2$. Choosing $\nu = \sqrt{\frac{n}{w}}\sigma$, we obtain a vector $\nu\,\mathbf{s}$ of length $\sigma\sqrt{n}$.

In the case of sparse secrets, combinatorial techniques can also be applied, see Chapters 5, 6, and 7. In the following, we describe a more naive approach. Given a secret $\mathbf{s}$ with at most $w < n$ non-zero entries, we guess k entries of $\mathbf{s}$ to be 0, therefore decreasing the dimension of the lattice to consider. For each guess, we then apply lattice reduction to recover the remaining components of the vector $(\mathbf{s} \mid \mathbf{e} \mid 1)$. Therefore, when estimating the overall cost for solving such instances, we find $\min_k\{1/p_k \cdot C(n-k)\}$ where $C(n)$ is the cost of running BKZ on a lattice of dimension n and p_k is the probability of guessing correctly.

3.4 Security Estimates

In what follows, we assume that the geometry of Bai-Galbraith's embedding lattice is sufficiently close to that of Kannan's embedding lattice so that we transfer the analysis as is. Furthermore, in the provided tables we will denote applying (3.2) using Kannan's embedding for our estimates as "Our(K)" and applying (3.2) using Bai and Galbraith's embedding [BG14b] as "Our(BG)". Unless stated otherwise, we will assume that calling BKZ with block size β in dimension d costs $8d\,2^{0.292\,\beta+16.4}$ operations [BDGL16, Alb17], in particular that sieving is used as the SVP subroutine.

3.4.1 Lizard

Lizard [CKLS16b, CKLS16a] is a public-key encryption scheme based on the Learning With Rounding problem, using a small, sparse secret. The authors provide a reduction to LWE, and security parameters against classic and quantum adversaries, following their analysis. In particular, they cost BKZ by a single call to sieving on a block of size β. They estimate this call to cost $\beta\,2^{c\,\beta}$ operations where $c = 0.292$ for classical adversaries, $c = 0.265$ for quantum ones and $c = 0.2075$ as a lower bound for sieving ("paranoid"). Applying the revised 2016 cost estimate for the primal attack to the parameters suggested in [CKLS16b] (using their sieving cost model as described above) reduces the expected costs, as shown in Table 3.2. We note that in the meantime the authors of Lizard have updated their parameters in [CKLS16a].

3.4.2 HElib

HElib [GHS12a, GHS12b] is a fully homomorphic encryption library implementing the BGV scheme [BGH13]. A recent work [Alb17] provides revised security estimates for HELib by employing a dual attack exploiting the small and sparse secret, using the same cost estimate for BKZ as given at the beginning of this section. In Table 3.3 we provide costs for a primal attack using Kannan's and Bai and Galbraith's embeddings. Primal attacks perform worse than the algorithm described ind [Alb17], but, as expected, under the 2016 estimate the gap narrows.

	Classical			Quantum			Paranoid		
n, $\log_2 q$, σ	386, 11, 2.04			414, 11, 2.09			504, 12, 4.20		
Cost	β	d	λ	β	d	λ	β	d	λ
[CKLS16b]	418	—	130.8	456	—	129.7	590	—	131.6
Our(K)	**372**	805	**117.2**	**400**	873	**114.6**	**567**	1120	**126.8**
Our(BG)	**270**	646	**88.5**	**297**	692	**86.9**	**372**	833	**85.9**

Table 3.2: Cost estimates λ for solving Lizard PKE [CKLS16b] as given in [CKLS16b] and using Kannan's resp. Bai and Galbraith's embedding under the 2016 estimate. The dimension of the LWE secret is n. In all cases, BKZ-β is estimated to cost $\beta\, 2^{c\,\beta}$ operations.

3.4.3 SEAL

SEAL [CLP17] is a fully homomorphic encryption library by Microsoft based on the FV scheme [FV12]. Up to date parameters are given in [CLP17], using the same cost model for BKZ as mentioned at the beginning of this section. In Table 3.4, we provide cost estimates for Kannan's and Bai and Galbraith's embeddings under the 2016 estimate. Note that the gap in solving time between the dual and primal attack reported in [Alb17] is closed for SEAL v2.1 parameters.

3.4.4 TESLA

TESLA [BG14a, ABBD15] is a signature scheme based on LWE. Post-quantum secure parameters in the quantum random oracle model were recently proposed in [ABB$^+$17]. In Table 3.5, we show that these parameters need to be increased to maintain the currently claimed level of security under the 2016 estimate. Note that [ABB$^+$17] maintains a gap of roughly $\log_2 n$ bits of security between the best known attack on LWE and claimed security to account for a loss of security in the reduction.

3.4.5 BCIV17

[BCIV17] is a somewhat homomorphic encryption scheme obtained as a simplification of the FV scheme [FV12] and proposed as a candidate for enabling privacy friendly energy consumption forecast computation in smart grid settings. The authors propose parameters for obtaining 80 bits of security, derived using the estimator from [APS15] available at the time of publication. As a consequence of applying (3.2), we observe a moderate loss of security, as reported in Table 3.6.

	80 bit security														
n	1024			2048			4096			8192			16384		
$\log_2 q$, σ	47, 3.2			87, 3.2			167, 3.2			326, 3.2			638, 3.2		
Cost	β	d	λ	β	d	λ	β	d	λ	β	d	λ	β	d	λ
[Alb17] SILKE$_{\text{sparse}}$	105	—	61.3	111	—	65.0	112	—	67.0	123	—	70.2	134	—	73.1
Our(K)	**156**	2096	**76.0**	**166**	4003	**79.8**	**171**	7960	**82.3**	**176**	15606	**84.7**	**180**	31847	**86.9**
Our(BG)	**137**	1944	**70.3**	**152**	3906	**75.9**	**163**	7753	**79.9**	**169**	16053	**82.9**	**173**	32003	**85.9**
	128 bit security														
n	1024			2048			4096			8192			16384		
$\log_2 q$, σ	38, 3.2			70, 3.2			134, 3.2			261, 3.2			511, 3.2		
Cost	β	d	λ	β	d	λ	β	d	λ	β	d	λ	β	d	λ
[Alb17] SILKE$_{\text{sparse}}$	138	—	73.2	145	—	77.4	151	—	81.2	163	—	84.0	149	—	86.4
Our(K)	**225**	2076	**96.1**	**238**	4050	**100.9**	**245**	8011	**103.9**	**250**	16017	**106.4**	**257**	31635	**109.4**
Our(BG)	**189**	1901	**86.6**	**211**	3830	**94.4**	**204**	7348	**99.3**	**185**	13543	**102.8**	**204**	28236	**105.9**

Table 3.3: Solving costs for LWE instances underlying HELib as given in [Alb17] and using Kannan's resp. Bai and Galbraith's embedding under the 2016 estimate. The dimension of the LWE secret is n. In all cases, BKZ-β is estimated to cost $8d\,2^{0.292\,\beta+16.4}$ operations.

n, $\log_2 q$, σ	1024, 35, 3.19			2048, 60, 3.19			4096, 116, 3.19			8192, 226, 3.19			16384, 435, 3.19		
Cost	β	d	λ	β	d	λ	β	d	λ	β	d	λ	β	d	λ
[CLP17]	230	—	97.6	282	—	115.1	297	—	119.1	307	—	123.1	329	—	130.5
[Alb17]+	255	—	104.9	298	—	118.4	304	—	121.2	310	—	124.0	328	—	130.2
Our(K)	**257**	2085	**105.5**	**304**	4041	**120.2**	**307**	8047	**122.0**	**312**	15876	**124.5**	**328**	31599	**130.1**
Our(BG)	**237**	1984	**99.6**	**288**	4011	**115.5**	**299**	8048	**119.7**	**309**	15729	**123.6**	**326**	31322	**129.5**

Table 3.4: Solving costs for parameter choices in SEAL v2.1 as given in [CLP17], using [Alb17] as implemented in the current [APS15] estimator commit `84014b6` ("[Alb17]+"), and using Kannan's resp. Bai and Galbraith's embedding under the 2016 estimate. In all cases, BKZ-β is estimated to cost $8d\,2^{0.292\,\beta+16.4}$ operations.

	TESLA-0			TESLA-1			TESLA-2		
n, $\log_2 q$, σ	644, 31, 55			804, 31, 57			1300, 35, 73		
Cost	β	d	λ	β	d	λ	β	d	λ
				Classical					
[ABB$^+$17]	—	—	110.0	—	—	142.0	—	—	204.0
[ABB$^+$17]+	255	—	110.0	358	—	140.4	563	—	200.9
Our(K)	**248**	1514	**102.4**	**339**	1954	**129.3**	**525**	3014	**184.3**
				Post-Quantum					
[ABB$^+$17]	—	—	71.0	—	—	94.0	—	—	142.0
[ABB$^+$17]+	255	—	68.5	358	—	90.7	563	—	136.4
Our(K)	**248**	1415	**61.5**	**339**	1954	**81.1**	**525**	3014	**122.4**

Table 3.5: Cost estimates for solving TESLA parameter sets [ABB$^+$17]. The entry "[ABB$^+$17]+" refers to reproducing the estimates from [ABB$^+$17] using a current copy of the estimator from [APS15] which uses the embedding factor $M = 1$ instead of $M = \|\mathbf{e}\|$, as a consequence the values in the respective rows are slightly lower than in [ABB$^+$17]. We compare with Kannan's embedding under the 2016 estimate. Classically, BKZ-β is estimated to cost $8d\, 2^{0.292\,\beta+16.4}$ operations; quantumly BKZ-β is estimated to cost $8d\,\sqrt{\beta^{0.0225\,\beta} \cdot 2^{0.4574\,\beta}/2^{\beta/4}}$ operations in [ABB$^+$17].

80 bit security							
$n = 4096$, $\log_2 q = 186$, $\sigma = 102$							
Attack	β	d	λ	Attack	β	d	λ
Our(K)	**156**	8105	**77.9**	**Our(BG)**	**147**	7818	**75.3**

Table 3.6: Solving costs for proposed Ring-LWE parameters in [BCIV17] using Kannan's resp. Bai and Galbraith's embedding under the 2016 estimate. In both cases, BKZ-β is estimated to cost $8d\, 2^{0.292\,\beta+16.4}$ operations.

4 On the Use of Sparsification when Embedding BDD into uSVP

Kannan's embedding attack [Kan87] to solve LWE (see Section 2.4.4) corresponds to a *deterministic* reduction from BDD_α to uSVP_γ with $\gamma = \frac{1}{2\alpha}$, or more refined, with $\alpha = (2\lfloor\gamma\rfloor)/(2\gamma^2 + \lfloor\gamma\rfloor\lfloor\gamma+1\rfloor)$, see [BSW16, LM09, LWXZ14]. In 2016, Bai et al. [BSW16] presented a *probabilistic* reduction from BDD_α to uSVP_γ with $\gamma = \frac{1}{\sqrt{2}\alpha}$, improving the relation between the factors α and γ.[5] To achieve this improvement, so-called sparsification techniques [Kho03, Kho04, DK13, DRS14, SD16] are used prior to the embedding into uSVP, which is then solved using lattice reduction. Informally, sparsification chooses a random sublattice of the BDD lattice. With a certain probability, the BDD solution is contained in this sublattice, and in this case, BDD in the sublattice is potentially easier to solve than in the original one. So far, the implications of this improved reduction and the use of sparsification to the concrete hardness of LWE and BDD have not been studied.

Contribution. In this chapter, we consider a sparsified embedding attack on LWE (or BDD) which is deduced from the reduction presented in [BSW16]. We provide a detailed theoretical performance analysis of the sparsified embedding attack in practice and compare it to Kannan's embedding approach. Our analysis is based on the 2016 estimate [ADPS16] analyzed in Chapter 3 and common heuristics used in lattice-based cryptography. Our results show that, in general, using the sparsified embedding approach does not lead to a better attack on LWE compared to Kannan's embedding approach. This is due to the fact that the decrease in success probability introduced by sparsification in general is not compensated for or exceeded by the obtained speedup in the success case.

Organization. The details of the sparsified embedding attack are described in Section 4.1. Our performance analysis based on the 2016 estimate and a comparison to Kannan's embedding attack are provided in Section 4.2.

[5] BDD_α is easier for smaller values of α, while uSVP_γ is easier for larger values of γ.

Publications. This chapter is based on the publication [6], which will be presented at ISPEC 2018.

4.1 The Sparsified Embedding Attack

In the following we describe a sparsified embedding attack on LWE which can be deduced from [BSW16]. The sparsified embedding approach is similar to Kannan's embedding (see Section 2.4.4). The main difference is that the BDD lattice $\Lambda_{(\mathbf{A},q)} = \{\mathbf{v} \in \mathbb{Z}_q^m \mid \mathbf{v} \equiv \mathbf{A}\mathbf{x} \pmod{q} \text{ for some } \mathbf{x} \in \mathbb{Z}^n\}$ is sparsified prior to embedding it into a uSVP lattice. The sparsification technique was first introduced by Khot [Kho03, Kho04], and specified in [DK13, DRS14, SD16]. Roughly speaking, sparsifying a lattice means choosing a random sublattice of some index p. In more detail, let p be the desired index and $\mathbf{B}$ be a basis of $\Lambda_{(\mathbf{A},q)}$. Sample $\mathbf{z}$ and $\mathbf{u}$ uniformly and independently from $\mathbb{Z}_p^m$ and set $\mathbf{w} = \mathbf{B}\mathbf{u}$. If $\|\mathbf{b}+\mathbf{w}\| < (m+1)l_0/\sqrt{2}$, where the parameter l_0 is chosen as described in [BSW16], resample $\mathbf{u}$ until $\|\mathbf{b}+\mathbf{w}\| \geq (m+1)l_0/\sqrt{2}$. The vector $\mathbf{z}$ is used to sparsify the lattice $\Lambda_{(\mathbf{A},q)}$ and $\mathbf{w}$ is used to offset the target vector $\mathbf{b}$. The sparsified lattice $\Lambda_{p,\mathbf{z}}$ of $\Lambda_{(\mathbf{A},q)}$ is now defined as

$$\Lambda_{p,\mathbf{z}} = \{\mathbf{v} \in \Lambda(\mathbf{B}) \mid \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{v}\rangle = 0 \mod p\}.$$

If $\mathbf{z} \neq \mathbf{0}$ then $\Lambda_{p,\mathbf{z}}$ is a sublattice of $\Lambda_{(\mathbf{A},q)}$ of index p as shown in the following lemma.

Lemma 4.1. *Let Λ be a d-dimensional full-rank lattice, $\mathbf{B}$ be a basis of Λ, p be some prime, $\mathbf{z} \in \mathbb{Z}_p^n \setminus \{\mathbf{0}\}$ and $\Lambda_{p,\mathbf{z}} = \{\mathbf{v} \in \Lambda(\mathbf{B}) \mid \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{v}\rangle = 0 \bmod p\}$. Then for the index of the subgroup $\Lambda_{p,\mathbf{z}}$ of Λ it holds that $[\Lambda : \Lambda_{p,\mathbf{z}}] = p$.*

Proof. Consider the homomorphism

$$\varphi : \Lambda \to (\mathbb{Z}_p, +), \quad \mathbf{v} \mapsto \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{v}\rangle \bmod p.$$

We first show that φ is surjective. Let j be an index with $z_j \neq 0$. Let a be some arbitrary element in $\mathbb{Z}_p$. Then for $\mathbf{v} = \mathbf{B}\mathbf{x}$, where $\mathbf{x} \in \mathbb{Z}^n$ with $x_i = 0$ for $i \neq j$ and $x_j = (z_j^{-1} \bmod p)a$, it holds that $\varphi(\mathbf{v}) = a$. Hence φ is surjective and by the isomorphism theorem we have

$$\Lambda/\Lambda_{p,\mathbf{z}} = \Lambda/\ker(\varphi) \simeq \mathsf{im}(\varphi) = \mathbb{Z}_p \quad \text{and} \quad [\Lambda : \Lambda_{p,\mathbf{z}}] = p.$$

□

A basis $\mathbf{B}_{p,\mathbf{z}}$ of $\Lambda_{p,\mathbf{z}}$ is constructed (as described in Lemma 9 of [BSW16]) and then embedded into

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B}_{p,\mathbf{z}} & \mathbf{b}+\mathbf{w} \\ \mathbf{0} & M \end{pmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)}$$

using the target vector $\mathbf{b} + \mathbf{w}$. How to choose the embedding factor M for the proof of the reduction is described in [BSW16]. However, as typical for Kannan's embedding approach, we choose $M = 1$. Finally, a shortest non-zero vector $\mathbf{v}$ of $\Lambda(\mathbf{B}')$ is recovered by lattice reduction and the vector consisting of its first m components is returned. Note that the output is not necessarily given by $\pm\mathbf{e}$, hence the attack is not always successful. This is the case because the attack can only succeed in recovering $\mathbf{e}$ if the vector closest to $\mathbf{b} + \mathbf{w}$ in $\Lambda_{(\mathbf{A},q)}$, namely $\mathbf{b} + \mathbf{w} - \mathbf{e}$, is also contained in $\Lambda_{p,\mathbf{z}}$. If the sparsified lattice $\Lambda_{p,\mathbf{z}}$ is chosen randomly as described above, the success probability of the attack is roughly $1/p$, see Corollary 2.17 in [SD16] and Lemma 13 in [BSW16]. For more details on sparsification, we refer to [BSW16]. The pseudocode for a simple version of the sparsified embedding attack on LWE is given in Algorithm 4.

Algorithm 4: The sparsified embedding approach

```
  Input: An LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e} \mod q) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, a prime $p$,
         and $l_0 > 0$, embedding factor $M$
1 Construct a lattice basis $\mathbf{B} \in \mathbb{Z}^{m\times m}$ of the lattice
    $\Lambda_{(\mathbf{A},q)} = \{\mathbf{v} \in \mathbb{Z}_q^m \mid \mathbf{v} \equiv \mathbf{Ax} \pmod q \text{ for some } \mathbf{x} \in \mathbb{Z}^n\}$ ;
2 Sample $\mathbf{z}$ and $\mathbf{u}$ uniformly and independently from $\mathbb{Z}_p^m$ and set $\mathbf{w} = \mathbf{Bu}$ until
    $\|\mathbf{b} + \mathbf{w}\| \geq (m+1)l_0/\sqrt{2}$;
3 Construct a lattice basis $\mathbf{B}_{p,\mathbf{z}}$ of the sparsified lattice
    $\Lambda_{p,\mathbf{z}} = \{\mathbf{v} \in \Lambda(\mathbf{B}) \mid \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{v}\rangle = 0 \mod p\}$;
4 Set $\mathbf{B}' = \begin{pmatrix} \mathbf{B}_{p,\mathbf{z}} & \mathbf{b}+\mathbf{w} \\ \mathbf{0} & M \end{pmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)}$;
5 Recover $\mathbf{v} = \begin{pmatrix} \mathbf{x} \\ y \end{pmatrix}$ by solving (u)SVP in $\Lambda(\mathbf{B}')$ using lattice reduction;
6 return $\mathbf{x}$;
```

4.2 Analysis

In [BSW16], it is shown that the sparsified embedding yields an improved reduction from BDD_α to uSVP_γ compared to Kannan's embedding in the sense that it gives better gaps ($\gamma = \frac{1}{\sqrt{2}\alpha}$ instead of $\gamma = \frac{1}{2\alpha}$). This improvement, however, comes at the cost of a probabilistic reduction instead of a deterministic one. In this section, we theoretically analyze and compare the practical behavior of both embedding approaches under common heuristics used in lattice-based cryptography. Note that the practical behavior substantially differs from the provable reductions, since in those reductions "worst cases" that can occur need to be taken into account while the practical behavior is determined by the average case. Let Λ_s be the embedded sparsified lattice of dimension d. From the 2016 estimate (cf. Chapter 3), it can be deduced that the sparsified embedding attack succeeds if the unique shortest non-zero vector is contained in Λ_s and the block size β satisfies

$$\sqrt{\beta/d}\lambda_1(\Lambda_s) \leq \delta^{2\beta-d}\det(\Lambda_s)^{1/d}.$$

In the following, we elaborate on this assumption by analyzing how to solve BDD using the two embedding approaches (the results carry over to LWE if viewed as an instance of BDD as described in Section 2.4.4).

4.2.1 Heuristics for Kannan's Embedding

For Kannan's embedding, most works considered with the practicality of the attack implicitly assume that there is no reduction loss in practice, i.e., that $\gamma = \frac{1}{\alpha}$ instead of $\gamma = \frac{1}{2\alpha}$. In the following, we elaborate on this assumption. For simplicity, we ignore the extra dimension induced by the embedding. Let Λ be the BDD lattice, d be the dimension of Λ, and Λ' be the uSVP lattice obtained by using Kannan's technique for the BDD lattice Λ and the BDD target vector $\mathbf{t}$ as described in Section 4.1. Let $\alpha = \mathsf{dist}(\mathbf{t}, \Lambda)/\lambda_1(\Lambda)$ be the factor of the BDD instance and $\gamma = \lambda_2(\Lambda')/\lambda_1(\Lambda')$ be the gap of the resulting uSVP instance. In practice, it is common (see for example [APS15, AGVW17]) to make the following heuristic assumptions.

1. Under the assumption that Λ is a random lattice, $\lambda_1(\Lambda)$ corresponds to the Gaussian heuristic for Λ.
2. As Kannan's embedding adds the uniquely distance short vector from $\mathbf{t}$ to the nearest lattice point to the lattice, we can assume that $\lambda_1(\Lambda')$ corresponds to $\mathsf{dist}(\mathbf{t}, \Lambda) = \alpha\lambda_1(\Lambda)$, i.e., $\lambda_1(\Lambda') = \alpha\lambda_1(\Lambda)$.
3. Under the assumption that except for this uniquely short vector Λ' behaves as a random lattice, we can assume that $\lambda_2(\Lambda')$ corresponds to the Gaussian heuristic for Λ', which is the same as the Gaussian heuristic for Λ, i.e., $\lambda_2(\Lambda') = \lambda_1(\Lambda)$.
4. In conclusion, we obtain $\frac{1}{\alpha} = \frac{\lambda_1(\Lambda)}{\lambda_1(\Lambda')} = \frac{\lambda_2(\Lambda')}{\lambda_1(\Lambda')} = \gamma$.

This shows that heuristically, Kannan's embedding approach performs much better in practice than guaranteed by the theoretical reduction, which only guarantees the gap $\frac{1}{2\alpha}$.

It remains to determine the necessary block size for BKZ to solve such an instance. According to the 2016 estimate (see Chapter 3), the Gaussian heuristic, and $\gamma = \frac{1}{\alpha}$, we get that the required block size β is the minimal β that satisfies

$$\alpha = \frac{1}{\gamma} \leq \sqrt{\frac{2\pi e}{\beta}}\delta^{2\beta-d} = \sqrt{\frac{2\pi e}{\beta}}\left((((\pi\beta)^{1/\beta}\beta)/(2\pi e))^{1/(2(\beta-1))}\right)^{2\beta-d}.$$

In the LWE case, parameterized by the secret dimension n, the number of samples m, the modulus q, and the standard deviation σ of the error distribution, we may instead use the condition

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

since according to the Gaussian heuristic the gap can be estimated as

$$\alpha = \frac{\lambda_1(\Lambda)}{\lambda_2(\Lambda)} = \frac{\sigma\sqrt{d}}{\sqrt{d/(2\pi e)}\det(\Lambda)^{1/d}} = \frac{\sigma\sqrt{2\pi e}}{(q^{m-n})^{1/d}}.$$

This condition takes the extra dimension introduced by the embedding into account (i.e., $d = m + 1$) and corresponds to the 2016 estimate for LWE (cf. Chapter 3).

4.2.2 Heuristics for the Sparsified Embedding

In this section, we analyze how the sparsified embedding approach performs in practice, assuming that the heuristics presented in Section 4.2.1 are reasonable. Let Λ, Λ', d, α, $\mathbf{t}$, and γ be as in Section 4.2.1. Let p be the prime number used for the sparsification of Λ and $\Lambda_s \subset \Lambda$ be some sparsified sublattice of Λ with $[\Lambda : \Lambda_s] = p$. Then it holds that $\det(\Lambda_s) = p \cdot \det(\Lambda)$. If the sparsification is random (as described in the reduction), then the probability to keep the closest vector in Λ to the target $\mathbf{t}$ in the sparsified lattice Λ_s is roughly $1/p$. So the probability that one can solve the BDD problem at all in the sparsified lattice is close to $1/p$. Assume that we are in the success case, i.e., the closest lattice vector in Λ to the target $\mathbf{t}$ is kept in the sparsified lattice Λ_s. Let Λ'_s be the embedded lattice of Λ_s. Again, for simplicity, we ignore the additional dimension of Λ'_s. Then, similarly to Section 4.2.1, we can apply the following heuristics.

1. $\lambda_1(\Lambda_s)$ corresponds to the Gaussian heuristic for Λ_s which yields $\lambda_1(\Lambda_s) = p^{1/d}\lambda_1(\Lambda)$.
2. $\lambda_1(\Lambda'_s)$ corresponds to $\mathsf{dist}(\mathbf{t}, \Lambda_s) = \mathsf{dist}(\mathbf{t}, \Lambda) = \alpha\lambda_1(\Lambda)$, i.e., $\lambda_1(\Lambda'_s) = \alpha\lambda_1(\Lambda) = \lambda_1(\Lambda')$.
3. $\lambda_2(\Lambda'_s)$ corresponds to the Gaussian heuristic for Λ'_s, which is the same as the Gaussian heuristic for Λ_s, i.e., $\lambda_2(\Lambda'_s) = \lambda_1(\Lambda_s) = p^{1/d}\lambda_1(\Lambda) = p^{1/d}\lambda_2(\Lambda')$.
4. Let γ_s be the uSVP gap in Λ'_s. Then we get $\gamma_s = \frac{\lambda_2(\Lambda'_s)}{\lambda_1(\Lambda'_s)} = \frac{p^{1/d}\lambda_2(\Lambda')}{\lambda_1(\Lambda')} = p^{1/d}\gamma = p^{1/d}\frac{1}{\alpha}$.

In conclusion, heuristically the gap of the sparsified embedding technique $\gamma_s = p^{1/d}\frac{1}{\alpha}$ improves by a factor of $p^{1/d}$ compared to Kannan's embedding, and of course it improves the gap $\frac{1}{\sqrt{2}\alpha}$ guaranteed by the theoretical reduction. Note however, that this improvement comes at the cost of a success probability of (roughly) $\frac{1}{p}$.

It remains to determine the necessary block size for BKZ to solve such an instance according to the 2016 estimate. Similar as above, for the success case with $\gamma_s = p^{1/d}\frac{1}{\alpha}$, we get that the required block size β is the minimal β that satisfies

$$\alpha = \frac{1}{\gamma} \leq p^{1/d}\sqrt{\frac{2\pi e}{\beta}}\delta^{2\beta-d} = p^{1/d}\sqrt{\frac{2\pi e}{\beta}}\left(\left(((\pi\beta)^{1/\beta}\beta)/(2\pi e)\right)^{1/(2(\beta-1))}\right)^{2\beta-d}.$$

In the LWE case parameterized by n, m, q, and σ as above we may instead use the condition

$$\sqrt{\beta}\sigma \leq \delta^{2\beta-(m+1)}(pq^{m-n})^{1/(m+1)}.$$

4.2.3 Comparison

As shown in Sections 4.2.1 and 4.2.2, the heuristic improvement of using sparsification in the embedding approach is a factor of $p^{1/d}$ in the uSVP gap which results in a smaller necessary block size for BKZ to solve the resulting uSVP problem. In the following, we further analyze this improvement. First, note that if $p = p(d)$ is chosen to be polynomial in the lattice dimension d, the improvement factor $p^{1/d}$ tends to 1 as d increases, i.e., asymptotically, the improvement vanishes. On the other hand, if $p = p(d)$ is chosen to be exponential in d, the success probability of roughly $1/p$ is negligible. Therefore, to possibly achieve an overall improvement in practice, taking the success probability into account, p must be chosen carefully for the specific instance.

In Table 4.1, we show the predicted minimal block sizes for BKZ according to the 2016 estimate required by Kannan's and the sparsified embedding approach for BDD instances of various parameter sets. As indicated by these examples, the benefit of using sparsification depends on different parameters. In Table 4.2, we show the same for the LWE instances analyzed in Chapter 3. The results show that, for the analyzed instances, one needs to considerably increase p in order to get a moderate decrease of the required block size. This, however, implies, that getting a moderate speed up in the success case comes at the price of a low success probability of roughly $1/p$. For the analyzed instances, one can therefore predict that the sparsified embedding approach performs worse than Kannan's (assuming a reasonable cost model for BKZ).

$d = 256$, $\alpha = 1/2$			$d = 256$, $\alpha = 1/4$		
$1 \leq p \leq 3$	$5 \leq p \leq 59$	$61 \leq p \leq 751$	$1 \leq p \leq 31$	$37 \leq p \leq 1899$	$1901 \leq p \leq 119563$
$\beta = 157$	$\beta = 156$	$\beta = 155$	$\beta = 101$	$\beta = 100$	$\beta = 99$
$d = 512$, $\alpha = 1/2$			$d = 512$, $\alpha = 1/4$		
$1 \leq p \leq 5$	$7 \leq p \leq 127$	$131 \leq p \leq 2447$	$1 \leq p \leq 3$	$5 \leq p \leq 409$	$419 \leq p \leq 42257$
$\beta = 350$	$\beta = 349$	$\beta = 348$	$\beta = 253$	$\beta = 252$	$\beta = 251$
$d = 1024$, $\alpha = 1/2$			$d = 1024$, $\alpha = 1/4$		
$1 \leq p \leq 11$	$13 \leq p \leq 349$	$353 \leq p \leq 9661$	$1 \leq p \leq 47$	$53 \leq p \leq 7309$	$7321 \leq p \leq 1063399$
$\beta = 748$	$\beta = 747$	$\beta = 746$	$\beta = 572$	$\beta = 571$	$\beta = 570$

Table 4.1: Minimal block sizes β according to the 2016 estimate for various dimensions d, factors α, and primes p. The exception $p = 1$ indicates that no sparsification is used.

$n = 65$, $m = 182$, $q = 521$, $\sigma = 8/\sqrt{2\pi}$		
$1 \leq p \leq 23$	$29 \leq p \leq 887$	$907 \leq p \leq 27953$
$\beta = 56$	$\beta = 55$	$\beta = 54$
$n = 100$, $m = 243$, $q = 2053$, $\sigma = 8/\sqrt{2\pi}$		
$1 \leq p \leq 113$	$127 \leq p \leq 21859$	$21863 \leq p \leq 4141603$
$\beta = 67$	$\beta = 66$	$\beta = 65$
$n = 108$, $m = 261$, $q = 2053$, $\sigma = 8/\sqrt{2\pi}$		
$1 \leq p \leq 163$	$167 \leq p \leq 36523$	$36527 \leq p \leq 8485031$
$\beta = 77$	$\beta = 76$	$\beta = 75$

Table 4.2: Minimal block sizes β according to the 2016 estimate for various LWE instances parameterized by the secret dimension n, the number of samples m, the modulus q, and the standard deviation σ of the error distribution and for various primes p. The exception $p = 1$ indicates that no sparsification is used.

5 Revisiting the Hybrid Lattice Reduction and Meet-in-the-Middle Attack

Over the recent years, several cryptographic schemes based on lattice problems with particularly small (e.g., binary) and/or sparse vectors have been proposed, e.g., [HPS98, BCLvV17b, BGG$^+$16, DDLL13, GLP12]. In order to evaluate the security of such schemes, it is not sufficient to estimate the runtimes of general lattice attacks (such as the ones discussed in Chapters 3 and 4), but in addition it is important to consider attacks that are specifically designed to solve such special instances of lattice problems. One such attack is the "hybrid lattice reduction and meet-in-the-middle attack" [HG07] (referred to as the hybrid attack in the following) against the NTRU encryption scheme [HPS98] proposed by Howgrave-Graham in 2007. Several works [HG07, HHGP$^+$07, HHHGW09, HPS$^+$17, Sch15] claim that the hybrid attack is by far the best known attack on NTRUEncrypt. In the following years, numerous cryptographers have applied the hybrid attack to their schemes in order to estimate their security. These considerations include more variants of the NTRU encryption scheme [HHHGW09, HPS$^+$17, Sch15], the recently proposed encryption scheme NTRU prime [BCLvV17b, BCLvV16], and the signature schemes BLISS [DDLL13] and GLP [GLP12, DDLL13]. However, so far a framework to apply the hybrid attack to a larger class of lattice problems with small or sparse secret vectors, in particular LWE with small or sparse error distributions, has not been proposed. In addition, all of the analyses of the hybrid attack mentioned above suffer from the use of over-simplifying assumptions which may distort the accuracy of the security estimates, as pointed out in [Sch15]. Therefore, an important challenge is to provide a detailed analysis of the hybrid attack in a framework which is applicable to a large class of lattice problems.

Contribution. In this chapter, we address this challenge in the following way. We present a generalized framework for the hybrid attack applied to the uSVP. This general framework for the hybrid attack can naturally be applied to many lattice-based cryptocraphic constructions. We provide a detailed analysis of the generalized version of the hybrid attack, improving on previous considerations in the literature. Our improvements include explicit calculations of the probability of finding collisions in the meet-in-the-middle search. Finally, we apply our improved analysis to reevaluate the security of the following cryptographic schemes against the hybrid attack: the NTRU [HPS+17], NTRU prime [BCLvV17b, BCLvV16], and R-BinLWEEnc [BGG+16] encryption schemes and the BLISS [DDLL13] and GLP [GLP12] signature schemes. Our results show that there exist both security over- and underestimates against the hybrid attack across the literature. We further compare our results to security estimates derived from the 2016 estimate (cf. Chapter 3) to showcase the improvement of the hybrid attack over a pure lattice reduction attack on uSVP with small and/or sparse secret vectors.

Organization. In Section 5.1, we provide some useful tools for q-ary lattices. Our uSVP framework for the hybrid attack is presented in Section 5.2. In Section 5.3, we provide our improved analysis of the hybrid attack in the generalized framework. We apply our new analysis of the hybrid attack to various cryptographic schemes to derive updated security estimates and compare our results to the primal attack under the 2016 estimate in Section 5.4.

Publications. This chapter is based on the publications [1], which was presented at AFRICACRYPT 2016, and [2], which will appear in the Journal of Mathematical Cryptology.

5.1 Tools for q-ary Lattices

In this section, we provide some useful tools for q-ary lattices.

5.1.1 Constructing a Suitable Basis for the Hybrid Attack

The hybrid attack requires a lattice of the form

$$\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m \times m}$$

for some dimensions m and r. In the following lemma we show that for q-ary lattices, where q is prime, there always exists a basis of this form for a suitable r depending on the determinant of the lattice. In the proof we also show how to construct such a basis.

Lemma 5.1. *Let q be prime, $m \in \mathbb{N}$, and $\Lambda \subset \mathbb{Z}^m$ a q-ary lattice.*

1. *There exists some $k \in \mathbb{Z}, 0 \leq n \leq m$ such that $\det(\Lambda) = q^k$.*
2. *Let $\det(\Lambda) = q^k$. Then there is a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times (m-k)}$ of rank $m-k$ (over $\mathbb{Z}_q$) such that $\Lambda = \Lambda_q(\mathbf{A})$.*

3. *Let* $\det(\Lambda) = q^k$ *and* $\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}$ *with* $\mathbf{A}_1 \in \mathbb{Z}_q^{k\times(m-k)}, \mathbf{A}_2 \in \mathbb{Z}_q^{(m-k)\times(m-k)}$ *be a matrix of rank* $m-k$ *(over* $\mathbb{Z}_q$*) such that* $\Lambda = \Lambda_q(\mathbf{A})$*. If* $\mathbf{A}_2$ *is invertible over* $\mathbb{Z}_q$*, then the columns of the matrix*

$$\mathbf{B}' = \left(\begin{array}{c|c} q\mathbf{I}_k & \mathbf{A}_1\mathbf{A}_2^{-1} \\ \hline \mathbf{0} & \mathbf{I}_{m-k} \end{array}\right) \in \mathbb{Z}^{m\times m} \tag{5.1}$$

form a basis of the lattice Λ.

Proof. 1. As $q\mathbb{Z}^m \subset \Lambda$ it holds that $\det(\Lambda) \mid \det(q\mathbb{Z}^m) = q^m$ and therefore $\det(\Lambda)$ is some non-negative power of q, because q is prime.

2. For the group index $[\Lambda : q\mathbb{Z}^m]$ we have $[\Lambda : q\mathbb{Z}^m] = \det(q\mathbb{Z}^m)/\det(\Lambda) = q^{m-k}$. Let $\mathbf{A}' \in \mathbb{Z}_q^{m\times m}$ be some lattice basis of Λ. Since $\Lambda/q\mathbb{Z}^m$ is in one-to-one correspondence to the $\mathbb{Z}_q$–vector space spanned by $\mathbf{A}'$, this vector space has to be of dimension $m-k$ and therefore $\mathbf{A}'$ has rank $m-k$ over $\mathbb{Z}_q$. This implies that there is some matrix $\mathbf{A}$ consisting of $m-k$ columns of $\mathbf{A}'$ such that $\Lambda = \Lambda(q\mathbf{I}_m \mid \mathbf{A}) = \Lambda_q(\mathbf{A})$.

3. By assumption $\mathbf{A}_2$ is invertible and thus we have

$$\begin{aligned} \Lambda &= \left\{\mathbf{v} \in \mathbb{Z}^m \mid \exists \mathbf{w} \in \mathbb{Z}^{(m-k)} : \mathbf{v} = \mathbf{A}\mathbf{w} \mod q\right\} \\ &= \left\{\mathbf{v} \in \mathbb{Z}^m \mid \exists \mathbf{w} \in \mathbb{Z}^{(m-k)} : \mathbf{v} = \begin{pmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} \mathbf{A}_2^{-1}\mathbf{w} \mod q\right\} \\ &= \left\{\begin{pmatrix} \mathbf{A}_1\mathbf{A}_2^{-1} \\ \mathbf{I}_{m-k} \end{pmatrix} \mathbf{w} \mid \mathbf{w} \in \mathbb{Z}^{(m-k)}\right\} + q\mathbb{Z}^m. \end{aligned}$$

Therefore the columns of the matrix

$$\left(q\mathbf{I}_m \middle| \begin{array}{c} \mathbf{A}_1\mathbf{A}_2^{-1} \\ \mathbf{I}_{m-k} \end{array}\right) \in \mathbb{Z}^{m\times(m+(m-k))}$$

form a generating set of the lattice Λ, which can be reduced to the basis $\mathbf{B}'$.

□

5.1.2 Modifying the GSA for q-ary Lattices

Typically, the Gram-Schmidt lengths of a lattice basis obtained after performing BKZ with a certain block size (or root Hermite factor) can be approximated the Geometric Series Assumption (GSA), see Chapter 2. However, for bases of q-ary lattices of the form as constructed in Lemma 5.1, this assumption can be modified to give better predictions. This has already been considered and confirmed with experimental results in previous works, see for example [HG07, HHHGW09, HPS$^+$17, Sch15]. In this section, we derive simple formulas predicting the Gram-Schmidt lengths of a reduced basis for q-ary lattices, given a basis of a certain form. We begin by sketching the reason why the unmodified GSA should be modified for q-ary lattices, given an input basis $\mathbf{B}$ of the form

$$\mathbf{B} = \left(\begin{array}{c|c} q\mathbf{I}_a & \star \\ \hline \mathbf{0} & \mathbf{I}_b \end{array}\right) \in \mathbb{Z}^{d\times d},$$

where $d = a + b$. How to construct such a basis for a q-ary lattice is shown in Section 5.1.1. For a relatively small block size (equivalently a large root Hermite factor) the GSA predicts that the first Gram-Schmidt vectors of the reduced basis have norm bigger than q. However, in practice this will not happen, since in this case the first vectors will simply not be modified by the reduction. This means, that instead of reducing the whole basis $\mathbf{B}$, one can just consider reducing the last vectors that will actually be reduced. Let k denote the (so far unknown) number of the last vectors that are actually reduced (i.e., their corresponding Gram-Schmidt vectors according to the GSA have norm smaller than q). In the following, we assume that the applied block size is small enough such that $k < d$ but sufficiently large such that $k > b$. We write $\mathbf{B}$ in the form

$$\mathbf{B} = \left(\begin{array}{c|c} q\mathbf{I}_{d-k} & \mathbf{D} \\ \hline \mathbf{0} & \mathbf{B}_1 \end{array}\right)$$

for some $\mathbf{B}_1 \in \mathbb{Z}^{k\times k}$ and $\mathbf{D} \in \mathbb{Z}^{(d-k)\times k}$. Now instead of $\mathbf{B}$ we only reduce $\mathbf{B}_1$ to $\mathbf{B}_1' = \mathbf{B}_1\mathbf{U}$ for some unimodular matrix $\mathbf{U} \in \mathbb{Z}^{k\times k}$. This yields a reduced basis

$$\mathbf{B}' = \left(\begin{array}{c|c} q\mathbf{I}_{d-k} & \mathbf{DU} \\ \hline \mathbf{0} & \mathbf{B}_1' \end{array}\right)$$

of $\mathbf{B}$. The Gram-Schmidt basis of this new basis $\mathbf{B}'$ is given by

$$(\mathbf{B}')^* = \left(\begin{array}{c|c} q\mathbf{I}_{d-k} & \mathbf{0} \\ \hline \mathbf{0} & (\mathbf{B}_1')^* \end{array}\right).$$

Therefore, the lengths of the Gram-Schmidt basis vectors $(\mathbf{B}')^*$ are q for the first $d-k$ vectors and then equal to the lengths of the Gram-Schmidt basis vectors $(\mathbf{B}_1')^*$, which are smaller than q. In order to predict the lengths of $(\mathbf{B}')^*$ we can apply the GSA to the lengths of the Gram-Schmidt basis vectors $(\mathbf{B}_1')^*$. What remains is to determine k. Assume applying BKZ on $\mathbf{B}_1$ with the given block size results in a reduced basis $\mathbf{B}_1'$ of root Hermite factor δ. By our construction we can assume that the first Gram-Schmidt basis vector of $(\mathbf{B}_1')^*$ has norm roughly equal to q, so the GSA implies

$$\delta^k \det(\Lambda(\mathbf{B}_1))^{\frac{1}{k}} = q.$$

Using the fact that $\det(\Lambda(\mathbf{B}_1)) = q^{k-b}$ and $k < d$, we can solve for k and obtain

$$k = \min\left(\left\lfloor \sqrt{\frac{b}{\log_q(\delta)}} \right\rfloor, d\right). \tag{5.2}$$

Summarizing, we expect that after lattice reduction our Gram-Schmidt basis $(\mathbf{B}_1')^*$ has lengths $\|\mathbf{b}_1^*\|, \dots, \|\mathbf{b}_d^*\|$, where

$$\|\mathbf{b}_i^*\| = \begin{cases} q, & \text{if } i \leq d-k \\ \delta^{-2(i-(d-k)-1)+k} q^{\frac{k-b}{k}}, & \text{else} \end{cases} \tag{5.3}$$

and k is given as in Equation 5.2.

Note that it might also happen that the last Gram-Schmidt lengths are predicted to be smaller than 1. In this case, these last vectors may also not be reduced in practice, since the basis matrix has the identity in the bottom right corner. Therefore, in this case the GSA can be further modified. However, for realistic attack parameters this phenomenon never occurred in our considerations and therefore we do not include it in our formulas and leave it to the reader to perform the calculations if needed.

5.2 The Hybrid Attack

In this section, we present a generalized version of the hybrid attack to solve unique shortest vector problems. Our framework for the hybrid attack is the following: the task is to find a (unique) shortest non-zero vector $\mathbf{v}$ in a lattice Λ, given a basis of Λ of the form

$$\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m \times m},$$

where $0 < r < m$ is the meet-in-the-middle dimension, $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$, and $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$. In Section, 5.1.1, it was shown that for q-ary lattices, where q is prime, one can always construct a basis of this form, provided that the determinant of the lattice is at most q^{m-r}. Additionally, in Section 5.4, we show that our framework can be applied to many lattice-based cryptographic schemes.

The main idea of the attack is the following. Let $\mathbf{v}$ be a shortest non-zero vector contained in the lattice Λ. We split the short vector $\mathbf{v}$ into two parts $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. The second part $\mathbf{v}_g$ represents the part of $\mathbf{v}$ that is recovered by guessing (meet-in-the-middle) during the attack, while the first part $\mathbf{v}_l$ is recovered with lattice techniques (solving BDD problems). Because of the special form of the basis $\mathbf{B}'$, we have that

$$\mathbf{v} = \begin{pmatrix} \mathbf{v}_l \\ \mathbf{v}_g \end{pmatrix} = \mathbf{B}' \begin{pmatrix} \mathbf{x} \\ \mathbf{v}_g \end{pmatrix} = \begin{pmatrix} \mathbf{Bx} + \mathbf{Cv}_g \\ \mathbf{v}_g \end{pmatrix}$$

for some vector $\mathbf{x} \in \mathbb{Z}^{m-r}$, hence $\mathbf{C}\mathbf{v}_g = -\mathbf{B}\mathbf{x} + \mathbf{v}_l$. This means $\mathbf{C}\mathbf{v}_g$ is close to the lattice $\Lambda(\mathbf{B})$, since it only differs from the lattice by the short vector $\mathbf{v}_l$, and therefore $\mathbf{v}_l$ can be recovered solving a BDD problem if $\mathbf{v}_g$ is know. The idea now is that if we can correctly guess the vector $\mathbf{v}_g$, we can hope to find $\mathbf{v}_l$ using the Nearest Plane algorithm (see Chapter 2) via $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$, which is the case if the basis $\mathbf{B}$ is sufficiently reduced. Solving the BDD problem using Nearest Plane is the lattice part of the attack. The lattice $\Lambda(\mathbf{B})$ in which we need to solve BDD has the same determinant as the lattice $\Lambda(\mathbf{B}')$ in which we want to solve uSVP, but it has smaller dimension, i.e., $m - r$ instead of m. Therefore, the newly obtained BDD problem is potentially easier to solve than the original uSVP instance.

In the following, we explain how one can speed up the guessing part of the attack by Odlyzko's meet-in-the-middle approach. Using this technique one is able to reduce the number of necessary guesses to the square root of the number of guesses needed in a naive brute-force approach. Odlyzko's meet-in-the-middle attack on NTRU was first described in [HGSW] and applied in the hybrid attack against NTRU in [HG07]. The idea is that instead of guessing $\mathbf{v}_g$ directly in a large set M of possible vectors, we guess sparser vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in a smaller set N of vectors such that $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$. In our attack the larger set M will be the set of all vectors with a fixed number $2c_i$ of the non-zero entries equal to i for all $i \in \{\pm 1, \dots, \pm k\}$, where $k = \|\mathbf{v}_g\|_\infty$. The smaller set N will be the set of all vectors with only half as many, i.e., only c_i, of the non-zero entries equal to i for all $i \in \{\pm 1, \dots, \pm k\}$. Assume that $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. First, we guess vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in the smaller set N. We then compute $\mathbf{v}_l' = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g')$ and $\mathbf{v}_l'' = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g'')$. We hope that if $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$, then also $\mathbf{v}_l' + \mathbf{v}_l'' = \mathbf{v}_l$, i.e., that Nearest Plane is additively homomorphic on those inputs. The probability that this additive property holds is one crucial element in the runtime analysis of the attack. We further need to detect when this property holds during the attack, i.e., we need to be able to recognize matching vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ with $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$ and $\mathbf{v}_l' + \mathbf{v}_l'' = \mathbf{v}_l$, which we call a collision. In order to do so, we store $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in (hash) boxes whose addresses depend on $\mathbf{v}_l'$ and $\mathbf{v}_l''$, respectively, such that they collide in at least one box. To define those addresses properly, note that in case of a collision we have $\mathbf{v}_l' = -\mathbf{v}_l'' + \mathbf{v}_l$. Thus $\mathbf{v}_l'$ and $-\mathbf{v}_l''$ differ only by a vector of infinity norm $y = \|\mathbf{v}_l\|_\infty$. Therefore, the addresses must be crafted such that for any $\mathbf{x} \in \mathbb{Z}^m$ and $\mathbf{z} \in \mathbb{Z}^m$ with $\|\mathbf{z}\|_\infty \leq y$ it holds that the intersection of the addresses of $\mathbf{x}$ and $\mathbf{x} + \mathbf{z}$ is non-empty, i.e., $\mathcal{A}_{\mathbf{x}}^{(m,y)} \cap \mathcal{A}_{\mathbf{x}+\mathbf{z}}^{(m,y)} \neq \emptyset$. Furthermore, the set of addresses should not be unnecessarily large so the hash tables do not grow too big and unwanted collisions are unlikely to happen. The following definition satisfies these properties.

Definition 5.1. *Let $m, y \in \mathbb{N}$. For a vector $\mathbf{x} \in \mathbb{Z}^m$ the set $\mathcal{A}_{\mathbf{x}}^{(m,y)} \subset \{0,1\}^m$ is defined as*

$$\mathcal{A}_{\mathbf{x}}^{(m,y)} = \left\{ \mathbf{a} \in \{0,1\}^m \;\middle|\; \begin{array}{l} (\mathbf{a})_i = 1 \text{ if } (\mathbf{x})_i > \lceil \frac{y}{2} - 1 \rceil \text{ for } i \in \{1, \dots, m\}, \\ (\mathbf{a})_i = 0 \text{ if } (\mathbf{x})_i < -\lfloor \frac{y}{2} \rfloor \text{ for } i \in \{1, \dots, m\} \end{array} \right\}.$$

Algorithm 5: The hybrid attack on uSVP without lattice reduction

```
Input: $m, r \in \mathbb{N}$ with $r < m$, $y, k \in \mathbb{N}$, $c_{-k}, \dots, c_k \in \mathbb{N}_0$ with $r = \sum_{i=-k}^{k} 2c_i$,
       $\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m \times m}$, where $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$ and
       $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$
1 while true do
2     guess $\mathbf{v}'_g \in \{-k, \dots, k\}^r$ with exactly $c_i$ entries equal to $i$ for all
        $i \in \{-k, \dots, k\}$;
3     calculate $\mathbf{v}'_l = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}'_g) \in \mathbb{Z}^{m-r}$ ;
4     store $\mathbf{v}'_g$ in all the boxes addressed by $\mathcal{A}^{(m-r,y)}_{\mathbf{v}'_l} \cup \mathcal{A}^{(m-r,y)}_{-\mathbf{v}'_l}$;
5     for all $\mathbf{v}''_g \neq \mathbf{v}'_g$ in all the boxes addressed by $\mathcal{A}^{(m-r,y)}_{\mathbf{v}'_l} \cup \mathcal{A}^{(m-r,y)}_{-\mathbf{v}'_l}$ do
6         set $\mathbf{v}_g = \mathbf{v}'_g + \mathbf{v}''_g$ and calculate $\mathbf{v}_l = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) \in \mathbb{Z}^{m-r}$;
7         if $\mathbf{v} = \begin{pmatrix} \mathbf{v}_l \\ \mathbf{v}_g \end{pmatrix} \in \Lambda(\mathbf{B}')$ and $\|\mathbf{v}_l\|_\infty \leq y$ and $\|\mathbf{v}_g\|_\infty \leq k$ then
8             return $\mathbf{v}$;
```

We illustrate Definition 5.1 with some examples.

Example. *Let $m = 5$ be fixed. For varying bounds y and input vectors $\mathbf{x}$ we have*

$$\mathcal{A}^{(5,1)}_{(7,0,-1,1,-5)} = \{(1,0,0,1,0), (1,1,0,1,0)\}$$
$$\mathcal{A}^{(5,2)}_{(8,0,-1,1,-2)} = \{(1,0,0,1,0), (1,1,0,1,0), (1,0,1,1,0), (1,1,1,1,0)\}$$
$$\mathcal{A}^{(5,3)}_{(2,-1,9,1,-2)} = \{(1,0,1,0,0), (1,0,1,1,0), (1,1,1,0,0), (1,1,1,1,0)\}$$
$$\mathcal{A}^{(5,4)}_{(2,-5,0,7,-2)} = \{(1,0,0,1,0), (1,0,0,1,1), (1,0,1,1,0), (1,0,1,1,1)\}$$

The hybrid attack on uSVP without precomputation is presented in Algorithm 5. A list of the attack parameters and the parameters used in the runtime analysis of the attack and their meaning is given in Table 5.1. In order to increase the chance of Algorithm 5 being successful one performs a lattice reduction step as precomputation. Therefore, the complete hybrid attack, presented in Algorithm 6, is in fact a combination of a lattice reduction step and Algorithm 5.

The Hybrid Attack on BDD

The hybrid attack can also be applied to BDD instead of uSVP by rewriting a BDD instance into a uSVP instance via Kannan's embedding, see Section 2.4.4. The embedded uSVP lattice has the same determinant as the BDD lattice and dimension $m+1$ instead of m. However, the additional dimension can be ignored, since the last

Algorithm 6: The hybrid attack on uSVP including lattice reduction

```
Input: $m, r \in \mathbb{N}$ with $r < m$, $\beta, y, k \in \mathbb{N}$, $c_{-k}, \dots, c_{-1}, c_1, \dots, c_k \in \mathbb{N}_0$ with
       $r = \sum_{i=-k}^{k} 2c_i$,
       $\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m \times m}$, where $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$ and
       $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$
1 BKZ-$\beta$ reduce $\mathbf{B}$ to some basis $\tilde{\mathbf{B}}$;
2 run Algorithm 5 on input $m, r, y, k, c_{-k}, \dots, c_{-1}, c_1, \dots, c_k, \left(\begin{array}{c|c} \tilde{\mathbf{B}} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right)$;
```

Parameter	Meaning
m	lattice dimension
r	meet-in-the-middle dimension
β	block size used for lattice reduction
δ	root Hermite factor corresponding to β
$\mathbf{B}'$	lattice basis of the whole lattice
$\mathbf{B}$	partially reduced lattice basis of the sublattice
c_i	number of i-entries guessed during attack
y	infinity norm bound on $\mathbf{v}_l$
k	infinity norm bound on $\mathbf{v}_g$
Y	expected Euclidean norm of $\mathbf{v}_l$
$\lVert\mathbf{b}_i^*\rVert$	Gram-Schmidt lengths corresponding to $\mathbf{B}$
r_i	scaled Gram-Schmidt lengths corresponding to $\mathbf{B}$

Table 5.1: Attack parameters and parameters in the runtime analysis

entry of the short vector $\mathbf{v}$ is known to be the embedding factor and therefore we do not have to guess it during the meet-in-the-middle phase. Note that by definition of BDD it is very likely that $\pm\mathbf{v}$ are the only short vectors in the lattice $\Lambda(\mathbf{B}'')$. By fixing the last coordinate to be the embedding factor, only $\mathbf{v}$ can be found by the attack.

5.3 Analysis

In this section, we analyze the runtime of the hybrid attack. First, in Heuristic 5.1 in Section 5.3.1, we estimate the runtime of the attack in case sufficient success conditions are satisfied. In Section 5.3.2, we then show how to determine the probability that those success conditions are satisfied, i.e., how to determine (a lower bound on) the success probability. We conclude the runtime analysis of the attack by showing how to optimize the attack parameters to minimize its runtime in

Section 5.3.3. We end the section by highlighting our improvements over previous analyses of the hybrid attack, see Section 5.3.3.

5.3.1 Runtime Analysis

We now present our main result about the runtime of the generalized hybrid attack. It shows that under sufficient conditions the attack is successful and estimates the expected runtime in the success case. We provide "over"- and "under"-estimates, where the under-estimates account for possible improvements which have not yet shown to be applicable. Not the they are not intended to be strict upper or lower bounds on the runtime of the attack.

Heuristic 5.1. *Let $m, r \in \mathbb{N}$ with $r < m$, $\beta, y, k \in \mathbb{N}$, $c_{-k}, \dots, c_{-1}, c_1, \dots, c_k \in \mathbb{N}_0$ with $r = \sum_{i=-k}^{k} 2c_i$, and $\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m \times m}$ with $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$ and $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$ be the inputs of Algorithm 5. Further let $Y \in \mathbb{R}_{\geq 0}$ and let $\|\mathbf{b}_1^*\|, \dots, \|\mathbf{b}_{m-r}^*\|$ denote the lengths of the Gram-Schmidt basis vectors of the basis $\mathbf{B}$. Further let $S \subset \Lambda(\mathbf{B}')$ denote the set of all non-zero lattice vectors $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t \in \Lambda(\mathbf{B}')$, where $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$ with $\|\mathbf{v}_l\|_\infty \leq y$, $\|\mathbf{v}_l\| \approx Y$, $\|\mathbf{v}_g\|_\infty \leq k$, exactly $2c_i$ entries of $\mathbf{v}_g$ are equal to i for all $i \in \{\pm 1, \dots \pm k\}$, and $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. Assume that the set S is non-empty.*

Then Algorithm 5 is successful and the expected number of loops can be estimated by

$$L = \binom{r}{c_{-k}, \dots, c_k} \left(p \cdot |S| \cdot \prod_{i \in \{\pm 1, \dots, \pm k\}} \binom{2c_i}{c_i} \right)^{-\frac{1}{2}},$$

where

$$p = \prod_{i=1}^{m-r} \left(1 - \frac{1}{r_i B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_{-r_i-1}^{-r_i} \int_{\max(-1, z-r_i)}^{z+r_i} (1-t^2)^{\frac{(m-r)-3}{2}} dt dz \right),$$

$B(\cdot,\cdot)$ denotes the Euler beta function (see [Olv10]), and

$$r_i = \frac{\|\mathbf{b}_i^*\|}{2Y} \quad \text{for all } i \in \{1, \dots, m-r\}.$$

Furthermore, the expected number of operations of Algorithm 5 for security under- and overestimates can be estimated by

$$T_{\mathsf{hyb,under}} = (m-r)/2^{1.06} L \quad \text{and} \quad T_{\mathsf{hyb,over}} = (m-r)^2/2^{1.06} L.$$

In the following remark we explain the meaning of the (attack) parameters that appear in Heuristic 5.1 in more detail.

Remark 5.1. *1) The main attack parameters of the hybrid attack are the meet-in-the-middle dimension r and the BKZ block size β used in the precomputation phase (see Algorithm 6). While r determines the dimensions of the search space and the BDD lattice, β determines the Gram-Schmidt lengths $\|\mathbf{b}_1^*\|, \dots, \|\mathbf{b}_{m-r}^*\|$ of the BKZ-β reduced basis of the BDD lattice. The Gram-Schmidt lengths achieved by lattice reduction can be estimated by the GSA (see Chapter 2) or its modified version for q-ary lattices presented in Section 5.1.2. Note that spending more time on lattice reduction increases the probability p in Heuristic 5.1 as well as the probability that the condition $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$ holds, as can be seen later in this section and Section 5.3.2.*

2) In order to obtain a high success probability of the attack, the parameters y, k, $c_{-k}, \dots, c_k$ must be chosen in such a way that the requirements of Heuristic 5.1 are likely to be fulfilled. Choosing those parameters depends heavily on the distribution of the short vectors $\mathbf{v} \in S$. In order to obtain more flexibility, this distribution is not specified in Heuristic 5.1. However, in Section 5.4, we show how one can choose the attack parameters and calculate the success probability for several distributions arising in various cryptographic schemes. At this point we only remark that y should be a (tight) upper bound on $\|\mathbf{v}_l\|_\infty$, k a (tight) upper bound on $\|\mathbf{v}_g\|_\infty$, and $2c_i$ the (expected) number of entries of $\mathbf{v}_g$ that is equal to i for $i \in \{\pm 1, \dots, \pm k\}$.

3) As indicated in the first remark, the complete attack (presented in Algorithm 6) is in fact a combination of precomputation (lattice reduction) and Algorithm 5. Therefore, the runtime of both phases must be considered when estimating the total runtime of the attack. Furthermore, to minimize the overall cost (up to a factor of at most 2), the runtimes of both individual phases have to be balanced. In particular, the block size for the BKZ algorithm must be chosen such that the precomputed basis offers the best trade-off between its quality with respect to the hybrid attack (i.e., amplifying the success probability and decreasing the number of operations) and the cost to compute such a basis. In addition, the dimension r must be chosen such that the cost of the meet-in-the-middle phase roughly matches the precomputation cost. More details on optimizing the total runtime are presented in Section 5.3.3.

In the following, we show how Heuristic 5.1 can be derived. For the rest of this section let all notations be as in Heuristic 5.1. We further assume in the following that the assumption of Heuristic 5.1, i.e., $S \neq \emptyset$, is satisfied. We first provide the following useful definition already given in [HG07], however with a slightly different notation.

Definition 5.2. *Let $n \in \mathbb{N}$. A vector $\mathbf{x} \in \mathbb{R}^n$ is called $\mathbf{y}$-admissible (with respect to the basis $\mathbf{B}$) for some vector $\mathbf{y} \in \mathbb{R}^n$ if $\mathrm{NP}_{\mathbf{B}}(\mathbf{x}) = \mathrm{NP}_{\mathbf{B}}(\mathbf{x} - \mathbf{y}) + \mathbf{y}$.*

This means, that if $\mathbf{x}$ is $\mathbf{y}$-admissible then $\mathrm{NP}_{\mathbf{B}}(\mathbf{x})$ and $\mathrm{NP}_{\mathbf{B}}(\mathbf{x}-\mathbf{y})$ yield the same lattice vector. The following lemma about Definition 5.2 showcases the relevance of the definition by relating it to the equation $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1) + \mathrm{NP}_{\mathbf{B}}(\mathbf{t}_2) = \mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1 + \mathbf{t}_2)$, which is necessary to hold for our attack to work.

Lemma 5.2. *Let $\mathbf{t}_1 \in \mathbb{R}^n, \mathbf{t}_2 \in \mathbb{R}^n$ be two arbitrary target vectors. Then the following are equivalent.*

1. $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1) + \mathrm{NP}_{\mathbf{B}}(\mathbf{t}_2) = \mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1 + \mathbf{t}_2)$.
2. $\mathbf{t}_1$ *is* $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*
3. $\mathbf{t}_2$ *is* $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*

Proof. Let $\mathbf{t} = \mathbf{t}_1 + \mathbf{t}_2$ and $\mathbf{y} = \mathrm{NP}(\mathbf{t})$. By symmetry it suffices to show

$$\mathrm{NP}(\mathbf{t}_1) + \mathrm{NP}(\mathbf{t}_2) = \mathbf{y} \quad \Leftrightarrow \quad \mathrm{NP}(\mathbf{t}_1) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) + \mathbf{y},$$

which is equivalent to showing

$$-\mathrm{NP}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y}).$$

By definition, $\mathbf{t} - \mathbf{y}$ is a lattice vector and therefore $\mathrm{NP}(\mathbf{x} - (\mathbf{t} - \mathbf{y})) = \mathrm{NP}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^m$. This leads to

$$\mathrm{NP}(\mathbf{t}_1 - \mathbf{y}) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{y} - (\mathbf{t} - \mathbf{y})) = \mathrm{NP}(\mathbf{t}_1 - \mathbf{t}) = \mathrm{NP}(-\mathbf{t}_2) = -\mathrm{NP}(\mathbf{t}_2).$$

□

Success of the Attack and Number of Loops

We now estimate the expected number of loops in case Algorithm 5 terminates. In the following, we use the subscript $\mathbf{B}$ for probabilities to indicate that the probability is taken over the randomness of the basis (with Gram-Schmidt length $\|\mathbf{b}_1^*\|, \ldots, \|\mathbf{b}_{m-r}^*\|$). In each loop of the algorithm we sample a vector $\mathbf{v}_g'$ in the set

$$W = \{\mathbf{w} \in \mathbb{Z}^r \mid \text{exactly } c_i \text{ entries of } \mathbf{w} \text{ are equal to } i \quad \forall i \in \{-k, \ldots, k\}\}.$$

The attack succeeds if $\mathbf{v}_g' \in W$ and $\mathbf{v}_g'' \in W$ such that $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$ and $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g') + \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g'') = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g' + \mathbf{C}\mathbf{v}_g'') = \mathbf{v}_l$ for some vector $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g) \in S$ are sampled in different loops of the algorithm. By Lemma 5.2 the second condition is equivalent to the fact that $\mathbf{C}\mathbf{v}_g'$ is $\mathbf{v}_l$-admissible. We assume that the algorithm only succeeds in this case. We are therefore interested in the following subset of W:

$$V = \left\{ \mathbf{w} \in W \,\middle|\, \begin{array}{c} \mathbf{v}_g - \mathbf{w} \in W \text{ and } \mathbf{C}\mathbf{w} \text{ is } \mathbf{v}_l\text{-admissible} \\ \text{for some } \mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g) \in S \end{array} \right\}.$$

For all $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g) \in S$, with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$ let $p(\mathbf{v})$ denote the probability

$$p(\mathbf{v}) = \Pr_{\mathbf{B},\mathbf{w} \leftarrow W}[\mathbf{Cw} \text{ is } \mathbf{v}_l\text{-admissible}]$$

and $p_1(\mathbf{v})$ denote the probability

$$p_1(\mathbf{v}) = \Pr_{\mathbf{w} \leftarrow W}[\mathbf{v}_g - \mathbf{w} \in W] = \frac{\prod\limits_{i \in \{\pm 1, \ldots, \pm k\}} \binom{2c_i}{c_i}}{|W|}, \text{ where } |W| = \binom{r}{c_{-k}, \ldots, c_k}.$$

By construction we have that $p_1(\mathbf{v})$ is constant for all $\mathbf{v} \in S$, so we can simply write p_1 instead of $p_1(\mathbf{v})$. It is reasonable to assume that $\mathbf{Cw}$ is randomly distributed modulo the parallelepiped $\mathcal{P}(\mathbf{B}^*)$, or without loss of generality in $\mathcal{P}(\mathbf{B}^*)$, and that $\mathbf{v}_l$ (which is of length Y) is distributed in a random direction relative to basis. We can therefore make the following reasonable assumption on $p(\mathbf{v})$.

Assumption 5.1. *For all $\mathbf{v} \in S$ we assume that*

$$p(\mathbf{v}) \approx p := \Pr_{\mathbf{B},\mathbf{x} \leftarrow \mathcal{P}(\mathbf{B}^*), \mathbf{y} \leftarrow S_{m-r}(Y)}[\mathbf{x} \textit{ is } \mathbf{y}\textit{-admissible}],$$

where

$$S_{m-r}(Y) = \{\mathbf{x} \in \mathbb{R}^{m-r} \mid \|\mathbf{x}\| = Y\}$$

is the surface of a sphere with radius Y centered around the origin.

Assuming independence of p and p_1 and disjoint events for the elements of S, we can make the following reasonable assumption (analogously to Lemma 6 and Theorem 3 of [HG07]).

Assumption 5.2. *We assume that*

$$\frac{|V|}{|W|} \approx \Pr_{\mathbf{B},\mathbf{w} \leftarrow W}[\mathbf{w} \in V] \approx p_1 p \, |S| \, .$$

From Assumption 5.2 it follows that $|V| \approx p_1 p \, |W| \, |S|$. As long as the product $p_1 p$ is not too small, we can therefore assume that $V \neq \emptyset$, which we do in the following. In this case the attack is successful, since by Lemma 5.2 if $\mathbf{v}_g' \in V$ then also $\mathbf{v}_g'' = \mathbf{v}_g - \mathbf{v}_g' \in V$ for all $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g) \in S$. Such two vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in V will eventually be guessed in two separate loops of the algorithm and they are recognized as a collision, since by the assumption $\|\mathbf{v}_l\|_\infty \leq y$ of Heuristic 5.1 they share at least one common address. By Assumption 5.2 we expect that during the algorithm we sample in V every $\frac{1}{p_1 p |S|}$ loops and by the birthday paradox we expect to find a collision $\mathbf{v}'_g \in V$ and $\mathbf{v}''_g \in V$ with $\mathbf{v}''_g + \mathbf{v}'_g = \mathbf{v}_g$ after $L \approx \frac{1}{p_1 p|S|}\sqrt{|V|}$ loops. In conclusion, we can estimate the expected number of loops by

$$L \approx \frac{\sqrt{|V|}}{p_1 p\,|S|} = \frac{\sqrt{|W|}}{\sqrt{p_1 p\,|S|}} = \binom{r}{c_{-k}, \ldots, c_k}\left(p\,|S| \prod_{i \in \{\pm 1, \ldots, \pm k\}} \binom{2c_i}{c_i}\right)^{-\frac{1}{2}}.$$

In order to conclude the estimation for the necessary number of loops, it remains to calculate the probability p, which is done in the following.

Heuristic 5.2. *The probability p is approximately*

$$p \approx \prod_{i=1}^{m-r}\left(1 - \frac{1}{r_i B(\frac{(m-r)-1}{2}, \frac{1}{2})}\int_{-r_i-1}^{-r_i}\int_{\max(-1,z-r_i)}^{z+r_i}(1-t^2)^{\frac{(m-r)-3}{2}}dtdz\right),$$

where $B(\cdot,\cdot)$ and $r_1, \ldots, r_{m-r}$ are defined as in Heuristic 5.1.

In order to calculate p one needs to estimate the lengths r_i, as discussed in the following remark.

Remark 5.2. *Note that the probability p depends on the scaled Gram-Schmidt lengths r_i and therefore on the quality of the basis, i.e., its root Hermite factor δ. For the scaling factor one needs to estimate $\|\mathbf{v}_l\|$. The Gram-Schmidt lengths obtained after performing lattice reduction can be predicted by the GSA (see Chapter 2) or its modified version for q-ary lattices (see Section 5.1.2).*

In the following, we justify Heuristic 5.2. Let $\mathbf{x}$ and $\mathbf{y}$ be as in Assumption 5.2. By Lemma 2.1 there exist unique lattice vectors $\mathbf{u}_1, \mathbf{u}_2 \in \Lambda(\mathbf{B})$ such that $\mathrm{NP}_{\mathbf{B}}(\mathbf{x}) = \mathbf{x} - \mathbf{u}_1 \in \mathcal{P}(\mathbf{B}^*)$ and $\mathrm{NP}_{\mathbf{B}}(\mathbf{x}-\mathbf{y}) + \mathbf{y} = \mathbf{x} - \mathbf{u}_2 \in \mathbf{y} + \mathcal{P}(\mathbf{B}^*)$. As without loss of generality we assume $\mathbf{x} \in \mathcal{P}(\mathbf{B}^*)$, we have $\mathbf{u}_1 = \mathbf{0}$. Then by definition $\mathbf{x}$ is $\mathbf{y}$-admissible if and only if $\mathbf{u}_2 = \mathbf{u}_1 = \mathbf{0}$, which is equivalent to $\mathbf{y} - \mathrm{NP}_{\mathbf{B}}(\mathbf{x}) \in \mathcal{P}(\mathbf{B}^*)$. Therefore, p is equal to the probability

$$p = \Pr_{\mathbf{B}, \mathbf{x} \leftarrow \mathcal{P}(\mathbf{B}^*), \mathbf{y} \leftarrow S_{m-r}(Y)}[\mathbf{y} - \mathrm{NP}_{\mathbf{B}}(\mathbf{x}) \in \mathcal{P}(\mathbf{B}^*)],$$

which we determine in the following.

There exists some orthonormal transformation that aligns $\mathcal{P}(\mathbf{B}^*)$ along the standard axes of $\mathbb{R}^{m-r}$. By applying this transformation, we may therefore assume that $\mathcal{P}(\mathbf{B}^*)$ is aligned along the standard axes of $\mathbb{R}^{m-r}$ (and still that $\mathbf{y}$ is a uniformly random vector of length Y). We can therefore approximate the probability p by

$$p \approx \Pr_{\mathbf{t} \overset{\$}{\leftarrow} \mathcal{R}, \mathbf{y} \overset{\$}{\leftarrow} S_{m-r}(Y)}[\mathbf{t} + \mathbf{y} \in \mathcal{R}], \tag{5.4}$$

where

$$\mathcal{R} = \left\{ \mathbf{x} \in \mathbb{R}^{m-r} \mid \forall i \in \{1, \dots, m-r\} : -\frac{\|\mathbf{b}_i^*\|}{2} \leq x_i < \frac{\|\mathbf{b}_i^*\|}{2} \right\}$$

is the rectangular parallelepiped centered around zero with edge lengths $\|\mathbf{b}_i^*\|$. We continue calculating this approximation of p. We can rewrite (5.4) as

$$p \approx \Pr_{t_i \xleftarrow{\$} \left[-\frac{\|\mathbf{b}_i^*\|}{2}, \frac{\|\mathbf{b}_i^*\|}{2}\right], \mathbf{y} \xleftarrow{\$} S_{m-r}(Y)} \left[\forall i \in \{1, \dots, m-r\} : t_i + (\mathbf{y})_i \in \left[-\frac{\|\mathbf{b}_i^*\|}{2}, \frac{\|\mathbf{b}_i^*\|}{2} \right] \right].$$

Rescaling everything by a factor of $1/Y$ leads to

$$p \approx \Pr_{t_i \xleftarrow{\$} [-r_i, r_i], \mathbf{y} \xleftarrow{\$} S_{m-r}(1)} [\forall i \in \{1, \dots, m-r\} : t_i + (\mathbf{y})_i \in [-r_i, r_i]],$$

where r_i are as defined in Heuristic 5.1.

In theory, the distributions of the coordinates of $\mathbf{y}$ are not independent, which makes calculating p very cumbersome. In practice, however, the probability that $t_i + (\mathbf{y})_i \in [-r_i, r_i]$ is big for all but the last few indices i. This is due to the fact that according to the GSA typically only the last values r_i are small. Consequently, we expect the dependence of the remaining entries not to be strong. This assumption was already established by Howgrave-Graham [HG07] and appears to hold for typical values of r_i appearing in practice. It is therefore reasonable to assume that

$$p \approx \prod_{i=1}^{m-r} \Pr_{t_i \xleftarrow{\$} [-r_i, r_i], (\mathbf{y})_i \xleftarrow{\$} P_{m-r}} [t_i + (\mathbf{y})_i \in [-r_i, r_i]],$$

where P_{m-r} denotes the probability distribution on the interval $[-1, 1]$ obtained by the following experiment: sample a vector $\mathbf{y}$ uniformly at random on the unit sphere in $\mathbb{R}^{(m-r)}$ and then output the first (equivalently, any arbitrary but fixed) coordinate of $\mathbf{y}$.

Next we explore the density function of P_{m-r}. The probability that $(\mathbf{y})_i \leq x$ for some $-1 < x < 0$, where $(\mathbf{y})_i \xleftarrow{\$} P_{m-r}$, is given by the ratio of the surface area of a hyperspherical cap of the unit sphere in $\mathbb{R}^{(m-r)}$ with height $h = 1 + x$ and the surface area of the unit sphere. This is illustrated in Figure 5.1 for $m - r = 2$. The surface area of a hyperspherical cap of the unit sphere in $\mathbb{R}^{m-r}$ with height $h < 1$ is given by (see [Li11])

$$A_{m-r}(h) = \frac{1}{2} A_{m-r} I_{2h-h^2} \left(\frac{(m-r)-1}{2}, \frac{1}{2} \right),$$

where $A_{m-r} = 2\pi^{(m-r)/2} / \Gamma((m-r)/2)$ is the surface area of the unit sphere and

$$I_x(a,b) = \frac{\int_0^x t^{a-1}(1-t)^{b-1} dt}{B(a,b)}$$

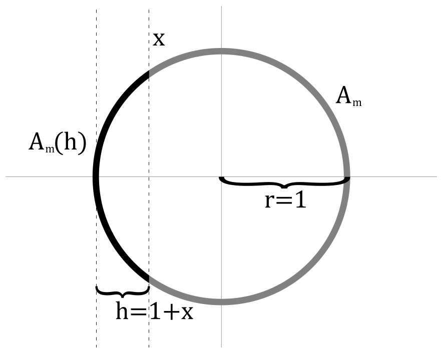

Figure 5.1: Two-dimensional hyperspherical cap

is the regularized incomplete beta function (see [Olv10]) and $B(a, b)$ is the Euler beta function.

Consequently, for $-1 < x < 0$, we have

$$\begin{aligned}
&\Pr_{(\mathbf{y})_i \xleftarrow{\$} P_{m-r}} [(\mathbf{y})_i \leq x] = \frac{A_{m-r}(1+x)}{A_{m-r}} \\
&= \frac{1}{2} I_{2(1+x)-(1+x)^2} \left(\frac{(m-r)-1}{2}, \frac{1}{2} \right) \\
&= \frac{1}{2} I_{1-x^2} \left(\frac{(m-r)-1}{2}, \frac{1}{2} \right) \\
&= \frac{1}{2B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_0^{1-x^2} t^{\frac{(m-r)-3}{2}} (1-t)^{-1/2} dt \\
&= \frac{1}{2B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_{-1}^{x} (1-t^2)^{\frac{(m-r)-3}{2}} (1-(1-t^2))^{-1/2} (-2t) dt \\
&= -\frac{1}{B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_{-1}^{x} (1-t^2)^{\frac{(m-r)-3}{2}} |t|^{-1} |t| \, dt \\
&= \frac{1}{B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_{-1}^{x} (1-t^2)^{\frac{(m-r)-3}{2}} dt. \qquad (5.5)
\end{aligned}$$

Together with

$$\Pr_{t_i \xleftarrow{\$} [-r_i, r_i]} [t_i \leq x] = \int_{-r_i}^{x} \frac{1}{2r_i} dt,$$

we can use a convolution to obtain

$$\Pr_{t_i \xleftarrow{\$} [-r_i, r_i], (\mathbf{y})_i \xleftarrow{\$} P_{m-r}} [t_i + (\mathbf{y})_i \leq x] = \frac{1}{2r_i B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_{-r_i-1}^{x} \int_{\max(-1, z-r_i)}^{\min(1, z+r_i)} (1-t^2)^{\frac{(m-r)-3}{2}} dt dz.$$

Using the fact that

$$\Pr_{t_i \xleftarrow{\$} [-r_i, r_i], (\mathbf{y})_i \xleftarrow{\$} P_{m-r}} [t_i + (\mathbf{y})_i \in [-r_i, r_i]] = 1 - 2 \left(\Pr_{t_i \xleftarrow{\$} [-r_i, r_i], (\mathbf{y})_i \xleftarrow{\$} P_{m-r}} [t_i + (\mathbf{y})_i < -r_i] \right),$$

concludes our calculation of the probability p. All integrals can be calculated for instance using SageMath [S+17].

Number of Operations

We now estimate the expected total number of operations of the hybrid attack under the conditions of Heuristic 5.1. In order to do so we need to estimate the runtime of one inner loop and multiply it by the expected number of loops. As in [HG07] we make the following assumption, which is plausible as long the sets of addresses are not extremely large.

Assumption 5.3. *We assume that the number of operations of one inner loop of Algorithm 5 is dominated by the number of operations of one Nearest Plane call.*

Note that Assumption 5.3 does not hold for all parameter choices[6], but it is reasonable to believe that it holds for many relevant parameter sets, as claimed in [HG07]. However, the claim in [HG07] is based on the observation that for random vectors in $\mathbb{Z}_q^m$ it is highly unlikely that adding a binary vector will flip the sign of many coordinates (i.e., that a random vector in $\mathbb{Z}_q^m$ has many minus one coordinates). While this is true, the vectors in question are in fact not random vectors in $\mathbb{Z}_q^m$ but outputs of a Nearest Plane call, and thus potentially shorter than typical vectors in $\mathbb{Z}_q^m$. Therefore it can be expected that adding a binary vector will flip more signs. Additionally, in general it is not only a binary vector that is added, but a vector of infinity norm at most y, which makes flipping signs more likely. However, it is reasonable to believe that Assumption 5.3 is still plausible for most relevant parameter sets and small y, and in the worst case the assumption leads to more conservative security estimates.

In [HHHGW09], Hirschhorn et al. give an experimentally verified number of bit operations (defined as in [LV01]) of one Nearest Plane call and state a conjecture on the runtime of Nearest Plane using precomputation. Based on their results, we make the following assumption for our security estimates (over and under).

Assumption 5.4. *Let $d \in \mathbb{N}$ be the lattice dimension. For our security overestimates, we assume that the number of bit operations of one Nearest Plane call is approximately $d^2/2^{1.06}$. For our security underestimates, we assume that the number of bit operations of one Nearest Plane call is approximately $d/2^{1.06}$.*

[6]For instance, if the infinity norm y is too big, it is likely to have exponentially many addresses per vector and storing a vector at all addresses takes more time than a Nearest Plane call.

In conclusion, under the conditions of Heuristic 5.1 the expected number of operations of Algorithm 5 for security under- and overestimates is approximately

$$T_{\mathsf{hyb,under}} = (m-r)/2^{1.06}L \quad \text{and} \quad T_{\mathsf{hyb,over}} = (m-r)^2/2^{1.06}L.$$

5.3.2 Determining the Success Probability.

In Heuristic 5.1 it is guaranteed that Algorithm 5 is successful if the lattice Λ contains a non-empty set S of short vectors of the form $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)$, where $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$, with $\|\mathbf{v}_l\| \approx Y$, $\|\mathbf{v}_l\|_\infty \leq y$, $\|\mathbf{v}_g\|_\infty \leq k$, exactly $2c_i$ entries of $\mathbf{v}_g$ are equal to i for all $i \in \{\pm 1, \dots \pm k\}$, and $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. In order to determine a lower bound on the success probability, one must calculate the probability that the set S of such vectors is non-empty, since

$$p_{\mathsf{succ}} \geq \Pr[S \neq \emptyset].$$

However, this probability depends heavily on the distribution of the short vectors contained in Λ and is therefore not done in Heuristic 5.1, allowing for more flexibility. In consequence, this analysis must be performed for the specific distribution at hand originating from the cryptographic scheme that is to be analyzed. The most involved part in calculating the success probability is typically calculating the probability p_{NP} that $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. From Equation 5.5, we can deduce that the probability p_{NP} is approximately

$$p_{\mathrm{NP}} \approx \prod_{i=1}^{m-r} \left(1 - \frac{2}{B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_{-1}^{\max(-r_i,-1)} (1-t^2)^{\frac{(m-r)-3}{2}} dt \right), \tag{5.6}$$

where r_i are defined as in Heuristic 5.1 and obtained as in Remark 5.2.

In [LP11], Lindner and Peikert calculated the success probability of the Nearest Plane(s) algorithm for the case that the difference vector is drawn from a discrete Gaussian distribution with standard deviation σ (as typical for, e.g., an LWE error distribution). In our case, this would result in the formula

$$p_{\mathrm{NP}} = \Pr\left[\mathrm{NP}_{\mathbf{B}}\left(\mathbf{C}\mathbf{v}_g\right) \approx \mathbf{v}_\ell\right] = \prod_{i=1}^{m-r} \mathrm{erf}\left(\frac{\|\mathbf{b}_i^*\| \sqrt{2}}{\sigma} \right). \tag{5.7}$$

In the following, we compare our formula (5.6) to (5.7) in the case of discrete Gaussian distributions with standard deviation σ. To this end, we evaluated both formulas for a lattice of dimension $d = m - r = 200$ of determinant 128^{100} for different standard deviations. For our formulas, we assumed that the norm of $\mathbf{v}_l$ is $\sigma\sqrt{200}$ as expected and that the basis follows the GSA with root Hermite factor 1.008. Our results, presented in Table 5.2, show that both formulas virtually give the same results for the analyzed instances. This indicates that our formula is a good generalization of the one provided in [LP11].

Gaussian parameter	$s = 1$	$s = 2$	$s = 4$	$s = 8$	$s = 16$
p_{NP} according to (5.6)	$2^{-0.033}$	$2^{-3.658}$	$2^{-27.775}$	$2^{-87.506}$	$2^{-188.445}$
p_{NP} according to (5.7)	$2^{-0.036}$	$2^{-3.669}$	$2^{-27.680}$	$2^{-87.217}$	$2^{-187.932}$

Table 5.2: Comparison of (5.6) and (5.7) for standard deviation $\sigma = s/\sqrt{2\pi}$ and varying Gaussian parameter s.

5.3.3 Optimizing the Runtime

The final step in our analysis is to determine the runtime of the complete hybrid attack (Algorithm 6) including precomputation, which involves the runtime of lattice reduction T_{red}, the runtime of the actual attack T_{hyb}, and the success probability p_{succ}. All these quantities depend on the attack parameter r and the quality of the basis $\mathbf{B}$, i.e., its root Hermite factor δ corresponding to the applied block size β (cf. Chapter 2). In order to unfold the full potential of the attack, one must minimize the runtime over all possible attack parameters r and β (or the corresponding δ instead of β). For our security overestimates, we assume that the total runtime (which is to be minimized) is given by

$$T_{\mathsf{total,over}}(\beta, r) = \frac{T_{\mathsf{red,over}}(\beta, r) + T_{\mathsf{hyb,over}}(\beta, r)}{p_{\mathsf{succ}}(\beta, r)}.$$

For our security underestimates, we conservatively assume that given a reduced basis with quality δ it is significantly easier (i.e., requires a smaller block size) to find another reduced basis with same quality δ (e.g., by randomizing and reducing an already reduced basis) than it is to find one given an arbitrary non-reduced basis. A similar assumption, however resulting in a basis with (slightly) worse quality $\delta' > \delta$ is made in [Alb17]. In the spirit of providing underestimates, however, we assume that $\delta' = \delta$. We therefore assume that even if the attack is not successful and needs to be run again, the large precomputation cost for lattice reduction only needs to be paid once, and hence

$$T_{\mathsf{total,under}}(\beta, r) = T_{\mathsf{red,under}}(\beta, r) + \frac{T_{\mathsf{hyb,under}}(\beta, r)}{p_{\mathsf{succ}}(\beta, r)}.$$

In order to calculate $T_{\mathsf{total,under}}(\beta, r)$ and $T_{\mathsf{total,over}}(\beta, r)$ one has to determine $T_{\mathsf{hyb,under}}(\beta, r)$, $T_{\mathsf{hyb,over}}(\beta, r)$, $T_{\mathsf{red,under}}(\beta, r)$, $T_{\mathsf{red,over}}(\beta, r)$, and $p_{\mathsf{succ}}(\beta, r)$. How to calculate $T_{\mathsf{hyb,under}}(\beta, r)$ and $T_{\mathsf{hyb,over}}(\beta, r)$ is shown in Heuristic 5.1. The success probability $p_{\mathsf{succ}}(\beta, r)$ is calculated in Section 5.3.2. Different approaches how to estimate the cost for BKZ-β depending on the assumed cost of the SVP oracle and the number of tours are discussed in Chapter 2. Since there is not yet a consensus in the cryptographic community as to which estimate to choose, our framework for analyzing the hybrid attack is designed such that the cost model for lattice reduction can be replaced by a different one while the rest of the analysis remains intact. Thus, if future research shows significant improvements in estimating the cost of lattice reduction, these cost models can be applied in our framework. For our security estimates in Section 5.4 we use the enumeration-based cost estimate for the SVP oracle in block size β provided in [APS15]

$$T_{\mathsf{SVP}}(\beta) = 2^{0.187281\beta\log_2(\beta)-1.0192\beta+16.1}.$$

For our security overestimates we use the BKZ 2.0 simulator[7] of [Che13, CN11] to determine the corresponding necessary number of rounds k and set

$$T_{\mathsf{red,over}}(\beta, r) = (m-r)k \cdot T_{\mathsf{SVP}}(\beta).$$

For our security underestimates we assume that only one tour with block size β is needed (e.g., by reducing the basis with smaller block sizes first, see [Che13, AWHT16]), ignore the cost of SVP calls in smaller dimensions than β, and use

$$T_{\mathsf{red,under}}(\beta, r) = (m-r-\beta+1) \cdot T_{\mathsf{SVP}}(\beta).$$

Runtime optimization. The optimization of the total runtime $T_{\mathsf{total}}(\beta, r)$ is performed in the following way. For each possible r we find the optimal β_r that minimizes the runtime $T_{\mathsf{total}}(\beta, r)$. Consequently, the optimal runtime is given by $\min\{T_{\mathsf{total}}(\beta_r, r)\}$, the smallest of those minimized runtimes. Note that for fixed r the optimal β_r for our security underestimates can easily be found in the following way. For fixed r the function $T_{\mathsf{red,under}}(\beta, r)$ is monotonically increasing in β and the function $\frac{T_{\mathsf{hyb,under}}(\beta,r)}{p_{\mathsf{succ}}(\beta,r)}$ is monotonically decreasing in β. Therefore $T_{\mathsf{total,under}}(\beta, r)$ is (close to) optimal (up to a factor of at most 2) when both those functions are balanced, i.e., take the same value. Thus the optimal β_r can for example be found by a simple binary search.

For our security overestimates, we assume the function $\frac{T_{\mathsf{red,over}}(\beta,r)}{p_{\mathsf{succ}}(\beta,r)}$ is monotonically increasing in β in the relevant range. Hence the (near) optimal $T_{\mathsf{total,over}}(\beta, r)$ can be found by balancing the functions $\frac{T_{\mathsf{red,over}}(\beta,r)}{p_{\mathsf{succ}}(\beta,r)}$ and $\frac{T_{\mathsf{hyb,over}}(\beta,r)}{p_{\mathsf{succ}}(\beta,r)}$ as above. Note that this assumption may note be true, but it surely leads to upper bounds on the optimal runtime of the attack.

Improvements Compared to Previous Analyses of the Hybrid Attack

We end this section by highlighting our two main improvements of the analysis of the hybrid attack and compare them to previous approaches which suffer from

[7]For our implementations we used the publicly available code from `https://github.com/NTRUOpenSourceProject/ntru-params`.

inaccuracies. We remark that some of those inaccuracies of previous analyses lead to overestimating the security of the schemes and others to underestimating it. In some analyses, both types occurred at the same time and somewhat magically almost canceled out each others effect on the security estimates for some parameter sets. Even though the security estimates in those cases are not necessarily wrong, they can not be relied upon, since without further analysis it is not clear if the security estimates are correct, over-, or underestimates. We straighten out this unsatisfying state of affairs by providing updated security estimates for various cryptographic schemes using our improved analysis of the hybrid attack, see Section 5.4.

Calculating the probability p One of the most frequently encountered problems that appeared in several works is the lack of a (correct) calculation of the probability p defined in Assumption 5.1. As can be seen in Heuristic 5.1, this probability plays a crucial role in the runtime analysis of the attack. Nevertheless, in several works [HHGP+07, DDLL13, HPS+17, Sch15, BCLvV17b, BCLvV16] the authors ignore the presence of this probability by setting $p = 1$ for the sake of simplicity. However, when analyzing the security of several lattice-based schemes in Section 5.4, even for the optimized attack parameters the probability p was sometimes as low as 2^{-80}, see Table 5.4. Note that the incorrect assumption $p = 1$ gives more power to the attacker, since it assumes that collisions can always be detected by the attacker although this is not the case, resulting in security underestimates. We also remark that in some works the probability p is not completely ignored but determined in a purely experimental way [HG07] or calculated using additional unnecessary assumptions [HHHGW09], introducing inaccuracies into the analysis. In our analysis, we explicitly calculate p under some reasonable assumptions.

Considering the success probability of Nearest Plane In most works [HG07, HHGP+07, HHHGW09, DDLL13, HPS+17, Sch15, BCLvV17b, BCLvV16], the authors demand a sufficiently good lattice reduction such that the Nearest Plane algorithm is guaranteed to unveil the searched short vector (or at least with very high probability). To be more precise, Lemma 1 of [HG07] is used to determine what sufficiently good exactly means. In our opinion, this demand is unrealistic, and instead we account for the probability of this event in the success probability, which reflects the attacker's power in a more accurate way. In particular we note that in most cases Lemma 1 of [HG07] is not applicable the way it is claimed in several works. We briefly sketch way this is the case. Often, Lemma 1 of [HG07] is applied to determine the necessary quality of a reduced basis such that Nearest Plane (on correct input) unveils a vector $\mathbf{v}$ of infinity norm at most y. However, this lemma is only applicable if the basis matrix is in triangular form, which is not the case is general. Therefore, one needs to transform the basis with an orthonormal matrix $\mathbf{Y}$ in order to obtain a triangular basis. This basis, however, does not span the same lattice but one that contains the transformed vector $\mathbf{vY}$, but (in general) not the vector $\mathbf{v}$. While the transformation $\mathbf{Y}$ preserves the Euclidean norm of the vector $\mathbf{v}$, it does not preserve its infinity norm. Therefore, the lemma can not be applied in a straight-forward manner with the same infinity norm bound y, which is done in most works. In fact, in the worst case the new infinity norm bound can be up to $\sqrt{d}y$, where d is the lattice dimension. In consequence one would have to apply Lemma 1 of [HG07] with infinity norm bound $\sqrt{d}y$ instead of y in order to get a rigorous statement, which demands a much better lattice reduction. This problem is already mentioned – but not solve – in [Sch15]. Note that the worst case, where (i) the vector $\mathbf{v}$ has Euclidean norm $\sqrt{d}y$ and (ii) all the weight of the transformed vector is on one coordinate such that $\sqrt{d}y$ is a tight bound on the infinity norm after transformation, is highly unlikely. Nevertheless, simply applying Lemma 1 of [HG07] with infinity norm bound y is overly conservative and no longer necessary in our analysis. In the following, we give an example to illustrate the different success conditions for the Nearest Plane algorithm.

Example. *Let $d = 512$ and $q = 1024$. We consider Nearest Plane on a BDD instance $\mathbf{t} \in \Lambda + \mathbf{e}$ in a d-dimensional lattice Λ of determinant $q^{d/2}$, where $\mathbf{e}$ is a random binary vector. Naivly applying Lemma 1 of [HG07] with infinity norm bound 1 would suggest that lattice reduction of quality $\delta_1 \approx 1.0068$ is sufficient to recover $\mathbf{e}$. Applying the cost model used for our security underestimates described in Section 5.3.3, lattice reduction of that quality would cost roughly $T_1 \approx 2^{91}$ operations. However, as described above, the lemma can not be applied with that naive bound. Instead, using the worst case bound $\sqrt{d}y$ on the infinity norm and applying Lemma 1 of [HG07] would lead to lattice reduction of quality $\delta_2 \approx 1.0007$, taking roughly $T_2 \approx 2^{357}$ operations, to guarantee the success of Nearest Plane. This shows the impracticality of this approach. Using our approach instead, assuming that that the Euclidean norm of a random binary vector is roughly $\|\mathbf{e}\| \approx \sqrt{d/2}$, one can balance the quality of lattice reduction and the success probability of Nearest Plane to obtain the optimal trade-off $\delta_3 \approx 1.0067$, taking roughly $T_3 \approx 2^{94}$ operations, with a success probability of roughly 2^{-31}.*

5.4 Security Estimates Against the Hybrid Attack

In the recent years, the hybrid attack has been applied to various lattice-based cryptographic schemes in order to estimate their security. However, the claimed security levels are unreliable due to simplifications in their analysis of the hybrid attack. Therefore, in this section, we apply our improved analysis of the hybrid attack provided in Section 5.3 to several schemes in order to reevaluate their security.

This section is structures as follows. Each scheme is analyzed in a separate subsection. We begin with the encryption schemes NTRU, NTRU prime and R-BinLWEEnc and end with the signature schemes BLISS and GLP. In each subsection, we first give a brief introduction to the scheme and summarize the inaccuracies in its previous security analysis against the hybrid attack. We then apply the hybrid attack to the scheme and analyze its cost according to Section 5.3. This analysis is performed the following four steps steps.

1) **Constructing the lattice.** We first construct a lattice of the required form which contains the secret key as a short vector.

2) **Determining the attack parameters.** We find suitable attack parameters c_i (depending on the meet-in-the-middle dimension r), infinity norm bounds y and k, and estimate the Euclidean norm Y.

3) **Determining the success probability.** We determine the success probability of the attack according to Section 5.3.2.

4) **Optimizing the runtime.** We optimize the runtime of the attack for our security under- and overestimate according to Section 5.3.3.

We end each subsection by providing a table of updated security estimates against the hybrid attack obtained by our analysis. In the tables we also provide the optimal attack parameters r, δ_r, β_r derived by our optimization process and the corresponding probability p with whom collisions can be detected. For comparison, we further provide the security estimates of the previous works. To showcase the improvement of the hybrid attack over solving uSVP with small or sparse secrets using lattice reduction only, we also provide security estimates that can be derived from the 2016 estimate (cf. Chapter 3). In our runtime optimization of the attack we optimized with a precision of up to one bit. As a result there may not be one unique optimal set of attack parameters r, δ_r, β_r and for the table we arbitrarily pick one of them.

5.4.1 NTRU

The NTRU encryption scheme was officially introduced in [HPS98] and is one of the best known lattice-based encryption schemes today due to its high efficiency. The hybrid attack was first developed to attack NTRU [HG07] and has been applied to various proposed parameter sets since [HG07, HHGP$^+$07, HHHGW09, HPS$^+$17, Sch15]. In this section, we restrict our studies to the recent NTRU parameters presented in [HPS$^+$17]. As the analysis in [HPS$^+$17] makes simplifying assumptions such as setting the probability p equal to one or simplifying the structure of the private keys, we conclude that these security estimates are not reliable. We therefore reevaluate the security of the NTRU EESS # 1 parameter sets given in Table 3 of [HPS$^+$17].

Constructing the Lattice

The NTRU cryptosystem is defined over the ring $R_q = \mathbb{Z}_q[X]/(X^N - 1)$, where $N, q \in \mathbb{N}$ and N is prime. The parameters N and q are public. Furthermore there exist public parameters $d_1, d_2, d_3, d_g \in \mathbb{Z}$. For the parameter sets considered in [HPS+17], the private key is a pair of polynomials $(f, g) \in R_q^2$. The polynomial g has coefficients in $\{-1, 0, 1\}$ with exactly $d_g + 1$ ones and d_g minus ones. The polynomial $f = 1+3F$ is invertible in R_q, where $F = A_1A_2 + A_3$ for some polynomials A_i with coefficients in $\{-1, 0, 1\}$ of which exactly d_i are equal to one and d_i equal to minus one. The corresponding public key is $(1, h)$, where $h = f^{-1}g$. In the following we assume that h and 3 are invertible in R_q. We further identify polynomials with their coefficient vectors. We can recover the private key by finding the secret vector $\mathbf{v} = (\mathbf{F}, \mathbf{g})$.[8] Since $h = (1 + 3F)^{-1}g$ we have $3^{-1}h^{-1}g = F + 3^{-1}$ and therefore it holds that

$$\mathbf{v} + \begin{pmatrix} \mathbf{3}^{-1} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 3^{-1}\mathbf{h}^{-1}\mathbf{g} + q\mathbf{w} \\ \mathbf{g} \end{pmatrix} = \left(\begin{array}{c|c} q\mathbf{I}_n & 3^{-1}\overline{\mathbf{H}} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right) \begin{pmatrix} \mathbf{w} \\ \mathbf{g} \end{pmatrix}$$

for some $\mathbf{w} \in \mathbb{Z}^n$, where $\overline{\mathbf{H}}$ is the rotation matrix of $\mathbf{h}^{-1}$. Hence $\mathbf{v}$ can be recovered by solving BDD on input $(-\mathbf{3}^{-1}, \mathbf{0})$ in the q-ary lattice

$$\Lambda = \Lambda\left(\left(\begin{array}{c|c} q\mathbf{I}_n & 3^{-1}\overline{\mathbf{H}} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right)\right),$$

since $(-\mathbf{3}^{-1}, \mathbf{0}) - \mathbf{v} \in \Lambda$.[9] A similar way to recover the private key was already mentioned in [Sch15]. The lattice Λ has dimension $2n$ and determinant q^n. Since we take the BDD approach for the hybrid attack, we assume that only $\mathbf{v}$, not its rotations or additive inverse, can be found by the attack, see Section 5.2. Hence we assume that the set S, as defined in Heuristic 5.1, contains of at most one element.

Determining the Attack Parameters

Let $\mathbf{v} = (\mathbf{F}, \mathbf{g}) = (\mathbf{v}_l, \mathbf{v}_g)$ with $\mathbf{v}_l \in \mathbb{Z}^{2n-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Since $\mathbf{g}$ is a ternary vector, we can set the infinity norm bound k on $\mathbf{v}_g$ equal to one. In contrast, determining an infinity norm bound on the vector $\mathbf{v}_l$ is not that trivial, since $\mathbf{F}$ is not ternary but of product form. For a specific parameter set this can either be done theoretically or experimentally. The same holds for estimating the Euclidean norm of $\mathbf{v}_l$. For our

[8] Note that we put $\mathbf{g}$ in the half of the vector $\mathbf{v}$ that is guessed in the meet-in-the-middle part of the attack. The reason for this choice is that we exactly know the structure of $\mathbf{g}$ but not the structure of the product form polynomial $\mathbf{F}$.

[9] It is also possible to construct a lattice that contains $(\mathbf{f}, \mathbf{g})$ as a short vector instead. However, since $f = 1 + 3F$ has norm larger than F, this leads to a less efficient attack.

runtime estimates we determined the expected Euclidean norm of $\mathbf{F}$ experimentally and set the expected Euclidean norm of $\mathbf{v}_l$ to

$$\|\mathbf{v}_l\| \approx \sqrt{\|\mathbf{F}\|^2 + \frac{n-r}{n} \cdot (2d_g + 1)}.$$

We set $2c_{-1} = \frac{r}{n} \cdot (d_g + 1)$ and $2c_1 = \frac{r}{n} \cdot d_g$ to be equal to the expected number of minus one entries and one entries, respectively, in $\mathbf{g}$.[10] For simplicity we assume that c_{-1} and c_1 are integers in the following in order to avoid writing down the rounding operates.

Determining the Success Probability

The next step is to determine the success probability p_{succ}, i.e., the probability that $\mathbf{v}$ has exactly $2c_{-1}$ entries equal to minus one, $2c_1$ entries equal to one, and $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$ holds, where $\mathbf{B}$ is as given in Heuristic 5.1. Assuming independence, the success probability is approximately

$$p_{\mathsf{succ}} = p_c \cdot p_{\mathrm{NP}},$$

where p_c is the probability that $\mathbf{v}$ has exactly $2c_{-1}$ entries equal to minus one and $2c_1$ entries equal to one and p_{NP} is defined and calculated as in Section 5.3.2. The probability p_c is given by

$$p_c = \frac{\binom{r}{2\tilde{c_0}, 2c_{-1}, 2c_1} \binom{n-r}{d_0 - 2\tilde{c_0}, d_g - 2c_{-1}, d_g + 1 - 2c_1}}{\binom{n}{d_0, d_g, d_g + 1}},$$

where $2\tilde{c_0} = r - 2c_{-1} - 2c_1$ and $d_0 = n - (d_g + 1) - d_g$. As explained earlier, since we use the BDD approach of the hybrid attack, we assume that $|S| = 1$ in case the attack is successful.

Optimizing the Runtime

We determined the optimal attack parameters to estimate the minimal runtime of the hybrid attack for the NTRU EESS # 1 parameter sets given in Table 3 of [HPS$^+$17]. The results, including the optimal r, corresponding δ_r and β_r, and resulting probability p that collisions can be found, are presented in Table 5.3. Our analysis shows that the security levels against the hybrid attack claimed in [HPS$^+$17] are lower than the actual security levels for all parameter sets. In addition, our results

[10] Note that this must not necessarily be the optimal choice for the c_i. However, we expect that this choice comes very close to the optimal one and therefore restrict our studies to this case.

show that while for all of the analyzed parameter sets the hybrid attack outperforms a pure lattice reduction attack (cf. Chapter 3), it does not perform better than a purely combinatorial meet-in-the-middle search, see Table 3 of [HPS+17]. Our results therefore do not support the common claim that the hybrid attack is necessarily the best attack on NTRU.

Parameter set	**n = 401**	**n = 439**	**n = 593**	**n = 743**
Optimal $r_{\mathsf{under}}/r_{\mathsf{over}}$	104/122	122/140	206/219	290/308
Optimal $\delta_{\mathsf{r,under}}$	1.00544	1.00509	1.00412	1.00352
Optimal $\delta_{\mathsf{r,over}}$	1.00552	1.00518	1.00420	1.00357
Optimal $\beta_{\mathsf{r,under}}$	252	279	381	477
Optimal $\beta_{\mathsf{r,over}}$	246	271	371	468
Corresp. p under/over	$2^{-70}/2^{-43}$	$2^{-56}/2^{-47}$	$2^{-67}/2^{-62}$	$2^{-78}/2^{-69}$
Security under/over in bits	**145/162**	**165/182**	**249/267**	**335/354**
In [HPS+17]	116	133	201	272
r, β used in [HPS+17]	$154, 197$	$174, 221$	$261, 316$	$350, 407$
2016 est. under/over	168/175	196/202	322/328	459/466
β_{2016}	283	318	463	608

Table 5.3: Optimal attack parameters and security levels against the hybrid attack and the primal attack under the 2016 estimate for the NTRU EESS # 1 parameter sets.

5.4.2 NTRU prime

The NTRU prime encryption scheme was recently introduced [BCLvV17b, BCLvV16] in order to eliminate worrisome algebraic structures that exist within NTRU [HPS98] or Ring-LWE based encryption schemes such as [LPR10, ADPS16]. The authors considered the application of the hybrid attack to their scheme to derive their security estimates. However, their analysis follows the methodology of [HPS+17] and therefore makes the same simplifying assumptions, leading to unreliable estimates, see Section 5.4.1. We therefore reevaluate the security of NTRU prime.

Constructing the Lattice

The Streamlined NTRU prime family of cryptosystems is parameterized by three integers $(n, q, t) \in \mathbb{N}^3$, where n and q are odd primes. The base ring for Streamlined NTRU prime is $R_q = \mathbb{Z}_q[X]/(X^n - X - 1)$. The private key is (essentially) a pair of polynomials $(g, f) \in R_q^2$, where g is drawn uniformly at random from the set of all ternary polynomials and f is drawn uniformly at random from the set of all ternary polynomials with exactly $2t$ non-zero coefficients. The corresponding public key is $h = g(3f)^{-1} \in R_q$. In the following we identify polynomials with their coefficient vectors. As described in [BCLvV17b, BCLvV16], the secret vector $\mathbf{v} = (\mathbf{g}, \mathbf{f})$ is contained in the q-ary lattice

$$\Lambda = \Lambda\left(\left(\begin{array}{c|c} q\mathbf{I}_n & 3\mathbf{H} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right)\right),$$

where $\mathbf{H}$ is the rotation matrix of h, since

$$\left(\begin{array}{c|c} q\mathbf{I}_n & 3\mathbf{H} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right)\begin{pmatrix}\mathbf{w}\\ \mathbf{f}\end{pmatrix} = \begin{pmatrix} q\mathbf{w} + 3\mathbf{hf} \\ \mathbf{f}\end{pmatrix} = \begin{pmatrix}\mathbf{g}\\ \mathbf{f}\end{pmatrix} = \mathbf{v}$$

for some $\mathbf{w} \in \mathbb{Z}^n$. The determinant of the lattice Λ is given by q^n and its dimension is equal to $2n$. Note that in the case of Streamlined NTRU prime the rotations of a ternary polynomial are not necessarily ternary due to the structure of the ring, but it is likely the some of them are. The authors of [BCLvV17b, BCLvV16] conservatively assume that the maximum number of good rotations of $\mathbf{v}$ that can be utilized by the attack is $n - t$, which we also assume in the following. Counting their additive inverses leaves us $2(n - t)$ short vectors that can be found by the attack.

Determining the Attack Parameters

Let $\mathbf{v} = (\mathbf{f}, \mathbf{g}) = (\mathbf{v}_l, \mathbf{v}_g)$ with $\mathbf{v}_l \in \mathbb{Z}^{2n-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Since $\mathbf{v}$ is ternary, we can set the infinity norm bounds y and k equal to one. The expected Euclidean norm of $\mathbf{v}_l$ is given by

$$\|\mathbf{v}_l\| \approx \sqrt{\frac{2}{3}n + \frac{n-r}{n}2t}.$$

We set $2c_1 = 2c_{-1} = \frac{r}{n} \cdot \frac{t}{2}$ equal to the expected number of one entries (or minus one entries, respectively) in $\mathbf{f}$. For simplicity, in the following we assume that c_1 is an integer.

Determining the Success Probability

Next, we determine the success probability $p_{\mathsf{succ}} = \Pr[S \neq \emptyset]$, where S denotes the following subset of the lattice Λ:

$$S = \left\{ \mathbf{w} \in \Lambda \mid \begin{array}{l} \mathbf{w} = (\mathbf{w}_l, \mathbf{w}_g) \text{ with } \mathbf{w}_l \in \{0, \pm 1\}^{2n-r}, \mathbf{w}_g \in \{0, \pm 1\}^r, \\ \text{exactly } 2c_i \text{ entries of } \mathbf{w}_g \text{ equal to } i \quad \forall i \in \{-1, 1\}, \\ \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{w}_g) = \mathbf{w}_l \end{array} \right\},$$

and $\mathbf{B}$ is as defined in Heuristic 5.1. We assume that S is a subset of all the rotations of $\mathbf{v}$ that can be utilized by the attack and their additive inverses. In particular, we assume that S has at most $2(n-t)$ elements. Note that if some vector $\mathbf{w}$ is contained in S, then we also have $-\mathbf{w} \in S$. Assuming independence, the probability p_S that $\mathbf{v} \in S$ is approximately given by

$$p_S \approx \frac{\binom{r}{2\tilde{c_0}, 2c_{-1}, 2c_1} \binom{n-r}{2t-4c_1} 2^{2t-4c_1}}{\binom{n}{2t} 2^{2t}} \cdot p_{\mathrm{NP}},$$

where $d_0 = n - 2t$ and $2\tilde{c_0} = r - 4c_1$ and p_{NP} is defined and calculated as in Section 5.3.2. Assuming independence, all of the $n-t$ good rotations of $\mathbf{v}$ are contained in S with probability p_S as well. Therefore, the probability p_{succ} that we have at least one good rotation is approximately

$$p_{\mathsf{succ}} = \Pr[S \neq \emptyset] \approx 1 - (1-p_S)^{n-t}.$$

Next, we estimate the size of the set S in the case $S \neq \emptyset$, i.e., Algorithm 5 is successful. In that case, at least one rotation is contained in S. Then also its additive inverse is contained in S, hence $|S| \geq 2$. We can estimate the size of S in case of success to be

$$|S| \approx 2 + 2(n-t-1)p_S,$$

where p_S is defined as above.

Optimizing the Runtime

We applied our new techniques to estimate the minimal runtimes for several NTRU prime parameter sets proposed in the Appendix of [BCLvV16]. Besides the "case study parameter set", for our analysis we picked one parameter set that offers the lowest security level and one that offers the highest according to the analysis of [BCLvV16]. Our resulting security estimates and corresponding attack parameters are presented in Table 5.4. The table further provides a comparison to the primal attack under the 2016 estimate (cf. Chapter 3). Our analysis shows that the authors of [BCLvV17b, BCLvV16] underestimate the security of their scheme and that the hybrid attack outperforms the primal attack for all parameter sets we evaluated.

5.4.3 R-BinLWEEnc

In [BGG+16], Buchmann et al. presented R-BinLWEEnc, a lightweight public-key encryption scheme based on binary Ring-LWE. To determine the security of their scheme the authors evaluate the hardness of binary LWE against the hybrid attack. They use a simplified version of the methodology presented in this chapter, which ignores the success probability of the Nearest Planes algorithm and uses the simplified

Parameter set	$\mathbf{n = 607}$ $\mathbf{q = 18749}$	$\mathbf{n = 739}$ $\mathbf{q = 9829}$	$\mathbf{n = 929}$ $\mathbf{q = 12953}$
Optimal $r_{\mathsf{under}}/r_{\mathsf{over}}$	148/162	235/257	328/353
Optimal $\delta_{\mathsf{r,under}}$	1.00466	1.00405	1.00346
Optimal $\delta_{\mathsf{r,over}}$	1.00466	1.00407	1.00346
Optimal $\beta_{\mathsf{r,under}}$	318	391	489
Optimal $\beta_{\mathsf{r,over}}$	318	388	489
Corresponding p under/over	$2^{-63}/2^{-54}$	$2^{-73}/2^{-60}$	$2^{-80}/2^{-65}$
Security under/over in bits	**197/211**	**258/273**	**346/363**
In [BCLvV16]	128	228	310
2016 est. under/over	235/241	344/350	478/485
β_{2016}	364	487	627

Table 5.4: Optimal attack parameters and security levels against the hybrid attack and the primal attack under the 2016 estimate for NTRU prime.

formulas of [LP11] to estimate the runtime for lattice reduction and Nearest Plane. Therefore we reevaluate the security of binary LWE against the hybrid attack in order to obtain updated security estimates for R-BinLWEEnc.

Constructing the Lattice

Let $m, n, q \in \mathbb{Z}$ with $m > n$ and $(\mathbf{A}, \mathbf{b}' = \mathbf{A}\mathbf{s} + \mathbf{e}' \mod q)$ be a binary LWE instance with $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$, and binary error $\mathbf{e}' \in \{0,1\}$.[11] To obtain a more efficient attack, we first subtract the vector $(0.5, \dots, 0.5, 0, \dots, 0)$ with $m - r$ non-zero and r zero entries from both sides of the equation $\mathbf{b}' = \mathbf{A}\mathbf{s} + \mathbf{e}' \mod q$ to obtain a new LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \mod q)$, where $\mathbf{e} \in \{\pm 0.5\}^{m-r} \times \{0,1\}^r$. This way, the expected norm of the first $m - r$ entries is reduced while the last r entries, which are guessed during the attack, remain unchanged. In the following, we only consider this transformed LWE instance with smaller error. We use Kannan's embedding with embedding factor 1 to transform this LWE instance into an instance of the uSVP. Ignoring the additional component introduced by the embedding (as we know it is equal to the embedding factor and hence does not need to be guessed), the dimension of the uSVP lattice is m and its determinant is q^{m-n}. In the [BGG+16] encryption scheme, $m = 2n$ samples are provided, which we use in our attack.

[11]Note that with our approach we only need that error vector $\mathbf{e}'$ is binary, and not also that the secret vector $\mathbf{s}$ is binary, as demanded in [BGG+16].

Determining the Attack Parameters

Let $\mathbf{v} = \mathbf{e} = (\mathbf{v}_l, \mathbf{v}_g)$ with $\mathbf{v}_l \in \{\pm 0.5\}^{m-r}$ and $\mathbf{v}_g \in \{0,1\}^r$. We set the infinity norm bound y on $\mathbf{v}_l$ to be 0.5. Since $\mathbf{v}_l$ is a uniformly random vector in $\{\pm 0.5\}^{m-r}$, the expected Euclidean norm of $\mathbf{v}_l$ is

$$\|\mathbf{v}_l\| \approx \sqrt{\frac{m-r}{4}}.$$

We set $2c_0 = 2c_1 = \frac{r}{2}$ to be the expected number of 0 and 1 entries of $\mathbf{v}_g$. In the following, we assume that $c_0 = c_1$ is an integer in order to not have to deal with rounding operators.

Determining the Success Probability

We can approximate the success probability p_{succ} by $p_{\mathsf{succ}} \approx p_c \cdot p_{\mathrm{NP}}$, where p_c is the probability that $\mathbf{v}_g$ has exactly $2c_0$ entries equal to 0 and $2c_1$ entries equal to 1 and p_{NP} is defined as in Section 5.3.2. Using the fact that $2c_0 + 2c_1 = r$, we therefore obtain

$$p_{\mathsf{succ}} \approx p_c \cdot p_{\mathrm{NP}} = 2^{-r} \binom{r}{2c_0} p_{\mathrm{NP}}.$$

We assume that if the attack is successful then $|S| = 1$, where S is defined as in Heuristic 5.1, since $\mathbf{e}$ and is assumed to be the only vector that can be found by the attack.

Optimizing the Runtime

We reevaluated the security of the R-BinLWEEnc parameter sets of [BGG+16]. Our security estimates, the optimal attack parameters r, δ_r and β_r, and the corresponding probability p are presented in Table 5.5. The table also provides a comparison to the primal attack under the 2016 estimate (cf. Chapter 3). The results show that the original security estimates given in [BGG+16] are within the security range we determined and that the hybrid attack outperforms the primal attack for the analyzed binary LWE instances.

5.4.4 BLISS

In the following, we analyze the signature scheme BLISS introduced in [DDLL13]. In the original paper, the authors considered the hybrid attack on their signature scheme for their security estimates, but the analysis is rather vague and simplified. For instance, the authors assume that collisions will always be detected and do not optimize the attack parameters, which ignores the fact that there is a non-trivial trade-off between lattice reduction and the meet-in-the-middle phase. We therefore provide updated security for the BLISS signature scheme.

Parameter set	**Set-I**	**Set-II**	**Set-III**
Optimal $r_{\mathsf{under}}/r_{\mathsf{over}}$	112/108	88/100	264/272
Optimal $\delta_{\mathsf{r,under}}$	1.00688	1.00731	1.00481
Optimal $\delta_{\mathsf{r,over}}$	1.00706	1.00738	1.00485
Optimal $\beta_{\mathsf{r,under}}$	173	156	304
Optimal $\beta_{\mathsf{r,over}}$	165	153	300
Corresponding p under/over	$2^{-28}/2^{-31}$	$2^{-31}/2^{-25}$	$2^{-43}/2^{-29}$
Security under/over in bits	**89/99**	**79/89**	**186/197**
In [BGG+16]	94	84	190
2016 est. under/over	122/128	101/108	316/323
β_{2016}	222	189	458

Table 5.5: Optimal attack parameters and security levels against the hybrid attack and the primal attack under the 2016 estimate for R-BinLWEEnc.

Constructing the Lattice

In the BLISS signature scheme the setup is as follows. Let n be a power of two, $d_1, d_2 \in \mathbb{N}$ such that $d_1 + d_2 \leq n$ holds, q a prime modulus with $q \equiv 1 \mod 2n$, and $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. The signing key is of the form $(s_1, s_2) = (f, 2g + 1)$, where $f \in \mathcal{R}_q^\times, g \in \mathcal{R}_q$, each with d_1 coefficients in $\{\pm 1\}$ and d_2 coefficients in $\{\pm 2\}$, and the remaining coefficients equal to 0. The public key is essentially $a = s_2/s_1 \in \mathcal{R}_q$. We assume that a is invertible in $\mathcal{R}_q$, which is the case with very high probability. Hence we obtain the equation $s_1 = s_2 a^{-1} \in \mathcal{R}_q$, or equivalently $f = 2ga^{-1} + a^{-1} \mod q$. In the following, we identify polynomials with their coefficient vectors.

In order to recover the signing key, it is sufficient to find the vector $\mathbf{v} = (\mathbf{f}, \mathbf{g})$. Similar to our previous analysis of NTRU in Section 5.4.1 we have that

$$\mathbf{v} + \begin{pmatrix} -\mathbf{a}^{-1} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 2\mathbf{g}\mathbf{a}^{-1} + q\mathbf{w} \\ \mathbf{g} \end{pmatrix} = \left(\begin{array}{c|c} q\mathbf{I}_n & 2\mathbf{A} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array} \right) \begin{pmatrix} \mathbf{w} \\ \mathbf{g} \end{pmatrix}$$

for some $\mathbf{w} \in \mathbb{Z}^n$, where $\mathbf{A}$ is the rotation matrix of $\mathbf{a}^{-1}$. Hence $\mathbf{v}$ can be recovered by solving BDD on input $(\mathbf{a}^{-1}, \mathbf{0})$ in the q-ary lattice

$$\Lambda = \Lambda\left(\left(\begin{array}{c|c} q\mathbf{I}_n & 2\mathbf{A} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array} \right) \right),$$

since $(\mathbf{a}^{-1}, \mathbf{0}) - \mathbf{v} \in \Lambda$. The determinant of the lattice Λ is q^n and its dimension is equal to $2n$.

Determining the Attack Parameters

In the following, let $\mathbf{v} = (\mathbf{f}, \mathbf{g}) = (\mathbf{v}_l, \mathbf{v}_g)$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Since we are using the hybrid attack to solve a BDD problem, it is not known how to utilize the rotations of $\mathbf{v}$ within the attack, see Section 5.2. We therefore assume that $\mathbf{v}$ is the only rotation useful in the attack, i.e., that the set the set of good rotations S contains at most $\mathbf{v}$. The first step is to determine proper bounds y on $\|\mathbf{v}_l\|_\infty$ and k on $\|\mathbf{v}_g\|_\infty$ and find suitable guessing parameters c_i. By construction we have $\|\mathbf{v}\|_\infty \leq 2$, thus we can set the infinity norm bounds $y = k = 2$. The expected Euclidean norm of $\mathbf{v}_l$ is given by

$$\|\mathbf{v}_l\| \approx \sqrt{d_1 + 4d_2 + \frac{n-r}{n}(1d_1 + 4d_2)}.$$

We set $2c_i$ equal to the expected number of i-entries in $\mathbf{v}_g$, i.e., $c_{-2} = c_2 = \frac{r}{n} \cdot \frac{1}{4} d_2$ and $c_{-1} = c_1 = \frac{r}{n} \cdot \frac{1}{4} d_1$. For simplicity we assume that c_1 and c_2 are integers in the following.

Determining the Success Probability

Next, we determine the success probability p_{succ}, which is the probability that $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$ and exactly $2c_i$ entries of $\mathbf{v}_g$ are equal to i for $i \in \{\pm 1, \ldots, \pm k\}$. The probability p_c that exactly $2c_i$ entries of the vector $\mathbf{v}_g$ are equal to i for all $i \in \{\pm 1, \ldots, \pm k\}$ is given by

$$\frac{\binom{r}{2\tilde{c_0}, 2c_{-2}, 2c_2, 2c_{-4}, 2c_4} \binom{n-r}{d_0 - 2\tilde{c_0}, d_1 - 4c_2, d_2 - 4c_4} 2^{d_1 + d_2 - 4(c_2 + c_4)}}{\binom{n}{d_0, d_1, d_2} 2^{d_1 + d_2}},$$

where $d_0 = n - d_1 - d_2$ and $2\tilde{c_0} = r - 2(c_{-2} + c_2 + c_{-4} + c_4)$. Assuming independence, the success probability is approximately given by

$$p_{\mathsf{succ}} \approx p_c \cdot p_{\mathrm{NP}},$$

where p_{NP} is defined as in Section 5.3.2. As explained earlier, we assume that $S \subset \{\mathbf{v}\}$, so if Algorithm 5 is successful we have $|S| = 1$.

Optimizing the Runtime

We performed the optimization process for the BLISS parameter sets proposed in [DDLL13]. The results are presented in Table 5.6. Besides the security levels against the hybrid attack, we provide the optimal attack parameters r, δ_r, and β_r

Parameter set	**BLISS-I,II**	**BLISS-III**	**BLISS-IV**
Optimal $r_{\mathsf{under}}/r_{\mathsf{over}}$	152/152	109/144	99/137
Optimal $\delta_{\mathsf{r,under}}$	1.00588	1.00532	1.00518
Optimal $\delta_{\mathsf{r,over}}$	1.00600	1.00541	1.00524
Optimal $\beta_{\mathsf{r,under}}$	223	261	271
Optimal $\beta_{\mathsf{r,over}}$	216	254	264
Corresp. p under/over	$2^{-35}/2^{-38}$	$2^{-58}/2^{-40}$	$2^{-67}/2^{-44}$
Security under/over in bits	**124/139**	**152/170**	**160/182**
In [DDLL13]	128	160	192
r used in [DDLL13]	194	183	201
2016 est. under/over	159/165	176/182	183/189
β_{2016}	270	292	301

Table 5.6: Optimal attack parameters and security levels against the hybrid attack and the primal attack under the 2016 estimate for BLISS.

leading to a minimal runtime of the attack as well as the corresponding probability p. The table further provides a comparison to the primal attack under the 2016 estimate (cf. Chapter 3). Our results show that the security estimates for the BLISS-I, BLISS-II, and BLISS-III parameter sets given in [DDLL13] are within the range of security we determined, whereas the BLISS-IV parameter set is less secure than originally claimed. In addition, the authors of [DDLL13] claim that there are at least 17 bits of security margins built into their security estimates, which is not the case for all parameter sets according to our analysis. Furthermore, our results show the the hybrid attack performs better than the primal attack on BLISS.

5.4.5 GLP

The GLP signature scheme was introduced in [GLP12]. In their original work, the authors did not consider the hybrid attack when deriving their security estimates. Later, in [DDLL13], the hybrid attack was also applied to the GLP-I parameter set. However, the analysis of the hybrid attack against GLP presented in [DDLL13] is simplified in the same way as the analysis of the BLISS signature scheme, see Section 5.4.4. Furthermore, the GLP-II parameter set has not been analyzed with respect to the hybrid attack so far. We therefore reevaluate the security of the GLP-I parameter set against the hybrid attack and firstly evaluate the hybrid attack security of the GLP-II parameter set.

Constructing the Lattice

For the GLP signature scheme the setup is as follows. Let n be a power of two, q a prime modulus with $q \equiv 1 \mod 2n$, and $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. The signing key is of the form (s_1, s_2), where s_1 and s_1 are sampled uniformly at random among all polynomials of $\mathcal{R}_q$ with coefficients in $\{-1, 0, 1\}$. The corresponding public key is then of the form $(a, b = as_1 + s_2) \in \mathcal{R}_q^2$, where a is drawn uniformly at random in $\mathcal{R}_q$. So we know that $0 = -b + as_1 + s_2$. Identifying polynomials with their coefficient vectors we therefore have that

$$\mathbf{v} := \begin{pmatrix} -1 \\ \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \in \Lambda := \Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{w} \in \mathbb{Z}^{2n+1} \mid \mathbf{A}\mathbf{w} \equiv \mathbf{0} \mod q\} \subset \mathbb{Z}^{2n+1},$$

where $\mathbf{A} = (\mathbf{b}|\mathsf{rot}(\mathbf{a})|\mathbf{I}_n)$ and $\mathsf{rot}(\mathbf{a})$ is the rotation matrix of $\mathbf{a}$ (cf. Section 3.3.1). By construction of the lattice we do not assume that rotations of $\mathbf{v}$ can by utilized by the attack. Therefore, with very high probability $\mathbf{v}$ and $-\mathbf{v}$ are the only non-zero ternary vectors contained in Λ, which we assume in the following. For the determinant of the lattice we have $\det \Lambda = q^n$, see Section 3.3.1.

Determining the Attack Parameters

Ignoring the first -1 coordinate, the short vector $\mathbf{v}$ is drawn uniformly from $\{-1, 0, 1\}^{2n+1}$. Let $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Then $\|\mathbf{v}_l\|_\infty \leq 1$ and $\|\mathbf{v}_g\|_\infty \leq 1$ hold, so we can set the infinity norm bounds y and k equal to one. The expected Euclidean norm of $\mathbf{v}_l$ is approximately

$$\|\mathbf{v}_l\| \approx \sqrt{2(2n + 1 - r)/3}.$$

We set $2c_{-1} = 2c_1 = \frac{r}{3}$ to be the expected number of ones and minus ones. For simplicity we assume that $c_{-1} = c_1$ is an integer in the following.

Determining the Success Probability

The success probability p_{succ} of the attack is approximately $p_{\mathsf{succ}} \approx p_c \cdot p_{\mathrm{NP}}$, where p_c is the probability that $\mathbf{v}_g$ hat exactly $2c_{-1}$ minus one entries and $2c_1$ one entries and p_{NP} is defined as in Section 5.3.2. Calculating p_c yields

$$p_{\mathsf{succ}} \approx p_c \cdot p_{\mathrm{NP}} = 3^{-r} \binom{r}{r/3, r/3, r/3} p_{\mathrm{NP}}.$$

As previously mentioned, we assume that if the attack is successful then $|S| = 2$.

Parameter set	**GLP-I**	**GLP-II**
Optimal $r_{\mathsf{under}}/r_{\mathsf{over}}$	30/54	168/192
Optimal $\delta_{\mathsf{r,under}}$	1.00776	1.00450
Optimal $\delta_{\mathsf{r,over}}$	1.00769	1.00451
Optimal $\beta_{\mathsf{r,under}}$	140	335
Optimal $\beta_{\mathsf{r,over}}$	143	334
Corresponding p under/over	$2^{-41}/2^{-25}$	$2^{-61}/2^{-49}$
Security under/over in bits	**71/88**	**212/233**
In [DDLL13], [GLP12]	75 to 80	≥ 256
r used in [DDLL13]	85	—
2016 est. under/over	71/77	237/243
β_{2016}	142	366

Table 5.7: Optimal attack parameters and security levels against the hybrid attack and the primal attack under the 2016 estimate for GLP.

Optimizing the Runtime

Weoptimized the runtime of the hybrid attack for the GLP parameter sets proposed in [GLP12]. The results, including the optimal attack parameters r, δ_r, and β_r and the probability p, are shown in Table 5.7. In addition, the table provides a comparison to the primal attack under the 2016 estimate (cf. Chapter 3). The security level of the GLP-I parameter set claimed in [DDLL13] is within the range of security we determined. Furthermore, for the GLP-I parameter set the hybrid attack performs similar to the primal attack. In [DDLL13], the authors did not analyze the hybrid attack for the GLP-II parameter set. Güneysu et al. [GLP12] claimed a security level of at least 256 bits (not considering the hybrid attack) for the GLP-II parameter set, whereas we show that it offers at most 233 bits of security against the hybrid attack and at most 243 bits against the primal attack considering the 2016 estimate.

6 Parallelizing the Hybrid Lattice Reduction and Meet-in-the-Middle Attack

The hybrid attack (see Chapter 5) is currently considered the best known attack on several instances of lattice problems with small or sparse secret vectors. In order to evaluate the security of certain lattice-based cryptosystems (such as [HPS98, BCLvV17b, BGG$^+$16, DDLL13, GLP12, CHK$^+$17, HS14]) it is therefore important to study the practical behavior of the hybrid attack. To reflect the full potential of the hybrid attack in practice it has to be parallelized.

Contribution. In this chapter, we show how to parallelize the hybrid attack using three strategies: running the attack on multiple randomized instances in parallel, parallelizing its meet-in-the-middle phase, and potentially using a parallel version of the BKZ lattice reduction algorithm. For simplicity, we restrict our studies to the hybrid attack on binary LWE, where the LWE error distribution is the uniform distribution on $\{0, 1\}$. We provide a theoretical and experimental analysis of our parallel hybrid attack, which shows that it scales well within reasonable parameter ranges. Our theoretical analysis depends on the efficiency of a potential parallel BKZ algorithm and the efficiency of the parallel meet-in-the-middle phase. It shows that the efficiency of the parallel hybrid attack is at least as good as the worse of these two efficiencies (as long as the number of nodes employed is within a certain range), but may in general be better. For our practical implementations, we employ OpenMP and the Message Passing Interface (MPI). Our experiments show that the parallel hybrid attack can considerably speed up the attack by running multiple, randomized instances in parallel with minimized MPI communication. We further analyze the efficiency of a parallel meet-in-the-middle search within the hybrid attack. Our meet-in-the-middle phase is shared-memory parallelized and we report an efficiency of about 90% on our system providing 24 physical cores per node. Our results suggest that the above-mentioned cryptosystems may in practice be broken significantly faster using our parallel hybrid attack.

Organization. In Section 6.1, we specify the serial hybrid attack on binary LWE as a foundation for our parallel version. In Section 6.2, we show how to parallelize the hybrid attack and analyze the runtime of the parallel hybrid attack from a theoretic point of view. Our experimental analysis is presented in Section 6.3.

Publications. This chapter is based on the publication [8], which will be presented at CSE 2018.

6.1 The Hybrid Attack on Binary LWE

In this section, we specify the serial hybrid attack on binary LWE. We largely follow the description given in Chapter 5 with slight modifications. Let $q \in \mathbb{N}$ and $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$ be a binary LWE instance with $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{b} \in \mathbb{Z}_q^m$, $\mathbf{s} \in \mathbb{Z}_q^n$, and $\mathbf{e} \in \{0,1\}^m$. We use Kannan's embedding (see Section 2.4.4) with embedding factor 1 to transform LWE into uSVP (containing $(\mathbf{e}, 1)$ as short binary vector) and then run the hybrid attack. Our modification from Chapter 5 are the following. As we know that the last component of the short binary vector is 1, we set the last entry of the vectors guessed in the meet-in-the-middle search equal to 0.5. Furthermore, we use the following sets of addresses for our meet-in-the-middle search.

$$\mathcal{A}_{\mathbf{x}} = \left\{ \mathbf{a} \in \{0,1\}^k \,\middle|\, \begin{array}{l} (\mathbf{a})_i = 1 \text{ if } (\mathbf{x})_i > 0 \text{ for } i \in \{1,\dots,k\}, \\ (\mathbf{a})_i = 0 \text{ if } (\mathbf{x})_i < 0 \text{ for } i \in \{1,\dots,k\} \end{array} \right\}. \tag{6.1}$$

The modified pseudocode for the hybrid attack on binary LWE is given in Algorithm 7. It takes as input a binary LWE instance, a guessing dimension r, and a block size β, which determines the quality of the precomputation. The attack aims at finding the LWE error vector. For simplicity, we assume that r is a multiple of 4 such that we can guess binary vectors with exactly $c = r/4$ non-zero entries in the meet-in-the-middle search of the attack. Lines 1 and 2 describe the precomputation phase of the attack with BKZ-β being its computational hotspot. Lines 5 to 13 describe the meet-in-the-middle phase of the attack. Note that the attack might have a low success probability as detailed in Chapter 5. The success probability (and the runtime of the attack) depends on the attack parameters r and β, which therefore need to be chosen carefully. Because of the possibly low success probability, in general, the attack needs to be randomized and repeated multiple times until successful.

6.2 Parallelizing the Hybrid Attack

In this section, we describe how one can parallelize the hybrid attack and analyze the resulting theoretical speedup. Throughout this chapter, we focus on the runtime as a metric for the attack. Our analysis depends on the number of nodes and cores per

Algorithm 7: The hybrid attack on binary LWE

```
Input: A modulus $q$, a binary LWE instance $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^m$, a
       guessing dimension $r \in \mathbb{N}$ with $4 \mid r < m+1$, a block size $\beta$
1  compute a basis $\mathbf{B}'$ of $\Lambda_{(\mathbf{A},\mathbf{b},q)}$ of the form
       $$\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_{r+1} \end{array}\right) \in \mathbb{Z}^{(m+1)\times(m+1)},$$
     where $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$ and $\mathbf{C} \in \mathbb{Z}^{(m-r)\times(r+1)}$;
2  BKZ-$\beta$-reduce the upper-left block $\mathbf{B}$;
3  set $c = r/4$;
4  while true do
5      guess $\mathbf{w}' \in \{0,1\}^r$ with exactly $c$ non-zero entries and set $\mathbf{v}_g' = (\mathbf{w}', 0.5)$;
6      calculate $\mathbf{v}_l' = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g')$ ;
7      store $\mathbf{v}_g'$ in all the boxes addressed by $\mathcal{A}_{\mathbf{v}_l'} \cup \mathcal{A}_{-\mathbf{v}_l'}$;
8      for all $\mathbf{v}_g'' \neq \mathbf{v}_g'$ in all the boxes addressed by $\mathcal{A}_{\mathbf{v}_l'} \cup \mathcal{A}_{-\mathbf{v}_l'}$ do
9          set $\mathbf{v}_g = \mathbf{v}_g' + \mathbf{v}_g''$;
10         if $\mathbf{v}_g \in \{0,1\}^{r+1}$ then
11             calculate $\mathbf{v}_l = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) \in \mathbb{Z}^{m-r}$;
12             if $\mathbf{v} = \begin{pmatrix} \mathbf{v}_l \\ \mathbf{v}_g \end{pmatrix} \in \Lambda_{(\mathbf{A},\mathbf{b},q)} \cap \{0,1\}^{m+1}$ then
13                 return $\mathbf{v}$;
```

node. For the rest of this section, k denotes the number of nodes and l the number of cores per node, hence in total we have kl cores. We assume that cores on the same node can communicate and share a common memory, whereas this is not the case across different nodes. Therefore, cores on the same node play a different role than cores on different nodes. We are interested in the efficiency of parallel algorithms, which is measured by

$$E(X_1, \ldots, X_h, C) = \frac{\left(\frac{T(X_1,\ldots,X_h,1)}{T(X_1,\ldots,X_h,C)}\right)}{C},$$

where C is the total number of cores, $T(X_1, \ldots, X_h, i)$ is the runtime of the algorithm on i cores, and $X_1, \ldots, X_h$ are the inputs of the algorithm.

Our measures to parallelize the hybrid attack are the following:

1. Running the attack on multiple randomized instances in parallel.
2. Potentially using a parallel version of BKZ.

3. Performing the meet-in-the-middle search in parallel.

In the following, we discuss these measures in more detail.

6.2.1 Running Multiple Instances in Parallel

The hybrid attack suffers from a possibly low success probability p_{succ} (cf. Chapter 5). It is therefore expected that the attack needs to be randomized and repeated approximately $1/p_{\mathsf{succ}}$ times until it is successful. This can be done in parallel on different cores. As different executions of the attack are independent, these cores can be located on different nodes.

In the following, we elaborate on how to randomize the instances for the attack. The two components of the overall success probability are i) the probability that the last components of the searched vector (in our case the binary LWE error vector) which are guessed in the meet-in-the-middle phase have a certain structure[12] and ii) the success probability of the Nearest Plane algorithm. The first probability depends on the structure of the searched vector, while the second depends on the quality of the reduced basis. Thus, our strategy to randomize the instances is twofold.

First, we permute the LWE samples by permuting the rows of the input LWE instance $(\mathbf{A}, \mathbf{b})$. Then it holds that $\tau(\mathbf{b}) = \tau(\mathbf{A})\mathbf{s} + \tau(\mathbf{e}) \bmod q$, where τ is some permutation of the rows of a matrix or the entries of a vector, respectively. In this way, we randomize the structure of the last components of the LWE error vector. It can also be viewed as guessing other entries of the LWE error vector than the last ones. Note that in this case, the attack potentially finds the permutation $\tau(\mathbf{e})$ instead of the original error vector $\mathbf{e}$.

Second, before BKZ-reducing the upper-left part $\mathbf{B}$ of the full basis $\mathbf{B}'$ (Line 2 of Algorithm 7), we randomize this part by multiplying it with a random unimodular matrix. This procedure randomizes the BKZ-reduced basis while preserving the lattice.

The benefit of running multiple randomized instances of the attack in parallel is experimentally verified in Section 6.3.5.

6.2.2 Using Parallel BKZ

The two most time-consuming steps of the hybrid attack are the BKZ lattice reduction (precomputation) step (Line 2 of Algorithm 7) and the meet-in-the-middle phase (Lines 5 to 13 of Algorithm 7). These steps may be parallelized. A summary of the state-of-the-art regarding a parallel BKZ algorithm is given in [MLC$^+$17]. To the best of the authors' knowledge, there are no results published about the performance and scalability of a parallel BKZ 2.0 algorithm. For this chapter, we assume that the

[12]For example, if the searched vector is binary, the structure would be a certain number of non-zero entries.

BKZ or BKZ 2.0 algorithm may be parallelized (in a black box manner), but assume that this needs to be done on a single node. We do not analyze the scalability of parallel BKZ, as this is out of the scope of this work.

6.2.3 Parallel Meet-in-the-Middle Search

Besides lattice reduction, the meet-in-the-middle phase (Lines 5 to 13 of Algorithm 7) is the most time-consuming part of the hybrid attack. For the meet-in-the-middle phase, an enormous number of vectors needs to be guessed and checked for possible collisions that lead to the solution. We propose to perform this guessing and collision search in parallel. To this end, all guessing and collision search threads (of one individual randomized instance only) need to operate on a shared hash map. We therefore assume that the parallel meet-in-the-middle search for one individual instance needs to be performed on a single node. We investigate the parallel efficiency of the meet-in-the-middle phase in Section 6.3.4.

Note that a bottleneck of the meet-in-the-middle search is its memory consumption. A reduced memory version (which comes at the cost of a slower runtime) of a pure meet-in-the-middle attack [HGSW] on NTRU has been presented in [vV16]. The attack is based on a "golden" collision search which has been parallelized in [vW96, vW99]. However, it is unclear if the memory reduction techniques of [vV16] can be applied to the hybrid attack. This is due to the fact that the meet-in-the-middle search of [vV16] can only find one possible solution, which may be unlikely to be found within the hybrid attack due to the low collision-finding probability. In contrast, for the meet-in-the-middle search of the hybrid attack there are many possible collisions, which makes it very likely that one of them will be found. We therefore do not consider the above techniques in this chapter.

6.2.4 Runtime Analysis

A detailed runtime analysis of the serial hybrid attack can be found in Chapter 5. In Chapter 5, over- and underestimates of the runtime of the hybrid attack are presented. The underestimates represent potential algorithmic improvements which have not yet been shown to be applicable in practice. Since this chapter is focused on the practicality of the hybrid attack, we only consider the overestimates.

Let β be the block size used for the lattice reduction step and r be the guessing dimension used in the hybrid attack. The parameters β and r can be chosen by the attacker, while the others (n, m, q) are fixed by the given LWE instance and therefore not mentioned explicitly in the following. Then, according to Chapter 5, the expected total runtime of the serial hybrid attack can be expressed as

$$T_{\mathsf{total}}(\beta, r) = \frac{T_{\mathsf{BKZ}}(\beta, r) + T_{\mathsf{hyb}}(\beta, r)}{p_{\mathsf{succ}}(\beta, r)},$$

where the runtime T_{BKZ} of BKZ, the runtime T_{hyb} of the meet-in-the-middle phase, and the overall success probability p_{succ} can be estimated as in Chapter 5.[13] In order to minimize the runtime of the serial hybrid attack, the total runtime must be minimized over all possible choices of β and r as described in Chapter 5.

In the following, we show how to make use of the available cores and determine the theoretical runtime $T_{\mathsf{total,p}}(\beta, r, k, l)$ of the parallel hybrid attack when using k nodes with l cores per node. Let $T_{\mathsf{BKZ,p}}(\beta, r, k, l)$ and $T_{\mathsf{hyb,p}}(\beta, r, k, l)$ denote the runtimes of parallel BKZ and the meet-in-the-middle guessing phase. As described in Section 6.2.1, we expect to find the solution after about

$$N(\beta, r) = \frac{1}{p_{\mathsf{succ}}(\beta, r)}$$

repetitions of the attack, which can be performed in parallel. We (optimistically) expect this to scale optimally until the total number of cores used exceeds $N(\beta, r)$. Hence we use approximately $\min(N(\beta, r), kl)$ cores to run about $\min(N(\beta, r), kl)$ randomized instances in parallel, reducing the time of the parallel hybrid attack to approximately

$$T_{\mathsf{total,p}}(\beta, r, k, l) = \max\left(1, \frac{N(\beta, r)}{kl}\right) \cdot (T_{\mathsf{BKZ,p}}(\beta, r, k, l) + T_{\mathsf{hyb,p}}(\beta, r, k, l)).$$

Per randomized instance, there remain about

$$\max\left(1, \frac{kl}{N(\beta, r)}\right)$$

cores to use for BKZ and the meet-in-the-middle phase, i.e., to reduce the parallel runtimes $T_{\mathsf{BKZ,p}}(\beta, r, k, l)$ and $T_{\mathsf{hyb,p}}(\beta, r, k, l)$. However, since we assume that BKZ as well as the meet-in-the-middle phase need to be parallelized on a single node, we can use at most l of them per instance. Summarizing, this results in the following heuristic to estimate the runtime of the parallel hybrid attack.

Heuristic 6.1. *Let β, r, $T_{\mathsf{BKZ}}(\beta, r)$, $T_{\mathsf{hyb}}(\beta, r)$, $p_{\mathsf{succ}}(\beta, r)$, and $N(\beta, r) = 1/p_{\mathsf{succ}}(\beta, r)$ be as above. Then the total runtime $T_{\mathsf{total,p}}(\beta, r, k, l)$ of the parallel hybrid attack on k nodes with l cores per node is approximately*

$$T_{\mathsf{total,p}}(\beta, r, k, l) = \max\left(1, \frac{N(\beta, r)}{kl}\right) \cdot \left(\frac{T_{\mathsf{BKZ}}(\beta, r)}{M_{\mathsf{BKZ}}(\beta, r, k, l)} + \frac{T_{\mathsf{hyb}}(\beta, r)}{M_{\mathsf{hyb}}(\beta, r, k, l)}\right),$$

where

$$M_{\mathsf{BKZ}}(\beta, r, k, l) = E_{\mathsf{BKZ}}(\beta, r, C(\beta, r, k, l)) \cdot C(\beta, r, k, l),$$

[13] Note that in Chapter 5, the estimated number of operations is given instead of the runtime. However, knowing how many operations can be performed per second, these can be transformed into each other.

$$M_{\mathsf{hyb}}(\beta, r, k, l) = E_{\mathsf{hyb}}(\beta, r, C(\beta, r, k, l)) \cdot C(\beta, r, k, l)$$

with

$$C(\beta, r, k, l) = \min\left(l, \max\left(1, \frac{kl}{N(\beta, r)}\right)\right),$$

and $E_{\mathsf{BKZ}}((\beta, r, i))$ and $E_{\mathsf{hyb}}((\beta, r, i))$ are the parallel efficiencies of BKZ and the meet-in-the-middle phase, respectively.

We make a few remarks regarding Heuristic 6.1.

Remark 6.1.

1. *We emphasize that for each combination of k and l, the attack parameters r and β must be re-optimized, as in general – when focusing on the runtime – this yields a better attack than naively using the same attack parameters as for the serial hybrid attack.*

2. *Note that as long as the total number of cores kl does not exceed the expected number of repetitions of the serial hybrid attack, i.e., as long as $kl \leq N(\beta_0, r_0)$ for the optimal attack parameters β_0, r_0 of the serial hybrid attack, one obtains 100% parallel efficiency of the hybrid attack by choosing β_0, r_0 as the attack parameters.*

3. *According to Heuristic 6.1, the parallel efficiency of the hybrid attack depends on the parallel efficiency of BKZ and the meet-in-the-middle phase. Note that the parallel efficiency of the entire attack is at least the minimum of these two efficiencies as long as the number of nodes does not exceed the expected number of repetitions of the attack of the serial hybrid attack, i.e., as long as $k \leq N(\beta_0, r_0)$ for the optimal attack parameters β_0, r_0 of the serial hybrid attack. In particular, if BKZ and the meet-in-the-middle phase scale ideally, so does the parallel hybrid attack as long as $k \leq N(\beta_0, r_0)$.*

4. *As can be seen in Heuristic 6.1, the runtime of the parallel hybrid attack does not only depend on the total number of cores kl that are used, but also on the configuration, i.e., on how many cores l there are per node. In particular, if $k > N(\beta, r)$ holds, increasing k may have a worse effect on the parallel efficiency of the attack than increasing l. As long as $k \leq N(\beta, r)$, however, this phenomenon does not occur, since in this case it holds that $kl/N(\beta, r) \leq l$ and hence we have $C(\beta, r, k, l) = \max(1, kl/N(\beta, r))$, which only depends on the product kl and not on the individual choices of k and l.*

5. *If there is only one core per node, i.e., $l = 1$, BKZ and the meet-in-the-middle-phase for each individual instance are not further parallelized. Hence, in this case, the efficiency of the hybrid attack is independent of the efficiencies of BKZ and the meet-in-the-middle phase.*

Examples and Discussion

We illustrate the theoretical efficiency of the parallel hybrid attack with some examples. We consider the binary LWE instance with parameters $n = 256, m = 512, q = 128$, which is underlying the first of the proposed instantiations of the encryption scheme by Buchmann et al. [BGG+16]. For simplicity, we do not shift the LWE error vector component-wise by 1/2, which leads to a slightly better attack (serial and parallel) as proposed in Chapter 5. For our examples, we assume that the efficiency functions $E_{\mathsf{BKZ}}(\beta, r, i)$ and $E_{\mathsf{hyb}}(\beta, r, i)$ are functions of the form

$$f_E(\beta, r, i) = \begin{cases} 1 & \text{for } i = 1 \\ E & \text{for } i > 1 \end{cases}$$

with $E \in \{0.1, 0.9\}$, giving four possible combinations. Note that efficiency functions of this form are somewhat pathological and not realistic for the practical behavior of parallel BKZ and a parallel meet-in-the-middle search. However, they allow us to showcase the effect of the individual efficiencies on the overall efficiency of the parallel hybrid attack. Furthermore, if the constants are viewed as possible lower bounds on the efficiency of BKZ and the meet-in-the-middle phase, respectively, our results can be interpreted as lower bounds on the theoretical efficiency of the parallel hybrid attack.

We combined the analysis of the serial hybrid attack provided in Chapter 5 with our analysis of the parallel hybrid attack and optimized the attack parameters for each individual configuration. For the number of operations required for BKZ-β in dimension d we use common $8d \cdot 2^{0.270\beta\ln(\beta) - 1.019\beta + 16.1}$ cost model [APS15] for enumeration-based BKZ. We use estimates for enumeration-based BKZ, as opposed to BKZ that uses sieving algorithms as SVP solvers, because enumeration algorithms currently seem to perform better in practice (as argued for example in [BCLvV17b]) and this chapter is considered with the practicality of the hybrid attack. In addition, the BKZ implementation used in our practical experiments uses enumeration as SVP solver. For the number of operations required by Nearest Plane in dimension d we use $d^2/(2^{1.06})$ as for our overestimates in Chapter 5.

Our results assuming efficiency $f_{0.1}$ for BKZ and the meet-in-the-middle phase are shown in Table 6.1. Our results assuming efficiency $f_{0.9}$ for BKZ and the meet-in-the-middle phase are shown in Table 6.2. Our results assuming efficiency $f_{0.1}$ for BKZ and efficiency $f_{0.9}$ for the meet-in-the-middle phase are shown in Table 6.3. Our results assuming efficiency $f_{0.9}$ for BKZ and efficiency $f_{0.1}$ for the meet-in-the-middle phase are shown in Table 6.4. According to our analysis, the serial hybrid attack requires roughly $2^{108.5}$ operations (including repetitions of the attack) and has a success probability of roughly $2^{-6.97}$.

In general, all of the above-mentioned tables confirm the behavior of the parallel hybrid attack described in Remark 6.1 and show that the parallel hybrid attack scales well within reasonable parameter ranges. We can make the following observations from the above mentioned tables.

1. In each case, the efficiency of the parallel hybrid attack is 100% as long as the total number of cores is at most 2^7, which is roughly the required number of repetitions of the serial hybrid attack.

2. The efficiency of the hybrid attack does not drop below the minimum of the two individual efficiencies of BKZ and the meet-in-the-middle phase as long as the number of nodes k is at most 2^7. Note however, that in general we achieve better efficiencies.

3. We can further observe that increasing the total number of cores by increasing number of cores per node has either the same or a better effect on the efficiency than doing so by increasing the number of nodes.

4. All tables indicate that for each number of nodes k there exists a number of cores per node l_k such that when increasing the number l of cores per node the efficiency remains constant and that this efficiency is gradually approached when increasing the l to l_k.

5. For $l = 1$, the tables confirm that the efficiency of the hybrid attack is independent of the efficiencies of BKZ and the meet-in-the-middle phase, i.e., the $l = 1$ column of all of the above-mentioned tables is the same.

6. Comparing Table 6.3 and Table 6.4, we see that having efficiency $f_{0.1}$ for BKZ and efficiency $f_{0.9}$ for the meet-in-the-middle phase has a better effect on the overall efficiency of the parallel hybrid attack than having efficiency $f_{0.9}$ for BKZ and efficiency $f_{0.1}$ for the meet-in-the-middle phase.

In Figure 6.1, we illustrate the improvement of optimizing the attack parameters individually for each configuration compared to using the optimal attack parameters of the serial hybrid attack for the $l = 1$ case.

6.3 Experiments and Results

In this section, we start by describing our implementation in Section 6.3.1, the test environment in Section 6.3.2, as well as the test cases employed in Section 6.3.3. Afterward, we present the results of our practical experiments in Sections 6.3.4, 6.3.5, and 6.3.6.

k \ l	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
2^0	100%	100%	100%	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	10%	10%
2^1	100%	100%	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	10%	10%	10%
2^2	100%	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	10%	10%	10%	10%
2^3	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	10%	10%	10%	10%	10%
2^4	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	10%	10%	10%	10%	10%	10%
2^5	100%	100%	100%	94%	72%	50%	44%	21%	20%	10%	10%	10%	10%	10%	10%	10%
2^6	100%	100%	94%	72%	50%	44%	21%	20%	10%	10%	10%	10%	10%	10%	10%	10%
2^7	100%	94%	72%	50%	44%	21%	20%	10%	10%	10%	10%	10%	10%	10%	10%	10%
2^8	94%	72%	50%	44%	21%	20%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%
2^9	72%	50%	44%	21%	20%	9%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%
2^{10}	52%	44%	21%	20%	9%	7%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
2^{11}	44%	21%	20%	9%	7%	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%	4%
2^{12}	24%	20%	9%	7%	4%	3%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
2^{13}	20%	9%	7%	4%	3%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
2^{14}	10%	7%	4%	3%	2%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%
2^{15}	8%	4%	3%	2%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%

Table 6.1: Parallel efficiency of the hybrid attack for k nodes with l cores each for binary LWE with $n = 256$, $m = 512$, $q = 128$ assuming efficiency $f_{0.1}$ for BKZ and the meet-in-the-middle phase.

k \ l	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
2^0	100%	100%	100%	100%	100%	100%	100%	100%	94%	90%	90%	90%	90%	90%	90%	90%
2^1	100%	100%	100%	100%	100%	100%	100%	94%	90%	90%	90%	90%	90%	90%	90%	90%
2^2	100%	100%	100%	100%	100%	100%	94%	90%	90%	90%	90%	90%	90%	90%	90%	90%
2^3	100%	100%	100%	100%	100%	94%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%
2^4	100%	100%	100%	100%	94%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%
2^5	100%	100%	100%	94%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%
2^6	100%	100%	94%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%
2^7	100%	94%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%	90%
2^8	94%	85%	85%	85%	85%	85%	85%	85%	85%	85%	85%	85%	85%	85%	85%	85%
2^9	72%	64%	64%	64%	64%	64%	64%	64%	64%	64%	64%	64%	64%	64%	64%	64%
2^{10}	52%	47%	47%	47%	47%	47%	47%	47%	47%	47%	47%	47%	47%	47%	47%	47%
2^{11}	44%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%
2^{12}	24%	21%	21%	21%	21%	21%	21%	21%	21%	21%	21%	21%	21%	21%	21%	21%
2^{13}	20%	18%	18%	18%	18%	18%	18%	18%	18%	18%	18%	18%	18%	18%	18%	18%
2^{14}	10%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%
2^{15}	8%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%

Table 6.2: Parallel efficiency of the hybrid attack for k nodes with l cores each for binary LWE with $n = 256$, $m = 512$, $q = 128$ assuming efficiency $f_{0.9}$ for BKZ and the meet-in-the-middle phase.

k \ l	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
2^0	100%	100%	100%	100%	100%	100%	100%	100%	94%	72%	56%	56%	56%	56%	56%	56%
2^1	100%	100%	100%	100%	100%	100%	100%	94%	72%	56%	56%	56%	56%	56%	56%	56%
2^2	100%	100%	100%	100%	100%	100%	94%	72%	56%	56%	56%	56%	56%	56%	56%	56%
2^3	100%	100%	100%	100%	100%	94%	72%	56%	56%	56%	56%	56%	56%	56%	56%	56%
2^4	100%	100%	100%	100%	94%	72%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%
2^5	100%	100%	100%	94%	72%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%
2^6	100%	100%	94%	72%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%
2^7	100%	94%	72%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%	56%
2^8	94%	72%	50%	49%	49%	49%	49%	49%	49%	49%	49%	49%	49%	49%	49%	49%
2^9	72%	50%	44%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%	39%
2^{10}	52%	44%	34%	34%	34%	34%	34%	34%	34%	34%	34%	34%	34%	34%	34%	34%
2^{11}	44%	21%	20%	19%	19%	19%	19%	19%	19%	19%	19%	19%	19%	19%	19%	19%
2^{12}	24%	20%	16%	16%	16%	16%	16%	16%	16%	16%	16%	16%	16%	16%	16%	16%
2^{13}	20%	9%	8%	8%	8%	8%	8%	8%	8%	8%	8%	8%	8%	8%	8%	8%
2^{14}	10%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%	7%
2^{15}	8%	4%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%

Table 6.3: Parallel efficiency of the hybrid attack for k nodes with l cores each for binary LWE with $n = 256$, $m = 512$, $q = 128$ assuming efficiency $f_{0.1}$ for BKZ and efficiency $f_{0.9}$ for the meet-in-the-middle phase.

k \ l	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
2^0	100%	100%	100%	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	16%	16%
2^1	100%	100%	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	16%	16%	16%
2^2	100%	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	16%	16%	16%	16%
2^3	100%	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	16%	16%	16%	16%	16%
2^4	100%	100%	100%	100%	94%	72%	50%	44%	21%	20%	16%	16%	16%	16%	16%	16%
2^5	100%	100%	100%	94%	72%	50%	44%	21%	20%	16%	16%	16%	16%	16%	16%	16%
2^6	100%	100%	94%	72%	50%	44%	21%	20%	16%	16%	16%	16%	16%	16%	16%	16%
2^7	100%	94%	72%	50%	44%	21%	20%	16%	16%	16%	16%	16%	16%	16%	16%	16%
2^8	94%	72%	50%	44%	21%	20%	13%	13%	13%	13%	13%	13%	13%	13%	13%	13%
2^9	72%	50%	44%	21%	20%	13%	13%	13%	13%	13%	13%	13%	13%	13%	13%	13%
2^{10}	52%	44%	21%	20%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%	9%
2^{11}	44%	21%	20%	9%	7%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%	6%
2^{12}	24%	20%	9%	7%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%	5%
2^{13}	20%	9%	7%	4%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%	3%
2^{14}	10%	7%	4%	3%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%	2%
2^{15}	8%	4%	3%	2%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%	1%

Table 6.4: Parallel efficiency of the hybrid attack for k nodes with l cores each for binary LWE with $n = 256$, $m = 512$, $q = 128$ assuming efficiency $f_{0.9}$ for BKZ and efficiency $f_{0.1}$ for the meet-in-the-middle phase.

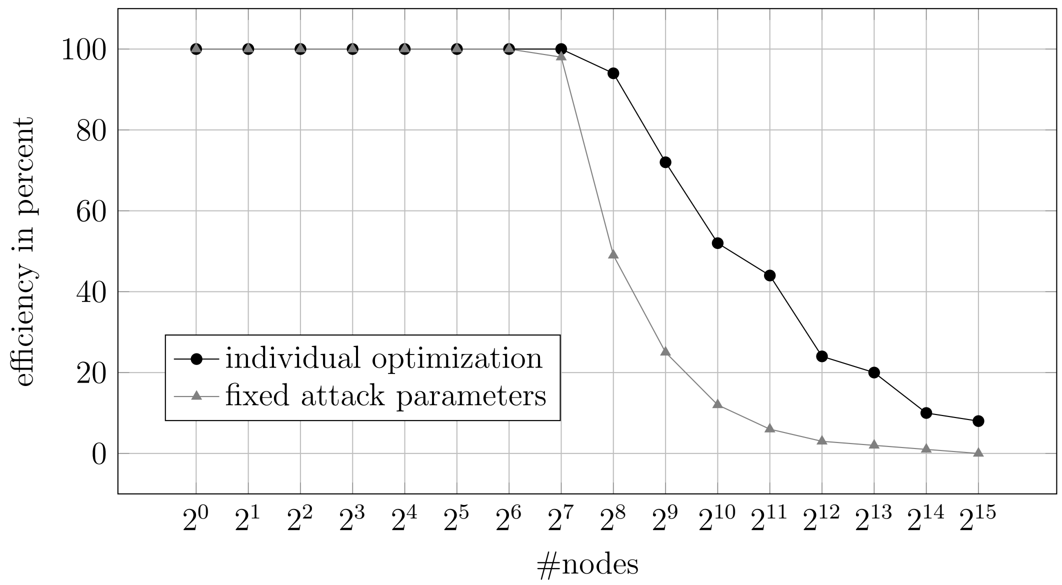

Figure 6.1: Comparing the efficiency of the parallel hybrid attack when optimizing the attack parameters for each configuration individually to using the optimal attack parameters of the serial hybrid attack for each configuration for binary LWE with $n = 256$, $m = 512$, $q = 128$ with varying number of nodes and one core per node.

6.3.1 Our Implementation

For our implementation for the experiments, we use different MPI processes for the running randomized instances of the attack in parallel and multiple threads to parallelize one run of the meet-in-the-middle phase. We employ the ZZ and RR data types provided by the NTL library for big integer and arbitrary floating point precision data types, e.g., to store the bases of the lattices. The lattice-related tasks, namely the Gram-Schmidt orthogonalization and the BKZ reduction are also performed within NTL and are not parallelized.
We implemented an iterative Nearest Plane algorithm since they in general perform better than recursive Nearest Plane algorithms. The loop within iterative Nearest Plane depends on previous iterations thus preventing (an obvious) parallel execution. Each loop contains two inner product calculations which are performed as basic operations in NTL on its own data types. The vector dimension is much smaller than ten thousand and parallelization of this calculation does not pay off. However, since the NTL functions and types are used as a black box, they may easily be replaced in the case of more efficient implementations. Our code is available online[14].

[14]`https://github.com/MiBu84/Hybrid-Attack-binary-LWE`

Parallel Meet-in-the-Middle Search

To parallelize the meet-in-the-middle phase on a shared-memory node, we employ the OpenMP standard. Each thread samples new random vectors independently of the others and executes its Nearest Plane calculations. The random sampling is based on pseudo-random integral number generation following the C++11 language standard and employs instances of std :: uniform_int_distribution which receive a uniform random number generator as argument. In our case, this generator is a Mersenne Twister pseudo-random generator of 32-bit numbers of type std :: mt19937. The required seed is generated by a weighed product of the actual OpenMP thread id, the number of threads, the id of the executing process, the size of the MPI communicator and prime numbers. The crucial point for a meet-in-the-middle phase with multiple threads is the parallel access of the threads to a shared hash map. To avoid inconsistencies in this map, we use the implementation of the concurrent hash map (revision 3.4) for the Intel TBB library. Hence, no manual synchronization is required from outside. As hash function for the concurrent map, we use the predefined standard specialization std :: hash <bitset> which is a very space efficient data structure. Its hash function calculation is also very performant. To determine the corresponding std :: bitset from a vector of NTL's ZZ type, we apply Definition 6.1.

Parallel Instances

To run multiple instances of the hybrid attack in parallel, we implemented a parallelization based on the MPI standard. Each MPI process reads the input data independently and executes the pre-computation which includes randomizing the basis through permutation and multiplication with a random unimodular matrix and lattice reduction with BKZ. To achieve random permutations, we use the std :: shuffle template function following the C++11 standard which randomly rearranges the elements in vectors. As an argument the std :: shuffle also receives a Mersenne Twister generator of type std :: mt19937. The BKZ reduction is executed with pruning activated where we set the pruning parameter of the corresponding NTL::BKZ_ function to 10. After preparing the basis in the described way, each process enters the OpenMP parallelized meet-in-the-middle phase as explained in Section 6.3.1. Each process periodically checks whether one of the other processes successfully finished through non-blocking communication. The periodicity of those checks is configured in such a way that the communication overhead is negligible compared to the calculations of each process. Hence, it may happen that the other processes still search for a solution for a short time, although one process already successfully finished the hybrid attack.

High Hybrid Flexibility

Our hybrid parallelization approach allows for a highly flexible execution of the hybrid attack, depending on the focus of the execution and the hardware available. The common case is a relatively low success probability of the attack which requires permutations of the LWE samples and randomizing the bases before BKZ reducing them.

Hence, a high number of processes is required in that case to amplify the success probability. Our hybrid implementation allows to run multiple MPI processes for the randomized instances on a single compute node while still being able to parallelize the meet-in-the-middle phase where each process spawns a group of threads. The low costs for the process management and the minimized communication overhead enable an efficient use of the computational resource.

6.3.2 Test Environment

Most tests are performed on our local high performance computer. The nodes employed are equipped with two Intel Haswell Xeon E5-2680v3 processors ($2\times12 = 24$ cores, no hyperthreading, max. 3.3 GHz), 30 MB of last-level cache and 64 GB RAM. Nodes are interconnected with Infiniband FDR-14. All nodes are allocated exclusively to avoid any interference from other calculations. Additionally, we employed an own compute node, called LARA, with two E5-2698 processors resulting in 32 physical cores operating at 2.3 GHz. We mention explicitly when LARA was employed for an experiment. CentOS 7 is the operating system in both cases. C++ code is compiled with the Intel C++ in version 18.0.0, C++11 language standard is chosen and the optimization level is set to full optimization (*ofast*). The Intel OpenMP implementation and OpenMPI in version 1.10.7 are employed. As libraries, we use NTL (10.5.0), GMP (6.1.2), boost (1.66.0) and Intel TBB (4.4).

6.3.3 Test Cases

For the experiments for the meet-in-the-middle phase, we created binary LWE instances that we know could be solved with our attack parameters. To that end, we created binary LWE instances, where the binary error vector is of the form $\mathbf{e} = (0, 1, 0, 1, \ldots)$ to ensure that the last components of the short vector always have the correct number of non-zero entries. Furthermore, we checked if the Nearest Plane call in Line 11 of Algorithm 7 for the correct vector $\mathbf{v}_g$ finds the first components of the short vector. In contrast, for the experiments on the general number of repetitions of the attack we used random instances and implemented a check if the solution can possibly be found in general, i.e., check if the Nearest Plane algorithm succeeds and if the number of non-zero entries in the last components is correct. In each case, we investigate the performance of the serial version and the effect of increasing the degree of parallelism.

6.3.4 Reducing the Runtime of the Meet-in-the-Middle Phase of the Attack

To evaluate the quality of our parallelization approach for the meet-in-the-middle phase, we define the measure of processed vectors per second $\#v/t$ within the phase. The number of vectors which are processed until success of the algorithm is logged and divided by the overall runtime of the meet-in-the-middle phase. We repeated our test case with $\beta = 24$ and $r = 20$ ten times while varying the number of threads between $1, 2, 4, 8, 16$ and 24. The binary LWE instance was parameterized by $n = 80$, $m = 160$, and $q = 521$. Figure 6.2 summarizes the results.

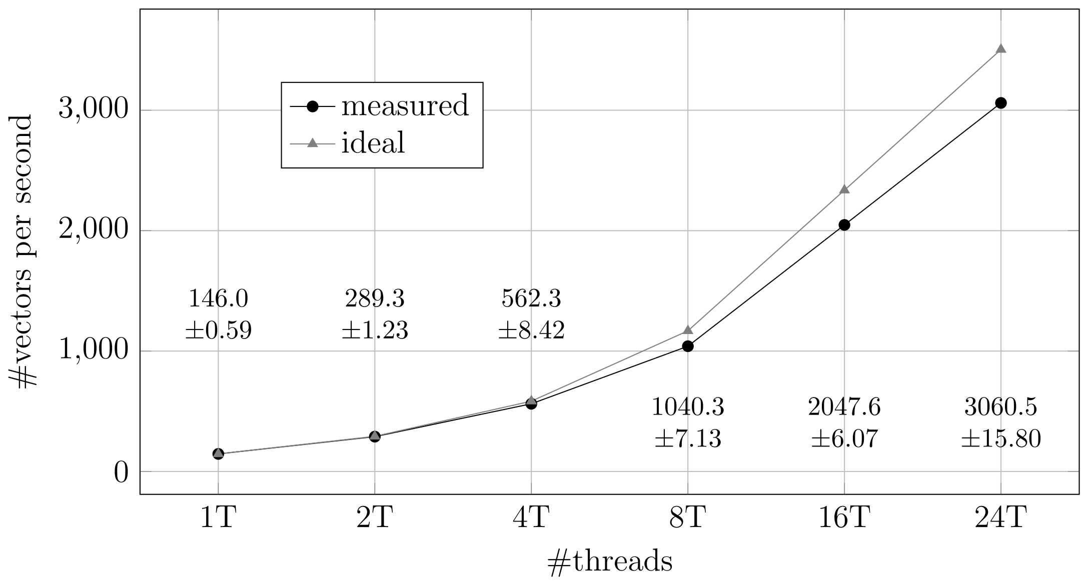

Figure 6.2: Scaling analysis of the meet-in-the-middle phase.

We show the average number of vectors processed per second for each number of threads as well as the standard deviations. The values are very stable and reproducible: For example, for one thread, the standard deviation is three orders of magnitude lower than the average value. Even in the worst case for four threads, the quotient of average value and standard deviation is higher than 66. We also see that the parallel scaling behavior is very good and nearly ideal up to four threads where efficiency values are above 96%. For a higher number of threads, we still achieve efficiency rates of more than 87% on our single node.

The second main point of our investigation of the meet-in-the-middle phase is the development of its overall runtime dependent on the number of OpenMP threads employed. To that end, we conducted three test suites differing in the values of r

and fixing $\beta = 24$, while keeping the LWE parameters from the test above. For each suite, we ran the meet-in-the-middle phase ten times and measured the time until success. The results for $r = 20$, $r = 24$ and $r = 28$ are shown in Figure 6.3.

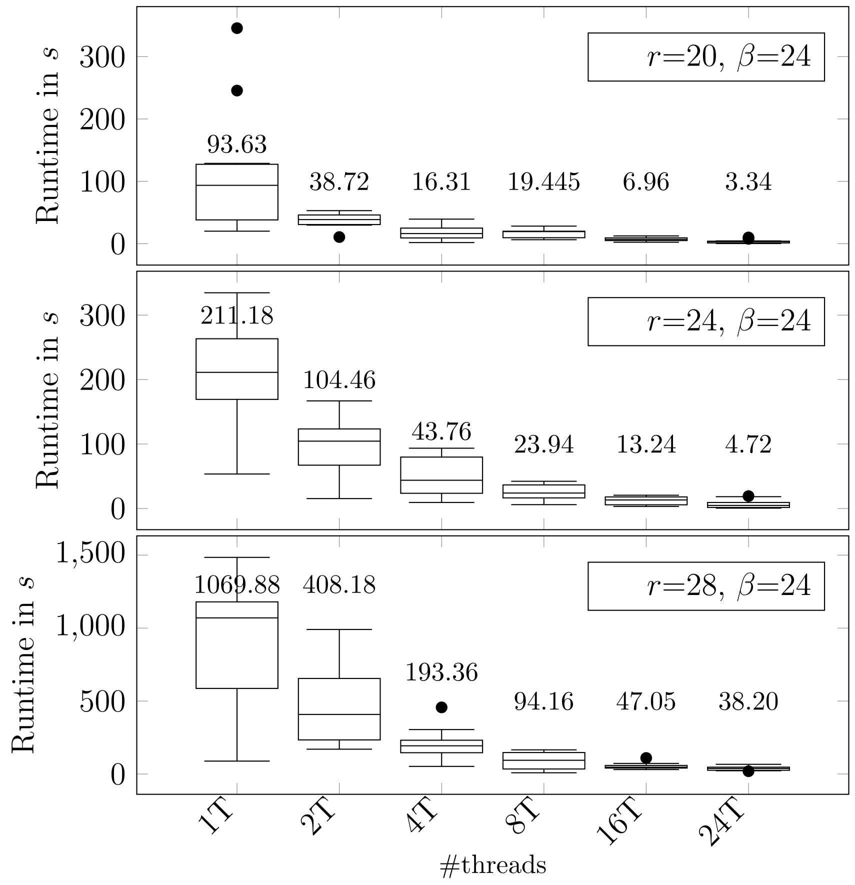

Figure 6.3: Runtime of meet-in-the-middle phase depending on the number of threads.

We employ box plots for the visualization of the data. The box represents the values between the 25- and 75-percentile, called Q_{25} and Q_{75}. This means that 50% of all measurement values lie in this range. Lines at bottom and top of the boxes represent the so called whiskers. In our case, we employ the definition of Tukey [Tuk77] meaning that the end of the whiskers indicated the lowest and highest measurement point, respectively, which lies within $1.5 \cdot (Q_{75} - Q_{25})$ of the lower and upper quartile, respectively. The median Q_{50} is shown by the horizontal lines within the boxes and its value is given in the diagram above the corresponding box. Outlying measurement points are drawn as filled circles.

First of all, we see that the serial runtime of the meet-in-the-middle phase increases when increasing the value of r, which is the expected behavior as the size of the search space increases. Second, in general, the runtime decreases when increasing the number of threads employed. We see that the speedup for the median time is even higher than the speedup in the number of vectors processed. For example, in the top case and the lower case with $r = 20$ and $r = 28$, respectively, of Figure 6.3, the median decreases by a factor of 28 when employing 24 threads compared to one thread while the factor is even 44 for $r = 24, \beta = 24$. This also results from the more dense distribution of the measurement values for a higher number of threads. For one thread, a wide range of possible runtimes is covered, while the region is small for 24 threads in all three cases. There are also no extreme outliers from eight threads on. Hence, increasing the number of threads also stabilizes the runtime of the attack phase.

6.3.5 Reducing the Overall Runtime of the Attack

In this section, we experimentally verify how using more processes to run multiple instances of the attack in parallel decreases the total runtime using our C++ implementation and its MPI parallelization. To that end, we spawn a varying number of MPI processes and each process randomizes and reduces the basis until one process finds a good basis, i.e., one for which the meet-in-the-middle phase can succeed. In this case, the attack will be successful. As the runtime of the meet-in-the-middle phase is analyzed in Section 6.3.4, we only check if a good basis is found and do not actually run the meet-in-the-middle search. We take the lowest number of randomization attempts required, where one attempt means randomizing and reducing one basis for each process in parallel. The binary LWE parameters are $n = 50$, $m = 100$, and $q = 67$, while $r = 4$ and $\beta = 3$. This test was repeated 20 times for a fixed number of MPI processes and the same input was used in all cases. The results are summarized in Figure 6.4. We again employ box plots with the same properties as given in Section 6.3.4. Figure 6.4 shows that the number of attempts required

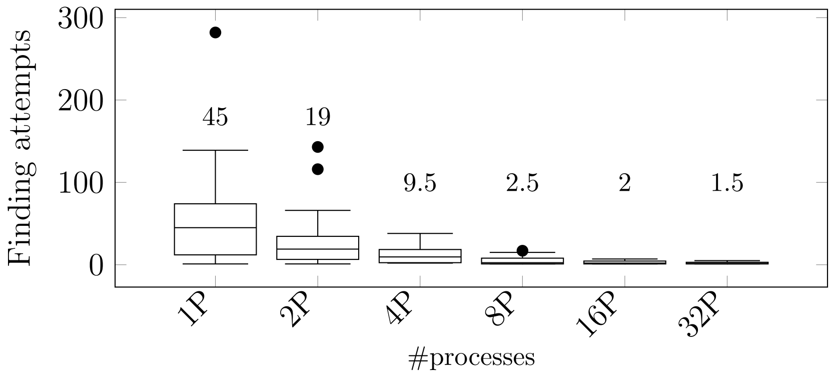

Figure 6.4: Number of attempts required to find a good basis when increasing the number of MPI processes.

decreases significantly when employing more processes. While for one thread up to

286 randomization attempts are performed, the maximum number is only 5 in the case of 32 processes. The median decreases from 45 to 1.5.

6.3.6 Analysis of the Hybrid Efficiency

The efficiency and flexibility of our hybrid implementation was investigated with experiments on LARA. The number of processes ($\#PROC$) and threads ($\#THR$) was varied in such a way that the product is 32. We call these 32 units of execution *workers* in the following. We ran our experiments in different configurations on a single node for binary LWE instances with $m = 160$, $n = 80$, and $q = 521$. All tests were repeated ten times. Table 6.5 gives an overview of the results.

In *Test 1*, we use $\beta = 20$ and $r = 20$ and let all processes enter the meet-in-the-middle phase at the same time on precomputed bases. The test stops when one thread finds a solution. We log the number of vectors each process processes during the runtime and calculate the average number of vectors processed per second on the whole machine by all workers. This number is shown in the third column including its standard deviation. We see that the number of vectors processed per second is virtually independent of the configuration. This also shows that sharing the resources on a single compute node is done efficiently and that our implementation works well with multiple processes on one node. The fourth column gives the average runtime t_{guess} (over the processes) in seconds.

		Test 1	
$\#PROC$	$\#THR$	$\#v$/s all workers	average t_{guess}
1	32	1475 ± 7.26	7.92 ± 7.26
2	16	1497 ± 25.22	13.13 ± 10.01
4	8	1494 ± 34.04	21.05 ± 11.09
8	4	1490 ± 53.98	24.22 ± 10.83
		Test 2	
$\#PROC$	$\#THR$	$\#v$/s succ. thread	average t_{BKZ}
2	16	1557 ± 3.14	184 ± 2.23
4	8	716 ± 33.65	188 ± 4.66
8	4	297 ± 46.03	199 ± 5.51

Table 6.5: Two experiments on parallel configurability. In Test 1 all workers perform guessing, in Test 2 half of the workers run BKZ the others guess. Runtimes in seconds.

In *Test 2* with $\beta = 24$ and $r = 28$ half of the workers run BKZ while the rest directly enters the guessing phase. The third column shows the vectors processed per second by the succeeding thread. Ideally, we would expect that this number is halved from row to row. From the second to the third row, the speed differs by a factor of 2.2, while the factor is 2.4 between row three to four. The increasing number of BKZ instances has a negative influence on the threads in the meet-in-the-middle phase, indicating that the memory interface of the system is the bottleneck in this case. Replacing NTL's BKZ implementation by a more memory efficient one will reduce this effect. The fourth column shows the average runtime of the BKZ calls t_{BKZ} which becomes somewhat slower when increasing the number of simultaneous runs.

The experiments on LARA demonstrate that our implementation is well prepared for various high performance computing setups since the increasing number of CPU cores in future compute nodes can be used to increase the number of randomized instances that are run in parallel as well as to increase the degree of parallelism per instance within the meet-in-the-middle phase (and possibly also within BKZ).

7 The Hybrid Lattice Reduction and Quantum Search Attack

While the hybrid attack (cf. Chapter 5) is currently considered the most practical attack on several instances of lattice problems, it has four main drawbacks. First, it is only practical for lattice problems with highly structured secret vectors such as LWE with binary or ternary error distribution. Second, the memory requirements of the meet-in-the-middle search are enormous. Third, the probability that collisions are detected during the meet-in-the-middle-phase can be extremely small, see Chapter 5. And finally, it does not take the scenario into account, where the attacker has access to a large-scale quantum computer. The natural question is therefore whether the hybrid attack can be improved such that all of the above drawbacks are eliminated.

Contribution. In this chapter, we present an improved quantum version of the hybrid attack which eliminates all these drawbacks of the classical hybrid attack and provide a detailed analysis of the attack. Our quantum hybrid attack replaces the meet-in-the-middle phase of the hybrid attack with a generalization of Grover's quantum search algorithm [Gro96] by Brassard et al. [BHMT02]. This quantum search is sensitive to the underlying distribution on the search space, which makes it more efficient than Grover's algorithm if the distribution from which the shortest non-zero vector is drawn is non-uniform (e.g., in the case of LWE with a discrete Gaussian error distribution). In addition, our quantum hybrid attack eliminates the huge memory cost and low collision finding probability caused by the meet-in-the-middle search of the classical hybrid attack. Our runtime analysis of the quantum hybrid attack includes optimizing the quantum search algorithm and the search space. Finally, we apply our quantum attack to various uSVP instances with small and/or sparse short vectors as well as to instances with short vectors that follow discrete Gaussian distributions. We compare our results to the classical hybrid attack and the primal attack under the 2016 estimate (cf. Chapter 3), highlighting the improvements of the quantum hybrid attack.

Organization. In Section 7.1, we present our new quantum hybrid attack. The runtime analysis of the attack is provided in Section 7.2. In Section 7.3, we show

how to further optimize the search space for the attack. Finally, in Section 7.4, we apply our quantum attack to several uSVP instances.

Publications. This chapter is based on the publication [3], which was presented at PQCrypto 2017. In addition, the concept of optimizing the search space of the quantum hybrid attack and the systematic runtime estimates for various discrete Gaussian and binary or ternary distributions are either part of [7] or novel in this thesis.

7.1 The Quantum Hybrid Attack

In this section, we introduce our new quantum hybrid attack. The main idea is to use quantum search algorithms to speed up the guessing part of the classical hybrid attack. The idea to replace the meet-in-the-middle phase by Grover's search algorithm was sketched in Schanck's thesis [Sch15]. However, an analysis of the runtime of such an attack is still missing in the literature. Furthermore, by using a modification of Grover's algorithm, our quantum hybrid attack is more efficient if the searched vector is not drawn from a uniform distribution (e.g., in the case of solving LWE with a discrete Gaussian error distribution).

This section is structured as follows. We give a brief summary of Grover's quantum search algorithm [Gro96] and its modified version developed by Brassard et al. [BHMT02] in Section 7.1.1. In Section 7.1.2, we show how to use this quantum search algorithm inside the hybrid attack to obtain a new quantum hybrid attack.

7.1.1 Amplitude Amplification

In 1996, Grover presented a quantum algorithm that can speed up the search in unstructured databases [Gro96]. Given a function $f : S \to \{0,1\}$ defined on a finite set S, we call $S_f := \{x \in S \mid f(x) = 1\}$ the set of marked elements. Grover's algorithm allows to find an element $x \in S_f$ in approximately $\frac{\pi}{4} \cdot \sqrt{|S| \,/\, |S_f|}$ evaluations of f (without any further knowledge about f), while classical algorithms require an average number of evaluations in the order of $|S| \,/\, |S_f|$.

The runtime of Grover's search algorithm is independent of how the marked elements have been chosen. The drawback is that additional information about the choice of the marked elements is not used. A generalization of Grover's search algorithm that can utilize the probability distribution on the search space was presented by Brassard et al. [BHMT02]. Their generalization uses an additional algorithm $\mathcal{A}$ sampling from some distribution on the search space S.

Theorem 7.1 ([BHMT02], Theorem 3)**.** *There exists a quantum algorithm* QSearch *with the following property. Let $\mathcal{A}$ be any quantum algorithm that uses no measurements (i.e., a unitary transformation), and let $f : S \to \{0,1\}$ be any Boolean function.*

Let a denote the initial success probability of $\mathcal{A}$ (i.e., $a = \Pr[f(x) = 1, x \xleftarrow{\$} \mathcal{A}]$). The algorithm QSearch *finds a good solution using an expected number of applications of $\mathcal{A}$, $\mathcal{A}^{-1}$ and f which is in $\Theta(1/\sqrt{a})$ if $a > 0$, and otherwise runs forever.*

The quantum algorithm $\mathcal{A}$ can be constructed as follows: Given an arbitrary (efficient) probabilistic sampling algorithm, it can be transformed into a deterministic algorithm that gets random bits as input. This algorithm in turn can be transformed into a quantum algorithm. Instantiating this quantum algorithm with the uniform distribution as superposition for the input bits leads to the wanted algorithm $\mathcal{A}$.

Note that the complexity of the algorithm QSearch is only given asymptotically. This is only necessary because the probability a is unknown. However, it can be shown that the hidden constant is indeed small, and hence we can ignore the Landau notation in our runtime estimates.

7.1.2 The Attack

In the following, we describe our new quantum hybrid attack (Algorithm 9). As always, we use the notation $\mathrm{NP}_{\mathbf{B}}(\mathbf{t})$ to indicate that Nearest Plane is called on the target vector $\mathbf{t}$ and input basis $\mathbf{B}$. The inputs for the quantum hybrid attack are a basis $\mathbf{B}' \in \mathbb{R}^{m \times m}$ of a uSVP lattice Λ of the form

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{0} & \mathbf{I}_r \end{pmatrix},$$

the distribution D_e on $\mathbb{Z}^m$ from which the shortest non-zero vector in Λ is drawn, an upper bound y on the norm of the shortest non-zero vector, and the attack parameters r and β. Similar to the classical hybrid attack (cf. Chapter 5), we use the idea that if $\mathbf{v} = (\mathbf{v}_\ell, \mathbf{v}_g) \in \Lambda$ with $\mathbf{v}_g \in \mathbb{R}^r$ is a shortest non-zero vector in Λ and $\mathbf{B}$ is sufficiently well reduced, we can guess $\mathbf{v}_g$ and hope to find $\mathbf{v}_\ell$ via $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$, since $\mathbf{C}\mathbf{v}_g = -\mathbf{B}\mathbf{x} + \mathbf{v}_l$. Now, the attack proceeds as follows. After choosing a suitable distribution for the sampling algorithm $\mathcal{A}$ used in the quantum search algorithm, the attack reduces the upper-left block $\mathbf{B}$ of the basis matrix $\mathbf{B}'$. It then runs *QSearch* with the function defined by Algorithm 8, which essentially checks if a guess $\mathbf{w}_g$ is correct by checking if $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{w}_g) = \mathbf{v}_l$.

As we show in Section 7.2, in general it is not optimal to use the distribution D_e for the sampling algorithm $\mathcal{A}$ to find the solution. Instead we use the following transformed distribution.

Definition 7.1. *Let X be an arbitrary distribution with finite support S. We write $T(X)$ for the distribution defined by*

$$\forall a \in S : \Pr[a = b | b \xleftarrow{\$} T(X)] = \frac{x_a^{\frac{2}{3}}}{\sum_{c \in S} x_c^{\frac{2}{3}}}.$$

Our quantum hybrid attack is presented in Algorithm 9. Recall that the attack parameter r indicates the guessing dimension and the parameter β is the block size used for lattice reduction algorithms.

Algorithm 8: Function $f_{\mathbf{B},\mathbf{C},y}(\mathbf{w}_g)$

```
1 $\mathbf{w}_\ell \leftarrow \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{w}_g)$;
2 Set $\mathbf{w} = (\mathbf{w}_\ell, \mathbf{w}_g)$;
3 if $\|\mathbf{w}\| \leq y$ then
4     return 1;
5 else
6     return 0;
```

Algorithm 9: Quantum hybrid attack

Input: A basis $\mathbf{B}' \in \mathbb{R}^{m \times m}$ of a uSVP lattice Λ of the form $\mathbf{B}' = \begin{pmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{0} & \mathbf{I}_r \end{pmatrix}$, a distribution D_e on $\mathbb{Z}^m$ from which the shortest non-zero vector in Λ is drawn, a bound y, the attack parameters $r, \beta \in \mathbb{N}$

```
1 Let $D$ be the distribution of the last $r$ entries of a vector $\mathbf{x}$, where $\mathbf{x} \xleftarrow{\$} D_e$;
2 Set $\mathcal{A}$ to be a quantum (sampling) algorithm without measuring for the
    distribution $T(D)$ as defined in Definition 7.1;
3 BKZ-$\beta$ reduce $\mathbf{B}$;
4 Let $\mathbf{v}'_g$ be the result of QSearch (Theorem 7.1) with function $f_{\mathbf{B},\mathbf{C},y}$
    (Algorithm 8) and quantum algorithm $\mathcal{A}$;
5 return $(\mathrm{NP}_{\mathbf{B}}(\mathbf{v}'_g), \mathbf{v}'_g)$;
```

7.2 Analysis

In this section, we analyze the expected runtime of the quantum hybrid attack and show how to minimize it over all choices of attack parameters.

7.2.1 Success Probability and Number of Function Applications

In the following, we show our main result about the runtime of our quantum hybrid attack.

Heuristic 7.1. *Let Λ, the matrices $\mathbf{B}, \mathbf{C}$, the distribution D, the algorithm $\mathcal{A}$, and the parameters m, y, r be defined as in Section 7.1. Let $\mathbf{v} = (\mathbf{v}_\ell, \mathbf{v}_g) \in \Lambda$ with $\mathbf{v}_g \in \mathbb{R}^r$ be a shortest non-zero vector and assume $\|\mathbf{v}\| \leq y$.*

The success probability p of the quantum hybrid attack is approximately

$$p \approx \prod_{i=1}^{m-r} \left(1 - \frac{2}{B(\frac{(m-r)-1}{2}, \frac{1}{2})} \int_{-1}^{\max(-r_i,-1)} (1-t^2)^{\frac{(m-r)-3}{2}} dt \right),$$

where $B(\cdot,\cdot)$ denotes the Euler beta function (see [Olv10]),

$$r_i = \frac{\|\mathbf{b}_i^*\|}{2\|\mathbf{v}_l\|} \quad \text{for all } i \in \{1,\dots,m-r\},$$

and $\|\mathbf{b}_1^*\|,\dots,\|\mathbf{b}_{m-r}^*\|$ denote the lengths of the Gram-Schmidt basis vectors corresponding to the basis $\mathbf{B}$.

In case of success, the expected number of applications of $f_{\mathbf{B},\mathbf{C},y}$, $\mathcal{A}$, and $\mathcal{A}^{-1}$ in Algorithm 9 is $\Theta(L)$, where

$$L = \left(\sum_{x \in \mathsf{supp}(D)} d_x^{\frac{2}{3}} \right)^{\frac{3}{2}}.$$

Furthermore, the choice of the distribution for the sampling algorithm $\mathcal{A}$ in Algorithm 7.1 is optimal.

We first determine the success probability of the attack. We then calculate and optimize the number of applications of f, $\mathcal{A}$, and $\mathcal{A}^{-1}$ and compare our results with Grover's search algorithm. In the following, let all notations be as in Heuristic 7.1 and assume that its requirements hold.

Success Probability

If $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_\ell$, we have $f_{\mathbf{B},\mathbf{C},y}(\mathbf{v}_g) = 1$ with overwhelming probability and *QSearch* recovers $\mathbf{v}_g$. Using the approximation of the probability that $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_\ell$ determined in Chapter 5 yields the success probability given in Heuristic 7.1.

Number of Applications of $f_{\mathbf{B},\mathbf{C},y}$, $\mathcal{A}$, and $\mathcal{A}^{-1}$

We now calculate the expected number of applications of $f_{\mathbf{B},\mathbf{C},y}$, $\mathcal{A}$ and $\mathcal{A}^{-1}$ (simply called loops in the following) in the quantum hybrid attack in the case the attack is successful. We show how the choice of the sampling algorithm $\mathcal{A}$ influences the number of loops, how to minimize this number over all possible choices of $\mathcal{A}$, and that our choice in Algorithm 9 is in fact optimal. In the following, let $S = \mathsf{supp}(D)$ be a finite set. The support S is the search space of our quantum algorithm. Let $\mathcal{A}$ be the initial sampling algorithm used in the quantum hybrid attack and A be the distribution with support S corresponding to $\mathcal{A}$. According to Theorem 7.1, for a fixed target element $x \in S$ the expected number of loops in the quantum hybrid attack is roughly $(\sqrt{a_x})^{-1}$. However, since the marked element (and its probability) is not known, we can only estimate the expected number of loops

$$L(A) = L\left((a_x)_{x\in S}\right) = \sum_{x\in S} \frac{d_x}{\sqrt{a_x}}. \tag{7.1}$$

In order to minimize the runtime of the quantum search we must determine the optimal distribution A that minimizes the number of loops $L(A)$. We emphasize that minimizing the number of loops is of independent interest for any quantum search algorithm based on [BHMT02] applied in a similar way as in our attack.

Minimal number of loops. We first minimize the expected number of loops over all possible choices of A. Without loss of generality we assume $S = \{1, \dots, k\}$ for some $k \in \mathbb{N}$. We minimize the expected number of loops by minimizing the function

$$L : (0,1)^k \to \mathbb{R}, \quad (a_1, \dots, a_k) \mapsto \sum_{i=1}^{k} \frac{d_i}{\sqrt{a_i}}, \tag{7.2}$$

in k variables $a_1, \dots, a_k \in (0,1)$ under the constraint

$$a_1 + \ldots + a_k = 1, \tag{7.3}$$

where $d_1, \dots, d_k \in (0,1)$ are fixed. In order to minimize L under the constraints, we define the Lagrange function corresponding to L and Equation (7.3)

$$\mathcal{L}(\lambda, a_1, \dots, a_k) = \left(\sum_{i=1}^{k} \frac{d_i}{\sqrt{a_i}}\right) + \lambda\left(-1 + \sum_{i=1}^{k} a_i\right). \tag{7.4}$$

To find the minimum of L we need to solve the following set of $k+1$ equations

$$\begin{aligned} &[\mathsf{E}_i]_{i\in\{1,\dots,k\}} && 0 = \mathcal{L}_{a_i}(\lambda, a_1, \dots, a_k) = -\frac{d_i}{2} a_i^{-\frac{3}{2}} + \lambda \\ &[\mathsf{E_c}] && a_1 + \ldots + a_k = 1, \end{aligned}$$

which gives

$$a_i = \frac{d_i^{\frac{2}{3}}}{\sum_{j=1}^{k} d_j^{\frac{2}{3}}} \quad \text{and} \quad \lambda = \frac{\left(\sum_{j=1}^{k} d_j^{\frac{2}{3}}\right)^{\frac{3}{2}}}{2}. \tag{7.5}$$

It remains to be shown that choosing the a_i according to Equation (7.5) leads in fact to a local *minimum* of L under the given constraints. If this is the case, this local minimum must indeed constitute the global minimum satisfying the constraints, since it is the only local minimum and L tends to infinity as one of the a_i approaches zero (hence the problem can be restricted to a compact domain). In order to show that the a_i constitute a local minimum, we compute the determinants of the leading principal minors of the bordered Hessian matrix evaluated in the a_i

$$H = \begin{pmatrix} 0 & 1 & 1 & \dots & 1 \\ 1 & x_1 & 0 & \dots & 0 \\ 1 & 0 & x_2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1 & 0 & \dots & 0 & x_k \end{pmatrix}, \quad \text{where } x_i = \frac{3d_i}{4a_i^{2.5}} > 0.$$

For $j \in \{1, \dots, k\}$ let

$$H_j = \begin{pmatrix} 0 & 1 & 1 & \dots & 1 \\ 1 & x_1 & 0 & \dots & 0 \\ 1 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 1 & 0 & \dots & 0 & x_j \end{pmatrix}$$

be the leading principal minors. As adding scalar multiples of columns to other columns does not change the determinant, we can use Gaussian elimination to see that the determinants of all but the first principal minors of H are given by

$$\det(H_j) = \det \begin{pmatrix} x_0 & 1 & 1 & \dots & 1 \\ 0 & x_1 & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ 0 & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & x_j \end{pmatrix} \quad \text{where } x_0 = -\left(\sum_{i=0}^{j} \frac{1}{x_i}\right) < 0.$$

Hence all determinants of the leading principal minors of H (except the first one) are negative and thus choosing the a_i according to Equation (7.5) leads in fact to a local minimum of L under the given constraints. Inserting these a_i into Equation (7.2) yields the minimal number of loops

$$L_{\mathsf{min}} = \sum_{i=1}^{k} \frac{d_i}{\sqrt{a_i}} = \sum_{i=1}^{k} \frac{d_i}{\sqrt{\frac{d_i^{\frac{2}{3}}}{\sum_{j=1}^{k} d_j^{\frac{2}{3}}}}} = \left(\sum_{j=1}^{k} d_j^{\frac{2}{3}}\right)^{\frac{1}{2}} \cdot \sum_{j=1}^{k} d_j^{\frac{2}{3}} = \left(\sum_{x \in S} d_x^{\frac{2}{3}}\right)^{\frac{3}{2}}. \qquad (7.6)$$

An important special case. While Equation (7.6) provides a simple formula for the minimal number of loops, evaluating it might be a computationally expensive task for a large support S. In the following we consider the case that the support is of the form $S = S_0^r$ for some $r \in \mathbb{N}$ and smaller set S_0 and that $D = P^r$ for some distribution P on S_0. Note that this is for instance the case for LWE if the components of the error vector are drawn independently from the same distribution. We show how in this case Equation (7.6) can be evaluated by computing a sum of $|S_0|$ summands and raising it to the r-th power instead of computing a sum of $|S_0|^r$ summands. This is true since Equation (7.6) can be rewritten and simplified to

$$\begin{aligned} L_{\mathsf{min}} = \left(\sum_{x\in S} d_x^{\frac{2}{3}}\right)^{\frac{3}{2}} &= \left(\sum_{y_1\in S_0}\cdots\sum_{y_{r-1}\in S_0}\sum_{y_r\in S_0}\prod_{i=1}^{r} p_{y_i}^{\frac{2}{3}}\right)^{\frac{3}{2}} = \\ &= \left(\sum_{y_1\in S_0}\cdots\sum_{y_{r-1}\in S_0}\prod_{i=1}^{r-1} p_{y_i}^{\frac{2}{3}}\left(\sum_{y_r\in S_0} p_{y_r}^{\frac{2}{3}}\right)\right)^{\frac{3}{2}} = \\ &= \left(\sum_{y_1\in S_0}\cdots\sum_{y_{r-1}\in S_0}\prod_{i=1}^{r-1} p_{y_i}^{\frac{2}{3}}\left(\sum_{y\in S_0} p_{y}^{\frac{2}{3}}\right)\right)^{\frac{3}{2}} = \\ &= \ldots = \left(\left(\sum_{y\in S_0} p_y^{\frac{2}{3}}\right)^r\right)^{\frac{3}{2}}, \end{aligned} \tag{7.7}$$

since each of the d_x is exactly the product of r of the p_y.

Comparison with Grover's search algorithm. If in our quantum hybrid attack the distribution D is the uniform distribution, then its number of loops matches the one of Grover's search algorithm

$$L_{\mathsf{min}} = \left(\sum_{x\in S} d_x^{\frac{2}{3}}\right)^{\frac{3}{2}} = \left(\sum_{x\in S}\left(\frac{1}{|S|}\right)^{\frac{2}{3}}\right)^{\frac{3}{2}} = \left(|S|\,\frac{1}{|S|^{\frac{2}{3}}}\right)^{\frac{3}{2}} = \sqrt{|S|}.$$

For a structured search space, however, QSearch (see Theorem 7.1) may give a significantly smaller number of loops. As an example we examine the distribution D on the set $S = \{-16, \ldots, 16\}^r$ used in the New Hope [ADPS16] key exchange scheme. Then $|S| = 33^r$ and using Grover's search algorithm inside the quantum hybrid attack would yield an expected number of loops of

$$L_{\mathsf{grover}} = \sqrt{33^r} \approx 2^{2.52r}.$$

In comparison, our quantum hybrid attack only requires

$$L_{\mathsf{our}} = \left(\left(\sum_{i=0}^{32} p_i^{\frac{2}{3}}\right)^r\right)^{\frac{3}{2}} \approx 2^{1.85r}, \quad \text{where } p_i = \binom{32}{i}\cdot 2^{-32}.$$

For $r = 200$ entries that are guessed during the quantum hybrid attack this amounts to a speedup factor of 2^{134} of our approach over using Grover's algorithm inside the hybrid attack. This example showcases the significant improvement of our quantum hybrid attack over one that is simply using Grover's search algorithm. It also demonstrates that our new quantum hybrid attack opens the possibility to apply the hybrid attack to larger, non-uniform search spaces.

7.2.2 Total Runtime of the Quantum Hybrid Attack

In this section we estimate the total runtime of the quantum hybrid attack by estimating the individual cost of one application of $f_{\mathbf{B},\mathbf{C},y}$, $\mathcal{A}$, and $\mathcal{A}^{-1}$, the precomputation (i.e., lattice reduction) cost, and combining the results with the ones of Section 7.2.1. The resulting runtime formula must then be optimized over all possible attack parameters.

Cost of $f_{\mathbf{B},\mathbf{C},y}$, $\mathcal{A}$, and $\mathcal{A}^{-1}$. The cost of the function $f_{\mathbf{B},\mathbf{C},y}$ is dominated by the cost of one Nearest Plane call, which was experimentally found to be roughly $k^2/2^{1.06}$ bit operations [HHHGW09], where k is the dimension of the lattice (in our case $k = m - r$), see Section 2.4.5. We assume that compared to this cost, the cost of the algorithm $\mathcal{A}$ and $\mathcal{A}^{-1}$ can be neglected.

Total Cost and Runtime Optimization. Consequently, the total runtime of the quantum hybrid attack can be estimated by

$$T_{\mathsf{total}} = \frac{T_{\mathsf{red}} + T_{\mathsf{hyb}}}{p},$$

where

$$T_{\mathsf{hyb}} = \left(\sum_{x \in S} d_x^{\frac{2}{3}} \right)^{\frac{3}{2}} \cdot (d-r)^2/2^{1.06},$$

T_{red} is the runtime of lattice reduction, and p is the success probability as given in Heuristic 7.1. The total runtime of the attack T_{total} depends on the attack parameters, i.e., the guessing dimension r and the applied block size β, and must therefore be optimized over all such choices as in Section 5.3.3.

7.2.3 Further Techniques

When embedding LWE or NTRU problems into uSVP, the (quantum) hybrid attack can be combined with further (known) techniques.

Choosing the lattice dimension. One of the simplest techniques is to choose a number of LWE samples that optimizes the attack. In the NTRU setting, this corresponds to the dimension reducing techniques described in [MS01], which allow to choose the lattice dimension between n and $2n$, where n is the degree of the polynomial defining the NTRU ring.

Rescaling parts of the lattice. If the LWE secret vector is uniquely small or sparse, rescaling techniques can be applied to balance the size of the LWE secret and the error vectors when using Bai and Galbraith's embedding [BG14b], see Section 3.3.1 for more details. In this case, we swap the positions of the secret and error vector in order to guess parts of the smaller or sparser secret in the hybrid attack.

Centering LWE error vectors. If the LWE error distribution is not centered around zero, shifting the center of the distribution to zero by subtracting a constant vector from the parts of the LWE equation which are not guessed can lead to a more efficient attack by reducing the norm of the error vector. This is illustrated for LWE with binary error in Section 5.4.3.

Considering rotations of the short vector. As accounted for in Chapter 5, it is possible that the uSVP lattice contains more than one uniquely short vector. In fact, this case can be seen as a variant of uSVP, which occurs for instance when embedding the NTRU problem into uSVP, as also rotations of the short vector are contained in the lattice. This can be taken into consideration by amplifying the success probability p_{succ} of one vector to $1-(1-p_{\mathsf{succ}})^k$, where k is the number of rotations to be considered (cf. Section 5.4.2). This assumes that each of the rotations has the same success probability and that they are independent.

7.3 Optimizing the Search Space

In the classical hybrid attack, one typically assumes that the last r entries of the short vector(s) have a fixed number of non-zero entries, i.e., Hamming weight, h_r. Consequently, one only guesses vectors of that weight and accounts for that restriction in the success probability of the attack. In the quantum hybrid attack as detailed above one instead guesses all possible vectors. However, both approaches may not be optimal as they are located at the opposite sides of the trade-off between success probability and number of vectors that need to be guessed. Instead, we propose to use the following approach for the quantum hybrid attack. Let the lattice dimension m and the guessing dimension r be fixed. Let χ be the distribution of the short vector and χ_r be the distribution of its last r components. Let M be the maximal possible guessing set for the last r components, i.e., the support of χ_r (e.g., for random binary or random ternary vectors this would be $\{0,1\}^r$ or $\{-1,0,1\}^r$ respectively). Further let $S \subset M$ denote the actual guessing set used in the attack. Let p_S denote the probability that $\mathbf{v}_g \in S$ if $\mathbf{v}_g \overset{\$}{\leftarrow} \chi_r$ and for $\mathbf{x} \in M$ let $q_{\mathbf{x}}$ denote the probability of $\mathbf{x}$ according to χ_r. Then it can be assumed that the runtime of the quantum hybrid attack is roughly

$$T_{\mathsf{total}} \approx \frac{T_{\mathsf{red}} + T_{\mathsf{qsearch}}}{p_{\mathsf{succ}}} \approx \frac{T_{\mathsf{red}} + \left(\sum_{\mathbf{x}\in S} \left(\frac{q_{\mathbf{x}}}{p_S}\right)^{2/3}\right)^{3/2} T_{\mathrm{NP}}}{(1-(1-p_{\mathrm{NP}}\cdot p_S)^k)}, \tag{7.8}$$

where k is the number of rotations of the short vector that can be found, p_{succ} is the overall success probability and p_{NP} is the estimated success probability of Nearest Plane (cf. Chapter 5). T_{total} can then be minimized over all possible choices of the guessing set S. In the following, we elaborate on how to optimize S. First, it is reasonable to construct S as a subset of M containing the most likely elements of M, i.e., no guess in $M \setminus S$ should have a higher probability of being a correct guess than some guess in S. If one respects this condition on S, one only has to optimize its size. In the following, we explain how to construct such sets S if

(i) χ is the uniform distribution on $\{0,1\}^m$,

(ii) χ is the uniform distribution on $\{-1,0,1\}^m$,

(iii) χ is the uniform distribution on the set of all vectors in $\{0,1\}^m$ with fixed Hamming weight h, or

(iv) χ is the uniform distribution on the set of all vectors in $\{-1,0,1\}^m$ with fixed Hamming weight h.

In cases (i) and (ii), every guess in M has the same probability of being correct. Hence we can pick any elements to construct S and only need to minimize (7.8), which in this case it equivalent to

$$\frac{T_{\mathsf{red}} + \sqrt{|S|}T_{\mathrm{NP}}}{\left(1-\left(1-p_{\mathrm{NP}}\cdot\frac{|S|}{|M|}\right)^k\right)},$$

over all possible choices of the size of S with $1 \le |S| \le |M|$. This can be done for instance by a binary search. Note that in the uniform case and if $k = 1$, the optimal choice is always to choose $S = M$.

We now consider the case (iii). For $\max(0, h-(m-r)) \le i \le \min(r,h)$ let S_i denote the set of all vectors $\{0,1\}^r$ with hamming weight i. Note that for each such i, every element $\mathbf{x} \in S_i$ has the same probability

$$q_{\mathbf{x}} = q_i := \frac{\binom{m-r}{h-i}}{\binom{m}{h}}$$

of being a correct guess. Let $i_0, \dots i_{\min(r,h)-\max(0,h-(m-r))}$ be ordered such that $q_{i_0} \geq \dots \geq q_{i_{\min(r,h)-\max(0,h-(m-r))}}$. Then we may construct S as a union $S = S_{i_0} \cup \dots \cup S_{i_{k-1}} \cup S'_{i_k}$ for some $k \in \mathbb{N}_0$, where S'_{i_k} is some subset of S_{i_k}. One then minimizes (7.8) over the choice of k and the size of the subset S'_{i_k} of S_{i_k}. A valid ordering of the i_j is for example given by choosing i_j such that $h - i_j$ is a closest integer in $\mathbb{N}_0 \setminus \{i_0, \dots, i_{j-1}\}$ to $\left\lfloor \frac{m-r}{2} \right\rfloor$.

Finally, we consider the case (iv), which is similar to case (iii). For $\max(0, h - (m-r)) \leq i \leq \min(r, h)$ let S_i denote the set of all vectors $\{-1, 0, 1\}^r$ with hamming weight i. Then for each such i, every element $\mathbf{x} \in S_i$ has the same probability

$$q_{\mathbf{x}} = q_i := \frac{2^{-i}\binom{m-r}{h-i}}{\binom{m}{h}}$$

of being a correct guess. Again, let $i_0, \dots i_{\min(r,h)-\max(0,h-(m-r))}$ be ordered such that $q_{i_0} \geq \dots \geq q_{i_{\min(r,h)-\max(0,h-(m-r))}}$. We may again construct S as a union $S = S_{i_0} \cup \dots \cup S_{i_{k-1}} \cup S'_{i_k}$ for some $k \in \mathbb{N}_0$, where S'_{i_k} is some subset of S_{i_k} by minimizing (7.8) over the choice of k and the size of the subset S'_{i_k} of S_{i_k}.

In Section 7.4.2, we provide examples that showcase the improvements gained by the above techniques.

7.4 Results

In this section, we present concrete runtime estimates of our quantum hybrid attack for various uSVP instances and provide a comparison to the classical hybrid and the primal attack. For all our runtime estimates in this section we assume that one Nearest Plane call in dimension d costs $d^2/(2^{1.06})$ operations. If not specified otherwise, we apply the enumeration-based cost model $\log_2(8d \cdot 2^{0.18728\beta\log_2(\beta)-1.0192\beta+16.1})$ for BKZ-β in dimension d.

7.4.1 Comparison to the Classical Hybrid and Primal Attack

In this section, we compare the quantum hybrid attack to the classical hybrid attack and the 2016 estimate for the primal attack (cf. Chapter 3). To this end, as in Chapters 5 and 6, we analyze a uSVP instance of fixed lattice dimension 512 and determinant 128^{256} with a random binary unique shortest non-zero vector, which underlies the first proposed parameter set of the encryption scheme by Buchmann et al. [BGG+16]. For our comparison, we do not shift the binary vector, as for instance discussed in Section 7.2.3. We apply the enumeration-based $\log_2(8d \cdot 2^{0.18728\beta\log_2(\beta)-1.0192\beta+16.1})$ cost model for BKZ. The results, including the optimal attack parameters, are shown in Table 7.1. The expected attack cost significantly drops from 2^{151} for the primal attack to 2^{109} for the classical attack. This cost is further reduced to 2^{90} when using the quantum hybrid attack.

Attack	**Quantum Hybrid**	**Classical Hybrid**	**Primal**
Cost	**90**	**109**	**151**
Guessing dimension	135	124	—
Block size	158	185	256

Table 7.1: Expected costs and attack parameters for the quantum hybrid attack, classical hybrid attack, and primal attack against a uSVP instance of fixed lattice dimension 512 and determinant 128^{256} with a random binary unique shortest non-zero vector.

How the runtime of the classical hybrid attack can be reduced using parallel computing techniques is shown in Chapter 6. A comparison between the quantum hybrid attack and an improved version of the primal attack for small or sparse secrets can be found in Section 8.5.

7.4.2 Small and Sparse Secret vectors

In this section, we analyze the behavior of the quantum hybrid attack on uSVP instances with small and sparse secret vectors and compare its performance to the primal attack under the 2016 estimate (cf. Chapter 3). To that end, we analyze uSVP instances in lattice dimension 512 with determinant 128^{256}, where the unique shortest non-zero vector is of the form $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$ with a uniformly random $\mathbf{v}_1 \in \{0,1\}^{256}$ and $\mathbf{v}_2$ is either uniformly random binary, uniformly random ternary, or random binary or ternary with a fixed Hamming weight. Such instances may for example appear in instantiations of NTRU or LWE with small and sparse secrets.

We compare the quantum hybrid attack with additional scaling or search-space optimization techniques to the quantum hybrid attack in its simple form and to the primal attack. For the quantum hybrid attack, we optimized its runtime according to Section 7.2.2.

Our runtime estimates and the corresponding attack parameters assuming either enumeration-based or quantum-sieving-based BKZ are shown in Table 7.2 and Table 7.3, respectively. The results show that for all except one (in the quantum-sieving regime) analyzed uSVP instances with binary and ternary shortest non-zero vectors, the quantum hybrid attack significantly outperforms the primal attack. The gap between the runtime of the quantum hybrid and the primal attack grows bigger and bigger as the vectors get more sparse. One can also notice that in general the size of the search space needs to be optimized as the naive choices do not yield optimal attacks, see Section 7.3.

A comparison between the quantum hybrid attack and an improved version of the primal attack for small or sparse secrets applied to lattice-based schemes is conducted in Section 8.5.

Quantum hybrid attack with scaling and the optimizing search space								
Structure	rand. ter.	rand. bin.	ter. $h = 64$	bin. $h = 64$	ter. $h = 32$	bin. $h = 32$	ter. $h = 16$	bin. $h = 16$
Expected cost	**116**	**90**	**88**	**78**	**66**	**61**	**49**	**46**
Guessing dim.	110	135	135	150	163	170	189	198
Block size	189	158	149	141	109	109	82	76
$\|S\|/\|M\|$	1	1	2^{-72}	2^{-30}	2^{-65}	2^{-35}	2^{-37}	2^{-28}
Quantum hybrid attack without scaling and optimizing the search space								
Structure	rand. ter.	rand. bin.	ter. $h = 64$	bin. $h = 64$	ter. $h = 32$	bin. $h = 32$	ter. $h = 16$	bin. $h = 16$
Expected cost	**116**	**90**	**95**	**81**	**76**	**66**	**57**	**52**
Guessing dim.	110	135	124	144	152	168	178	201
Block size	189	158	164	146	141	126	109	101
$\|S\|/\|M\|$	1	1	1	1	1	1	1	1
Primal attack under the 2016 estimate								
Structure	rand. ter.	rand. bin.	ter. $h = 64$	bin. $h = 64$	ter. $h = 32$	bin. $h = 32$	ter. $h = 16$	bin. $h = 16$
Expected cost	**158**	**151**	**139**	**139**	**132**	**132**	**129**	**129**
Block size	266	256	241	241	231	231	226	226

Table 7.2: Expected costs and corresponding attack parameters for uSVP instances of lattice dimension 512, determinant q^{256}, and a unique shortest non-zero vector of the form $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$ with a uniformly random $\mathbf{v}_1 \in \{0,1\}^{256}$ and $\mathbf{v}_2$ is either uniformly random binary, uniformly random ternary, or random binary or ternary with a fixed Hamming weight h. We optimized the guessing dimension, the block size, and the size of the search space S relative to the maximal search space M, i.e. $|S|/|M|$. Assuming the enumeration-based cost model $\log_2(8d \cdot 2^{0.18728\beta\log_2(\beta)-1.0192\beta+16.1})$ for BKZ-β in dimension d.

Quantum hybrid attack with scaling and optimizing the search space

Structure	rand. ter.	rand. bin.	ter. $h = 64$	bin. $h = 64$	ter. $h = 32$	bin. $h = 32$	ter. $h = 16$	bin. $h = 16$
Expected cost	**101**	**84**	**83**	**75**	**66**	**62**	**53**	**46**
Guessing dim.	96	128	124	140	165	175	197	212
Block size	240	193	178	158	117	109	76	65
$\lvert S\rvert / \lvert M\rvert$	1	1	2^{-64}	2^{-27}	2^{-57}	2^{-32}	2^{-25}	2^{-17}

Quantum hybrid attack without scaling and the optimizing search space

Structure	rand. ter.	rand. bin.	ter. $h = 64$	bin. $h = 64$	ter. $h = 32$	bin. $h = 32$	ter. $h = 16$	bin. $h = 16$
Expected cost	**101**	**84**	**87**	**77**	**73**	**66**	**58**	**52**
Guessing dim.	96	128	114	128	144	158	192	207
Block size	240	193	201	171	158	126	109	88
$\lvert S\rvert / \lvert M\rvert$	1	1	1	1	1	1	1	1

Primal attack under the 2016 estimate

Structure	rand. ter.	rand. bin.	ter. $h = 64$	bin. $h = 64$	ter. $h = 32$	bin. $h = 32$	ter. $h = 16$	bin. $h = 16$
Expected cost	**99**	**96**	**92**	**92**	**90**	**90**	**88**	**88**
Block size	266	256	241	241	231	231	226	226

Table 7.3: Expected costs and corresponding attack parameters for uSVP instances of lattice dimension 512, determinant q^{256}, and a unique shortest non-zero vector of the form $\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2)$ with a uniformly random $\mathbf{v}_1 \in \{0,1\}^{256}$ and $\mathbf{v}_2$ is either uniformly random binary, uniformly random ternary, or random binary or ternary with a fixed Hamming weight h. We optimized the guessing dimension, the block size, and the size of the search space S relative to the maximal search space M, i.e. $\lvert S\rvert / \lvert M\rvert$. Assuming the quantum-sieving-based cost model $\log_2(8d \cdot 2^{0.265\beta+16.4})$ for BKZ-β in dimension d.

7.4.3 Gaussian Distributions

In this section, we show that the quantum hybrid attack is suitable for uSVP instances where the unique shortest non-zero vector is drawn from a (narrow) discrete Gaussian distribution. We analyze uSVP instances with lattice dimension 512, determinant q^{256}, and a unique shortest non-zero vector whose components are drawn from a discrete Gaussian distribution D_σ of standard deviation σ for different q and σ with respect to the quantum hybrid attack and the primal attack under the 2016 estimate (cf. Chapter 3). Note that theoretically, the discrete Gaussian distributions have infinite support, while our analysis requires finite support. However, using a standard tailbound argument [LP11] one can show that with overwhelming probability the absolute value of D_σ is bounded by 14σ. We therefore assume that the distributions D_σ have finite support $\{-\lceil 14\sigma \rceil, \ldots, \lceil 14\sigma \rceil\}$. For the quantum hybrid attack, we optimized the runtime of the attack according to Section 7.2.2 using the $\log_2(8d \cdot 2^{0.18728\beta\log_2(\beta)-1.0192\beta+16.1})$ cost model for BKZ.

The expected attack costs are shown in Table 7.4. The corresponding attack parameters (guessing dimension and block size for the quantum hybrid attack and block size for the primal attack) are shown in Table 7.5. Note that the table is designed such that (assuming the Gaussian heuristic for the second successive minimum $\lambda_2(\Lambda)$) both going from one column to the next (i.e., decreasing q) and going from one row to the next (i.e., increasing σ) decreases the uSVP gap $\lambda_2(\Lambda)/\lambda_1(\Lambda)$ by a factor of 2. The results show that for certain instantiations of uSVP with a Gaussian shortest non-zero vector the quantum hybrid attack outperforms the primal attack. This is not the case for the classical hybrid attack and was enabled by replacing the meet-in-the-middle search by a quantum search that is sensitive to the underlying distribution. In the following, we explain the results shown in Table 7.4 in more detail. For fixed dimension, assuming the Gaussian heuristic for the second successive minimum, the 2016 estimate only depends on the uSVP gap (cf. Section 4.2.1). Hence, for the same gap we obtain the same cost for the primal attack, and decreasing the gap by increasing sigma and decreasing the gap by decreasing the determinant has the same effect on the expected cost under the 2016 estimate. This is not true for the quantum hybrid attack. In this case, decreasing the gap by increasing sigma results in a worse runtime than decreasing the gap by decreasing the determinant. This can be explained by the negative effect of increasing sigma on the quantum search phase. As a consequence, the runtime of the quantum hybrid attack increases when keeping the uSVP gap constant while increasing sigma. Therefore, for each fixed uSVP gap and varying sigma, there typically exists a crossover point at which the quantum hybrid attack becomes more efficient than the primal attack. Note that if one assumes quantum sieving to be feasible as an SVP oracle in BKZ, these crossover points might not be within reasonable parameters for Gaussian distributions, rendering the quantum hybrid less efficient than the primal attack in this case.

σ \ q	$64 \cdot 256$	$16 \cdot 256$	$4 \cdot 256$	$1 \cdot 256$
1	$(60, 57)$	$(76, 76)$	$(98, 104)$	$(128, 151)$
2	$(82, 76)$	$(107, 104)$	$(143, 151)$	$(194, 230)$
4	$(113, 104)$	$(152, 151)$	$(207, 230)$	$(288, 375)$
8	$(158, 151)$	$(217, 230)$	$(321, 375)$	$(423, 672)$

Table 7.4: Expected costs $(T_{\mathsf{qhybrid}}, T_{\mathsf{primal}})$ for the quantum hybrid attack and the primal attack for uSVP instances of lattice dimension 512, determinant q^{256}, and a unique shortest non-zero vector whose components are drawn from a discrete Gaussian distribution of standard deviation σ.

σ \ q	$64 \cdot 256$	$16 \cdot 256$	$4 \cdot 256$	$1 \cdot 256$
1	$((35, 109), 113)$	$((47, 132), 146)$	$((62, 160), 191)$	$((84, 197), 257)$
2	$((34, 138), 146)$	$((50, 178), 191)$	$((66, 216), 257)$	$((93, 275), 356)$
4	$((38, 178), 191)$	$((52, 222), 257)$	$((74, 283), 356)$	$((102, 356), 519)$
8	$((43, 225), 257)$	$((59, 281), 356)$	$((83, 358), 519)$	$((124, 479), 814)$

Table 7.5: Optimal attack parameters for the quantum hybrid and block sizes for the primal attack $((r_{\mathsf{qhybrid}}, \beta_{\mathsf{qhybrid}}), \beta_{2016})$ corresponding to Table 7.4.

8 Security Estimates for Lattice-based Candidates for NIST's Standardization

In 2015, the US National Institute of Standards and Technology (NIST) initiated a process of standardizing post-quantum Public-Key Encryption (PKE) schemes, Key Encapsulation Mechanisms (KEM), and Digital Signature Algorithms (SIG), resulting in a call for proposals in 2016 [Nat16]. Among the accepted submissions, 23 are either based on the hardness of LWE or NTRU problems. In their submissions, the authors were asked to provide security estimates for their schemes and categorize them into one or more of five security categories. However, the different submissions used numerous different cost models to estimate their scheme's security, making it hard to compare security levels across the submissions.

Contribution. In this chapter, we analyze the security of the LWE and NTRU-based NIST submissions with respect to the primal attack under the 2016 estimate (cf. Chapter 3) and the quantum hybrid attack (cf. Chapter 7). To this end, we apply the primal attack to all schemes, utilizing the [APS15] estimator[15] using all of the different cost models for lattice reduction proposed in the NIST submissions. This enables a fair comparison of security levels across the submissions. We further analyze selected schemes with respect to the quantum hybrid attack. Depending on the assumed cost of lattice reduction, our results yield either significantly lower or comparable attack costs for the quantum hybrid attack when compared to the primal attack.

Organization. After recalling the definition of NIST's security categories in Section 8.1, we summarize the analyzed schemes and extract the proposed parameters from the submissions to NIST in Section 8.2. A summary of the proposed cost models for BKZ as part of a NIST submission is given in Section 8.3. Our analysis of the proposed schemes with respect to the primal attack is presented in Section 8.4.

[15] `https://bitbucket.org/malb/lwe-estimator`, commit `1850100`.

Our analysis of selected schemes with respect to the quantum hybrid attack is shown in Section 8.5.

Publications. This chapter is based on the publication [5], which will be presented at SCN 2018. In addition, the considerations with respect to the quantum hybrid attack are novel in this thesis.

8.1 NIST's Security Categories

The goal of NIST's standardization process [Nat16] is to meet the cryptographic requirements for communication (e.g., via the internet) in an era where large-scale quantum computers exist. The call for proposals received 69 "complete and proper" submissions, out of which 23 are based on either the LWE or the NTRU family of lattice problems. Participants were invited to submit their cryptographic schemes, along with different parameter sets aimed at meeting the requirements of one or more of the following security categories.

1. Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 128-bit key (e.g. AES128)
2. Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for collision search on a 256-bit hash function (e.g. SHA256/ SHA3-256)
3. Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 192-bit key (e.g. AES192)
4. Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for collision search on a 384-bit hash function (e.g. SHA384/ SHA3-384)
5. Any attack that breaks the relevant security definition must require computational resources comparable to or greater than those required for key search on a block cipher with a 256-bit key (e.g. AES 256)

([Nat16])

These categories roughly indicate how classical and quantum attacks on the proposed schemes compare to attacks on AES and SHA-3 in the post-quantum context. As part of their submissions participants were asked to provide cryptanalysis supporting their security claims, and to use this cryptanalysis to roughly estimate the size of the security parameter for each parameter set.

8.2 Proposed Schemes

The three tables below specify the parameter sets for the schemes considered. Table 8.1 gives the parameters for the NTRU-based schemes. In Table 8.2 these parameters are converted into the LWE-based context as detailed in Section 8.4. Table 8.3 gives the parameters for the LWE-based schemes in terms of plain LWE, that is, ignoring the potential ring or module structure.

Throughout, n is the dimension of the problem and q the modulus. The polynomial ϕ, if present, is the polynomial used to define the base ring $\mathcal{R}_q = \mathbb{Z}_q[x]/(\phi)$ from which Ring-/Module-LWE or NTRU elements are drawn. In Tables 8.2 and 8.3, the value σ is the standard deviation of the (discrete Gaussian) distribution χ from which the LWE errors are drawn. If the error distribution is not a discrete Gaussian, our approaches are explained in Section 8.4. If the secret distribution is "normal", i.e. in the normal form, this means it is the same distribution as the error, namely χ. If not, the distribution given determines the secret distribution. We use the following notation for these distributions. For integers a and b we use (a, b) to denote the uniform distribution on the integer interval from a to b. Furthermore, for some positive integer $k \leq n$ we use $((-1, 1), k)$ to denote the uniform distribution on the set of vectors in $\{-1, 0, 1\}^n$ with Hamming weight k.

Name	n	q	$\|f\|$	$\|g\|$	NIST	Assumption	ϕ	Primitive
NTRUEncrypt	443	2048	16.94	16.94	1	NTRU	$x^n - 1$	KEM, PKE
	743	2048	22.25	22.25	$1, 2, 3, 4, 5$	NTRU	$x^n - 1$	KEM, PKE
	1024	1073750017	23168.00	23168.00	$4, 5$	NTRU	$x^n - 1$	KEM, PKE
Falcon	512	12289	91.71	91.71	1	NTRU	$x^n + 1$	SIG
	768	18433	112.32	112.32	$2, 3$	NTRU	$x^n - x^{n/2} + 1$	SIG
	1024	12289	91.71	91.71	$4, 5$	NTRU	$x^n + 1$	SIG
NTRU HRSS	700	8192	20.92	20.92	1	NTRU	$\sum_{i=0}^{n-1} x^i$	KEM
S/L NTRU Prime	761	4591	16.91	22.52	5	NTRU	$x^n - x - 1$	KEM
pqNTRUsign	1024	65537	22.38	22.38	$1, 2, 3, 4, 5$	NTRU	$x^n - 1$	SIG

Table 8.1: Parameter sets for NTRU-based schemes with secret dimension n, modulus q, small polynomials f and g, and ring $\mathbb{Z}_q[x]/(\phi)$. The NIST column indicates the NIST security category aimed at.

Name	n	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
NTRUEncrypt	443	2048	0.80	$((-1,1),287)$	1	NTRU	x^n-1	KEM, PKE
	743	2048	0.82	$((-1,1),495)$	1, 2, 3, 4, 5	NTRU	x^n-1	KEM, PKE
	1024	1073750017	724.00	normal	4, 5	NTRU	x^n-1	KEM, PKE
Falcon	512	12289	4.05	normal	1	NTRU	x^n+1	SIG
	768	18433	4.05	normal	2, 3	NTRU	$x^n-x^{n/2}+1$	SIG
	1024	12289	2.87	normal	4, 5	NTRU	x^n+1	SIG
NTRU HRSS	700	8192	0.79	$((-1,1),437)$	1	NTRU	$\sum_{i=0}^{n-1} x^i$	KEM
SNTRU Prime	761	4591	0.82	$((-1,1),286)$	5	NTRU	x^n-x-1	KEM
pqNTRUSign	1024	65537	0.70	$((-1,1),501)$	1, 2, 3, 4, 5	NTRU	x^n-1	SIG

Table 8.2: LWE parameter sets for NTRU-based schemes, with dimension n, modulus q, standard deviation of the error σ, and ring $\mathbb{Z}_q[x]/(\phi)$. The parameters are obtained following Section 8.4. The NIST column indicates the NIST security category aimed at.

Name	n	k	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
KCL-RLWE	1024	—	12289	2.83	normal	5	RLWE	x^n+1	KEM
KCL-MLWE	768	3	7681	1.00	normal	4	MLWE	$x^{n/k}+1$	KEM
	768	3	7681	2.24	normal	4	MLWE	$x^{n/k}+1$	KEM
BabyBear	624	2	1024	1.00	normal	2	ILWE	$q^{n/k}-q^{n/(2k)}-1$	KEM
	624	2	1024	0.79	normal	2	ILWE	$q^{n/k}-q^{n/(2k)}-1$	KEM
MamaBear	936	3	1024	0.94	normal	5	ILWE	$q^{n/k}-q^{n/(2k)}-1$	KEM
	936	3	1024	0.71	normal	4	ILWE	$q^{n/k}-q^{n/(2k)}-1$	KEM
PapaBear	1248	4	1024	0.87	normal	5	ILWE	$q^{n/k}-q^{n/(2k)}-1$	KEM
	1248	4	1024	0.61	normal	5	ILWE	$q^{n/k}-q^{n/(2k)}-1$	KEM
CRYSTALS-Dilithium	768	3	8380417	3.74	$(-6,6)$	1	MLWE	$x^{n/k}+1$	SIG
	1024	4	8380417	3.16	$(-5,5)$	2	MLWE	$x^{n/k}+1$	SIG
	1280	5	8380417	2.00	$(-3,3)$	3	MLWE	$x^{n/k}+1$	SIG
CRYSTALS-Kyber	512	2	7681	1.58	normal	1	MLWE	$x^{n/k}+1$	KEM, PKE
	768	3	7681	1.41	normal	3	MLWE	$x^{n/k}+1$	KEM, PKE
	1024	4	7681	1.22	normal	5	MLWE	$x^{n/k}+1$	KEM, PKE
Ding Key Exchange	512	—	120883	4.19	normal	1	RLWE	x^n+1	KEM
	1024	—	120883	2.60	normal	$3, 5$	RLWE	x^n+1	KEM
EMBLEM	770	—	16777216	25.00	$(-1,1)$	1	LWE	—	KEM, PKE
	611	—	16777216	25.00	$(-2,2)$	1	LWE	—	KEM, PKE
R EMBLEM	512	—	65536	25.00	$(-1,1)$	1	RLWE	x^n+1 †	KEM, PKE
	512	—	16384	3.00	$(-1,1)$	1	RLWE	x^n+1 †	KEM, PKE
Frodo	640	—	32768	2.75	normal	1	LWE	—	KEM, PKE
	976	—	65536	2.30	normal	3	LWE	—	KEM, PKE
NewHope	512	—	12289	2.00	normal	1	RLWE	x^n+1	KEM, PKE
	1024	—	12289	2.00	normal	5	RLWE	x^n+1	KEM, PKE
HILA5	1024	—	12289	2.83	normal	5	RLWE	x^n+1	KE

Name	n	k	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
KINDI	768	3	16384	2.29	$(-4,4)$	2	MLWE	$x^{n/k}+1$	KEM, PKE
	1024	2	8192	1.12	$(-2,2)$	4	MLWE	$x^{n/k}+1$	KEM, PKE
	1024	2	16384	2.29	$(-4,4)$	4	MLWE	$x^{n/k}+1$	KEM, PKE
	1280	5	16384	1.12	$(-2,2)$	5	MLWE	$x^{n/k}+1$	KEM, PKE
	1536	3	8192	1.12	$(-2,2)$	5	MLWE	$x^{n/k}+1$	KEM, PKE
LAC	512	—	251	0.71	normal	$1,2$	PLWE	x^n+1	KE, KEM, PKE
	1024	—	251	0.50	normal	$3,4$	PLWE	x^n+1	KE, KEM, PKE
	1024	—	251	0.71	normal	5	PLWE	x^n+1	KE, KEM, PKE
LIMA-2p	1024	—	133121	3.16	normal	3	RLWE	x^n+1	KEM, PKE
	2048	—	184321	3.16	normal	4	RLWE	x^n+1	KEM, PKE
LIMA-sp	1018	—	12521473	3.16	normal	1	RLWE	$\sum_{i=0}^{n} x^i$	KEM, PKE
	1306	—	48181249	3.16	normal	2	RLWE	$\sum_{i=0}^{n} x^i$	KEM, PKE
	1822	—	44802049	3.16	normal	3	RLWE	$\sum_{i=0}^{n} x^i$	KEM, PKE
	2062	—	16900097	3.16	normal	4	RLWE	$\sum_{i=0}^{n} x^i$	KEM, PKE
Lizard	1024	—	2048	1.12	$((-1,1),140)$	1	LWE, LWR	—	KEM, PKE
	1024	—	1024	1.12	$((-1,1),128)$	1	LWE, LWR	—	KEM, PKE
	1024	—	2048	1.12	$((-1,1),200)$	3	LWE, LWR	—	KEM, PKE
	1024	—	2048	1.12	$((-1,1),200)$	3	LWE, LWR	—	KEM, PKE
	2048	—	4096	1.12	$((-1,1),200)$	5	LWE, LWR	—	KEM, PKE
	2048	—	2048	1.12	$((-1,1),200)$	5	LWE, LWR	—	KEM, PKE
RLizard	1024	—	1024	1.12	$((-1,1),128)$	1	RLWE, RLWR	x^n+1	KEM, PKE
	1024	—	2048	1.12	$((-1,1),264)$	3	RLWE, RLWR	x^n+1	KEM, PKE
	2048	—	2048	1.12	$((-1,1),164)$	3	RLWE, RLWR	x^n+1	KEM, PKE
	2048	—	4096	1.12	$((-1,1),256)$	5	RLWE, RLWR	x^n+1	KEM, PKE
LOTUS	576	—	8192	3.00	normal	$1,2$	LWE	—	KEM, PKE
	704	—	8192	3.00	normal	$3,4$	LWE	—	KEM, PKE
	832	—	8192	3.00	normal	5	LWE	—	KEM, PKE
uRound2.KEM	500	—	16384	2.29	$((-1,1),74)$	1	LWR	—	KEM
	580	—	32768	4.61	$((-1,1),116)$	2	LWR	—	KEM
	630	—	32768	4.61	$((-1,1),126)$	3	LWR	—	KEM
	786	—	32768	4.61	$((-1,1),156)$	4	LWR	—	KEM
	786	—	32768	4.61	$((-1,1),156)$	5	LWR	—	KEM
uRound2.KEM	418	—	4096	4.61	$((-1,1),66)$	1	RLWR	$\sum_{i=0}^{n} x^i$	KEM
	522	—	32768	36.95	$((-1,1),78)$	2	RLWR	$\sum_{i=0}^{n} x^i$	KEM
	540	—	16384	18.47	$((-1,1),96)$	3	RLWR	$\sum_{i=0}^{n} x^i$	KEM
	700	—	32768	36.95	$((-1,1),112)$	4	RLWR	$\sum_{i=0}^{n} x^i$	KEM
	676	—	32768	36.95	$((-1,1),120)$	5	RLWR	$\sum_{i=0}^{n} x^i$	KEM
uRound2.PKE	500	—	32768	4.61	$((-1,1),74)$	1	LWR	—	PKE
	585	—	32768	4.61	$((-1,1),110)$	2	LWR	—	PKE
	643	—	32768	4.61	$((-1,1),114)$	3	LWR	—	PKE
	835	—	32768	2.29	$((-1,1),166)$	4	LWR	—	PKE
	835	—	32768	2.29	$((-1,1),166)$	5	LWR	—	PKE
uRound2.PKE	420	—	1024	1.12	$((-1,1),62)$	1	RLWR	$\sum_{i=0}^{n} x^i$	PKE
	540	—	8192	4.61	$((-1,1),96)$	2	RLWR	$\sum_{i=0}^{n} x^i$	PKE
	586	—	8192	4.61	$((-1,1),104)$	3	RLWR	$\sum_{i=0}^{n} x^i$	PKE
	708	—	32768	18.47	$((-1,1),140)$	$4,5$	RLWR	$\sum_{i=0}^{n} x^i$	PKE
nRound2.KEM	400	—	3209	3.61	$((-1,1),72)$	1	RLWR	$\sum_{i=0}^{n} x^i$	KEM
	486	—	1949	2.18	$((-1,1),96)$	2	RLWR	$\sum_{i=0}^{n} x^i$	KEM
	556	—	3343	3.76	$((-1,1),88)$	3	RLWR	$\sum_{i=0}^{n} x^i$	KEM
	658	—	1319	1.46	$((-1,1),130)$	$4,5$	RLWR	$\sum_{i=0}^{n} x^i$	KEM
nRound2.PKE	442	—	2659	1.47	$((-1,1),74)$	1	RLWR	$\sum_{i=0}^{n} x^i$	PKE
	556	—	3343	1.86	$((-1,1),88)$	2	RLWR	$\sum_{i=0}^{n} x^i$	PKE
	576	—	2309	1.27	$((-1,1),108)$	3	RLWR	$\sum_{i=0}^{n} x^i$	PKE
	708	—	2837	1.57	$((-1,1),140)$	$4,5$	RLWR	$\sum_{i=0}^{n} x^i$	PKE
LightSaber	512	2	8192	2.29	normal	1	MLWR	$x^{n/k}+1$	KEM, PKE

Name	n	k	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
NTRU LPrime	761	—	4591	0.82	$((-1,1),250)$	5	RLWR	$x^n - x - 1$	KEM
Saber	768	3	8192	2.29	normal	3	MLWR	$x^{n/k} + 1$	KEM, PKE
FireSaber	1024	4	8192	2.29	normal	5	MLWR	$x^{n/k} + 1$	KEM, PKE
qTESLA	1024	—	8058881	8.49	normal	1	RLWE	$x^n + 1$	SIG
	2048	—	12681217	8.49	normal	3	RLWE	$x^n + 1$	SIG
	2048	—	27627521	8.49	normal	5	RLWE	$x^n + 1$	SIG
Titanium.PKE	1024	—	86017	1.41	normal	1	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	PKE
	1280	—	301057	1.41	normal	1	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	PKE
	1536	—	737281	1.41	normal	3	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	PKE
	2048	—	1198081	1.41	normal	5	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	PKE
Titanium.KEM	1024	—	118273	1.41	normal	1	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	KEM
	1280	—	430081	1.41	normal	1	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	KEM
	1536	—	783361	1.41	normal	3	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	KEM
	2048	—	1198081	1.41	normal	5	PLWE	$x^n + \sum_{i=1}^{n-1} f_i x^i + f_0$ *	KEM

Table 8.3: Parameter sets for LWE-based schemes with secret dimension n, MLWE rank k (if any), modulus q, standard deviation of the error σ. If the LWE samples come from a Ring- or Module-LWE instance, the ring is $\mathbb{Z}_q[x]/(\phi)$. The NIST column indicates the NIST security category aimed at. *For Titanium no ring is explicitly chosen but the scheme simultaneously relies on a family of rings where $f_i \in \{-1, 0, 1\}$, $f_0 \in \{-1, 1\}$. †For R EMBLEM we list the parameters from the reference implementation since a suitable ϕ could not be found for those proposed in [SPL+17, Table 2].

8.3 Proposed Costs for Lattice Reduction

There exist multiple different cost models for the runtime of BKZ in the literature, e.g., [CN11, APS15, ADPS16]. The main differences between these models are whether they rely on sieving or enumeration as an SVP subroutine and how many calls to the SVP oracle are assumed (cf. Chapter 2). A summary of every cost model applied in the NIST submissions can be found in Table 8.4.

The most commonly considered SVP oracle among the NIST submissions is sieving. In the literature, its cost on a random lattice of dimension β is estimated as $2^{c\beta+o(\beta)}$, where $c = 0.292$ classically [BDGL16], with Grover speedups lowering this to $c = 0.265$ [Laa15a] in the quantum setting. A "paranoid" lower bound is given in [ADPS16] as $2^{0.2075\beta+o(\beta)}$ based on the "kissing number". Some authors replace $o(\beta)$ by the constant 16.4 [APS15], based on experiments in [Laa15b], some authors omit it. A "min space" variant of sieving is also considered in [BDGL16], which uses $c = 0.368$ with Grover speedups lowering this to $c = 0.2975$ [Laa15a].

Alternatively, enumeration is considered in some of the submissions. In particular, it can be found to be estimated as $2^{c_1\beta\log_2\beta+c_2\beta+c_3}$ [Kan83, MW15] or as $2^{c_1\beta^2+c_2\beta+c_3}$ [FP85, CN11], with Grover speedups considered to half the exponent [ANS18]. The estimates $0.187\beta\log_2\beta - 1.019\beta + 16.1$ [APS15] and $0.000784\beta^2 + 0.366\beta - 0.9$ [HPS+15] are based on fitting the same data from [Che13].

With respect to the number of SVP oracle calls required by BKZ, a popular choice among the submissions was to follow the "Core-SVP" model introduced in [ADPS16], that conservatively assumes that only a single call to the SVP oracle. Alternatively, the number of calls has also been estimated to be $8d$ (for example, in [Alb17]), where d is the dimension of the embedding lattice and β is the BKZ block size.

LOTUS [PHAM17] is the only submission not to provide a closed formula for estimating the cost of BKZ. Given their preference for enumeration, we fit their estimated cost model to a curve of shape $2^{c_1\beta\log_2\beta+c_2\beta+c_3}$ following [MW15]. We fit a curve to the values given by (39) in [PHAM17], the script used is available in the public repository.

The NTRU Prime submission [BCLvV17a] utilizes the BKZ 2.0 simulator of [CN11] to determine the necessary block size and number of tours to achieve a certain root Hermite factor prior to applying their BKZ cost model. In contrast, we apply the asymptotic formula from [Che13] to relate block size and root Hermite factor, and consider BKZ to complete in 8 tours while matching their cost asymptotic for a single enumeration call.

8.4 Estimates for the Primal Attack

For our experiments we make use of the LWE estimator[16] from [APS15], which allows one to specify arbitrary cost models for BKZ. We wrap it in a script that loops though the proposed schemes and cost models, estimating the cost of the appropriate variants of the primal attack. Note that the estimator considers choosing the optimal number of LWE samples, rescaling the LWE secret, and dimension reducing techniques for small or sparse secret variants when costing the primal attack according to the 2016 estimate. The results may therefore differ from a plain application of the 2016 estimate (cf. Chapter 3). For the following reason, we restrict the number of LWE samples provided to an attacker to n or $2n$. In the RLWE KEM setting – which is the most common for the schemes considered in this chapter – the public key is one RLWE sample $(a, b) = (a, a \cdot s + e)$ for some short s, e and encapsulations consist of two RLWE samples $v \cdot a + e'$ and $v \cdot b + e'' + \tilde{m}$ where $\tilde{m}$ is some encoding of a random string and v, e', e'' are short. Thus, depending on the target, the adversary is given either n or $2n$ plain LWE samples. However, note that in a typical setting the adversary does not get to enjoy the full power of having two samples at its disposal, because, firstly, the random string $\tilde{m}$ increases the noise in

[16] `https://bitbucket.org/malb/lwe-estimator`, commit 1850100.

Model	**Schemes**
0.292β 0.265β	CRYSTALS [LDK+17, SAB+17] SABER [DKRV17] Falcon [PFH+17] ThreeBears [Ham17] HILA5 [Saa17] Titanium [SSZ17] KINDI [Ban17] NTRU HRSS [SHRS17] LAC [LLJ+17] NTRUEncrypt [ZCHW17a] New Hope [PAA+17] pqNTRUSign [ZCHW17b]
$0.292\beta + 16.4$ $0.265\beta + 16.4$	LIMA [SAL+17]
0.368β 0.2975β	NTRU HRSS [SHRS17]
$0.292\beta + \log_2(\beta)$ $0.265\beta + \log_2(\beta)$	Frodo [NAB+17] KCL [ZjGS17] Lizard [CPL+17] Round2 [GMZB+17]
$0.292\beta + 16.4 + \log_2(8d)$	Ding Key Exchange [DTGW17] EMBLEM [SPL+17]
$0.265\beta + 16.4 + \log_2(8d)$	qTESLA [BAA+17]
$0.187\beta \log_2 \beta - 1.019\beta + 16.1$	NTRU HRSS [SHRS17] pqNTRUSign [ZCHW17b] NTRUEncrypt [ZCHW17a]
$\frac{1}{2}(0.187\beta \log_2 \beta - 1.019\beta + 16.1)$	NTRU HRSS [SHRS17]
$0.000784\beta^2 + 0.366\beta - 0.9 + \log_2(8d)$	NTRU Prime [BCLvV17a]
$0.125\beta \log_2 \beta - 0.755\beta + 2.25$	LOTUS [PHAM17]

Table 8.4: Cost models proposed as part of a PQC NIST submission. The name of a model is the base-2 logarithm of its cost.

$v \cdot b + e'' + \tilde{m}$ by a factor of 2 and, secondly, because many schemes drop lower order bits from $v \cdot b + e'' + \tilde{m}$ to save bandwidth. Due to the way decryption works this bit dropping can be quite aggressive, and thus the noise in the second sample can be quite large compared to the original noise rate. In the case of Module-LWE, a ciphertext in transit produces a smaller number of LWE samples, but n samples can still be recovered from the public key. In this chapter, we consider the n and $2n$ scenarios for all schemes and leave distinguishing which scenario applies to which scheme for future work.

Our code to estimate the security of the schemes is available at `https://github.com/estimate-all-the-lwe-ntru-schemes`. Our results are given in Tables 8.5, 8.6, 8.7, 8.8, 8.9, and 8.10. A user friendly version of these tables is available at `https://estimate-all-the-lwe-ntru-schemes.github.io`. In particular, the HTML version supports filtering and sorting the table. It also contains SageMath source code snippets to reproduce each entry.

Scheme	Claim	NIST	0.265β	0.265β $+16.4$	0.2975β	0.265β $+\log_2(\beta)$	$0.265\beta+16.4$ $+\log_2(8d)$	0.292β	0.292β $+16.4$	0.368β	0.292β $+\log_2(\beta)$	$0.292\beta+16.4$ $+\log_2(8d)$
BabyBear-0624-0.79-1024	141.00	2	143	159	160	152	172	157	173	198	166	187
BabyBear-0624-1.00-1024	152.00	2	153	169	172	163	183	169	185	213	178	199
CRYSTALS-Dilithium-0768-3.74-8380417	91.00	1	92	108	104	101	122	102	118	128	110	132
CRYSTALS-Dilithium-1024-3.16-8380417	125.00	2	130	146	146	139	160	143	159	180	152	173
CRYSTALS-Dilithium-1280-2.00-8380417	158.00	3	159	175	179	168	190	175	191	221	185	206
CRYSTALS-Kyber-0512-1.58-7681	102.00	1	103	119	115	111	132	113	129	143	122	143
CRYSTALS-Kyber-0768-1.41-7681	161.00	3	163	179	183	172	193	180	196	226	189	210
CRYSTALS-Kyber-1024-1.22-7681	218.00	5	221	237	248	230	251	243	259	306	253	273
Ding Key Exchange-0512-4.19-120883	—	1	92	108	103	100	121	101	117	127	110	131
Ding Key Exchange-1024-2.60-120883	—	3, 5	191	207	214	200	221	210	226	265	220	241
EMBLEM-0611-25.00-16777216	128.30	1	69	85	78	77	99	76	92	96	84	106
EMBLEM-0770-25.00-16777216	128.30	1	90	106	101	98	120	99	115	125	107	129
FireSaber-1024-2.29-8192	245.00	5	257	273	288	267	287	283	300	357	293	314
Frodo-0640-2.75-32768	103.00	1	129	145	145	138	159	142	158	179	151	172
Frodo-0976-2.30-65536	150.00	3	188	204	211	197	218	207	223	261	216	237
HILA5-1024-2.83-12289	255.00	5	258	274	289	268	288	284	300	358	294	314
KCL-MLWE-0768-1.00-7681	147.00	4	149	165	167	158	179	164	180	207	173	194
KCL-MLWE-0768-2.24-7681	183.00	4	185	201	208	194	215	204	220	257	213	234
KCL-RLWE-1024-2.83-12289	255.00	5	258	274	289	268	288	284	300	358	294	314
KINDI-0768-2.29-16384	164.00	2	171	187	191	180	201	188	204	237	197	218
KINDI-1024-1.12-8192	207.00	4	221	237	248	230	251	243	259	306	253	273
KINDI-1024-2.29-16384	232.00	4	238	254	268	248	269	263	279	331	273	293
KINDI-1280-1.12-16384	251.00	5	264	280	297	274	295	291	307	367	301	322
KINDI-1536-1.12-8192	330.00	5	352	368	396	363	383	388	404	489	399	419
LAC-0512-0.71-251	128.00	1, 2	136	152	152	145	165	149	165	188	158	179
LAC-1024-0.50-251	192.00	3, 4	262	278	294	271	292	288	304	363	298	318
LAC-1024-0.71-251	256.00	5	293	309	329	303	323	323	339	407	333	353
LIMA-2p-1024-3.16-133121	208.80	3	198	214	222	207	228	218	234	274	227	248
LIMA-2p-2048-3.16-184321	444.50	4	430	446	482	440	461	473	489	596	484	505
LIMA-sp-1018-3.16-12521473	139.20	1	125	141	140	133	155	137	153	173	146	168
LIMA-sp-1306-3.16-48181249	167.80	2	153	169	171	162	183	168	184	212	177	199
LIMA-sp-1822-3.16-44802049	247.90	3	233	249	261	243	264	257	273	323	266	288
LIMA-sp-2062-3.16-16900097	303.50	4	291	307	327	302	323	321	337	405	331	353
LOTUS-0576-3.00-8192	—	1, 2	143	159	160	152	172	157	173	198	166	187
LOTUS-0704-3.00-8192	—	3, 4	180	196	203	190	210	199	215	250	208	229
LOTUS-0832-3.00-8192	—	5	219	235	246	229	249	241	257	304	251	271
LightSaber-0512-2.29-8192	115.00	1	114	130	128	123	143	125	142	158	134	155
Lizard-1024-1.12-1024	131.00	1	158	175	178	167	188	174	191	219	183	204
Lizard-1024-1.12-2048	130.00	1	126	143	142	135	155	139	155	175	148	168
Lizard-1024-1.12-2048	193.00	3	187	203	210	197	217	206	222	260	216	236
Lizard-1024-1.12-2048	195.00	3	220	236	246	229	250	242	258	304	251	272
Lizard-2048-1.12-2048	264.00	5	319	336	358	330	350	352	368	443	362	382
Lizard-2048-1.12-4096	257.00	5	264	281	297	274	295	291	308	367	301	322
MamaBear-0936-0.71-1024	219.00	4	220	236	247	230	251	243	259	306	253	273
MamaBear-0936-0.94-1024	237.00	5	239	255	269	249	269	264	280	332	273	294

Scheme	Claim	NIST	0.265β	0.265β $+16.4$	0.2975β	0.265β $+\log_2(\beta)$	$0.265\beta+16.4$ $+\log_2(8d)$	0.292β	0.292β $+16.4$	0.368β	0.292β $+\log_2(\beta)$	$0.292\beta+16.4$ $+\log_2(8d)$
NTRU LPrime-0761-0.82-4591	225.00	5	141	157	159	151	171	156	172	196	165	186
NewHope-0512-2.00-12289	101.00	1	103	119	115	111	132	113	129	143	122	143
NewHope-1024-2.00-12289	233.00	5	235	251	264	245	266	259	275	327	269	290
PapaBear-1248-0.61-1024	292.00	5	293	309	329	303	323	323	339	407	333	353
PapaBear-1248-0.87-1024	320.00	5	324	340	363	334	354	356	372	449	367	387
R EMBLEM-0512-25.00-65536	128.10	1	102	118	114	111	131	112	128	141	121	142
R EMBLEM-0512-3.00-16384	128.30	1	92	108	103	100	121	101	117	127	110	131
RLizard-1024-1.12-1024	147.00	1	223	240	245	233	253	242	258	286	251	272
RLizard-1024-1.12-2048	195.00	3	225	241	252	234	255	247	264	312	257	278
RLizard-2048-1.12-2048	291.00	3	389	405	416	398	419	412	428	468	421	442
RLizard-2048-1.12-4096	318.00	5	429	445	473	439	460	466	482	554	476	496
Saber-0768-2.29-8192	180.00	3	185	201	207	194	215	203	220	256	213	233
Titanium.KEM-1024-1.41-118273	128.00	1	168	184	188	177	198	185	201	233	194	215
Titanium.KEM-1280-1.41-430081	160.00	1	194	210	218	204	225	214	230	270	223	245
Titanium.KEM-1536-1.41-783361	192.00	3	230	246	258	240	261	254	270	320	263	285
Titanium.KEM-2048-1.41-1198081	256.00	5	314	330	352	324	345	346	362	436	356	377
Titanium.PKE-1024-1.41-86017	128.00	1	173	189	194	183	204	191	207	240	200	221
Titanium.PKE-1280-1.41-301057	160.00	1	201	217	226	211	232	222	238	279	231	252
Titanium.PKE-1536-1.41-737281	192.00	3	231	247	260	241	262	255	271	321	265	286
Titanium.PKE-2048-1.41-1198081	256.00	5	314	330	352	324	345	346	362	436	356	377
nRound2.KEM-0400-3.61-3209	74.00	1	79	95	88	87	107	87	103	109	95	115
nRound2.KEM-0486-2.18-1949	97.00	2	101	117	113	109	130	111	127	139	119	140
nRound2.KEM-0556-3.76-3343	106.00	3	116	132	129	124	145	127	144	156	136	156
nRound2.KEM-0658-1.46-1319	139.00	4, 5	144	160	161	153	173	158	175	199	167	188
nRound2.PKE-0442-1.47-2659	74.00	1	79	96	89	88	108	88	104	110	96	117
nRound2.PKE-0556-1.86-3343	97.00	2	105	122	118	114	134	116	132	144	124	145
nRound2.PKE-0576-1.27-2309	106.00	3	111	128	125	120	141	123	139	154	131	152
nRound2.PKE-0708-1.57-2837	138.00	4, 5	143	160	161	152	173	158	174	199	167	187
qTESLA-1024-8.49-8058881	128.00	1	157	173	176	166	188	173	189	218	182	203
qTESLA-2048-8.49-12681217	192.00	3	348	364	391	359	380	384	400	483	394	415
qTESLA-2048-8.49-27627521	256.00	5	326	342	366	336	357	359	375	452	369	390
uRound2.KEM-0418-4.61-4096	75.00	1	82	98	92	90	111	90	107	111	98	119
uRound2.KEM-0500-2.29-16384	74.00	1	76	93	86	84	105	84	100	105	92	113
uRound2.KEM-0522-36.95-32768	97.00	2	107	123	120	115	136	117	134	143	126	146
uRound2.KEM-0540-18.47-16384	106.00	3	113	130	127	122	142	125	141	156	133	154
uRound2.KEM-0580-4.61-32768	96.00	2	95	111	106	103	124	104	121	131	113	134
uRound2.KEM-0630-4.61-32768	106.00	3	105	121	118	114	134	116	132	145	124	145
uRound2.KEM-0676-36.95-32768	139.00	5	147	163	165	156	177	162	178	202	171	191
uRound2.KEM-0700-36.95-32768	140.00	4	152	168	170	161	181	167	183	205	176	197
uRound2.KEM-0786-4.61-32768	138.00	5	138	154	155	147	168	152	168	191	161	182
uRound2.KEM-0786-4.61-32768	139.00	4	138	154	155	147	168	152	168	191	161	182
uRound2.PKE-0420-1.12-1024	74.00	1	81	98	91	90	110	89	106	109	98	118
uRound2.PKE-0500-4.61-32768	74.00	1	77	93	86	85	106	84	101	105	93	113
uRound2.PKE-0540-4.61-8192	97.00	2	103	119	115	111	132	113	130	142	122	142
uRound2.PKE-0585-4.61-32768	96.00	2	95	112	107	104	125	105	121	132	114	134

Scheme	Claim	NIST	0.265β	0.265β $+16.4$	0.2975β	0.265β $+\log_2(\beta)$	$0.265\beta+16.4$ $+\log_2(8d)$	0.292β	0.292β $+16.4$	0.368β	0.292β $+\log_2(\beta)$	$0.292\beta+16.4$ $+\log_2(8d)$
uRound2.PKE-0586-4.61-8192	107.00	3	113	130	127	122	143	125	141	157	134	154
uRound2.PKE-0643-4.61-32768	106.00	3	107	123	120	115	136	118	134	148	126	147
uRound2.PKE-0708-18.47-32768	138.00	4, 5	144	160	161	153	173	158	175	199	167	188
uRound2.PKE-0835-2.29-32768	138.00	4	137	154	154	146	167	151	168	190	160	181
uRound2.PKE-0835-2.29-32768	138.00	5	137	154	154	146	167	151	168	190	160	181

Table 8.5: Cost of the primal attack against LWE-based schemes assuming n LWE samples using sieving. The column Scheme indicates each instantiation of a scheme using the format NAME-n-σ-q.

Scheme	Claim	NIST	0.265β	0.265β $+16.4$	0.2975β	0.265β $+\log_2(\beta)$	$0.265\beta + 16.4$ $+\log_2(8d)$	0.292β	0.292β $+16.4$	0.368β	0.292β $+\log_2(\beta)$	$0.292\beta + 16.4$ $+\log_2(8d)$
BabyBear-0624-0.79-1024	141.00	2	143	159	160	152	172	157	173	198	166	187
BabyBear-0624-1.00-1024	152.00	2	153	169	172	163	183	169	185	213	178	199
CRYSTALS-Dilithium-0768-3.74-8380417	91.00	1	91	107	103	100	121	101	117	127	109	131
CRYSTALS-Dilithium-1024-3.16-8380417	125.00	2	129	145	145	138	160	142	158	179	151	173
CRYSTALS-Dilithium-1280-2.00-8380417	158.00	3	159	175	178	168	190	175	191	221	184	206
CRYSTALS-Kyber-0512-1.58-7681	102.00	1	103	119	115	111	132	113	129	143	122	143
CRYSTALS-Kyber-0768-1.41-7681	161.00	3	163	179	183	172	193	180	196	226	189	210
CRYSTALS-Kyber-1024-1.22-7681	218.00	5	221	237	248	230	251	243	259	306	253	273
Ding Key Exchange-0512-4.19-120883	—	1	90	106	101	98	119	99	115	125	107	128
Ding Key Exchange-1024-2.60-120883	—	3, 5	190	206	214	200	221	210	226	264	219	240
EMBLEM-0611-25.00-16777216	128.30	1	69	85	78	77	99	76	92	96	84	106
EMBLEM-0770-25.00-16777216	128.30	1	90	106	101	98	120	99	115	125	107	129
FireSaber-1024-2.29-8192	245.00	5	257	273	288	267	287	283	300	357	293	314
Frodo-0640-2.75-32768	103.00	1	128	144	144	137	158	141	157	178	150	171
Frodo-0976-2.30-65536	150.00	3	188	204	211	197	218	207	223	261	216	237
HILA5-1024-2.83-12289	255.00	5	257	273	288	267	287	283	299	357	293	314
KCL-MLWE-0768-1.00-7681	147.00	4	149	165	167	158	179	164	180	207	173	194
KCL-MLWE-0768-2.24-7681	183.00	4	185	201	207	194	215	203	219	256	213	233
KCL-RLWE-1024-2.83-12289	255.00	5	257	273	288	267	287	283	299	357	293	314
KINDI-0768-2.29-16384	164.00	2	170	186	191	179	200	187	203	236	196	217
KINDI-1024-1.12-8192	207.00	4	221	237	248	230	251	243	259	306	253	273
KINDI-1024-2.29-16384	232.00	4	238	254	267	248	269	262	278	331	272	293
KINDI-1280-1.12-16384	251.00	5	264	280	297	274	295	291	307	367	301	322
KINDI-1536-1.12-8192	330.00	5	352	368	396	363	383	388	404	489	399	419
LAC-0512-0.71-251	128.00	1, 2	136	152	152	145	165	149	165	188	158	179
LAC-1024-0.50-251	192.00	3, 4	262	278	294	271	292	288	304	363	298	318
LAC-1024-0.71-251	256.00	5	293	309	329	303	323	323	339	407	333	353
LIMA-2p-1024-3.16-133121	208.80	3	196	213	220	206	227	216	233	272	226	247
LIMA-2p-2048-3.16-184321	444.50	4	429	446	482	440	461	473	489	596	484	504
LIMA-sp-1018-3.16-12521473	139.20	1	124	140	139	133	154	136	153	172	145	167
LIMA-sp-1306-3.16-48181249	167.80	2	152	169	171	162	183	168	184	211	177	199
LIMA-sp-1822-3.16-44802049	247.90	3	232	249	261	242	264	256	272	322	266	287
LIMA-sp-2062-3.16-16900097	303.50	4	291	308	327	301	323	321	337	404	331	352
LOTUS-0576-3.00-8192	—	1, 2	141	157	159	151	171	156	172	196	165	186
LOTUS-0704-3.00-8192	—	3, 4	179	195	201	189	209	197	213	249	207	227
LOTUS-0832-3.00-8192	—	5	218	234	244	227	248	240	256	302	249	270
LightSaber-0512-2.29-8192	115.00	1	113	130	127	122	143	125	141	157	134	154
Lizard-1024-1.12-1024	131.00	1	158	175	178	167	188	174	191	219	183	204
Lizard-1024-1.12-2048	130.00	1	126	143	142	135	155	139	155	175	148	168
Lizard-1024-1.12-2048	193.00	3	187	203	210	197	217	206	222	260	216	236
Lizard-1024-1.12-2048	195.00	3	220	236	246	229	250	242	258	304	251	272
Lizard-2048-1.12-2048	264.00	5	319	336	358	330	350	352	368	443	362	382
Lizard-2048-1.12-4096	257.00	5	264	281	297	274	295	291	308	367	301	322
MamaBear-0936-0.71-1024	219.00	4	220	236	247	230	251	243	259	306	253	273
MamaBear-0936-0.94-1024	237.00	5	239	255	269	249	269	264	280	332	273	294

Scheme	Claim	NIST	0.265β	0.265β $+16.4$	0.2975β	0.265β $+\log_2(\beta)$	$0.265\beta+16.4$ $+\log_2(8d)$	0.292β	0.292β $+16.4$	0.368β	0.292β $+\log_2(\beta)$	$0.292\beta+16.4$ $+\log_2(8d)$
NTRU LPrime-0761-0.82-4591	225.00	5	141	157	159	151	171	156	172	196	165	186
NewHope-0512-2.00-12289	101.00	1	103	119	115	111	132	113	129	143	122	143
NewHope-1024-2.00-12289	233.00	5	235	251	264	245	266	259	275	327	269	290
PapaBear-1248-0.61-1024	292.00	5	293	309	329	303	323	323	339	407	333	353
PapaBear-1248-0.87-1024	320.00	5	324	340	363	334	354	356	372	449	367	387
R EMBLEM-0512-25.00-65536	128.10	1	102	118	114	111	131	112	128	141	121	142
R EMBLEM-0512-3.00-16384	128.30	1	92	108	103	100	121	101	117	127	110	131
RLizard-1024-1.12-1024	147.00	1	223	240	245	233	253	242	258	286	251	272
RLizard-1024-1.12-2048	195.00	3	225	241	252	234	255	247	264	312	257	278
RLizard-2048-1.12-2048	291.00	3	389	405	416	398	419	412	428	468	421	442
RLizard-2048-1.12-4096	318.00	5	429	445	473	439	460	466	482	554	476	496
Saber-0768-2.29-8192	180.00	3	184	201	207	194	214	203	219	256	212	233
Titanium.KEM-1024-1.41-118273	128.00	1	168	184	188	177	198	185	201	233	194	215
Titanium.KEM-1280-1.41-430081	160.00	1	194	210	218	204	225	214	230	270	223	245
Titanium.KEM-1536-1.41-783361	192.00	3	230	246	258	240	261	254	270	320	263	285
Titanium.KEM-2048-1.41-1198081	256.00	5	314	330	352	324	345	346	362	436	356	377
Titanium.PKE-1024-1.41-86017	128.00	1	173	189	194	183	204	191	207	240	200	221
Titanium.PKE-1280-1.41-301057	160.00	1	201	217	226	211	232	222	238	279	231	252
Titanium.PKE-1536-1.41-737281	192.00	3	231	247	260	241	262	255	271	321	265	286
Titanium.PKE-2048-1.41-1198081	256.00	5	314	330	352	324	345	346	362	436	356	377
nRound2.KEM-0400-3.61-3209	74.00	1	79	95	88	87	107	87	103	109	95	115
nRound2.KEM-0486-2.18-1949	97.00	2	101	117	113	109	130	111	127	139	119	140
nRound2.KEM-0556-3.76-3343	106.00	3	116	132	129	124	145	127	144	156	136	156
nRound2.KEM-0658-1.46-1319	139.00	4, 5	144	160	161	153	173	158	175	199	167	188
nRound2.PKE-0442-1.47-2659	74.00	1	79	96	89	88	108	88	104	110	96	117
nRound2.PKE-0556-1.86-3343	97.00	2	105	122	118	114	134	116	132	144	124	145
nRound2.PKE-0576-1.27-2309	106.00	3	111	128	125	120	141	123	139	154	131	152
nRound2.PKE-0708-1.57-2837	138.00	4, 5	143	160	161	152	173	158	174	199	167	187
qTESLA-1024-8.49-8058881	128.00	1	154	170	173	163	184	170	186	214	179	200
qTESLA-2048-8.49-12681217	192.00	3	344	360	387	355	376	380	396	478	390	411
qTESLA-2048-8.49-27627521	256.00	5	322	338	362	333	354	355	371	448	366	387
uRound2.KEM-0418-4.61-4096	75.00	1	82	98	92	90	111	90	107	111	98	119
uRound2.KEM-0500-2.29-16384	74.00	1	76	93	86	84	105	84	100	105	92	113
uRound2.KEM-0522-36.95-32768	97.00	2	107	123	120	115	136	117	134	143	126	146
uRound2.KEM-0540-18.47-16384	106.00	3	113	130	127	122	142	125	141	156	133	154
uRound2.KEM-0580-4.61-32768	96.00	2	95	111	106	103	124	104	121	131	113	134
uRound2.KEM-0630-4.61-32768	106.00	3	105	121	118	114	134	116	132	145	124	145
uRound2.KEM-0676-36.95-32768	139.00	5	147	163	165	156	177	162	178	202	171	191
uRound2.KEM-0700-36.95-32768	140.00	4	152	168	170	161	181	167	183	205	176	197
uRound2.KEM-0786-4.61-32768	138.00	5	138	154	155	147	168	152	168	191	161	182
uRound2.KEM-0786-4.61-32768	139.00	4	138	154	155	147	168	152	168	191	161	182
uRound2.PKE-0420-1.12-1024	74.00	1	81	98	91	90	110	89	106	109	98	118
uRound2.PKE-0500-4.61-32768	74.00	1	77	93	86	85	106	84	101	105	93	113
uRound2.PKE-0540-4.61-8192	97.00	2	103	119	115	111	132	113	130	142	122	142
uRound2.PKE-0585-4.61-32768	96.00	2	95	112	107	104	125	105	121	132	114	134

Scheme	Claim	NIST	0.265β	0.265β $+16.4$	0.2975β	0.265β $+\log_2(\beta)$	$0.265\beta + 16.4$ $+\log_2(8d)$	0.292β	0.292β $+16.4$	0.368β	0.292β $+\log_2(\beta)$	$0.292\beta + 16.4$ $+\log_2(8d)$
uRound2.PKE-0586-4.61-8192	107.00	3	113	130	127	122	143	125	141	157	134	154
uRound2.PKE-0643-4.61-32768	106.00	3	107	123	120	115	136	118	134	148	126	147
uRound2.PKE-0708-18.47-32768	138.00	4, 5	144	160	161	153	173	158	175	199	167	188
uRound2.PKE-0835-2.29-32768	138.00	4	137	154	154	146	167	151	168	190	160	181
uRound2.PKE-0835-2.29-32768	138.00	5	137	154	154	146	167	151	168	190	160	181

Table 8.6: Cost of the primal attack against LWE-based schemes assuming $2n$ LWE samples using sieving. The column Scheme indicates each instantiation of a scheme using the format NAME-n-σ-q.

Scheme	Claim	NIST	0.265β	0.265β $+16.4$	0.2975β	0.265β $+\log_2(\beta)$	$0.265\beta+16.4$ $+\log_2(8d)$	0.292β	0.292β $+16.4$	0.368β	0.292β $+\log_2(\beta)$	$0.292\beta+16.4$ $+\log_2(8d)$
Falcon-0512-4.05-12289	103.00	1	128	144	144	137	158	141	157	178	150	171
Falcon-0768-4.05-18433	172.00	2, 3	193	209	217	203	223	213	229	268	223	243
Falcon-1024-2.87-12289	230.00	4, 5	259	275	291	269	289	285	301	359	295	316
NTRU HRSS-0700-0.79-8192	123.00	1	123	139	138	132	153	136	152	171	145	165
NTRUEncrypt-0443-0.80-2048	84.00	1	85	101	95	93	114	93	109	117	101	123
NTRUEncrypt-0743-0.82-2048	159.00	1, 2, 3, 4, 5	159	175	179	169	189	175	191	221	185	205
NTRUEncrypt-1024-724.00-1073750017	198.00	4, 5	248	264	279	258	279	274	290	345	283	304
S/L NTRU Prime-0761-0.82-4591	248.00	5	140	156	158	149	170	155	171	195	164	184
pqNTRUsign-1024-0.70-65537	149.00	1, 2, 3, 4, 5	152	168	171	162	183	168	184	211	177	198

Table 8.7: Cost of the primal attack against NTRU-based schemes using sieving. The column Scheme indicates each instantiation of a scheme using the format NAME-n-σ-q, where the equivalent LWE values are provided as seen in Section 8.4.

Scheme	Claim	NIST	$\frac{1}{2}(0.187\beta \log_2 \beta$ $-1.019\beta + 16.1)$	$0.125\beta \log_2 \beta 5$ $-0.755\beta + 2.25$	$0.187\beta \log_2 \beta$ $-1.019\beta + 16.1$	$0.000784\beta^2 + 0.366\beta$ $-0.9 + \log_2(8d)$
BabyBear-0624-0.79-1024	141.00	2	190	204	380	436
BabyBear-0624-1.00-1024	152.00	2	210	227	420	487
CRYSTALS-Dilithium-0768-3.74-8380417	91.00	1	106	106	211	236
CRYSTALS-Dilithium-1024-3.16-8380417	125.00	2	168	178	335	381
CRYSTALS-Dilithium-1280-2.00-8380417	158.00	3	221	240	441	516
CRYSTALS-Kyber-0512-1.58-7681	102.00	1	122	125	244	273
CRYSTALS-Kyber-0768-1.41-7681	161.00	3	228	248	456	535
CRYSTALS-Kyber-1024-1.22-7681	218.00	5	340	381	679	861
Ding Key Exchange-0512-4.19-120883	—	1	105	105	210	234
Ding Key Exchange-1024-2.60-120883	—	3, 5	281	310	561	683
EMBLEM-0611-25.00-16777216	128.30	1	71	67	142	163
EMBLEM-0770-25.00-16777216	128.30	1	102	101	203	227
FireSaber-1024-2.29-8192	245.00	5	414	469	828	1105
Frodo-0640-2.75-32768	103.00	1	167	176	333	377
Frodo-0976-2.30-65536	150.00	3	275	304	549	666
HILA5-1024-2.83-12289	255.00	5	416	471	832	1110
KCL-MLWE-0768-1.00-7681	147.00	4	202	218	404	467
KCL-MLWE-0768-2.24-7681	183.00	4	269	297	538	650
KCL-RLWE-1024-2.83-12289	255.00	5	416	471	832	1110
KINDI-0768-2.29-16384	164.00	2	242	265	484	573
KINDI-1024-1.12-8192	207.00	4	340	381	679	861
KINDI-1024-2.29-16384	232.00	4	376	424	751	977
KINDI-1280-1.12-16384	251.00	5	429	487	858	1156
KINDI-1536-1.12-8192	330.00	5	622	718	1243	1882
LAC-0512-0.71-251	128.00	1, 2	178	190	356	405
LAC-1024-0.50-251	192.00	3, 4	424	481	847	1137
LAC-1024-0.71-251	256.00	5	492	562	983	1377
LIMA-2p-1024-3.16-133121	208.80	3	294	326	587	722
LIMA-2p-2048-3.16-184321	444.50	4	800	933	1599	2665
LIMA-sp-1018-3.16-12521473	139.20	1	159	167	317	358
LIMA-sp-1306-3.16-48181249	167.80	2	209	225	417	484
LIMA-sp-1822-3.16-44802049	247.90	3	364	410	728	940
LIMA-sp-2062-3.16-16900097	303.50	4	488	557	975	1364
LOTUS-0576-3.00-8192	—	1, 2	191	205	381	437
LOTUS-0704-3.00-8192	—	3, 4	261	287	521	625
LOTUS-0832-3.00-8192	—	5	336	376	672	849
LightSaber-0512-2.29-8192	115.00	1	141	146	281	315
Lizard-1024-1.12-1024	131.00	1	219	237	372	391
Lizard-1024-1.12-2048	130.00	1	162	170	322	362
Lizard-1024-1.12-2048	193.00	3	273	302	480	505
Lizard-1024-1.12-2048	195.00	3	318	336	480	505
Lizard-2048-1.12-2048	264.00	5	533	552	695	720
Lizard-2048-1.12-4096	257.00	5	430	488	664	689
MamaBear-0936-0.71-1024	219.00	4	339	380	678	859
MamaBear-0936-0.94-1024	237.00	5	378	425	755	982

Scheme	Claim	NIST	$\frac{1}{2}(0.187\beta\log_2\beta$ $-1.019\beta+16.1)$	$0.125\beta\log_2\beta5$ $-0.755\beta+2.25$	$0.187\beta\log_2\beta$ $-1.019\beta+16.1$	$0.000784\beta^2+0.366\beta$ $-0.9+\log_2(8d)$
NTRU LPrime-0761-0.82-4591	225.00	5	189	202	365	398
NewHope-0512-2.00-12289	101.00	1	122	125	244	273
NewHope-1024-2.00-12289	233.00	5	369	416	738	955
PapaBear-1248-0.61-1024	292.00	5	491	561	981	1375
PapaBear-1248-0.87-1024	320.00	5	558	641	1115	1627
R EMBLEM-0512-25.00-65536	128.10	1	121	123	242	270
R EMBLEM-0512-3.00-16384	128.30	1	105	105	210	234
RLizard-1024-1.12-1024	147.00	1	272	276	370	390
RLizard-1024-1.12-2048	195.00	3	346	378	570	609
RLizard-2048-1.12-2048	291.00	3	466	476	593	615
RLizard-2048-1.12-4096	318.00	5	594	623	802	837
Saber-0768-2.29-8192	180.00	3	269	296	537	648
Titanium.KEM-1024-1.41-118273	128.00	1	237	258	473	559
Titanium.KEM-1280-1.41-430081	160.00	1	287	318	574	702
Titanium.KEM-1536-1.41-783361	192.00	3	359	404	718	923
Titanium.KEM-2048-1.41-1198081	256.00	5	537	616	1073	1547
Titanium.PKE-1024-1.41-86017	128.00	1	247	271	494	587
Titanium.PKE-1280-1.41-301057	160.00	1	301	334	601	742
Titanium.PKE-1536-1.41-737281	192.00	3	361	406	722	930
Titanium.PKE-2048-1.41-1198081	256.00	5	537	616	1073	1547
nRound2.KEM-0400-3.61-3209	74.00	1	84	79	133	152
nRound2.KEM-0486-2.18-1949	97.00	2	117	116	187	206
nRound2.KEM-0556-3.76-3343	106.00	3	133	130	196	215
nRound2.KEM-0658-1.46-1319	139.00	4, 5	186	190	286	306
nRound2.PKE-0442-1.47-2659	74.00	1	85	80	134	153
nRound2.PKE-0556-1.86-3343	97.00	2	120	117	181	199
nRound2.PKE-0576-1.27-2309	106.00	3	134	134	211	230
nRound2.PKE-0708-1.57-2837	138.00	4, 5	187	193	292	313
qTESLA-1024-8.49-8058881	128.00	1	217	235	433	506
qTESLA-2048-8.49-12681217	192.00	3	612	707	1224	1847
qTESLA-2048-8.49-27627521	256.00	5	563	647	1125	1649
uRound2.KEM-0418-4.61-4096	75.00	1	86	80	131	150
uRound2.KEM-0500-2.29-16384	74.00	1	80	75	126	145
uRound2.KEM-0522-36.95-32768	97.00	2	119	114	173	192
uRound2.KEM-0540-18.47-16384	106.00	3	134	132	204	223
uRound2.KEM-0580-4.61-32768	96.00	2	109	110	188	207
uRound2.KEM-0630-4.61-32768	106.00	3	126	128	213	232
uRound2.KEM-0676-36.95-32768	139.00	5	187	189	278	297
uRound2.KEM-0700-36.95-32768	140.00	4	187	188	271	290
uRound2.KEM-0786-4.61-32768	138.00	5	181	188	294	314
uRound2.KEM-0786-4.61-32768	139.00	4	181	188	294	314
uRound2.PKE-0420-1.12-1024	74.00	1	84	78	126	145
uRound2.PKE-0500-4.61-32768	74.00	1	80	75	126	146
uRound2.PKE-0540-4.61-8192	97.00	2	120	118	187	206
uRound2.PKE-0585-4.61-32768	96.00	2	110	110	184	203

Scheme	Claim	NIST	$\frac{1}{2}(0.187\beta\log_2\beta$ $-1.019\beta+16.1)$	$0.125\beta\log_2\beta5$ $-0.755\beta+2.25$	$0.187\beta\log_2\beta$ $-1.019\beta+16.1$	$0.000784\beta^2+0.366\beta$ $-0.9+\log_2(8d)$
uRound2.PKE-0586-4.61-8192	107.00	3	136	135	210	229
uRound2.PKE-0643-4.61-32768	106.00	3	128	128	205	224
uRound2.PKE-0708-18.47-32768	138.00	4, 5	188	194	294	313
uRound2.PKE-0835-2.29-32768	138.00	4	180	189	298	320
uRound2.PKE-0835-2.29-32768	138.00	5	180	189	298	320

Table 8.8: Cost of the primal attack against LWE-based schemes assuming n LWE samples using enumeration. The column Scheme indicates each instantiation of a scheme using the format NAME-n-σ-q.

Scheme	Claim	NIST	$\frac{1}{2}(0.187\beta\log_2\beta$ $-1.019\beta+16.1)$	$0.125\beta\log_2\beta5$ $-0.755\beta+2.25$	$0.187\beta\log_2\beta$ $-1.019\beta+16.1$	$0.000784\beta^2+0.366\beta$ $-0.9+\log_2(8d)$
BabyBear-0624-0.79-1024	141.00	2	190	204	380	436
BabyBear-0624-1.00-1024	152.00	2	210	227	420	487
CRYSTALS-Dilithium-0768-3.74-8380417	91.00	1	104	104	208	233
CRYSTALS-Dilithium-1024-3.16-8380417	125.00	2	167	177	334	379
CRYSTALS-Dilithium-1280-2.00-8380417	158.00	3	220	239	440	515
CRYSTALS-Kyber-0512-1.58-7681	102.00	1	122	125	244	273
CRYSTALS-Kyber-0768-1.41-7681	161.00	3	228	248	456	535
CRYSTALS-Kyber-1024-1.22-7681	218.00	5	340	381	679	861
Ding Key Exchange-0512-4.19-120883	—	1	102	101	203	227
Ding Key Exchange-1024-2.60-120883	—	3, 5	280	309	559	680
EMBLEM-0611-25.00-16777216	128.30	1	71	66	141	162
EMBLEM-0770-25.00-16777216	128.30	1	102	101	203	227
FireSaber-1024-2.29-8192	245.00	5	414	469	828	1105
Frodo-0640-2.75-32768	103.00	1	165	174	329	372
Frodo-0976-2.30-65536	150.00	3	275	304	549	666
HILA5-1024-2.83-12289	255.00	5	414	469	828	1105
KCL-MLWE-0768-1.00-7681	147.00	4	202	218	404	467
KCL-MLWE-0768-2.24-7681	183.00	4	269	296	537	648
KCL-RLWE-1024-2.83-12289	255.00	5	414	469	828	1105
KINDI-0768-2.29-16384	164.00	2	241	263	481	569
KINDI-1024-1.12-8192	207.00	4	340	381	679	861
KINDI-1024-2.29-16384	232.00	4	375	423	750	975
KINDI-1280-1.12-16384	251.00	5	429	487	858	1156
KINDI-1536-1.12-8192	330.00	5	622	718	1243	1882
LAC-0512-0.71-251	128.00	1, 2	178	190	356	405
LAC-1024-0.50-251	192.00	3, 4	424	481	847	1137
LAC-1024-0.71-251	256.00	5	492	562	983	1377
LIMA-2p-1024-3.16-133121	208.80	3	291	323	582	714
LIMA-2p-2048-3.16-184321	444.50	4	799	932	1598	2662
LIMA-sp-1018-3.16-12521473	139.20	1	157	166	314	355
LIMA-sp-1306-3.16-48181249	167.80	2	208	225	416	483
LIMA-sp-1822-3.16-44802049	247.90	3	363	409	726	937
LIMA-sp-2062-3.16-16900097	303.50	4	487	556	973	1362
LOTUS-0576-3.00-8192	—	1, 2	189	202	377	431
LOTUS-0704-3.00-8192	—	3, 4	258	284	516	618
LOTUS-0832-3.00-8192	—	5	333	373	666	841
LightSaber-0512-2.29-8192	115.00	1	140	145	279	313
Lizard-1024-1.12-1024	131.00	1	219	237	372	391
Lizard-1024-1.12-2048	130.00	1	162	170	322	362
Lizard-1024-1.12-2048	193.00	3	273	302	480	505
Lizard-1024-1.12-2048	195.00	3	318	336	480	505
Lizard-2048-1.12-2048	264.00	5	533	552	695	720
Lizard-2048-1.12-4096	257.00	5	430	488	664	689
MamaBear-0936-0.71-1024	219.00	4	339	380	678	859
MamaBear-0936-0.94-1024	237.00	5	378	425	755	982

Scheme	Claim	NIST	$\frac{1}{2}(0.187\beta\log_2\beta$ $-1.019\beta+16.1)$	$0.125\beta\log_2\beta5$ $-0.755\beta+2.25$	$0.187\beta\log_2\beta$ $-1.019\beta+16.1$	$0.000784\beta^2+0.366\beta$ $-0.9+\log_2(8d)$
NTRU LPrime-0761-0.82-4591	225.00	5	189	202	365	398
NewHope-0512-2.00-12289	101.00	1	122	125	244	273
NewHope-1024-2.00-12289	233.00	5	369	416	738	955
PapaBear-1248-0.61-1024	292.00	5	491	561	981	1375
PapaBear-1248-0.87-1024	320.00	5	558	641	1115	1627
R EMBLEM-0512-25.00-65536	128.10	1	121	123	242	270
R EMBLEM-0512-3.00-16384	128.30	1	105	105	210	234
RLizard-1024-1.12-1024	147.00	1	272	276	370	390
RLizard-1024-1.12-2048	195.00	3	346	378	570	609
RLizard-2048-1.12-2048	291.00	3	466	476	593	615
RLizard-2048-1.12-4096	318.00	5	594	623	802	837
Saber-0768-2.29-8192	180.00	3	268	295	535	645
Titanium.KEM-1024-1.41-118273	128.00	1	237	258	473	559
Titanium.KEM-1280-1.41-430081	160.00	1	287	318	574	702
Titanium.KEM-1536-1.41-783361	192.00	3	359	404	718	923
Titanium.KEM-2048-1.41-1198081	256.00	5	537	616	1073	1547
Titanium.PKE-1024-1.41-86017	128.00	1	247	271	494	587
Titanium.PKE-1280-1.41-301057	160.00	1	301	334	601	742
Titanium.PKE-1536-1.41-737281	192.00	3	361	406	722	930
Titanium.PKE-2048-1.41-1198081	256.00	5	537	616	1073	1547
nRound2.KEM-0400-3.61-3209	74.00	1	84	79	133	152
nRound2.KEM-0486-2.18-1949	97.00	2	117	116	187	206
nRound2.KEM-0556-3.76-3343	106.00	3	133	130	196	215
nRound2.KEM-0658-1.46-1319	139.00	4, 5	186	190	286	306
nRound2.PKE-0442-1.47-2659	74.00	1	85	80	134	153
nRound2.PKE-0556-1.86-3343	97.00	2	120	117	181	199
nRound2.PKE-0576-1.27-2309	106.00	3	134	134	211	230
nRound2.PKE-0708-1.57-2837	138.00	4, 5	187	193	292	313
qTESLA-1024-8.49-8058881	128.00	1	211	228	422	490
qTESLA-2048-8.49-12681217	192.00	3	604	697	1208	1813
qTESLA-2048-8.49-27627521	256.00	5	555	638	1110	1619
uRound2.KEM-0418-4.61-4096	75.00	1	86	80	131	150
uRound2.KEM-0500-2.29-16384	74.00	1	80	75	126	145
uRound2.KEM-0522-36.95-32768	97.00	2	119	114	173	192
uRound2.KEM-0540-18.47-16384	106.00	3	134	132	204	223
uRound2.KEM-0580-4.61-32768	96.00	2	109	110	188	207
uRound2.KEM-0630-4.61-32768	106.00	3	126	128	213	232
uRound2.KEM-0676-36.95-32768	139.00	5	187	189	278	297
uRound2.KEM-0700-36.95-32768	140.00	4	187	188	271	290
uRound2.KEM-0786-4.61-32768	138.00	5	181	188	294	314
uRound2.KEM-0786-4.61-32768	139.00	4	181	188	294	314
uRound2.PKE-0420-1.12-1024	74.00	1	84	78	126	145
uRound2.PKE-0500-4.61-32768	74.00	1	80	75	126	146
uRound2.PKE-0540-4.61-8192	97.00	2	120	118	187	206
uRound2.PKE-0585-4.61-32768	96.00	2	110	110	184	203

Scheme	Claim	NIST	$\frac{1}{2}(0.187\beta \log_2 \beta$ $-1.019\beta + 16.1)$	$0.125\beta \log_2 \beta 5$ $-0.755\beta + 2.25$	$0.187\beta \log_2 \beta$ $-1.019\beta + 16.1$	$0.000784\beta^2 + 0.366\beta$ $-0.9 + \log_2(8d)$
uRound2.PKE-0586-4.61-8192	107.00	3	136	135	210	229
uRound2.PKE-0643-4.61-32768	106.00	3	128	128	205	224
uRound2.PKE-0708-18.47-32768	138.00	4, 5	188	194	294	313
uRound2.PKE-0835-2.29-32768	138.00	4	180	189	298	320
uRound2.PKE-0835-2.29-32768	138.00	5	180	189	298	320

Table 8.9: Cost of the primal attack against LWE-based schemes assuming $2n$ LWE samples using enumeration. The column Scheme indicates each instantiation of a scheme using the format NAME-n-σ-q.

Scheme	Claim	NIST	$\frac{1}{2}(0.187\beta\log_2\beta$ $-1.019\beta+16.1)$	$0.125\beta\log_2\beta5$ $-0.755\beta+2.25$	$0.187\beta\log_2\beta$ $-1.019\beta+16.1$	$0.000784\beta^2+0.366\beta$ $-0.9+\log_2(8d)$
Falcon-0512-4.05-12289	103.00	1	165	175	330	373
Falcon-0768-4.05-18433	172.00	2, 3	286	316	571	697
Falcon-1024-2.87-12289	230.00	4, 5	418	474	836	1118
NTRU HRSS-0700-0.79-8192	123.00	1	157	165	313	350
NTRUEncrypt-0443-0.80-2048	84.00	1	93	92	186	208
NTRUEncrypt-0743-0.82-2048	159.00	1, 2, 3, 4, 5	221	240	441	516
NTRUEncrypt-1024-724.00-1073750017	198.00	4, 5	396	448	792	1043
S/L NTRU Prime-0761-0.82-4591	248.00	5	187	200	370	410
pqNTRUsign-1024-0.70-65537	149.00	1, 2, 3, 4, 5	208	225	416	480

Table 8.10: Cost of the primal attack against NTRU-based schemes using enumeration. The column Scheme indicates each instantiation of a scheme using the format NAME-n-σ-q, where the equivalent LWE values are provided as seen in Section 8.4.

In the following, we illuminate some of the choices and assumptions we made to arrive at our estimates.

Secret distributions. Many submissions consider uniform, bounded uniform, or sparse bounded uniform secret distributions. In the case of Lizard [CPL+17], LWE secrets are drawn from the distribution $\mathcal{ZO}_n(\rho)$ for some $0 < \rho < 1$. $\mathcal{ZO}_n(\rho)$ is the distribution over $\{-1, 0, 1\}^n$ where each component s_i of a vector $\mathbf{s} \leftarrow \mathcal{ZO}_n(\rho)$ satisfies $\Pr[s_i = 1] = \Pr[s_i = -1] = \rho/2$ and $\Pr[s_i = 0] = 1 - \rho$. We model this distribution as a fixed weight bounded uniform distribution, where the Hamming weight h matches the expected number of non-zero components of an element drawn from $\mathcal{ZO}_n(\rho)$.

Error distributions. While the LWE estimator assumes the distribution of error vector components to be a discrete Gaussian, many submissions use alternatives. Binomial distributions are treated as discrete Gaussians with the corresponding standard deviation. Similarly, bounded uniform distributions $U_{[a,b]}$ are also treated as discrete Gaussians with standard deviation $\sqrt{\mathbb{V}_{U_{[a,b]}}[\mathbf{e}_i]}$, where $\mathbb{V}$ denotes the variance of the distribution. In the case of LWR, we use a standard deviation of $\sqrt{\frac{(q/p)^2-1}{12}}$, following [Ngu18].

Success probability. The LWE estimator supports defining a target success probability for the primal. The only proposal we found that explicitly uses this functionality is LIMA [SAL+17], which chooses to use a target success probability of 51%. For our estimates we imposed this to be the estimator's default 99% for all schemes, since it seems to make little to no difference for the final estimates as amplification in this range is rather cheap.

Known limitations. While the estimator can scale short secret vectors with entries sampled from a bounded uniform distribution, it does not attempt to shift secret vectors whose entries have unbalanced bounds to optimize the scaling. Similarly, it does not attempt to guess entries of such secrets to reduce the dimension. We note, however, that only the KINDI submission [Ban17] uses such a secret vector distribution. In this case, the deviation from a distribution centered at zero is small and we thus ignore it.

NTRU. For estimating NTRU-based schemes, we also utilize the LWE estimator to evaluate the primal attack (and its improvements) on NTRU. In particular, we treat the NTRU problem as a uSVP instance and account for the presence of rotations by amplifying the success probability p of dropping the correct columns of the short vector to $1 - (1 - p)^k$, where k is the number of rotations. Further speedups as presented in [KF17] which exploit the structure of the NTRU lattice do not affect the schemes submitted to NIST and are therefore not considered. In more detail, let $(\mathbf{f}, \mathbf{g}) \in \mathbb{Z}^{2n}$ be the secret NTRU vector. We treat $\mathbf{f}$ as the LWE secret and $\mathbf{g}$ as the LWE error (or vice versa, as their roles can be swapped). The LWE secret dimension n is set to the degree of the NTRU polynomial ϕ. The standard deviation of the LWE error distribution is set to $\|\mathbf{g}\| / \sqrt{n}$. The LWE modulus q is set to the NTRU modulus. The secret distribution is set to the distribution of $\mathbf{f}$. We limit the number of LWE samples to n. The estimator is set to consider the n rotations of $\mathbf{g}$ when estimating the cost of the primal attack on NTRU.

Beyond key recovery. We consider key recovery attacks on all schemes. In the case of LWE-based schemes, we also consider message recovery attacks by setting the number of samples to be $m = 2n$ and trying to recover the ephemeral secret key set as part of key encapsulation. A straight-forward primal uSVP message recovery attack for NTRU-based schemes as described in Footnote 2 of [SHRS17] is not expected to perform better than the primal uSVP key recovery attack, and is therefore omitted in this work.

In the case of signatures, it is also possible to attempt forgery attacks. All four lattice-based signatures schemes submitted to the NIST process claim that the problem of forging a signature is strictly harder than that of recovering the signing key. In particular, Dilithium and pqNTRUSign provide analyses which explicitly determine that larger BKZ block sizes are required for signature forgery than key recovery. Falcon argues similarly without giving explicit block sizes and qTESLA presents a tight reduction in the QROM from the RLWE problem to signature forgery, in particular from exactly the RLWE problem one would have to solve to recover the signing key. As such, since one may trivially forge signatures given possession of the signing key, forgery attacks are not considered further in their security analyses.

Several complications arise when attempting to estimate the complexity of signature forgery compared to key recovery. These include the requirement for a signature forging adversary to satisfy the conditions in the Verify algorithm, which for the four proposed schemes consists of solving different, sometimes not well studied, problems, such as the SIS problem in the ℓ_∞-norm for Dilithium and qTESLA and the modular equivalence required between the message and signature in pqNTRUSign. In attempts to determine how one might straightforwardly estimate the complexity of signature forgery against the Dilithium and qTESLA schemes, custom analysis was required which was heavily dependent on the intricacies of the scheme in question, ruling out a scheme-agnostic approach to security estimation in the case of signature forgeries.

8.4.1 Discussion

Our data highlights that cost models for lattice reduction do not necessarily preserve the ordering of the schemes under consideration. That is, under one cost model some scheme A can be considered harder to break than a scheme B, while under another cost model scheme B appears harder to break.

An example for the schemes EMBLEM and uRound2.KEM was highlighted in [Ber18]. Consider the EMBLEM parameter set with $n = 611$ and the uRound2.KEM parameter set with $n = 500$. In the 0.292β cost model, the cost of the primal attack for EMBLEM-611 is estimated as[17] 76 and for uRound2.KEM-500 as 84. For the same attack in the $0.187\beta \log_2 \beta - 1.019\beta + 16.1$ cost model, the cost is estimated for EMBLEM-611 as 142 and for uRound2.KEM-500 as 126. Similar swaps can be observed for several other pairs of schemes and cost models. In most cases the estimated securities of the two schemes are very close to each other (differing by, say, 1 or 2) and thus a swap of ordering does not fundamentally alter our understanding of their relative security as these estimates are typically derived by heuristically searching through the space of possible parameters and computing with limited precision. In some cases, though, such as the one highlighted in [Ber18], the differences in security estimates can be significant. There are two classes of such cases as described in the following.

Sparse secrets. The first class of cases involves instances with sparse secrets. The LWE estimator applies guessing strategies (cf. [Alb17]) when costing the primal attack. The basic idea is that for a sparse secret, many of the entries of the secret vector are zero, and hence can be ignored. We guess τ entries to be zero, and drop the corresponding columns from the attack lattice. In dropping τ columns from a n-dimensional LWE instance, we obtain a $(n - \tau)$-dimensional LWE instance with a more dense secret distribution, where the density depends on the choice of τ and the original value of the Hamming weight h. On the one hand, there is a probability of failure when guessing which columns to drop. On the other hand there may exist a τ for which the $(n - \tau)$-dimensional LWE instance is easier to solve, and in particular requires a smaller BKZ blocksize β. The trade-off between running BKZ on smaller lattices and having to run it multiple times can correspond to an overall lower expected attack cost. The probability of failure when guessing secret entries does not depend on the cost model, but rather on the weight and dimension of the secret, making this kind of attack more effective for very sparse secrets. In the case of comparing an enumeration cost model versus a sieving one, we have that the cost of enumeration is fitted as $2^{\Theta(\beta \log_2 \beta)}$ or $2^{\Theta(\beta^2)}$ whereas the cost of sieving is $2^{\Theta(\beta)}$. The steeper curve for enumeration means that as we increase τ, and hence decrease β, savings are potentially larger, justifying a larger number τ of entries

[17]Any discrepancies in value from those cited in [Ber18] are due to rounding introduced to the estimator output since.

guessed. Concretely, the computed optimal guessing dimension τ can be much larger than in the sieving regime. This phenomenon can also be observed when comparing two different sieving models or two different enumeration models.

In Figure 8.1, we illustrate this for the EMBLEM and uRound2.KEM example. EMBLEM does not have a sparse secret, while uRound2.KEM does. For EMBLEM the best guessing dimension, giving the lowest overall cost, is $\tau = 0$ in both cost models. For uRound2.KEM, we see that the optimal guessing dimension varies depending on the cost model. In the 0.292β cost model, the lowest overall expected cost is achieved for $\tau = 1$ while in the $0.187\beta \log_2 \beta - 1.019\beta + 16.1$ model the optimal choice is $\tau = 197$.

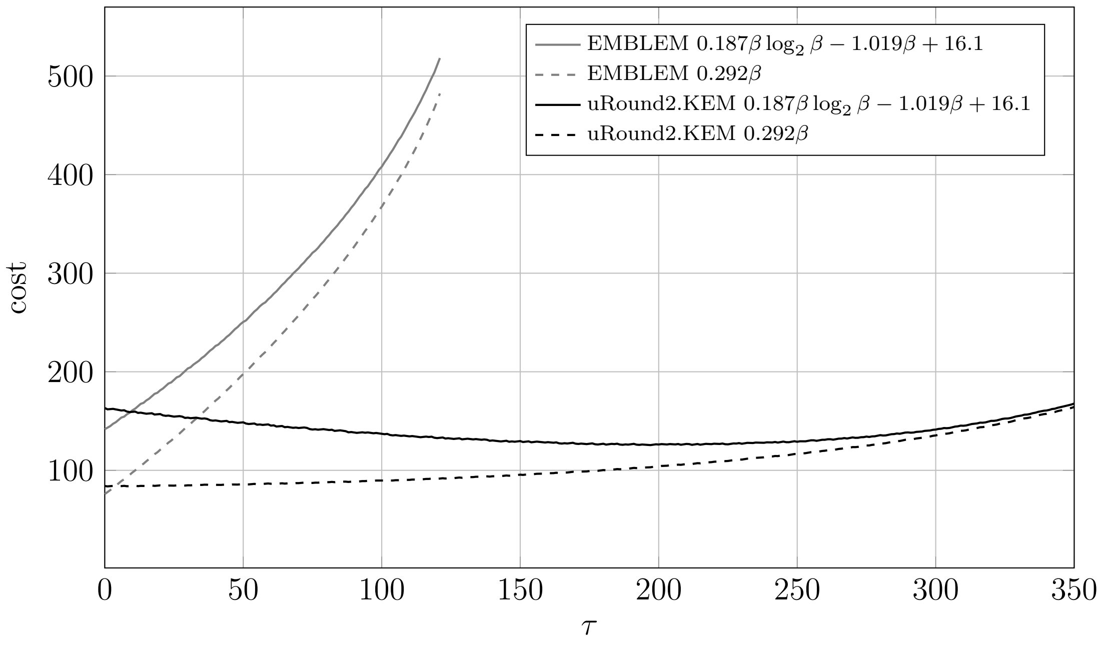

Figure 8.1: Estimates of the cost of the primal attack when guessing τ secret entries for the schemes EMBLEM-611 and uRound2.KEM-500 using the sieving-based cost model 0.292β and the enumeration-based cost model $0.187\beta \log_2 \beta - 1.019\beta + 16.1$.

Multiple hardness assumptions. Lizard (RLizard) is based on two different hardness assumptions, namely LWE (RLWE) and LWR (RLWR). Secret key recovery corresponds to the underlying LWE problem, and ephemeral key recovery corresponds to the underlying LWR problem. There are Lizard parameter sets for which ephemeral key recovery is harder than secret key recovery (i.e, the underlying LWR problem is harder than the underlying LWE problem), and there are also parameter sets for which the converse is true. To deal with this issue, for each parameter set, in each cost model, we always choose the lower of the two possible costs.

Quantum security. In [Nat16], NIST defined five security categories that schemes should target in the presence of an adversary with access to a large scale quantum computer (cf. Section 8.1). They furthermore propose as a plausible assumption that such a device would support a maximum quantum circuit depth $\texttt{MAXDEPTH} \leq 2^{96}$ (although they do not mention a preferred set of universal gates to consider). However, not all schemes take this limitation into account, and many of the submissions instead use an asymptotic quantum cost model that considers the best known (or assumed) theoretical Grover speed-up, resulting in possible overestimates of the adversary's power.

This use of quantum models introduces a further difficulty when trying to compare schemes based on the outputs of the [APS15] estimator. For example, the security definition of Category 1 requires that attacks on schemes should be as hard as AES128 key recovery. Some schemes address this by tuning their parameters to match a quantum-hardness of at least 2^{128}, in the vein of "128 bit security". On the other hand, other schemes claiming the same category match a quantum-hardness of at least 2^{64} since key recovery on AES128 can be considered as a search problem in an unstructured list of size 2^{128}, which Grover can complete in $O(2^{n/2})$ time. This results in schemes with rather different cycle counts and memory usage claiming the same security category, as can be seen from the "claimed security" column in the estimates table.

8.5 Estimates for the Quantum Hybrid Attack

In this section, we analyze two selected schemes with respect to their security against the quantum hybrid attack and compare the results to the security estimates against the primal attack provided in Section 8.4. Note that the quantum hybrid attack may be applied to more of the submitted schemes. For our analysis, we pick one scheme with particularly sparse ternary secret vectors, namely the LWR-based parameter sets of the uRound2 [GMZB+17] KEM, and one scheme with random ternary secret vectors, namely the RLWE-based parameter sets of the EMBLEM [SPL+17] KEM/PKE. For a comparison between these two schemes with respect to the primal attack, see also Section 8.4.1. When analyzing the schemes, we restricted our considerations to the case where n samples are provided. Furthermore, we restrict our analysis to the most commonly used enumeration- and quantum-sieving-based BKZ cost models, i.e., $0.187\beta \log_2 \beta - 1.019\beta + 16.1$ and 0.265β. We used Bai and Galbraith's embedding [BG14b] to embed RLWE and LWR into uSVP (ignoring the additional dimension introduced by the embedding factor and flipping the positions of the secret and error vector). We considered rescaling and dimension reducing techniques (as discussed in Section 7.2.2) and optimizing the search space according to Section 7.3. To that end, we proceeded as follows. For each combination of number of LWE/LWR samples m and relative size of the search space $|S| \,/\, |M|$, we optimized the attack parameters r (guessing dimension) and β (block size) as described in Sections 5.3.3 and 7.2.2) with optimal rescaling factor. To get reasonably close to the optimum, we tried all combinations with $20 \mid m$, $5 \mid \log_2(|S| / |M|)$, and $5 \mid r$.

Results. Our results for the LWR-based uRound2 KEM for the $0.187\beta \log_2 \beta - 1.019\beta + 16.1$ and 0.265β cost models are presented in Tables 8.11 and 8.12. The results for the RLWE-based EMBLEM KEM/PKE are presented in Tables 8.13 and 8.14. For both schemes, the quantum hybrid attack significantly outperforms the primal attack up to a factor of 2^{109} in the enumeration-regime. For uRound2 in the quantum-sieving-regime, the quantum hybrid attack performs slightly better than the primal attack. For EMBLEM, however, the quantum hybrid attack is outperformed by the primal attack in the quantum-sieving-regime. This can be explained by noting that guessing entries of the secret vector is typically less beneficial in the sieving-regime than in the enumeration-regime, in particular for uniform ternary secrets compared to sparse secrets.

Quantum hybrid attack					
Parameter set	I	II	III	IV	V
Expected cost	**91**	**126**	**140**	**185**	**185**
Guessing dim.	225	260	295	400	400
Block size	163	215	231	282	282
$\lvert S\rvert / \lvert M\rvert$	2^{-160}	2^{-185}	2^{-205}	2^{-255}	2^{-255}
m	260	320	360	420	420
Primal attack (cf. Table 8.8)					
Parameter set	I	II	III	IV	V
Expected cost	**126**	**188**	**213**	**294**	**294**

Table 8.11: Expected costs and corresponding attack parameters for the LWR-based uRound2 KEM parameter sets (cf. Table 8.3) under the $0.187\beta \log_2 \beta - 1.019\beta + 16.1$ BKZ cost model.

Quantum hybrid attack					
Parameter set	I	II	III	IV	V
Expected cost	**73**	**95**	**105**	**134**	**134**
Guessing dim.	170	180	205	275	275
Block size	240	316	349	453	453
$\lvert S\rvert / \lvert M\rvert$	2^{-145}	2^{-150}	2^{-170}	2^{-220}	2^{-220}
m	360	460	480	540	540
Primal attack (cf. Table 8.5)					
Parameter set	I	II	III	IV	V
Expected cost	**76**	**95**	**105**	**138**	**138**

Table 8.12: Expected costs and corresponding attack parameters for the LWR-based uRound2 KEM parameter sets (cf. Table 8.3) under the 0.265β BKZ cost model.

Quantum hybrid attack		
Parameter set	I	II
Expected cost	**179**	**162**
Guessing dim.	190	165
Block size	294	268
$\lvert S\rvert / \lvert M\rvert$	1	1
m	380	400
Primal attack (cf. Table 8.8)		
Parameter set	I	II
Expected cost	**210**	**242**

Table 8.13: Expected costs and corresponding attack parameters for the RLWE-based EMBLEM (R EMBLEM) KEM/PKE parameter sets (cf. Table 8.3) under the $0.187\beta\log_2\beta - 1.019\beta + 16.1$ BKZ cost model.

Quantum hybrid attack		
Parameter set	I	II
Expected cost	**120**	**108**
Guessing dim.	115	105
Block size	412	382
$\|S\| / \|M\|$	1	1
m	500	460
Primal attack (cf. Table 8.5)		
Parameter set	I	II
Expected cost	**92**	**102**

Table 8.14: Expected costs and corresponding attack parameters for the RLWE-based EMBLEM (R EMBLEM) KEM/PKE parameter sets (cf. Table 8.3) under the 0.265β BKZ cost model.

9 Conclusion

In this chapter, we conclude our work and give possible future research directions. This work presented several techniques to estimate the hardness of lattice problems (in particular instances of the uSVP) and in consequence to estimate the concrete security of lattice-based schemes.

We showed that the 2016 estimate [ADPS16] constitutes a reliable estimate for the minimal block size that guarantees the success of the BKZ [SE94, CN11, Che13] lattice reduction algorithm in solving uSVP. As the block size determines the runtime of the BKZ algorithm, this directly translates to cost estimates for one of the most efficient attacks on lattice-based schemes, the primal attack, which embeds lattice problems into uSVP instances and solves them via BKZ.

We further investigated the practical implications of using sparsification techniques [Kho03, Kho04, DK13, DRS14, SD16] when embedding lattice problems into uSVP instances. While the use of such techniques yield improved theoretical reductions [BSW16], our analysis shows that they typically do not lead to better attacks in practice. This is due to the fact that the low success probabilities introduced by these techniques is typically not compensated for by the expected speedup in the success case.

In addition to the above approaches to solve uSVP in general, we investigated hybrid attacks, which outperform the general approaches for certain uSVP instances. Typical targets for such attacks are uSVP instances with particularly small and/or sparse secret vectors. To this end, we adapted the hybrid attack [HG07] on the NTRU encryption scheme [HPS98] to solve the uSVP and presented an improved analysis of the attack. The new uSVP framework makes the attack applicable to a wider class of lattice-based cryptosystems (e.g., LWE-based schemes) while the improved analysis enables reliable runtime estimates, which were previously not available due to inaccuracies in the existing analyses.

We showed how to accelerate the hybrid attack in two different ways. The first is using parallel computing techniques of classical computers. We showed how to parallelize the hybrid attack and analyzed the expected speedup. Our theoretical analysis and practical experiments demonstrate that the parallel hybrid attack scales well within reasonable parameter ranges.

The second way we improved the hybrid attack is using quantum computing, which needs to be taken into account when evaluating the post-quantum security of cryptographic schemes. By replacing the classical meet-in-the-middle search of the attack with a quantum search [BHMT02] which is sensitive to the distribution on the search space we not only made the hybrid attack faster, but also applicable to a wider range of uSVP instances. Besides outperforming the classical hybrid attack, our results show that the quantum hybrid attack also outperforms the primal attack for several uSVP instances with small and sparse secret vectors as well as vectors that follow a (narrow) discrete Gaussian distribution.

Finally, we used our derived results for the primal and quantum hybrid attack to evaluate the security of the lattice-based schemes which were accepted to NIST's process of standardizing post-quantum public-key cryptography [Nat16], highlighting the practical implications of this work.

Future work. All of the attacks discussed in this work make heavy use of the BKZ lattice reduction algorithm. The runtime of the BKZ algorithms is determined by its block size. In this work, we showed how to determine the optimal block size for the respective attacks. To determine the runtime of BKZ with a certain block size, we applied estimates that exist in the current literature. However, the numerous existing estimates provide vastly different results as highlighted in Chapter 8. The main source of these differences is that BKZ is either assumed to rely on enumeration algorithms [Kan83, FP85, MW15] as SVP oracle or on sieving algorithms [AKS01, LMvdP15, BDGL16]. While sieving algorithms offer better asymptotic complexities, they require access to exponentially large memory, which may render them less efficient in practice despite the better asymptotics. Currently, there exists no consensus in the cryptographic community as to which estimates to use for BKZ. Settling this debate by deriving an accurate and realistic cost model for BKZ is one of the most important topics in the cryptanalysis of lattice-based cryptography. Note that the results presented in this thesis are applicable to all cost models of BKZ, and hence relevant independently of what future works shows with respect to the runtime of BKZ.

In our analysis of the 2016 estimate for the primal attack, we made the assumption that BKZ uses a perfect SVP oracle as subroutine. Future research may investigate if it is possible to obtain an improved estimate by relaxing this assumption and allowing SVP oracles with certain success probabilities (possibly different success probabilities at different stages of BKZ) as used in BKZ 2.0 [CN11, Che13]. Lowering the success probability of the SVP oracle can considerably decrease the runtime of BKZ, but the effect on the 2016 estimate so far is unclear.

For the hybrid attack, we used Babai's Nearest Plane algorithm [Bab86] to check if a guess is correct. Future work can investigate if it is beneficial to replace the Nearest Plane algorithm by a different BDD solver, or even only an algorithm that decides whether a given CVP instance is in fact a BDD instance. However, the fact that Nearest Plane can be divided into an expensive precomputation phase and a cheap BDD phase seems to make it particularly suitable for the hybrid attack.

With respect to the parallel hybrid attack we identified the interference of the execution of multiple BKZ executions on a single compute node and the parallel speedup of the guessing as a bottleneck in our current implementation. It results from an overextension of system's memory interface through multiple BKZ runs executed in parallel. Replacing NTL's BKZ implementation by a more cache friendly and memory efficient one will remove this effect. Furthermore, an analysis of the performance and scalability of a parallel BKZ implementation was out of scope and is left for future work.

An open question regarding the quantum hybrid attack is whether is can be improved by a quantum meet-in-the-middle search [BHT98, XWW^{+}12, WMM13] as briefly discussed in [Sch15]. Besides the problem of requiring huge quantum memory, this would introduce the low collision finding probabilities as encountered in the classical hybrid attack. We therefore may conjecture that using a quantum meet-in-the-middle search does not improve the quantum hybrid attack, however, a detailed analysis of such a modification has not yet been conducted.

As most of the proposed quantum algorithms for lattice problems, our quantum hybrid attack uses (a generalization of) Grover's quantum search algorithm [Gro96]. The further investigation of dedicated quantum algorithms designed to solve specific problems, as for example used for lattices with additional algebraic structure [CDPR16, BS16, Bia17, CDW17], remains open for future work. In addition, while parts of this thesis were focused on weaknesses in lattice problems introduced by small or sparse secret vectors, the study of potential weaknesses of lattice problems introduced by additional algebraic structure as in [ELOS15, ABD16, KF17] is an important future research topic.

Bibliography

[ABB$^+$17] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting TESLA in the quantum random oracle model. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*, pages 143–162. Springer International Publishing, 2017. 3, 19, 20, 38, 41

[ABBD15] Erdem Alkim, Nina Bindel, Johannes Buchmann, and Özgür Dagdelen. TESLA: Tightly-secure efficient signatures from standard lattices. Cryptology ePrint Archive, Report 2015/755, 2015. `http://eprint.iacr.org/2015/755`. 38

[ABD16] Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 153–178. Springer, Heidelberg, August 2016. 18, 159

[ACF$^+$15] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Algebraic algorithms for LWE problems. *ACM Comm. Computer Algebra*, 49(2):62, 2015. 18

[ADPS16] 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*, pages 327–343. USENIX Association, 2016. 1, 3, 5, 14, 15, 19, 21, 22, 23, 31, 43, 75, 114, 130, 131, 157

[AFFP14] Martin R. Albrecht, Jean-Charles Faugère, Robert Fitzpatrick, and Ludovic Perret. Lazy modulus switching for the BKW algorithm on LWE. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 429–445. Springer, Heidelberg, March 2014. 18

[AFG14] 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, *ICISC 13*, volume 8565 of *LNCS*, pages 293–310. Springer, Heidelberg, November 2014. 16, 19, 21

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP 2011, Part I*, volume 6755 of *LNCS*, pages 403–415. Springer, Heidelberg, July 2011. 18

[AGVW17] 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. 46

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *28th ACM STOC*, pages 99–108. ACM Press, May 1996. 17

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd ACM STOC*, pages 601–610. ACM Press, July 2001. 15, 158

[Alb17] Martin R. Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 103–129. Springer, Heidelberg, April / May 2017. 15, 17, 20, 37, 38, 39, 40, 68, 131, 150

[ANS18] Yoshinori Aono, Phong Q. Nguyen, and Yixin Shen. Quantum lattice enumeration and tweaking discrete pruning. Cryptology ePrint Archive, Report 2018/546, 2018. `http://eprint.iacr.org/2018/546`. 15, 131

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015. 14, 15, 16, 19, 21, 38, 40, 41, 46, 69, 92, 125, 130, 131, 152

[AWHT16] Yoshinori Aono, Yuntao Wang, Takuya Hayashi, and Tsuyoshi Takagi. Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 789–819. Springer, Heidelberg, May 2016. 14, 69

[BAA$^+$17] Nina Bindel, Sedat Akleylek, Erdem Alkim, Paulo S. L. M. Barreto, Johannes Buchmann, Edward Eaton, Gus Gutoski, Juliane Kramer, Patrick Longa, Harun Polat, Jefferson E. Ricardini, and Gustavo Zanon. qtesla. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[Bab86] László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, Mar 1986. 16, 17, 24, 158

[Ban17] Rachid El Bansarkhani. Kindi. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132, 148

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography*. Springer Publishing Company, Incorporated, 1st edition, 2009. 1

[BCD$^+$16] Joppe W. 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 Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1006–1018. ACM Press, October 2016. 19

[BCIV17] Joppe W. Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Privacy-friendly forecasting for the smart grid using homomorphic encryption and the group method of data handling. In Marc Joye and Abderrahmane Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017, Proceedings*, pages 184–201. Springer International Publishing, 2017. 20, 38, 41

[BCLvV16] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. *IACR Cryptology ePrint Archive*, 2016:461, 2016. 51, 52, 70, 75, 76, 77, 78

[BCLvV17a] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. Ntru prime. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. 12, 131, 132

[BCLvV17b] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime: Reducing attack surface at low cost. In Carlisle Adams and Jan Camenisch, editors, *SAC 2017*, volume 10719 of *LNCS*, pages 235–260. Springer, Heidelberg, August 2017. 51, 52, 70, 75, 76, 77, 85, 92

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In Robert Krauthgamer, editor, *27th SODA*, pages 10–24. ACM-SIAM, January 2016. 15, 19, 37, 130, 158

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - kyber: A cca-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 353–367. IEEE, 2018. 19

[Ber18] Daniel J. Bernstein, 2018. Comment on PQC forum in response to an earlier version of this work. Available at https://groups.google.com/a/list.nist.gov/d/msg/pqc-forum/h4_LCVNejCI/FyV5hgnqBAAJ. 150

[BG14a] Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In Josh Benaloh, editor, *CT-RSA 2014*, volume 8366 of *LNCS*, pages 28–47. Springer, Heidelberg, February 2014. 1, 3, 19, 38

[BG14b] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14*, volume 8544 of *LNCS*, pages 322–337. Springer, Heidelberg, July 2014. 2, 36, 37, 116, 152

[BGG+16] Johannes A. Buchmann, Florian Göpfert, Tim Güneysu, Tobias Oder, and Thomas Pöppelmann. High-performance and lightweight lattice-based public-key encryption. In *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security, CPSSAsiaCCS, Xi'an, China, May 30 - June 3, 2016*, pages 2–9, 2016. 51, 52, 77, 78, 79, 80, 85, 92, 118

[BGH13] Zvika Brakerski, Craig Gentry, and Shai Halevi. Packed ciphertexts in LWE-based homomorphic encryption. In Kaoru Kurosawa and

Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 1–13. Springer, Heidelberg, February / March 2013. 37

[BHMT02] Gilles Brassard, P. Høyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. In *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series*, pages 53–74. American Mathematical Society, 2002. Earlier version in arxiv:quant-ph/0005055. 4, 107, 108, 112, 158

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*, volume 1380 of *Lecture Notes in Computer Science*, pages 163–169. Springer, 1998. 159

[Bia17] Jean-François Biasse. Approximate short vectors in ideal lattices of $\mathbb{Q}(\zeta_{p^e})$ with precomputation of $\mathrm{Cl}(\mathcal{O}_K)$. In Carlisle Adams and Jan Camenisch, editors, *SAC 2017*, volume 10719 of *LNCS*, pages 374–393. Springer, Heidelberg, August 2017. 18, 159

[BKW00] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *32nd ACM STOC*, pages 435–440. ACM Press, May 2000. 18

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Heidelberg, April 2012. 11

[BS16] Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Robert Krauthgamer, editor, *27th SODA*, pages 893–902. ACM-SIAM, January 2016. 18, 159

[BSW16] 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, *ICALP 2016*, volume 55 of *LIPIcs*, pages 76:1–76:12. Schloss Dagstuhl, July 2016. 3, 5, 43, 44, 45, 157

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd FOCS*, pages 97–106. IEEE Computer Society Press, October 2011. 1

[BVWW16] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In Madhu Sudan, editor, *ITCS 2016*, pages 147–156. ACM, January 2016. 1

[CDPR16] Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 559–585. Springer, Heidelberg, May 2016. 18, 159

[CDW17] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 324–348. Springer, Heidelberg, April / May 2017. 18, 159

[Che13] Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, Paris 7, 2013. 13, 14, 15, 19, 24, 31, 69, 131, 157, 158

[CHK$^+$17] Jung Hee Cheon, Kyoohyung Han, Jinsu Kim, Changmin Lee, and Yongha Son. A practical post-quantum public-key cryptosystem based on splWE. In Seokhie Hong and Jong Hwan Park, editors, *ICISC 16*, volume 10157 of *LNCS*, pages 51–74. Springer, Heidelberg, November / December 2017. 3, 19, 85

[CIV16] Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Provably weak instances of ring-LWE revisited. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 147–167. Springer, Heidelberg, May 2016. 18

[CJL16] Jung Hee Cheon, Jinhyuck Jeong, and Changmin Lee. An algorithm for ntru problems and cryptanalysis of the ggh multilinear map without a low-level encoding of zero. *LMS Journal of Computation and Mathematics*, 19(A):255–266, 2016. 18

[CKLS16a] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! Practical post-quantum public-key encryption from LWE and LWR. Cryptology ePrint Archive, Report 2016/1126, 2016. `http://eprint.iacr.org/2016/1126`. 3, 19, 37

[CKLS16b] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! Practical post-quantum public-key encryption from LWE and LWR. Cryptology ePrint Archive, Report 2016/1126 (20161222:071525), 2016. `http://eprint.iacr.org/2016/1126/20161222:071525`. 37, 38

[CLP17] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic library - SEAL v2.1. In Michael Brenner, Kurt Rohloff, Joseph Bonneau, Andrew Miller, Peter Y. A. Ryan, Vanessa Teague, Andrea Bracciali, Massimiliano Sala, Federico Pintore, and Markus Jakobsson, editors, *FC 2017 Workshops*, volume 10323 of *LNCS*, pages 3–18. Springer, Heidelberg, April 2017. 3, 19, 38, 40

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2011. 2, 13, 14, 15, 16, 19, 24, 69, 130, 131, 157, 158

[CPL$^+$17] Jung Hee Cheon, Sangjoon Park, Joohee Lee, Duhyeong Kim, Yongsoo Song, Seungwan Hong, Dongwoo Kim, Jinsu Kim, Seong-Min Hong, Aaram Yun, Jeongsu Kim, Haeryong Park, Eunyoung Choi, Kimoon kim, Jun-Sub Kim, and Jieun Lee. Lizard. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132, 148

[CS97] Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 52–61. Springer, Heidelberg, May 1997. 12

[DDLL13] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 40–56. Springer, Heidelberg, August 2013. 51, 52, 70, 79, 81, 82, 84, 85

[DK13] Daniel Dadush and Gábor Kun. Lattice sparsification and the approximate closest vector problem. In Sanjeev Khanna, editor, *24th SODA*, pages 1088–1102. ACM-SIAM, January 2013. 3, 43, 44, 157

[DKRV17] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. 132

[DRS14] Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. On the closest vector problem with a distance guarantee. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 98–109. IEEE Computer Society, 2014. 3, 43, 44, 157

[DTGW17] Jintai Ding, Tsuyoshi Takagi, Xinwei Gao, and Yuntao Wang. Ding key exchange. Technical report, National Institute of Standards and Technology, 2017. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions. 132

[EHL14] Kirsten Eisenträger, Sean Hallgren, and Kristin E. Lauter. Weak instances of PLWE. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014*, volume 8781 of *LNCS*, pages 183–194. Springer, Heidelberg, August 2014. 18

[ELOS15] Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. Provably weak instances of ring-LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 63–92. Springer, Heidelberg, August 2015. 18, 159

[ELOS16] Yara Elias, Kristin E. Lauter, Ekin Ozman, and Katherine E. Stange. Ring-LWE cryptography for the number theorist. In Ellen E. Eischen, Ling Long, Rachel Pries, and Katherine E. Stange, editors, *Directions in Number Theory*, pages 271–290. Springer International Publishing, 2016. 18

[FP85] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–463, May 1985. 15, 131, 158

[FPL17] The FPLLL development team. fplll, a lattice reduction library. Available at https://github.com/fplll/fplll, 2017. 20, 22, 24, 26, 30

[FPY17] The FPYLLL development team. fyplll, a Python (2 and 3) wrapper for fplll. Available at https://github.com/fplll/fpylll, 2017. 20, 22, 24, 26, 30

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144. 38

[GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. Cryptology ePrint Archive, Report 2012/099, 2012. `http://eprint.iacr.org/2012/099`. 37

[GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 850–867. Springer, Heidelberg, August 2012. 37

[GJMS17] Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski. Coded-BKW with sieving. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 323–346. Springer, Heidelberg, December 2017. 18

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 23–42. Springer, Heidelberg, August 2015. 18

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 530–547. Springer, Heidelberg, September 2012. 51, 52, 82, 84, 85

[GMZB$^+$17] Oscar Garcia-Morchon, Zhenfei Zhang, Sauvik Bhattacharya, Ronald Rietman, Ludo Tolhuizen, and Jose-Luis Torre-Arce. Round2. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132, 152

[GN08a] Nicolas Gama and Phong Q. Nguyen. Finding short lattice vectors within Mordell's inequality. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 207–216. ACM Press, May 2008. 2, 13

[GN08b] Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 31–51. Springer, Heidelberg, April 2008. 3, 19, 20, 21

[GNR10] Nicolas Gama, Phong Q. Nguyen, and Oded Regev. Lattice enumeration using extreme pruning. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 257–278. Springer, Heidelberg, May / June 2010. 25

[Göp16] Florian Göpfert. *Securely Instantiating Cryptographic Schemes Based on the Learning with Errors Assumption.* PhD thesis, Technische Universität Darmstadt, 2016. `http://tuprints.ulb.tu-darmstadt.de/5850/`. 19, 21

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM. 4, 107, 108, 159

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013. 1

[Ham17] Mike Hamburg. Three bears. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[HG07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 150–169. Springer, Heidelberg, August 2007. 3, 5, 14, 51, 53, 56, 60, 62, 64, 66, 70, 71, 72, 157

[HGSW] N. Howgrave-Graham, J. H. Silverman, and W. Whyte. A meet-in-the-middle attack on an NTRU private key. `https://www.securityinnovation.com/uploads/Crypto/NTRUTech004v2.pdf`. 56, 89

[HHGP$^+$07] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H Silverman, and William Whyte. Hybrid lattice reduction and meet in the middle resistant parameter selection for NTRUEncrypt. *Submission/-contribution to ieee p1363*, 1:2007–02, 2007. 3, 51, 70, 72

[HHHGW09] Philip S. Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 09*, volume 5536 of *LNCS*, pages 437–455. Springer, Heidelberg, June 2009. 3, 17, 51, 53, 66, 70, 72, 115

[HKM17] Gottfried Herold, Elena Kirshanova, and Alexander May. On the asymptotic complexity of solving lwe. *Designs, Codes and Cryptography*, Jan 2017. 19, 21

[HPS96] Jeffery Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A new high speed public-key cryptosystem. Technical report, Draft distributed at CRYPTO96, 1996. available at `https://cdn2.hubspot.net/hubfs/49125/downloads/ntru-orig.pdf`. 12

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998. 1, 3, 12, 51, 72, 75, 85, 157

[HPS11] Guillaume Hanrot, Xavier Pujol, and Damien Stehlé. Analyzing blockwise lattice algorithms using dynamical systems. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 447–464. Springer, Heidelberg, August 2011. 2, 14

[HPS$^+$15] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. Cryptology ePrint Archive, Report 2015/708, 2015. `http://eprint.iacr.org/2015/708`. 131

[HPS$^+$17] Jeffrey Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 3–18. Springer, Heidelberg, February 2017. 3, 51, 52, 53, 70, 72, 73, 74, 75

[HS14] Shai Halevi and Victor Shoup. Algorithms in HElib. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2014. 85

[JF11] David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2011. 1

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th ACM STOC*, pages 193–206. ACM Press, April 1983. 15, 131, 158

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, Aug 1987. 2, 16, 43

[KF15] Paul Kirchner and Pierre-Alain Fouque. An improved BKW algorithm for LWE with applications to cryptography and lattices. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 43–62. Springer, Heidelberg, August 2015. 18

[KF17] Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 3–26. Springer, Heidelberg, April / May 2017. 18, 149, 159

[Kho03] Subhash Khot. Hardness of approximating the shortest vector problem in high Lp norms. In *44th FOCS*, pages 290–297. IEEE Computer Society Press, October 2003. 3, 43, 44, 157

[Kho04] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. In *45th FOCS*, pages 126–135. IEEE Computer Society Press, October 2004. 3, 43, 44, 157

[Laa15a] T Laarhoven. *Search problems in cryptography: From fingerprinting to lattice sieving*. PhD thesis, Eindhoven University of Technology, 2015. 130

[Laa15b] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015. 15, 19, 130

[LDK$^+$17] Vadim Lyubashevsky, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals-dilithium. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[Li11] Shengqiao Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011. 34, 64

[LLJ$^+$17] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, and Zhenfei Zhang. Lac. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[LLL82] A.K. Lenstra, Jr. Lenstra, H.W., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982. 2, 25

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 577–594. Springer, Heidelberg, August 2009. 16, 43

[LMvdP15] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Finding shortest lattice vectors faster using quantum search. *Designs, Codes and Cryptography*, 77(2–3):375–400, December 2015. 15, 158

[LO83] J. C. Lagarias and Andrew M. Odlyzko. Solving low-density subset sum problems. In *24th FOCS*, pages 1–10. IEEE Computer Society Press, November 1983. 20

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, Heidelberg, February 2011. 1, 17, 18, 67, 78, 122

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, Heidelberg, May / June 2010. 11, 75

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 75(3):565–599, June 2015. 11

[LV01] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001. 17, 66

[LWXZ14] Mingjie Liu, Xiaoyun Wang, Guangwu Xu, and Xuexin Zheng. A note on BDD problems with λ_2-gap. *Inf. Process. Lett.*, 114(1-2):9–12, 2014. 43

[MLC$^+$17] Artur Mariano, Thijs Laarhoven, Fábio Correia, Manuel Rodrigues, and Gabriel Falcão. A practical view of the state-of-the-art of lattice-based cryptanalysis. *IEEE Access*, 5:24184–24202, 2017. 88

[Mos15] Michele Mosca. Cybersecurity in an era with quantum computers: Will we be ready? Cryptology ePrint Archive, Report 2015/1075, 2015. `http://eprint.iacr.org/2015/1075`. 1

[MS01] Alexander May and Joseph H. Silverman. Dimension reduction methods for convolution modular lattices. In *Cryptography and Lattices, International Conference, CaLC 2001, Providence, RI, USA, March 29-30, 2001, Revised Papers*, pages 110–125, 2001. 12, 116

[MW15] Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In Piotr Indyk, editor, *26th SODA*, pages 276–294. ACM-SIAM, January 2015. 15, 131, 158

[MW16] Daniele Micciancio and Michael Walter. Practical, predictable lattice basis reduction. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 820–849. Springer, Heidelberg, May 2016. 2, 14

[NAB$^+$17] Michael Naehrig, Erdem Alkim, Joppe Bos, Leo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. Frodokem. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[Nat16] National Institute of Standards and Technology. Submission requirements and evaluation criteria for the Post-Quantum Cryptography standardization process. `http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf`, December 2016. 1, 5, 16, 125, 126, 152, 158

[Ngu18] Phong Nguyen, 2018. Comment on PQC forum. Available at `https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/nZBIBvYmmUI`. 11, 148

[Olv10] Frank WJ Olver. *NIST handbook of mathematical functions.* Cambridge University Press, 2010. 59, 65, 111

[PAA$^+$17] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Leo Ducas, Antonio de la Piedra, Peter Schwabe, and Douglas Stebila. Newhope. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[Pei16a] Chris Peikert. A decade of lattice cryptography. *Found. Trends Theor. Comput. Sci.*, 10(4):283–424, March 2016. 1

[Pei16b] Chris Peikert. How (not) to instantiate ring-LWE. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 411–430. Springer, Heidelberg, August / September 2016. 18

[PFH$^+$17] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 12, 132

[PHAM17] Le Trieu Phong, Takuya Hayashi, Yoshinori Aono, and Shiho Moriai. Lotus. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 131, 132

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):1–40, Sep 2009. 1, 10

[S$^+$17] William Stein et al. *Sage Mathematics Software Version 7.5.1.* The Sage Development Team, 2017. Available at `http://www.sagemath.org`. 24, 66

[Saa17] Markku-Juhani O. Saarinen. Hila5. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[SAB$^+$17] Peter Schwabe, Roberto Avanzi, Joppe Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehle. Crystals-kyber. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[SAL$^+$17] Nigel P. Smart, Martin R. Albrecht, Yehuda Lindell, Emmanuela Orsini, Valery Osheter, Kenny Paterson, and Guy Peer. Lima. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132, 148

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987. 2

[Sch03] 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. 14

[Sch15] John M. Schanck. *Practical Lattice Cryptosystems: NTRUEncrypt and NTRUMLS.* PhD thesis, University of Waterloo, 2015. 3, 12, 51, 53, 70, 71, 72, 73, 108, 159

[SD16] Noah Stephens-Davidowitz. Discrete Gaussian sampling reduces to CVP and SVP. In Robert Krauthgamer, editor, *27th SODA*, pages 1748–1764. ACM-SIAM, January 2016. 3, 43, 44, 45, 157

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994. 13, 19, 157

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997. 1

[SHRS17] John M. Schanck, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. Ntru-hrss-kem. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 12, 132, 149

[SPL$^+$17] Minhye Seo, Jong Hwan Park, Dong Hoon Lee, Suhri Kim, and Seung-Joon Lee. Emblem and r.emblem. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 130, 132, 152

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 617–635. Springer, Heidelberg, December 2009. 11

[SSZ17] Ron Steinfeld, Amin Sakzad, and Raymond K. Zhao. Titanium. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[Tuk77] John W Tukey. Exploratory data analysis. *Addison-Wesley Series in Behavioral Science: Quantitative Methods, Reading, Mass.: Addison-Wesley, 1977*, 1977. 102

[vV16] Christine van Vredendaal. Reduced memory meet-in-the-middle attack against the ntru private key. *LMS Journal of Computation and Mathematics*, 19(A):43–57, 2016. 89

[vW96] Paul C. van Oorschot and Michael J. Wiener. Improving implementable meet-in-the-middle attacks by orders of magnitude. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 229–236. Springer, Heidelberg, August 1996. 89

[vW99] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, 1999. 89

[WAT18] Yuntao Wang, Yoshinori Aono, and Tsuyoshi Takagi. An experimental study of kannan's embedding technique for the search lwe problem. In *The 19th International Conference on Information and Communications Security, ICICS 2017*, volume 10631 of *LNCS*. Springer, November 2018. 16

[WMM13] Hong Wang, Zhi Ma, and ChuanGui Ma. An efficient quantum meet-in-the-middle attack against ntru-2005. *Chinese Science Bulletin*, 58(28-29):3514–3518, 2013. 159

[XWW$^+$12] Zhijian Xiong, Jinshuang Wang, Yanbo Wang, Tao Zhang, and Liang Chen. An improved mitm attack against ntru. *International Journal of Security and Its Applications*, 6(2):269–274, 2012. 159

[ZCHW17a] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. Ntruencrypt. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132

[ZCHW17b] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, and William Whyte. pqntrusign. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 12, 132

[ZjGS17] Yunlei Zhao, Zhengzhong jin, Boru Gong, and Guangye Sui. Kcl (pka okcn/akcn/cnke). Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`. 132