Cryptanalysis and Applications of Lattice-based Encryption Schemes

Benjamin R. Curtis

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
Royal Holloway, University of London

2020

Declaration

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled at the Centre for Doctoral Training in Cyber Security in the Department of Information Security as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Ben Curtis

Acknowledgements

There are many people that I need to thank for their support throughout the course of my PhD. Firstly, my supervisor Professor Keith Martin. Keith has been a constant source of encouragement and has provided me with many opportunities over the past five years. Thanks for all the wisdom!

Next, my advisor Professor Martin Albrecht. Martin has shown exceptional levels of patience as I have gotten to grips with various aspects of lattice-based cryptography. I have learned a lot from Martin and have appreciated all of his time and effort.

I have been lucky to have had mentorship from many other PhD students, including Dr Rachel Player, Dr Thomas Wunderer, Dr James Alderman, Dr Naomi Farley, and Dr Anamaria Costache. At some point during my PhD, you have provided support and knowledge which has been invaluable to me.

There are many other people who have become good friends over the past five years, especially James, Lydia, Nick, Ivan, Pip, Andreas, Alex, Jason, Torben, Fernando, Giovanni, Joanne, Ela, and Rob. To all of those who I have befriended at conferences, especially Sander, Ryan, and Marcella - it has been a lot of fun.

I also need to thank Professor Johannes Buchmann and Dr Thomas Wunderer for hosting me at TU Darmstadt for a research visit in October 2018, as well as Professor Kurt Rohloff and Professor Yuriy Polyakov for hosting me at New Jersey Institute of Technology for a research visit in April 2019. As well as this, the `homomorphicencryption.org` community has been welcoming and has provided me with several opportunities.

A special acknowledgement goes to Amit who has taught me many things over the past five years, and still takes the time to explain things to me today. Alongside this you have been a great friend.

Thanks to all of my family members, particularly Mum, Dad, Whoa, and Pop, for providing me with support and encouragement.

Finally, thanks to Eve for all the love, laughs, and for always being there over the past few years. Here's to many more!

Abstract

The work presented in this thesis is focused around the security of the Learning with Errors (LWE) problem, as well as applications of homomorphic encryption schemes.

In Chapter 1, we provide an overview of the topics discussed in this thesis: lattice-based cryptography, secure computation, cryptanalysis, and standardisation.

In Chapter 2, we introduce necessary background concepts. Specifically, we outline some notions related to lattice-based cryptography and cryptanalysis.

In Chapter 3, we consider trade-offs in "Batch Bounded Distance Decoding". We consider *guess-and-verify decoding* (g-v decoding), a porting of the decoding attack on LWE into the case of small and/or sparse secret vectors. This results in a combinatorial trade-off, where components of the secret vector are guessed before batches of BDD instances are solved in a smaller dimension. This attack technique has similarities with the hybrid lattice-reduction and meet-in-the-middle (hybrid-decoding) attack, and we compare and contrast these two techniques throughout. We conclude that, under certain assumptions, our g-v decoding technique outperforms a variant of the hybrid-decoding attack.

In Chapter 4, we analyse submissions to the NIST standardisation process for post-quantum cryptographic algorithms. Specifically, we consider all parameter sets submitted to the first round, for every lattice-based scheme, as well as the cost models used for lattice reduction. We estimate the security of every parameter set, under every cost model, considering both the uSVP and dual attacks (where appropriate). This allows for individual schemes to be compared more easily. As a result of this analysis, we observe that cost models for the BKZ algorithm are not order preserving. That is, if scheme A is "more secure" than scheme B under cost model 1, the same is not necessarily true under cost model 2. Finally we outline the current state of the NIST standardisation process, and provide some estimates for the schemes which have reached the third round.

In Chapter 5, we consider homomorphic encryption-style parameter sets, and explore hybrid attacks. Hybrid attacks are competitive in regimes where the LWE secret is small and/or sparse, so need to be considered for parameter sets used in homomorphic encryption schemes. We consider the effect of secret sparsity on security estimates, and consider the trade-off between bootstrapping complexity and security.

Finally, in Chapter 6, we consider an application of homomorphic encryption: "Private Outsourced Kriging Interpolation". Kriging is a spatial interpolation algorithm which has

applications in geoscience. We consider the outsourcing of this algorithm using homomorphic encryption, and outline techniques which can be used to protect the sensitive parameters in order to provide an efficient solution.

Contents

List of Figures

List of Tables

List of Algorithms

List of Publications

This thesis is based on the following publications:

1. [ACF+17]: James Alderman, Benjamin R. Curtis, Oriol Farràs, Keith M. Martin, and Jordi Ribes-González. Private Outsourced Kriging Interpolation. The 5th Workshop on Encrypted Computing and Applied Homomorphic Cryptography. In International Conference on Financial Cryptography and Data Security Workshops (pp. 75-90). Springer, volume 10323 of Lecture Notes in Computer Science, 2017.

2. [ACD+18]: Martin R. Albrecht, Benjamin R. Curtis, Alex Davidson, Amit Deo, Rachel Player, Eamonn Postlethwaite, Fernando Virdia and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! 11th International Conference on Security in Communications Networks (SCN) (pp. 351-367). Springer, volume 11035 of Lecture Notes in Computer Science, 2018.

3. [ACW20]: Martin R. Albrecht, Benjamin R. Curtis and Thomas Wunderer. Exploring Trade-offs in Batch Bounded Distance Decoding. Selected Areas of Cryptography 2019 (pp. 467-491). Springer, volume 11959 of Lecture Notes in Computer science, 2019.

4. [CP19]: Benjamin R. Curtis and Rachel Player. On the Feasibility and Impact of Standardising Sparse-secret LWE Parameter Sets for Homomorphic Encryption. In Proceedings of the 7th ACM Workshop on Encrypted Computing and Applied Homomorphic Cryptography (pp. 1-10). Association for Computing Machinery, 2019.

Chapter 1

Introduction

Contents

1.1 Lattice-based Cryptography

Cryptography underpins society as we know it. From secure messaging applications to online payments, cryptography allows us to perform many daily tasks over the internet in a secure manner – often without the knowledge that we are even using it. Currently used cryptographic primitives whose security relies on RSA [RSA78] or discrete logarithm problems [DH76] can be broken by sufficiently large quantum computers using Shor's algorithm [Sho97]. Thus begins the search for quantum-secure cryptography, known more commonly as *post-quantum cryptography*. The US National Institute of Standards and Technology (NIST) started a standardisation procedure in 2016 [Nat16], with the goal being to design, analyse, and standardise a portfolio of public-key encryption schemes and digital signature algorithms. A variety of candidate post-quantum secure algorithms have been proposed, including code-based submissions, lattice-based submissions, isogeny-based submissions, and multivariate-based submissions.

Of the candidate post-quantum secure algorithms submitted to this process, lattice-based cryptographic algorithms represent a strong candidate for standardisation and have been tested in practice [Bra16]. There are three major roadblocks to the wide-scale deployment of lattice-based cryptosystems: usability, confidence in the underlying security assumptions, and

standardisation. Indeed, cryptosystems must be usable, and secure, for wide-scale deployment to be considered and, moreover, standardisation is essential for many forms of industrial or governmental usage. In this thesis, we consider each of these aspects individually.

The hardness of lattice-based cryptographic primitives typically relies on the *Learning with Errors* (LWE) problem. LWE (and its variants) has given rise to many advanced encryption techniques such as homomorphic encryption, which has lead to a variety of additional interesting areas of research.

1.2 Secure Computation and Applications

Whilst typical encryption techniques allow us to secure data-at-rest and data-in-transit, once data has been encrypted we are no longer able to perform operations on this data. *Fully homomorphic encryption* [Gen09] allows for computation to occur on data which remains encrypted, giving rise to an abundance of privacy-preserving applications. One of the major bottlenecks with fully homomorphic encryption is the efficiency of the *bootstrapping step* used to refresh a noisy ciphertext. Solutions used in practice tend to avoid this expensive procedure in favour of *somewhat homomorphic encryption*, which allows for a limited computation, such as polynomial functions up to some maximal degree d, to be evaluated in a secure manner.

There are a variety of other secure computation technologies available for use in the wild today. Garbled circuits [Yao86] allow for the private evaluation of boolean circuits, oblivious RAM [Gol87] allows for access patterns to be protected, and secure multiparty computation allows several parties to jointly compute a function over their private inputs. These techniques can be combined with homomorphic encryption to yield efficient privacy-preserving solutions to interesting problems, e.g. [CCD+19]. We are now in an era where secure computation is being used in commercial products [Env20, Unb20, Dua20, Zam20], which gives an indication of the practicality of these techniques.

In this thesis, we are interested in both the application, and security, of homomorphic encryption encryption schemes. In order to provide increased efficiency, the parameter sets used typically do not come under the umbrella of the various security reductions outlined in the literature. These adaptations come with a concrete security loss[1], and it is important

[1]Here we are referring to both the *tightness of reductions*, for example the reduction from LWR to LWE,

to quantify this loss of security so that homomorphic encryption-style parameter sets can be used with confidence.

1.3 Cryptanalysis

The security of LWE-based cryptographic constructions is measured in terms of the running times of the best-known attacks. The LWE Estimator of Albrecht et al. [APS15, Est20] is a common tool used for estimating the running time of a subset of these attacks on a given set of LWE parameters (n, α, q). For other variants of the LWE, such as those used in homomorphic encryption schemes where the secret vector is small and/or sparse, *hybrid attacks* are also competitive.

All of the attacks considered in this thesis involve the usage of lattice-reduction algorithms, which, generically, find short vectors in projected sublattices of size β, referred to as the *blocksize*, in order to find a short basis of a given input lattice. To begin, we reduce the LWE problem to a lattice problem such as the unique Shortest Vector problem (uSVP), the Bounded Distance Decoding problem (BDD) or the Short Integer Solutions problem (SIS) before solving the lattice problem at hand via lattice-reduction techniques.

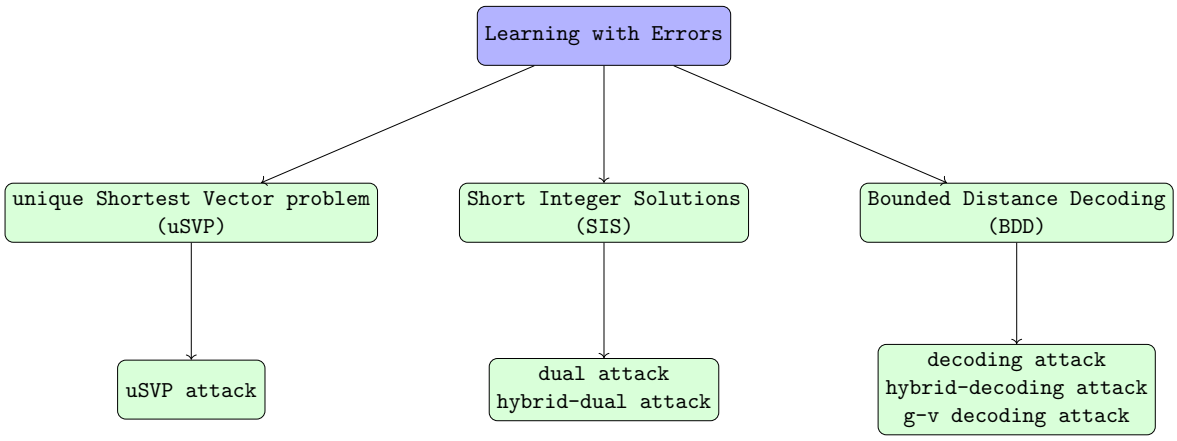


Figure 1.1: An outline of the various attack techniques used to solve the Learning with Errors problem.

Of particular interest in this thesis are hybrid attacks, which exploit any smallness and/or sparsity in the LWE secret. Often, it is advantageous to guess components of the LWE secret

and the chosen secret distribution.

(or error) vector before performing lattice reduction in a smaller dimension. This creates a trade-off between guessing correctly and the reduced cost of performing lattice reduction in a smaller dimension. Hybrid attacks involve a myriad of trade-offs, and are notoriously difficult to optimise. To this end, we explore a variety of these trade-offs as well as the assumptions which are made in analyses of hybrid attacks. As part of our work, we have released open-source code which allows for the complexity of hybrid attacks to be estimated under a variety of assumptions[2]. We have also contributed to the LWE estimator, to help keep the output estimates in line with state-of-the-art attacks.

1.4 Standardisation

There are two ongoing standardisation procedures of interest in this thesis. The first, as discussed in Section 1.1, is the NIST standardisation process for post-quantum algorithms, and we consider the security of all of the schemes submitted to first and third rounds of this process.

The second standardisation procedure is the ongoing effort to standardise aspects of homomorphic encryption, and the `homomorphicencryption.org` consortium are leading this effort. The Homomorphic Encryption Security Standard (HE Standard) [ACC+18] recommends secure parameters for use in homomorphic encryption schemes, and we discuss and analyse some of these parameter selections in this thesis. We also consider potential future extensions to the HE Standard, and outline several points for consideration in future work.

1.5 Organisation and Contributions

In Chapter 2 we outline all necessary background content required. This includes preliminary notions from lattice-based cryptography and cryptanalysis. We then introduce our four contributions in Chapters 3, 4, 5, and 6.

1. In Chapter 3, we discuss the Batch Bounded Distance Decoding problem and its application to solving the Small-secret Learning with Errors problem. We compare this technique

[2]This code can be found at `github.com/bencrts/hybrid_attacks`.

to Howgrave-Graham's hybrid lattice-reduction and meet-in-the-middle attack (hybrid-decoding attack), and discuss how important assumptions are in these analyses. We show that, under certain assumptions, our attack technique outperforms a variant of the hybrid-decoding attack and, under other sets of assumptions, the converse is true. This chapter corresponds to the publication *Exploring trade-offs in batch bounded distance decoding* detailed in the List of Publications section of this thesis.

2. In Chapter 4 we discuss the concrete security of the submissions to the NIST post-quantum standardisation process. We highlight the importance of lattice reduction cost models and estimate the security of each parameter set for every scheme under all cost models considered as part of a first round submission. This technique allows for the security of two given schemes to be compared in a fair manner. As part of this work, we observe that lattice-reduction cost models are not *order preserving*, meaning that if scheme A is harder to break than scheme B under cost model 1, the same is not necessarily true under cost model 2. Moreover, we provide an update regarding the current state of the standardisation process. This chapter corresponds to the publication *Estimate all the {LWE, NTRU} schemes!* detailed in the List of Publications section of this thesis.

3. In Chapter 5 we discuss homomorphic encryption standardisation, and focus on the security of Sparse-secret LWE parameter sets. We outline the current state of the HE standard, and consider potential extensions to this standard, including sparse secret distributions. We outline parameter sets which balance security and efficiency, with a focus on the cost of the expensive bootstrapping procedure required in fully homomorphic encryption schemes. This chapter corresponds to the publication *On the feasibility and impact of standardising sparse-secret LWE* detailed in the List of Publications section of this thesis.

4. In Chapter 6 we discuss an application of homomorphic encryption: the private outsourcing of Kriging interpolation. Kriging is a spatial interpolation algorithm which provides the best linear unbiased prediction (BLUP) of an observed phenomenon, by taking a weighted average of samples within a specified neighbourhood. Kriging is widely used in areas such as geo-statistics where, as an example, it may be used to predict the quality of mineral deposits at an unobserved location based on previous measurements. In our work, we tweak the underlying algorithms to allow this process to be securely outsourced in an efficient manner. In particular, we build a construction which allows for the Kriging process to carried out when the measurement values are encrypted via a

homomorphic encryption scheme. This chapter corresponds to the publication *Private outsourced kriging interpolation* detailed in the List of Publications section of this thesis.

1.6 Source Code

Where possible, source code used as part of this thesis has been made publicly available.

1. The code used in Chapter 2 is available here: `github.com/bencrts/thesis/code/background`
2. The code used in Chapter 3 is available here: `github.com/bencrts/thesis/code/batchbdd`
3. The code used in Chapter 4 is available here: `github.com/estimate-all-the-lwe-ntru-schemes/estimate-all-the-lwe-ntru-schemes.github.io` and here: `github.com/bencrts/thesis/code/nist`
4. The code used in Chapter 5 is available here: `github.com/bencrts/thesis/code/hestandard`
5. The code used in Chapter 6 is available here: `github.com/bencrts/thesis/code/kriging`

In the event that any of the above links become broken, please contact the author for a copy of the desired source code.

Chapter 2

Background and Notation

Contents

In this chapter, we introduce all relevant background mathematics required in this thesis. We introduce the notion of a lattice and consider various properties of lattices, as well as heuristics considered in lattice-based cryptography and cryptanalysis.

2.1 Mathematical Objects

Euclidean space of dimension n is denoted $\mathbb{R}^n$. All logarithms are to the base two, unless otherwise stated. Column vectors are denoted by lower case bold letters, e.g $\mathbf{b}$, and matrices by upper case bold letters, e.g. $\mathbf{B}$. The transpose of the matrix $\mathbf{B}$ is denoted by $\mathbf{B}^\mathsf{T}$. The i^{th} component of a vector $\mathbf{b}$ is denoted by b_i, and the $(i,j)^{th}$ entry of a matrix $\mathbf{B}$ is denoted by $B_{i,j}$, where all indices start from one. We write $\mathbf{B}_i$ for the i^{th} column of $\mathbf{B}$. The inner product of two vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ is written as $\langle \mathbf{b}_1, \mathbf{b}_2 \rangle$. We write $(\mathbf{b}_1, \mathbf{b}_2)$ for the vector formed by concatenating the entries of the two vectors $\mathbf{b}_1$ and $\mathbf{b}_2$. The same notation is used for the concatenation of the vector $\mathbf{b}_1$ and the scalar c as $(\mathbf{b}_1, c)$. Similarly, we denote the concatenation of k column vectors, each of length d, into a $(d \times k)$ matrix as $[\mathbf{b}_1 \mid \mathbf{b}_2 \mid \cdots \mid \mathbf{b}_k]$. We write $\mathbf{B}_{(\tau)}$ to represent the $d \times (k-\tau)$ sub-matrix of $\mathbf{B}$ constructed via dropping the first τ columns of $\mathbf{B}$, i.e. $\mathbf{B}_{(\tau)} = [\mathbf{b}_{\tau+1} \mid \mathbf{b}_{\tau+2} \mid \cdots \mid \mathbf{b}_k]$. Similarly we use $\mathbf{b}_{(\tau)}$ to denote dropping the first τ components of the vector $\mathbf{b}$, i.e. $\mathbf{b}_{(\tau)} = (b_{\tau+1}, b_{\tau+2}, \ldots, b_k)$. We identify polynomials $f = \sum_{i=1}^{n} f_i x^{i-1}$ with their coefficient vectors $\mathbf{f} = (f_1, f_2, \ldots, f_n)$. The Euclidean norm of a vector $\mathbf{v}$ is defined to be $\|\mathbf{v}\| = \sqrt{\sum_{i=1}^{n} v_i^2}$.

Definition 2.1 (Discrete Gaussian Distribution) *A discrete Gaussian distribution centred at μ and with width parameter r samples elements with probability:*

$$\exp\left(-\pi \frac{(x-\mu)^2}{r^2}\right).$$

Throughout this thesis, we typically consider discrete Gaussian distributions over the integers to be centred at zero (i.e. $\mu = 0$). The standard deviation of such a distribution is $\sigma = \frac{r}{\sqrt{2\pi}}$, and we denote this distribution by D_σ.

Throughout this thesis the term *ring* is reserved for a *commutative ring* R with a multiplicative identity element 1_R (we denote the *additive identity* element of R by 0_R).

Example 2.1 *For an integer q, we consider values in the set $\mathbb{Z} \cap (-q/2, q/2]$ to be representatives of $\mathbb{Z}_q$. For example, if $x \in \mathbb{Z}_2$, x can take values in the set:*

$$\mathbb{Z}_2 = \{0, 1\}$$

and, if $x \in \mathbb{Z}_3$, x can take values in the set:

$$\mathbb{Z}_3 = \{-1, 0, 1\}.$$

Definition 2.2 (Ideal) *The set $\mathcal{I} \neq \emptyset$ is an ideal of the ring R if:*

1. *I is a subgroup of the additive group of R, and*
2. *for each $r \in R$ and $x \in I$ we have that $xr \in I$.*

Definition 2.3 (Field) *A field K is a ring which also has the property that it is closed under multiplicative inverses, i.e. the non-zero elements of K form a multiplicative group.*

Example 2.2 *For a ring R, an ideal generated by a single ring element $f \in R$ is denoted by (f), and this ideal is made up of elements in the set:*

$$(f) = \{fr \mid r \in R\}.$$

Definition 2.4 (Maximal Ideal) *A proper ideal $I \subsetneq R$ of a ring R is maximal if for any ideal J satisfying $I \subseteq J$, we have either $J = I$ or $J = R$.*

Definition 2.5 (Quotient Ring) *Given a ring R and an ideal I of R, we can define the quotient ring R/I which is formed of cosets $r + I$, for $r \in R$, of the additive group of I in R. We define the addition and multiplication operations in the quotient ring R/I in the following way:*

$$(I + r_1) + (I + r_2) = I + (r_1 + r_2)$$
$$(I + r_1)(I + r_2) = (I + r_1 r_2),$$

for all $r_1, r_2 \in R$.

We can think of a quotient ring as a set of *equivalence classes* under the equivalence relation:

$$[x] \sim [y] \text{ if and only if } x - y \in I.$$

That is, two elements $x, y \in R/I$ are equivalent if and only if their difference $x - y$ is contained within the ideal I. We note that an ideal I in the ring R is *maximal* if and only if the quotient ring R/I is a field. A *ring homomorphism* ϕ is a structure-preserving map between two rings R_1 and R_2.

Definition 2.6 (Ring Homomorphism) *The map $\phi : R \to S$ is a ring homomorphism from the ring R to the ring S if it satisfies the three conditions:*

1. $\phi(1_R) = 1_S$,
2. $\phi(r + s) = \phi(r) + \phi(s)$, *and*
3. $\phi(rs) = \phi(r)\phi(s)$,

for all $r \in R$ and $s \in S$, where 1_R and 1_S are the multiplicative identity elements of the rings R and S respectively.

Throughout this thesis we are particularly interested in polynomial rings of the form $\mathbb{Z}_q[X]/(f)$ for some polynomial $f \in \mathbb{Z}_q[X]$. In this case, two elements $g, h \in \mathbb{Z}_q[X]/(f)$ are in the same equivalence class if and only if $g - h \in (f)$, i.e. $g - h = fk$ for some $k \in \mathbb{Z}[X]$. Multiplication in polynomial quotient rings works as in typical polynomial multiplication, with the additional condition that we work modulo the quotient polynomial f. For a given ring R, we denote the ring R/qR as R_q.

When we consider elements $a \in R_q$, we can represent a as a vector $\mathbf{a}$ of n coefficients in $\mathbb{Z}_q$. Therefore, for two ring elements $a, b \in R_q$, we can compute the polynomial sum $a + b$ by considering an element-wise addition of the vectors $\mathbf{a}$ and $\mathbf{b}$. That is:

$$\mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n),$$

and $\mathbf{a} + \mathbf{b}$ is the coefficient vector of $a + b$. For multiplication, the situation is a little more complicated: multiplication of two ring elements, i.e. ab corresponds to matrix/vector multiplication. That is, there exists a matrix $\mathbf{P}_a$ such that, for any ring element $b \in R_q$ we can compute the product $c = ab$ via:

$$\mathbf{c} = \mathbf{P}_a \mathbf{b}.$$

More specifically, the matrix $\mathbf{P}_a$ is of the form:

$$\mathbf{P}_a = \begin{pmatrix} a_1 & a_2 & \ldots & a_n \\ (xa)_1 & (xa)_2 & \ldots & (xa)_n \\ \vdots & \vdots & \ddots & \vdots \\ (x^{n-1}a)_1 & (x^{n-1}a)_2 & \ldots & (x^{n-1}a)_n \end{pmatrix},$$

where $(x^i a)_j$ is the j^{th} component of the coefficient vector of the polynomial $x^i a$.

Example 2.3 *Let $R_q = \mathbb{Z}_3[X]/(x^5-1)$. This ring can be identified with the set of polynomials with coefficients in $\mathbb{Z}_3$ up to degree 4, i.e. the set:*

$$\{a_1 + a_2x + a_3x^2 + a_4x^3 + a_5x^4 \mid a_i \in \mathbb{Z}_3, 1 \leq i \leq 5\}.$$

Consider the polynomial $g = a_1 + a_2x + a_3x^2 + a_4x^3 + a_5x^4$. We note that:

$$\begin{aligned} xg &= x(a_1 + a_2x + a_3x^2 + a_4x^3 + a_5x^4) \\ &= a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 \\ &= a_5 + a_1x + a_2x^2 + a_3x^3 + a_4x^4, \end{aligned}$$

since $x^5 \equiv 1 \bmod (x^5 - 1)$. In this case, the matrix $\mathbf{P}_g$ is given by:

$$\mathbf{P}_g = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 \\ a_5 & a_1 & a_2 & a_3 & a_4 \\ a_4 & a_5 & a_1 & a_2 & a_3 \\ a_3 & a_4 & a_5 & a_1 & a_2 \\ a_2 & a_3 & a_4 & a_5 & a_1 \end{pmatrix}.$$

Definition 2.7 (Irreducible Polynomial) *Given a field K, a non-constant polynomial $f \in K[X]$ is irreducible over K if it cannot be factored into two non-constant polynomials $g, h \in K[X]$.*

In typical examples considered in this thesis, the (non-constant) polynomial f is irreducible in $\mathbb{Z}_q[X]$, and we consider rings of the form $\mathbb{Z}_q[X]/(f)$.

Example 2.4 *The polynomial $f = X^2 + 1$ is irreducible over $\mathbb{Q}$ since we cannot write:*

$$X^2 + 1 = (X + a)(X + b),$$

with $a, b \in \mathbb{Q}$. We note that f is not irreducible over $\mathbb{C}$ since we can write f as:

$$X^2 + 1 = (X + i)(X - i).$$

Example 2.4 motivates the discussion of an *extension field*.

Definition 2.8 (Extension Field) *If F, K are two fields such that F is a subfield of K, then we refer to K as an* extension field *of F.*

Example 2.5 *Consider the field $\mathbb{Q}$. Clearly $\pm\sqrt{2} \notin \mathbb{Q}$, and so the polynomial $p = x^2 - 2 \in \mathbb{Q}[X]$ is irreducible over $\mathbb{Q}$. Since p is irreducible over $\mathbb{Q}$, the ideal (p) in $\mathbb{Q}[X]$ is maximal and therefore the quotient ring $R = \mathbb{Q}[X]/(x^2 - 2)$ is a field. Since $\mathbb{Q}$ is a subfield of R, we have that R is an extension field of $\mathbb{Q}$.*

We also consider the notion of a *module* M over a ring R.

Definition 2.9 (Module) *Given a ring R, an abelian group M is called an R-*Module *if there exists an operation $\odot : R \times M \to M$ such that, for all $m, n \in M$ and $r, s \in R$, we have:*

1. $r \odot (m + n) = r \odot m + r \odot n$,
2. $(r + s) \odot m = r \odot m + s \odot m$,
3. $(r \cdot s) \odot m = r \odot (s \odot m)$, *and*
4. $1_R \odot m = m$,

where $\cdot, +$ denote the regular ring operations.

We will consider various cryptographic constructions built over rings and modules. In a similar manner to the notion of a vector space K^d built considering d-tuples of field elements from K, we can build a module by forming tuples of elements from a ring R.

Example 2.6 *Consider the ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. We define:*

$$M := \{(x_1, x_2, \ldots, x_k) \mid x_1, x_2, \ldots, x_k \in R_q\}.$$

The set M is an R_q-module since for $s \in R_q$ and $(x_1, x_2, \ldots, x_k) \in M$ we can define a multiplication operation $\odot : R_q \times M \to M$ such that:

$$s \odot (x_1, x_2, \ldots, x_k) = (sx_1, sx_2, \ldots, sx_k),$$

leading to the four conditions in Definition 2.9 being satisfied. In this example, k is referred to as the module rank*. Setting $k = 1$, we can see that R_q itself is an R_q-module.*

2.2 Lattices

Definition 2.10 (Lattice) *A lattice L in $\mathbb{R}^d$ is the set:*

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

of all integer combinations of a set of n linearly independent vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ in $\mathbb{R}^d$, where $d \geq n$.

The integer n is known as the *rank* of the lattice, and the integer d is the *dimension* of the lattice. If $n = d$ then the lattice is known as *full rank*. The set of linearly independent column vectors $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n\}$ is known as a *basis* of the lattice, and is typically represented in matrix form $\mathbf{B} = [\mathbf{b}_1 \mid \mathbf{b}_2 \mid \cdots \mid \mathbf{b}_n] \in \mathbb{R}^{d \times n}$. We write $L(\mathbf{B})$ to denote the lattice generated by the columns of the matrix $\mathbf{B}$. In particular, this lattice is made up of the set:

$$L(\mathbf{B}) = \{\mathbf{B}\mathbf{v} \mid \mathbf{v} \in \mathbb{Z}^n\}.$$

In Figure 2.1 we present an example of two different bases of the same lattice.

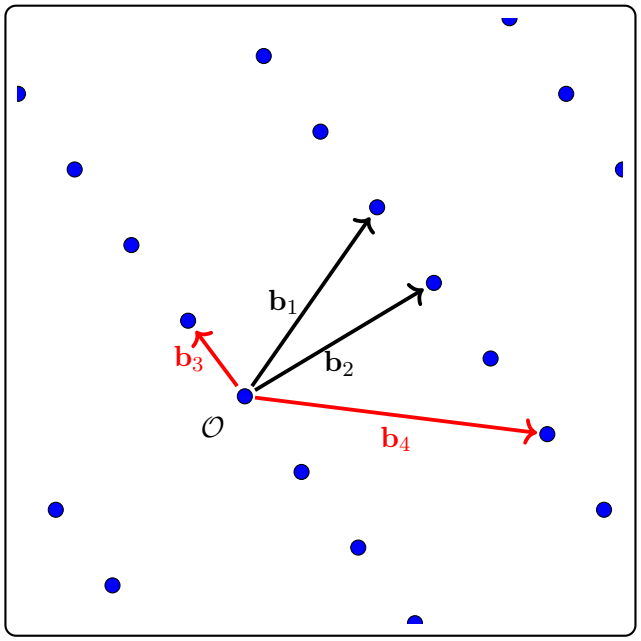


Figure 2.1: The vectors $\{\mathbf{b}_1, \mathbf{b}_2\}$ form a basis of the lattice consisting of the blue points, as do the vectors $\{\mathbf{b}_3, \mathbf{b}_4\}$.

Definition 2.11 (Span) *The span of a set of vectors S over a field K is defined to be:*

$$\mathsf{span}(S) = \left\{ \sum_{i=1}^{\ell} c_i s_i \middle| \ell \in \mathbb{N}, c_i \in K, s_i \in S \right\}.$$

That is, the span of the set S is the set of finite linear combinations of elements of S, with coefficients drawn from the underlying field K. Given a full-rank lattice L, the *Dual lattice* L^* is the set of vectors in the span of L which have integer inner product with all lattice points in L.

Definition 2.12 (Dual Lattice) *Let L be a full rank lattice of dimension d. The dual lattice of L is defined to be:*

$$L^* = \{\mathbf{v} \in \mathsf{span}(L) \mid \forall \mathbf{w} \in L, \langle \mathbf{v}, \mathbf{w} \rangle \in \mathbb{Z}\}.$$

It can be shown that the dual lattice is a lattice.

Proposition 2.1 *Let L be a lattice with basis $\mathbf{B}$. The dual lattice of $L(\mathbf{B})$, denoted by $L(\mathbf{B})^*$, is a lattice with basis $\mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}$.*

To prove this result, we show that $L(\mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}) \subseteq L(\mathbf{B})^*$ and $L(\mathbf{B})^* \subseteq L(\mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1})$. We will need to use the fact that:

$$\begin{aligned}(\mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1})^\top\mathbf{B} &= ((\mathbf{B}^\top\mathbf{B})^{-1})^\top\mathbf{B}^\top\mathbf{B} \\ &= ((\mathbf{B}^\top\mathbf{B})^{-1})\mathbf{B}^\top\mathbf{B} \\ &= \mathbf{I}\end{aligned}$$

Proof. Set $\mathbf{C} = \mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}$. Note that:

$$\begin{aligned}\mathbf{C}\mathbf{x} &= \mathbf{B}(\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{x} \\ &= \mathbf{B}((\mathbf{B}^\top\mathbf{B})^{-1}\mathbf{x})\end{aligned}$$

and therefore we have that $\mathbf{C}\mathbf{x} \in \mathsf{span}(L(\mathbf{B}))$. Moreover, for $\mathbf{y} = \mathbf{C}\mathbf{x} \in \mathsf{span}(L(\mathbf{B}))$ and

$\mathbf{w} = \mathbf{B}\mathbf{v} \in L(\mathbf{B})$, we have:

$$\begin{aligned}\langle \mathbf{y}, \mathbf{w} \rangle &= \langle \mathbf{C}\mathbf{x}, \mathbf{w} \rangle \\ &= \langle \mathbf{x}, \mathbf{C}^\top \mathbf{w} \rangle \\ &= \langle \mathbf{x}, \mathbf{C}^\top \mathbf{B}\mathbf{v} \rangle \\ &= \langle \mathbf{x}, (\mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1})^\top \mathbf{B}\mathbf{v} \rangle \\ &= \langle \mathbf{x}, \mathbf{v} \rangle \in \mathbb{Z}\end{aligned}$$

and, therefore, we have that $\mathbf{C}\mathbf{x} \in L(\mathbf{B})^*$ by Definition 2.12. This gives us $L(\mathbf{C}) \subseteq L^*(\mathbf{B})$.

To prove that $L(\mathbf{B})^* \subseteq L(\mathbf{C})$, we consider $\mathbf{z} \in L(\mathbf{B})^*$ and aim to show that $\mathbf{z} \in L(\mathbf{C})$. Since $\mathbf{z} \in L(\mathbf{B})^*$, we have that $\mathbf{z} \in \mathsf{span}(L(\mathbf{B}))$ and, for all $\mathbf{r} \in L(\mathbf{B})$, we have $\langle \mathbf{z}, \mathbf{r} \rangle \in \mathbb{Z}$ by Definition 2.12. Further, we have that $\mathbf{z} = \mathbf{B}\mathbf{s}$ for some vector $s \in \mathbb{R}^d$, and, therefore:

$$\begin{aligned}\mathbf{z} &= \mathbf{B}\mathbf{s} \\ &= \mathbf{B}(\mathbf{B}^\top \mathbf{B})^{-1}(\mathbf{B}^\top \mathbf{B})\mathbf{s} \\ &= \mathbf{C}(\mathbf{B}\mathbf{z})\end{aligned}$$

and, since each component of $\mathbf{B}\mathbf{z}$ is an integer, we have $L(\mathbf{B})^* \subseteq L(\mathbf{C})$. Therefore, we have $L(\mathbf{B})^* \subseteq L(\mathbf{C})$, as required. □

We will also encounter q-ary lattices when considering cryptanalytic attacks on lattice-based cryptosystems.

Definition 2.13 (q-ary Lattice) *A q-ary lattice is a lattice L which satisfies:*

$$q\mathbb{Z}^d \subseteq L \subseteq \mathbb{Z}^d,$$

for some integers q, d.

Example 2.7 *For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, where $q, m, n \in \mathbb{N}$, the lattice:*

$$L_q(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{v}\mathbf{A} = 0 \bmod q\},$$

is q-ary: clearly $L \subseteq \mathbb{Z}^n$, and if $\mathbf{y} \in q\mathbb{Z}^m$ then $\mathbf{y}\mathbf{A} \equiv \mathbf{0} \bmod q$ meaning that $q\mathbb{Z}^n \subseteq L$. Therefore, we have $q\mathbb{Z}^n \subseteq L \subseteq \mathbb{Z}^n$ as required.

Definition 2.14 (Unimodular Matrix) *Let* $\mathbf{U} \in \mathbb{Z}^{d \times d}$ *be a matrix with all integer entries. The matrix* $\mathbf{U}$ *is referred to as* unimodular *if the determinant of* $\mathbf{U}$ *is in the set* $\{1, -1\}$.

Proposition 2.2 *Let* $\mathbf{B}_1$ *and* $\mathbf{B}_2$ *be the matrix representations of two lattice bases. Then:*

$$L(\mathbf{B}_1) = L(\mathbf{B}_2),$$

if and only if $\mathbf{B}_1 = \mathbf{U}\mathbf{B}_2$ *for some unimodular matrix* $\mathbf{U}$.

There are infinitely many unimodular matrices of dimension $d \geq 2$. For dimension two we note that:

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc,$$

and there are infinitely many solutions to the integer equation $ad - bc = \pm 1$. The same holds true for all larger dimensions (by induction). For dimension $d \geq 2$, a lattice has infinitely many bases and there are a variety of proofs for this result. Informally, for full rank lattices, this can be seen as a combination of the facts that:

1. there are infinitely many unimodular matrices of dimension $d \geq 2$,
2. full rank lattices admit an invertible basis matrix (thus $\mathbf{U}_1\mathbf{B} = \mathbf{U}_2\mathbf{B} \iff \mathbf{U}_1 = \mathbf{U}_2$), and
3. Proposition 2.2.

Definition 2.15 (Fundamental Parallelepiped) *Given a lattice basis* $\mathbf{B}$*, we define the set:*

$$\mathcal{P}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^n \text{ such that } \forall i, 0 \leq x_i < 1\}.$$

as the fundamental parallelepiped *of the lattice basis* $\mathbf{B}$.

Note that the shape of this parallelepiped depends on the lattice basis $\mathbf{B}$ under consideration. Two such parallelepipeds can be seen in Figure 2.2.

We are also interested in the *shifted fundamental parallelepiped.*

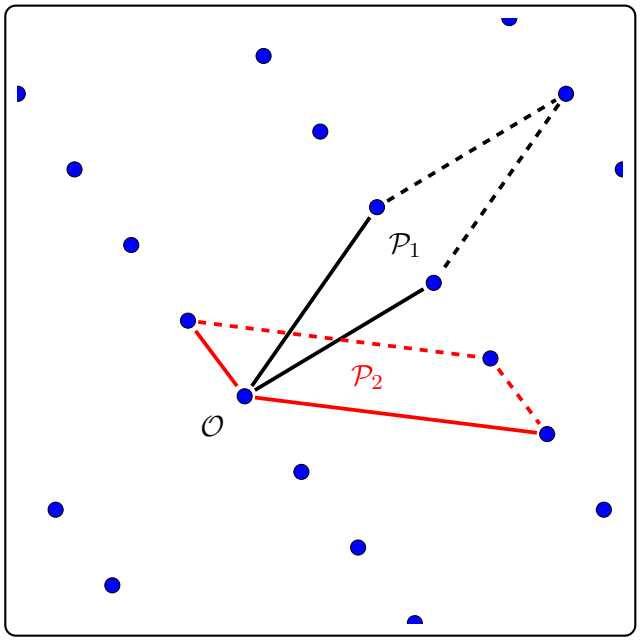


Figure 2.2: Two fundamental parallelepipeds, $\mathcal{P}_1$ and $\mathcal{P}_2$, of the bases $\mathbf{B}_1$ and $\mathbf{B}_2$.

Definition 2.16 (Shifted Fundamental Parallelepiped) *Given a lattice basis* $\mathbf{B}$*, we define the set:*

$$\mathcal{P}(\mathbf{B}) = \left\{ \mathbf{Bx} \middle| \mathbf{x} \in \mathbb{R}^n \text{ such that } \forall i, \frac{-1}{2} \leq x_i < \frac{1}{2} \right\}.$$

as the shifted fundamental parallelepiped of the lattice basis $\mathbf{B}$*.*

Definition 2.17 (Volume of a Lattice) *The volume of a lattice* $L(\mathbf{B})$ *is defined to be:*

$$\mathit{Vol}(L(\mathbf{B})) = \sqrt{\det(\mathbf{B}^\mathsf{T}\mathbf{B})}.$$

Note that if $\mathbf{B}_1 = \mathbf{UB}$ for some unimodular matrix $\mathbf{U} \in \mathbb{Z}^{(d \times d)}$, then:

$$\begin{aligned}
\mathsf{Vol}(L(\mathbf{B}_1)) &= \sqrt{\det(\mathbf{B}_1^\mathsf{T}\mathbf{B}_1)} \\
&= \sqrt{\det(\mathbf{B}^\mathsf{T}\mathbf{U}^\mathsf{T}\mathbf{UB})} \\
&= \sqrt{\det(\mathbf{B}^\mathsf{T}\mathbf{B})} \\
&= \mathsf{Vol}(L(\mathbf{B})),
\end{aligned}$$

since by definition $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}$. This is as expected, since $\mathbf{B}_1$ and $\mathbf{B}$ are bases of the same lattice, and volume is a lattice invariant.

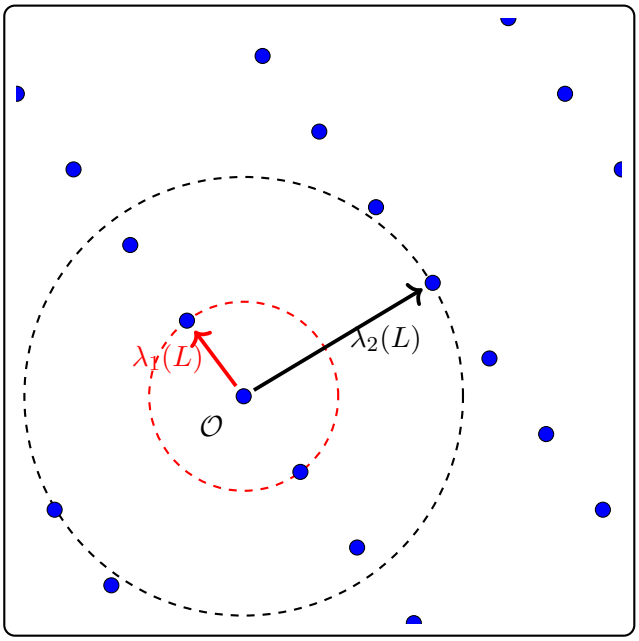


Figure 2.3: An example of $\lambda_1(L)$ and $\lambda_2(L)$ in the lattice represented by the blue dots.

Definition 2.18 (Successive Minima) *We denote by $\lambda_i(L)$ the i^{th} successive minima of the lattice L, i.e. the radius of the smallest ball, centred at the origin, containing at least i linearly independent lattice vectors.*

An example of $\lambda_1(L)$ and $\lambda_2(L)$ for a lattice L is given in Figure 2.3.

2.3 Gram-Schmidt Orthogonalisation

For a lattice basis $\mathbf{B} = [\mathbf{b}_1 \mid \mathbf{b}_2 \mid \cdots \mid \mathbf{b}_d]$, we can define the corresponding *Gram-Schmidt orthogonalised* (GSO) vectors $\mathbf{B}^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_d^*\}$ as:

$$\mathbf{b}_1^* = \mathbf{b}_1$$
$$\mathbf{b}_i^* = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^*,$$

where the *Gram-Schmidt coefficients* $\mu_{i,j}$ are given by:

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

The vector $\mathbf{b}_i^*$ is the vector $\mathbf{b}_i$ with contributions in the directions of $\{\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_{i-1}^*\}$ removed, i.e. $\mathbf{b}_i^*$ is the component of the vector $\mathbf{b}_i$ which is *orthogonal* to the hyperplane defined by the set of vectors $\{\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_{i-1}^*\}$. This means that the vectors $\mathbf{b}_i^*$ and $\mathbf{b}_j^*$, for $i \neq j$, are pairwise orthogonal, i.e. $\langle \mathbf{b}_i^*, \mathbf{b}_j^* \rangle = 0$.

The volume of the lattice $L(\mathbf{B})$ can be computed as the product of the lengths of the GSO vectors:

$$\mathsf{Vol}(L(\mathbf{B})) = \prod_{i=1}^{d} \|\mathbf{b}_i^*\|,$$

and, as outlined earlier, can also be computed as the determinant of the *Gram matrix* $\mathbf{B}^\mathsf{T}\mathbf{B}$, comprising of the coefficients $\mu_{i,j}$, i.e. $(\mathbf{B}^\mathsf{T}\mathbf{B})_{i,j} = \mu_{i,j}$. We note that the vectors $\mathbf{B}^* = [\mathbf{b}_1^* \mid \mathbf{b}_2^* \mid \cdots \mid \mathbf{b}_d^*]$ in general do not form a basis of the lattice $L(\mathbf{B})$, however these two bases do *span* the same space i.e. $\mathsf{span}(\mathbf{B}) = \mathsf{span}(\mathbf{B}^*)$.

When considering cryptanalytic attacks against lattice-based cryptosystems, the lengths of the GSO vectors for a given lattice basis $\mathbf{B}$ are an important quantity, we which refer to as the *GSO profile.*

Definition 2.19 (GSO Profile) *For a lattice L of rank n, we define the GSO Profile of the lattice basis $\mathbf{B}$ to be the set of GSO lengths, that is:*

$$\{\|\mathbf{b}_i^*\| \mid 1 \leq i \leq d\}.^1$$

2.4 Projections

Definition 2.20 (Projection) *Given a vector $\mathbf{v}$ and a non-zero vector $\mathbf{u}$, we define the projection of $\mathbf{v}$ onto the direction of $\mathbf{u}$ to be:*

$$\pi_{\mathbf{u}}(\mathbf{v}) = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\langle \mathbf{u}, \mathbf{u} \rangle} \mathbf{u}.$$

We are particularly interested in projections onto, and orthogonal to, *basis vectors* of a lattice L.

[1] Note that the GSO profile is sometimes defined in terms of the *squared* GSO lengths.

Definition 2.21 (Parallel Projection) *Given a lattice with basis* $\mathbf{B}$ *we write* $\pi^{\parallel}_{i,\mathbf{B}}(\mathbf{x})$, $1 \leq i \leq d$, *to denote the* parallel projection *of* $\mathbf{x}$ *onto the space spanned by the set of vectors* $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}\}$, *that is:*

$$\pi^{\parallel}_{i,\mathbf{B}}(\mathbf{x}) = \sum_{j=1}^{i-1} \pi_{\mathbf{b}^*_j}(\mathbf{x}) = \sum_{j=1}^{i-1} \frac{\langle \mathbf{b}^*_j, \mathbf{x} \rangle}{\langle \mathbf{b}^*_j, \mathbf{b}^*_j \rangle} \cdot \mathbf{x}.$$

Definition 2.22 (Orthogonal Projection) *Given a lattice with basis* $\mathbf{B}$, *we write* $\pi^{\perp}_{i,\mathbf{B}}(\mathbf{x})$, $1 \leq i \leq d$, *to denote the* orthogonal projection *of* $\mathbf{x}$ *onto the space spanned by the vectors* $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{i-1}\}$, *that is:*

$$\begin{aligned} \pi^{\perp}_{i,\mathbf{B}}(\mathbf{x}) &= \mathbf{x} - \pi^{\parallel}_{i,\mathbf{B}}(\mathbf{x}) \\ &= \mathbf{x} - \sum_{j=1}^{i-1} \frac{\langle \mathbf{b}^*_j, \mathbf{x} \rangle}{\langle \mathbf{b}^*_j, \mathbf{b}^*_j \rangle} \cdot \mathbf{x}. \end{aligned}$$

Note that $\pi^{\perp}_{1,\mathbf{B}}(\mathbf{x})$ is the identity function, i.e. $\pi^{\perp}_{1,\mathbf{B}}(\mathbf{x}) = \mathbf{x}$, and $\pi^{\parallel}_{1,\mathbf{B}}(\mathbf{x}) = \mathbf{0}$. Moreover, since the lattice basis is clear from context, we usually drop the basis $\mathbf{B}$ and write $\pi^{\perp}_i$ or $\pi^{\parallel}_i$. Since the orthogonal projection is usually the projection of interest, we sometimes write $\pi^{\perp}_i := \pi_i$.

Example 2.8 *Mapping* $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ *to the point* $\mathbf{y} = (x_1, x_2, \ldots, x_{n-1}, 0)^T$ *is a parallel projection onto the space spanned by the set of unit vectors* $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n-1}\}$. *Such a projection can be represented by the matrix:*

$$\mathbf{P} = \begin{pmatrix} \mathbf{I}_{n-1} & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

so that $\mathbf{y} = \mathbf{P}\mathbf{x}$. *The corresponding orthogonal protection onto the space spanned by the set of unit vectors* $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{n-1}\}$, *which takes* $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ *to the point* $\mathbf{y} = (0, 0, \ldots, 0, x_n)^T$ *can be represented by the matrix:*

$$\mathbf{Q} = \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{I}_1 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

so that $\mathbf{y} = \mathbf{Q}\mathbf{x}$.

We are particularly interested in projecting orthogonally to a subset of basis vectors for some lattice $L(\mathbf{B})$, which generates a *projected sublattice*:

$$\Lambda_i(\mathbf{B}) = \{\pi^{\perp}_i(\mathbf{x}) \mid \mathbf{x} \in L(\mathbf{B})\},$$

we note that $\Lambda_i(\mathbf{B})$ is indeed a sublattice of $L(\mathbf{B})$, with a basis given by:

$$\{\pi_i^{\perp}(\mathbf{b}_i), \pi_i^{\perp}(\mathbf{b}_{i+1}), \ldots, \pi_i^{\perp}(\mathbf{b}_d)\}.$$

Note that $\pi_1^{\perp}(\mathbf{B}) = L(\mathbf{B})$ as is expected. By considering the projection function outlined in Definition 2.22, we note that we can write the Gram-Schmidt vectors as:

$$\begin{aligned}\mathbf{b}_i^* &= \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j}\mathbf{b}_j^* \\ &= \mathbf{b}_i - \sum_{j=1}^{i-1} \pi_{\mathbf{b}_j^*}(\mathbf{b}_j) \\ &= \pi_i^{\perp}(\mathbf{b}_i).\end{aligned}$$

2.5 Hard Problems in Lattices

We are interested in several of the hard problems in lattices which underpin lattice-based cryptography. We first consider the *Shortest Vector Problem* which challenges us to find the shortest vector in a lattice.

Definition 2.23 (Shortest Vector Problem (SVP)) *Given a lattice basis* $\mathbf{B}$*, find a shortest non-zero lattice vector in* $L(\mathbf{B})$*, i.e. a vector* $\mathbf{x} \in L(\mathbf{B})$ *of length* $\|\mathbf{x}\| = \lambda_1(L(\mathbf{B}))$*.*

We can also define a selection of problems closely related to the shortest vector problem. The γ*-Approximate Shortest Vector Problem* gives us a bound γ and asks us to find a vector whose length is upper-bounded by $\gamma\lambda_1(L(\mathbf{B}))$.

Definition 2.24 (γ-Approximate Shortest Vector Problem (Approx-SVP)) *Given a lattice basis* $\mathbf{B}$ *and an approximation factor* γ*, find a non-zero lattice vector in* $\mathbf{x} \in L(\mathbf{B})$ *of length* $\|\mathbf{x}\| \leq \gamma\lambda_1(L(\mathbf{B}))$*.*

The γ-unique Shortest Vector Problem (uSVP) provides a guarantee that a shortest vector exists, is unique (up to sign), and is significantly shorter (in particular, by a factor of γ) than the second shortest vector $\lambda_2(L)$ in the lattice L.

Definition 2.25 (γ-unique Shortest Vector Problem (uSVP)) *Given a lattice basis* $\mathbf{B}$ *guaranteed to contain a non-zero shortest vector satisfying* $\lambda_2(L(\mathbf{B}))/\lambda_1(L(\mathbf{B})) = \gamma$*, find this shortest vector.*

The Bounded Distance Decoding Problem (BDD) provides us with a public lattice basis $L(\mathbf{B})$, as well as a target point $\mathbf{t}$, and a parameter γ. Our goal is to find the closest lattice vector to the given target point, under the guarantee that the target point is very close to the lattice.

Definition 2.26 (Distance Between a Point and a Lattice) *We define* $\mathit{dist}(\mathbf{t}, L(\mathbf{B}))$ *to be the distance between the target point* $\mathbf{t}$ *and the closest point to* $\mathbf{t}$ *contained within the lattice* $L(\mathbf{B})$*, that is:*

$$\mathit{dist}(\mathbf{t}, L(\mathbf{B})) = \min\{\|\mathbf{x} - \mathbf{t}\| \mid \mathbf{x} \in L(\mathbf{B})\}.$$

Definition 2.27 (γ-Bounded Distance Decoding Problem ($\mathbf{BDD}_\gamma$)) *Given the basis of a lattice* $L(\mathbf{B})$*, a vector* $\mathbf{t} \notin L(\mathbf{B})$*, and a parameter* $0 < \gamma$ *such that* $\mathit{dist}(\mathbf{t}, L(\mathbf{B})) < \gamma\lambda_1(\mathbf{B})$*, find the lattice vector* $\mathbf{v} \in L(\mathbf{B})$ *which is closest to* $\mathbf{t}$*.*

The Short Integer Solutions Problem (SIS) asks us to find a vector below a certain length, which is in the left kernel of a given, public, matrix.

Definition 2.28 (γ-Shortest Integer Solutions Problem (SIS)) *Let* $\mathbf{A} \in \mathbb{Z}_q^{(m \times n)}$ *be a uniformly random matrix. Find a non-zero vector* $\mathbf{x} \in \mathbb{Z}_q^m$ *satisfying* $\|\mathbf{x}\| \leq \gamma$ *and* $\mathbf{xA} \equiv \mathbf{0} \bmod q$*.*

We are interested in lattice-based cryptography which is built on the hardness of two families of problems: the *NTRU problem* and the *Learning with Errors Problem*.

Definition 2.29 (NTRU [HPS96]) *Let* n, q *be positive integers,* $\phi \in \mathbb{Z}[x]$ *be a monic polynomial of degree* n*, and* $R_q = \mathbb{Z}_q[x]/(\phi)$*. Let* $f, g \in R_q$*, with* f *invertible, be small polynomials (i.e. having small coefficients) and* $h = g \cdot f^{-1} \in R_q$*.*

- *Search-NTRU is the problem of recovering f or g given h.*
- *Decision-NTRU is the problem of deciding if $h = g \cdot f^{-1}$ or uniform.*

Typically, the quotient polynomial is chosen as $\phi = X^n + 1$. However, other instantiations of the NTRU problem (such as NTRUPrime [BCLv19]) consider alternative quotient polynomials, such as $\phi = X^n - X - 1$. An important object is the matrix $\mathbf{P}_h$ as defined in Section 2.1, for a choice of $\phi = X^n - 1$ this matrix takes the form:

$$\mathbf{P}_h = \begin{pmatrix} h_1 & h_2 & \dots & h_n \\ h_n & h_1 & \dots & h_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_2 & h_3 & \dots & h_1 \end{pmatrix},$$

where $h_1, \dots, h_n$ are the coefficients of the degree $(n-1)$ polynomial $h \in R_q$. In NTRU-based cryptographic constructions, the ring element h (and therefore the matrix $\mathbf{P}_h$) is typically made public. Representing this matrix requires storage of only n integers modulo q, i.e. $n \log_2(q)$ bits. Note that, for other choices of ϕ, this matrix can take a different form, related to the structure of multiplication by x^i.

We next define variants of the *Learning with Errors problem*. We begin by outlining the *Module Learning with Errors Problem* (Module-LWE), before defining the Ring Learning with Errors Problem (Ring-LWE) and the Learning with Errors Problem (LWE), which can both be viewed as special cases of Module-LWE.

Definition 2.30 (Module Learning with Errors (Module-LWE) [LS15]) *Let n, q, k be positive integers such that $d = n/k \in \mathbb{Z}$. Define the rings $R = \mathbb{Z}[X]/(X^d + 1)$ and $R_q = R/qR$. Let χ be a probability distribution on R and $\mathbf{s}$ be a secret module element in R_q^k.*

- *We define the Module-LWE Distribution $ML_{s,\chi,q}$ as the distribution on $R_q^k \times R_q$ given by choosing $\mathbf{a}_i \in R_q^k$ uniformly at random, choosing $e_i \in R$ according to χ and considering it as an element of R_q, and outputting $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) \in R_q^k \times R_q$.*
- *Search-Module-LWE is the problem of recovering the ring element s from a collection $\{(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_{i=1}^m$ of samples drawn according to $ML_{\mathbf{s},\chi,q}$.*

- *Decision-Module-LWE is the problem of distinguishing whether samples* $\{(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_{i=1}^{m}$ *are drawn from the Module-LWE distribution* $ML_{\mathbf{s},\chi,q}$ *or uniformly from* $R_q^k \times R_q$.

The distribution χ is typically a discrete Gaussian distribution over $\mathbb{Z}$, as defined in Definition 2.1, centred at zero and with width parameter αq. Here, the value α is referred to as the *LWE error rate*, and we recall that a discrete Gaussian distribution with width parameter αq has standard deviation $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$.

Note that a module element $\mathbf{v} \in R_q$ corresponds to a vector of k polynomials of degree d. Therefore, by considering the co-efficient vectors of the module elements, and recalling that $\mathbf{P}_a$ is the matrix representation of multiplication by the ring element a, we can represent a single Module-LWE sample, consisting of the k ring elements $a_1, a_2, \cdots, a_k \in R_q$, by the system of equations:

$$\mathbf{b} = [\mathbf{P}_{a_1} \mid \mathbf{P}_{a_2} \mid \cdots \mid \mathbf{P}_{a_k}]\mathbf{s} + \mathbf{e}.$$

More explicitly, for our choice of $f = X^d + 1$, we have that each $\mathbf{P}_{a_j}$ is of the form:

$$\mathbf{P}_{a_j} = \begin{pmatrix} a_1 & a_2 & a_3 & \cdots & a_{d-2} & a_{d-1} & a_d \\ -a_n & a_1 & a_2 & \cdots & a_{d-3} & a_{d-2} & a_{d-1} \\ -a_{n-1} & -a_n & a_1 & \cdots & a_{d-4} & a_{d-3} & a_{d-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -a_3 & -a_4 & -a_5 & \cdots & -a_d & a_1 & a_2 \\ -a_2 & -a_3 & -a_4 & \cdots & -a_{d-1} & -a_d & a_1 \end{pmatrix},$$

and, therefore, we have:

$$\mathbf{b} = \left(\begin{array}{cccc|c|cccc} a_{1,1} & a_{1,2} & \dots & a_{1,d} & \dots & a_{k,1} & a_{k,2} & \dots & a_{k,d} \\ -a_{1,d} & a_{1,1} & \dots & a_{1,d-1} & \dots & -a_{k,d} & a_{k,1} & \dots & a_{k,d-1} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -a_{1,2} & -a_{1,3} & \dots & a_{1,1} & \dots & -a_{k,2} & -a_{k,3} & \dots & a_{k,1} \end{array}\right) \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}.$$

Representing the matrix $\mathbf{A} = [\mathbf{P}_{a_1} \mid \mathbf{P}_{a_2} \mid \cdots \mid \mathbf{P}_{a_k}]$ requires storage of $n = dk$ integers modulo q, i.e. $n \log_2(q)$ bits. Setting the module rank $k = 1$ gives rise to the *Ring Learning with Errors Problem.*

Definition 2.31 (Ring Learning with Errors (Ring-LWE) [LPR10]) *Define the rings $R = \mathbb{Z}[X]/(X^n + 1)$ and $R_q = R/qR$. Let n, q be positive integers, χ be a probability distribution on R and s be a secret ring element in R_q.*

- *We define the Ring-LWE Distribution $RL_{s,\chi,q}$ as the distribution on $R_q \times R_q$ given by choosing $a_i \in R_q$ uniformly at random, choosing $e_i \in R$ according to χ and considering it as an element of R_q, and outputting $(a_i, a_i s + e_i) \in R_q \times R_q$.*
- *Search-Ring-LWE is the problem of recovering the ring element s from a collection $\{(a_i, a_i s + e_i)\}_{i=1}^m$ of samples drawn according to $RL_{\mathbf{s},\chi,q}$.*
- *Decision-Ring-LWE is the problem of distinguishing whether samples $\{(a_i, a_i s + e_i)\}_{i=1}^m$ are drawn from the Ring-LWE distribution $RL_{\mathbf{s},\chi,q}$ or uniformly from $R_q^d \times R_q$.*

In our definition of Ring-LWE we have considered the quotient ring $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ as a basis for the problem. In fact, we can define Ring-LWE with a variety of rings, although this can lead to security issues [Pei16]. A single Ring-LWE sample consists of $a_i s + e_i$ where $a_i, s, e_i \in R_q$ are all ring elements. As in Module-LWE, we can represent Ring-LWE samples in matrix/vector form:

$$\mathbf{b} = \mathbf{P}_a \mathbf{s} + \mathbf{e}.$$

More explicitly, for our choice of $f = X^n + 1$, we have:

$$\mathbf{b} = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ -a_n & a_1 & \cdots & a_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ -a_2 & -a_3 & \cdots & a_1 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}.$$

Representing the matrix $\mathbf{P}_a$ requires storage of n integers modulo q, i.e. $n \log_2(q)$ bits. Finally, if we consider the ring elements in R_q as vectors in $\mathbb{Z}_q^n$, and ignore the algebraic structure induced by the ring R_q, we arrive at the *Learning with Errors problem.*

Definition 2.32 (Learning with Errors (LWE) [Reg05]) *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 the* LWE Distribution $L_{\mathbf{s},\chi,q}$ *as the distribution on* $\mathbb{Z}_q^n \times \mathbb{Z}_q$ *given by choosing* $\mathbf{a}_i \in \mathbb{Z}_q^n$ *uniformly at random, choosing* $e_i \in \mathbb{Z}$ *according to* χ *and considering it as an element of* $\mathbb{Z}_q$*, and outputting* $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$*.*
- Search-LWE *is the problem of recovering the vector* $\mathbf{s}$ *from a collection* $\{(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_{i=1}^m$ *of samples drawn according to* $L_{\mathbf{s},\chi,q}$*.*
- Decision-LWE *is the problem of distinguishing whether samples* $\{(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_{i=1}^m$ *are drawn from the LWE distribution* $L_{\mathbf{s},\chi,q}$ *or uniformly from* $\mathbb{Z}_q^n \times \mathbb{Z}_q$*.*

If the components of the LWE secret follow the error distribution, then this is known as *normal-form* LWE. A single LWE sample consists of $\langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$, where the n coefficients of the vector $\mathbf{a}_i$ are drawn uniformly at random from $\mathbb{Z}_q$, and the scalar e_i is drawn from the error distribution χ. We can represent m LWE samples by the system of equations $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$, where:

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_m \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ \vdots \\ e_m \end{pmatrix}.$$

In LWE-based cryptographic constructions the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ are typically made public, whereas the secret $\mathbf{s}$ and the error $\mathbf{e}$ remain private. To represent the matrix $\mathbf{A}$ requires storage of $n \times m$ integers modulo q, i.e. $nm \log_2(q)$ bits.

To summarise, Module-LWE considers vectors of k polynomials over the ring $\mathbb{Z}[X]/(X^{n/k}+1)$. Setting the module rank $k = 1$, and considering elements of the ring $\mathbb{Z}[X]/(X^n + 1)$, gives rise to the Ring-LWE problem and, moreover, ignoring any structure induced by this ring, we can retrieve the Learning with Errors problem over $\mathbb{Z}_q^n$. Therefore, one can represent:

- a single Module-LWE sample by k Ring-LWE samples of the appropriate dimension,
- a single Ring-LWE sample by n LWE samples, and
- a single Module-LWE sample as nk LWE samples.

Throughout this thesis, we will always view Ring-LWE and Module-LWE samples simply as LWE samples of the appropriate dimension. Based on current knowledge, this approach is

reasonable since there are no known approaches to solve Ring-LWE and Module-LWE which exploits the additional algebraic structure in the ring.

2.5.1 Related Problems

There are a variety of problems closely related to the Learning with Errors problem. One variant of interest is the *Learning with Rounding Problem* [BPR12], which replaces the addition of a random error term e_i with a deterministic rounding process. Informally, this deterministic rounding process splits the space $\mathbb{Z}_q$ into p sections, each of size approximately $\frac{q}{p}$. Specifically, the operation:

$$\lfloor x \rceil_p = \left\lfloor \frac{p}{q} x \right\rceil,$$

is used.

Definition 2.33 (Learning with Rounding (LWR) [BPR12]) *Let $n, q, p < q$ be positive integers and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$.*

- *We define the LWR Distribution $L_{\mathbf{s},q,p}$ as the distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ given by choosing $\mathbf{a}_i \in \mathbb{Z}_q^n$ uniformly at random and outputting $(\mathbf{a}_i, \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$.*
- *Search-LWR is the problem of recovering the vector $\mathbf{s}$ from a collection $\{(\mathbf{a}_i, \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p)\}_{i=1}^m$ of samples drawn according to $L_{\mathbf{s},q,p}$.*
- *Decision-LWR is the problem of distinguishing whether samples $\{(\mathbf{a}_i, \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p)\}_{i=1}^m$ are drawn from the LWR distribution $L_{\mathbf{s},q,p}$ or uniformly from $\mathbb{Z}_q^n \times \mathbb{Z}_p$.*

We note that we can similarly define the *Ring-Learning with Rounding Problem* (Ring-LWR), and the *Module-Learning with Rounding Problem* (Module-LWR) in a similar manner to LWR and LWE. We can also view Learning with Rounding samples as LWE samples, where the error term is generated according to the parameters used in the rounding process as outlined above:

$$\lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rceil_p = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i,$$

for an appropriately chosen error term e_i. In Table 2.1, we provide a comparison of the LWE, Ring-LWE, Module-LWE, LWR, and NTRU problems.

Assumption	Secret	Error	Public Coefficients	Sample
NTRU	$f \in R_q$, invertible	$g \in R_q$	$h \in R_q$	$(h, g \cdot f^{-1}) \in R_q \times R_q$
LWE	$\mathbf{s} \in \mathbb{Z}_q^n$	$e_i \in \mathbb{Z}$	$\mathbf{a}_i \in \mathbb{Z}_q^n$	$(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$
Ring-LWE	$s \in R_q$	$e_i \in R$	$a_i \in R_q$	$(a_i, a_i s + e_i) \in R_q \times R_q$
Module-LWE	$\mathbf{s} \in R_q^k$	$e_i \in R$	$\mathbf{a}_i \in R_q^k$	$(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i) \in (R_q)^k \times R_q$
LWR	$\mathbf{s} \in \mathbb{Z}_q^n$	`n/a`	$\mathbf{a}_i \in \mathbb{Z}_q^n$	$(\mathbf{a}_i, \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rfloor_p) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$
Ring-LWR	$s \in R_q$	`n/a`	$a_i \in R_q$	$(a_i, \lfloor a_i s \rfloor_p) \in R_q \times R_p$
Module-LWR	$\mathbf{s} \in R_q^k$	`n/a`	$\mathbf{a}_i \in R_q^k$	$(\mathbf{a}_i, \lfloor \langle \mathbf{a}_i, \mathbf{s} \rangle \rfloor_p) \in R_q^k \times R_p$

Table 2.1: A comparison of variants of the Learning with Errors, Learning with Rounding, and NTRU problems.

2.5.2 Small Secrets

In many cases, for efficiency purposes, it can be useful to restrict the secret distribution of $\mathbf{s}$ from uniformly random over $\mathbb{Z}_q^n$ to some other distribution with a smaller support e.g. $\{0,1\}^n$. Several typical examples can be found in Definition 2.34, following the notation of [Alb17].

Definition 2.34 (Small Secret Distributions) *Let n, q be positive integers.*

- *$\mathcal{B}$ is any probability distribution on $\mathbb{Z}_q^n$ where each component is ≤ 1 in absolute value.*
- *$\mathcal{B}^+$ is the probability distribution on $\mathbb{Z}_q^n$ where each component is independently sampled uniformly at random from $\{0,1\}$.*
- *$\mathcal{B}^-$ is the probability distribution on $\mathbb{Z}_q^n$ where each component is independently sampled uniformly at random from $\{-1,0,1\}$.*
- *$\mathcal{B}_h^+$ is the probability distribution on $\mathbb{Z}_q^n$ where components are sampled uniformly at random from $\{0,1\}$ with the additional guarantee that at most h components are non-zero.*
- *$\mathcal{B}_h^-$ is the probability distribution on $\mathbb{Z}_q^n$ where components are sampled uniformly at random from $\{-1,0,1\}$ with the additional guarantee that exactly h components are non-zero.*

- $\mathcal{B}^{-}_{(h_1,h_2)}$ *is the probability distribution on* $\mathbb{Z}_q^n$ *where components are sampled uniformly at random from* $\{-1, 0, 1\}$ *with the additional guarantee that exactly* h_1 *components are equal to* -1 *and exactly* h_2 *components are equal to* 1.

We define the *Small-secret Learning with Errors* problem to be the same as in Definition 2.32 except the secret is drawn from one of the distributions defined in Definition 2.34. The hardness of Small-secret LWE has been studied in [BLP+13] and more recently in [Mic18]. The concrete loss of security which arises from the use of small secrets has been studied in [BGPW16, Alb17, ACD+18, Wun19, CP19]. Note that we can consider small-secrets in the context of any of the Module-LWE/R, Ring-LWE/R, or LWE/R problems.

2.6 SVP Solvers, CVP Solvers, and Lattice Reduction Algorithms

The security of Learning with Errors-based cryptosystems depends on the hardness of finding short vectors within q-ary lattices. There are two main families of algorithms used to find short vectors. The first are sieving algorithms [AKS01, LMvdP15, BDGL16], which use large lists of short vectors and iteratively generate new vectors by checking for shorter linear combinations within this list, requiring the use of exponential memory. The second are enumeration algorithms [Kan83, FP85, MW15], which perform a lattice-point search around a given target point and only require the usage of polynomial memory. There are a variety of estimates in the literature considering the running time of these SVP solvers, which are surveyed in Chapter 4.

Informally, a *lattice reduction algorithm* takes as input a public, 'bad', basis $\mathbf{B}$ of some lattice L and attempts to find a shorter and more orthogonal basis $\mathbf{B}'$ of the same lattice. As an example, the BKZ algorithm [SE94] does this by calling an SVP solver on projected sublattices of dimension β at a cost of:

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

Where $T_{\mathsf{SVP}}(\beta)$ is the cost of an SVP solver in dimension β, and c denotes the number of required calls to this SVP solver. The FPYLLL library [FPL20] contains implementations of the lattice reduction algorithms outlined in this section, and we will use this library throughout to provide examples.

2.6.1 LLL

The *Lenstra, Lenstra and Lovász (LLL) algorithm* [LLL82] was originally designed as an algorithm to perform polynomial factorisation. This algorithm has since been used to perform lattice reduction and, indeed, is the first polynomial time lattice reduction algorithm. In a lattice of dimension d, LLL finds an (approximate) shortest vector $\mathbf{v}$ whose length is at most $\gamma\lambda_1(L)$ where $\gamma = 2^{O(d)}$. Let $L(\mathbf{B})$ be the lattice with basis matrix $[\mathbf{b}_1 \mid \mathbf{b}_2 \mid \cdots \mid \mathbf{b}_d]$, GSO vectors $\mathbf{B}^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \cdots, \mathbf{b}_d^*\}$, and Gram-Schmidt coefficients $\mu_{i,j}$.

Definition 2.35 (LLL-reduced Basis) *We say that the basis* $\mathbf{B}$ *is* LLL-reduced *if the following two conditions are satisfied:*

1. *for* $1 \leq j < i \leq d$, *we have that* $|\mu_{i,j}| \leq \frac{1}{2}$, *and*
2. *for* $t = 1, 2, \ldots, d-1$, *and some* $\kappa \in (\frac{1}{4}, 1]$, *each pair of vectors indexed* $(t, t+1)$ *satisfies:*

$$\kappa\|\mathbf{b}_t^*\|^2 \leq \|\mathbf{b}_{t+1}^*\|^2 + \mu_{t+1,t}^2\|\mathbf{b}_t\|^2.$$

The first condition (size reduction) ensures that the lengths of the GSO vectors are in descending order i.e. $\|\mathbf{b}_t\| \geq \|\mathbf{b}_{t+1}^*\|$. The second (Lovász) condition can be re-arranged to:

$$(\kappa - \mu_{t+1,t}^2)\|\mathbf{b}_t^*\|^2 \leq \|\mathbf{b}_{t+1}^*\|^2,$$

and by the first condition we have that $|\mu_{t,t+1}| \leq \frac{1}{2}$, which gives us $\mu_{t,t+1}^2 \leq \frac{1}{4}$, and, since $\kappa \in (\frac{1}{4}, 1]$, this leads us to $(\kappa - \mu_{t+1,t}^2) \in (0, \frac{3}{4}]$. This provides a bound on the difference in length between successive Gram-Schmidt vectors, i.e. $\mathbf{b}_t^*$ and $\mathbf{b}_{t+1}^*$, for $1 \leq t \leq d-1$. Note that we can write:

$$\|\mathbf{b}_t^*\|^2 \leq \frac{1}{(\kappa - \mu_{t+1,t}^2)}\|\mathbf{b}_{t-1}^*\|^2 \leq \frac{1}{(\kappa - \frac{1}{4})}\|\mathbf{b}_{t-1}^*\|^2,$$

and, choosing e.g. $\kappa = \frac{1}{2}$, we can note the relation:

$$\|\mathbf{b}_t^*\|^2 \leq 4\|\mathbf{b}_{t-1}^*\|^2.$$

Intuitively, LLL works in the following way: we perform size reduction, i.e. iteratively compute:

$$\mathbf{b}_t \leftarrow \mathbf{b}_t - \sum_{j=1}^{t-1} \lfloor \mu_{t,j} \rceil \mathbf{b}_j,$$

and then we check that the Lovász condition is satisfied. If this condition is not satisfied at e.g. index ℓ, we swap the position of the vector $\mathbf{b}_\ell$ with the previous vector $\mathbf{b}_{\ell-1}$ and restart the process. We outline the LLL algorithm in Algorithm 1.

Input: A basis for the lattice $L(\mathbf{B})$ and $\kappa \in (\frac{1}{4}, 1]$
Result: A κ-LLL-reduced basis for the lattice $L(\mathbf{B})$
Compute the GSO vectors $\mathbf{b}_1^*, \mathbf{b}_2^*, \ldots, \mathbf{b}_d^*$
for *i = 2 to d* **do**
  for *j = i-1 to 1* **do**
    $\mathbf{b}_i \leftarrow \mathbf{b}_i - \lfloor \mu_{i,j} \rceil \mathbf{b}_j$
  end
end
if *there exists a t such that* $(\kappa - \mu_{t+1,t}^2)\|\mathbf{b}_t^*\|^2 \leq \|\mathbf{b}_{t+1}^*\|^2$ **then**
  Swap $\mathbf{b}_t$ and $\mathbf{b}_{t+1}$ and restart
end

Algorithm 1: The LLL Algorithm.

We discuss heuristics related to the LLL algorithm, such as the expected shape of the output basis, in Section 2.8.

2.6.1.1 Performing LLL in FPYLLL

We can use the FPYLLL library [FPL20] to perform LLL reduction in practice. Let us consider a q-ary lattice of dimension $d = 220$, with $q = 2^{15}$. This lattice has basis matrix given by:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_{110} & \mathbf{0} \\ \mathbf{A} & 2^{15} \cdot \mathbf{I}_{110} \end{pmatrix},$$

where the components of $\mathbf{A}$ are chosen uniformly at random from $\mathbb{Z}_q$. We perform LLL-reduction on such a lattice basis using the SageMath code in Figure 2.4, which generates an output GSO Profile as in Figure 2.5. Clearly, by considering Algorithm 1, we expect that the lengths of the GSO vectors will decrease as the index i increases, i.e. $\|\mathbf{b}_i^*\| \geq \|\mathbf{b}_{i+1}^*\|$ for $1 \leq i < d$.

```
from fpylll import *

A = IntegerMatrix.random(220,"qary", k=110, q=2**15)[::-1]
A = LLL.reduction(A)
M = GSO.Mat(A)
M.update_gso()
N = []

for i in range(len(M.r())):
    N.append((i, log(M.r()[i],2**15).n()))

for i in N:
    print(i)
```

Figure 2.4: Code used to generate a random q-ary lattice of dimension 220 with modulus $q = 2^{15}$ and determinant q^{110}, and perform LLL-reduction.

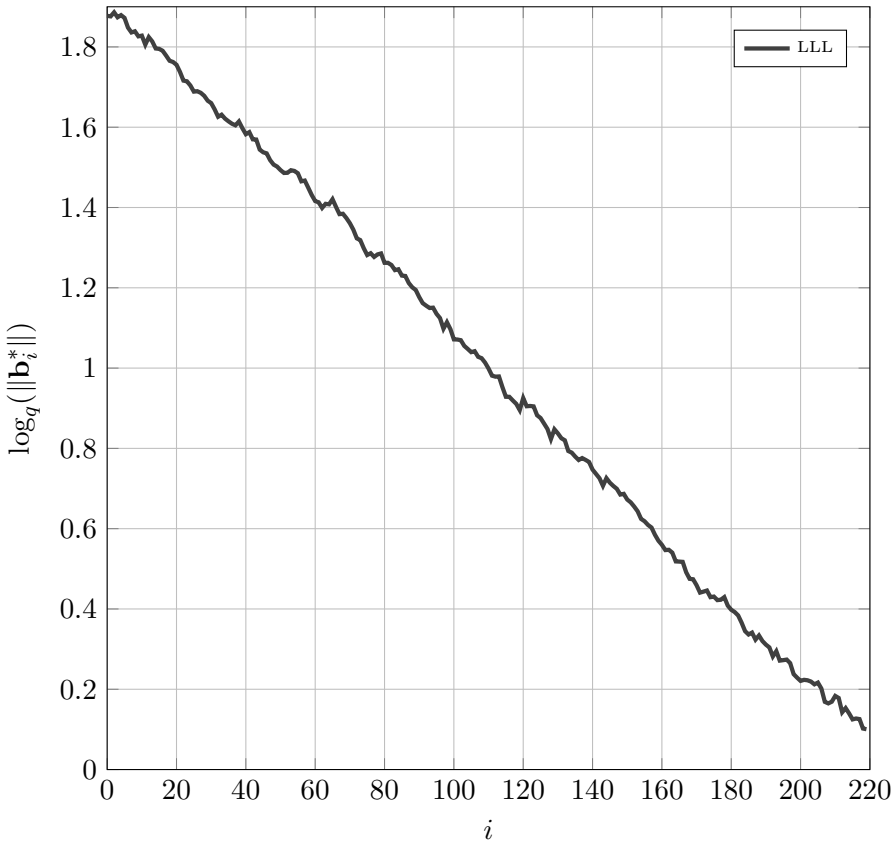


Figure 2.5: GSO lengths of a random q-ary lattice of dimension 220 with modulus $q = 2^{15}$ and determinant q^{110} which has been LLL-reduced using FPYLLL [FPL20].

2.6.2 BKZ

For the majority of this thesis the lattice reduction algorithm under consideration will be the *Block Korkine Zolotarev algorithm* (BKZ) [CN11]. The high level idea of BKZ is to produce a shorter lattice basis by finding the short vectors in projected sublattices of dimension β, referred to as the *blocksize*, as a subroutine. Let $L(\mathbf{B})$ be the lattice with basis matrix $\mathbf{B} = [\mathbf{b}_1 \mid \mathbf{b}_2 \mid \cdots \mid \mathbf{b}_n]$, and corresponding GSO vectors $\mathbf{B}^* = \{\mathbf{b}_1^*, \mathbf{b}_2^*, \cdots, \mathbf{b}_n^*\}$.

Definition 2.36 (HKZ-reduced Basis) *We say that the basis* $\mathbf{B}$ *is* HKZ-reduced *if the following two conditions are satisfied:*

1. *The lattice basis is size reduced, and*
2. *for* $i \in \{1, \ldots, d\}$ *we have that* $\mathbf{b}_i = \lambda_1(\pi_i(L(\mathbf{B})))$.

Recall that the function $\pi_1(\cdot)$ is the identity function, meaning that for $i = 1$ the second condition states that $\mathbf{b}_1 = \lambda_1(L(\mathbf{B}))$. We write $\mathbf{B}_{[i,j]}$ to denote the projected sublattice with basis matrix:

$$[\pi_i(\mathbf{b}_i) \mid \pi_i(\mathbf{b}_{i+1}) \mid \cdots \mid \pi_i(\mathbf{b}_j)] .$$

Definition 2.37 (BKZ-β-reduced Basis) *The lattice basis* $\mathbf{B}$ *is* BKZ-β-reduced*, for some* $\beta \geq 2$*, if the following two conditions are satisfied:*

1. *The lattice basis is size reduced, and*
2. *For* $i \in \{1, \ldots, d\}$ *we have that* $\mathbf{b}_i = \lambda_1(L(\mathbf{B}_{[i,\min(i+\beta-1,d)]}))$.

The BKZ algorithm is complex, and can be difficult to present concisely. We provide an intuition as to how BKZ works, and point to the literature [CN11] for full details. In this thesis use BKZ as a black-box: an algorithm which, given as input a lattice basis, provides us with another lattice basis which satisfies certain properties. These properties can then be used to produce running times for cryptanalytic attacks. Informally, the BKZ algorithm executes the following set of steps:

1. The process begins by considering the first "block" of β vectors $\mathbf{b}_i, \mathbf{b}_{i+1}, \ldots, \mathbf{b}_{i+\beta-1}$ for $i = 1$. We find a new, shorter, vector $\mathbf{b}_\psi$, which is a linear combination of the vectors under consideration, by solving SVP in this "block".

2. We then insert this vector $\mathbf{b}_\psi$ into the lattice basis for $L(\mathbf{B})$ (in position i), and, since we now have a set of $d + 1$ vectors in d-dimensional space, we use LLL to remove the resulting linear dependency, that is, we remove the resulting "zero" vector to retrieve a new basis $\mathbf{B}''$ for the full lattice L.

3. We increase the index i, iterating through the set $i \in \{1, 2, \ldots, d - \beta + 1\}$.

4. We repeat this process c times, halting when either (a) an upper bound on c is achieved, or (b) the output basis satisfies a set of pre-defined conditions (which may include, for example, halting when the quality of the basis is no longer improving at a predetermined rate).

Each cycle through the index set $i \in \{1, 2, \ldots, d - \beta + 1\}$ is referred to as a *tour*. We note that the number of tours considered inside BKZ affects both the running time and the output quality.

2.6.2.1 Performing BKZ in FPYLLL

We can use the FPYLLL library [FPL20] to perform BKZ reduction for a variety of blocksizes β. Using the code outlined in Figure 2.6, we can generate GSO profiles for a random q-ary matrix as in Section 2.6.1.1: i.e. with $d = 220, q = 2^{15}$, and $m = n = 110$. This code generates basis profiles using blocksizes $\beta \in \{10, 30, 50, 70\}$, and the output can be seen in Figure 2.7. In a similar manner to Figure 2.5, we expect the lengths of the GSO vectors to be decreasing, i.e. $\|\mathbf{b}_i^*\| \geq \|\mathbf{b}_{i+1}^*\|$ for $1 \leq i < d$.

```
from fpylll import *
from fpylll.algorithms.bkz2 import BKZReduction as BKZ2

FPLLL.set_precision(128)
A = IntegerMatrix.random(220,"qary", k=110, q=2**15)[::-1]
M = GSO.Mat(A, float_type = "mpfr")
M.update_gso()
params = BKZ.EasyParam(block_size = 50, flags=BKZ.VERBOSE)
bkz = BKZ2(M)
bkz(params)
N = []

for i in range(len(M.r())):
        N.append((i, log(M.r()[i],2**15).n()))

for i in N:
        print(i)
```

Figure 2.6: Code used to generate the BKZ-50 reduced lattice basis considered in Figure 2.7.

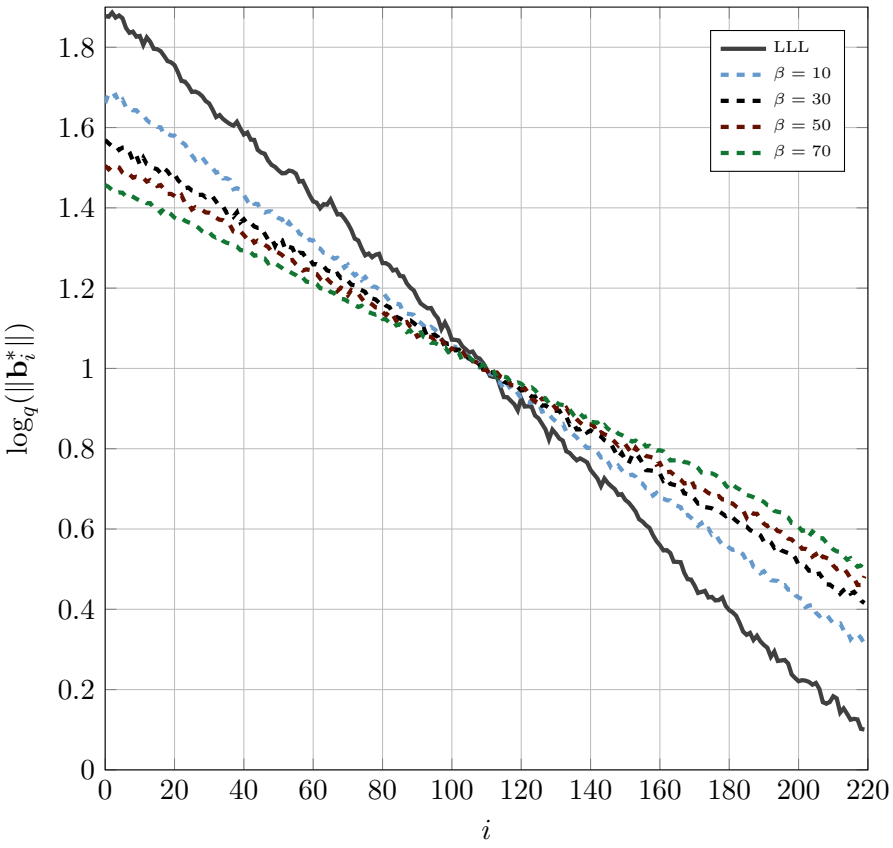


Figure 2.7: Output GSO lengths of a random q-ary lattice of dimension 220 with modulus $q = 2^{15}$ and determinant q^{110} which has been BKZ-β reduced for $\beta \in \{10, 30, 50, 70\}$ using FPYLLL [FPL20]. For comparison, we also include the LLL basis profile used in Figure 2.5.

2.7 Babai's Nearest Plane Algorithm

Babai's Nearest Plane algorithm [Bab86] is a polynomial time algorithm which (approximately) solves the CVP problem. That is, given a lattice $L(\mathbf{B})$ with $\mathbf{B} \in \mathbb{R}^{d \times n}$ and target point $\mathbf{t} \in \mathbb{R}^d$, Babai's Nearest Plane algorithm outputs a vector $\mathbf{v} \in L(\mathbf{B})$ which satisfies:

$$\|\mathbf{v} - \mathbf{t}\| \leq 2^{\frac{n}{2}} \mathsf{dist}(\mathbf{t}, L(\mathbf{B})).$$

We describe Babai's Nearest Plane algorithm in Algorithm 2.

Input: A basis for the lattice $L(\mathbf{B})$ and a target point $\mathbf{t} \in \mathbb{R}^m$
Result: A vector $\mathbf{v}$ which satifies$\|\mathbf{v} - \mathbf{t}\| \leq 2^{\frac{n}{2}} \mathsf{dist}(\mathbf{t}, L(\mathbf{B}))$
Run LLL on $\mathbf{B}$ with $\delta = 3/4$. Set $\mathbf{b} = \mathbf{t}$
for *$i = n$ to 1* **do**
  $\mathbf{b} = \mathbf{b} - \left\lceil \frac{\langle \mathbf{b}, \mathbf{b}_i^* \rangle}{\langle \mathbf{b}_i^*, \mathbf{b}_i^* \rangle} \right\rfloor \mathbf{b}_i$
end
Output $\mathbf{t} - \mathbf{b}$

Algorithm 2: Babai's Nearest Plane algorithm.

If we consider the lattice $L(\mathbf{B})$ tiled by the shifted fundamental parallelepiped defined in Definition 2.16, then, for a given target vector $\mathbf{t}$, Babai's Nearest Plane algorithm returns the lattice point which is contained in the same fundamental region of the target vector $\mathbf{t}$ (here, we assume that the parallelepiped is centred on the vector $\mathbf{t}$). Although Babai's Nearest Plane algorithm is deterministic, in the case of decoding attacks on NTRU and LWE, we can define a success probability with respect to the target point $\mathbf{t}$ [LN13], and we outline approaches for computing this probability in Chapter 3.

2.8 Cryptanalytic Heuristics

In order to provide concrete security estimates for lattice-based cryptosystems, we use the running times of the best attacks. To retrieve these running times, we require usage of many *heuristics*. Throughout this section we outline the cryptanalytic heuristics used in this thesis. We begin with the *root-Hermite factor*, a quantity which captures the *quality* of a lattice basis after lattice reduction has been performed.

Definition 2.38 (Root-Hermite Factor) *For a basis* $\mathbf{B}$ *of a lattice* L *of dimension* d*, the*

root-Hermite factor is defined to be:

$$\delta = \left(\frac{\|\mathbf{b}_1\|}{\mathit{Vol}(L)^{1/d}} \right)^{1/d}.$$

For bases produced by the BKZ algorithm with $\beta \geq 40$, this value is well approximated by $\delta^{2(\beta-1)} = \frac{\beta}{2\pi e}(\beta\pi)^{1/\beta}$ [Che13]. As the value of β increases, the value δ decreases towards one.

Example 2.9 *Consider the BKZ-reduced lattice bases in Figure 2.7. In Table 2.2, for each value of the blocksize β, we compute the root-Hermite Factor δ and also calculate the corresponding approximation $\frac{\beta}{2\pi e}(\beta\pi)^{1/\beta}$.*

β	$\|\mathbf{b}_1\|$	δ	$\frac{\beta}{2\pi e}(\beta\pi)^{1/\beta}$
10	5616.891	1.01574	0.98947
30	3486.508	1.01353	1.01240
50	2501.535	1.01201	1.01206
70	1944.753	1.01085	1.01084

Table 2.2: The root-Hermite factor for a random q-ary lattice of dimension 220 with modulus $q = 2^{15}$ and determinant q^{110} which has been BKZ-β reduced for $\beta \in \{10, 30, 50, 70\}$.

The *Geometric Series Assumption* gives a prediction for the output shape of the Gram-Schmidt basis vectors.

Definition 2.39 (Geometric Series Assumption [Sch03]) *Let $\{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_d\}$ be a basis, of quality δ, of a lattice L, output by some lattice reduction algorithm. The Geometric Series Assumption states that the lengths $\|\mathbf{b}_i^*\|$ for $(1 \leq i \leq d)$ of the Gram-Schmidt vectors of this basis are approximated by:*

$$\|\mathbf{b}_i^*\| \approx \psi^{i-1}\|\mathbf{b}_1\|,$$

for some $0 < \psi < 1$.

We consider the formula for the volume of a lattice, as defined in Definition 2.17, in order to

determine the value of ψ:

$$\begin{aligned}
\mathsf{Vol}(L)^{1/d} &= \prod_{i=1}^{d} \|\mathbf{b}_i^*\| \\
&= \left(\prod_{i=1}^{d} \psi^i \|\mathbf{b}_1\|\right)^{1/d} \\
&= \left(\psi^{\frac{1}{2}d(d-1)} \|\mathbf{b}_1\|^d\right)^{1/d} \\
&= \psi^{\frac{1}{2}(d-1)} \|\mathbf{b}_1\| \\
&= \sqrt{\psi^{d-1}} \|\mathbf{b}_1\|.
\end{aligned}$$

Making use of $\|\mathbf{b}_1\| = \delta^d \mathsf{Vol}(L)^{1/d}$, we can conclude that $\delta^{-d} = \sqrt{\psi^{d-1}}$. This leads to $\psi = \delta^{-2d/(d+1)}$. Typically, this value is approximated as $\psi \approx \delta^{-2}$ and therefore we have:

$$\begin{aligned}
\|\mathbf{b}_i^*\| &= \psi^{i-1} \|\mathbf{b_1}\| \\
&\approx (\delta^{-2})^{i-1} \|\mathbf{b}_1\| \\
&= \delta^{-2i+2} \delta^d \mathsf{Vol}(L)^{1/d} \\
&= \delta^{-2i+2+d} \mathsf{Vol}(L)^{1/d}.
\end{aligned}$$

We note that, for certain blocksizes β (e.g. $\beta \geq 40$ [APS15]), the GSA appears to predict the output shape of lattice reduction well. Wunderer [Wun19] suggests a 'q-ary' GSA, which takes into consideration the case where the length of the first GSO vector is shorter than predicted by the GSA (in this case, since the vector is already shorter than predicted by the GSA, and lattice reduction will not *increase* the length of the input vectors, the GSA does not fit well). There are also a variety of BKZ simulators [CN11, BSW18, FPL20], which attempt to capture the concavity (i.e. non-linearity) of the final block of GSO vectors [YD17]. This non-linearity can be observed in Figure 2.7. We discuss both the 'q-ary' GSA and BKZ simulators in more detail in Chapter 3.

The *Gaussian Heuristic* gives us a simple method of predicting how many lattice points are within a "regular" region of a given volume. We can then use this heuristic to predict the length of the shortest vector within a lattice.

Definition 2.40 (Gaussian Heuristic) *Let L be a d-dimensional lattice. The Gaussian Heuristic states that if K is a measurable set satisfying $K \subseteq \mathsf{Span}(L)$, then the number of lattice points in $K \cap L$ satisfies:*

$$|K \cap L| \approx \frac{\mathsf{Vol}(K)}{\mathsf{Vol}(L)}.$$

We can use the Gaussian Heuristic to predict the length of the shortest vector, $\lambda_1(L)$, in a random lattice L. To do this we suppose that K is an n dimensional ball, centred at the origin, with radius $R = \lambda_1(L)$. Clearly, this means that the set $K \cap L$ contains *only* the shortest vector in the lattice, i.e. we have $|K \cap L| = 1$, yielding $\mathsf{Vol}(K) = \mathsf{Vol}(L)$. Using the well-known formula for the volume of a n-sphere of radius R, this leads us to:

$$\frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}\lambda_1(L)^d \approx \mathsf{Vol}(L).$$

Re-arranging leads to the equation for an approximation of the length of the shortest vector in L:

$$\lambda_1(L) \approx \mathsf{Vol}(L)^{1/d}\frac{\Gamma(\frac{d}{2}+1)^{1/d}}{\sqrt{\pi}}. \tag{2.1}$$

Assuming that Equation 2.1 is in fact an equality, and setting $V_d(1) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}$, we can take logarithms on both sides to produce:

$$\begin{aligned}
\log(\lambda_1(L)) &= \log\left(\frac{\mathsf{Vol}(L)^{1/d}}{V_d(1)}\right) \\
&= \frac{1}{d}\log(\mathsf{Vol}(L)) - \log(V_d(1)) \\
&= \frac{1}{d}\log\left(\prod_{i=1}^{d}\|\mathbf{b}_i^*\|\right) - \log\left(\frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)}\right) \\
&= \frac{1}{d}\log\left(\prod_{i=1}^{d}\|\mathbf{b}_i^*\|\right) - \frac{d}{2}\log(\pi) - \log\left(\Gamma\left(\frac{d}{2}+1\right)\right) \\
&= \frac{1}{d}\sum_{i=1}^{d}\log(\|\mathbf{b}_i^*\|) - \frac{d}{2}\log(\pi) - \log\left(\Gamma\left(\frac{d}{2}+1\right)\right).
\end{aligned} \tag{2.2}$$

In Chapter 3 we will use Equation 2.2 to predict the length of the shortest vector in a variety of different lattices, in order to guarantee success for a specific attack technique.

2.9 Solving the Learning with Errors Problem

Throughout this thesis, we are interested in considering techniques used to solve variants of the Learning with Errors problem. In particular, in this section, we outline the *dual, uSVP,* and *decoding* attacks. All three of these attacks require the usage of a lattice reduction algorithm, typically BKZ, parametrised by a blocksize β. Variants of these attacks specialised to the presence of a small secret are discussed in Chapter 4.

2.9.1 The Dual Attack

The Dual (or Distinguishing) attack [MR09, Alb17] is a method of solving Decision-LWE. Specifically, this problem distinguishes between LWE samples of the form $\mathbf{b} = \mathbf{As} + \mathbf{e}$, as defined in Definition 2.32, and uniformly random input. This method solves Decision-LWE via finding short vectors in the q-ary lattice considered in the SIS problem outlined in Section 2.5:

$$L^* = \{\mathbf{x} \in \mathbb{Z}^m \mid \mathbf{xA} \equiv \mathbf{0} \bmod q\}.$$

Upon retrieving a short vector $\mathbf{v} \in L^*$, we then consider the inner product $\langle \mathbf{v}, \mathbf{b} \rangle$. We have:

$$\langle \mathbf{v}, \mathbf{b} \rangle = \langle \mathbf{v}, \mathbf{As} + \mathbf{e} \rangle = \langle \mathbf{vA}, \mathbf{s} \rangle + \langle \mathbf{v}, \mathbf{e} \rangle = \langle \mathbf{v}, \mathbf{e} \rangle \bmod q,$$

since $\mathbf{vA} \equiv \mathbf{0} \bmod q$. If $\mathbf{b}$ is formed of LWE samples, then the inner product $\langle \mathbf{v}, \mathbf{e} \rangle$ is small [ACF+15]. If $\mathbf{b}$ is uniformly random modulo q, then we would expect the inner product $\langle \mathbf{v}, \mathbf{e} \rangle$ to also be uniformly random. This observation can be used to distinguish LWE from random and thus solve Decision-LWE. We note that the success probability of this attack depends on the length of the vector $\mathbf{v}$, and thus the BKZ blocksize β. This can be seen by considering the output distribution of the inner products $\langle \mathbf{v}, \mathbf{b} \rangle$: as the length of $\mathbf{v}$ grows, this distribution becomes closer to uniformly random. In particular, the result from [LP11] gives the distinguishing advantage as close to:

$$\exp(-\pi(\|\mathbf{v}\|\alpha)^2),$$

where α is the LWE error rate, as defined in Definition 2.30, i.e. $\alpha = \frac{\sigma\sqrt{2\pi}}{q}$.

2.9.2 The uSVP Attack

The uSVP attack [Kan87, ADPS16, AGVW17] is a method of solving Search-LWE via the γ-unique shortest vector problem (uSVP). This approach consists of embedding the LWE error vector $\mathbf{e}$ into a lattice L with a uSVP structure. It is known that the recovery of this vector via lattice reduction can be guaranteed under certain conditions [ADPS16]. In more detail, to solve LWE via uSVP we construct the q-ary lattice:

$$L_{\mathsf{uSVP}} = \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{y} \equiv \mathbf{Ax} \bmod q \text{ for some } \mathbf{x} \in \mathbb{Z}^n\},$$

which has basis matrix:

$$\mathbf{B} = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & t \end{pmatrix}.$$

Here t is the embedding factor which denotes the distance between $\mathbf{b}$ and the lattice $L(\mathbf{A})$, and this value is typically set to be e.g. $t = 1$ or $t = \|\mathbf{e}\|$.[2] We can see that:

$$\begin{pmatrix} \mathbf{A} & \mathbf{b} \\ 0 & t \end{pmatrix} \begin{pmatrix} -\mathbf{s} \\ 1 \end{pmatrix} = \begin{pmatrix} -\mathbf{A}\mathbf{s} + \mathbf{b} \\ t \end{pmatrix} = \begin{pmatrix} \mathbf{e} \\ t \end{pmatrix} \bmod q,$$

and, setting $t = 1$, we see that the lattice $L(\mathbf{B})$ contains the short vector $(\mathbf{e}, 1)$. To recover this short vector, we choose the BKZ blocksize β which satisfies the success condition from [ADPS16] (and this success condition was experimentally verified in [AGVW17]). Specifically, we choose β to satisfy the *[ADPS16] success condition*:

$$\sqrt{\frac{\beta}{d}} \cdot \lambda_1(L) \leq \delta^{2\beta - d} \cdot \mathsf{Vol}(L)^{1/d},$$

where d is the dimension of the lattice $L(\mathbf{B})$. The overall cost of this approach, constrained by the [ADPS16] success condition, is:

$$T_{\mathsf{uSVP}} = \min_{\beta, d} \left\{ T_{\mathsf{BKZ}}(\beta, d) \right\}.$$

2.9.3 The Decoding Attack

The decoding attack [LP11, LN13, BG14] is a method of solving Search-LWE via the bounded distance decoding problem (BDD). This approach consists of constructing a lattice in which the LWE error vector $\mathbf{e}$ is the offset between this lattice and a known target point. Specifically, we construct a lattice with basis matrix:

$$\mathbf{B} = \begin{pmatrix} \mathbf{A} & q\mathbf{I}_m \end{pmatrix},$$

and we note that:

$$\begin{pmatrix} \mathbf{A} & q\mathbf{I}_m \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ * \end{pmatrix} = \mathbf{A}\mathbf{s} + q* = \mathbf{A}\mathbf{s} \bmod q.$$

This tells us that the vector $\mathbf{A}\mathbf{s}$ is contained in the lattice $L(\mathbf{B})$. Moreover, since $\mathbf{A}\mathbf{s} = \mathbf{b} - \mathbf{e}$, we have that the publicly known point $\mathbf{b}$ is of distance $\|\mathbf{e}\|$ from the lattice point $\mathbf{A}\mathbf{s} \bmod q$. Since the vector $\mathbf{e}$ is short, we can use a BDD solver to recover the lattice vector $\mathbf{A}\mathbf{s} \bmod q$ and, therefore, the LWE error vector $\mathbf{e}$ (since $\mathbf{b}$ is publicly known). Overall, the cost of this approach is:

$$T_{\mathsf{Dec}} = \min_{\beta, d} \left\{ \frac{T_{\mathsf{BKZ}}(\beta, d) + T_{\mathsf{BDD}}(\beta, d)}{p_{\mathsf{BDD}}} \right\},$$

[2] Although a choice of $t = \|\mathbf{e}\|$ makes this problem more balanced, in practice, we typically choose $t = 1$ for efficiency, see [AFG13].

where $T_{\mathsf{BDD}}(\beta, d)$ is the cost of running the chosen BDD solver on a BKZ-β reduced lattice of dimension d, and p_{BDD} is the associated success probability of the BDD solver.

2.9.4 Alternative Techniques

There are many other methods to solve LWE, such as Coded-BKW [GJS15], Meet-in-the-middle [APS15, CHHS19], and Arora-Ge [AG11, ACFP14]. A summary of these techniques can be found in [APS15, Pla19]. For the parameter sets we are interested in, these techniques are not competitive with the uSVP, dual, and decoding attacks, as noted in [ACD$^+$18], and we therefore do not consider them in this thesis.

2.10 The Learning with Errors Estimator

The Learning with Errors Estimator (LWE Estimator [APS15]) outputs concrete security estimates for a given set of LWE parameters (n, α, q). By default, the running times of the uSVP, decoding, and dual approaches are given as output. Estimates for several other techniques are also available, although we will not consider them in this thesis. In this section we outline the LWE Estimator.

2.10.1 Input and Output

There are many input and output options in the LWE Estimator. In Table 2.3 we outline the input variables to the top level function `estimate_lwe`, which is used to retrieve security estimates.

Quantity	Meaning
n	The LWE dimension
α	The LWE error rate
q	The LWE modulus
m	The number of LWE samples available
`secret_distribution`	The LWE secret distribution
`reduction_cost_model`	The cost model considered for BKZ
`skip`	List of attacks not considered in the output

Table 2.3: Input parameters to the LWE Estimator, used to retrieve security estimates for LWE parameter sets.

The two most interesting inputs are `secret_distribution` and `reduction_cost_model`. These enable the user to specify a specific LWE secret distribution (such as $\{0,1\}^n$ or D_σ), as well as a custom cost model for lattice reduction.

2.10.1.1 The LWE Secret Distribution: `secret_distribution`

By default, the secret distribution used in the estimator is D_σ, where σ is the width of the error distribution, i.e. the LWE Estimator assumes *normal form LWE* by default. Available inputs are:

- `True` : each component is chosen uniformly at random from D_σ (default)
- `False` : each component is chosen uniformly at random from $\mathbb{Z}_q$
- (a, b) : each component is chosen from the uniform distribution between a and b
- $((a, b), h)$: vectors with a fixed Hamming weight h, where the non-zero components are chosen from the uniform distribution between a and b

It should be noted that, whilst there are small-secret variants of the uSVP, dual, and decoding attacks (outlined in Chapter 4), the LWE estimator does not trigger these variants for *all* small-secret distributions. It is therefore important to consult the source code [Est20] to determine the attack strategies being used.

2.10.1.2 The Lattice Reduction Cost Model: `reduction_cost_model`

There are four in-built cost models for the BKZ algorithm in the LWE Estimator: `sieve`, `qsieve`, `enum`, and `lp`. Each of those cost models corresponds to a result in the literature. The `lp` model corresponds to the runtime estimation of Linder and Peikert [LP11], which is known to be outdated, as discussed in [Alb17]. The `enum` model corresponds to the recent result of [ABF+20], which represents state-of-the-art enumeration-based lattice reduction. The `sieve` model corresponds to results in [BDGL16], which applies improvements in nearest-neighbour algorithms to sieving for short vectors to retrieve a complexity of $2^{0.292n}$ in dimension n, and the model used here assumes $8d$ calls to such an SVP oracle for a dimension d lattice. The `qsieve` result corresponds to the results in [LMv14], which considers the application of quantum search techniques to sieving, resulting in a complexity of $2^{0.265n}$ in dimension n, and the estimator again assumes $8d$ calls for lattices of dimension d. These cost models correspond to the following formulae:

$$\begin{aligned}
\texttt{sieve} &: 8d \cdot 2^{0.292\beta+16.4}, \\
\texttt{qsieve} &: 8d \cdot 2^{0.265\beta+16.4}, \\
\texttt{enum} &: \begin{cases} 2^{0.1839\beta\log(\beta)-0.995\beta+16.25+\log(100,2)} & \text{if } \frac{3}{2}\beta \geq d \text{ or } \beta \leq 92 \\ 2^{0.125\beta\log(\beta)-0.547\beta+10.4+\log(100,2)} & \text{otherwise} \end{cases} \quad {}^{3}, \text{ and} \\
\texttt{lp} &: 2^{\frac{1.8}{\log(\delta_0)}-110+\log(2.3\cdot 10^9)}.
\end{aligned}$$

Note that the model `enum` is considered as the default option in the current variant of the Estimator [Est20]. When retrieving security estimates, it may be desirable to use a custom cost model. We consider how to define the cost models:

$$\begin{aligned}
\texttt{example-model-1} &: T_{\mathsf{BKZ}}(\beta, d) = 2^{0.292\beta}, \text{ and} \\
\texttt{example-model-2} &: T_{\mathsf{BKZ}}(\beta, d) = 4d \cdot 2^{0.265\beta+16.4},
\end{aligned}$$

in Figure 2.8.

```
example_model_1 = lambda beta, d, B: ZZ(2)**RR(0.292*beta)
example_model_2 = lambda beta, d, B: ZZ(2)**RR(0.265*beta + 16.4 + log(4*d, 2))
```

Figure 2.8: An example of two custom cost models to be used in the LWE Estimator.

To retrieve an estimate with the cost model $T_{\mathsf{BKZ}}(\beta, d) = 2^{0.292\beta}$, we call the top-level function `estimate_lwe` with the input `reduction_cost_model = example-model-1`. Such

[3] As of commit `8daf3f7`, this model replaces $2^{0.270\beta\log(\beta)-1.019\beta+16.103+\log(100)}$.

```
load("https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py")
n = 256
alpha = 0.002
q = 17500
_ = estimate_lwe(n, alpha, q)
```

Figure 2.9: Example call to the LWE Estimator with our example parameter set $n = 256, \alpha = 0.002$, and $q = 17500$.

a functionality allows for a custom cost models to be considered when computing security estimates. In Chapter 4, we consider further lattice reduction cost models used in the literature.

2.10.1.3 Outputs

When the LWE Estimator outputs security estimates, it also provides optimal attack parameters. To help illustrate the output of the LWE Estimator, we consider an example.

Example 2.10 *We call the top-level function* `estimate_lwe` *for the parameters* $n = 256, \alpha = 0.002$ *and* $q = 17500$ *in Figure 2.9.*

This code will return complexity estimates and optimal attack parameters for the uSVP, decoding, and dual attacks. For the uSVP attack, the output is:

$$\texttt{rop:} \approx 2^{159.5},\ \texttt{red:} \approx 2^{159.5},\ \delta_0 : 1.005093,\ \beta : 278,\ d : 677,\ m\texttt{:}420,$$

for the decoding attack, the output is:

$$\texttt{rop:} \approx 2^{171.2},\ m\texttt{:}438,\ \texttt{red:} \approx 2^{171.2},\ \delta_0 : 1.005181,\ \beta : 271,\ d : 694,\ \texttt{babai:} \approx 2^{157.0}$$

$$\texttt{babai\_op:}2^{172.1},\ \texttt{repeat:}2^{16.2},\ \epsilon :\approx 2^{-14.0}$$

and for the dual attack, the output is:

$$\texttt{rop:} \approx 2^{180.8},\ m\texttt{:}471,\ \texttt{red:} \approx 2^{180.8},\ \delta_0 : 1.004745,\ \beta : 310,\ \texttt{repeat:}2^{141.0},\ d : 727,\ c : 1$$

In Table 2.4, we explain each of the output values for the usvp, dual, and decoding attacks.

Quantity	Attack/s	Meaning
`rop`	all	The cost of the attack in ring operations (in the ring $\mathbb{Z}_q$)
β	all	The optimal blocksize used inside BKZ
δ_0	all	The root-Hermite factor associated to the choice of β[a]
m	all	The optimal number of LWE samples used in the attack
d	all	The dimension of the lattice reduced via BKZ
`red`	all	The (total) cost of lattice reduction
`repeat`	dual, dec	The number of times the attack needs to be repeated
ϵ	dec	Overall success probability of the algorithm
`babai`	dec	The (total) number of nodes enumerated inside the decoding algorithm
`babai_op`	dec	The (total) cost of the decoding algorithm
k	usvp, dec, dual	The number of guessed components of the LWE secret
`postprocess`	usvp, dec, dual	The maximal hamming weight considered in the search space
c	dual	The scaling factor considered in the dual lattice

Table 2.4: Outputs from the LWE Estimator.

[a] We use the notation δ for this quantity.

The attack parameters k and `postprocess` only appear in the case that certain small-secret distributions are used, and these attack parameters will be discussed in full detail in Chapter 4.

2.10.2 Implemented Features

There are a variety of attack techniques considered in the LWE Estimator and, as new attack techniques surface, these techniques may or may be supported. Moreover, there may be attack techniques which can leverage (for example) small secret distributions, which may be implemented in the case of sparse ternary secrets, but not in the case of uniform ternary secrets. Although the code is open source, it may not be immediately obvious when a new attack technique has been implemented. To help with this issue, we present in Figure 2.10 a diagram of attack techniques considered, and identify which attack techniques are considered for which secret distributions. We comment on each of the attacks listed in Figure 2.10.

- `classic uSVP` refers to the uSVP attack outlined in Section 2.9.2,
- `guess + uSVP` refers to the small-secret variant of the uSVP attack outlined in Chapter 4,
- `LP11 decoding` corresponds to the decoding attack outlined in section 2.9.3, implemented using Linder and Peikert's Nearest Planes algorithm as the BDD solver [LP11],
- `guess + LP11 decoding` corresponds to the small-secret variant of the decoding attack outlined in Chapter 3 (implemented using Linder and Peikert's Nearest Planes algorithm as the BDD solver [LP11]),
- `LN13 decoding` refers to both the decoding attack outlined in section 2.9.3 (implemented using pruned enumeration as the BDD solver [LN13]),
- `guess + LN13 decoding` corresponds to the small-secret variant of this attack as outlined in Chapter 3 (implemented using pruned enumeration as the BDD solver [LN13]),
- `g-v decoding` refers to the guess-and-verify decoding technique outlined in Chapter 3,
- `hybrid-decoding` refers to the hybrid lattice reduction and meet-in-the-middle attack [How07], discussed in Chapters 3 and 4,
- `classic dual` corresponds to the dual attack as outlined in Section 2.9.1,
- `SILKE` refers to the techniques outlined in [Alb17], as discussed in Chapter 4, and

- `hybrid-dual` corresponds to the hybrid dual and meet-in-the-middle attack [CHHS19].

2.11 Public-key Encryption from LWE

Public-key encryption [DH76, RSA78] provides a way for two parties to communicate over an insecure channel, without the need to have agreed on a pre-shared secret. We begin this section by informally defining the notion of a *public-key encryption* scheme.

Definition 2.41 (Public-key Encryption Scheme) *A public-key encryption scheme is made up of three algorithms* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *satisfying:*

1. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^n)$. *The key generation algorithm* KGen *takes as input the security parameter, and outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*, where* pk *is the public key and* sk *is the private key.*
2. $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. *The encryption algorithm* Enc *takes as input the public key* pk *and message* m*, and returns a ciphertext* c*.*
3. $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$ *The decryption algorithm* Dec *takes as input the secret key* sk *a ciphertext* c*, and returns a message* m*.*

We also require correctness, which ensures that a ciphertext decrypts to the correct underlying message.

Definition 2.42 (Correctness of Public-key Encryption) *A public-key encryption scheme is referred to as correct if for any* $m \in \mathcal{M}$*, we have:*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c) = m \mid (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(1^n), c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)\right] = 1.$$

Example 2.11 *We present the Ring-LWE-based public-key encryption scheme outlined by Regev [Reg05], and show how it can be used to encrypt a bit* $b \in \{0, 1\}$*. Note that the cryptosystem parameters need to be correctly tuned in order to guarantee security. Let* n, m, q

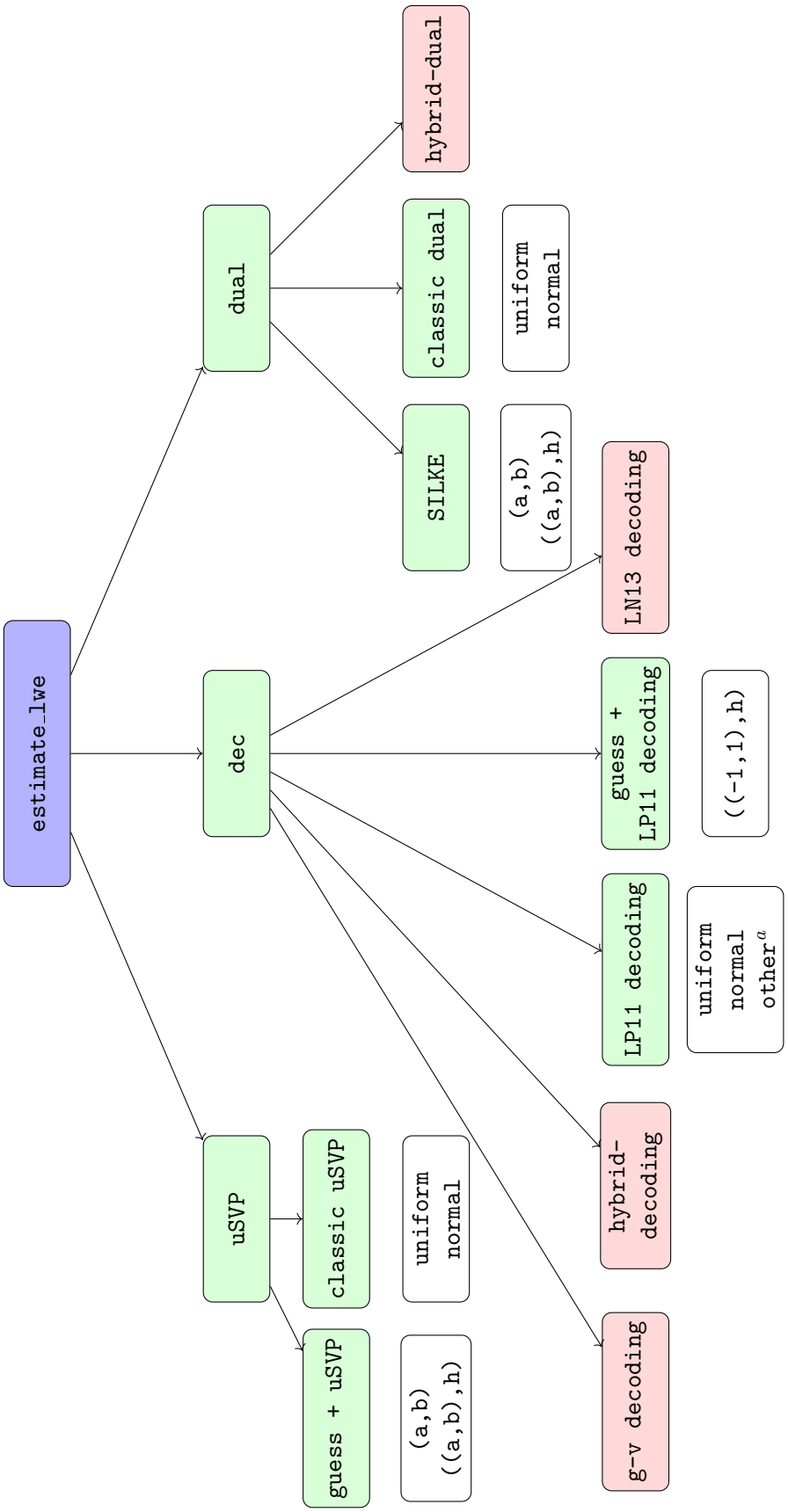


Figure 2.10: Structure of the top level function `estimate_lwe` in the LWE Estimator. Implemented attacks are listed in green, whilst those not implemented are listed in red. Supported secret distributions are listed below implemented attacks. Here we assume that $a \leq 0$ and $b > 0$.

[a] This refers to all secret disributions not contained within the set {`uniform`, `normal`, `((-1,1),h)`}.

be integers and let χ *be a probability distribution*[4] *on* $\mathbb{Z}_q$*. The secret key* sk *is a uniform random vector* $s \in \mathbb{Z}_q^n$*, i.e* $\mathsf{sk} = \mathbf{s}$*. The public key is selected as follows: we draw several samples* $(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)_{i=1}^m$ *from the LWE distribution* $L_{\mathbf{s},\chi,q}$*. To encrypt a bit* $b \in \{0,1\}$*, we choose a random set* $\mathcal{S}$ *from the set of all* 2^m *bit strings of length* m *and the encryption function outputs:*

$$\mathsf{Enc}_{\mathsf{pk}}(b) = \begin{cases} (\sum_{i \in \mathcal{S}} \mathbf{a}_i, \sum_{i \in \mathcal{S}} b) & \text{if } b = 0 \\ (\sum_{i \in \mathcal{S}} \mathbf{a}_i, \lfloor \frac{q}{2} \rfloor + \sum_{i \in \mathcal{S}} b) & \text{if } b = 1 \end{cases}.$$

The decryption algorithm takes as input some vector of the form $(\mathbf{a}, b)$*, and returns* 0 *if* $b - \langle \mathbf{a}, \mathbf{s} \rangle$ *is closer to* 0 *than to* $\lfloor \frac{q}{2} \rfloor$*, and 1 otherwise.*

2.12 Homomorphic Encryption

The search for *Fully Homomorphic Encryption* (FHE) dates back to 1978 [RAD78] and was an open question until 2009, when the problem was solved by Gentry [Gen09]. A variety of intermediary steps were taken [Pai99, BGN05] prior to Gentry's discovery, but none of the schemes prior to Gentry's work were *fully* homomorphic. Since then, there have been a variety of schemes [FV12, BGV14, CGGI16] and several libraries [HEl20, Mic20, Pal20, HEA20, TFH20, Con20], and there are numerous interesting applications [ACC+17], including privacy-preserving machine learning [GBDL+16, BCL+18, JKLS18, CCD+19, CKR+20]. The iDash competition [Ida19] runs each year, challenging participants to produce a privacy-preserving solution using homomorphic encryption. As an example, Track 2 in the 2019 iDash competition was entitled *Secure Genotype Imputation using Homomorphic Encryption.* Currently, a standardisation effort [BDH+17, ACC+17, ACC+18] is taking place, and companies are beginning to commercialise homomorphic encryption techniques.

We begin this section with some definitions, following the notation of [Vai11].

Definition 2.43 (Homomorphic Encryption Scheme) *A homomorphic encryption scheme is made up of four algorithms* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *satisfying:*

1. $(\mathsf{pk}, \mathit{ek}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^n)$*. The key generation algorithm* KGen *takes as input the security parameter, and outputs a tuple of keys* $(\mathsf{pk}, \mathit{ek}, \mathsf{sk})$*, where* pk *is the public encryption key,* ek *is a public evaluation key, and* sk *is the private key.*

[4] We note that Regev considers a Gaussian distribution in order to ensure security.

2. $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. *The encryption algorithm* Enc *takes as input the public key* pk *and message* m*, and returns a ciphertext* c*.*

3. $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$. *The decryption algorithm* Dec *takes as input the secret key* sk *a ciphertext* c*, and returns a message* m*.*

4. $c_f \leftarrow \mathsf{Eval}(\mathit{ek}, f, c_1, c_2, \ldots, c_k)$. *The evaluation algorithm* Eval *takes as input the public evaluation key* ek*, a function* f*, typically represented as an algebraic circuit, and* k *ciphertexts* $c_1, c_2, \ldots, c_k$*, with each* $c_i = \mathsf{Enc}(\mathsf{pk}, m_i)$*, and outputs a ciphertext* c_f*.*

Definition 2.44 ($\mathcal{C}$-homomorphism) *Denote by* $\mathcal{C}$ *a class of functions. A homomorphic encryption scheme* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *is* $\mathcal{C}$*-homomorphic if for any function* $f \in \mathcal{C}$*, we have:*

$$\Pr[\mathsf{Dec}\,(\mathsf{sk}, (\mathsf{Eval}(\mathit{ek}, f, c_1, c_2, \ldots c_k))) \neq f(m_1, m_2 \ldots, m_k)] = \mathit{negl}(\lambda)$$

where $(\mathsf{pk}, \mathit{ek}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$*, and* $c_i \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$*, for* $1 \leq i \leq k$*.*

Definition 2.12 essentially outlines correctness for homomorphic encryption schemes, in terms of a set of functions $\mathcal{C}$. That is, a $\mathcal{C}$-homomorphic encryption scheme is correct for all functions $f \in \mathcal{C}$.

Definition 2.45 (Compactness) *A homomorphic encryption scheme* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *is* compact *if there exists a polynomial* $s = s(\lambda)$ *such that the output length of* $\mathsf{Eval}(\cdot)$ *is at most* s*-bits long.*

The definition of compactness tells us two things. Firstly, that the length of a ciphertext output by the evaluate algorithm is "not too long" (i.e. bounded by some polynomial $s(\lambda)$) and, secondly, that the length of this output is independent of both the input function f and the number of ciphertexts $c_1, c_2, \ldots, c_k$.

Definition 2.46 (Fully Homomorphic Encryption Scheme) *A homomorphic encryption scheme is* fully homomorphic *if it is both compact and homomorphic for the class of all arithmetic circuits over* $\mathit{GF}(q)$*.*

That is, a fully homomorphic encryption scheme can evaluate any arithmetic circuit. As a basic example of a homomorphic encryption scheme, we consider the RSA algorithm.

Example 2.12 *The 'Textbook RSA' algorithm is multiplicatively homomorphic. Let p, q be two k-bit primes, $n = pq$, and choose e such that $\mathsf{gcd}(e, \phi(n)) = 1$ (here ϕ is Euler's Totient function [OLBC10]). We choose d such that $ed \equiv 1 \bmod \phi(n)$, and set $\mathsf{pk} = e, \mathsf{sk} = d$. Encryption is performed on a message m via computing the function:*

$$\mathsf{Enc}(e, m) = m^e \bmod n,$$

and decryption is performed on c via the computing the function:

$$\mathsf{Dec}(d, c) = c^d \bmod n.$$

We now highlight the multiplicatively homomorphic nature of this scheme. Let $c_1 = m_1^e \bmod n$ and $c_2 = m_2^e \bmod n$, we have:

$$\begin{aligned} c_1 c_2 &= (m_1^e \bmod \phi(n))\,(m_2^e \bmod \phi(n)) \\ &= (m_1 m_2)^e \bmod \phi(n) \\ &= \mathsf{Enc}(e, m_1 m_2). \end{aligned}$$

That is, by performing multiplication on two ciphertexts*, we retrieve an encryption of the multiplication of the underlying* plaintexts*, i.e we have multiplicatively homomorphic encryption.*

In this thesis we cryptanalyse LWE-based homomorphic encryption-style parameter sets in Chapters 3 and 5. Also in Chapter 5, we outline in detail three lattice-based homomorphic encryption schemes: BGV [BGV12], BFV [FV12], and CKKS [CKKS17]. In Chapter 6, we make use of the *Paillier* homomorphic encryption scheme [Pai99].

Chapter 3

Batch Bounded Distance Decoding

Contents

This chapter is based on the following publication: Martin R. Albrecht, Benjamin R. Curtis and Thomas Wunderer. Exploring Trade-offs in Batch Bounded Distance Decoding. Selected Areas of Cryptography 2019 (pp. 467-491). Springer, volume 11959 of Lecture Notes in Computer science, 2019. Additional details have been added in this thesis.

In this chapter, we explore trade-offs in Howgrave-Graham's hybrid lattice-reduction and meet-in-the-middle attack (hybrid-decoding attack) on NTRU and LWE. Specifically, we consider

the effect of using a BDD solver with a higher probability of success on the landscape of trade-offs.

The author of this thesis contributed towards (a) the writing of the paper, (b) the design of the g-v decoding algorithm, (c) the implementation of the attack scripts, and (d) the running of all experiments.

3.1 Introduction and Contribution

As defined in Chapter 2, the *Bounded Distance Decoding problem* (BDD) with parameter $\alpha > 0$ asks a challenger to find the closest vector in a lattice $L \subset \mathbb{R}^d$ to some given target vector $\mathbf{t} \in \mathbb{R}^d$, under the *guarantee* that the distance between the target vector $\mathbf{t}$ and the lattice Λ is at most $\alpha\lambda_1(L)$. We can use algorithms which solve BDD to solve variants of the Learning with Errors problem and, therefore, we can estimate the security of lattice-based cryptographic primitives based on the running times of these algorithms. In this chapter, we are particularly interested in solving the small-secret variant of the Learning with Errors problem, as defined in Section 2.5.2, where the components s_i of the secret vector are drawn from some distribution over the set $\{-1, 0, 1\}$.

Due to the shortness and/or sparsity of the secret vector $\mathbf{s}$, *batches* of BDD instances emerge from a combinatorial approach where several components of the target vector are *guessed* before decoding. In this chapter, we explore trade-offs in solving batches of (candidate) BDD instances, which we refer to as "Batch BDD" throughout, and apply our techniques to the NTRU Prime [BCLv19] and Round5 [GZB$^+$19] schemes submitted to the NIST standardisation process [Nat16], as well as a sparse-secret parameter set previously used for the homomorphic encryption library HElib [HEl20].

We compare our techniques to a variant of Howgrave-Graham's hybrid meet-in-the-middle and lattice reduction attack (hybrid-decoding attack) [How07] under a variety of different assumptions. Specifically, we consider Howgrave-Graham's attack when a "classical" guessing approach is used, and also when a meet-in-the-middle guessing approach is used (under conservative assumptions).

3.1.1 An Overview of Decoding Attacks

In this section, we outline the relationships between four techniques to solve LWE via BDD:

1. the decoding attack,
2. the drop-and-solve decoding attack, which combines the decoding attack with zero-forcing techniques,
3. the hybrid-decoding attack, which solves batches of candidate BDD instances using Babai's Nearest Plane algorithm, and
4. the g-v decoding attack, outlined in this chapter, which solves batches of candidate BDD instances using pruned enumeration.

The purpose of this section is to explain the relationship between these four techniques via high level descriptions. The technical details of each of these four techniques can be found later in this chapter.

3.1.1.1 The Decoding Attack

To solve Small-secret LWE via BDD we use a variation of the *decoding attack* outlined in Section 2.9.3. For a collection of LWE samples $\{(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_{i=1}^{m}$ of dimension n, we begin by finding a basis of sufficient quality for the dimension $(n+m)$ lattice:

$$\{(\mathbf{x}, \mathbf{A}\mathbf{x} \bmod q) \mid \mathbf{x} \in \mathbb{Z}^n\},$$

followed by usage of a BDD solver to solve the resulting BDD instance, of which a typical choice would be pruned enumeration [LP11, LN13].

The decoding attack can be summarised in two phases:

1. a *lattice reduction phase* producing a lattice basis of a sufficient quality, and
2. a *decoding phase* where a BDD solver is run against the reduced lattice from step 1, using a known target vector as input.

3.1.1.2 The Drop-and-solve Decoding Attack

We can combine the decoding attack with dimension reduction techniques by guessing (typically as zero, and referred to as "zero-forcing" [MS01]) e.g. the first κ components of the secret vector $\mathbf{s}$. This approach begins by finding a basis of sufficient quality for the dimension $(n + m - \kappa)$ lattice:

$$\{(\mathbf{y}, \mathbf{A}_{(\kappa)}\mathbf{y} \bmod q) \mid \mathbf{y} \in \mathbb{Z}^{n-\kappa}\},$$

followed, again, by usage of a BDD solver to solve the resulting dimension-reduced BDD instance. The LWE secret can be recovered (with some probability, associated to the BDD solver) via solving the BDD instance in the case that the guessed κ components of the secret are all zero. This technique is referred to as the *drop-and-solve decoding attack* in the literature, and follows the mantra of dimension reduction outlined in [MS01, Alb17, ACD$^+$18].

The drop-and-solve decoding attack can be summarised in three phases:

1. a *zero-forcing phase* which selects a zero-forcing dimension $\kappa \leq n$, and generates the resulting $(n + m - \kappa)$ dimension lattice basis to be reduced,
2. a *lattice reduction phase* producing a lattice basis of a sufficient quality, and
3. a *decoding phase* where a BDD solver is run against the reduced lattice from step 1, using a known target vector as input.

3.1.1.3 The Hybrid-decoding Attack

The hybrid-decoding attack [How07, BGPW16, GvW17, Wun19] admits *batches* of BDD instances which emerge from guessing some components of the target vector $(\mathbf{s}, \mathbf{e})$, as above, combined with an algebraic observation which allows *several points* to be decoded against the same lattice. This approach begins by finding a basis of sufficient quality for the dimension $(n + m - \tau)$ lattice:

$$\{(\mathbf{y}, \mathbf{A}_{(\tau)}\mathbf{y} \bmod q) \mid \mathbf{y} \in \mathbb{Z}^{n-\tau}\},$$

followed by multiple calls to a decoding algorithm (Babai's Nearest Plane algorithm), the cost of which is polynomial in the lattice dimension. This makes Babai's algorithm an appropriate choice of BDD solver to be employed in the hybrid-decoding attack, since the adversary has to call the BDD solver many times: once for each guess.

The hybrid-decoding attack can be summarised in three phases:

1. a *dimension reduction phase* which selects a guessing dimension $\tau \leq n$, and generates the resulting $(n + m - \tau)$ dimension lattice basis to be reduced,
2. a *lattice reduction phase* producing a lattice basis of a sufficient quality, and
3. a *guess-and-verify phase* where guesses are verified by running Babai's Nearest Plane against the reduced lattice basis from step 1, and a target vector derived from the particular guess under consideration.

Therefore, unlike the decoding and drop-and-solve decoding techniques, the hybrid-decoding attack permits *multiple decodings per lattice reduction step*. That is, in phase three we are solving batches of (candidate) BDD instances: one for each guess. This guess-and-verify step is usually considered to be realised using a meet-in-the-middle (mitm) approach.

3.1.1.4 The Guess-and-verify Decoding Attack

The uSVP approach outlined in Section 2.9.2 and the hybrid-decoding approach can be considered as the endpoints of a continuum of strategies for solving Batch-BDD. The final enumeration is either (essentially) as expensive as the initial lattice reduction algorithm (as in uSVP) or optimised to be as cheap as possible, in order to decode a large number of points (as in the hybrid-decoding attack). In this chapter, we will explore this continuum of strategies for solving Small-secret LWE instances.

Specifically, we present a *guess-and-verify* decoding approach (g-v decoding) which, like the hybrid-decoding attack, makes use of a guessing approach to reduce the dimension of the associated BDD problem. However, in our g-v decoding attack, we employ a more expensive BDD solver than Babai's Nearest Plane. Here, as in the hybrid-decoding attack, this approach begins by finding a basis of sufficient quality for the dimension $(n + m - \tau)$ lattice:

$$\{(\mathbf{y}, \mathbf{A}_{(\tau)}\mathbf{y} \bmod q) \mid \mathbf{y} \in \mathbb{Z}^{n-\tau}\},$$

followed by multiple calls to a decoding algorithm. However, the decoding algorithm considered here is pruned enumeration. To establish the dimension of the projected sublattice in which we perform enumeration (i.e. the *enumeration dimension*, denoted η), we deploy (a slight

variant of) the success condition for uSVP from [ADPS16], outlined in Section 2.9.2 and, in Section 3.2, we explain and compare these success conditions with a concrete example.

The g-v decoding attack can be summarised in three phases:

1. a *dimension reduction phase* which selects a guessing dimension $\tau \leq n$, and generates the resulting $(n + m - \tau)$ dimension lattice basis to be reduced,
2. a *lattice reduction phase* producing a lattice basis of a sufficient quality, and
3. a *guess-and-verify phase* where guesses are verified by running pruned enumeration in dimension η against the reduced lattice basis from step 1, and a target vector derived from the particular guess under consideration.

Therefore, as opposed to applying a low probability BDD solver on a large number of (candidate) BDD instances (e.g. Babai's Nearest Plane algorithm, as in the hybrid-decoding attack), our g-v decoding technique applies a heavier enumeration process, with a higher probability of success, to a (typically) smaller[1] number of (candidate) BDD instances. In doing so, we hope to achieve a positive trade-off in the overall running time of the attack when considered in comparison to the hybrid-decoding attack.

3.1.1.5 Summary

These four attacks (decoding, drop-and-solve decoding, hybrid-decoding, and guess-and-verify decoding) all share the same genealogy. However, it is not the case that one of these techniques *always* outperforms the others. Indeed, the ordering of these attack techniques depends on the assumptions being used, and the parameter set under consideration.

[1] This is because the cost of performing pruned enumeration in some dimension η is typically more expensive than Babai's Nearest Plane algorithm. Therefore, for the same "enumeration budget", we can perform less BDD calls when using pruned enumeration compared to the Babai's Nearest Plane algorithm.

Attack	Guessing?	Multiple Guesses?	Enumeration algorithm
Decoding	No	No	Pruned enumeration
Drop-and-solve decoding	Yes	No	Pruned enumeration
Hybrid-decoding	Yes	Yes	Babai's Nearest Plane
Guess-and-verify decoding	Yes	Yes	Pruned Enumeration

Table 3.1: A summary of attacks found in the literature: decoding, drop-and-solve-decoding, and hybrid-decoding, as well as our guess-and-verify decoding technique.

3.2 A Comparison of Success Conditions

The success condition from [ADPS16] for uSVP considers the length of the GSO vectors in the context of the uSVP attack. As outlined in Section 2.9.2, for a given uSVP lattice L of dimension d, the unusually short vector in which the LWE secret is embedded can be recovered by BKZ using blocksize β, if the condition:

$$\sqrt{\frac{\beta}{d}} \cdot \lambda_1(\Lambda) \approx \sqrt{\beta} \cdot \sigma \leq \delta^{2\beta-d} \cdot \det(L)^{1/d}, \tag{3.1}$$

is satisfied. The variant of this condition used in this work considers the Gaussian Heuristic directly, and the code used to compute the dimension η, in which we perform enumeration, can be seen in Figure 3.1. This condition is exactly that the length of the projection of the shortest vector (in the lattice $\pi_i(L)$, as predicted by the Gaussian Heuristic) is less than the expected length of the target, offset, vector, i.e. $\sigma\sqrt{d-i}$. Specifically, the condition states:

$$\lambda_1(\pi_i(L)) \approx \mathsf{Vol}(\pi_i(L))^{1/(d-i)} \cdot \frac{\Gamma(\frac{d-i}{2}+1)^{1/(d-i)}}{\sqrt{\pi}} \leq \sigma\sqrt{d-i}. \tag{3.2}$$

For the first index j for which this condition is satisfied, we set the enumeration dimension η as $\eta = d - (j-1)$ and, if this condition is never satisfied, the code reverts to considering an enumeration dimension of $\eta = 2$. The choice of $\eta = 2$ corresponds to performing lattice reduction such that Babai's Nearest Plane algorithm succeeds with probability close to 1 (which is typically not optimal).

Since the ADPS16 success condition is defined such that $\eta = \beta$, i.e. requires the final block of the lattice basis to be HKZ reduced, we consider this case in an example. We consider a q-ary lattice of dimension $d = 1522$, with $q = 4591$ and determinant $q^{d/2}$. For blocksize $\beta = 561$

```
def ball_log_vol(n):
        Return volume of ‘n‘—dimensional unit ball
        :param n: dimension

        return (n/2.) * log(pi) — lgamma(n/2. + 1)


def gaussian_heuristic(r):
        Return squared norm of shortest vector as predicted by the Gaussian heuristic.
        :param r: vector of squared Gram—Schmidt norms

        n = len(list(r))
        log_vol = sum([log(x) for x in r])
        log_gh =  1./n * (log_vol — 2 * ball_log_vol(n))

        return exp(log_gh)


def _enum_dim(r, alpha, q):
        Return eta for a given lattice shape and LWE noise.
        :param r: squared GSO norms
        :param alpha: LWE noise rate
        :param q: LWE modulus

        stddev = est.stddevf(alpha*q)
        d = len(r)
        for i, ri in enumerate(r):
                if gaussian_heuristic(r[i:]) < stddev**2 * (d—i):
                        return ZZ(d—(i—1))

        return ZZ(2)
```

Figure 3.1: Code used to generate the required enumeration dimension which guarantees success in the g-v decoding approach. Note that this code considers the squared norms of the vectors $\|\mathbf{b}_i^*\|$, and thus varies slightly from Equation 2.2. The functions `gaussian_heuristic()` and `ball_log_vol()` are taken from the FPLLL library [FPL20].

satisfying Equation 3.1 we have $\eta_{\mathsf{ADPS}} = \beta = 561$. Using Equation 3.2, we can determine that $\eta_{\mathsf{gh}} = 562$. The success condition considered in our work, i.e. Equation 3.2, explicitly constructs a BDD_α instance with $\alpha = 1$. Note that we are not recovering the short vector directly via lattice reduction, but instead via the usage of an enumeration algorithm with an associated probability of success. The success condition we consider essentially states the following:

> *When lattice reduction is conducted with blocksize β, the short vector can be recovered via solving $BDD_{\alpha=1}$ in dimension η, where η satisfies Equation 3.2.*

whereas the [ADPS16] success condition essentially states:

> *When lattice reduction is conducted with blocksize β satisfying the [ADPS16] success condition, i.e. Equation 3.1, the short vector is recovered directly, and is contained within the output basis provided by BKZ.*

We note that, however, the condition used in our work allows the parameters β and η to be *uncoupled*. That is, we no longer require the condition that $\eta = \beta$ as in Equation 3.1.

3.3 The Hybrid-decoding Attack

The hybrid meet-in-the-middle and lattice reduction (hybrid-decoding) attack was introduced by Howgrave-Graham [How07], and has been considered in the context of LWE with binary error [BGPW16], with a quantum search [GvW17], and is typically used to set parameters for encryption schemes based on the NTRU problem [HPS$^+$15]. This attack combines the ideas of May and Silverman's *dimension reduction techniques* [MS01] against the NTRU cryptosystem, with an algebraic observation which allows multiple (candidate) BDD instances to be solved per lattice reduction step. As opposed to solving BDD in the dimension $d = m + n$ lattice with basis:

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

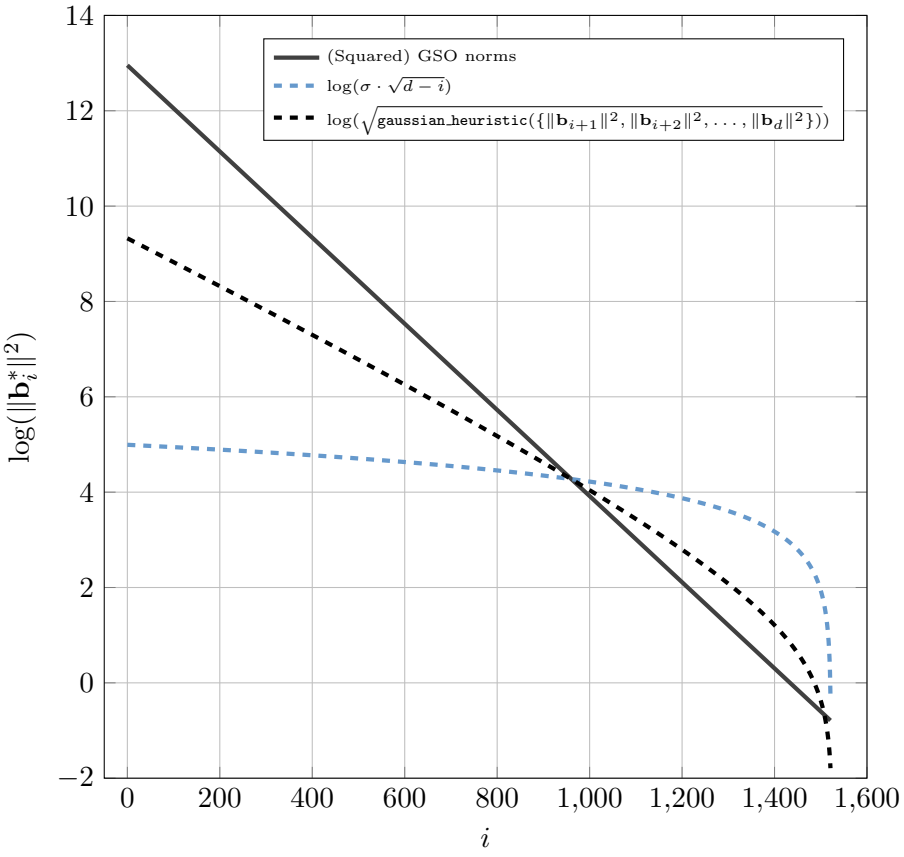


Figure 3.2: A comparison of success conditions. Here, the GSO norms are computed using the Geometric Series Assumption. The dashed blue line represents the expected length of the projected target vector in the projected sublattice $\pi_i^{\perp}(L)$. The dashed black line represents the length of the shortest vector in the same projected sublattice $\pi_i^{\top}(L)$ as predicted by the Gaussian Heuristic. Note that we have used the GSA here to produce the GSO norms. This analysis can also be carried out using a BKZ Simulator to generate the GSO norns, and we consider both of these techniques (GSA and BKZ Simulator) throughout this chapter.

the hybrid-decoding attack sets a guessing dimension τ, carries out lattice reduction in dimension $(d-\tau)$ and solves BDD on a dimension $(d-\tau)$ lattice by decoding on various points associated to guesses in the τ-dimensional guessing space. In more detail: the first phase of the hybrid-decoding attack involves choosing a guessing dimension τ and generating the lattice basis determined by the matrix:

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{A}_{(\tau)} \\ 0 & \mathbf{I}_{n-\tau} \end{pmatrix},$$

recall that $\mathbf{A}_{(\tau)}$ denotes the matrix $\mathbf{A}$ where (without loss of generality) the first τ columns have been dropped. The second phase involves performing lattice reduction on this basis to retrieve $\mathsf{BKZ}_\beta(\mathbf{B})$, and the third phase involves solving a batch of candidate (BDD) instances

using Babai's Nearest Plane algorithm [Bab86] on target points related to vectors in the τ-dimensional guessing space. We have:

$$\begin{pmatrix} q\mathbf{I}_m & \mathbf{A}_{(\tau)} \\ 0 & \mathbf{I}_{n-\tau} \end{pmatrix} \begin{pmatrix} * \\ \mathbf{s}_{(\tau)} \end{pmatrix} = \begin{pmatrix} q* + \mathbf{A}_{(\tau)}\mathbf{s}_{(\tau)} \\ \mathbf{s}_{(\tau)} \end{pmatrix}.$$

If the first τ components of $\mathbf{s}$ are all zero, then it is the case that $\mathbf{A}_{(\tau)}\mathbf{s}_{(\tau)} = \mathbf{A}\mathbf{s}$, which allows recovery of the LWE secret (with some probability, determined by the BDD solver) by decoding on the point $(\mathbf{b}, \mathbf{0})$. If the first τ components of the LWE secret $\mathbf{s}$ are *not* all equal to zero, then we are required to start searching for the first τ components of the secret (e.g over the space $\{-1, 0, 1\}^\tau$). When $s \leftarrow \mathcal{B}^-$, the coefficients s_i of the secret vector are contained within the set $\{-1, 0, 1\}$, and we observe that:

$$\mathbf{A}\mathbf{s} = \sum_{\{i|s_i=1\}} \mathbf{A}_i - \sum_{\{j|s_j=-1\}} \mathbf{A}_j. \tag{3.3}$$

Recalling that the unit vectors are represented by $\mathbf{u}_k$, we can make a guess:

$$\mathbf{v}_g = \sum_{k=1}^{\tau} c_k \mathbf{u}_k,$$

for some values $c_k \in \{-1, 0, 1\}$. For this new guess, we can decode on the point $(\mathbf{b} - \sum_{k=1}^{\tau} c_k \mathbf{A}_k, 0)$. This can be seen to be the correct point to attempt decoding on since, for the correct guess $\mathbf{v} = \sum_{k=1}^{\tau} s_k \mathbf{u}_k$, we have:

$$\begin{aligned} \mathbf{b} - \sum_{k=1}^{\tau} s_k \mathbf{A}_k &= \mathbf{A}\mathbf{s} - \sum_{k=1}^{\tau} s_k \mathbf{A}_k + \mathbf{e} \\ &= \mathbf{A}_{(\tau)}\mathbf{s}_{(\tau)} + \mathbf{e} \bmod q. \end{aligned}$$

Therefore, we have that $(\mathbf{b} - \sum_{i=1}^{\tau} s_k \mathbf{A}_k, 0)$ is separated from the lattice point $(\mathbf{A}_{(\tau)}\mathbf{s}_{(\tau)} \bmod q, \mathbf{s}_{(\tau)})$ by the vector $(-\mathbf{e}, \mathbf{s}_{(\tau)})$, as is required. This means that we can consider a *batch* of BDD instances, by decoding on several target points of the form $(\mathbf{b} - \sum_{k=1}^{\tau} c_k \mathbf{A}_k, 0)$.

Typically, this guessing is realised via the usage of a time-memory trade-off approach outlined in [Wun19]. That is, we can consider Odlyzko's meet-in-the-middle technique described in [HGSW03] for the guessing phase of the attack. Specifically, the guessed part of the secret $\mathbf{v}_g$ is split into two sub-guesses $\mathbf{v}'_g$ and $\mathbf{v}''_g$ satisfying $\mathbf{v}_g = \mathbf{v}'_g + \mathbf{v}''_g$, and we in turn have *two* applications of Babai's Nearest Plane: one for each "half" of the original guess:

$$\mathbf{v}'_l = \mathsf{NP}_{\mathbf{B}}(\mathbf{b} - \sum_{k=1}^{\tau} (\mathbf{v}'_g)_{(k)} \mathbf{A}_k, 0), \text{ and}$$

$$\mathbf{v}_l'' = \mathsf{NP}_{\mathbf{B}}(-\sum_{k=1}^{\tau}(\mathbf{v}_g'')_{(k)}\mathbf{A}_k, 0).$$

We hope that the BDD solver is additively homomorphic, i.e. we hope that if $\mathbf{v}_g = \mathbf{v}_g' + \mathbf{v}_g''$ then we have $\mathbf{v}_l = \mathbf{v}_l' + \mathbf{v}_l''$. We store each decoded vector in a hash table using a locality sensitive hash function [Wun19] which permits collision finding. A collision involves detecting pairs $(\mathbf{v}_g', \mathbf{v}_g'')$ which satisfy $\mathbf{v}_g = \mathbf{v}_g' + \mathbf{v}_g''$ and $\mathbf{v}_l = \mathbf{v}_l' + \mathbf{v}_l''$. Wunderer [Wun19] gives a concrete example of hash tables which guarantee that collisions can be detected. The probability that the BDD solver is additively homomorphic has been analysed in the case that the BDD solver is Babai's Nearest Plane in [Wun19]. However, this refined model is not employed e.g. in submissions to the NIST PQC process [BCLv17, SHRS17, ZCHW17a]. To summarise, there are two probabilities at play when considering a meet-in-the-middle approach:

1. the probability that the BDD solver is homomorphic, and
2. the probability that collisions (as outlined above) are detected; which we can set to 1 by assuming the hash tables of Wunderer [Wun19].

3.3.1 Common Assumptions in Analyses of the Hybrid-decoding Attack

Throughout any analysis of the hybrid-decoding attack, there are several stages in which assumptions are required in order to generate concrete security estimates. In this section, we outline locations within an analysis in which assumptions are required, and we also outline common choices which are made in the literature.

1. The lattice reduction cost model.

The cost of lattice reduction is an important assumption in an analysis of the hybrid-decoding attack. Here we are referring explicitly to the cost of the BKZ algorithm, that is:

$$T_{\mathsf{BKZ}}(\beta, d),$$

where an example cost is $T_{\mathsf{BKZ}}(\beta, d) = 8d \cdot 2^{0.292\beta + 16.4}$.

2. The shape of the basis output by lattice reduction.

As discussed in Section 3.3.2, typical choices are to consider the Geometric Series Assumption [Sch03], the adapted q-ary Geometric Series Assumption [Wun19], or a BKZ simulator e.g. [CN11]. The output shape has an effect on the success probability of Babai's Nearest Plane algorithm (note the inclusion of the GSO lengths in the p_{babai} terms in 3.), and thus the overall running time of the attack.

3. The success probability of Babai's Nearest Plane, p_{babai}.

Typical choices are either to (conservatively) set this probability to be 1, or to use the formula from Wunderer [Wun19] which approximates the probability as:

$$p_{\mathsf{babai}} \approx \prod_{1 \leq i \leq d} \left(1 - \frac{2}{B(\frac{d-1}{2}, \frac{1}{2})} \int_{\min(r_i,1)}^{1} (1-t^2)^{(d-3)/2}\, dt\right).$$

Here $B(\cdot)$ denotes the Beta function [OLBC10], and $r_i = \frac{\|\mathbf{b}_i^*\|}{2\|(\mathbf{s}_{(\tau)}, \mathbf{e})\|}$. Previous works [LP11] have also assumed that Babai's Nearest Plane algorithm succeeds with probability:

$$p_{\mathsf{babai}} \approx \prod_{1 \leq i \leq d} \mathsf{erf}\left(\frac{\|\mathbf{b}_i^*\|\sqrt{\pi}}{2\alpha q}\right).$$

under the assumption of a continuous LWE error distribution. In our work, we use Wunderer's formula, as is standard with state-of-the-art analyses of the hybrid-decoding attack.

4. The running time of Babai's Nearest Plane algorithm, T_{babai}.

Typical choices are either to (conservatively) set this running time to be a single operation, i.e. 1, or to consider a cost which is either linear in the lattice dimension, such as $d/2^{1.06}$, or quadratic in the lattice dimension, such as $d^2/2^{1.06}$. In our work, we consider the quadratic cost, i.e. $d^2/2^{1.06}$.

5. The search strategy under consideration.

Typical choices are to consider a meet-in-the-middle approach, a quantum search, or a classical guessing (exhaustive search) approach. As an example, all three techniques are considered in the NTRU Prime submission to the second round of the NIST standardisation procedure [BCLv19].

6. The meet-in-the-middle probability, p_{mitm}.

Typical choices are to (conservatively) set this probability to be 1, or to consider the formula from [BGPW16, Wun19]:

$$p_{\mathsf{mitm}} = \prod_{i=1}^{d} \left(1 - \frac{1}{r_i B\left(\frac{d-1}{2}, \frac{1}{2}\right)} \int_{-r_i-1}^{-r_i} \int_{\max(-1, z-r_i)}^{z+r_i} \left(1-t^2\right)^{\frac{d-3}{2}} dt dz\right),$$

where the values r_i and function $B(\cdot)$ are as in 3. This probability can be evaluated in Sage [S$^+$20] using Wunderer's scripts [Wun20]. An alternative formulae was recently provided in [SC19]:

$$p_s = \prod_{i=1}^{d} \left(\operatorname{erf}\left(\frac{R_i\sqrt{\pi}}{\alpha q}\right) + \frac{\alpha q}{R_i} \frac{e^{-\left(\frac{R_i\sqrt{\pi}}{\alpha q}\right)^2} - 1}{\pi}\right),$$

under the assumption of a continuous LWE error distribution, as in [LP11].

7. Whether or not to consider the cost of memory.

Most analyses of the hybrid-decoding attack only consider the *time* cost. That is, they do not consider memory costs of (a) lattice reduction (especially in the sieving regime, where the memory cost is exponential in the blocksize β), and (b) the meet-in-the-middle step. The second round NTRU Prime submission to the NIST standardisation procedure [BCLv19], considers different memory scenarios. That is, they consider an analysis which includes memory costs (referred to as "real" memory) and also consider an analysis which ignores memory costs (referred to as "free" memory).

8. The length of the target, offset, vector.

There is also a decision to be made about the length of the target vector. Suppose we set our guessing dimension to τ, then our target vector is:

$$(\mathbf{s}_{(\tau)}, \mathbf{e}) = (s_{\tau+1}, \ldots, s_n, \mathbf{e}).$$

Suppose further that $\mathbf{s}$ is a ternary vector of Hamming weight h, so that $\|\mathbf{s}\| = h$ and $\|\mathbf{e}\| = m\sigma^2$, then we can assume that:

$$\|(s_{\tau+1}, \ldots, s_n, \mathbf{e})\| = \sqrt{h\,\frac{n-\tau}{n} + m\,\sigma^2}.$$

That is, we consider the *expected length* of the target offset. We note that for any given instance of the attack, the target vector may be longer, or shorter, than this assumed length, and this has an impact on the overall running time of the attack.

In Section 3.6 we consider these assumptions for both the hybrid-decoding attack as well as our g-v decoding approach. We begin by making an initial set of assumptions, which match the assumptions made in the NTRU Prime submission to the NIST standardisation procedure, and swap out one assumption at a time until we reach the assumptions considered in our work. At each intermediate stage we consider the complexities of both attacks under the given assumptions, and we present the results in Tables 3.12 and 3.13. This highlights the differences in attack complexity that certain assumptions can make, and shows that it is vital to clearly state assumptions when analysing the hybrid-decoding attack.

3.3.2 Modelling Lattice Reduction for q-ary Bases

For lattice reduction we consider the BKZ algorithm [SE94] parametrised by a block size β which determines the running time (at least exponential in β) and output quality. In this chapter, when BKZ is instantiated with an enumeration algorithm [FP85, Kan83], we consider the cost of lattice reduction using blocksize β on a lattice of dimension d to be:

$$T_{\mathsf{BKZ}}(\beta, d) = 8\,d \cdot 2^{0.18728 \cdot \beta \cdot \log(\beta) - 1.019\beta + 16.1} \text{ enum. nodes},$$

which is taken from [APS15] based on experiments from [CN11]. To translate from the number of nodes visited during enumeration to CPU cycles, the literature typically assumes one node ≈ 100 CPU cycles [FPL20]. This means that in terms of ring-operations the cost of lattice reduction using blocksize β on a lattice of dimension d is:

$$T_{\mathsf{BKZ}}(\beta, d) = 8\,d \cdot 2^{0.18728\beta\log(\beta) - 1.019\beta + 16.1 + \log(100)}.$$

When BKZ is instantiated with a sieving algorithm [AKS01, BDGL16], we consider the cost of lattice reduction using blocksize β on a lattice of dimension d to be:

$$T_{\mathsf{BKZ}}(\beta, d) = 8\,d \cdot 2^{0.292\beta + 16.4},$$

where the constant term is arbitrarily picked as in [APS15].

There are several models in the literature for the behaviour of the BKZ algorithm on q-ary lattices. In most of the literature on solving LWE via BDD, we use the public LWE matrix

$\mathbf{A} \in \mathbb{Z}_q^{(m \times n)}$ to construct a lattice basis of the form:

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{0} \\ \mathbf{A} & q\mathbf{I}_m \end{pmatrix}, \tag{3.4}$$

for which it is commonly assumed that the Geometric Series Assumption, defined in Definition 2.39, is relatively accurate after running BKZ-β with $\beta \ll m + n$. The literature on analysing the hybrid-decoding attack considers lattice bases of the form:

$$\begin{pmatrix} q\mathbf{I}_m & \mathbf{A} \\ 0 & \mathbf{I}_n \end{pmatrix}, \tag{3.5}$$

where the qs are in the top left hand corner of the basis matrix, and the 1s are located in the bottom right of the basis matrix. In Figure 3.3, we plot the initial GSO vector lengths from the basis as in Equation 3.5.

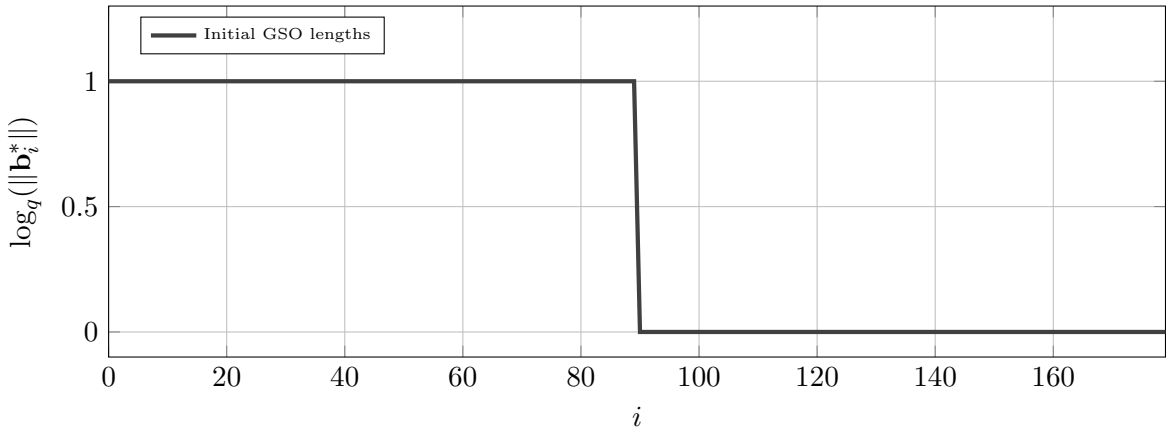


Figure 3.3: Example of the initial GSO lengths for a q-ary lattice of dimension $d = 180$ with $q = 17$ constructed as in Equation 3.5.

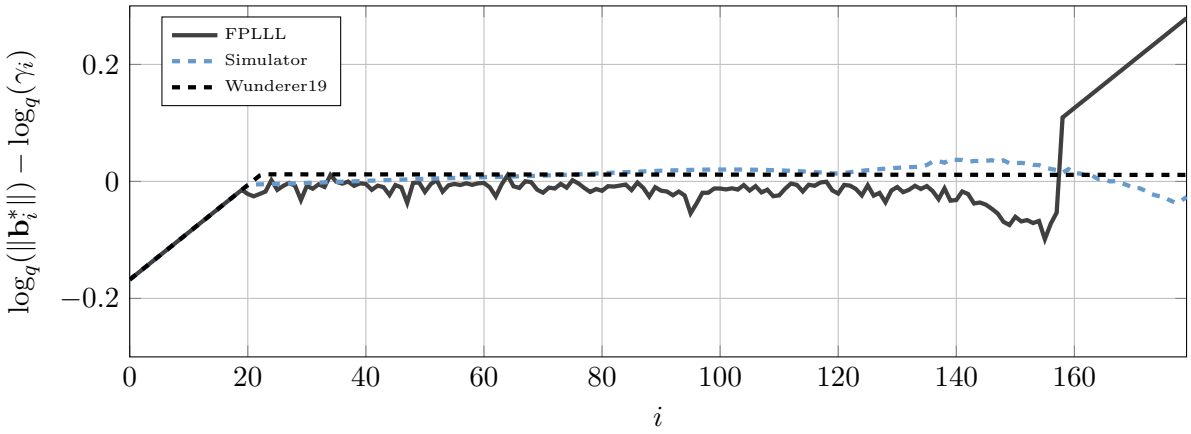
Wunderer [Wun19] notes that writing the basis matrix in the form of Equation 3.5 suggests that the GSA does not hold. Indeed, when the GSA predicts that $\|\mathbf{b}_1^*\| > q$ we notice that this is longer than the first vector *already contained* within the GSO basis. Since lattice reduction does not make the GSO vectors *longer*, we therefore will obtain $\|\mathbf{b}_1^*\| = q$. As a consequence, lattice reduction is expected to produce a "Z-shaped" basis [How07], comprised of leading qs, trailing ones and a middle part approximated by the GSA. In Figure 3.4 we give an illustrative example of this phenomena of the output of lattice reduction as implemented in FPLLL [FPL20] which clearly illustrates the Z-shape.

The structure of this Z-shape has been modelled by Wunderer [Wun19], via insisting on the

condition $\|\mathbf{b}_i^*\| \leq q$ for all $1 \leq i \leq d$. Let k be the number of lattice vectors which follow a GSA-style behaviour, meaning that the remaining $(d-k)$ vectors have length q. Explicitly, we have:

$$\|\mathbf{b}_i^*\| = \begin{cases} q & \text{if } i \leq d-k \\ \delta^{-2(i-(d-k)-1)+k} \cdot q^{\frac{k-n}{k}} & \text{otherwise} \end{cases}.$$

If we make the additional assumption that the first vector in the "GSA-block", i.e. $\mathbf{b}_{(d-k+1)}$, satisfies $\|\mathbf{b}^*_{(d-k+1)}\| \approx q$ [Wun19], we can compute a value for k, i.e. $k = \min\left(\left\lfloor\sqrt{\frac{n}{\log_q(\delta)}}\right\rfloor, d\right)$. Wunderer refers to this shape as the "q-ary GSA", and we follow this notation throughout. In Figure 3.4 we observe that, for this example, the number of leading q's predicted by Wunderer closely resembles that produced by fpylll and the BKZ simulator. However, [Wun19] makes no attempt to model the number of trailing ones. Some works fix the number of trailing ones by choosing a sublattice to reduce [HPS+15], although there is currently no work in the literature that offers a way to predict the number of trailing ones which occur when BKZ is run on the full lattice.



We define $\gamma_i = \alpha^{i-1}\,\delta^d \det(\Lambda)^{1/d}$ and so the GSA corresponds to the line at $y = 0$.

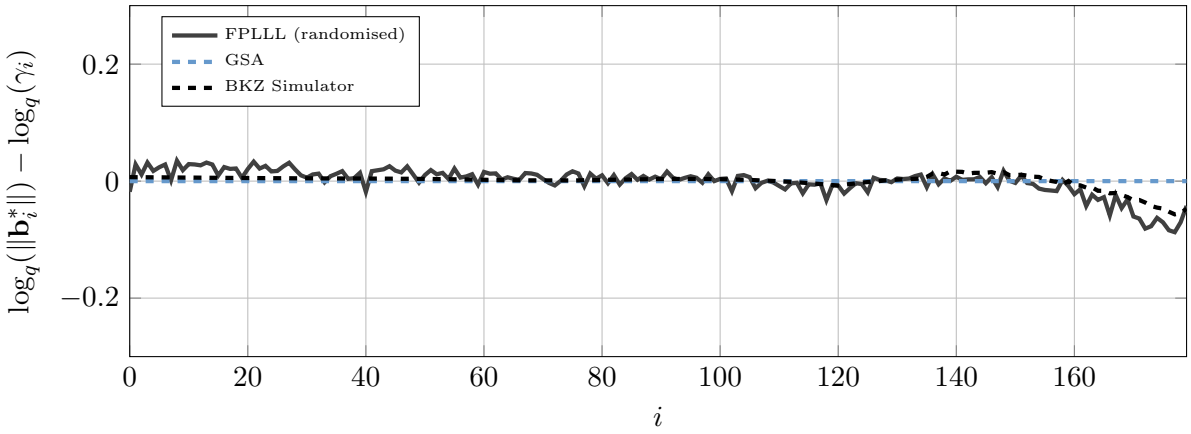
Figure 3.4: Example of BKZ-60 reduction on a q-ary lattice of dimension $d = 180$ with $q = 17$ and volume 17^{80} for bases constructed as in Equation (3.5), along with the output of BKZ simulation and the heuristic from [Wun19].

For the remainder of this chapter, we do not consider the q-ary GSA. That is, we make the assumption that the q-ary structure of the lattice bases on which we perform lattice reduction does not impact the output basis shape. Explicitly, we do not assume that leading qs or training ones occur. This assumption can be made to hold by re-randomising the input basis for lattice reduction. Instead of reducing a lattice basis $\mathbf{B}$ directly, we reduce the lattice basis given by $\mathbf{U}_i\,\mathbf{B}$ for some unimodular matrix $\mathbf{U}_i$. Considering the techniques in this work in a regime exploiting the q-ary structure is an interesting area for future work. In Figure 3.5,

we consider the GSA against such a randomised basis:

$$\mathbf{B}' = \mathbf{U}_i \, \mathbf{B},$$

to determine how closely this example follows the GSA and/or a BKZ simulator. In Figure 3.5, we plot the output GSO lengths against the GSA and the BKZ simulator. Here, we can see that both the GSA and the BKZ simulator provide a good approximation to the GSO lengths, although the BKZ simulator captures the HKZ-reduced behaviour in the last block more accurately.



We define $\gamma_i = \alpha^{i-1} \cdot \delta^d \cdot \det(\Lambda)^{1/d}$ and so the GSA corresponds to the line at $y = 0$.

Figure 3.5: Example of BKZ-60 reduction on a q-ary lattice of dimension $d = 180$ with $q = 17$ and volume 17^{80} for a basis constructed as in Equation 3.5, which has been re-randomised via the application of a unimodular matrix. We also plot the output of the BKZ simulator on a basis for the same lattice.

3.4 A Spectrum of Decoding Approaches for Solving Small-secret LWE

In this section we outline the expected costs of:

1. the classical "decoding" approach in the LWE literature,
2. the "drop-and-solve decoding" approach, which is a combination of zero-guessing and solving a single BDD instance in a reduced dimension, and
3. our *guess-and-verify* decoding approach, where multiple BDD instances are solved per lattice reduction step.

In our guess-and-verify decoding approach, the attack parameters are chosen such that $p_{\mathsf{bdd}} \approx 1$ and $p_{\mathsf{babai}} \approx 1$[2], although for clarity we include these probabilities in the running times presented in this chapter. Note that the probability $p_{\mathsf{bdd}} \approx 1$ ensures that the BDD algorithm succeeds when called on the correct target point. Exploring trade-offs which arise by varying these probabilities is interesting future work.

In order to produce these estimates, we make use of both enumeration-based, and sieving-based, BDD solvers. When using enumeration to solve BDD in dimension η, we assume a cost of:

$$T_{\mathsf{bdd}}(\eta) = 2^{0.18728\eta\log(\eta) - 1.019\eta + 16.1} \text{ enum. nodes},$$

where again we assume that one node $\approx$ 100 CPU cycles [FPL20]. Such an enumeration is assumed to succeed with probability close to one, i.e. $p_{\mathsf{bdd}} \approx 1$. When using sieving to solve BDD in dimension η, we assume a cost of:

$$T_{\mathsf{bdd}}(\eta) = 2^{0.292\eta + 16.4},$$

based on the results of [Laa16], which suggest that sieving for short vectors has the same asymptotic cost as sieving for close vectors. We assume that this sieving process succeeds with probability close to one, i.e. $p_{\mathsf{bdd}} \approx 1$. We note that it is always clear from context whether an enumeration-based, or a sieving-based, BDD solver is being deployed.

To highlight the differences in the three decoding approaches discussed, we make use of a running example to illustrate the behaviour of the approaches under consideration. We consider the small-secret LWE parameter set:

$$n = 653, q = 4621, \sigma \approx \sqrt{2/3}, \chi_s = \mathcal{B}_{100}^{-},$$

and use this parameter set as a reference throughout. Assuming the GSA, a combinatorial dual attack costs $2^{214.4}$ ring-operaions ($\beta = 198$), and a combinatorial uSVP attack costs $2^{209.6}$ ring-operations ($\beta = 223$), according to the LWE Estimator [APS15][3], under the enumeration-based BKZ cost model mentioned in Section 3.3.2.

[2]For g-v decoding, the probability p_{babai} is the probability that the solution found by the BDD algorithm is lifted to the full lattice successfully.

[3]Using the LWE Estimator as of commit `428d6ea`.

3.4.1 Small-secret Decoding

As discussed in Section 3.1, the decoding approach for small-secret LWE [BG14] $\mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e}$ constructs a lattice for which the vector $(\mathbf{b}, 0)$ is separated by the short vector $(-\mathbf{e}, \mathbf{s})$ to the lattice point $(\mathbf{A}\mathbf{s} \bmod q, \mathbf{s})$. The basis of this lattice is given by the columns of the matrix $\mathbf{B}$, where:

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{A} \\ 0 & \mathbf{I}_n \end{pmatrix}.$$

In more detail, we have that:

$$\begin{pmatrix} q\mathbf{I}_m & \mathbf{A} \\ \mathbf{0} & \mathbf{I}_n \end{pmatrix} \begin{pmatrix} * \\ \mathbf{s} \end{pmatrix} = \begin{pmatrix} q* + \mathbf{A}\mathbf{s} \\ \mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} -\mathbf{e} \\ \mathbf{s} \end{pmatrix} \bmod q.$$

After lattice reduction on the lattice $L(\mathbf{B})$, we perform enumeration around the target point $(\mathbf{b}, 0)$ which, with some probability, will return the lattice point separated by $(-\mathbf{e}, \mathbf{s})$, allowing for recovery of the LWE secret.

Typically, the cost of lattice reduction and decoding are balanced, and the output BDD probability determines the number of times the algorithm is repeated. The total expected running time is:

$$T_{\mathsf{Dec}} = \frac{T_{\mathsf{BKZ}}(\beta_{\mathsf{Dec}}, d) + T_{\mathsf{bdd}}(\eta_{\mathsf{Dec}})}{p_{\mathsf{babai}} \cdot p_{\mathsf{bdd}}}.$$

Lattice reduction is carried out on the full lattice with block size β_{Dec}, and a BDD solver is used on a projected sublattice of dimension η_{Dec}, where η_{Dec} can be determined using Equation 3.2. Here p_{babai} is the probability of lifting the candidate solution from $\pi_{d-\eta_{\mathsf{Dec}}+1}(L(\mathbf{B}))$ to the full lattice. Since η_{Dec} is determined using our variant of the condition from [ADPS16] we have $p_{\mathsf{babai}} \approx 1$ [AGVW17]. For our running example parameter set, assuming the GSA, this approach has a cost of $2^{293.0}$ ring-operations with an optimal blocksize $\beta_{\mathsf{Dec}} = 419$, where $\eta_{\mathsf{Dec}} = 429$.

3.4.2 Drop-and-solve Decoding

The drop-and-solve decoding approach for small-secret LWE combines the decoding approach with dimension reduction techniques [MS01]. The attack begins by constructing a lattice for which the vector $(\mathbf{b}, 0)$ is hopefully separated by the short vector $(-\mathbf{e}, \mathbf{s}_{(\tau)})$ to the lattice

point $(\mathbf{A}_{(\tau)}\mathbf{s}_{(\tau)} \bmod q, \mathbf{s}_{(\tau)})$. The basis of this lattice is given by the columns of the matrix $\mathbf{B}$, where:

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{A}_{(\tau)} \\ 0 & \mathbf{I}_{n-\tau} \end{pmatrix}.$$

Assuming that we have correctly guessed τ zero components, we have that:

$$\begin{pmatrix} q\mathbf{I}_m & \mathbf{A}_{(\tau)} \\ \mathbf{0} & \mathbf{I}_{\tau} \end{pmatrix} \begin{pmatrix} * \\ \mathbf{s}_{(\tau)} \end{pmatrix} = \begin{pmatrix} q* + \mathbf{A}_{(\tau)}\mathbf{s}_{(\tau)} \\ \mathbf{s}_{(\tau)} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} + \begin{pmatrix} -\mathbf{e} \\ \mathbf{s}_{(\tau)} \end{pmatrix} \bmod q.$$

In this approach, we guess τ zero components of $\mathbf{s}$ and then run the decoding attack in dimension $(d - \tau)$ [MS01, Alb17, ACD+18]. If we are unsuccessful, we restart with a fresh guess for the positions of zeros.

The core idea is that the lower running time of the dimension-reduced problem will trade-off positively against the probability of guessing zeros. If we correctly guess, for example, that the first τ components of $\mathbf{s}$ are all zeros, then $(s_{\tau+1}, \ldots, s_n, \mathbf{e})$ can be found via solving BDD in the dimension-reduced lattice. The total expected running time of this strategy is:

$$T_{\mathsf{dsDec}} = \frac{T_{\mathsf{BKZ}}(\beta_{\mathsf{dsDec}}, d - \tau) + T_{\mathsf{bdd}}(\eta_{\mathsf{dsDec}})}{p_{\mathsf{babai}} \cdot p_{\mathsf{bdd}} \cdot p_0},$$

where p_0 is the probability of correctly guessing τ zero components in the LWE secret $\mathbf{s}$. Lattice reduction is carried out on a lattice of dimension $(d - \tau)$ with block size β_{dsDec}, and enumeration is carried out in the projected sublattice in dimension η_{dsDec}, as outlined in Section 3.2. For our running example parameter set, assuming the GSA, this attack returns a complexity of $2^{208.2}$ ring-operations with optimal values of $\beta_{\mathsf{dsDec}} = 170$ and $\tau = 315$.

3.4.3 Guess-and-verify Decoding

As in the drop-and-solve decoding approach, our guess-and-verify approach begins by performing lattice reduction on a lattice basis of the form:

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{A}_{(\tau)} \\ 0 & \mathbf{I}_{n-\tau} \end{pmatrix},$$

and combines this technique with the following algebraic observation: if our initial zero guess is incorrect, we can account for this by decoding on multiple points against the same lattice

basis. For example, after decoding on the point $(\mathbf{b}, 0)$ with unsuccessful verification, we can begin checking all Hamming weight one guesses by decoding on the $2 \cdot \binom{\tau}{1}$ points of the form:

$$(\mathbf{b} \pm \mathbf{A}_i, 0) \text{ for all } 1 \leq i \leq \tau,$$

and then all Hamming weight two guesses by decoding on the $2^2 \cdot \binom{\tau}{2}$ points of the form:

$$(\mathbf{b} \pm \mathbf{A}_i \pm \mathbf{A}_j, 0) \text{ for all } 1 \leq i, j \leq \tau, i \neq j,$$

and repeating this process up to Hamming weight ψ for some optimal value $\psi \leq \min(h, \tau)$. This allows us to make multiple guesses per lattice reduction step, exactly as in the hybrid-decoding attack. There is no obvious reason to restrict the BDD solver in any guess-and-verify decoding attack to Babai's Nearest Plane algorithm (as in the hybrid-decoding attack). Instead we may employ a stronger BDD solver, which in turn permits a reduction in the cost of preprocessing or the usage of a lower guessing dimension.

In this *g-v decoding* attack, we consider a BDD dimension as defined by a variant of the success condition from [ADPS16], as outlined in Section 3.2. The overall expected cost of this approach is:

$$T_{\mathsf{gvDec}} = \frac{T_{\mathsf{BKZ}}(\beta_{\mathsf{gvDec}}, d - \tau) + \|\mathcal{S}_{t_{\mathsf{gvDec}}}\| \cdot T_{\mathsf{BDD}}(\eta_{\mathsf{gvDec}})}{p_{\mathsf{babai}} \cdot p_{\mathsf{BDD}} \cdot (\sum_{i=0}^{t_{\mathsf{gvDec}}} p_i)},$$

where p_i is the probability that the guessed sub-vector of the secret $\mathbf{s}$ has Hamming weight i, and $\|\mathcal{S}_{t_{\mathsf{gvDec}}}\|$ is the size of the guessing set, i.e. the number of target points decoded against. For our running example parameter set, assuming the GSA, this attack returns a complexity of $2^{186.1}$ ring-operations with $\beta_{\mathsf{gvDec}} = 225$ and $\tau = 335$, with optimal choices of $\eta_{\mathsf{gvDec}} = 49$ $t_{\mathsf{gvDec}} = 16$ so that $\|\mathcal{S}_{t_{\mathsf{gvDec}}}\| = \sum_{i=0}^{16} \binom{335}{i} \cdot 2^i \approx 2^{105.5}$.

We note that "guess-and-verify" decoding encompasses the usual decoding strategy ($\tau = 0, \|\mathcal{S}_{t_{\mathsf{gvDec}}}\| = 1, \sum p_i = 1$) and the "drop-and-solve" strategy ($\tau > 0, \|\mathcal{S}_{t_{\mathsf{gvDec}}}\| = 1, t_{\mathsf{gvDec}} = 0$). On the other hand, as specified here, it does not encompass the hybrid-decoding attack, (even without time-memory trade-offs) since we insist on picking (β, η) such that $p_{\mathsf{babai}} \approx 1$, which is typically not the case for optimal hybrid-decoding attack parameters.

To summarise, we present a table of results outlining the complexity and optimal parameters for the attack techniques considered in this section.

Technique	β	τ	Ring operations
uSVP	198	297	$2^{214.4}$
dual	223	315	$2^{209.6}$
decoding	419	`n/a`	$2^{293.0}$
drop-and-solve decoding	170	315	$2^{208.2}$
guess-and-verify decoding	225	335	$2^{186.1}$

Table 3.2: Complexity estimates for uSVP, dual, and various decoding techniques on our example parameter set with $n = 653, q = 4621, \sigma \approx \sqrt{\frac{2}{3}}$, and $\chi_{\mathbf{s}} = \mathcal{B}^{-}_{100}$.

3.5 Target Parameter Sets and Estimates

In this section we apply our techniques to parameter sets for the NTRU Prime [BCLv17] and Round5 [BGL+18] submissions to the NIST PQC standardisation process, and, in order to compare with previous works, we also target a sparse-secret parameter set, we consider an old parameter set previously used in the homomorphic encryption library HElib [HEl20], which was studied in [Alb17].[4] We compare our results against the LWE Estimator under the same assumptions, i.e. considering the cost models in Section 3.3.2 and the Geometric Series Assumption. We also present our results considering usage of the BKZ simulator. Full results can be seen in Tables 3.7, 3.8, 3.9 and 3.10. In each section, we present a subset of the results for a classical guessing approach (i.e. exhaustive search).

3.5.1 NTRU Prime

We consider the NTRU LPrime parameter set from [BCLv19]. The construction is based on LWE with a fixed Hamming weight ternary secret and a random ternary error. Specifically, the parameter set considered is:

$$n = 761, q = 4591, \sigma \approx \sqrt{2/3}, \chi_s = \mathcal{B}^{-}_{250}.$$

In the enumeration regime, when the output basis shape is determined by the BKZ simulator, g-v decoding outperforms the non-mitm hybrid-decoding attack by approximately 23-bits, as

[4] For up-to-date information on parameter selection in the homomorphic encryption library HElib, we refer the reader to [HS20].

can be seen in Table 3.3. In the sieving regime, solving Batch-BDD is less efficient than solving BDD (i.e. combinatorics do not improve the running time). Our techniques are thus not interesting in this regime, for this parameter set.

Attack	GSO Profile	Enumeration	Sieving
(non-mitm) hybrid-decoding	GSA	**325.7**	220.8
g-v decoding	GSA	337.6	**181.0**†
(non-mitm) hybrid-decoding	Simulator	385.3	303.3
g-v decoding	Simulator	**362.1**	**185.3**†

Table 3.3: Summary of results for NTRU Prime for a classical guessing (i.e. exhaustive search) approach. Estimates marked with † correspond to standard BDD decoding. Full results can be found in Tables 3.7, 3.8, 3.9, and 3.10.

3.5.2 Round5

For Round5, we consider the NIST level 3 parameter set from [GZB+19]. Round 5 is based on the Learning with Rounding problem (LWR) with a ternary, fixed hamming weight, secret. In the case of LWR, we have another parameter p which is an additional modulus considered in the deterministic rounding process. In this case, we can set $\sigma \approx \sqrt{\frac{(q/p)^2-1}{12}}$ as in [ACD+18]. We therefore model this parameter set as LWE with:

$$n = 756, \sigma \approx 4.61, q = 2^{12}, p = 2^8, \chi_s = \mathcal{B}^-_{242}.$$

We also consider the IoT specific use-case parameter set from [GZB+19]. We can model this parameter set as LWE with:

$$n = 372, \sigma \approx 4.61, q = 2^{11}, p = 2^7, \chi_s = \mathcal{B}^-_{178}.$$

In the enumeration regime, when the output basis shape is determined by the BKZ Simulator, g-v decoding outperforms the non-mitm hybrid-decoding attack for the NIST level 3 parameter set by approximately 16-bits, and the IoT parameter set by approximately 19-bits, as can be seen in Tables 3.4 and 3.5. In the sieving regime, for both parameter sets, solving Batch-BDD is less efficient than solving BDD (i.e. combinatorics do not improve the running time). Our techniques are thus not interesting in this regime, for this parameter set.

Attack	GSO Profile	Enumeration	Sieving
(non-mitm) hybrid-decoding	GSA	**392.5**	275.2
g-v decoding	GSA	416.9	**218.9**†
(non-mitm) hybrid-decoding	Simulator	466.6	394.9
g-v decoding	Simulator	**449.6**	**224.1**†

Table 3.4: Summary of results for Round 5 for a classical guessing (i.e. exhaustive search) approach. Estimates marked with † correspond to standard BDD decoding. Full results can be found in Tables 3.7, 3.8, 3.9, and 3.10.

Attack	GSO Profile	Enumeration	Sieving
(non-mitm) hybrid-decoding	GSA	**205.5**	156.9
g-v decoding	GSA	214.3	**122.4**†
(non-mitm) hybrid-decoding	Simulator	240.6	205.6
g-v decoding	Simulator	**221.4**	**124.3**†

Table 3.5: Summary of results for Round 5 IoT for a classical guessing (i.e. exhaustive search) approach. Estimates marked with † correspond to standard BDD decoding. Full results can be found in Tables 3.7, 3.8, 3.9, and 3.10.

3.5.3 HElib

We also consider our approach in the context of a *sparse* LWE parameter set, in order to gauge the performance of g-v decoding for sparse secrets. To compare with previous works, we consider the sparse-secret parameter set outlined in [Alb17]. Specifically, the parameter set we consider is:

$$n = 1024, q = 2^{47}, \sigma \approx 3.19, \chi_s = \mathcal{B}_{64}^{-}.$$

Since the optimal blocksize β satisfies $\beta \ll m + n$, the output of the BKZ simulator is very close to the GSA and thus the results in each case are similar, as can be seen in Table 3.6. We can see in Tables 3.7 and 3.9 that the results for g-v decoding and non-mitm hybrid-decoding are essentially the same.

Attack	GSO Profile	Enumeration	Sieving
(non-mitm) hybrid-decoding	GSA	69.9	69.8
g-v decoding	GSA	69.1	69.8

Table 3.6: Summary of results for HElib for a classical guessing (i.e. exhaustive search) approach. Full results can be found in Tables 3.7, 3.8, 3.9, and 3.10.

3.5.4 Results and Notation

In the Tables 3.7, 3.8, 3.9, and 3.10, τ is the (fixed) optimal guessing dimension, β is optimal the blocksize used in lattice reduction, η is the enumeration dimension considered, BDD cost is the total cost of solving the candidate BDD instances in the dimension η projected sublattice, $|S|$ is the size of the search space considered, i.e. the number of points on which we decode, d is the dimension of the lattice considered, #pp denotes the maximal Hamming weight considered in the search space, and rop is the cost of running the algorithm in ring-operations. Note that "g-v decoding" is the technique described in this work. Where meaningful, we highlight the "best in class" values in bold. Finally, we note that the "λ" values outline the security claims of each scheme, considering similar (pre-quantum) cost models and (pre-quantum) attacks; we note that such values of λ can be generated using vastly different assumptions.

3.5.5 Results in the Enumeration Regime

Tables 3.7 and 3.8 represent results in the enumeration regime, for a classical guessing strategy and a (square-root) meet-in-the-middle guessing strategy, respectively. Here, if the GSA is assumed, then the hybrid-decoding attack (typically) outperforms g-v decoding. Considering NTRU LPrime in Table 3.7, we see a complexity of 325.7-bits for the non-mitm hybrid attack and a complexity of 337.6-bits for the g-v decoding attack. If the BKZ simulator is assumed, then g-v decoding (typically) outperforms the hybrid-decoding attack in the case of a "classical" guessing approach, but not in the case of a square-root meet-in-the-middle approach. Considering NTRU LPrime in Table 3.7, we see a complexity of 385.3-bits for the non-mitm hybrid attack, and a complexity of 362.1-bits for the g-v decoding attack.

This tells us that, under certain assumptions, g-v decoding outperforms a variant of the

attack	τ	β	η	BDD cost	$\lvert S\rvert$	repeats	d	#pp	$\log_2(\mathsf{rop})$
NTRU LPrime: $n = 761, q = 4591, \sigma = \sqrt{2/3}, h = 250$									$\lambda = 222$
uSVP (GSA)	92	458	458	n/a	1	$2^{57.2}$	1220	n/a	384.6
Dual (GSA)	69	495	n/a	n/a	n/a	$2^{320.9}$	1281	11	374.0
g-v decoding (GSA)	285	430	102	$2^{336.1}$	$2^{252.8}$	$2^{34.1}$	1026	55	337.6
non-mitm hybrid (GSA)	275	400	n/a	$2^{324.1}$	$2^{255.6}$	$2^{49.6}$	1036	57	**325.7**
g-v decoding	225	435	272	$2^{360.9}$	$2^{146.4}$	$2^{53.9}$	1086	28	**362.1**
non-mitm hybrid	305	395	n/a	$2^{384.3}$	$2^{252.3}$	$2^{113.1}$	1006	53	385.3
Round5: $n = 756, q = 2^{12}, \sigma \approx 4.61, h = 242$									$\lambda = 270$
uSVP (GSA)	230	449	449	n/a	1	$2^{160.1}$	936	n/a	478.9
Dual (GSA)	63	626	n/a	n/a	n/a	$2^{413.5}$	1227	19	489.2
g-v decoding (GSA)	365	490	117	$2^{415.2}$	$2^{297.9}$	$2^{60.2}$	814	62	416.9
non-mitm hybrid (GSA)	335	445	n/a	$2^{391.0}$	$2^{295.7}$	$2^{76.8}$	844	64	**392.5**
g-v decoding	290	490	320	$2^{448.2}$	$2^{157.2}$	$2^{92.6}$	889	28	**449.6**
non-mitm hybrid	365	420	n/a	$2^{465.5}$	$2^{274.2}$	$2^{172.9}$	814	55	466.6
Round5 (IoT): $n = 372, q = 2^{11}, \sigma \approx 4.61, h = 178$									$\lambda = 129$
uSVP (GSA)	0	335	335	n/a	n/a	1	682	n/a	220.0
Dual (GSA)	32	334	n/a	n/a	n/a	$2^{174.7}$	661	14	221.7
g-v decoding (GSA)	65	315	224	$2^{213.1}$	$2^{79.2}$	$2^{9.0}$	616	22	214.3
non-mitm hybrid (GSA)	115	270	n/a	$2^{203.6}$	$2^{149.5}$	$2^{36.9}$	566	43	**205.5**
g-v decoding	50	320	266	$2^{220.4}$	$2^{51.6}$	$2^{12.8}$	631	13	**221.4**
non-mitm hybrid	120	270	n/a	$2^{239.6}$	$2^{150.8}$	$2^{71.5}$	561	42	240.6
HElib-1024: $n = 1024, q = 2^{47}, \sigma \approx 3.19, h = 64$									
uSVP (GSA)	140	105	105	n/a	1	$2^{14.0}$	1670	n/a	75.5
Dual (GSA)	189	107	n/a	n/a	n/a	$2^{22.3}$	1680	7	68.4
g-v decoding (GSA)	185	100	48	$2^{66.7}$	$2^{29.5}$	$2^{9.9}$	1624	4	69.1
non-mitm hybrid (GSA)	210	100	n/a	$2^{67.5}$	$2^{36.6}$	$2^{10.7}$	1599	5	69.9

Table 3.7: Estimates in the enumeration regime, where BKZ and the BDD solver are instantiated with enumeration algorithms.

attack	τ	β	η	BDD cost	$\lvert S\rvert$	repeats	d	#pp	$\log_2(\mathsf{rop})$
NTRU LPrime: $n = 761, q = 4591, \sigma = \sqrt{2/3}, h = 250$									$\lambda = 222$
sqrt g-v decoding (GSA)	370	350	43	$2^{243.5}$	$2^{412.2}$	$2^{11.5}$	941	102	245.0
sqrt hybrid (GSA)	360	335	n/a	$2^{240.1}$	$2^{402.8}$	$2^{20.0}$	951	100	**241.3**
sqrt g-v decoding	370	380	119	$2^{273.6}$	$2^{400.0}$	$2^{15.5}$	941	97	**274.7**
sqrt hybrid	395	350	n/a	$2^{274.9}$	$2^{428.3}$	$2^{42.1}$	916	104	275.9
Round5: $n = 756, q = 2^{12}, \sigma \approx 4.61, h = 242$									$\lambda = 270$
sqrt g-v decoding (GSA)	445	395	37	$2^{283.9}$	$2^{490.0}$	$2^{14.1}$	734	120	285.5
sqrt hybrid (GSA)	425	365	n/a	$2^{277.0}$	$2^{453.9}$	$2^{32.0}$	754	109	**278.0**
sqrt g-v decoding	450	430	131	$2^{324.8}$	$2^{474.6}$	$2^{22.7}$	729	113	325.9
sqrt hybrid	460	390	n/a	$2^{320.5}$	$2^{496.6}$	$2^{54.3}$	719	120	**321.6**
Round5 (IoT): $n = 372, q = 2^{11}, \sigma \approx 4.61, h = 178$									$\lambda = 129$
sqrt g-v decoding (GSA)	175	250	48	$2^{156.7}$	$2^{250.8}$	$2^{4.0}$	506	80	157.8
sqrt hybrid (GSA)	165	225	n/a	$2^{151.6}$	$2^{234.5}$	$2^{17.4}$	516	74	**152.9**
g-v decoding	170	270	101	$2^{174.3}$	$2^{237.2}$	$2^{6.5}$	511	73	175.4
sqrt hybrid	180	240	n/a	$2^{172.2}$	$2^{256.4}$	$2^{27.1}$	501	81	**173.4**
HElib-1024: $n = 1024, q = 2^{47}, \sigma \approx 3.19, h = 64$									
sqrt g-v decoding (GSA)	210	95	36	$2^{59.9}$	$2^{59.7}$	$2^{5.2}$	1599	9	62.0
sqrt hybrid (GSA)	270	85	n/a	$2^{60.8}$	$2^{63.1}$	$2^{9.1}$	1539	9	61.8

Table 3.8: Estimates in the enumeration regime considering a "meet-in-the-middle" approach which does not consider probabilities of failure in the meet-in-the-middle phase. Such an approach considers a square-root speed-up in the guessing phase.

attack	τ	β	η	BDD cost	$\lvert S\rvert$	repeats	d	#pp	$\log_2(\mathsf{rop})$
NTRU LPrime: $n = 761, q = 4591, \sigma = \sqrt{2/3}, h = 250$									$\lambda = 155$
uSVP (GSA)	0	532	532	n/a	n/a	1	1352	n/a	185.1
Dual (GSA)	45	586	n/a	n/a	n/a	$2^{148.0}$	1383	14	203.1
g-v decoding (GSA)	0	515	549	$2^{179.7}$	1	1	1351	n/a	**181.0**†
non-mitm hybrid (GSA)	170	580	n/a	$2^{218.9}$	$2^{178.1}$	$2^{21.4}$	1181	43	220.8
g-v decoding	0	530	562	$2^{183.5}$	1	1	1351	n/a	**185.3**†
non-mitm hybrid	230	615	n/a	$2^{302.1}$	$2^{189.6}$	$2^{93.3}$	1121	40	303.3
Round5: $n = 756, q = 2^{12}, \sigma \approx 4.61, h = 242$									$\lambda = 193$
uSVP (GSA)	0	664	664	n/a	n/a	1	1266	n/a	223.6
Dual (GSA)	46	748	n/a	n/a	n/a	$2^{198.6}$	1325	13	251.0
g-v decoding (GSA)	0	645	679	$2^{217.7}$	1	1	1265	n/a	**218.9**†
non-mitm hybrid (GSA)	225	705	n/a	$2^{273.7}$	$2^{215.4}$	$2^{39.3}$	1040	49	275.2
g-v decoding	0	660	699	$2^{223.5}$	1	1	1265	n/a	**224.1**†
non-mitm hybrid	290	700	n/a	$2^{393.9}$	$2^{214.8}$	$2^{160.3}$	975	43	394.9
Round5 (IoT): $n = 372, q = 2^{11}, \sigma \approx 4.61, h = 178$									$\lambda = 96$
uSVP (GSA)	0	335	335	n/a	n/a	1	682	n/a	126.6
Dual (GSA)	22	396	n/a	n/a	n/a	$2^{104.2}$	710	1	145.0
g-v decoding (GSA)	0	315	349	$2^{121.6}$	1	1	681	n/a	**122.4**†
non-mitm hybrid (GSA)	85	375	n/a	$2^{155.2}$	$2^{119.5}$	$2^{18.3}$	596	38	156.9
sqrt g-v decoding	0	320	358	$2^{123.9}$	1	1	681	n/a	**124.3**†
non-mitm hybrid	95	380	n/a	$2^{204.5}$	$2^{122.1}$	$2^{65.1}$	586	35	205.6
HElib-1024: $n = 1024, q = 2^{47}, \sigma \approx 3.19, h = 64$									
uSVP (GSA)	0	137	137	n/a	n/a	1	1939	n/a	70.3
Dual (GSA)	80	115	n/a	n/a	n/a	$2^{19.6}$	1741	7	67.1
g-v decoding (GSA)	85	125	50	$2^{66.6}$	$2^{30.0}$	$2^{2.6}$	1853	5	69.8
non-mitm hybrid (GSA)	155	115	n/a	$2^{66.2}$	$2^{40.1}$	$2^{5.6}$	1783	6	69.8

Table 3.9: Estimates in the sieving regime, where BKZ and the BDD solver are instantiated with sieving algorithms. Estimates marked with † correspond to standard BDD decoding.

attack	τ	β	η	BDD cost	$\lvert S\rvert$	repeats	d	#pp	$\log_2(\mathsf{rop})$
NTRUL Prime: $n = 761, q = 4591, \sigma = \sqrt{2/3}, h = 250$									$\lambda = 155$
sqrt g-v decoding (GSA)	0	515	549	$2^{179.7}$	1	1	1351	0	**181.0**†
sqrt hybrid (GSA)	260	475	n/a	$2^{181.3}$	$2^{296.5}$	$2^{13.9}$	1091	75	182.7
sqrt g-v decoding	0	530	562	$2^{183.5}$	1	1	1351	0	**185.3**†
sqrt hybrid	315	550	n/a	$2^{230.4}$	$2^{341.1}$	$2^{40.9}$	1036	83	231.7
Round5: $n = 756, q = 2^{12}, \sigma \approx 4.61, h = 242$									$\lambda = 193$
sqrt g-v decoding (GSA)	0	645	679	$2^{217.7}$	1	1	1265	0	218.9†
sqrt hybrid (GSA)	320	565	n/a	$2^{216.4}$	$2^{350.7}$	$2^{22.3}$	945	86	**217.5**
sqrt g-v decoding	0	660	699	$2^{223.5}$	1	1	1265	0	**224.1**†
sqrt hybrid	390	660	n/a	$2^{288.4}$	$2^{405.8}$	$2^{67.0}$	875	96	289.6
Round5 (IoT): $n = 372, q = 2^{11}, \sigma \approx 4.61, h = 178$									$\lambda = 96$
sqrt g-v decoding (GSA)	0	315	349	$2^{121.6}$	1	1	681	0	**122.4**†
sqrt hybrid (GSA)	135	300	n/a	$2^{126.9}$	$2^{196.9}$	$2^{11.4}$	546	65	128.2
sqrt g-v decoding	0	320	358	$2^{123.9}$	1	1	681	0	**124.3**†
sqrt hybrid	155	335	n/a	$2^{155.2}$	$2^{218.1}$	$2^{29.2}$	526	68	156.3
HElib-1024: $n = 1024, q = 2^{47}, \sigma \approx 3.19, h = 64$									
sqrt g-v decoding (GSA)	195	100	31	$2^{63.1}$	$2^{53.4}$	$2^{5.3}$	1743	8	65.1
sqrt hybrid (GSA)	235	95	n/a	$2^{61.7}$	$2^{72.1}$	$2^{5.2}$	1703	11	63.6

Table 3.10: Estimates in sieving regime for a "meet-in-the-middle" approach which does not consider probabilities of failure in the meet-in-the-middle phase. Such an approach considers a square-root speed-up in the guessing phase.

hybrid-decoding attack. In order to compare directly to the (full) hybrid-decoding attack, a proper analysis is required of the meet-in-the-middle probability for the g-v decoding approach, and this is left to future work.

3.5.6 Results in the Sieving Regime

Tables 3.9 and 3.10 represent results in the sieving regime, for a classical guessing strategy and a (square-root) meet-in-the-middle guessing strategy, respectively. Here, the results are less interesting since combinatorics do not result in a complexity improvement for g-v decoding and, therefore, the g-v decoding results correspond to the standard decoding attack.

Finally, we note that some of the complexities for the uSVP and decoding attacks considered in these tables outperform the dual attack. Intuitively, since the dual attack solves decision-LWE and the uSVP and decoding attacks solve search-LWE, we might expect that the dual attack should have a lower complexity. This could highlight the fact that there may be potential improvements for the dual attack which have yet-to-be discovered, or, it could outline the fact that some of the assumptions used in our analysis of the hybrid attack, and the variant of the dual attack considered in the LWE Estimator, are mis-aligned. Moreover, we note that this behaviour can also be observed in a variety of other estimates in the literature. For example, there are several examples of this behaviour in the estimates at `https://estimate-all-the-lwe-ntru-schemes.github.io`, which are discussed in detail in Chapter 4.

3.6 Assumptions Case Study: NTRU Prime

As discussed in Section 3.3.1, there are several points during a hybrid-decoding attack-based security analysis where assumptions are required. This can make comparing two analyses of the complexity of the hybrid-decoding attack cumbersome. In order to cross-check our hybrid-decoding attack estimates, we align our code with the assumptions made in the NTRU Prime security script. That is, we consider the set of assumptions $\mathcal{A}_0$ outlined in Table 3.11. Explicitly, assumption set $\mathcal{A}_0$ consists of the following assumptions:

1. core-style BKZ models ("pre-quantum sieving" (i.e. $2^{0.292\beta}$) and "pre-quantum enumeration"

Technique	Assumption	$\mathcal{A}_0$	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	Our work
BKZ SVP calls	1	✓	✓	✓			
	$8d$				✓	✓	✓
p_{babai}	$\prod_{1\leq i\leq d}\left(1-\frac{2}{B(\frac{d-1}{2},\frac{1}{2})}\int_{\min(r_i,1)}^{1}(1-t^2)^{(d-3)/2}\right)$	✓	✓	✓	✓	✓	✓
T_{babai}	1	✓	✓				
	$\frac{d^2}{2^{1.06}}$			✓	✓	✓	✓
BKZ output shape	q-ary GSA	✓					
	BKZ Simulator		✓	✓	✓	✓	✓
Guessing strategy	MiTM (sqrt)	✓	✓	✓	✓		
	Classic					✓	✓
Target norm	$\sqrt{m\,\sigma^2+h\,\frac{n-\tau}{n}}$	✓	✓	✓	✓	✓	✓
Lattice scaling	$\mathbf{s}\mapsto\eta\mathbf{s}:\|\mathbf{s}\|\approx\|\mathbf{e}\|$						✓
Mitm probability	1	✓	✓	✓	✓	✓	✓
Memory considered?	yes						

Table 3.11: Sets of assumptions considered in this case study.

(i.e. $2^{0.18728\beta\log(\beta)-1.019\beta+16.1}$), both with "free memory", in the language of [BCLv19]),

2. the formula for the success probability of Babai's Nearest Plane algorithm from [Wun19], with an associated cost of one operation,
3. the q-ary GSA as the output basis shape of the BKZ algorithm,
4. a meet-in-the-middle guessing phase, with associated collision probability of one,
5. the target norm of the vector recovered via the BDD algorithm is $\sqrt{m\,\sigma^2+h\,\frac{n-\tau}{n}}$,
6. free memory, and
7. no lattice scaling is considered.

We note that [BCLv19] contains estimates which consider the cost of memory. We do not compare against these estimates as we do not consider memory costs in our analysis. After

considering the assumption set $\mathcal{A}_0$, we move through assumptions until we reach those used in our work. In particular, we consider several sets of assumptions: $\mathcal{A}_i$ for $0 \leq i \leq 4$.

1. Assumption set $\mathcal{A}_1$ corresponds to $\mathcal{A}_0$ with the q-ary GSA swapped for the BKZ simulator, since this is a more accurate measure of the GSO basis output shape given by BKZ.

2. Assumption set $\mathcal{A}_2$ corresponds to $\mathcal{A}_1$ with the cost of Babai's Nearest Plane algorithm altered from one operation to be polynomial in the dimension of the lattice, i.e. $\frac{d^2}{2^{1.06}}$ operations.

3. Assumption set $\mathcal{A}_3$ corresponds to $\mathcal{A}_2$ with the core- style cost models changed to cost models which consider eight tours. As an example, this means swapping from e.g. $T_{\mathsf{BKZ}}(\beta, d) = 2^{0.292\beta+16.4}$ to $T_{\mathsf{BKZ}}(\beta, d) = 8d \cdot 2^{0.292\beta+16.4}$.

4. Assumption set $\mathcal{A}_4$ corresponds to $\mathcal{A}_3$ with the guessing strategy changed from a meet-in-the-middle approach to a classical guessing strategy, allowing us to drop the inaccurate assumption that collisions occur with probability one.

5. The only difference between assumption set $\mathcal{A}_4$ and the assumptions considered in our work is that we consider lattice scaling.

These assumption sets are summarised in Table 3.11. We now compare the outputs of the NTRU Prime script[5], our script for the hybrid-decoding attack (labelled *our script (hybrid)*), and our script for the g-v decoding attack (labelled *our script (g-v decoding)*) under these various assumption sets, to see how the complexities of the techniques compare in each case. Table 3.12 outlines the complexities in the enumeration regime, and Table 3.13 outlines the complexities in the sieving regime. Note that we consider the same search spaces as the NTRU LPrime script in each case, i.e. $\beta \in \{40, 80, 120, \dots\}$ and $\tau \in \{0, 40, 80, \dots\}$. We note the closeness of the results under assumptions $\mathcal{A}_0$ between the NTRU Prime script and our script (hybrid) given in Tables 3.12 and 3.13, and comment that this lends confidence to the estimates given by our scripts. Moreover, our code is open source and can be found at the url listed in Section 1.6.

[5]The code used in the NTRU Prime submission can be found here `https://ntruprime.cr.yp.to/estimate-20190329.sage`

ass	alg	τ	β	η	BDD cost	$\lvert S\rvert$	repeats	d	#pp	$\log_2(\mathsf{rop})$
NTRU LPrime: $n = 761, q = 4591, \sigma = \sqrt{2/3}, h = 250$										
$\mathcal{A}_0$	NTRU Prime script	360	320	n/a	$2^{220.9}$	–	$2^{32.5}$	881	–	222.1
	our script (hybrid)	360	320	n/a	$2^{221.3}$	$2^{375.4}$	$2^{33.6}$	951	89	**222.9**
	our script (g-v decoding)	360	360	83	$2^{239.0}$	$2^{380.5}$	$2^{18.1}$	951	91	240.5
$\mathcal{A}_1$	our script (hybrid)	400	360	n/a	$2^{257.6}$	$2^{442.8}$	$2^{36.2}$	911	109	**258.7**
	our script (g-v decoding)	360	400	139	$2^{270.1}$	$2^{390.6}$	$2^{14.4}$	951	95	271.2
$\mathcal{A}_2$	our script (hybrid)	400	360	n/a	$2^{273.0}$	$2^{404.5}$	$2^{52.2}$	911	94	274.5
	our script (g-v decoding)	360	400	139	$2^{270.1}$	$2^{390.6}$	$2^{14.4}$	951	95	**271.2**
$\mathcal{A}_3$	our script (hybrid)	400	360	n/a	$2^{276.2}$	$2^{442.8}$	$2^{36.2}$	911	109	**277.9**
	our script (g-v decoding)	360	400	139	$2^{283.6}$	$2^{409.8}$	$2^{8.7}$	951	103	284.9
$\mathcal{A}_4$	our script (hybrid)	320	400	n/a	$2^{386.9}$	$2^{256.3}$	$2^{111.8}$	991	53	388.1
	our script (g-v decoding)	240	440	280	$2^{376.0}$	$2^{141.3}$	$2^{68.1}$	1071	26	**380.0**

Table 3.12: Enumeration-based estimates, where each section corresponds to a set of assumptions outlined in Table 3.11. "– " denotes a value which is not compatible with our notation (for example, our script considers a simple sqrt speed-up in the search space, the NTRU Prime script considers splitting the search space as in a meet-in-the-middle approach).

ass	alg	τ	β	η	BDD cost	$\lvert S\rvert$	repeats	d	#pp	$\log_2(\mathsf{rop})$
NTRU LPrime: $n = 761, q = 4591, \sigma = \sqrt{2/3}, h = 250$										
$\mathcal{A}_0$	NTRU Prime script	240	480	n/a	$2^{156.0}$	–	$2^{18.8}$	1081	–	159.4
	our script (hybrid)	240	480	n/a	$2^{158.1}$	$2^{279.7}$	$2^{18.3}$	1111	72	**159.4**
	our script (g-v decoding)	0	560	562	$2^{164.1}$	1	1	1351	0	165.0†
$\mathcal{A}_1$	our script (hybrid)	320	600	n/a	$2^{209.9}$	$2^{348.2}$	$2^{35.8}$	1031	85	211.5
	our script (g-v decoding)	0	560	593	$2^{173.2}$	1	1	1351	0	**173.2**†
$\mathcal{A}_2$	our script (hybrid)	360	320	n/a	$2^{221.3}$	$2^{375.4}$	$2^{33.6}$	951	89	222.9
	our script (g-v decoding)	0	560	593	$2^{173.2}$	1	1	1351	0	**173.2**†
$\mathcal{A}_3$	our script (hybrid)	320	600	0	$2^{227.2}$	$2^{338.2}$	$2^{39.1}$	1031	81	228.3
	our script (g-v decoding)	0	560	593	$2^{176.2}$	1	1	1351	0	**177.8**†
$\mathcal{A}_4$	our script (hybrid)	200	640	0	$2^{297.6}$	$2^{180.7}$	$2^{97.6}$	1151	40	298.6
	our script (g-v decoding)	0	560	593	$2^{176.2}$	1	1	1351	0	**177.8**†

Table 3.13: Sieving-based estimates, where each section corresponds to a set of assumptions outlined in Table 3.11. "– " denotes a value which is not compatible with our notation (for example, our script considers a simple sqrt speed-up in the search space, the NTRU Prime script considers splitting the search space as in a meet-in-the-middle approach). Estimates marked with a † correspond to a uSVP estimate with η, β uncoupled.

There are two main observations from Tables 3.12 and 3.13. The first observation is that we believe assumption set $\mathcal{A}_4$ represents assumptions which are more realistic than those used in the preceding assumption sets and, as we can see in the enumeration regime (Table 3.12), the g-v decoding attack outperforms (a variant of) the hybrid attack, under these assumptions, for this parameter set. In the sieving regime, this is not the case, and here g-v decoding corresponds to the standard BDD decoding approach.

The second observation is to highlight the differences that varying assumptions can make on attacks of this type. If we consider assumption sets $\mathcal{A}_0$ and $\mathcal{A}_3$ in the enumeration regime (Table 3.12), we note that the difference in complexity of the g-v decoding estimates is $\approx 285 - 241 = 44$-bits, and the difference in complexity of the hybrid attack estimates is $\approx 278 - 223 = 55$-bits. This shows the importance of outlining the assumptions used in any security analysis which considers decoding attacks.

3.7 Conclusion

In this chapter, we introduced a *guess-and-verify decoding* technique to solve the Small-secret Learning with Errors problem. Throughout, we have compared this technique with the hybrid-decoding attack. These two attack techniques consider a myriad of trade-offs and are trick to optimise, due to the large number of degrees of freedom in the attack parameters (e.g. $(\tau, \beta, m, |S|)$). Conservative assumptions can be made (such as those discussed throughout this chapter) in order to generate an *underestimate* of security fairly efficiently. This is a reasonable approach, provided that the process is transparent. However, it is also important to estimate the complexity of these attacks as accurately as possible. Throughout this chapter, we have considered a variety of sets of assumptions and have shown that, under certain assumptions our g-v decoding technique outperforms a (non-mitm) variant of the hybrid-decoding attack and, under other sets of assumptions, the converse is true.

Chapter 4

Security Estimates for the NIST Standardisatio Process for Post-quantum Cryptographic Algorithms

Contents

This chapter is based on the following publication: Martin R. Albrecht, Benjamin R. Curtis, Alex Davidson, Amit Deo, Rachel Player, Eamonn Postlethwaite, Fernando Virdia and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! 11th International Conference on

Security in Communications Networks (SCN) (pp. 351-367). Springer, volume 11035 of Lecture Notes in Computer Science, 2018. Additional details have been added in this thesis.

In this chapter, we survey the submissions to the first round of the NIST standardisation process for post-quantum cryptographic algorithms. In particular, we consider every cost model for lattice reduction used as part of a submission and estimate the complexity of all parameter sets for each submission under every cost model, which allows the security claims of two schemes to be compared more easily. Finally, we provide a status update regarding the third round of the NIST standardisation process.

The author of this thesis contributed towards (a) the collection of data used in the tables and (b) the writing of the paper. Furthermore, all updates for the third round are novel.

4.1 Introduction and Contribution

In 2015, the United States' National Institute of Standards and Technology (NIST) began a process aimed at the standardisation of post-quantum public-key encryption schemes (PKE), key encapsulation mechanisms (KEM), and digital signature algorithms (SIG). The initial call for proposals was in 2016 [Nat16]. The aim of the process is to ensure that cryptographic requirements can be met in an era where quantum computers exist. Participants were invited to submit their designs along with an associated cryptanalysis, and parameter sets aimed at meeting one or more target security categories out of the five defined by NIST. The five security categories are listed as follows:

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).

One way to interpret these security levels is the following: each AES-n security level corresponds to n-bits of classical security and $n/2$-bits of quantum security (based on the complexity of key search, i.e. $\mathcal{O}(2^n)$ classically and $\mathcal{O}(2^{n/2})$ using Grover's algorithm), and each SHA-n security level corresponds to $n/2$-bits of classical security and $n/3$-bits of quantum security (based on the complexity of collision search, i.e. $\mathcal{O}(2^{n/2})$ classically and $\mathcal{O}(2^{n/3})$ using Grover's algorithm), leading to the security levels in Table 4.1.

Security Level	**Classical Security (bits)**	**Quantum Security (bits)**
1 (AES-128)	128	64
2 (SHA-256)	128	≈ 85
3 (AES-192)	192	96
4 (SHA-384)	192	128
5 (AES-256)	256	128

Table 4.1: An example interpretation of the five NIST security levels.

Different interpretations of these five different security levels, alongside the use of vastly different assumptions considered by each submission, means that the security (and efficiency) of any two schemes can be difficult to compare fairly. In particular, the *cost model* for the BKZ algorithm, as defined in Section 2.10.1.2, varies across submissions. This means that it is possible for two *identical* parameter sets to be associated to two different security levels λ_1 and λ_2, of which the difference can be significant. In this chapter, we are interested in the *lattice-based* submissions to the NIST standardisation process. We extract the proposed parameter sets, and lattice reduction cost models, used in each LWE-based and NTRU-based submission. To provide a clearer view on the effect of the chosen cost model, we cross-estimate the security of each parameter set under every cost model for all first round submissions.

The five security categories are defined as target security levels when considering adversaries who have access to a quantum computer. Furthermore, NIST propose the assumption that such a quantum computing device would support a maximum circuit depth $\texttt{MAXDEPTH} \leq 2^{96}$ [Nat16]. We note that not all schemes take this limitation into account, and instead opt for using an asymptotic cost model that considers the best known theoretical Grover speed-up, resulting in overestimates of the adversary's power.

This has caused confusion within the submissions and this confusion introduces further difficulties when making a comparison of two schemes under a (quantum) cost model. Consider category 1, which states that attacks on schemes should be at least as hard as AES-128 key recovery. Some schemes interpret this by generating parameter sets with $\lambda \geq 128$ under a quantum cost model, whereas other schemes claiming the same category of hardness interpret this to mean $\lambda \geq 64$, since key recovery for AES-128 can be completed in time $\mathcal{O}(2^{n/2})$ using Grover's algorithm. This results in schemes listing parameter sets with vastly different claimed security being in the same category. To make this clear we list the "claimed security" levels of all schemes in our tables of estimates.

We restrict our attention to the uSVP variant of the primal lattice attack [ADPS16, AGVW17] and the dual lattice attack [MR09], and we recall that both of these techniques were introduced in Chapter 2. We note that, for certain schemes where the LWE secret is small and/or sparse, we consider the small-secret variants of the uSVP attack [BG14] and dual attack [Alb17] which we outline in Section 4.5. We do not consider algebraic [AG11, ACFP14] or combinatorial [AFFP14, GJS15, KF15, GJMS17] attacks, since those algorithms are not competitive for the parameter sets considered in this work.

We do not consider the primal attack via a combination of lattice reduction and BDD enumeration often referred to as a "lattice decoding" attack [Sch03, LP11] as considered in Chapter 3. However, note that the primal uSVP attack can be considered as a simplified variant of the decoding attack in the enumeration regime. For NTRU, we restrict our attention to the primal uSVP attack (possibly combined with guessing zero-entries of the short vector). We do not consider the hybrid lattice reduction and meet-in-the-middle attack [How07, Wun19] or "guessing + nearest plane" after lattice reduction, as considered in Chapter 3.

4.1.1 Related Work

NIST categorised each scheme according to the family of underlying problem (lattice-based, code-based, SIDH-based, MQ-based, hash-based, other) in [Moo17]. This analysis was refined in [Fuj17]. NIST then provided a first performance comparison of all complete and proper schemes in [Nat17]. Bernstein provided a comparison of all schemes based on the sizes of their ciphertexts and keys in [Ber17].

4.2 First Round Submissions

In total, 82 submissions were made to the standardisation process and, of these submissions, 69 were deemed "complete and proper" by NIST. These 69 submissions, of which 23 were based on either the LWE or NTRU family of problems, formed the *first round submissions*. The 18 LWE-based submissions and five NTRU-based submissions are outlined in Table 4.2.

Assumption	**Schemes**	
LWE Variants	Crystals-Dilithium	Crystals-Kyber
	Ding Key Exchange	Emblem
	Frodo	HILA5
	KCL	KINDI
	LAC	Lima
	LOTUS	Lizard
	NewHope	Saber
	ThreeBears	Titanium
	uRound2	qTesla
NTRU Variants	Falcon	NTRU HRSS
	NTRUEncrypt	NTRU Prime
	pqNTRUSign	

Table 4.2: Complete and proper lattice-based submissions to the NIST standardisation process.

4.3 Costing Lattice Reduction

There are a variety of approaches within the literature to cost the running time of the BKZ algorithm [CN11, APS15, ADPS16]. The main divergence stems from whether cost models are based on using enumeration-based algorithms as the SVP solver (the enumeration regime), or sieving-based algorithms as the SVP solver (the sieving regime). A second divergence stems from how many calls to the SVP oracle are expected to be required to recover a vector of length $\approx \delta^d \mathsf{Vol}(\Lambda)^{1/d}$.

The cost of BKZ with blocksize β on a lattice of dimension d can be written in the form:

$$\begin{aligned} T_{BKZ}(\beta, d) &= a \cdot 2^{f(\beta,d)} \\ &= 2^{f(\beta,d)+\log(a)}, \end{aligned}$$

where a denotes the number of calls to an SVP oracle of cost $2^{f(\beta,d)}$. We refer to the exponent, i.e. $f(\beta, d) + \log(a)$, as a *BKZ cost model*. In this section, we review the cost models used as part of all first round submissions. We note that the cost models considered in this chapter do not consider the *coefficient size*, that is the bit-length of the modulus q. We assume that the output costs are in ring-operations, i.e. a number of operations in $\mathbb{Z}_q$, but we do not consider the additional costs (e.g. in *binary operations*) brought by varying the value of q. For the schemes considered in this work, where the moduli are reasonably close in size, this is not particularly an issue. However, for homomorphic encryption schemes, where modulus sizes larger than 2^{1000} are considered, it may be important to also consider the coefficient size in the cost of lattice reduction.

4.3.1 Enumeration-based Cost Models

Let d be the dimension of the lattice used inside BKZ with blocksize β. In the literature, the cost of enumeration can be found to be estimated as $2^{c_1\beta\log\beta+c_2\beta+c_3}$ [Kan83, MW15] or as $2^{c_1\beta^2+c_2\beta+c_3}$ [FP85, CN11], with Grover speedups considered to half the exponent. There are four enumeration-based cost models used as part of the first round submissions to the NIST standardisation process. The estimates $0.187\beta\log\beta - 1.019\beta + 16.1$ [APS15] and $0.000784\beta^2 + 0.366\beta - 0.9$ [HPS$^+$15] are both based on fitting the same data from [Che13]. 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

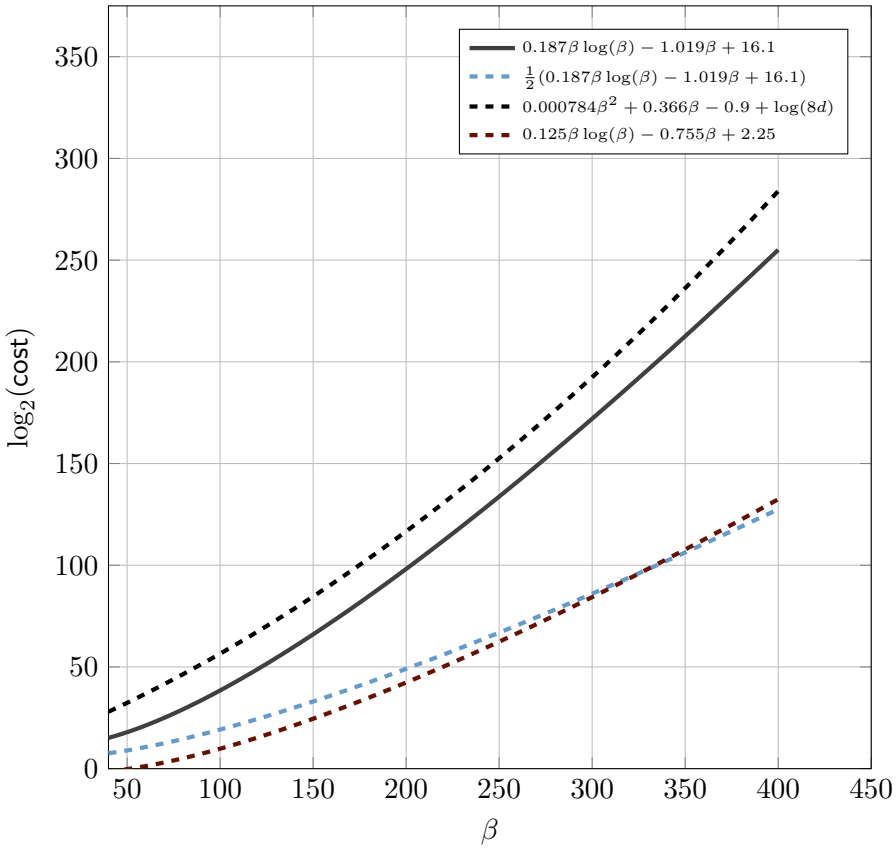


Figure 4.1: Enumeration-based cost models used as part of a first round submission to the NIST standardisation process for a lattice of dimension $d = 1024$ with $40 \leq \beta \leq 400$.

to a curve of shape $2^{c_1\beta\log\beta + c_2\beta + c_3}$ following [MW15]. We fit a curve to the values given by [PHAM17, (39)], the script used is available in the public repository [Lot18]. To summarise, the four enumeration-based cost models considered in the first round submissions are:

1. $0.187\beta\log(\beta) - 1.019\beta + 16.1$,
2. $\frac{1}{2}(0.187\beta\log(\beta) - 1.019\beta + 16.1)$,
3. $0.000784\beta^2 + 0.366\beta - 0.9 + \log(8d)$, and
4. $0.125\beta\log(\beta) - 0.755\beta + 2.25$.

To illustrate the differences in these models, we plot the costs for a lattice of dimension $d = 1024$ for $40 \leq \beta \leq 400$ in Figure 4.1.

4.3.2 Sieving-based Cost Models

There are ten sieving-based cost models used as part of first round submissions to the NIST standardisation process. Recall that sieving algorithms require the usage of exponential memory. However, in our work, we do not consider the memory costs associated to sieving algorithms, and simply consider the time cost. Let d be the dimension of the lattice used inside BKZ with blocksize β. The 0.292β model comes from [BDGL16], and the 0.265β model accounts for speed-ups associated Grover's algorithm, and appears in [Laa15a]. There is a min-space variant 0.368β which also appears in [BDGL16], where Grover speed-ups account for the 0.2975β model, occurring in [Laa15a]. The constant 16.4 comes from experiments in [Laa15b], interpolated by [APS15]. The $\log(\beta)$ term appears in submissions which assume that a sieving cost model of the form $2^{c\beta}$ requires $\beta 2^{c\beta}$ CPU cycles.

With respect to the number of SVP oracle calls required by BKZ, a popular choice was to follow the "Core-SVP" model introduced in [ADPS16], that considers a single call. 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. To summarise, there are ten sieving-based cost models considered in the first round submissions, all of which are the result of combinations of the above assumptions. Of the ten cost models, the five classical models are:

1. 0.292β,
2. $0.292\beta + 16.4$,
3. 0.368β,
4. $0.292\beta + \log(\beta)$, and
5. $0.292\beta + 16.4 + \log(8d)$.

The five quantum models are:

6. 0.265β,
7. $0.265\beta + 16.4$,
8. 0.2975β,

9. $0.265\beta + \log(\beta)$, and
10. $0.265\beta + 16.4 + \log(8d)$.

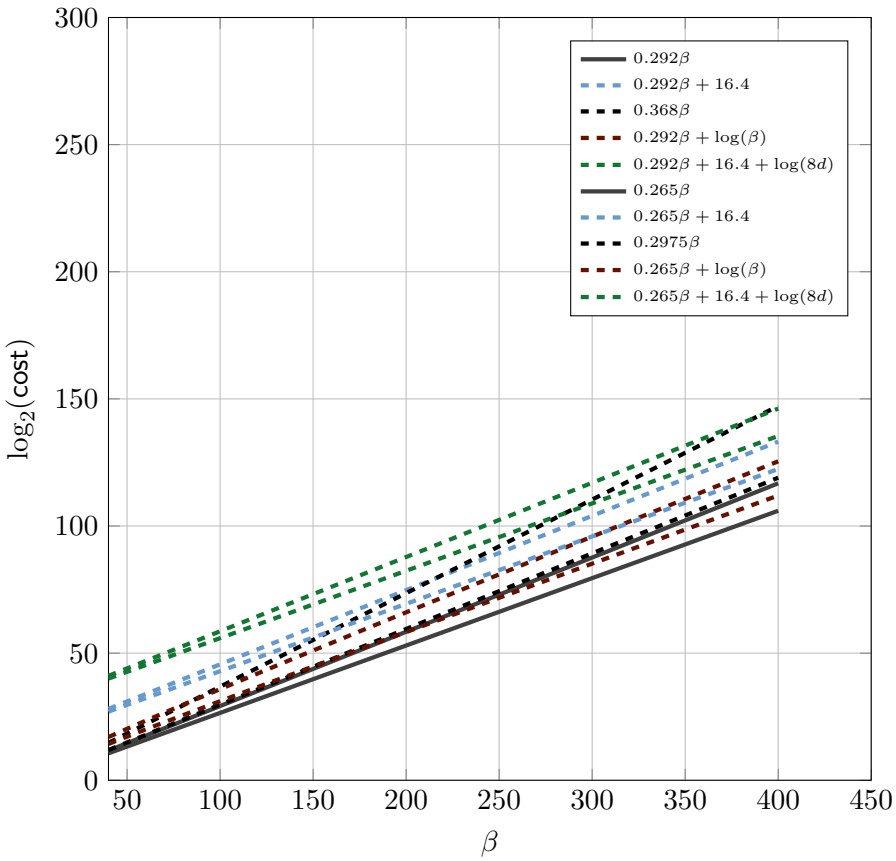


Figure 4.2: Sieving-based cost models, used as part of a first round submission to the NIST standardisation process, for a lattice of dimension $d = 1024$ with $40 \leq \beta \leq 400$.

4.3.3 Cost Models Used in the Submissions

In this section we match the models outlined in Section 4.3 to the first round submissions. Specifically, in Table 4.3, we outline the cost models considered by each individual submission. We note that some submissions consider several models, and cost their scheme under each model (for example, LIMA [SAL+17] considers both the $0.292\beta + 16.4$ and $0.265\beta + 16.4$ models).

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". As discussed in Chapter 3,

it is typically assumed that processing one node costs about 100 CPU cycles [CN11]. For sieving, the elementary operation is typically an operation on word-sized integers, costing about one CPU cycle (recall, however, that the cost models including a $\log(\beta)$ term assume that a sieving cost model of the form $2^{c\beta}$ requires $\beta 2^{c\beta}$ CPU cycles). For quantum algorithms the unit is typically the number of Grover iterations required, and 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.

4.4 Parameter Sets

In this section we outline the individual parameter sets considered inside *every submission*. In this work we consider Learning with Rounding-based, Ring/Module LWE-based, and NTRU-based parameter sets.

Learning with Rounding. Recall from Chapter 2 that the Learning with Rounding (LWR) problem replaces the addition of a noise term (used in LWE) with a deterministic rounding process. An instance of the LWR problem is of the form:

$$\left(\mathbf{a}, b := \left\lfloor \frac{p}{q}\langle \mathbf{a}, \mathbf{s}\rangle \right\rceil\right) \in \mathbb{Z}_q^n \times \mathbb{Z}_p,$$

and we can interpret this 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 chosen uniformly from the set $\{\frac{-q}{2p} + 1, \ldots, \frac{q}{2p}\}$ [Ngu18]. We can therefore view LWR samples as LWE samples with modulus q and error distribution uniform over the set $\{\frac{-q}{2p} + 1, \ldots, \frac{q}{2p}\}$.

Ring/Module Learning with Errors. We view Ring-LWE and Module-LWE instances as LWE instances by considering the coefficients of elements in R_q as vectors in $\mathbb{Z}_q^n$ and ignoring any algebraic structure of R_q. This approach is standard when considering the complexity of algorithms solving the Ring-LWE and Module-LWE problems due to the lack of cryptanalytic techniques exploiting the algebraic structure.

NTRU. Let $(\mathbf{f}, \mathbf{g}) \in \mathbb{Z}^{2n}$ be the NTRU secret as introduced in Chapter 2. We treat $\mathbf{f}$ as the LWE secret $\mathbf{s}$, and $\mathbf{g}$ as the LWE error $\mathbf{e}$ (we note that this can also be considered vice-versa).

Model	Schemes
	CRYSTALS [LDK+17, SAB+17]
	SABER [DKRV17]
	Falcon [PFH+17]
	ThreeBears [Ham17]
	HILA5 [Saa17]
0.292β	Titanium [SSZ17]
0.265β	KINDI [El 17]
	NTRU HRSS [SHRS17]
	LAC [LLJ+17]
	NTRUEncrypt [ZCHW17a]
	New Hope [PAA+17]
	pqNTRUSign [ZCHW17b]
	Round2 [GZB+17]
$0.292\beta + 16.4$ $0.265\beta + 16.4$	LIMA [SAL+17]
0.368β 0.2975β	NTRU HRSS [SHRS17]
	Frodo [NAB+17]
$0.292\beta + \log(\beta)$	KCL [ZJGS17]
$0.265\beta + \log(\beta)$	Lizard [CPL+17]
	Round2 [GZB+17]
$0.292\beta + 16.4 + \log(8d)$	Ding Key Exchange [DTGW17]
	EMBLEM [SPL+17]
$0.265\beta + 16.4 + \log(8d)$	qTESLA [BAA+17]
	NTRU HRSS [SHRS17]
$0.187\beta \log \beta - 1.019\beta + 16.1$	pqNTRUSign [ZCHW17b]
	NTRUEncrypt [ZCHW17a]
$\frac{1}{2}(0.187\beta \log \beta - 1.019\beta + 16.1)$	NTRU HRSS [SHRS17]
$0.000784\beta^2 + 0.366\beta - 0.9 + \log(8d)$	NTRU Prime [BCLv17]
$0.125\beta \log \beta - 0.755\beta + 2.25$	LOTUS [PHAM17]

Table 4.3: All cost models proposed as part of a submission to the first round of the NIST standardisation procedure. The name of a model is the log (to the base 2) of its cost.

The LWE degree n is exactly the degree of the NTRU polynomial ϕ, the standard deviation of the LWE error distribution is set to $\|\mathbf{g}\|_2/\sqrt{n}$, and the LWE modulus q is exactly the NTRU modulus. We account for the presence of rotations by amplifying the success probability p of guessing entries of the secret correctly to $1-(1-p)^k$, where k is the number of rotations.

In Table 4.4 we present the parameters considered in the NTRU-based submissions. In Table 4.5, these parameters have been represented as LWE samples using the techniques discussed above. In Table 4.6, we outline the parameters considered in the LWE and LWR-based schemes.

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
SNTRU 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 4.4: Parameter sets for NTRU-based schemes with secret dimension n, modulo 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 4.5: LWE parameter sets for NTRU-based schemes, with dimension n, modulo q, standard deviation of the error σ, and ring $\mathbb{Z}_q[x]/(\phi)$. 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	Ring-LWE	x^n+1	KEM
KCL-MLWE	768	3	7681	1.00	normal	4	Module-LWE	$x^{n/k}+1$	KEM
	768	3	7681	2.24	normal	4	Module-LWE	$x^{n/k}+1$	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	Module-LWE	$x^{n/k}+1$	SIG
	1024	4	8380417	3.16	$(-5,5)$	2	Module-LWE	$x^{n/k}+1$	SIG
	1280	5	8380417	2.00	$(-3,3)$	3	Module-LWE	$x^{n/k}+1$	SIG
CRYSTALS-Kyber	512	2	7681	1.58	normal	1	Module-LWE	$x^{n/k}+1$	KEM, PKE
	768	3	7681	1.41	normal	3	Module-LWE	$x^{n/k}+1$	KEM, PKE
	1024	4	7681	1.22	normal	5	Module-LWE	$x^{n/k}+1$	KEM, PKE
Ding Key Exchange	512	—	120883	4.19	normal	1	Ring-LWE	x^n+1	KEM
	1024	—	120883	2.60	normal	3, 5	Ring-LWE	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	Ring-LWE	x^n+1 †	KEM, PKE
	512	—	16384	3.00	$(-1,1)$	1	Ring-LWE	x^n+1 †	KEM, PKE

4.4 Parameter Sets

Name	n	k	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
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	Ring-LWE	$x^n + 1$	KEM, PKE
	1024	—	12289	2.00	normal	5	Ring-LWE	$x^n + 1$	KEM, PKE
HILA5	1024	—	12289	2.83	normal	5	Ring-LWE	$x^n + 1$	KE
KINDI	768	3	16384	2.29	$(-4, 4)$	2	Module-LWE	$x^{n/k} + 1$	KEM, PKE
	1024	2	8192	1.12	$(-2, 2)$	4	Module-LWE	$x^{n/k} + 1$	KEM, PKE
	1024	2	16384	2.29	$(-4, 4)$	4	Module-LWE	$x^{n/k} + 1$	KEM, PKE
	1280	5	16384	1.12	$(-2, 2)$	5	Module-LWE	$x^{n/k} + 1$	KEM, PKE
	1536	3	8192	1.12	$(-2, 2)$	5	Module-LWE	$x^{n/k} + 1$	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	Ring-LWE	$x^n + 1$	KEM, PKE
	2048	—	184321	3.16	normal	4	Ring-LWE	$x^n + 1$	KEM, PKE
LIMA-sp	1018	—	12521473	3.16	normal	1	Ring-LWE	$\sum_{i=0}^{n} x^i$	KEM, PKE
	1306	—	48181249	3.16	normal	2	Ring-LWE	$\sum_{i=0}^{n} x^i$	KEM, PKE
	1822	—	44802049	3.16	normal	3	Ring-LWE	$\sum_{i=0}^{n} x^i$	KEM, PKE
	2062	—	16900097	3.16	normal	4	Ring-LWE	$\sum_{i=0}^{n} x^i$	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	Ring-LWE, Ring-LWR	$x^n + 1$	KEM, PKE
	1024	—	2048	1.12	$((-1, 1), 264)$	3	Ring-LWE, Ring-LWR	$x^n + 1$	KEM, PKE
	2048	—	2048	1.12	$((-1, 1), 164)$	3	Ring-LWE, Ring-LWR	$x^n + 1$	KEM, PKE
	2048	—	4096	1.12	$((-1, 1), 256)$	5	Ring-LWE, Ring-LWR	$x^n + 1$	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	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM
	522	—	32768	36.95	$((-1, 1), 78)$	2	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM
	540	—	16384	18.47	$((-1, 1), 96)$	3	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM
	700	—	32768	36.95	$((-1, 1), 112)$	4	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM
	676	—	32768	36.95	$((-1, 1), 120)$	5	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM

4.4 Parameter Sets

Name	n	k	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
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	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
	540	—	8192	4.61	$((-1,1),96)$	2	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
	586	—	8192	4.61	$((-1,1),104)$	3	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
	708	—	32768	18.47	$((-1,1),140)$	4, 5	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
nRound2.KEM	400	—	3209	3.61	$((-1,1),72)$	1	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM
	486	—	1949	2.18	$((-1,1),96)$	2	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM
	556	—	3343	3.76	$((-1,1),88)$	3	Ring-LWRR	$\sum_{i=0}^{n} x^i$	KEM
	658	—	1319	1.46	$((-1,1),130)$	4, 5	Ring-LWR	$\sum_{i=0}^{n} x^i$	KEM
nRound2.PKE	442	—	2659	1.47	$((-1,1),74)$	1	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
	556	—	3343	1.86	$((-1,1),88)$	2	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
	576	—	2309	1.27	$((-1,1),108)$	3	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
	708	—	2837	1.57	$((-1,1),140)$	4, 5	Ring-LWR	$\sum_{i=0}^{n} x^i$	PKE
LightSaber	512	2	8192	2.29	normal	1	Module-LWR	$x^{n/k}+1$	KEM, PKE
NTRU LPrime	761	—	4591	0.82	$((-1,1),250)$	5	Ring-LWR	$x^n - x - 1$	KEM
Saber	768	3	8192	2.29	normal	3	Module-LWR	$x^{n/k}+1$	KEM, PKE
FireSaber	1024	4	8192	2.29	normal	5	Module-LWR	$x^{n/k}+1$	KEM, PKE
qTESLA	1024	—	8058881	8.49	normal	1	Ring-LWE	x^n+1	SIG
	2048	—	12681217	8.49	normal	3	Ring-LWE	x^n+1	SIG
	2048	—	27627521	8.49	normal	5	Ring-LWE	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 4.6: Parameter sets for LWE-based schemes with secret dimension n, Module-LWE rank k (if any), modulo 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 relies on a family of rings where $f_i \in \{-1,0,1\}$ and $f_0 \in \{-1,1\}$. † 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].

In Tables 4.5 and 4.6 we see that a total of 104 parameter sets were submitted to the first round. Of these parameter sets, nine consider the NTRU assumption, 23 consider the LWE and/or LWR assumptions, 39 consider the Ring-LWE and/or Ring-LWR assumptions, 16 consider the Module-LWE and/or Module-LWR assumptions, and a further 17 consider alternative LWE-based assumptions[1].

4.5 Small Secret Variants of the uSVP and Dual Attacks

The uSVP and dual attacks can be optimised for variants of Small-secret LWE. In a similar manner to the decoding attack in Chapter 3, we can combine the uSVP and dual attacks with combinatorics to retrieve a complexity improvement. In this section, we introduce the small-secret variants of the uSVP and dual attacks.

4.5.1 uSVP

The uSVP attack on Small-secret LWE considers the Bai and Galbraith embedding [BG14]. As opposed to the uSVP variant outlined in Chapter 2, where the target vector is $(\mathbf{e}, 1)$, the Bai and Galbraith embedding constructs the lattice with basis matrix $\mathbf{B}$, where:

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

The vector $(\mathbf{s}, \mathbf{e}, 1)$ is embedded in the lattice $L(\mathbf{B})$, and this can be seen since:

$$\begin{pmatrix} \mathbf{I}_n & \mathbf{0} & \mathbf{0} \\ -\mathbf{A} & q\mathbf{I}_m & \mathbf{b} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{s} \\ * \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{s} \\ -\mathbf{A}\mathbf{s} + q* + \mathbf{b} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{s} \\ \mathbf{e} \\ 1 \end{pmatrix} \bmod q.$$

In the case of *normal-form* LWE, we can view this variant of the uSVP attack as identical to the approach considered in Chapter 2. However, we note that in the small secret variant of this attack, we can employ *dimension reduction* techniques [MS01, APS15, Alb17] to reduce the overall complexity. That is, we guess τ components of the LWE secret, before solving

[1] These assumptions are Polynomial-LWE [SSTX09], and Integer Module-LWE [Ham17].

uSVP in the $(n + m - \tau)$ dimension lattice with basis:

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_{n-\tau} & \mathbf{0} & \mathbf{0} \\ -\mathbf{A}_{(\tau)} & q\mathbf{I}_m & \mathbf{b} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix},$$

where we recall that $\mathbf{A}_{(\tau)}$ corresponds to the matrix $\mathbf{A}$ with the first τ columns removed. If we correctly guess zeros then the vector $(s_{\tau+1}, s_{\tau+2}, \ldots, s_n, \mathbf{e}, 1)$ is embedded in the lattice $L(\mathbf{B})$ reduced by BKZ, and this allows for complete recovery of the LWE secret. If we *incorrectly* guess the zero components, then we have to restart this process with a fresh guess of τ zero components. The running time of the dimension reduced problem, constrained by the [ADPS16] success condition outlined in Chapter 2, is:

$$T_{uSVP} = \min_{\beta,\tau,m} \left\{ \frac{1}{p_\tau} \cdot T_{BKZ}(\beta, n + m - \tau) \right\},$$

where p_τ is the probability of correctly guessing τ zero components of the LWE secret, and $T_{BKZ}(\beta, n + m - \tau)$ is the cost of running BKZ on a dimension $(n + m - \tau)$ lattice with blocksize β. We note that setting $\tau = 0$ corresponds to not performing any guessing.

4.5.2 Dual

Recall the dual attack from Section 2.9.1, which finds short vectors $\mathbf{v}$ in the lattice:

$$\Lambda^* = \{\mathbf{x} \in \mathbb{Z}_q^m \mid \mathbf{x}\mathbf{A} \equiv \mathbf{0} \bmod q\},$$

before computing inner products of the form:

$$\langle \mathbf{v}, \mathbf{b} \rangle = \langle \mathbf{v}, \mathbf{A}\mathbf{s} + \mathbf{e} \rangle = \langle \mathbf{v}\mathbf{A}, \mathbf{s} \rangle + \langle \mathbf{v}, \mathbf{e} \rangle = \langle \mathbf{v}, \mathbf{e} \rangle \bmod q.$$

Inner products of this form allow us to distinguish LWE from random. For small and/or sparse secrets, Albrecht suggests several improvements to this attack [Alb17], which we summarise in the following four subsections.

4.5.2.1 Combinatorics

For sparse secrets, Albrecht suggests splitting the matrix $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$ where $\mathbf{A}_0 \in \mathbb{Z}_q^{m \times (n-\tau)}$ and $\mathbf{A}_1 \in \mathbb{Z}_q^{m \times \tau}$. Splitting the LWE secret $\mathbf{s} = (\mathbf{s}_0, \mathbf{s}_1)$ in the same manner, we can see that:

$$\mathbf{A}\mathbf{s} = \mathbf{A}_0\mathbf{s}_0 + \mathbf{A}_1\mathbf{s}_1.$$

In this variant of the attack, we begin by finding short vectors in the lattice:

$$\Lambda' = \{\mathbf{v} \in \mathbb{Z}^m \mid \mathbf{v}\mathbf{A}_0 \equiv 0 \bmod q\},$$

which corresponds to guessing τ components of the LWE secret as zero, i.e. $(\mathbf{s}_1 = 0)$ and hoping that $\mathbf{A}\mathbf{s} = \mathbf{A}_0\mathbf{s}_0$, or equivalently that $\mathbf{A}_1\mathbf{s}_1 \equiv 0 \bmod q$ (in a similar manner to the hybrid-decoding attack outlined in Chapter 3, and the small-secret uSVP attack outlined in Section 4.5.1). Note that setting $\tau = 0$ corresponds to not performing any guessing, and therefore reduces to the original dual attack. Suppose, then, that we proceed with the following steps:

1. we collect ν short vectors $\{\mathbf{v}_i\}_{i=1}^{\nu}$ each contained within the lattice Λ', typically by performing lattice reduction ν times, and
2. compute a collection of inner products $\{\tilde{e}_i = \langle \mathbf{v}_i, \mathbf{b}\rangle\}_{i=1}^{\nu}$.

We have:

$$\begin{aligned}\tilde{e}_i = \langle \mathbf{v}_i, \mathbf{b}\rangle = \langle \mathbf{v}_i, \mathbf{A}\mathbf{s} + \mathbf{e}\rangle &= \langle \mathbf{v}_i\mathbf{A}, \mathbf{s}\rangle + \langle \mathbf{v}_i, \mathbf{e}\rangle \\ &= \langle \mathbf{v}_i\mathbf{A}_0, \mathbf{s}_0\rangle + \langle \mathbf{v}_i\mathbf{A}_1, \mathbf{s}_1\rangle + \langle \mathbf{v}_i, \mathbf{e}\rangle \\ &= \langle \mathbf{v}_i\mathbf{A}_1, \mathbf{s}_1\rangle + \langle \mathbf{v}_i, \mathbf{e}\rangle,\end{aligned}$$

since $\mathbf{v}_i\mathbf{A}_0 \equiv 0 \bmod q$. If our guess of τ zeros (i.e. $\mathbf{s}_1 = \mathbf{0}$) was correct, then these computed terms are of the form $\langle \mathbf{v}_i, \mathbf{e}\rangle$ since $\langle \mathbf{v}_i\mathbf{A}_1, \mathbf{s}_1\rangle = 0$. In this case, the $\tilde{e}_i$ terms follow a Discrete Gaussian distribution and we have solved Decision-LWE.

Otherwise, we know that our guess of τ zeros is *incorrect*, and we can therefore make additional guesses in the τ-dimensional guessing space (which, for example, if $\mathbf{s} \leftarrow \mathcal{B}^-$ would be $\{-1, 0, 1\}^\tau$). For each guess, we compare the corresponding distributions provided by the inner products against the uniform distribution modulo q. Specifically, we know that the additional term is of the form $\langle \mathbf{v}_i\mathbf{A}_1, \mathbf{s}_1\rangle$ and we can therefore search over possible secrets $\mathbf{s}_1$ and attempt to remove this additional term from the inner product/s. In particular, for candidate guesses $\mathbf{s}'$ we construct terms of the form:

$$e_i'' = e_i' - \langle \mathbf{v}_i\mathbf{A}_1\mathbf{s}'\rangle.$$

For the correct candidate secret, i.e. $\mathbf{s}' = \mathbf{s}_1$, the terms e_i'' are of the form $\langle \mathbf{v}_i, \mathbf{e}\rangle$ as is required.

4.5.2.2 Amortising the Cost of Lattice Reduction

Albrecht also considers re-randomisation techniques to amortise the cost of lattice reduction. Recall that, for a successful attack, we may require ν short vectors from the lattice $L(\mathbf{B})$. To retrieve these vectors, one could imagine re-randomising the input basis before running the BKZ algorithm on a new, randomised, lattice basis:

$$\tilde{\mathbf{B}}_i \leftarrow \mathsf{BKZ}_\beta(\mathbf{U}_i\mathbf{B}),$$

ν times, and using the shortest vectors in the lattice bases $\{\tilde{\mathbf{B}}_i\}_{i=1}^{\nu}$. Albrecht notes that this is unnecessary and suggests performing an initial, heavy, lattice reduction step:

$$\tilde{\mathbf{B}} \leftarrow \mathsf{BKZ}_\beta(\mathbf{B}),$$

and applying randomisation *after* this lattice reduction has taken place. That is, we can use the shortest vectors in the lattice bases:

$$\mathbf{B}'_i \leftarrow \mathsf{BKZ}_{\beta'}(\mathbf{U}_i\tilde{\mathbf{B}}),$$

for some blocksize $\beta' \ll \beta$. The idea is to take the shortest vector from each of these re-randomised lattice bases, and use the resulting set of short vectors to distinguish against the uniform distribution modulo q. The length of the short vectors generated via this re-randomisation process are *longer* than those generated using using a fresh call to BKZ-β. Albrecht suggests that these vectors have their norms increased by a factor of two compared to the initial vector retrieved directly from BKZ-β, and provides experimental evidence to back up this heuristic [Alb17]. However, these longer vectors can be generated at a *significantly lower cost* via this amortisation technique. The hope is that this trade-off in vector length vs computation time provides a speed-up in the overall attack cost.

4.5.2.3 Scaled Normal Form

The final technique stems from the observation that it is sufficient to find a short vector $(\mathbf{v}, \mathbf{w})$ in the lattice given by:

$$\Lambda'' = \left\{ (\mathbf{v}, \mathbf{w}) \in \mathbb{Z}^m \times \left(\frac{1}{c} \cdot \mathbb{Z}\right)^n \middle| \mathbf{v}\mathbf{A} \equiv c\mathbf{w} \bmod q \right\},$$

If we find such a short vector, then we note that:

$$\langle \mathbf{v}, \mathbf{b} \rangle = \langle \mathbf{v}, \mathbf{A}\mathbf{s} + \mathbf{e} \rangle = \langle \mathbf{v}\mathbf{A}, \mathbf{s} \rangle + \langle \mathbf{v}, \mathbf{e} \rangle = \langle c\mathbf{w}, \mathbf{s} \rangle + \langle \mathbf{v}, \mathbf{e} \rangle,$$

and, assuming that $\mathbf{w}, \mathbf{s}$ are both sufficiently short, we can *still* distinguish LWE from random modulo q, albeit with a different advantage, since the additional error term $\langle c\mathbf{w}, \mathbf{s}\rangle$ is small. A scaling factor c is computed to balance the contributions of the two terms (i.e. $\langle c\mathbf{w}, \mathbf{s}\rangle$ and $\langle \mathbf{v}, \mathbf{e}\rangle$), which is computed as:

$$c = \frac{\alpha q}{\sqrt{2\pi h}}\sqrt{m-n},$$

in [Alb17].

4.5.2.4 Combining Techniques

The small-secret variant of the dual attack combines these techniques (where appropriate) to leverage both the smallness *as well as* any sparsity of the LWE secret vector $\mathbf{s}$. We note that these techniques are implemented in the LWE Estimator for both binary and ternary secrets.

4.5.3 Multiple Hardness Assumptions

The Lizard (RLizard) scheme is based on two hardness assumptions, LWE (Ring-LWE) and LWR (Ring-LWR). Secret key recovery corresponds to the underlying LWE problem, and ephemeral key recovery corresponds to the underlying LWR problem. There exists parameter sets for which secret key recovery is harder than ephemeral key recovery (i.e. the underlying LWE problem is harder than the underlying LWR problem), and there also exists parameter sets for which the converse is true, i.e. ephemeral key recovery is harder than secret key recovery. To deal with this issue, in each cost model, for every attack, we consider both the cost of ephemeral key recovery and the cost of secret key recovery and always choose the lower of the two estimates.

4.5.4 Number of Samples

LWE as defined in Definition 2.32 provides the adversary with an arbitrary number of samples denoted by m. When using the LWE Estimator we can set the value $m = \infty$ and allow the adversary to have access to an infinite number of samples. In practice, however, this is not the case. In particular, in the Ring-LWE KEM setting – which is the most common for the

schemes considered in this chapter – the public key is one Ring-LWE sample:

$$(a, b) = (a, as + e),$$

for some short s, e, and encapsulations consist of *two* Ring-LWE samples

$$va + e',$$

and:

$$vb + e'' + \tilde{m},$$

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.

In a typical setting, though, the adversary does not get to enjoy the full power of having two Ring-LWE samples at its disposal, because, firstly, the random string $\tilde{m}$ increases the noise in $vb + e'' + \tilde{m}$ and, secondly, because many schemes drop lower order bits from $vb + e'' + \tilde{m}$ 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. 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 work, we consider the $m = n$ and $m = 2n$ scenarios for all schemes. We note that, for many schemes, n samples are sufficient to run the most efficient variant of either attack.

4.6 First Round Security Estimates

We provide estimates for all first round parameter sets considered as part of a submission, under all BKZ cost models considered as part of a submission, for both the uSVP and dual attacks (where appropriate). This corresponds to over 150 parameter sets, under 14 cost models, yielding more than 2000 individual security estimates. Our results can be found at `https://estimate-all-the-lwe-ntru-schemes.github.io/docs/`. In this chapter, we present a small subset of the results. Specifically, in Table 4.7, we present the results for a single parameter set considered in each of the schemes EMBLEM, CRYSTALS-Kyber, NTRU Prime, and uRound2.KEM, under the two cost models 0.292β and $0.187\beta \log(\beta) - 1.019\beta + 16.1$. In each case we highlight the associated value of n for clarity.

Scheme	Assumption	Claimed Security	NIST Category	Attack	$0.187\beta\log(\beta) - 1.019\beta + 16.1$	0.292β
EMBLEM ($n = 611$)	LWE	128	1	dual	142	76
EMBLEM ($n = 611$)	LWE	128	1	primal-usvp	152	91
Kyber ($n = 512$)	Ring-LWE	128	1	dual	289	141
Kyber ($n = 512$)	Ring-LWE	128	1	primal-usvp	244	113
NTRULPrime ($n = 761$)	Ring-LWR	128	5	dual	365	179
NTRULPrime ($n = 761$)	Ring-LWR	128	5	primal-usvp	365	156
uRound2.KEM ($n = 500$)	LWR	128	1	dual	127	94
uRound2.KEM ($n = 500$)	LWR	128	1	primal-usvp	126	84

Table 4.7: Estimates for the first round submissions EMBLEM, NewHope, NTRU Prime, and uRound2.KEM under the 0.292β and $0.187\beta\log(\beta) - 1.019\beta + 16.1$ cost models. Estimates are taken directly from `https://estimate-all-the-lwe-ntru-schemes.github.io/docs/`.

4.6.1 Observation: Cost Swaps

One of the interesting observations from our data is that cost models for lattice reduction do not necessarily preserve the ordering of the schemes under consideration. More explicitly: if scheme A is considered harder to break than scheme B under cost model 1, the same does not necessarily hold for cost model 2. That is, under cost model 2, scheme B could be considered harder to break than scheme A.

To find an example of this, we consider EMBLEM and uRound2.KEM, as highlighted in [Ber18]. As can been seen in Table 4.8, in the sieving-based cost model 0.292β, the associated security estimate of EMBLEM is 76 bits and uRound2.KEM is 84 bits, making EMBLEM easier to break. However, under the enumeration-based cost model $0.187\beta\log(\beta) - 1.019\beta + 16.1$ the associated security estimate of EMBLEM is 142 bits and uRound2.KEM is 126 bits, making uRound2.KEM easier to break.

	EMBLEM	uRound2.KEM
0.292β	**76**	84
$0.187\beta\log(\beta) - 1.019\beta + 16.1$	142	**126**

Table 4.8: Security estimates for the first round variants of EMBLEM and uRound2.KEM. Best in class are highlighted in bold, and we can see this is an example of a cost swap: in the 0.292β model uRound2.KEM has a higher security estimate (84-bits vs 76-bits), whereas in the $0.187\beta\log(\beta) - 1.019\beta + 16.1$ model EMBLEM has a higher security estimate (142-bits vs 126-bits).

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 bits) 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. As part of our work, we identified two cases in which this can happen: the first involves sparse secrets, and the second concerns the dual attack.

4.6.1.1 Sparse secrets

The first class of cases involves instances with sparse secrets. The LWE Estimator applies guessing strategies when costing the dual attack [Alb17] and the primal attack. The basic idea, as discussed throughout this chapter, is that 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 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. This 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 cost model, we have that the cost of enumeration is fitted as $2^{\Theta(\beta \log \beta)}$ 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 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 4.3, 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 $0.187\beta \log \beta - 1.019\beta + 16.1$ model the optimal choice is $\tau = 197$.

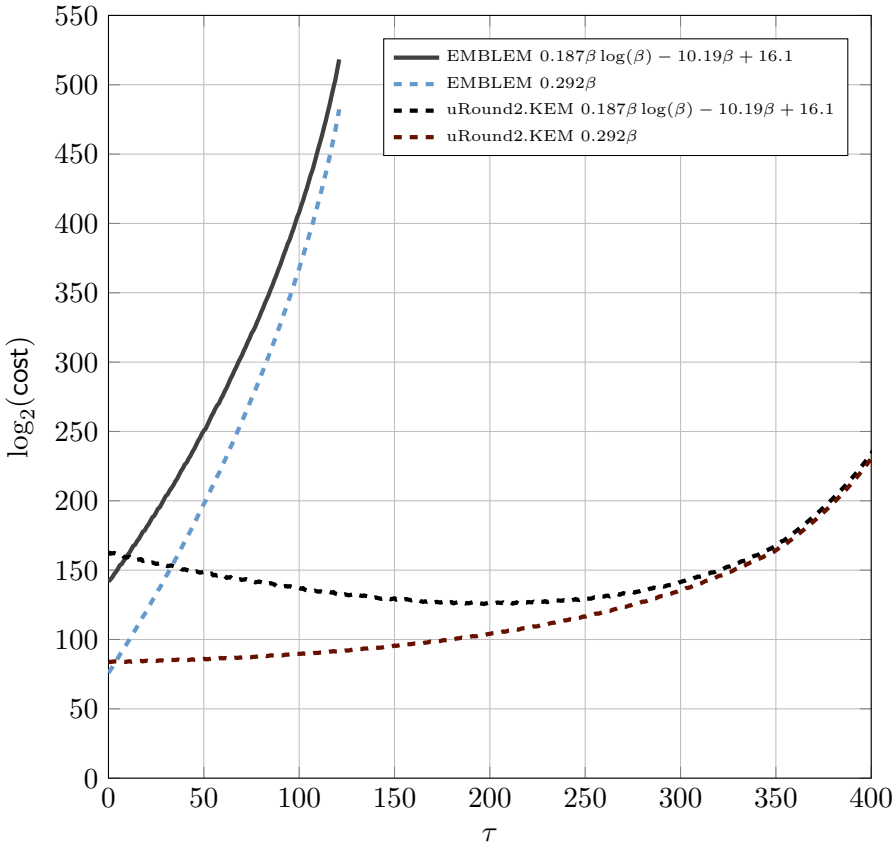


Figure 4.3: Estimates of the cost of the primal attack when guessing τ secret entries for the schemes EMBLEM ($n = 611$) and uRound2.KEM ($n = 500$).

4.6.1.2 The Dual attack

The second class of cases can be observed for the dual attack. Recall that the dual attack runs lattice reduction to find a small vector $\mathbf{v}$ in the scaled dual lattice of $\mathbf{A}$, and then considers $\langle \mathbf{v}, \mathbf{b} \rangle$ which is short when $\mathbf{A}, \mathbf{b}$ is an LWE sample. In more detail, the advantage of distinguishing $\langle \mathbf{v}, \mathbf{b} \rangle$, as discussed in Chapter 2, is:

$$\varepsilon = \exp(-\delta^{2\,d} c_0),$$

for some constant c_0 depending on the instance and with d being the dimension of the lattice under consideration [LP11]. To amplify this advantage to a constant advantage, we have to repeat the experiment roughly $1/\varepsilon^2$ times. Therefore, the overall cost of the attack is approximately:

$$T_{\mathsf{BKZ}}(\beta, d)/\exp(-\delta^{2\,d} c_0)^2.$$

In the sieving regime we have $T_{\mathsf{BKZ}}(\beta, d) \approx 2^{c_1\beta}$, and in the enumeration regime we have: $T_{\mathsf{BKZ}}(\beta, d) \approx \beta^{c_2\beta}$ (from enumeration costing $2^{\Theta(\beta \log \beta)}$). For large β we have $\delta \approx \beta^{1/2\beta}$ [Che13], and thus we have overall log costs of roughly:

$$c_1\,\beta + 2\,\log(e)\,\beta^{d/\beta}\,c_0,$$

and

$$c_2\,\beta\log(\beta) + 2\,\log(e)\,\beta^{d/\beta}\,c_0.$$

We wish to minimise both expressions (under the constraint that $\beta \geq 2$) and the optimal trade-off depends on c_0, c_1 and c_2. In particular, the optimal β in the sieving regime is not necessarily the optimal β in the enumeration regime.

4.7 Second Round Submissions

In January 2019, NIST announced the second round submissions. Of the 69 submissions from the first round, 26 submissions made it through to the second round [Moo19]. The 23 lattice-based submission from the first round became 12. At this point, designers were allowed to make more significant changes to their submissions including e.g. the merging of multiple submissions, design changes, and new parameter sets. The second round submissions are outlined in Table 4.9.

4.8 The Third Round

In July 2020, NIST announced the third round submissions. Of the 26 submissions from the second round, NIST announced seven third round *finalists*, as well as eight *candidate* algorithms. Of these eight finalists, five are lattice-based submissions, and of the seven candidate algorithms, two are lattice-based submissions [Moo20]. The third round submissions are outlined in Table 4.10.

4.8.1 Cost Models

Next, we consider which of the cost models from the first round, given in Table 4.3, are considered in third round submissions, and we present these cost models in Table 4.11.

Assumption	Schemes	Notes
LWE Variants	Crystals-Dilithium [LDK+19]	
	Crystals-Kyber [SAB+19]	
	Frodo [NAB+19]	
	LAC [LLJ+19]	
	NewHope [PAA+19]	
	Round5 [GZB+19]	Merge of HILA5 and Round2
	Saber [DKRV19]	
	ThreeBears [Ham19]	
	qTesla [BAA+19]	
NTRU Variants	Falcon [PFH+19]	
	NTRU [ZCH+19]	Merge of NTRUEncrypt and NTRU HRSS
	NTRU Prime [BCLv19]	

Table 4.9: Second round lattice-based submissions to the NIST standardisation process.

Assumption	Schemes	Notes
LWE Variants	Crystals-Dilithium [LDK+20]	Finalist
	Crystals-Kyber [SAB+20]	Finalist
	Frodo [NAB+20]	Candidate
	Saber [DKR+20]	Finalist
NTRU Variants	Falcon [PFH+20]	Finalist
	NTRU [ZCH+20]	Finalist
	NTRU Prime [BCLv20]	Candidate

Table 4.10: Third round lattice-based submissions to the NIST standardisation process.

Model	Schemes
	SABER [DKR+20]
0.292β	FALCON [PFH+20]
0.265β	NTRU Prime [BCLv20]
	Crystals-Kyber [SAB+20]
	Crystals-Dilithium [LDK+20]
$0.396\beta^{\dagger}$	NTRU Prime [BCLv20]
$0.3496\beta^{\dagger}$	NTRU [ZCH+20]
$0.4150\beta^{\dagger}$	NTRU [ZCH+20]
$0.292\beta + \log(\beta)$	
$0.265\beta + \log(\beta)$	Frodo [NAB+20]
$0.2075\beta + \log(\beta)^{\dagger}$	
$0.187\beta \log \beta - 1.019\beta + 16.1$	NTRU Prime [BCLv20]
$\frac{1}{2}(0.187\beta \log \beta - 1.019\beta + 16.1)$	NTRU Prime [BCLv20]

Table 4.11: All cost models proposed as part of a submission to the third round of the NIST standardisation procedure. The name of a model is the log (to the base 2) of its cost. Cost models which were not used as part of our analysis for the first round submissions are marked with a †. The 0.396β model considers the 0.292β model mapped to the AT metric [Ber20]. The 0.3496β and 0.4150β models are used in the NTRU submission under the assumption of a "local" model of computation. The $0.2075\beta + \log(\beta)$ model was mentioned in the round one Frodo submission, but wasn't used to produce estimates.

4.8.2 Parameter Sets and Estimates

Next, we consider the seven lattice-based schemes listed in Table 4.10. We present updated parameter selections in Tables 4.12, 4.13, and 4.14. Finally, we present Core-SVP estimates, both in the classical regime (0.292β) and the quantum regime (0.265β) for the Round 3 parameter sets in Tables 4.15 and 4.16.

Name	n	q	$\|f\|$	$\|g\|$	NIST	Assumption	ϕ	Primitive
NTRU	509	2048	$\sqrt{2/3}\sqrt{509}$	$\sqrt{q/8-2}$	1	NTRU	x^n-1	KEM, PKE
	677	2048	$\sqrt{2/3}\sqrt{677}$	$\sqrt{q/8-2}$	3	NTRU	x^n-1	KEM, PKE
	821	4096	$\sqrt{2/3}\sqrt{821}$	$\sqrt{q/8-2}$	5	NTRU	x^n-1	KEM, PKE
	701	8192	20.92	20.92	3	NTRU	x^n-1	KEM, PKE
Falcon	512	12289	$1.17\sqrt{q/2}$	$1.17\sqrt{q/2}$	1	NTRU	x^n+1	SIG
	1024	12289	$1.17\sqrt{q/2}$	$1.17\sqrt{q/2}$	5	NTRU	x^n+1	SIG
SNTRU Prime	653	4621	$\sqrt{288}$	$\sqrt{653}\sqrt{2/3}$	1	NTRU	x^n-x-1	KEM
	761	4591	$\sqrt{286}$	$\sqrt{761}\sqrt{2/3}$	2	NTRU	x^n-x-1	KEM
	857	5167	$\sqrt{322}$	$\sqrt{857}\sqrt{2/3}$	2	NTRU	x^n-x-1	KEM
	953	6343	$\sqrt{396}$	$\sqrt{953}\sqrt{2/3}$	3	NTRU	x^n-x-1	KEM
	1013	7177	$\sqrt{448}$	$\sqrt{1013}\sqrt{2/3}$	4	NTRU	x^n-x-1	KEM
	1277	7879	$\sqrt{492}$	$\sqrt{1277}\sqrt{2/3}$	5	NTRU	x^n-x-1	KEM

Table 4.12: Parameter sets for third round NTRU-based schemes with secret dimension n, modulo q, small polynomials f and g, and ring $\mathbb{Z}_q[x]/(\phi)$. The NIST column indicates the NIST security category aimed at. Each parameter set from SNTRU Prime has been assigned two security levels in the round 3 submission, and we always choose the lowest of the two.

Name	n	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
NTRU	509	2048	$\sqrt{2/3}$	$((-1,1), q/8-2)$	1	NTRU	x^n-1	KEM, PKE
	677	2048	$\sqrt{2/3}$	$((-1,1), q/8-2)$	3	NTRU	x^n-1	KEM, PKE
	821	4096	$\sqrt{2/3}$	$((-1,1), q/8-2)$	5	NTRU	x^n-1	KEM, PKE
	701	8192	0.79	$((-1,1), 437)$	4	NTRU	x^n-1	KEM, PKE
Falcon	512	12289	$1.17\sqrt{q/2n}$	normal	1	NTRU	x^n+1	SIG
	1024	12289	$1.17\sqrt{q/2n}$	normal	5	NTRU	x^n+1	SIG
SNTRU Prime	653	4621	$\sqrt{2/3}$	$((-1,1), 288)$	1	NTRU	x^n-x-1	KEM
	761	4591	$\sqrt{2/3}$	$((-1,1), 286)$	2	NTRU	x^n-x-1	KEM
	857	5167	$\sqrt{2/3}$	$((-1,1), 322)$	2	NTRU	x^n-x-1	KEM
	953	6343	$\sqrt{2/3}$	$((-1,1), 396)$	3	NTRU	x^n-x-1	KEM
	1013	7177	$\sqrt{2/3}$	$((-1,1), 448)$	4	NTRU	x^n-x-1	KEM
	1277	7879	$\sqrt{2/3}$	$((-1,1), 492)$	5	NTRU	x^n-x-1	KEM

Table 4.13: LWE parameter sets for third round NTRU-based schemes, with dimension n, modulo q, standard deviation of the error σ, and ring $\mathbb{Z}_q[x]/(\phi)$. The NIST column indicates the NIST security category aimed at. Each parameter set from SNTRU Prime has been assigned two security levels in the round 3 submission, and we always choose the lowest of the two.

Name	n	k	q	σ	Secret dist.	NIST	Assumption	ϕ	Primitive
CRYSTALS-Dilithium	1024	4	8380417	$\sqrt{24/12}$	$(-2, 2)$	2	Module-LWE	$x^{n/k}+1$	SIG
	1280	5	8380417	$\sqrt{80/12}$	$(-4, 4)$	3	Module-LWE	$x^{n/k}+1$	SIG
	1792	7	8380417	$\sqrt{24/12}$	$(-2, 2)$	5	Module-LWE	$x^{n/k}+1$	SIG
CRYSTALS-Kyber	512	2	3329	$\sqrt{3/2}$	normal	1	Module-LWE	$x^{n/k}+1$	KEM, PKE
	768	3	3329	1	normal	3	Module-LWE	$x^{n/k}+1$	KEM, PKE
	1024	4	3329	1	normal	5	Module-LWE	$x^{n/k}+1$	KEM, PKE
Frodo	640	—	32768	2.80	normal	1	LWE	—	KEM, PKE
	976	—	65536	2.30	normal	3	LWE	—	KEM, PKE
	1344	—	65536	1.40	normal	5	LWE	—	KEM, PKE
NTRU LPrime	653	—	4621	$\sqrt{2/3}$	$((-1, 1), 252)$	1	Ring-LWR	$x^n - x - 1$	KEM
	761	—	4591	$\sqrt{2/3}$	$((-1, 1), 250)$	2	Ring-LWR	$x^n - x - 1$	KEM
	857	—	5167	$\sqrt{2/3}$	$((-1, 1), 281)$	2	Ring-LWR	$x^n - x - 1$	KEM
	953	—	6343	$\sqrt{2/3}$	$((-1, 1), 345)$	3	Ring-LWR	$x^n - x - 1$	KEM
	1013	—	7177	$\sqrt{2/3}$	$((-1, 1), 392)$	4	Ring-LWR	$x^n - x - 1$	KEM
	1277	—	7879	$\sqrt{2/3}$	$((-1, 1), 429)$	5	Ring-LWR	$x^n - x - 1$	KEM
LightSaber	512	2	8192	$\sqrt{63/12}$	$(-5, 5)$	1	Module-LWR	$x^{n/k}+1$	KEM, PKE
Saber	768	3	8192	$\sqrt{63/12}$	$(-4, 4)$	3	Module-LWR	$x^{n/k}+1$	KEM, PKE
FireSaber	1024	4	8192	$\sqrt{63/12}$	$(-3, 3)$	5	Module-LWR	$x^{n/k}+1$	KEM, PKE

Table 4.14: Parameter sets for third round LWE-based schemes with secret dimension n, Module-LWE rank k (if any), modulo 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. Note that, for the SABER submission, we consider the binomial secret drawn from B_η to be uniform over the interval $(-\frac{\eta}{2}, \frac{\eta}{2})$. Each parameter set from NTRU LPrime has been assigned two security levels in the round 3 submission, and we always choose the lowest of the two.

Name	n	k	q	σ	Secret dist.	NIST	Dual	uSVP
CRYSTALS-Dilithium	1024	4	8380417	$\sqrt{24/12}$	$(-2, 2)$	2	39	124
	1280	5	8380417	$\sqrt{80/12}$	$(-4, 4)$	3	202	183
	1792	7	8380417	$\sqrt{24/12}$	$(-2, 2)$	5	292	252
CRYSTALS-Kyber	512	2	3329	$\sqrt{3/2}$	normal	1	148	119
	768	3	3329	1	normal	3	218	182
	1024	4	3329	1	normal	5	303	255
Frodo	640	—	32768	2.80	normal	1	170	142
	976	—	65536	2.30	normal	3	241	207
	1344	—	65536	1.40	normal	5	314	272
NTRU LPrime	653	—	4621	$\sqrt{2/3}$	$((-1, 1), 252)$	1	152	131
	761	—	4591	$\sqrt{2/3}$	$((-1, 1), 250)$	2	179	156
	857	—	5167	$\sqrt{2/3}$	$((-1, 1), 281)$	2	204	177
	953	—	6343	$\sqrt{2/3}$	$((-1, 1), 345)$	3	227	198
	1013	—	7177	$\sqrt{2/3}$	$((-1, 1), 392)$	4	241	211
	1277	—	7879	$\sqrt{2/3}$	$((-1, 1), 429)$	5	309	271
LightSaber	512	2	8192	$\sqrt{63/12}$	$(-5, 5)$	1	168	125
Saber	768	3	8192	$\sqrt{63/12}$	$(-4, 4)$	3	244	203
FireSaber	1024	4	8192	$\sqrt{63/12}$	$(-3, 3)$	5	321	278
NTRU	509	—	2048	$\sqrt{2/3}$	$((-1, 1), q/8 - 2)$	1	—	108
	677	—	2048	$\sqrt{2/3}$	$((-1, 1), q/8 - 2)$	3	—	149
	821	—	4096	$\sqrt{2/3}$	$((-1, 1), q/8 - 2)$	5	—	180
	701	—	8192	0.79	$((-1, 1), 437)$	4	—	135
Falcon	512	—	12289	$1.17\sqrt{q/2n}$	normal	1	—	141
	1024	—	12289	$1.17\sqrt{q/2n}$	normal	5	—	285
SNTRU Prime	653	—	4621	$\sqrt{2/3}$	$((-1, 1), 288)$	1	—	130
	761	—	4591	$\sqrt{2/3}$	$((-1, 1), 286)$	2	—	155
	857	—	5167	$\sqrt{2/3}$	$((-1, 1), 322)$	2	—	176
	953	—	6343	$\sqrt{2/3}$	$((-1, 1), 396)$	3	—	197
	1013	—	7177	$\sqrt{2/3}$	$((-1, 1), 448)$	4	—	211
	1277	—	7879	$\sqrt{2/3}$	$((-1, 1), 492)$	5	—	272

Table 4.15: "Core-SVP" estimates for third round NTRU-based schemes, with dimension n, modulo q, standard deviation of the error σ, ring $\mathbb{Z}_q[x]/(\phi)$ and with $m = 2n$ samples. The NIST column indicates the NIST security category aimed at.

Name	n	k	q	σ	Secret dist.	NIST	Dual	uSVP
CRYSTALS-Dilithium	1024	4	8380417	$\sqrt{24/12}$	$(-2, 2)$	2	127	113
	1280	5	8380417	$\sqrt{80/12}$	$(-4, 4)$	3	186	166
	1792	7	8380417	$\sqrt{24/12}$	$(-2, 2)$	5	275	229
CRYSTALS-Kyber	512	2	3329	$\sqrt{3/2}$	normal	1	137	108
	768	3	3329	1	normal	3	202	166
	1024	4	3329	1	normal	5	279	232
Frodo	640	—	32768	2.80	normal	1	155	128
	976	—	65536	2.30	normal	3	223	188
	1344	—	65536	1.40	normal	5	285	247
NTRU LPrime	653	—	4621	$\sqrt{2/3}$	$((-1, 1), 252)$	1	142	119
	761	—	4591	$\sqrt{2/3}$	$((-1, 1), 250)$	2	166	141
	857	—	5167	$\sqrt{2/3}$	$((-1, 1), 281)$	2	188	161
	953	—	6343	$\sqrt{2/3}$	$((-1, 1), 345)$	3	208	179
	1013	—	7177	$\sqrt{2/3}$	$((-1, 1), 392)$	4	223	191
	1277	—	7879	$\sqrt{2/3}$	$((-1, 1), 429)$	5	280	246
LightSaber	512	2	8192	$\sqrt{63/12}$	$(-5, 5)$	1	156	113
Saber	768	3	8192	$\sqrt{63/12}$	$(-4, 4)$	3	222	184
FireSaber	1024	4	8192	$\sqrt{63/12}$	$(-3, 3)$	5	300	253
NTRU	509	—	2048	$\sqrt{2/3}$	$((-1, 1), q/8 - 2)$	1	—	98
	677	—	2048	$\sqrt{2/3}$	$((-1, 1), q/8 - 2)$	3	—	136
	821	—	4096	$\sqrt{2/3}$	$((-1, 1), q/8 - 2)$	5	—	163
	701	—	8192	0.79	$((-1, 1), 437)$	4	—	123
Falcon	512	—	12289	$1.17\sqrt{q/2n}$	normal	1	—	128
	1024	—	12289	$1.17\sqrt{q/2n}$	normal	5	—	259
SNTRU Prime	653	—	4621	$\sqrt{2/3}$	$((-1, 1), 288)$	1	—	118
	761	—	4591	$\sqrt{2/3}$	$((-1, 1), 286)$	2	—	140
	857	—	5167	$\sqrt{2/3}$	$((-1, 1), 322)$	2	—	160
	953	—	6343	$\sqrt{2/3}$	$((-1, 1), 396)$	3	—	180
	1013	—	7177	$\sqrt{2/3}$	$((-1, 1), 448)$	4	—	192
	1277	—	7879	$\sqrt{2/3}$	$((-1, 1), 492)$	5	—	247

Table 4.16: Quantum "Core-SVP" estimates for third round NTRU-based schemes, with dimension n, modulo q, standard deviation of the error σ, ring $\mathbb{Z}_q[x]/(\phi)$, and with $m = 2n$ samples. The NIST column indicates the NIST security category aimed ats.

In Tables 4.15 and 4.16 we see the 30 parameter sets analysed using the 0.292β and 0.265β BKZ cost models, under the uSVP attack and the dual attack (where appropriate).

4.9 Conclusion

In this chapter, we have considered the first, second, and third round submissions to the NIST standardisation process. We extracted the parameter sets from all first round submissions, and estimated the security of the uSVP and dual attacks (where appropriate) for each scheme, under every cost model considered as part of a submission. This allows for the security of any two schemes to be compared in a more easy manner. Moreover, we have also shown that cost models for the BKZ algorithm are not *order preserving*, and have provided a summary of the schemes which progressed into the second and third rounds. In the case of the third round schemes, we have provided an updated analysis on the security of these schemes under the Core-SVP model, i.e. $2^{0.292\beta}$ in the classic case, and $2^{0.265\beta}$ in the quantum case.

Chapter 5

Homomorphic Encryption Standardisation

Contents

This chapter is based on the following publication: Benjamin R. Curtis and Rachel Player. On the Feasibility and Impact of Standardising Sparse-secret LWE Parameter Sets for Homomorphic Encryption. In Proceedings of the 7th ACM Workshop on Encrypted Computing and Applied Homomorphic Cryptography (pp. 1-10). Association for Computing Machinery, 2019. Additional details have been added in this thesis.

In this chapter we investigate the security of homomorphic encryption-style LWE parameter sets against hybrid attacks. We consider the effect of secret sparsity on both the performance and security of these schemes.

The author of this thesis contributed towards (a) the writing of the paper, (b) the writing of the code used for experiments, as well as (c) running the experiments and presenting the experimental data.

5.1 Introduction and Contribution

The `homomorphicencryption.org` consortium have begun an effort to standardise both an API [BDH+17] and advice on secure parameter selection for LWE-based homomorphic encryption schemes. The Homomorphic Encryption Security Standard (HE Standard) [ACC+18] recommends parameter sets for usage in homomorphic encryption schemes achieving target security levels $\lambda \in \{128, 192, 256\}$.

Recall from Chapter 2 that LWE instances can always be transformed into *Normal form*, where the secret follows the error distribution [ACPS09]. For error distributions D_σ which satisfy $\sigma = \mathcal{O}(\sqrt{n})$, reductions exist from worst-case hard lattice problems to LWE [Reg05]. In this setting, hardness results for a binary secret $\mathbf{s} \in \{0,1\}^n$ can be obtained at the cost of increasing the LWE dimension [BLP+13]. We note that there are currently no known hardness results for ternary secret LWE, or sparse-secret LWE.

Implementations of LWE-based homomorphic encryption libraries typically choose an error distribution which is much narrower than those considered in security reductions. Indeed, an early example uses $\sigma = 3.19$ [GHS12], which remains a popular choice today. We note that for $\sigma = 3.19$, currently known security reductions do not apply[1]. However, $\sigma = 3.19$ is used for *all* of the the currently recommended parameter sets in the HE Standard [ACC+18].

The HE Standard specifies parameters (n, q, σ) achieving a security level $\lambda \in \{128, 192, 256\}$ according to the LWE Estimator [APS15], described in Chapter 2. The parameters considered are power-of-two ring dimensions $n \in \{1024, 2048, ..., 32768\}$, and a fixed Discrete Gaussian error distribution D_σ with standard deviation $\sigma = 3.19$. For each ring dimension n, a bit-length $\log q$ is standardised. For a given modulus q, the constraint on the error distribution can be equivalently expressed as fixing the parameter $\alpha = \frac{8}{q}$, where α is defined such that $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$, as outlined in Definition 2.30. Typical secret distributions $\chi_{\mathbf{s}}$ used in homomorphic

[1] Further, we note that this is not just the case for homomorphic encryption schemes, and that some of the parameters typically considered in e.g. lattice-based KEMs (see Chapter 4) are not covered by these reductions either.

encryption schemes are:

1. *uniform ternary*, i.e. $\mathbf{s}$ is chosen uniformly at random from the set $\{-1, 0, 1\}^n$,
2. *uniform*, i.e. $\mathbf{s}$ is chosen uniformly at random from the set $\mathbb{Z}_q^n$,
3. *error*, i.e. each coefficient of $\mathbf{s}$ is sampled from the error distribution D_σ, or
4. *uniform binary*, i.e. $\mathbf{s}$ is chosen uniformly at random from the set $\{0, 1\}^n$,

and we note that the HE Standard supports the first three secret distributions. All major implementations of homomorphic encryption use a uniform binary or a ternary secret distribution. Moreover, many implementations use a *sparse* secret, for which all but a certain Hamming weight h of the coefficients are zero. As an example, HEAAN [HEA20], uses by default a sparse ternary secret of Hamming weight $h = 64$. An important issue motivating the use of sparse secrets is the complexity of bootstrapping.

It is worth noting that, in this chapter, we are not concerned with the *depth* of computation, or any potential issues with correctness. Our goal is, for power-of-two ring dimensions n, to provide a maximal permissible modulus $\log q$ which allows a user to attain a desired level of security λ, as in the HE Standard [ACC+18]. Determining the optimal parameters for a specific computation of multiplicative depth L is beyond the scope of this work.

5.1.1 Bootstrapping Complexities for CKKS, BFV, and BGV

We provide a brief overview of the complexity of the bootstrapping procedures for the CKKS [CKKS17], BGV [BGV12], and BFV [FV12] schemes. We have deliberately chosen to omit details on encoding and decoding as these details are not required. Recall that $R_q = \mathbb{Z}_q[X]/(X^n + 1)$. Throughout this section we let χ be some distribution which samples small polynomials from the ring R_q. We also set $R_3 = \mathbb{Z}_3[X]/(X^n + 1)$ and we recall here that $\mathbb{Z}_3 = \{-1, 0, 1\}$ and, therefore, the co-efficient vector $\mathbf{g}$ of non-zero polynomials $g \leftarrow R_3$ satisfies $\|\mathbf{g}\|_\infty = 1$.

5.1.1.1 CKKS

We briefly outline the CKKS homomorphic encryption scheme as presented in [CKKS17]. Consider values $p, q_0 > 0$,$q_\ell = p^\ell q_0$ for $0 < \ell \leq L$ for some $L \in \mathbb{N}$, and set $P \approx q_L$.

- $\mathsf{KGen}(1^n)$. The key generation algorithm takes as input the security parameter and generates three keys: a secret key sk, a public encryption key pk, and a public evaluation key ek.
  - To generate the secret key, we sample a small polynomial $s \leftarrow R_3$ and set:
$$\mathsf{sk} = (1, s) \in R_{q_L}^2.$$
  - To generate the public key, we sample $a \leftarrow R_{q_L}, e \leftarrow \chi$ and set:
$$\mathsf{pk} = (-(as + e) \bmod q_L, a) \in R_{q_L}^2.$$
  - To generate the evaluation key we sample $a' \leftarrow R_{Pq_L}, e' \leftarrow \chi$ and set:
$$\mathsf{ek} = (-a's + e + Ps^2 \bmod Pq_L, a') \in R_{Pq_L}^2.$$
- $\mathsf{Enc}(\mathsf{pk}, m)$. The encryption algorithm takes as input a public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) \in R_{q_L}^2$ and a message $m \in R$, samples $v \leftarrow R_3$ and $e_1, e_2 \leftarrow \chi$, and outputs:
$$c = v\mathsf{pk} + (m + e_1, e_2) \in R_{q_L}^2.$$
- $\mathsf{Dec}(\mathsf{sk}, c)$. The decryption algorithm takes as input a secret key $\mathsf{sk} = s \in R_3$ and a ciphertext $c = (c_1, c_2) \in R_{q_L}^2$, and outputs:
$$m' \leftarrow (c_1 + c_2 s) \bmod q_L.$$
- $\mathsf{Add}(c, c')$. The addition algorithm takes as input two ciphertexts $c, c' \in R_{q_\ell}^2$ and outputs:
$$c_+ = (c_1 + c_1', c_2 + c_2') \bmod q_\ell.$$
- $\mathsf{Mult}(c, c')$. The multiplication algorithm takes as input two ciphertexts $c = (c_1, c_2), c' = (c_1', c_2') \in R_{q_\ell}^2$ and computes the values $d_1 = c_1 c_1'$, $d_2 = c_1 c_2' + c_2 c_1'$, and $d_3 = c_2 c_2'$. The algorithm outputs:
$$c_\times = (d_1, d_2) + \lfloor P^{-1} \cdot d_3 \cdot \mathsf{ek} \rceil \bmod q_\ell.$$

The decryption function in the CKKS scheme is a modular reduction, i.e. evaluating:

$$m' \leftarrow \langle ct, sk \rangle \bmod q_L.$$

In order to bootstrap we need to homomorphically evaluate the decryption function, we note that this decryption function can be represented as a scaled sine function [CHK+18]:

$$\langle ct, sk \rangle \bmod q_L = \frac{q}{2\pi} \cdot \sin\left(\frac{2\pi}{q_L} \cdot \langle ct, sk \rangle\right) + O\left(\epsilon^3 \cdot q_L\right),$$

when $\|[\langle ct, sk \rangle]_{q_L}\| \leq \epsilon \cdot q_L$. Bootstrapping can therefore be implemented by evaluating a Chebyshev interpolant (polynomial) in degree $d = \mathcal{O}(K + \log q)$, where q is the ciphertext modulus, and K is a constant depending on the secret distribution. This evaluation requires $\mathcal{O}(\sqrt{d})$ ciphertext multiplications [CCS19a]. For sparse secrets with Hamming weight h, the heuristic argument of [CHK+18] shows that we have $K = \mathcal{O}(\sqrt{h})$, while for a uniform ternary secret we have $K = \mathcal{O}(\sqrt{n})$. We note that a passive attack against CKKS, as well as potential countermeasures, has been outlined in [LM20].

5.1.1.2 BGV

We briefly outline the BGV homomorphic encryption scheme as presented in [CLP19]. Here, the ciphertext space is R_q and the plaintext space is R_t. We note that w is a base, and ℓ denotes the number of terms in the decomposition of an integer modulo q into base w, i.e. $w = \lfloor \log_w(q) \rfloor$.

- $\mathsf{KGen}(1^n)$. The key generation algorithm takes as input the security parameter and generates three keys: a secret key sk, a public encryption key pk, and a public evaluation key ek.
  - To generate the secret key, we sample a small polynomial $s \leftarrow R_3$ and set:
    $$\mathsf{sk} = s.$$
  - To generate the public key, we sample $a \leftarrow R_q, e \leftarrow \chi$ and set:
    $$\mathsf{pk} = (-(as + te) \bmod q, a).$$
  - To generate the evaluation key, for $1 \leq i \leq \ell$ we sample $a_i \leftarrow R_q$ and $e_i \leftarrow \chi$, and set:
    $$\mathsf{ek} = \left\{\left(-(a_i s + te_i) + w^i s^2 \bmod q, a_i\right)\right\}_{i=1}^{\ell}.$$

- $\mathsf{Enc}(\mathsf{pk}, m)$. The encryption algorithm takes as input a public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) \in R_q^2$ and a message $m \in R_t$, samples $v \leftarrow R_3$ and $e_1, e_2 \leftarrow \chi$, and outputs a ciphertext of the form:

$$c = (\mathsf{pk}_1 v + te_1 + m, \mathsf{pk}_2 v + te_2) \in R_q^2.$$

- $\mathsf{Dec}(\mathsf{sk}, c)$. The decryption algorithm takes as input a secret key $\mathsf{sk} = s \in R_q$ and aciphertext $c = (c_1, c_2) \in R_q^2$, and outputs:

$$m = (c_1 + c_2 s \bmod q) \bmod t.$$

- $\mathsf{Add}(c, c')$. The addition algorithm takes as input two ciphertexts $c, c' \in R_q^2$ and outputs:

$$c_+ = (c_1 + c_1' \bmod q, c_2 + c_2' \bmod q).$$

- $\mathsf{Mult}(c, c')$. The multiplication algorithm takes as input two ciphertexts $c, c' \in R_q^2$ and computes the values $d_1 = c_1 c_1' \bmod q$, $d_2 = c_1 c_2' + c_2 c_1' \bmod q$, and $d_3 = c_2 c_2' \bmod q$. From there, we output $c_\times = (d_1, d_2, d_3)$, that is:

$$c_\times = (c_1 c_1' \bmod q, c_1 c_2' + c_2 c_1' \bmod q, c_2 c_2' \bmod q).$$

- $\mathsf{Relinearize}(\mathsf{ek}, c)$. The relinearisation algorithm takes as input an evaluation key $\mathsf{ek} = \{(\mathsf{ek}_{i,1}, \mathsf{ek}_{i,2})\}_{i=1}^{\ell}$ and a ciphertext $c = (c_1, c_2, c_3)$, expresses c_3 in base w, i.e. computes $c_3 = \sum_{i=1}^{\ell} c_3^{(i)} w^i$. Computes $d_1 = c_1 + \sum_{i=1}^{\ell} \mathsf{ek}_{i,1} c_3^{(i)}$ and $d_2 = c_2 + \sum_{i=1}^{\ell} \mathsf{ek}_{i,2} c_2^{(i)}$, and outputs $c_{\mathsf{relin}} = (d_1, d_2)$, that is:

$$c_{\mathsf{relin}} = (c_1 + \sum_{i=1}^{\ell} \mathsf{ek}_{i,1} c_3^{(i)}, c_2 + \sum_{i=0}^{\ell} \mathsf{ek}_{i,2} c_3^{(i)}).$$

The decryption function in the BGV scheme is:

$$m' \leftarrow (c_1 + c_2 s \bmod q) \bmod t,$$

and bootstrapping therefore requires the evaluation of a circuit of depth $\log(\|s\|_1) + \log t$ [CH18], where t is the plaintext modulus. This evaluation requires $\mathcal{O}(\log^{3/2} \|s\|_1 + \log^{1/2} \|s\|_1 \cdot \log t + \log^2 t)$ ciphertext multiplications [CH18, Table 2].

5.1.1.3 BFV

We briefly outline the BFV homomorphic encryption scheme as presented in [CLP19]. Here, the ciphertext space is R_q and the plaintext space is R_t. We also define the quantity $\Delta = \lfloor q/t \rfloor$. We note that w is a base, and ℓ denotes the number of terms in the decomposition of an integer modulo q into base w, i.e. $w = \lfloor \log_w(q) \rfloor$.

- $\mathsf{KGen}(1^n)$. The key generation algorithm takes as input the security parameter and generates three keys: a secret key sk, a public encryption key pk, and a public evaluation key ek.
  - To generate the secret key, we sample a small polynomial $s \leftarrow R_3$ and set:
  $$\mathsf{sk} = s.$$
  - To generate the public key, we sample $a \leftarrow R_q, e \leftarrow \chi$ and set:
  $$\mathsf{pk} = (-(as + e) \bmod q, a).$$
  - To generate the evaluation key, for $1 \leq i \leq \ell$ we sample $a_i \leftarrow R_q$ and $e_i \leftarrow \chi$, and set:
  $$\mathsf{ek} = \left\{ \left(-(a_i s + e_i) + w^i s^2 \bmod q, a_i \right) \right\}_{i=1}^{\ell}.$$
- $\mathsf{Enc}(\mathsf{pk}, m)$. The encryption algorithm takes as input a public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) \in R_q^2$ and a message $m \in R_t$, samples $v \leftarrow R_3$ and $e_1, e_2 \leftarrow \chi$, and outputs a ciphertext of the form: and outputs a ciphertext of the form:
$$c = (\mathsf{pk}_1 v + e_1 + \Delta m, \mathsf{pk}_2 v + e_2) \in R_q^2.$$
- $\mathsf{Dec}(\mathsf{sk}, c)$. The decryption algorithm takes as input a secret key $\mathsf{sk} = s \in R_3$ and a ciphertext $c = (c_1, c_2) \in R_q^2$, and outputs:
$$m' = \left\lfloor \frac{t}{q}(c_1 + c_2 s) \bmod q \right\rceil \bmod t.$$
- $\mathsf{Add}(c, c')$. The addition algorithm takes as input two ciphertexts $c, c' \in R_q^2$ and outputs:
$$c_+ = (c_1 + c_1' \bmod q, c_2 + c_2' \bmod q).$$

- $\mathsf{Mult}(c, c')$. The multiplication algorithm takes as input two ciphertexts $c, c' \in R_q^2$ and computes the values $d_1 = \left\lfloor \frac{t}{q} c_1 c_1' \right\rceil \bmod q$, $d_2 = \left\lfloor \frac{t}{q}(c_1 c_2' + c_2 c_1') \right\rceil \bmod q$, and $d_3 = \left\lfloor \frac{t}{q} c_2 c_2' \right\rceil \bmod q$. From there, we output $c_\times = (d_1, d_2, d_3)$, that is:

$$c_\times = \left(\left\lfloor \frac{t}{q} c_1 c_1' \right\rceil \bmod q, \left\lfloor \frac{t}{q}(c_1 c_2' + c_2 c_1') \right\rceil \bmod q, \left\lfloor \frac{t}{q} c_2 c_2' \right\rceil \bmod q \right).$$

- $\mathsf{Relinearize}(\mathsf{ek}, c)$. The relinearisation algorithm takes as input an evaluation key $\mathsf{ek} = \{(\mathsf{ek}_{i,1}, \mathsf{ek}_{i,2})\}_{i=1}^{\ell}$ and a ciphertext $c = (c_1, c_2, c_3)$, expresses c_3 in base w, i.e. computes $c_3 = \sum_{i=1}^{\ell} c_3^{(i)} w^i$. Computes $d_1 = c_1 + \sum_{i=0}^{\ell} \mathsf{ek}_{i,1} c_3^{(i)} \bmod q$ and $d_2 = c_2 + \sum_{i=1}^{\ell} \mathsf{ek}_{i,2} c_3^{(i)} \bmod q$, and outputs $c_{\mathsf{relin}} = (d_1, d_2)$, that is:

$$c_{\mathsf{relin}} = (c_1 + \sum_{i=1}^{\ell} \mathsf{ek}_{i,1} c_3^{(i)} \bmod q, c_2 + \sum_{i=1}^{\ell} \mathsf{ek}_{i,2} c_3^{(i)}) \bmod q.$$

The decryption function in the BFV scheme can be written as:

$$m' \leftarrow \left\lfloor \frac{(c_1 + c_2 \cdot s \bmod q)}{\Delta} \right\rceil,$$

which requires the evaluation of a circuit of depth $\log(\|s\|_1) + \log\log t$. This, in turn, requires $\mathcal{O}((\log \|s\|_1 + \log t)^{1/2} \log \|s\|_1)$ ciphertext multiplications [CH18, Table 2].

5.1.1.4 Summary

For sparse ternary secret with Hamming weight h, we have $\|s\|_1 = h$, whereas for a uniform ternary secret we expect $\|s\|_1 = \mathcal{O}(n)$. Current implementations for bootstrapping in CKKS, BGV or BFV use sparse secrets for efficiency reasons [CH18, CHK+18]. We also note that there are works aimed towards improving bootstrapping techniques for non-sparse keys [BMTPH20].

We summarise this discussion in Table 5.1.

5.1.1.5 Sparse Secrets

Sparse secret distributions are not currently supported in the HE Standard. This is likely due to the loss of security as compared to a uniform ternary secret for a fixed set of LWE parameters (n, q, σ), as well as uncertainty over the attack landscape for these distributions.

Scheme	CKKS	BGV	BFV
Decryption circuit depth	$\mathcal{O}(K + \log q)$	$\log(\|s\|_1) + \log t$	$\log(\|s\|_1) + \log\log t$
Ciphertext multiplications	$\mathcal{O}(\sqrt{d})$	$\mathcal{O}(\log^{3/2}\|s\|_1 + \log^{1/2}\|s\|_1 \cdot \log t + \log^2 t)$	$\mathcal{O}((\log\|s\|_1 + \log t)^{1/2}\log\|s\|_1)$
Parameter sizes when $\mathbf{s} \leftarrow \mathcal{B}^-$	$K = \mathcal{O}(\sqrt{n})$	$\|s\|_1 = \mathcal{O}(n)$	$\|s\|_1 = \mathcal{O}(n)$
Parameter sizes when $\mathbf{s} \leftarrow \mathcal{B}^-_h$	$K = \mathcal{O}(\sqrt{h})$	$\|s\|_1 = h$	$\|s\|_1 = h$

Table 5.1: A summary of the bootstrapping complexities for CKKS, BGV and BFV.

The loss of security is intuitive, as this corresponds to shrinking the size of the keyspace. Moreover, several attacks are known which can exploit the sparsity of an LWE secret [How07, Alb17, HHC19]. As discussed in Chapters 3 and 4, these attack techniques trade-off the probability of guessing τ secret components against the lower cost of solving a lattice problem in dimension $(d - \tau)$.

Another way in which the recommended parameter sets in the HE Standard do not always reflect implementation choices is in the maximal supported dimension $n = 2^{15}$. For example, many implementations of bootstrapping, such as [CHK$^+$18, CCS19a, HHC19], choose ring dimension $n = 2^{16}$. In addition, advanced applications of homomorphic encryption, such as logistic regression training [KSK$^+$18, KSW$^+$18], have been reported using dimension $n = 2^{16}$ or $n = 2^{17}$. Such large dimensions are used to support the choice of q, which is chosen to be large enough to allow for evaluation of a circuit of a specific depth.

5.1.2 Structure and Contributions

The discussion from Section 5.1.1.5 motivates the widening of the recommended parameter sets to include sparse secrets, or parameter sets for larger dimension $n > 2^{15}$. In this chapter we consider such possible extensions.

An outline of the structure and contributions of this chapter is as follows. In Section 5.2 we make some comments on small and sparse secret LWE. In Section 5.3 we discuss algorithms used to solve small and sparse LWE, including introducing the hybrid dual attack [CHHS19]. In Section 5.4 we outline some of the parameter sets recommended in the current variant of the HE Standard. In Section 5.5 we assess the impact on security and performance of using a sparse secret of Hamming weight h instead of a uniform ternary secret, for various choices of h. In Section 5.5.4 we show how the methodology of the Standard could be used

to select parameters with larger power-of-two dimension $n \geq 2^{16}$. In Section 5.6 we discuss open problems and future work.

5.2 Comments on Small and Sparse-secret LWE

Definition 2.34 outlines some small-secret distributions of interest. Recall that the uniform ternary distribution is denoted by $\mathcal{B}^-$, and the uniform binary distribution is denoted by $\mathcal{B}^+$.

5.2.1 Keyspace Size

When considering combinatorial attacks against LWE, the size of the keyspace is an important quantity, since this determines the maximal size of the guessing set in combinatorial attacks. To illustrate this, we consider the following examples:

1. when the secret is drawn from $\mathbb{Z}_q^n$, the size of the keyspace is $\|\mathcal{S}_{\mathbb{Z}_q^n}\| = q^n$,
2. when the secret is drawn from $\mathcal{B}^-$, the size of the keyspace is $\|\mathcal{S}_{\mathcal{B}^-}\| = 3^n$,
3. when the secret is drawn from $\mathcal{B}_h^-$, the size of the keyspace is $\|\mathcal{S}_{\mathcal{B}_h^-}\| = \binom{n}{h} \cdot 2^h$,
4. when the secret is drawn from $\mathcal{B}^+$, the size of the keyspace is $\|\mathcal{S}_{\mathcal{B}^+}\| = 2^n$, and
5. when the secret is drawn from $\mathcal{B}_h^+$, the size of the keyspace is $\|\mathcal{S}_{\mathcal{B}^+}\| = \binom{n}{h}$.

In Figure 5.1 we highlight the size of the keyspace $\mathcal{B}_h^-$, and $\mathcal{B}_h^+$, for each potential value of h when the associated ring dimension is $n = 1024$. If $h = 64$ then the size of the keyspace is $\approx 2^{405}$, whereas if the secret is is drawn from $\mathcal{B}^-$ then the size of the keyspace is $\approx 2^{1623}$. We note that the LWE Estimator, as well as our scripts for hybrid attacks, assume that uniformly random ternary secrets have fixed Hamming weight $h = \lfloor \frac{2n}{3} \rfloor$, and that uniformly random binary secrets have fixed Hamming weight $h = \lfloor \frac{n}{2} \rceil$. For the examples considered in Figure 5.1, we have that $h = \lfloor \frac{2 \times 1024}{3} \rceil = 683$ and $\|\mathcal{S}_{\mathcal{B}_{683}^-}\| = \binom{1024}{683} \cdot 2^{683} \approx 2^{1618}$, and $h = \lfloor \frac{1024}{2} \rceil = 512$ and $\|\mathcal{S}_{\mathcal{B}_{512}^+}\| = \binom{1024}{512} \approx 2^{1018}$.

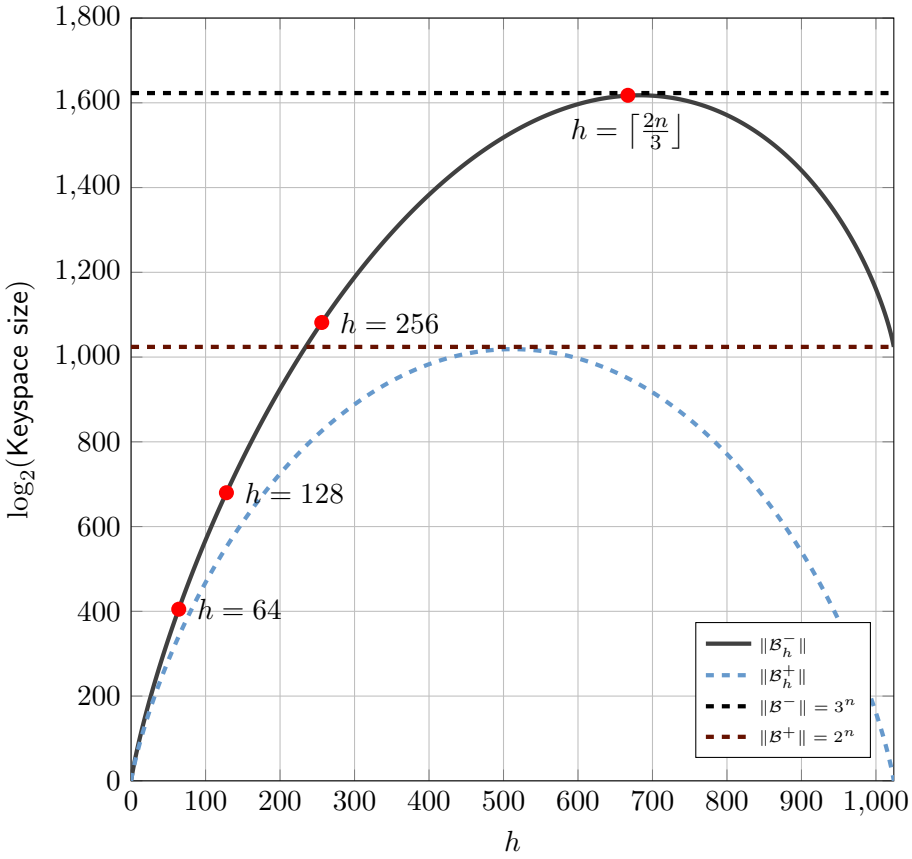


Figure 5.1: Example LWE (secret) keyspace sizes with $n = 1024$ for binary, ternary, fixed-weight binary, and fixed-weight ternary secrets.

5.2.2 Secret Density

Keeping a fixed Hamming weight (e.g. $h = 64$) for a variety of ring dimensions means that the *density* $\kappa = \frac{h}{n}$ of the secret decreases as n grows. One approach to scaling sparse secrets is to fix the *density parameter*:

$$\kappa = \frac{h}{n}.$$

For example, we could consider $\kappa = \frac{1}{16}$ such that:

$$(n, h) \in \{(1024, 64), (2048, 128), (4096, 256), \dots\}.$$

This follows the approach of several submissions to the ongoing NIST post-quantum standardisation effort: for example, Lizard [CKLS18] uses $h = \frac{n}{8}$, i.e. $\kappa = \frac{1}{8}$. For larger ring dimensions used in homomorphic encryption libraries this approach can lead to a large Hamming weight h,

leading to a more expensive bootstrapping operation. For example, for a ring dimension $n = 32768$, choosing $\kappa = \frac{1}{16}$ would require $h = 2048$.

Another approach is to fix the ratio between the Hamming weight h of the secret and the security parameter λ, i.e. fix the value ζ, where:

$$\zeta = \frac{h}{\lambda}.$$

This approach would mean that, for each target security level $\lambda_{\mathsf{target}}$, the value of the Hamming weight h for *every* ring dimension $n = 2^k$ is fixed. For example, if $\zeta = 1$, then for a fixed security level λ we consider secrets of Hamming weight $h = \lambda$. Such an approach means that the (theoretical) complexity of bootstrapping would remain the same for each dimension n with an associated security level λ. In this work we consider the *second* approach, i.e. fixing the value of $\zeta = \frac{h}{\lambda}$.

5.3 Algorithms for solving Small-secret LWE

As discussed in Chapter 2, the concrete security of LWE-based parameter sets is typically determined by considering the best known attacks. That is, given an LWE parameter set (n, α, q) and a corresponding secret distribution, we set λ to be the logarithm of the running time of the fastest attack. The current version of the HE Standard [ACC+18] uses the LWE Estimator to determine parameters, based on the running time of three attacks: uSVP, decoding and dual. Hybrid attacks [How07, CHHS19] are typically among the most competitive in the case of sparse secrets, although they are not currently supported by the LWE Estimator, and therefore have not been considered in the latest version of the HE Standard.

Following the HE Standard, in this chapter we only consider *sieving*-based cost models for BKZ. Specifically, we view BKZ as a black box which runs in (pre-quantum) time:

$$T_{\mathsf{BKZ}}(\beta, d) = 2^{0.292\beta + 16.4 + \log(8d)},$$

and, if instantiated with quantum algorithms to solve SVP, runs in time:

$$T_{\mathsf{BKZ}}(\beta, d) = 2^{0.265\beta + 16.4 + \log(8d)}.$$

Estimates for the primal decoding attack [LP11, LN13] reported by the LWE Estimator do not assume state-of-the-art techniques, hence may be inaccurate and are often not competitive.

More precisely, the Estimator currently assumes the decoding attack is implemented with the Nearest Planes algorithm [LP11] as opposed to the more efficient pruned enumeration [LN13]. Moreover, the Estimator does not consider combinatorial techniques for the decoding attack as considered in [ACW20] and Chapter 3. As an example, for the example parameter set $n = 653, q = 4621, \sigma \approx \sqrt{2/3}, \chi_s = \mathcal{B}_{100}^{-}$ considered in Section 3.4, the decoding estimate provided by the LWE Estimator estimates the complexity as $2^{229.9}$, whereas the batch-BDD techniques outlined in Chapter 3 report a complexity of $2^{186.1}$. For this reason, we do not report dec estimates in this chapter.

We consider the uSVP, hybrid-decoding, dual and hybrid-dual attacks on LWE. For uSVP and dual, we consider the small-secret variant of these attacks as described in Chapter 4. For the hybrid-decoding attack, as discussed in Chapter 3, we outline the assumptions considered in this Chapter in Section 5.3.1. We describe the hybrid-dual attack in Section 5.3.2.

5.3.1 Hybrid-decoding Attack Assumptions

As discussed in Chapter 3, an analysis of the hybrid-decoding attack requires the usage of several assumptions. In Chapter 3 we considered multiple sets of assumptions. For clarity, we outline the assumptions considered in this chapter:

- The output GSO basis shape of lattice reduction is given by the Geometric Series Assumption [Sch03].
- The (heuristic) success probability of Babai's Nearest Plane algorithm follows the analysis in [Wun18] and is given by:

$$p \approx \prod_{1 \leq i \leq d} \left(1 - \frac{2}{B(\frac{d-1}{2}, \frac{1}{2})} \cdot \int_{\min(r_i,1)}^{1} (1-t^2)^{(d-3)/2}\, dt\right),$$

  where d is the dimension of the lattice under consideration, and $r_i = \|\mathbf{b}_i^*\|/2\|\mathbf{v}\|$ where $\|\mathbf{v}\|$ is the (expected) norm of the target vector, and $B(\cdot,\cdot)$ denotes the Beta function [OLBC10].
- The cost of running Babai's Nearest Plane algorithm in a lattice of dimension d is given by $T_{\mathsf{babai}} = \frac{d^2}{2^{1.06}}$.
- The meet-in-the-middle search phase provides a square-root speed-up as compared to an exhaustive search.

- The associated meet-in-the-middle probability is set to be $p_{\mathsf{mitm}} = 1$, thus providing an explicit *underestimate* of security.
- The meet-in-the-middle search phase has access to unlimited memory.

Under these assumptions, the cost of the hybrid-decoding attack is given by:

$$\min_{\beta,\tau,d,t} \frac{T_{\mathsf{BKZ}}(\beta, d) + \frac{d^2}{2^{1.06}} \cdot \sqrt{\sum_{i=0}^{t} \|\mathcal{S}_i\|}}{p \cdot \sum_{i=0}^{t} p_i},$$

where β is the BKZ blocksize, d is the dimension in which lattice reduction is performed, τ is the guessing dimension (i.e. the number of guessed components of the secret), t is the maximal Hamming weight considered in the search space, and $\sum_{i=0}^{t} \|\mathcal{S}_i\|$ is the size of the search space, i.e. the number of points on which we decode. Here p is the probability of success of Babai's Nearest Plane algorithm, and p_i is the probability that the guessed part of the secret has Hamming weight i. Estimates for the cost of the hybrid-decoding attack are generated using custom code[2].

5.3.2 The Hybrid-dual Attack

Albrecht's variant of the dual attack, as described in Chapter 4, was recently adapted by Cheon *et al.* [CHHS19] to include a meet-in-the-middle step in the combinatorial search phase of the attack, giving rise to a *hybrid-dual attack*. It is shown in [CHHS19] that when fixing a maximal memory of 2^{80}, the hybrid dual attack outperforms the dual attack for certain homomorphic encryption-style parameter sets with a sparse ternary secret. Cheon *et al.* [CHHS19] provide a script[3] that can be used to estimate the security of a given parameter set against the hybrid dual attack. Recall from Chapter 4 that the small-secret variant of the dual attack proceeds by performing lattice reduction on the lattice Λ'', where:

$$\Lambda'' = \left\{ (\mathbf{v}, \mathbf{w}) \in \mathbb{Z}^m \times \left(\frac{1}{c} \cdot \mathbb{Z}\right)^n \middle| \mathbf{v}\mathbf{A} \equiv c\mathbf{w} \bmod q \right\}.$$

For an LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e})$, with sparse secret of Hamming weight h, the crux of the meet-in-the-middle process is based on considering the noisy relationship:

$$\mathbf{A}\mathbf{s}_1 \approx \mathbf{b} - \mathbf{A}\mathbf{s}_2$$

[2]This code can be found at `https://github.com/bencrts/hybrid_attacks/hybrid_decoding.py`
[3]The script is available at `https://github.com/swanhong/HybridLWEAttack`.

for a pair $\mathbf{s}_1, \mathbf{s}_2$ satisfying $\mathbf{s}_1 + \mathbf{s}_2 = \mathbf{s}$. The attack begins by considering the list which constructs $\mathbf{A}\mathbf{v}_1$ for candidate values $\mathbf{v}_1$ of $\mathbf{s}_1$:

$$\mathcal{T} = \{\mathbf{A}\mathbf{v}_1 \mid \mathbf{v}_1 \in \{-1, 0, 1\}^n : \mathcal{HW}(\mathbf{v}_1) \leq h'\},$$

which is constructed for some $h' \leq h$. Note that h' is an attack parameter and can be optimised over. A hash table $\mathcal{H}$, initialised as 2^m empty linked lists with indexes in $\{0,1\}^m$, is generated from this list by appending an entry $\mathbf{t} \in \mathcal{T}$ into the position indexed with $\mathsf{sgn}(\mathbf{t})$ where:

$$\mathsf{sgn}(\mathbf{t})_i = \begin{cases} 1 & \text{if } t_i \in [0, q/2) \\ 0 & else \end{cases}.$$

After this process is completed for all $t \in \mathcal{T}$, we begin a search over $\mathcal{H}$. The search proceeds by checking whether $\mathbf{b} - \mathbf{A}\mathbf{v}_2$ is close to the list $\mathcal{T}$ for candidate choices $\mathbf{v}_2$, where the closeness depends on the size of the error vector $\mathbf{e}$. That is, we are interested in the norm of:

$$\|\mathbf{A}\mathbf{s}_1 - (\mathbf{b} - \mathbf{A}\mathbf{s}_2)\|_\infty = \|\mathbf{e}\|_\infty,$$

and the meet-in-the-middle process is successful if $\|\mathbf{e}\|_\infty < B$, for some B. This is referred to as a *B-noisy collision.*

Definition 5.1 (B-noisy collision [CHHS19]) *For a vector $\mathbf{a} \in \mathbb{Z}_q^m$, a vector $\mathbf{t} \in \mathbb{Z}_q^m$ is referred to as a B-noisy collision of $\mathbf{a}$ if $\|\mathbf{a} - \mathbf{t}\|_\infty \leq B$ for some $B < \frac{q}{2}$.*

In the analysis of [CHHS19], the parameter B is chosen as:

$$B = \left(2 + \frac{1}{\sqrt{2\pi}}\sqrt{\frac{m}{m+n}} \cdot \alpha q \|\mathbf{y}\|\right),$$

where $\mathbf{y}$ is the short vector retrieved via lattice reduction on the lattice Λ''. If we can find a B-noisy collision of $\mathbf{A}\mathbf{s}_1$, for a candidate $\mathbf{s}_1$, then we can find $\mathbf{b} - \mathbf{A}\mathbf{s}_2$ and therefore solve decisional LWE.

The search phase takes as input the constructed hash table $\mathcal{H}$, a query point $\mathbf{q}$ (which corresponds to $\mathbf{b} - \mathbf{A}\mathbf{v}_2$ for some candidate $\mathbf{v}_2$ of $\mathbf{s}_2$) as well as a distance bound B. Defining:

$$\mathsf{sgn}'(\mathbf{t})_i = \begin{cases} \mathsf{sgn}(t_i) & \text{if } t_i \in V_B \\ \times & else \end{cases},$$

where $V_B = [-q/2 + B, -B) \cup [B, q/2 - B)$, we then consider:

$$\mathsf{sgn}'(\mathbf{q}),$$

where $\mathbf{q}$ is a candidate for $\mathbf{s}_2$. $\mathsf{sgn}'(\mathbf{q})$ represents a *list* of binary strings generated by replacing each $\times$ by 0 or 1. Then, for each element $\mathbf{x}$ in this list, we check if $\mathcal{H}$ contains a set indexed by $\mathbf{x}$. If this is the case, then for every $\mathbf{t}$ within this list we check if:

$$\|\mathbf{q} - \mathbf{t}\|_\infty < B,$$

and, if this is the case, we return this vector.

To summarise, the hybrid-dual attack consists of three phases:

1. a *lattice reduction phase*, where we perform lattice reduction on the lattice Λ'',
2. a *hash-table construction phase*, where we generate the hash table $\mathcal{H}$, and
3. a *searching phase*, where we search for B-noisy collisions using the hash table $\mathcal{H}$.

The total cost of phases 2. and 3. is the sum of:

(a) constructing the list $\mathcal{T}$,

(b) generating the hash table $\mathcal{H}$, and

(c) the cost of the search algorithm applied to the candidate secrets.

These costs are computed as $N_{\mathcal{T}}(n^2 + m)$ for steps (a) and (b) and $\mathcal{O}(N_q 2^{4mB/q})$ for step (c) [CHHS19].

5.3.2.1 Hybrid-dual Assumptions

In this thesis, we do not use the script provided by [CHHS19].[4] Instead, we *conservatively* assume that the meet-in-the-middle process admits a square-root speed-up with no probability loss – thus providing an explicit *underestimate* of security, for the purposes of generating conservative parameters. This matches our approach with the hybrid-decoding attack (i.e. $p_{\mathsf{mitm}} = 1$). To do this, we consider Albrecht's attack script [Est20] and, in the search phase, balance the cost of searching with the *square* of the lattice reduction cost (as opposed to simply balancing the cost of searching with the cost of lattice reduction). The changes to the source code can be found in Figure 5.2.

[4] This is because the script was not designed to be used for non-sparse (i.e. uniform-ternary) secrets.

```
if postprocess:
    repeat = current["repeat"]
    dim    = current["d"]

    for i in range(1, k):
        cost_post_i = (2 * repeat * dim * k) + (repeat * binomial(k, i) * (b-a)**i  * i)
        probability_i = success_probability_drop(n, h, k, i, rotations=rotations)
        if cost_post + cost_post_i >= cost_lat**2:
            postprocess = i
            break

cost_post += cost_post_i
probability += probability_i

current["rop"] = cost_lat + sqrt(cost_post)
```

Figure 5.2: Changes applied to the dual attack source code inside the LWE Estimator to produce our hybrid-dual estimates. This replaces lines 1944-1959 in commit `428d6ea` of the LWE Estimator.

5.4 Currently Recommended Parameters

In Tables 5.2 we reproduce a subset of the parameter sets $(n, \log q, \alpha)$ recommended in the current version of the HE Standard [ACC+18] to achieve target security level λ for $\lambda \in \{128, 192, 256\}$, for power-of-two ring dimensions between 1024 and 32768 and for a secret having coefficients chosen uniformly in $\{-1, 0, 1\}$. Table 5.2 reports the estimated cost of running the uSVP, decoding, and dual attacks on these parameter sets under the sieving cost model:

$$T(\beta, d) = 2^{0.292\beta + 16.4 + \log(8d)}.$$

n	$\log q$	α	usvp	decoding	dual	$\lambda_{\mathsf{target}}$
1024	27	8/q	**131.6**	160.2	138.7	128
2048	54	8/q	**129.7**	144.4	134.2	
4096	109	8/q	**128.1**	134.9	129.9	
8192	218	8/q	**128.5**	131.5	129.2	
16384	438	8/q	**128.1**	129.9	129.0	
32768	881	8/q	**128.5**	129.1	**128.5**	
1024	19	8/q	**193.0**	259.5	207.7	192
2048	37	8/q	**197.5**	233.0	207.8	
4096	75	8/q	**194.7**	212.2	198.5	
8192	152	8/q	**192.2**	200.4	194.6	
16384	305	8/q	**192.1**	196.2	193.2	
32768	611	8/q	**192.7**	194.2	193.7	
1024	14	8/q	**265.6**	406.4	293.8	256
2048	29	8/q	**259.1**	321.7	273.5	
4096	58	8/q	**260.4**	292.6	270.1	
8192	118	8/q	**256.7**	270.4	260.6	
16384	237	8/q	**256.9**	264.2	259.8	
32768	476	8/q	**256.4**	260.2	258.2	

Table 5.2: Currently standardised LWE parameters at the 128-, 192- and 256-bit security level for uniform ternary secret specified in [ACC+18, Table 1] and estimates of their security against usvp, decoding, and dual attacks under the BKZ cost model $T(\beta, d) = 2^{0.292\beta+16.4+\log(8d)}$, where β is the blocksize and d is the dimension. The best performing attack for each parameter set is highlighted in bold.

5.5 Investigating Sparse-secrets

It may be desirable to extend the HE Standard to include parameter sets with sparse secret distributions, due to the complexity of bootstrapping outlined in Section 5.1.1. In this section, we consider the feasibility and impact of including sparse ternary secret distributions in the HE Standard. To begin, in Figure 5.3 we compare the four attack techniques discussed in

this chapter for the parameter set $n = 1024, q = 2^{40}$ and $\sigma \approx 3.2$ with a sparse ternary secret of Hamming weight $h \in \{64, 128, 256, 512\}$, to see which attacks perform best as the density of the secret changes.

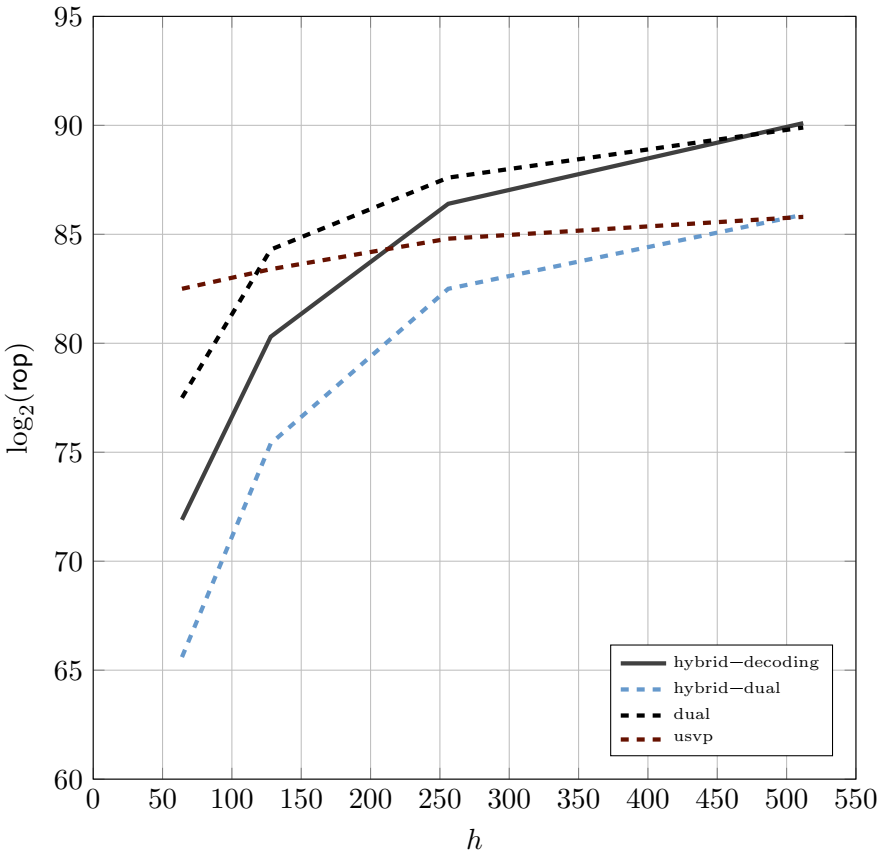


Figure 5.3: A comparison of the usvp, dual, hybrid−dual and hybrid−decoding attacks under the BKZ cost model $T(\beta, d) = 2^{0.292\beta + 16.4 + \log(8d)}$, for the parameter set $n = 1024, q = 2^{40}$ and $\sigma \approx 3.2$ with a sparse ternary secret with a variety of Hamming weights $h \in \{64, 128, 256, 512\}$.

5.5.1 Using Sparse Secrets with Existing Recommended Parameter Sets

Next, we consider the impact of using a sparse ternary secret of Hamming weight $h = 128$ for the sets of parameters $(n, \log q, \alpha)$ as recommended in Table 5.2 for uniform ternary secret. That is, we take the parameter sets currently recommended in the HE standard for uniform-ternary secrets, and swap the secret distribution to be sparse ternary of Hamming weight $h = 128$, in order to determine the effect on the security. In Table 5.3 we report the output of the LWE Estimator giving an estimate of the concrete security of the parameter sets $(n, \log q, \alpha, h)$ with a sparse ternary secret of Hamming weight $h = 128$. For consistency,

we have used the same sieving cost model $T(\beta, d) = 2^{0.292\beta+16.4+\log(8d)}$ which is used to generate [ACC+18, Table 1].

n	$\log q$	α	h	usvp	dual	$\lambda_{\mathsf{target}}$
1024	27	8/q	128	**124.9**	127.8	128
2048	54	8/q	128	125.0	**122.0**	
4096	109	8/q	128	124.9	**117.9**	
8192	218	8/q	128	126.4	**117.2**	
16384	438	8/q	128	127.0	**117.1**	
32768	881	8/q	128	127.5	**116.5**	
1024	19	8/q	128	**178.2**	179.2	192
2048	37	8/q	128	186.5	**173.7**	
4096	75	8/q	128	186.6	**165.2**	
8192	152	8/q	128	186.4	**167.5**	
16384	305	8/q	128	187.7	**159.1**	
32768	611	8/q	128	188.8	**161.9**	
1024	14	8/q	128	**235.4**	238.5	256
2048	29	8/q	128	231.9	**217.3**	
4096	58	8/q	128	234.3	**210.2**	
8192	118	8/q	128	232.8	**207.9**	
16384	237	8/q	128	233.8	**195.8**	
32768	476	8/q	128	234.6	**214.7**	

Table 5.3: Impact of using a sparse ternary secret of Hamming weight $h = 128$, using the currently standardised LWE parameter sets at the target 128-, 192- and 256-bit security level for uniform ternary secret specified in Table 5.2. Estimates of the security of each parameter set against usvp and dual attacks under the BKZ cost model $T(\beta, d) = 2^{0.292\beta+16.4+\log(8d)}$ are presented, where β is the blocksize and d is the dimension. The best performing attack for each parameter set is highlighted in bold.

It can be seen from Table 5.3 that introducing a sparse secret of Hamming weight $h = 128$, there is a noticeable security loss, by up to 11 bits at the target 128-bit security level, by up to 32 bits at the target 192-bit security level, and by up to 60 bits at the target 256-bit security level. Table 5.3 does not take into account hybrid attacks, which are likely to be

competitive, and hence the security loss may be even greater.

5.5.2 Sparsity vs. Performance Trade-off

In Table 5.4 we illustrate the effect of using a sparse ternary secret with various Hamming weights h on the bit size $\log q$ with $n = 1024$ and $\sigma = 3.19$ fixed. For comparison, we also note the bit-length $\log Q$ which is currently recommended to achieve target security level λ for the same n, σ with a uniform ternary secret in the HE Standard. A smaller modulus q may impact on practical performance of the schemes. For example when we need to ensure that q is large enough to support the full computation, to ensure correct decryption. The lower q required by introducing a sparse secret may necessitate moving to a higher dimension n to support the computation, which in turn will be slower.

5.5.3 Sparsity as a Proportion of Target Security: Exploration of Choices for ζ

In Table 5.5 we present an exploration of possible choices for the value $\zeta = \frac{h}{\lambda}$, illustrating the reduction in bit-length $\log q$ required to retain the desired level of security when using a sparse ternary secret compared to a uniform ternary secret. Table 5.5 uses the cost model $T(\beta, d) = 2^{0.292\beta + 16.4 + \log(8d)}$, i.e. BKZ.sieve available in the LWE Estimator, and considers the following attacks: uSVP, dual, hybrid-decoding, and hybrid-dual. We provide as a point of comparison $\log Q$, the bit-length of the modulus Q currently recommended in [ACC+18] for the given parameters $(n, \sigma = 3.19)$ with uniform ternary secret at target security level λ.

Table 5.5 indicates that a choice such as $\zeta = 1$ gives a reasonable trade-off between performance and security. In this case, we can retain secure parameters with at most a 24% drop in the bit-length of the modulus $\log q$ compared to that recommended at target security level λ for the same values of n and $\sigma = 3.19$) and uniform ternary secret. This choice corresponds to a sparse ternary secret with Hamming weight $h \in \{128, 192, 256\}$, depending on the desired security level, which allows for more efficient bootstrapping when compared to uniform-ternary secrets.

h	n	λ	$\log q$	usvp	dual	$\log Q$
64	1024	128	24	**131.0**	133.8	27
		192	16	**193.8**	198.3	19
		256	11	**265.2**	279.1	14
128	1024	128	26	**130.4**	131.4	27
		192	17	**198.6**	199.8	19
		256	12	**269.6**	279.7	14
256	1024	128	27	**128.1**	134.1	27
		192	18	**194.9**	208.6	19
		256	13	**267.0**	293.2	14
512	1024	128	27	**132.0**	137.6	27
		192	18	**202.9**	219.0	19
		256	14	**261.3**	292.6	14
$\lceil \frac{2n}{3} \rfloor$	1024	128	27	**133.3**	138.7	27
		192	19	**194.7**	207.7	19
		256	14	**265.6**	293.8	14

Table 5.4: Bit-length $\log q$ of moduli required to provide target security level λ, for $\lambda \in \{128, 192, 256\}$, for various secret densities. We note that the LWE Estimator treats uniform ternary secrets as ternary fixed-weight secrets with Hamming weight $h = \lceil \frac{2n}{3} \rfloor$. The best performing attack for each parameter set is highlighted in bold.

5.5.4 Standardising Larger Dimensions n

With current progress in applied homomorphic encryption it is becoming necessary to work in dimensions larger than $n = 2^{15}$, the largest dimension currently supported in the HE Standard. Several recent papers [CHK+18, CCS19a, HHC19, KSK+18] have reported implementations in dimension $n = 2^{16}$, and an implementation in dimension $n = 2^{17}$ was reported in [KSW+18]. A natural extension of the current standard would therefore be to standardise parameter sets for dimension $n = 2^k$ for some $k \geq 16$, since power-of-two n remain the most widely used in practice. Moreover, power-of-two n enables convenient coefficient-wise error sampling and

n	λ	$\log q^{(\zeta=\frac{1}{2})}$	$\log q^{(\zeta=\frac{3}{4})}$	$\log q^{(\zeta=1)}$	$\log q^{(\zeta=\frac{3}{2})}$	$\log Q$
1024	128	14	19	21	23	27
	192	10	13	14	16	19
	256	7	10	11	12	14
2048	128	27	37	41	46	54
	192	19	26	29	32	37
	256	15	19	22	24	29
4096	128	55	74	83	92	109
	192	37	52	57	64	75
	256	30	39	44	49	58
8192	128	111	148	171	186	218
	192	84	100	114	130	152
	256	60	79	89	98	118
16384	128	223	300	342	377	438
	192	157	201	232	265	305
	256	115	161	176	202	237
32768	128	496	619	699	767	881
	192	350	411	479	523	611
	256	263	313	361	408	476

Table 5.5: The reduction in bit-length $\log q$ of the modulus q required to retain the desired level of security against the dual, usvp, hybrid-dual and hybrid-decoding attacks, under our assumptions, when using a sparse ternary secret parameterised by $\zeta = \frac{h}{\lambda}$ compared to a uniform ternary secret. The lattice reduction cost model is $T(\beta, d) = 2^{0.292\beta+16.4+\log(8d)}$, and a (conservative) estimate for both the hybrid-dual and hybrid-decoding attacks are obtained by considering a square-root speed-up in the search space, and ignoring any meet-in-the-middle probabilities.

would require no change to the currently standardised error distribution.

For $n = 2^{16}$, it is straightforward to apply the methodology used in the current HE Standard [ACC+18],

i.e. to use the LWE Estimator to find an appropriate $\log q$ to meet security requirements for fixed $\sigma = 3.19$ and a currently standardised secret distribution. We present the results of such an analysis for a uniform ternary secret distribution in Table 5.6, which gives an estimate of the security of the proposed parameter sets $(n = 2^{16}, \sigma = 3.19, \log q)$ against the usvp and dual attacks under a sieving lattice reduction cost model.

λ	n	$\log q$	usvp	dual
128	65536	1782	128.3	128.4
192	65536	1242	192.5	192.0
256	65536	963	256.7	257.7

Table 5.6: Required bit-length $\log q$ of moduli required to attain target security level λ under the usvp and dual attacks, with $\lambda \in \{128, 192, 256\}$, for dimension $n = 65536$, under the sieving-based cost model $T(\beta, d) = 2^{0.292\beta + 16.4 + \log(8d)}$.

For $n \geq 2^{17}$ the same methodology works in theory, although it can become cumbersome to run hybrid attack estimates many times for such very large parameter sets. To find suitable moduli q achieving target security λ for higher values of n, we can extrapolate using the data we already have using the apparent linear relationship between n and $\log q$. That is, for a fixed σ, $n \log q$ is essentially constant for a fixed target security level. This means we can easily extrapolate entries for larger values of n, without having to explicitly run new experiments. We illustrate in Figure 5.4 this for a sparse ternary secret of fixed Hamming weight $h = \lambda$ (i.e. $\zeta = 1$). When considering pre-quantum estimates, we can represent $\log(q)$ as a linear function of n by extrapolation from the data in Table 5.5:

$$\begin{aligned}
\mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(128,1)}(n) &= 0.021370n - 3.601989 \\
\mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(192,1)}(n) &= 0.014630n - 3.139303, \text{and} \\
\mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(256,1)}(n) &= 0.011007n - 1.184080.
\end{aligned}$$

These linear models were found using the `find_fit` function in SageMath [S+20]. For readability, we round the coefficients to six decimal places in each case.

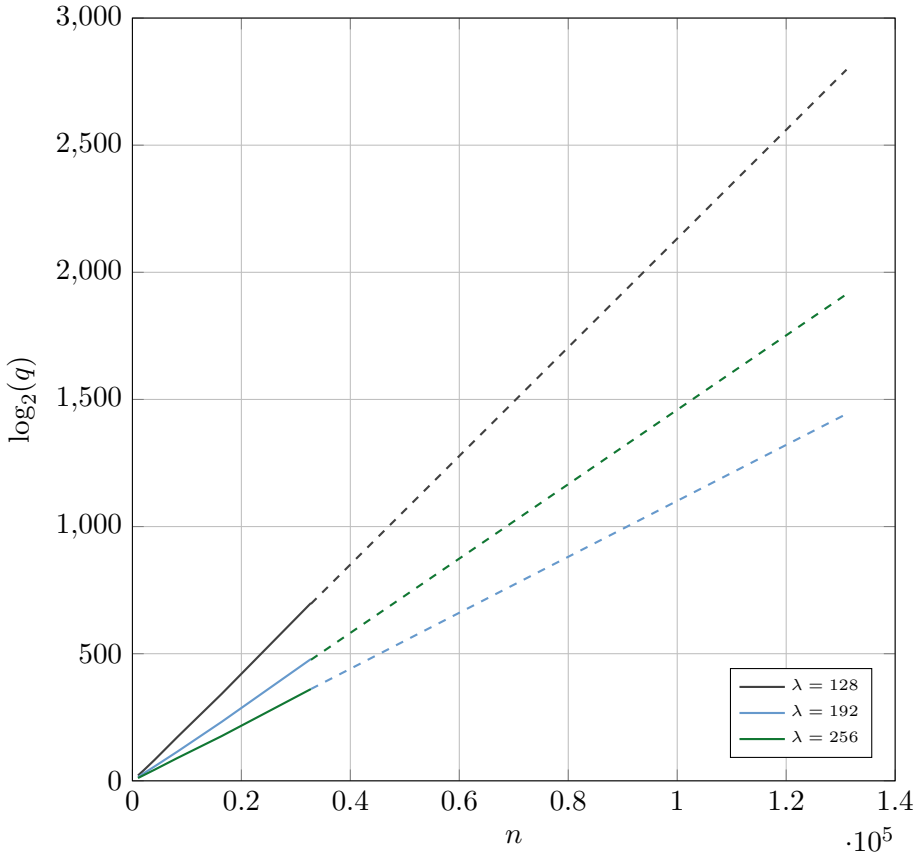


Figure 5.4: Extrapolation to $n = 65536$ and $n = 131072$ using the data from Table 5.5 for the value $\zeta = 1$. Here, we consider the lattice reduction cost model $T_{\mathsf{BKZ}}(\beta, d) = 2^{0.292\beta+16.4+\log(8d)}$ and extrapolate using the SageMath function `find_fit`. Note that the solid lines represent values covered by data points, and the dashed lines represents extrapolation.

We can use these extrapolations to generate values for $\log(q)$. For $n = 65536$, we see that:

$$\begin{aligned}\lfloor \mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(128,1)}(65536)\rfloor &= 1396,\\ \lfloor \mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(192,1)}(65536)\rfloor &= 955,\\ \lfloor \mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(256,1)}(65536)\rfloor &= 720,\end{aligned}$$

that is, for $\lambda = 128$ we have $\log(q) = 1396$, for $\lambda = 192$ we have $\log(q) = 955$, and for $\lambda = 256$ we have $\log(q) = 720$. For $n = 131072$, we see that:

$$\begin{aligned}\lfloor \mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(128,1)}(131072)\rfloor &= 2797,\\ \lfloor \mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(192,1)}(131072)\rfloor &= 1914, \text{ and}\\ \lfloor \mathsf{logq}^{\mathsf{sieve}}_{(\lambda,\zeta)=(256,1)}(131072)\rfloor &= 1441,\end{aligned}$$

that is, for $\lambda = 128$ we have $\log(q) = 2797$, for $\lambda = 192$ we have $\log(q) = 1914$, and for $\lambda = 256$ we have $\log(q) = 1441$.

5.5.4.1 Standardising the Standard Deviation σ

The choice of standard deviation $\sigma = 3.19$ is somewhat arbitrary, and we could consider including wider standard deviations. In Table 5.7 we consider the impact of various standard deviations $\sigma \in \{0.80, 2.90, 3.20, 32, 320, 3200\}$ on the bit-length $\log q$ of the required modulus to achieve a target level of security λ for fixed $\lambda = 128$, and fixed weight ternary secret with Hamming weight $h = 128$. As an illustrative example, we choose the case of $n = 8192$. We consider security against the usvp and dual attacks under the sieving-based cost model $T(\beta, d) = 2^{0.292\beta + 16.4 + \log(8d)}$ for lattice reduction. However, we note that, as σ decreases, it may be beneficial to guess components of the error vector, and the LWE Estimator does not consider such techniques.

σ	n	$\log q$	usvp	dual
0.80	8192	194	141.8	128.5
2.90	8192	196	141.7	128.5
3.20	8192	196	141.8	128.6
32	8192	204	138.2	128.4
320	8192	203	141.5	128.1
3200	8192	211	137.9	128.1

Table 5.7: Maximal moduli q required to attain target security level $\lambda = 128$ for various values of σ. In each case, we consider a ring dimension of $n = 8192$ and a fixed weight ternary secret with Hamming weight $h = 128$.

In general, we can see that as the value of σ increases, we attain a larger permissible value of $\log(q)$ for the fixed value of $n = 8192$.

5.6 Conclusion

In this chapter, we have considered the current state of the Homomorphic Encryption Security Standard. We have considered the impact and feasibility of considering sparse secret parameter sets in future variants of the standard. We note that the current version of the HE Standard does not consider hybrid attacks, and urge that any parameter sets are analysed against hybrid attacks before standardisation.

We have shown that, when sparse secrets are considered, the LWE modulus q can require a significant reduction in size in order to attain the target security level. This can be problematic in homomorphic encryption schemes, where we are looking for a large modulus q in order to be able to carry out computations correctly. The attack landscape for sparse-secret LWE is fast moving, with new contributions [CHHS19, EJK20] emerging in recent years.

Chapter 6

Privacy-preserving Kriging Interpolation

Contents

This chapter is based on the following publication:

James Alderman, Benjamin R. Curtis, Oriol Farràs, Keith M. Martin, and Jordi Ribes-González. Private Outsourced Kriging Interpolation. The 5th Workshop on Encrypted Computing and Applied Homomorphic Cryptography. In International Conference on Financial Cryptography and Data Security Workshops (pp. 75-90). Springer, volume 10323 of Lecture Notes in Computer Science. 2017. Additional details have been added in this thesis.

In this chapter we present a solution which enables private outsourced Kriging interpolation. We outline techniques that can be used to "factor out" sensitive parameters from the system of equations which need to be solved by the server. Using these technique, alongside additively homomorphic encryption, we can securely outsource Kirging interpolation.

The author of this thesis contributed towards (a) the design of the private outsourced kriging interpolation scheme, and (b) the writing of the paper. No contribution was made towards the implementation, and therefore minimal details are provided in this chapter.

6.1 Introduction and Contribution

Kriging is a spatial interpolation algorithm which provides the best linear unbiased prediction (BLUP) of an observed phenomenon, by taking a weighted average of samples within a specified neighbourhood. It is widely used in areas such as geo-statistics where, as an example, it may be used to predict the quality of mineral deposits at an unobserved location based on previous measurements. There are many variants of Kriging, but we focus on the widely used *Ordinary Kriging* variant, and refer to it simply as Kriging throughout this chapter. Kriging can be outsourced to a cloud service provider, although measurements and predictions can be highly sensitive and must therefore be protected.

In this chapter, we present a method for the private outsourcing of Kriging interpolation. We use a modified variant of the Kriging algorithm in combination with homomorphic encryption (as described in Chapter 2). Our solution allows crucial information relating to measurement values to be hidden from the cloud service provider. Crucially, we only require *additively homomorphic encryption* which allows for an efficient solution to be built.

Kriging is a form of linear interpolation that predicts the value $[z_0^*]$ of some phenomenon at an unobserved location $\mathbf{q}_0 = (x_0, y_0)$ in a two-dimensional region. The quality of Kriging prediction relies on the *parameters* considered, as well as the *variogram model* which describes the spatial continuity of the data. These parameters, and the variogram model, are chosen prior to interpolation. The Kriging prediction is then formed as a weighted sum of *prior measurements*, where measurements closer to the query point $\mathbf{q}_0$ are given a greater weight in the sum than those which are further away. This reflects the assumption that measurements taken at nearby locations are more likely to be 'similar' than measurements taken further apart.

Kriging was designed with geo-statistical applications in mind (e.g. to predict the best location to mine based on the mineral deposits found at previous measurements within a region), but has also found applications in a variety of settings including remote sensing [RDB94], real-

estate appraisal [KH14], and computer simulations [Kle09]. Kriging has been identified as a good candidate process to be outsourced, based on the practical and legislative requirements of industrial users [Cla16, Ing16].

Many users may need access to a Kriging prediction service (indeed, legal frameworks may require such data to be shared amongst relevant authorities [Eur07]). A secured storage server may be preferable to distributing copies of the entire dataset to each authorised user, especially when datasets are large and/or user devices are constrained. Furthermore, Kriging might need to be performed over data owned by *multiple* organizations, with an independent cloud service provider performing processing duties on behalf of all concerned parties.[1] Centralised outsourcing also makes sense when remote sensors take frequent measurements and push the results to a central database.

Consider a client $\mathcal{C}$ that owns a Kriging dataset (a set of measurements taken at various locations) which it wishes to outsource to an *honest-but-curious* cloud service provider $\mathcal{S}$. In the honest-but-curious model, the cloud service provider $\mathcal{S}$ follows the protocol as described, but also attempts to learn as much as possible from the resulting transcript of messages. Our results do not translate to the malicious setting, since we have no way of verifying the computations made by the sever in our solution. A maliciously-secure protocol for the private outsourcing of Kriging interpolation is left for future work. The client $\mathcal{C}$ would like to make use of both the storage *and* computational power of the server $\mathcal{S}$ to make a Kriging prediction service on its dataset available to multiple users. Further, other *data generating nodes* may be authorised by $\mathcal{C}$ to add/remove data (measurements) to/from the outsourced dataset.

A trivial solution consists of encrypting all data using a symmetric encryption scheme and using the server only for storage-as-a-service. To compute a Kriging prediction, all relevant data is retrieved, decrypted and computed on locally. Unfortunately, this solution may not be efficient, particularly if client devices have limited computational power or storage capacity. Moreover, this solution requires high bandwidth during queries. This may be an issue if, for example, a surveyor in the field requires an on-line Kriging prediction service. In this case, mobile data services may be expensive, intermittently available, or slow.

An alternative is to compute the entire Kriging process on encrypted data by encrypting all data using FHE. Unfortunately, Kriging involves several computations that are currently

[1] In this case, it may be preferable to use *Multi-key Homomorphic Encryption* [LTV12, PS16, CCS19b, CDKS19]. We do not consider this setting in our work, and assume that there is a single data owner $\mathcal{C}$.

challenging when using FHE, including computing square roots and natural exponentiations. It is certainly possible to outsource the Kriging process and protect *all* information using FHE. However, this results in prohibitively high encryption and decryption costs as well as a large amount of interactivity and local computation, which may diminish the benefits of the client/server setting. Preliminary experiments using the SEAL library [Mic20] (without optimization of code or parameter choices) did not yield promising results when computing a Kriging prediction using a dataset of only three measurements.

Our proposed solution uses additively homomorphic encryption to outsource Kriging interpolation efficiently. We make a trade-off by protecting only the most sensitive parameters. That is, we protect the measurement values in the dataset, the generated Kriging predictions and (a subset of) the variogram parameters chosen by the client. We do not hide locations (of prior measurements or queries), noting that prior measurement locations may well be externally observable (e.g. if measurements come from previous mining operations). We discuss issues surrounding data leakage in more detail in Section 6.6.

Our main contribution is to show that the Kriging process can be adapted such that the sensitive variogram parameters may be 'factored out' from the online computation provided by the server, whilst the remainder of the Kriging computation may be performed on *encrypted* measurement values using an additively homomorphic encryption scheme. We thus gain a practical, efficient, and secure solution to privately outsource Kriging. An outline of our protocol is as follows:

1. $\mathcal{C}$ uploads an encrypted dataset, consisting of n measurements, to $\mathcal{S}$.
2. $\mathcal{S}$ prepares the Kriging dataset for future queries. This process includes plaintext operations that are also necessary in an unprotected outsourced Kriging scheme.
3. $\mathcal{C}$ makes a query to $\mathcal{S}$ requesting a Kriging prediction at a location (x_0, y_0), which is done in plaintext with virtually no cost.
4. $\mathcal{S}$ computes the interpolation on encrypted measurements.
5. $\mathcal{C}$ decrypts the result.

Cryptographically-secured Kriging has been studied previously in a different setting, where a *server* owns a dataset and clients may query the dataset at a previously unsampled location [TP13]:

the queried location and resulting prediction should be private from the server, whilst the dataset held by the server should be private from the client. Two solutions are proposed in [TP13] which, unlike our solution, support only one variogram model and require high communication complexity, interactivity and local computation. The first is based on creating random 'dummy' queries to hide the queried location, and using an oblivious transfer protocol to hide predictions for all but the legitimate query location. The second solution uses the Paillier encryption scheme [Pai99] in an interactive protocol requiring multiple round-trips between client and server. In [TP14] collaborative private Kriging was investigated, where users combine their datasets to gain more accurate Kriging predictions.

The remainder of this chapter is structured as follows. In Section 6.2 we describe the Kriging interpolation process. In Section 6.3 we define our system model and analyse the required security properties of each piece of data in our setting. In Section 6.4 we introduce the idea of a *canonical* variogram, which we use in our construction to allow the server to compute a Kriging prediction without relying on the sensitive parameters. Our construction is given in Section 6.5 and we discuss its performance in Section 6.6. Finally, in Section 6.7, we conclude and outline some potential directions for future work.

6.2 Kriging Interpolation

This section outlines the background theory of Kriging interpolation. For more detail, see [CD99, Cre92, Kri51, Wac13]. To highlight which objects are to be kept secret from the sever, we use brackets, e.g. $[x]$ represents that we want to keep the value x secret (via encryption or other techniques). The Kriging process starts with a set of measurements taken at some locations in a spatial region, and produces predicted measurements at unsampled locations. We denote this spatial region by $R \subset \mathbb{R}^2$, and the locations of prior measurements by $P = (\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n)$, where each $\mathbf{r}_i = (x_i, y_i) \in R$. The Euclidean distance[2] between two locations $\mathbf{r}_i, \mathbf{r}_j \in R$ is:

$$d(\mathbf{r}_i, \mathbf{r}_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

We refer to the set of taken measurements by $S = ([z_1], [z_2], \ldots, [z_n])$, where $[z_i]$ corresponds to a measurement taken at the corresponding location $\mathbf{r}_i \in P$. We define the *Kriging dataset* to be the tuple (P, S), which consists of all measurement values and their corresponding

[2]We note that some Kriging datasets consider the *Great Circle distance*, i.e. the distance between two points along the surface of a sphere, as the distance metric; we do not consider such cases in this chapter.

locations. That is:

$$(P, S) = ((\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n), ([z_1], [z_2], \ldots, [z_n])).$$

The Kriging process allows a client to query an arbitrary location $\mathbf{r}_0 \in R$, in order to receive a prediction $[z_0^*]$ of the true value $[z_0]$ that would be measured at $\mathbf{r}_0$.

6.2.1 Overview of the Kriging Procedure

Informally, Kriging consists of three phases:

1. *Computing the experimental variogram*: one of the underlying assumptions of the Kriging process is that two measurements of a phenomenon will be similar when measured at nearby locations. Using the sampled dataset, one can plot the *experimental variogram* to show the dependence between measurements sampled at locations at certain distances h.

2. *Fitting a variogram model*: the experimental variogram is not usually sufficient to use in the Kriging prediction directly, since there may not be sampled data at every required distance. Therefore, one chooses a *variogram model* and empirically selects model parameters to fit a curve to the points of the experimental variogram.

3. *Computing the prediction*: using the variogram, we can determine the appropriate weights for each measurement (based on the distance between each measurement and the queried location). The Kriging prediction is then computed as a weighted sum of the measured samples.

6.2.2 The Variogram

Let $N(h)$ be the set of all pairs of measurements taken approximately distance h apart, that is:

$$N(h) = \{([z_i], [z_j]) \mid d(\mathbf{r}_i, \mathbf{r}_j) \in (h - \Delta, h + \Delta)\}.^3$$

[3]The approximation tolerance Δ can be increased when the Kriging dataset does not include enough sample points at a close enough distance.

The *experimental variogram* γ^* plots, for every distance h such that $N(h) \neq \emptyset$:

$$\gamma^*(h) = \frac{1}{2N(h)} \sum_{(z_i,z_j)\in N(h)} ([z_i] - [z_j])^2. \tag{6.1}$$

A suitable variogram function $\gamma : \mathbb{R}^{\geq 0} \to \mathbb{R}$ must satisfy a set of conditions [CD99, Cre92]. The most commonly used models require:

- $\gamma(0) = 0$,
- $\gamma(h)$ is positive and bounded, and
- the existence of the limits $\lim_{h\to 0^+} \gamma(h)$ and $\lim_{h\to\infty} \gamma(h)$.

These models are parametrized by the following four variables:

1. the *nugget effect* $[\eta]$: the limit of $\gamma(h)$ as $h \to 0^+$,
2. the *sill* $[\nu]$: the limit of $\gamma(h)$ as $h \to \infty$,
3. the *range* ρ: which controls how fast $\gamma(h)$ approaches $[\nu]$ as h increases, and
4. the *partial sill* $[\mu]$*:* the difference between the sill and the nugget, i.e. $[\mu] = [\nu] - [\eta]$.

Typically, a variogram model is chosen from a set of standard parametric variogram models. This model is then fitted to the experimental variogram by empirically adjusting the nugget effect, sill, and range parameters. The most common choices of bounded variogram models are, for $h > 0$:

1. the *bounded linear model*: $\gamma(h) = [\nu] - ([\nu] - [\eta]) \left(1 - \frac{h}{\rho}\right) \cdot 1_{(0,\rho)}(h)$,
2. the *exponential variogram model*: $\gamma(h) = [\nu] - ([\nu] - [\eta])e^{-h/\rho}$,
3. the *spherical variogram model*: $\gamma(h) = [\nu] - ([\nu] - [\eta]) \left(1 - \frac{3h}{2\rho} + \frac{h^3}{2\rho^3}\right) \cdot 1_{(0,\rho)}(h)$, and
4. the *Gaussian variogram model*: $\gamma(h) = [\nu] - ([\nu] - [\eta])e^{-h^2/\rho^2}$,

where $1_I(x) = 1$ if $x \in I$, and $1_I(x) = 0$ otherwise.

6.2.3 The Normal Equations

Let γ be one of the variogram models discussed in Section 6.2.2 with parameters chosen to fit the experimental variogram. To construct the best linear unbiased predictor of the phenomenon at a queried location $\mathbf{r}_0 = (x_0, y_0) \in R$, we first form the *Kriging matrix* $\mathbf{K} \in \mathbb{R}^{(n+1)\times(n+1)}$ with elements:

- $\mathbf{K}_{i,j} = \gamma(d(\mathbf{r}_i, \mathbf{r}_j))$ for $1 \leq i, j \leq n$,
- $\mathbf{K}_{n+1,i} = \mathbf{K}_{i,n+1} = 1$ for $i \neq n+1$, and
- $\mathbf{K}_{n+1,n+1} = 0$.

That is,

$$\mathbf{K} = \begin{pmatrix} \gamma(d(\mathbf{r}_1,\mathbf{r}_1)) & \gamma(d(\mathbf{r}_1,\mathbf{r}_2)) & \dots & \gamma(d(\mathbf{r}_1,\mathbf{r}_n)) & 1 \\ \gamma(d(\mathbf{r}_2,\mathbf{r}_1)) & \gamma(d(\mathbf{r}_2,\mathbf{r}_2)) & \dots & \gamma(d(\mathbf{r}_2,\mathbf{r}_n)) & 1 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \gamma(d(\mathbf{r}_n,\mathbf{r}_1)) & \gamma(d(\mathbf{r}_n,\mathbf{r}_2)) & \dots & \gamma(d(\mathbf{r}_n,\mathbf{r}_n)) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}.$$

Define $\mathbf{v} \in \mathbb{R}^{n+1}$ with $v_i = \gamma(d(\mathbf{r}_0, \mathbf{r}_i))$ for $1 \leq i \leq n$, and $v_{n+1} = 1$. Finally, let $\mathbf{w} \in \mathbb{R}^{(n+1)}$ satisfy $\mathbf{K}\mathbf{w} = \mathbf{v}$:

$$\begin{pmatrix} \gamma(d(\mathbf{r}_1,\mathbf{r}_1)) & \gamma(d(\mathbf{r}_1,\mathbf{r}_2)) & \dots & \gamma(d(\mathbf{r}_1,\mathbf{r}_n)) & 1 \\ \gamma(d(\mathbf{r}_2,\mathbf{r}_1)) & \gamma(d(\mathbf{r}_2,\mathbf{r}_2)) & \dots & \gamma(d(\mathbf{r}_2,\mathbf{r}_n)) & 1 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \gamma(d(\mathbf{r}_n,\mathbf{r}_1)) & \gamma(d(\mathbf{r}_n,\mathbf{r}_2)) & \dots & \gamma(d(\mathbf{r}_n,\mathbf{r}_n)) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \\ w_{n+1} \end{pmatrix} = \begin{pmatrix} \gamma(d(\mathbf{r}_0,\mathbf{r}_0)) \\ \gamma(d(\mathbf{r}_0,\mathbf{r}_1)) \\ \vdots \\ \gamma(d(\mathbf{r}_0,\mathbf{r}_n)) \\ 1 \end{pmatrix}.$$

The *(Ordinary) Kriging prediction* $[z_0^*]$ of the measured phenomenon at the location $\mathbf{r}_0$ is computed as the weighted sum of the sampled measurements, with the weights defined by $\mathbf{w}$. Specifically, we have:

$$[z_0^*] = \sum_{i=1}^{n} w_i [z_i].$$

The set of linear equations defined by $\mathbf{K}$, $\mathbf{w}$, and $\mathbf{v}$ are known as the *Normal Equations*. They are derived by ensuring that the induced linear predictor is unbiased (this is done by forcing the first n weights to sum to one, i.e. $\sum_{i=1}^{n} w_i = 1$) while minimizing the variance of the induced linear predictor [Wac13]. The resulting minimized variance $[\sigma_0^{*2}]$ is called the *(Ordinary) Kriging variance*, and it is described by the following expression:

$$[\sigma_0^{*2}] = w_{n+1} + \sum_{i=1}^{n} w_i \gamma(d(\mathbf{r}_0, \mathbf{r}_i)).$$

The Kriging variance allows for the construction of confidence intervals for each prediction and this describes the error associated with the prediction. We define a variogram function to be *non-degenerate* if $[\eta] \neq [\nu]$, and we restrict our attention to non-degenerate variograms in this chapter. Using the degenerate variogram (also called the *nugget effect* variogram [Wac13]) results in the average Kriging predictor $[z_0^*] = \sum_{i=1}^{n} [z_i]/n$ at all unsampled locations $r_0 \notin P$, with Kriging variance $[\sigma_0^{*2}] = n + 1$.

6.2.4 Toy Example

Throughout this chapter, we make use of a small example dataset for illustrative purposes. To do this, we use the PyKrige library [PYK20] following in a similar fashion to the example code provided with the library. Specifically, we expand their *Ordinary Kriging* example code and utilise their example dataset. This example code considers the toy dataset, consisting of five points, outlined in Table 6.1.

x	y	$[z]$
0.3	1.2	[0.47]
1.9	0.6	[0.56]
1.1	3.2	[0.74]
3.3	4.4	[1.47]
4.7	3.8	[1.74]

Table 6.1: The toy dataset used throughout this section, as in the ordinary kriging example used in the PyKrige library [PYK20]. The x_i, y_i values are coordinates of the measurement $[z_i]$.

In our example, we choose to set the value of Δ to be 0. Therefore, since the size of our

```
def compute_experimental_variogram(data):

vals = []
Nh = 1
variogram_vals = []
h = []

for i in range(len(data)):
for j in range(i+1,len(data)):
xij = data[i][0] - data[j][0]
yij = data[i][1] - data[j][1]
zij = data[i][2] - data[j][2]
hij =  sqrt(xij**2 - yij**2)
h.append(hij)
vals.append(zij**2)

variogram_vals = [1/(2*Nh) * z for z in vals]

return (h, variogram_vals)
```

Figure 6.1: Code used to plot the experimental variogram of a dataset, such as the example dataset presented in Table 6.1.

dataset is five points, we note that there are at most $\binom{5}{2} = 10$ distinct sets of the form:

$$N(h_k) = \{([z_i], [z_j]) \mid d(\mathbf{r}_i, \mathbf{r}_j) = h_k\},$$

for $1 \leq k \leq 10$. For our dataset, it turns out that each value h_k is *distinct*, and we therefore have exactly ten sets $\{N(h_k)\}_{k=1}^{10}$. We plot h against $\gamma^*(h)$ in Figure 6.2.

We can use the custom code outlined in Figure 6.1 to determine a good variogram model. We note that PyKrige allows the usage of *binning* when computing a variogram, which corresponds to a selection of $\Delta \neq 0$ and considering Euclidean distances in the range $(h - \Delta, h + \Delta)$. As noted above, we consider $\Delta = 0$ in our example. We begin by plotting the experimental variogram according to Equation 6.1 in Figure 6.1.

We can also use Pykrige to plot the experimental variogram, which we do using the example code provided with the PyKrige library, and fit a linear variogram to the data (since Figure 6.2 shows that this is a good model for fitting). Recall that the linear model is of the form:

$$\gamma(h) = [\eta] - ([\eta] - [\nu])(1 - \frac{h}{\rho}),$$

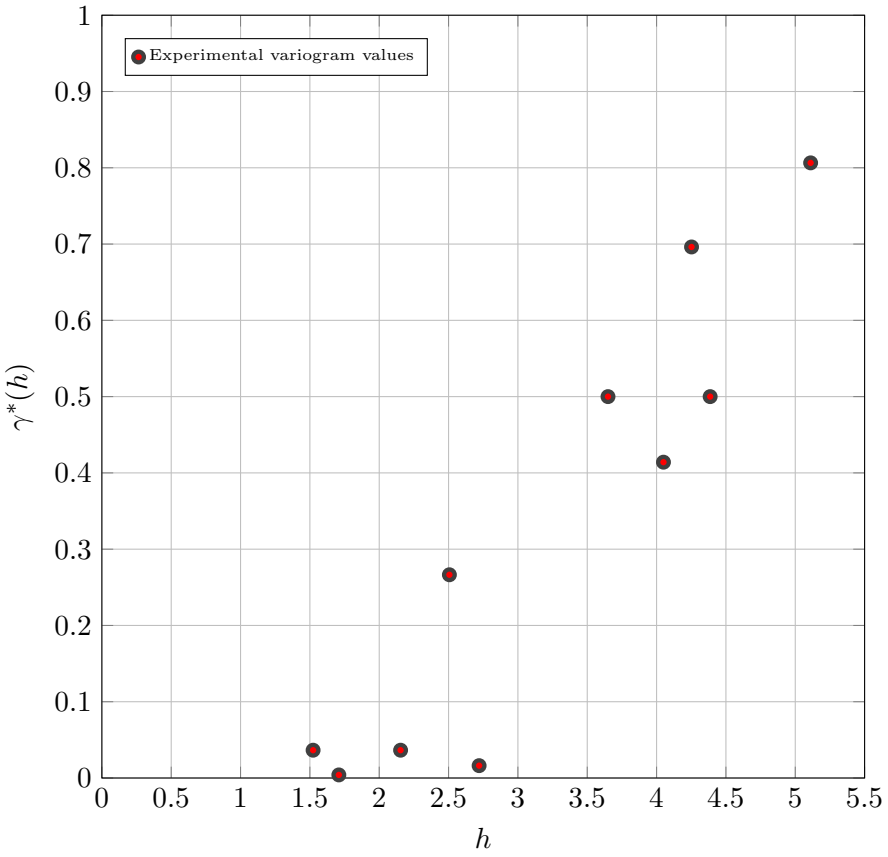


Figure 6.2: The experimental variogram for our example dataset, presented in Table 6.1, with $\Delta = 0$

and PyKrige fits a model of the form:

$$\gamma(h) = 5.2517 \cdot 10^{-11} + 0.1168h,$$

which corresponds to choices of $\rho = 1$, $[\eta] \approx [0.1168]$, $[\nu] = [5.2517 \cdot 10^{-11}]$. Note that PyKrige does not consider a range in their linear models, so we have omitted the Indicator function used in Section 6.2.2. For this variogram we perform Kriging interpolation on a 10×10 grid between the values $x, y \in [0, 5]$, and the results are displayed in Figure 6.4. We can, of course, change the granularity of this prediction and predict over an $(r \times s)$ grid for any $r, s \in \mathbb{Z}$.

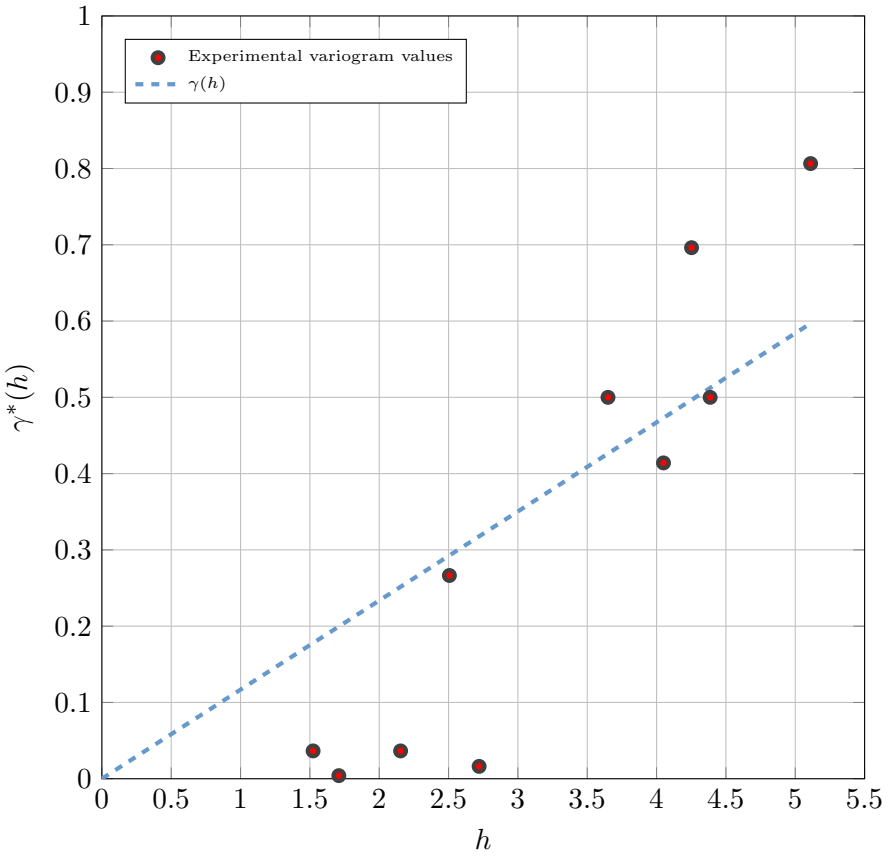


Figure 6.3: A linear model fitted to the experimental variogram using the Pykrige library, for our example dataset presented in Table 6.2.

$y \mid x$	0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
5.0	0.90	0.95	1.02	1.11	1.20	1.31	1.42	1.52	1.61	1.67	1.72
4.5	0.83	0.89	0.96	1.05	1.15	1.27	1.40	1.51	1.61	1.69	1.73
4.0	0.77	0.81	0.88	0.97	1.08	1.21	1.34	1.47	1.59	1.70	1.74
3.5	0.70	0.74	0.79	0.88	1.00	1.13	1.27	1.40	1.52	1.64	1.69
3.0	0.64	0.67	0.71	0.81	0.93	1.05	1.18	1.31	1.43	1.53	1.59
2.5	0.58	0.61	0.66	0.75	0.85	0.97	1.09	1.21	1.32	1.42	1.48
2.0	0.53	0.56	0.61	0.69	0.78	0.89	1.01	1.12	1.23	1.32	1.39
1.5	0.49	0.51	0.56	0.62	0.71	0.81	0.92	1.04	1.14	1.23	1.30
1.0	0.46	0.48	0.52	0.56	0.63	0.74	0.85	0.96	1.06	1.15	1.22
0.5	0.45	0.46	0.49	0.53	0.58	0.69	0.80	0.90	1.00	1.08	1.15
0	0.44	0.46	0.49	0.52	0.58	0.67	0.76	0.86	0.94	1.02	1.09

Figure 6.4: Example interpolation grid using the linear model fitted to the experimental variogram presented in Figure 6.3.

6.3 Private Outsourced Kriging Interpolation

We now introduce the setting that we consider in this chapter. We have a client $\mathcal{C}$ that owns a Kriging dataset (P, S), as defined in Section 6.2, along with a variogram γ computed as in Section 6.2.4. We also have an honest-but-curious server $\mathcal{S}$ that is willing to perform outsourced Kriging on behalf of the client $\mathcal{C}$. Finally, we have additional users $\mathcal{U}$ that are authorised by $\mathcal{C}$ to make Kriging queries to $\mathcal{S}$, and we may also consider additional data generating nodes (e.g. several remote sensors placed in locations of interest) that may update the outsourced dataset by producing additional measurement data or removing previous (e.g. outdated) measurements. The requirements of each party are as follows:

- The data owner must choose the variogram model to be used, as well as the associated parameters. They upload a Kriging dataset to the server, and should be able to update data and request Kriging predictions.
- The data users may request Kriging predictions and update data. Here, update data refers to the addition of a new measurement $[z_i]$ to the set of measurements S, and the addition of the corresponding location $\mathbf{r}_i$ to the set of locations P. After an update has taken place, all future Kriging predictions use the updated dataset.
- The data generating nodes should only be able to update data.
- The honest-but-curious server should only be able to perform Kriging predictions, and should do so without learning the data used in the computation.

We now informally describe the protocol, which proceeds as follows:

1. The data owner $\mathcal{C}$ chooses the variogram model to be used and runs the Outsource algorithm to generate the (protected) dataset to be sent to the server, as well as 'keys' that are issued to authorise a party to update the outsourced dataset, or to perform Kriging queries.
2. Upon receipt of the protected data, the server may run the Setup algorithm to process the data and perform any necessary pre-computation. After this step has been completed, the system is ready to accept queries and perform the associated predictions.

3. The data owner, or an authorised data user (in possession of the query key), may request a Kriging prediction at a specified location by running the Query algorithm to generate a query token Q.

4. This query token Q is sent to the server, who runs the Interpolate algorithm using the processed database to generate an encrypted prediction and an encoding of the Kriging variance (the estimation of the error in the prediction).

5. A party authorised to perform queries may learn the prediction and variance by running the Decrypt algorithm.

6. To dynamically update the outsourced dataset, an authorised party (in possession of the update key) may run the AddRequest algorithm on a specified location $\mathbf{r}'$ and measurement $[z']$, or the DeleteRequest algorithm on a specified location $\mathbf{r}$. These algorithms produce an addition token $\alpha_{\mathbf{r}',[z']}$ or deletion token $\delta_{\mathbf{r}}$, respectively, that is sent to the server.

7. Upon receipt of an addition or deletion token, the server runs the Add or Delete algorithm, respectively, to update the database accordingly.

We note that, throughout execution of all of the algorithms discussed above, the server cannot access the underlying predictions due to the encryption provided by the homomorphic encryption scheme. In this chapter, we assume that any user authorised to generate a Kriging query is also permitted to update the dataset. If this is not the case, then the construction can be modified to include a digital signature computed on any addition or deletion token, where the signing key is contained in the update key (and not the query key). The server should be trusted to reject any tokens that do not have a valid signature (which is the case since we are working in the honest-but-curious model). In this case, only users in possession of the private signature key would be able to update the dataset. We now formalise this discussion.

Definition 6.1 (Private Outsourced Kriging Interpolation Scheme) *A private outsourced Kriging interpolation scheme is made up of the following algorithms:*

- $(UK, QK, C) \xleftarrow{\$} \mathsf{Outsource}(1^\lambda, P, S, \gamma)$*: a probabilistic algorithm run by* $\mathcal{C}$ *which takes as input a security parameter* λ*, the Kriging dataset which is made up of measurement locations* P *and measurement values* S*, and the chosen variogram* γ*. It produces:*

  - *an update key UK that may be used to update the outsourced dataset, which is given to parties authorised by $\mathcal{C}$ to update the dataset,*
  - *a query key QK which may be used to form Kriging queries, which is kept by $\mathcal{C}$, and*
  - *an outsourceable dataset C that may be transmitted to the server.*

- $\mathrm{DB} \leftarrow \mathsf{Setup}(C)$*: a deterministic algorithm run by $\mathcal{S}$ which takes as input the outsourceable dataset C. This algorithm enables $\mathcal{S}$ to perform any necessary processing that will enable it to compute Kriging predictions, and produces a processed outsourced dataset* DB*.*

- $Q \xleftarrow{\$} \mathsf{Query}(QK, \mathbf{r}_0)$*: a probabilistic algorithm run by $\mathcal{C}$ or a data user in $\mathcal{U}$ which takes as input a query key QK and a query location $\mathbf{r}_0 = (x_0, y_0) \in R$ for which a Kriging prediction should be computed. The output is a query token Q which is sent to $\mathcal{S}$.*

- $(\tilde{Z}_0, {\tilde{\sigma}_0}^{*2}) \leftarrow \mathsf{Interpolate}(Q, \mathrm{DB})$*: a deterministic algorithm run by $\mathcal{S}$ that, given a query token Q and the database* DB*, returns an encrypted Kriging interpolation $\tilde{Z}_0$ and the partially computed Kriging variance $\tilde{\sigma_0^{*2}}$.*

- $([z_0^*], [\sigma_0^{*2}]) \leftarrow \mathsf{Decrypt}(QK, \tilde{Z}_0, {\tilde{\sigma}_0}^{*2})$*: a deterministic algorithm run by $\mathcal{C}$ or a user in $\mathcal{U}$ that takes the query key QK, the Kriging results $\tilde{Z}_0$ and ${\tilde{\sigma}_0}^{*2}$ from the server and outputs the Kriging prediction $[z_0^*]$ and the Kriging variance $[\sigma_0^{*2}]$ at the queried location.*

- $\alpha_{\mathbf{r}', [z']} \leftarrow \mathsf{AddRequest}(UK, \mathbf{r}', [z'])$*: a deterministic algorithm run by $\mathcal{C}$, a data user in $\mathcal{U}$ or a data generating node, which takes the update key UK, a location $\mathbf{r}'$, a measurement value $[z']$, and outputs an addition token $\alpha_{\mathbf{r}', [z']}$.*

- $\mathrm{DB}' \leftarrow \mathsf{Add}(\mathrm{DB}, \alpha_{\mathbf{r}', [z'],})$*: a deterministic algorithm run by $\mathcal{S}$ which takes the current outsourced database* DB *and an addition token $\alpha_{\mathbf{r}', [z'],}$ and outputs an updated database* DB′ *representing the Kriging dataset $(P \cup \{\mathbf{r}'\}, S \cup \{[z']\})$.*

- $\delta_{\mathbf{r}} \leftarrow \mathsf{DeleteRequest}(UK, \mathbf{r})$*: a deterministic algorithm run by $\mathcal{C}$, a data user in $\mathcal{U}$ or a data generating node which takes as input a location $\mathbf{r} \in P$ and the update key UK and outputs a deletion token $\delta_{\mathbf{r}}$.*

- $\mathrm{DB}' \leftarrow \mathsf{Delete}(\mathrm{DB}, \delta_{\mathbf{r}})$*: a deterministic algorithm by the server which takes as input the current database* DB*, as well as a deletion token $\delta_{\mathbf{r}}$, and outputs an updated database* DB′ *representing the Kriging dataset $(P \setminus \{\mathbf{r}\}, S \setminus \{[z_{\mathbf{r}}]\})$, where $[z_{\mathbf{r}}] \in S$ is the measurement corresponding to location $\mathbf{r} \in P$ in* DB*.*

Next, we analyse the security requirements of each component within a Kriging system.

1. The measurement values $[z_i] \in S$ are highly sensitive and must be protected at all times.
2. We consider the coordinates $\mathbf{r}_i \in P$ of previous measurements to not be sensitive. This is reasonable since in some applications these locations may be externally observable, such as the case where they are the locations of previous mining activity.
3. The queried location $\mathbf{r}_0$ at which a new prediction should be computed may reveal areas of particular interest to the user. The sensitivity of this relies on the setting and individual user requirements. However, in practice, Kriging queries are often made at *every* location within a region to produce a heat map of a phenomenon (as considered in our toy example), which may limit the sensitivity of individual query locations. Further, the basic assumption of Kriging is that the quality of predictions degrades with distance. Therefore, the best Kriging results will be obtained when the queried location is broadly within the region of prior (observed) measurements.
4. The computed prediction $[z_0^*]$ is highly sensitive as it may form the basis of future decisions and may also be business-critical, and therefore must be protected.
5. The choice of variogram model (without the variogram parameters) may reveal something about the overall trend of the spatial dependencies of the measurements. We assume that this is not particularly sensitive information, and we leave a detailed leakage analysis to future work.
6. The range parameter ρ of the variogram is a constant scaling of the region R denoting the inter-measurement distance h at which the spatial dependency becomes negligible. For distances $h > \rho$, the variogram approaches the variance of the measurements [Wac13], which is represented by the sill $[\nu]$.
7. The nugget effect $[\eta]$ reveals the spatial dependency at very small distances.

We assume that the range is not sensitive (as it merely scales the region R), but that information revealed by the nugget and sill may be sensitive. Even in applications where this direct information on the variance and spatial dependency of measurements is deemed non-sensitive, it may be the case that the variogram parameters are commercially sensitive. These parameters must be chosen empirically to best fit the experimental data, a process which may be time-consuming. The quality of predictions depends on how well the variogram matches the experimental variogram. We summarise our security requirements in Table 6.2.

Data	$\mathbf{r}_i$	$[z_i]$	(x_0, y_0)	$[z_0^*]$	γ model	ρ	$[\nu]$	$[\eta]$
Protection	✗	✓	✗	✓	✗	✗	✓	✓

Table 6.2: Data protection offered by our private outsourced Kriging scheme.

6.4 Our Techniques

In this section we introduce the main concepts used in our construction: the canonical variogram. We show how to factor out the variogram parameters in the Normal equations which allows us to remove these parameters from the outsourced dataset, and use them to recover the final prediction *on the client side*. The crux of our solution for the private outsourcing of Kriging interpolation is to observe how the Kriging prediction varies according to the variogram nugget effect $[\eta]$, the sill $[\nu]$, and range ρ in the non-degenerate case. We define a *canonical* variogram for each variogram model by arbitrarily fixing the parameters $[\eta] = \rho = 1$ and $[\nu] = 0$, although our results clearly translate to other choices.

Since the Kriging process is inherently linear, we show how to 'factor out' the sensitive parameters $[\eta]$ and $[\nu]$ from the variogram to leave just the canonical variogram. Using this result, in combination with an additively homomorphic scheme, an untrusted server can compute a *related* Kriging prediction, and variance, without any knowledge of $[\eta]$, $[\nu]$ and the actual measurements. The variogram parameters can then be used by the client locally to compute the final prediction.

6.4.1 The Canonical Normal Equations

Definition 6.2 (Canonical Variogram) *Let $\gamma(h)$ be a non-degenerate variogram function with nugget effect $[\eta]$, sill $[\nu]$ and range ρ. We define its associated* canonical variogram *as the function $\tilde{\gamma} : \mathbb{R}^{\geq 0} \to \mathbb{R}$ satisfying $\tilde{\gamma}(0) = 0$ and:*

$$\tilde{\gamma}(h) = -\frac{1}{[\nu] - [\eta]}\gamma(\rho h) + \frac{[\nu]}{[\nu] - [\eta]} \quad \text{for } h > 0. \tag{6.2}$$

Note that for any non-degenerate variogram function coming from the parametric variogram models defined in Section 6.2.2, the canonical variogram depends only on the *model* and not

on any of the parameters. Given a Kriging dataset (P, S) of n measurements, a query position $\mathbf{r}_0 \notin P$ and a variogram function γ with nugget effect $[\eta]$, sill $[\nu]$ and range ρ, let $\mathbf{K}\mathbf{w} = \mathbf{v}$ be the corresponding Normal equations as defined in Section 6.2. Our main result is that it is sufficient to consider a canonical version of the Normal equations that depends only on the chosen variogram model, as well as P and the range parameter ρ of γ.

Definition 6.3 (Canonical Normal Equations) *We define the canonical Normal equations as the linear system obtained from the Normal equations* $\mathbf{K}\mathbf{w} = \mathbf{v}$ *by replacing:*

1. *every* $\mathbf{r}_i \in P$ *by* $\mathbf{r}_i/\rho$*, i.e.* $(x_i, y_i) \mapsto (\frac{x_i}{\rho}, \frac{y_i}{\rho})$*,*
2. *the query position* $\mathbf{r}_0$ *by* $\mathbf{r}_0/\rho$*, similarly to above, and*
3. *the variogram* $\gamma(h)$ *by the canonical variogram* $\tilde{\gamma}(h)$*.*

Note that when the positions are scaled we have

$$d(\mathbf{r}_i/\rho, \mathbf{r}_j/\rho) = \frac{1}{\rho} d(\mathbf{r}_i, \mathbf{r}_j).$$

We denote the canonical Normal equations as $\tilde{\mathbf{K}}\tilde{\mathbf{w}} = \tilde{\mathbf{v}}$. The matrix $\tilde{\mathbf{K}}$ is given by:

$$\begin{pmatrix} \tilde{\gamma}(d(\mathbf{r}_1/\rho, \mathbf{r}_2/\rho)) & \tilde{\gamma}(d(\mathbf{r}_1/\rho, \mathbf{r}_2/\rho) & \dots & \tilde{\gamma}(d(\mathbf{r}_1/\rho, \mathbf{r}_n/\rho)) & 1 \\ \tilde{\gamma}(d(\mathbf{r}_2/\rho, \mathbf{r}_1/\rho)) & \tilde{\gamma}(d(\mathbf{r}_2/\rho, \mathbf{r}_2/\rho)) & \dots & \tilde{\gamma}(d(\mathbf{r}_2/\rho, \mathbf{r}_n/\rho)) & 1 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ \tilde{\gamma}(d(\mathbf{r}_n/\rho, \mathbf{r}_1/\rho)) & \tilde{\gamma}(d(\mathbf{r}_n/\rho, \mathbf{r}_2/\rho)) & \dots & \tilde{\gamma}(d(\mathbf{r}_n/\rho, \mathbf{r}_n/\rho)) & 1 \\ 1 & 1 & \dots & 1 & 0 \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)},$$

and the canonical Normal equations are given by the system of equations:

$$\tilde{\mathbf{K}} \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_n \\ \mathbf{w}_{n+1} \end{pmatrix} = \begin{pmatrix} \tilde{\gamma}(d(\mathbf{r}_0/\rho, \mathbf{r}_0/\rho)) \\ \tilde{\gamma}(d(\mathbf{r}_0/\rho, \mathbf{r}_1/\rho)) \\ \vdots \\ \tilde{\gamma}(d(\mathbf{r}_0/\rho, \mathbf{r}_n/\rho)) \\ 1 \end{pmatrix}.$$

Note that, since the canonical variogram is parameterless, the canonical Normal equations involve only the variogram model and the locations in P scaled by $1/\rho$. This observation

allows us to take advantage of the linearity of the Kriging predictor in order to protect the measurements and interpolation value, whilst hiding the sill and nugget parameters $[\nu]$ and $[\eta]$ from the server by storing them locally. We now discuss the solution to the canonical Normal equations in Proposition 6.1.

Proposition 6.1 *Let $\mathbf{K}, \mathbf{K}' \in \mathbb{R}^{(n+1)\times(n+1)}$ be real matrices, and let $\mathbf{v}, \mathbf{v}' \in \mathbb{R}^{n+1}$ be real vectors such that:*

1. *there exists $a, b \in \mathbb{R}$ such that $\mathbf{K}'_{i,j} = a\mathbf{K}_{i,j} + b$ and $v'_i = av_i + b$ for all $1 \leq i, j \leq n$,*
2. *$\mathbf{K}_{i,n+1} = \mathbf{K}_{n+1,i} = \mathbf{K}'_{i,n+1} = \mathbf{K}'_{n+1,i} = \mathbf{v}_{n+1} = \mathbf{v}'_{n+1} = 1$ for all $1 \leq i \leq n$,*
3. *$\mathbf{K}_{n+1,n+1} = \mathbf{K}'_{n+1,n+1} = 0$.*

Then, if $\mathbf{w} \in \mathbb{R}^{n+1}$ satisfies $\mathbf{K}\mathbf{w} = \mathbf{v}$, the vector $\mathbf{w}' \in \mathbb{R}^{n+1}$ defined by:

$$w'_i = w_i \text{ for all } 1 \leq i \leq n, \text{ and}$$
$$w'_{n+1} = aw_{n+1},$$

satisfies $\mathbf{K}'\mathbf{w}' = \mathbf{v}'$.

Proof. Note that:

$$(\mathbf{K}'\mathbf{w}')_i = a\mathbf{v}_i + b\sum_{i=1}^{n} \mathbf{w}_i,$$

for $1 \leq i \leq n$, and we have $(\mathbf{K}'\mathbf{w}')_{n+1} = 1$. Since $\sum_{i=1}^{n} \mathbf{w}_i = 1$ (by the last equation of the system $\mathbf{K}\mathbf{w} = \mathbf{v}$), the result follows. □

This result extends an observation in [Cre92], which states that summing a constant to the variogram does not alter the solutions of the Normal equations, and that such a transformation of the variogram may sometimes be necessary in order to obtain a numerically stable Kriging prediction. We apply this proposition to the Normal equations with $a = -1/([\nu] - [\eta])$ and $b = [\nu]/([\nu] - [\eta])$, and consider the canonical Normal equations. By the definitions of the Kriging prediction and the Kriging variance in Section 6.2, we obtain Proposition 6.2.

Proposition 6.2 *Let $[z_0^*]$ and $[\tilde{z}_0{}^*]$ be the Kriging predictions computed from the Normal and the canonical Normal equations respectively. Denote by $[\sigma_0^{*2}]$ and $\tilde{\sigma}_0{}^{*2}$ the Kriging variance*

associated with each of the predictors. Then:

$$[\tilde{z_0}^*] = [z_0^*] \quad \textit{and} \quad \tilde{\sigma_0}^{*2} = -\frac{1}{[\nu]-[\eta]}[\sigma_0^{*2}] + \frac{[\nu]}{[\nu]-[\eta]}.$$

In the case that the variogram is non-degenerate, the Kriging prediction is independent of the sill $[\nu]$ and nugget $[\eta]$ parameters of the variogram, whilst the range parameter $[\rho]$ scales positions. When we apply a linear transformation to the variogram, the Kriging variance of the obtained Kriging predictor changes according to the same transformation.

6.5 Our Construction

We now outline the operation of each of the algorithms in our scheme. Denote by $\mathcal{H} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ an IND-CPA-secure additive homomorphic encryption scheme. Then:

- $(C, UK, QK) \xleftarrow{\$} \mathsf{Outsource}(1^\lambda, P, S, \gamma)$: If γ is a degenerate variogram function, halt and return $\perp$; in this case, our protocol fails.[4] Otherwise, generate a keypair for the homomorphic encryption scheme:

  $$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda).$$

  Recall that: $P \subseteq \mathbb{R}^2$ is the ordered set of locations $(\mathbf{r}_i)_{i=1}^n$, $S \subseteq \mathbb{R}$ is the ordered set of measurements $([z_i])_{i=1}^n$, and that the variogram γ is made up of three parameters: the nugget $[\eta]$, the sill $[\nu]$ and the range ρ. Let $\tilde{\gamma}$ be the canonical variogram associated with γ, as defined in Section 6.4. Define the update key UK and the query key QK as:

  $$UK = (\mathsf{pk}, \rho) \text{ and } QK = (\mathsf{sk}, [\eta], [\nu], \rho).$$

  To account for the factor of ρ in the input to γ in Equation 6.2, compute the scaled locations:

  $$\tilde{P} = ((x_i/\rho, y_i/\rho))_{i=1}^n.$$

  Finally, encrypt each measurement in S and define the ordered set:

  $$\mathrm{Z} = (\mathsf{Enc}(\mathsf{pk}, [z_i]))_{i=1}^n.$$

  Output $C = (\tilde{P}, \mathrm{Z}, \tilde{\gamma})$, along with UK and QK.

[4] However, if γ is degenerate, the variogram is constant (the so-called 'nugget effect model') and models a purely random variable with no spatial correlation. In this case the prediction is $[z_0^*] = \sum[z_i]/n$ for $r_0 \notin P$ and the variance is $[\sigma_0^{*2}] = n + 1$.

- $\mathrm{DB} \leftarrow \mathsf{Setup}(C)$: Instantiate the matrix $\tilde{\mathbf{K}}$ from the canonical Normal equations using the positions $\mathbf{r}'_i \in \tilde{P}$, $1 \leq i \leq n$, and the canonical variogram $\tilde{\gamma}$. Set

  1. $\tilde{\mathbf{K}}_{i,j} = \tilde{\gamma}(d(\mathbf{r}'_i, \mathbf{r}'_j))$ for $1 \leq i, j \leq n$,
  2. $\tilde{\mathbf{K}}_{n+1,i} = \tilde{\mathbf{K}}_{i,n+1} = 1$ for $i \neq n+1$, and
  3. $\tilde{\mathbf{K}}_{n+1,n+1} = 0$.

  Return $\mathrm{DB} = (\tilde{\mathbf{K}}, C)$.

- $Q \xleftarrow{\$} \mathsf{Query}(\mathbf{r}_0, QK)$: Let $\mathbf{r}_0 = (x_0, y_0)$ and, recalling that ρ is contained within QK, return $Q = (x_0/\rho, y_0/\rho)$.

- $(\tilde{Z}_0, {\tilde{\sigma_0}^*}^2) \leftarrow \mathsf{Interpolate}(Q, \mathrm{DB})$: Recall that $C = (\tilde{P}, \mathrm{Z}, \tilde{\gamma})$. If $Q \in \tilde{P}$, then the exact measurement is contained in the outsourced dataset and no prediction is required. Let j be the index such that $Q = \mathbf{r}_j$, and return $(Z_j, \perp)$, where $\perp$ is a distinguished symbol denoting that the prediction is exact.

  Otherwise, compute the vector $\mathbf{v}$ from the canonical Normal equations using the locations $\mathbf{r}'_i \in \tilde{P}$, the query position Q and the canonical variogram $\tilde{\gamma}$:

  1. $v_i = \tilde{\gamma}(d(Q, \mathbf{r}'_i))$ for $1 \leq i \leq n$, and
  2. $v_{n+1} = 1$.

  Compute the solution $\tilde{\mathbf{w}}$ to the canonical Normal equation $\tilde{\mathbf{K}}\tilde{\mathbf{w}} = \tilde{\mathbf{v}}$ in plaintext. This step essentially computes the Kriging coefficients $\mathbf{w}$ using the canonical variogram and the scaled locations *without* requiring the parameters of the variogram. Then, using the homomorphic property of the encryption, compute the encrypted prediction $\tilde{Z}_0$:

$$\tilde{Z}_0 = \sum_{i=1}^{n} \tilde{w}_i \cdot Z_i \text{ and } {\tilde{\sigma_0}^*}^2 = \tilde{w}_{n+1} + \sum_{i=1}^{n} \tilde{w}_i \cdot \tilde{\gamma}(Q, \mathbf{r}'_i).$$

  Return the encrypted prediction $\tilde{Z}_0$, and the partially computed Kriging variance (error estimation) ${\tilde{\sigma_0}^*}^2$.

- $([z_0^*], [\sigma_0^{*2}]) \leftarrow \mathsf{Decrypt}(\tilde{Z}_0, {\tilde{\sigma_0}^*}^2, QK)$: First decrypt the Kriging prediction:

$$[{\tilde{z_0}^*}] = \mathsf{Dec}(\mathsf{sk}, \tilde{Z}_0),$$

  where sk is contained within QK. Then, if ${\tilde{\sigma_0}^*}^2 = \perp$, set $[\sigma_0^{*2}] = 0$. Else, compute the Kriging variance:

$$[\sigma_0^{*2}] = [\nu] - ([\nu] - [\eta]){\tilde{\sigma_0}^*}^2.$$

This final step essentially adds back in the parameters of the variogram, which were removed for outsourcing, using the result from Proposition 6.2.

- $\alpha_{\mathbf{r}',[z']} \leftarrow \mathsf{AddRequest}(\mathbf{r}', [z'], UK)$: Let $\mathbf{r}_a = \frac{\mathbf{r}'}{\rho}$ and compute the ciphertext:

$$Z_a = \mathsf{Enc}(\mathsf{pk}, [z']),$$

where ρ and pk are contained within UK. Output the addition token:

$$\alpha_{\mathbf{r}',[z']} = (\mathbf{r}_a, Z_a).$$

- $\mathrm{DB}' \leftarrow \mathsf{Add}(\mathrm{DB}, \alpha_{\mathbf{r}',[z']})$: Recall that $\alpha_{\mathbf{r}',[z']} = (\mathbf{r}_a, Z_a)$. Compute the updated dataset: if $\mathbf{r}_a \in \tilde{P}$ then let j be the index such that $\mathbf{r}_j = \mathbf{r}_a$ and modify $Z_j \in \mathrm{Z}$ to be Z_a. Otherwise, set $C' = (\tilde{P} \cup \{\mathbf{r}_a\}, \mathrm{Z} \cup \{Z_a\}, \tilde{\gamma})$. Return the output of $\mathsf{Setup}(C')$.

- $\delta_{\mathbf{r}} \leftarrow \mathsf{DeleteRequest}(\mathbf{r}, UK)$: Return $\delta_{\mathbf{r}} = \mathbf{r}/\rho$.

- $\mathrm{DB}' \leftarrow \mathsf{Delete}(\mathrm{DB}, \delta_{\mathbf{r}})$: If $\delta_{\mathbf{r}} \notin \tilde{P}$, return DB as there is nothing to remove. Otherwise, let j be the index such that $\mathbf{r} = \mathbf{r}_j$ in $\tilde{P}$. Compute the updated dataset $C' = (\tilde{P} \setminus \{\mathbf{r}_j\}, \mathrm{Z} \setminus \{Z_j\}, \tilde{\gamma})$ and return the output of $\mathsf{Setup}(C')$.

6.6 Discussion

The correctness of the scheme is immediate from Proposition 6.2 as well as the correctness of the homomorphic encryption scheme $\mathcal{H}$. These homomorphic properties enable addition and scalar multiplication of ciphertexts, whilst ensuring that the results decrypt appropriately. Proposition 6.2 shows that the Kriging prediction, as well as the Kriging variance, can be computed by applying a linear transformation to the result computed using the canonical (parameterless) variogram. Correctness of the updates is apparent because the addition and deletion tokens format the data in the same way as the original dataset. Since we are using the honest-but-curious model, the server will modify the dataset correctly. The remainder of the update algorithms then simulate a new setup procedure running Setup on a new Kriging dataset from $\mathsf{Outsource}$.

In terms of security, measurement values are always in encrypted form whilst outsourced. Leakage is bounded by the variogram model as well as both the queried and observed locations (scaled by the inverse of range parameter ρ). Therefore, assuming no collusion

between the server and users, the data is confidential from the server. The homomorphic encryption scheme enables the computation to be performed on the measurements whilst they are encrypted, and therefore at no point during the computation is the data revealed to the server.

It is also clear that neither the variogram parameters η and ν, nor any values computed from them, are ever revealed to the server. The final parameter of the variogram, the range ρ, is never explicitly given to the server. However, the server does learn the coordinates (scaled by ρ) of the measurements. Hence, the range could be revealed *if* the server has existing knowledge of the measurement locations. Of the three variogram parameters, we believe that the range is the least sensitive: it reveals how quickly the variogram approaches the sill (i.e. the distance at which the spatial correlation between measurements becomes negligible) but does not reveal anything relating to the measurement values themselves.

Whilst the queried location is revealed in the plain to the server, we note that the mechanism of Tugrul and Polat [TP13] may easily be used to gain a weak form of secrecy: during the Query algorithm, the party carrying out the query may choose $(q-1)$ additional locations from the region, and scale each by ρ. The query token then is made up of q scaled locations, randomly permuted. The server must perform Interpolate for each location, and the client discards all results except the one that it is interested in. Unlike [TP13], we do not require an oblivious transfer protocol, since the querier is authorised to learn as many queries on the dataset as it wishes. However, as in [TP13], the server may guess the location of interest with probability $1/q$ (but cannot learn the prediction at this location). Data generating nodes cannot learn Kriging predictions as they do not have the decryption key and $\mathcal{H}$ is assumed to be IND-CPA secure.

6.6.1 Implementation

We have used the PHE library [PHE16] to implement our construction, and consider the Paillier encryption scheme [Pai99]. We note that the plaintext space of the Paillier encryption scheme is $\mathbb{Z}_n$ for some n. In order to encode the values $[z_i] \in \mathbb{R}$ into this plaintext space, we can use the `phe.encoding` function in PHE. This function takes as input a number $x \in \mathbb{R}$, multiplies by a large constant, usually 10^k for some value of k, and rounds the result to recover $\lceil x10^k \rfloor \in \mathbb{Z}$. The implementation is intended as a proof of concept to evaluate the efficiency of the proposed solution. All code is executed locally on an Amazon EC2 instance

with a 2.5GHz Intel Xeon processor and 1GB memory running Ubuntu 14.04.4. All timings are averaged over 30 iterations, each on a new randomly generated dataset.

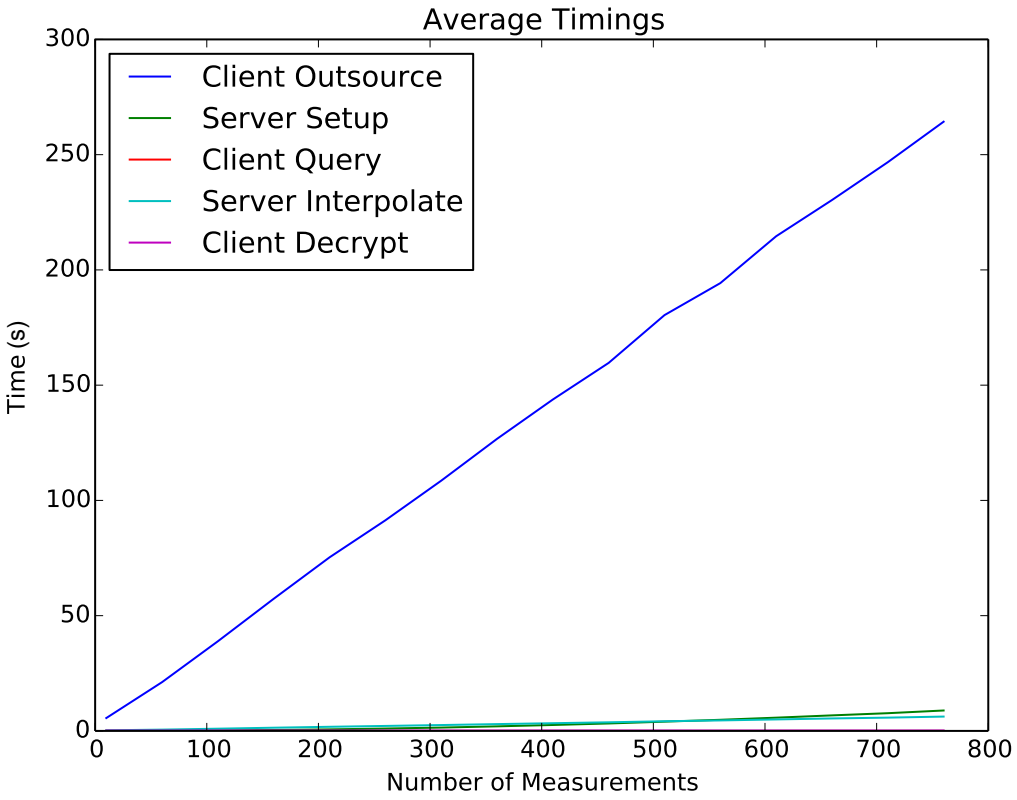


Figure 6.5: Graphs showing the timing costs of each algorithm.

Figures 6.5 and 6.6 give some basic timing results for the implementation our construction. Figure 6.5 shows the algorithm costs. Note that the cost of the Outsource algorithm dominates all others, and this is due to the cost of n encryptions.

Therefore, for clarity, we also present Figure 6.6, which shows the same results with the exclusion of the Outsource algorithm. It can be seen that, with the exception of the (high) one-time cost of Outsource (which may be amortised over many queries), the remaining client-side processes are very efficient. The server must perform quadratic work to perform Setup, but this will be required only during initial setup and when the outsourced dataset is updated. The online workload of the client is very low, whilst the server's online workload is linear in the size of the dataset. Further, we note that the server's online workload greater than the client's workload, which makes outsourcing worthwhile.

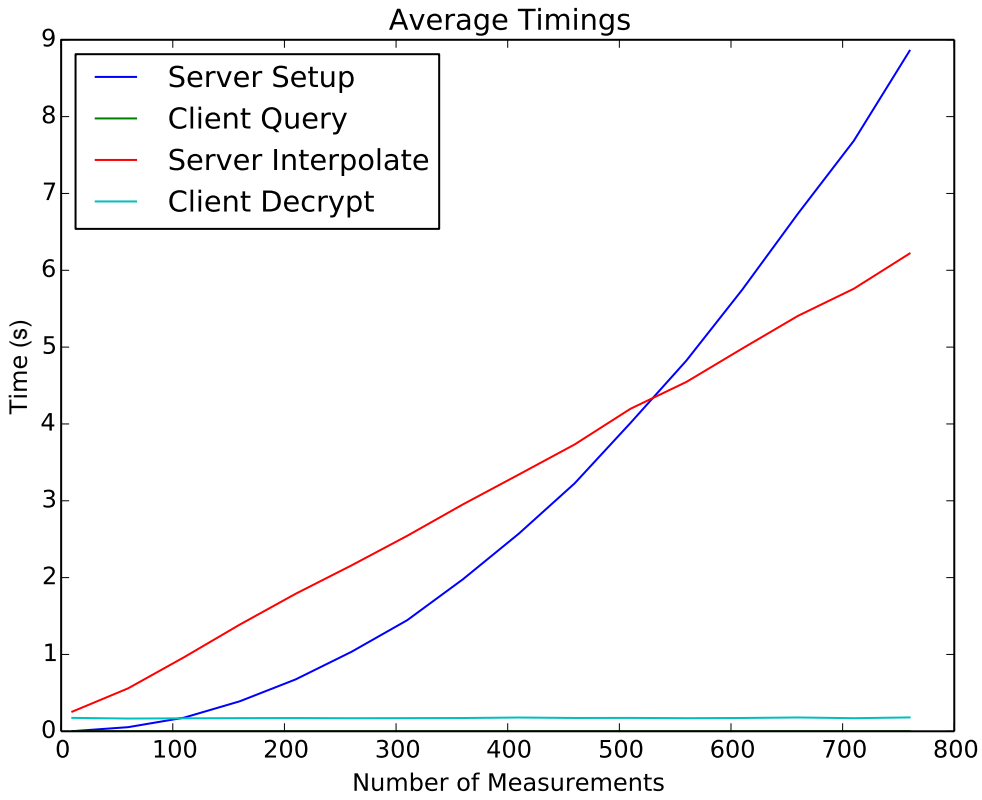


Figure 6.6: Graphs showing the timing costs of each algorithm, excluding Outsource.

We believe that these experiments demonstrate the performance and scalability of our solution. We have considered the well-known Meuse dataset [BMML15], which contains 155 measurements, and we believe this to be reasonable compared to what is used in practice.

6.7 Conclusion

The Kriging interpolation technique describes the best unbiased linear prediction of an observed phenomenon in a geographical region, based on a set of measurements, and it is used in a wide range of applications. In this chapter, we have presented a construction that allows for Kriging interpolation to be securely outsourced to a cloud service provider, such that the measurement values and sensitive variogram parameters are withheld from the server. This solution allows the Kriging interpolation technique to be performed in a privacy-preserving manner, under the assumption that the cloud service provider acts in an honest-but-curious

manner. We have implemented this solution in the PHE library, making use of the Paillier encryption scheme, and have provided some timing results for our implementation.

Chapter 7

Conclusion and Future Work

Contents

7.1 Conclusion

In this thesis we have considered a variety of topics within lattice-based cryptography and homomorphic encryption. Our work has focused around three areas: *security, standardisation, and applications*. We have studied the *security* of lattice-based cryptosystems in Chapters 3, 4, and 5, *standardisation efforts* in Chapters 4 and 5, and *applications* in Chapter 6.

7.1.1 Security

Understanding the security of the Learning with Errors problem is of central importance to the future of lattice-based cryptography. In this thesis, we have considered variants of the *uSVP, dual, decoding, hybrid-dual,* and *hybrid-decoding* attacks which can be used to determine the security of a given set of LWE parameters.

Specifically, in Chapter 3 we introduced a *guess-and-verify decoding* attack (g-v decoding) on small-secret LWE. This approach follows the approach of the hybrid-decoding attack, but considers a more expensive, higher probability, BDD solver. This alters the landscape of trade-offs and we show that, when the BKZ simulator is assumed, aswell as an enumeration cost model for BKZ, our g-v decoding technique outperforms a (non-mitm) variant of the hybrid-decoding attack.

In Chapters 4 and 5 we have considered the security of a variety of different LWE-based parameter sets currently involved in standardisation processes. As part of this work, we have also contributed to the LWE Estimator, to ensure that current state-of-the-art attacks are switched on for binary secrets, and have also released custom code[1].

7.1.2 Standardisation

Standardisation efforts for public-key encryption schemes, digital signature algorithms, and homomorphic encryption schemes based on the Learning with Errors problem are well underway.

The NIST standardisation process, discussed in Chapter 4, aims to standardise a suite of public-key encryption and digital signature algorithms designed for use in an era where quantum computers exist. In this thesis, we have analysed the security of all of the first round submissions against the uSVP and dual attacks (where appropriate). This resulted in a large set of security estimates, which allows for any two schemes to be compared in a fair manner. Moreover, our work highlighted that cost models for BKZ are not *order preserving*. We also provided an update regarding the current state of the NIST standardisation process, including some security estimates for the schemes in the third round.

The `homomorphicencryption.org` standardisation effort, discussed in Chapter 5, aims to standardise LWE-based parameter sets to be used in homomorphic encryption schemes[2]. In our work we have considered potential extensions to the latest variant of the Homomorphic Encryption Security Standard. Specifically, we considered the feasibility and impact of standardising LWE-based parameter sets with a *sparse secret distribution*. We present a variety of parameter sets which balance security requirements with the cost of the expensive

[1] Available at `https://github.com/bencrts/hybrid_attacks` and `https://github.com/estimate-all-the-lwe-ntru-schemes/estimate-all-the-lwe-ntru-schemes.github.io`.

[2] The consortium is also in the process of standardising an API for use in homomorphic encryption schemes.

bootstrapping step.

7.1.3 Applications

Applications of homomorphic encryption schemes are plentiful and ever-increasing. In Chapter 6, we have considered the outsourcing of an interpolation algorithm called *Kriging* to an honest-but-curious cloud server. In our work, we showed how to "factor out" sensitive parameters in order to allow the system of equations used to determine the weights to be solved by the server in a secure manner. When used in combination with an additively homomorphic encryption scheme, this allows for the weights to be applied to the encrypted measurement values, and for the sum to be computed securely. Finally, the sensitive parameters are re-applied on the client-side, to allow for the interpolation value to be determined.

7.2 Future Work

In this section we conclude this thesis with some comments on potential future work.

7.2.1 Security

Cryptanalysis of lattice-based cryptography is a fast-moving field. Further work is required to accurately estimate the cost of the hybrid-decoding attack, and this includes verification of the various heuristics considered in the hybrid-decoding attack. Future areas to explore also includes considering the impact of *quantum enumeration algorithms* on the concrete running time of the hybrid-decoding attack, and studying *sieving-based BDD solvers* in more detail. Moreover, an accurate analysis of the meet-in-the-middle probability for the g-v decoding approach would allow for a direct comparison with the hybrid-decoding attack. Further work is also needed to fully understand the effects of using a sparse secret on the concrete security of LWE.

7.2.2 Standardisation

As the NIST standardisation process continues into the final rounds, it is important for the literature to attempt to converge onto BKZ cost models for both the enumeration and sieving cases. Once this has been achieved, estimating attack complexities becomes a more simple process. Moreover, it is also important for the *memory cost* of cryptanalytic attacks to be considered, as the majority of cryptanalytic arguments in the submissions to the NIST standardisation process focus solely on the time cost of attacks. As an example, we have seen that *hybrid-decoding attacks*, as outlined in Chapter 3, can outperform the dual and uSVP attacks. However, these attacks require exponential memory.

In terms of homomorphic encryption standardisation, the methodology in the current variant of the HE Standard relies on the LWE Estimator. However we note that the LWE Estimator has a number of limitations, including outdated decoding estimates and lack of support for hybrid attacks. Moreover, the attack landscape for sparse-secret LWE is fast moving, with several new contributions emerging in recent years. Further work is needed to ensure that any parameter sets standardised for homomorphic encryption schemes are analysed under hybrid attacks.

7.2.3 Applications

Our proposed construction for the private outsourcing of Kriging interpolation may be extended in several ways. For example, we could consider extending our protocol to protect the locations of the query points. This can be achieved by increasing interactivity, communication complexity, and client computation in the query process. However, this approach requires the server to compute square roots and natural exponentials over encrypted data (in an efficient manner) which remains an open problem.

Bibliography

[ABF+20] Martin R. Albrecht, Shi Bai, Pierre-Alain Fouque, Paul Kirchner, Damien Stehlé, and Weiqiang Wen. Faster enumeration-based lattice reduction: Root hermite factor $k^{1/(2k)}$ time $k^{k/8+o(k)}$. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 186–212, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

[ACC+17] David Archer, Lily Chen, Jung Hee Cheon, Ran Gilad-Bachrach, Roger A. Hallman, Zhicong Huang, Xiaoqian Jiang, Ranjit Kumaresan, Bradley A. Malin, Heidi Sofia, Yongsoo Song, and Shuang Wang. Applications of homomorphic encryption. Technical report, HomomorphicEncryption.org, Redmond WA, USA, July 2017.

[ACC+18] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[ACD+18] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the LWE, NTRU schemes! In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 351–367, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany.

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

[ACF+17] James Alderman, Benjamin R. Curtis, Oriol Farràs, Keith M. Martin, and Jordi Ribes-González. Private outsourced kriging interpolation. 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, *Financial Cryptography and Data Security*, pages 75–90, Cham, 2017. Springer International Publishing.

[ACFP14] Martin R. Albrecht, Carlos Cid, Jean-Charles Faugère, and Ludovic Perret. Algebraic algorithms for LWE. Cryptology ePrint Archive, Report 2014/1018, 2014. `http://eprint.iacr.org/2014/1018`.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.

[ACW20] Martin R. Albrecht, Benjamin R. Curtis, and Thomas Wunderer. Exploring trade-offs in batch bounded distance decoding. In Kenneth G. Paterson and Douglas Stebila, editors, *Selected Areas in Cryptography – SAC 2019*, pages 467–491, Cham, 2020. Springer International Publishing.

[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.

[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: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 429–445, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[AFG13] Martin R Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving lwe by reduction to unique-svp. In *International Conference on Information Security and Cryptology*, pages 293–310. Springer, 2013.

[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: 38th International Colloquium on Automata, Languages and Programming, Part I*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415, Zurich, Switzerland, July 4–8, 2011. Springer, Heidelberg, Germany.

[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, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 297–322, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd Annual ACM Symposium on Theory of Computing*, pages 601–610, Crete, Greece, July 6–8, 2001. ACM Press.

[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, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 103–129, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[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.

[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`.

[BAA+19] 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, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

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

[BCL+18] Ahmad Al Badawi, Jin Chao, Jie Lin, Chan Fook Mun, Sim Jun Jie, Benjamin Hong Meng Tan, Xiao Nan, Khin Mi Mi Aung, and Vijay Ramaseshan Chandrasekhar. The alexnet moment for homomorphic encryption: Hcnn, the first homomorphic cnn on encrypted data with gpus. *arXiv preprint arXiv:1811.00778*, 2018.

[BCLv17] 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`.

[BCLv19] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[BCLv20] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU Prime. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[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 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 10–24, Arlington, VA, USA, January 10–12, 2016. ACM-SIAM.

[BDH+17] Michael Brenner, Wei Dai, Shai Halevi, Kyoohyung Han, Amir Jalali, Miran Kim, Kim Laine, Alex Malozemoff, Pascal Paillier, Yuriy Polyakov, Kurt Rohloff, Erkay Savaş, and Berk Sunar. A standard api for rlwe-based homomorphic encryption. Technical report, HomomorphicEncryption.org, Redmond WA, USA, July 2017.

[Ber17] Daniel J. Bernstein. Table of ciphertext and key sizes for the NIST candidate algorithms. Available at `https://groups.google.com/a/list.nist.gov/d/msg/pqc-forum/1lDNio0sKq4/xjqy4K6SAgAJ`, 2017.

[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`.

[Ber20] Daniel J. Bernstein. Discussion of the 0.396β cost model. Available at `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/tsGTo5n3Zf4/m/fmMG2rlFCAAJ`, 2020.

[BG14] Shi Bai and Steven D. Galbraith. Lattice decoding attacks on binary LWE. In Willy Susilo and Yi Mu, editors, *ACISP 14: 19th Australasian Conference on Information Security and Privacy*, volume 8544 of *Lecture Notes in Computer Science*, pages 322–337, Wollongong, NSW, Australia, July 7–9, 2014. Springer, Heidelberg, Germany.

[BGL+18] Sauvik Bhattacharya, Óscar García-Morchón, Thijs Laarhoven, Ronald Rietman, Markku-Juhani O. Saarinen, Ludo Tolhuizen, and Zhenfei Zhang. Round5: Compact and fast post-quantum public-key encryption. Cryptology ePrint Archive, Report 2018/725, 2018. `https://eprint.iacr.org/2018/725`.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of cryptography conference*, pages 325–341. Springer, 2005.

[BGPW16] 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 David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16: 8th International Conference on Cryptology in Africa*, volume 9646 of *Lecture Notes in Computer Science*, pages 24–43, Fes, Morocco, April 13–15, 2016. Springer, Heidelberg, Germany.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):13, 2014.

[BLP+13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 575–584, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[BMML15] Peter A Burrough, Rachael McDonnell, Rachael A McDonnell, and Christopher D Lloyd. *Principles of geographical information systems*. Oxford University Press, 2015.

[BMTPH20] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. Cryptology ePrint Archive, Report 2020/1203, 2020. `https://eprint.iacr.org/2020/1203`.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[Bra16] Matt Braithwaite. Experimenting with post-quantum cryptography. *Google Security Blog*, 2016. `https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html`.

[BSW18] Shi Bai, Damien Stehlé, and Weiqiang Wen. Measuring, simulating and exploiting the head concavity phenomenon in BKZ. In Thomas Peyrin and Steven Galbraith, editors, *Advances in Cryptology – ASIACRYPT 2018, Part I*, volume 11272 of *Lecture Notes in Computer Science*, pages 369–404, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

[CCD+19] Hao Chen, Ilaria Chillotti, Yihe Dong, Oxana Poburinnaya, Ilya P. Razenshteyn, and M. Sadegh Riazi. SANNS: scaling up secure approximate k-nearest neighbors search. *CoRR*, abs/1904.02033, 2019.

[CCS19a] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 34–54, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[CCS19b] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Multi-key homomorphic encryption from TFHE. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology – ASIACRYPT 2019, Part II*, volume 11922 of *Lecture Notes in Computer Science*, pages 446–472, Kobe, Japan, December 8–12, 2019. Springer, Heidelberg, Germany.

[CD99] Jean-Paul Chilès and Pierre Delfiner. Multivariate methods. *Geostatistics: Modeling Spatial Uncertainty, Second Edition*, pages 299–385, 1999.

[CDKS19] Hao Chen, Wei Dai, Miran Kim, and Yongsoo Song. Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019: 26th Conference on Computer and Communications Security*, pages 395–412. ACM Press, November 11–15, 2019.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 3–33, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[CH18] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved FHE bootstrapping. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 315–337, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[Che13] Yuanmi Chen. *Réduction de réseau et sécurité concrète du chiffrement complètement homomorphe*. PhD thesis, Paris 7, 2013.

[CHHS19] J. H. Cheon, M. Hhan, S. Hong, and Y. Son. A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret lwe. *IEEE Access*, 7:89497–89506, 2019.

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 360–384, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[CKLS18] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! A practical post-quantum public-key encryption from LWE and LWR. In Dario Catalano and Roberto De Prisco, editors, *SCN 18: 11th International Conference on Security in Communication Networks*, volume 11035 of *Lecture Notes in Computer Science*, pages 160–177, Amalfi, Italy, September 5–7, 2018. Springer, Heidelberg, Germany.

[CKR+20] Hao Chen, Miran Kim, Ilya Razenshteyn, Dragos Rotaru, Yongsoo Song, and Sameer Wagh. Maliciously secure matrix multiplication with applications to private deep learning. Cryptology ePrint Archive, Report 2020/451, 2020. `https://eprint.iacr.org/2020/451`.

[Cla16] CLARUS: User centered privacy and security in the cloud. `http://clarussecure.eu`, 2016. Accessed: 11/12/2016.

[CLP19] Anamaria Costache, Kim Laine, and Rachel Player. Evaluating the effectiveness of heuristic worst-case noise analysis in fhe. Cryptology ePrint Archive, Report 2019/493, 2019. `https://eprint.iacr.org/2019/493`.

[CN11] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: Better lattice security estimates. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[Con20] Zama. Concrete operates on ciphertexts rapidly by extending TFHE. https://github.com/zama-ai/concrete, 2020. Github repository, commit `fff309e`.

[CP19] Benjamin R. Curtis and Rachel Player. On the feasibility and impact of standardising sparse-secret lwe parameter sets for homomorphic encryption. In *Proceedings of the 7th ACM Workshop on Encrypted Computing and Applied Homomorphic Cryptography*, WAHC'19, page 1–10, New York, NY, USA, 2019. Association for Computing Machinery.

[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`.

[Cre92] Noel Cressie. Statistics for spatial data. *Terra Nova*, 4(5):613–617, 1992.

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[DKR+20] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[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`.

[DKRV19] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. SABER. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[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.

[Dua20] Duality technologies. corporate website. https://dualitytech.com/, 2020. Accessed July 22nd 2020.

[EJK20] Thomas Espitau, Antoine Joux, and Natalia Kharchenko. On a hybrid approach to solve binary-lwe. Cryptology ePrint Archive, Report 2020/515, 2020. https://eprint.iacr.org/2020/515.

[El 17] 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.

[Env20] ENVEIL. corporate website. https://www.enveil.com/, 2020. Accessed July 22nd 2020.

[Est20] The LWE estimator. https://bitbucket.org/malb/lwe-estimator, 2020.

[Eur07] European Parliament. Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an infrastructure for spatial information in the European Community (INSPIRE). *Official Journal of the European Union*, 50(L108), 2007.

[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.

[FPL20] The FPLLL Development Team. FPLLL, a lattice reduction library. Available at https://github.com/fplll/fplll, 2020.

[Fuj17] Ryo Fujita. Table of underlying problems of the NIST candidate algorithms. Available at https://groups.google.com/a/list.nist.gov/d/msg/pqc-forum/1lDNio0sKq4/7zXvtfdZBQAJ, 2017.

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

[GBDL+16] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to

encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pages 201–210, 2016.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

[GJMS17] Qian Guo, Thomas Johansson, Erik Mårtensson, and Paul Stankovski. Coded-BKW with sieving. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 323–346, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[GJS15] Qian Guo, Thomas Johansson, and Paul Stankovski. Coded-BKW: Solving LWE using lattice codes. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 23–42, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious RAMs. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 182–194, New York City, NY, USA, May 25–27, 1987. ACM Press.

[GvW17] Florian Göpfert, Christine van Vredendaal, and Thomas Wunderer. A hybrid lattice basis reduction and quantum search attack on LWE. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 184–202, Utrecht, The Netherlands, June 26–28 2017. Springer, Heidelberg, Germany.

[GZB+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`.

[GZB+19] Oscar Garcia-Morchon, Zhenfei Zhang, Sauvik Bhattacharya, Ronald Rietman, Ludo Tolhuizen, Jose-Luis Torre-Arce, Hayo Baan, Markku-Juhani O. Saarinen, Scott Fluhrer, Thijs Laarhoven, and Rachel Player. Round5. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[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`.

[Ham19] Mike Hamburg. Three Bears. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[HEA20] SNU cryptography lab. HEAAN. `https://github.com/snucrypto/HEAAN`, 2020. Github repository, commit `48a1ed`.

[HEl20] HElib Development Team. HElib. `https://github.com/homenc/HElib/`, 2020. commit ID `507ceff`.

[HGSW03] Nick Howgrave-Graham, Joseph Silverman, and William Whyte. A meet-in-the-middle attack on an ntru private key. NTRU Cryptosystems Technical Report, 2003.

[HHC19] Kyoohyung Han, Minki Hhan, and Jung Hee Cheon. Improved homomorphic discrete fourier transforms and FHE bootstrapping. *IEEE Access*, 7:57361–57370, 2019.

[How07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[HPS96] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A new high speed public key cryptosystem, 1996. Draft Distributed at Crypto'96, available at `http://web.securityinnovation.com/hubfs/files/ntru-orig.pdf`.

[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.

[HS20] Shai Halevi and Victor Shoup. Design and implementation of helib: a homomorphic encryption library. Cryptology ePrint Archive, Report 2020/1481, 2020. https://eprint.iacr.org/2020/1481.

[Ida19] Idash security and privacy workshop, 2019. http://www.humangenomeprivacy.org/2019/.

[Ing16] InGeoCloudS: inspired geo-data cloud services. https://www.ingeoclouds.eu/, 2016. Accessed: 11/12/2016.

[JKLS18] Xiaoqian Jiang, Miran Kim, Kristin E. Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018: 25th Conference on Computer and Communications Security*, pages 1209–1222, Toronto, ON, Canada, October 15–19, 2018. ACM Press.

[Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems. In *15th Annual ACM Symposium on Theory of Computing*, pages 193–206, Boston, MA, USA, April 25–27, 1983. ACM Press.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, pages 415–440, 1987.

[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, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 43–62, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[KH14] Michael Kuntz and Marco Helbich. Geostatistical mapping of real estate prices: an empirical comparison of kriging and cokriging. *International Journal of Geographical Information Science*, 28(9):1904–1921, 2014.

[Kle09] Jack P.C. Kleijnen. Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707 – 716, 2009.

[Kri51] DG Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[KSK+18] Andrey Kim, Yongsoo Song, Miran Kim, Keewoo Lee, and Jung Hee Cheon. Logistic regression model training based on the approximate homomorphic encryption. Cryptology ePrint Archive, Report 2018/254, 2018. `https://eprint.iacr.org/2018/254`.

[KSW+18] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, and Xiaoqian Jiang. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR Med Inform*, 6(2):e19, Apr 2018.

[Laa15a] Thijs Laarhoven. *Search problems in cryptography: From fingerprinting to lattice sieving*. PhD thesis, Eindhoven University of Technology, 2015.

[Laa15b] Thijs Laarhoven. Sieving for shortest vectors in lattices using angular locality-sensitive hashing. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 3–22, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[Laa16] Thijs Laarhoven. Sieving for closest lattice vectors (with preprocessing). In Roberto Avanzi and Howard M. Heys, editors, *SAC 2016: 23rd Annual International Workshop on Selected Areas in Cryptography*, volume 10532 of *Lecture Notes in Computer Science*, pages 523–542, St. John's, NL, Canada, August 10–12, 2016. Springer, Heidelberg, Germany.

[LDK+17] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[LDK+19] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[LDK+20] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[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`.

[LLJ+19] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[LLL82] A.K. Lenstra, Jr. Lenstra, H.W., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[LM20] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. Cryptology ePrint Archive, Report 2020/1533, 2020. `https://eprint.iacr.org/2020/1533`.

[LMv14] Thijs Laarhoven, Michele Mosca, and Joop van de Pol. Finding shortest lattice vectors faster using quantum search. Cryptology ePrint Archive, Report 2014/907, 2014. `http://eprint.iacr.org/2014/907`.

[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.

[LN13] Mingjie Liu and Phong Q. Nguyen. Solving BDD by enumeration: An update. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 293–309, San Francisco, CA, USA, February 25 – March 1, 2013. Springer, Heidelberg, Germany.

[Lot18] Script for fitting lotus enumeration model. `https://github.com/estimate-all-the-lwe-ntru-schemes/estimate-all-the-lwe-ntru-schemes.github.io/blob/master/lotus_fitting.sage`, 2018.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[LS15] Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptography*, 75(3):565–599, 2015.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219–1234, New York, NY, USA, May 19–22, 2012. ACM Press.

[Mic18] Daniele Micciancio. On the hardness of lwe with binary error. Technical report, UCSD, February 2018. `http://cseweb.ucsd.edu/~daniele/papers/BinLWE.pdf`.

[Mic20] Microsoft Research. SEAL. `https://github.com/Microsoft/SEAL`, 2020. Github repository, commit `c9af10d`.

[Moo17] Dustin Moody. The NIST post quantum cryptography "competition". Available at `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/asiacrypt-2017-moody-pqc.pdf`, 2017.

[Moo19] Dustin Moody. NIST announcement of 2nd round candidates. Available at `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/bBxcfFFUsxE`, 2019.

[Moo20] Dustin Moody. NIST announcement of 3rd round candidates. Available at `https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/0ieuPB-b8eg`, 2020.

[MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum*

Cryptography, pages 147–191. Springer, Heidelberg, Berlin, Heidelberg, New York, 2009.

[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.

[MW15] Daniele Micciancio and Michael Walter. Fast lattice point enumeration with minimal overhead. In Piotr Indyk, editor, *26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 276–294, San Diego, CA, USA, January 4–6, 2015. ACM-SIAM.

[NAB+17] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo 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`.

[NAB+19] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo 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, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[NAB+20] Michael Naehrig, Erdem Alkim, Joppe Bos, Léo 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, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[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.

[Nat17] National Institute of Standards and Technology. Performance testing of the NIST candidate algorithms. Available at https://drive.google.com/file/d/1g-l0bPa-tReBD0Frgnz9aZXpO06PunUa/view, 2017.

[Ngu18] P. Nguyen, 2018. Comment on PQC forum. Available at https://groups.google.com/a/list.nist.gov/forum/#!topic/pqc-forum/nZBIBvYmmUI.

[OLBC10] Frank WJ Olver, Daniel W Lozier, Ronald F Boisvert, and Charles W Clark. *NIST handbook of mathematical functions hardback and CD-ROM*. Cambridge university press, 2010.

[PAA+17] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo 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.

[PAA+19] Thomas Poppelmann, Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Peter Schwabe, Douglas Stebila, Martin R. Albrecht, Emmanuela Orsini, Valery Osheter, Kenneth G. Paterson, Guy Peer, and Nigel P. Smart. NewHope. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[Pal20] PALISADE development team. PALISADE. https://gitlab.com/palisade/palisade-release, 2020. GitLab repository, commit 7eec669e.

[Pei16] Chris Peikert. How (not) to instantiate ring-LWE. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16: 10th International Conference on Security in Communication Networks*, volume 9841 of *Lecture Notes in Computer Science*, pages 411–430, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany.

[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.

[PFH+19] 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, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions.

[PFH+20] 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, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions.

[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.

[PHE16] python-paillier: a library for partially homomorphic encryption in python, Data61|CSIRO. https://github.com/NICTA/python-paillier, 2016. Accessed: 11/12/2016.

[Pla19] Rachel Player. *Parameter selection in Lattice-based cryptography.* PhD thesis, Royal Holloway University of London, 2019.

[PS16] Chris Peikert and Sina Shiehian. Multi-key FHE from LWE, revisited. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 217–238, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

[PYK20] Geostat framework. pykrige. https://github.com/GeoStat-Framework/PyKrige, 2020.

[RAD78] R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations on Secure Computation, Academia Press*, pages 169–179, 1978.

[RDB94] Richard E. Rossi, Jennifer L. Dungan, and Louisa R. Beck. Kriging in the shadows: Geostatistical interpolation for remote sensing. *Remote Sensing of Environment*, 49(1):32 – 40, 1994.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[S+20] William Stein et al. *Sage Mathematics Software Version 8.0.* The Sage Development Team, 2020. `http://www.sagemath.org`.

[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`.

[SAB+17] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2017. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions`.

[SAB+19] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[SAB+20] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[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.

[SC19] Yongha Son and Jung Hee Cheon. Revisiting the hybrid attack on sparse secret lwe and application to he parameters. In *Proceedings of the 7th ACM Workshop on Encrypted Computing and Applied Homomorphic Cryptography*, WAHC'19, page 11–20, New York, NY, USA, 2019. Association for Computing Machinery.

[Sch03] Claus Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *STACS 2003*, pages 145–156, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[SE94] Claus-Peter Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–199, 1994.

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

[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.

[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.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

[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.

[TFH20] TFHE Development Team. TFHE: Fast fully homomorphic encryption library. `https://tfhe.github.io/tfhe/`, 2020. Github repository, commit `a085efe`.

[TP13] Bulent Tugrul and Huseyin Polat. Estimating kriging-based predictions with privacy. *International Journal of Innovative Computing, Information and Control, Accepted for publication*, 2013.

[TP14] Bulent Tugrul and Huseyin Polat. Privacy-preserving kriging interpolation on partitioned data. *Knowledge-Based Systems*, 62:38–46, 2014.

[Unb20] Unbound tech. corporate website. `https://www.unboundtech.com/`, 2020. Accessed July 22nd 2020.

[Vai11] Vinod Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 5–16. IEEE, 2011.

[Wac13] Hans Wackernagel. *Multivariate geostatistics: an introduction with applications*. Springer Science & Business Media, 2013.

[Wun18] Thomas Wunderer. *On the Security of Lattice-Based Cryptography Against Lattice Reduction and Hybrid Attacks*. PhD thesis, Technische Universität, Darmstadt, 2018.

[Wun19] Thomas Wunderer. A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack. *J. Mathematical Cryptology*, 13(1):1–26, 2019.

[Wun20] Thomas Wunderer. Hybrid attack scripts. `https://github.com/lducas/LatRedHybrid`, 2020.

[Yao86] A. C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, 1986.

[YD17] Yang Yu and Léo Ducas. Second order statistical behavior of LLL and BKZ. In Carlisle Adams and Jan Camenisch, editors, *SAC 2017: 24th Annual International Workshop on Selected Areas in Cryptography*, volume 10719 of *Lecture Notes in Computer Science*, pages 3–22, Ottawa, ON, Canada, August 16–18, 2017. Springer, Heidelberg, Germany.

[Zam20] Zama. corporate website. `https://zama.ai/`, 2020. Accessed July 22nd 2020.

[ZCH+19] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, William Whyte, John M. Schanck, Andreas Hulsing, Joost Rijneveld, Peter Schwabe, and Oussama Danba. NTRUEncrypt. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions`.

[ZCH+20] Zhenfei Zhang, Cong Chen, Jeffrey Hoffstein, William Whyte, John M. Schanck, Andreas Hulsing, Joost Rijneveld, Peter Schwabe, Oussama Danba, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. NTRUEncrypt. Technical report, National Institute of Standards and Technology, 2020. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`.

[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`.

[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`.

[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`.