Some Algorithms for Learning with Errors

Robert Fitzpatrick

A Thesis submitted for the degree of Doctor of Philosophy

Information Security Group

Royal Holloway and Bedford New College, University of London

August 21, 2014

Acknowledgements

This thesis is dedicated to my wife, Yu-Hsuan Tai, for her patience, understanding, encouragement and unceasing support during the past three years. My deepest thanks and appreciation go to my supervisor, Carlos Cid, for guiding me along the Ph.D path and for providing invaluable counsel on associated matters. My thanks also go to Maire O'Neill and CSIT, Queens University Belfast for providing the funding to make possible this research. Many thanks go to my advisor, Keith Martin. Special thanks go to Martin Albrecht, with whom I had the good fortune to begin collaborating towards the end of my first year and from whom I learned much about how research is done. I would also like to thank my collaborators in work both published and unpublished: Ludovic Perret, Jean-Charles Faugére, Nigel Smart, Daniel Cabarcas, Florian Göpfert, Michael Schneider, Keita Xagawa and Todo Yosuke.

To my parents, Fred and Mary, for my upbringing and the ethics received and also for their support and understanding during the last 3 years and long before. To Po-Wah Yau, for many interesting conversations. My thanks also go to Bo-Yin Yang and the Institute of Information Science at Academia Sinica, Taipei, for hosting me on multiple occasions during my time in Taiwan.

Declaration

I, Robert Fitzpatrick, hereby declare that this thesis and the work presented in it is entirely my own. Parts of this thesis are based on the following papers:

- On the Complexity of the BKW Algorithm on LWE - Designs, Codes and Cryptography, Springer-Verläg, October 2013
- Lazy Modulus Switching for the BKW Algorithm on LWE - PKC 2014, Buenos Aires, Argentina, March 2014
- Practical Cryptanalysis of a Public-Key Encryption Scheme based on New Multivariate Quadratic Assumptions - PKC 2014, Buenos Aires, Argentina, March 2014
- On the Efficacy of Solving LWE by Reduction to Unique-SVP - ICISC 2013, Seoul, South Korea, November 2013

In the case of joint authorship, all authors contributed equally.

Signed:

Robert Fitzpatrick

Abstract

The Learning with Errors (LWE) problem, introduced in 2005 by Regev, is a generalisation to larger modulii of the Learning Parity with Noise problem and has proven to be a remarkably versatile primitive for the construction of cryptographic schemes. With an average-to-worst-case reduction from (assumed) hard lattice problems, the LWE problem is highly attractive as a basis for post-quantum secure constructions. However, the concrete hardness of the LWE problem remains little investigated, with parameter choices in the literature tending to be conservative or completely absent as a result. In this work we study the various known approaches for solving LWE instances and develop new and modified approaches. Two main approaches exist - those employing lattice basis reduction and combinatorial approaches. For the latter we present the first analysis and adaptation of the BKW algorithm to the LWE case, illustrating that this approach is asymptotically more efficient than known lattice-based approaches with surprisingly low 'crossover' dimension'. We also present a modification of the BKW algorithm, optimised for LWE instances in which the secret vector entries are unusually small, demonstrating this modified algorithm to be significantly more efficient than previously-known approaches. We additionally examine the recently-proposed public-key scheme of Huang, Liu and Yang and show that, by viewing this scheme as a weak LWE-like problem we can break all challenges in a matter of hours. With regard to algorithms which rely on lattice basis reduction, we present the first experimental study of the efficacy of applying Kannan's embedding approach, showing that this approach out-performs the classical distinguishing approach in the high-advantage regime.

Contents

1 Introduction **14**

1.1 Foreword 14

1.2 Post-Quantum Cryptography 17

1.3 Notation and Preliminary Definitions 18

1.4 Lattices and Related Hard Problems 18

1.4.1 Lattice Reduction. 22

1.4.2 Experiments on Random Lattices and q-ary Lattices 25

1.5 Noise Models 27

1.6 Small Integer Solutions and Learning with Errors 28

1.6.1 Learning Parities with Noise 28

1.6.2 Learning with Errors 29

1.6.3 Reduction from Hard Lattice Problems 30

1.7 Solving LWE using Lattice Reduction 33

1.8 Generating LWE Instances 36

2 Naïve Algorithms for LWE and the BKW Algorithm **37**

2.1 Introduction 38

2.2 Preliminaries 41

2.3 The BKW Algorithm 42

2.3.1 Sample Reduction 44

2.3.2 Hypothesis Testing 51

2.3.3 Back-Substitution . . . 58
2.3.4 Complexity of BKW . . . 58
2.4 Applications . . . 61
2.4.1 Regev's Original Parameters . . . 62
2.4.2 Lindner and Peikert's Parameters . . . 62
2.4.3 Albrecht et al.'s Polly-Cracker . . . 63
2.5 Comparison with Alternative Approaches . . . 63
2.5.1 Short Integer Solutions: Lattice Reduction . . . 64
2.5.2 Short Integer Solutions: Combinatorial . . . 66
2.5.3 Bounded Distance Decoding: Lattice Reduction . . . 67
2.6 Experimental Results . . . 68
2.7 Magnification and Independence of Noise . . . 70
2.7.1 Balls, Bins and Markov Chains . . . 70
2.7.2 The General Case . . . 74
2.7.3 Putting it all together . . . 75
2.7.4 Implications . . . 78
2.7.5 Mitigation . . . 78
2.8 Closing Remarks . . . 79

3 Lazy Modulus Switching for the BKW Algorithm 80
3.1 Introduction . . . 80
3.1.1 Organisation of the Chapter and Main Results . . . 81
3.2 Modifying BKW: Lazy Modulus Switching . . . 83
3.2.1 The Basic Idea . . . 84
3.2.2 Sample Reduction for Short Secrets . . . 85
3.2.3 Selecting p . . . 86
3.3 Improved Algorithm: Stunting Growth by Unnatural Selection . . . 90
3.3.1 The Basic Idea . . . 91
3.3.2 Algorithms . . . 92

3.3.3 Selecting p . . . 94
3.4 Complexity . . . 98
3.5 Implementation . . . 100
3.5.1 Correctness of Algorithm 6: Constructing $\mathbf{M}$ . . . 101
3.6 Parameters . . . 102
3.6.1 Independence Assumption . . . 107
3.7 Closing Remarks . . . 109
3.8 Justification of Assumption 4 . . . 109

4 Applying Kannan's Embedding Approach to LWE 113
4.1 Introduction . . . 113
4.1.1 Related Work . . . 114
4.1.2 Organisation . . . 114
4.2 Background and Notation . . . 114
4.2.1 Concrete Hardness of uSVP . . . 116
4.3 The Embedding Approach . . . 117
4.3.1 Construction of Embedding Lattices . . . 117
4.3.2 On the Determination of τ when $t = \lceil \|\mathbf{e}\| \rceil$ . . . 118
4.3.3 On the Determination of τ when $t < \lceil \|\mathbf{e}\| \rceil$ . . . 120
4.4 Application to LWE and comparisons . . . 121
4.4.1 Regev's Parameters . . . 124
4.4.2 Lindner and Peikert's Parameters . . . 124
4.5 Limits of the Embedding Approach . . . 126
4.5.1 Comparisons . . . 128
4.5.2 Closing Remarks . . . 129

5 Practical Cryptanalysis of a MQPKC after Reduction to LWE 130
5.1 Introduction . . . 130
5.1.1 Overview of the Results . . . 131
5.2 A New Multivariate Quadratic Assumption and LWE with Small Secrets 133

5.3 Analysis of the Parameters 136
5.4 Improved Embedding Attack 138
5.4.1 Estimation of the Expected Gap 140
5.5 Practical Attacks against HLY Challenges 141
5.6 Closing Remarks 148

A An LWE Instance Generator **162**
A.1 Introduction 162
A.2 The Generator 164
A.2.1 Instances from the Literature 166
A.2.2 Utility Functions 166

List of Figures

2.1 BKZ running times in seconds s for given values of δ_0. 65
2.2 Distribution of right key component ranks for 1000 experiments on $n = 25$, $t = 2.3$, $d = 1$, $\mathrm{p}_{success} = 0.99$. 69
2.3 Example of Noise Tree 74

3.1 Children, parents and strangers. 91
3.2 Histogram of distribution of 0th component in T^ℓ for $1 \leq \ell \leq 4$ with parameters $q = 32003$, $b = 2$, $p = 2^9$, $n = 10$, and $o = 2^{20}$. 102
3.3 Cost of solving Decision-LWE using various algorithms discussed in this work. 107
3.4 Distribution of $c_i - \langle \tilde{\mathbf{a}}_i, \mathbf{s} \rangle$ for parameters $q = 4093, n = 30, a = 15, \sigma = 3.0$. 108
3.5 σ/σ_n for $1 \leq \tau \leq 3$ and $\tau = 128$. 111
3.6 c_τ in function of τ 112

4.1 Experimental Success Rates, Regev-LWE, LLL, $n \in \{35, 40, 45\}$, $t = \|\mathbf{e}\|$ 122
4.2 Minimum lattice dimension, Regev-LWE, success rate 10%, $t = \|\mathbf{e}\|$. 123
4.3 Minimum lattice dimension, Regev-LWE, success rate 10%, $t = 1$. . . 123
4.4 $m_{\|\mathbf{e}\|}/m_1$ Ratios, Regev-LWE, LLL and BKZ-5 124
4.5 Minimum lattice dimension, Lindner & Peikert Parameterisation, success rate 10%, $t = \|\mathbf{e}\|$ 125

4.6 Minimum lattice dimension, Lindner & Peikert Parameterisation, success rate 10%, $t = 1$ 126

5.1 Logarithmic Running Times, G_LLL_FP, G_LLL_QP, and G_LLL_RR146

List of Tables

1.1 Mean experimental Hermite root factors, comparison with best-known upper bounds, [38] 26

2.1 Cost of solving Search-LWE for parameters suggested in [86] with $d = 1, t = 3, \epsilon = 0.99$ with BKW. 63
2.2 Cost of solving Search-LWE for parameters suggested in [61] with $d = 1, t = 2.7, \epsilon = 0.99$ with BKW. 64
2.3 Cost of finding $G \approx \mathbf{s}$ for parameters suggested in [7] with $d = 2, \epsilon = 0.99$. 64
2.4 Cost of solving Decision-LWE with BKZ as in [61] and BKW as in Corollary 3. 65
2.5 Cost of distinguishing LWE samples from uniform as reported as "Distinguish" in [61], compared to Corollary 3. BKZ estimates obtained using BKZ 2.0 simulator-derived cost estimate. 66
2.6 Cost of solving Search-LWE reported as "Decode" in [61], compared to the cost solving Decision-LWE with BKW 68

3.1 c_τ for small values of τ 94

3.2 Cost for solving Decision-LWE with advantage ≈ 1 for BKW, BKZ and MITM where q, σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_2^n)$. We run BKZ $1/\epsilon$ times, "$\log \mathbb{Z}_2$" gives the number of "bit operations", "$\log L_{\mathbf{s},\chi}^{(n)}$" the number of LWE oracle calls and "log mem" the memory requirement of $\mathbb{Z}_q$ elements. All logarithms are base-2. 104

3.3 Cost for solving Decision-LWE with advantage ≈ 1 for BKW and BKZ variants where q, σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_2^n)$ after one-shot modulus reduction with $p = q\sqrt{n/12}\sigma_s/\sigma$. 104

3.4 Cost for solving Decision-LWE with advantage ≈ 1 with the algorithms discussed in this work when $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_2^n)$. 105

3.5 Cost for solving Decision-LWE with advantage ≈ 1 for BKW, BKZ and MITM where q and σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$. 105

3.6 Cost for solving Decision-LWE with advantage ≈ 1 for BKW and BKZ variants where q, σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$ after one-shot modulus reduction with $p = q\sqrt{n/12}\sigma_s/\sigma$. 106

3.7 Cost for solving Decision-LWE with advantage ≈ 1 with the algorithms discussed in this work when $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$. 106

3.8 Expected Number of Noise-Element Collisions, no Table-Switching . 108

4.1 Selected Lindner & Peikert LWE Parameters 125

4.2 Parameters for finding $\mathbf{e}$ with success rate 10%, Regev's and Lindner & Peikert's parameters. 125

4.3 Estimated cost of finding $\mathbf{e}$ with success rate 0.099, Regev's parameters. 128

4.4 Estimated cost of solving decision-LWE, advantage ~ 0.099, Regev's parameters, dual-lattice distinguisher 129

5.1 Suggested parameters in [49]. 141

5.2 Extended parameters: We fix $m = 2n$, $k = 12$, $\zeta = 10$, $\beta = 10$, and $\lambda = 5$. We calculate q by NextPrime$(4k\zeta\beta^2 mn^{2+\lambda})$ = NextPrime$(3840n^8)$ to satisfy the correctness condition. 145

5.3 Experimental results using G_LLL_FP, G_LLL_QP, and G_LLL_RR. 147

Chapter 1

Introduction

1.1 Foreword

Cryptographic constructions rely, by necessity, upon random choices: consider, for instance, a cryptosystem with a deterministic key-generation process: the key so derived could not be secret. We rely therefore on the intractability of random instances of (assumed) hard problems for the security of cryptographic constructions. Herein lies the central difficulty in the design of secure cryptosystems: while we can readily ascertain that the *worst cases* of many amenable problems are hard, we have no way (in general) of determining the *average-case* hardness. As exemplified by the majority of proposed cryptosystems to date, there can exist a substantial gap between these two regimes.

The holy grail of (public-key) cryptography has been expressed as the development of cryptosystems, the security of which relies on the question $\mathbf{P} = \mathbf{NP}$?: this, in a sense, being the best we can hope to achieve. A necessary step in this direction is the development of cryptosystems in which the average-case hardness of the underlying hard problem is somehow 'connected' to the worst-case hardness. Such a result was achieved by Miklos Ajtai in his seminal work of 1996 [2], where it was shown that there exists an average-case problem, defined on a certain class of random lattices,

the solution (with non-negligible probability) of which would imply (*inter alia*) the solution of the worst-case problem of determining the length of a shortest non-zero vector in a related class of lattices, to within polynomial factors. This remarkable result led to the subsequent development of the field of lattice-based cryptography, motivated chiefly by the (still without analogue) prospect of basing the security of cryptographic constructions on the worst-case hardness of a class of problems.

Another reason for the remarkableness of Ajtai's 1996 result is that, previously, lattices and lattice basis reduction were viewed chiefly as a 'destructive' influence in cryptography, having been employed in the cryptanalysis of several schemes, the effective termination of entire areas of research (e.g. knapsack cryptosystems) and the development of attacks on low-exponent RSA, all largely through the application of the famed 1982 LLL algorithm of Lenstra, Lenstra and Lovasz [60].

While the development of lattice-based 'minicrypt' primitives was relatively straightforward, the development of 'cryptomania' primitives has had a somewhat more laboured history. The 1997 work of Ajtai and Dwork [3] contained a proposal of a public-key cryptosystem, the hardness of which could be provably reduced to the worst-case hardness of n^8-unique SVP on a certain class of lattices. Several subsequent developments along this path were made, culminating in the 2003 work of Regev [85] in which a public-key cryptosystem based on $n^{1.5}$-unique SVP was developed. The year 2005, however, saw a marked development with Regev's proof [86] that the worst-case hardness of GapSVP could be reduced to the average-case hardness of the Learning with Errors (LWE) problem, a problem which has the benefit of being intuitive and easy to understand. There followed a 'growth-spurt' in the development of lattice-based cryptomania primitives, a result of the inherent flexibility of the LWE problem, with applications including homorphic encryption, oblivious transfer protocols, identity-based encryption and more.

Besides the average-case to worst-case security guarantees, the LWE problem has the added attraction that the only operations necessary for en/decryption are modular multiplication and addition in a ring of polynomial size. Two disadvantages remained, however - the quantum nature of the security reduction of LWE to GapSVP and the relative inefficiency (in both storage and communication complexity terms) of LWE-based primitives. The first of these was resolved in 2013 [22] to obtain a classical reduction of worst-case lattice problems to LWE, while the second of these is an area of ongoing research with the use of ideal lattices to improve greatly the efficiency of LWE-based primitives, while retaining worst-case hardness, albeit only to problems in a certain sub-class of lattices.

While theoretical progress has been swift in the development and application of such primitives, practical considerations, including concrete instantiation of such schemes has lagged somewhat behind. In particular, still very few algorithms are known for solving the LWE problem and those that are known are less than perfectly understood. As a result, someone intending to employ such schemes would have a somewhat difficult task in understanding the applicability of current algorithms and the corresponding parameters to choose to attain a given security level. As an example of the need for better understanding of parameter choices for LWE we examine a recently proposed provably-secure public-key cryptosystem and show that, due to the close relationship between the hard problem underlying this scheme and LWE, all parameter choices proposed in [49] are weak, indeed weak enough to be practically attacked. Thus, this work aims to fill this gap by analysing some algorithms for LWE and modifying such algorithms where advantageous.

We hope that this work will assist future users of lattice-based cryptography in understanding the various methods of solving the LWE and Ideal-LWE problems and hence further bolster confidence in their strength and complexity.

1.2 Post-Quantum Cryptography

In 1994 a cryptographic earthquake occurred, taking the form of Shor's announcement [92] of a quantum algorithm for factoring an RSA modulus n in $(\log_2 n)^{2+o(1)}$ operations (using a large-enough quantum computer), with the current best-known classical algorithms requiring $\sim e^{(1.923+o(1))(\ln n)^{1/3}(\ln \ln n)^{2/3}}$ operations. The implications of Shor's discovery gave rise to the discipline known as 'post-quantum cryptography', the aim of which, as the name suggests, is to develop and strengthen confidence in public-key and signature schemes against which only Grover's enhanced exhaustive-search algorithm can be employed by a quantum computer (to the best of our current knowledge).

This discipline now encompasses both constructions dating back to 1977 and post-Shor constructions. The sub-field of code-based cryptography stems from the McEliece scheme of 1977, multivariate cryptography from the early 1980s and lattice-based cryptography from 1996. This work is concerned with the last of these sub-field, lattice-based cryptography. From 2005 onwards, the principal hard problems in this sub-field have been the Learning with Errors (LWE) problem and the Short Integer Solution (SIS) problem, a pair of closely-related problems to which we can reduce the *worst-case* hardness of certain (assumed) hard lattice problems. However, the practical hardness of both the LWE and SIS problems are not well understood, with the development of algorithms for their solution appearing to be at relatively early stages of development. Thus, when actually proposing concrete instantiations of lattice-based schemes for real-world use, a major stumbling-block is the lack of understanding of the complexity of even currently-known algorithms. This work aims to advance our understanding of such questions and hence hopefully bring real-world use of lattice-based cryptography one small step closer.

With recent work on algorithms for the discrete logarithm problem [14], the walls

seem to be 'closing in' on traditional number-theoretic public-key cryptography, with confidence, particularly in the long-term security scenario, of such schemes slowly but steadily diminishing.

1.3 Notation and Preliminary Definitions

We always start counting at zero and denote vectors in lower-case bold. Given a vector $\mathbf{a}$, we denote by $\mathbf{a}_{(i)}$ the i-th entry in $\mathbf{a}$, i.e. a scalar, and by $\mathbf{a}_{(i,j)}$ the subvector of $\mathbf{a}$ spanning the entries at indices $i, \ldots, j-1$. When given a list of vectors, we index its elements by subscript, e.g. we use $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2$ to denote the first three members of the list. We denote matrices in upper-case bold.

Definition 1 (Negligible Function)**.** *A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ is said to be* negligible *if, for every constant $c \in \mathbb{R}_{>0}$, there exists a $k_0 \in \mathbb{N}$ such that $|\varepsilon(k)| < k^{-c}, \forall k > k_0$.*

1.4 Lattices and Related Hard Problems

A (full-rank) lattice Λ in $\mathbb{R}^n$ is a discrete additive subgroup. For a general introduction, the reader is referred to [70]. We view a lattice as being generated by a (non-unique) basis $\mathbf{B} = \{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\} \subset \mathbb{R}^n$ of linearly-independent vectors. We assume that the vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}$ form the rows of the $n \times n$ matrix $\mathbf{B}$. That is:

$$\Lambda = \mathcal{L}(\mathbf{B}) = \mathbb{Z}^n \cdot \mathbf{B} = \left\{ \sum_{i=0}^{n-1} x_i \cdot \mathbf{b}_i \mid x_0, \ldots, x_{n-1} \in \mathbb{Z} \right\}.$$

The rank of a lattice Λ is the dimension of the linear span $\mathrm{span}(\Lambda)$ of Λ. The basis $\mathbf{B}$ is not unique - we call two bases $\mathbf{B}$ and $\mathbf{B}'$ *equivalent* if and only if $\mathbf{B}' = \mathbf{B}\mathbf{U}$ where $\mathbf{U}$ is a unimodular matrix - an integer matrix with $|\det(\mathbf{U})| = 1$ and note that such unimodular matrices form the general linear group $GL_n(\mathbb{Z})$.

In modern lattice-based cryptography and hence also in this work, attention is mainly restricted to a particular class of lattices, called q-ary lattices:

Definition 2 (q-ary Lattice)**.** *A lattice $\mathcal{L}$ is said to be a q-ary lattice if:*

$$q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$$

Equivalently, if we let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and view $\mathbf{A} : \mathbb{Z}^m \to \mathbb{Z}^n$ and $\mathbf{A}^T : \mathbb{Z}^n \to \mathbb{Z}^m$ as group homomorphisms, with $\pi_q : \mathbb{Z}^n \to \mathbb{Z}_q^n$ denoting the natural projection, then we have that the following two sets constitute full-rank lattices:

$$\mathcal{L}(\mathbf{A}^T) = \mathrm{Im}(\mathbf{A}^T) + q\mathbb{Z}^m = \{\mathbf{z} \in \mathbb{Z}^m \mid \exists \mathbf{x} \in \mathbb{Z}^n : \mathbf{z} = \mathbf{A}^T\mathbf{x}(\mod q)\} \subseteq \mathbb{Z}^m$$

$$\mathcal{L}^{\perp}(\mathbf{A}) = \mathrm{Ker}(\pi_q(\mathbf{A})) = \{\mathbf{z} \in \mathbb{Z}^m \mid \mathbf{A}\mathbf{z} = 0(\mod q)\} \subseteq \mathbb{Z}^m$$

The determinant or volume $\mathrm{vol}\,(\Lambda)$ of a (full-rank) lattice Λ is the determinant of any given basis of Λ, hence $\mathrm{vol}\,(\Lambda) = \det(\mathbf{B})$. The *dual* of a lattice Λ, denoted by Λ^*, is the lattice consisting of the set of all vectors $\mathbf{z} \in \mathbb{R}^n$ such that $\langle \mathbf{y}, \mathbf{z} \rangle \in \mathbb{Z}$ for all vectors $\mathbf{y} \in \Lambda$. Given a lattice Λ, we denote by $\lambda_i(\Lambda)$ the i-th minimum of Λ

$$\lambda_i(\Lambda) := \inf\left\{r \mid \dim(\mathrm{span}(\Lambda \cap \bar{\mathcal{B}}_n(\mathbf{0}, r))) \geq i\right\}$$

where $\bar{\mathcal{B}}_n(\mathbf{0}, r)$ denotes the closed, zero-centred n-dimensional (Euclidean) ball of radius r. We define the minimum distance from a given point $\mathbf{t} \in \mathbb{R}^n$ to the lattice by $\mathrm{dist}(\Lambda, \mathbf{t}) = \min\{\|\mathbf{t} - \mathbf{x}\|_2 \mid \mathbf{x} \in \Lambda\}$.

Minkowski's second theorem gives us a bound on the geometric mean of the successive minima. Given an n-dimensional lattice Λ and any $1 \leq k \leq n$ we have

$$\left(\prod_{i=1}^{k} \lambda_i(\Lambda)\right)^{1/k} \leq \sqrt{\gamma_n} \cdot \mathrm{vol}\,(\Lambda)^{1/n}$$

where γ_n denotes Hermite's constant of dimension n.

However, determining the exact value of γ_n is a long-standing open problem in the geometry of numbers, with the exact values being known for only $1 \leq n \leq 8$ and $n = 24$ [27]. Heuristically speaking, given a random lattice Λ of dimension n and a Euclidean ball $\bar{\mathcal{B}}_n(\mathbf{x}, r)$, we expect that the number of lattice points which lie in

$\Lambda \cap \bar{\mathcal{B}}_n(\mathbf{x}, r)$ to be approximately equal to $\frac{\mathrm{vol}\left(\bar{\mathcal{B}}_n(\mathbf{x},r)\right)}{\mathrm{vol}(\Lambda)}$. From this observation we can derive what is known as the 'Gaussian heuristic'.

Now, the lattices we mainly consider in this work are not random in the sense of Goldstein and Mayer [43] - for more details on the nature of random lattices, the reader is referred to [43]. However, it is generally assumed in the literature, as in this work, that the Gaussian heuristic holds well for the lattices considered here. If the above approximate equality were to hold for any such ball, then by considering the unit ball in $\bar{\mathcal{B}}_m(\mathbf{0}, 1) \subset \mathbb{R}^n$, we would have

$$| \Lambda \cap \bar{\mathcal{B}}_m(\mathbf{0}, 1) | \approx \frac{\pi^{m/2}}{\Gamma(1 + m/2) \cdot \mathrm{vol}\,(\Lambda)}.$$

where Γ denotes the standard gamma function $\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$, $z \in \mathbb{C}$. Hence we would expect that

$$\lambda_1(\Lambda) \approx \left(\frac{\mathrm{vol}\,(\Lambda)}{\mathrm{vol}\left(\bar{\mathcal{B}}_m(\mathbf{0}, 1)\right)} \right)^{1/m} = \frac{\mathrm{vol}\,(\Lambda)^{1/m} \cdot \Gamma(1 + m/2)^{1/m}}{\sqrt{\pi}}$$

For random lattices, it is known that, with overwhelming probability, the above holds (for all successive minima) [2]. This provides the motivation for the Hermite-SVP problem, which we define below. More generally, we list below the five main general-lattice problems of relevance to this work:

The approximate Shortest Vector problem (γ-SVP):
Input. A lattice $\Lambda = \mathcal{L}(\mathbf{B})$.
Question. Find a vector $\mathbf{v} \in \Lambda$ such that $0 < \|\mathbf{v}\| \leq \gamma \cdot \lambda_1(\Lambda)$.

The approximate Hermite Shortest Vector problem (γ-HSVP):
Input. A lattice $\Lambda = \mathcal{L}(\mathbf{B})$.
Question. Find a vector $\mathbf{v} \in \Lambda$ such that $0 < \|\mathbf{v}\| \leq \gamma \cdot \det(\Lambda)^{\frac{1}{m}}$.

Any algorithm which solves γ-SVP also solves $\gamma\sqrt{\gamma_n}$-HSVP while, reciprocally, we

can use a γ-HSVP algorithm linearly-many times to solve γ^2-SVP [66]. If an algorithm solves κ-HSVP for lattices of dimension n, we say that such an algorithm attains a 'Hermite Root Factor' $\delta_0 := \kappa^{1/n}$ when applied to such lattices.

The approximate Shortest Vector Problem (γ-SVP) ($\gamma \geq 1$) is NP-Hard under randomized reduction for any $\gamma < 2^{(\log n)^{1/2-\epsilon}}$, where $\epsilon > 0$ is an arbitrarily small constant [54].

The bounded distance decoding problem (BDD$_\eta$):
Input. A lattice Λ and a vector $\mathbf{t}$ such that $\text{dist}(\mathbf{t}, \Lambda) < \eta \cdot \lambda_1(\Lambda)$.
Question. Find the lattice vector $\mathbf{y}$ which is closest to $\mathbf{t}$.

We note that, when considering BDD$_\eta$ from a complexity theory approach, arbitrary values for η can be considered while in practical settings, the problem is often defined with the restriction that $\eta \leq \frac{1}{2}$. The case of solving BDD$_{\eta > \frac{1}{2}}$ corresponds to list-decoding in coding parlance. BDD$_\eta$ is known to be NP-hard for any constant factor $\eta > \frac{1}{\sqrt{2}}$ [65].

The GapSVP (promise) problem (GapSVP$_\gamma$):
Input. A lattice Λ, a radius $r > 0$ and approximation factor $\gamma > 1$.
Question. Is $\lambda_1(\Lambda) \leq r$? If so return YES, else if $\lambda_1(\Lambda) > \gamma \cdot r$ return NO, and otherwise return YES or NO.

GapSVP$_\gamma$ is NP-Hard for any constant γ [54].

The Generalised GapSVP (promise) problem (GapSVP$_{\zeta,\gamma}$):
Input. A radius $r > 0$ and approximation factors $\zeta > \gamma > 1$ and a lattice basis $\mathbf{B}$ satisfying:

1. $\lambda_1(\Lambda) \leq \zeta$

2. $\min_i \|\bar{\mathbf{b}_i}\| \geq 1$ where the $\bar{\mathbf{b}_i}$'s are the Gram-Schmidt vectors of the basis $\mathbf{B}$

3. $1 \leq d \leq \zeta/\gamma$

Question. Is $\lambda_1(\Lambda) \leq r$? If so return YES, else if $\lambda_1(\Lambda) > \gamma \cdot r$ return No and otherwise return YES or NO.

The unique shortest vector problem (USVP):
Input. A lattice Λ such that $\lambda_2(\Lambda)/\lambda_1(\Lambda) = g$.
Question. Find a shortest (non-zero) lattice vector.

1.4.1 Lattice Reduction.

The notion of lattice basis reduction can be briefly described as 'improving the quality' of a given basis, though there exists no general metric for the 'quality' of a basis - all basis reduction algorithms operate by applying a sequence of elementary transformations to a given basis with the aim, loosely speaking, of obtaining a basis, the constituent vectors of which are both relatively short and pairwise orthogonal. The simplest lattice reduction algorithm is that of Gauss/Lagrange reduction for two-dimensional lattices which always returns a shortest (non-zero) vector. In a seminal 1982 paper [60], Lenstra, Lenstra and Lovász proposed what has come to be known as the LLL algorithm and which can be essentially described as combining Gram-Schmidt orthogonalisation with the Gauss reduction algorithm on 2-dimensional projected sub-lattices. The LLL algorithm provably returns a basis of a lattice, the first vector of which is no more than $\left(\frac{2}{\sqrt{3}}\right)^n$ times the length of the first minimum of the lattice. Although this bound is tight (i.e. there exist lattices for which this approximation factor holds), in practice LLL performs much more effectively than we can expect or, indeed, satisfactorily explain. The LLL algorithm delivers what is known as an 'LLL-reduced' basis, which we now define.

We first recall the Gram-Schmidt orthogonalisation process.

Definition 3. *Given n linearly independent vectors $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1} \in \mathbb{R}^n$, the Gram-Schmidt orthogonalisation of $\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}$ is defined by*

$$\tilde{\mathbf{b_i}} = \mathbf{b}_i - \sum_{j=0}^{i-1} \mu_{i,j} \tilde{\mathbf{b}_j} \quad \text{where} \quad \mu_{i,j} = \frac{\langle \mathbf{b}_i, \tilde{\mathbf{b}_j} \rangle}{\langle \tilde{\mathbf{b}_j}, \tilde{\mathbf{b}_j} \rangle}$$

The LLL Conditions are:

1. A lattice basis $\mathbf{B} = \{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$ is said to be 'size-reduced' if

$$|\mu_{i,j}| \leq \frac{1}{2}, \quad \forall 0 \leq j < i < n$$

2. A lattice basis $\mathbf{B} = \{\mathbf{b}_0, \ldots, \mathbf{b}_{n-1}\}$ is said to satisfy the Lovász condition for $\delta \in (\frac{1}{4}, 1]$ if

$$\delta \|\tilde{\mathbf{b}}_{k-1}\|^2 \leq \|\tilde{\mathbf{b}}_k\|^2 + \mu_{k,k-1}^2 \|\tilde{\mathbf{b}}_{k-1}\|^2 \quad \forall 1 \leq k < n$$

If both conditions are satisfied, we say that $\mathbf{B}$ is LLL-reduced. As the name suggests, the LLL algorithm delivers such a basis. It easily follows that LLL solves γ-SVP with $\gamma \approx 1.154^n$ and hence also γ-HSVP with $\gamma \approx 1.074^n$.

The original motivations for and applications of the LLL algorithm were for factorising polynomials with rational coefficients and for solving integer linear programming problems. Applications specific to cryptography were quickly realised, with cryptanalyses of knapsack-based cryptosystems, low-exponent RSA and more. Indeed, in this thesis we present an attack on a recently-proposed cryptosystem in which LLL is sufficient to recover the private key in a matter of hours on a desktop computer.

However, despite its simplicity and relative (i.e. 1982) antiquity, the behaviour of LLL remains somewhat mysterious. Victor Shoup, the creator of the NTL library (which contains one of the most widely used implementations of LLL) summarises this succinctly:

> *I think it is safe to say that nobody really understands how the LLL algorithm works. The theoretical analyses are a long way from describing what "really" happens in practice.*

Several attempts have been made [79, 90] to clarify our understanding of the practical behaviour of LLL, but it would seem that there is still much to be done. LLL, however, despite its utility, is the 'weakest' and simplest of the arbitrary-dimension lattice basis reduction algorithms, with variants of Schnorr's block reduction algorithm (BKZ) [91] forming the best-known class of algorithms for practical use today. In this thesis, we consider only the use of LLL and BKZ, but for brevity do not give a detailed introduction to the workings of these algorithms.

Briefly, however, the BKZ algorithm is mainly parameterised by a block-size, denoted by β, with $\beta = 2$ corresponding (essentially) to the LLL algorithm. While in the LLL algorithm, we employ Gauss reduction to recover shortest non-zero vectors in projected 2-dimensional lattices, BKZ-β finds and incorporates shortest non-zero vectors in projected β-dimensional lattices. However, with block-sizes greater than 2, Gauss reduction must be replaced by more costly search algorithms for shortest vectors in the projected lattices. With 'pure' BKZ, a block-size of up to around 30 is practically feasible in low dimension, however with 'pruned enumeration' strategies, block-sizes of ~ 60 are readily achievable in practise, with the work of [39] demonstrating that it is possible to find shortest vectors in 110-dimensional lattices in a few days using a relatively-modest parallel implementation. BKZ-β (provably) solves γ-SVP for $\gamma = \gamma_\beta^{(n-1)/(\beta-1)}$ and γ-HSVP for $\gamma = \sqrt{\gamma_\beta}^{1+(n-1)/(\beta-1)}$ where γ_β denotes Hermite's constant in dimension β.

As mentioned in the quote from Shoup, the practical behaviour of LLL (and similarly of BKZ) is somewhat removed from the theoretical analyses, with the algorithms generally performing much (and consistently) better than can be explained by the theoretical analyses. Folklore for many years, the 2008 work of Gama and Nguyen

[38] constituted the first attempt to document comprehensively such effects. That work detailed the results of a large number of experiments applying LLL, BKZ (and a third algorithm DEEP which we do not discuss), to solve HSVP and USVP instances.

In recent years, several piecemeal improvements have been proposed for BKZ, the most effective of which are:

1. Early termination. *Motivation*: most of the improvement of the quality of the basis occurs during the initial stages of BKZ, with quickly-diminishing returns generally occurring in later stages;

2. Extreme pruning. *Motivation*: substantial speed-up of the searches for shortest vectors in projected sub-lattices. Pruning heuristic rules can be applied to gain exponential speed-ups in exchange for probabilistic success;

3. Pre-processing of local bases. *Motivation*: substantial speed-up of the searches for shortest vectors in projected sub-lattices can occur if these local bases are pre-processed with BKZ with a relatively small block-size.

However, unfortunately, no publicly-available implementations of BKZ which incorporate such improvements are available.

1.4.2 Experiments on Random Lattices and q-ary Lattices

Due to our current imperfect understanding of the practical behaviour of lattice basis reduction algorithms, we are forced to rely on experimental evidence to make projections. The work of Nguyen and Stehle [79] examined some aspects of the practical behaviour of LLL while the work of Gama and Nguyen [38] examined some aspects of both the behaviour of LLL and BKZ (in addition to DEEP). Both of these works, clearly required some notion of random lattices and random bases on which to conduct experiments. *What is a 'random' lattice?*. This subjected is touched on briefly in [79, 38]. In a mathematical sense, the answer is provided by Siegel (1945)

[94], with efficient methods for sampling such random lattices being proposed, for instance by Goldstein and Mayer [43]. In contrast, the lattices we consider in this work are somewhat removed from 'random lattices' as such. However, in all cases examined and in the course of experimentation, no departure from the results for 'random lattices' was observed.

The 2008 work of Gama and Nguyen confirmed what had been known as folklore - that (for the average case) LLL generally solved Approx-SVP and Hermite-SVP with approximation factors much better than could be justified by the theoretical analyses, though with approximation factors still exponential in the lattice dimension and that similar phenomena also held for BKZ. The experimental results indicate that the experimental approximation and Hermite root factors are governed by (apparently) Gaussian distributions. In addition, it transpires that, in practice, the Hermite root factors rapidly converging to constants with increasing dimension. In Table 1.1 we partially reproduce Table 1 from [38], illustrating the experimentally-derived Hermite root-factors for LLL, BKZ-20 and BKZ-28 with the best upper bounds.

	LLL	BKZ-20	BKZ-28
Experimental δ_0	1.0219	1.0128	1.0109
Best proved upper-bound	1.0754	1.0337	1.0282

Table 1.1: Mean experimental Hermite root factors, comparison with best-known upper bounds, [38]

.

Also highly relevant for our purposes was a brief examination in [38] of the performance of LLL and BKZ on lattices possessing a λ_2/λ_1 gap, this being (to the best of our knowledge) the first work to examine such cases in practise. While it had also been folklore that the presence of such a gap led to LLL having improved performance as a λ_1 oracle, this phenomena was (and still is) poorly understood. In Chapter 4 we report the results of a number of experiments on unique-SVP instances

related to LWE.

1.5 Noise Models

GAUSSIANS. Given an $s > 0$, we denote the Gaussian function with domain $\mathbb{R}^n$ and parameter s by

$$\rho_s(\mathbf{x}) := \exp\left(-\pi\|\mathbf{x}/s\|^2\right)$$

We then denote by ν_s the Gaussian probability density function with parameter s:

$$\nu_s(\mathbf{x}) := \rho_s(\mathbf{x})/s^n$$

If we then have a countable set $C \subset \mathbb{R}^n$, we can define the discrete Gaussian distribution $D_{C,s}$ to be

$$D_{C,s}(\mathbf{x}) := \frac{\rho_s(\mathbf{x})}{\sum_{\mathbf{y}\in C}\rho_s(\mathbf{y})}$$

for any $\mathbf{x} \in C$. We typically take such a countable set to be a lattice.

Given a real $\alpha > 0$ and a modulus q, we let Ψ_α denote the distribution on $\mathbb{Z}_q$ obtained by sampling a (real) normal variate of mean 0 and standard deviation $\alpha q/\sqrt{2\pi}$, rounding this variate to the nearest integer then reducing modulo q.

We state a straightforward lemma which, in conjuction with an assumption regarding sums of discrete Gaussian variates, will be useful in our computations later.

Lemma 1. *Let $X_0, \ldots, X_{m-1}$ be independent random variates, with $X_i \sim \mathcal{N}(\mu, \sigma^2)$. Then their sum $X = \sum_{i=0}^{m-1} X_i$ is also normally distributed, with $X \sim \mathcal{N}(m\mu, m\sigma^2)$.*

In the case of X_i following a discrete Gaussian distribution, it does not necessarily follow that a sum of such random variables is distributed in a way analogous to the statement above. However, throughout this work, we assume that this does hold i.e., that Lemma 1 applies to the discrete Gaussian case - while we do not know how to prove this, this assumption causes no apparent discrepancies in our experimental

results. For a detailed discussion on sums of discrete Gaussian random variables, the interested reader is referred to [1].

1.6 Small Integer Solutions and Learning with Errors

The Small Integer Solution (SIS) problem is defined as

Definition 4 (SIS). *Let n and q be integers, $n \in \text{poly}(n)$ and let $\beta \in \mathbb{R}, \beta > 0$. Let $\mathcal{U}(\mathbb{Z}_q^{n\times m})$ denote the uniform distribution on all $n \times m$ matrices with entries in $\mathbb{F}_q$. Given a matrix $\mathbf{A}$ sampled from $\mathcal{U}(\mathbb{Z}_q^{n\times m})$ for some $m \in \text{poly}(n)$, find a non-zero integer vector $\mathbf{z} \in \mathbb{Z}^m$ such that $\mathbf{Az} = \mathbf{0} \bmod q$ and $\|\mathbf{z}\|_2 \leq \beta$.*

Note that β should be large enough for a solution to exist ($\beta > \sqrt{n \log q}$ typically suffices), while taking $\beta \geq q$ renders the problem trivially easy to solve. In [2], Ajtai showed that, given appropriate parameters, the ability to solve SIS with non-negligible probability is at least as hard as solving several lattice approximation problems on n-dimensional lattices in the *worst-case*, to within polynomial factors. We now define the Learning with Errors (LWE) problem, but first introduce a simpler sub-problem - Learning Parities with Noise (LPN).

1.6.1 Learning Parities with Noise

Definition 5. *Let $\langle \cdot, \cdot \rangle_2$ denote the binary inner product. Let $n \in \mathbb{N}$ and $\varepsilon \in (0, 1/2)$. Let $\mathbf{s} \leftarrow \mathcal{U}(\{0,1\}^n)$. Define Ber_ε to be the Bernoulli distribution with parameter ε. We define $\text{B}_{\mathbf{s},\varepsilon}$ to be the distribution given by*

$$\{\mathbf{a} \leftarrow \mathcal{U}(\{0,1\}^n), e \leftarrow \text{Ber}_\varepsilon : (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle_2 \oplus e)\}$$

By abuse of notation, we also use $\text{B}_{\mathbf{s},\varepsilon}$ to denote an oracle which outputs (independent) samples from this distribution. The LPN problem is then, given access to such an oracle, to recover $\mathbf{s}$.

LPN is attractive for several reasons - firstly, it is known to be NP-hard [20] and possesses a search-to-decision reduction [53] i.e. if we can distinguish between outputs

of $\mathrm{B}_{\mathbf{s},\varepsilon}$ and $\mathcal{U}(\{0,1\}^{n+1})$ then we can recover $\mathbf{s}$ efficiently. Additionally, primitives based on LPN are extremely simple to implement, leading to substantial interest in such primitives for implementation on lightweight devices (RFID tags and the like), starting with [48] and followed by many others.

The first sub-exponential algorithm for solving LPN was given by Blum, Kalai and Wasserman in [21], an algorithm possessing time and space complexity $2^{\mathcal{O}(n/\log n)}$. This algorithm, henceforth referred to as the BKW algorithm is discussed at more length in Chapters 2 and 3.

1.6.2 Learning with Errors

We can view the LWE problem as the generalisation of LPN to a larger modulus and replacing the Bernoulli noise distribution with a (typically) Gaussian distribution.

Definition 6 (LWE). *Let n, q be positive integers, χ be a probability distribution on $\mathbb{Z}_q$ and $\mathbf{s}$ be a secret vector following the uniform distribution on $\mathbb{Z}_q^n$. We denote by $L_{\mathbf{s},\chi}^{(n)}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a}$ from the uniform distribution on $\mathbb{Z}_q^n$, choosing $e \in \mathbb{Z}$ according to χ, and returning $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*
– Search-LWE *is the problem of finding $\mathbf{s} \in \mathbb{Z}_q^n$ given pairs $(\mathbf{a}_i, c_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\chi}^{(n)}$.*
– Decision-LWE *is the problem of deciding whether pairs $(\mathbf{a}_i, c_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ are sampled according to $L_{\mathbf{s},\chi}^{(n)}$ or the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

The modulus q is typically taken to be polynomial in n. We note that the above definition has been largely superseded in more recent works by 'normal-form LWE' in which the entries of both the secret vector and the noise elements follow χ.

Throughout this thesis, we often view a set of LWE samples as forming a single 'Matrix-LWE' sample, consisting of a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a vector $\mathbf{c}$ formed by

$\mathbf{c} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$.

If q is prime, it easily follows that search and decision-LWE are equivalent by observing that if we have a decision-LWE oracle and an LWE sample $(\mathbf{a}, c)$, if we choose $b \in \mathbb{Z}_q$ and submit $(\mathbf{a}, c + \langle \mathbf{a}, (b, 0, \ldots, 0) \rangle)$ to our oracle, it will respond in the affirmative only when $b = -\mathbf{s}_0$, the primality of q ensuring uniform distribution otherwise. Hence we can recover $\mathbf{s}$ in $qn \in \text{poly}(n)$ trials.

In the case of composite q, [82] extends the above approach to modulii of the form $q = q_0 \times \ldots \times q_{v-1}$ where each q_i is a prime of polynomial size but not too small i.e. $q_i \in \omega(\sqrt{\log n})$. In [71], the second restriction on the size of q_i is removed, at the cost of an increase in the parameter of the noise Gaussian.

1.6.3 Reduction from Hard Lattice Problems

Here, we briefly outline the main ideas of the known reductions from (assumed) hard worst-case lattice problems to LWE. These are (in chronological order): the quantum reduction due to Regev; the classical reduction (with exponential modulus) due to Peikert and the classical reduction due to Brakerski et. al.

Regev's Reduction

On a high level, Regevs proof consists of a quantum and a classical part. Firstly, we state a proposition from [87], noting that we now define the noise distribution χ to be a Gaussian distribution over $\mathbb{R}$ of standard deviation αq which has been rounded to the nearest integer and reduced modulo q - we follow Regev by denoting such a distribution by Ψ_α.

Proposition 1 (Proposition 2.1, [87]). *Let $q \geq 2$ be an integer and $\alpha \in (0, 1)$ be a real number. Assume we are given access to an oracle that solves the LWE problem with modulus q and errors distributed according to Ψ_α. Then, given as input any lattice L and a basis $\mathbf{B}$, a large enough polynomial number of samples from the discrete*

Gaussian distribution $D_{L^*,r}$ for some $r \geq \sqrt{2}q \cdot \eta_\varepsilon(L^*)$ and a point $\mathbf{x}$ within distance $\alpha q/(\sqrt{2}r)$ of L, we can output the (unique) closest lattice point to $\mathbf{x}$ in polynomial time.

The main idea of the proof is as follows: let $\mathbf{v}$ be the solution of the BDD problem. We wish to generate LWE samples with secret $\mathbf{s} = \mathbf{B}^{-1}\mathbf{v} \bmod q$. We create such a sample by taking a sample $\mathbf{y}$ from $D_{L^*,r}$ and setting $\mathbf{a} = \mathbf{B}^T\mathbf{y} \bmod q$, $c = \lfloor \langle \mathbf{y}, \mathbf{x} \rangle + e \rfloor \bmod q$. The condition on r ensures that $\mathbf{a}$ is (almost) uniformly distributed and e is some additional Gaussian term. We can then observe that

$$\langle \mathbf{y}, \mathbf{x} \rangle = \langle \mathbf{y}, \mathbf{v} + \mathbf{e} \rangle = \langle \mathbf{B}^T\mathbf{y}, \mathbf{B}^{-1}\mathbf{v} \rangle + \langle \mathbf{y}, \mathbf{e} \rangle$$

where $\mathbf{e}$ is an error vector of norm $\|\mathbf{e}\| \leq \alpha q/\sqrt{2}r$. We have neglected some technicalities regarding the emergent noise distribution, though it should be apparent that the ability to sample from $D_{L^*,r}$ thus furnishes us with the ability to solve BDD with parameter $\alpha q/(\sqrt{2}r)$ for some r not too small. The main difficulty, then, is in generating samples from $D_{L^*,r}$.

We now require the 'bootstrapping' lemma from [86]:

Lemma 2 (Lemma 3.2, [86])**.** *There exists an efficient algorithm that, given any n-dimensional lattice L and $r > 2^{2n}\lambda_n(L)$, outputs a samples from a distribution that is within statistical distance $2^{-\Omega(n)}$ of $D_{L,r}$.*

The proof proceeds by reducing the lattice basis using LLL, thus obtaining a basis of maximal length at most $2^n\lambda_n(L)$. If we then take a sample $\mathbf{y}$ from the (continuous) n-dimensional Gaussian ν_r of parameter r and then output $\mathbf{y} - (\mathbf{y} \bmod \mathcal{P}(L))$, the resulting lattice vector follows a distribution close to $D_{L,r}$.
The quantum component of Regev's reduction now arises:

Lemma 3 (Lemma 3.3, [86])**.** *Let $\varepsilon = \varepsilon(n)$ be a negligible function, $\alpha = \alpha(n) \in (0,1)$ be a real number and $q = q(n) \geq 2$ be an integer. Assume we have access to an oracle which solves LWE with modulus q and noise parameter α, given a polynomial number*

of samples. Then, there exists a constant $c > 0$ and an efficient quantum algorithm that, given any n-dimensional lattice L, a number $r > \sqrt{2}p\eta_\varepsilon(L)$, and n^c samples from $D_{L,r}$, produces a sample from $D_{L,r\sqrt{n}/(\alpha q)}$.

We do not discuss the proof here and refer the reader to [86] for further details. The final stage of the proof is to give a reduction from GapSVP. For this, we reproduce Theorem 3.1 from [82] and give a high-level description of the main idea of the proof.

Theorem 1 (Theorem 3.1, [82])**.** *Let $\alpha \in (0,1)$ be a real number and $\gamma = \gamma(n) \geq n/(\alpha\sqrt{\log n})$. Let $\zeta = \zeta(n) \geq \gamma$ and $q = q(n) \geq (\zeta/\sqrt{n}) \cdot \omega(\sqrt{\log n})$. There is a probabilistic polynomial time reduction from solving* $\mathrm{GapSVP}_{\zeta,\gamma}$ *in the worst case (with overwhelming probability) to solving LWE with modulus q and noise parameter α using* $\mathrm{poly}(n)$ *samples.*

The main idea of the proof is to show that, by choosing a random lattice point and then perturbing this by a point chosen at random from the n-dimensional ball $\mathcal{B}_n(u)$ of a certain radius u, the BDD instances which result can be used as input (along with samples from a Gaussian on the dual lattice) to the LWE oracle. We then use our LWE oracle to solve these BDD instances and hence solve $\mathrm{GapSVP}_{\zeta,\gamma}$ by altering the radius u of the ball.

However, it is not currently known how to perform a step analogous to that in Lemma 3 classically, hence Peikert's reduction is inherently 'lossy' compared to that of Regev with only a reduction from $\mathrm{GapSVP}_{\zeta,\gamma}$ in $\sqrt{n}$-dimensional lattices to n-dimensional LWE.

However, the hardness of $\mathrm{GapSVP}_{\zeta,\gamma}$ is less than clear, with the problem being only equivalent to GapSVP_{γ} only when ζ is large, such cases corresponding to LWE with exponential modulus.

Brakerski et. al.'s Reduction

Recently, Brakerski et. al. [22] gave the first classical reduction of worst-case lattice problems to LWE with polynomial modulus by employing techniques originally introduced by Brakerski and Vaikuntanathan [23] for controlling noise growth during fully-homomorphic encryption. We examine such techniques in more detail in Chapter 3 with regard to solving LWE instances. However, [22] takes such approaches further, giving techniques for (essentially) transforming LWE samples of modulus q and dimension n into (almost faithful) LWE samples of modulus q' and dimension n', as long as $n \log q$ is preserved. By applying such techniques, it easily follows that we can transform LWE samples with polynomial modulus into samples with exponential modulus (and reduced dimension), allowing classical reduction from GapSVP_γ.

We discuss such techniques further in Chapters 3 and 5.

1.7 Solving LWE using Lattice Reduction

In practise, the most efficient methods for solving LWE involve viewing a set of LWE samples as determining either a BDD instance or an SIS instance and attempting to recover the lattice point corresponding to the secret vector or to find a short vector in the (scaled) dual lattice, respectively. Here, we briefly cover the standard lattice-based methods for solving LWE instances, mainly for later reference and comparison.

We may classify lattice-based algorithms for solving LWE into two families. The first family reduces LWE to the problem of finding a short vector in the (scaled) dual lattice (commonly known as the Short Integer Solution (SIS) problem) constructed from a set of LWE samples. The second family solves the Bounded Distance Decoding (BDD) problem in the primal lattice. For both families lattice reduction algorithms may be applied. We may either use lattice reduction to find a short vector in the dual lattice or apply lattice reduction and (a variant of) Babai's algorithm to solve

BDD [61]. Indeed, the expected complexity of lattice algorithms is often exclusively considered when parameters for LWE-based schemes are discussed. However, while the effort on improving lattices algorithms is intense [88, 28, 77, 39, 80, 47, 75, 84], our understanding of the behaviour and concrete complexity of these algorithms in high dimensions is still somewhat limited.

In [73], the authors briefly examine an approach for solving LWE by distinguishing between valid matrix-LWE samples of the form $(\mathbf{A}, \mathbf{c}) = (\mathbf{A}, \mathbf{A}^T\mathbf{s} + \mathbf{e})$ and samples drawn from the uniform distribution over $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$. Given the matrix $\mathbf{A}$, one way of constructing such a distinguisher is to find a short vector $\mathbf{u}$ in the dual lattice $\Lambda(\mathbf{A})^\perp$ such that $\mathbf{A}\mathbf{u} = \mathbf{0} \mod q$. If $\mathbf{c}$ belongs to the uniform distribution over $\mathbb{Z}_q^n$, then $\langle \mathbf{u}, \mathbf{c}\rangle$ belongs to the uniform distribution on $\mathbb{Z}_q$. On the other hand, if $\mathbf{c} = \mathbf{A}^T\mathbf{s} + \mathbf{e}$, then $\langle \mathbf{u}, \mathbf{c}\rangle = \langle \mathbf{u}, \mathbf{A}^T\mathbf{s} + \mathbf{e}\rangle = \langle \mathbf{u}, \mathbf{e}\rangle$, where samples of the form $\langle \mathbf{u}, \mathbf{e}_i\rangle$ are governed by another discrete, wrapped Gaussian distribution. Following the work of Micciancio and Regev [73], the authors of [61] give estimates for the complexity of distinguishing between LWE samples and uniform samples by estimating the cost of the BKZ algorithm in finding a short enough vector. In particular, given $n, q, \sigma = \alpha q$, we set $s = \sigma \cdot \sqrt{2\pi}$ and compute $\beta = q/s \cdot \sqrt{\log(1/\epsilon)/\pi}$. From this β we then compute the required root Hermite factor $\delta = 2^{\log_2^2(\beta)/(4n\log_2 q)}$. Note that this presupposes access to $m = \sqrt{n \log_2 q / \log_2 \delta}$ samples.

Alternatively, we may attempt to solve the LWE problem in the primal lattice by decoding to the closest lattice point. The classic algorithm for solving such CVP instances is due to Babai and consists of successively projecting the noisy point onto hyperplanes within the lattice and rounding. Applying Babai's algorithm to an LLL-reduced basis and a noisy lattice point yields a lattice point which is exponentially far from the true closest lattice point. If we wish for Babai's algorithm to recover the closest lattice point for LWE instances, we would generally require very strong lattice reduction. A simple modification (originally due to Klein [58] and applied to

the LWE case in [61]) can be made to Babai's algorithm, leading to a tree-based algorithm in which we carry out (possibly) more than one projection at each stage of the algorithm. Combining this modified algorithm with strong lattice reduction allows somewhat more efficient attacks than the dual-lattice attack. A further improvement to this algorithm was proposed in [63] in which fast enumeration techniques, as applied in exact-SVP algorithms, were applied to the tree arising from the modified Babai algorithm.

It is against these algorithms that we wish to compare the results obtained in Chapters 2 and 3. Rigorous comparison, however, is problematic if not impossible, given the current lack of understanding of the concrete complexity and optimal application of lattice-reduction algorithms. As mentioned in 1.4.2, the results of [38] indicate that, in practise, LLL achieves a $\delta_0 \approx 1.0219$ while BKZ-20 and BKZ-28 achieve $\delta_0 \approx 1.0128$ and $\delta_0 \approx 1.0109$, respectively and provide conjectures that the current limits of 'practical' lattice reduction appear to be a root Hermite factor of ≈ 1.01, with $\delta_0 = 1.005$ being far beyond reach (in high dimension). However, estimation of the running time of BKZ in high dimension with a large block-size is difficult - no good upper-bounds are known, with the best being super-exponential in n. For BKZ 2.0 [28], with several additional degrees of freedom, even less is presently known. To attempt a conservative prediction of the running time of BKZ 2.0 with large block-size, the authors of [61] assume that δ_0 is the dominant influence on the running-time of BKZ in high dimension and proposed a simple extrapolation of running times as a function of δ_0 leading to the model

$$\log_2 T_{sec} = 1.8/\log_2 \delta_0 - 110. \tag{1.1}$$

We can translate this figure into bit operations by assuming $2.3 \cdot 10^9$ bit operations per second on a 2.3 GHz CPU. However, the accuracy and hence utility of such models is debatable, with such models giving infeasibly low complexity estimates for the application of LLL. Alternative models of which we are aware are

$\log_2 T_{sec} = \exp(1/\log_2(\delta_0)^{1.001} - 43.4)$ [88] proposed by Rueckert & Schneider and the 'BKZ 2.0 simulator' of Chen & Nguyen [28], neither of which appear wholly satisfactory.

An alternative method for solving LWE (and for BDD in general) using lattice reduction is to employ Kannan's embedding method. We discuss this approach in detail in Chapters 4 and 5 and hence do not discuss it here.

1.8 Generating LWE Instances

For all experiments involving LWE/Ring-LWE instances featured in this thesis, we employed an LWE instance generator, allowing the generation of LWE instances adhering to all parameters proposed to date. The author participated in the development of a SAGE module to allow the unified generation of instances, along with Martin Albrecht, Daniel Cabarcas, Florian Göpfert and Michael Schneider. This project arose from a visit to TU Darmstadt and the amalgamation and consolidation of various pieces of code for LWE instance generation from the various authors and has since been included in SAGE. A brief description of the LWE instance generator is provided in Appendix A, as in [8].

Chapter 2

Naïve Algorithms for LWE and the BKW Algorithm

The contents of this chapter are based on the paper "On the Complexity of the BKW Algorithm on LWE", which appeared in Designs, Codes and Cryptography [6] and was a work conducted in collaboration with M. R. Albrecht, C. Cid, J. C. Faugére and L. Perret.

The Blum-Kalai-Wasserman (BKW) algorithm is the prototypical method for solving LPN and LWE by searching for short dual-lattice vectors by finding low-weight linear combinations of initial vectors. In this chapter we present a study of the complexity of the Blum-Kalai-Wasserman (BKW) algorithm when applied to LWE and provide refined estimates for the data and computational effort requirements for solving *concrete* instances of the LWE problem. We apply this refined analysis to suggested parameters for various LWE-based cryptographic schemes from the literature and compare with alternative approaches based on lattice reduction. As a result, we provide new upper bounds for the concrete hardness of these LWE-based schemes. Rather surprisingly, it appears that BKW algorithm outperforms known estimates for lattice reduction algorithms starting in dimension $n \approx 250$ when LWE

is reduced to SIS. However, this assumes access to an unbounded number of LWE samples.

2.1 Introduction

Let us first consider some naïve algorithms for solving LWE with $q \in \text{poly}(n)(n)$. Perhaps the simplest algorithm would be to request enough LWE samples until we obtain $\text{poly}(n)$ samples of the form $(\mathbf{a}, c)$ where $\mathbf{a} = (1, 0, \ldots, 0)$, allowing us to recover $\mathbf{s}_0$. We then repeat for all other elements of $\mathbf{s}$. Since the probability of obtaining such samples is q^{-n}, we require $2^{\mathcal{O}(n \log n)}$ samples and also a running time of the same order. A marginally more sophisticated algorithm arises from the observation that, after observing $\mathcal{O}(n)$ samples, if we can find an $\mathbf{s}'$ which 'approximately satisfies' these equations, then $\mathbf{s}' = \mathbf{s}$ with overwhelming probability. We thus reduce the sample complexity to $\mathcal{O}(n)$, though retaining the time complexity of $2^{\mathcal{O}(n \log n)}$. In more detail, consider a modulus N and χ being a discrete Gaussian on $\mathbb{Z}$ with parameter $\frac{N}{a}$ for a some constant. Then, assuming we can treat this discrete Gaussian in a continuous fashion, we fix a tail-cut and obtain $\Pr[e \leftarrow \chi \colon |e| > c] = 1 - \text{erf}(\frac{c\sqrt{\pi}}{s})$. Thus, if we have k samples, the probability that all error terms have absolute value less than c is $(\text{erf}(\frac{c\sqrt{\pi}}{s}))^k$. Conversely, for a wrong guess, the probability that any given error term has absolute value less than c is $2c/N$ and hence the probability that all k error terms have absolute value less than c is $(2c/N)^k$. Then, by taking a union bound over all such wrong guesses, we have the probability of any such wrong guess having all error terms $< c$ is less than $(N-1) \cdot (\frac{2c}{N})^k$. Thus, after only a very small number of samples, only the secret (with extremely high probability) approximately satisfies the equations presented by the samples.

A third possible approach is to take a set of n LWE samples and perform Gaussian elimination on the matrix $[\mathbf{A} \mid \mathbf{c}]$, then take majority vote over all such reduced samples. However, since the variance of the noise present in the elements of $\mathbf{c}$ in-

creases linearly with the number of additions, it should be apparent that such an approach can only work for the very smallest of examples.

The BKW algorithm, in contrast, requires $2^{\mathcal{O}(n)}$ samples and time and can best be viewed as a form of structured Gaussian elimination in which we endeavour to minimise the number of additions performed. BKW and algorithms related to it are generally referred to as 'combinatorial' methods for solving LWE. Combinatorial algorithms for tackling the LWE problem remain rarely investigated from an algorithmic point of view. For example, the main subject of this chapter – the BKW algorithm – specifically applied to the LWE problem has received little treatment in the literature[1]. However, since the BKW algorithm can be viewed as an oracle producing short vectors in the dual lattice spanned by the $\mathbf{a}_i$ (i.e., it reduces LWE to SIS) it shares some similarities with combinatorial (exact) SVP solvers.

Finally, Arora and Ge [11] proposed a new algebraic technique for solving LWE which does not rely upon lattice reduction. The algorithm has a total complexity (time and space) of $2^{\tilde{\mathcal{O}}(\sigma^2)}$ and is thus subexponential when $\sigma \leq \sqrt{n}$, remaining exponential when $\sigma > \sqrt{n}$. It is worth noting that Arora and Ge achieve the $\sqrt{n}$ hardness-threshold found by Regev [86], and thus provide a subexponential algorithm precisely in the region where the reduction to GapSVP fails. We note however that currently the main relevance of Arora-Ge's algorithm is asymptotic as the constants hidden in $\tilde{\mathcal{O}}(\cdot)$ are rather large [5]; it is an open question whether one can improve its practical efficiency.

Firstly, we present a detailed study of a dedicated version of the Blum, Kalai and Wasserman (BKW) algorithm [21] for LWE with discrete Gaussian noise. The BKW

[1] However, a study of the algorithm to the LPN case was conducted in [37], which inspired [6]. The authors of [37] also gave revised security estimates for some HB-type authentication protocols relying on the hardness of LPN.

algorithm is known to have (time and space) complexity $2^{\mathcal{O}(n)}$ when applied to LWE instances with a prime modulus polynomial in n [86]; in this chapter we provide both the leading constant of the exponent in $2^{\mathcal{O}(n)}$ and concrete cost of BKW when applied to Search- and Decision-LWE. That is, by studying in detail all steps of the BKW algorithm, we 'de-asymptotic-ify' the understanding of the hardness of LWE under the BKW algorithm and provide concrete values for the expected number of operations for solving instances of the LWE problem. More precisely, we show the following theorem in Section 2.3.4.

Theorem 2 (Search-LWE, simplified)**.** *Let* $(\mathbf{a}_i, c_i)$ *be samples following* $L^{(n)}_{\mathbf{s},\chi}$*, set* $a = \lfloor \log_2(1/(2\alpha)^2) \rceil$*,* $b = n/a$ *and* q *a prime. Let* d *be a small constant* $0 < d < \log_2(n)$*. Assume* α *is such that* $q^b = q^{n/a} = q^{n/\lfloor \log_2(1/(2\alpha)^2) \rceil}$ *is superpolynomial in* n*. Then, given these parameters, the cost of the BKW algorithm to solve* Search-LWE *is*

$$\left(\frac{q^b - 1}{2}\right) \cdot \left(\frac{a(a-1)}{2} \cdot (n+1)\right) + \left\lceil \frac{q^b}{2} \right\rceil \cdot \left(\left\lceil \frac{n}{d} \right\rceil + 1\right) \cdot d \cdot a + \mathrm{poly}(n) \approx \left(a^2 n\right) \cdot \frac{q^b}{2}$$

operations in $\mathbb{Z}_q$*. Furthermore,*

$$a \cdot \left\lceil \frac{q^b}{2} \right\rceil + \mathrm{poly}(n) \text{ calls to } L^{(n)}_{\mathbf{s},\chi} \text{ and storage of } \left(a \cdot \left\lceil \frac{q^b}{2} \right\rceil \cdot n\right) \text{ elements in } \mathbb{Z}_q \text{ are needed.}$$

We note that the above result is a corollary to Theorem 3 which depends on a value m - a number of 'reduced' samples. However, since at present no closed form expressing m is known, the above simplified statement avoids m by restricting choices on parameters of the algorithm. We also show the following simple corollary on the algorithmic hardness of Decision-LWE.

Corollary 1 (Decision-LWE)**.** *Let* $(\mathbf{a}_i, c_i)$ *be samples following* $L^{(n)}_{\mathbf{s},\chi}$*,* $0 < b \leq n$ *be a parameter,* $0 < \epsilon < 1$ *the targeted success rate and* $a = \lceil n/b \rceil$ *the addition depth. Then, the expected cost of the BKW algorithm to distinguish* $L^{(n)}_{\mathbf{s},\chi}$ *from random with success probability* ϵ *is*

$$\left(\tfrac{q^b-1}{2}\right) \cdot \left(\tfrac{a(a-1)}{2} \cdot (n+1) - \tfrac{ba(a-1)}{4} - \tfrac{b}{6}\left((a-1)^3 + \tfrac{3}{2}(a-1)^2 + \tfrac{1}{2}(a-1)\right)\right)$$

additions/subtractions in $\mathbb{Z}_q$ to produce elimination tables,

$$m \cdot \left(\tfrac{a}{2} \cdot (n+2)\right) \text{ with } m = \epsilon / \exp\left(-\tfrac{\pi^2 \sigma^2 2^{a+1}}{q^2}\right)$$

additions/subtractions in $\mathbb{Z}_q$ to produce samples. Furthermore, $a \cdot \left\lceil \frac{q^b}{2} \right\rceil + m$ calls to $L^{(n)}_{\mathbf{s},\chi}$ and storage for $\left(\frac{q^b}{2}\right) \cdot a \cdot \left(n + 1 - b\frac{a-1}{2}\right)$ elements in $\mathbb{Z}_q$ are needed.

This corollary is perhaps the more useful result for cryptographic applications which rely on Decision-LWE and do not assume a prime modulus q. Here, we investigate the search variant first because the decision variant follows easily. However, we emphasize that there are noticeable differences in the computational costs of the two variants.

In Section 2.4, we apply the BKW algorithm to various parameter choices for LWE from the literature [86, 61, 7] and compare with alternative approaches in Section 2.5. It appears that the BKW algorithm outperforms known estimates for lattice reduction algorithms when LWE is reduced to SIS (called "Distinguishing" in [61]) starting in dimension $n \approx 250$ (but, assuming access to an unbounded number of LWE samples). However, reducing LWE to BDD (called "Decoding" in [61]) and applying a combination of lattice reduction and decoding outperforms BKW for the parameter sets considered in this chapter. However, since the concrete behaviour of lattice reduction algorithms is not fully understood, the commonly used running-time estimates tend to be optimistic. In contrast, for combinatorial algorithms such as BKW, we have a much better understanding of the concrete complexity, leading to greater confidence in the recovered bounds. Finally, we report experimental results for small instances of LWE in Section 3.5.

2.2 Preliminaries

Computational Model. We express concrete costs as computational costs and storage requirements. We measure the former in $\mathbb{Z}_q$ operations and the latter in

the number of $\mathbb{Z}_q$ elements requiring storage. However, as the hardness of LWE is related to the quantity $n \log q$ [22], relying on these measures would render results for different instances incommensurable. We hence normalise these magnitudes by considering "bit-operations" where one ring operation in $\mathbb{Z}_q$ is equivalent to $\log_2 q$ such bit operations. The specific multiplier $\log_2 q$ is derived from the fact that the majority of operations are additions and subtractions in $\mathbb{Z}_q$ as opposed to multiplications in $\mathbb{Z}_q$. In particular, we ignore the cost of "book keeping" and of fixed-precision floating point operations occuring during the algorithm (where the precision is typically a small multiple of n, cf. Section 2.4).

We make the assumption that we have unrestricted access to an LWE oracle, allowing us to obtain a large number of independent LWE samples which may not be available in practise. This assumption is usually made for combinatorial algorithms and the Arora-Ge algorithm, while lattice reduction algorithms usually require only a small number of LWE samples. However, as we discuss later, the optimal strategies for employing lattice based approaches for solving LWE appear to require executing a large number of small-advantage executions, each requiring independent LWE samples. While the cryptosystems considered in this work do not provide such an LWE oracle it is known [87] that given roughly $n \log q$ LWE samples one can produce many more LWE samples at the cost of a modest increase in the noise through inter-addition. While employing these approaches would render our proofs inadequate, it is assumed that in practice similar results would still hold. Similar notions (in the case of LPN) were considered in [37], although, as in this work, the authors did not analyse the impact of these steps.

2.3 The BKW Algorithm

The BKW algorithm was proposed by Blum, Kalai and Wasserman [21] as a method for solving the LPN problem, with sub-exponential complexity, requiring $2^{\mathcal{O}(n/\log n)}$

samples and time. The algorithm can be adapted for tackling Search- and Decision-LWE, with complexity $2^{\mathcal{O}(n)}$, when the modulus is taken to be polynomial in n.

To describe and analyse the BKW algorithm we use terminology and intuitions from linear algebra. Recall that noise-free linear systems of equations are solved by (a) transforming them into a triangular shape, (b) recovering a candidate solution in one variable for the univariate linear equation produced and (c) extending this solution by back substitution. Similarly, if we are only interested in *deciding* whether a linear system of equations does have a common solution, the standard technique is to produce a triangular basis and express other rows as linear combinations of this basis, i.e., to attempt to reduce them to zero.

The BKW algorithm – when applied to Search-LWE – can be viewed as consisting of three stages somewhat analogous to those of linear system solving:

(a) sample reduction is a form of Gaussian elimination which, instead of treating each component independently, considers 'blocks' of b components per iteration, where b is a parameter of the algorithm.

(b) hypothesis testing tests candidate sub-solutions to recover components of the secret vector $\mathbf{s}$.

(c) back substitution such that the whole process can be continued on a smaller LWE instance.

On a high-level, to aid intuition, if the standard deviation of χ was zero and hence all equations were noise-free, we could obviously recover $\mathbf{s}$ by simple Gaussian elimination. When we have non-zero noise, however, the number of row-additions conducted during Gaussian elimination result in the noise being 'amplified' to such levels that recovery of $\mathbf{s}$ would generally be impossible. Thus the motivation behind the BKW algorithm can be thought of as using a greater number of rows but eliminating many variables with single additions of rows rather than just one. If we can perform a

Gaussian elimination-like reduction of the sample matrix using few enough row additions, the resulting noise in the system is still 'low enough' to allow us to recover one or a few components of $\mathbf{s}$ at a time. While, mainly for convenience of analysis, we choose a nested-oracle approach below to define the algorithm, the above intuitive approach is essentially equivalent.

The way we study the complexity of the BKW algorithm for solving the LWE problem is closely related to the method described in [37]: given an oracle that returns samples according to the probability distribution $L^{(n)}_{\mathbf{s},\chi}$, we use the algorithm's first stage to construct an oracle returning samples according to another distribution, which we call $B^{(n)}_{\mathbf{s},\chi,a}$, where $a = \lceil n/b \rceil$ denotes the number of 'levels' of addition. The complexity of the algorithm is related to the number of operations performed in this transformation, to obtain the required number of samples for hypothesis testing. We now study the complexity of the first stage of the BKW algorithm.

2.3.1 Sample Reduction

Given $n \in \mathbb{Z}$, select a positive integer $b \leq n$ (the window width), and let $a := \lceil n/b \rceil$ (the addition depth). Given an LWE oracle (which by abuse of notation, we will also denote by $L^{(n)}_{\mathbf{s},\chi}$), we denote by $B^{(n)}_{\mathbf{s},\chi,\ell}$ a related oracle which outputs samples where the first $b \cdot \ell$ coordinates of each $\mathbf{a}_i$ are zero, generated under the distribution (which again by abuse of notation, we denote by $B^{(n)}_{\mathbf{s},\chi,\ell}$) obtained as follows:

- if $\ell = 0$, then $B^{(n)}_{\mathbf{s},\chi,0}$ is simply $L^{(n)}_{\mathbf{s},\chi}$;
- if $0 < \ell < a$, the distribution $B^{(n)}_{\mathbf{s},\chi,\ell}$ is obtained by taking the difference of two vectors from $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ that agree on the elements $(\mathbf{a}_{((\ell-1)\cdot b)}, \ldots, \mathbf{a}_{(\ell \cdot b-1)})$.

We can then describe the first stage of the BKW algorithm as the (recursively constructed) series of sample oracles $B^{(n)}_{\mathbf{s},\chi,\ell}$, for $0 \leq \ell < a$. Indeed, we define $B^{(n)}_{\mathbf{s},\chi,0}$ as the oracle which simply returns samples from $L^{(n)}_{\mathbf{s},\chi}$, while $B^{(n)}_{\mathbf{s},\chi,\ell}$ is constructed from $B^{(n)}_{\mathbf{s},\chi,\ell-1}$, for $\ell \geq 1$. We will make use of a set of tables T (maintained across oracle

calls) to store (randomly-chosen) vectors that will be used to reduce samples arising from our oracles. More explicitly, given a parameter $b \leq n$ for the window width, and letting $a = \lceil n/b \rceil$, we can describe the oracle $B^{(n)}_{\mathbf{s},\chi,\ell}$ as follows:

1. For $\ell = 0$, we can obtain samples from $B^{(n)}_{\mathbf{s},\chi,0}$ by simply calling the LWE oracle $L^{(n)}_{\mathbf{s},\chi}$ and returning the output.

2. For $\ell = 1$, we repeatedly query the oracle $B^{(n)}_{\mathbf{s},\chi,0}$ to obtain (at most) $(q^b - 1)/2$ samples $(\mathbf{a}, c)$ with distinct non-zero vectors for the first b coordinates of $\mathbf{a}$. We only collect $(q^b - 1)/2$ such vectors because we exploit the symmetry of $\mathbb{Z}_q$ and that of the noise distribution. We use these samples to populate the table T^1, indexed by the first b entries of $\mathbf{a}$. We store $(\mathbf{a}, c)$ in the table. During this course of this population, whenever we obtain a sample $(\mathbf{a}', c')$ from $B^{(n)}_{\mathbf{s},\chi,0}$, if either the first b entries of $\mathbf{a}'$ (resp. their negation) match the first b entries of a vector $\mathbf{a}$ such that the pair $(\mathbf{a}, c)$ is already in T^1, we return $(\mathbf{a}' \pm \mathbf{a}, c' \pm c)$, as a sample from $B^{(n)}_{\mathbf{s},\chi,1}$. Note that, if the first b entries in $\mathbf{a}'$ are zero, we return $(\mathbf{a}', c')$ as a sample from $B^{(n)}_{\mathbf{s},\chi,1}$. Further calls to the oracle $B^{(n)}_{\mathbf{s},\chi,1}$ proceed in a similar manner, but using (and potentially adding entries to) the same table T^1.

3. For $1 < \ell < a$, we proceed as above: we make use of the table T^ℓ (constructed by calling $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ up to $(q^b - 1)/2$ times) to reduce any output sample from $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ which has the b entries in its ℓ-th block already in T^ℓ, to generate a sample from $B^{(n)}_{\mathbf{s},\chi,\ell}$.

Pseudo-code for the oracle $B^{(n)}_{\mathbf{s},\chi,\ell}$, for $0 < \ell < a$, is given in Algorithm 3 (the case $\ell = a$ will be discussed below).

Algorithm 1: $B^{(n)}_{\mathbf{s},\chi,\ell}$ for $0 < \ell < a$

Input: b – an integer $0 < b \leq n$

Input: ℓ – an integer $0 < \ell < a$

begin
  $T^{\ell} \leftarrow$ array indexed by $\mathbb{Z}_q^b$ maintained across all runs of $B^{(n)}_{\mathbf{s},\chi,\ell}$;
  query $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ to obtain $(\mathbf{a}, c)$;
  if $\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}$ *is all zero vector* **then**
    return $(\mathbf{a}, c)$;
  while $T^{\ell}_{\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}} = \varnothing$ **and** $T^{\ell}_{-\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}} = \varnothing$ **do**
    $T^{\ell}_{\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}} \leftarrow (\mathbf{a}, c)$;
    query $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ to obtain $(\mathbf{a}, c)$;
    if $\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}$ *is all zero vector* **then**
      return $(\mathbf{a}, c)$;
  if $T^{\ell}_{\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}} \neq \varnothing$ **then**
    $(\mathbf{a}', c') \leftarrow T^{\ell}_{\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}}$;
    return $(\mathbf{a} - \mathbf{a}', c - c')$;
  else
    $(\mathbf{a}', c') \leftarrow T^{\ell}_{-\mathbf{a}_{(b\cdot(\ell-1),b\cdot\ell)}}$;
    return $(\mathbf{a} + \mathbf{a}', c + c')$;

Then, given an LWE oracle $L^{(n)}_{\mathbf{s},\chi}$ outputting samples of the form $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e)$, where $\mathbf{a}, \mathbf{s} \in \mathbb{Z}_q^n$, the oracle $B^{(n)}_{\mathbf{s},\chi,a-1}$ can be seen as another LWE oracle outputting samples of the form $(\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + e')$, where $\mathbf{a}', \mathbf{s}' \in \mathbb{Z}_q^k$, with $k = n \bmod b$, if b does not divide n, or $k = b$ otherwise, and e' is generated with a different distribution (related to the original error distribution and the value a). The vector $\mathbf{s}'$ is defined to be the last k components of $\mathbf{s}$. For the remainder of this section we will assume that $n \bmod b = 0$, and therefore $k = b$ (this is done to simplify the notation, but all the results obtained can be easily adapted when the last block has length $k < b$).

We note that, in the analysis below, we make the assumption for simplicity that all tables are completely filled during the elimination stage of the algorithm, thus giving conservative time and space bounds. In practise, especially in the final tables,

this will not be the case and birthday-paradox arguments could be applied to derive a more realistic (lower) complexity. Typically, if the number of samples required for hypothesis-testing is small, the birthday paradox implies that the storage required for the final table can be reduced by a square-root factor.

Moreover, we introduce an additional parameter $d \leq b$ which does not exist in the original BKW algorithm [21]. This parameter is used to reduce the number of hypotheses that need to be tested in the second stage of the algorithm, and arises from the fact we work with primes $q > 2$. At times, instead of working with the last block of length b, which could lead to potentially exponentially many hypotheses q^b to be tested, we may employ a final reduction phase – $B^{(n)}_{\mathbf{s},\chi,a}$ – to reduce the samples to $d < b$ non-zero entries in $\mathbf{a}$. Thus d will represent the number of components in our final block, i.e., the number of elements of the secret over which we conduct hypothesis tests. So after running the $B^{(n)}_{\mathbf{s},\chi,a-1}$ algorithm, we may decide to split the final block to obtain vectors over $\mathbb{Z}_q^d$. If so, we run the reduction function described above once more, which we will denote by $B^{(n)}_{\mathbf{s},\chi,a}$. In the simple case where we do not split the last block (i.e., $d = b$), we adopt the convention that we will also call the $B^{(n)}_{\mathbf{s},\chi,a}$ function, but it will perform no extra action (i.e., it simply calls $B^{(n)}_{\mathbf{s},\chi,a-1}$). Thus we have that for a choice of $0 \leq d \leq b$, the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ will output samples of the form $(\mathbf{a}', \langle \mathbf{a}', \mathbf{s}' \rangle + e')$, where $\mathbf{a}', \mathbf{s}' \in \mathbb{Z}_q^d$. We pick $d = 0$ in the decision variant.

On choosing b and d. In general, as discussed above, choosing the parameter d to be small (e.g. 1 or 2) leads to the best results. However, in general one could also relax the condition $d \leq b$ to $d \leq n$ where $d = n$ is equivalent to straight-forward exhaustive search. Finally, a natural question is – should all the blocks be of equal length or should some be shorter than others? Intuitively, choosing blocks which are all of equal size (or as close as possible) appears to be the optimal strategy, though we do not formally investigate this here. To ease the presentation, we assume throughout this chapter that this is the case.

From the constructions above, it follows that the cost of one call to $B^{(n)}_{\mathbf{s},\chi,\ell}$ is at most $\lceil q^b/2 \rceil$ calls to $B^{(n)}_{\mathbf{s},\chi,\ell-1}$. We also need at most one addition of two outputs of $B^{(n)}_{\mathbf{s},\chi,\ell-1}$, which have the first $b \cdot \ell$ entries in common. This operation requires $n+1-b\cdot\ell$ additions in $\mathbb{Z}_q$. Furthermore, since we maintain T^ℓ across different runs of $B^{(n)}_{\mathbf{s},\chi,\ell}$, this cost is amortised across different runs of $B^{(n)}_{\mathbf{s},\chi,\ell}$. Hence, when $\ell > 0$ the cost of calling $B^{(n)}_{\mathbf{s},\chi,\ell}$ a total of m times is upper bounded by:

$$m + \frac{q^b - 1}{2} \text{ calls to } B^{(n)}_{\mathbf{s},\chi,\ell-1} \text{ and } m \text{ additions of outputs of } B^{(n)}_{\mathbf{s},\chi,\ell-1}.$$

Overall we obtain Lemma 4.

Lemma 4. *Let $n \geq 1$ be the dimension of the LWE secret vector, q be a positive integer, and $d, b \in \mathbb{Z}$ with $1 \leq d \leq b \leq n$, and define $a = \lceil n/b \rceil$. The cost of calling $B^{(n)}_{\mathbf{s},\chi,a}$, which returns samples $(\mathbf{a}_i, c_i)$ with at most the d rightmost entries of $\mathbf{a}_i$ non-zero, m times is upper bounded by*

$$\left(\tfrac{q^b-1}{2}\right)\cdot\left(\tfrac{a(a-1)}{2}\cdot(n+1) - \tfrac{ba(a-1)}{4} - \tfrac{b}{6}\left((a-1)^3 + \tfrac{3}{2}(a-1)^2 + \tfrac{1}{2}(a-1)\right)\right) + m\cdot\left(\tfrac{a}{2}\cdot(n+2)\right) <$$

$$\left(\tfrac{q^b-1}{2}\right)\cdot\left(\tfrac{a(a-1)}{2}\cdot(n+1)\right) + m\cdot\left(\tfrac{a}{2}\cdot(n+2)\right)$$

additions in $\mathbb{Z}_q$ and $a \cdot \left\lceil \frac{q^b}{2} \right\rceil + m$ calls to $L^{(n)}_{\mathbf{s},\chi}$.

Proof. We may think of the $B^{(n)}_{\mathbf{s},\chi,\ell}$ oracles as constructing the matrix

$$B = \begin{pmatrix}
T^1 & & & & & & & & & & \\
0 & \dots & 0 & T^2 & & & & & & & \\
0 & \dots & 0 & 0 & \dots & 0 & T^3 & & & & \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & & & \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & T^{a-1} & \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & T^a \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & \dots & 0 & M
\end{pmatrix}$$

where, for $1 \leq i \leq a-1$, T^i represents a submatrix of $(q^b-1)/2$ rows and $n+1-b\cdot(i-1)$ columns (the $+1$ accounts for the c_i column). According to our convention, T^a is either a submatrix of $(q^{b-d}-1)/2$ rows and $n+1-b\cdot(a-1)$ columns if we split the last block (i.e. $d < b$), or an empty matrix. Finally M

represents a submatrix with m rows and $d+1$ columns. The matrix B has therefore at most $(a-1)\cdot((q^b-1)/2)+((q^{b-d}-1)/2)+m < a\cdot\left\lceil\frac{q^b}{2}\right\rceil+m$ rows and hence needs as many calls to $L^{(n)}_{\mathbf{s},\chi}$ to be constructed. This proves the second claim.

For the upper-bound on the number of additions necessary in $\mathbb{Z}_q$, we have the following (we treat the worst case, where full construction of all T tables is necessary before obtaining any samples from $B^{(n)}_{\mathbf{s},\chi,a}$): The construction of a T-tables is required, only $a-1$ (at most) of which require additions.

1. The construction of table T^1 requires 0 ring additions.
2. The construction of table T^2 requires at most $((q^b-1)/2)\cdot(n+1-b)$ additions.
3. The construction of table T^3 requires at most $((q^b-1)/2)\cdot((n+1-b)+(n+1-2b))$ additions.
4. In general, for $2<i<a$, the construction of table T^i requires at most
$$\left(\frac{q^b-1}{2}\right)\cdot\left((i-1)\cdot(n+1)-\sum_{j=1}^{i-1}j\cdot b\right) \quad = \quad \left(\frac{q^b-1}{2}\right)\cdot(i-1)\cdot\left((n+1)-\frac{i}{2}\cdot b\right).$$
5. The construction of T^a - the above expression is an upper bound for $i=a$.
6. Thus, the construction of all the T^i tables requires at most
$$\left(\frac{q^b-1}{2}\right)\cdot\sum_{j=2}^{a}\left((j-1)\cdot((n+1)-\frac{j}{2}\cdot b)\right)=\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\sum_{k=1}^{a-1}\frac{k(k+1)}{2}\cdot b\right)$$
$$=\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\frac{ba(a-1)}{4}-\frac{b}{6}\left((a-1)^3+\frac{3}{2}(a-1)^2+\frac{1}{2}(a-1)\right)\right)$$
additions in $\mathbb{Z}_q$.
7. Now, for the construction of our m final samples, the construction of each of these samples requires at most
$$\begin{aligned}(n+1-b)+(n+1-2b)+\ldots+(n+1-a\cdot b) &= \sum_{i=1}^{a}(n+1-ib)\\ &< a\cdot\left((n+1)-\frac{n}{2}\right)\\ &= \frac{a}{2}\cdot(n+2)\end{aligned}$$

additions (in $\mathbb{Z}_q$).

8. Thus, the number of additions (in $\mathbb{Z}_q$) incurred through calling $B^{(n)}_{\mathbf{s},\chi,a}$ m times is upper-bounded by:

$$\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\frac{ba(a-1)}{4}-\frac{b}{6}\left((a-1)^3+\frac{3}{2}(a-1)^2+\frac{1}{2}(a-1)\right)\right)+m\cdot\left(\frac{a}{2}\cdot(n+2)\right)$$

and this concludes the proof of the lemma. □

The memory requirements for storing the tables T^i are established in Lemma 5 below.

Lemma 5. *Let $n \geq 1$ be the dimension of the secret, q be a positive integer, and $d, b \in \mathbb{Z}$ with $1 \leq d \leq b \leq n$, and define $a = \lceil n/b \rceil$. The memory required to store the table T^i is upper bounded by*

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

$\mathbb{Z}_q$ elements, each of which requires $\lceil \log_2(q) \rceil$ bits of storage.

Proof. The table T^1 has $\frac{q^b}{2}$ entries each of which holds $n+1$ elements of $\mathbb{Z}_q$. The table T^2 has the same number of entries but holds on $n+1-b$ elements of $\mathbb{Z}_q$. Overall, we get that all tables together hold

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

$\mathbb{Z}_q$ elements. □

Note however that, while the original LWE oracle $L^{(n)}_{\mathbf{s},\chi}$ may output zero vectors (which offer no information for the hypothesis tests in the search variant) with probability q^{-n}, the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ may output such zero vectors with noticeable probability. In particular, calling $B^{(n)}_{\mathbf{s},\chi,a}$ m times does not guarantee that we get m samples with non-zero coefficients in $\mathbf{a}_i$. The probability of obtaining a zero vector from $B^{(n)}_{\mathbf{s},\chi,a}$ is $\frac{1}{q^d}$, and thus expect to have to call the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ around $q^d/(q^d-1)\cdot m$ times to obtain $\approx m$ useful samples with good probability.

2.3.2 Hypothesis Testing

To give concrete estimates for the time and data complexity of solving a Search-LWE instance using BKW, we formulate the problem of solving an LWE instance as the problem of distinguishing between two different distributions. Assume we have m samples in $\mathbb{Z}_q^d \times \mathbb{Z}_q$ from $B^{(n)}_{\mathbf{s},\chi,a}$. It follows that we have $\mathbb{Z}_q^d$ hypotheses to test. In what follows, we examine each hypothesis in turn and derive a hypothesised set of noise values as a result (each one corresponding to one of the m samples). We show that if we have guessed incorrectly for the subvector $\mathbf{s}'$ of $\mathbf{s}$ then the distribution of these hypothesised noise elements will be (almost) uniform while if we guess correctly then these hypothesised noise elements will be distributed according to χ_a. That is, if we have that the noise distribution associated with samples from $L^{(n)}_{\mathbf{s},\chi}$ is $\chi = \chi_{\alpha,q}$, then it follows from Lemmas 1 and 4 that the noise distribution of samples obtained from $B^{(n)}_{\mathbf{s},\chi,\ell}$ follows $\chi_{\sqrt{2^\ell}\alpha,q}$ if all inputs are independent, i.e., we are adding 2^ℓ discrete Gaussians and produce a discrete Gaussian with standard deviation increased by a factor of $\sqrt{2^\ell}$. For the sake of simplicity, we denote this distribution by χ_ℓ in the remainder of this chapter and also assume that the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ performs non-trivial operations on the output of $B^{(n)}_{\mathbf{s},\chi,a-1}$, i.e., the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ performs a further reduction step. In other words we assume that the final oracle $B^{(n)}_{\mathbf{s},\chi,a}$ results in a further increase in the standard deviation of the noise distribution associated with the final samples which are used to test hypotheses for elements of $\mathbf{s}$. We hence make the following assumption in this section:

Assumption 1. *If we let* $\mathbf{s}' := \mathbf{s}_{(n-d,n)} = (\mathbf{s}_{(n-d)}, \ldots, \mathbf{s}_{(n-1)})$*, then the output of* $B^{(n)}_{\mathbf{s},\chi,a}$ *is generated as*

$$\mathbf{a} \leftarrow_\$ \mathbb{Z}_q^d, e \leftarrow_\$ \chi_a : (\mathbf{a}, \langle \mathbf{a}, \mathbf{s}' \rangle + e).$$

Remark: This section only refers to the Search-LWE problem in which we assume q is prime for ease of analysis and exposition. This restriction does not apply to our results below on the decision variant.

For our hypothesis-testing strategies, we think of each of the samples returned by $B^{(n)}_{\mathbf{s},\chi,a}$ as giving rise to many equations

$$f_i = -c_i \pm j + \sum_{k=0}^{d-1} (\mathbf{a}_i)_{(k)} x_{(k)} \text{ for } 0 \leq j < q/2.$$

Given a number of these samples, in order to get an estimate for $\mathbf{s}'$, we run through q^d hypotheses and compute an array of scores S indexed by the possible guesses in $\mathbb{Z}_q^d$. That is, a function W assigns a weight to elements in $\mathbb{Z}_q$ which represent the noise under the hypothesis $\mathbf{s}' = \mathbf{v}$. For each guess $\mathbf{v}$ we sum over the weighted noises $W(-c_i + \sum_{k=0}^{d-1} (\mathbf{a}_i)_{(k)} \cdot v_{(k)})$. If W is such that the counter $S_\mathbf{v}$ grows proportionally to the likelihood that $\mathbf{v}$ is the correct guess, then the counter $S_{\mathbf{s}'}$ will grow fastest. Pseudo-code is given in Algorithm 2.

Algorithm 2: Analysing candidates.

Input: F – a set of m samples following $B^{(n)}_{\mathbf{s},\chi,a}$
Input: W – a weight function mapping members of $\mathbb{Z}_q$ to real numbers

```
1  begin
2      S ← array filled with zeros indexed by Z_q^d;
3      for v ∈ Z_q^d do
4          w_v ← ∅;
5          for f_i ∈ F do
6              write f_i as −c_i + Σ_{k=0}^{d−1} (a_i)_(k) · x_(k);
7              j ← ⟨a_i, v⟩ − c_i;
8              w_v ← w_v ∪ {W(j)};
9          S_v ← Σ_{w_i ∈ w_v} w_i/m;
10     return S
```

Lemma 6. *Running hypothesis testing costs $m \cdot q^d$ operations in $\mathbb{Z}_q$.*

Proof. Evaluating $\langle \mathbf{a}_i, \mathbf{v} \rangle - c_i$ at some point in $\mathbb{Z}_q^d$ naively costs $2d$ operations in $\mathbb{Z}_q$ which implies an overall cost of $2d \cdot m \cdot q^d$. However, we can reorder the elements in $\mathbb{Z}_q^d$ such that the element at index h differs from the element at index $h+1$ by an addition of a unit vector in $\mathbb{Z}_q^d$. Evaluating $\langle \mathbf{a}_i, \mathbf{v} \rangle - c_i$ on all $\mathbb{Z}_q^d$ points ordered in such a way reduces to one operation in $\mathbb{Z}_q$: addition of $\mathbf{a}_{i,(j)}$ where j is the index at

which two consecutive elements differ. Hence, Algorithm 2 costs $m \cdot q^d$ operations in $\mathbb{Z}_q$. □

Recall that χ_a is the distribution of the errors under a right guess. We define $\mathcal{U}_a(\mathbf{v})$ to be the distribution of errors under a wrong guess $\mathbf{v} \neq \mathbf{s}'$.

The Neyman-Pearson lemma states the following: Given two point hypotheses $\mathrm{H}_0 \colon \theta = \theta'$ and $\mathrm{H}_1 \colon \theta = \theta''$, then, given a set of i.i.d. random variables $\{X_1, \ldots, X_n\}$ the most powerful test of which distribution the random variables adhere to is the likelihood ratio test in which we consider:

$$\frac{L(\theta' \colon \{X_1, \ldots, X_n\})}{L(\theta'' \colon \{X_1, \ldots, X_n\}))} \tag{2.1}$$

Now, since H_1 is not a point set in our case, the Neyman-Pearson lemma cannot be applied directly. Instead, in a simplifying step, we use Neyman-Pearson as the motivation for taking the average of all possible denominators in 2.1. We can heuristically justify this approach as most denominators are close to this average.

Now, when dealing with collections of i.i.d. random variables, the likelihood function L can generally be factored into a product of likelihood functions for the individual random variables, hence it is more convenient to use the *log-likelihood function*, allowing us to sum the individual log-likelihood values. Thus, after fixing a guess for $\mathbf{s}'$, we obtain a corresponding set of hypothesised error values j_i, $\lceil -q/2 \rceil \leq j_i \leq \lfloor q/2 \rfloor$ - for each one we can define the statistic $W(j_i)$ to be

$$W(j_i) := \log_2 \left(\frac{\Pr[e \leftarrow_\$ \chi_a : e = j_i]}{\mathbf{E}(\Pr[e \leftarrow_\$ \mathcal{U}_a(\mathbf{v}) : e = j_i]}) \right). \tag{2.2}$$

In our setting, our adaptation of the Neyman-Pearson lemma indicates that the sum of the statistics W provides the most powerful test of which distribution the X_i follow and hence we use this sum as our final distinguishing statistic. Thus, the remainder of this section will be used to establish the distribution of $W(j_i)$ in the cases of right and wrong guesses, with the goal of establishing the conditions under which we can identify the correct guess by way of the respective values of $W(j_i)$.

We firstly examine the relationships between $\tilde{p}_j := \mathbf{E}(\Pr[e \leftarrow_\$ \mathcal{U}_a(\mathbf{v}) : e = j]$ and $p_j := \Pr[e \leftarrow_\$ \chi_a : e = j])$ and then proceed to examine the distribution of our final distinguishing characteristic in the cases of correct and incorrect guesses, allowing us to explicitly quantify the power of our final distinguisher in terms of m.

Our principal assumption in this section is that the sums of the statistics $W(j_i)$ behave according to the central limit theorem, allowing us to derive approximate Gaussian distribution descriptions of the sums of these statistics in both the correct and incorrect guess cases. Under this assumption, if enough samples are available to us then, with very high probability, we can easily distinguish between members of these Gaussian distributions, allowing us to isolate the correct value of $\mathbf{s}'$.

Lemma 7. *Given a wrong guess* $\mathbf{v}$ *for* $\mathbf{s}'$*, for each element* $f_i = -c_i + \sum_{k=0}^{d-1} (\mathbf{a}_i)_{(k)} x_{(k)} \in \mathbb{Z}_q[x]$*, with* $c_i = \langle \mathbf{a}_i, \mathbf{s}' \rangle - e_i$*, the probability of error* j *appearing is*

$$\tilde{p}_j := \mathbf{E}(\Pr[e \leftarrow_\$ \mathcal{U}_a(\mathbf{v}) : e = j]) = \frac{q^{d-1} - p_j}{q^d - 1} \tag{2.3}$$

if q *is prime.*

Proof. For an incorrect guess $\mathbf{v}$, $\Pr[e \leftarrow_\$ \mathcal{U}_a(\mathbf{v}) : e = j] = p_{j - \mathbf{a}_i^T(\mathbf{s}' - \mathbf{v})}$. Thus

$$\begin{aligned}
\tilde{p}_j &= \mathbf{E}(p_{j - \mathbf{a}_i^T(\mathbf{s}' - \mathbf{v})}) = \sum_k p_{j-k} \Pr[\mathbf{a}_i^T(\mathbf{s}' - \mathbf{v}) = k] \\
&= p_j \Pr[\mathbf{a}_i^T(\mathbf{s}' - \mathbf{v}) = 0] + \sum_{k \neq 0} p_{j-k} \Pr[\mathbf{a}_i^T(\mathbf{s}' - \mathbf{v}) = k] \\
&= p_j \frac{q^{d-1} - 1}{q^d - 1} + \sum_{k \neq 0} p_{j-k} \frac{q^{d-1}}{q^d - 1} \\
&= \frac{q^{d-1} - p_j}{q^d - 1}
\end{aligned}$$

□

For the final back-substitution stage, we wish to ensure that the score for the correct guess $\mathbf{v} = \mathbf{s}'$ is highest among the entries of S. Thus, what remains to be established is the size $m = |F|$ needed such that the score for the right guess $\mathbf{v} = \mathbf{s}'$

is the highest. Under our sample independence assumptions, by the Central Limit theorem, the distribution of $S_{\mathbf{v}}$ approaches a Normal distribution as m increases. Hence, for sufficiently large m we may approximate the discrete distribution $S_{\mathbf{v}}$ by a normal distribution [12]. If $\mathcal{N}(\mu, \sigma^2)$ denotes a Normal distribution with mean μ and standard deviation σ we denote the distribution for $\mathbf{v} = \mathbf{s}'$ by $D_c = \mathcal{N}(\mathrm{E}_c, \mathrm{Var}_c)$ and for $\mathbf{v} \neq \mathbf{s}'$ by $D_w = \mathcal{N}(\mathrm{E}_w, \mathrm{Var}_w)$. Recall that we now wish to determine the number of samples m necessary to allow us to distinguish between variates following D_w and those following D_c.

Establishing m hence first of all means establishing $\mathrm{E}_c, \mathrm{E}_w, \mathrm{Var}_c$ and Var_w. We start with E_c.

Lemma 8. *Let* $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ *be samples following* $B^{(n)}_{\mathbf{s},\chi,a}$, q *be a positive integer,* $\mathbf{v} \in \mathbb{Z}_q^d$, $p_j := \Pr(e \leftarrow_{\$} \chi_a : e = j)$, $w_j := W(j)$ *and* $S_{\mathbf{v}} = \frac{1}{m}\sum_{i=0}^{m-1} W(\langle \mathbf{a}_i, \mathbf{v}\rangle - c_i)$. *When* $\mathbf{v} = \mathbf{s}'$, $E(S_{\mathbf{v}})$ *is given by:*

$$E_c = E(S_{\mathbf{v}} \mid \mathbf{v} = \mathbf{s}') = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j w_j = p_0 w_0 + 2 \cdot \sum_{j=1}^{j=\lfloor q/2 \rfloor} p_j w_j. \tag{2.4}$$

Proof. First, we remark that:

$$\Pr[\langle \mathbf{a}_i, \mathbf{s}'\rangle = c_i + u] \quad = \quad \Pr[\langle \mathbf{a}_i, \mathbf{s}'\rangle = \langle \mathbf{a}_i, \mathbf{s}'\rangle + e_i + u] = \Pr[-e_i = u] = p_u.$$

The expected value for $S_{\mathbf{v}}$ in the case of a correct guess is then given by:

$$\mathrm{E}_c := \mathrm{E}(S_{\mathbf{v}} \mid \mathbf{v} = \mathbf{s}') \quad = \quad \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} \Pr[e_i = j] \cdot W(j) = \sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j w_j.$$

Finally, for all $j, 1 \leq j \leq \lfloor q/2 \rfloor$, we have $p_{-j} w_{-j} = p_j w_j$. Thus:

$$\sum_{j=\lceil -q/2 \rceil}^{\lfloor q/2 \rfloor} p_j w_j = p_0 w_0 + 2 \cdot \sum_{j=1}^{j=\lfloor q/2 \rfloor} p_j w_j.$$

□

We now examine $\mathrm{E}_w = \mathrm{E}(S_{\mathbf{v}} \mid \mathbf{v} \neq \mathbf{s}')$. To begin with, we fix a wrong guess $\mathbf{v}$, such that $\mathbf{v} = \mathbf{s}' + \mathbf{t}$ with $\mathbf{t} \neq 0$.

Lemma 9. *Let* $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ *be samples following* $B^{(n)}_{\mathbf{s},\chi,a}$, q *be a positive integer,* $\mathbf{v} \in \mathbb{Z}_q^d$, $\tilde{p}_j := \mathbf{E}[\Pr(e \leftarrow_{\$} \mathcal{U}_a(\mathbf{v}) : e = j)]$, $p_j := \Pr(e \leftarrow_{\$} \chi_a : e = j)$, $w_j := W(j)$, *and* $S_{\mathbf{v}} = \frac{1}{m}\sum_{i=0}^{m-1} W(\langle \mathbf{a}_i, \mathbf{v}\rangle - c_i)$. *If* $\mathbf{v} \neq \mathbf{s}'$, *we have:*

$$E_w = E(S_{\mathbf{v}} \mid \mathbf{v} \neq \mathbf{s}') = \sum_{j=\lceil -q/2 \rfloor}^{\lfloor q/2 \rfloor} \tilde{p}_j w_j = \sum_{j=\lceil -q/2 \rfloor}^{\lfloor q/2 \rfloor} \frac{q^{d-1} - p_j}{q^d - 1} w_j. \tag{2.5}$$

Since the proof of Lemma 9 is analogous to Lemma 8 we omit it here. We now look at the variances Var_c and Var_w.

Lemma 10. *Let* $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ *be samples following* $B^{(n)}_{\mathbf{s},\chi,a}$, q *be a positive integer,* $\mathbf{v} \in \mathbb{Z}_q^d$, $p_j := \Pr(e \leftarrow_{\$} \chi_a : e = j)$, $\tilde{p}_j := \mathbf{E}[\Pr(e \leftarrow_{\$} \mathcal{U}_a(\mathbf{v}) : e = j)]$, $w_j := W(j)$ *and* $S_{\mathbf{v}} = \sum_{i=0}^{m-1} \frac{1}{m} W(\langle \mathbf{a}_i, \mathbf{v}\rangle - c_i)$.

If $\mathbf{v} = \mathbf{s}'$, *then*

$$\mathrm{Var}_c := \mathrm{Var}(S_{\mathbf{v}} \mid \mathbf{v} = \mathbf{s}') = \frac{1}{m} \sum_{j=\lceil -q/2 \rfloor}^{\lfloor q/2 \rfloor} p_j \cdot (w_j - E_c)^2. \tag{2.6}$$

If $\mathbf{v} \neq \mathbf{s}'$, *then*

$$\mathrm{Var}_w := \mathrm{Var}(S_{\mathbf{v}} \mid \mathbf{v} \neq \mathbf{s}') = \frac{1}{m} \sum_{j=\lceil -q/2 \rfloor}^{\lfloor q/2 \rfloor} \tilde{p}_j \cdot (w_j - E_w)^2. \tag{2.7}$$

Proof. In the case of $\mathbf{v} = \mathbf{s}'$ we have that for $m = 1$,

$$\mathrm{Var}_c = \sum_{j=\lceil -q/2 \rfloor}^{\lfloor q/2 \rfloor} p_j \cdot \big(w_j - \mathrm{E}_c\big)^2.$$

In the case of adding then normalising m samples we can use the fact that when adding random variables of zero covariance, the sum of the variances is the variance of the sum. Thus the variance in the case of adding m samples and normalising is given by:

$$\mathrm{Var}_c = m \cdot \sum_{j=\lceil -q/2 \rfloor}^{\lfloor q/2 \rfloor} p_j \cdot \left(\frac{w_j}{m} - \frac{\mathrm{E}_c}{m}\right)^2 = \frac{1}{m} \sum_{j=\lceil -q/2 \rfloor}^{\lfloor q/2 \rfloor} p_j \cdot (w_j - \mathrm{E}_c)^2$$

A similar argument holds in the case of Var_w. □

Finally, given E_c, E_w, Var_c, and Var_w, we can estimate the rank of the right secret in dependence of the number of samples m considered. We denote by Y_h the random variable determined by the rank of a correct score $S_{\mathbf{s}'}$ in a list of h elements. Now, for a list of length q^d and a given rank $0 \leq r < q^d$, the probability of Y_{q^d} taking rank r is given by a binomial-normal compound distribution. Finally, we get Lemma 11, which essentially states that for whatever score the right secret gets, in order for it to have rank zero the remaining $q^d - 1$ secrets must have smaller scores.

Lemma 11. *Let E_c, E_w, Var_c and Var_w be as in Lemmas 8, 9 and 10. Let also Y_{q^d} be the random variable determined by the rank of a correct score $S_{\mathbf{s}'}$ in the list S of q^d elements. Then, the number of samples m required for Y_{q^d} to take rank zero with probability ϵ' is recovered by solving*

$$\epsilon' = \int_x \left[\frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x - E_w}{\sqrt{2\mathrm{Var}_w}}\right)\right)\right]^{(q^d-1)} \cdot \left(\frac{1}{\sqrt{2\pi\mathrm{Var}_c}} e^{-\frac{(x-E_c)^2}{2\mathrm{Var}_c}}\right) \mathrm{d}x,$$

for m.

Proof. Y_{q^d} follows a binomial-normal compound distribution given by $\Pr[Y_{q^d} = r] =$

$$\int_x \left(\binom{q^d - 1}{r} \cdot \Pr[e \leftarrow_\$ D_w : e \geq x]^r \cdot \Pr[e \leftarrow_\$ D_w : e < x]^{(q^d-r-1)} \cdot \Pr[e \leftarrow_\$ D_c : e = x]\right) \mathrm{d}x.$$

Plugging in $r = 0$ and $\Pr[Y_{q^d} = r] = \epsilon'$ we get:

$$\begin{aligned} \epsilon' &= \int_x \Pr[e \leftarrow_\$ D_w : e < x]^{(q^d-1)} \cdot \Pr[e \leftarrow_\$ D_c : e = x] \, \mathrm{d}x \\ &= \int_x \left[\frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x - \mathrm{E}_w}{\sqrt{2\mathrm{Var}_w}}\right)\right)\right]^{(q^d-1)} \cdot \left(\frac{1}{\sqrt{2\pi\mathrm{Var}_c}} e^{-\frac{(x-\mathrm{E}_c)^2}{2\mathrm{Var}_c}}\right) \mathrm{d}x \end{aligned}$$

as required. □

Using Lemma 11 we can hence estimate the number of non-zero samples m we need to recover subvector $\mathbf{s}'$.

Remark: We note that Algorithm 2 not only returns an ordering of the hypotheses

but also a score for each hypothesis. Hence, we can simply sample from $B^{(n)}_{\mathbf{s},\chi,a}$ until the distance between the first and second highest rated hypothesis is above a certain threshold.

2.3.3 Back-Substitution

Given a candidate solution for $\mathbf{s}'$ which is correct with very high probability we can perform back-substitution in our tables T^i similarly to solving a triangular linear system. It is easy to see that back-substitution costs $2d$ operations per row. Furthermore, by Lemma 4 we have $a \cdot (\lceil q^b/2 \rceil)$ rows in all tables T^i.

After back-substitution, we start the BKW algorithm again in stage one where all the tables T^i are already filled. To recover the next d components of $\mathbf{s}$ then, we ask for m fresh samples which are reduced using our modified tables T^i and perform hypothesis testing on these m samples.

2.3.4 Complexity of BKW

We can now state the main theorem of this chapter.

Theorem 3 (Search-LWE). *Let $(\mathbf{a}_i, c_i)$ be samples following $L^{(n)}_{\mathbf{s},\chi}$, $0 < b \leq n$, $d \leq b$ parameters, $0 < \epsilon < 1$ the targeted success rate and q prime. Let $a = \lceil n/b \rceil$ and m be as in Lemma 11 when $\epsilon' = (\epsilon)^{1/\lceil n/d \rceil}$. Then, the expected cost of the BKW algorithm to recover $\mathbf{s}$ with success probability ϵ is*

$$\left(\frac{q^b - 1}{2}\right) \cdot \left(\frac{a(a-1)}{2} \cdot (n+1) - \frac{ba(a-1)}{4} - \frac{b}{6}\left((a-1)^3 + \frac{3}{2}(a-1)^2 + \frac{1}{2}(a-1)\right)\right) \tag{2.8}$$

additions/subtractions in $\mathbb{Z}_q$ to produce the elimination tables,

$$\frac{q^d}{q^d - 1} \cdot \frac{\lceil \frac{n}{d} \rceil + 1}{2} \cdot m \cdot \left(\frac{a}{2} \cdot (n+2)\right) \tag{2.9}$$

additions/subtractions in $\mathbb{Z}_q$ to produce samples for hypothesis testing. For the hypothesis-testing step

$$\left\lceil \frac{n}{d} \right\rceil \cdot \left(m \cdot q^d\right) \tag{2.10}$$

arithmetic operations in $\mathbb{Z}_q$ are required and

$$\left(\left\lceil \frac{n}{d} \right\rceil + 1\right) \cdot d \cdot a \cdot \left\lceil \frac{q^b}{2} \right\rceil \tag{2.11}$$

operations in $\mathbb{Z}_q$ for back-substitution. Furthermore,

$$a \cdot \left\lceil \frac{q^b}{2} \right\rceil + \frac{q^d}{q^d - 1} \cdot \left\lceil \frac{n}{d} \right\rceil \cdot m \tag{2.12}$$

calls to $L^{(n)}_{\mathbf{s},\chi}$ and storage for

$$\left(\frac{q^b}{2}\right) \cdot a \cdot \left(n + 1 - b\frac{a-1}{2}\right) \tag{2.13}$$

elements in $\mathbb{Z}_q$ are needed.

Proof. In order to recover $\mathbf{s}$ we need every run of stage 1 to be successful, hence we have $\epsilon = (\epsilon')^{\lceil n/d \rceil}$ and consequently $\epsilon' = (\epsilon)^{1/\lceil n/d \rceil}$.

Furthermore, we have:

- The cost of constructing the tables T^i in Equation (2.8) follows from Lemma 4.
- Lemma 4 and the fact that with probability $\frac{1}{q^d}$ the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ returns an all-zero sample establish that to produce m non-zero samples for hypothesis testing, $\frac{q^d}{q^d-1} \cdot m \cdot \left(\frac{a}{2} \cdot (n+2)\right)$ operations are necessary. We need to produce m such samples $\left\lceil \frac{n}{d} \right\rceil$ times. However, as we proceed the number of required operations linearly approaches zero. Hence, we need $\frac{\left\lceil \frac{n}{d} \right\rceil + 1}{2} \cdot \frac{q^d}{q^d-1} \cdot m \cdot \left(\frac{a}{2} \cdot (n+2)\right)$ operations as in Equation (2.9).
- The cost of Algorithm 2 in Equation (2.10) which also is run $\left\lceil \frac{n}{d} \right\rceil$ times follows from Lemma 6.
- There are $a \cdot \left\lceil \frac{q^b}{2} \right\rceil$ rows in all tables T^i each of which requires $2d$ operations in back-substitution. We need to run back-substitution $\left\lceil \frac{n}{d} \right\rceil$ times, but each time the cost decreases linearly. From this follows Equation (2.11).
- The number of samples needed in Equation (2.12) follows from Lemma 4 and that with probability $\frac{1}{q^d-1}$ the oracle $B^{(n)}_{\mathbf{s},\chi,a}$ returns a sample which is useless to us.

- The storage requirement in Equation (2.13) follows from Lemma 5.

□

We would like to express the complexity of the BKW algorithm as a function of n, q, α explicitly. In this regard, Theorem 3 does not deliver yet. However, from the fact that we can distinguish χ_a and $\mathcal{U}_a$ in subexponential time if the standard deviation $\sqrt{2^a}\alpha q < q/2$ (i.e, the standard deviation of the discrete Gaussian distribution over $\mathbb{Z}$ corresponding to χ_a), we can derive the following simple corollary eliminating m.

Corollary 2. *Let $(\mathbf{a}_i, c_i)$ be samples following $L^{(n)}_{\mathbf{s},\chi}$, set $a = \lfloor \log_2(1/(2\alpha)^2) \rceil$, $b = n/a$ and q a prime. Let d be a small constant $0 < d < \log_2(n)$. Assume α is such that $q^b = q^{n/a} = q^{n/\lfloor \log_2(1/(2\alpha)^2) \rceil}$ is superpolynomial in n. Then, given these parameters, the cost of the BKW algorithm to solve* Search-LWE *is*

$$\left(\frac{q^b - 1}{2}\right) \cdot \left(\frac{a(a-1)}{2} \cdot (n+1)\right) + \left\lceil \frac{q^b}{2} \right\rceil \cdot \left(\left\lceil \frac{n}{d} \right\rceil + 1\right) \cdot d \cdot a + \mathrm{poly}(n) \approx \left(a^2 n\right) \cdot \frac{q^b}{2}$$

operations in $\mathbb{Z}_q$. Furthermore,

$$a \cdot \left\lceil \frac{q^b}{2} \right\rceil + \mathrm{poly}(n) \text{ calls to } L^{(n)}_{\mathbf{s},\chi} \text{ and storage of } \left(a \cdot \left\lceil \frac{q^b}{2} \right\rceil \cdot n\right) \text{ elements in } \mathbb{Z}_q \text{ are needed.}$$

Proof. From the condition $\sqrt{2^a} \cdot \alpha \cdot q < q/2$ follows that we must set $a = \log_2(1/(2\alpha)^2)$. If a is set this way we have that we can distinguish χ_a from $\mathcal{U}(\mathbb{Z}_q)$ in $\mathrm{poly}(n)$. Now, since q^b is superpoynomial in n we have that $m \leq q^b$ and Theorem 3 is dominated by terms involving q^b. □

In many cryptographic applications solving the Decision-LWE problem is equivalent to breaking the cryptographic assumption. Furthermore, in many such constructions q may not be a prime. Hence, we also establish the cost of distinguishing $L^{(n)}_{\mathbf{s},\chi}$ from random with a given success probability for arbitrary moduli q.

Corollary 3 (Decision-LWE)**.** *Let $(\mathbf{a}_i, c_i)$ be samples following $L^{(n)}_{\mathbf{s},\chi}$, $0 < b \leq n$ be a parameter, $0 < \epsilon < 1$ the targeted success rate and $a = \lceil n/b \rceil$ the addition depth.*

Then, the expected cost of the BKW algorithm to distinguish $L^{(n)}_{\mathbf{s},\chi}$ *from random with success probability* ϵ *is*

$$\left(\frac{q^b-1}{2}\right)\cdot\left(\frac{a(a-1)}{2}\cdot(n+1)-\frac{ba(a-1)}{4}-\frac{b}{6}\left((a-1)^3+\frac{3}{2}(a-1)^2+\frac{1}{2}(a-1)\right)\right) \tag{2.14}$$

additions/subtractions in $\mathbb{Z}_q$ *to produce elimination tables,*

$$m\cdot\left(\frac{a}{2}\cdot(n+2)\right) \text{ with } m=\epsilon/\exp\left(-\frac{\pi^2\sigma^2 2^{a+1}}{q^2}\right) \tag{2.15}$$

additions/subtractions in $\mathbb{Z}_q$ *to produce samples. Furthermore,*

$$a\cdot\left\lceil\frac{q^b}{2}\right\rceil+m \tag{2.16}$$

calls to $L^{(n)}_{\mathbf{s},\chi}$ *and storage for*

$$\left(\frac{q^b}{2}\right)\cdot a\cdot\left(n+1-b\frac{a-1}{2}\right) \tag{2.17}$$

elements in $\mathbb{Z}_q$ *are needed.*

Proof. No hypothesis testing, back-substitution and accounting for all zero samples is necessary and hence any terms referring to those can be dropped from Theorem 3. Choosing $m=\exp\left(-\frac{\pi^2\sigma^2 2^{a+1}}{q^2}\right)/\epsilon$ leads to a distinguishing advantage of ϵ (cf. [61]). □

2.4 Applications

In this section we apply Theorem 3 to various sets of parameters suggested in the literature. In order to compute concrete costs we rely on numerical approximations in various places such as the computation of p_j. We used $2n-4n$ bits of precision for all computations, increasing this precision further did not appear to change our results. The solving step for m of Lemma 11 is accomplished by a simple search implemented in Sage [97]. As a subroutine of this search we rely on numerical integration which we performed using the mpmath library [50] as shipped with Sage.

In all cases below we always set $\epsilon = 0.99$ and $a := \lceil t \cdot \log_2 n \rfloor$ where t is a small constant, which is consistent with the complexity of the BKW algorithm $q^{\mathcal{O}(n/\log_2(n))} = 2^{\mathcal{O}(n)}$ if $q \in \text{poly}(n)$.

2.4.1 Regev's Original Parameters

In [86] Regev proposes a simple public-key encryption scheme with the suggested parameters $q \approx n^2$ and $\alpha = 1/(\sqrt{n} \cdot \log_2^2 n\sqrt{2\pi})$. We consider the parameters in the range $n = 32, \ldots, 256$. In our experiments $t = 3.0$ produced the best results, i.e., higher values of t resulted in m growing too fast. Plugging these values into the formulas of Theorem 3 we get an overall complexity of

$$\frac{mn^9 + \frac{1}{6}\,2^{\frac{2}{3}n}n^5 + \left[\left(3\,n + \frac{9}{2}\right) \cdot \left(2^{\frac{2}{3}n}n^4 + 1\right)\right] \log_2\left(n\right)^2 + \frac{1}{6}\,n}{2\left(n^4 - 1\right)}$$

operations in $\mathbb{Z}_q$ after simplification. If $m < 2^{(\frac{2}{2.6}n)}$ then this expression is dominated by

$$\frac{\frac{1}{6}n^5 + \left(3\,n^5 + \frac{9}{2}n^4\right) \cdot \log_2\left(n\right)^2}{2\left(n^4 - 1\right)} 2^{\frac{2}{3}n} \in 2^{\frac{2}{3}n + \mathcal{O}(\log n)}.$$

However, since we compute m numerically, we have to rely concrete values for various n to verify that with these settings indeed m does not grow too fast. Table 2.1 lists the estimated number of calls to $L_{\mathbf{s},\chi}^{(n)}$ ("$\log_2 \#L_{\mathbf{s},\chi}^{(n)}$"), the estimated number of required ring ("$\log_2 \#\mathbb{Z}_q$") and bit ("$\log_2 \#\mathbb{Z}_2$") operations, the costs in terms of ring operations for each of the three stages sampling, hypothesis testing and back substitution.

2.4.2 Lindner and Peikert's Parameters

In [61], Lindner and Peikert propose new attacks and parameters for LWE. Table 2.2 lists concrete costs of the BKW algorithm for solving LWE under the parameter choices from [61] as interpreted in the LWE instance generator (Section 1.8). In our computations $t = 2.7$ produced the best results, i.e., higher values of t resulted in m growing too fast.

n	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$ in				$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
		sample	hypo.	subs.	total		
32	19.93	32.76	34.94	29.31	35.25	38.57	25.64
48	28.90	43.70	45.66	40.69	46.02	49.50	35.81
64	34.22	54.36	52.22	51.86	54.85	58.43	45.87
80	42.19	65.50	61.16	62.94	65.78	69.44	56.60
96	49.83	76.52	69.58	73.91	76.75	80.47	67.31
112	58.79	87.51	79.22	84.84	87.72	91.49	78.02
128	67.44	98.46	88.44	95.75	98.67	102.48	88.74
144	76.40	109.35	97.91	106.61	109.56	113.40	99.43
160	86.37	120.23	108.34	117.46	120.43	124.30	110.12
176	97.34	131.09	119.71	128.29	131.28	135.18	120.82
192	106.30	141.93	129.06	139.10	142.12	146.04	131.51
208	117.27	152.76	140.37	149.91	152.95	156.89	142.20
224	128.56	163.57	151.98	160.70	163.76	167.72	152.88
240	139.52	174.37	163.24	171.48	174.56	178.54	163.57
256	150.49	185.17	174.49	182.26	185.35	189.35	174.25

Table 2.1: Cost of solving Search-LWE for parameters suggested in [86] with $d = 1, t = 3, \epsilon = 0.99$ with BKW.

2.4.3 Albrecht et al.'s Polly-Cracker

In [7] a somewhat homomorphic encryption scheme is proposed based on the hardness of computing Gröbner bases with noise. Using linearisation the equation systems considered in [7] may be considered as LWE instances. Table 2.3 lists concrete costs for recovering the secret Gröbner basis using this strategy for selected parameters suggested in [7]. In Table 2.3 "λ" is the targeted bit-security level and n the number of variables in the linearised system. We note that we did not exploit the structure of the secret for Table 2.3.

2.5 Comparison with Alternative Approaches

Now, given the complexity estimates in Section 2.4 we may ask how these relate to existing approaches in the literature. Hence, we briefly describe some alternative

n	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$ in				$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
		sample	hypo.	subs.	total		
32	7.64	35.91	33.65	33.92	36.46	39.78	28.84
48	9.97	45.75	36.56	43.58	46.04	49.52	37.94
64	15.61	54.82	42.62	52.53	55.09	58.67	46.49
80	22.25	63.39	49.58	61.02	63.65	67.31	54.66
96	30.90	71.62	58.49	69.18	71.86	75.58	62.57
112	40.86	79.57	68.68	77.08	79.81	83.58	70.25
128	54.15	87.32	82.16	84.78	87.58	91.39	77.76
144	37.54	102.31	67.71	99.73	102.53	106.37	92.54
160	46.51	110.38	76.83	107.77	110.60	114.47	100.43
176	56.47	118.31	86.93	115.68	118.53	122.43	108.20
192	70.76	126.13	101.35	123.47	126.34	130.26	115.87
208	80.73	133.84	111.43	131.15	134.05	137.99	123.44
224	94.34	141.45	125.15	138.74	141.66	145.62	130.92
240	109.62	148.98	140.53	146.25	149.18	153.16	138.33
256	126.23	156.42	157.24	153.68	157.96	161.96	145.67

Table 2.2: Cost of solving Search-LWE for parameters suggested in [61] with $d = 1, t = 2.7, \epsilon = 0.99$ with BKW.

λ	n	q	α	t	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$ in				$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
						sample	hypo.	subs.	total		
80	136	1999	0.005582542...	2.2	93.58	109.40	121.59	105.71	121.59	125.04	100.23
	231	92893	0.000139563...	3.4	127.23	157.47	167.09	154.40	167.09	171.13	146.54
128	153	12227	0.002797408...	2.4	84.05	132.07	117.45	129.66	132.32	136.08	122.39
	253	594397	0.000034967...	3.8	100.66	175.15	146.00	171.88	175.29	179.55	163.89

Table 2.3: Cost of finding $G \approx \mathbf{s}$ for parameters suggested in [7] with $d = 2, \epsilon = 0.99$.

strategies for solving the LWE problem.

2.5.1 Short Integer Solutions: Lattice Reduction

Table 2.4 compares the number of bit and ring operations using the BKW and BKZ algorithm as described in [61] i.e. obtaining a short vector $\mathbf{v}$ in the (scaled) dual lattice allows to distinguish between LWE samples and uniform with advantage (approximately) $\exp(-\pi \cdot (\|\mathbf{v}\| \cdot \alpha \cdot \sqrt{2\pi})^2)$. In Table 2.4 running times and the number

of required samples for BKZ include the $1/\epsilon$ factor, hence both approaches distinguish with probability close to 1.

n	q	αq	BKW					NTL-BKZ Lindner/Peikert Model				
			t	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$	$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$	$\log_2 \epsilon$	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$	$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
Regev [86]												
128	16411	11.81	3.2	81.62	93.84	97.65	83.85	-18	26.47	61.56	65.36	26.47
256	65537	25.53	3.1	126.26	179.76	183.76	168.79	-29	38.50	175.48	179.48	38.50
512	262147	57.06	3.1	337.92	350.80	354.97	338.02	-48	58.52	386.75	390.92	58.52
Lindner & Peikert [61]												
128	2053	2.70	2.9	63.86	82.40	85.86	72.73	-18	26.25	54.50	57.96	26.25
256	4099	3.34	2.8	105.08	151.45	155.04	140.64	-29	38.22	156.18	159.77	38.22
512	4099	2.90	2.6	157.78	278.01	281.59	266.14	-50	60.14	341.87	345.45	60.14

Table 2.4: Cost of solving Decision-LWE with BKZ as in [61] and BKW as in Corollary 3.

Hence, Table 2.4 illustrates that for the families of parameters considered here, we expect the BKW algorithm to be asymptotically faster than the BKZ algorithm with a crossover around $n = 250$ at the cost of requiring a lot more samples and memory.

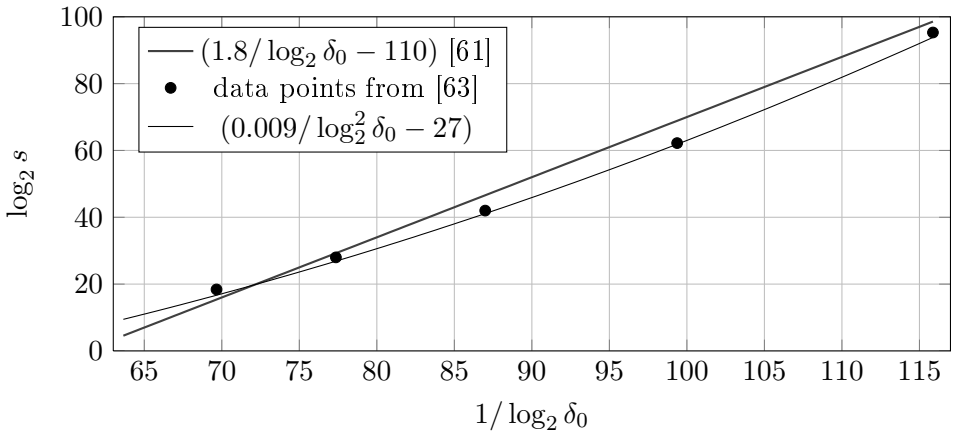

Figure 2.1: BKZ running times in seconds s for given values of δ_0.

Table 2.4 contains analogous approximations in which the BKZ entries are obtained using the non-linear BKZ 2.0 model.

n	q	αq	BKW					BKZ 2.0 Simulator Model				
			t	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$	$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$	$\log_2 \epsilon$	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$	$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
Regev [86]												
128	16411	11.81	3.2	81.62	93.84	97.65	83.85	-14	22.50	61.90	65.71	22.50
256	65537	25.53	3.1	126.26	179.76	183.76	168.79	-35	44.48	174.46	178.46	44.48
512	262147	57.06	3.1	337.92	350.80	354.97	338.02	-94	104.47	518.62	522.79	104.47
Lindner & Peikert [61]												
128	2053	2.70	2.9	63.86	82.40	85.86	72.73	-14	22.28	57.06	60.52	22.28
256	4099	3.34	2.8	105.08	151.45	155.04	140.64	-33	42.21	151.16	154.74	42.21
512	4099	2.90	2.6	157.78	278.01	281.59	266.14	-86	96.09	424.45	428.03	96.09

Table 2.5: Cost of distinguishing LWE samples from uniform as reported as "Distinguish" in [61], compared to Corollary 3. BKZ estimates obtained using BKZ 2.0 simulator-derived cost estimate.

Thus, we can reasonably conclude that, under the assumptions made above, employing lattice reduction in a pure distinguishing approach is out-performed by BKW in surprisingly low dimension.

2.5.2 Short Integer Solutions: Combinatorial

Recall that if we consider the set of samples from $L^{(n)}_{\mathbf{s},\chi}$ used during the course of the BKW algorithm as determining a q-ary lattice, and the noisy vector as denoting a point close to a lattice point, we may consider the BKW algorithm as analysed in Corollary 3 as a combinatorial approach for sampling a sparse $\mathbf{u}$ in the dual lattice with entries $\in \{-1, 0, 1\}$. Hence, it is related to a combinatorial approach for finding short dual-(q-ary)lattice vectors as briefly sketched in [73, p. 156] (cf. also [68]). These algorithms, however, operate somewhat differently to BKW - given a relatively small set of LWE samples, these are divided into a small number of subsets. Within each subset, we compute all linear combinations of the members of that subset such that the coefficients of these linear combinations are in $\{-b', \ldots, b'\}$ (note that the parameter b' is unrelated to the parameter b used in this chapter).

The algorithm sketched by [73] uses the generalised birthday paradox to produce

collisions among samples produced by inter-addition, with the parameters of the algorithm being chosen such that we expect to obtain a single short vector in the dual-lattice vector.

There are some significant differences between such algorithms and BKW, stemming from the assumption of the former that we only have access to a very small number of LWE samples. This requires the 'expansion' of the sample set in such a way that when we search for collisions, we are almost certain to find enough. On the other hand, due to this expansion, the initial samples must be separated into disjoint lists. Thus, probably the easiest way to describe the algorithm in [73] in terms of BKW is to imagine a variant of BKW where, if a sample is not in a given table, we add this sample plus *all* $\{-b, \ldots, b\}$-bounded linear combinations of this sample with the pre-existing table entries, to the table. To give a strict analogue to [73], on finding a subsequent collision, we would need to store a list of all noise elements which had been added to a sample and ensure that, if we want to eliminate a sample, that the set of noise elements belonging to the sample and the set of noise elements belonging to the table entry are disjoint.

However, the fundamental difference stems from the assumption that the number of samples is restricted. If this is not the case (as we assume), then it is clear that BKW delivers a much shorter dual lattice vector.

2.5.3 Bounded Distance Decoding: Lattice Reduction

As discussed in the introduction, in [61], the authors propose a method to solve LWE instances which consists of q-ary lattice reduction, then employing a decoding stage to determine the secret. The decoding stage used is essentially a straightforward modification of Babai's well-known nearest plane algorithm for CVP. The authors estimate the running-time of the BKZ algorithm in producing a basis 'reduced-enough' for the decoding stage of the algorithm to succeed, then add the cost of the decoding

stage.

To obtain comparable complexity results, we calculate upper bounds on the bit operation counts for two data-points based on the running times reported in [61] multiplied by the clock speed of the CPU used. As can be seen from Table 2.6, unsurprisingly these indicate substantially lower complexities than for BKW. In addition, the memory requirements of this approach are small compared to the memory requirements of BKW.

n	q	αq	BKW					NTL-BKZ Lindner/Peikert Model				
			t	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$	$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$	$\log_2 \epsilon$	$\log_2 m$	$\log_2 \#\mathbb{Z}_q$	$\log_2 \#\mathbb{Z}_2$	$\log_2 \#L^{(n)}_{\mathbf{s},\chi}$
136	2003	5.19	2.6	67.49	93.77	97.23	84.15	-25	33.46	91.35	94.81	33.46
214	16381	2.94	3.4	76.90	128.36	132.16	117.54	-18	26.95	82.31	86.11	26.95

Table 2.6: Cost of solving Search-LWE reported as "Decode" in [61], compared to the cost solving Decision-LWE with BKW

2.6 Experimental Results

In order to verify the results of this chapter, we implemented stages 1 and 2 of the BKW algorithm. Our implementation considers LWE with short secrets but we ignore the transformation cost to produce samples with a short secret. Also, our implementation supports arbitrary bit-width windows b, not only multiplies of $\lceil \log_2(q) \rceil$. However, due to the fact that our implementation does not use a balanced representation of finite field elements internally – which simplifies dealing with arbitrary bit-width windows – our implementation does not *fully* implement the half-table improvement. That is, for simplicity, our implementation only uses the additive inverse of a vector if this is trivially compatible with our internal data representation. Furthermore, our implementation does not bit-pack finite field elements. Elements always take up 16 bits of storage. Overall, the memory consumption of our implementation in stage 1 is worse by a factor of up to four compared to the estimates

given in this chapter and the computational work in stage is worse by a factor of up to two. Finally, since our implementation is not optimised we do not report CPU times. With these considerations in mind, our estimates are confirmed by our implementation. For example, consider Regev's parameters for $n = 25$ and $t = 2.3$ and $d = 1$. By Lemma 11 picking $m = 2^{12.82}$ will result in a success probability of $\mathrm{p}_{success} \approx 0.99959$ per component and $\mathrm{P}_{success} \approx 0.99$ overall. Lemma 4 estimates a computational cost of $2^{30.54}$ ring operation and $2^{24.19}$ calls to $L^{(n)}_{\mathbf{s},\chi}$ in stage 1. We ran our implementation with $m = \lceil 2^{12.82} \rfloor$ and window bitsize $w = 22 = \frac{n \log_2(q)}{2.3 \log_2(n)}$. It required $2^{29.74}$ ring operations and $2^{23.31}$ calls to $L^{(n)}_{\mathbf{s},\chi}$ to recover one component of $\mathbf{s}$. From this we conclude that Theorem 3 is reasonably tight.

To test the accuracy of Lemma 11 we ran our implementation with the parameters $n = 25$, $q = 631$, $\alpha \cdot q = 5.85$, $w = 24 = \frac{n \log_2(q)}{2.1 \log_2(n)}$ and $m = 2^7$. Lemma 11 predicts a success rate of 53%. In 1000 experiments we ranked zero for the correct key component 665 times, while Lemma 11 predicted 530. Hence, it seems our predictions are slightly pessimistic. The distribution of the ranks of the correct component of $\mathbf{s}$ in 1000 experiments is plotted in Figure 2.2.

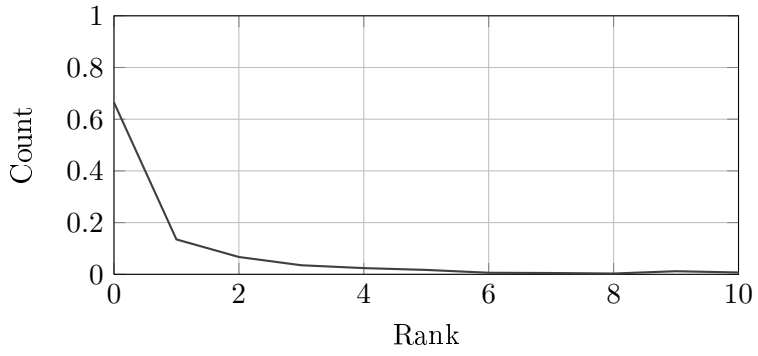

Figure 2.2: Distribution of right key component ranks for 1000 experiments on $n = 25$, $t = 2.3$, $d = 1$, $\mathrm{p}_{success} = 0.99$.

2.7 Magnification and Independence of Noise

During the course of the BKW algorithm, the structure implies that we will sometimes encounter a situation where the same noise element is added more than once to a sample as it passes through the tables. This is generally undesirable from a point of view of minimising noise growth as it leads to random walk scenario where (in the standard BKW algorithm) a noise element is either repeatedly added to or subtracted from itself[2]. Perhaps of greater concern is the possibility of two outputs from $B^{(n)}_{\mathbf{s},\chi,a}$ being 'imperfectly independent'. In this section we treat such issues and argue that, under the parameters employed in this chapter, such effects can be ignored.

We remark that the BKW algorithm as dealt with here (and also in [37]) is distinct from the original proposition in [21]. Namely, in that work, the algorithm proceeds by calling $B^{(n)}_{\mathbf{s},\chi,a-1}$ to obtain a *single* reduced sample, then discards the contents of all tables built up during this construction and starts again from the beginning to obtain a second sample and so on for a third sample etc. The motivation for this discarding of tables is to preserve perfect independence between the noise in the final samples. To make the algorithm even remotely practical, however, the contents of the tables are retained (and added to) between deriving final samples. Such an assumption was also made in [37], though the authors apparently failed to appreciate this loss of perfect independence. In this section we provide an analysis of some of the effects of this loss of perfect independence along with a simple modification to BKW to lessen the incidence of shared noise.

2.7.1 Balls, Bins and Markov Chains

Upon obtaining a sample $(\tilde{\mathbf{a}}_i, \tilde{c}_i)$ from $B^{(n)}_{\mathbf{s},\chi,a}$ it is easy to see that, with high probability, $\tilde{c}_i = \langle \tilde{\mathbf{a}}_i, \tilde{\mathbf{s}} \rangle + e_i$ where $e_i = \sum_{j=0}^{2^a-1} \tilde{e}_j$, where the $\tilde{e}_j$ (with the exception of $\tilde{e}_0$) derive from the table entries which have been 'hit' during the 'reduction' of the

[2] In contrast, this behaviour is somewhat desirable when dealing with LPN.

sample. Now, previous analyses of the BKW algorithm rely on each $\tilde{e}_j$ in such a set being distinct, allowing to argue that the final sum follows a particular noise distribution. If these values are not distinct then, in general, the final noise will follow a distribution of greater variance. In addition, it is clear that the distinguishing phase of the algorithm is most effective when the noise present in each reduced LWE sample is independent from any other. If, say each reduced sample 'contained' 100 noise elements, but, say, 30 were shared between any two reduced samples, the resulting loss of entropy leads to a more costly distinguishing phase.

Note: In the following, we deal with the execution of the BKW algorithm as described previously, with the difference that we never 'switch' table entries, this being, in a sense which should be apparent later, the 'worst-case'. We additionally assume all necessary tables are filled before deriving any output samples from $B^{(n)}_{\mathbf{s},\chi,a}$ - for each such table T^i we denote by ${}^*\mathrm{T}^i$ the set of independent noise elements which occur in T^i (but in no earlier table) as a result of this construction. Clearly, in later tables, these elements also appear (with some probability) in increasing number, increasing the probability that a sample from $B^{(n)}_{\mathbf{s},\chi,a}$ will incorporate more than one occurrence of such noise entries.

Clearly, given $(\tilde{\mathbf{a}}, \tilde{c}) \leftarrow B^{(n)}_{\mathbf{s},\chi,a}$, $\tilde{c}$ 'contains' 2^a noise elements, with

$$n_i = 2^{a-1-i}$$

samples from ${}^*\mathrm{T}^i$. These noise elements are not 'equal', however - some will have been chosen at random from ${}^*\mathrm{T}^i$, while others will have been chosen from a subset of ${}^*\mathrm{T}^i$. For example, if we set $a = 3$ and consider ${}^*\mathrm{T}^0$, then a sample $(\tilde{\mathbf{a}}, \tilde{c}) \leftarrow B^{(n)}_{\mathbf{s},\chi,2}$ will 'contain' four members of ${}^*\mathrm{T}^0$, three of which are drawn at random from ${}^*\mathrm{T}^0$. The fourth, however, is drawn at random from a (generally) smaller set of noise elements, each member of which was drawn at random, with replacement, from ${}^*\mathrm{T}^0$. We denote drawing a noise element from ${}^*\mathrm{T}^0$ as drawing a sample from a '0-th fold'

of ${}^*\mathrm{T}^0$ and the latter case as drawing a sample from a 1st fold of ${}^*\mathrm{T}^0$. We denote such sets by ${}^*\mathrm{T}^0_0$ and ${}^*\mathrm{T}^0_1$ with ${}^*\mathrm{T}^i_j$ denoting the general case.

Then, given n_i elements from ${}^*\mathrm{T}^i$ occurring in a final sample, a simple counting argument shows that the number of such elements from each fold of ${}^*\mathrm{T}^i$ is given by the binomial coefficients. Namely, the number of elements from ${}^*\mathrm{T}^i_k$ is given by the k-th entry of the $(a-1-i)-th$ row of Pascal's triangle, *i.e.* we have $\binom{a-1-i}{k}$ elements from ${}^*\mathrm{T}^i_k$.

We need the following lemma, of which we shall omit the proof

Lemma 12. *Given a set F_0 of d_0 distinctly-labelled balls, we denote by a 'fold' of F_0 the random selection with replacement of d_0 members of F_0. Repeating this sequentially n times leads to an 'n-fold' of F_0, with the original set being denoted by a 0-fold of itself. Then, if we take an n-fold of F_0 and draw two elements at random (with replacement), we denote the probability that these elements collide by $p_{(n,d_0)}$. Then, if we assume that each random variable is independent*

$$p_{(n,d_0)} = \sum_{j=1}^{d_n} \frac{E_n[j] \cdot j^2}{d_0^2}$$

where

$$E_0[j] = \begin{cases} d_0 & \text{if } j = 1 \\ 0 & \text{if } j \neq 1 \end{cases}$$

$$E_i[j] = \sum_{k=1}^{d_i} E_{i-1}[k] \cdot \binom{d_0}{j} \cdot \left(\frac{k}{d_0}\right)^j \cdot \left(1 - \frac{k}{d_0}\right)^{d_0 - j}$$

and

$$d_i = d_0 - \sum_{m=1}^{i-1} E_m[0]$$

More generally we wish to answer the question: if we take an element from ${}^*\mathrm{T}^0_i$ and an element from ${}^*\mathrm{T}^0_j$ $(j \geq i)$, what is the probability that they collide? We denote this general probability by $p_{(i \leftrightarrow j, d_0)}$, with $p_{(n,d_0)}$ corresponding to $p_{(n \leftrightarrow n, d_0)}$.

We denote the multiplicity of an element e in ${}^*\mathrm{T}_k^0$ by $\#_k(e)$.

We approach the problem from a discrete-time, time-homogeneous Markov Chain perspective, treating the multiplicities of elements in folds as being states and the successive folds as being time-steps and then examine the transition probabilities. We denote by random variables X_k the multiplicity of a given element after k folds or time-steps. In notation standard for Markov chains, the probability of transitioning from state i to state j in n steps is

$$p_{ij}^{(n)} := \Pr[X_{k+n} = j \mid X_k = i]$$

Now, we define the state-transition matrix $\mathbf{A}$ such that

$$\mathbf{A}_{i,j} = p_{ij}^{(1)} = \Pr[\#_{k+1}(e) = j \mid \#_k(e) = i]$$

Then the k-step transition probability matrix is simply $\mathbf{A}^k$. For example, in the case of $d_0 = 8$, we have

$$\mathbf{A} = \begin{pmatrix} 1.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.344 & 0.393 & 0.196 & 0.056 & 0.010 & 0.001 & 0.000 & 0.000 & 0.000 \\ 0.100 & 0.267 & 0.311 & 0.208 & 0.086 & 0.023 & 0.003 & 0.000 & 0.000 \\ 0.023 & 0.112 & 0.235 & 0.282 & 0.211 & 0.101 & 0.030 & 0.005 & 0.000 \\ 0.003 & 0.031 & 0.109 & 0.219 & 0.273 & 0.219 & 0.109 & 0.031 & 0.003 \\ 0.000 & 0.005 & 0.030 & 0.101 & 0.211 & 0.282 & 0.235 & 0.112 & 0.023 \\ 0.000 & 0.000 & 0.003 & 0.023 & 0.086 & 0.208 & 0.311 & 0.267 & 0.100 \\ 0.000 & 0.000 & 0.000 & 0.001 & 0.010 & 0.056 & 0.196 & 0.393 & 0.344 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{pmatrix}$$

The stationary distributions are given by the (two distinct) left eigenvectors of $\mathbf{A}$ of multiplicity one. The stochastic state row-vector relation is given by $\mathbf{x}^{k+1} = \mathbf{x}^k\mathbf{A}$. In general, $\mathbf{x}^{k+n} = \mathbf{x}^k\mathbf{A}^n$.

The starting state-vector is always

$$\mathbf{x} = \mathbf{x}^0 = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \end{pmatrix}$$

We now proceed to outline three simple cases which will be required in our analysis

Assumption 2. *Let* $\mathbf{d} = (0, 1/d_0, 2/d_0, \dots, (d_0-1)/d_0, 1)$. *Then we can approximate the probabilities of collisions between noise elements* a, b *by the following*

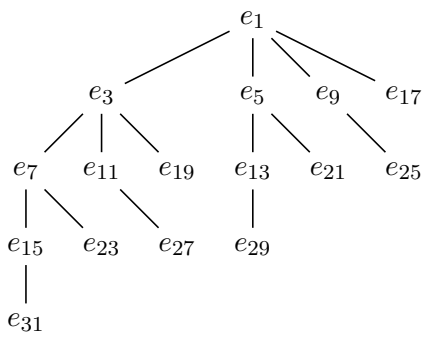

Figure 2.3: Example of Noise Tree

1. *In the case of a and b being drawn from the same fold ${}^*\mathrm{T}^i_j$ of ${}^*\mathrm{T}^i$*

$$\Pr[a = b] \approx \sum_{m=1}^{d_0} \Pr[\#a = m] \cdot \frac{m}{d_0} = \sum_{m=1}^{d_0} \frac{i^2}{d_0} \mathbf{A}^k_{1,i}$$

We denote this by $p_{(j\leftrightarrow,j)}$.

2. *In the case of a and b being drawn from two folds of ${}^*\mathrm{T}^i$, neither of which is a derivative of the other*

$$\Pr[a = b] \approx \frac{1}{d_0}$$

3. *In the case of a being drawn from a fold $F_0 =^* \mathrm{T}^i_j$ of ${}^*\mathrm{T}^i$ and b being drawn from a fold $F_1 =^* \mathrm{T}^i_{j+x}$ of T^i where F_1 is an x-th derivative of F_0*

$$\Pr[a = b] \approx \sum_{m=1}^{d_0} \left(\Pr[\#a = m] \cdot \sum_{n=1}^{d_0} \frac{n}{d_0} \cdot \left(\mathbf{A}^x_{(m,n)} \right) \right) = \sum_{m=1}^{d_0} \left(m \cdot \mathbf{A}^k_{1,i} \cdot \langle \mathbf{d}, \mathbf{A}^x_{(m,*)} \rangle \right)$$

We denote this by $p_{(j\leftrightarrow(j+x))}$.

2.7.2 The General Case

In the remainder of this section we quantify the expected number of collisions between elements of noise trees constructed implicitly during the execution of the BKW algorithm. The arguments employed are largely derived from in-depth examination of such trees and extrapolation, with rigorous proofs being exceedingly cumbersome,

hence we rely on a multitude of counting arguments obtained by examining such trees.

Now, we are interested in taking a k-th level tree and considering the probability of collision between any two tree elements. We denote the tree with 2 elements by K^1, that with 4 elements by K^2, etc., thus the tree in Figure 2.3 corresponds to K^4.

Within each K^i, we need to consider $\binom{2^i}{2}$ possible collisions. To go from K^1 to K^2 we take each element of K^1 and append a copy of K^1, joining the root node to the element. Or, to go from K^1 to K^3, we would append a copy of K^2 to each element of K^1.
Then we can observe that $\#p_{(0\leftrightarrow 0)}$ in tree K^j is given by

$$\#p_{(0\leftrightarrow 0)}(j) = \sum_{r=0}^{j-2} \binom{2r+2}{r}$$

2.7.3 Putting it all together

Given a tree K^j, the possible number of $(0 \leftrightarrow 0)$ collisions on level L is given by

$$\#p_{(0,j,L)} := \sum_{b=1}^{j-L+1} \left(\binom{j-b}{j-L-b+1} \cdot \sum_{c=b+1}^{j-L+1} \binom{j-c}{j-L-c+1} \right)$$

Then, in general, we have

$$\#p_{(k,j,L)} = \sum_{d=1}^{j-L+1} \binom{d+k-2}{d-1} \cdot \#p_{(0,j-k-d+1,L-k)}$$

Hence, *in a specific row*, the total expected number of collisions is given by

$$\sum_{t=0}^{L-1} \#p_{(t,j,L)} \cdot p_{(t\leftrightarrow t)}$$

Thus, given a given row, we can calculate the expected number of collisions between elements. For instance, let $j = 14$ and consider the 8th row of K^{14}.

So, this gives us the total expected number of collisions in a certain row of the tree.

Given the tree K^x, the expected number of collisions with direct-descendant elements is

$$\sum_{b=0}^{x} \left(\sum_{a=1}^{\binom{x}{b}} \left(\sum_{j=1}^{x-1} \binom{x-1}{j} \cdot p_{j \leftrightarrow b} \right) \right)$$

The expected number of collisions with indirect descendants is given as follows. Consider $b = 1$. Then

$$\mathrm{E}[\mathrm{ind}_{x,b=1}]^{\Sigma} = \sum_{j=b-1}^{x-1} \binom{j}{j-b+1} \cdot \left(\sum_{p=j+1}^{x-1} \left(\sum_{t=b}^{p} \binom{p}{t} \cdot p_{0 \leftrightarrow 0} \right) \right)$$

Now consider $x = 4$, $b = 2$. We have

$$\begin{aligned} \mathrm{E}[\mathrm{ind}_{x=4,b=2}]^{\Sigma} &= \sum_{j=b-1}^{x-1} \binom{j}{j-b+1} \cdot \left(\sum_{p=j+1}^{x-1} \left(\sum_{t=b}^{p} \binom{p}{t} \cdot p_{0 \leftrightarrow 0} \right) \right) + \\ &+ \mathrm{E}[\mathrm{ind}_{x=2,b=1}]^{1} + \mathrm{E}[\mathrm{ind}_{x=3,b=1}]^{1} \end{aligned}$$

Similarly, for $x = 4, b = 3$ we have

$$\begin{aligned} \mathrm{E}[\mathrm{ind}_{x=4,b=3}]^{\Sigma} &= \sum_{j=b-1}^{x-1} \binom{j}{j-b+1} \cdot \left(\sum_{p=j+1}^{x-1} \left(\sum_{t=b}^{p} \binom{p}{t} \cdot p_{0 \leftrightarrow 0} \right) \right) + \\ &+ \mathrm{E}[\mathrm{ind}_{x=3,b=2}]^{1} + \mathrm{E}[\mathrm{ind}_{x=2,b=2}]^{2} \end{aligned}$$

We define $B_{(b,l,x)}$ to be

$$B_{(b,l,x)} = \sum_{j=b-1}^{x-1} \binom{j}{j-b+1} \cdot \left(\sum_{p=j+1}^{x-1} \left(\sum_{t=b}^{p} \binom{p}{t} \cdot p_{(l \leftrightarrow l)} \right) \right)$$

Then, given tree K^4, the expected number of indirect collisions is given by

$$\begin{aligned} & B_{(1,0,4)} + \\ + \; & B_{(2,0,4)} + B_{(1,1,2)} + B_{(1,1,3)} + \\ + \; & B_{(3,0,4)} + B_{(2,2,3)} \end{aligned}$$

Similarly, given tree K^5, the expected number of indirect collisions is given by

$$\begin{aligned} & B_{(1,0,5)} + \\ + \quad & B_{(2,0,5)} + B_{(1,1,2)} + B_{(1,1,3)} + B_{(1,1,4)} + \\ + \quad & B_{(3,0,5)} + B_{(2,2,3)} + B_{(2,2,4)} + \\ + \quad & B_{(4,0,5)} + B_{(3,3,4)} \end{aligned}$$

So, in general, given tree K^x, the expected number of indirect collisions is given by

$$\mathrm{E}_b(x) \quad = \quad B_{(1,0,x)} + \sum_{a=2}^{x-1} \left(B_{(a,0,x)} + \sum_{u=a}^{x-1} B_{(a-1,a-1,u)} \right)$$

So, putting this all together, given tree K^x, we have

1. The expected number of collisions between elements lying in the same level of the tree

2. The expected number of collisions between elements and their direct descendants

3. The expected number of collisions between elements and their indirect descendants

Thus, finally, given tree K^x, the expected number of collisions is

$$A_x + B_x + C_x$$

where

$$\begin{aligned} A_x &= \sum_{r=0}^{x} \left(\sum_{t=0}^{r-1} \left(p_{(t \leftrightarrow t)} \cdot \sum_{d=1}^{x-r+1} \binom{d+t-2}{d-1} \cdot \left(\sum_{b=1}^{x-r+1} \left(\binom{x-b}{x-r-b+1} \cdot \sum_{c=b+1}^{x-r+1} \binom{x-c}{x-r-c+1} \right) \right) \right) \right) \\ B_x &= \sum_{r=0}^{x} \left(\sum_{a=1}^{\binom{x}{r}} \left(\sum_{j=1}^{x-1} p_{j \leftrightarrow r} \cdot \binom{x-1}{j} \right) \right) \\ C_x &= B_{(1,0,x)} + \sum_{a=2}^{x-1} \left(B_{(a,0,x)} + \sum_{u=a}^{x-1} B_{(a-1,a-1,u)} \right) \end{aligned}$$

However, despite obtaining such expressions for the expected number of collisions, extracting concrete figures is problematic due to the cost of repeatedly multiplying

the Markov matrix by itself leading to prohibitive running times for anything other than small examples. To attempt a circumvention of this, we employ the above expressions for small examples and then fit a 2-dimensional model to allow us to extrapolate to larger instances.

For any reasonable parameters, the term A_x dominates, thus we only consider A_x to obtain an estimate of the expected number of collisions. To do so, we fitted a two-parameter model to small estimates, obtaining

$$\log_2(A_x^{d_0}) \approx 3.0103 \cdot x - c(d_0)$$

where

$$c(d_0) \approx 97.6417 \cdot d_0^{0.0138} - 93.2746$$

Thus

$$A_x^{d_0} \approx 2^{3.0103 \cdot x - 97.6417 \cdot d_0^{0.0138} + 93.2746}$$

and we can approximate the number of collisions in a given sample from $B_{\mathbf{s},\chi,a}^{(n)}$ by

$$\sum_{i=0}^{a} A_i^{d_0}$$

2.7.4 Implications

In the LPN case, the above considerations could only be beneficial when we consider self-intersections, since the noise in a single sample cannot grow as a result. However, the impact on independence can be substantial and detrimental. In the LWE case, both cases can impact the efficacy of the algorithm.

2.7.5 Mitigation

A simple modification to the BKW algorithm leads to substantial mitigation of the incidence of shared noise elements. Simply, when we have a table T^i and a sample $(\mathbf{a}, c)$ which is to be reduced by $(\mathbf{a}', c')$ in table T^i, we compute $(\mathbf{a} - \mathbf{a}', c - c')$ and, instead of simply returning this vector, we additionally replace $(\mathbf{a}', c')$ in T^i by $(\mathbf{a}, c)$,

discarding $(\mathbf{a}', c')$ from the table. This is somewhat related to a modification in the next chapter, though for unrelated reasons. However, we do not analyse the effects of this modification here.

2.8 Closing Remarks

In this chapter we provided a concrete analysis of the cost of running the BKW algorithm on LWE instances both for the search and the decision variants of the LWE problem. We also applied this analysis to various sets of parameters found in the literature. From this we conclude that the BKW algorithm outperforms lattice reduction algorithms in an SIS setting for the parameter sets proposed in [86, 61] starting around dimension $n \approx 250$ at the cost of requiring many more samples and storage. On the other hand, lattice reduction in a BDD setting currently outperforms the BKW algorithm as analysed in this chapter.

We also provide a partially-heuristic analysis of the expected number of (undesirable) collisions which result from more practical versions of BKW, showing such effects to be insignificant in the LWE case and illustrating how such effects can be mitigated.

Furthermore, in this chapter we ignored the so-called LWE "normal form" where the secret follows the noise distribution χ (cf. [22]) and other small secret variants of LWE. For the BKW algorithm as presented in this chapter, only hypothesis testing is affected by the size of the secret and hence we do not expect the algorithm to benefit from considering small secrets in this form. A dedicated variant of the BKW algorithm tackling small secrets is thus a clear direction for investigation, one which we visit in the next chapter.

Chapter 3

Lazy Modulus Switching for the BKW Algorithm

The contents of this chapter are based on the paper "Lazy Modulus Switching for the BKW Algorithm" which was presented at PKC 2014, Buenos Aires, Argentina and was a work conducted in collaboration with M. R. Albrecht, J. C. Faugére and L. Perret.

3.1 Introduction

The motivation for this chapter comes from the observation that recent constructions based on LWE do not sample the secret uniformly at random but rather from some distribution which produces small entries (e.g. [10, 4, 44, 41, 83]). From a theoretical point of view, this is motivated by the observation that every LWE instance can be transformed into an instance where the secret follows the same distribution as the noise [10].[1] However, many constructions use secrets which are considerably smaller. For example, binary-LWE samples the secret from $\{0,1\}^*$ [22] or $\{-1,0,1\}^*$ [41]. The presence of such small secrets provokes the question of what implications such choices have on the security of LWE. Is solving LWE with, say, binary secrets easier

[1] also in [57] for the LPN case.

than standard LWE ? From a theoretical point of view, [22] proves that their binary-LWE is as secure as LWE. In this chapter, we try to address the question from an algorithmic point of view; i.e. what is the actual impact of small secrets on concrete parameters.

3.1.1 Organisation of the Chapter and Main Results

While none of the algorithms discussed previously (BKW, dual-lattice distinguishing, primal lattice reduction + decoding/enumeration) take advantage of the presence of small secrets, we may combine them with the *modulus switching technique*. The modulus switching technique was initially introduced to improve the performance of homomorphic encryption schemes [23] and was recently used to reduce the hardness of LWE with polynomially sized moduli to GAPSVP [22]. Modulus switching is essentially the same as computing with a lower precision similar to performing floating point computations with a low fixed precision. Namely, let $\big(\mathbf{a}, c = \langle \mathbf{a}, \mathbf{s}\rangle + e\big) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ be LWE sample where $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret vector, and $e \in \mathbb{Z}_q$ is an error. Let also some $p < q$ and consider $\big(\, \lfloor p/q \cdot \mathbf{a} \rceil \,, \lfloor p/q \cdot c \rceil \,\big)$ with

$$
\begin{aligned}
\left\lfloor \frac{p}{q} \cdot c \right\rceil &= \left\lfloor \frac{p}{q}\big(\langle \mathbf{a}, \mathbf{s}\rangle + q \cdot u + e\big) \right\rceil, \text{ for some } u \in \mathbb{Z} \\
\left\lfloor \frac{p}{q} \cdot c \right\rceil &= \left\lfloor \langle \frac{p}{q} \cdot \mathbf{a}, \mathbf{s}\rangle_p + \frac{p}{q} \cdot e \right\rceil = \left\lfloor \langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s}\rangle_p + \langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s}\rangle_p + \frac{p}{q} \cdot e \right\rceil \\
&= \langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s}\rangle_p + \langle \frac{p}{q} \cdot \mathbf{a} - \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s}\rangle_p + \frac{p}{q} \cdot e + e', \text{ where } e' \in [-0.5, 0.5] \\
&= \langle \left\lfloor \frac{p}{q} \cdot \mathbf{a} \right\rceil, \mathbf{s}\rangle_p + e'' + \frac{p}{q} \cdot e + e'. \qquad (3.1)
\end{aligned}
$$

where $\langle \mathbf{x}, \mathbf{y}\rangle_p$ denotes the modulo p inner product of $\mathbf{x}$ and $\mathbf{y}$.

Since $p/q \cdot \mathbf{a} - \lfloor p/q \cdot \mathbf{a} \rceil$ takes values $\in [-0.5, 0.5]$ we have that e'' is small if $\mathbf{s}$ is small. We may hence compute with the smaller 'precision' p at the cost of a slight increase of the noise rate by a '*rounding error*' e''. Modulus switching allows then to map a LWE instance $\bmod q$ to a scaled instance of LWE $\bmod p$. Thus, modulus switching can be used in the solving of small secret instances of LWE, a folklore ap-

proach which has not been explicitly studied in the literature. Namely, if we pick p such that e'' is not much larger than $p/q \cdot e$ then, for example, the running time of the BKW algorithm improves from $(a^2 n) \cdot \frac{q^b}{2}$ to $(a^2 n) \cdot \frac{p^b}{2}$. Since typically $b \approx n/\log n$ this may translate to substantial improvements. Indeed, we can pick p such that $|\langle p/q \cdot \mathbf{a} - \lfloor p/q \cdot \mathbf{a} \rceil, \mathbf{s} \rangle| \approx p/q \cdot |e|$. This implies $\sigma_s \cdot \sqrt{\frac{n}{12}} \approx p/q \cdot \sigma$, or

$$p \approx \min\left\{q, \frac{\sigma_s}{\sigma} \cdot \sqrt{\frac{n}{12}} \cdot q\right\}$$

where σ_s is the standard deviation of elements in the secret $\mathbf{s}$. In this chapter, we refine this approach and present a variant of the BKW algorithm which fuses modulus switching and BKW-style reduction. In particular, this chapter contains two main contributions. Firstly, in Section 3.2 we present a modulus switching strategy for the BKW algorithm in which switching is delayed until necessary. In a nutshell, recall that the BKW algorithm performs additions of elements which collide in certain components. Our variant will search for such collisions in 'low precision' $\mathbb{Z}_p$ but will perform arithmetic in 'high precision' $\mathbb{Z}_q$. From now on, we call *rounding error* the inner product of the sub-vector of 'low bits' of $\mathbf{a}$ with the secret $\mathbf{s}$. Our strategy permits to decrease rounding errors and allows to reduce p by a factor of $\sqrt{a}$. Secondly, this perspective enables us to choose reductors in the BKW algorithm which minimise the rounding errors further (Section 3.3). Namely, we favour components a with small distance $|\lfloor p/q \cdot a \rceil - p/q \cdot a|$ in already reduced components, called 'child components' in this work. Our strategy ensures that the probability of finding such elements is highest for those components which are considered first by the BKW algorithm, i.e. those components which contribute most to the noise. We note that the first contribution relies on standard independence assumptions only, while the second contribution relies on stronger assumptions, which however seem to hold in practice. We then discuss the complexity of our variants in Section 3.4. For typical choices of parameters – i.e. $q \approx n^c$ for some small constant $c \geq 1$, $a = \log_2 n$ and $b = n/\log_2 n$ – the complexity of BKW as analysed in Chapter 2 is $\mathcal{O}\left(2^{cn} \cdot n \log_2^2 n\right)$. For small secrets, a naive modulus switching technique allows the reduction of this

complexity to $\mathcal{O}\left(2^{n\left(c+\frac{\log_2 d}{\log_2 n}\right)} \cdot n \log_2^2 n\right)$ where $0 < d \leq 1$ is a small constant. If the secret distribution does not depend on n and if an unbounded number of LWE samples is available our improved version of BKW allows to get a complexity of:

$$\mathcal{O}\left(2^{n\left(c+\frac{\log_2 d - \frac{1}{2}\log_2\log_2 n}{\log_2 n}\right)} \cdot n \log_2^2 n\right).$$

We then study the behaviour of this algorithm by applying it to various instances of LWE with binary secrets. In Section 3.5 we report on experiments conducted with a proof-of-concept implementation of our algorithm. In Section 3.6, we compare the results with plain BKW and BKZ under modulus switching and a simple meet-in-the-middle approach or generalised birthday attack. We show that our lazy-modulus-switching variant of the BKW algorithm provides better results than applying plain BKW after modulus reduction. We also demonstrate that under the parameters considered here this algorithm also – as n increases – outperforms the most optimistic estimates for BKZ when we apply BKZ to the same task as that to which we apply BKW: finding short vectors in the (scaled-)dual lattice - we obtain this perspective by viewing the rounding error as an increase in the noise rate while still finding short vectors in the (scaled)-dual p-ary lattice determined by our modulus-reduced LWE samples. Indeed, our results indicate that our algorithm outperforms BKZ 2.0 when both are used to find a short vector in the (scaled-)dual lattice in dimension as low as ≈ 256 when considering LWE parameters from [86] with binary secret. However, we stress again that we always assume an unbounded number of samples to be available for solving.

3.2 Modifying BKW: Lazy Modulus Switching

Similarly to the approach taken in Chapter 2, we consider BKW – applied to Decision-LWE – as consisting of two stages: *sample reduction* and *hypothesis testing* (we neglect back-substitution in this chapter). In this chapter, we only modify the first stage.

3.2.1 The Basic Idea

We briefly recall the principle of standard BKW, as examined in Chapter 2. Assume we are given samples of the form $(\mathbf{a}, c)$ following either $L^{(n)}_{\mathbf{s},\chi}$ or $\mathcal{U}\mathbb{Z}_q^n \times \mathcal{U}\mathbb{Z}_q$. Our goal is to distinguish between the two cases. Standard BKW proceeds by producing samples $(\mathbf{a}^*, c^*)$ with $\mathbf{a}^*$ being all zero such that statistical tests can be applied to c^* to decide whether they follow $\mathcal{U}(\mathbb{Z}_q)$ or some distribution related to $L^{(n)}_{\mathbf{s},\chi}$. This is achieved by grouping the n components of all vectors into a groups of b components each (assuming a and b divide n for simplicity). If two vectors collide on all b entries in one group, the first is subtracted from the second, producing a vector with at least b all zero entries. These vectors are then again combined to produce more all zero entries and so forth until all a groups are eliminated to zero. However, as we add up vectors the noise increases. Overall, after ℓ addition levels the noise has standard deviation $\sqrt{2^\ell}\alpha q$. Our algorithm, too, will be parametrized by a positive integer $b \leq n$ (the window width), and $a := \lceil n/b \rceil$ (the addition depth).

Recall that the complexity of the BKW algorithm is essentially q^b. However, b only depends on the ratio $\alpha q/\sqrt{2\pi}q = \alpha/\sqrt{2\pi}$ and thus not on q. Hence, it is clear that applying modulus reduction before running the BKW algorithm may greatly improve its running time: b is preserved whilst q is reduced to p. However, instead of applying modulus reduction in 'one shot' prior to executing BKW, we propose switching to a lower precision only when needed. For this, we actually never switch the modulus but simply consider elements in $\mathbb{Z}_q$ 'through the perspective' of $\mathbb{Z}_p$. We then essentially only consider the top-most $\log_2 p$ bits of $\mathbb{Z}_q$.

Under this perspective, given samples of the form $(\mathbf{a}, c)$ we aim to produce $\left(\tilde{\mathbf{a}}, \tilde{c} = \langle \tilde{\mathbf{a}}, \mathbf{s} \rangle + \tilde{e}\right)$, where $\tilde{\mathbf{a}}$ is short enough, i.e.

$$|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle| \approx \sqrt{2^a}\alpha q. \tag{3.2}$$

Although other choices are possible, this choice means balancing the noise $\tilde{e}$ after a levels of addition and the contribution of $|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle|$ such that neither dominates. From now on, we shall call *rounding error* the term $\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle$. So, condition (3.2) is such that after a levels of additions performed by the BKW algorithm the escalated initial noise and the noise coming from rounding errors have the same size.

3.2.2 Sample Reduction for Short Secrets

Let $(\mathbf{a}_0, c_0), \ldots, (\mathbf{a}_{m-1}, c_{m-1})$ be samples which follow $L^{(n)}_{\mathbf{s},\chi}$ or $\mathcal{U}(\mathbb{Z}_q^n) \times \mathcal{U}(\mathbb{Z}_q)$. We now explain how to produce samples $(\tilde{\mathbf{a}}_i, \tilde{c}_i)_{i \geq 0}$ that satisfy condition (3.2). For simplicity, we assume from now on that $p = 2^\kappa$. [2]

The main idea of the algorithm is to search for collisions among the first b components of samples $(\mathbf{a}_i, c_i)$ by only considering their top $\log_2 p$ bits. If such a collision is found, we proceed as in the normal BKW algorithm, i.e. we subtract the colliding samples to clear the first b components. In our case, we clear the top-most $\log_2 p$ bits of the first b components. Hence, instead of managing elimination tables for every bit of all components, we only manage elimination tables for the most significant κ bits. Put differently, all arithmetic is performed in $\mathbb{Z}_q$ but collisions are searched for in $\mathbb{Z}_p$ after rescaling or modulus switching.

As in chapter 2, we realise the first stage of the BKW algorithm as a (recursively constructed) series of oracles $B^{(n)}_{\mathbf{s},\chi,\ell}$. In our case, we have $0 \leq \ell < a$, where $B^{(n)}_{\mathbf{s},\chi,a-1}$ produces the final output and $B^{(n)}_{\mathbf{s},\chi,-1}$ calls the LWE oracle. We will make use of a set of tables T^ℓ (maintained across oracle calls) to store (randomly-chosen) vectors that will be used to reduce samples arising from our oracles. However, compared to Chapter 2 our oracles $B^{(n)}_{\mathbf{s},\chi,\ell}$ take an additional parameter p which specifies the precision which we consider. Hence, if $p = q$ then we recover the algorithm from Chapter

[2] While we do not have to restrict our attention to p of the form 2^κ, we choose it for ease of exposition and implementation.

2 where we perform no modulus reduction at all. In particular, $B^{(n)}_{\mathbf{s},\chi,\ell}$ proceeds as follows:

1. For $\ell = -1$, we can obtain samples from $B^{(n)}_{\mathbf{s},\chi,-1}$ by simply calling the LWE oracle $L^{(n)}_{\mathbf{s},\chi}$ and returning the output.

2. For $\ell = 0$, we repeatedly query the oracle $B^{(n)}_{\mathbf{s},\chi,0}$ to obtain (at most) $(p^b - 1)/2$ samples $(\mathbf{a}, c)$ with distinct non-zero vectors $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$. We use these samples to populate the table T^0, indexed by $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$. We store $(\mathbf{a}, c)$ in the table. During this course of this population, whenever we obtain a sample $(\mathbf{a}', c')$ from $B^{(n)}_{\mathbf{s},\chi,-1}$, if $\left\lfloor p/q \cdot \mathbf{a}'_{(0,b)} \right\rceil$ (resp. the negation) match $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$ such that the pair $(\mathbf{a}, c)$ is already in T^1, we return $(\mathbf{a}' \pm \mathbf{a}, c' \pm c)$, as a sample from $B^{(n)}_{\mathbf{s},\chi,0}$. Note that, if $\lfloor p/q \cdot \mathbf{a}_{(0,b)} \rceil$ is zero, we return $(\mathbf{a}', c')$ as a sample from $B^{(n)}_{\mathbf{s},\chi,0}$. Further calls to the oracle $B^{(n)}_{\mathbf{s},\chi,0}$ proceed in a similar manner, but using (and potentially adding entries to) the same table T^0.

3. For $0 < \ell < a$, we proceed as above: we make use of the table T^ℓ (constructed by calling $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ up to $(p^b - 1)/2$ times) to reduce any output sample from $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ with $\lfloor p/q \cdot \mathbf{a}_{(b\cdot\ell, b\cdot\ell+b)} \rceil$ by an element with a matching such vector, to generate a sample returned by $B^{(n)}_{\mathbf{s},\chi,\ell}$.

Pseudo-code for the modified oracle $B^{(n)}_{\mathbf{s},\chi,\ell}$, for $0 \leq \ell < a$, is given in Algorithm 3.

3.2.3 Selecting p

Yet, we still have to establish the size of p to satisfy Condition 3.2. In contrast to the technique described in (3.3), we emphasise that in our approach we do not actually multiply by p/q. Let σ_r be the standard deviation of uniformly random elements in $\mathbb{Z}_{\lfloor q/p \rceil}$. Performing one-shot modulus switching in this setting would mean to split $\mathbf{a}$ into two vectors, $\mathbf{a}'$ with the 'high order' bits and $\mathbf{a}''$ with 'low order' bits. In essence, when performing one-shot modulus reduction leads us to immediately discard or ignore all low order bits at the outset, then to proceed with BKW on

Algorithm 3: $B^{(n)}_{\mathbf{s},\chi,\ell}$ for $0 \leq \ell < a$.

Input: b – an integer $0 < b \leq n$

Input: ℓ – an integer $0 \leq \ell < a$

Input: p – an integer $0 < p \leq q$

begin

  $T^{\ell} \leftarrow$ table with p^b rows maintained across all runs of $B^{(n)}_{\mathbf{s},\chi,\ell}$;

  repeat

    query $B^{(n)}_{\mathbf{s},\chi,\ell-1}$ to obtain $(\mathbf{a}, c)$;

    $\mathbf{z} \leftarrow \left\lfloor \frac{p \cdot \mathbf{a}_{(b\cdot\ell, b\cdot(\ell+1))}}{q} \right\rceil$;

    if $\mathbf{z}$ *is all zero* **then**

      return $(\mathbf{a}, c)$;

    else if $T_{\mathbf{z}} \neq \varnothing$ **then**

      break;

    $T_{\mathbf{z}} \leftarrow (\mathbf{a}, c)$;

    $\overline{\mathbf{z}} \leftarrow \left\lfloor \frac{-p \cdot \mathbf{a}_{(b\cdot\ell, b\cdot(\ell+1))}}{q} \right\rceil$;

    $T_{\overline{\mathbf{z}}} \leftarrow (-\mathbf{a}, -c)$;

  until *the world ends*;

  $(\mathbf{a}', c') \leftarrow T_{\mathbf{z}}$;

  return $(\mathbf{a} - \mathbf{a}', c - c')$;

the remaining high-order bits. By discarding the low order bits at the outset, we immediately increase the noise present in the resulting modulus-switched samples.

The standard deviation of each component of $\mathbf{a}''$ is σ_r, thus the standard deviation of the product of such a component with a component of $\mathbf{s}$ is $\sqrt{\sigma_r^2\sigma_s^2}$. Considering all such products, it is apparent that the initial one-shot modulus switching adds noise of standard deviation $\sqrt{n \cdot \sigma_r^2\sigma_s^2}$ to the modulus-switched samples. Hence, after applying BKW to these pre-processed samples, the standard deviation of the noise contributed by modulus-switching in the final output would be

$$\sqrt{n \cdot 2^a \cdot \sigma_r^2\sigma_s^2} = \sqrt{a\,b \cdot 2^a \cdot \sigma_r^2\sigma_s^2}. \tag{3.3}$$

In more detail, we can observe that by performing one-shot modulus reduction, we immediately gain the additional noise term e'', as in Equation 3.1, which has variance $n \cdot \sigma_s^2$·. However, as the following lemma establishes, we may consider smaller p because the final noise contributed by modulus switching under Algorithm 3 is smaller than in (3.3). This is because if $(\tilde{\mathbf{a}}_i, \tilde{c}_i)$ are final output samples then the entries $\tilde{\mathbf{a}}_{i,(b\cdot a-1)}$ will be significantly smaller than $\tilde{\mathbf{a}}_{i,(0)}$.

Yet, to formalise this, we need to make a (standard) simplifying assumption, namely that the outputs of the BKW algorithm (at every stage) are independent. That is, we make the assumption that, during the course of the algorithm described, all components of each sample from $B^{(n)}_{\mathbf{s},\chi,\ell}$ are independent from every other sample. In 3.6.1 we examine the application of arguments from 2.7 to typical table sizes arising from this modified algorithm to illustrate that we can, to all intents and purposes, treat all outputs as being independent.

Assumption 3. *We assume that all outputs of $B^{(n)}_{\mathbf{s},\chi,\ell}$ are independent.*

Assumption 3 allows to establish the following lemma:

Lemma 13. *Let $n \geq 1$ be the dimension of the* LWE *secret vector, q be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$. Let also σ_r be the standard deviation of uniformly random*

elements in $\mathbb{Z}_{\lfloor q/p \rceil}$. *Under Assumption 3, the components of* $\tilde{\mathbf{a}} = \mathbf{a} - \mathbf{a}'$ *returned by* $B^{(n)}_{\mathbf{s},\chi,\ell}$ *satisfy:*

$$\mathrm{Var}(\tilde{\mathbf{a}}_{(i)}) = 2^{\ell - \lfloor i/b \rfloor}\sigma_r^2, \text{ for } 0 \leq \lfloor i/b \rfloor \leq \ell$$

and $\mathrm{Var}\big(\mathcal{U}\mathbb{Z}_q\big)$ *for* $\lfloor i/b \rfloor > \ell$.

Proof. We assume $b = 1$ without loss of generality and proceed by induction on ℓ.
Initialization. If $\ell = 0$, then $i = 0$. The output of $B^{(n)}_{\mathbf{s},\chi,0}$ is the sum of two random vectors in $\mathbb{Z}_q^n$ which collide in component zero when considered in $\mathbb{Z}_p$. The variance of the result is hence that of a random element in $\mathbb{Z}_{\lfloor q/p \rceil}$, i.e. σ_r^2, in component zero, all other components follow the uniform distribution in $\mathbb{Z}_q$.

If $\ell = 1$, then $i = 0$ and 1. Also, we have two elimination tables T^0 and T^1. Outputs of $B^{(n)}_{\mathbf{s},\chi,1}$ are the sum of two outputs of $B^{(n)}_{\mathbf{s},\chi,0}$. Under Assumption 3 these are independent and the sum of their variances is the variance of their sum. The variance of $\tilde{\mathbf{a}}_{(0)}$ is hence $2\sigma_r^2$ and $\tilde{\mathbf{a}}_{(1)}$ has variance σ_r^2 similarly to case $\ell = 0$. All other components are uniformly random in $\mathbb{Z}_q$.

Induction. More generally, for $\ell > 0$ the output of $B^{(n)}_{\mathbf{s},\chi,\ell}$ is the sum of two outputs of $B^{(n)}_{\mathbf{s},\chi,\ell-1}$. Hence, its components satisfy $\mathrm{Var}(\tilde{\mathbf{a}}_{(i)}) = 2 \cdot 2^{\ell-1-i}\sigma_r^2$ for $0 < i < \ell$ and σ_r^2 for $\mathbf{a}_{(\ell)}$. □

Using Lemma 13 we may adapt our choice of p, because the noise contributed by modulus switching for a given p is smaller:

Corollary 4. *Let* $n \geq 1$ *be the dimension of the* LWE *secret vector,* q *be a modulus,* $b \in \mathbb{Z}$ *with* $1 \leq b \leq n$. *Let* σ_r *be the standard deviation of uniformly random elements in* $\mathbb{Z}_{\lfloor q/p \rceil}$ *and* σ_s *be the standard deviation of the distribution from which the secret* $\mathbf{s}$ *is sampled. Let* $(\tilde{a}, \tilde{c})$ *be an output of* $B^{(n)}_{\mathbf{s},\chi,a-1}$. *Under Assumption 3, the noise added by lazy modulus switching in the final output of* $B^{(n)}_{\mathbf{s},\chi,a-1}$, *that is* $|\langle \tilde{\mathbf{a}}, \mathbf{s} \rangle|$, *has*

standard deviation

$$\sqrt{b \cdot \left(\sum_{i=0}^{a-1} 2^{a-i-1}\right) \cdot \sigma_r^2 \sigma_s^2} = \sqrt{b \cdot (2^a - 1) \cdot \sigma_r^2 \sigma_s^2}.$$

Proof. From Lemma 13, we are adding n (assumed to be) independent random variables, each of which takes the form $\tilde{\mathbf{a}}_i \cdot \mathbf{s}_i$ where $\tilde{\mathbf{a}}_i$ is distributed according to the interval of b elements in which i lies. The corollary then follows by adding the variances of b such random variables from each interval. □

The main observation of this part is obtained by comparing Corollary 4 with the standard deviation (3.3). We see that the standard deviation obtained using our lazy modulus switching is divided by a factor $\sqrt{a}$ w.r.t. to a naive use of modulus-switching, i.e. as in (3.3). As a consequence, we may reduce p by a factor $\sqrt{a}$.

3.3 Improved Algorithm: Stunting Growth by Unnatural Selection

Based on the strategy in the previous section, we now introduce a pre-processing step which allows to further reduce the magnitude of the noise present in the outputs of $B^{(n)}_{\mathbf{s},\chi,a-1}$ by reducing rounding errors further. For this, it will be useful to establish notation to refer to various components of $\mathbf{a}_i$ in relation to $B^{(n)}_{\mathbf{s},\chi,\ell}$.

Children: are all those components with index $j < b \cdot \ell$, i.e, those components that were reduced by some $B^{(n)}_{\mathbf{s},\chi,k}$ with $k < \ell$: *they grow up so quickly.*

Parents: are those components of $\mathbf{a}_i$ with index $b \cdot \ell \leq j < b \cdot \ell + b$, i.e. those components among which collisions are searched for in $B^{(n)}_{\mathbf{s},\chi,\ell}$: *collisions among parents produce children.*

Strangers: with respect to $B^{(n)}_{\mathbf{s},\chi,\ell}$ are all other components $j \geq b \cdot \ell + b$: *they are indifferent towards each other.*

So, for example, if $n = 10$ and $b = 2$ and we are considering $\ell = 2$ then $\mathbf{a}_{(0-3)}$ are child components, $\mathbf{a}_{(4-5)}$ are parents and $\mathbf{a}_{(6-9)}$ are strangers (cf. Figure 3.1).

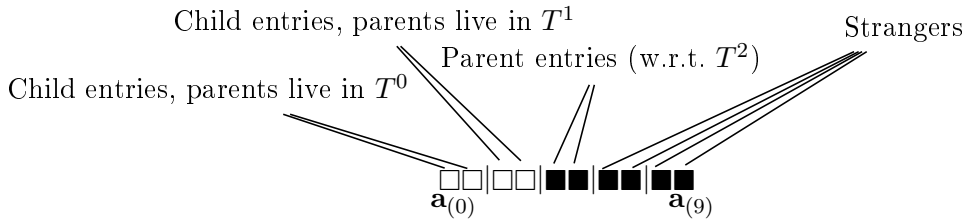

Figure 3.1: Children, parents and strangers.

3.3.1 The Basic Idea

For the general idea and intuition, assume $b = 1$ and that $\tilde{\mathbf{a}}_i$ are outputs of $B^{(n)}_{\mathbf{s},\chi,0}$ and we hence have $\mathrm{Var}(\tilde{\mathbf{a}}_{i,(0)}) = \sigma_r^2$. Now, some of these $\tilde{\mathbf{a}}_i$ will be stored in Table T^1 by $B^{(n)}_{\mathbf{s},\chi,1}$ based on the value in the parent component $\tilde{\mathbf{a}}_{i,(1)}$. All future outputs of $B^{(n)}_{\mathbf{s},\chi,1}$ which collide with $\tilde{\mathbf{a}}_i$ in the parent component at index 1 will have $\tilde{\mathbf{a}}_i$ added/-subtracted to it, we are hence adding a value with $\mathrm{Var}(\tilde{\mathbf{a}}_{i,(0)}) = \sigma_r^2$ in index 0.

Now, however, if the $\tilde{\mathbf{a}}_{i,(0)}$ happened to be unusually short, all $B^{(n)}_{\mathbf{s},\chi,\ell}$ for $\ell > 0$ would output vectors with a shorter $\tilde{\mathbf{a}}_{i,(0)}$ added/subtracted in, i.e. would also have unusually small child components (although to a lesser degree). That is, improving the outputs of $B^{(n)}_{\mathbf{s},\chi,1}$ – i.e. decreasing the magnitude of the $\tilde{\mathbf{a}}_{i,(0)}$ stored in T^1 – has a knock-on effect on all later outputs. More generally, improving the outputs of $B^{(n)}_{\mathbf{s},\chi,\ell}$ will improve the outputs of $B^{(n)}_{\mathbf{s},\chi,k}$ for $k > \ell$.

On the other hand, improving the outputs of $B^{(n)}_{\mathbf{s},\chi,\ell}$ where ℓ is small, is easier than for larger values of ℓ. In the algorithm as described so far, when we obtain a collision between a member of T^ℓ and an output $(\mathbf{a}_i, c_i)$ of $B^{(n)}_{\mathbf{s},\chi,\ell-1}$, we reduce $(\mathbf{a}_i, c_i)$ using the colliding member of T^ℓ, retaining this member in the table. Alternatively we can reduce $(\mathbf{a}_i, c_i)$ using the *in-situ* table entry, replace the table entry with (the now reduced) $(\mathbf{a}_i, c_i)$ and return the former table entry as the output of $B^{(n)}_{\mathbf{s},\chi,\ell}$. If we selectively employ this alternative strategy using the relative magnitudes of the

child components of $(\mathbf{a}_i, c_i)$ and the table entry as a criterion, we can improve the 'quality' of our tables as part of a pre-processing phase.

That is, in $B^{(n)}_{\mathbf{s},\chi,\ell}$ for each collision in a parent component we may inspect the child components for their size and keep that in T^ℓ where the child components are smallest. Phrased in the language of 'children' and 'parents': we do not let 'nature', i.e. randomness, run its course but intervene and select children based on their size. As the number of child components is $b \cdot \ell$ it becomes more difficult as ℓ increases to find vectors where all child components are short.

3.3.2 Algorithms

This leads to a modified algorithm $B_{small,\mathbf{s},\chi}(b, \ell, p)$ given in Algorithm 4. Using Algorithms 3 and 4 we may then present our revised version of the BKW algorithm in Algorithm 5 where we first use Algorithm 4 to produce 'good' tables and then use Algorithm 3 to sample $(\tilde{\mathbf{a}}_i, \tilde{c}_i)$ as before. We note that, in Algorithm 4, we employ the ℓ_1 norm for selecting desirable child entries. However, modifying this choice to use any other norm would be straightforward.

Algorithm 4: $B_{small,\mathbf{s},\chi}(b, \ell, p)$ for $0 \leq \ell < a$.

Input: b – an integer $0 < b \leq n$
Input: ℓ – an integer $0 \leq \ell < a$
Input: p – an integer $0 < p \leq q$

```
1 begin
2     T^ℓ ← table with p^b rows maintained across all runs of B_{small,s,χ}(b, ℓ, p);
3     Find (a', c') ← T^ℓ_z that collides with a fresh sample (a, c) from B^(n)_{s,χ,ℓ−1} as in
      Algorithm 3;
4     if Σ_{i=0}^{b·ℓ−1} |a'_(i)| > Σ_{i=0}^{b·ℓ−1} |a_(i)| then
5         T^ℓ_z ← (a, c);
6     return (a − a', c − c');
```

Algorithm 5: BKW with lazy modulus switching.

Input: b – an integer $0 < b \leq n$

Input: a – an integer such that $ab = n$

Input: p – an integer $0 < p < q$

Input: o – an integer $0 \leq o$

Input: m – an integer $0 \leq m$

begin

  $o_t \leftarrow o/(a+1)$;

  `// populate elimination tables with random entries`

  for $0 \leq i < o_t$ **do**

    $(\tilde{\mathbf{a}}, c) \leftarrow B^{(n)}_{\mathbf{s},\chi,a-1}$; `//` $(\tilde{\mathbf{a}}, c)$ `is discarded`

  `// sample small entries`

  for $0 \leq i < a$ **do**

    for $0 \leq j < o_t$ **do**

      $(\tilde{\mathbf{a}}, c) \leftarrow B_{small,\mathbf{s},\chi}(b, i, p)$; `//` $(\tilde{\mathbf{a}}, c)$ `is discarded`

  for $0 \leq i < m$ **do**

    $(\tilde{\mathbf{a}}_i, c_i) \leftarrow B^{(n)}_{\mathbf{s},\chi,a-1}$;

  Run distinguisher on c_i and return output;

τ	1	2	3	4	5
c_τ	0.405799353869	0.692447899282	0.789885269135	0.844195936036	0.854967912468
τ	6	7	8	9	10
c_τ	0.895446987232	0.91570933651	0.956763578012	0.943424544282	0.998715322134

Table 3.1: c_τ for small values of τ

3.3.3 Selecting p

It remains to be established what the effect of such a strategy is, i.e. how fast children grow up or how fast rounding errors accumulate. In particular, given n vectors $\mathbf{x}_i$ sampled from some distribution $\mathcal{D}$ where each component has standard deviation σ, i.e. $\mathrm{Var}(\mathbf{x}_{i,(j)}) = \sigma^2$ we are interested in the standard deviation σ_n of each component for $\mathbf{x}^* = \min_{abs}(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ where $\min_{abs}$ picks that vector where $\sum_{j=0}^{b\cdot\ell-1} |\mathbf{x}_{(j)}|$ is minimal. At this point we know no closed algebraic expression for σ_n. However, we found – as detailed in Section 3.8 – that σ_n can be estimated as follows:

Assumption 4. *Let the vectors $\mathbf{x}_0, \ldots, \mathbf{x}_{n-1} \in \mathbb{Z}_q^\tau$ be sampled from some distribution $\mathcal{D}$ such that $\sigma^2 = \mathrm{Var}(\mathbf{x}_{i,(j)})$ where $\mathcal{D}$ is any distribution on (sub-)vectors observable in Algorithm 5. Let $\mathbf{x}^* = \min_{abs}(\mathbf{x}_0, \ldots, \mathbf{x}_{n-1})$ where $\min_{abs}$ picks that vector $\mathbf{x}^*$ with $\sum_{j=0}^{b\cdot\ell-1} |\mathbf{x}^*_{(j)}|$ minimal. The standard deviation $\sigma_n = \sqrt{\mathrm{Var}(\mathbf{x}^*_{(0)})} = \cdots = \sqrt{\mathrm{Var}(\mathbf{x}^*_{(\tau-1)})}$ of components in $\mathbf{x}^*$ satisfies*

$$\sigma/\sigma_n \geq c_\tau \sqrt[\tau]{n} + (1 - c_\tau)$$

with c_τ as in Table 3.1 for $\tau \leq 10$ and

$$c_\tau = 0.201514\sqrt{\tau} + 0.323621 \approx \frac{1}{5}\sqrt{\tau} + \frac{1}{3}$$

otherwise.

With Assumption 4 we can now estimate the size of the entries of the variance matrix associated with our elimination tables. That is, a matrix $\mathbf{M}$ where the entry $\mathbf{M}_{(i,j)}$ holds the variance of entries $(b \cdot j, \ldots, b \cdot j + b - 1)$ in T^i.

It is clear that $\mathbf{M}_{(i,j)} = \mathrm{Var}(\mathcal{U}(\mathbb{Z}_q))$ for $0 \leq i < a$ and $i \leq j$ as no reductions

take place for entries on and above 'the main diagonal'. Now, in Algorithm 5 the child components in T^1 are reduced by calling Algorithm 4 $o/(a+1)$ times. Each table T^ℓ has $p^b/2$ rows and we can hence apply Assumption 4 on T^1 with $\tau = b$ and $n = \frac{2o}{(a+1)p^b} + 1$ uniform samples (i.e. $\mathcal{D}$ is the uniform distribution) with standard deviation σ_r. Note that this assumes (idealistically) that each table entry is 'hit' exactly n times during this process. While the expected value of 'hits' each table entry receives is n, ideally we wish to ensure that the majority of table entries are 'hit' at least a certain number of times. Clearly, the number of 'hits' for a given table entry is governed by a binomial distribution - if we consider the problem from a 'balls and bins' perspective, we have the random and independent placing of $o/(a+1)$ balls into $p^b/2$ bins. Then we can approximate the expected number of bins containing j balls by a Poisson random variable with parameter $o/((a+1) \cdot p^b/2)$, implying we can approximate the number of such bins by

$$\frac{(o/((a+1) \cdot p^b/2))^j}{j!} \cdot e^{-\frac{2o}{(a+1) \cdot p^b}}$$

Thus we can approximate the number of bins containing less than M balls by

$$p^b/2 \cdot e^{-\frac{o}{(a+1) \cdot p^b/2}} \cdot \sum_{k=0}^{M-1} \frac{((o/((a+1) \cdot p^b/2)))^k}{k!}$$

Now, it is well known that when the parameter is large enough, the Poisson distribution itself may be approximated closely by a Gaussian distribution. The parameter for the Poisson distribution is $\frac{o}{(a+1) \cdot p^b/2}$ hence, by a standard result, we can approximate this Poisson distribution by a Gaussian of mean $\frac{o}{(a+1) \cdot p^b/2}$ and of variance also $\frac{o}{(a+1) \cdot p^b/2}$.

Thus, under the assumption that these approximations hold, we can approximate the probability that a given bin contains less than x balls by the standard CDF for Gaussians:

$$\frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x - \frac{o}{(a+1) \cdot p^b/2}}{\sqrt{\frac{2 \cdot o}{(a+1) \cdot p^b/2}}}\right)\right).$$

However, in Algorithm 6 below we use the mean and take the distribution of balls in bins into account by reducing the number of observed samples by a fudge factor of 2 in our calculations.

By Assumption 4 we hence get

$$\mathbf{M}_{(1,0)} = \frac{\sigma_r^2}{\left(c_b \sqrt[b]{n} + 1 - c_b\right)^2}$$

Moving on to $\mathbf{M}_{(2,0)}$ we have $\tau = 2b$. Hence, using the same reasoning as in Lemma 13 we expect

$$\mathbf{M}_{(2,0)} = \frac{\sigma_r^2 + \mathbf{M}_{(1,0)}}{\left(c_{2b} \sqrt[2b]{n} + 1 - c_{2b}\right)^2} = \frac{\sigma_r^2 + \frac{\sigma_r^2}{\left(c_b \sqrt[b]{n} + 1 - c_b\right)^2}}{\left(c_{2b} \sqrt[2b]{n} + 1 - c_{2b}\right)^2}$$

and so on for $\mathbf{M}_{(3,0)}, \mathbf{M}_{(4,0)}, \ldots$.

An algorithm for constructing $\mathbf{M}$ is given in Algorithm 6 which we expect this algorithm to give a reasonable approximation of the variances of components of entries in T^ℓ and back up this expectation with empirical evidence in Section 3.5.
Using the matrix $\mathbf{M}$ computed by Algorithm 6, we can estimate the variances of components of $\tilde{\mathbf{a}}_i$ as output by $B^{(n)}_{\mathbf{s},\chi,a-1}$. This result follows immediately from Assumption 4.

Lemma 14. *Let $n \geq 1, q$ be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$ and σ_r be the standard deviation of $\mathcal{U}\mathbb{Z}_{\lfloor q/p \rceil}$. Define $a := \lceil n/b \rceil$ and pick some $p < q$ and let $\mathbf{M}$ be the output of Algorithm 6 under these parameters. Let $(\tilde{\mathbf{a}}_i, c_i)$ be samples returned by $B^{(n)}_{\mathbf{s},\chi,a-1}$. Finally, define $\mathbf{v}$ as the $a-$vector of variances of the components of $\tilde{\mathbf{a}}$ where $\mathbf{v}_{(k)}$ holds the variance of the components $\tilde{\mathbf{a}}_{(b \cdot k)}$ to $\tilde{\mathbf{a}}_{(b \cdot k + b - 1)}$. Under Assumption 4, the components of $\mathbf{v}$ satisfy:*

$$\mathbf{v}_{(i)} = \sigma_r^2 + \sum_{j=i+1}^{a} \mathbf{M}_{(j,i)}.$$

This now allows us to given an expression for the noise distribution output by $B^{(n)}_{\mathbf{s},\chi,a-1}$.

Algorithm 6: Constructing $\mathbf{M}$.

begin
  $T \leftarrow 2 \cdot p^b/2$; // fudge factor: 2
  $n \leftarrow \frac{o}{(a+1)\cdot T} + 1$;
  $\mathrm{Var}_{red} = \mathrm{Var}(\mathcal{U}(\mathbb{Z}_{\lfloor q/p \rceil})) = \sigma_r^2$; // the variance of freshly reduced elements
  $\mathbf{M}$ is an $a \times a$ matrix;
  for $0 \leq r < a$ **do**
    for $0 \leq c < a$ **do**
      $\mathbf{M}_{(r,c)} \leftarrow \mathrm{Var}(\mathcal{U}(\mathbb{Z}_q))$; // elements on and above main diagonal are not reduced
  for $1 \leq t < a$ **do**
    // row t is the sum of previous rows + one fresh element being added for each index
    for $0 \leq i < t$ **do**
      $\mathbf{M}_{(t,i)} \leftarrow \mathrm{Var}_{red} + \sum_{j=i+1}^{t-1} \mathbf{M}_{(j,i)}$;
    $\tau \leftarrow b \cdot \ell$;
    for $0 \leq i < t$ **do**
      $\mathbf{M}_{(t,i)} \leftarrow \frac{\mathbf{M}_{(t,i)}}{\left(c_\tau \sqrt[\tau]{n} + 1 - c_\tau\right)^2}$;

Lemma 15. *Let $n \geq 1$ be the dimension of the* LWE *secret vector, q be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$. Define $a := \lceil n/b \rceil$ and pick some $p < q$ and let $\mathbf{v}$ be as in Lemma 14. Let $(\tilde{\mathbf{a}}_i, \tilde{c}_i)$ be outputs of $B^{(n)}_{\mathbf{s},\chi,a-1}$. We assume that Assumptions 3 and 4 hold. Then as a increases the distribution of $\tilde{c}_i$ approaches a discrete Gaussian distribution modulo q with standard deviation*

$$\sigma_{total} := \sqrt{2^a\sigma + b\,\sigma_r^2\sigma_s^2 \sum_{i=0}^{a-1} \mathbf{v}_{(i)}} \leq \sqrt{2^a\sigma + (2^a - 1) \cdot b \cdot \sigma_r^2\sigma_s^2}.$$

Proof. The standard deviation follows from Assumption 3 and Lemma 14. Since the distribution is formed by adding up 2^a vectors it approaches a discrete Gaussian distribution when considered over $\mathbb{Z}$ as a increases by the Central Limit Theorem. □

Assumption 5. *We assume that Lemma 15 holds for $128 \leq n$, i.e. the values of n considered in this work.*

3.4 Complexity

Finally, we analyse the complexity of the algorithms presented until now. To do so, we assume that Assumptions 3, 4, and 5 hold. Lemma 15 allows us to estimate the numbers of samples needed to distinguish the outputs of $B^{(n)}_{\mathbf{s},\chi,a-1}$ if $B^{(n)}_{\mathbf{s},\chi,-1}$ returns LWE samples from uniform. For this, we rely on standard estimates from [61] (as used in Chapter 2) for the number of samples required to distinguish. This estimate provides a good approximation for the advantage obtainable in distinguishing between $\mathcal{U}(\mathbb{Z}_q)$ and a discrete Gaussian reduced mod q with standard deviation σ_{total}. In particular, we compute the advantage as

$$\text{Adv} = \exp\left(-\pi\left(\frac{\sigma_{total} \cdot \sqrt{2\pi}}{q}\right)^2\right). \tag{3.4}$$

We can now state the overall complexity of running the algorithm in Theorem 4. Remark that the proof of next two results are omitted; they follow by an easy adaptation of the proof of Lemma 5.

Theorem 4. *Let $n \geq 1$ be the dimension of the* LWE *secret vector, q be a modulus, $b \in \mathbb{Z}$ with $1 \leq b \leq n$ and σ_s the standard deviation of the secret vector components. Let also σ_r be the variance of random elements in $\mathbb{Z}_{\lfloor q/p_{\text{small}} \rceil}$. Define $a := \lceil n/b \rceil$ and pick a pair (p_{small}, o) such that $b\,\sigma_r^2\sigma_s^2 \sum_{i=0}^{a-1} \mathbf{v}_{(i)} \leq 2^a\sigma$, where $\mathbf{v}_{(i)}$ is defined as in Lemma 15. Then $B^{(n)}_{\mathbf{s},\chi,a-1}$ will return $(\tilde{\mathbf{a}}_0, \tilde{c}_0), \ldots, (\tilde{\mathbf{a}}_{m-1}, \tilde{c}_{m-1})$ where $\tilde{c}_i$ has standard deviation $\leq \sqrt{2^{a+1}} \cdot \sigma$. Furthermore, this costs*

$$\frac{p_{\text{small}}^b}{2} \cdot \left(\frac{a(a-1)}{2} \cdot (n+1) \right) + (m+o)\; n\, a$$

additions in $\mathbb{Z}_q$ and $a \cdot \left(\frac{p_{\text{small}}^b}{2}\right) + m + o$ calls to $L^{(n)}_{\mathbf{s},\chi}$.

The memory requirement for storing each table is established in Corollary 5 below.

Corollary 5. *The memory required to store the table T^i is upper-bounded by*

$$\frac{p_{\text{small}}^b}{2} \cdot a \cdot (n+1)$$

elements in $\mathbb{Z}_q$, each of which requires $\lceil \log_2(q) \rceil$ bits of storage.

To clarify the impact of Theorem 4, we consider $o = 0$ – i.e. the case discussed in Section 3.2 – on classical parameters of LWE.

Corollary 6. *Let $q \approx n^c$, for some constant $c > 0$, and $\alpha = n^{1/2-c}$ such that $\sigma \approx \alpha q \approx \sqrt{n}$. Let $d \approx \min\{1, \sigma_s/\sqrt{12}\}$. Furthermore, let $a = \log_2 n$ and $b = n/\log_2 n$ be the usual choices of parameters for BKW. Assume σ_s does not depend on n. Then, solving Decision-*LWE *costs at most*

$$\mathcal{O}\left(2^{n\left(c + \frac{\log_2 d - \frac{1}{2}\log_2\log_2 n}{\log_2 n}\right)} \cdot n \log_2^2 n \right)$$

operations in $\mathbb{Z}_q$. We also need to store $\mathcal{O}\left(2^{n\left(c + \frac{\log_2 d - \frac{1}{2}\log_2\log_2 n}{\log_2 n}\right)} \cdot n \log_2 n \right)$ elements in $\mathbb{Z}_q$.

Proof. First, we recall that the time complexity of the BKW algorithm, under these parameters and as given in Corollary 3, is $\approx a^2\, n\, q^b$. Note that the memory needed

is also $\approx a\, n\, q^b$. With the parameters considered, this yields a time complexity dominated by $\mathcal{O}\left(n^{cn/\log_2 n} \cdot n \log_2^2 n\right) = \mathcal{O}\left(2^{c\,n} \cdot n \log_2^2 n\right)$.

This can be improved by first performing a one-shot modulus switching, as explained in Section 3.2.3, and then using BKW on this pre-processed instances. A good choice is to take $p_{\text{small}} \approx \min\{q,\ \frac{\sigma_s}{\sigma}\sqrt{\frac{n}{12}} \cdot q\}$, which simplifies to $\min\{q,\ \frac{\sigma_s}{\sqrt{12}} \cdot q\}$ or $d \cdot q$, with $0 < d \leq 1$, under these parameter choices. This allows to reduces the complexity to $\mathcal{O}\left((dn^c)^{n/\log_2 n} \cdot n \log_2^2 n\right) = \mathcal{O}\left(d^{n/\log_2 n} \cdot 2^{cn} \cdot n \log_2^2 n\right) = \mathcal{O}\left(2^{n\left(c+\frac{\log_2 d}{\log_2 n}\right)} \cdot n \log_2^2 n\right)$. Since $\log_2 d < 0$, this indeed improves the complexity of the plain BKW.

Applying lazy modulus switching, once can reduce p_{small} by an additional factor of $\sqrt{a} = \sqrt{\log_2 n}$ (Corollary 4). This gives:

$$\mathcal{O}\left(\left(\frac{dn^c}{\sqrt{\log_2 n}}\right)^{n/\log_2 n} \cdot n \log_2^2 n\right) = \mathcal{O}\left(2^{n\left(c+\frac{\log_2 d'}{\log_2 n}\right)} \cdot n \log_2^2 n\right), \text{ with } d' = \left(\frac{d}{\sqrt{\log_2 n}}\right).$$

Finally $\log_2 d' = \log_2 d - \frac{1}{2}\log_2 \log_2 n$, and then:

$$\mathcal{O}\left(\left(\frac{dn^c}{\sqrt{\log_2 n}}\right)^{n/\log_2 n} \cdot n \log_2^2 n\right) = \mathcal{O}\left(2^{n\left(c+\frac{\log_2 d - \frac{1}{2}\log_2 \log_2 n}{\log_2 n}\right)} \cdot n \log_2^2 n\right).$$

The very same arguments yields the memory complexity claimed. □

3.5 Implementation

We implemented the algorithm presented in this chapter, i.e. Algorithm 5, to verify the assumptions made in this chapter and to confirm the expected behaviour of the algorithm. Our implementation supports machine size ring elements and integer values for b. We mention that our implementation is not optimised and hence we do not report CPU times.

3.5.1 Correctness of Algorithm 6: Constructing M

The behaviour our algorithm depends critically on the quality of approximation made in Algorithm 6. We hence verified that the matrix $\mathbf{M}$ returned by that algorithm matches the actual variances observed in practice. We start, with an example where the prediction is quite precise. We considered the parameters $q = 65521$, $a = 20$, $b = 2$, $p = 2^{11}$ and $o = 2^{26}$.

Algorithm 6 returns the following matrix

$$\log_2 \mathbf{M} = \begin{pmatrix}
28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
6.6 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
7.7 & 7.0 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
8.6 & 7.9 & 7.1 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
9.5 & 8.9 & 8.1 & 7.2 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
10.5 & 9.8 & 9.0 & 8.1 & 7.2 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
11.4 & 10.8 & 10.0 & 9.1 & 8.2 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
12.4 & 11.7 & 10.9 & 10.0 & 9.1 & 8.2 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
13.3 & 12.7 & 11.9 & 11.0 & 10.1 & 9.2 & 8.2 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
14.3 & 13.6 & 12.8 & 11.9 & 11.1 & 10.1 & 9.2 & 8.3 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
15.3 & 14.6 & 13.8 & 12.9 & 12.0 & 11.1 & 10.2 & 9.2 & 8.3 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
16.2 & 15.6 & 14.7 & 13.9 & 13.0 & 12.1 & 11.1 & 10.2 & 9.2 & 8.3 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
17.2 & 16.5 & 15.7 & 14.8 & 13.9 & 13.0 & 12.1 & 11.2 & 10.2 & 9.3 & 8.3 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
18.2 & 17.5 & 16.7 & 15.8 & 14.9 & 14.0 & 13.1 & 12.1 & 11.2 & 10.2 & 9.3 & 8.3 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
19.1 & 18.5 & 17.7 & 16.8 & 15.9 & 15.0 & 14.0 & 13.1 & 12.2 & 11.2 & 10.2 & 9.3 & 8.3 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
20.1 & 19.4 & 18.6 & 17.8 & 16.9 & 15.9 & 15.0 & 14.1 & 13.1 & 12.2 & 11.2 & 10.3 & 9.3 & 8.3 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
21.1 & 20.4 & 19.6 & 18.7 & 17.8 & 16.9 & 16.0 & 15.0 & 14.1 & 13.1 & 12.2 & 11.2 & 10.3 & 9.3 & 8.3 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
22.1 & 21.4 & 20.6 & 19.7 & 18.8 & 17.9 & 17.0 & 16.0 & 15.1 & 14.1 & 13.2 & 12.2 & 11.2 & 10.3 & 9.3 & 8.3 & 7.4 & 28.4 & 28.4 & 28.4 \\
23.0 & 22.4 & 21.6 & 20.7 & 19.8 & 18.9 & 17.9 & 17.0 & 16.1 & 15.1 & 14.1 & 13.2 & 12.2 & 11.3 & 10.3 & 9.3 & 8.3 & 7.4 & 28.4 & 28.4 \\
24.0 & 23.3 & 22.5 & 21.7 & 20.8 & 19.9 & 18.9 & 18.0 & 17.0 & 16.1 & 15.1 & 14.2 & 13.2 & 12.2 & 11.3 & 10.3 & 9.3 & 8.3 & 7.4 & 28.4
\end{pmatrix}$$

whereas our implementation constructed tables with the follow variance matrix

$$\log_2 \mathbf{M}' = \begin{pmatrix}
28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
6.6 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
7.6 & 6.9 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
8.5 & 7.8 & 7.1 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
9.5 & 8.8 & 8.0 & 7.2 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
10.4 & 9.7 & 8.9 & 8.1 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
11.4 & 10.7 & 9.9 & 9.1 & 8.2 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
12.4 & 11.6 & 10.8 & 10.0 & 9.2 & 8.3 & 7.3 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
13.3 & 12.6 & 11.8 & 11.0 & 10.1 & 9.2 & 8.3 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
14.3 & 13.6 & 12.8 & 12.0 & 11.1 & 10.2 & 9.3 & 8.3 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
15.3 & 14.5 & 13.8 & 12.9 & 12.1 & 11.2 & 10.3 & 9.3 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
16.3 & 15.5 & 14.7 & 13.9 & 13.1 & 12.2 & 11.3 & 10.3 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
17.3 & 16.5 & 15.7 & 14.9 & 14.0 & 13.2 & 12.2 & 11.3 & 10.3 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
18.3 & 17.5 & 16.7 & 15.9 & 15.0 & 14.1 & 13.2 & 12.3 & 11.3 & 10.4 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
19.3 & 18.5 & 17.7 & 16.9 & 16.0 & 15.1 & 14.2 & 13.3 & 12.3 & 11.4 & 10.4 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
20.3 & 19.5 & 18.7 & 17.9 & 17.0 & 16.1 & 15.2 & 14.3 & 13.3 & 12.4 & 11.4 & 10.4 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
21.3 & 20.5 & 19.7 & 18.9 & 18.0 & 17.1 & 16.2 & 15.3 & 14.3 & 13.4 & 12.4 & 11.4 & 10.4 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
22.2 & 21.5 & 20.7 & 19.8 & 19.0 & 18.1 & 17.2 & 16.3 & 15.3 & 14.3 & 13.4 & 12.4 & 11.4 & 10.4 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 & 28.4 \\
23.2 & 22.4 & 21.7 & 20.8 & 20.0 & 19.1 & 18.2 & 17.3 & 16.3 & 15.3 & 14.4 & 13.4 & 12.4 & 11.4 & 10.4 & 9.4 & 8.4 & 7.4 & 28.4 & 28.4 \\
24.2 & 23.4 & 22.6 & 21.8 & 21.0 & 20.1 & 19.2 & 18.2 & 17.3 & 16.3 & 15.4 & 14.4 & 13.4 & 12.4 & 11.4 & 10.4 & 9.4 & 8.4 & 7.4 & 28.4
\end{pmatrix}$$

To highlight the limitations of Algorithm 6 we consider the parameters $q = 65521, n = 30, b = 3, \sigma = 1.0, p = 2^8$ and $o = 2^{29}$. Algorithm 6 predicts

$$\log_2 \mathbf{M}' = \begin{pmatrix}
28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
11.8 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
12.9 & 12.5 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
13.8 & 13.4 & 12.8 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
14.6 & 14.2 & 13.6 & 12.9 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
15.5 & 15.1 & 14.5 & 13.8 & 13.0 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
16.3 & 15.9 & 15.3 & 14.6 & 13.9 & 13.1 & 28.4 & 28.4 & 28.4 & 28.4 \\
17.2 & 16.8 & 16.2 & 15.5 & 14.8 & 13.9 & 13.1 & 28.4 & 28.4 & 28.4 \\
18.1 & 17.7 & 17.1 & 16.4 & 15.6 & 14.8 & 14.0 & 13.1 & 28.4 & 28.4 \\
19.0 & 18.6 & 18.0 & 17.3 & 16.5 & 15.7 & 14.9 & 14.0 & 13.2 & 28.4
\end{pmatrix}$$

but implementation produced tables with variance matrix

$$\log_2 \mathbf{M} = \begin{pmatrix}
28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
11.7 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
12.6 & 12.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
13.4 & 13.2 & 12.8 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
14.1 & 13.9 & 13.5 & 13.0 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
14.8 & 14.6 & 14.2 & 13.7 & 13.1 & 28.4 & 28.4 & 28.4 & 28.4 & 28.4 \\
15.4 & 15.3 & 15.0 & 14.5 & 13.9 & 13.2 & 28.4 & 28.4 & 28.4 & 28.4 \\
16.1 & 16.0 & 15.7 & 15.3 & 14.7 & 14.0 & 13.2 & 28.4 & 28.4 & 28.4 \\
16.9 & 16.7 & 16.4 & 16.0 & 15.5 & 14.9 & 14.1 & 13.3 & 28.4 & 28.4 \\
17.6 & 17.4 & 17.2 & 16.8 & 16.3 & 15.7 & 15.0 & 14.2 & 13.3 & 28.4
\end{pmatrix}$$

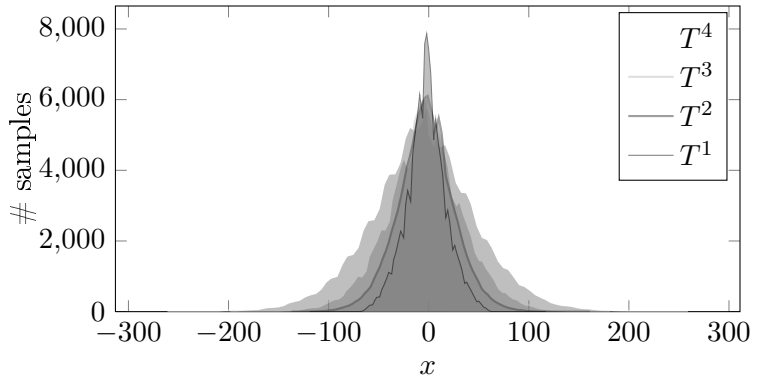

Figure 3.2: Histogram of distribution of 0th component in T^ℓ for $1 \leq \ell \leq 4$ with parameters $q = 32003$, $b = 2$, $p = 2^9$, $n = 10$, and $o = 2^{20}$.

which means we overestimated the variance of child components in our tables T. The main reason for this effect is that our approximation of the shrinking of variances when taking the minimum is based on the uniform distribution. However, the distributions actually governing the entries of our tables T^ℓ are not uniform, as discussed in 3.3. Figure 3.5.1 gives histograms for these distributions.
Overall, for the instances considered our estimates are pessimistic, which means we expect our algorithm to perform better in practice than predicted. A more refined model for its behaviour is hence a good topic for future work.

3.6 Parameters

To understand the behaviour of our more careful modulus switching technique for concrete parameters, we compare it with one-shot modulus switching. Specifically, we consider the standard BKW algorithm as analysed in Chapter 2. We also compare with the BKZ (2.0) algorithm when applied to SIS instances derived from LWE samples and with a simple meet-in-the-middle (MITM) approach or generalised birthday attack.

INSTANCES. We choose $n \in [128, 256, 512, 1024, 2048]$ and – using the LWE generator [8] – pick $q \approx n^2$ and $\sigma = \frac{q}{\sqrt{2\pi n \log_2^2 n}}$ as in Regev's original encryption scheme [86]. We then consider both variants of what is known as binary-LWE in the literature: we first assume $\mathbf{s} \leftarrow_\$ \mathcal{U}(\mathbb{Z}_2^n)$ as in [22] and then $\mathbf{s} \leftarrow_\$ \mathcal{U}(\{-1, 0, 1\}^n)$ as in [41]. However, in contrast to those works we assume an unbounded number of samples being available to the attacker to establish the performance of the algorithms discussed here under optimal conditions.

BKW. For complexity estimates of the plain BKW algorithm we rely on Chapter 2. Recall that, there, the BKW algorithm takes a parameter t which controls the addition depth $a := t \log_2 n$. Here we first pick $t = 2(\log_2 q - \log_2 \sigma)/\log_2 n$ which ensures that the standard deviation of the noise after a levels of additions grows only as large as the modulus. We then slowly increase t in steps of 0.1 until the performance of the algorithm is not estimated to improve any further because too many samples are needed to perform the distinguishing step. In the same manner as in chapter 2, we translate operations in $\mathbb{Z}_q$ into "bit operations" by multiplying by $\log_2 q$.

MITM. One can also solve small secret LWE with a meet-in-the-middle attack that requires (somewhat optimistically) $\approx \mathfrak{c}^{n/2}$ time and space where $\mathfrak{c}$ is the cardinality of the set from which each component of the secret is sampled (so $\mathfrak{c} = 2$ or $\mathfrak{c} = 3$ for binary-LWE depending on the definition used): compute and store a sorted list of all $\mathbf{A}\mathbf{s}'$ where $\mathbf{s}' = (\mathbf{s}_{(0)}, \ldots, \mathbf{s}_{(n/2)-1}, 0, 0, \ldots, 0)$ for all possible $\mathfrak{c}^{n/2}$ choices for $\mathbf{s}'$. Then compute $\mathbf{c} - \mathbf{A}\mathbf{s}''$ where we have $\mathbf{s}'' = (0, 0, \ldots, 0, \mathbf{s}_{(n/2)}, \ldots, \mathbf{s}_{n-1})$ and check for a vector that is close to this value in the list.

RESULTS FOR $\mathbf{s} \leftarrow_\$ \mathcal{U}(\mathbb{Z}_2^n)$. In Table 3.2 we give the number of bit operations ("$\log \mathbb{Z}_2$"), calls to the LWE oracle ("$\log L_{\mathbf{s},\chi}^{(n)}$") and memory requirement ("$\log \mathrm{mem}$") for BKW without any modulus reduction to establish the baseline. All costs are

	MITM		BKZ [61]			BKZ 2.0 [63]			BKW [6]			
n	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	t	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$
128	67.8	64	-18	26.5	65.4	-14	22.5	65.7	3.18	83.9	97.6	90.0
256	132.0	128	-29	38.5	179.5	-35	44.5	178.5	3.13	167.2	182.1	174.2
512	260.2	256	-48	58.5	390.9	-94	104.5	522.8	3.00	344.7	361.0	352.8
1024	516.3	512	-82	93.5	785.0	-265	276.5	1606.2	2.99	688.0	705.5	697.0
2048	1028.5	1024	-133	145.3	1327.6	-636	648.1	3929.5	2.71	1135.3	1153.6	1145.3

Table 3.2: Cost for solving Decision-LWE with advantage ≈ 1 for BKW, BKZ and MITM where q, σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_2^n)$. We run BKZ $1/\epsilon$ times, "$\log \mathbb{Z}_2$" gives the number of "bit operations", "$\log L^{(n)}_{\mathbf{s},\chi}$" the number of LWE oracle calls and "$\log \mathrm{mem}$" the memory requirement of $\mathbb{Z}_q$ elements. All logarithms are base-2.

	BKZ [61]			BKZ 2.0 [63]			BKW [6]			
n	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	t	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$
128	-20	28.3	68.5	-16	24.3	68.6	2.84	76.1	89.6	81.2
256	-31	40.3	172.5	-36	45.3	169.5	2.74	149.6	164.0	156.7
512	-49	59.3	360.2	-89	99.2	457.8	2.76	289.8	305.6	297.9
1024	-80	91.3	701.6	-232	243.2	1312.8	2.78	563.1	580.2	572.2
2048	-133	145.3	1327.6	-636	648.1	3929.5	2.71	1135.3	1153.6	1145.3

Table 3.3: Cost for solving Decision-LWE with advantage ≈ 1 for BKW and BKZ variants where q, σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_2^n)$ after one-shot modulus reduction with $p = q\sqrt{n/12}\sigma_s/\sigma$.

given for the high advantage case, i.e. if $\epsilon \ll 1$ we multiply the cost by $1/\epsilon$.

Table 3.3 gives the running times after modulus reduction with $p = q\sqrt{n/12}\sigma_s/\sigma$. In particular, Table 3.3 lists the expected running time (number of oracle calls and where applicable memory requirements) of running BKW and BKZ after applying modulus reduction.

Finally, Table 3.4 gives the expected costs for solving these LWE instances using the techniques described in this work. We list two variants: one with and one without "unnatural selection" . This is because these techniques rely on more assumptions

	this work (w/o unnatural selection)						this work					
n	t	$\log p$	$\log o$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$	t	$\log p$	$\log o$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$
128	2.98	10	0	64.7	78.2	70.8	2.98	6	60	60.0	74.2	46.3
256	2.83	11	0	127.8	142.7	134.9	2.83	5	117	117.0	132.5	67.1
512	2.70	11	0	235.1	251.2	243.1	2.70	8	225	225.0	241.8	180.0
1024	2.59	12	0	477.4	494.8	486.5	2.59	10	467	467.0	485.0	407.5
2048	2.50	12	0	897.8	916.4	907.9	2.50	10	834	834.0	853.2	758.9

Table 3.4: Cost for solving Decision-LWE with advantage ≈ 1 with the algorithms discussed in this work when $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_2^n)$.

	MITM		BKZ [61]			BKZ 2.0 [63]			BKW [6]			
n	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	t	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$
128	105.2	101.4	-18	26.5	65.4	-14	22.5	65.7	3.18	83.9	97.6	90.0
256	206.9	202.9	-29	38.5	179.5	-35	44.5	178.5	3.13	167.2	182.1	174.2
512	409.9	405.8	-48	58.5	390.9	-94	104.5	522.8	3.00	344.7	361.0	352.8
1024	815.8	811.5	-82	93.5	785.0	-265	276.5	1606.2	2.99	688.0	705.5	697.0
2048	1627.5	1623.0	-141	153.6	1523.6	-773	785.4	5100.0	3.00	1369.8	1388.7	1379.9

Table 3.5: Cost for solving Decision-LWE with advantage ≈ 1 for BKW, BKZ and MITM where q and σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$.

than the rest of this work which means we have greater confidence in the predictions avoiding such assumptions. Yet, as discussed in Section 3.5, these assumptions seem to hold reasonably well in practice.

Results for $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$. Tables 3.5, 3.6 and 3.7 correspond to Tables 3.2, 3.3 and 3.4 and adopt the same notation.

Discussion. The results in this section (summarised in Figure 3.6) indicate that the variants of the BKW algorithms discussed in this work compare favourably for the parameters considered. In particular, in the case $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{0, 1\}^n)$ these BKW variants are the only algorithms which beat the simple MITM approach. For $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\})^n$ that advantage naturally increases. The results in this table also indicate that the unnatural selection strategy has little impact on the overall time complexity. However, it allows to reduce the data complexity, in some cases,

	BKZ [61]			BKZ 2.0 [63]			BKW [6]			
n	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \epsilon$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	t	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$
128	-21	29.3	70.2	-16	24.4	69.8	2.85	76.8	90.2	82.4
256	-31	40.3	175.3	-37	46.3	172.8	2.85	150.4	165.6	153.7
512	-50	60.3	365.0	-90	100.2	467.0	2.76	293.8	309.6	301.9
1024	-81	92.3	710.1	-236	247.2	1339.1	2.78	570.3	587.4	579.4
2048	-134	146.3	1342.3	-647	659.2	4006.5	2.71	1149.0	1167.3	1159.1

Table 3.6: Cost for solving Decision-LWE with advantage ≈ 1 for BKW and BKZ variants where q, σ are chosen as in [86] and $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$ after one-shot modulus reduction with $p = q\sqrt{n/12}\sigma_s/\sigma$.

	this work (w/o unnatural selection)						this work					
n	t	$\log p$	$\log o$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$	t	$\log p$	$\log o$	$\log L^{(n)}_{\mathbf{s},\chi}$	$\log \mathbb{Z}_2$	$\log \mathrm{mem}$
128	2.98	10	0	64.7	78.2	70.8	2.98	6	61	61.0	75.2	46.3
256	2.83	11	0	127.8	142.7	134.9	2.83	5	118	118.0	133.5	67.1
512	2.70	11	0	235.1	251.2	243.1	2.70	8	225	225.0	241.8	180.0
1024	2.59	12	0	477.4	494.8	486.5	2.59	10	467	467.0	485.0	407.5
2048	2.50	12	0	971.4	990.7	907.9	2.50	12	961	961.0	980.2	907.9

Table 3.7: Cost for solving Decision-LWE with advantage ≈ 1 with the algorithms discussed in this work when $\mathbf{s} \leftarrow_{\$} \mathcal{U}(\{-1, 0, 1\}^n)$.

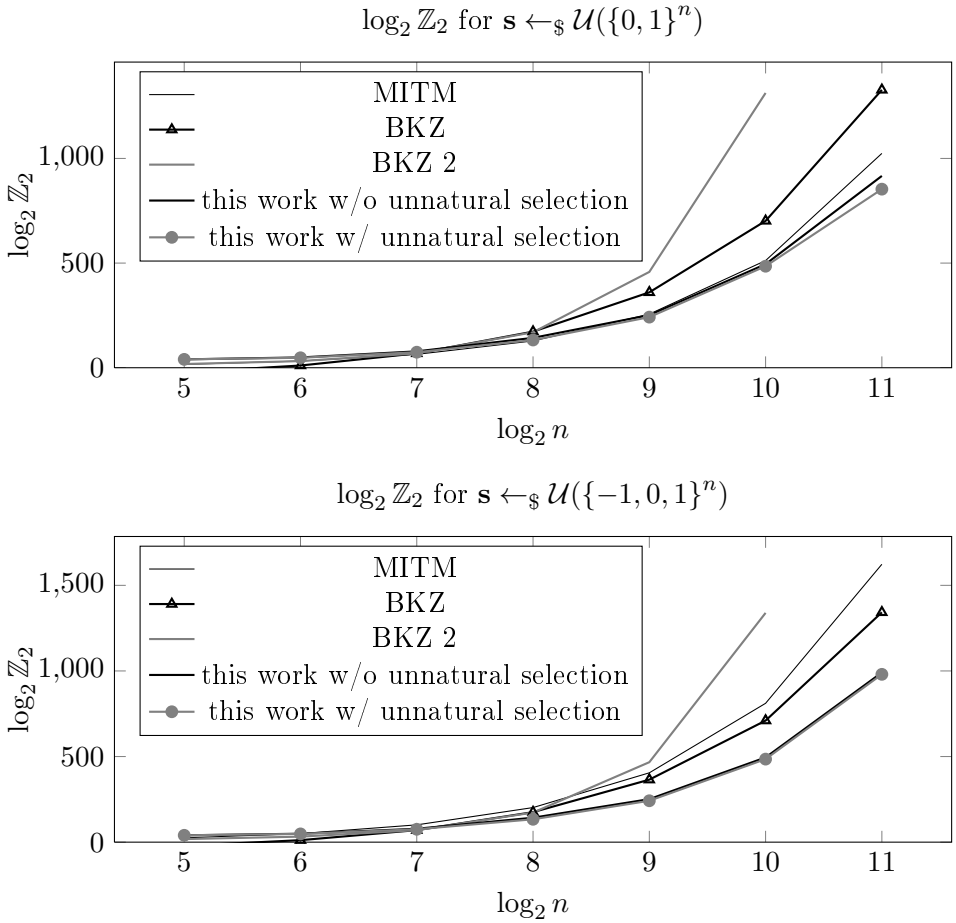

Figure 3.3: Cost of solving Decision-LWE using various algorithms discussed in this work.

considerably. In particular, e.g. considering line 1 of Table 3.7, we note that applying this technique can make the difference between a feasible ($\approx 80 \cdot 1024^4$ bytes) and infeasible ($\approx 1260 \cdot 1024^6$ bytes) attack for a well-equipped attacker.

3.6.1 Independence Assumption

To verify the independence assumption, i.e. Assumption 3, and in particular that the noise part behaves like discrete Gaussian modulo q with $s = \sqrt{2^a}\alpha q$, we ran our implementation for $q = 4093, \sigma = 3.0$, and $b = 2$ with $n = 10$, 25 and 30. In

n	t	$\log p$	$\log o$	$\log_2 \sum A_i^{p^b}$
128	2.98	6	61	14.81
256	2.83	5	118	-7.99
512	2.70	8	225	-323.91
1024	2.59	10	467	< -1000
2048	2.50	12	961	< -1000

Table 3.8: Expected Number of Noise-Element Collisions, no Table-Switching

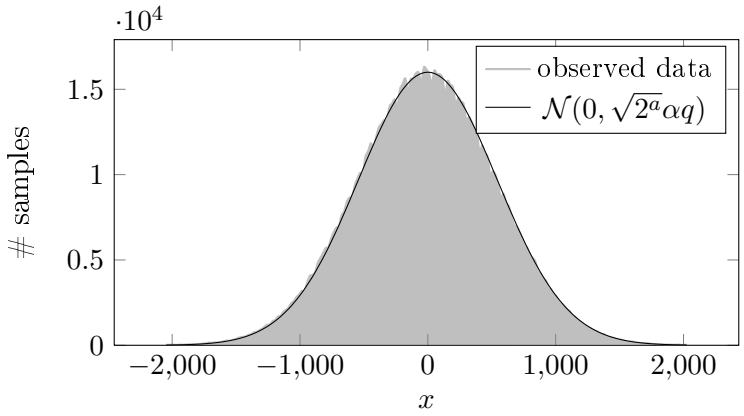

Figure 3.4: Distribution of $c_i - \langle \tilde{\mathbf{a}}_i, \mathbf{s} \rangle$ for parameters $q = 4093, n = 30, a = 15, \sigma = 3.0$.

each case, we asked for 2^{20} samples and extracted the noise by computing $c_i - \langle \tilde{\mathbf{a}}_i, \mathbf{s} \rangle$ and analysed the resulting distributions. In all our experiments, the noise followed a discrete Gaussian closely. In Figure 3.4 we plot a histogram for the experimental data (in grey) for the case $n = 30$ and the expected distribution for $\mathcal{N}(0, \sqrt{2^a}q)$.

If we consider the application of arguments from 2.7 and assume, pessimistically that no replacement of table elements is carried out, we obtain estimates, given in Table 3.8 for the expected number of collisions between noise elements, given the parameters in Table 3.4, thus illustrating that, though more severe than in the standard BKW case, such phenomena can be safely neglected.

3.7 Closing Remarks

In this chapter we investigated applying modulus switching to exploit the presence of a small secret in LWE instances and demonstrated that it can make a significant impact on the complexity of solving such instances. We also adapted the BKW algorithm to perform modulus-switching 'on-the-fly', showing that this approach is superior to performing 'one-shot' modulus reduction on LWE samples prior to solving. Our first variant improves the target modulus by a factor of $\sqrt{\log_2 n}$ in typical scenarios; our second variant mainly improves the memory requirements of the algorithm, one of the key limiting aspects of the BKW algorithm. Our algorithms, however, rely on various assumptions which, though appearing sound, are unproven. Our estimates should thus be considered heuristic, as are performance estimates for all currently-known algorithms for solving LWE. Verifying these assumptions is hence a promising direction for future research. Furthermore, one of the main remaining obstacles for applying the BKW algorithm to cryptographic constructions based on LWE is that it requires an unbounded number of samples ot proceed. Lifting this requirement, if only heuristically, is hence a pressing research question.

3.8 Justification of Assumption 4

We arrive at the approximation in Assumption 4 as follows. We chose $\mathcal{D} = \mathcal{U}\mathbb{Z}_{2^{32}}$ and observed that σ/σ_n behaves linearly when $\tau = 1$ (cf. Figure 3.5). As we increase τ we expect the problem of finding short vectors to become exponentially harder. We also require $\sigma/\sigma_1 = 1$ for any τ. The approximation $c_\tau \sqrt[\tau]{x} + (1 - c_\tau) = \sigma/\sigma_n$ satisfies all these conditions. Furthermore, experimental evidence suggests that there exist c_τ such that a function of this form approximates the observed data closely (cf. Figure 3.5). We then used curve fitting (as implemented in Sage [97] via calls to SciPy [51] which in turn calls MINPACK [74]) on experimental data for $1 \leq n \leq 128$ to find c_τ for $1 \leq \tau \leq 512$; Table 3.1 lists the first 10 such values. The expression $c_\tau \approx 1/5\sqrt{\tau} + 1/3$ was recovered by curve fitting the pairs $(1, c_1), \ldots, (512, c_{512})$

recovered before (cf. Figure 3.6 for a plot of this data and $1/5\sqrt{\tau}+1/3$). However, for small values of τ this expression diverges too much from the observed, which is why we are using explicit values instead. Finally, we assume that all distributions observed during the course of running our algorithm 'shrink' at least as fast as the uniform distribution. An assumption we experimentally verified for the normal distribution and which seems to be confirmed by our experimental results in Section 3.5.

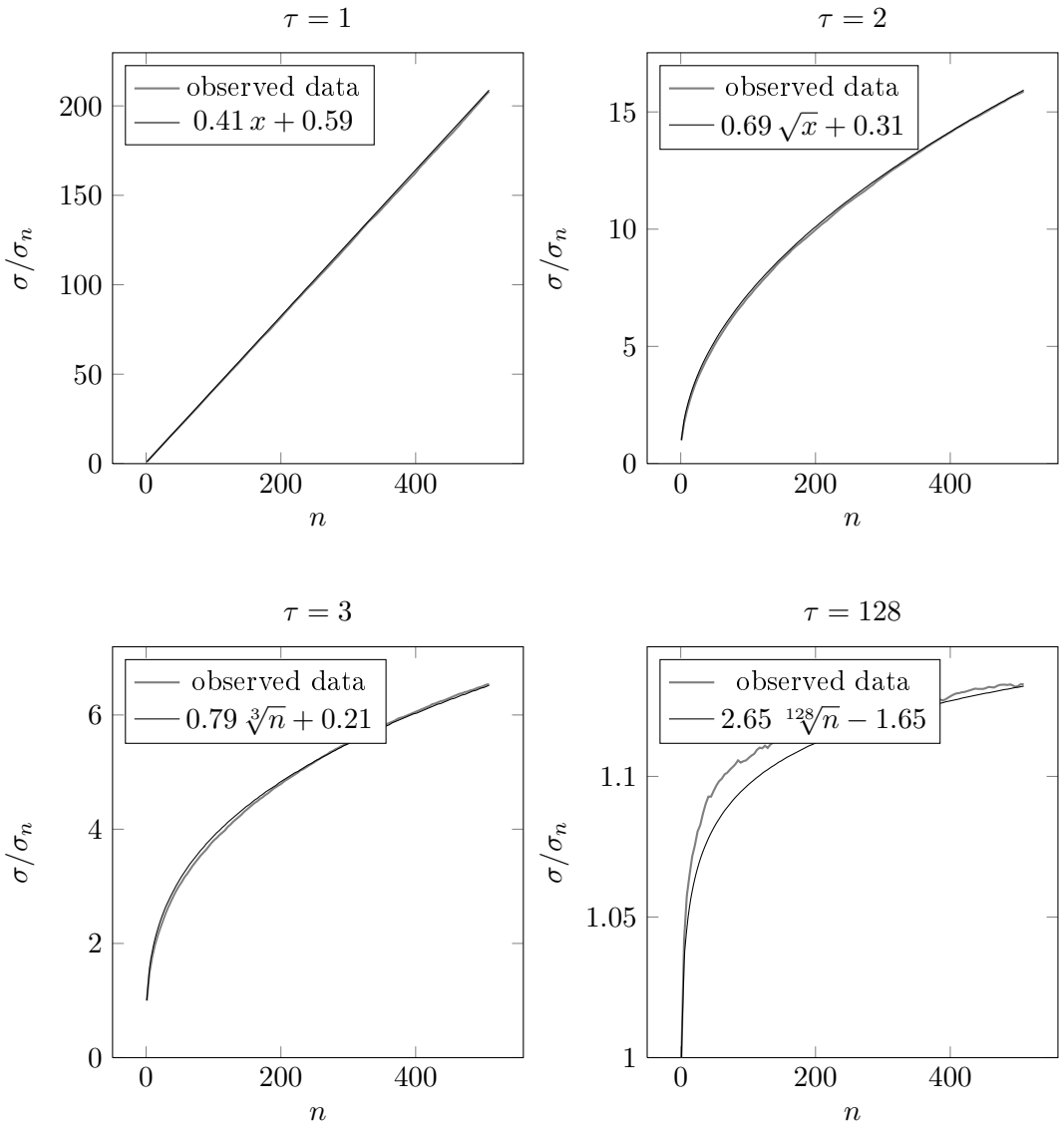

Figure 3.5: σ/σ_n for $1 \leq \tau \leq 3$ and $\tau = 128$.

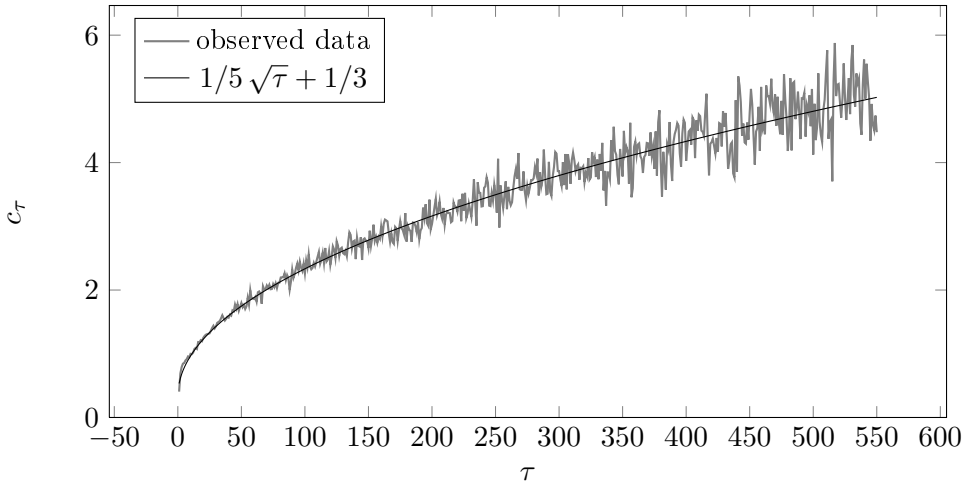

Figure 3.6: c_τ in function of τ

Chapter 4

Applying Kannan's Embedding Approach to LWE

The contents of this chapter are based on the paper "On the Efficacy of Solving LWE by Reduction to Unique-SVP" which appeared at ICISC 2013, Seoul, Korea [9] and was a work conducted in collaboration with M. R. Albrecht and F. Göpfert.

4.1 Introduction

In Chapter 1, we discussed, albeit briefly, the most common techniques for solving LWE instances through the use of lattice basis reduction. In this chapter we examine an approach, Kannan's embedding technique [52], which is lesser-known and which has received little treatment in the literature for the LWE case. We also present the results of experiments using LLL and BKZ. In [67] it is shown that while the embedding approach has been successfully employed in past works (see for instance [76]), the approach remains somewhat mysterious with our current understanding of the efficacy of the approach being comparatively poor.

4.1.1 Related Work

In [64] Liu et. al. investigate similar questions, though their work lacks an experimental component which, in our opinion, forms an indispensable part of any such work, given the current state of knowledge regarding the concrete complexity of unique-SVP. The current understanding of how a particular gap is related to the success of a particular reduction algorithm in disclosing a shortest vector, is poor. In [38] the results of a number of experiments were reported in which the authors examined the success of a number of algorithms in disclosing a shortest vector when (at least) a good approximation to the gap was known (though not in bounded-distance decoding/LWE cases). A simple model was proposed as a criterion for the success of a particular algorithm and particular class of lattices, with 'explaining the unique-SVP phenomenon' being posed as an open question. This general question was partly answered by the work of Luzzi et. al. [62], explaining one of the questions posed by [38], i.e. why a κ-Hermite SVP algorithm (characterised by δ_0) solves unique-SVP with gap of order $\delta_0^{\dim(\mathcal{L})}$ rather than requiring a gap of order $\delta_0^{2\cdot\dim(\mathcal{L})}$.

4.1.2 Organisation

We provide some background discussion in Section 4.2 and discuss the embedding gap in Section 4.3.1. In Section 4.4 we apply the embedding approach to lattices derived from LWE instances. Finally, in Section 4.5 we discuss the limits of the embedding approach and compare our results with results from the literature.

4.2 Background and Notation

If we have an algorithm which solves κ-Hermite SVP for lattices of dimension n, we say that the algorithm attains a *root Hermite factor* of $\delta_0 := \kappa^{1/n}$. It is an experimentally-verified heuristic [38] that the root Hermite factor a given algorithm attains converges swiftly with increasing dimension. Now, if we have an algorithm which can solve Hermite-SVP with approximation factor κ, we can use this algorithm

linearly many times [66] to solve Approx-SVP with approximation factor κ^2. Hence, we can use our κ-HSVP algorithm to solve uSVP instances in which the gap is at least κ^2. Similarly, if we have a root Hermite factor δ_0 characterising our Hermite-SVP algorithm, we can solve uSVP instances of gap δ_0^{2m}. This was improved in [62] to δ_0^m by proving a reduction from unique-SVP to Hermite-SVP. However, one of the conclusions from [38], as we discuss later, is that, as with the gulf between the theoretical and practical performance of lattice reduction algorithms, we can generally solve uSVP instances with much smaller gap.

All experiments reported in this chapter were carried out using the NTL implementation of BKZ and all LWE instances were generated using the afore-mentioned LWE instance generator. In this work, due to the probabilistic nature of our experiments (as opposed to one-off reductions), we are constrained to experiments in relatively low block-size. Here, we report the results of experiments using block-sizes of 5 and 10 (with no pruning) which collectively took around 2 months on two servers. Partial results for BKZ with a block-size of 20 indicate no discrepancy with our conclusions. While much larger block-sizes, in particular with pruning, have been achieved, our experiments required the execution of a substantial number (100) of reductions for each attempt at a solution. For simplicity, we assume throughout that enough LWE samples are exposed by a cryptosystem to allow the employment of the embedding technique in the optimal lattice dimension. While this may not always be the case in reality, in this chapter (as in this thesis in general) we are concerned primarily with the LWE problem "in theory".

The following tail bound on discrete Gaussians is needed. This tail bound is obtained by employing the tail-bound on $D_{\mathbb{Z},s}$ from [13] and then observing that the product distribution of n copies of this distribution gives $D_{\mathbb{Z}^n,s}$.

Lemma 16. *Let $c \geq 1$ and $C = c \cdot \exp((1-c^2)/2) < 1$. Then for any real $s > 0$ and*

any integer $n \geq 1$ *we have*

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

4.2.1 Concrete Hardness of uSVP

It is folklore that the presence of a significant gap between the first and second minima of a lattice makes finding a shortest non-zero vector somewhat easier than would otherwise be the case, with an exponential gap allowing a shortest non-zero vector to be disclosed by application of LLL. However, in cases with sub-exponential gap, the success of lattice reduction algorithms in disclosing shortest non-zero vectors is poorly understood with a brief investigation in [38] being (to the best of our knowledge) the only practical investigation of such effects.

In [38] it was posited that given a lattice-reduction algorithm which we assume to be characterised by a root Hermite factor δ_0 and a (full-rank) m-dimensional lattice Λ, the algorithm will be successful in disclosing a shortest non-zero vector with high probability when $\lambda_2(\Lambda)/\lambda_1(\Lambda) \geq \tau \cdot \delta_0^m$, where τ was taken to be a constant depending both on the nature of the lattices examined and also on the lattice reduction algorithm applied. In [38] values of τ ranging between 0.18 and 0.48 were experimentally-derived for various classes of lattices (though not LWE-derived lattices) and algorithms. However, the phrase 'with high probability' was not elaborated on in [38] and thus it is unclear as to whether a fixed threshold was used throughout the experiments in [38] or a variable threshold.

We can speculate that the probabilistic behaviour observed in our experiments arises from phenomena somewhat analogous to the bearing of starting conditions on finding global minima in optimisation problems, though such considerations are beyond the scope of this work.

4.3 The Embedding Approach

In this section we outline and examine our application of Kannan's embedding technique, the resulting λ_2/λ_1-gap distributions and the resulting implications for the success of the approach.

4.3.1 Construction of Embedding Lattices

Given a set of m LWE samples $(\mathbf{a}_i, c_i)$, we construct a Matrix-LWE instance of dimension m by constructing a matrix $\mathbf{A}'$ by taking the $\mathbf{a}_i$ vectors to be the columns of $\mathbf{A}'$ and form the vector $\mathbf{c}$ from the c_i's to obtain $\mathbf{c} = \mathbf{A}'^T\mathbf{s} + \mathbf{e}$. We consider the problem of being given a matrix-LWE instance $(\mathbf{A}', \mathbf{c})$ of dimension m and forming a lattice basis as follows. We take the matrix $\mathbf{A}' \in \mathbb{Z}_q^{n\times m}$ and calculate the reduced echelon form $\mathbf{A}'' \in \mathbb{Z}_q^{n\times m}$. For the right permutation matrix $\mathbf{P} \in \mathbb{Z}^{m\times m}$, we obtain the form $\mathbf{A}'' = \begin{pmatrix} \mathbf{I} & \overline{\mathbf{A}} \end{pmatrix}$ with $\mathbf{I} \in \mathbb{Z}_q^{n\times n}$ and $\overline{\mathbf{A}} \in \mathbb{Z}_q^{n\times(m-n)}$. If we interpret this matrix as a matrix over $\mathbb{Z}$, extend it with $\begin{pmatrix} \mathbf{0} & q\mathbf{I} \end{pmatrix} \in \mathbb{Z}^{(m-n)\times m}$ and define

$$\mathbf{A} = \begin{pmatrix} \mathbf{I} & \overline{\mathbf{A}} \\ \mathbf{0} & q\mathbf{I} \end{pmatrix} \mathbf{P}^{-1},$$

$\mathbf{A}$ is a basis of the lattice $\{\mathbf{v} \in \mathbb{Z}^m \mid \exists \mathbf{x} \in \mathbb{Z}_q^n : \mathbf{x}\mathbf{A} = \mathbf{v} \mod q\}$. Now, given this $\mathbf{A}$ and a target vector $\mathbf{t} \in \mathbb{Z}_q^m$ and attempting to solve the LWE instance by reducing the embedding lattice basis

$$\mathbf{B}_{(\mathbf{A},\mathbf{t},t)} := \begin{pmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{t} & t \end{pmatrix}$$

where $t > 0$ is an embedding factor to be determined. We then define $\Lambda_e := \mathcal{L}(\mathbf{B})$. Note that, with overwhelming probability, $\det(\Lambda_\mathbf{e}) = t \cdot q^{m-n}$, i.e. $\mathbf{A}'$ has full rank over $\mathbb{Z}_q$.

It is well-known [67] that $1/(2\gamma)$-BDD can be reduced to solving γ-USVP by setting the embedding factor $t \geq \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{A}))$. In practice, however, employing a

smaller embedding factor generally allows us to create a unique-SVP instance with larger λ_2/λ_1 gap than by setting $t = \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{A}))$. However, by setting $t < \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{A}))$, with non-zero probability there exists a vector $\mathbf{v} \in \Lambda$ such that $\|\mathbf{v} + c \cdot [\mathbf{t} \quad t]\| < \|[\mathbf{e} \quad t]\|$ where $c \in \mathbb{Z}$ and, in general, if $t < \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{A}))$, we will have $\lambda_2(\Lambda_e) < \lambda_1(\mathcal{L}(\mathbf{A}))$. Thus when we reduce t, quantification of the resulting λ_2/λ_1 gap becomes difficult.

To the best of our knowledge, no good model exists to determine the distribution of the lattice gap when taking an embedding factor smaller than $\|\mathbf{e}\|$. To attempt circumvention of such difficulties, we conduct experiments on LWE-derived unique-SVP lattices, examining firstly the λ_2/λ_1 gap required for success when we set $t = \lceil \text{dist}(\mathbf{t}, \mathcal{L}(\mathbf{A})) \rceil$ (where we know λ_2/λ_1) and then for the case $t = 1$, under the assumption that the 'necessary gap' is unlikely to change, allowing us to derive analogous models.

4.3.2 On the Determination of τ when $t = \lceil \|\mathbf{e}\| \rceil$

As mentioned in 4.2.1, we employ the simple model of Gama and Nguyen for predicting the success of a particular basis-reduction algorithm in recovering a shortest non-zero vector, namely that there exist values of τ such that, for a given probability, basis-reduction algorithm and lattice class, the basis-reduction algorithm finds a shortest non-zero vector with probability greater or equal than the given probability over the random choice of lattices in the class whenever $\lambda_2(\Lambda)/\lambda_1(\Lambda) \geq \tau \cdot \delta_0^m$ where Λ represents a random choice of lattice in the class with dimension m. Thus, if we are able to sample such lattices randomly, determining a particular value of τ requires us to know (at least approximately) the λ_2/λ_1 gap of the lattices we are dealing with.

In the q-ary lattices we consider (i.e. lattices of the form $\mathcal{L}(\mathbf{A})$), unfortunately, there is no known good bound (in the Euclidean norm) on the first minimum when $m < 5n \log_2 q$. The case of $m \geq 5n \log_2 q$ is dealt with in [96]. For the case of ran-

dom lattices (in the sense of [43]), it is known that with overwhelming probability the minima of such an n-dimensional lattice are all asymptotically close to the Gaussian heuristic i.e.

$$\frac{\lambda_i(\Lambda)}{\mathrm{vol}(\Lambda)^{1/n}} \approx \frac{\Gamma(1+n/2)^{1/n}}{\sqrt{\pi}}.$$

Now the q-ary lattices (e.g. $\mathcal{L}(\mathbf{A})$) widely employed in lattice-based cryptography are not random in this sense. However, in all cases, it appears that the Gaussian heuristic appears to hold exceedingly well for such lattices (at least for the first minimum and with the added property that we always have vectors of norm q within the lattice), thus we assume throughout that the first minimum of such lattices is lower-bounded by the Gaussian heuristic with overwhelming probability.

For the first minimum of the embedding lattices, we only deal with this explicitly in the 'known-λ_1' case where we take this to be $\lambda_1(\Lambda_e) = \sqrt{2} \cdot \|\mathbf{e}\|$.
Then we can state the following lemma, the proof of which we omit

Lemma 17. *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, let $s > 0$ and let $c > 1$. Let $\mathbf{e}$ be drawn from $D_{\mathbb{Z}^m,s}$. Under the assumption that $\lambda_1(\Lambda(\mathbf{A})) \geq \mathrm{GH}_{q,n,m}$[1] and that the rows of $\mathbf{A}$ are linearly-independent over $\mathbb{Z}_q$, we can create an embedding lattice Λ_e with λ_2/λ_1-gap greater than*

$$\frac{\min\left\{q, \frac{q^{1-\frac{n}{m}}\Gamma(1+\frac{m}{2})^{\frac{1}{m}}}{\sqrt{\pi}}\right\}}{\frac{cs\sqrt{m}}{\sqrt{\pi}}} \approx \frac{\min\left\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\right\}}{\frac{cs\sqrt{m}}{\sqrt{\pi}}}$$

with probabillity greater than $1 - (c \cdot \exp((1-c^2)/2))^m$.

We wish to obtain the value of m for which we can expect to gain the largest gaps (again, probabilistically).

[1] We employ the notation $\mathrm{GH}_{q,n,m}$ to denote the application of the Gaussian heuristic to an LWE lattice formed from m LWE samples of dimension n, with modulus q.

Corollary 7. *Under the assumptions stated in Lemma 17 and for a fixed value of c ($c > 1$), we can construct embedding lattices with the largest possible gap when*

$$q = \frac{q^{1-\frac{n}{m}}\Gamma(1+\frac{m}{2})^{\frac{1}{m}}}{\sqrt{\pi}}.$$

Proof. We assume that the approximation is close enough such that the maximum occurs for the same value of m. Consider the functions:

- $f_0(m) = \frac{\sqrt{\pi}q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}}{cs\sqrt{m}}$ where $c > 1$, $s > 0$.
- $f_1(m) = \frac{q\sqrt{\pi}}{cs\sqrt{m}}$ where $c, m > 1$ and $s > 0$.

Then $f_1(m)$ is clearly monotonically-decreasing and $f_0(m)$ has the form $f_0(m) = d \cdot q^{1-\frac{n}{m}}$, where d is a positive constant, hence is clearly monotonically-increasing under the conditions given. □

Thus, in our experiments, it appears valid to derive values of τ by assuming the Gaussian heuristic holds and that the (Euclidean) norm of the noise vector is equal to the expected value.

4.3.3 On the Determination of τ when $t < \lceil \|\mathbf{e}\| \rceil$

However, as mentioned, the employment of an embedding factor smaller than the norm of the noise vector $\mathbf{e}$ generally leads to a modest decrease in the size of the second minimum of the resulting lattice. In all cases observed, however, this decrease in the second minimum is less than the corresponding decrease in the first minimum (as a result of making the target vector shorter), leading to a more effective attack. However, quantification of the resulting gap is not simple – we know of no efficient method for determining the distribution of the λ_2/λ_1 gap under such conditions.

In an attempt to circumvent the lack of knowledge of the distribution of the λ_2/λ_1 gap when we take an embedding factor t such that $t < \|\mathbf{e}\|$, we assume that (for the same probabilistic success of a given basis-reduction algorithm) the same size of gap

is required as in the case where we take $t = \lceil \|\mathbf{e}\| \rceil$ and then derive a modified value for τ. That is, we assume that the basis-reduction algorithm is in some sense oblivious to the embedding factor, with the size of the gap being the 'deciding factor'. While this is a somewhat arbitrary assumption, we believe it to be reasonable and intuitive. We denote the value of τ when $t = \lceil \|\mathbf{e}\| \rceil$ by $\tau_{\|\mathbf{e}\|}$ and the analogous value of τ when $t = 1$ by τ_1. Given a particular value of n and knowing $\tau_{\|\mathbf{e}\|}$, we hence know (approximately) the gap required, denoted by $g_{\|\mathbf{e}\|}$ and hence a corresponding minimum lattice dimension which we denote by $m_{\|\mathbf{e}\|}$. Then, denoting by m_1 the minimum lattice dimension in the case $t = 1$ and assuming that the minimum required gap in the second case, denoted by g_1, is the same, we can write

$$\tau_1 = \min \left\{ \tau_{\|\mathbf{e}\|} \cdot \delta_0^{(m_{\|\mathbf{e}\|} - m_1)}, 1 \right\}.$$

However, for easier and more intuitive comparison, we wish to express τ values for the case $t = 1$ when using the gaps from the $t = \lceil \|\mathbf{e}\| \rceil$ cases. For this comparison, we simply use the λ_2/λ_1 gaps from the case $t = \lceil \|\mathbf{e}\| \rceil$ and plug in the minimum dimension values from the case $t = 1$. We denote these 'illustrative' values of τ by τ'.

4.4 Application to LWE and comparisons

We now examine in more detail the model of [38] when applied to such unique-SVP instances. One difficulty with this model is that, while Gama and Nguyen state that success will occur with 'high probability', this probability is not explained. In the cases examined in this work, it appears to be often impossible to exceed a certain success probability regardless of the lattice λ_2/λ_1 gap (when fixing a particular algorithm and parameterisation). For instance, Figure 4.1 demonstrates success probabilities for LLL for the case of Regev's parameterisation with $n \in \{35, 40, 45\}$ ($t = \|\mathbf{e}\|$) and increasing values of m, with between 50 and 100 cases being run for each value of m.

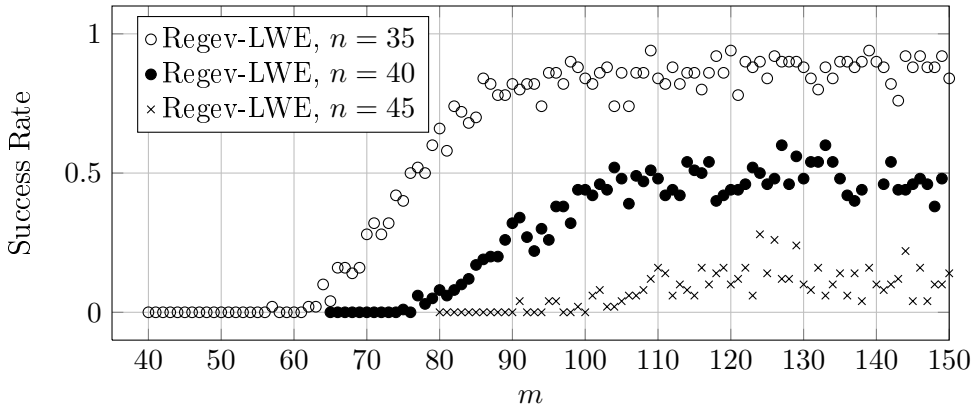

Figure 4.1: Experimental Success Rates, Regev-LWE, LLL, $n \in \{35, 40, 45\}$, $t = \|\mathbf{e}\|$

We treat only the LWE parameterisations proposed by Regev [86] and Lindner & Peikert [61] and view each family of LWE instances as being parameterised by a value of n, from which values of s and q are derived. We then wish to examine the conditions under which applying the embedding approach yields a basis in which the target vector is present (though not necessarily the shortest vector in this reduced basis).

As in [38], our experiments indicate that the target vector lies in the reduced basis with some (fixed) probability whenever the gap is large enough such that

$$\frac{\lambda_2(\Lambda_m)}{\lambda_1(\Lambda_m)} \geq \tau \cdot \delta_0^m$$

where τ is some real constant such that $0 < \tau \leq 1$ depending on the desired probability level, the 'nature' of the lattices considered and the basis-reduction algorithm used. Our experiments proceed by fixing values of n to obtain corresponding LWE parameterisations then generating instances with increasing values of m – using [8] – until finding the minimum such value that recovery of the target vector is possible with the desired probability. We denote such values of m by $m_{\min}(n)$. In the $t = \lceil \|e\| \rceil$ case, plugging this value $m_{\min}(n)$ in $\frac{\lambda_2(\Lambda_m)}{\lambda_1(\Lambda_m)} = \tau \cdot \delta_0^m$ for m where we use Lemma 17 then recovers $\tau_{\|e\|}$. From this value and experimental data for $t = 1$ we

can then derive $\tau_1 = \min\left\{\tau_{\|\mathbf{e}\|} \cdot \delta_0^{(m_{\|\mathbf{e}\|}-m_1)}, 1\right\}$ and τ' by solving $\frac{\lambda_2(\Lambda_m)}{\lambda_1(\Lambda_m)} = \tau' \cdot \delta_0^{m_1}$. Throughout, the experimental data points indicate the minimum lattice dimension for which the lattice basis reduction algorithm succeeds in recovering the target vector with success rate 10%.

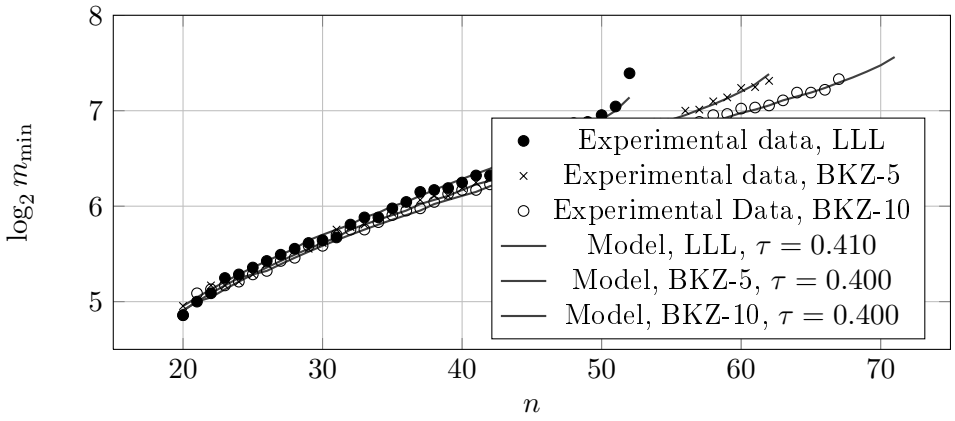

Figure 4.2: Minimum lattice dimension, Regev-LWE, success rate 10%, $t = \|\mathbf{e}\|$.

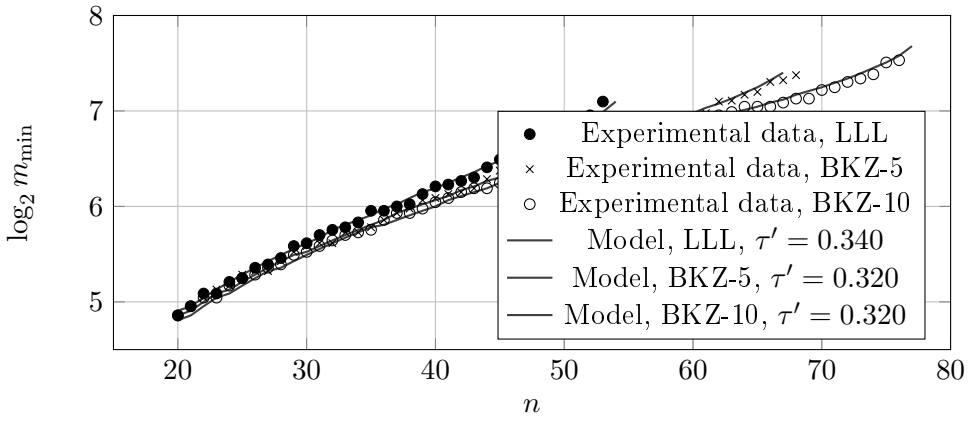

Figure 4.3: Minimum lattice dimension, Regev-LWE, success rate 10%, $t = 1$.

In all experiments carried out, we artificially force that every $\|\mathbf{e}\|$ takes value $\approx \mathrm{E}[\|\mathbf{e}\|]$. This allows us to gain a good estimate of the λ_2/λ_1 gap in the $t = \|\mathbf{e}\|$ case. In addition, for the $m_{\min}$ calculations, we used the experimentally-derived root Hermite factors with linear interpolation.

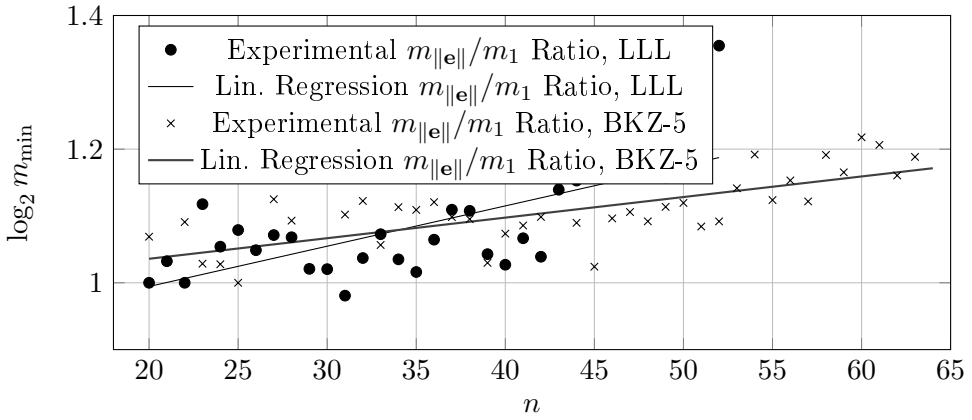

Figure 4.4: $m_{\|\mathbf{e}\|}/m_1$ Ratios, Regev-LWE, LLL and BKZ-5

4.4.1 Regev's Parameters

We firstly examine the case of Regev's original parameters as proposed in [86]. We take $q \approx n^2$ and set $\alpha = 1/(\sqrt{n} \cdot \log_2^2 n)$, $s = \alpha q$. Figure 4.2 illustrates the predicted feasible regions when $t = \lceil \|\mathbf{e}\| \rceil$. Similarly, Figure 4.3 gives analogous plots in the case $t = 1$, using the 'illustrative' values of τ' mentioned in Section 4.3.3. Figure 4.4 gives the $m_{\|\mathbf{e}\|}/m_1$ ratio for LLL and BKZ-5, illustrating the greater efficiency of using $t = 1$.

Based on the results as displayed above, we obtain parameters for embedding factors of $\lceil \|\mathbf{e}\| \rceil$ and 1, given in Table 4.2. We note that, while using an embedding factor $t = 1$ is most efficient, obtaining $\tau_1 > \tau_{\|\mathbf{e}\|}$ possibly seems counter-intuitive. However, the assumption of a fixed gap required for success to occurs (with probability ≈ 0.1) indeed leads to a larger value for τ_1.

4.4.2 Lindner and Peikert's Parameters

For Lindner and Peikert's parameters, as in the Regev-LWE case, we choose a series of values for n and generate parameters accordingly, then apply LLL, BKZ-5 and BKZ-10 to solve such instances as far as is possible. Specifically, Table 4.1 gives

a selection of the parameters considered as produced by the LWE generator. We

n	20	30	40	50	60	70	80
q	2053	2053	2053	2053	2053	2053	2053
s	9.026	8.566	8.225	7.953	7.728	7.536	7.369

Table 4.1: Selected Lindner & Peikert LWE Parameters

proceed similarly to the Regev-LWE case, with minimum lattice dimensions being given in Figure 4.5 for the $t = \|\mathbf{e}\|$ case and in Figure 4.6 for the $t = 1$ case. Table 4.2 also gives the derived values of τ for Lindner and Peikert's parameterisation.

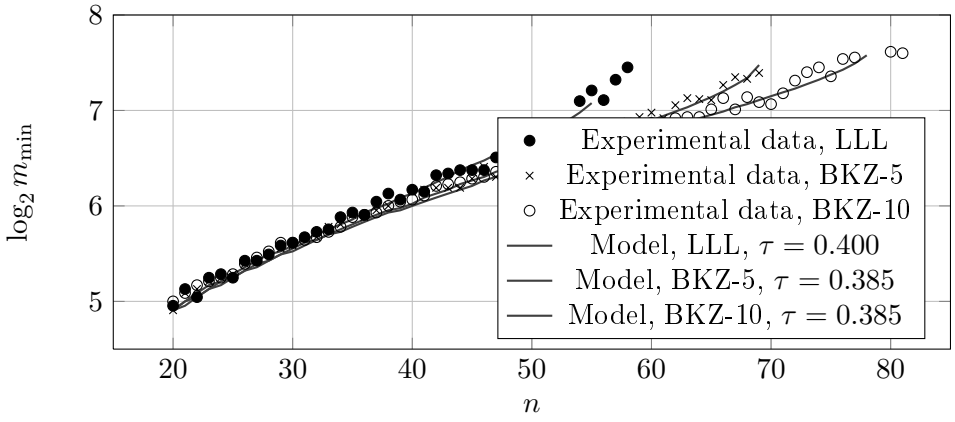

Figure 4.5: Minimum lattice dimension, Lindner & Peikert Parameterisation, success rate 10%, $t = \|\mathbf{e}\|$

Regev				Lindner and Peikert			
	LLL	BKZ-5	BKZ-10		LLL	BKZ-5	BKZ-10
τ $(t = \|\mathbf{e}\|)$	0.410	0.400	0.400	τ $(t = \|\mathbf{e}\|)$	0.400	0.385	0.385
τ $(t = 1)$	0.467	0.464	0.444	τ $(t = 1)$	0.435	0.431	0.439
τ' $(t = 1)$	0.340	0.320	0.320	τ' $(t = 1)$	0.330	0.310	0.310

Table 4.2: Parameters for finding $\mathbf{e}$ with success rate 10%, Regev's and Lindner & Peikert's parameters.

We note that the values of τ derived seem consistent and do not vary widely between parameterisations. Of course, the value of τ may be expected to change when using

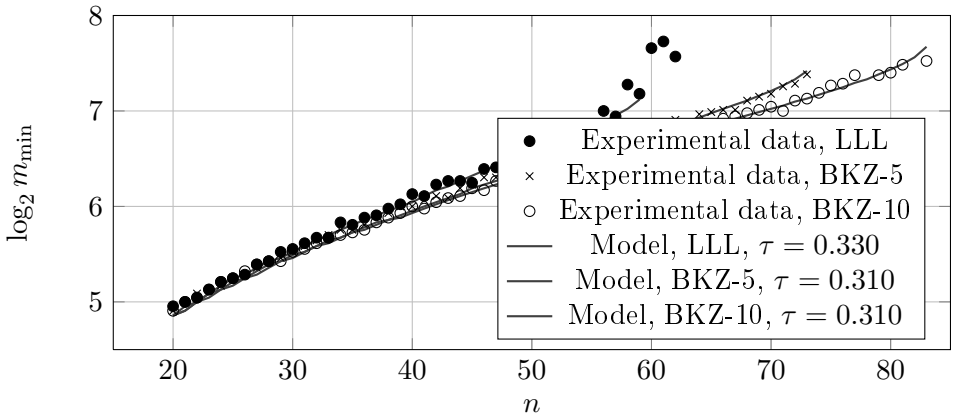

Figure 4.6: Minimum lattice dimension, Lindner & Peikert Parameterisation, success rate 10%, $t = 1$

'stronger' algorithms than BKZ-10 or BKZ-20, however our limited experiments, and the results reported in [38] appear to indicate that the use of 'stronger' basis reduction algorithms leads to modest decreases in the values of τ. Thus, when we project these results in Section 4.5.1, we use the experimentally-derived τ values and thus expect the resulting complexity predictions to be somewhat conservative.

4.5 Limits of the Embedding Approach

Using the above model, we can derive an estimation of the limits of applicability of the embedding approach. Given a values (δ_0, τ), we can define the maximum value of n for which we can recover the target vector using the embedding approach to be

$$n_{\max} := \max \left\{ n \colon \exists m \quad \text{s.t.} \frac{\lambda_2(\Lambda_e(n,m))}{\lambda_1(\Lambda_e(n,m))} = \tau \cdot \delta_0^m \right\}$$

The goal is to determine the values of $n_{\max}$. Lemma 2 shows that we can construct a gap of size (under the assumption that we use basis-reduction algorithms with δ_0 small enough that $q^{1-(n/m)}\sqrt{m/(2\pi e)} < q$)

$$\frac{\lambda_2}{\lambda_1} \approx \frac{q^{1-\frac{n}{m}}\sqrt{\frac{1}{2e}}}{cs}.$$

If we want to solve an LWE instance with secret-dimension n, we have to find m such that

$$\frac{q^{1-\frac{n}{m}}\sqrt{\frac{1}{2e}}}{cs\cdot\tau\cdot\delta_0^m}\geq 1.$$

In order to determine the optimal m, we want to maximize the function

$$f_n(m)=\frac{q^{1-\frac{n}{m}}\sqrt{\frac{1}{2e}}}{c\cdot s\cdot\tau\cdot\delta_0^m}.$$

the first derivative of which is zero only when

$$\frac{n\log q}{m^2}=\log\delta_0,$$

and therefore $m=\sqrt{\frac{n\log q}{\log\delta_0}}$ is the optimal sub-dimension. In other words, we expect the attack to succeed if

$$\frac{q^{\left(1-\frac{n}{\sqrt{\frac{n\log q}{\log\delta_0}}}\right)}\sqrt{\frac{1}{2e}}}{c\cdot s\cdot\tau\cdot\delta_0^{\sqrt{\frac{n\log q}{\log\delta_0}}}}\geq 1$$

Thus, we only need to consider the optimal sub-dimension to ascertain whether we can expect the attack to succeed (with the given probability). Since, in our experiments we force $\|\mathbf{e}\|\approx\mathrm{E}[\|\mathbf{e}\|]$, we increase the value of c to cover all but the upper-tail of the distribution of $\|\mathbf{e}\|$. We can then state the following:

Assumption 6. *Given a fixed LWE parameterisation and a given value of τ (derived as above using $\|\mathbf{e}\|\approx\mathrm{E}[\|\mathbf{e}\|]$ instances and also corresponding to a fixed δ_0) corresponding to a fixed success rate p_s, we can solve general instances from the parameterisation with secret-dimension n with a particular value of m with probability*

$$p_c\geq p_s\cdot\left(1-(c\cdot\exp((1-c^2)/2))^m\right) \tag{4.1}$$

if

$$\frac{q^{\left(1-\frac{n}{\sqrt{\frac{n\log q}{\log\delta_0}}}\right)}\sqrt{\frac{1}{2e}}}{c\cdot s\cdot\tau\cdot\delta_0^{\sqrt{\frac{n\log q}{\log\delta_0}}}}\geq 1 \tag{4.2}$$

We note that this assumption follows immediately from the above discussion and Lemma 16. Thus, given a target success probability, we attempt to satisfy conditions 4.1 and 4.2.

n	64	96	128	160	192	224	256	288	320
δ_0	1.0159	1.0111	1.0085	1.0069	1.0058	1.0050	1.0045	1.0040	1.0036
$\log_2(\text{sec}) = 1.8/\log_2\delta_0 - 110$	negl.	negl.	37.41	71.44	105.74	140.16	167.88	202.54	237.20
$\log_2(\text{sec}) = 0.009/\log_2^2\delta_0 - 27$	negl.	negl.	33.36	64.45	102.29	146.83	187.50	244.37	307.85

Table 4.3: Estimated cost of finding $\mathbf{e}$ with success rate 0.099, Regev's parameters.

4.5.1 Comparisons

We briefly compare the application of BKZ in both the embedding approach and the short dual-lattice vector distinguishing approach. For all embedding approach predictions, we take success probability slightly lower than 0.1, employing Assumption 6 - we choose c such that condition 4.1 holds for $p_c \geq 0.099$. While the dual-lattice distinguishing approach is not the best-known attack, it is easy to analyse in comparison to reduction-then-decode algorithms. We consider the application of BKZ in both situations. In the distinguishing approach, we can choose a desired distinguishing advantage ϵ and set $\gamma = q/s \cdot \sqrt{\ln(1/\epsilon)/\pi}$, from which we can compute a required root Hermite factor of $\delta_0 = 2^{\log_2^2(\gamma)/(4n\log_2 q)}$. So, for instance, with $n = 128$, we require $\delta_0 \approx 1.0077$ to gain a distinguishing advantage of ≈ 0.099, i.e. significantly worse than the 1.0085 required for the embedding attack. In Table 4.4 we give comparable estimated costs for distinguishing between LWE samples and uniformly random samples using the approach of Micciancio and Regev.

However, we note that the expression of Lindner and Peikert for the advantage of the dual-lattice distinguishing approach gives an upper-bound on the advantage obtained through the use of a specific algorithm. While the approximation is close overall, in the high-advantage regime the model is somewhat optimistic in estimating the

n	64	96	128	160	192	224	256	288	320
δ_0	1.0144	1.0099	1.0077	1.0063	1.0053	1.0046	1.0040	1.0036	1.0033
$\log_2(\text{sec}) = 1.8/\log_2\delta_0 - 110$	negl.	negl.	53.15	89.99	126.44	162.56	198.39	234.00	269.38
$\log_2(\text{sec}) = 0.009/\log_2^2\delta_0 - 27$	negl	negl	46.93	84.10	128.29	179.35	237.18	301.70	372.81

Table 4.4: Estimated cost of solving decision-LWE, advantage ~ 0.099, Regev's parameters, dual-lattice distinguisher

advantage obtainable.

More rigorous comparison to the dual-lattice distinguishing attack is difficult, however, since the optimal strategy for said attack is to run a large number of low-advantage attacks and we can only analyse the embedding approach for high-advantages due to the (current) practical component of the analysis.

4.5.2 Closing Remarks

In conclusion, we provide evidence that the model of Gama and Nguyen is applicable to the solution of unique-SVP instances constructed from LWE instances and experimentally derive the constants which embody the performance of the approach. Based on the models used and assumptions made, we show that the embedding approach outperforms the dual-lattice distinguishing approach of Micciancio and Regev (in the high-advantage regime). We view a more in-depth comparison of the efficiency of the embedding technique and enumeration techniques as a pressing research question. The practical behaviour of lattice-reduction algorithms on unique-SVP instances remains mysterious, with (to the best of our knowledge) no recent progress in explaining the phenomena observed.

Chapter 5

Practical Cryptanalysis of a MQPKC after Reduction to LWE

The contents of this chapter are based on the paper "Practical Cryptanalysis of a Public-Key Encryption Scheme Based on New Multivariate Quadratic Assumptions" which was presented at PKC 2014, Buenos Aires, Argentina and was a work conducted in collaboration with M. R. Albrecht, J. C. Faugere, L. Perret, Y. Todo and K. Xagawa.

5.1 Introduction

A member of the family of conjectured quantum-secure alternatives to number-theoretic cryptography, the field of multivariate quadratic ($\mathcal{MQ}$) cryptography is based on the hardness of solving systems of quadratic polynomial equations. This line of research dates back to the mid-eighties with the design of C* [69], later followed by many other proposals, e.g. [89, 59, 30, 81, 55, 102, 103]. While the constructions that have emerged from this field possess remarkable speed and simplicity and are commonly considered to be an interesting alternative to constructions based on number-theoretic problems, the field suffers from a lack of clear security reductions to well-understood problems, leading to a series of attacks, [56, 29, 34, 45, 36, 33, 35, 31].

Until recently, no such schemes with security proofs had been proposed, leading to a history of attacks and improvements, resulting in steadily diminishing confidence in the feasibility of using multivariate cryptography in practise. Recently, the field has seen efforts to develop provably-secure constructions, with the proposal of the QUAD stream cipher [19] being an elegant example.

At PKC 2012 Huang, Liu and Yang (HLY) proposed a new public-key encryption scheme [49] whose security can be provably reduced to the the hardness of solving a system of non-linear equations. The key innovation of HLY [49] is a $\mathcal{MQ}$ scheme in which the public key is noise-free and non-linear but ciphertexts are noisy and linear. Hence, the scheme proposed by HLY can be viewed as a hybrid between the Learning with Errors (LWE) problem [86] and $\mathcal{MQ}$ cryptosystems. The semantic security of the scheme [49] can be provably reduced to the difficulty of solving a system of non-linear equations which is somewhat structured as the coefficients of the non-linear parts of the polynomials are chosen according to a discrete Gaussian distribution. The main assumption of [49] is that this new problem is not easier than the problem of solving a random system of quadratics equations.

5.1.1 Overview of the Results

We describe the HLY proposal in Section 5.2. The new hard problem introduced by HLY is as follows:

Definition 7 ($\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$)**.** *Let n be positive integer, $m = cn$ for some $c \geq 1$, q be a polynomially bounded prime, a constant $\beta, 0 < \beta < q/2$ and $\mathbf{s}$ be a secret vector in $H_\beta := [-\beta, \ldots, \beta]^n \subseteq \mathbb{Z}_q^n$. We denote by $\mathbb{Z}_q^{\Phi_\zeta}[x_1, \ldots, x_n]$ the distribution on quadratic polynomials of $\mathbb{Z}_q[x_1, \ldots, x_n]$ obtained by sampling the monomials of degree 2 according to a discrete Gaussian distribution Φ_ζ of standard deviation $\zeta \in \mathcal{O}(1)$ and centred on zero and by sampling the others coefficients (linear, and constant parts) uniformly at random. $MQ^{(n)}_{\mathbf{s},\Phi}$ is the probability distribution on the $\mathbb{Z}_q[x_1, \ldots, x_n]^m \times$*

$\mathbb{Z}_q^m$ *obtained by sampling* $\mathbf{p} = (p_1, \ldots, p_m)$ *from* $\mathbb{Z}_q^{\Phi_\alpha}[\mathbf{x}]^m$, *and returning* $(\mathbf{p}, \mathbf{c}) = \big(\mathbf{p}, \mathbf{p}(\mathbf{s})\big) \in \mathbb{Z}_q[x_1, \ldots, x_n]^m \times \mathbb{Z}_q^m$. $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ *is the problem of finding* $\mathbf{s} \in H_\beta^n$ *given a pair* $\big(\mathbf{p}, \mathbf{p}(\mathbf{s})\big) \leftarrow_\$ MQ_{\mathbf{s},\Phi}^{(n)}$.

The main assumption made in [49] is that $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ is not easier than the problem of solving a random system of quadratic equations (Assumption 7). Remark that the latter problem is notoriously known as a hard problem from a theoretical [40] and practical point of view [15, 16, 17]. In this chapter, we show that $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ is in fact related to a much easier problem. The starting point of our analysis is to simply remark that $\mathrm{MQ}(n, m\Phi_\zeta, H_\beta)$ closely resembles a LWE problem with a discrete Gaussian with variance $\gamma^2 = \mathcal{O}\left(n^2\beta^2\zeta^2\right)$ (centred at zero).

We use this fact, together with the Micciancio-Regev distinguisher and the lattice-reduction complexity model of Lindner and Peikert to derive a new necessary conditions on the security of the HLY scheme (Section 5.3). In particular, such scheme has at most τ-bit security – with regard to constructing a distinguisher of advantage d – if $(n, \beta, c, k, \tau, d)$ verifies

$$\exp\left(-\frac{\pi^2}{8} n^{-4} \cdot \frac{(ck)^{-2}}{6\beta^2} \cdot \left(2^{1.8/(\tau+78.9)}\right)^{2cn}\right) = d$$

For example, with $\beta = c = 2$, $k = 12$, $d = 0.5$, setting $n = 1190$ satisfies this condition for $\tau = 80$. With $n = 1190$, however, the public-key is of size ≈ 1.17 GB.

It appears then that all parameters suggested in [49] (reproduced in Table 5.1) are too small to verify our new security condition. Indeed, we have been able to mount several practical attacks: distinguishing attack with Micciancio-Regev, and a key-recovery attack with the embedding technique, and an improved key-recovery attack exploiting the presence of a small secret (Section 5.4). We successfully run the two first attacks in roughly one day for the first challenge (i.e. Case 1)) and in roughly three days for the second challenge (i.e. Case 2) proposed by the authors [49]. The last practical attack is even more efficient. For the first challenge, we recovered the

secret-key in less than 5 minutes and less than 30 minutes for the second challenge. The experimental results are detailed in Section 5.5.

5.2 A New Multivariate Quadratic Assumption and LWE with Small Secrets

In [49] the authors introduced a variant of the classical Polynomial System Solving Problem (PoSSo).

Definition 8. *Let* $f_0, \ldots, f_{m-1} \in \mathbb{Z}_q[x_0, \ldots, x_{n-1}]$ *be non-linear polynomials.* PoSSo *is the problem of finding – if any –* $\mathbf{s} \in \overline{\mathbb{Z}_q}^n$ *such that* $f_0(\mathbf{s}) = 0, \ldots, f_{m-1}(\mathbf{s}) = 0$.

It is well known [40] that this problem is NP-hard. Note that PoSSo remains NP-hard [40] even if we suppose that the input polynomials are quadratics. In this case, PoSSo is also called MQ. HLY proposed a variant of MQ where the monomials of highest degree (i.e. 2) in the system have their coefficients chosen according to a discrete Gaussian distribution of standard deviation $\zeta \in \mathcal{O}(1)$ and centered on zero. Following [49], we denote this distribution by Φ_ζ.[1] The remaining coefficients (linear, and constant parts) are chosen uniformly at random. We denote this distribution on $\mathbb{Z}_q[x_1, \ldots, x_n]$ by $\mathbb{Z}_q^{\Phi_\alpha}[\mathbf{x}]$. As mentioned in [49], $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ is rather close to LWE. In fact, each $\big(\mathbf{p}, \mathbf{p}(\mathbf{s})\big) \leftarrow_\$ MQ^{(n)}_{\mathbf{s},\Phi}$ can be mapped to a LWE instance. To do so, we just consider the matrix $A_\mathbf{p} \in \mathbb{Z}_q^{n \times m}$ corresponding to the linear part of $\mathbf{p}$. We then remark that each component of $\mathbf{p}(\mathbf{s}) - \mathbf{s} \cdot A_\mathbf{p} - \mathbf{p}(\mathbf{0})$ is the sum of $\frac{n(n+1)}{2}$ discrete Gaussians each having variance $\left(\frac{(2\beta+1)^2-1}{12}\right)^2 \cdot \zeta^2$. From now, we assume that this sum is a discrete Gaussian of variance $\gamma^2 = \frac{n(n+1)}{2} \cdot \left(\frac{(2\beta+1)^2-1}{12}\right)^2 \cdot \zeta^2$ (centred at zero).

It is proven in [49] that $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ has decision to search equivalence. Such equivalence makes the problem appealing to design an encryption scheme. The

[1] The parameter ζ is called α in [49] but this notation clashes with the standard notation for LWE.

public-key of the scheme proposed in [49] is a pair of the form $\big(\mathbf{p}, \mathbf{p}(\mathbf{s})\big) = (\mathbf{p}, \mathbf{c}) \in \mathbb{Z}_q^{\Phi_\alpha}[\mathbf{x}]^m \times \mathbb{Z}_q^m$. To encrypt a bit b, we choose $\mathbf{r} \in H_{n^\lambda} := [-n^\lambda, \ldots, n^\lambda]^m \subset \mathbb{Z}_q^m$ with λ being a new parameter. We then compute : $c = \big(A_{\mathbf{p}} \cdot \mathbf{r}^T, \langle\, \mathbf{r}, \mathbf{c} - p(\mathbf{0}) \,\rangle + b \cdot \lfloor q/2 \rceil\big)$. Thus, each encryption of zero produces a LWE sample whose error has variance: $m \cdot n^{2\lambda} \cdot \gamma^2$. As a consequence, we expect the noise to have size $\sqrt{\frac{2}{\pi}} \cdot \sqrt{m} \cdot n^\lambda \cdot \gamma$. Note that [49] also proposed a Key Encapsulation Mechanism (KEM) scheme, based on the same new hard problem, but which we do not discuss here.

Regarding the security, [49] showed that breaking the semantic security of the encryption scheme is equivalent to solving $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$. More precisely:

Theorem 5 ([49]). *Let $\mathcal{A}$ be an adversary breaking the semantic security of the scheme working in time T with advantage ϵ. Then, there exists a probabilistic algorithm $\mathcal{B}$ solving $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ in time at most $T \cdot \frac{128}{\epsilon^2} \cdot (2\beta + 1) \cdot (n^2 \log q)^2$, with success probability at least $\epsilon/(4\,q)$.*

A similar result holds for the KEM scheme, i.e. breaking the semantic security of the KEM scheme allows to solve $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$. Such a reduction is then used to establish concrete parameters for the proposed encryption scheme. The basic hypothesis for setting the parameter is to assume that solving $\mathbf{p} - \mathbf{p}(\mathbf{s}) = \mathbf{0}$, for $\big(\mathbf{p}, \mathbf{p}(\mathbf{s})\big) \leftarrow_\$ MQ^{(n)}_{\mathbf{s},\Phi}$, is essentially not easier than solving a random system of equations [49].

Assumption 7 (HLY Hardness Hypothesis). *Solving $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ is as hard as solving a random system of m quadratic equations in n variables modulo q with a pre-assigned solution in H_β^n.*

The fact that the secret is in H_β^n implies that one can always add n equations of degree $2\,\beta + 1$ of the form $\prod_{j \in H_\beta}(x_i - j)$. Clearly, the evaluation of such equations on any $\mathbf{s} \in H_\beta^n$ will be zero.

Arguably, this connection between the semantic security and hardness of PoSSo

is the main difference between the HLY scheme and the classical encryption scheme based on LWE. Indeed, the HLY scheme is very similar to a textbook LWE encryption scheme equipped with a Gaussian of standard deviation $\sqrt{m} \cdot n^{\lambda} \cdot \gamma$ with a very small secret. A noteworthy difference lies in the fact that we also consider small (i.e. of norm bounded by n^{λ}) linear combinations of public samples. In the classical LWE encryption scheme due to Regev [86], we consider only linear combinations with coefficients in $\{-1, 0, 1\}$ of the public samples. However, [73] also suggest to sample $\mathbf{r}$ from $\{-r, -r+1, \ldots, r\}^m$.

Assumption 7 allows to estimate the cost of the best attack against $\mathrm{MQ}(n, m, \Phi_{\zeta}, H_{\beta})$. A well-established approach to solve PoSSo is to compute a Gröbner basis [18, 24, 25]. The cost of solving a (zero-dimensional, i.e. finite number of solutions) system of m non-linear equations in n variables with the F_5 algorithm [15, 32] is $\mathcal{O}\left(\binom{n+D_{reg}}{D_{reg}}^{\omega}\right)$, where D_{reg} is the maximum degree reached during the Gröbner basis computation, and ω is the matrix multiplication exponent (or the linear-algebra constant) as defined in [100, Chapter 12]. We recall [101, 99] that $\omega \in [2, 2.3727]$.

In general, it is a hard problem to predict *a priori* the degree of regularity of a given system of equations. However, Assumption 7 implies that the system of non-linear equations involved is no easier to solve than semi-regular equations [15, 16, 17]. Precisely, D_{reg} is bounded from below by the index of the first non-positive coefficient of: $\sum_{k \geq 0} c_k z^k = \frac{(1-z^2)^m (1-z^{(2\beta+1)})^n}{(1-z)^n}$. This is the degree of regularity of a system of m equations of degree 2 plus n equations of degree $2\beta+1$ in n variables.[2] From now on, we will denote by $T_{\mathrm{ref}}(m, n, q)$ the cost of solving such system with F_5 algorithm, and by ϵ_{ref} the success probability. Usually, a Gröbner basis computation always succeeds, but one can relax this condition by randomly fixing variables. Precisely, a

[2] Note that this quantity can be explicitly computed for any value of n, m and β.

success probability ϵ_{ref} allows to fix

$$r_{\text{ref}} = \left\lceil \log_{2\,\beta+1}\left(\frac{1}{\epsilon_{\text{ref}}}\right)\right\rceil$$

variables for systems sampled according to $MQ^{(n)}_{\mathbf{s},\Phi}$.

It is worth mentioning and commending that [49] propose concrete parameters for their scheme (reproduced in Table 5.1). The parameters are chosen as follows. Assume there exists an adversary $\mathcal{A}$ breaking the semantic security of the HLY encryption in time $T_{\text{dist}} = 2^{\ell}$ with advantage $\epsilon_{\text{dist}} = 2^{-s}$. According to Theorem 5, we can construct an algorithm $\mathcal{B}$ solving $\text{MQ}(n, m, \Phi_{\zeta}, H_{\beta})$ in time $T_{\text{search}}(T_{\text{dist}}, \epsilon_{\text{dist}}, n, q)$ with success probability $\epsilon_{\text{search}}(\epsilon_{\text{dist}}, q)$. From Assumption 7, the best algorithm for solving $\text{MQ}(n, m, \Phi_{\zeta}, H_{\beta})$ works in time $T_{\text{ref}}(m, n - r_{\text{ref}}, q)$ with a success probability ϵ_{ref}. The parameters m, n, q are chosen such that

$$T_{\text{search}}(T_{\text{dist}}, \epsilon_{\text{dist}}, n, q) < T_{\text{ref}}(m, n, q) \text{ and } \epsilon_{\text{search}}(\epsilon_{\text{dist}}, q) < \epsilon_{\text{ref}}.$$

Under the HLY hypothesis (Assumption 7), this means that no adversary can break the semantic security of the scheme in time less than 2^{ℓ} with success probability better than 2^{-s}.

5.3 Analysis of the Parameters

In this section, we show that security and efficiency are essentially incompatible for HLY. To do so, we derive a set of conditions on the parameters that would thwart the simplest known attack against LWE-style systems such as those discussed above. That is, we want to find parameters such that both computing a Gröbner basis and lattice attacks (in particular the non-optimal Micciancio-Regev approach) are exponentially hard in the security parameter τ. Below, we recall the constraints on the parameters from [49]:

1. $k \cdot \zeta \cdot n^{2+\lambda} \cdot m \cdot \beta^2 \leq q/4$ (to allow for correct decryption)

2. $m \cdot \log(2n^\lambda + 1) \geq (n+1)\log q + 2k$ (to make sure the subset sum problem is hard)

3. n, m, q, ζ, β (to satisfy the condition in the MQ assumption such that $MQ(n, m, q, \Psi_\zeta, H_\beta)$ is hard to solve).

For the number of equations, we may restrict $m = c \cdot n$ where c is a constant (we remark that the challenges proposed in [49] have $c = 2$). In this case, we can assume that MQ is hard (that is, the cost of computing a Gröbner basis is exponential in the number of variables [15, 16, 17]). From Condition 2, we then get :

$$\begin{aligned} m \cdot \log(2n^\lambda + 1) &\geq (n+1)\log q + 2k \geq n \log q, \\ c \cdot n \cdot \log(2n^\lambda + 1) &\geq n \log q, \\ c \cdot \log(2n^\lambda + 1) &\geq \log q. \end{aligned}$$

This means that $2n^\lambda$ should be roughly (or at least) $q^{1/c}$. Hence, the first condition yields:

$$\begin{aligned} k \cdot \zeta \cdot n^{2+\lambda} \cdot m \cdot \beta^2 &\leq q/4 \\ k \cdot \zeta \cdot n^{2+\lambda} \cdot c \cdot n \cdot \beta^2 &\leq 2^{(c-2)} n^{c\lambda} \\ \zeta \cdot n^2 \cdot \beta^2 &\leq (ck)^{-1} 2^{(c-2)} n^{(c-1)\lambda - 1} \end{aligned}$$

as a bound on the noise in each of the m samples. As mentioned in the introduction to this thesis, (heuristically) lattice reduction will produce vectors of length

$$v = q^{n/m} \cdot \delta_0^m \quad = \quad q^{1/c} \cdot \delta_0^{cn} \leq 2n^\lambda \cdot \delta_0^{cn}.$$

By combining this with the above, we get a distinguishing advantage $\big($as defined in Equation 3.4$\big)$ of

$$\begin{aligned} \exp\left(-\frac{\pi s^2 v^2}{q^2}\right) &= \exp\left(-\frac{\pi s^2 4n^{2\lambda}\delta_0^{2cn}}{q^2}\right) = \exp\left(-\frac{2\pi^2\sigma^2 4n^{2\lambda}\delta_0^{2cn}}{q^2}\right) = \exp\left(-\frac{2\pi^2\sigma^2 4n^{2\lambda}\delta_0^{2cn}}{4^c n^{2c\lambda}}\right), \\ &= \exp\left(-(4^{(1-c+\frac{1}{2})}\pi^2\sigma^2 n^{2\lambda(1-c)}\delta_0^{2cn})\right). \end{aligned}$$

Now, we can write:

$$\sigma^2 = \quad \zeta^2 \cdot \frac{n(n+1)}{2} \cdot \left(\frac{(2\beta+1)^2 - 1}{12}\right)^2$$

which, after some manipulation, gives:

$$\sigma^2 \lessapprox \frac{4^{c-2} \cdot (ck)^{-2} \cdot n^{2\lambda c - 2\lambda - 4} \cdot (16\beta^2 + 32\beta + 16)}{288 \cdot \beta^2}$$

Hence we can lower-bound the distinguishing advantage by:

$$\exp\left(-(4^{\frac{3}{2}-c}\pi^2\sigma^2 n^{2\lambda(1-c)}\delta_0^{2cn})\right) \approx \exp\left(-\frac{\pi^2}{8}n^{-4}\delta_0^{2cn} \cdot \frac{(ck)^{-2}}{6\beta^2}\right)$$

We now introduce a parameter τ, representing the bit-complexity of solving such instances using the model of Lindner and Peikert. We then replace δ_0 by $2^{(1.8/(\tau+78.9))}$ (employing the estimates of Lindner & Peikert for the running time of BKZ 2.0 to deliver an estimate of the number of bit operations required to obtain such a root Hermite factor) and require that the advantage is constant in terms of τ. In other words

$$\exp\left(-\frac{\pi^2}{8}n^{-4} \cdot \frac{(ck)^{-2}}{6\beta^2} \cdot \left(2^{1.8/(\tau+78.9)}\right)^{2cn}\right) = d. \tag{5.1}$$

For example, for $\tau = 80$, with $\beta = 2$, $c = 2$, $k = 12$ and $d = 0.5$, setting $n = 1190$ satisfies this condition. For $\tau = 128$, the same parameters require $n = 1600$. We note, however, that setting $n = 1190$ already results in a public key of considerable size (optimistically setting $\zeta = 10$):

$$\frac{m \cdot \binom{n+2}{2} \cdot \log_2(2\pi\zeta)}{8 \cdot 1024^3} \approx 1.17 \text{ GB}, \tag{5.2}$$

while setting $n = 1600$ results in a public-key of size 2.85 GB.

Furthermore, we stress that these parameters do not take other potential attack methods into account and should be viewed as a somewhat loose *upper-bound* on the complexity of solving such instances. In particular, this discussion does not reflect the possibility of exploiting the small secret for example through modulus reduction and the approach discussed next.

5.4 Improved Embedding Attack

We present an improved version of the embedding attack described in Chapter 4. To do so, we exploit the fact that the secret key $\mathbf{s}$ is *extremely short*. Recall that the

coefficients of the secret lie in a small subset $H = [-\beta, \beta] \subset \mathbb{Z}_q$. Typically, Huang, Liu and Yang suggested to take $\beta = 2$ (Table 5.1).

Let $\big(\mathbf{p}, \mathbf{p}(\mathbf{s})\big) = (\mathbf{p}, \mathbf{c}) \leftarrow_{\$} MQ^{(n)}_{\mathbf{s},\Phi}$ be a public-key of HLY scheme. Let $A_{\mathbf{p}} \in \mathbb{Z}_q^{n \times m}$ be the matrix corresponding to the linear part of $\mathbf{p}$. According to the fact that we can map an $MQ^{(n)}_{\mathbf{s},\Phi}$ instance to and LWE instance, we can write:

$$\mathbf{c} \equiv \mathbf{s} \cdot A_{\mathbf{p}} + \mathbf{e} + \mathbf{p}(\mathbf{0}) \bmod q,$$

where $\mathbf{e} \equiv \mathbf{p}(\mathbf{s}) - \mathbf{s} \cdot A_{\mathbf{p}} - \mathbf{p}(\mathbf{0}) \bmod q$. Notice that each coefficient of $\mathbf{e}$ is the sum of $n(n+1)/2$ discrete Gaussians. From now on, we let $\mathbf{y} \equiv \mathbf{c} - \mathbf{p}(\mathbf{0}) \equiv \mathbf{s} \cdot A_{\mathbf{p}} + \mathbf{e} \bmod q$ to ignore the constant part.

The basic idea is to consider the lattice defined by the following basis $\mathbf{B}$:

$$\mathbf{B} = \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & \mathbf{0} \\ A_{\mathbf{p}} & \mathbf{I}_n & \mathbf{0} \\ -\mathbf{y} & \mathbf{0} & 1 \end{pmatrix}.$$

Since $\mathbf{y} \equiv \mathbf{s}A_{\mathbf{p}} + \mathbf{e} \pmod q$, there exists $\mathbf{k} \in \mathbb{Z}^m$ satisfying $\mathbf{y} = \mathbf{s} \cdot A_{\mathbf{p}} + \mathbf{e} + q\mathbf{k} \in \mathbb{Z}^m$. Notice that, using this $\mathbf{k}$, the lattice $\mathcal{L}(\mathbf{B})$ contains a short vector $\mathbf{w} = [-\mathbf{e} \mid \mathbf{s} \mid 1] \in \mathbb{Z}^{m+n+1}$:

$$[\mathbf{k} \mid \mathbf{s} \mid 1] \cdot \begin{pmatrix} q\mathbf{I}_m & \mathbf{0} & \mathbf{0} \\ A_{\mathbf{p}} & \mathbf{I}_n & \mathbf{0} \\ -\mathbf{y} & \mathbf{0} & 1 \end{pmatrix} = [-\mathbf{e} \mid \mathbf{s} \mid 1].$$

Applying the reduction algorithm to the lattice $\mathcal{L}(\mathbf{B})$ is less efficient than the basic embedding attack. The dimension $m + n + 1$ is larger than $m + 1$ and the short vector $\mathbf{w} = [-\mathbf{e} \mid \mathbf{s} \mid 1] \in \mathbb{Z}^{m+n+1}$ contains $\mathbf{e}$ entirely.

However, we can consider *a truncated lattice* defined by an $(m' + n + 1)$-dimension

right-bottom submatrix $\mathbf{B}'$ of $\mathbf{B}$. By this truncation, we have the following relations:

$$[\mathbf{k}' \mid \mathbf{s} \mid 1] \cdot \begin{pmatrix} q\mathbf{I}_{m'} & \mathbf{0} & \mathbf{0} \\ A'_{\mathbf{p}} & \mathbf{I}_n & \mathbf{0} \\ -\mathbf{y}' & \mathbf{0} & 1 \end{pmatrix} = [-\mathbf{e}' \mid \mathbf{s} \mid 1] \in \mathbb{Z}^{m'+n+1}.$$

We note that $\mathbf{w}' = [-\mathbf{e}' \mid \mathbf{s} \mid 1]$ should be shorter than the previous $\mathbf{w}$. Hence, we could expect a 'less powerful' basis reduction algorithm to be required for recovery of $\pm\mathbf{w}'$ as compared to one required for recovery of the previous $\mathbf{w}$. We finally note that, if $m' < m - n$, then the dimension is smaller than that of the lattice in the direct approach.

5.4.1 Estimation of the Expected Gap

For an N dimensional lattice $\mathcal{L}$, we have

$$\lambda_1(\mathcal{L}) \approx \sqrt{N/2\pi e} \cdot \mathrm{vol}(\mathcal{L})^{1/N}$$

according to the Gaussian heuristic. In our case, assuming this heuristic holds, we have

$$\lambda_1(\mathcal{L}(\mathbf{B}')) \approx \sqrt{(n + m' + 1)/2\pi e} \cdot q^{m'/(n+m'+1)}.$$

Next, we estimate $\|\mathbf{w}'\| = \sqrt{\|\mathbf{s}\|^2 + \|\mathbf{e}'\|^2 + 1}$. Since the components of $\mathbf{s}$ are chosen from $\{-\beta, \ldots, \beta\}$ uniformly at random, the expected value of $\|\mathbf{s}\|^2$ is $n \cdot \frac{1}{2\beta+1}\sum_{i=-\beta}^{\beta} i^2 = n\beta(\beta+1)/3$. As mentioned above, each coefficient of $\mathbf{e}'$ follows a discrete Gaussian of standard deviation γ. Hence, $\mathrm{E}[\|\mathbf{e}'\|]$ can be estimated as $\sqrt{m'} \cdot \gamma$. Summarizing the above, we obtain (using $\beta = 2$):

$$\mathrm{E}[\|\mathbf{w}'\|] \approx \sqrt{m'\gamma^2 + 2n + 1} \approx \sqrt{m' \cdot \zeta^2 \cdot n \cdot (n+1) + 2n}.$$

Hence, the expected gap is expected to be

$$\frac{\lambda_1(\mathcal{L}(\mathbf{B}'))}{\|\mathbf{w}'\|} \approx \sqrt{\frac{n + m' + 1}{2\pi e n(m'\zeta^2(n+1) + 2)}} \cdot q^{m'/(n+m'+1)}. \tag{5.3}$$

Case	n	m	ζ	β	q	Hardness (T, μ)
1	200	400	10	2	$\approx 2^{74}$	$(2^{156}, 2^{-100})$
2	256	512	10	2	$\approx 2^{76}$	$(2^{205}, 2^{-104})$

Table 5.1: Suggested parameters in [49].

We finally note that, when comparing the efficacy of embedding attacks, the expected gaps should be compared with those of lattices of *the same dimension*. If the dimensions differ, we derive less information regarding the success of the lattice-reduction algorithm in finding the shortest vector.

5.5 Practical Attacks against HLY Challenges

From the discussion in Section 5.3, we expect that all parameters suggested in [49] should be weak against a lattice-reduction attack. The number of variables being considered is much smaller that what is required by condition (5.1). The goal of this part is to provide experimental results to confirm the analysis. To mount the attack, we also make use of the fact that we can look at the hard problem from [49] as an LWE instance and then solve these instances using lattice reduction. In particular, we consider all the parameter sets proposed in [49] (Table 5.1).

The column "Hardness" (T, μ) is a strict lower bound [49] on the complexity of solving $\mathrm{MQ}(n, m, \Phi_\zeta, H_\beta)$ under Assumption 7. The parameters of Case (1) are chosen such that no adversary running in time less than 2^{82} can break the semantic security of the HLY bit-encryption scheme with advantage better than 2^{-11}. For the KEM, Case (1) provides a security of $(2^{85}, 2^{-10})$ (which denotes (time, advantage). Case (2) was expected to provide a security level of $(2^{130}, 2^{-11})$ for the bit encryption scheme (and a security level of $(2^{130}, 2^{-10})$ for the KEM scheme).

Case (1)

Distinguishing. We have $m = 400$ equations in $n = 200$ unknowns. Coefficients for quadratic terms are chosen from a discrete Gaussian with standard deviation $\zeta = 10$ and the secret is in $[-\beta, \ldots, \beta]$ for $\beta = 2$. If we ignore all quadratic terms and only consider the linear part, we have an LWE-style instance with $m = 400, n = 200, q = 18031317546972632788519$ and standard deviation

$$\gamma = \sqrt{\frac{n \cdot (n+1)}{2} \cdot \zeta^2 \cdot \left(\frac{(2\beta+1)^2 - 1}{12}\right)^2} = \sqrt{\frac{200 \cdot 201}{2} \cdot 10^2 \cdot \left(\frac{5^2 - 1}{12}\right)^2} \approx 2^{11.47}.$$

In this instance, the optimal sub-lattice dimension for applying LLL is $\sqrt{n \log(q)/\log(1.0219)} \approx 688$. However, applying LLL in dimension 400 is expected to return a vector of norm $v = q^{n/m} \cdot \delta_0^m \approx 2^{49.47}$ which is more than sufficient to distinguish between such LWE samples and random with advantage $\epsilon = \exp\left(-\frac{\pi s^2 v^2}{q^2}\right) \approx 0.9999$. We ran the LLL algorithm as implemented in fpLLL [78, 95] on lattice instances as in Case (1), i.e., with $m = 400, n = 200, q = 18031317546972632788519$. More precisely, we ran LLL (using Sage's default parameters [97]) on the 400×400 dual lattice. The shortest vector recovered by LLL had norm $2^{49.76}$ while we predicted a norm of $2^{49.47}$. The entire computation took 26 hours on a single core.

Modulus Reduction. A slightly more efficient variant is to perform modulus reduction before performing LLL in order to keep coefficients small. We may apply the modulus reduction technique with the above parameters and pick $p \approx 2^{65.00}$ and $\gamma \approx 2^{3.59}$. Applying LLL in dimension 400 is expected to return a vector of norm $v = 2^{45.00}$ which translates into a distinguishing advantage of $\epsilon \approx 1$.

Embedding. We may also consider the embedding attack. We apply LLL to the 401×401 extended primal lattice and using a (conservative) embedding factor $\lceil \sqrt{m} \cdot \sigma \rceil$. The λ_2/λ_1 gap in this case is approximately

$$\frac{\mathrm{vol}(\mathcal{L}(\mathbf{B}))^{1/m} \cdot \Gamma(1 + m/2)^{1/m}}{\sqrt{2\pi m}\sigma} \approx \frac{q^{\frac{m-n}{m}} \sqrt{\frac{m}{2\pi e}}}{\sqrt{2m}\sigma} \approx 2^{22.94}.$$

The attack recovered the 'noise' from the public key, allowing the private key (or an equivalent) to be recovered by simple linear algebra. We note that this attack obviates the need for a separate search-to-distinguishing phase, as required in the dual-lattice method, the attack taking again ∼26 hours using a 1.4GHz Intel Core2Duo CPU.

Improved Embedding. We set $m' = 66 \approx 200/3$. In this dimension, the running times varied from 268.69 to 295.34 seconds on a Core i7 CPU using the NTL library [93] with GMP [42]. We note that the expected gap (5.3) in this case is $\approx 2^{6.267}$. In each case, we ran the BKZ algorithm (G_BKZ_FP with $\delta = 0.99$, block size = 30, and prune = 10) on the 267-dimensional lattices constructed from the public-keys. We computed m' by incrementing m' from 1 until we were successfully able to recover a shortest vector.

Case (2)

Distinguishing. We have $m = 512$ equations in $n = 256$ unknowns modulo $q \approx 2^{75.47}$. Coefficients for quadratic terms are chosen from a discrete Gaussian with standard deviation $\zeta = 10$ and the secret is in $[-2, \ldots, 2]$ for $\beta = 2$. This gives a standard deviation $\gamma = \sqrt{\frac{256 \cdot 257}{2} \cdot 10^2 \cdot \left(\frac{5^2-1}{12}\right)^2} \approx 2^{11.82}$. Applying LLL in dimension 512 is expected to return a vector of norm $v = q^{n/m} \cdot \delta_0^m \approx 2^{53.74}$ which is more than sufficient to distinguish between such LWE samples and random with advantage $\epsilon = \exp\left(-\frac{\pi s^2 v^2}{q^2}\right) \approx 1$.

Modulus Reduction. Using modulus reduction, we pick $p \approx 2^{66.36}$ and $\gamma \approx 2^{3.76}$. Applying LLL in dimension 512 is expected to return a vector of norm $v = 2^{16.00}$ which translates into a distinguishing advantage of $\epsilon \approx 1$.

Improved Embedding. We set $m' = 90$ as a slightly larger integer than a third of n. Again, we ran the improved embedding attack ten times and successfully

recovered the secret keys from all ten public-keys. The running times varied from 898.14 to 1119.53 seconds. We note that the expected gap is $\approx 2^{7.176}$.

Beyond The Challenges.

To examine how the improved embedding attack scales, we consider larger parameters than those provided by the two challenges of Huang, Liu and Yang. In order to extend these challenges, we fix $\zeta = 10$, $\beta = 2$, $m = 2n$, $k = 12$, and $\lambda = 5$ and calculate q. From the correctness condition in [49] (see also Section 5.3), we should set $q \geq \text{NextPrime}(4k\zeta\beta^2 mn^{2+\lambda}) = \text{NextPrime}(3840n^8)$. From the provable security side, in order to employ the leftover hash lemma, HLY [49] require q to satisfy $m \cdot \log(2n^\lambda + 1) \geq (n+1)\log q + 2k$. We here take q as small as possible, that is, we take $q = \text{NextPrime}(3840n^8)$, which always satisfies the correctness constraint and the security constraint.

Employing an Core i7 CPU (3.4GHz), we ran the LLL algorithm on lattices constructed from the public keys with the parameter n increasing from 100 in increments of 25. We computed the necessary m' for each n by increasing m' from 30 by increments of 10 until we were able to successfully recover a shortest vector in such a case. Table 5.3 summarises the results of experiments using G_LLL_FP, G_LLL_QP, and G_LLL_RR (with precision 150), respectively. Due to precision issues, G_LLL_FP fails at $n = 300$ while G_LLL_QP fails at $n = 450$. Due to time constraints, we only ran G_LLL_RR for n up to 325. Figure 5.1 shows the logarithm of T_{FP}, T_{QP}, and T_{RR}, the running times (in seconds) for these algorithms. We can approximate these times by $\log_2(T_{\text{FP}}) = 6.9675\log(n) - 27.238$, $\log_2(T_{\text{QP}}) = 7.3037\log(n) - 27.208$, and $\log_2(T_{\text{RR}}) = 6.4345\log(n) - 18.502$, By using $T_{\text{cycle}} = T \cdot 3.4 \cdot 10^9$, we obtain bit-operation complexity estimates $\log_2(T_{\text{FP,cycle}}) = 6.9675\log(n) + 4.425$, $\log_2(T_{\text{QP,cycle}}) = 7.3037\log(n) + 4.459$, and $\log_2(T_{\text{RR,cycle}}) = 6.4345\log(n) + 13.161$.

If we wished to extend our experiments using G_LLL_RR for $n = 450$, our model indicates that around $2^{20.808}$ seconds $\approx$ 21 days would be required. However, we

n	q	$\lvert pk\rvert$ (MB)
100	$3840000000000000001 \approx 2^{65.058}$	0.875
125	$228881835937500000029 \approx 2^{67.533}$	1.655
150	$984150000000000000073 \approx 2^{69.737}$	2.794
175	$3377813085937500000071 \approx 2^{71.517}$	4.361
200	$9830400000000000000007 \approx 2^{73.057}$	6.423
225	$25222688085937500000079 \approx 2^{74.417}$	9.046
250	$58593750000000000000013 \approx 2^{75.633}$	12.300
275	$12560090683593750000067 \approx 2^{76.733}$	16.247
300	$25194240000000000000043 \approx 2^{77.737}$	20.960
325	$47796721933593750000059 \approx 2^{78.661}$	26.501
350	$86472015000000000000017 \approx 2^{79.516}$	32.940
375	$15016937255859375000000047 \approx 2^{80.313}$	40.343
400	$25165824000000000000000041 \approx 2^{81.057}$	48.778
425	$40873578755859375000000001 \approx 2^{81.757}$	58.311
450	$64570081500000000000000029 \approx 2^{82.417}$	69.009

Table 5.2: Extended parameters: We fix $m = 2n$, $k = 12$, $\zeta = 10$, $\beta = 10$, and $\lambda = 5$. We calculate q by $\text{NextPrime}(4k\zeta\beta^2 mn^{2+\lambda}) = \text{NextPrime}(3840n^8)$ to satisfy the correctness condition.

expect that application of LLL would prove insufficient to recover the private key (with probability ~ 1) in this manner for values of n greater than ~ 500. For such values of n, lattice reduction algorithms achieving lower root Hermite factors will be required [3] - we expect this to be the case due to observations made in [38] (as discussed in Chapter 4) that we can expect to solve unique-SVP instances with a certain probability p whenever we have

$$\frac{\lambda_2(\mathcal{L})}{\lambda_1(\mathcal{L})} \geq \tau_p \cdot \delta_0^{\dim(\mathcal{L})}$$

for some $\tau_p \in (0, 1]$. Though there are 'structural' differences in the lattices employed in this work and those in Chapter 4, we expect the model above to also hold reasonably well, with $\tau_{0.1} \approx 0.4$.

In any case, our experimental results suggest that the security bounds derived in Section 5.3 are already very pessimistic; even bigger keys than (5.2), for example, should be considered to thwart the improved embedding attack.

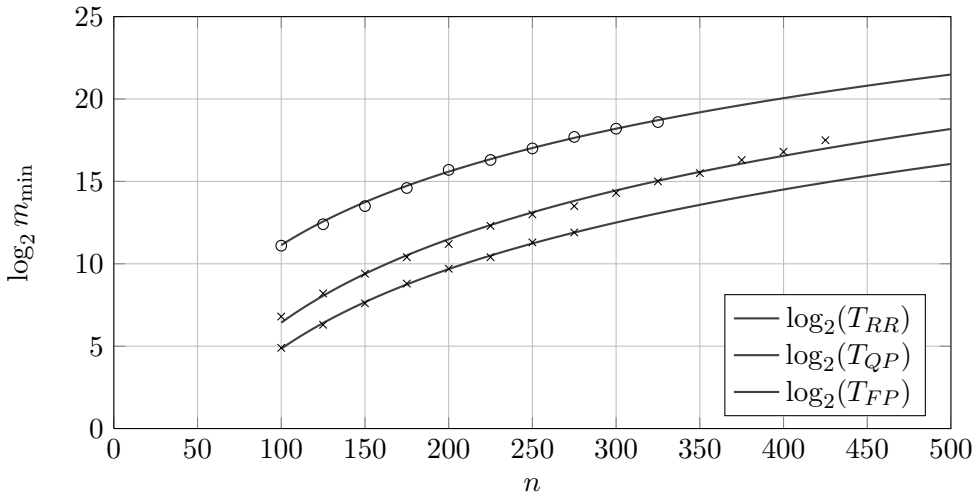

Figure 5.1: Logarithmic Running Times, G_LLL_FP, G_LLL_QP, and G_LLL_RR

[3] However, further improved embedding attacks may enable larger values of n to be attacked using only LLL, but we do not deal with this here.

			G_LLL_FP			G_LLL_QP			G_LLL_RR (with precision 150)		
n	q	$\|pk\|$(MB)	m'	exp. gap	T_{FP} (sec.)	m'	exp. gap	T_{QP} (sec.)	m'	exp. gap	T_{RR} (sec.)
100	$\approx 2^{65.058}$	0.875	30	$2^{3.942}$	$31 \approx 2^{4.954}$	30	$2^{3.942}$	$115 \approx 2^{6.845}$	30	$2^{3.942}$	$2412 \approx 2^{11.237}$
125	$\approx 2^{67.533}$	1.655	40	$2^{4.959}$	$82 \approx 2^{6.358}$	40	$2^{4.959}$	$294 \approx 2^{8.200}$	40	$2^{4.959}$	$5960 \approx 2^{12.541}$
150	$\approx 2^{69.737}$	2.794	50	$2^{5.748}$	$177 \approx 2^{7.468}$	50	$2^{5.748}$	$626 \approx 2^{9.290}$	50	$2^{5.748}$	$11974 \approx 2^{13.548}$
175	$\approx 2^{71.517}$	4.361	70	$2^{8.433}$	$466 \approx 2^{8.864}$	70	$2^{8.433}$	$1411 \approx 2^{10.463}$	70	$2^{8.433}$	$27190 \approx 2^{14.731}$
200	$\approx 2^{73.057}$	6.423	80	$2^{8.689}$	$810 \approx 2^{9.662}$	80	$2^{8.689}$	$2441 \approx 2^{11.253}$	80	$2^{8.689}$	$50976 \approx 2^{15.638}$
225	$\approx 2^{74.417}$	9.046	100	$2^{10.494}$	$1456 \approx 2^{10.508}$	100	$2^{10.494}$	$4513 \approx 2^{12.140}$	100	$2^{10.494}$	$86427 \approx 2^{16.399}$
250	$\approx 2^{75.633}$	12.300	120	$2^{11.940}$	$2487 \approx 2^{11.280}$	120	$2^{11.940}$	$7587 \approx 2^{12.889}$	120	$2^{11.940}$	$135423 \approx 2^{17.047}$
275	$\approx 2^{76.733}$	16.247	140	$2^{13.134}$	$3784 \approx 2^{11.886}$	130	$2^{11.916}$	$11720 \approx 2^{13.517}$	140	$2^{13.134}$	$203450 \approx 2^{17.634}$
300	$\approx 2^{77.737}$	20.960	—	—	fail	160	$2^{14.143}$	$19285 \approx 2^{14.235}$	160	$2^{14.143}$	$292092 \approx 2^{18.156}$
325	$\approx 2^{78.661}$	26.501				200	$2^{16.891}$	$32016 \approx 2^{14.967}$	190	$2^{15.969}$	$439574 \approx 2^{18.746}$
350	$\approx 2^{79.516}$	32.940				230	$2^{18.324}$	$44158 \approx 2^{15.430}$			
375	$\approx 2^{80.313}$	40.343				280	$2^{20.972}$	$82369 \approx 2^{16.330}$			
400	$\approx 2^{81.057}$	48.778				330	$2^{23.151}$	$119767 \approx 2^{16.870}$			
425	$\approx 2^{81.757}$	58.311				400	$2^{26.013}$	$175007 \approx 2^{17.417}$			
450	$\approx 2^{82.417}$	69.009				—	—	fail			

Table 5.3: Experimental results using G_LLL_FP, G_LLL_QP, and G_LLL_RR.

5.6 Closing Remarks

We presented a review and practical cryptanalysis of the public-key encryption scheme of Huang, Liu and Yang by exploiting the close connection between the hard problem underlying the scheme and the LWE problem, demonstrating from a practical and theoretical point of view that the assumptions of Huang, Liu and Yang are too optimistic. We further examine the possibility of finding a set of parameters for the scheme which would offer the desired security level against lattice attacks, reaching the conclusion that such an instantiation would only be possible at the cost of an enormous public key size even when not taking into account additional structural properties such as the presence of a small secret.

Bibliography

[1] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. Sampling discrete Gaussians efficiently and obliviously. Cryptology ePrint Archive, Report 2012/714, 2012. `http://eprint.iacr.org/`.

[2] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

[3] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293, 1997.

[4] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer Verlag, 2009.

[5] Martin Albrecht, Carlos Cid, Jean-Charles Faugere, Robert Fitzpatrick, and Ludovic Perret. On the complexity of the arora-ge algorithm against lwe. In *SCC '12: Proceedings of the 3nd International Conference on Symbolic Computation and Cryptography*, pages 93–99, Castro-Urdiales, July 2012.

[6] 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*, pages 1–30, 2013.

[7] Martin R. Albrecht, Pooya Farshim, Jean-Charles Faugere, and Ludovic Perret. Polly Cracker, revisited. In *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 179–196, Berlin, Heidelberg, New York, 2011. Springer Verlag. full version available as Cryptology ePrint Archive, Report 2011/289, 2011 `http://eprint.iacr.org/`.

[8] Martin R. Albrecht, Robert Fitzpatrick, Daniel Cabracas, Florian Gopfert, and Michael Schneider. A generator for LWE and Ring-LWE instances, 2013. available at `http://www.iacr.org/news/files/2013-04-29lwe-generator.pdf`.

[9] Martin R. Albrecht, Robert Fitzpatrick, and Florian Gopfert. On the efficacy of solving LWE by reduction to Unique-SVP. Information Security and Cryptology - ICISC 2013, 2013.

[10] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Advances in Cryptology – CRYPTO 2009*, Lecture Notes in Computer Science, pages 595–618, Berlin, Heidelberg, New York, 2009. Springer Verlag.

[11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In Luca Aceto, Monika Henzinger, and Jiri Sgall, editors, *ICALP*, volume 6755 of *Lecture Notes in Computer Science*, pages 403–415. Springer Verlag, 2011.

[12] Thomas Baigneres, Pascal Junod, and Serge Vaudenay. How far can we go beyond Linear Cryptanalysis? In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 432–450, Berlin, Heidelberg, New York, 2004. Springer Verlag.

[13] Wojciech Banaszczyk. New bounds in some transference theorems in the geometry of numbers. *Mathematische Annalen*, 296(1):625–635, 1993.

[14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. *IACR Cryptology ePrint Archive*, 2013:400, 2013.

[15] Magali Bardet. *etude des systèmes algebriques surdetermines. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Universite Paris VI, 2004.

[16] Magali Bardet, Jean-Charles Faugere, and Bruno Salvy. Complexity of Gröbner basis computation for semi-regular overdetermined sequences over F_2 with solutions in F_2. Technical Report 5049, INRIA, December 2003. Available at `http://www.inria.fr/rrrt/rr-5049.html`.

[17] Magali Bardet, Jean-Charles Faugere, and Bruno Salvy. On the complexity of Grobner basis computation of semi-regular overdetermined algebraic equations. In *Proc. International Conference on Polynomial System Solving (ICPSS)*, pages 71–75, 2004.

[18] Thomas Becker and Volker Weispfenning. *Gröbner Bases - A Computational Approach to Commutative Algebra.* Springer Verlag, Berlin, Heidelberg, New York, 1991.

[19] Côme Berbain, Henri Gilbert, and Jacques Patarin. Quad: A multivariate stream cipher with provable security. *J. Symb. Comput.*, 44(12):1703–1723, 2009.

[20] Elwyn R Berlekamp, Robert J McEliece, and Henk CA Van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

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

[22] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehle. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 575–584. ACM, 2013.

[23] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011*, pages 97–106. IEEE, 2011.

[24] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, University of Innsbruck, 1965.

[25] Bruno Buchberger, Georges E. Collins, Rudiger G. K. Loos, and Rudolph Albrecht. Computer algebra symbolic and algebraic computation. *SIGSAM Bull.*, 16(4):5–5, 1982.

[26] Daniel Cabarcas, Florian Göpfert, and Patrick Weiden. Provably secure lwe-encryption with uniform secret. *IACR Cryptology ePrint Archive*, 2013:164, 2013.

[27] J.W.S. Cassels. *An Introduction to ther Geometry of Numbers.* Classics in Mathematics. Springer Berlin Heidelberg, 1997.

[28] Yuanmi Chen and Phong Q. Nguyen. BKZ 2.0: better lattice security estimates. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 1–20, Berlin, Heidelberg, 2011. Springer Verlag.

[29] Nicolas Courtois and Louis Goubin. Cryptanalysis of the TTM cryptosystem. In *Advances in Cryptology – ASIACRYPT '00*, volume 1976 of *Lecture Notes in Computer Science*, pages 44–57. Springer, 2000.

[30] Jintai Ding and Bo-Yin Yang. Multivariate public key cryptography. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 193–234. Springer Verlag, Berlin, Heidelberg, New York, 2009.

[31] Vivien Dubois, Pierre-Alain Fouque, Adi Shamir, and Jacques Stern. Practical cryptanalysis of sflash. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2007.

[32] Jean-Charles Faugere. A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, pages 75–83, New York, 2002. ACM.

[33] Jean-Charles Faugere, Françoise Levy dit Vehel, and Ludovic Perret. Cryptanalysis of minrank. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 280–296. Springer Verlag, 2008.

[34] Jean-Charles Faugere and Antoine Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer Verlag, 2003.

[35] Jean-Charles Faugere and Ludovic Perret. Cryptanalysis of 2R– Schemes. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 357–372. Springer Berlin / Heidelberg, August 2006.

[36] Jean-Charles Faugere and Ludovic Perret. Polynomial equivalence problems: Algorithmic and theoretical aspects. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer Verlag, 2006.

[37] Pierre-Alain Fouque and eric Levieil. An improved LPN algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, 5th International Conference, SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer Verlag, 2006.

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

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

[40] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[41] Craig Gentry, Shai Halevi, and Nigel P. Smart. Advances in cryptology – crypto 2012. volume 7417 of *Lecture Notes in Computer Science*, pages 850–867, Berlin, Heidelberg, New York, 2012. Springer Verlag.

[42] GMP. GMP: The GNU multiple precision arithmetic library. `http://gmplib.org/`.

[43] Daniel Goldstein and Andrew Mayer. On the equidistribution of Hecke points. *Forum Mathematicum*, 15:165–189, 2003.

[44] Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-Chih Yao, editor, *ICS*, pages 230–240. Tsinghua University Press, 2010.

[45] Louis Granboulan, Antoine Joux, and Jacques Stern. Inverting hfe is quasipolynomial. In *CRYPTO*, pages 345–356, 2006.

[46] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2012.

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

[48] Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2001.

[49] Yun-Ju Huang, Feng-Hao Liu, and Bo-Yin Yang. Public-key cryptography from new multivariate quadratic assumptions. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography – PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 190–205. Springer Verlag, 2012.

[50] Fredrik Johansson et al. *mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 0.17)*, February 2011. `http://code.google.com/p/mpmath/`.

[51] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

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

[53] Jonathan Katz, Ji Sun Shin, and Adam Smith. Parallel and concurrent security of the hb and hb$^+$ protocols. *J. Cryptology*, 23(3):402–421, 2010.

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

[55] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature. In *Advances in Cryptology – EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 206–222. Springer-Verlag, 1999.

[56] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*, pages 19–30. Springer, 1999.

[57] Paul Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011. `http://eprint.iacr.org/`.

[58] Philip Klein. Finding the closest lattice vector when it's unusually close. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 937–941. Society for Industrial and Applied Mathematics, 2000.

[59] Neal Koblitz. *Algebraic Aspects of Cryptography.*, volume 3 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 1998.

[60] Lovász L. Lenstra H.W. jr., Lenstra A.K. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[61] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, volume 6558 of *LNCS*, pages 319–339. Springer, 2011.

[62] Cong Ling, Shuiyin Liu, Laura Luzzi, and Damien Stehlé. Decoding by embedding: Correct decoding radius and dmt optimality. *CoRR*, abs/1102.2936, 2011.

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

[64] Mingjie Liu, Xiaoyun Wang, Guangwu Xu, and Xuexin Zheng. Shortest lattice vectors in the presence of gaps. Cryptology ePrint Archive, Report 2011/139, 2011. `http://eprint.iacr.org/`. Last accessed 4th March 2012.

[65] Yi-Kai Liu, Vadim Lyubashevsky, and Daniele Micciancio. On bounded distance decoding for general lattices. In Josep Díaz, Klaus Jansen, Jose D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 450–461. Springer, 2006.

[66] László Lovász. *An algorithmic theory of numbers, graphs, and convexity.* CBMS-NSF regional conference series in applied mathematics. Philadelphia, Pa. Society for Industrial and Applied Mathematics, 1986.

[67] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 577–594. Springer, 2009.

[68] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. Swifft: A modest proposal for fft hashing. In Kaisa Nyberg, editor, *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer Verlag, 2008.

[69] Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology – EUROCRYPT 1988*, volume 330 of *LNCS*, pages 419–453. Springer–Verlag, 1988.

[70] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 2002.

[71] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology–EUROCRYPT 2012*, pages 700–718. Springer, 2012.

[72] Daniele Micciancio and Chris Peikert. Hardness of sis and lwe with small parameters. *IACR Cryptology ePrint Archive*, 2013:69, 2013.

[73] 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 Verlag, Berlin, Heidelberg, New York, 2009.

[74] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. *User Guide for MINPACK-1*, 1980.

[75] Ivan Morel, Damien Stehle, and Gilles Villard. H-LLL: using householder inside LLL. In Jeremy R. Johnson, Hyungju Park, and Erich Kaltofen, editors, *Symbolic and Algebraic Computation, International Symposium, ISSAC 2009*, pages 271–278. ACM, 2009.

[76] Phong Q. Nguyen. Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto '97. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 288–304. Springer, 1999.

[77] Phong Q. Nguyen. Lattice reduction algorithms: Theory and practice. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 2–6. Springer Verlag, 2011.

[78] Phong Q. Nguyen and Damien Stehle. Floating-point lll revisited. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 215–233. Springer, 2005.

[79] Phong Q. Nguyen and Damien Stehle. LLL on the average. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS*, volume 4076 of *Lecture Notes in Computer Science*, pages 238–256. Springer, 2006.

[80] Phong Q. Nguyen and Damien Stehle. Low-dimensional lattice basis reduction revisited. *ACM Transactions on Algorithms*, 5(4), 2009.

[81] Jacques Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms. In *Advances in Cryptology – EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996.

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

[83] Krzysztof Pietrzak. Subspace lwe. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 548–563. Springer, 2012.

[84] Xavier Pujol and Damien Stehle. Solving the shortest lattice vector problem in time $2^{2.465n}$. *IACR Cryptology ePrint Archive*, 2009:605, 2009.

[85] Oded Regev. New lattice based cryptographic constructions. In *STOC*, pages 407–416, 2003.

[86] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.

[87] Oded Regev. The learning with errors problem (invited survey). In *25th Annual IEEE Conference on Computational Complexity, CCC 2010*, pages 191–204. IEEE Computer Society, 2010.

[88] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. *IACR Cryptology ePrint Archive*, 2010:137, 2010.

[89] M. Sala, T. Mora, L. Perret, S. Sakata, and C. Traverso. *Gröbner Bases, Coding, and Cryptography*. Springer, Berlin, Heidelberg, New York, 2009.

[90] Michael Schneider, Johannes Buchmann, and Richard Lindner. Probabilistic analysis of lll reduced bases. 2010.

[91] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.

[92] Peter W. Shor. Polynominal time algorithms for discrete logarithms and factoring on a quantum computer. In Leonard M. Adleman and Ming-Deh A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, page 289. Springer, 1994.

[93] Victor Shoup. NTL: A library for doing number theory. `http://shoup.net/ntl/`.

[94] Carl Ludwig Siegel. A mean value theorem in geometry of numbers. *The Annals of Mathematics*, 46(2):340–347, 1945.

[95] Damien Stehle et al. *fpLLL 4.0.4*. fpLLL Development Team, 2013. `http://perso.ens-lyon.fr/damien.stehle/fplll/`.

[96] Damien Stehle, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer Verlag, 2009.

[97] W. A. Stein et al. *Sage Mathematics Software (Version 5.2)*. The Sage Development Team, 2012. `http://www.sagemath.org`.

[98] W. A. Stein et al. *Sage Mathematics Software*. The Sage Development Team, 2013. `http://www.sagemath.org`.

[99] A. J. Stothers. *On the Complexity of Matrix Multiplication*. PhD thesis, University of Edinburgh, 2010.

[100] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra (2. ed.)*. Cambridge University Press, 2003.

[101] Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In Howard J. Karloff and Toniann Pitassi, editors, *STOC*, pages 887–898. ACM, 2012.

[102] Christopher Wolf. *Multivariate quadratic polynomials in public key cryptography*. Univ. Leuven Heverlee, 2005.

[103] Bo-Yin Yang and Jiun-Ming Chen. Building secure tame-like multivariate public-key cryptosystems: The new tts. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP*, volume 3574 of *Lecture Notes in Computer Science*, pages 518–531. Springer, 2005.

Appendix A

An LWE Instance Generator

The contents of this appendix are based on "A Generator for LWE and Ring-LWE Instances" which was a work conducted in collaboration with M. R. Albrecht, D. Cabarcas, F. Göpfert and M. Schneider during a visit of M. R. Albrecht and the author to TU Darmstadt in 2013. The code described herein was included in SAGE [98].

A.1 Introduction

The LWE and Ring-LWE problems (reproduced in definitions 9 and 10) have received widespread attention from the cryptographic community in recent years.

Definition 9 (LWE). *Let n, q be positive integers, χ be a probability distribution on $\mathbb{Z}_q$ and $\mathbf{s}$ be a secret vector following the uniform distribution on $\mathbb{Z}_q^n$. We denote by $L_{\mathbf{s},\chi}^{(n)}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a}$ from the uniform distribution on $\mathbb{Z}_q^n$, choosing $e \in \mathbb{Z}$ according to χ, and returning $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

– Search-LWE *is the problem of finding $\mathbf{s} \in \mathbb{Z}_q^n$ given pairs $(\mathbf{a}_i, c_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\chi}^{(n)}$.*

– Decision-LWE *is the problem of deciding whether pairs $(\mathbf{a}_i, c_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ are sampled according to $L_{\mathbf{s},\chi}^{(n)}$ or the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*

Under certain conditions on the modulus and noise distribution χ (as discussed in Chapter 1), the solving certain (assumed hard) worst-case lattice problems can be reduced to solving (average-case) LWE. In recent work [22], the authors go beyond the *quantum* reduction of Regev [86], to give a classical reduction, further bolstering confidence in the hardness of LWE.

Definition 10 (Ring-LWE (informal))**.** *Let $n \geq 1$ be a power of 2 and let $q \equiv 1 \mod 2n$, $q \in \text{poly}(n)$ be a prime modulus. Let $f(x) = x^n + 1 \in \mathbb{Z}[x]$, implying that $f(x)$ is irreducible over $\mathbb{Q}$. Set $R = \mathbb{Z}[x]/\langle f(x)\rangle$, the ring of integer polynomials modulo $f(x)$ and set $R_q = R/\langle q\rangle$. Then the ring-LWE problem can (informally) be defined as: choosing $s \in R_q$ to be a uniformly random ring element and defining an error distribution χ on R such that the 'weight' of χ (in general terms) is concentrated on 'small' elements of R. Similarly to decision-LWE, the (decision) Ring-LWE problem is to distinguish between pairs $(a, a * s + e)$ (where $a \leftarrow \mathcal{U}(R_q)$, $e \leftarrow \chi$ and $*$ denotes multiplication in R_q) and pairs $(a, c) \leftarrow \mathcal{U}(R_q \times R_q)$.*

It should be noted that, despite the significant extra structure (and thus enhanced efficiency in constructions based on Ring-LWE) present in instances of the Ring-LWE problem, no algorithms are currently known which can exploit this structure in a significant manner, thus the present algorithms for the LWE and Ring-LWE problems are essentially the same.

This appendix aims to provide such benchmark instances and to facilitate research on the concrete hardness of LWE instances by making easy-to-use instance generators for the LWE and Ring-LWE problem available. This includes both generic classes for these problems (`LWE` and `RingLWE`) as well as specific generators for various proposals from the literature [86, 61, 26]. Our generators are written for and have been included in the SAGE mathematics software [98]. We start with an example where we construct an LWE oracle following [86]:

```
sage: from lwe import Regev
sage: Regev(n=128)  # Regev's parameters with security parameter n=128
```

```
LWE(128, 16411, DiscreteGaussianSamplerRejection(11.809841, 16411, 4), '
    uniform', None)
```

A.2 The Generator

The `LWE` and `RingLWE` constructors accept as parameter a noise distribution $\mathcal{D}$ which, when invoked, returns an element in $\mathbb{Z}$. We provide instances of such noise distribution oracles called "samplers" in our work. Somewhat contrary to intuition, sampling faithfully from Gaussian and (in particular) discretised Gaussian distributions is not a straight-forward task. The approach we employ is simple rejection sampling, in which elements are drawn uniformly at random from the support of a distribution, then rejected with probability inversely proportional to the probability density at the chosen point.

`DiscreteGaussianSamplerRejection` samples element in $\mathbb{Z}$ according to a a discrete Gaussian distribution centred at zero with standard deviation σ. We stress that this sampler is parameterised by the target standard deviation instead of the parameter $s = (\sigma \cdot \sqrt{2\pi})$ often found in the literature. We give an example below:

```
sage: from lwe import DiscreteGaussianSamplerRejection
sage: D = DiscreteGaussianSamplerRejection(3.0)
sage: variance([D() for _ in range(2000)]).sqrt().n()
3.01445160173946
```

Note that rejection sampling is currently the default strategy for sampling from a discrete Gaussian distribution. Hence, the default Gaussian sampler `DiscreteGaussianSampler` points to `DiscreteGaussianSamplerRejection`.

Some recent works (*e.g.*[46, 72]) raise the possibility of employing the uniform distribution over a small subset of $\mathbb{Z}$ in place of a discrete Gaussian, thus we include `UniformSampler` which samples uniformly between a lower and an upper bound as illustrated in the following example:

```
sage: from lwe import UniformSampler
```

```
sage: D = UniformSampler(-2,2)
sage: L = [D() for _ in range(2000)]
sage: [L.count(i) for i in (-2,-1,0,1,2)]
[399, 409, 380, 395, 417]
```

For each of these classes there also exist analogous polynomial variants for Ring-LWE.

The basic LWE class characterises LWE instances by the dimension n, a modulus q and a noise distribution $\mathcal{D}$ defined over $\mathbb{Z}$. We can construct LWE instances as follows:

```
sage: from lwe import DiscreteGaussianSampler, LWE
sage: D = DiscreteGaussianSampler(3)
sage: LDist = LWE(n=20, q=401, D=D)
sage: LDist
LWE(20, 401, DiscreteGaussianSamplerRejection(3.000000, 53, 4), 'uniform',
    None)
sage: LDist()
((19, 118, 249, 127, 347, 269, 232, 383, 16, 265, 247, 133, 102, 74, ... ,
    272), 63)
```

The `LWE` class also accepts a parameter `secret_dist` which may be either "uniform" or "noise". In the later case, the secret is sampled from $\mathcal{D}$. It also accepts a parameter `m` to limit the number samples returned, as is the case in some proposals, for instance [61]. After this limit is reached an `IndexError` is raised whenever another sample is requested. We highlight this using `RingLWE`:

```
sage: from lwe import RingLWE, DiscreteGaussianPolynomialSampler
sage: D = DiscreteGaussianPolynomialSampler(euler_phi(8), stddev=3)
sage: rlwe = RingLWE(N=8, q=next_prime(400), D=D, m=5); rlwe
RingLWE(8, 401, DiscreteGaussianPolynomialSamplerRejection(4, 3.000000, 53, 4)
    , x^4 + 1, 'uniform', 5)
sage: rlwe() # note that samples are tuples of vectors over IntegerModRing(q)
((314, 333, 270, 367), (89, 276, 152, 388))
sage: _ = [ rlwe() for _ in range(4) ]
sage: rlwe()
...
IndexError: Number of available samples exhausted.
```

A.2.1 Instances from the Literature

We also provide high-level interfaces to generate LWE instances from the literature. In particular, the following instance generators are supported.

- `Regev` instantiates an LWE oracle given a security parameter n following [86].

```
sage: from lwe import Regev
sage: Regev(128)
LWE(128, 16411, DiscreteGaussianSamplerRejection(11.809841, 16411, 4), '
    uniform', None)
```

- `LindnerPeikert` instantiates an LWE oracle given a security parameter n following [61].

```
sage: from lwe import LindnerPeikert
sage: LindnerPeikert(128)
LWE(128, 2053, DiscreteGaussianSamplerRejection(2.705800, 53, 4), 'noise
    ', None)
```

- `UniformNoiseLWE` instantiates an LWE oracle given a security parameter n following [26].

```
sage: from lwe import UniformNoiseLWE
sage: UniformNoiseLWE(128)
LWE(128, 389164331, UniformSampler(0, 486), 'noise', 177)
```

- `RingLindnerPeikert` instantiates a Ring-LWE oracle given a security parameter n following a generalisation of [61].

```
sage: from lwe import RingLindnerPeikert
sage: RingLindnerPeikert(128)
RingLWE(128, 2053, DiscreteGaussianPolynomialSamplerRejection(64,
    3.050908, 53, 4), x^64 + 1, 'noise', None)
```

A.2.2 Utility Functions

The function `samples` provides easy, one-line access to all generators. It accepts a number m requested samples, a security parameter n, an LWE instance generator,

a seed for the random number generator and a special parameter "balanced": by default, elements in $\mathbb{Z}_q$ are represented as integers in $x \in [0, q-1]$ by Sage. If, instead, we wish to use the balanced representation $x \in [-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$, the parameter "balanced" can be used to apply the balanced representation to scalars and vectors in and over $\mathbb{Z}_q$ (with output given as integers or $\mathbb{Z}$-module elements, respectively).

```
sage: S = samples(20, 10, 'RingLindnerPeikert', seed=1337); S
[((706, 21, 602, 420), (197, 136, 639, 177)),
 ...
 ((709, 855, 682, 57), (504, 166, 894, 963))]
sage: len(S)
20
```

With balanced-representation output:

```
sage: samples(20, 10, 'RingLindnerPeikert', seed=1337, balanced=True)
[((-325, 21, -429, 420), (197, 136, -392, 177)),
 ...
 ((-322, -176, -349, 57), (504, 166, -137, -68))]
```