UNIVERSITY OF CALIFORNIA SAN DIEGO

Gadgets and Gaussians in Lattice-Based Cryptography

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Electrical Engineering (Communication Theory and Systems)

by

Nicholas James Genise

Committee in charge:

Professor Daniele Micciancio, Chair
Professor Young-Han Kim, Co-Chair
Professor Mihir Bellare
Professor Massimo Franceschetti
Professor Shachar Lovett
Professor Alon Orlitsky

2019

The Dissertation of Nicholas James Genise is approved and is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California San Diego

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I start by thanking those who helped me during my time at UCSD. They are not listed in an order of contribution, but are split into those who offered academic advice and those who offered personal advice. Of course, no partition is perfect and many of the academics gave great guidance in general matters while friends and family often gave great guidance in academic-related matters.

I am grateful for my advisor, Daniele Micciancio, for his guidance, motivation, high expectations, and for showing me how to perform research. He took me on as a student from a department other than his own, multiple years in the PhD program, and with no background in his research area. From everything to sending me to the DARPA Safeware PI meeting (twice) to consistent clear communications (even in times of academic chaos), Daniele exceeded his duties as an adivsor.

I am grateful for the many professors at UCSD and the University of Michigan for showing me the beautiful areas of cryptography, coding and information theory, and mathematics in general. Further, I would like to thank the professors who gave me their guidance and honesty. These include, but are not limited to, Mihir Bellare, Young-Han Kim, Ken Zeger, Shachar Lovett, Massimo Franceschetti, Alex Vardy, Paul Siegel, Laura Balzano, Volker Elling, Kevin Xu, David Winter, Demostheis Teneketzis, and Achilleas Anastasopoulos.

I am grateful for the researchers and my fellow interns at Visa Research, where I interned during the summer of 2018 mentored by Yilei Chen and Pratyay Mukherjee. My experience at Visa was the "shot in the arm" I needed at the time. I learned so much more about cryptography than I thought possible during a summer internship. Everything from the (nearly daily) research meetings with Yilei and Pratyay to our large group lunches, I deeply enjoyed my time at Visa.

I am grateful for the many friendships I have gained during my time at UCSD. The number of graduate students who have impacted me are too many to list. Though, of particular importance to my time at UCSD were Srinjoy Das, Geoffrey Ganzberger, Michael Klug, and Sankeerth Rao.

I would like to thank my family for the constant support during the challenging times of the PhD. My mother and father always encouraged me to do well, and from them is where I first learned the importance of high expectations for oneself. Further, my older brother, Ben, often gave me a perspective I was missing when mulling over import decisions.

Lastly, I would like to thank my wonderful girlfriend Haley Richins for her unconditional support. Her constant encouragement helped push me forward in challenging times.

Chapter 2 of this dissertation is reprinted as it appears (with minor modifications) from the publication "Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus," presented by this dissertation's author at EUROCRYPT 2018 and is joint work with Daniele Micciancio.

Chapter 3 of this dissertation is reprinted as it appears (with minor modifications) from the publication "Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More," presented by this dissertation's author at EUROCRYPT 2019 and is joint work with Daniele Micciancio, and Yuriy Polyakov.

VITA

2013	Bachelor of Science and Engineering, Electrical Engineering, University of Michigan, Ann Arbor
2016	Master of Science, Electrical Engineering, University of California, San Diego
2019	Doctor of Philosophy in Electrical Engineering, University of California, San Diego

FIELDS OF STUDY

Major Field: Electrical Engineering (Communication Theory and Systems).

ABSTRACT OF THE DISSERTATION

Gadgets and Gaussians in Lattice-Based Cryptography

by

Nicholas James Genise

Doctor of Philosophy in Electrical Engineering (Communication Theory and Systems)

University of California San Diego, 2019

Professor Daniele Micciancio, Chair
Professor Young-Han Kim, Co-Chair

This dissertation explores optimal algorithms employed in lattice-based cryptographic schemes. Chapter 3 focuses on optimizing discrete gaussian sampling on "gadget" and algebraic lattices. These gaussian sampling algorithms are used in lattice-cryptography's most efficient trapdoor mechanism for the SIS and LWE problems: "MP12" trapdoors. However, this trapdoor mechanism was previously not optimized and inefficient (or not proven to be statistically correct) for structured lattices (ring-SIS/LWE), lattice-cryptography's most efficient form, where the modulus is often a prime. The algorithms in this chapter achieve optimality in this regime and have (already) resulted in a drastic efficiency improvement in independent implementations.

Chapter 4 digs deeper into the gadget lattice's associated algorithms. Specifically, we explore efficiently sampling a simple subgaussian distribution on gadget lattices, and we optimize LWE decoding on gadget lattices. These subgaussian sampling algorithms correspond to a randomized bit-decomposition needed in lattice-based schemes with homomorphic properties like fully homomorphic encryption (FHE). Next, we introduce a general class of "Chinese Remainder Theorem" (CRT) gadgets. These gadgets allow advanced lattice-based schemes to avoid multi-precision arithmetic when the applications modulus is larger than 64 bits.

The algorithms presented in the first two chapters improve the efficiency of many lattice-based cryptosystems: digital signature schemes, identity-based encryption schemes, as well as more advanced schemes like fully-homomorphic encryption and attribute-based encryption.

In the final chapter, we take a closer look at the random matrices used in trapdoor lattices. First, we revisit the constants in the concentration bounds of subgaussian random matrices. Then, we provide experimental evidence for a simple heuristic regarding the singular values of matrices with entries drawn from commonly used distributions in cryptography. Though the proofs in this chapter are dense, cryptographers need a strong understanding of the singular values of these matrices since their maximum singular value determines the concrete security of the trapdoor scheme's underlying SIS problem.

Chapter 1

Introduction

1.1 Motivation

A fundamental consequence of Shor's algorithm [78] is the inevitable insecurity of today's number-theoretic cryptography if large-scale quantum computers are built. An entire subfield of cryptography, which this dissertation lies, is dedicated to the advancement of efficient cryptosystems that are secure even when the adversary is in possession of a quantum computer (known as post-quantum cryptography). To date, lattice-based cryptography is the strongest post-quantum candidate. A lattice is a group of periodic points in Euclidean space. Though lattices were first used in cryptography to break cryptosystems [58], Ajtai [4] and Regev's [75] breakthroughs, respectively named the short integer solution problem (SIS) and the learning with errors problem (LWE), demonstrated cryptographic building blocks from lattice problems which admit proofs of average-case hardness from worst-case hardness assumptions. These proofs are seemingly unique to lattices. In short, they provide strong evidence one can pick keys in certain lattice-based schemes at random without worry.

Besides security against quantum attacks, another strength of lattices is the many advanced cryptosystems one can build from lattice assumptions, like SIS, LWE, and NTRU [56]. For a number of powerful cryptographic primitives, the only known constructions are based on lattices. These schemes include fully homomorphic encryption [77, 44, 26, 25, 29, 28, 47] (where a server can compute on encrypted data without the decryption key), and homomorphic digital

signatures [18, 50] (a secure method for verifying an experiment was honestly computed from a public data set). In addition, lattices yield advanced constructions like identity-based encryption (a user's public key is simply their name/identity) [46], as well as fine-grained access schemes like attribute-based encryption (ABE) [19] and constraint-hiding constrained pseudorandom functions [21, 33] (schemes where a user's key allows decryption of subsets of the encrypted data), all of which are believed to be secure against quantum threats.

Today, there is a great effort to introduce specialized algorithms specifically aimed at improving practical implementations [45, 22]. These specialized algorithms are mostly used in advanced lattice-based schemes like fully homomorphic encryption and attribute-based encryption. However, they often serve as building blocks used in simpler schemes like digital signatures or identity-based encryption.

This dissertation improves the efficiency of many of these schemes through optimizing the underlying lattice algorithms in the state-of-the-art lattice trapdoor procedures [66], as well as optimizing the gadget-related algorithms at the heart of nearly all of lattice-based cryptography's schemes with advanced, homomorphic properties.

1.2 Results

Our first contribution is the optimization of trapdoor discrete gaussian sampler for the SIS trapdoors of [66] for an arbitrary modulus q. Previously, the discrete gaussian sampling algorithm for SIS trapdoors was only optimized when the modulus was a power of two, $q = 2^k$. This is undesirable, for advanced primitives can only be efficiently implemented in the ring setting [64], where the modulus is large (polynomial-sized) prime or the product primes with each prime around 64 bits. Specifically for this contribution, we are concerned with sampling the gadget lattice, or "G-lattice," which is, more or less, a bit-decomposition lattice with ideal decoding properties that depends on the modulus q. The gadget sampler employs a novel idea: factor the G-lattice's basis into sparse, triangular matrices, sample a discrete gaussian on the first

matrix's lattice, then use the second matrix as an efficient linear transformation.

Our next contribution is giving an efficient perturbation sampling algorithm [72] specialized over the ring setting [64]. This algorithm is an FFT-like [36] algorithm which computes the FFO [41] matrix factorization on-the-fly resulting in less memory consumption than the algorithm [41] which can be adapted to a discrete gaussian sampler. Further, we prove the statistical correctness of our algorithm. This latter fact is crucial for applications, since the theoretical breakthrough of using discrete gaussian sampling in lattice trapdoors is that the preimages statistically hide the trapdoor [46]. Our statistical analysis uses a new convolution theorem via the Schur-complement decomposition of the lattice basis [80]. This analysis is represented in Theorem 3.4.1. It avoids the technicalities of the standard discrete gaussian convolution theorem [72] and is well-suited for our recursive algorithm. These contributions are given in-detail in Chapter 3 and are published in [42].

Our next contribution is the optimization of gadget-lattice algorithms [66, 11]. These include randomized bit-decompositions (subgaussian sampling) [10] and LWE decoding [66]. Both of these algorithms use the sparse matrix factorization from Chapter 3. The subgaussian sampling algorithm follows the same algorithmic blueprint as the discrete gaussian sampler introduced earlier in this dissertation: sample and map via the sparse matrices in the basis' factorization. The distribution sampled is a simple randomized rounding version of Babai's nearest plane algorithm [13] where we round to the nearest plane with probability according to the target's distance from the plane.

In addition, we introduce a general set of gadgets specifically tailored for the Chinese Remainder Theorem (CRT) setting. Specifically, the CRT setting is when a modulus is chosen so homomorphic computations can be performed over the CRT isomorphism, in-parallel. This allows implementations of advanced lattice-based schemes to keep arithmetic under the native 64 bits native in modern machines, hence avoiding multi-precision numbers, when the modulus has primes factors all under 64 bits. This trick can be used further in the ring setting by factoring the primes over the underlying ring of integers in the scheme [45]. These CRT gadget matrices

allow for parallel computations in gaussian sampling (using the algorithms from Chapter 3), subgaussian sampling, and LWE decoding (the latter two algorithms are introduced in Chapter 4). These CRT algorithms are crucial for the practical efficiency of lattice-based schemes since one can merely increase the modulus (more memory and parallelism) while keeping efficient 64-bit arithmetic. We implement our algorithms in the PALISADE library [73], an open source lattice cryptography library. We demonstrate the practical impact of our algorithms by showing a nearly 300x speedup in an attribute-based encryption scheme's ciphertext evaluation times [19]. These contributions (gadget algorithms) are published in [43].

The last contribution is an in-depth analysis of the subgaussian matrices used in lattice-based trapdoors [11, 66]. The underlying system parameters of many lattice-based schemes depend on the trapdoor matrix's largest singular value. Unfortunately, the state-of-the-art in subgaussian analysis [79] has unknown constants floating around in its singular value concentration bounds. This unknown constant directly affects the concrete security of the scheme since the hardness of the underlying SIS problem scales with this singular value. Our contributions here are two-fold: first to prove a concentration bound with exact constants, then we experimentally evaluate the constants appearing in the commonly used distributions in lattice-based schemes. These are presented in Chapter 5.

1.3 Outline

The next chapter covers basic definitions used throughout the dissertation. Then, Chapter 3 includes the discrete gaussian sampling algorithms: gadget and perturbation sampling. Next, we present the lattice gadget toolkit in Chapter 4. This includes the subgaussian sampling and LWE decoding algorithms as well as the general class of CRT gadgets. Lastly, our subgaussian matrix analysis is presented in Chapter 5.

Chapter 2

Preliminaries

2.1 Linear Algebra

The (forward) *Gram-Schmidt orthogonalization* of an ordered set of linearly independent vectors $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ is $\widetilde{\mathbf{B}} = \{\widetilde{\mathbf{b}}_1, \ldots, \widetilde{\mathbf{b}}_k\}$ where each $\widetilde{\mathbf{b}}_i$ is the component of $\mathbf{b}_i$ orthogonal to $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$ (and the backward GSO is defined as $\mathbf{b}_i^\dagger = \mathbf{b}_i \perp \text{span}(\mathbf{b}_{i+1}, \ldots, \mathbf{b}_n)$). An *anti-cylic* matrix is an $n \times n$ matrix of the form

$$\begin{bmatrix} a_0 & -a_{n-1} & \ldots & -a_1 \\ a_1 & a_0 & \ldots & -a_2 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & a_{n-2} & \ldots & a_0 \end{bmatrix}.$$

For any two (symmetric) matrices $\Sigma, \Gamma \in \mathbb{R}^{n \times n}$, we write $\Sigma \succeq \Gamma$ if $\mathbf{x}^T(\Sigma - \Gamma)\mathbf{x} \geq 0$ for all (nonzero) vectors $\mathbf{x} \in \mathbb{R}^n$, and $\Sigma \succ \Gamma$ if $\mathbf{x}^T(\Sigma - \Gamma)\mathbf{x} > 0$. It is easy to check that $\succeq$ is a partial order relation. Relations $\preceq$ and $\prec$ are defined symmetrically. When one of the two matrices $\Gamma = s\mathbf{I}$ is scalar, we simply write $\Sigma \succeq s$ or $\Sigma \preceq s$. A symmetric matrix $\Sigma \in \mathbb{R}^{n \times n}$ is called *positive definite* if $\Sigma \succ 0$, and *positive semidefinite* if $\Sigma \succeq 0$. Equivalently, Σ is positive semidefinite if and only if it can be written as $\Sigma = \mathbf{B}\mathbf{B}^T$ for some (square) matrix $\mathbf{B}$, called a *square root* of Σ and denoted $\mathbf{B} = \sqrt{\Sigma}$. (Notice that any $\Sigma \succ 0$ has infinitely many square roots $\mathbf{B} = \sqrt{\Sigma}$.) Σ is positive definite if and only if its square root $\mathbf{B}$ is a square non-singular matrix. When $\mathbf{B}$ is upper

(resp. lower) triangular, the factorization $\Sigma = \mathbf{B}\mathbf{B}^T$ is called the upper (resp. lower) triangular *Cholesky decomposition* of Σ. The Cholesky decomposition of any positive definite $\Sigma \in \mathbb{R}^{n\times n}$ can be computed with $O(n^3)$ floating point arithmetic operations. For any scalar s, $\Sigma \succ s$ if and only if all eigenvalues of Σ are strictly greater than s. In particular, positive definite matrices are nonsingular.

For any $n \times n$ matrix $\mathbf{S}$ and non-empty index sets $I, J \subseteq \{1,\ldots,n\}$, we write $\mathbf{S}[I,J]$ for the $|I| \times |J|$ matrix obtained by selecting the elements at positions $(i,j) \in I \times J$ from $\mathbf{S}$. When $I = J$, we write $\mathbf{S}[I]$ as a shorthand for $\mathbf{S}[I,I]$. For any nonsingular matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ and index partition $I \cup \bar{I} = \{1,\ldots,n\}$, $I \cap \bar{I} = \emptyset$, the $I \times I$ matrix

$$\mathbf{S}/I = \mathbf{S}[I] - \mathbf{S}[I,\bar{I}] \cdot \mathbf{S}[\bar{I}]^{-1} \cdot \mathbf{S}[\bar{I},I]$$

is called the *Schur complement* of $\mathbf{S}[\bar{I}]$, often denoted by $\mathbf{S}/\mathbf{S}[\bar{I}] = \mathbf{S}/I$. In particular, if $\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix}$ then the Schur complement of $\mathbf{A}$ is the matrix $\mathbf{S}/\mathbf{A} = \mathbf{D} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$. For any index set I, a symmetric matrix $\mathbf{S}$ is positive definite if and only if both $\mathbf{S}[I]$ and its Schur's complement $\mathbf{S}/\mathbf{S}[I]$ are positive definite.

Let $\Sigma = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix} \succ 0$. We can factor Σ in terms of a principal submatrix, say $\mathbf{D}$, and its Schur complement, $\Sigma/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^T$, as follows:

$$\Sigma = \begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Sigma/\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{B}^T & \mathbf{I} \end{bmatrix}.$$

The next two theorems regarding the spectra of principal submatrices and Schur complements of positive definite matrices are used in the analysis of our algebraic discrete Gaussian sampling algorithms. In both theorems, λ_i is the ith (in non-increasing order, with multiplicity) eigenvalue of a symmetric matrix.

Theorem 2.1.1 (Cauchy) *For any symmetric matrix* $\mathbf{S} \in \mathbb{R}^{n\times n}$, $I \subseteq \{1,\ldots,n\}$ *and* $1 \le i \le |I|$

$$\lambda_i(\mathbf{S}) \ge \lambda_i(\mathbf{S}[I]) \ge \lambda_{i+n-|I|}(\mathbf{S}).$$

Theorem 2.1.2 ([80, Corollary 2.3]) *For any positive definite* $\Sigma \in \mathbb{R}^{n\times n}$, $I \subseteq \{1,\ldots,n\}$ *and* $1 \le i \le |I|$

$$\lambda_i(\Sigma) \ge \lambda_i(\Sigma/I) \ge \lambda_{i+n-|I|}(\Sigma).$$

In other words, the eigenvalues of principal submatrices and Schur complements of a positive definite matrix are bounded from below and above by the smallest and largest eigenvalues of the original matrix.

Theorem 2.1.3 (Geršgorin) *Let* $\mathbf{M}$ *be an* $n \times n$ *matrix with complex entries. For each row* i*, let* r_i *be the sum of its non-diagonal entries' magnitudes:* $r_i = \sum_{j\neq i} |\mathbf{M}(i,j)|$*. Then, the eigenvalues of* $\mathbf{M}$ *are all in*

$$\bigcup_i \{z \in \mathbb{C} : |z - M(i,i)| \le r_i\}.$$

2.2 Lattices

A lattice is a discrete subgroup of $\mathbb{R}^n$. Equivalently, a lattice Λ can be represented as the set of all integer combinations of a basis $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_k] \in \mathbb{Z}^{n\times k}$, $\Lambda = \{\sum_1^k z_i \mathbf{b}_i : z_i \in \mathbb{Z}\} = \mathscr{L}(\mathbf{B})$. Notice that any permutation of basis vectors is another lattice basis. In fact, any matrix $\mathbf{U} \in \mathrm{GL}(k,\mathbb{Z})$ gives a new basis $\mathbf{B}' = \mathbf{B}\mathbf{U}$. Hence, any lattice $k > 1$ has infinitely many basis matrices. We only consider full-rank lattices ($k = n$). A lattice is an integer lattice if it is a subgroup of $\mathbb{Z}^n$. The dual lattice of Λ, denoted as Λ^*, is the set $\Lambda^* = \{\mathbf{z} \in \mathbb{R}^n : \langle \mathbf{z}, \Lambda\rangle \subseteq \mathbb{Z}\}$. Given a basis $\mathbf{B}$ for Λ, its dual basis is $\mathbf{B}^{-t}$ which is also a basis for Λ^*. We will consider direct sums of lattices, $\Lambda = \Lambda_1 \oplus \cdots \oplus \Lambda_l$ and their dual lattices $\Lambda^* = \Lambda_1^* \oplus \cdots \oplus \Lambda_l^*$. The number $\lambda_i(\Lambda)$ is the radius of the smallest ball containing i linearly independent lattice vectors.

Given a basis $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_n]$ for a lattice Λ, its (forward) Gram-Schmidt orthogonalization (GSO) is the set of vectors $\widetilde{\mathbf{B}} = [\widetilde{\mathbf{b}}_1, \cdots, \widetilde{\mathbf{b}}_n]$ where $\widetilde{\mathbf{b}}_i$ is the component of $\mathbf{b}_i$ orthogonal to span$(\mathbf{b}_1, \cdots, \mathbf{b}_{i-1})$. The GSO is not another basis for the lattice in-general, but it gives us a tiling of $\mathbb{R}^n$ given by $\mathbb{R}^n = \cup_{\mathbf{x}\in\Lambda}(\mathbf{x} + \mathscr{P}_{1/2}(\widetilde{\mathbf{B}}))$ where $\mathscr{P}_{1/2}(\widetilde{\mathbf{B}}) := \widetilde{\mathbf{B}} \cdot (-1/2, 1/2]^n$. Note that the GSO depends on the order of the vectors given. We define the reverse order GSO analogously. The algorithms presented in this dissertation will all be instantiations of Babai's greedy decoding algorithm known as the *nearest plane algorithm* [13].

Theorem 2.2.1 *There is an algorithm which given $\mathbf{B}, \widetilde{\mathbf{B}}, \mathbf{t} \in \mathbb{R}^n$ returns the unique lattice point in $\mathbf{t} + \mathscr{P}_{1/2}(\mathbf{B}^*)$ in time $O(n^2)$ and memory $O(n^3)$*[1].

2.3 Discrete and Subgaussians

Let $A \subset \mathbb{R}^n$ be a discrete set, and let the (spherical) Gaussian function with width s and center $\mathbf{c} \in \mathbb{R}^n$ be $\rho_{s,\mathbf{c}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x} - \mathbf{c}\|^2/s^2)$. Let $\rho_{s,\mathbf{c}}(A) = \sum_{\mathbf{y}\in A} \rho_{s,\mathbf{c}}(\mathbf{y})$. The smoothing parameter of a lattice [67] for some $\varepsilon > 0$, is dentoted as $\eta_\varepsilon(\Lambda)$, and it is defined as the minimum $s > 0$ such that $\rho(s \cdot \Lambda^*) \leq 1 + \varepsilon$. When $s = 1$ and $\mathbf{c} = \mathbf{0}$, we denote this as $\rho(\cdot)$. Then, the discrete Gaussian distribution has probability $\rho_{s,\mathbf{c}}(\mathbf{x})/\rho_{s,\mathbf{c}}(A)$ for each $\mathbf{x} \in A$. This distribution is denoted as $D_{A,s,\mathbf{c}}$. Polynomial time discrete Gausisan sampling algorithms for general lattices and their cosets, with width above the GSO length of the input basis (times a small factor, $\omega(\sqrt{\log n})$ or $O(\sqrt{\log n})$), are given in [46, 27].

Subgaussian distributions

Subgaussian distributions are those on $\mathbb{R}$ which have tails dominated by gaussians [79]. An equivalent formulation is through a distribution's moment generating function, and the definition below is most-commonly used throughout lattice-based cryptography [66, 65].

[1] This assumes the GSO has entries each described in $O(n)$ bits, but often we use floating point numbers in implementations.

Definition 2.3.1 *A random variable X over the reals is* subgaussian with parameter $s > 0$ *if for all* $t \in \mathbb{R}$*, it holds*

$$\mathbb{E}[e^{2\pi tX}] \leq e^{\pi s^2 t^2}.$$

From here we can derive the gaussian concentration bound.

Corollary 2.3.1 *A subgaussian random variable, X, with parameter* $s > 0$ *satisfies*

$$\Pr\{|X| \geq t\} \leq 2\exp(-\pi t^2/s^2)$$

for all $t > 0$*.*

Proof: Let $\delta \in \mathbb{R}$ be arbitrary. Then,

$$\begin{aligned}\Pr\{X \geq t\} &= \Pr\{\exp(2\pi\delta X) \geq \exp(2\pi\delta t)\} \leq \exp(-2\pi\delta t) \cdot \mathbb{E}[\exp(2\pi\delta X)] \\ &\leq \exp(-2\pi\delta t + \pi\delta^2 s^2).\end{aligned}$$

This is minimized at $\delta = t/s^2$, so we have

$$\Pr\{X \geq t\} \leq \exp(-\pi t^2/s^2).$$

The symmetric case, $X \leq -t$, is analogous and the proof is completed by a union bound. □

A random vector $\mathbf{x}$ over $\mathbb{R}^n$ is *subgaussian* with parameter $\alpha > 0$ if $\langle \mathbf{x}, \mathbf{u} \rangle$ is subgaussian with parameter α for all unit vectors $\mathbf{u}$. If each coefficient of a random vector is subgaussian with parameter α conditioned on the previous coefficients taking any values, then the vector is subgaussian with parameter α.

Lemma 2.3.1 *Let* $\mathbf{x}$ *be a discrete random vector over* $\mathbb{R}^n$ *such that each coordinate* x_i *is subgaussian with parameter* α_i *given the previous coordinates take any values. Then,* $\mathbf{x}$ *is a subgaussian vector with parameter* $\max_i\{\alpha_i\}$.

Proof: We expand the moment generating function:

$$\begin{aligned}\mathbb{E}[\exp(2\pi t \langle \mathbf{x}, \mathbf{u} \rangle)] &= \sum_{\chi} \Pr\{\mathbf{x} = (\chi_1, \cdots, \chi_n)\} \exp(2\pi t \langle \chi, \mathbf{u} \rangle) \\ &\leq \exp(\pi t^2 \sum_i \alpha_i^2 u_i^2) \\ &\leq \exp(\pi t^2 \max_i {\alpha_i}^2 \|\mathbf{u}\|^2).\end{aligned}$$

□

2.4 Gadgets

A gadget lattice is described by a matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times nk}$ and the lattice itself, often called the "G-lattice," is the set $\Lambda_q^{\perp}(\mathbf{G}) = \{\mathbf{x} \in \mathbb{Z}^{nk} : \mathbf{G}\mathbf{x} = \mathbf{0} \mod q\}$. All gadget matrices, $\mathbf{G}$, in this dissertation have the same block-diagonal form: $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t$ for some $\mathbf{g}^t$. This allows us to focus on the smaller lattice $\Lambda_q^{\perp}(\mathbf{g}^t)$ since

$$\Lambda_q^{\perp}(\mathbf{G}) = \Lambda_q^{\perp}(\mathbf{g}^t) \oplus \cdots \oplus \Lambda_q^{\perp}(\mathbf{g}^t), \ n \text{ times.}$$

The most common gadget is the power-of-b gadget $\mathbf{g}^t := (1, b, b^2, \cdots, b^{k-1})$ where $k := \lceil \log_b q \rceil$. This lattice has a simple basis

$$\mathbf{B}_q = \begin{bmatrix} b & & & & q_0 \\ -1 & b & & & q_1 \\ & -1 & \ddots & & \vdots \\ & & \ddots & b & q_{k-2} \\ & & & -1 & q_{k-1} \end{bmatrix}$$

where $(q_0, \ldots, q_{k-1}) = [q]_b^k = \mathbf{q}$ is the base-b representation of the modulus q. We cheat when $q = b^k$ and say $\mathbf{q} = (0, \cdots, 0, b)$. Notice that for a fixed b, k, all gadget lattices have the same

$k-1$-dimensional sublattice generated by the basis' first $k-1$ columns. Despite these lattices being almost the same, this last dimension, defined by $\mathbf{q}$, has a significant impact on the efficiency of Babai's nearest plane algorithm [13] on $\Lambda_q^{\perp}(\mathbf{g}^t)$. This is because the reverse GSO of $\mathbf{B}_q$ is dense whenever $q \neq b^k$.

Chapter 3

Discrete Gaussian Sampling on Algebraic and Gadget Lattices

3.1 Introduction

We present improved algorithms for gaussian preimage sampling using the lattice trapdoors of [66], called "MP12 trapdoors" throughout this dissertation. For simplicity, we view the trapdoor scheme through the lens of a hash-and-sign digital signature scheme [46]. The scheme's verification key is an SIS matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ (short and fat since $m \geq n \log q$) and the signing key is a randomly chosen matrix with small entries $\mathbf{T}$. Signing is done by hashing a message with a random salt $H(m||r) = \mathbf{u} \in \mathbb{Z}_q^n$, and using the trapdoor to return a short, gaussian vector $\mathbf{x} \in \mathbb{Z}^m$ satisfying[1]

$$\mathbf{A}\mathbf{x} \mod q = \mathbf{u} \ \ \& \ \ \|\mathbf{x}\|_2 = \text{"short"} \leq \beta.$$

MP12 trapdoor sampling consist of two discrete gaussian sampling steps:

1. online discrete gaussian sampling on message-dependent shift of a public gadget lattice, and

2. an offline perturbation sampling which only depends on the trapdoor $\mathbf{T}$.

[1] We hide the trapdoor and gadget in $\mathbf{A}$ by sampling $\mathbf{A}, \mathbf{T}$ together as $\mathbf{A} = [\mathbf{A}'|\mathbf{G} - \mathbf{A}'\mathbf{T}] \mod q$ where $\mathbf{A}'$ is a truly random matrix. $\mathbf{A}$ is close to statistically random by the left-over hash lemma. Now, we can sign a message by sampling a perturbation $\mathbf{p}$ and sampling a short $\mathbf{y}$ satisfying $\mathbf{G}\mathbf{y} = \mathbf{u} - \mathbf{A}\mathbf{p} \mod q$. The final signature is $\mathbf{x} := \mathbf{p} + \mathbf{T}\mathbf{y}$.

Table 3.1. Running time and storage of the (G-sampling) algorithm. G-Sampling running times are scaled by a factor n to take into account that each sample requires n independent calls to the underlying G-sampling operation.

	MP12	MP12	**This work**
modulus q	2^k	any	any
G-Sampling precomp.	—	$O(\log^3 q)$	—
G-Sampling space	$O(\log q)$	$O(\log^2 q)$	$O(\log q)$
G-Sampling time	$O(n \log q)$	$O(n \log^2 q)$	$O(n \log q)$

Optimizing the online stage is most crucial for applications since we can always precompute and store perturbations (whereas there are q^n different possible signatures $\mathbf{u}$).

Contribution

We present a new algorithm for the online sampling stage capable of handling any modulus q (including the large prime moduli required in the ring setting) while achieving the same level of performance of the specialized discrete gaussian algorithm of [66] for a power-of-two modulus $q = 2^k$. This improves the running time of [66] for arbitrary modulus from cubic $\log^3 q$ (or quadratic $\log^2 q$, using precomputation and a substantial amount of storage) to just linear in $\log q$ and with minimal storage requirements. These G-sampling improvements are summarized in Table 3.1.

The second contribution of this chapter is an optimized discrete gaussian sampling on structured lattices, or equivalently an efficient algorithm to sample the integer lattice with a structure covariance. The covariances in MP12's ring-SIS/LWE perturbations have blocks corresponding to polynomial multiplication over a finite-dimensional polynomial ring, $R = \mathbb{Z}[x]/(x^n + 1)$. A similar algorithm can be adapted from Ducas and Prest [41], though without a proof of statistical correctness. Further, the algorithm presented here is more memory efficient (linear space consumption versus quasi-linear space) than the adapting from Ducas and Prest's algorithm [41]. We emphasize that the proof of statistical correctness is critical for applications since the preimages provided by MP12 cannot leak information about the trapdoor. Otherwise,

the attacks of Nguyen and Regev could recover the trapdoor [69].

Technical details

The main idea behind our efficient gaussian sampling algorithm is quite simple. Surprisingly, the commonly used basis of the (power-of-b) gadget lattice, denoted $\mathbf{B}_q$, has a factorization allowing us to efficiently sample a different lattice:

$$\mathbf{B}_q = \mathbf{BD}.$$

Both $\mathbf{B}$ and $\mathbf{D}$ are sparse and triangular, so we can sample a discrete gaussian on the G-lattice by

1. sampling a discrete gaussian on the lattice generated by $\mathbf{D}$,
2. and apply $\mathbf{B}$ as a linear transformation.

The matrix $\mathbf{D}$ has entries in $[0,1]$, so we can sample a narrow discrete gaussian on the lattice generated by $\mathbf{D}$, and the matrix $\mathbf{B}$ has small entries, so the width of the distribution is not increased by much when we use $\mathbf{B}$ as a linear transformation. This also applies when $q = b^k$ since $\mathbf{D} = \mathbf{I}_k$ when $q = b^k$, but the algorithm of Micciancio and Peikert [66] is much simpler in this case. Further, we add a perturbation, like Peikert's algorithm [72], to get a spherical sample. We have this perturbation step not for security but for efficiency in generating the offline perturbations.

Next, our algebraic perturbation algorithm follows the simple recursive structure of fast-Fourier transform [36]. That is, we view the n-dimensional ring $R = \mathbb{Z}[x]/(x^n + 1)$ as a two dimensional module over the smaller $n/2$-dimensional ring. Algorithmically, this is done to a change of basis corresponding to the Schur complement [80] and a convolution [72].

The actual steps are quite technical. Though, we needed to prove a new convolution theorem suited more towards these Schur complement convolutions. We emphasize that a proof of statistical correctness is crucial for applications, since we need to trapdoor samples to statistically hide the trapdoor [46].

In addition, we sketch how one would compute the perturbations in the canonical embedding for non-power-of-two cyclotomic rings. The algorithm is much simpler in the canonical embedding due to its diagonal structure of multiplication matrices.

3.2 Background

We denote the complex numbers as $\mathbb{C}$, the real numbers as $\mathbb{R}$, the rational numbers as $\mathbb{Q}$, and the integers as $\mathbb{Z}$. A number is denoted by a lower case letter, $z \in \mathbb{Z}$ for example. We denote the conjugate of a complex number y as y^*. When q is a positive integer, $\log q$ is short for its rounded up logarithm in base two, $\lceil \log_2 q \rceil$. A floating point number with mantissa length m representing $x \in \mathbb{R}$ is denoted as $\bar{x}$. The index set of the first n natural numbers is $[n] = \{1, \ldots, n\}$. Vectors are denoted by bold lower case letters, $\mathbf{v}$, and are in column form ($\mathbf{v}^t$ is a row vector) unless stated otherwise. The inner product of two vectors is $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^t \mathbf{y}$. We denote matrices with bold upper case letters $\mathbf{B}$ or with upper case Greek letters (for positive-definite matrices). The transpose of a matrix is $\mathbf{B}^t$. The entry of $\mathbf{B}$ in row i and column j is denoted $B_{i,j}$. Unless otherwise stated, the norm of a vector is the ℓ_2 norm. The norm of a matrix $\|\mathbf{B}\| = \max_i \|\mathbf{b}_i\|$ is the maximum norm of its column vectors. Given two probability distributions over a countable domain D, the statistical distance between them is $\Delta_{\mathrm{SD}}(X,Y) = \frac{1}{2} \sum_{\omega \in D} |X(\omega) - Y(\omega)|$. In order to avoid tracing irrelevant terms in our statistical distance computations, we define $\hat{\varepsilon} = \varepsilon + O(\varepsilon^2)$.

We denote a random variable x sampled from a distribution $\mathscr{D}$ as $x \leftarrow \mathscr{D}$. A random variable distributed as $\mathscr{D}$ is denoted $x \sim \mathscr{D}$. We denote an algorithm $\mathscr{A}$ with oracle access to another algorithm $\mathscr{B}$ (distribution $\mathscr{D}$) as $\mathscr{A}^{\mathscr{B}}$ ($\mathscr{A}^{\mathscr{D}}$).

The max-log, or ML, distance between two distributions was recently introduced by [68] in order to prove tighter bounds for concrete security. The *ML distance* between two discrete distributions over the same support, S, as

$$\Delta_{\mathrm{ML}}(\mathscr{P}, \mathscr{Q}) = \max_{x \in S} |\ln \mathscr{Q}(x) - \ln \mathscr{P}(x)|.$$

Let $\mathscr{P}, \mathscr{Q}$ be distributions over a countable domain again and let S be the support of $\mathscr{P}$.

The *Rényi divergence* of order infinity of $\mathscr{Q}$ from $\mathscr{P}$ is

$$R_\infty(\mathscr{P}||\mathscr{Q}) = \max_{x \in S} \frac{\mathscr{P}(x)}{\mathscr{Q}(x)}.$$

Rényi divergence is used in [14] to yield a tighter security analysis than one using statistical distance.

3.2.1 Gaussians and Lattices

A *lattice* $\Lambda \subset \mathbb{R}^n$ is a discrete subgroup of $\mathbb{R}^n$. Specifically, a lattice of *rank* k is the integer span $\mathscr{L}(\mathbf{B}) = \{z_1\mathbf{b}_1 + \cdots + z_k\mathbf{b}_k \mid z_i \in \mathbb{Z}\}$ of a basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\} \subset \mathbb{R}^n$ $(k \leq n)$. There are infinitely many bases for a given lattice since right-multiplying a basis by a unimodular transformation gives another basis. The *dual lattice* of Λ, denoted by Λ^*, is the lattice $\{\mathbf{x} \in \text{span}(\Lambda) | \langle \mathbf{x}, \Lambda \rangle \subseteq \mathbb{Z}\}$. It is easy to see that $\mathbf{B}^{-t}$ is a basis for $\mathscr{L}(\mathbf{B})^*$ for a full rank lattice $(n = k)$.

The n-dimensional *gaussian* function $\rho : \mathbb{R}^n \to (0, 1]$ is defined as $\rho(\mathbf{x}) := \exp(-\pi\|\mathbf{x}\|^2)$. Applying an invertible linear transformation $\mathbf{B}$ to the gaussian function yields

$$\rho_{\mathbf{B}}(\mathbf{x}) = \rho(\mathbf{B}^{-1}\mathbf{x}) = \exp(-\pi \cdot \mathbf{x}^t\Sigma^{-1}\mathbf{x})$$

with $\Sigma = \mathbf{B}\mathbf{B}^t \succ 0$. For any $\mathbf{c} \in \text{span}(\mathbf{B}) = \text{span}(\Sigma)$, we also define the shifted gaussian function (centered at $\mathbf{c}$) as $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x}) = \rho_{\sqrt{\Sigma}}(\mathbf{x} - \mathbf{c})$. Normalizing the function $\rho_{\mathbf{B},\mathbf{c}}(\mathbf{x})$ by the measure of $\rho_{\mathbf{B},\mathbf{c}}$ over the span of $\mathbf{B}$ gives the *continuous gaussian distribution* with covariance $\Sigma/(2\pi)$, denoted by $D_{\sqrt{\Sigma},\mathbf{c}}$. Let $S \subset \mathbb{R}^n$ be any discrete set in $\mathbb{R}^n$, then we define $\rho_{\sqrt{\Sigma}}(S) := \sum_{\mathbf{s} \in S} \rho_{\sqrt{\Sigma}}(\mathbf{s})$. The *discrete gaussian* distribution over a lattice Λ, denoted by $D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$, is defined by restricting the support of the distribution to Λ. Specifically, a sample $\mathbf{y} \leftarrow D_{\Lambda,\sqrt{\Sigma},\mathbf{c}}$ has probability mass function $\rho_{\sqrt{\Sigma},\mathbf{c}}(\mathbf{x})/\rho_{\sqrt{\Sigma},\mathbf{c}}(\Lambda)$ for all $\mathbf{x} \in \Lambda$. Discrete gaussians on lattice cosets $\Lambda + \mathbf{c}$, for $\mathbf{c} \in \text{span}(\Lambda)$, are defined similarly setting $\Pr\{\mathbf{y} \leftarrow D_{\Lambda+\mathbf{c},\sqrt{\Sigma},\mathbf{p}}\} = \rho_{\sqrt{\Sigma},\mathbf{p}}(\mathbf{y})/\rho_{\sqrt{\Sigma},\mathbf{p}}(\Lambda + \mathbf{c})$ for all $\mathbf{y} \in \Lambda + \mathbf{c}$. For

brevity we let $D_{\Lambda+\mathbf{c},\sqrt{\Sigma},\mathbf{p}}(\mathbf{y}) := \Pr\{\mathbf{y} \leftarrow D_{\Lambda+\mathbf{c},\sqrt{\Sigma},\mathbf{p}}\}$.

For a lattice Λ and any (typically small) positive $\varepsilon > 0$, the *smoothing parameter* $\eta_\varepsilon(\Lambda)$ [67] is the smallest $s > 0$ such that $\rho(s \cdot \Lambda^*) \leq 1+\varepsilon$. A one-dimensional discrete gaussian with a tail-cut, t, is a discrete gaussian $D_{\mathbb{Z},c,s}$ restricted to a support of $\mathbb{Z} \cap [c - t \cdot s, c + t \cdot s]$. We denote this truncated discrete gaussian as $D^t_{\mathbb{Z},c,s}$. In order to use the ML distance in Section 3.3, we will restrict all tail-cut discrete gaussians to a universal support of $\mathbb{Z} \cap [c - t \cdot s_{max}, c + t \cdot s_{max}]$ for some s_{max}.

Lemma 3.2.1 ([46, Lemma 4.2]) *For any $\varepsilon > 0$, any $s \geq \eta_\varepsilon(\mathbb{Z})$, and any $t > 0$,*

$$\Pr_{x \leftarrow D_{\mathbb{Z},s,c}}[|x - c| \geq t \cdot s] \leq 2e^{-\pi t^2} \cdot \frac{1+\varepsilon}{1-\varepsilon}.$$

More generally, for any positive definite matrix Σ and lattice $\Lambda \subset \mathrm{span}(\Sigma)$, we write $\sqrt{\Sigma} \geq \eta_\varepsilon(\Lambda)$, or $\Sigma \succeq \eta^2_\varepsilon(\Lambda)$, if $\rho(\sqrt{\Sigma}^t \cdot \Lambda^*) \leq 1+\varepsilon$. The reader is referred to [67, 46, 72] for additional background on the smoothing parameter.

Here we recall two bounds and a discrete gaussian convolution theorem to be used later.

Lemma 3.2.2 ([46, Lemma 3.1]) *Let $\Lambda \subset \mathbb{R}^n$ be a lattice with basis $\mathbf{B}$, and let $\varepsilon > 0$. Then,*

$$\eta_\varepsilon(\Lambda) \leq \|\widetilde{\mathbf{B}}\| \sqrt{\log(2n(1+1/\varepsilon))/\pi}.$$

Lemma 3.2.3 ([72, Lemma 2.5]) *For any full rank n-dimensional lattice Λ, vector $\mathbf{c} \in \mathbb{R}^n$, real $\varepsilon \in (0,1)$, and positive definite $\Sigma \succeq \eta^2_\varepsilon(\Lambda)$,*

$$\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{c}) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_{\sqrt{\Sigma}}(\Lambda).$$

Theorem 3.2.1 ([72, Theorem 3.1]) *For any vectors $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$, lattices $\Lambda_1, \Lambda_2 \subset \mathbb{R}^n$, and positive definite matrices $\Sigma_1, \Sigma_2 \succ 0$, $\Sigma = \Sigma_1 + \Sigma_2 \succ 0$, $\Sigma_3^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} \succ 0$, if $\sqrt{\Sigma_1} \succeq \eta_\varepsilon(\Lambda_1)$*

and $\sqrt{\Sigma_3} \succeq \eta_\varepsilon(\Lambda_2)$ *for some* $0 < \varepsilon \leq 1/2$*, then the distribution*

$$X = \{\mathbf{x} \mid \mathbf{p} \leftarrow D_{\Lambda_2+\mathbf{c}_2,\sqrt{\Sigma_2}}, \mathbf{x} \leftarrow D_{\Lambda_1+\mathbf{c}_1,\sqrt{\Sigma_1},\mathbf{p}}\}$$

is within statistical distance $\Delta(X,Y) \leq 8\varepsilon$ *from the discrete gaussian* $Y = D_{\Lambda_1+\mathbf{c}_1,\sqrt{\Sigma}}$.

Below we have the correctness theorem for the standard, randomized version of Babai's nearest plane algorithm. The term *statistically close* is the standard cryptographic notion of negligible statistical distance[2]. We emphasize that the algorithm reduces to sampling $D_{\mathbb{Z},s,c}$.

Theorem 3.2.2 ([46, Theorem 4.1]) *Given a full-rank lattice basis* $\mathbf{B} \in \mathbb{R}^{n\times n}$*, a parameter* $s \geq \|\tilde{\mathbf{B}}\|\omega(\sqrt{\log n})$*, and a center* $\mathbf{c} \in \mathbb{R}^n$*, there is an* $O(n^2)$*-time, with a* $O(n^3)$*-time preprocessing, probabilistic algorithm whose output is statistically close to* $D_{\mathscr{L}(\mathbf{B}),s,\mathbf{c}}$.

3.2.2 Cyclotomic Fields

Let n be a positive integer. The *n-th cyclotomic field over* $\mathbb{Q}$ is the number field $\mathscr{K}_n = \mathbb{Q}[x]/(\Phi_n(x)) \cong \mathbb{Q}(\zeta)$ where ζ is an *n-th primitive root of unity* and $\Phi_n(x)$ is the minimal polynomial of ζ over $\mathbb{Q}$. The nth *cyclotomic ring* is $\mathscr{O}_n = \mathbb{Z}[x]/(\Phi_n(x))$. Let $\varphi(n)$ be Euler's totient function. $\mathscr{K}_n$ is a $\varphi(n)$-dimensional $\mathbb{Q}$-vector space, and we can view $\mathscr{K}_n$ as a subset of $\mathbb{C}$ by viewing ζ as a complex primitive n-th root of unity.

Multiplication by a fixed element f, $g \mapsto f \cdot g$, is a linear transformation on $\mathscr{K}_n$ as a $\mathbb{Q}$-vector space. We will often view field elements as $\varphi(n)$-dimensional rational vectors via the *coefficient embedding*. This is defined by $f(x) = \sum_{i=0}^{\varphi(n)-1} f_i x^i \mapsto (f_0, \cdots, f_{\varphi(n)-1})^t$ mapping a field element to its vector of coefficients under the *power basis* $\{1, x, \cdots, x^{\varphi(n)-1}\}$ (or equivalently $\{1, \zeta, \cdots, \zeta^{\varphi(n)-1}\}$). We can represent a field element as the matrix in $\mathbb{Q}^{\varphi(n)\times\varphi(n)}$ representing the linear transformation by its multiplication in the coefficient embedding. This matrix is called

[2]Precisely, a function $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is negligible if for every $c > 1$ there exists an N such that for all $n > N$, $f(n) < n^{-c}$.

a field element's coefficient *multiplication* matrix. When n is a power of two, an element's coefficient multiplication matrix is anti-cyclic.

An *isomorphism* from the field F to the field K is a bijection $\theta : F \to K$ such that $\theta(fg) = \theta(f)\theta(g)$, and $\theta(f+g) = \theta(f) + \theta(g)$ for all $f, g \in F$. An *automorphism* is an isomorphism from a field to itself. For example, if we view the cyclotomic field $\mathscr{K}_n$ as a subset of the complex numbers, then the *conjugation* map $f(\zeta) \mapsto f(\zeta)^* = f(\zeta^*)$ is an automorphism and can be computed in linear time $O(n)$. In power-of-two cyclotomic fields, the conjugation of a field element corresponds to the matrix transpose of an element's anti-cyclic multiplication matrix.

Another embedding is the *canonical* embedding which maps an element $f \in \mathscr{K}_n$ to the vector of evaluations of f, as a polynomial, at each root of $\Phi_n(x)$. When n is a power of two, the linear transformation between the coefficient embedding and the canonical embedding is a scaled isometry.

Let n be a power of two, then the field $\mathscr{K}_{2n}$ is a two-dimensional $\mathscr{K}_n$-vector space as see by splitting a polynomial $f(x) \in \mathscr{K}_{2n}$ into $f(x) = f_0(x^2) + x \cdot f_1(x^2)$ for $f_i \in \mathscr{K}_n$. Now, we can view the linear transformation given by multiplication by some $f \in \mathscr{K}_{2n}$ as a linear transformation over $\mathscr{K}_n \oplus \mathscr{K}_n \cong \mathscr{K}_{2n}$. Let $\phi_{2n} : \mathscr{K}_{2n} \to \mathbb{Q}^{n \times n}$ be the injective ring homomorphism from the field to an element's anti-cyclic matrix. Then, we have the following relationship where $\mathbf{P}$ below is a simple re-indexing matrix known as a stride permutation (increasing evens followed by increasing odds in $\{0, 1, \ldots, n-1\}$),

$$\mathbf{P}\phi_n(f)\mathbf{P}^t = \begin{bmatrix} \phi_{n/2}(f_0) & \phi_{n/2}(x \cdot f_1) \\ \phi_{n/2}(f_1) & \phi_{n/2}(f_0) \end{bmatrix}.$$

3.3 Sampling G-lattices

For any positive integers $b \geq 2$, $k \geq 1$ and non-negative integer $u < b^k$, we write $[u]_b^k$ for the base-b expansion of u, i.e., the unique vector $(u_0, \ldots, u_{k-1})$ with entries $0 \leq u_i < b$ such that $u = \sum_i u_i b^i$. Typically, $b = 2$ and $[u]_2^k$ is just the k-digits binary representation of u, but larger values of b may be used to obtain interesting efficiency trade-offs. Throughout this section, we consider the values of b and k as fixed, and all definitions and algorithms are implicitly parameterized by them.

In this section we study the so-called *G-lattice sampling problem*, i.e., the problem of sampling the discrete Gaussian distribution on a lattice coset

$$\Lambda_u^{\perp}(\mathbf{g}^t) = \{\mathbf{z} \in \mathbb{Z}^k : \mathbf{g}^t \mathbf{z} = u \bmod q\}$$

where $q \leq b^k$, $u \in \mathbb{Z}_q$, $k = \lceil \log_b q \rceil$, and $\mathbf{g} = (1, b, \ldots, b^{k-1})$. G-lattice sampling is used in many lattice schemes employing a trapdoor. Both schemes with polynomial modulus, like IBE [24, 17, 3, 2], group signatures [62, 70, 63, 51], and others (double authentication preventing and predicate authentication preventing signatures, constraint-hiding PRFs) [20, 33, 34], and schemes with super-polynomial modulus [31, 32, 48, 23, 60, 50, 1] (ABE, watermarking, etc.), use G-lattice sampling.

An efficient algorithm to solve this problem is given in [66] for the special case when $q = b^k$ is a power of the base b. The algorithm, shown in Figure 3.1, is simple. This algorithm reduces the problem of sampling the k-dimensional lattice coset $\Lambda_u^{\perp}(\mathbf{g}^t)$ for $u \in \mathbb{Z}_q$ to the much simpler problem of sampling the *one-dimensional* lattice cosets $u + b\mathbb{Z}$ for $u \in \mathbb{Z}_b$. The simplicity of the algorithm is due to the fact that, when $q = b^k$ is an exact power of b, the lattice $\Lambda^{\perp}(\mathbf{g}^t)$ has

Algorithm 2: SampleG when $q = b^k$.

Input: $(q = b^k, s = b \cdot \omega(\sqrt{\log n}), u)$

Output: $\mathbf{x} \sim D_{\Lambda_u^\perp(\mathbf{g}^t),s}$.

1 **for** $i = 0, \cdots, k-1$ **do**

2 $\quad \mathbf{x}_i \leftarrow D_{b\mathbb{Z}+u,s}$.

3 $\quad u := (u - x_i)/b \in \mathbb{Z}$.

4 **return** $\mathbf{x} = (\mathbf{x}_0, \cdots, \mathbf{x}_{k-1})$.

Figure 3.1. A sampling algorithm for G-lattices when the modulus q is a perfect power of the base b. The algorithm is implicitly parameterized by a base b and dimension k.

a special basis

$$\mathbf{B}_{b^k} = \begin{bmatrix} b & & & & \\ -1 & b & & & \\ & -1 & \ddots & & \\ & & \ddots & b & \\ & & & -1 & b \end{bmatrix}$$

which is sparse, triangular, and with small integer entries. (In particular, its Gram-Schmidt orthogonalization $\widetilde{\mathbf{B}}_{b^k} = b\mathbf{I}$ is a scalar matrix.) As a result, the general lattice sampling algorithm of [61, 46] (which typically requires $O(k^3)$-time preprocessing, and $O(k^2)$ storage and online running time) can be specialized to the much simpler algorithm in Figure 3.1 that runs in linear time $O(k)$, with minimal memory requirements and no preprocessing at all.

We give a specialized algorithm to solve the same sampling problem when $q < b^k$ is an arbitrary modulus. This is needed in many cryptographic applications where the modulus q is typically a prime. As already observed in [66] the lattice $\Lambda^\perp(\mathbf{g}^t)$ still has a fairly simple and

sparse basis matrix

$$\mathbf{B}_q = \begin{bmatrix} b & & & & q_0 \\ -1 & b & & & q_1 \\ & -1 & \ddots & & \vdots \\ & & \ddots & b & q_{k-2} \\ & & & -1 & q_{k-1} \end{bmatrix}$$

where $(q_0, \ldots, q_{k-1}) = [q]_b^k = \mathbf{q}$ is the base-b representation of the modulus q. This basis still has good geometric properties, as all vectors in its (left-to-right) Gram-Schmidt orthogonalization have length at most $O(b)$. So, it can be used with the algorithm of [61, 46] to generate good-quality gaussian samples on the lattice cosets with small standard deviation. However, since the basis is no longer triangular, its Gram-Schmidt orthogonalization is not sparse anymore, and the algorithm of [61, 46] can no longer be optimized to run in linear time as in Figure 3.1. In applications where $q = n^{O(1)}$ is polynomial in the security parameter n, the matrix dimension $k = O(\log n)$ is relatively small, and the general sampling algorithm (with $O(k^2)$ storage and running time) can still be used with an acceptable (albeit significant) performance degradation. However, for larger q this becomes prohibitive in practice. Moreover, even for small q, it would be nice to have an optimal sampling algorithm with $O(k)$ running time, linear in the matrix dimension, as for the exact power case. Here we give such an algorithm, based on the convolution methods of [72], but specialized with a number of concrete technical choices that result in a simple and fast implementation, comparable to the specialized algorithm of [66] for the exact power case.

The reader may notice that the alternating columns of $\mathbf{B}_q$, $\mathbf{b}_1, \mathbf{b}_3, \ldots$ and $\mathbf{b}_2, \mathbf{b}_4, \ldots$, are pair-wise orthogonal. Let us call these sets $\mathbf{B}_1$ and $\mathbf{B}_2$, respectively. Then, another basis for $\Lambda^{\perp}(\mathbf{g}^t)$ is $(\mathbf{B}_1, \mathbf{B}_2, \mathbf{q})$ and this might suggest that the GSO of this basis is sparse. Unfortunately, this leads to a GSO of $(\mathbf{B}_1, \mathbf{B}_2^*, \mathbf{q}^*)$ where $\mathbf{B}_2^*$ is a dense, upper triangular block. Let $\mathbf{b}$ be the $i-th$ vector in $\mathbf{B}_2$. Then, there are $2+i-1$ non-orthogonal vectors to $\mathbf{b}$ preceding it in $\mathbf{B}_1$ and

$\mathbf{B}_2^*$, filling in the upper portion of $\tilde{\mathbf{b}}$.

Overview

The idea is the following. Instead of sampling $\Lambda_u^\perp(\mathbf{g}^t)$ directly, we express the lattice basis $\mathbf{B}_q = \mathbf{TD}$ as the image (under a linear transformation $\mathbf{T}$) of some other matrix $\mathbf{D}$ with simple (sparse, triangular) structure. Next, we sample the discrete gaussian distribution (say, with variance σ^2) on an appropriate coset of $\mathscr{L}(\mathbf{D})$. Finally, we map the result back to the original lattice applying the linear transformation $\mathbf{T}$ to it. Notice that, even if $\mathscr{L}(\mathbf{D})$ is sampled according to a spherical gaussian distribution, the resulting distribution is no longer spherical. Rather, it follows an ellipsoidal gaussian distribution with (scaled) covariance $\sigma^2\mathbf{TT}^t$. This problem is solved using the convolution method of [72], i.e., initially adding a perturbation with complementary covariance $s^2\mathbf{I} - \sigma^2\mathbf{TT}^t$ to the target, so that the final output has covariance $\sigma^2\mathbf{TT}^t + (s^2\mathbf{I} - \sigma^2\mathbf{TT}^t) = s^2\mathbf{I}$. In summary, at a high level, the algorithm performs (at least implicitly) the following steps:

1. Compute the covariance matrix $\Sigma_1 = \mathbf{TT}^t$ and an upper bound r on the spectral norm of $\mathbf{TT}^t$
2. Compute the complementary covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$
3. Sample $\mathbf{p} \leftarrow D_{\Lambda_1,\sigma\sqrt{\Sigma_2}}$, from some convenient lattice Λ_1 using the Cholesky decomposition of Σ_2
4. Compute the preimage $\mathbf{c} = \mathbf{T}^{-1}(\mathbf{u} - \mathbf{p})$
5. Sample $\mathbf{z} \leftarrow D_{\mathscr{L}(\mathbf{D}),-\mathbf{c},\sigma}$
6. Output $\mathbf{u} + \mathbf{Tz}$

The technical challenge is to find appropriate matrices $\mathbf{T}$ and $\mathbf{D}$ that lead to an efficient implementation of all the steps. In particular, we would like $\mathbf{T}$ to be a simple matrix (say,

sparse, triangular, and with small integer entries) so that $\mathbf{T}$ has small spectral norm, and both linear transformations $\mathbf{T}$ and $\mathbf{T}^{-1}$ can be computed efficiently. The matrix $\mathbf{D}$ (which is uniquely determined by $\mathbf{B}$ and $\mathbf{T}$) should also be sparse and triangular, so that the discrete gaussian distribution on the cosets of $\mathscr{L}(\mathbf{D})$ can be efficiently sampled. Finally (and this is the trickiest part in obtaining an efficient instantiation) the complementary covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$ should also have a simple Cholesky decoposition $\Sigma_2 = \mathbf{L}\mathbf{L}^t$ where $\mathbf{L}$ is triangular, sparse and with small entries, so that perturbations can be generated efficiently. Ideally, all matrices should also have a simple, regular structure, so that they do not need to be stored explicitly, and can be computed on the fly with minimal overhead.

In the next subsection we provide an instantiation that satisfies all of these properties. Next, in Subsection 3.3.2 we describe the specialized sampling algorithm resulting from the instantiation, and analyze its correctness and efficiency properties.

3.3.1 Instantiation

In this subsection, we describe a specific choice of linear transformations and matrix decompositions that satisfies all our desiderata, and results in an efficient instantiation of the convolution sampling algorithm on G-lattices.

A tempting idea may be to map the lattice basis $\mathbf{B}_q$ to the basis $\mathbf{B}_{b^k}$, and then use the efficient sampling algorithm from Figure 3.1. However, this does not quite work because it results in a pretty bad transformation $\mathbf{T}$ which has both poor geometrical properties and a dense matrix representation. It turns out that a good choice for a linear transformation $\mathbf{T}$ is given precisely by the matrix $\mathbf{T} = \mathbf{B}_{b^k}$ describing the basis when q is a power of b. We remark that $\mathbf{T}$ is used as a linear transformation, rather than a lattice basis. So, the fact that it equals $\mathbf{B}_{b^k}$ does not seem to carry any special geometric meaning, it just works! In particular, what we do here

should not be confused with mapping $\mathbf{B}_q$ to $\mathbf{B}_{b^k}$. The resulting factorization is

$$\mathbf{B}_q = \begin{bmatrix} 2 & & & & q_0 \\ -1 & 2 & & & q_1 \\ & -1 & \ddots & & \vdots \\ & & \ddots & 2 & q_{k-2} \\ & & & -1 & q_{k-1} \end{bmatrix} = \begin{bmatrix} 2 & & & & \\ -1 & 2 & & & \\ & -1 & \ddots & & \\ & & \ddots & 2 & \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & & & & d_0 \\ & 1 & & & d_1 \\ & & \ddots & & \vdots \\ & & & 1 & d_{k-2} \\ & & & & d_{k-1} \end{bmatrix} = \mathbf{B}_{b^k}\mathbf{D}$$

where the entries of the last column of $\mathbf{D}$ are defined by the recurrence $d_i = \frac{d_{i-1}+q_i}{b}$ with initial condition $d_{-1} = 0$. Notice that all the d_i are in the range $[0,1)$, and $b^{i+1} \cdot d_i$ is always an integer. In some sense, sampling from $\mathscr{L}(\mathbf{D})$ is even easier than sampling from $\mathscr{L}(\mathbf{B}_{b^k})$ because the first $k-1$ columns of $\mathbf{D}$ are orthogonal and the corresponding coordinates can be sampled independently in parallel. (This should be contrasted with the sequential algorithm in Figure 3.1.)

We now look at the geometry and algorithmic complexity of generating perturbations. The covariance matrix of $\mathbf{T} = \mathbf{B}_{b^k}$ is given by

$$\Sigma_1 = \mathbf{B}_{b^k}\mathbf{B}_{b^k}^t = \begin{bmatrix} b^2 & -b & & & & \\ -b & (b^2+1) & -b & & & \\ & \ddots & \ddots & \ddots & & \\ & & -b & (b^2+1) & -b & \\ & & & -b & (b^2+1) \end{bmatrix}.$$

The next step is to find an upper bound r^2 on the spectral norm of Σ_2, and compute the Cholesky decomposition $\mathbf{L}\mathbf{L}^t$ of the complementary covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$. By the Gershgorin circle theorem, all eigenvalues of Σ_1 are in the range $(b \pm 1)^2$. So, we may set $r = b+1$. Numerical computations also suggest that this choice of r is optimal, in the sense that the spectral norm of Σ_1 approaches $b+1$ as k tends to infinity. The Cholesky decomposition is customarily defined by taking $\mathbf{L}$ to be a *lower* triangular matrix. However, for sampling purposes, an upper

triangular $\mathbf{L}$ works just as well. It turns out that using an upper triangular $\mathbf{L}$ in the decomposition process leads to a much simpler solution, where all (squared) entries have a simple, closed form expression, and can be easily computed on-line without requiring any preprocessing computation or storage. (By contrast, numerical computations suggest that the standard Cholesky decomposition with lower triangular $\mathbf{L}$ is far less regular, and even precomputing it requires exponentially higher precision arithmetic than our upper triangular solution.) So, we let $\mathbf{L}$ be an upper triangular matrix, and set $r = b+1$.

For any r, the perturbation's covariance matrix $\Sigma_2 = r^2\mathbf{I} - \Sigma_1$ has Cholesky decomposition $\Sigma_2 = \mathbf{L}\cdot\mathbf{L}^t$ where $\mathbf{L}$ is the sparse upper triangular matrix defined by the following equations:

$$\mathbf{L} = \begin{bmatrix} l_0 & h_1 & & & \\ & l_1 & h_2 & & \\ & & \ddots & \ddots & \\ & & & & h_{k-1} \\ & & & & l_{k-1} \end{bmatrix} \quad \text{where} \quad \begin{aligned} l_0^2 + h_1^2 &= r^2 - b^2 \\ l_i^2 + h_{i+1}^2 &= r^2 - (b^2+1) \quad (i = 1,\ldots,k-2) \\ l_{k-1}^2 &= r^2 - (b^2+1) \\ l_i h_i &= b \qquad\qquad\qquad (i = 1,\ldots,k-1) \end{aligned}$$

It can be easily verified that these equations have the following simple closed form solution:

$$r = b+1, \quad l_0^2 = b\left(1+\frac{1}{k}\right)+1, \quad l_i^2 = b\left(1+\frac{1}{k-i}\right), \quad h_{i+1}^2 = b\left(1-\frac{1}{k-i}\right) \tag{3.1}$$

We observe that also the inverse transformation $\mathbf{B}_{b^k}^{-1}$ has a simple, closed-form solution: the ith column of $\mathbf{B}_{b^k}^{-1}$ equals $(0,\cdots,0,\frac{1}{b},\ldots,(\frac{1}{b})^{k-i})$. Notice that this matrix is not sparse, as it has $O(k^2)$ nonzero entries. However, there is no need to store it and the associated transformation can still be computed in linear time by solving the sparse triangular system $\mathbf{Tx} = \mathbf{b}$ by back-substitution.

3.3.2 The Algorithm

The sampling algorithm, SAMPLEG, is shown in Figure 3.2. It takes as input a modulus q, an integer variance s, a coset u of $\Lambda^{\perp}(\mathbf{g}^t)$, and outputs a sample statistically close to $D_{\Lambda_u^{\perp}(\mathbf{g}^t),s}$. SAMPLEG relies on subroutines PERTURB and SAMPLED where PERTURB(σ) returns a perturbation, $\mathbf{p}$, statistically close to $D_{\mathscr{L}(\Sigma_2),\sigma\cdot\sqrt{\Sigma_2}}$, and SAMPLED$(\sigma,\mathbf{c})$ returns a sample $\mathbf{z}$ such that $\mathbf{Dz}$ is statistically close to $D_{\mathscr{L}(\mathbf{D}),-\mathbf{c},\sigma}$.

Both PERTURB and SAMPLED are instantiations of the randomized nearest plane algorithm [61, 46]. Consequently, both algorithms rely on a subroutine SAMPLEZ$_t(\sigma,c,\sigma_{max})$ which returns a sample statistically close to one-dimensional discrete gaussian with it a tail-cut t, $D^t_{\mathbb{Z},\sigma,c}$ over the *fixed* support of $\mathbb{Z}\cap[c-t\cdot\sigma_{max},c+t\cdot\sigma_{max}]$. We fix the support of all one dimensional discrete gaussians for compatibility with ML distance. In addition, we only feed SAMPLEZ centers $c\in[0,1)$ since we can always shift by an integer.

Storage

The scalars c_i in SAMPLEG, representing $\mathbf{c}=\mathbf{B}_{b^k}^{-1}(\mathbf{u}-\mathbf{p})$, and d_i in SAMPLED, representing the last column of $\mathbf{D}$, are rational numbers of the form x/b^i for a small integer x and $i\in[k]$. The numbers l_i,h_i are positive numbers of magnitude less than $\sqrt{2b+1}$.

A naive implementation of the algorithms store floating point numbers c_i, d_i, h_i, and l_i for a total storage of $4k$ floating point numbers. However, this can be adapted to constant time storage since they are determined by simple recurrence relations (c_i, d_i) or simple formulas (h_i, l_i).

Time Complexity

Assuming constant time sampling for SAMPLEZ and scalar arithmetic, SAMPLEG runs in time $O(k)$. Now let us consider all operations: there are $6k$ integer additions/subtractions, $3k+2$ integer multiplications, $3(k+1)$ floating point divisions, $2k$ floating point multiplications, and $2k$ floating point additions. The analysis below shows we can use double precision floating point numbers for most applications.

Algorithm 3: SampleG

Input: $(s = b \cdot \omega(\sqrt{\log n}), \mathbf{u} = [u]_b^k, \mathbf{q} = [q]_b^k$

Output: $\mathbf{x} \sim D_{\Lambda_u^\perp(\mathbf{g}^t),s}$.

$\sigma := s/(b+1)$.
$\mathbf{p} \leftarrow$ PERTURB(σ).
for $i = 0, \cdots, k-1$ **do**
  $c_i := (c_{i-1} + u_i - p_i)/b$.
$\mathbf{z} \leftarrow$ SAMPLED$(\sigma, \mathbf{c})$.
for $i = 0, \cdots, k-2$ **do**
  $t_i := b \cdot z_i - z_{i-1} + q_i \cdot z_{k-1} + u_i$.
$t_{k-1} := q_{k-1} \cdot z_{k-1} - z_{k-2} + u_{k-1}$.
return t.

Algorithm 4: Perturb

Input: σ

Output: $\mathbf{p} \sim D_{\mathscr{L}(\Sigma_2),\Sigma_2}$

$\beta := 0$.
for $i = 0, \cdots, k-1$ **do**
  $c_i := \beta/l_i$, and $\sigma_i := \sigma/l_i$.
  $z_i \leftarrow \lfloor c_i \rfloor +$ SAMPLE$\mathrm{Z}_t(\sigma_i, \lfloor c_i \rceil_{[0,1)}, s)$.
  $\beta := -z_i h_i$.
$p_0 := (2b+1)z_0 + bz_1$.
for $i = 1, \cdots, k-1$ **do**
  $p_i := b(z_{i-1} + 2z_i + z_{i+1})$.
return p.

Algorithm 5: SampleD

Input: $(\sigma, \mathbf{c})$.

Output: $\mathbf{z} \in \mathbb{Z}^k : \mathbf{Dz} \sim D_{\mathscr{L}(\mathbf{D}),-\mathbf{c},\sigma}$.

$z_{k-1} \leftarrow \lfloor -c_{k-1}/d_{k-1} \rfloor$.
$z_{k-1} \leftarrow z_{k-1} +$ SAMPLE$\mathrm{Z}_t(\sigma/d_{k-1}, \lfloor -c_{k-1}/d_{k-1} \rceil_{[0,1)}, s)$.
$\mathbf{c} := \mathbf{c} - z_{k-1}\mathbf{d}$.
for $i \in \{0, \cdots, k-2\}$ **do**
  $z_i \leftarrow \lfloor -c_i \rfloor +$ SAMPLE$\mathrm{Z}_t(\sigma, \lfloor -c_i \rceil_{[0,1)}, s)$.
return z.

Figure 3.2. Any scalar with an index out of range is 0, i.e. $c_{-1} = z_{-1} = z_k = 0$. SAMPLE$\mathrm{Z}_t(\sigma, c, \sigma_{max})$ is a discrete gaussian over $\mathbb{Z}$ exactly or approximately with centers in $[0,1)$ and a fixed truncated support $\mathbb{Z} \cap [c - t \cdot \sigma_{max}, c + t \cdot \sigma_{max}]$ (t is the tail-cut parameter). We denote $x - \lfloor x \rfloor$ as $\lfloor x \rceil_{[0,1)}$.

Statistical Analysis and Floating Point Precision

We now perform a statistical analysis on SAMPLEG with a perfect one-dimensional sampler (and no tail-bound), then with a tail-bounded imperfect sampler in terms of ML distance. This allows us to measure loss in concrete security. We direct the reader to [68, Section 3] for more details on the ML distance and a complete concrete security analysis.

The following lemma is needed in order to make sense of the "Σ_3 condition" in Theorem 3.2.1.

Lemma 3.3.1 *Let Σ_3 be defined by $\Sigma_3^{-1} = \frac{(b+1)^2}{s^2}[\Sigma_1^{-1} + [(b+1)^2\mathbf{I} - \Sigma]^{-1}]$, then its eigenvalues are $\Theta(s^2/b)$. Moreover, if λ_i is the $i-th$ eigenvalue of Σ_1, then the $i-th$ eigenvalue of Σ_3 is $(s/[b+1])^2 \cdot \frac{\lambda_i[(b+1)^2-\lambda_i]}{(b+1)^2}$.*

Proof: Let $\Sigma_1 = \mathbf{Q}^t\mathbf{D}\mathbf{Q}$ be its diagonalization. Then, $\Sigma_1^{-1} = \mathbf{Q}^t\mathbf{D}^{-1}\mathbf{Q}$ and the rest follows from algebraic manipulations of the individual eigenvalues along with the Gershgorin circle theorem on Σ_1. □

Let $C_{\varepsilon,k} = \sqrt{\log(2k(1+1/\varepsilon))/\pi}$. Now we can easily bound s from below. We need the following three conditions for s: $s \geq (b+1)\eta_\varepsilon(\mathbf{D})$, $\sqrt{\Sigma_3} \geq \eta_\varepsilon(\Sigma_2)$, and $s \geq (b+1)\eta_\varepsilon(\mathbf{L})$. The middle condition determines s with a lower bound of $s \geq \sqrt{2b} \cdot (2b+1) \cdot C_{\varepsilon,k}$ (the last two conditions both have $s = \Omega(b^{1.5} \cdot C_{\varepsilon,k})$).

Corollary 3.3.1 *Fix $0 < \varepsilon \leq 1/2$ and let $s \geq \sqrt{2b} \cdot (2b+1) \cdot C_{\varepsilon,k}$. Then,* SAMPLEG *returns a perturbation within a statistical distance $\Theta(k\hat{\varepsilon})$ from $D_{\Lambda_u^\perp(\mathbf{g}^t),s}$ for any $q < b^k$ when* PERTURB *and* SAMPLED *use a perfect one-dimensional sampler,* SAMPLEZ*. In addition, the Rényi divergence of order infinity of $D_{\Lambda_u^\perp(\mathbf{g}^t),s}$ from* SAMPLEG *with a perfect one-dimensional sampler is less than or equal to $1 + \Theta(k\hat{\varepsilon})$.*

The statistical distance bound of $\Theta(k\hat{\varepsilon})$ results in about a loss of $\log\log q$ bits in security if $\varepsilon = 2^{-\kappa}$ for a security parameter κ by [68, Lemma 3.1]. (The multiplicative factor of k comes from the randomized nearest plane algorithm's analysis: see [46, Theorem 4.1].)

Next, we turn to the ML distance for a tighter analysis on the bits of security lost in using SAMPLEG with an imperfect one-dimensional sampler. Since the centers, c, and variances, s, given to SAMPLEZ are computed from two or three floating point computations, we assume both $\bar{c}$ and $\bar{s}$ are within a relative error of 2^{-m} of c and s.

Proposition 3.3.1 *Fix an $\varepsilon > 0$ and let $s \geq (b+1) \cdot \eta_\varepsilon(\mathbb{Z})$. For any one-dimensional sampler* $\text{SAMPLEZ}_t(\bar{\sigma}, \bar{c}, s)$ *that takes as inputs approximated centers $\bar{c} \in [0,1)$ and variances $\bar{\sigma} \in [s/(b+1), s \cdot b/(b+1)]$ represented as floating point numbers with mantissa length m,*

$$\Delta_{\text{ML}}(\text{SAMPLEG}^{D^t_{\mathbb{Z},\sigma,c}}, \text{SAMPLEG}^{\text{SAMPLEZ}_t(\bar{\sigma},\bar{c})}) \leq$$
$$2k[O(b^2t^2 2^{-m}) + \max_{\bar{\sigma},\bar{c}} \Delta_{\text{ML}}(\text{SAMPLEZ}_t(\bar{\sigma}, \bar{c}, s), D^t_{\mathbb{Z},\bar{\sigma},\bar{c}})].$$

Before we begin the proof, we note that $d_{k-1} = q/b^k \in [1/b, 1]$ since $k = \lceil \log_b q \rceil$. This implies that every variance fed to SAMPLEZ is in the range $[s/(b+1), s \cdot b/(b+1)] \subseteq [s/(b+1), s]$. We restrict all truncated one-dimensional discrete gaussians to $\mathbb{Z} \cap [c - t \cdot s, c + t \cdot s]$ since it is unclear when $\mathbb{Z} \cap [c - t \cdot \sigma, c + t \cdot \sigma] = \mathbb{Z} \cap [c - t \cdot \bar{\sigma}, c + t \cdot \bar{\sigma}]$ when using floating point variances $\bar{\sigma}$. The ML distance is undefined when these two sets are not equal.

Proof: First, we use the triangle inequality on ML distance in order to pair together terms for an easier analysis.

$$\Delta_{\text{ML}}(\text{SAMPLEG}^{D^t_{\mathbb{Z},\sigma,c}}, \text{SAMPLEG}^{\text{SAMPLEZ}_t(\bar{\sigma},\bar{c},s)}) \leq$$
$$\Delta_{\text{ML}}(\text{SAMPLEG}^{D^t_{\mathbb{Z},\sigma,c}}, \text{SAMPLEG}^{D^t_{\mathbb{Z},\bar{\sigma},c}}) + \Delta_{\text{ML}}(\text{SAMPLEG}^{D^t_{\mathbb{Z},\bar{\sigma},c}}, \text{SAMPLEG}^{D^t_{\mathbb{Z},\bar{\sigma},\bar{c}}}) +$$
$$\Delta_{\text{ML}}(\text{SAMPLEG}^{D^t_{\mathbb{Z},\bar{\sigma},\bar{c}}}, \text{SAMPLEG}^{\text{SAMPLEZ}_t(\bar{\sigma},\bar{c},s)}).$$

Next, we use the data processing inequality on ML distance where we treat SAMPLEG as a function of $2k$ correlated samples from a one-dimensional discrete gaussian sampler. From [Lemma 3.2, MW17], we get the following inequality:

$\Delta_{\mathrm{ML}}(\mathrm{SAMPLEG}^{D^t_{\mathbb{Z},\sigma,c}}, \mathrm{SAMPLEG}^{\mathrm{SAMPLEZ}_t(\bar{\sigma},\bar{c},s)}) \leq$

$2k \cdot \max_{\sigma_i,c_i}[\Delta_{\mathrm{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}) + \Delta_{\mathrm{ML}}(D^t_{\mathbb{Z},\bar{\sigma}_2,c_2}, D^t_{\mathbb{Z},\bar{\sigma}_2,\bar{c}_2}) +$

$\Delta_{\mathrm{ML}}(D^t_{\mathbb{Z},\bar{\sigma}_3,\bar{c}_3}, \mathrm{SAMPLEZ}_t(\bar{\sigma}_3,\bar{c}_3,s))]$.

The maximum is taken over all $c_i \in [0,1)$ and $\sigma_i \in [s/(b+1), s\cdot b/(b+1)]$. Let $\mathbb{Z}^t = \mathbb{Z} \cap [c - t\cdot s, c + t\cdot s]$. We bound $\max_{\sigma_1,c_1} \Delta_{\mathrm{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1})$ as follows:

$$\begin{aligned}\max_{\sigma_1,c_1} \Delta_{\mathrm{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}) &= \max_{\sigma_1,c_1,x\in\mathbb{Z}^t} |\ln D^t_{\mathbb{Z},\sigma_1,c_1}(x) - \ln D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}(x)| \\ &= \max_{\sigma_1,c_1,x\in\mathbb{Z}^t} \left|\pi(x-c)^2\left[\frac{1}{\sigma_1^2} - \frac{1}{\bar{\sigma}_1^2}\right] + \ln\frac{\rho_{\bar{\sigma}_1,c_1}(\mathbb{Z})}{\rho_{\sigma_1,c_1}(\mathbb{Z})}\right|.\end{aligned}$$

Since $\sigma_1, \bar{\sigma}_1 \geq \eta_\varepsilon(\mathbb{Z})$, we can approximate $\rho_{\sigma_1,c}(\mathbb{Z}) \in [(1-\varepsilon)^2, (1+\varepsilon)^2]\cdot\sigma$ and $\rho_{\bar{\sigma}_1,c}(\mathbb{Z}) \in [(1-\varepsilon)^2, (1+\varepsilon)^2]\cdot\bar{\sigma}$. Using the bound on the relative error of $\bar{\sigma}_1$ ($\bar{\sigma}_1 \in [1-2^{-m}, 1+2^{-m}]\cdot\sigma_1$), we can bound the expression with a simplified form below.

$$\begin{aligned}\max_{\sigma_1,c_1} \Delta_{\mathrm{ML}}(D^t_{\mathbb{Z},\sigma_1,c_1}, D^t_{\mathbb{Z},\bar{\sigma}_1,c_1}) \leq & \\ \max_{\sigma_1}\left|\pi\frac{t^2s^2}{\sigma_1^2}\cdot\frac{\bar{\sigma}_1^2-\sigma_1^2}{\sigma_1^2} + 2\hat{\varepsilon} + 2\hat{^-}m\right| \leq & \\ \pi t^2(b+1)^2(2^{-m+1}+2^{-2m}) + \hat{\varepsilon} + 2\hat{^-}m. &\end{aligned}$$

The proof for $\Delta_{\mathrm{ML}}(D^t_{\mathbb{Z},\bar{\sigma}_2,c_2}, D^t_{\mathbb{Z},\bar{\sigma}_2,\bar{c}_2})$ is nearly identical except we get a term linear in t, yielding a bound of $O(t\cdot 2^{-m})$. □

Assuming a cryptosystem using a perfect sampler for $D_{\Lambda^\perp_{\bar{u}}(g^t),s}$ has κ bits of security, we can combine the results of Corollary 3.3.1, Proposition 3.3.1, and [68, Lemma 3.3] to conclude that swapping $D_{\Lambda^\perp_{\bar{u}}(g^t),s}$ with SAMPLEG yields about $\kappa - 2\log(tb^2) - 3\log\log q - 5$ bits of security when $m = \kappa/2$, $\Delta_{\mathrm{ML}}(\mathrm{SAMPLEZ}_t(\bar{s},\bar{c}), D^t_{\mathbb{Z},\bar{s},\bar{c}}) < 2^{-\kappa/2}$, and $\varepsilon = 2^{-\kappa}$.

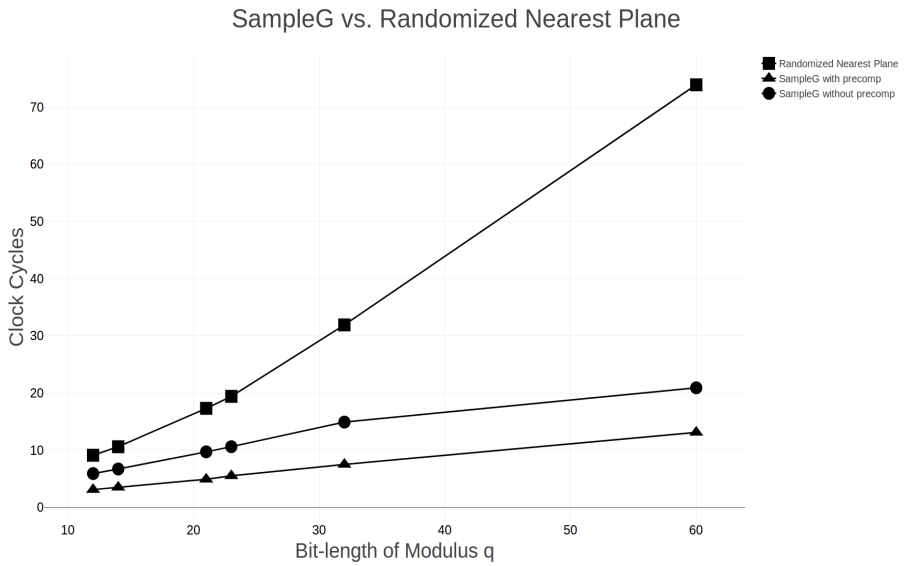

Figure 3.3. Measured clock cycles with $q = \{4.1 \cdot 10^3, 1.22 \cdot 10^5, 1.68 \cdot 10^7, 8.38 \cdot 10^7, 4.30 \cdot 10^9, 9 \cdot 10^{18}\}$ and $s = 100$ averaged over 100,000 runs. The clock cycles for the last three moduli are $\{19.4, 31.9, 73.9\}$ for GPV and $\{5.5, 7.5, 13.1\}$ for SAMPLEG with pre-computation.

3.3.3 Implementation and Comparison

In this subsection, we compare simple implementations of both SAMPLEG and the generic randomized nearest plane algorithm [46, Section 4] used in the G-lattice setting. The implementations were carried out in C++ with double precision floating point numbers for non-integers on an Intel i7-2600 3.4 GHz CPU. Clock cycles were measured with the "time.h" library and the results are charted in Figure 2.3.

The one-dimensional sampler, SAMPLEZ, was an instantiation of a discrete version of Karney's sampler [59], which is a modified rejection sampler. The moduli q were chosen from the common parameters subsection of [57, Section 4.2], in addition to an arbitrary 60-bit modulus. Most practical schemes require no more than a 30-bit modulus [16] for lattice dimension (n) up to 1024. More advanced schemes however, like ABE-encryption [19, 31], and predicate encryption [49], require a super-polynomial modulus often 90 or more bits (assuming the circuits in the ABE and predicate schemes are of log-depth).

For the generic, randomized nearest plane sampler, we pre-computed and stored the

Gram-Schmidt orthogonalization of the basis $\mathbf{B}_q$ and we only counted the clock cycles to run the algorithm thereafter. We had two versions of SAMPLEG: the first was the algorithm as-is, and the second would store pre-computed perturbations from PERTURB(σ), one for each G-lattice sample. This version of SAMPLEG with pre-computation saved about a factor of two in clock cycles.

3.4 Perturbation Sampling in Cyclotomic Rings

The lattice preimage sampling algorithm of [66] requires the generation of $n(2+\log q)$-dimensional gaussian perturbation vectors $\mathbf{p}$ with covariance $\Sigma_p = s^2 \cdot \mathbf{I} - \alpha^2 \mathbf{T} \cdot \mathbf{T}^t$ where $\mathbf{T} \in \mathbb{Z}^{(2+\log q)n \times n \log q}$ is a matrix with small entries serving as a lattice trapdoor, α is a small constant factor and s is an upper bound on the spectral norm of $\alpha\mathbf{T}$. In [66] this is accomplished using the Cholesky factorization of Σ_p, which takes $O(n \log q)^3$ pre-computation and $O(n \log q)^2$ storage and running time.

The trapdoor matrix $\mathbf{T}$ of [66] has some additional structure: $\mathbf{T}^t = [\bar{\mathbf{T}}^t, \mathbf{I}]$ for some $\bar{\mathbf{T}} \in \mathbb{Z}^{2n \times n \log q}$. Moreover, when working with algebraic lattices, $\bar{\mathbf{T}} = \phi_n(\tilde{\mathbf{T}})$ is the image (under a ring embedding $\phi_n \colon R_n \to \mathbb{Z}^{n \times n}$) of some matrix $\tilde{\mathbf{T}} \in R_n^{2 \times \log q}$ with entries in a ring R_n of rank n. (Most commonly, $R_n = \mathscr{O}_{2n} = \mathbb{Z}[x]/(x^n+1)$ is the ring of integers of the $(2n)$th cyclotomic field $\mathscr{K}_{2n}$ for $n = 2^k$ a power of two.) In [16] it is observed that, using the sparsity of Σ_p, the preprocessing storage and on-line computation cost of noise perturbation reduce to $O(n^2 \log q)$.[3] This is a factor $\log q$ improvement over a generic implementation, but it is still quadratic in the main security parameter n. This can be a significant improvement in practice, but the overall cost of the algorithm remains substantial. When using generic trapdoors $\bar{\mathbf{T}} \in \mathbb{Z}^{2n \times n \log q}$, there is little hope to improve the running time below $O(n^2 \log q)$, because just reading the matrix $\bar{\mathbf{T}}$ takes this much time. However, when using algebraic lattices, the trapdoor $\bar{\mathbf{T}} = \phi_n(\tilde{\mathbf{T}})$ admits a compact representation $\tilde{\mathbf{T}}$ consisting of only $2n \log q$ integers, so one may hope to reduce the running time to linear or quasi-linear in n.

[3] Sparsity also reduces the preprocessing running time to $O(\log q \cdot n^2 + n^3) = O(n^3)$, but still cubic in n.

In this section we give an alternative algorithm to generate integer perturbation vectors $\mathbf{p}$ with covariance Σ_p when $\bar{\mathbf{T}} = \phi_n(\tilde{\mathbf{T}})$. Our algorithm takes full advantage of the ring structure of R_n, compactly representing Σ_p and all other matrices generated during the execution of the algorithm as the image of matrices with entries in the ring R_n. In particular, similarly to [40, 41], our algorithm has time and space complexity quasi-linear in n, but does not require any preprocessing/storage. The algorithm can be expressed in a modular way as the combination of three steps:

1. First, the problem of sampling a $O(n \log q)$-dimensional integer vectors $\mathbf{p}$ with covariance Σ_p is reduced to the problem of sampling a $2n$-dimensional integer vector with covariance expressed by a 2×2 matrix over R_n.

2. Next, the problem of sampling an integer vector with covariance in $R_n^{2\times 2}$ is reduced to sampling two n-dimensional integer vectors, each with a covariance expressed by a single ring element in R_n.

3. Finally, if $n > 1$, the sampling problem with covariance in R_n is reduced to sampling an n-dimensional perturbation with covariance expressed by a 2×2 matrix over the smaller ring $R_{n/2}$.

Iterating the last two steps $\log n$ times reduces the original problem to sampling in $R_1 = \mathbb{Z}$. Details about each step are given in the next subsections. We remark that the algorithm is described as a recursive procedure only for simplicity of presentation and analysis, and it can be implemented just as easily using a simple nested loop, similarly to many FFT-like algorithms.

3.4.1 Discrete Perturbation Algorithm for Power of Two Cyclotomics

In this subsection we present the perturbation algorithm algorithm which produces $n(2+\log q)$-dimensional perturbations from a discrete gaussian on $\mathbb{Z}^{n(2+\log q)}$ in time $\tilde{O}(n \log q)$.

The entry point of the algorithm is the SAMPLEPZ procedure, which takes as input two integer parameters n, q, matrices $\tilde{\mathbf{T}} \in R_n^{2\times \log q}$, $\Sigma_2 \in R_n^{2\times 2}$, and three positive real numbers

$s^2, \alpha^2, z = (\alpha^{-2} - s^{-2})^{-1}$, and is expected to produce an $n(2+\log q)$-dimensional vector $\mathbf{p}$ with (non-spherical) discrete gaussian distribution $D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}}$ of covariance

$$\begin{aligned}\Sigma_p &= s^2 \cdot \mathbf{I} - \alpha^2 \begin{bmatrix} \phi_n(\tilde{\mathbf{T}}) \\ \mathbf{I} \end{bmatrix} \cdot \begin{bmatrix} \phi_n(\tilde{\mathbf{T}})^t & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \Sigma_2 & -\alpha^2 \phi_n(\tilde{\mathbf{T}}) \\ -\alpha^2 \phi_n(\tilde{\mathbf{T}})^t & (s^2 - \alpha^2)\mathbf{I} \end{bmatrix}.\end{aligned}$$

The algorithm calls two subroutines:

- SAMPLEZ$(s^2 - \alpha^2)$ which samples a one-dimensional discrete gaussian variable of variance $s^2 - \alpha^2$ centered at 0, and can be implemented using any standard technique, and
- SAMPLE2Z(a, b, d), which, on input three ring elements a, b, d compactly describing a positive definite matrix
$$\Sigma_2 = \begin{bmatrix} \phi_n(a) & \phi_n(b) \\ \phi_n(b)^t & \phi_n(d) \end{bmatrix},$$
is expected to sample a $(2n)$-dimensional vector $p \leftarrow D_{\mathbb{Z}^{2n}, \sqrt{\Sigma_2}}$.

In turn, SAMPLE2Z (also described in Figure 3.4) makes use of a procedure SAMPLEFZ(f) which on input a ring element f with positive definite $\phi_n(f)$, returns a sample $p \leftarrow D_{\mathbb{Z}^n, \sqrt{\phi_n(f)}}$.

Algorithm 6: SamplePz

Input: $(n, q, s, \alpha, \tilde{\mathbf{T}}, \Sigma_2, z)$.

Output: $(\mathbf{p}, \mathbf{q}) \sim D_{\mathbb{Z}^{n(2+\log q)}, \Sigma_p}$.

for $i = 0, \cdots, n\log q - 1$ **do**
  $q_i \leftarrow \text{SAMPLEZ}(s^2 - \alpha^2)$.
$(c_0, c_1) := -\frac{\alpha^2}{s^2-\alpha^2}\tilde{\mathbf{T}}\mathbf{q}$.
$c'(x) := c_0(x^2) + x \cdot c_1(x^2))$.
$\mathbf{p} \leftarrow \text{SAMPLE2Z}(a, b, d, c')$.
return $(\mathbf{p}, \mathbf{q})$.

Algorithm 7: Sample2z

Input: (a, b, c, d).

Output: $(q_0, q_1) \sim D_{\mathbb{Z}^{2n}, \Sigma_2}$.

Let $c(x) = c_0(x^2) + x \cdot c_1(x^2)$.
$q_1 \leftarrow \text{SAMPLEFZ}(d, c_1)$.
$c_0 := c_0 + bd^{-1}(q_1 - c_1)$.
$q_0 \leftarrow$ SAMPLEFZ$(a - bd^{-1}b^*, c_0)$.
return (q_0, q_1).

Algorithm 8: SampleFz

Input: (f, c).

Output: $q \sim D_{\mathbb{Z}^n, c, \phi_n(f)}$.

if $\dim(f) = 1$ **then**
  return SAMPLEZ(f, c).
else
  Let $f(x) = f_0(x^2) + x \cdot f_1(x^2)$.
  $(q_0, q_1) \leftarrow \text{SAMPLE2Z}(f_0, f_1, f_0, c)$.
  let $q(x) = q_0(x^2) + x \cdot q_1(x^2)$.
return q.

Figure 3.4. Sampling algorithm SAMPLEPZ for integer perturbations where $\mathbf{T} = \phi_n(\tilde{\mathbf{T}})$ is a compact trapdoor over a power of two cyclotomic ring. Note, $\tilde{\mathbf{T}}_i$ is a *row* vector over R_n for each $i \in \{0, 1\}$. The scalar $z = (\alpha^{-2} - s^{-2})^{-1}$.

Efficiency

Multiplications are done in the field $\mathscr{K}_i$, for an element's dimension $i \in \{1,2,\ldots,2n\}$, in time $\Theta(i \log i)$ by using the Chinese remainder transform (*CRT*) [65].

By treating scalar arithmetic as constant time, SAMPLEPZ has a time complexity of $\Theta(n \log n \log q)$ because the transformation by $\tilde{\mathbf{T}}$ is $\Theta(n \log n \log q)$ and SAMPLEFZ has complexity $\Theta(n \log^2 n)$ (represented by the recurrence $R(n) = 2R(n/2) + 2\log n/2 + 4.5n$). The algorithm requires $2n \log q$ scalar storage for the trapdoor $\tilde{\mathbf{T}}$.

Note, SAMPLEFZ is even more efficient, $\Theta(n \log n)$, if one were to store the polynomials in $\mathscr{K}_i$ in the canonical embedding (Fourier domain). One would change SAMPLEPZ to give SAMPLE2Z the Fourier/canonical representations of a, b, d, c_0, c_1 and perform an inverse CRT/FFT on $\mathbf{p} = (\mathbf{p}_0, \mathbf{p}_1)$. This allows us to use the FFT's butterfly transformation to convert to the Fourier representation of $f(x) = f_0(x^2) + x f_1(x^2) \in \mathscr{K}_{2n}$ to the Fourier representation of $f_0, f_1 \in \mathscr{K}_n$ and multiplication/inversion is now linear time (we would only invert the non-zero entries in the Fourier domain since this corresponds to pulling back to the field, inverting, then pushing forward to the cyclic ring via the embedding given by the Chinese remainder theorem) [41, Lemma 1]. (Moving from the canonical embedding to the FFT domain is linear time since we place zeros for the non-primitive roots of unity [41, Section A.2].) This, however, does not change the asymptotic time complexity of SAMPLEPZ since generating $\mathbf{q}$ in the canonical embedding is now $\Theta(n \log n \log q)$.

Correctness

One would use Peikert's convolution theorem, Theorem 3.2.1, in an initial attempt to prove the correctness of the algorithms in Figure 3.4. However, this would only ensure the correctness of the marginal distributions of $\mathbf{p}$ in SAMPLEPZ and q_0 in SAMPLE2Z and not their respective joint distributions, $(\mathbf{p}, \mathbf{q})$ and (q_0, q_1). Even if it were enough, tracking the Σ_3 condition in Theorem 3.2.1 through the recursive calls of the algorithms above is tedious. Instead, we derive a convolution lemma without a Σ_3 condition for the joint distribution of our discrete

gaussian convolutions on the simple lattice $\mathbb{Z}^n$.

First, we show the gaussian function $\rho_{\sqrt{\Sigma}}(\cdot)$ factors in a useful manner with respect to a Schur complement decomposition.

Lemma 3.4.1 *Let* $\Sigma = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^t & \mathbf{D} \end{bmatrix} \succ \mathbf{0}$ *be a positive definite with* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *and* $\mathbf{D} \in \mathbb{R}^{m\times m}$ *and* $\Sigma/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^t$ *is* $\mathbf{D}$*'s Schur complement, and let* $\mathbf{x}_1 \in \mathbb{R}^n$ *and* $\mathbf{x}_2 \in \mathbb{R}^m$ *be arbitrary. Then, the gaussian function* $\rho_{\sqrt{\Sigma}}(\mathbf{x})$ *factors as* $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbf{x}_1 - \mathbf{B}\mathbf{D}^{-1}\mathbf{x}_2) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbf{x}_2) = \rho_{\sqrt{\Sigma}}(\mathbf{x})$ *where* $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^{n+m}$.

Proof:(Sketch) This is seen through defining the inverse of Σ in terms of $\Sigma/\mathbf{D}$ and writing out $\rho_{\sqrt{\Sigma}}(\mathbf{x})$ in terms of $\Sigma/\mathbf{D}$. The matrix factorization

$$\Sigma = \begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \Sigma/\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{B}^t & \mathbf{I} \end{bmatrix}$$

yields the formula for Σ^{-1} needed to show the result. □

A consequence of the above lemma is that the gaussian sum $\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m})$ expands in terms of the gaussian functions $\rho_{\sqrt{\mathbf{D}}}(\cdot)$ and $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\cdot)$,

$$\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m}) = \sum_{\mathbf{y}_2 \in \mathbb{Z}^m} \rho_{\sqrt{\mathbf{D}}}(\mathbf{y}_2) \cdot \rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{B}\mathbf{D}^{-1}\mathbf{y}_2).$$

We will use the following lemma for the correctness proof. It states that if a discrete gaussian on the integer lattice is wide enough in its slimmest direction, then the lower dimensional discrete gaussians with covariance shaped with principal submatrices of the original are wide enough on their respective $\mathbb{Z}^{n'}$s.

Lemma 3.4.2 *Let* $\varepsilon > 0$, $\Sigma \succ 0$ *be a positive definite matrix in* $\mathbb{R}^{n\times n}$*, and let* $I_0 \subset [n]$ *be an arbitrary, non-empty subset. If* $\Sigma \succeq \eta_\varepsilon^2(\mathbb{Z}^n)$*, then* $\Sigma[I_0] \succeq \eta_\varepsilon^2(\mathbb{Z}^{|I_0|})$ *and* $\Sigma/\bar{I}_0 \succeq \eta_\varepsilon^2(\mathbb{Z}^{n-|I_0|})$ *for any principal submatrix - Schur complement pair,* $(\Sigma[I_0], \Sigma/\bar{I}_0)$*, of* Σ.

Proof: Note, a consequence of $\Sigma \succeq \eta_\varepsilon^2(\mathbb{Z}^n)$ is that Σ's minimum eigenvalue, $\lambda_{min}(\Sigma)$, is greater than $\eta_\varepsilon^2(\mathbb{Z}^n)$. Let $\mathbf{M} := \Sigma[I_0] \in \mathbb{R}^{n_0 \times n_0}$ for $n_0 = |I_0|$. $\mathbf{M}$ is diagonalizable so let $\mathbf{M} = \mathbf{Q}^t \Lambda \mathbf{Q}$ be its diagonalization. Notice, we have the following inequality from the interlacing theorems which imply $\lambda_{min}(\mathbf{M}) \geq \lambda_{min}(\Sigma)$,

$$\mathbf{x}^t \mathbf{M} \mathbf{x} = \mathbf{x}^t \mathbf{Q}^t \Lambda \mathbf{Q} \mathbf{x} = \mathbf{y}^t \Lambda \mathbf{y} = \sum_{i \in [n_0]} \lambda_i y_i^2 \geq \lambda_{min}(\Sigma) \|\mathbf{y}\|^2 = \lambda_{min}(\Sigma) \|\mathbf{x}\|^2.$$

Next, we can bound the quantity $\rho_{\sqrt{\mathbf{M}^{-1}}}((\mathbb{Z}^{n_0})^*) = \rho_{\sqrt{\mathbf{M}^{-1}}}(\mathbb{Z}^{n_0})$ by $1+\varepsilon$:

$$\rho_{\sqrt{\mathbf{M}^{-1}}}(\mathbb{Z}^{n_0}) = \sum_{\mathbf{x} \in \mathbb{Z}^{n_0}} e^{-\pi \mathbf{x}^t \mathbf{M} \mathbf{x}} \leq \sum_{\mathbf{x} \in \mathbb{Z}^{n_0}} e^{-\pi \lambda_{min}(\Sigma) \|\mathbf{x}\|^2}$$

$$\leq \sum_{\mathbf{x} \in \mathbb{Z}^n} e^{-\pi \lambda_{min}(\Sigma) \|\mathbf{x}\|^2} \leq 1 + \varepsilon.$$

The jump from $\mathbb{Z}^{n_0}$ to $\mathbb{Z}^n$ comes from the relation $\mathbb{Z}^{n_0} \subset \mathbb{Z}^n$. The proof for the Schur complement is identical. □

Next, we state and prove our main convolution lemma.

Lemma 3.4.3 *For any real $0 < \varepsilon \leq 1/2$, positive integers n, m, vector $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2) \in \mathbb{R}^{n+m}$, and positive definite matrix $\Sigma = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^t & \mathbf{D} \end{bmatrix} \succeq \eta_\varepsilon^2(\mathbb{Z}^{n+m})$, $\mathbf{A} \in \mathbb{Z}^{n \times n}$, $\mathbf{B} \in \mathbb{Z}^{n \times m}$, and $\mathbf{D} \in \mathbb{Z}^{m \times m}$ (where $\Sigma/\mathbf{D} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{B}^t$ is the Schur complement of $\mathbf{D}$) the random process*

- $\mathbf{x}_2 \leftarrow D_{\mathbb{Z}^m, \sqrt{\mathbf{D}}, \mathbf{c}_2}$.
- $\mathbf{x}_1 \leftarrow D_{\mathbb{Z}^n, \sqrt{\Sigma/\mathbf{D}}, \mathbf{c}_1 + \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}_2 - \mathbf{c}_2)}$.

produces a vector $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{Z}^{n+m}$ such that the Rényi divergence of order infinity of $D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}$ from $\mathbf{x}$ is less than or equal to $1 + 4\varepsilon$.

Proof:

First, we write out the probability and use Lemma 3.4.1 to simplify the numerator. Let $\mathbf{x}' = (\mathbf{x}'_1, \mathbf{x}'_2)$ below.

$$\begin{aligned}\Pr[\mathbf{x}_1 = \mathbf{x}'_1, \mathbf{x}_2 = \mathbf{x}'_2] &= \frac{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbf{x}'_1 - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}'_2 - \mathbf{c}_2)) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbf{x}'_2 - \mathbf{c}_2)}{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}'_2 - \mathbf{c}_2)) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbb{Z}^m - \mathbf{c}_2)} \\ &= \frac{\rho_{\sqrt{\Sigma}}(\mathbf{x}' - \mathbf{c})}{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{x}'_2 - \mathbf{c}_2)) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbb{Z}^m - \mathbf{c}_2)}\end{aligned}$$

Regarding the denominator, we use Lemma 3.4.2 to see that $\Sigma/\mathbf{D} \succeq \eta^2_\varepsilon(\mathbb{Z}^n)$ since $\Sigma \succeq \eta^2_\varepsilon(\mathbb{Z}^{n+m})$. Now, we can use Lemma 3.2.3 for the first gaussian sum (dependent on $\mathbf{x}'_2$) in the denominator to see,

$$\Pr[\mathbf{x}_1 = \mathbf{x}'_1, \mathbf{x}_2 = \mathbf{x}'_2] \in \alpha \cdot D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}(\mathbf{x}') \cdot \left[\left(\frac{1-\varepsilon}{1+\varepsilon}\right), 1\right]^{-1}$$

where $\alpha = \frac{\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m} - \mathbf{c})}{\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n) \cdot \rho_{\sqrt{\mathbf{D}}}(\mathbb{Z}^m - \mathbf{c}_2)}$.

Next, we show $\alpha \approx 1$. Using Lemma 3.4.1 we expand

$$\rho_{\sqrt{\Sigma}}(\mathbb{Z}^{n+m} - \mathbf{c}) = \sum_{\mathbf{y}_2 \in \mathbb{Z}^m} \rho_{\sqrt{\mathbf{D}}}(\mathbf{y}_2 - \mathbf{c}_2) \cdot \rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{y}_2 - \mathbf{c}_2)).$$

The sum $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{y}_2 - \mathbf{c}_2))$ is approximately $\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n)$ because $\Sigma/\mathbf{D} \succeq \eta^2_\varepsilon(\mathbb{Z}^n)$ as a consequence of Lemma 3.4.2 and $\Sigma \succeq \eta^2_\varepsilon(\mathbb{Z}^{n+m})$. In other words,

$$\rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n - \mathbf{c}_1 - \mathbf{B}\mathbf{D}^{-1}(\mathbf{y}_2 - \mathbf{c}_2)) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_{\sqrt{\Sigma/\mathbf{D}}}(\mathbb{Z}^n)$$

and $\alpha \in \left[\left(\frac{1-\varepsilon}{1+\varepsilon}\right), 1\right]$.

Finally, we have the approximation

$$\Pr[\mathbf{x}_1 = \mathbf{x}'_1, \mathbf{x}_2 = \mathbf{x}'_2] \in \left[\left(\frac{1-\varepsilon}{1+\varepsilon}\right), \left(\frac{1+\varepsilon}{1-\varepsilon}\right)\right] \cdot D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}(\mathbf{x}').$$

Given the restriction on $\varepsilon \in (0, 1/2]$, we have the relation we desire

$$\Pr[\mathbf{x}_1 = \mathbf{x'}_1, \mathbf{x}_2 = \mathbf{x'}_2] \in [1-4\varepsilon, 1+4\varepsilon] \cdot D_{\mathbb{Z}^{n+m}, \sqrt{\Sigma}, \mathbf{c}}(\mathbf{x}').$$

□

Next, we bound the Rényi divergence of order infinity between the output of SAMPLEPZ and the desired distribution. We need to ensure each discrete gaussian convolution in the algorithm is non-degenerate. We do not analyze the statistical loss from the floating point computations. As shown in Lemma 3.4.3, we need $\Sigma/\mathbf{D} \succeq \eta_\varepsilon^2(\mathbb{Z}^{n_0})$ and $\mathbf{D} \succeq \eta_\varepsilon^2(\mathbb{Z}^{n_1})$ at *each* of the n discrete gaussian convolutions. This is met through a simple condition on Σ_p as hinted to in Lemma 3.4.2.

Theorem 3.4.1 *Let* $0 < \varepsilon \leq 1/2$. *If* $\Sigma_p \succeq \eta_\varepsilon^2(\mathbb{Z}^{n(2+\log q)})$, *then* SAMPLEPZ *returns a perturbation with a Rényi divergence of order infinity* $R_\infty(D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}} || \text{SAMPLEPZ}) \leq 1 + 12n\hat{\varepsilon}$.

Proof: Since each covariance given to SAMPLEFZ is a Schur complement or a principal submatrix of a Schur complement of Σ_p, Lemma 3.4.2 and the interlacing theorems (Theorem 2.1.1 and Theorem2.1.2) imply the conditions for Lemma 3.4.3 are met. As there are $n-1$ convolutions (inner nodes of a full binary tree of depth $\log n$), a quick induction argument shows the probability distribution of the output of SAMPLEPZ is in the interval $[(1-4\varepsilon)^{3(n-1)}, (1+4\varepsilon)^{3(n-1)}] \cdot D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}}(\mathbf{x})$. Then, we have $R_\infty(D_{\mathbb{Z}^{n(2+\log q)}, \sqrt{\Sigma_p}} || \text{SAMPLEPZ}) \leq (1+4\varepsilon)^{3(n-1)} \approx 1 + 12n\hat{\varepsilon}$. □

For common parameters $\varepsilon = 2^{-128}$ and $n = 1024$, we have $1 - (1+4\varepsilon)^{3(n-1)} \approx 2^{-114}$.

In summary, this shows the FFT-like recurrence in perturbation sampling the integer lattice with an algebraic covariance in power of two cyclotomic rings through repeated convolutions. The relative simplicity of the power of two case relies on the fact that matrix transpose corresponds to the conjugation field automorphism. Hermitian transpose corresponds to the conjugation automorphism in the general cyclotomic case. Therefore, we would use the canonical

embedding for efficient perturbation sampling in general cyclotomic rings, which we shown in the next subsction.

Storage and Efficiency

Recent results suggest double precision floating point numbers are enough to preserve security in lattice-based cryptosystems for commonly used parameters [74, 14], but one can use the lazy floating point techniques of [40] for SAMPLEZ and still yield a version of SAMPLEPZ that has quasi-linear time complexity on average if longer floating point precision is needed. This would involve tweaking SAMPLEFZ and SAMPLE2Z to record their path through the tree of recursions and pass the path to SAMPLEZ. Then, SAMPLEZ could re-compute its center and variance in high precision with access to previous samples and the trapdoor $\tilde{\mathbf{T}}$.

SAMPLEFZ runs in time $\tilde{O}(n)$, and SAMPLEPZ runs in time $\tilde{O}(n \log q)$ as a result. Storage consists of storing the trapdoor $\tilde{\mathbf{T}}$ which is $2n \log q$ small integers (less than q, $O(\log n)$ bits each), and the algorithm stores $4n$ floating point numbers for the input of SAMPLE2Z.

3.4.2 General Cyclotomic Rings

Here we sketch sampling methods for arbitrary cyclotomics. Both the techniques in the previous subsection and the techniques of [40, 72] apply to this setting as well. Let $n = \varphi(n')$, q be a positive integer, $\mathscr{O}_{n'} = \mathbb{Z}[x]/(\Phi_{n'}(x))$ be the n'-th cyclotomic ring with $\mathscr{K}_{n'}$ as the n'-th cyclotomic field over $\mathbb{Q}$, and let $\tilde{\mathbf{T}} \in \mathscr{O}_{n'}^{2\times \log q}$ be a ring trapdoor matrix. Note, the ring $\mathscr{O}_{n'}$ is a lattice in the canonical embedding. Let $\mathrm{rad}(n')$ be the product of all distinct prime divisors of n'. Define the diagonal matrix of a field element, $f \in \mathscr{K}_{n'}$, as $\Psi(f)_{i,i} = f(\zeta^i)$ where ζ is a complex primitive n'-th root of unity and each i is a distinct element in the group of units modulo n', $i \in \mathbb{Z}_{n'}^*$. This is the multiplication matrix of an element in the canonical embedding. Notice, $\Psi(f)^\dagger = \Psi(f^*)$ since conjugation is an automorphism of all cyclotomic fields over $\mathbb{Q}$. We apply $\Psi(\cdot)$ element-wise to vectors and matrices over $\mathscr{K}_{n'}$.

Now, our goal is to efficiently sample the lattice $D_{\mathscr{O}_{n'}^{2+\log q},\sqrt{\Sigma_p}}$ where

$$\Sigma_p = s^2\mathbf{I} - \alpha^2 \begin{bmatrix} \Psi(\tilde{\mathbf{T}}) \\ I \end{bmatrix} \begin{bmatrix} \Psi(\tilde{\mathbf{T}})^\dagger & I \end{bmatrix}$$

and $\Psi(\tilde{\mathbf{T}})^\dagger$ is the Hermitian transpose of $\Psi(\tilde{\mathbf{T}})$. Notice, the Hermitian transpose of an element's diagonal matrix results in the diagonal matrix of a different field element since complex conjugation is an automorphism of $\mathscr{K}_{n'}$. Sampling $D_{\mathscr{O}_{n'}^{2+\log q},\sqrt{\Sigma_p}}$ reduces to sampling discrete gaussians over $\mathbb{Z}$ in nearly the same steps as the previous subsection with $z = (\alpha^{-2} - s^{-2})^{-1}$:

1. Sampling $\mathbf{p} \leftarrow D_{\mathscr{O}_{n'}^{2+\log q},\sqrt{\Sigma_p}}$ reduces to sampling $D_{\mathscr{O}_{n'}^2,\sqrt{\Sigma_{2\times 2}}}$ where

$$\Sigma_{2\times 2} = s^2\mathbf{I} - z\cdot\Psi(\tilde{\mathbf{T}})\Psi(\tilde{\mathbf{T}})^\dagger = \begin{bmatrix} \Psi(a) & \Psi(b) \\ \Psi(b^*) & \Psi(d) \end{bmatrix}$$

   by first sampling $\mathbf{p}_2 \leftarrow D_{\mathscr{O}_{n'}^{\log q},\sqrt{s^2-\alpha^2}}$, updating the randomized center $\mathbf{c} := \frac{-\alpha^2}{s^2-\alpha^2}\Psi(\tilde{\mathbf{T}})\mathbf{p}_2$, then sampling $\mathbf{p}_1 \leftarrow D_{\mathscr{O}_{n'}^2,\mathbf{c},\sqrt{\Sigma_{2\times 2}}}$.

2. Sampling $D_{\mathscr{O}_{n'}^2,\mathbf{c},\sqrt{\Sigma_{2\times 2}}}$ reduces to sampling $D_{\mathscr{O}_{n'},\sqrt{\Psi(f)}}$ for a positive definite field element f by sampling $\mathbf{q}_2 \leftarrow D_{\mathscr{O}_{n'},\mathbf{c}_2,\sqrt{\Psi(d)}}$, then by updating the center $\mathbf{c}_1 := \mathbf{c}_1 + \Psi(bd^{-1})(\mathbf{q}_2 - \mathbf{c}_2)$ and sampling $\mathbf{q}_2 \leftarrow D_{\mathscr{O}_{n'},\mathbf{c}_2,\sqrt{\Psi(a-bd^{-1}b^*)}}$.

Similar to how SAMPLEPZ must sample $D_{\mathbb{Z}^n\log q,\sqrt{s^2-\alpha^2}}$, the first step above requires sampling the discrete gaussian $D_{\mathscr{O}_{n'}^{\log q},\sqrt{s^2-\alpha^2}}$. This can be done in $O(\mathrm{rad}(n')n'\log q)$ since spherical discrete gaussians over the ring $\mathscr{O}_{n'}$ can be sampled in time $O(\mathrm{rad}(n')n')$ [65].

By using the randomized nearest plane algorithm to sample discrete gaussians on $\mathscr{O}_{n'}$ with diagonal covariances, sampling $\mathbf{p}$ statistically close to $D_{\mathscr{O}_{n'}^{2+\log q},\sqrt{\Sigma_p}}$ is $O(\mathrm{rad}(n')n'\log q) + \tilde{O}(n\log q)$. Note, the randomized rounder of [40, 72] could be used to sample $D_{\mathscr{O}_{n'},\sqrt{\Psi(f)}}$. We conclude on observing that the above holds for any number field which has complex conjugation

as an automorphism, though might not be as efficient because the ring/lattice of interest may have no sparse basis in the canonical embedding.

Acknowledgement

This is reprinted as it appears (with minor modifications) from the publication "Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus," presented by this dissertation's author at EUROCRYPT 2018 [42] and is joint work with Daniele Micciancio.

Chapter 4

The Lattice Gadget Toolkit

4.1 Introduction

The second chapter builds on the first chapter by optimizing all algorithmic tasks on power-of-b gadgets, as well as introducing an efficient class of gadgets geared towards the Chinese Remainder Transformation/Representation (CRT) of $\mathbb{Z}_q$ when $q = q_1 q_2 \cdots q_l$ for coprime q_i. These CRT gadgets are used in many advanced lattice-based cryptosystems where the modulus q is often much larger than the native 64 bits in most modern hardware. In other words, we can pick each coprime factor q_i as an integer just less than 64 bits and utilize the CRT isomorphism

$$\mathbb{Z}_q \cong \mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2} \times \cdots \times \mathbb{Z}_{q_l}.$$

Gadgets

Gadgets in lattice-based cryptography are defined with the following two functions in-mind. Both functions are parameterized by a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for $m \geq n \log q$. The first is the short-integer-solutions (SIS) [4] one-way function: $f_{\mathbf{A}}(\mathbf{x}) := \mathbf{A}\mathbf{x} \mod q$ where $\mathbf{x}$ is an integer vector with l_2 norm less than some parameter β. We usually set β so the function has collisions ($\beta = \sqrt{m}$ for $\mathbf{x} \in \{0, \pm 1\}^m$). And, we denote SIS "inversion" as $\mathbf{x} \leftarrow \mathbf{A}^{-1}(\mathbf{u})$ where $\mathbf{u} \in \mathbb{Z}_q^n$ is given as an input and $\mathbf{x}$ is any short integer vector satisfying $\mathbf{A}\mathbf{x} \mod q = \mathbf{u} \mod q$. Note that $\mathbf{A}^{-1}(\cdot)$ is not a matrix, nor is it a proper function inversion, but it can be randomized. In fact, the gaussian sampling algorithm in the previous chapter is a discrete gaussian SIS inversion on a

fixed matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$. Next, we define the LWE function [75] as $g_{\mathbf{A}}(\mathbf{s}, \mathbf{e}) := \mathbf{s}^t\mathbf{A} + \mathbf{e}^t \mod q$ where $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}^m$ has small entries. We will only care about parameter regimes where the LWE function is invertible (injective).

Loosely, a gadget $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is any matrix such that the SIS and LWE functions are easy to invert. The most commonly used gadgets are of the form $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}^t$ where $\mathbf{g}^t = (1, b, \cdots, b^{k-1})$ and $k = \lceil \log_b(q) \rceil$. The case $b = 2$ corresponds to a bit decomposition for a deterministic SIS inversion, $\mathbf{g}^{-1}(u) := \mathbf{x} \in \{0,1\}^k$ such that $\mathbf{g}^t\mathbf{x} \mod q = u$.

These gadgets first appeared in lattice-based trapdoors [4, 12] as well as second generation FHE schemes [28] in the form of a bit-decomposition. Micciancio and Peikert gave an optimized lattice trapdoor scheme in [66] which reduces inverting the SIS and LWE functions on a (pseudo)random $\mathbf{A}$ to inverting them on a fixed gadget $\mathbf{G}$. Further, they were the first to rigorously analyze the gadget lattice, or "G-lattice," $\Lambda_q^{\perp}(\mathbf{G}) := \{\mathbf{x} \in \mathbb{Z}^{nk} : \mathbf{G}\mathbf{x} = \mathbf{0} \mod q\}$. Alperin-Sheriff and Peikert [10] realized the potential for a simple, randomized bit-decomposition as $\mathbf{G}^{-1}(\cdot)$. Note, here we are thinking of a much simpler subgaussian distribution than a discrete gaussian preferably avoiding the use of floating point numbers (only achieved when $q = b^k$ in [10]). Alperin-Sheriff and Peikert use subgaussian analysis [79] in order to rigorously analyze these randomized bit-decompositions.

Recently there has been an effort to introduce gadgets more compatible with the CRT representation of $\mathbb{Z}_q$ [53, 22]. Traditional power-of-b gadgets require one to convert to the large modulus $\mathbb{Z}_q$ before computing $\mathbf{G}^{-1}(\cdot)$. The use of the large modulus here requires multi-precision numbers, nullifying the benefits of the CRT representation. Let us denote these CRT-compatible gadgets as $\mathbf{g}_{CRT}^t$ or $\mathbf{G}_{CRT}$. The gadgets presented in [53, 22] were only applicable in the specific setting where our modulus q has small prime factors, also known as a smooth modulus. Restricting to smooth moduli is undesirable since implementations often use a modulus where each coprime factor is a prime around 64 bits. Moduli with this form allow the use of the "double-CRT" representation [45]. Of particular interest is performing a discrete gaussian or subgaussian $\mathbf{g}_{CRT}^{-1}(\cdot)$ efficiently with only ever having to represent numbers in the CRT representation.

Contribution

Fix an integer $n > 0$ as our security parameter, an integer modulus $q > 0$, an integer base $0 < b \leq \sqrt{q}$. Note that we can have $b = q$, but then our power-of-b gadget will be trivial $\mathbf{g} = (1), \mathbf{G} = \mathbf{I}_n$.

This chapter achieves the following on a power-of-b gadget $\mathbf{G}$ for any modulus q:

- compute a subgaussian $\mathbf{g}^{-1}(\cdot)$ with minimal distribution width, minimal randomness, and in linear time and space $O(k)$,
- decode the LWE function on $\mathbf{g}^t$ in linear time $O(k)$ while maximizing the error tolerance $\|\mathbf{e}\|_\infty$.

Previously, these tasks were only optimized when $q = b^k$ [66, 10].

Further, this chapter introduces a general set of CRT gadgets for a wide class of moduli $q = q_1 q_2 \cdots q_l$ that achieves the following:

- discrete gaussian and subgaussian inversion can be performed in linear time and in-parallel with a distribution width *independent* of modulus' factors all while keeping arithmetic in the CRT representation,
- and perform LWE decoding in linear time in-parallel while keeping arithmetic in the CRT representation.

Mathematically, this efficiency is due to the structure of the CRT gadget's G-lattice:

$$\Lambda_q^\perp(\mathbf{g}_{CRT}^t) = \Lambda_q^\perp(\mathbf{g}_1^t) \oplus \Lambda_q^\perp(\mathbf{g}_2^t) \oplus \cdots \oplus \Lambda_q^\perp(\mathbf{g}_l^t)$$

where each $\mathbf{g}_i^t$ is a power-of-b_i gadget chosen by the user with the requirement $b_i < q_i$. Therefore, we can choose each $b_i = 2$ in order to have a narrow distribution on the (CRT) G-lattice.

We note that this set of gadgets directly generalizes [22] since their gadgets correspond to $b_i = q_i$, or $\mathbf{g}_i^t = (1)$, and they generalize the gadgets of [53] in spirit[1].

[1]The CRT gadgets of [53] correspond to a different isomorphism on $\mathbb{Z}_q$. The isomorphism in [53] is called the

Technical details

Our linear time and space subgaussian bit decomposition algorithm for power-of-b gadgets $\mathbf{g}^t$ follows the same algorithmic blueprint as our efficient discrete gaussian sampling algorithm from the previous chapter. That is, we use the G-lattice's sparse, triangular basis factorization

$$\mathbf{B}_q = \mathbf{BD}$$

where $\mathbf{B}_q$ is a basis of the G-lattice $\Lambda_q^\perp(\mathbf{g}) = \mathscr{L}(\mathbf{B}_q)$ and $\mathbf{B}, \mathbf{D}$ are sparse, triangular matrices. Now our algorithm is analogous to the previous chapter's solution:

1. sample a subgaussian distribution on the lattice generated on $\mathscr{L}(\mathbf{D})$,
2. and use $\mathbf{B}$ as a linear transformation.

The first step is done by performing a "bent-coin" randomized version of Babai's nearest plane algorithm [13]. We can sample a narrow distribution on $\mathscr{L}(\mathbf{D})$ since $\mathbf{D}$ has small entries. With a careful use of arithmetic in this "bent-coin" nearest plane algorithm, we have an algorithm which avoids floating point numbers, samples a subgaussian distribution exactly, has a trade-off between randomness consumed and distribution width (more randomness needed for a tighter distribution), and uses $O(k)$ time and space.

Our linear time and space LWE decoding algorithm is merely an exercise in lattice duality. The LWE decoding problem on $\mathbf{g}^t$ can be seen as the problem of decoding on the lattice

$$\Lambda_q(\mathbf{g}^t) = \{\mathbf{x} \in \mathbb{Z}^k : \exists s \in \mathbb{Z}_q \text{ s.t. } \mathbf{x} = s\mathbf{g}^t \mod q\} = \mathbb{Z}\mathbf{g}^t + q\mathbb{Z}^k.$$

This lattice is a scaled version of the G-lattice's dual

$$\Lambda_q(\mathbf{g}^t) = q\Lambda_q^\perp(\mathbf{g}^t)^*$$

invariant factor decomposition.

since for any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ we have $\Lambda_q(\mathbf{A}) = q\Lambda_q^{\perp}(\mathbf{A})^*$ (where $\mathscr{L}^*$ denotes the dual lattice of $\mathscr{L}$). Therefore, the sparse basis factorization of our favorite basis for $\Lambda_q^{\perp}(\mathbf{g}^t)$

$$\mathbf{B}_q = \mathbf{BD}$$

yields a triangular factorization for the dual

$$\Lambda_q(\mathbf{g}^t) = \mathscr{L}(q\mathbf{B}_q^{-t}), \mathbf{B}_q^{-t} = \mathbf{B}^{-t}\mathbf{D}^{-t}.$$

Luckily, the simple structure of $\mathbf{D}$ yields an efficient Babai's nearest plane decoding to $\mathscr{L}(q\mathbf{D}^{-t})$. Now the algorithmic blueprint should be clear:

1. given $\mathbf{y} = s\mathbf{g}^t + \mathbf{e}^t$ apply $\mathbf{B}^t$ as a linear transformation,
2. then decode $\mathbf{B}^t\mathbf{y}$ to $q\mathscr{L}(\mathbf{D}^{-t})$ in linear time.

The linear transformation $\mathbf{B}$ slightly increases the noise's entry size, but this is overall a small price to pay. The result is a linear time and space power-of-b gadget decoding algorithm with error tolerance $\|\mathbf{e}\|_\infty \leq \frac{q}{2(b+1)}$.

Our family of CRT gadgets employ a simple algebraic trick: delegate the inverse CRT transformation to the gadget. The CRT isomorphism $\varphi : \mathbb{Z}_q \to \mathbb{Z}_{q_1} \times \mathbb{Z}_{q_2} \times \cdots \times \mathbb{Z}_{q_l}$ has a simple form, $\varphi(x) = (x \mod q_1, \cdots, x \mod q_l)$, but its inverse is a little more complicated. In particular, it is of the form $\varphi^{-1}(x_1, \cdots, x_l) = \varphi^{-1}(\mathbf{x}) = \langle \alpha, \mathbf{x} \rangle$ where $\alpha = (\alpha_1, \cdots, \alpha_l)$ is a vector of integers with entries dependent on q's factorization. Now, the CRT gadget is of the form

$$\mathbf{g}_{CRT}^t = (\alpha_1\mathbf{g}_1^t, \cdots \alpha_l\mathbf{g}_l^t).$$

A few algebraic manipulations reveal an ideal G-lattice structure:

$$\Lambda_q^{\perp}(\mathbf{g}_{CRT}^t) = \Lambda_q^{\perp}(\mathbf{g}_1^t) \oplus \Lambda_q^{\perp}(\mathbf{g}_2^t) \oplus \cdots \oplus \Lambda_q^{\perp}(\mathbf{g}_l^t).$$

4.2 Background

We indicate numbers with lowercase letters, such as $z \in \mathbb{Z}$, vectors as bold lowercase letters, $\mathbf{z} \in \mathbb{Z}^n$, and matrices as uppercase bold letters, $\mathbf{M} \in \mathbb{R}^{n \times n}$. The default norm used is the l_2 norm of a vector unless stated otherwise, though we will often use the max, or l_∞, norm. For a real number r, denote $\lceil r \rfloor$ as the deterministic rounding function to a nearest integer of r. Rounding a real vector is applied analogously, entry-wise. Many computations will be done over the integers modulo q, $\mathbb{Z}_q$. We view $\mathbb{Z}_q$ through its balanced coset representatives in $(-q/2, q/2]$ unless stated otherwise. For a positive integer base b and a non-negative integer $u < b^k$, u's b-ary decomposition is a vector $[u]_b^k = (u_0, \cdots, u_{k-1}) \in \{0, \cdots, b-1\}^k$ and satisfies $\sum_i b^i u_i = u$. When $b = 2$, this is simply u's binary decomposition. Recall the Chinese Remainder Theorem for modular arithmetic. Let q be a positive integer with a prime factorization of $q = p_1^{e_1} \cdots p_l^{e_l} = q_1 \cdots q_l$. Then by the Chinese Remainder Theorem (CRT), we have $\mathbb{Z}_q \cong \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_l}$. The isomorphism $\phi(\cdot)$ is given by $\phi(a) = (a \mod q_1, \cdots, a \mod q_l)$ and its inverse is $\phi^{-1}(a_1, \cdots, a_l) = \sum_i (a_i) q_i^* \hat{q}_i$ where $q_i^* := \frac{q}{q_i}$ and $\hat{q}_i := (q_i^*)^{-1} \mod q_i$.

For a probability distribution χ, we denote $e \leftarrow \chi$ to mean e is sampled from χ. When χ is trivial (often over a number x), we will use $e \leftarrow x$ to be variable assignment as well.

4.2.1 Subgaussian Random Variables

A random variable X over $\mathbb{R}$ is *subgaussian* [65, 79] with parameter $\alpha > 0$ if its (scaled) moment generating function satisfies $\mathbb{E}[\exp(2\pi t X)] \leq \exp(\pi \alpha^2 t^2)$ for all $t \in \mathbb{R}$. Scaling a subgaussian X by any $c \in \mathbb{R}$ to $c \cdot X$ yields a subgaussian random variable with parameter $|c|\alpha$. If X is subgaussian with parameter α, then its tails are dominated by a Gaussian parameterized by α, $\Pr\{|X| \geq t\} \leq 2\exp(-\pi t^2/\alpha^2)$. Any B-bounded centered ($\mathbb{E}[X] = 0$) random variable X is subgaussian with parameter $B\sqrt{2\pi}$. When X is subgaussian with parameter α and Y conditioned on X taking any value is subgaussian with parameter β, $X + Y$ is subgaussian with parameter $\sqrt{\alpha^2 + \beta^2}$. This property is called *Pythagorean additivity*. The proof of the following Lemma is

derived by expanding $\mathbb{E}[\exp(2\pi t(X+Y))]$.

Lemma 4.2.1 *Let X, Y be discrete random variables over $\mathbb{R}$ such that X is subgaussian with parameter α and Y conditioned on X taking any value is subgaussian with parameter β. Then, $X+Y$ is subgaussian with parameter $\sqrt{\alpha^2+\beta^2}$.*

Proof: Expanding the moment generating function gives the result:

$$\begin{aligned}\mathbb{E}[\exp(2\pi t(X+Y))] &= \sum_z \sum_\chi \Pr\{Y = z-\chi | X = \chi\} \Pr\{X = \chi\} \exp(2\pi t z) \\ &= \sum_\chi \Pr\{X = \chi\} \exp(2\pi t \chi) \mathbb{E}[\exp(2\pi t Y) | X = \chi] \\ &\leq \exp(\pi t^2(\alpha^2+\beta^2)).\end{aligned}$$

□

A random vector $\mathbf{x}$ over $\mathbb{R}^n$ is *subgaussian* with parameter $\alpha > 0$ if $\langle \mathbf{x}, \mathbf{u} \rangle$ is subgaussian with parameter α for all unit vectors $\mathbf{u}$. Using a similar calculation to the above, one can show that if each coefficient of a random vector is subgaussian with parameter α conditioned on the previous coefficients taking any values, then the vector is subgaussian with parameter α. The slightly more general fact below is needed for our algorithms. Its proof is analogous to the proof of Lemma 4.2.1.

Lemma 4.2.2 *Let $\mathbf{x}$ be a discrete random vector over $\mathbb{R}^n$ such that each coordinate x_i is subgaussian with parameter α_i given the previous coordinates take any values. Then, $\mathbf{x}$ is a subgaussian vector with parameter $\max_i\{\alpha_i\}$.*

Proof: As before, we expand the moment generating function:

$$\begin{aligned}\mathbb{E}[\exp(2\pi t \langle \mathbf{x}, \mathbf{u} \rangle)] &= \sum_{\chi} \Pr\{\mathbf{x} = (\chi_1, \cdots, \chi_n)\} \exp(2\pi t \langle \chi, \mathbf{u} \rangle) \\ &\leq \exp(\pi t^2 \sum_i \alpha_i^2 u_i^2) \\ &\leq \exp(\pi t^2 \max_i {\alpha_i}^2 \|\mathbf{u}\|^2).\end{aligned}$$

The jump to the inequalities skips the straightforward calculations (nearly the same calculations as in Lemma 4.2.1). □

We emphasize this fact, for without it one is left with an unnecessary $\sqrt{n}$ term in the subgaussian parameter of subgaussian vectors. Now, that the sum of independently generated random vectors $\mathbf{x}$ and $\mathbf{y}$ subgaussian with parameters α and β is a subgaussian vector with parameter $\sqrt{\alpha^2 + \beta^2}$ immediately follows.

A main algorithm presented in this chapter will rely on a linear transformation of a discrete subgaussian vector.

Lemma 4.2.3 (Simplified [65, Corollary 2.3]) *Let* $\mathbf{x}$ *be a subgaussian random vector with parameter* α *and let* $\mathbf{M}$ *be a linear transformation. Then,* $\mathbf{Mx}$ *is a subgaussian vector with parameter* $\alpha \lambda_{max}(\mathbf{M}\mathbf{M}^t)^{1/2}$ *where* $\lambda_{max}(\cdot)$ *is the largest eigenvalue.*

4.2.2 q-ary Lattices

Throughout this chapter we will mostly be concerned with *q-ary lattices*. These are full-rank integer lattices with $q \cdot \mathbb{Z}^k$ as a sublattice. Fix an integer $q > 0$ to be used as a modulus and let $m > w > n$. A matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ is *primitive* if $\mathbf{A}\mathbb{Z}_q^m = \mathbb{Z}_q^n$. Given an $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define the following lattices: $\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{0} \mod q\}$, and $\Lambda_q(\mathbf{A}) = \{\mathbf{v} \in \mathbb{Z}^m : \exists\, \mathbf{s} \in \mathbb{Z}^n,\ \mathbf{v}^t = \mathbf{s}^t\mathbf{A} \mod q\}$. These lattices satisfy the following duality relation: $\Lambda_q^{\perp}(\mathbf{A})^* = q \cdot \Lambda_q(\mathbf{A})$. Further, the cosets of $\Lambda_q^{\perp}(\mathbf{A})$, $\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}) := \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{A}\mathbf{z} = \mathbf{u} \mod q\}$, are in bijection with $\mathbb{Z}_q^n$ when $\mathbf{A}$ is primitive. Let $\mathbf{G}$ be an arbitrary, primitive matrix over $\mathbb{Z}_q$. The following sampling problem,

defined on the integer cosets of $\Lambda_q^{\perp}(\mathbf{G})$, is needed for many advanced lattice crypto-schemes.

Definition 4.2.1 *For a primitive $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$, the subgaussian decomposition problem with parameter α for $\mathbf{G}$ is to sample vectors $\mathbf{x} \in \mathbb{Z}^w$ subgaussian with parameter α such that $\mathbf{u} = \mathbf{G}\mathbf{x} \mod q$ for arbitrary $\mathbf{u}$ given as input.*

Another name for this problem is subgaussian sampling. A generic adaptation of Babai's algorithm (analyzed in Section 4.4, called the subgaussian nearest plane algorithm) is used in [10] (AP14) to achieve subgaussian decomposition for a specific $\mathbf{G}$. In general, this generic algorithm runs in time $O(k^2)$, and uses space $O(k^3)$. Another, related problem is the discrete Gaussian sampling problem.

Definition 4.2.2 *For a primitive $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$, the discrete Gaussian sampling problem with width s for $\mathbf{G}$ is to sample vectors $\mathbf{x} \in \mathbb{Z}^w$ distributed as $D_{\mathbb{Z}^w,s}$ conditioned on $\mathbf{G}\mathbf{x} \mod q = \mathbf{u}$ for arbitrary $\mathbf{u}$ given as input.*

Efficient solutions with small s for commonly used $\mathbf{G}$'s are given in [66, 42]. Both of the above sampling problems have polynomial time solutions using randomized versions of Babai's algorithm. In addition, we will consider decoding the q-ary code defined by $\mathbf{G}$ for an arbitrary, primitive $\mathbf{G}$.

Definition 4.2.3 *For a primitive $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$, the LWE decoding problem with tolerance δ on $\mathbf{G}$ is to return $\mathbf{s}$ given $\mathbf{s}^t\mathbf{G} + \mathbf{e}^t \mod q$ for an error $\|\mathbf{e}\|_\infty < \delta$.*

Specifically, we want to efficiently decode $\mathbf{G}$ while maximizing $\delta \in [0, q/2)$. An efficient LWE decoding algorithm for a specific, commonly used $\mathbf{G}$ ($b = 2$ in the paragraph below) with tolerance $q/4$ is provided in [66].

A $\mathbf{G}$ commonly used in lattice-based schemes is defined as follows. Fix an integer $b \in (1, q)$, known as the *base*, and let $k = \lceil \log_b q \rceil$. The block-diagonal gadget matrix is $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^t$ with blocks $\mathbf{g}^t := (1, b, \cdots, b^{k-1})$. A common basis for $\Lambda_q^{\perp}(\mathbf{g}^t)$ [42] $\mathbf{S}_q$ has a sparse, triangular factorization $\mathbf{S}_q = \mathbf{SD}$ [42] (restated in Section 4.5.2 in this chapter).

4.3 Gadget Matrices

In order to guide our search for gadget matrices with efficient inversion and sampling algorithms, we give a simple general definition of gadget. The definition is modeled after the properties required by the digit decomposition problem, perhaps the simplest and most natural application of gadgets. But, as we will see, this simple characterization is enough to guarantee (theoretical) solutions to all problems that arise in the application of gadgets in lattice cryptography.

Definition 4.3.1 *For any finite additive group A, an A-gadget of size w and quality β is a vector $\mathbf{g} \in A^w$ such that any group element $u \in A$ can be written as an integer combination $u = \sum_i g_i \cdot x_i$ where $\mathbf{x} = (x_1, \ldots, x_w)$ has norm at most $\|\mathbf{x}\| \leq \beta$.*

We are primarily interested in gadgets for $A = \mathbb{Z}_q^n$, in which case the gadget is conveniently represented as a matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ such that for any $\mathbf{u} \in \mathbb{Z}_q^n$ there is a vector $\mathbf{x} \in \mathbb{Z}^w$ of length $\|\mathbf{x}\| \leq \beta$ such that $\mathbf{G}\mathbf{x} = \mathbf{u} \pmod{q}$. We defined gadgets in terms of abstract groups to emphasize that the dimension n and modulus q should be thought of as part of the problem specification (typically mandated by the target application), while the w and β describe the size and quality of the solution. In particular, for any given n and q, one may consider multiple gadgets achieving different values of w and β. Naturally, smaller w and β are preferable, but as we will see there is a natural tradeoff between these two values, and one may increase β in order to reduce w and vice versa.

Before establishing a formal connection between the above definition and the notion of gadget informally defined in previous work, we make some important observations.

- The matrix $\mathbf{G}$ is necessarily primitive, i.e., $\mathbf{G}\mathbb{Z}_q^w = \mathbb{Z}_q^n$. Moreover, any primitive matrix is a $\mathbb{Z}_q^n$-gadget for a sufficiently large $\beta = \max_{\mathbf{u}} \min\{\|\mathbf{x}\| : \mathbf{G}\mathbf{x} = \mathbf{u} \pmod{q}\}$.
- If $\mathbf{g} \in \mathbb{Z}^k$ is a $\mathbb{Z}_q$-gadget of quality β, then $\mathbf{G} = \mathbf{I} \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n \times w}$ is a $\mathbb{Z}_q^n$-gadget of size $w = kn$ and quality $\sqrt{n}\beta$.

- All definitions and constructions are easily adapted to ideal lattices (as used in the Ring-SIS and Ring-LWE problems) simply by considering "structured gadgets" of the form $\mathbf{G} \otimes [\alpha_1, \ldots, \alpha_n]$ where $[\alpha_1, \ldots, \alpha_n]$ is an appropriate $\mathbb{Z}$-basis of the underlying ring.

Based on the above observations, constructions may focus on the case $n = 1$, i.e., gadget vectors $\mathbf{g} \in \mathbb{Z}_q^w$, and then extend the solution to larger n (and possibly to the ring setting) using general techniques. In fact, this is how larger gadgets are built in all applications we are aware of. However, all the results in this section hold for arbitrary matrices, not necessarily with this tensor structure. So, for the sake of generality, we use matrix notation.

In order to justify our abstract definition of gadget, we show that it guarantees all other properties of gadgets used by lattice cryptography: it maps the gaussian distribution to an almost uniform vector $\mathbf{G}D_{\mathbb{Z},s}^w \approx \mathbb{Z}_q^n$ (as needed by the trapdoor generation algorithm of [66]), and it supports efficient algorithms to invert the LWE function $g_{\mathbf{G}}(\mathbf{x}, \mathbf{e})$, for discrete gaussian sampling on $f_{\mathbf{G}}^{-1}(\mathbf{u})$, and for subgassian decomposition with respect to $\mathbf{G}$. All these properties are proved by bounding the relevant parameters of the lattice $\Lambda_q^\perp(\mathbf{G})$ defined by $\mathbf{G}$.

Theorem 4.3.1 *For any gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ of quality β, the lattice $L = \Lambda_q^\perp(\mathbf{G})$ has a basis $\mathbf{S}$ with orthogonalized length $\|\tilde{\mathbf{S}}\| \leq 2\beta + \sqrt{w}$, successive minima $\lambda_1(L), \ldots, \lambda_w(L) \leq 2\beta + \sqrt{w}$ and smoothing parameter $\eta(L) \leq (2\beta + \sqrt{w})\omega(\sqrt{\log n})$.*

Proof: We first bound the covering radius $\mu(L)$. Let $\mathbf{x} \in \mathbb{R}^w$ be arbitrary, and let $\mathbf{y} = \lfloor \mathbf{x} \rceil$ be a nearest point in $\mathbb{Z}^w$ to $\mathbf{x}$. There exists some integer vector $\mathbf{z}$ of norm at most β such that $\mathbf{G}\mathbf{z} = -\mathbf{G}\mathbf{y} \mod q$. Therefore, the vector $\mathbf{y} + \mathbf{z}$ is in $\Lambda_q^\perp(\mathbf{G})$ and is at distance at most $\beta + \sqrt{w}/2$ from $\mathbf{x}$ by two applications of the triangle inequality.

The other bounds immediately follow from general relations (satisfied by any lattice) $\lambda_w(L) \leq 2\mu(L)$ and $\eta(L) \leq \lambda_n(L)\omega(\sqrt{\log n})$. Finally, any lattice has a basis with orthogonalized length $\|\tilde{\mathbf{S}}\| \leq \lambda_w(L)$. □

Note, the proof and theorem easily generalizes to any finite abelian group. Using

the bound on the smoothing parameter, and the short (orthogonalized) basis $\mathbf{S} \in \mathbb{Z}^{w \times w}$, we immediately get the following applications.

Corollary 4.3.1 *For any gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times w}$ of quality β and $s \geq (2\beta + \sqrt{w})\sqrt{\omega(\log n)}$, the distribution $\mathbf{G}D_{\mathbb{Z},s}^w$ is statistically close to the uniform distribution over $\mathbb{Z}_q^n$. Moreover, there are polynomial-time algorithms for the following problems:*

- *Discrete Gaussian Sampling for the function $f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G}\mathbf{x} \pmod q$ and input distribution $D_{\mathbb{Z},s}^w$ with $s \geq (2\beta + \sqrt{w})\sqrt{\omega(\log n)}$.*
- *Subgaussian Decomposition w.r.t $\mathbf{G}$ with parameter $s \geq (2\beta + \sqrt{w}) \cdot \sqrt{2\pi}$.*
- *LWE decoding of $g_{\mathbf{G}}(\mathbf{s}, \mathbf{e})$ for any $\mathbf{s} \in \mathbb{Z}_q^n$ and $\|\mathbf{e}\|_\infty \leq q/2 \cdot (2\beta + \sqrt{w})$.*

We remark that the general solutions provided by this corollary are of theoretical interest, and not suitable for practice. They are provided here only as a general feasibility result, in order to identify classes of good gadget matrices. The rest of the paper is dedicated to showing that by carefully choosing the gadget vector $\mathbf{g}$, one can obtain constructions and algorithms that are not only theoretically efficient, but also easy to implement and extremely fast.

4.4 Subgaussian Nearest Plane

Now we describe the subgaussian nearest plane algorithm, $SG_{NP}(\mathbf{B}, \mathbf{t})$, used in throughout this chapter. It is a randomized version of Babai's algorithm (Theorem 2.2.1), and it was first used in a theoretical sense by Alperin-Sheriff and Peikert [10]. The algorithm takes as input a target $\mathbf{t} \in \mathbb{R}^n$, a lattice basis $\mathbf{B}$, and its GSO. It returns a lattice point $\mathbf{x}$ that is randomly chosen so that $\mathbf{x} - \mathbf{t}$ is a subgaussian vector. The subgaussian parameter, as we will see, is $\max_i \|\tilde{\mathbf{b}}_i\|$.

Let $\mathbf{B}_k^*$ be the GSO of the basis $\mathbf{B}_k$. As in the nearest plane algorithm, we partition the lattice $\mathscr{L}(\mathbf{B}_k)$ into parallel planes $\mathscr{L}(\mathbf{B}_k) = \bigcup_{i \in \mathbb{Z}} (\mathscr{L}(\mathbf{B}_{k-1}) + i \cdot \mathbf{b}_k)$, but here we randomly round to the plane j or $j-1$, the planes between which the target lies, with a probability which depends

on the target's distance from the nearest plane instead of directly rounding. In more detail, given an input vector $\mathbf{t}$, we perform the following: First project the target orthogonally onto the span of $\mathbf{b}_k^*$ and store the coefficient $t \leftarrow \langle \mathbf{t}, \mathbf{b}_k^* \rangle / \|\mathbf{b}_k^*\|^2$. If $t \in \mathbb{Z}$, set $c \leftarrow t$. Otherwise, pick $c \leftarrow \lfloor t \rfloor + 1$ with probability $t \mod 1$ and $c \leftarrow \lfloor t \rfloor$ otherwise. Finally, return $c \cdot \mathbf{b}_k + SG_{NP}(\mathbf{B}_{k-1}, \mathbf{t} - c \cdot \mathbf{b}_k)$. The following lemma is easily proved by induction through expanding the expectation and the definition of the GSO.

Lemma 4.4.1 *The expected value of* $SG_{NP}(\mathbf{B}_k, \mathbf{t})$ *is* $\mathbf{t}$ *projected to* $\text{span}(\mathbf{B}_k)$.

Proof: We prove this by induction on the dimension. For the base case, let the basis and target be $b, t \in \mathbb{R}$, respectively. Let $p = t/b \mod 1$ (or equivalently $t/b - \lfloor t/b \rfloor$). Then the expectation of $SG_{NP}(b,t)$ is

$$\mathbb{E}[SG_{NP}(b,t)] = b[\lfloor t/b \rfloor (1-p) + (\lfloor t/b \rfloor + 1)p] = b(t/b) = t.$$

For the inductive step, we assume $\mathbb{E}[SG_{NP}(\mathbf{B}_{k-1}, \mathbf{t})] = \text{span}_{\mathbf{B}_{k-1}}(\mathbf{t})$. The coefficient c's expectation is $c' := \langle \mathbf{t}, \mathbf{b}_k^* \rangle / \|\mathbf{b}_k^*\|^2$ via the same computation as the base case. The law of iterated expectation gives us

$$\mathbb{E}[SG_{NP}(\mathbf{B}_k, \mathbf{t})] = c'\mathbf{b}_k + \sum_{i=1}^{k-1} \frac{\langle \mathbf{t} - c'\mathbf{b}_k, \mathbf{b}_i^* \rangle}{\|\mathbf{b}_i^*\|^2} \mathbf{b}_i^*.$$

The result follows from the definition of the GSO. □

To see $SG_{NP}(\mathbf{B}_k, \mathbf{t}) - \mathbf{t}$ is subgaussian, we view the space in the basis generated by $\mathbf{B}_k^*$. Since the coefficients of the output vector in this basis are chosen independently and by the definition of the GSO, $SG_{NP}(\mathbf{B}_k, \mathbf{t}) - \mathbf{t}$ is a subgaussian vector with parameter $\max_i \|\mathbf{b}_i^*\|$.

Lemma 4.4.2 $SG_{NP}(\mathbf{B}_k, \mathbf{t}) - \mathbf{t}$ *is a subgaussian vector with parameter* $\max_i \|\mathbf{b}_i^*\|$.

Proof: The result is seen through expressing $SG_{NP}(\mathbf{B}_k, \mathbf{t}) - \mathbf{t}$ in the GSO basis $\mathbf{B}_k^*$. Each coordinate c_i (coefficient of $\mathbf{b}_i^*$) is subgaussian with parameter $\sqrt{2\pi}$ and is independent of the

previous coordinates. As in Lemma 4.4.1's proof, the result follows from the definition of the GSO. □

4.5 Subgaussian Gadget Decomposition

In this section we present our main algorithms for the problem of *subgaussian gadget decomposition* using the gadget matrix $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^t$. Since this decomposition $\mathbf{G}^{-1}(\mathbf{u}) = (\mathbf{g}^{-1}(u_i))_{i=1}^n$ can be computed one component at a time (even in-parallel!) we restrict our attention to efficiently computing the subgaussian function $\mathbf{g}^{-1} : \mathbb{Z}_q \to \mathbb{Z}^k$ in the one-dimensional case, i.e., for $n = 1$.

The gadgets and algorithms in this section are parametrized by a "base" integer b, which we consider as fixed throughout the section, but can be used to achieve different efficiency/quality trade-offs. We distinguish two cases, depending on whether the modulus is a power $q = b^k$ of the base b, or an arbitrary integer $q < b^k$. In either case, no assumption is made about the factorization of the modulus q. Later in this chapter we will extend the gadgets and algorithms from this section to provide optimized treatment of large moduli with useful co-prime factorization $q = \prod_i q_i$, where the input $u \in \mathbb{Z}_q$ is given in CRT form $(u \bmod q_1, \ldots, u \bmod q_l)$.

All algorithms in this section use the same gadget $\mathbf{g}^t := (1, b, \cdots, b^{k-1})$, for $k = \lceil \log_b q \rceil$, but with different subgaussian decomposition procedures depending on the whether q is a power of b. Notice that $\mathbf{g}^t$ is a $\mathbb{Z}_q$-gadget of size k and quality $\beta = \sqrt{k}(b/2)$.

The main result of this section is summarized in the following theorem[2].

Theorem 4.5.1 *For any integer base $b > 1$, integer modulus $q > 1$, $k = \lceil \log_b q \rceil$ and gadget $\mathbf{g}^t = [1, b, \cdots, b^{k-1}]$, there is a subgaussian decomposition algorithm $\mathbf{g}^{-1}$ as follows:*

- *If $q = b^k$, the algorithm runs in linear $O(k)$ time (and space), uses $\log_2 q$ random bits, and achieves subgaussian parameter at most $(b-1)\sqrt{2\pi}$.*

[2]This theorem is most relevant when q is a relatively small modulus (say $q < 2^{64}$), so that arithmetic operations modulo q can be performed with unit cost. For larger moduli, the theorem will be used as a building block for a more complex algorithm using RNS/CRT representation for the elements of $\mathbb{Z}_q$.

- *If $q \neq b^k$, the algorithm runs in linear $O(k)$ time (and space), uses at most $k \log_2 q$ random bits, and achieves subgaussian parameter at most $(b+1)\sqrt{2\pi}$,*

Notice how the generic solution obtained by applying Theorem 4.3.1 to our gadget $\mathbf{g}$ only implies a polynomial time inversion algorithm with subgaussian parameter $(b+1) \cdot \sqrt{2k\pi}$, and quadratic $O(k^2)$ time complexity (after a cubic time $O(k^3)$ preprocessing). Depending on implementation details, this generic solution would also require the use of high precision floating point numbers[3] and a substantial amount of randomness for high precision sampling. By contrast, the solution described in Theorem 4.5.1 is much more efficient (linear time and space, with no need for preprocessing) and also achieves a smaller subgaussian parameter by a factor of $\sqrt{k}$. Moreover, our specialized algorithms use a relatively small (almost optimal) number of random bits, and can be implemented without the need for high-precision floating-point arithmetic.

A proof of Theorem 4.5.1 is given by the algorithms presented and analyzed in the next two subsections for the two separate cases $q = b^k$ and $q < b^k$.

4.5.1 Power-of-Base Case

Here we consider the subgaussian decomposition problem for the gadget

$$\mathbf{g} = (1, b, \ldots, b^{k-1})$$

when $q = b^k$, and the input is given as a positive coset representative $u \in \{0, 1, \cdots, q-1\}$. Conceptually, our solution to this problem is just a specialized/optimized version of the randomized-rounding variant of Babai's nearest plane algorithm [13, 10]. The general algorithm uses the Gram-Schmidt orthogonalization of a basis for the lattice $\Lambda_q^{\perp}(\mathbf{g}^t)$ associated to the gadget $\mathbf{g}$. The optimization is based on the observation (from [66]) that for our gadget $\mathbf{g}$ and modulus $q = b^k$,

[3] For a general integer basis $\mathbf{B}$, the GSO can have numbers with denominators as large as $\prod_i \|\mathbf{b}_i\|^2$.

the lattice $\Lambda_q^\perp(\mathbf{g}^t)$ has a very simple basis $\mathbf{S}$, and an even simpler GSO $\tilde{\mathbf{S}}$:

$$\mathbf{S} = \begin{pmatrix} b & & & \\ -1 & \ddots & & \\ & \ddots & b & \\ & & -1 & b \end{pmatrix}, \qquad \widetilde{\mathbf{S}} = b \cdot \mathbf{I}.$$

Using this special structure, there is no need to explicitly compute and store the GSO, and the randomized-rounding nearest-plane algorithm can be implemented in linear time and space $O(k)$. The specialized algorithm is best illustrated when $b = 2$, in which case it computes a randomized "bit" decomposition of u as follows:

1. For $i = 0, \cdots, k-1$:

   (a) if u is even, then set $x_i \leftarrow 0$,

   (b) if u is odd, then choose $x_i \leftarrow \{-1, +1\}$ uniformly at random

   Update $u \leftarrow (u - x_i)/2$.

2. Return $\mathbf{x} = (x_0, x_1, \cdots, x_{k-1})$.

This is essentially the same as the standard (deterministic) bit decomposition algorithm, except that when the bit is 1, we use a random ± 1 digit. Since ± 1 have the same parity modulo 2, the algorithm works as expected, with the only difference that now each digit is a zero-mean random variable, and the final output is subgaussian with parameter $\sqrt{2\pi}$.

We can modify this algorithm to an arbitrary base b as follows. Let $y := u \mod b \in \{0, \cdots, b-1\}$ for an input $u \in \mathbb{Z}_q$. Then, at each step, we pick the coset representative (of u with respect to $\mathbb{Z}_b$) with expectation 0 from the set $\{y - b, y\}$. The resulting algorithm is given in Figure 9. One can verify that this is the subgaussian nearest plane algorithm applied to the lattice $\mathscr{L}(\mathbf{S}) = \Lambda_q^\perp(\mathbf{g}^t)$, so the correctness of the algorithm is straightforward. Efficiency is also easily

Algorithm 9: $\mathbf{g}^{-1}(u)$ for $q = b^k$.

Input: $u \in \{0, 1, \cdots, q-1\}$

Output: subgaussian $\mathbf{x} \in \Lambda_u^{\perp}(\mathbf{g}^t)$ with parameter $(b-1)\sqrt{2\pi}$

```
Let x ← 0
for i ← 0, ··· , k − 1 do
    Let y ← u mod b ∈ {0, ··· , b − 1}.
    if y = 0 then
        x_i ← 0.
    else
        with probability y/b, x_i ← y − b, and x_i ← y otherwise.
    u ← (u − x_i)/b.
return x
```

analyzed by inspection. Notice that the algorithm is randomness efficient as it needs only one random number in $\mathbb{Z}_b$ for every interation, for a total of $k \cdot \log_2(b) = \log_2(q)$ random bits.

We remark that a similar algorithm is analyzed in [9], though with a loose bound on its subgaussian parameter (there is an unnecessary $\sqrt{k}$ factor in their subgaussian analysis). This section's main contribution is how to generalize the algorithm to arbitrary modulus q, as described in the next subsection.

4.5.2 Arbitrary Modulus, Arbitrary Base

Unfortunately, the (randomized) nearest plane algorithm $\Lambda_q^{\perp}(\mathbf{g}^t)$ does not specialize well when the modulus q is not a power of b. The reason is that, while we can still use the same gadget $\mathbf{g} = (1, b, \ldots, b^{k-1})$, the corresponding lattice $\Lambda_q^{\perp}(\mathbf{g}^t)$ has a slightly different basis $\mathbf{S}_q$ whose GSO is not diagonal, and not sparse. Our solution uses a technique developed in [42] for the discrete Gaussian sampling problem. Specifically, we use the fact that $\mathbf{S}_q$ admits a sparse,

triangular factorization

$$\mathbf{S}_q = \begin{pmatrix} b & & & q_0 \\ -1 & \ddots & & \vdots \\ & \ddots & b & q_{k-2} \\ & & -1 & q_{k-1} \end{pmatrix} = \begin{pmatrix} b & & & \\ -1 & \ddots & & \\ & \ddots & b & \\ & & -1 & b \end{pmatrix} \begin{pmatrix} 1 & & & d_0 \\ & \ddots & & \vdots \\ & & 1 & d_{k-2} \\ & & & d_{k-1} \end{pmatrix} = \mathbf{SD} \qquad (4.1)$$

where $(q_0, \cdots, q_{k-1})$ are the (base b) digits of q, and the last column of $\mathbf{D}$ is defined by the simple recurrence $d_i = \frac{d_{i-1}+q_i}{b}$ with initial condition $d_{-1} = 0$. (Note that $b^{i+1}d_i = q \mod b^{i+1} \in \{0, \cdots, b^{i+1}-1\}$.)

Then, on input $u \in \{0, 1, \cdots, q-1\}$, we proceed as follows:

1. Compute an arbitrary element $\mathbf{u} \in \mathbb{Z}^k$ of the lattice coset $\Lambda_u^\perp(\mathbf{g}^t)$, for example $\mathbf{u} = (u, 0, \ldots, 0)$.

2. Map $\mathbf{u}$ to $\mathbf{t} = \mathbf{S}^{-1}\mathbf{u}$ by solving a sparse system of linear equations $\mathbf{St} = \mathbf{u} \pmod q$.

3. Pick a subgaussian sample from the lattice coset $\mathscr{L}(\mathbf{D}) + \mathbf{t}$.

4. Apply the (sparse) linear transformation $\mathbf{S}$ to the sample, to obtain a subgaussian sample from $\Lambda_u^\perp(\mathbf{g}^t)$.

Here the (randomized) nearest plane algorithm admits a simple and efficient specialization because it is applied to a basis, $\mathbf{D}$, which has a diagonal GSO. The linear transformations $\mathbf{S}^{-1}$ and $\mathbf{S}$ can also be computed in linear time because $\mathbf{S}$ is sparse and triangular. As a result, the algorithm runs in linear time $O(k)$ and does not require any pre-processing. Finally, we get an output with subgaussian parameter $(b+1)\sqrt{2\pi}$ since $\mathbf{S}$ has small spectral norm.

The actual algorithm is given in Algorithm 10. The algorithm directly implements the outline given above, but it is specialized/optimized to avoid the explicit computation of the sparse matrices $\mathbf{S}, \mathbf{D}$, and to use only integer numbers (avoids floating point numbers). Details about the correctness and analysis of the algorithm are provided in the rest of this section.

Algorithm 10: $\mathbf{g}^{-1}(u)$

Input: $u \in \{0,1,\cdots,q-1\}$

Output: subgaussian $\mathbf{x} \in \Lambda_u^{\perp}(\mathbf{g}^t)$ with parameter $(b+1)\sqrt{2\pi}$

Let $\mathbf{u} \leftarrow [u]_b^k$, $\mathbf{x},\mathbf{y} \leftarrow \mathbf{0}$

$\mathbf{x} \leftarrow \mathbf{0}, \mathbf{q} = [q]_b^k$.

set $x_{k-1} \leftarrow 0$ with probability $(q-u)/q$ and $x_{k-1} \leftarrow -1$ otherwise.

for $i = k-2,\cdots,0$ **do**

  $u \leftarrow u - u_{i+1}b^{i+1}, q \leftarrow q - q_{i+1}b^{i+1}$.

  Let $c \leftarrow -(u + x_{k-1}q)$.

  if $c < 0$ **then**

    $p \leftarrow (c + b^{i+1})$, $z \leftarrow -1$.

  else

    $p \leftarrow c$, $z \leftarrow 0$.

  set $x_i \leftarrow z+1$ with probability p/b^{i+1} and $x_i \leftarrow z$ otherwise.

for $i \in \{0,\cdots,k-2\}$ **do**

  $y_i \leftarrow b \cdot x_i - x_{i-1} + x_{k-1} \cdot q_i + u_i$.

$y_{k-1} \leftarrow -x_{k-2} + x_{k-1} \cdot q_{k-1} + u_{k-1}$.

return $\mathbf{y}$.

Lemma 4.5.1 *The first loop of Algorithm 10 performs the subgaussian nearest plane algorithm on the lattice generated by* $\mathbf{D}$ *around target* $\mathbf{t} := -\mathbf{S}^{-1}[u]_b^k$.

Proof: Let $\mathbf{d}$ be the last column of $\mathbf{D}$. The last entry of $\mathbf{t}$ is $t_{k-1} = -u/b^k$ and the last entry of $\mathbf{d}$ is $d_{k-1} = q/b^k$. Therefore, we are randomly rounding x_{k-1} around the center $\left\langle \mathbf{t}, \tilde{\mathbf{d}} \right\rangle / \|\tilde{\mathbf{d}}\|^2 = -u/q \in (-1,0]$.

For the remainder of the loop, we note that $\mathbf{t} = -\mathbf{S}^{-1} \cdot \mathbf{u}$ has entries $t_i = -(\sum_{j=0}^{i} u_j \cdot b^j)/b^{i+1}$, represented by the recurrence relation $t_i = t_{i-1}/b + u_i/b$, $t_0 = -u_0/b$. This matches the recurrence relation for $\mathbf{d}$, $d_i = (\sum_{j=0}^{i} q_j \cdot b^j)/b^{i+1}$ since $\mathbf{d} = \mathbf{S}^{-1}[q]_b^k$, so we can compute the remaining centers for the nearest plane algorithm by these recurrences. Specifically, we are performing a randomized rounding around the centers $c_i = t_i - x_{k-1}d_i = -(\sum_{l=0}^{i} u_l \cdot b^l + x_{k-1} \cdot \sum_{j=0}^{i} q_j \cdot b^j)/b^{i+1} \in (-1,1)$. These centers are stored as c in the pseudocode. The variable z represents the two parallel planes (copies of $\mathscr{L}([\mathbf{d}_1,\cdots,\mathbf{d}_{i-1}])$ shifted by integer multiples of $\mathbf{d}_i$) separated by $\tilde{\mathbf{d}}_i$. The lemma follows. □

By storing $\mathbf{d} = \mathbf{S}^{-1}[q]_b^k$ in-advance, one can change the code to sample the first $k-1$ coordinates of $\mathbf{x}$ in-parallel since $\mathscr{L}(\mathbf{d}_0, \cdots, \mathbf{d}_{k-2}) = \mathbb{Z}^{k-1} \oplus \{0\}$. The proof of Theorem 4.5.1 follows below.

Proof: For the case $q = b^k$, Algorithm 9 returns a subgaussian sample $\mathbf{x} \in \Lambda_u^{\perp}(\mathbf{g}^t)$ with parameter $(b-1)\sqrt{2\pi}$ in time and space $O(\log_b q)$ while consuming $\log_2(q)$ of random bits by inspection, and Lemma 4.2.2.

Alternatively, let $q \neq b^k$. Now by Lemma 4.5.1, $\mathbf{x}$ after the first loop is so that $\mathbf{Dx}$ is the output of subgaussian nearest plane algorithm on $\mathbf{D}$ centered around $-\mathbf{S}^{-1}\mathbf{u}$. By Lemma 4.2.3, $\mathbf{S}_q\mathbf{x} + \mathbf{u}$ is a subgaussian vector with parameter $\sqrt{\lambda_{max}(\mathbf{S}\cdot\mathbf{S}^t)}\sqrt{2\pi}$, where $\lambda_{max}(\mathbf{S}\cdot\mathbf{S}^t)$ is the maximum eigenvalue of $\mathbf{S}\cdot\mathbf{S}^t$. A routine calculation for $\mathbf{S}\cdot\mathbf{S}^t$'s entries and the Geršgorin Circle Theorem (Theorem 2.1.3) imply $\lambda_{max}(\mathbf{S}\cdot\mathbf{S}^t) \leq (b+1)^2$. Since during each iteration in the first loop we draw a random number in $\mathbb{Z}_{b^i}$ to represent p, the algorithm consumes exactly $\log_2 b(1+2+\cdots+k) = \log_2 b \cdot (k^2+k)/2$ random bits. □

4.6 Gadget Decoding

Here we discuss our main algorithm for the problem of *LWE gadget decoding*, defined in this chapter's introduction, on the gadget matrix $\mathbf{G} = \mathbf{I}_n \otimes \mathbf{g}^t$ with entries in $\mathbb{Z}_q$, for an arbitrary modulus q. Given a $\mathbf{v}^t = \mathbf{s}^t\mathbf{G} + \mathbf{e}^t \in \mathbb{Z}_q^{nk}$ as input, we can break the vector into n components of length k, then decode (in-parallel) each component with respect to $\mathbf{g}^t$. Therefore, we focus on decoding $\mathbf{g}^t$ as a gadget for $\mathbb{Z}_q$.

Our algorithm and its respective gadgets are parameterized by an integer "base" b. We consider b as fixed in this section, though varying b for a fixed modulus q yields efficiency/quality trade-offs for these gadgets. Later in this chapter, we present a CRT gadget that can be used to efficiently decode an input given in CRT form.

Let $k = \lceil \log_b q \rceil$ and the gadget be $\mathbf{g}^t = (1, b, \cdots, b^{k-1})$. The vector $\mathbf{g}^t$ is a size k gadget of quality $(b/2)\sqrt{k}$ for $\mathbb{Z}_q$. The results in this section are summarized in the following theorem.

Theorem 4.6.1 *For every modulus q, and gadget* $\mathbf{g}^t = (1, b, \cdots, b^{k-1})$*, there is a time and space* $O(k)$ *algorithm decoding* $\mathbf{g}^t$ *with tolerance* $q/2(b+1)$.

A proof of Theorem 4.6.1 is given by the algorithm presented in this section. Note, Theorem 4.3.1 implies a polynomial time decoding algorithm for $\mathbf{g}^t$ with error tolerance $\|\mathbf{e}\|_\infty \leq q/2\sqrt{k}(b+1)$. Our decoding algorithm is more efficient and has a higher error tolerance by a factor $\sqrt{k}$ than the general gadgets decoding guarantee given by Theorem 4.3.1.

An optimized, linear time and space $O(k)$, decoding algorithm is given in [66] for the case $q = b^k$. The reason for this algorithm's efficiency is that the commonly used basis for $\Lambda_{b^k}(\mathbf{g}^t)$ results in a linear time nearest plane algorithm. In more detail, a basis for $\Lambda_{b^k}(\mathbf{g}^t)$ in this case is the triangular matrix $\mathbf{B}_{b^k} = b^k \cdot \mathbf{S}^{-t}$, where $\mathbf{S}$ is the commonly used basis for $\Lambda^{\perp}_{b^k}(\mathbf{g}^t)$ presented in the preliminaries, and this basis has a GSO of $(q/b) \cdot \mathbf{I}$.

However, the simple decoding idea presented in [66] fails when $q \neq b^k$. Because $\Lambda_q(\mathbf{g}^t)$'s commonly used basis has a dense GSO, Babai's nearest plane algorithm takes time $O(k^2)$ and space $O(k^3)$ when naively applied on $\Lambda^{\perp}_q(\mathbf{g}^t)$.

Efficient Decoding Algorithm

The intuition for our algorithm is best initially viewed through the case when $q = b^k$. Given an input $\mathbf{v}$, another way to decode the lattice $\Lambda_{b^k}(\mathbf{g}^t)$ is to use $\mathbf{S}^t$ as a linear transformation, decode $\mathbf{S}^t\mathbf{v}$ to the lattice $b^k \cdot \mathbb{Z}^k$ with the nearest plane algorithm, then map the nearest point in $b^k \cdot \mathbb{Z}^k$ back to $\Lambda_{b^k}(\mathbf{g}^t)$. This leads to a slightly stronger condition on the noise vector $\mathbf{e}$ since we now need $\mathbf{S}^t\mathbf{e} \in \mathscr{P}_{1/2}(q \cdot \mathbf{I})$, which is satisfied if $\|\mathbf{e}\|_\infty < q/2(b+1)$. Though there is no need to do this given the algorithm in [66], this is essentially what we will do in the case when $q \neq b^k$.

Overview

The overview of our efficient decoding algorithm for an arbitrary modulus is as follows. First recall the sparse, triangular factorization of $\Lambda^{\perp}_q(\mathbf{g}^t)$'s commonly used basis given in the preliminaries, $\mathbf{S}_q = \mathbf{SD}$. The duality relation for q-ary lattices, $\Lambda_q(\mathbf{g}^t) = q \cdot \Lambda^{\perp}_q(\mathbf{g}^t)^*$, dictates that a basis for $\Lambda_q(\mathbf{g}^t)$ is $q \cdot \mathbf{S}_q^{-t} = \mathbf{S}^{-t}(q \cdot \mathbf{D}^{-t})$. Luckily, the matrix $\mathbf{D}^{-t}$ is sparse with a diagonal

Algorithm 11: DECODEG$(\mathbf{v}, b, \mathbf{r}[q]_b^k)$

Input: $\mathbf{v} \in \mathbb{Z}^k$, b, and $\mathbf{q} = [q]_b^k$.
Output: $s \in \mathbb{Z}_q$ where $\mathbf{v} = s\mathbf{g}^t + \mathbf{e}^t$ as long as $\|\mathbf{e}\|_\infty < q/2(b+1)$.

for $i \leftarrow 0, \cdots, k-2$ **do**
    $v_i \leftarrow bv_i - v_{i+1}$.
$v_{k-1} \leftarrow b \cdot v_{k-1}$.
Let $\mathbf{x} \leftarrow \mathbf{0}$ and reg $\leftarrow 0$.
for $i \leftarrow 0, \cdots, k-2$ **do**
    $x_i \leftarrow \lceil v_i/q \rfloor$ and reg $\leftarrow$ reg$/b + b^{k-1} \cdot q_i$.
    $v_{k-1} \leftarrow v_{k-1} + x_i \cdot$ reg.
$x_{k-1} \leftarrow \lceil v_{k-1}/b^k \rfloor$.
Let $s \leftarrow x_{k-1}$ and reg $\leftarrow 0$.
for $i \leftarrow k-2, \cdots, 0$ **do**
    reg $\leftarrow b \cdot$ reg $+ q_{i+1}$.
    $s \leftarrow s + x_i \cdot$ reg
return $s \mod q$.

GSO, and $\mathscr{P}_{1/2}(q \cdot \tilde{\mathbf{D}}^{-t}) \supseteq \mathscr{P}_{1/2}(q \cdot \mathbf{I})$ (meaning we can decode as long as $\|\mathbf{e}\|_\infty < q/2(b+1)$). Therefore, we can decode $\mathbf{g}^t$ by the following.

1. Given $\mathbf{v}$, first apply $\mathbf{S}^t$ as a linear transformation.

2. Then, decode the vector $\mathbf{S}^t\mathbf{v}$ to the lattice generated by $q\mathbf{D}^{-t}$ using the nearest plane algorithm.

Both steps can be computed in linear time and space, $O(k)$, given the sparsity of $\mathbf{S}$ and $q\mathbf{D}^{-t}$, and $q\mathbf{D}^{-t}$'s diagonal GSO.

The pseudocode for our algorithm is shown in DECODEG. In short, the algorithm has three components, where each is represented by a loop in the pseudocode. These components are to first compute the linear transformation on the input $\mathbf{v} \leftarrow \mathbf{S}^t\mathbf{v}$, then to run the nearest plane algorithm on the lattice generated by $q \cdot \mathbf{D}^{-t}$, and finally to return s represented as the first entry of the nearest lattice point in $\Lambda_q(\mathbf{g}^t)$ modulo q. The proof of Theorem 4.6.1 follows from Lemmas 4.6.1 and 4.6.2 below.

Lemma 4.6.1 *The second loop in* DECODEG *is an instantiation of Babai's nearest plane algorithm on the lattice $q \cdot \mathbf{D}^{-t}$ given target $\mathbf{S}^t\mathbf{v}$, running in time and space $O(k)$.*

Proof: Recall the structure of $\mathbf{D}$ from the previous chapter, $\mathbf{D} = [\mathbf{M}|\mathbf{d}]$ where $\mathbf{M}^t = [\mathbf{I}_{k-1}|\mathbf{0}]$ and $\mathbf{d}$ has entries $d_i = (q \mod b^{i+1})/b^{i+1}$, with $q \mod b^{i+1} \in \{0, 1, \cdots, b^{i+1} - 1\}$. Then, it follows that $q\mathbf{D}^{-t}$ has a similar triangular, sparse structure. This is given by $q \cdot \mathbf{D}^{-t} = \begin{pmatrix} q\mathbf{I}_{k-1} & \mathbf{0} \\ \mathbf{c}^t & b^k \end{pmatrix}$ and the vector $\mathbf{c} \in \mathbb{Z}^{k-1}$ has entries $c_i = -b^{k-1-i} \cdot (q_0 + bq_1 + \cdots + b^i q_i) = -b^{k-1-i} \cdot (q \mod b^{i+1}) \in [-q, 0]$. Further, the entries of $\mathbf{c}$ satisfy the recurrence relation $-c_i = \frac{-(c_{i-1})}{b} + b^{k-1}q_i$ with the initial condition $-c_0 = b^{k-1}q_0$. The variable reg in DECODEG stores c_i, and it is updated using the recurrence relation for $\mathbf{c}$. The vector $\mathbf{x}$ in the pseudocode stores the coefficients of the nearest lattice point expressed in the basis $q\mathbf{D}^{-t}$. The Lemma follows by inspection. □

Lemma 4.6.2 *The last loop in* DECODEG *computes $s \mod q$ in time and space $O(k)$.*

Proof: Represent the first row of $\mathbf{B} = \mathbf{S}^{-t}q\mathbf{D}^{-t}$ as $\mathbf{h}$, and note $\langle \mathbf{h}, \mathbf{x} \rangle = s \mod q$. A careful analysis of $q\mathbf{D}^{-t}$ and $\mathbf{S}^{-t}$ gives us an expression for $\mathbf{h}$'s entries: $h_i = q_{i+1} + bq_{i+2} + \cdots + b^{k-i-2}q_{k-1} = \frac{q-(q \mod b^{i+1})}{b^{i+1}}$ for $i \in \{0, 1, \cdots, k-2\}$ and $h_{k-1} = 1$. All but the last entry of $\mathbf{h}$ satisfy the recurrence relation $h_i = q_{i+1} + b \cdot h_{i+1}$ for $i \in \{0, \cdots, k-2\}$, with an initial value of q_{k-1} (which is *not* the actual value of $\mathbf{h}$'s last entry). We use this recurrence relation to compute $\mathbf{h}$'s entries one at a time in the last loop, stored in the variable reg. The Lemma follows by inspection. □

4.7 Gadgets for the CRT Representation

Many applications of lattice gadgets require a large modulus that, for secure and functional sets of parameters, surpasses the native 64-bit integer arithmetic in a modern machine's hardware. One common method to circumvent the use of multi-precision numbers is to pick a

modulus of the form $q = \prod q_i$ with each q_i less than 64 bits. Then, one can store an element $u \in \mathbb{Z}_q$ as its *Chinese Remainder* representation (CRT form[4]) as $(u \mod q_1, \cdots, u \mod q_l)$ and perform computations via the Chinese Remainder Theorem, utilizing the ring isomorphism $\mathbb{Z}_q \cong \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_l}$. Simple forms of the gadget matrix (e.g. power of two matrix) are not compatible with this representation because the binary digits of a number cannot be easily recovered from the CRT components without a costly reconstruction phase involving large numbers modulo q.

In this section, we discuss a gadget for the CRT form. As usual, the gadget admits a compact (implicit) representation, and does not need to be computed and stored explicitly. Most importantly, the gadget allows us to use the algorithms in Sections 4.5 and 4.6 in order to perform subgaussian decomposition, discrete Gaussian sampling, and LWE gadget decoding all given input represented in CRT form. This has several theoretical and practical advantages: (1) the algorithms can be directly used by efficient applications that already store their numbers in CRT form, (2) our algorithms can be easily parallelized as they operate on each CRT component independently, (3) all algorithms only require arithmetic on small numbers (at most $\max_i q_i$) even if the modulus $q = \prod_i q_i$ may be very big. (Efficient solutions to Discrete Gaussian Sampling for the individual moduli q_i, as needed by our CRT DGS algorithm, are given in [66, 42].) We remark that a balanced, deterministic digit decomposition is provided in [38, 71], and an LWE decoding algorithm for a CRT/RNS hybrid gadget for general rings is given in the library's code[5] (without an analysis). Our results are summarized in the following theorem. We emphasize the analysis below assumes integer operations, including reductions modulo q_i, are done in constant time. This is because our algorithms are best implemented when each q_i is less than 64 bits, avoiding the use of multi-precision numbers.

Theorem 4.7.1 *Let q have factorization $q = \prod_{i=1}^{l} q_i$ into coprime factors $\{q_i\}$, $(b_i)_{i=1}^{l}$ be an l-tuple of bases with $b_i < q_i$ for all i, and let $k = \sum k_i$ where $k_i = \lceil \log_{b_i} q_i \rceil$. There exists a*

[4]This is also known as the residue number system (RNS) in previous works.

[5]https://github.com/cpeikert/Lol/blob/master/lol/Crypto/Lol/Gadget.hs

Algorithm 12: Sampling in CRT form.

Input: $(u_1, \cdots, u_l)$
Output: $\mathbf{g}_{CRT}^{-1}(u_1, \cdots, u_l)$.
1 **for** $i \in \{1, \cdots, l\}$ **do**
2 $\quad \mathbf{x}_i \leftarrow \mathbf{g}_i^{-1}(u_i)$.
3 **return** $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_l)$.

Algorithm 13: Decoding in CRT form.

Input: $\mathbf{v}^t = s \cdot \mathbf{g}_{CRT} + \mathbf{e}^t \mod q$
Output: $(s_1, \cdots, s_l)$.
1 Let $\mathbf{v} = (\mathbf{v}_1, \cdots, \mathbf{v}_l)$ for each $\mathbf{v}_i \in \mathbb{Z}_q^{k_i}$.
2 **for** $i \in \{1, \cdots, l\}$ **do**
3 $\quad s_i \leftarrow \text{DECODECRT}(\mathbf{v}_i)$
4 **return** $(s_1, \cdots, s_l)$.

Figure 4.1. Pseudocode for the parallel algorithms given in Theorem 4.7.1. We let $\mathbf{g}_i^{-1}(\cdot)$ denote either the subgaussian decomposition algorithm given in Section 4.5 or a discrete Gaussian sampler.

gadget, $\mathbf{g}_{CRT}^t$, for $\mathbb{Z}_q$ of size k and quality $\max_i b_i/2$. Further, the gadget satisfies the following properties:

- *Subgaussian decomposition can be performed in-parallel with l processors, each using time and space $O(k_i)$, consuming less than $k_i \log_2 q_i$ random bits ((\log_2(q_i)$ random bits if $q_i = b_i^{k_i}$)) and with parameter at most $(\max_i(b_i)+1)\sqrt{2\pi}$.*
- *For any $\varepsilon > 0$, discrete Gaussian sampling can be performed in-parallel with l processors, each in time and space $O(k_i)$ with width $s \geq O(b_j^{1.5})\eta_\varepsilon(\mathbb{Z}^{k_j})$ for index j maximizing $\sqrt{2b_j}(b_j+1) \cdot \eta_\varepsilon(\mathbb{Z}^{k_j})$.*
- *$\mathbf{g}_{CRT}^t$ is decodable in-parallel with l processors in time and space $O(k_i)$ with tolerance $q/2(\max_i(b_i)+1)$.*

As expected, each processor gets slightly more efficient whenever $q_i = b_i^{k_i}$. The algorithms are represented in Figure 4.1.

The CRT Gadget

For each coprime factor q_i, fix the *base-b_i* gadget vector as $\mathbf{g}_i^t := (1, b_i, \cdots, b_i^{k_i-1})$ where $k_i = \lceil \log_{b_i}(q_i) \rceil$. Let $k = \sum_i k_i$, $q_i^* = q/q_i$, and $\hat{q}_i = (q_i^*)^{-1} \mod q_i$. Consider the gadget vector, which we call the *general CRT gadget*, $\mathbf{g}_{CRT}^t = (q_1^*\hat{q}_1 \cdot \mathbf{g}_1^t, \cdots, q_l^*\hat{q}_l \cdot \mathbf{g}_l^t) \mod q \in \mathbb{Z}_q^{1\times k}$. This is

Algorithm 14: DECODECRT$(\mathbf{v}_i, b_i, \mathbf{t} = [q_i]_{b_i}^{k_i}, q, q_i^*)$

Input: $\mathbf{v}_i \in \mathbb{Z}^{k_i}$, b_i, q_i^*, q, and $\mathbf{t} = [q_i]_{b_i}^{k_i}$.

Output: $s \mod q_i$ where $\mathbf{v} = s\mathbf{g}^t + \mathbf{e}^t \mod q$ as long as $\|\mathbf{e}\|_\infty < q/2(b_i+1)$.

for $j \leftarrow 0, \cdots, k_i - 1$ **do**
  $v_j \leftarrow b_j v_j - v_{j+1}$.
Let $\mathbf{x} \leftarrow \mathbf{0}$.
for $j \in \{0, \cdots, k_i - 2\}$ **do**
  $x_j \leftarrow \lceil v_j/q \rfloor$.
$x_{k-1} \leftarrow \lceil (v_{k-1} - \left\langle \mathbf{c}, \mathbf{x}_0^{k-2} \right\rangle)/(q_i^* b_i^{k_i}) \rfloor$.
Let $s_i \leftarrow x_{k-1}$ and reg $\leftarrow 0$.
for $j \leftarrow k_i - 2, \cdots, 0$ **do**
  reg $\leftarrow b \cdot$ reg $+ t_{j+1} \cdot q_i^*$.
  $s_i \leftarrow s_i + x_j \cdot$ reg
return $s_i \mod q_i$.

a generalization of the gadgets (or implicit in algorithms) used in [22, 53, 55, 15]. As before, the gadget matrix is the block-diagonal matrix $\mathbf{G} := \mathbf{I}_n \otimes \mathbf{g}_{CRT}^t$. Theorem 4.7.1 follows from the fact $\Lambda_q^\perp(\mathbf{g}_{CRT}^t) = \Lambda_{q_1}^\perp(\mathbf{g}_1^t) \oplus \cdots \oplus \Lambda_{q_l}^\perp(\mathbf{g}_l^t)$, Theorem 4.5.1, and Proposition 3.1 in [42]. The parallel decoding algorithm is obtained by a slight adaptation to DECODEG presented in Section 4.6, and is analyzed in the Section 4.7.1. We prove the direct sum decomposition of $\Lambda_q^\perp(\mathbf{g}_{CRT}^t)$.

Proof: For the inclusion $\supseteq$, let $\mathbf{x}_i \in \Lambda_{q_i}^\perp(\mathbf{g}_i^t)$ be arbitrary with $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_l)$ as their concatenation. Then, $\langle \mathbf{x}_i, \mathbf{g}_i^t \rangle = aq_i \in q_i \cdot \mathbb{Z}$ and $\langle \mathbf{x}, \mathbf{g}_{CRT} \rangle \mod q = \sum_{i=1}^l q_i^* \hat{q}_i \langle \mathbf{x}_i, \mathbf{g}_i \rangle \mod q = 0 + \cdots + 0 \mod q$. We prove the converse by inducting on l, the number of q's coprime factors. The base case is routine. Now consider $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_l) \in \Lambda_q^\perp(\mathbf{g}_{CRT}^t)$ with $\mathbf{x}_i \in \Lambda_{q_i}^\perp(\mathbf{g}_i^t)$ for $i = 0, \cdots, l-1$ and $\mathbf{x}_l \in \mathbb{Z}^{k_l}$. By the inductive hypothesis, $\langle \mathbf{x}, \mathbf{g}_{CRT} \rangle \mod q = q_l^* \hat{q}_l \cdot \langle \mathbf{x}_l, \mathbf{g}_l^t \rangle = 0 \mod q$. Viewing this equation in $\mathbb{Z}$ and dividing both sides by q_l^* implies $\hat{q}_l \cdot \langle \mathbf{x}_l, \mathbf{g}_l \rangle \mod q_l = 0$. Finally, we conclude $\langle \mathbf{x}_l, \mathbf{g}_l \rangle \mod q_l = 0$ since $\hat{q}_l$ is a multiplicative unit in $\mathbb{Z}_{q_l}$. □

4.7.1 Decoding the CRT Gadget

Here we show how the efficient gadget decoding algorithm from Section 4.6 adapts to the general CRT gadget described in Section 4.7. Recall the decomposition of $\mathbf{g}^t$'s lattice,

$\Lambda_q^{\perp}(\mathbf{g}^t) = \Lambda_{q_1}^{\perp}(\mathbf{g}_1^t) \oplus \cdots \oplus \Lambda_{q_l}^{\perp}(\mathbf{g}_l^t) = \mathscr{L}(\mathbf{S}_{q_1}) \oplus \cdots \oplus \mathscr{L}(\mathbf{S}_{q_l})$. The duality relation for q-ary lattices yields $\Lambda_q(\mathbf{g}^t) = q \cdot (\Lambda_q^{\perp}(\mathbf{g}^t))^* = q \cdot \left(\bigoplus_i \mathscr{L}(\mathbf{S}_{q_i}^{-t}\mathbf{D}_{q_i}^{-t})\right) = \left(\bigoplus_i \mathscr{L}(\mathbf{S}_{q_i}^{-t} q_i^* \cdot (q_i \cdot \mathbf{D}_{q_i}^{-t}))\right)$.

Now we have a clear way to decode the general CRT gadget. First, break the input into l blocks, $\mathbf{v}^t = s\mathbf{g}^t + \mathbf{e}^t \mod q = (\mathbf{v}_1^t, \cdots, \mathbf{v}_l^t)$ where $\mathbf{v}_i^t = s \cdot q_i^* \hat{q}_i \mathbf{g}_i^t + \mathbf{e}_i^t \mod q$. Then, we compute the following. First, transform $\mathbf{v}_i$ to $\mathbf{S}_{q_i}^t \mathbf{v}_i$. Then, decode $\mathbf{S}_{q_i}^t \mathbf{v}_i$ to the lattice $q_i^*(q_i \mathbf{D}_{q_i}^{-t})$. Finally, return $s \mod q_i$. The pseudocode is given as the algorithm DECODECRT. Another change is that we store the vector $\mathbf{c}$ in memory. Recall, $\mathbf{c}$ has $k-2$ entries of the form $c_j = -b_i^{k_i-1-j}(q_i \mod b_i^j)$. Note that the correctness condition of our algorithm is still $\|\mathbf{e}^t\|_\infty < q/2(\max_i(b_i)+1)$.

Decoding in CRT Form

Here we describe how DECODECRT can decode $\mathbf{v} = s\mathbf{g} + \mathbf{e}$ where the input is given in its CRT representation. The ideas sketched here follow from [55]. The linear transformation $\mathbf{v} \to \mathbf{S}^t\mathbf{v}$ is easily computed given the CRT form of $\mathbf{v}$. Really, we are only concerned with divisions and integer rounding. In the second loop, note that $x_j \leftarrow \lceil v_j/q \rfloor = \lceil \sum_{o=1}^{l} [(v \mod q_o) \cdot (\hat{q_o}/q_o)] \rfloor$. Next we consider the line $x_{k-1} \leftarrow \lceil (v_{k-1} + \left\langle \mathbf{c}, \mathbf{x}_0^{k-2} \right\rangle)/(q_i^* b_i^{k_i}) \rfloor$. First, note that $v_{k-1}/(b_i^{k_i} q_i^*) = b_i^{-k_i} \cdot \sum_{o=1}^{l} (v_{k-1} \mod q_o) \cdot \hat{q}_o(q_i/q_o)$. This should be a small number in nearly all practical instantiations. Lastly, we note that we return s in CRT form, but we can alter the algorithm to return $s \in (-q/2, q/2]$ via a simple change. The s computed in the last loop is actually $s \cdot q_i^* \hat{q}_i$. So, we can remove the $\mod q_i$ in the return statement and sum up the output from the l parallel processors, $\sum_i (s \cdot q_i^* \hat{q}_i) = s \cdot \sum_i (q_i^* \hat{q}_i) = s \cdot 1 \mod q$.

4.8 Toolkit Implementation and Its Application

4.8.1 Software Implementation

We implemented most of the algorithms presented in this work in PALISADE [73], a modular open-source lattice cryptography library that includes ring-based implementations of homomorphic encryption, proxy re-encryption, identity-based encryption, attribute-based encryption, and other lattice schemes. More concretely, we added a new lattice gadget toolkit

module to PALISADE that implements the following algorithms:

- Subgaussian gadget decomposition (Algorithm 10) for arbitrary moduli and gadget bases.
- Efficient gadget in CRT representation, enabling both trapdoor sampling and subgaussian gadget decomposition in the CRT representation.
- Subgaussian gadget decomposition for cyclotomic rings both in positional and CRT number systems, which wraps around Algorithm 10.

The toolkit module complements/improves the lattice gadget algorithms previously added to PALISADE, such as trapdoor sampling for cyclotomic rings proposed in [42] and implemented in [52, 37]. The full lattice gadget capability will be included in the next major public release of PALISADE.

4.8.2 Optimized Variant of Key-Policy Attribute-Based Encryption

We use the lattice gadget toolkit algorithms to build and implement a full RNS/CRT variant of the short-secret Key-Policy Attribute-Based Encryption (KP-ABE) scheme originally proposed in [19] and implemented for cyclotomic rings in [39]. The KP-ABE scheme is a complex cryptographic primitive that can be used for attribute-based access control applications, as well as a building block for audit log encryption, targeted broadcast encryption, predicate encryption, functional encryption, and some forms of program obfuscation [19, 49].

Overview

ABE is a public key cryptography primitive that enables the decryption of a ciphertext by a user only if a specific access policy (defined over ℓ attributes) is satisfied. In the key-policy scenario, a message is encrypted using the attribute values as public keys, and a specific access policy is typically defined afterwards. When the access policy becomes known, a secret key for the policy is generated (using trapdoor sampling in our KP-ABE scheme), and the ciphertexts

and public keys are homomorphically evaluated over the policy circuit (using a GSW-type homomorphic multiplication in our KP-ABE scheme).

The short-secret KP-ABE scheme is a tuple of functions, namely `Setup`, `Encrypt`, `EvalPK`, `KeyGen`, `EvalCT`, and `Decrypt`, whose definitions are:

- $\textsc{Setup}(1^\lambda, \ell) \rightarrow \{\texttt{MPK}, \texttt{MSK}\}$: Given a security parameter λ and the number of attributes ℓ, a trusted private key generator (PKG) generates a master public key `MPK` and a master secret key `MSK`. `MPK` contains the ABE public parameters while `MSK` includes the trapdoor that is used by PKG to generate secret keys for access policies.

- $\textsc{Encrypt}(\mu, \mathbf{x}, \texttt{MPK}) \rightarrow \mathbf{C}$: Using `MPK` and attribute values $\mathbf{x} \in \{0,1\}^\ell$, sender encrypts the message μ and outputs the ciphertext $\mathbf{C}$.

- $\textsc{EvalPK}(\texttt{MPK}, \mathbf{x}, f) \rightarrow \texttt{PK}_f$: Homomorphically evaluate `MPK` over a policy (Boolean circuit) $f : \{0,1\}^\ell \rightarrow \{0,1\}$ to generate a public key $\texttt{PK}_f$ for the policy f.

- $\textsc{KeyGen}(\texttt{MSK}, \texttt{MPK}, \texttt{PK}_f) \rightarrow \texttt{SK}_f$: Given `MSK`, `MPK` and policy-specific $\texttt{PK}_f$, PKG generates the secret key $\texttt{SK}_f$ corresponding to f. PKG sends $\texttt{SK}_f$ to the receiver that is authorized to decrypt ciphertexts encrypted under f.

- $\textsc{EvalCT}(\mathbf{C}, \mathbf{x}, f) \rightarrow \mathbf{C}_f$: Homomorphically evaluate $\mathbf{C}$ over the policy f to generate the ciphertext $\mathbf{C}_f$.

- $\textsc{Decrypt}(\mathbf{C}_f, \texttt{SK}_f) \rightarrow \bar{\mu}$: Given the homomorphically computed ciphertext $\mathbf{C}_f$ and corresponding secret key $\texttt{SK}_f$, find $\bar{\mu}$, which is the same as the original message μ if the receiver has the secret key matching the policy f.

The most computationally expensive operations are EvalPK and EvalCT, which homomorphically evaluate a circuit of depth $\lceil \log_2 \ell \rceil$ using the GSW homomorphic multiplication approach. At each level of a Boolean circuit composed of NAND gates (which are used for benchmark evaluation in [39]), the algorithms compute matrix products $\mathbf{B}_{2i}\mathbf{G}^{-1}(-\mathbf{B}_i)$ and

$\left(\mathbf{G}^{-1}(-\mathbf{C}_i)\right)^t \mathbf{C}_{2i}$ for public keys and ciphertexts, respectively. Here, $\mathbf{B}_i \in R_q^{1\times m}$, $\mathbf{C}_i \in R_q^m$, $R_q = \mathbb{Z}_q[x]/\langle x^n+1\rangle$, and $m = \lceil \log_b q \rceil + 2$ (the latter corresponds to the Ring-LWE trapdoor construction). Note that that the gadget $\mathbf{G}$ is extended in this case to m by adding two zero entries to the decomposed digits.

The work [39] presents a CPU implementation of the ring variant of the KP-ABE scheme along with an efficient GPU implementation for policy evaluation and encryption. The CPU implementation was done for a binary gadget base and used the conversion from CRT to the positional number system for digit decomposition both in trapdoor sampling and gadget decomposition. To avoid the linear noise growth $O(nm)$ in gadget decomposition, the authors used a balanced digit decomposition, namely the binary non-adjacent form (NAF), that replaces digits in (0,1) with a zero-centered representation in (-1,0,1). Although this approach allows one to achieve a heuristic growth close to $O(\sqrt{nm})$ in the case of the KP-ABE scheme, the noise properties depend on the randomness of the input, i.e., this approach is deterministic.

The CPU runtimes for policy evaluation and encryption operations in [39] were far from practical (the CPU results only for ℓ up to 8 are presented), and hence the authors developed an efficient GPU implementation for these operations.

For detailed algorithms of the KP-ABE scheme, the reader is referred to [39].

Our Optimizations

We present a full CRT/RNS ring variant of the KP-ABE scheme that leverages the lattice gadget toolkit to significantly (by more than one order of magnitude) speed up the policy evaluation operations. In particular, our implementation includes the following optimizations as compared to [39]:

- The subgaussian gadget decomposition in CRT representation to minimize the noise growth instead of the NAF decomposition with the conversion from CRT representation to positional system. This provides a theoretical guarantee of the square-root noise growth. To achieve the repeatability of randomized decomposition in EVALPK and EVALCT, we

use the same seed for the random operations in subgaussian gadget decomposition. The seed is treated as part of the master public key.

- The CRT variant of trapdoor sampling using the gadget decomposition technique discussed in this paper in contrast to the multiprecision digit decomposition in [39].
- The RNS/CRT scaling proposed in [54] for decryption in contrast to the multiprecision scaling.
- Increased gadget base b (both in trapdoor and subgaussian gadget decomposition) instead of the binary base.

Parameter Selection

As the correctness constraint in [39] was derived for the classical binary-base gadget decomposition, we provide here a modified version incorporating the effect of a larger gadget base for the case of subgaussian gadget decomposition:

$$q > 4C_1 s\sigma\sqrt{mn}\left(b\sqrt{mn}\right)^d, \tag{4.2}$$

where $C_1 = 128$, $s = C \cdot \sigma^2(b+1) \cdot (\sqrt{n\log_b q} + \sqrt{2n} + 4.7)$, $C = 1.8$, $\sigma \approx 4.578$, and $d = \lceil \log_2 \ell \rceil$. Here, C and C_1 are empirical parameters chosen the same way as in [39].

The differences compared to [39] are the b factor in the exponentiation base (as the digits vary between $-b$ and b in subgaussian gadget decomposition) and a (b+1) factor in the expression for s (contributed by Gaussian sampling; see [42, 37] for a more detailed discussion of the Gaussian distribution parameter for arbitrary gadget bases).

4.9 Experimental Results

We ran the experiments in PALISADE version 1.2, which includes NTL version 10.5.0 and GMP version 6.1.2. The evaluation environment was a commodity desktop computer system

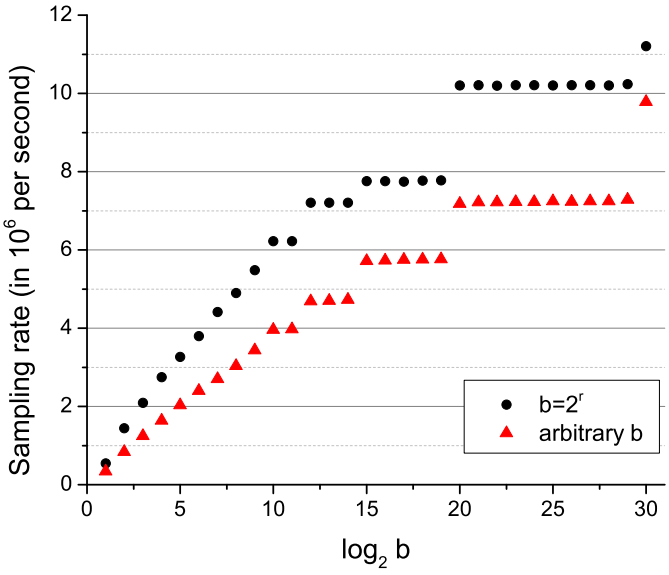

Figure 4.2. Runtime baseline of subgaussian sampling rate for native uniformly random integers (w.r.t a 60-bit modulus). When $b = 2^r$, the modulo reduction in digit decomposition is performed by simple bit shifting. When b is arbitrary, the slower hardware modulo operation is used.

with an Intel Core i7-3770 CPU with 4 cores rated at 3.40GHz and 16GB of memory, running Linux CentOS 7. The compiler was g++ (GCC) 5.3.1.

4.9.1 Subgaussian Gadget Decomposition

The experiments described in this section were all performed in the single-threaded mode. The goal of these results is to provide the performance baselines for subgaussian gadget decomposition, demonstrate the benefits of the efficient gadget in CRT representation, and illustrate the effect of subgaussian sampling on the noise growth in GSW-type products.

Figure 4.2 shows the dependence of subgaussian gadget decomposition rate (per decomposed integer) on the gadget base for native (64-bit) integers. The results are shown both for a power-of-two base, which supports fast modulo reduction by bit shifting, and an arbitrary base, which requires a division-based modulo operation on x86 architectures. In our implementation, the native arithmetic is a building block for performing operations in CRT representation for

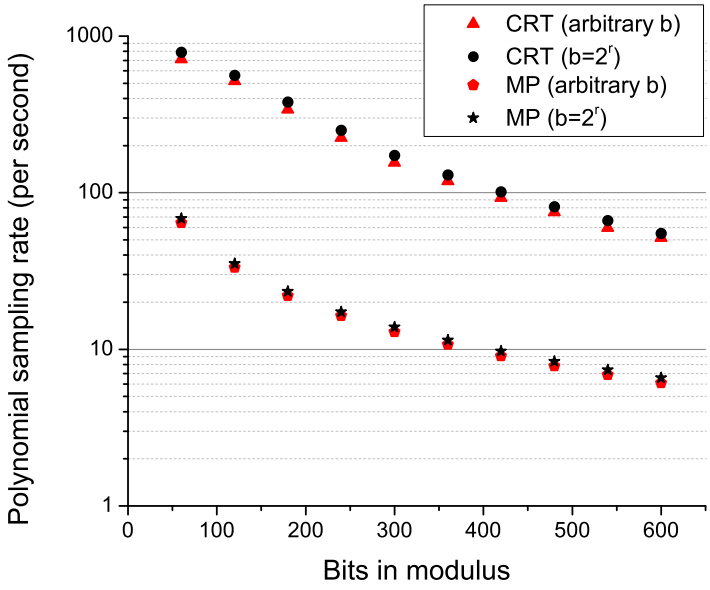

Figure 4.3. Comparison of sampling rates for CRT and multiprecision (MP) variants of subgaussian gadget decomposition for ring elements with 4096 coefficients and 60-bit CRT moduli at $r = \lceil \log_2 b \rceil = 20$.

integers that are larger than 60 bits, and, therefore, these results can be used to estimate the runtimes for larger CRT-represented integers. Figure 4.2 illustrates that the sampling rate increases in a discrete manner as we raise the gadget base because the number of digits is determined by $\lceil 60/\log_2 b \rceil$. The runtime is dominated by the randomized operations (as the difference between a power-of-two-base and arbitrary-base scenarios is relatively small), thus limiting the advantages of choosing the faster power-of-two bases. This suggests that a CRT representation in terms of powers of primes, where the primes are used as the residue bases, might be preferred in some instances (where an efficient implemention of arithmetic over prime powers is available) over power-of-two bases.

Figure 4.3 illustrates the benefits of using the efficient gadget in CRT representation when working with cyclotomic rings. The conversion from CRT representation to the positional system followed by digit decomposition w.r.t a large modulus slows down subgaussian gadget decomposition rate by almost one order of magnitude. We also observe that the difference in

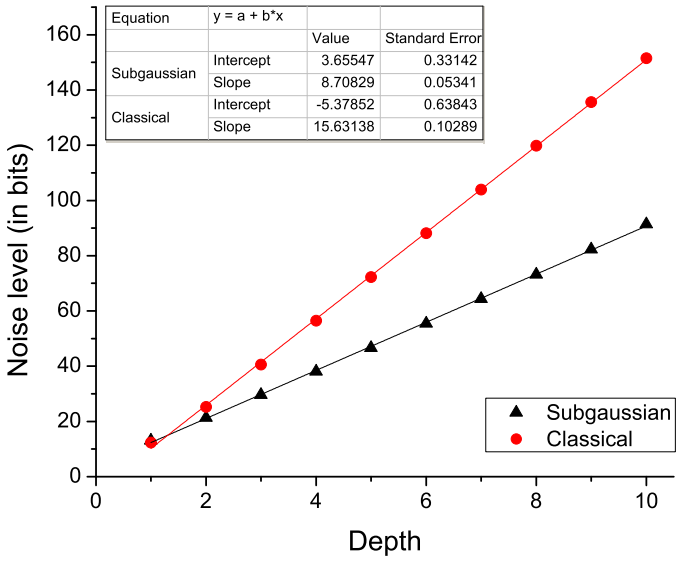

Figure 4.4. Noise growth for GSW-type multiplication in the ring-based KP-ABE variant (k = 180, n = 1024, b = 2). The base in the exponentiation is $(mn)^{\beta}$, where $m = k+2 = 182$ and β describes the rate of noise growth. The slope of the linear interpolation is $\beta \log_2(mn)$.

performance between a power-of-two base and an arbitrary base is relatively small for both cases.

Figure 4.4 demonstrates the differences in the noise growth of GSW-type products using the subgaussian and classical binary gadget decomposition methods. For this experiment, we generated an error vector in R^m and iteratively multiplied it by $\mathbf{G}^{-1}(\mathbf{U}_i)$, where $\mathbf{U}_i$ is a vector of uniformly random ring elements in R_q^m at level i. We applied the tree multiplication approach (rather than a sequential evaluation in a right-associative manner, which reduces the noise when dealing with a chained product of fresh encryptions in GSW [30, 10]) to emulate the noise growth in evaluating a Boolean policy circuit in the KP-ABE scheme. We considered both the cases when the same $\mathbf{U}$ was used at all levels (correlated ciphertexts) and different $\mathbf{U}_i$ at each level. The results were approximately the same for both scenarios because the classical gadget decomposition matrix is centered at 0.5 (see [39] for a more detailed discussion of the classical gadget decomposition case).

Figure 4.4 suggests that the noise growth in the subgaussian gadget decomposition case

Table 4.1. Comparison of performance results for our KP-ABE variant (in bold) vs. the implementation in [39] (in parentheses). EVALCT* = EVALPK + EVALCT corresponds to the scenario when the policy evaluation of public keys and ciphertexts is done at the same time.

ℓ	k	$\log_2 n$	r	KEYGEN [ms]	ENCRYPT [ms]	EVALCT* [s]	EVALPK [s]	DECRYPT [ms]	RAM [MB]
2	50 (44)	11 (11)	5 (1)	**40** (126)	**7** (33)	**0.023** (0.44)	**0.021** (0.42)	**1.8** (3.0)	**19** (58.5)
4	100 (52)	12 (12)	20 (1)	**64** (143)	**15** (57)	**0.072** (1.76)	**0.064** (1.68)	**3.9** (3.5)	**36.4** (86.3)
8	120 (60)	13 (13)	15 (1)	**151** (317)	**56** (222)	**0.59** (10.8)	**0.53** (10.4)	**8.9** (7.7)	**94.1** (255)
16	180 (70)	13 (13)	20 (1)	**177** (419)	**157** (1,483)	**1.68** (429)	**1.48** (427)	**11.5** (18.1)	**230** (2,867)
32	180	13	15	**206**	**414**	**5.67**	**5.0**	**13.46**	**508**
64	204	13	17	**226**	**1,052**	**13.1**	**11.2**	**16.39**	**1,229**
128	300	14	25	**568**	**6,454**	**98.3**	**85.5**	**45.43**	**7,024**

has a square-root dependence on mn ($\beta \approx 0.5$) while the classical gadget decomposition approach results in almost linear noise growth ($\beta \approx 0.9$). Note that the intercept is lower for classical gadget decomposition because the infinity norm of digits is 1 (only 0 or 1 are possible) vs. 2 in the case of subgaussian decomposition (the allowed integer values are in the range from -2 to 2). However, this advantage does not propagate to the second level of the circuit as the square-root dependence of subgaussian gadget decomposition already plays a more dominant role here.

4.9.2 Key-Policy Attribute-Based Encryption

Table 4.1 shows the performance results for our implementation along with the corresponding results for the implementation in [39]. The first three rows for the results in [39] were obtained using native (64-bit) integer arithmetic and the last row used a multiprecision backend in PALISADE based on NTL/GMP. The experiments were run for 4 threads on a commodity desktop system, i.e., Intel Core i7-3770 CPU with 4 cores at 3.40GHz and 16GB of memory running CentOS 7. Both variants were implemented in PALISADE v1.2.

To choose the ring dimension n for both implementations, we ran the LWE security estimator[6] (commit 560525) [6] to find the lowest security levels for the uSVP, decoding, and dual attacks following the standard homomorphic encryption security recommendations [5]. We selected the least value of the number of security bits λ for all 3 attacks on classical computers

[6]https://bitbucket.org/malb/lwe-estimator

based on the estimates for the BKZ sieve reduction cost model. All results are presented for at least 128 bits of security.

Table 4.1 suggests there is a speed-up of 2.1x to 3.2x for key generation, where the lattice trapdoor sampling subroutine is called. The speed-up for encryption is 3.8x to 9.5x, which is mostly attributed to the use of a larger gadget base. The speed-ups for the main bottleneck operations of homomorphic public key and ciphertext evaluation are in the range from 18x to 289x, which is a combined effect of subgaussian gadget decomposition in CRT and a larger gadget base. The decryption runtimes are comparable, and already fast for both implementations. The memory requirements for our optimized variant are 2.4x to 12.5x smaller.

Note that the performance of the KP-ABE variant in [39] dramatically degrades after switching from the native arithmetic (when $k \leq 60$ bits) to the multiprecision backend (for gadget decomposition), which is observed for $\ell = 16$ in Table 4.1. This implies the efficient gadgets in CRT representation are critical for supporting deeper Boolean circuits with CPU systems.

We also profiled the contributions of subgaussian gadget decomposition and the number theoretic transforms (NTT) of the digit-decomposed matrix (needed for matrix multiplication) to the runtimes for homomorphic policy evaluation of ciphertexts (EVALCT*). The contribution of subgaussian gadget decomposition was in the range from 15% to 22% w.r.t. the total homomorphic policy evaluation runtime. The contribution of the related NTTs was between 47% and 63%, suggesting that the latter is the main bottleneck of homomorphic circuit evaluation in our KP-ABE variant.

Acknowledgement

This chapter is reprinted as it appears (with minor modifications) from the publication "Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More," presented by this dissertation's author at EUROCRYPT 2019 [43] and is joint work with Daniele Micciancio, and Yuriy Polyakov.

Chapter 5

Subgaussian Analysis for Lattice Trapdoors

5.1 Introduction

Anyone tasked with measuring the concrete security of a cryptographic scheme built on the MP12 [66] lattice trapdoor, which relies on the short integer solution problem [4, 67], needs to understand the concentration of the largest singular value of a distribution of random matrices with subgaussian entries. Previously, the state of the art in concentration bounds for these matrices' singular values scaled with a mysterious constant value [79]. This is unsatisfying for the cryptographer interested in estimating the costs of a scheme prior to implementation.

The contribution of this chapter is twofold: first to find the exact constants in these singular values' concentration bounds, then to find the actual singular values appearing in-practice. Note, the provable bounds are for a broad class of random matrices and, as expected, the bounds are not as tight as one sees in-practice on the set of commonly-used distributions.

5.2 A Concentration Bound on Subgaussian Matrices with Exact Constants

In this section we prove the following theorem:

Theorem 5.2.1 *Let* $\mathbf{A}$ *be an* $m \times n$ *random matrix whose rows* $\mathbf{a}_i$ *are independent, zero-mean,*

σ-isotropic, subgaussian random vectors with parameter $s > 0$. Then, for any $t \geq 0$

$$\sigma\left[\sqrt{m} - C(s^2/\sigma^2)\left(\sqrt{n}+t\right)\right] \leq s_n(\mathbf{A}) \leq s_1(\mathbf{A}) \leq \sigma\left[\sqrt{m} + C(s^2/\sigma^2)\left(\sqrt{n}+t\right)\right]$$

for $C = \sqrt{4e^{1+2/e}\ln 9}/\sqrt{\pi}$ and with probability at least $1-2e^{-t^2}$.

Note, the normalized matrix $(1/\sigma)\mathbf{A}$ has isotropic rows. We remark the proof is nearly identical to [79] except here we pay special attention to the constants in the proof.

5.2.1 Useful Lemmas

We will need the following Fact.

Lemma 5.2.1 *Let X be a subgaussian random variable with parameter $s > 0$, then*

$$\mathbb{E}[|X|^k]^{1/k} \leq \frac{s\sqrt{k}}{e^{-1/e}\sqrt{2\pi}} = s \cdot O\left(\sqrt{k}\right)$$

for all $k > 0$.

Proof: Since $|X|$ is a non-negative random variable, we have

$$\mathbb{E}[|X|^k] = \int_0^\infty \Pr\{|X|^k > t\}dt \leq 2\int_0^\infty \exp(-\pi t^{2/k}/s^2)dt.$$

The inequality is by Fact 2.3.1. Next, we perform a change of variables. Let $u := \pi t^{2/k}/s^2$ so we have the following:

$$\mathbb{E}[|X|^k] \leq \left(\frac{s}{\sqrt{\pi}}\right)^k k\int_0^\infty \exp(-u)u^{k/2-1}du = \left(\frac{s}{\sqrt{\pi}}\right)^k k\cdot\Gamma(k/2)^{1/k}.$$

Now the bounds $\Gamma(k/2) \leq (k/2)^{k/2}$ (from Stirling's approximation) and $k^{1/k} \leq e^{1/e}$ (calculus) for all $k > 0$ give us

$$\mathbb{E}[|X|^k]^{1/k} \leq \frac{s}{\sqrt{2\pi}}e^{1/e}\sqrt{k}.$$

Lemma 5.2.2 *Let X be a positive random variable. Then, for all $i, j \in \mathbb{Z}^+$, we have*

$$cov(X^i, Y^j) = \mathbb{E}[X^{i+j}] - \mathbb{E}[X^i]\mathbb{E}[X^j] \geq 0.$$

Proof: Recall Jensen's inequality: $\varphi(\mathbb{E}[Y]) \leq \mathbb{E}[\varphi(Y)]$ for all random variables Y and all convex functions φ. This gives us

$$\mathbb{E}[X^i]^{(i+j)/i} \leq \mathbb{E}[X^{i+j}],\ \mathbb{E}[X^j]^{(i+j)/j} \leq \mathbb{E}[X^{i+j}].$$

Let us raise the first inequality to the $i/(i+j)$-th power and the second by $j/(i+j)$-th power. This gives us

$$\mathbb{E}[X^i] \leq \mathbb{E}[X^{i+j}]^{i/(i+j)}$$
$$\mathbb{E}[X^j] \leq \mathbb{E}[X^{i+j}]^{j/(i+j)}$$

and we multiply the two inequalities to get the result.

ε-Nets

An ε-net on the n-dimensional unit sphere, $S^{n-1} := \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 = 1\}$, is a finite set $N_\varepsilon \subset S^{n-1}$ such that for all $\mathbf{y} \in S^{n-1}$, there is a vector $\mathbf{x}$ in the net within a Euclidean distance $\varepsilon > 0$ of $\mathbf{y}$, $\|\mathbf{x} - \mathbf{y}\| \leq \varepsilon$. We have the following fact on the size of an ε-net on the unit sphere [79, Lemma 5.2].

Fact 5.2.2 *For all $\varepsilon > 0$, there exists an ε-net N_ε on the unit sphere S^{n-1} such that*

$$|N_\varepsilon| \leq \left(\frac{2}{\varepsilon} + 1\right)^n.$$

When $\mathbf{A} \in \mathbb{R}^{n\times n}$ is symmetric, we can estimate its norm from a net by the following lemma [79, Lemma 5.4].

Lemma 5.2.3 *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a symmetric matrix, $N_\varepsilon \subset S^{n-1}$ be an ε-net on the n-dimensional unit sphere for some $\varepsilon \in (0,1)$. Then,*

$$\|\mathbf{A}\| = \sup_{\mathbf{y}\in S^{n-1}} |\langle \mathbf{A}\mathbf{y}, \mathbf{y}\rangle| \leq \frac{1}{1-2\varepsilon} \max_{\mathbf{x}\in N_\varepsilon} |\langle \mathbf{A}\mathbf{x}, \mathbf{x}\rangle|.$$

Approximate isometries and isotropy

A random vector over $\mathbb{R}^n$ is $\sigma-$*isotropic* if $\mathbb{E}[\mathbf{x}\mathbf{x}^t] = \sigma^2\mathbf{I}_n$. When $\sigma = 1$, we simply say the random vector is isotropic. Let $\mathbf{x}$ be isotropic and $\mathbf{y} \in \mathbb{R}^n$ be arbitrary, then $\mathbb{E}[\langle \mathbf{x}, \mathbf{y}\rangle^2] = \mathbf{y}^t\mathbb{E}[\mathbf{x}\mathbf{x}^t]\mathbf{y} = \|\mathbf{y}\|_2^2$.

Below is a lemma giving a condition which implies a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is an approximate isometry from $\mathbb{R}^n$ to $\mathbb{R}^m$, measured by a small $\delta > 0$ [79, Lemma 5.36].

Lemma 5.2.4 *Let $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\delta > 0$, $\alpha > 1$, and $\|\mathbf{A}^t\mathbf{A} - \mathbf{I}_n\| \leq \alpha \max(\delta, \delta^2)$. Then,*

$$1 - \alpha\delta \leq s_n(\mathbf{A}) \leq s_1(\mathbf{A}) \leq 1 + \alpha\delta.$$

5.2.2 A Berstein-type Bound

Here we prove two lemmas: a bound on $(X^2 - \mathbb{E}[X^2])$'s moment generating function for a subgaussian X and a Berstein-type concentration lemma needed for the main theorem [76, Lemma 1.12].

Lemma 5.2.5 *Let X be a subgaussian random variable with parameter $s > 0$. Then, the random variable $Z := X^2 - \mathbb{E}[X^2]$ satisfies*

$$\mathbb{E}[e^{2\pi tZ}] \leq e^{C_B\cdot\pi t^2 s^4}$$

for all $|t| \leq 1/(8e^{2/e+1}s^2)$ and $C_B = 32e^{4/e+2}/\pi$.

Proof: We start with simply expanding the moment generating function and applying Jensen's inequality, $\varphi(\mathbb{E}[X]) \leq \mathbb{E}[\varphi(X)]$ for any convex function $\varphi(\cdot)$.

$$\begin{aligned}
\mathbb{E}[\exp(2\pi tZ)] &= 1 + \sum_{k\geq 2} \frac{(2\pi t)^k \mathbb{E}[(X^2 - \mathbb{E}[X^2])]^k}{k!} \\
&\leq 1 + \sum_{k\geq 2} \frac{(2\pi t)^k \mathbb{E}[(X^2 - \mathbb{E}[X^2])^k]}{k!} \\
&\leq 1 + \sum_{k\geq 2} \frac{2^{k-1}(2\pi t)^k \cdot 2 \cdot \mathbb{E}[X^{2k}]}{k!}.
\end{aligned}$$

The first inequality uses Jensen's inequality, for the function $\phi(y) = y^k$ with $y \geq 0$. The second inequality is given by pairing 2^{k-1} terms of the form $\mathbb{E}[X^{2i}X^{2j}] \pm E[X^{2i}]\mathbb{E}[X^{2j}]$, where $i + j = k$, and each term $\mathbb{E}[X^{2i}X^{2j}] \pm E[X^{2i}]\mathbb{E}[X^{2j}]$ is less than $2\mathbb{E}[X^{2k}]$ by Lemma 5.2.2. Now, we can use Lemma 5.2.1 to simplify.

$$\begin{aligned}
&= 1 + \sum_{k\geq 2} \frac{(4\pi t)^k \mathbb{E}[X^{2k}]}{k!} \\
&\leq 1 + \sum_{k\geq 2} \frac{(4\pi t)^k (se^{1/e}\sqrt{2k}/\sqrt{2\pi})^{2k}}{k!} \\
&= 1 + \sum_{k\geq 2} \frac{(4ts^2 e^{2/e} k)^k}{k!}.
\end{aligned}$$

Next, we use the bound $k! \geq (k/e)^k$ to get

$$\begin{aligned}
\mathbb{E}[\exp(2\pi tZ)] &\leq 1 + \sum_{k\geq 2} (4ts^2 e^{1+2/e})^k \\
&= 1 + (4ts^2 e^{1+2/e})^2 \sum_{k\geq 0} (4ts^2 e^{1+2/e})^k.
\end{aligned}$$

And finally, we restrict $t \in (0, \frac{1}{8s^2e^{1+2/e}})$ to get

$$\begin{aligned}\mathbb{E}[\exp(2\pi tZ)] &\leq 1 + 2(4ts^2e^{1+2/e})^2 \\ &\leq \exp(C_B \cdot \pi t^2 s^4)\end{aligned}$$

for $C_B := 32e^{2+4/e}/\pi$ (the last inequality uses $1 + x \leq e^x$ for $x \geq 0$).

Lemma 5.2.6 *Let $\{X_i\}_1^n$ be iid subgaussian random variables with parameter $s > 0$, and let*

$$\bar{X} := \frac{1}{n}\sum_i (X_i^2 - \mathbb{E}[X_i^2]),$$

then for all $t > 0$, we have

$$\Pr\{|\bar{X}| > t\} \leq 2 \cdot \exp\left(-\frac{n}{C_B} \cdot \min\left\{\frac{t^2}{s^4}, \frac{t}{s^2}\right\}\right)$$

for $C_B := \frac{32e^{2+4/e}}{\pi}$.

Proof: We proceed with the usual exponential Markov inequality. Fix $C_B = 32e^{4/e+2}/\pi$ as in the previous lemma. Let $\delta \in (0, 1/(8e^{2/e+1}s^2)]$ be arbitrary, $Z_i := X_i^2 - \mathbb{E}[X_i^2]$, and $S = n\bar{X} = \sum_i Z_i$.

$$\begin{aligned}\Pr[S > nt] &= \Pr[\exp(2\pi\delta S) > \exp(2\pi\delta nt)] \\ &\leq \exp(-2\pi\delta nt) \cdot \mathbb{E}[\exp(2\pi\delta S)] \\ &= \exp(-2\pi\delta nt) \cdot \prod_i \mathbb{E}[\exp(2\pi\delta Z_i)] \\ &\leq \exp(-2\pi\delta nt + nC_B\pi\delta^2 s^4).\end{aligned}$$

The first inequality is Markov's, and the last inequality is from Lemma 5.2.5. Now we minimize

the exponent as a function of δ. This yields $\delta = \frac{t}{C_B s^4}$ and in the case $\frac{t}{C_B s^4} \leq \frac{1}{8e^{1+2/e}s^2}$, we have

$$\Pr[\bar{X} > t] \leq \exp\left(-\frac{n}{C_B}\frac{t^2}{s^4}\right)$$

by substitution. If $\frac{t}{C_B s^4} > \frac{1}{8e^{1+2/e}s^2}$ (equivalently, $t > \frac{4e^{1+2/e}s^2}{\pi}$),

$$\Pr[\bar{X} > t] \leq \exp\left(-n\left[\frac{\pi t}{4e^{1+2/e}s^2} - \frac{1}{2}\right]\right)$$

by substitution with $\delta = \frac{1}{8e^{1+2/e}s^2}$. The further restriction on t ($t > \frac{4e^{1+2/e}s^2}{\pi}$ or equivalently $\frac{t\pi}{8e^{1+2/e}s^2} > 1/2$) gives us $\frac{\pi t}{4e^{1+2/e}s^2} - \frac{1}{2}$ is always at least $\frac{\pi t}{8e^{1+2/e}s^2}$, which in-turn is always at least $\frac{t}{C_B s^2}$. This proves the lemma.

5.2.3 Proof of Theorem 5.2.1

Proof: First, we will use a net on the unit sphere to approximate a fixed $\mathbf{A}$'s singular values, then we will use the Bernstein-like lemma from the previous subsection, Lemma 5.2.6, for a concentration bound on $\mathbf{A}$'s distribution. And finally, we will use a union bound over the entire net.

Let, $C_B = \frac{32e^{2+4/e}}{\pi}$ as in Lemma 5.2.6, $C = 2\sqrt{\ln(9)\cdot C_B} = 8e^{1+2/e}\sqrt{\ln 9}/\sqrt{\pi}$, $\delta := C(\sqrt{n/m} + t/\sqrt{m})$, and $\varepsilon := s^2/\sigma^2 \cdot \max(\delta, \delta^2)$. Notice that Lemma 5.2.1's proof gives us $\sigma < s$, or $s/\sigma > 1$.

Step 1: approximation.

Here we will use a net along with Lemma 5.2.4, which allows us to reduce the proof to showing $\|\mathbf{A}'^t\mathbf{A}' - I_n\| \leq (s/\sigma)^2 \max(\delta, \delta^2)$ where $\mathbf{A}' := \frac{1}{\sigma\cdot\sqrt{m}}\mathbf{A}$. Let $N \subset S^{n-1}$ be a $1/4$-net on the n-dimensional unit sphere. Lemma 5.2.3 tells us

$$\|\frac{1}{m\sigma^2}\mathbf{A}^t\mathbf{A} - \mathbf{I}_n\| \leq 2\max_{\mathbf{x}\in N}|\left\langle(\frac{1}{m\sigma^2}\mathbf{A}^t\mathbf{A} - \mathbf{I}_n)\mathbf{x}, \mathbf{x}\right\rangle| = 2\max_{\mathbf{x}\in N}\left|\frac{1}{m\sigma^2}\|\mathbf{A}\mathbf{x}\|_2^2 - 1\right|.$$

Next, we show with high probability (over $\mathbf{A}$'s rows), we have

$$\max_{\mathbf{x}\in N}\left|\frac{1}{m\sigma^2}\|\mathbf{A}\mathbf{x}\|_2^2-1\right|\leq\varepsilon/2.$$

Step 2: concentration.

Here we use the Bernstein-type lemma, Lemma 5.2.6, to get a concentration bound on $\left|\frac{1}{m\sigma^2}\|\mathbf{A}\mathbf{x}\|_2^2-1\right|$ for a fixed $\mathbf{x}$ in the net. Express $\|\mathbf{A}\mathbf{x}\|_2^2$ as a sum indexed by $\mathbf{A}$'s rows:

$$\|\mathbf{A}\mathbf{x}\|_2^2=\sum_{i=1}^{m}\langle\mathbf{a}_i,\mathbf{x}\rangle^2.$$

Now we are concerned with the probability $\Pr\left[\left|\frac{1}{m\sigma^2}\sum\langle\mathbf{a}_i,\mathbf{x}\rangle^2-1\right|\leq\varepsilon/2\right]$, and we can use Lemma 5.2.6 since $\mathbb{E}[\langle\mathbf{a}_i,\mathbf{x}\rangle^2]=\sigma^2$ by the definition of isotropic. For simplicity, re-scale the matrix to $\bar{\mathbf{A}}:=\mathbf{A}/\sigma$ with rows $\bar{\mathbf{a}}_i$. This matrix, $\bar{\mathbf{A}}$, has independent, centered, subgaussian rows with parameter s/σ. Then, we have

$$\begin{aligned}\Pr\left[\left|\frac{1}{m}\sum(\langle\bar{\mathbf{a}}_i,\mathbf{x}\rangle^2-1)\right|\leq\varepsilon/2\right]&\leq 2\exp\left(-\frac{m}{C_B}\min\left\{\frac{\sigma^2\varepsilon}{2s^2},\frac{\sigma^4\varepsilon^2}{4s^4}\right\}\right)\\&\leq 2\exp\left(-\frac{m}{4C_B}\min\left\{\max(\delta,\delta^2),\max(\delta^2,\delta^4)\right\}\right)\\&\leq 2\exp\left(-\frac{m}{4C_B}\delta^2\right)\end{aligned}$$

from Lemma 5.2.6[1]. (The min-max argument results in δ^2 in both cases $\delta<1$ and $\delta\geq 1$.)

Using the inequality $(a+b)^2\geq a^2+b^2$ for non-negative a,b gives us

$$\Pr\left[\left|\frac{1}{m}\sum\langle\bar{\mathbf{a}}_i,\mathbf{x}\rangle^2-1\right|\leq\varepsilon/2\right]\leq 2\exp(-\ln(9)n-t^2)$$

since $C_0=2\sqrt{\ln 9\cdot C_B}$.

[1]We consider $\delta\geq 1$ and $\delta<1$ by using Lemma 5.2.6 in this step of the proof.

$\mathscr{X}$	$\bar{s_1}$	$\sigma(\sqrt{m}+C_{\mathscr{X}}(s/\sigma)^2\sqrt{n})$	observed $C_{\mathscr{X}}$	Sample Var
$\mathscr{P}$	71.26	71.43	$.99/4\pi$	.04
$\mathscr{U}\{-1,1\}$	100.74	101.01	$.99/2\pi$	.05
$\mathscr{N}(0,1)$	100.71	101.01	.99	.043
$\mathscr{D}_{\mathbb{Z},\sqrt{2\pi}}$	100.77	101.01	.99	.06
$\mathscr{X}$	$\bar{s_n}$	$\sigma(\sqrt{m}-C_{\mathscr{X}}(s/\sigma)^2\sqrt{n})$	observed $C_{\mathscr{X}}$	Sample Var
$\mathscr{P}$	39.60	39.43	$.99/4\pi$	.017
$\mathscr{U}\{-1,1\}$	56.00	55.76	$.99/2\pi$	.043
$\mathscr{N}(0,1)$	55.92	55.76	.99	.036
$\mathscr{D}_{\mathbb{Z},\sqrt{2\pi}}$	56.00	55.76	.99	.037

Figure 5.1. Data from 50 random matrices of dimension $m = 6144 \times n = 512$ for each distribution $\mathscr{X}$. The third column is the expected singular value using each distribution's calculated $C_{\mathscr{X}}$: 1, $1/2\pi$, and $1/4\pi$ for discrete/continuous gaussians, $\mathscr{U}\{-1,1\}$, and $\mathscr{P}$ respectively.

Step 3: union bound.

Finally, we take a union bound over all $\mathbf{x} \in N$ and from fact 5.2.2 we have that $|N| \leq 9^n$. This gives us

$$\begin{aligned}\Pr[\exists\, \mathbf{x} \in N : \frac{1}{m}\sum \langle \bar{\mathbf{a}}_i, \mathbf{x}\rangle^2 - 1 > \varepsilon/2] &\leq 9^n \cdot 2\exp(-\ln(9)n - t^2) \\ &= 2\exp(-t^2).\end{aligned}$$

The proof is complete by Lemma 5.2.4.

5.2.4 Experiments

Here we present empirical data on the singular values of random matrices with independent entries drawn from commonly-used distributions in lattice-based cryptography. These distributions are the continuous gaussian, the discrete gaussian over $\mathbb{Z}$, $\mathscr{U}\{-1,1\}$, and the distribution given by choosing 0 with probability $1/2$ and ± 1 each with probability $1/4$, which we denote as $\mathscr{P}$. For each distribution, we sampled 50 $m = 6144$ by $n = 512$ random matrices

and measured their singular values, and assumed the singular values were approximately

$$s_1 \approx \sigma\left(\sqrt{m} + C_{\mathscr{X}}(s/\sigma)^2\sqrt{n}\right)$$
$$s_n \approx \sigma\left(\sqrt{m} - C_{\mathscr{X}}(s/\sigma)^2\sqrt{n}\right)$$

where $C_{\mathscr{X}}$ is a small constant dependent on the distribution $\mathscr{X}$. These results are given in Figure 5.1.

Continuous and Discrete Gaussians

The continuous gaussian $\mathscr{D}_\sigma$ is subgaussian with parameter $\sigma/\sqrt{2\pi}$ since[2] $\mathbb{E}[e^{2\pi t\mathscr{X}}] = e^{\pi t^2\sigma^2}$ where $\mathscr{X} \sim D_\sigma$. Further, the discrete gaussian $D_{\Lambda,s}$ is subgaussian with parameter s, independent of the smoothing parameter of Λ, for any lattice Λ [66, Lemma 2.8]. Assuming that the discrete gaussian is smooth, then one can expect the standard deviation of $D_{\mathbb{Z},r}$ to be close to the standard deviation of the continuous gaussian it approximates, $r/\sqrt{2\pi}$. This implies the ratio between the subgaussian parameter and the standard deviation of (discrete) gaussians is one or nearly one. Under this assumption, we observed $C_{\text{gaussian}} = 1$.

Uniform over $\{-1,1\}$

Here $\sigma = 1$ and $\mathbb{E}[e^{2\pi tX}] = \cosh 2\pi t \leq e^{2\pi^2 t^2}$, or the subgaussian parameter is at most 2π. We observed $C_{\mathscr{U}\{-1,1\}} = 1/2\pi$ in our experiment.

The Distribution $\mathscr{P}$

By nearly the same steps as the previous distribution, $\mathscr{P}$ is subgaussian with parameter 2π and $\sigma = 1/\sqrt{2}$. Then, we observed a $C_{\mathscr{P}} = 1/4\pi$. We note that for all four distributions we observed $C_{\mathscr{X}}(s/\sigma)^2 \approx 1$.

As a second experiment, we repeated the first experiment for a fixed $\mathscr{X} = \mathscr{U}\{-1,1\}$ but with varying dimensions. This experiment's data for s_1 is shown in Figure 5.2, graphed

[2] $D_\sigma = N(0,\sigma/\sqrt{2\pi})$ where $N(0,\sigma)$ is the normal distribution with variance σ^2.

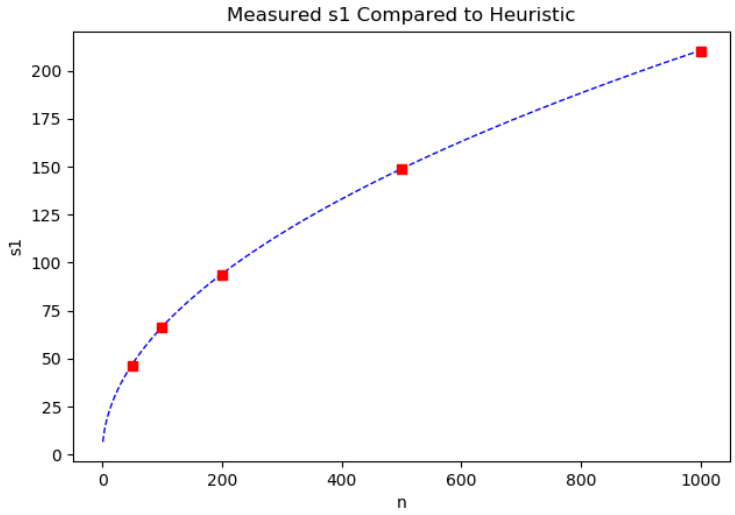

Figure 5.2. Here $\mathscr{X} = \{-1, 1\}$. For each $n = 50, 100, 200, 500, 1000$, the experiment sampled $N = 50$ random n by $32n$ matrices and averaged their largest singular value. The measured constant $C_{\mathscr{X}}$ approached $1/2\pi$ from below as n increased $(.92/2\pi, .96/2\pi, .97/2\pi, .99/2\pi, .99/2\pi)$.

with the expected largest singular value. We remark that we saw the same behavior for all four distributions when we varied the dimension.

5.2.5 Applications

Here we show how the updated singular value estimates from the previous subsection impact concrete security of lattice trapdoor schemes. As an example, we use the [66] trapdoor scheme with entries drawn independently from $\mathscr{P}$. Since the singular values scale with $\sigma = 1/\sqrt{2}$, the concrete security of the underlying SIS problem increases. See Figure 5.3 for the difference in a commonly-used parameter regime.

In order to estimate security, we followed [8, 7], in using sieving as the SVP oracle with time complexity $2^{.292k+16.4}$ in the block size, k. BKZ is expected to return a vector of length $\delta^{2n}\det^{1/2n}$ for a lattice of dimension $2n$ (Minkowski's theorem tells us a short enough lattice vector exists when we only use $2n$ columns of $\mathbf{A}$). Hence, we found the smallest block size k achieving the needed δ corresponding to forging a signature, $\frac{s\sqrt{m}}{\sqrt{q}} = \delta^{2n}$. Finally, we used

Parameters	Original	Updated
n	512	512
q	2^{24}	2^{24}
s	2881	2037
m	24804	24804
Bit Sec.	124	136
δ	1.0046	1.0043
k	324	364

Figure 5.3. The change in concrete security of the underlying SIS problem in MP12 when the trapdoor is drawn from $\mathscr{P}^{m\times n}$. We give the smallest BKZ block size k achieving the δ needed to find a vector of length $s\sqrt{m}$ in (a subspace of) the lattice $\Lambda_q^{\perp}(\mathbf{A})$.

the heuristic $\delta \approx (\frac{k}{2\pi e}(\pi k)^{1/k})^{1/2(k-1)}$ to determine the relationship between k and δ, and we set the total time complexity of BKZ with block-size k, dimension $2n$ as $8\cdot(2n)\cdot \text{time}(\text{SVP}) = 8\cdot(2n)\cdot 2^{.292k+16.4}$ [35, 8].

Bibliography

[1] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In *CRYPTO 2017*, pages 3–35, 2017.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572. Springer, 2010.

[3] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy IBE) from lattices. In Marc Fischlin, Johannes A. Buchmann, and Mark Manulis, editors, *Public Key Cryptography - PKC 2012 - 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings*, volume 7293 of *Lecture Notes in Computer Science*, pages 280–297. Springer, 2012.

[4] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996.

[5] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Jeffrey Hoffstein, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Cambridge MA, March 2018.

[6] Martin Albrecht, Samuel Scott, and Rachel Player. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169203, 10 2015.

[7] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. Estimate all the {LWE, NTRU} schemes! In *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, pages 351–367, 2018.

[8] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Mathematical Cryptology*, 9(3):169–203, 2015.

[9] Jacob Alperin-Sheriff and Daniel Apon. Weak is better: Tightly secure short signatures from weak prfs. *IACR Cryptology ePrint Archive*, 2017:563, 2017.

[10] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014.

[11] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, pages 75–86, 2009.

[12] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. *Theory Comput. Syst.*, 48(3):535–553, 2011.

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

[14] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. In *ASIACRYPT 2015*, pages 3–24, 2015.

[15] Jean-Claude Bajard, Julien Eynard, M. Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. In *Selected Areas in Cryptography - SAC'16*, volume 10532 of *LNCS*, pages 423–442, 2016.

[16] Rachid El Bansarkhani and Johannes A. Buchmann. Improvement and efficient implementation of a lattice-based signature scheme. In *Selected Areas in Cryptography - SAC 2013*, pages 48–67.

[17] Rikke Bendlin, Sara Krehbiel, and Chris Peikert. How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In *Applied Cryptography and Network Security, ACNS 2013*, pages 218–236.

[18] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 149–168. Springer, 2011.

[19] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 533–556. Springer, 2014.

[20] Dan Boneh, Sam Kim, and Valeria Nikolaenko. Lattice-based DAPS and generalizations: Self-enforcement in signature schemes. In *Applied Cryptography and Network Securitym ACNS 2017*, pages 457–477, 2017.

[21] Dan Boneh, Kevin Lewi, and David J. Wu. Constraining pseudorandom functions privately. In *Public Key Cryptography (2)*, volume 10175 of *Lecture Notes in Computer Science*, pages 494–524. Springer, 2017.

[22] Guillaume Bonnoron, Léo Ducas, and Max Fillinger. Large FHE gates from tensored homomorphic accumulator. In *AFRICACRYPT'18*, volume 10831 of *LNCS*, pages 217–251, 2018.

[23] Xavier Boyen and Qinyi Li. Attribute-based encryption for finite automata from LWE. In Man Ho Au and Atsuko Miyaji, editors, *Provable Security, ProvSec 2015*, volume 9451 of *Lecture Notes in Computer Science*, pages 247–267. Springer, 2015.

[24] Xavier Boyen and Qinyi Li. Towards tightly secure lattice short signature and id-based encryption. In *ASIACRYPT 2016*, pages 404–434, 2016.

[25] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

[26] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325. ACM, 2012.

[27] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 575–584, 2013.

[28] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 97–106, 2011.

[29] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

[30] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In *Innovations in Theoretical Computer Science - ITCS'14*, pages 1–12, 2014.

[31] Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from LWE: unbounded attributes and semi-adaptive security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016*, volume 9816 of *Lecture Notes in Computer Science*, pages 363–384. Springer, 2016.

[32] Zvika Brakerski, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Obfuscating conjunctions under entropic ring LWE. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 147–156. ACM, 2016.

[33] Ran Canetti and Yilei Chen. Constraint-hiding constrained prfs for nc^1 from LWE. In *EUROCRYPT 2017*, pages 446–476, 2017.

[34] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 577–607. Springer, 2018.

[35] Yuanmi Chen. *Réduction de réseau et sécurité concréte du chiffrement complétement homomorphe*. PhD thesis, Paris 7, 2013.

[36] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[37] David Bruce Cousins, Giovanni Di Crescenzo, Kamil Doruk Gür, Kevin King, Yuriy Polyakov, Kurt Rohloff, Gerard W. Ryan, and Erkay Savas. Implementing conjunction obfuscation under entropic ring LWE. In *Symposium on Security and Privacy - SSP'18*, pages 354–371, 2018.

[38] Eric Crockett and Chris Peikert. $\Lambda o \lambda$: Functional lattice cryptography. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 993–1005. ACM, 2016.

[39] Wei Dai, Yarkin Doröz, Yuriy Polyakov, Kurt Rohloff, Hadi Sajjadpour, Erkay Savas, and Berk Sunar. Implementation and evaluation of a lattice-based key-policy ABE scheme. *IEEE Trans. Information Forensics and Security*, 13(5):1169–1184, 2018.

[40] Léo Ducas and Phong Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2012.

[41] Léo Ducas and Thomas Prest. Fast fourier orthogonalization. In Sergei A. Abramov, Eugene V. Zima, and Xiao-Shan Gao, editors, *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation, ISSAC 2016*, pages 191–198. ACM, 2016.

[42] Nicholas Genise and Daniele Micciancio. Faster gaussian sampling for trapdoor lattices with arbitrary modulus. In *EUROCRYPT'18*, volume 10820 of *LNCS*, pages 174–203, 2018.

[43] Nicholas Genise, Daniele Micciancio, and Yuriy Polyakov. Building an efficient lattice gadget toolkit: Subgaussian sampling and more. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, pages 655–684, 2019.

[44] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[45] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 850–867, 2012.

[46] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 197–206. ACM, 2008.

[47] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[48] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. *J. ACM*, 62(6):45:1–45:33, 2015. Prelim. version in STOC 2013.

[49] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523. Springer, 2015.

[50] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings ACM on Symposium on Theory of Computing, STOC 2015*, pages 469–477. ACM, 2015.

[51] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A group signature scheme from lattice assumptions. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 395–412. Springer, 2010.

[52] Kamil Doruk Gur, Yuriy Polyakov, Kurt Rohloff, Gerard W. Ryan, and Erkay Savas. Implementation and evaluation of improved gaussian sampling for lattice trapdoors. *IACR Cryptology ePrint Archive*, 2017:285, 2017.

[53] Shai Halevi, Tzipora Halevi, Victor Shoup, and Noah Stephens-Davidowitz. Implementing bp-obfuscation using graph-induced encoding. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 783–798, 2017.

[54] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved RNS variant of the BFV homomorphic encryption scheme. *IACR Cryptology ePrint Archive*, 2018:117, 2018.

[55] Shai Halevi, Yuriy Polyakov, and Victor Shoup. An improved RNS variant of the BFV homomorphic encryption scheme. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA,*

March 4-8, 2019, Proceedings, volume 11405 of *Lecture Notes in Computer Science*, pages 83–105. Springer, 2019.

[56] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

[57] James Howe, Thomas Pöppelmann, Máire O'Neill, Elizabeth O'Sullivan, and Tim Güneysu. Practical lattice-based digital signature schemes. *ACM Trans. Embedded Comput. Syst.*

[58] Antoine Joux and Jacques Stern. Lattice reduction: A toolbox for the cryptanalyst. *J. Cryptology*, 11(3):161–185, 1998.

[59] Charles F. F. Karney. Sampling exactly from the normal distribution. *ACM Trans. Math. Softw.*, 42(1):3:1–3:14, January 2016.

[60] Sam Kim and David J. Wu. Watermarking cryptographic functionalities from standard lattice assumptions. In *CRYPTO 2017*, pages 503–536, 2017.

[61] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In David B. Shmoys, editor, *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, pages 937–941. ACM/SIAM, 2000.

[62] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013*, volume 8270 of *Lecture Notes in Computer Science*, pages 41–61. Springer, 2013.

[63] Adeline Langlois, San Ling, Khoa Nguyen, and Huaxiong Wang. Lattice-based group signature scheme with verifier-local revocation. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014*, volume 8383 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2014.

[64] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43, 2013.

[65] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-lwe cryptography. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 35–54. Springer, 2013.

[66] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.

[67] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007.

[68] Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. In *CRYPTO 2017*, pages 455–485, 2017.

[69] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.

[70] Phong Q. Nguyen, Jiang Zhang, and Zhenfeng Zhang. Simpler efficient group signatures from lattices. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 401–426. Springer, 2015.

[71] Chris Peikert. personal communication.

[72] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 80–97. Springer, 2010.

[73] Yuri Polyakov, Kurt Rohloff, and Gerard W. Ryan. PALISADE lattice cryptography library. https://git.njit.edu/palisade/PALISADE, Accessed October 2018.

[74] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In *Proceedings of the 16th International Workshop on Cryptographic Hardware and Embedded Systems — CHES 2014 - Volume 8731*, pages 353–370, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[75] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Symposium on Theory of Computing - STOC'05*, pages 84–93, 2005.

[76] Philippe Rigollet. 18.s997 high-dimensional statistics. License: Creative Commons BY-NC-SA, Spring 2015. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu.

[77] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[78] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, pages 124–134. IEEE Computer Society, 1994.

[79] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *CoRR*, abs/1011.3027, 2010.

[80] Fuzhen Zhang. *The Schur Complement and Its Applications*, volume 4. Springer Science, 2006.