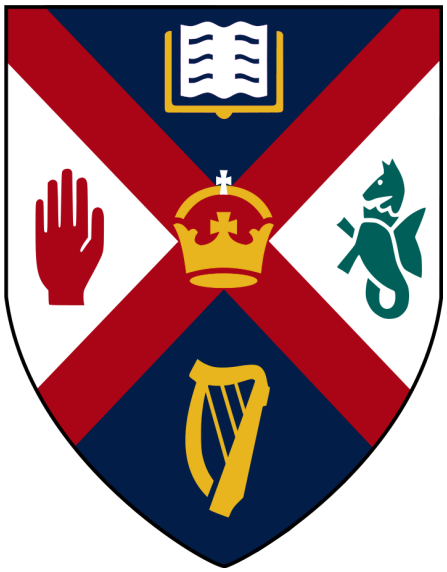
Practical Lattice-Based Cryptography in Hardware



James Victor Howe MSc BSc

Supervisor: Professor Máire O'Neill

School of Electronics, Electrical Engineering, and Computer Science,

Queen's University Belfast.

This dissertation is submitted for the degree of

Doctor of Philosophy

May 2018

Nomenclature

N Defines the security parameter of a given cryptoscheme.

λ Defines the arithmetic precision of a given cryptoscheme.

μ Defines the message data used with a digital signature scheme.

σ The standard deviation of the discrete Gaussian distribution.

τ The tail-cut parameter of the discrete Gaussian distribution.

AES Advanced encryption standard.

ASIC Application-specific integrated circuit.

AVX Advanced vector extensions.

BDD Bounded distance decoding.

BG The lattice-based digital signature scheme of Bai and Galbraith [2014b].

BKZ The Block Korkine-Zolotarev lattice reduction algorithm.

BLISS The bimodal lattice-based digital signature scheme of Ducas et al. [2013].

CCA Chosen-ciphertext attack.

CPA Chosen-plaintext attack.

CPU Central processing unit.

CVP The closest vector problem.

DCK The decisional compact knapsack problem.

DDoS Distributed denial of service.

DNS Domain name system.

DSA Digital signature algorithm.

DSS Digital signature scheme.

ECC Elliptic-curve cryptography.

ECDSA Elliptic-curve digital signature algorithm.

EU-CMA Existentially unforgeable under a chosen message attack.

FDH Full-domain hash.

FF Flip-flop.

FFT Fast Fourier transform.

FPGA Field-programmable gate array.

GapCVP The approximate decision version of the closest vector problem (CVP).

GapSVP The approximate decision version of the shortest vector problem (SVP).

GGH The Goldreich-Goldwasser-Halevi [Goldreich et al., 1996] lattice-based cryptosystem.

GLP The Güneysu-Lyubashevsky-Pöppelmann [Güneysu et al., 2012] lattice-based digital signature scheme.

GPU Graphics processing unit.

GPV The Gentry-Peikert-Vaikuntanathan [Gentry et al., 2008] cryptosystem.

IoT Internet of things.

LBC Lattice-based cryptography.

LUT Lookup table.

LWE The learning with errors problem.

LYU The lattice-based digital signature scheme of Lyubashevsky [2012].

NP Non-deterministic polynomial-time.

NTRU The N^{th} degree truncated polynomial ring lattice-based cryptosystem of Hoffstein et al. [1998].

NTT Number theoretic transform.

PAR Post-place and route.

PE Processing elements.

PKC Public-key cryptography.

PKE Public-key encryption.

PQC Post-quantum cryptography.

PRNG Pseudo-random number generator.

PSF Preimage sampleable (trapdoor) function.

Ring-LWE The ring variant of the learning with errors (LWE) problem.

Ring-SIS The ring variant of the short integer solution (SIS) problem.

Ring-TESLA The lattice-based digital signature scheme of Akleylek et al. [2016].

RSA The Rivest-Shamir-Adleman [Rivest et al., 1978] cryptosystem.

SIMD Single instruction, multiple data.

SIS The short integer solution problem.

SIVP The shortest independent vector problem.

SVP The shortest vector problem.

TESLA The lattice-based digital signature scheme of Alkim et al. [2015].

V2X Vehicle-to-everything.

I would like to dedicate this thesis to my loving parents ...

Declaration

I hereby declare that, except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 80,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

James Victor Howe MSc BSc

May 2018

Acknowledgements

And I would like to acknowledge ...

Abstract

Secure communications channels have become essential for the transmission of sensitive information over the Internet or between embedded devices, requiring protocols such as public-key encryption and digital signatures. Furthermore, these requirements for data security and privacy are becoming more important with growing numbers of connected devices, due to the popularity of the Internet of Things.

So far, practitioners have relied on cryptography based on the hardness of the factoring assumption (RSA) or the discrete logarithm problem (DSA/ECDSA). However, should a quantum computer be realised, the hardness of these related problems will be seriously weakened. This issue not only affects future communications but also secure messages sent today, which could be intercepted and stored, then decrypted by a device built a decade from now. Preparing for this is therefore paramount, and hence quantum-safe alternatives are needed to provide long-term security.

This thesis is focused on novel hardware designs of such quantum-safe alternatives, proposing architectures for public-key cryptographic schemes whose security is based on hardness problems which are to-date no easier to solve with a quantum computer. More specifically, the hardware designs proposed are for lattice-based cryptography. From a theoretical point-of-view, lattice-based cryptography has been well-studied, but research into improving its performance in hardware is still on-going. The proposed novel hardware designs are for discrete Gaussian

sampling techniques, a lattice-based public-key encryption scheme, and digital signature scheme.

Discrete Gaussian samplers are ubiquitously used within lattice-based cryptography. Novel hardware designs are proposed for all the major discrete Gaussian sampling methods, which are followed by a comprehensive evaluation. The novel hardware architectures proposed are for the Bernoulli, cumulative distribution table (CDT), and Knuth-Yao sampling techniques, which all operate in independent time. Thus, as well as improving on previous research, the novel designs protect the samplers against timing side-channel analysis. A discrete Gaussian distribution testing suite is also proposed to verify the randomness of these samplers.

The proposed hardware design of the lattice-based public-key encryption scheme is based on a stronger security assumption, the standard lattice assumption, in comparison to previous research on ideal lattices. The novel architecture utilises algorithmic and architectural optimisations, designed for the targeted Spartan-6 FPGA platform. When integrated with a constant-time discrete Gaussian sampler, the entire encryption scheme runs in independent time. Previously, standard lattices were thought to be completely impractical in hardware, as such the more practical ideal lattices were viewed as preferable, due to its practicality and smaller key sizes. The results here show that standard lattices are practical, and could be considered, especially for applications that require strong security.

The proposed digital signature hardware design is also based on a scheme proven to be more secure than previous research. Multiple hardware designs are investigated for different multipliers, which utilise a number of parallel Comba polynomial multipliers. This was examined since the multiplication stage was found to be the architecture's bottleneck. Results are provided for design goals of low-area, with higher throughput results provided with the inclusion of additional parallel multipliers.

Overall, the results show that lattice-based cryptography is not only a viable candidate for quantum-safe cryptography, but also performs well in its own right when compared to current state-of-the-art classical cryptoschemes.

Table of contents

List of figures

List of tables

CHAPTER 1

Introduction

In this chapter, the fundamental motivation for the research in this thesis is presented, that is, the emerging threat public-key cryptography (PKC) is facing with the almost inevitable construction of a quantum computer. Currently used PKC, such as RSA and elliptic-curve cryptography (ECC) which are deployed on a large number of diverse devices, are particularly susceptible. The proposed alternative considered in this thesis is lattice-based cryptography, a branch of cryptography that is resilient to quantum attacks via a quantum computer, which offers efficient asymmetric encryption and signature schemes. Additionally, as shown in this thesis, it is a competitive alternative to current PKC in its own right.

1.1 Motivation

Modern computing has drastically evolved in recent years due to the increase and widespread use of the Internet, e-commerce, and cloud services. These systems are continually evolving due to requirements from specialised platforms, such as those required by the Internet of things (IoT) or for vehicle-to-everything (V2X) communications, but also add additional attack vectors for adversaries. This was

shown recently by a cyber-attack (more specifically, a distributed denial of service (DDoS) attack) using compromised IoT devices, which severely disrupted Internet services by attacking major domain name system (DNS) infrastructures [Woolf, October 2016]. Security is therefore a growing concern, both for hardware and software systems.

Additionally, these systems also require *long-term* security and potentially need to be protected against attacks for the foreseeable future. Therefore, designing practical cryptographic schemes for these evolving and specialist platforms is paramount. With the onset of quantum computers ever looming, the computational power it could provide would cause instant insecurity to many of today's universally used cryptographic schemes by virtue of Shor's algorithm [Shor, 1997].

Specifically, schemes based on the discrete-logarithm problem or number-theoretic hard problems, which subsume almost all public-key encryption schemes used on the Internet, including elliptic-curve cryptography (ECC), RSA and DSA would be vulnerable. This is due to Shor's algorithm, which has the ability to efficiently compute discrete logarithms and factoring. Even symmetric-key encryption schemes, such as the Advanced Encryption Standard (AES), would have a quadratic brute-force attack speed-up via Grover's algorithm [Grover, 1996, 1997], essentially making AES-128 as secure as AES-64.

Accordingly, this has motivated the era of post-quantum cryptography (PQC), also known as quantum-safe cryptography or quantum-resilient cryptography, which refers to the construction of cryptographic algorithms to withstand quantum reductions. PQC is generally split into five types:

- *Code-based cryptography* [Overbeck and Sendrier, 2009] is based on the hard problem of decoding a random linear code. That is, the secret code-generating matrix is hidden with permutating, scrambling, and noise operations, in which recovery is only possible with the original code-generating

matrix. An example of a code-based cryptosystem is the McEliece [McEliece, 1978] cryptosystem.

- *Hash-based cryptography* is based on the hardness of finding pre-images of cryptographic hash functions, the classical example being the Merkle signature scheme [Merkle, 1990]. This type of cryptography currently only offers digital signature schemes, a more modern example of which is by Bernstein et al. [2015] and the XMSS signature scheme by Buchmann et al. [2011].

- *Multivariate-quadratic cryptography* [Matsumoto and Imai, 1988] is based on the hard problem of finding the solution for a set of multivariate-quadratic systems over a finite field. This type of cryptography also currently only offers digital signature schemes. One of the first signature schemes proposed of this type was the "Unbalanced Oil and Vinegar" (UOV) signature scheme by Kipnis et al. [1999]. UOV was later improved upon with the signature scheme "Rainbow" by Ding and Schmidt [2005].

- *Supersingular Isogeny Cryptography* [Jao and Feo, 2011] is a more recent post-quantum candidate, offering key exchange, encryption, and digital signatures. Its hardness is mainly based on the computation of endomorphism rings of supersingular curves, which is equivalent to computing isogenies between supersingular elliptic curves. Its post-quantum security has been recently studied by De Feo et al. [2014] and Galbraith et al. [2016]. Recently, a supersingular isogenies Diffie-Hellman key exchange hardware design was proposed by Koziel et al. [2017].

- *Lattice-based cryptography* [Ajtai, 1996, Regev, 2005] is based on the hardness of solving certain lattice problems, thought to be hard for both classical and quantum computers [Lyubashevsky and Micciancio, 2009]. These lattice

problems are essentially finding the shortest or closet vectors in a lattice, a problem synonymous to the minimum distance problem in coding theory. The current state-of-the-art is the LPR public-key encryption scheme by Lyubashevsky et al. [2013a] and the BLISS digital signature scheme by Ducas et al. [2013].

Amongst these important areas of post-quantum research, lattice-based cryptography is disputably the most auspicious. The main advantage of lattice-based cryptography over other post-quantum cryptosystems is that it allows for extended functionality and is, at the same time, efficient for the basic primitives of public-key encryption and digital signature schemes. Computational problems that exist within the lattice environment, such as finding the shortest vector (SVP) or finding the closest vector (CVP), are thought to be resilient to quantum-computer attacks [Ajtai et al., 2001, Dinur et al., 2003] which imply its conjectured intractability. Such properties show promise, with regards to security, for replacing current asymmetric schemes that would be susceptible to attacks in a post-quantum world.

On the practical front, some constructions of public-key encryption schemes and digital signature schemes based on lattice problems are now more practical than traditional schemes based on RSA. One of the most recent hardware designs of a lattice-based encryption scheme in hardware is shown by Roy et al. [2013a] (which improves on Göttert et al. [2012], Pöppelmann and Güneysu [2013]) with results outperforming those of ECC (over curve P224) by 20x [Güneysu and Paar, 2008]. Additionally, the hardware design is also an order of magnitude faster than a comparable RSA hardware design, provides a higher security level, and consumes less device resources. With regards to digital signature schemes, the two most notable hardware architectures by Güneysu et al. [2012] and Pöppelmann

et al. [2014] also show a speed improvement of 1.7x and 14x compared to an equivalent RSA design [Suzuki and Matsumoto, 2011].

The first use of lattices as a cryptographic primitive is due to Ajtai [1996], who proposed a problem now known as the short integer solution (SIS) problem. The concept remained purely academic, due to its limited capabilities and inefficiencies, until recently; lattice-based cryptography has now become available as a future alternative to number-theoretic cryptography. Recent research allows virtually any cryptographic primitive, such as those already discussed, to be built on the hardness of lattice problems.

Also, there has been a transition into a particular class of lattices, predominantly ideal lattices, as a source of computational hardness. Although the robustness of hardness assumptions on ideal lattices, in comparison to general lattices, has not been explicitly proven, it is generally considered that most problems relevant for cryptography still remain hard [Langlois and Stehlé, 2014, Lyubashevsky et al., 2010]. Additionally, using ideal lattices offers a significant speed-up and reduction in key sizes for almost all cryptographic primitives, in particular, in encryption schemes and digital signatures.

However, it will be some time before lattice-based cryptoschemes begin to replace current public-key cryptography and their integration into practical applications needs to be explored. For example, ECC was proposed by Miller [1986] and Koblitz [1987] but it took 20 years until it appeared in actual security systems. And while cryptanalysis is still an ongoing effort, the most critical issue to date with lattice-based cryptography is its practicability, and it is clear that in order for it to replace widely used number-theoretic primitives, its constructions must be shown to be similarly efficient on many of the embedded platforms prevalent in today's digital and pervasive environment.

This motivates the theme of this thesis; evaluating lattice-based cryptography as an alternative to currently used quantum susceptible cryptography such as

RSA and ECC. In particular, a random number generator required with lattice-based cryptography, a lattice-based encryption scheme, and a lattice-based digital signature scheme have been focused on. The focus of this research has been to design hardware optimised architectures for these lattice-based algorithms.

Recently, there has been a major shift towards field-programmable gate array (FPGA) use in cloud services, since FPGAs are able to contribute significant amounts of computing power at a lower cost in comparison to central processing units (CPUs). This can be seen in Amazon [Freund, December 2016] and Microsoft's [Metz, December 2016] inclusion of FPGAs in their data centres, where "the hardware costs less than 30 percent of everything else in the [Microsoft's] server, consumes less than 10 percent of the power, and processes data twice as fast as the company could without it." These savings in hardware and energy costs provided by FPGAs are also echoed by Intel's $16.7 billion acquisition of Altera [Darrow, February 2016].

Additionally, FPGAs are also being used for encryption and compression [Metz, December 2016]. This is furthered by an in-depth article in Forbes Insights by Intel [Intel, January 2017], which states that:

> "Security is another essential element for autonomous vehicles and, in many cases, for much of IoT. In the near future, autonomous cars may be connected to the Internet, sharing relevant data over the cloud, and, possibly, from car to car. The vehicle must be able to apply security algorithms to ensure that the data received is coming from a trusted source. The digital algorithms used for sending data, and for validating and decrypting incoming data, will be complex, and constantly changing. Like dealing with unexpected situations, handling the security screening will require substantial computing capabilities and FPGAs are ideally suited to performing these tasks. They operate

quickly and can be reconfigured when necessary to upgrade security settings."

Indeed, the FPGA platform is very suitable for cryptography and the prototyping of cryptographic designs. This is due to the device's flexible re-programmability, and features such as digital signal processor (DSP) blocks. The FPGA devices considered in this thesis are generally low-cost, with hardware designs usually targeting the Xilinx Spartan-6 family of devices [Xilinx, 2011]. However, the proposed designs are not restricted to this family of FPGAs and can be adapted to other platforms.

FPGAs are also highly suitable for cryptographic designs since they allow for the parallelising of operations. Software based designs do not have this quality and therefore are restricted to sequential and consecutive designs. Additionally, CPUs have a fixed hardware structure, with constant memory structure and connections. FPGAs have fixed resources (such as slices), but the functions they perform and the interconnections between them are defined by the user.

The nature of FPGAs also means designs can be fairly compared, for any hardware design targeting the same device. Cryptography, especially lattice-based cryptography, is a very fast paced research area, with parameter and algorithmic changes common, thus FPGAs being reprogrammable is ideal.

1.2 Aims and Scope

The central aim of this thesis is to improve the efficiency, and therefore practicality, of lattice-based schemes through the use of optimised hardware architectures targeting FPGA platforms. It is also possible to port these hardware designs to application-specific integrated circuit (ASIC) devices [Oder et al., 2016].

The proposed novel architectures exploit the aforementioned qualities of the FPGA, gaining efficiencies by considering novel architectural and algorithmic

optimisations in comparison to existing research. More specifically, novel hardware designs are proposed for discrete Gaussian samplers, a lattice-based encryption scheme, and a lattice-based digital signature scheme, producing results that outperform previous research in terms of FPGA area consumption and speed. Where possible, the proposed hardware designs are also compared to their equivalences in software.

The discrete Gaussian sampler component is one of the main modules within *all* of lattice-based cryptography and can be seen as analogous to pseudo-random number generators (PRNGs), which are required for most cryptographic schemes. This "normalised noise" is added onto computations of secret-data to "hide" its values. There are a number of techniques for deriving discrete Gaussian samples, such as basic rejection sampling or by using sampling methods designed for this purpose. These specialised techniques are the Bernoulli sampler [Ducas et al., 2013], the cumulative distribution table (CDT) sampler [Peikert, 2010], the Knuth-Yao sampler [Knuth and Yao, 1976], and the discrete Ziggurat sampler [Marsaglia et al., 2000, Buchmann et al., 2013], which either generate samples via arithmetic or are table based.

A hardware design of the lattice-based public-key encryption scheme by Lindner and Peikert [2011] is investigated due to a number of security concerns within ideal lattice-based cryptography. The use of standard lattices by Lindner and Peikert [2011] appeases these security concerns and also provides stronger security in comparison to the state-of-the-art in ideal lattice-based encryption.

The lattice-based digital signature scheme by Akleylek et al. [2016] is also considered due to its higher security assumption, that is, a provably secure instantiation with worst-case to average-case hardness. These are qualities not provided by the state-of-the-art in lattice-based digital signature schemes.

1.3 Novel Contributions

The following novel contributions are presented in this thesis:

1. The first comprehensive analysis of the discrete Gaussian sampling component in hardware is undertaken [Howe et al., 2016a], with the following novel contributions:

   (a) Practical hardware designs of discrete Gaussian samplers are presented, and compared with current state-of-the-art architectures, for appropriate practical parameters, throughput, memory consumption, and resource count. The hardware designs follow novel optimisation strategies, with the results competing with, and in many cases, significantly outperforming, previous research.

   (b) Novel designs of the first independent-time discrete Gaussian samplers in hardware are proposed which provide resistance against timing analysis attacks.

   (c) Based on the performance results, concrete recommendations are given for the most appropriate sampler to use in particular applications.

   (d) A test suite is proposed for use with the discrete Gaussian component, which tests the correctness of any generic sampler output.

2. The first hardware design of a *standard* lattice-based cryptoscheme is presented:

   (a) The hardware architecture is based upon a standard-LWE public-key encryption scheme, on a Spartan-6 FPGA, and is designed to balance area and performance. Results are provided for encryption and decryption algorithms, showing for the first time that standard lattice-based cryptography is indeed practical.

(b) With the inclusion of a time-independent discrete Gaussian sampler, the entire encryption scheme also operates in independent time, therefore bypassing timing side-channel analysis.

3. The first hardware design of a provably secure lattice-based digital signature scheme is presented:

   (a) Separate signing and verifying hardware designs are proposed, with the design goal of low-area, provided with a variety of options for higher throughput, which can produce between 104-785 signatures and 102-776 verifications per second.

   (b) The first evaluation, in hardware, of a lattice-based digital signature scheme which uses an alternative multiplication method to the number theoretic transform (NTT), that is, schoolbook polynomial multiplication. The modular multiplication and low-Hamming weight components are completely adaptable to *any* parameter set, which is not the case if NTTs are used.

   (c) The research also investigates parameter selection for the signature scheme. As such, hardware-friendly parameters are proposed, which are shown to reduce the area consumption in comparison to the parameter set proposed by the authors.

1.4 Organisation of Thesis

The thesis is organised as follows:

In Chapter 2, the field of lattice-based cryptography is comprehensively introduced, and a detailed background is given. Related previous research is also discussed, in particular theoretical contributions as well as software and hardware research with regards to discrete Gaussian samplers, lattice-based encryption,

and lattice-based digital signature schemes. The chapter then presents a discrete Gaussian testing suite, a generic software-based statistical testing platform for the discrete Gaussian distribution. The purpose of this testing suite is to verify that hardware or software based discrete Gaussian samplers are actually outputting the correct distribution, which could otherwise lead to security issues within a lattice-based cryptoscheme.

In Chapter 3, novel hardware designs of efficient and time-independent discrete Gaussian samplers are proposed. Hardware designs are proposed for all the majorly used practical techniques for generating discrete Gaussian noise, with separate designs for encryption and signature applications, using and not using the on-chip memory on FPGA devices (BRAMs). The theoretical background and previous research is comprehensive, with conclusions and results provided with application-specific recommendations. All hardware designs and recommendations are evaluated on FPGA for demonstration of practicability.

In Chapter 4, the first hardware design of a standard lattice-based cryptoscheme is proposed. The novel encryption hardware designs significantly exceed expectations (standard schemes were previously considered *infeasible and impractical* in hardware) and even closely compete with ideal lattice-based encryption schemes. This research uses a discrete Gaussian sampler from Chapter 3 to produce an encryption scheme which runs completely in constant-time, offering resistance to timing side-channel analysis.

In Chapter 5, the first hardware design of a compact lattice-based digital signature scheme is proposed, which additionally is the first lattice-based hardware design with a provably secure instantiation. This chapter also considers the performance of a lattice-based signature scheme which does not employ the restrictive NTT polynomial multiplier. A generic schoolbook multiplier is proposed with Barrett modular reduction, and a variety of results are produced, optimised to trade-off between efficiency and compactness. The hardware designs also

incorporate a low-Hamming weight multiplier and the SHA3 post-quantum secure hash function, both of which are designed to operate outside the Sign and Verify critical path, and operate in parallel to the subsequent Sign/Verify operation.

Finally, the thesis is concluded in Chapter 6, which concludes each of the chapters of the thesis. Furthermore, a discussion on future areas of research is also given.

CHAPTER 2

Background and Theory

In this chapter, the fundamental mathematics, mathematical notations, and computationally hard problems of lattice-based cryptography (LBC) are defined. Within the mathematics of LBC, the computational hardness problems they are built on are discussed, as well as the hardness problems themselves. Additionally, the important lattice-based cryptoschemes such as key exchange, encryption, and signature schemes are introduced and discussed. Any security levels discussed in this chapter (and throughout this thesis) is classical security, that is security against a classical adversary, as opposed to post-quantum security from a quantum adversary. As well as the schemes themselves, the underlying modules they share, such as polynomial multipliers and discrete Gaussian samplers, are defined, with previous and related research also given. The necessary mathematical definitions used within these modules are also outlined, such as schoolbook multiplication (as used in Chapter 5) and the use of Gaussian convolutions (as used in Chapter 3). Significant portions of this chapter appear in the already published work by Howe et al. [2015].

Side-channel analysis is introduced in Section 2.7 as this is furthered in Chapter 3 for discrete Gaussian hardware designs. Another contribution of this chapter is the proposal of a test suite for discrete Gaussian samplers. Statistical test

suites are common for use in pseudo-random number generators (PRNGs), and as LBC becomes more prevalent, it is important to develop a method to test the correctness of discrete Gaussian samplers. Moreover, due to the theoretical requirements for discrete Gaussian samplers within LBC, certain statistical tests for distribution correctness become inappropriate, and therefore a number of tests are surveyed. The final test suite provides eleven tests which assess the exactness of a discrete Gaussian sampler, which can be used on any sampling technique. This research appears in the publication by Howe and O'Neill [2017].

2.1 Introduction

Post-quantum cryptography as a research field has grown substantially recently, essentially due to the growing concerns posed by quantum computers. The proviso being to provide long-term and highly secure cryptography, practical in comparison to RSA and elliptic-curve cryptography (ECC), but more importantly being adequately safe from quantum computers. This requirement is also hastened by the need for "future proofing" currently secure data, ensuring current IT infrastructures are quantum-safe *before* large-scale quantum computers are realised [Campagna et al., 2015].

As such, government agencies, companies, and standards agencies are planning transitions towards quantum-safe algorithms. The Committee on National Security Systems (CNSS) [CNSS, 2015] and the National Technical Authority for Information Assurance (CESG/NCSC) [CESG, 2016] are now planning drop-in quantum-safe replacements for current cryptosystems. The ETSI Quantum-Safe Cryptography (QSC) Industry Specification Group (ISG) [Campagna et al., 2015] is also highly active in researching industrial requirements for quantum-safe real-world deployments. NIST [NIST, 2016] have also called for quantum-resistant

cryptographic algorithms for new public-key cryptography standards, similar to previous AES and SHA-3 competitions.

Lattice-based cryptography [Ajtai, 1996, Regev, 2005] (LBC) is one of the most promising areas within post-quantum cryptography, and offers versatile, efficient, and high performance security services. LBC bases its hardness on finding the shortest (or closest) vector in a lattice, which is currently resilient to *all known* quantum reductions and hence attacks by a quantum computer. Furthermore, lattice-based cryptography also offers extended functionality whilst being more efficient than ECC and RSA based primitives of public-key encryption [Pöppelmann and Güneysu, 2014] and digital signature schemes [Howe et al., 2015].

Cryptography based on the hardness of lattice problems has gained a lot of attention from a theoretical point of view, but to date this can not be said for research into its practical performance. The main focus of this thesis is to extend the practical investigation of lattice-based cryptography (LBC), and to bring the field closer to becoming feasible for real-world instantiations. This is achieved by outperforming previous work, with regards to RSA, ECC, and other lattice-based designs, in terms of speed and/or area consumption and proposing countermeasures against certain types of attack vectors (mainly timing analysis). This chapter covers the relevant background on the theory of lattices and the cryptography based upon its hardness. In practice, the cryptoschemes do not operate with lattices directly but the computational learning problems they are based on are shown to be as hard as solving hard lattice problems, which will be explained more explicitly in this chapter.

The following sections focus on the preliminaries of LBC. The prevailing notation used throughout this thesis is presented in Section 2.2.1 and 2.2.2. The high-level theoretical encryption and digital signature scheme model is revisited in Sections 2.2.3 and 2.2.4, as these concepts will be used in Chapter 4 and 5,

respectively. Furthering this, the theory of lattices is discussed in Section 2.2.5 as well as the main worst-case hardness problems within lattice theory in Section 2.2.6. The hardness problems which are used within LBC are detailed in Section 2.2.7. Lattice-based cryptoschemes are then described, which begins with an introduction to types of cryptoschemes in Section 2.3, encryption algorithms in Section 2.4, DSS algorithms in Section 2.5, and the core components within these cryptoschemes in Section 2.6. Side-channel analysis is then introduced in Section 2.7 which is followed by the proposal for the discrete Gaussian testing suite in Section 2.8.

The chapter is then concluded in Section 2.9.

2.2 Preliminaries

2.2.1 Notation

Throughout this thesis, the following notation will be used. All vectors are column vectors, which are expressed with bold-face lower case letters, and matrices are represented by collections of column vectors, such as $\mathbf{M} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n)$, and are expressed with bold-face upper case letters. All logarithms used are to the base 2, so $\log_2(\cdot) = \log(\cdot)$. The ℓ_p-norm of a vector $\mathbf{v}$ is denoted as $\|\mathbf{v}\|_p$, where for the Euclidean length ($p = 2$) it is simplified to $\|\mathbf{v}\|$. An element $x \in \mathbb{Z}_q$ is exactly the element $x \in \mathbb{Z}$ reduced modulo q, represented in the range $[-\frac{q-1}{2}, \frac{q-1}{2}]$. Therefore, the operation $\mathbf{y} = \mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{x} \in \mathbb{Z}^m$, results in the element $\mathbf{y} \in \mathbb{Z}_q^n$. For an element s chosen uniformly at random from the set $\mathcal{S}$, the notation $s \xleftarrow{\$} \mathcal{S}$ is used.

2.2.2 Discrete Gaussians and Polynomial Rings

The Gaussian distribution, with standard deviation $\sigma \in \mathbb{R}$, centre $\mathbf{c} \in \mathbb{R}^n$, and evaluated at $\mathbf{x} \in \mathbb{R}^n$ is defined by a weight proportional to $\rho_{\mathbf{c},\sigma}(\mathbf{x}) = \exp(\frac{-\|\mathbf{x}-\mathbf{c}\|^2}{2\sigma^2})$. When the centre $\mathbf{c} = \mathbf{0}$ the notation is simply $\rho_\sigma(\mathbf{x})$. The centred discrete Gaussian distribution over $\mathbb{Z}^m$ is defined as $D_\sigma^m = \rho_\sigma(\mathbf{x})/\rho_\sigma(\mathbb{Z})^m$. The polynomial ring $\mathcal{R} = \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$ is defined such that all elements can be represented by polynomials of degree $n-1$, with coefficients in the range $[-\frac{q-1}{2}, \frac{q-1}{2}]$, with the subset $\mathcal{R}_k$ consisting of all polynomials with coefficients in the range $[-k, k]$. The set of the units of $\mathcal{R}$ is denoted as $\mathcal{R}^\times$.

2.2.3 Encryption Definitions

Formally, a public-key encryption (PKE) scheme is a tuple of probabilistic polynomial time algorithms, (KeyGen(1^n), Enc_{pk}, Dec_{sk}), which define the scheme's key generation, encryption, and decryption operations, respectively. KeyGen(1^n) takes the security parameter 1^n as input and outputs a key pair (pk, sk), where pk is the scheme's public-key used for encryption and sk is the scheme's secret-key used for decryption. The PKE operation $\text{Enc}_{pk}(m)$ takes the message data $m \in \mathcal{M}$, over the message space $\mathcal{M}$, and public-key pk, and outputs a ciphertext c. The Decryption operation $\text{Dec}_{sk}(c)$ takes the encrypted ciphertext data c and secret-key sk, and outputs the message m. A PKE scheme is $\delta(n)$-correct if and only if, for all sufficiently large security parameters N and for any $m \in \mathcal{M}$, $\mathbf{Pr}[\text{Dec}_{sk}(\text{Enc}_{pk}(m)) = m] \geq \delta(n)$ for $(pk, sk) \leftarrow \text{KeyGen}(1^n)$, where the probability is taken over the randomness of the algorithms (KeyGen(1^n), Enc_{pk}, and Dec_{sk}).

A PKE scheme is considered secure if it is shown to be unforgeable against a chosen plaintext attack (CPA). This is a model of security where an adversary should not be able to distinguish between two ciphertext outputs, given access to the public-key and the encryption oracle, of chosen message inputs. A model

with a higher level of security is the chosen ciphertext attack (CCA) model, in which an adversary can gain access to plaintext information from ciphertexts of their choice, that is the decryption oracle. A PKE is therefore CCA-secure if an adversary cannot gain any secret information given access to this decryption oracle. The PKE scheme by Lindner and Peikert [2011] is used for the hardware design in Chapter 4, which is CPA secure [Bansarkhani and Buchmann, 2015].

2.2.4 Digital Signature Definitions

Formally, a digital signature scheme (DSS) is a tuple of probabilistic polynomial time algorithms, (KeyGen(1^n), Sign_{sk}, Ver_{pk}), which define the scheme's key generation, signing, and verification operations, respectively. KeyGen(1^n) outputs the secret-key sk and public-key pk, $\text{Sign}_{sk}(\mu)$ takes as input a message $\mu \in \mathcal{M}$ and outputs a corresponding signature σ using sk, and $\text{Verify}_{pk}(\mu, \sigma)$ takes as input the message μ and signature σ, and outputs 1 if and only if (μ, σ) is a valid message/signature pair, otherwise outputs 0. A signature scheme is complete if $\forall sk, pk \leftarrow \text{KeyGen}$, $\forall \mu \in \mathcal{M}$ and any $\sigma \leftarrow \text{Sign}_{sk}$, it produces $\text{Verify}_{pk}(\mu, \sigma) = 1$.

For a DSS to be secure, it must be proven to be existentially unforgeable under a chosen message attack (EU-CMA) [Goldwasser et al., 1988]. Which means that an adversary wins if, given access to the verification key and signing oracle O_{Sign} (that is, pairs $(\mu_1, \sigma_1), (\mu_2, \sigma_2), \ldots, (\mu_q, \sigma_q)$) they are able to generate (in polynomial time) a valid signature of μ, according to $\text{Verify}_{pk}(\mu, \sigma)$, given that μ was not amongst those messages μ_i queried to O_{Sign}. Additionally, for a DSS in the random oracle model, where collision resistant hash functions are used, an adversary also has access to a hash oracle O_H.

A higher level of security is strong unforgeability; whereby given the same paradigm, an adversary wins if they are able to generate (in polynomial time) a valid signature of μ according to $\text{Verify}_{pk}(\mu, \sigma)$ given that (μ, σ) was not amongst

those (μ_i, σ_i) queried to O_{Sign}. A DSS is described as $(q_{\text{Sign}}, q_H, t, \epsilon)$-strongly unforgeable if, given at most q_{Sign} queries to the signing oracle, at most q_H queries to the hash oracle and running in time at most t; there is no adversary that succeeds with at least probability ϵ. The digital signature scheme by Akleylek et al. [2016], used for the hardware design in Chapter 5 is EU-CMA secure.

2.2.5 Lattice Definitions

The general definition of a lattice is a set of points in n-dimensional space with periodic structure. More formally, a lattice $\mathcal{L}$ is defined as

$$\mathcal{L} = \{x_1\boldsymbol{b}_1 + x_2\boldsymbol{b}_2 + \cdots + x_n\boldsymbol{b}_n \mid x_i \in \mathbb{Z}\},$$

given n-linearly independent vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$ known as *basis* vectors. Alternatively, a lattice can be defined as a discrete co-compact subgroup of $\mathbb{R}^m$. The *rank* of a lattice is n and the dimension m. A lattice is known as *full-rank* when $n = m$.

The *fundamental region* of a lattice is a convex region, known as a lattice's parallelepiped, that contains exactly one representative of each co-set (for a proof see Minkowski's theorem [Helfrich, 1985]). The exact area of the fundamental region is the area that is spanned by the basis vectors. Fundamental regions are disjoint and collectively span the entire lattice. The *determinant* of a lattice defines the density of the lattice points. Given a basis matrix $\boldsymbol{B}$, the determinant of a lattice is defined as $\det(\mathcal{L}) = |\det(\boldsymbol{B})|$.

A number of different bases will generate the same lattice which motivates *equivalence*. Two bases will generate the same lattice when any of the following occur: vectors of the basis matrix are permuted, vectors of the basis matrix are negated, or vectors are added to integer multiples of other vectors, or more concisely: the multiplication of the basis matrix $\boldsymbol{B}$ by any unimodular matrix $\boldsymbol{U}$.

This leads to the observation that, given a unimodular matrix $\boldsymbol{U}$, two bases $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are equivalent if and only if $\boldsymbol{B}_2 = \boldsymbol{B}_1\boldsymbol{U}$.

The *minimum distance* of a lattice $\mathcal{L}$, also referred to as the shortest (nonzero) vector, is defined as $\lambda_1(\mathcal{L}) = \min\{\|\boldsymbol{v}\| : \boldsymbol{v} \in \mathcal{L}\backslash\{\mathbf{0}\}\}$. This is then generalised to define the k^{th} *successive minima* as:

$$\lambda_k(\mathcal{L}) = \min\{r : \dim(\text{span}(\mathcal{L} \cap \mathcal{B}(r))) \geq k\},$$

such that the ball $\mathcal{B}(\mathbf{0}, r) = \{\boldsymbol{v} : \|\boldsymbol{v}\| \leq r\}$, of radius r and centre $\mathbf{0}$, contains at least k linearly independent vectors.

2.2.6 Computationally Hard Lattice Problems

Certain classes of optimisation problems for lattices, such as the shortest vector problem (SVP) and its inhomogeneous counterpart the closest vector problem (CVP), are analogous to problems in coding theory. The security of lattice-based cryptography is based on the conjectured intractability of SVP, a problem synonymous to the minimum distance problem in coding theory. There are many variations of SVP and the two most commonly considered in the literature are presented here. The first, SVP_γ, states that given a lattice basis $\mathbf{B}$, find a non-zero vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ such that $\|\mathbf{v}\| \leq \gamma\lambda_1(\mathcal{L}(\mathbf{B}))$. The second, GapSVP_γ, an approximate decision version, states that given a lattice basis $\mathbf{B}$ and integer d, output 1 if $\lambda_1(\mathcal{L}(\mathbf{B})) \leq d$ or 0 if $\lambda_1(\mathcal{L}(\mathbf{B})) > \gamma d$. Clearly, both problems are more difficult when γ is small, conversely becoming less difficult as γ increases. Lyubashevsky and Micciancio [2009] investigated the NP-hardness of SVP (as well as the bounded distance decoding (BDD) problem) and its variants for different γ constraints. They also provide proofs of the equivalence of the computational problems within certain approximation factors.

Analogous to the nearest codeword problem in coding theory, the task given by CVP is to find the closest lattice point (likewise, codeword) given a (target) vector $\mathbf{t} \in \mathbb{R}^m$. More formally, CVP_γ states that, given a lattice $\mathcal{L}$ and a vector $\mathbf{t} \in \mathbb{R}^m$, find a lattice point $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v} - \mathbf{t}\| \leq \gamma \text{d}(\mathbf{t}, \mathcal{L})$, where $\text{d}(\mathbf{t}, \mathcal{L})$ denotes the minimum distance between an arbitrary point in a vector space and a lattice point. The approximate decision version GapCVP_γ states that, given a lattice basis $\mathbf{B}$, a vector $\mathbf{t} \in \mathbb{R}^m$ and $d \in \mathbb{R}$, output 1 if $\text{d}(\mathbf{t}, \mathcal{L}) \leq d$ or 0 if $\text{d}(\mathbf{t}, \mathcal{L}) > \gamma d$. Equivalences have been shown (most thoroughly by Micciancio [2008]) between CVP and the shortest independent vector problem (SIVP) in their exact versions, under deterministic polynomial time rank-preserving reductions.

2.2.7 The Bases of Lattice-Based Cryptography

Concordantly, there are problems based on the worst-case hardness of lattices which form the foundation of cryptosystems. They are namely the learning with errors problem (LWE) and the short integer solution problem (SIS) and, as shown by Micciancio and Peikert [2013], both assert the exceptional property that they are as hard to solve in the average-case as they are in the worst-case of lattice problems (such as SVP or CVP). This worst-case hardness quality renders *all* cryptographic constructions based on it secure, under the assumption that worst-case lattice problems are hard. In other words, breaking the cryptographic construction implies an efficient algorithm for solving any instance of some underlying lattice problem, such as SVP or CVP.

The SIS problem was first proposed by Ajtai [1996] as an alternative to cryptosystems, such as RSA, based on the hardness of factorising large numbers. The problem is defined as follows: given random vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m \in \mathbb{Z}_q^n$, and integers n and q, find a short non-trivial solution $s_1, s_2, \ldots, s_m \in \mathbb{Z}^m$ such that;

$$s_1\boldsymbol{a}_1 + s_2\boldsymbol{a}_2 + \cdots + s_m\boldsymbol{a}_m \equiv \mathbf{0} \mod q,$$

(alternatively $\mathbf{As} \equiv \mathbf{0} \mod q$). Restricting the shortness of the solution, such that $0 \leq \|\mathbf{s}\| \leq \beta < q$, alters the problem from trivial to computationally hard. Additionally, β must also be large enough to ensure a solution exists. Setting $\sqrt{n \log q} < \beta < q$ satisfies this with Micciancio and Peikert [2013] showing β 'nearly' equal to q retains the hardness assumption.

The relationship this problem has to lattices is as follows. Let S be the set of all integer solutions $\mathbf{s} = (s_1, s_2, \ldots, s_m)$, then S is a lattice implying the solution to the SIS problem is simply to find a short vector in S. Common uses of the SIS problem are shown by Micciancio and Peikert [2013] and include one-way functions and collision-resistant hash functions, while Lyubashevsky [2009, 2012] shows its use in DSSs which are discussed in Section 2.3.

The LWE problem, first proposed by Regev [2005], has many applications but can be predominantly attributed to uses in public-key cryptography (PKC) and CCA-secure cryptosystems. The LWE definition is as follows: given some uniformly distributed $\mathbf{a}_i \in \mathbb{Z}_q^n$, integers n and q, and $b_i \equiv \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i \mod q$, where the secret-key $\mathbf{s}$ is chosen uniformly at random from $\mathbb{Z}_q^n$ and each e_i follows some small error distribution, find $\mathbf{s}$ given access to pairs $(\mathbf{a}_i, b_i)$. The problem naturally produces two forms, namely its search and decision variants. The search variant asks an adversary to find $\mathbf{s} \in \mathbb{Z}_q^n$ given $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{b} \equiv \mathbf{A}^T\mathbf{s} + \mathbf{e} \mod q$, whereas the decision variant asks an adversary to distinguish between $(\mathbf{a}_i, b_i)$ and $(\mathbf{a}_i, u_i)$ where u_i is chosen uniformly at random. Regev [2009] (with a sample preserving reduction shown by Micciancio and Mol [2011]) shows a search-to-decision reduction, meaning that any efficient distinguisher between LWE and uniform distributions can be used to recover the secret-key. It should be noted that the small error distribution has been widely studied [Dwarakanath and Galbraith,

2014, Micciancio and Peikert, 2013] and is chosen independently and identically from a Gaussian-like distribution. Taking the standard deviation $\sigma = \alpha q$, a quantum reduction was shown by Regev [2005], whereby LWE is as hard in the average-case as approximating lattice problems in the worst-case with an approximation factor $\tilde{\mathcal{O}}(n/\alpha)$ and $\alpha q \geq 2\sqrt{n}$. Classical reductions were shown by Peikert [2008] with an exponential modulus and by Brakerski et al. [2013] with a polynomial modulus; the latter introducing an efficient algorithm showing hardness of LWE for worst-case instances of standard lattice problems. The use of standard lattices is furthered in Chapter 4. Moreover, Brakerski et al. [2013] discuss the hardness of the respective ring variants, showing a hardness proof for ring-LWE and Ring-SIS with exponential modulus under the hardness of problems on general lattices.

The relationship LWE has with known hard problems of lattices is as follows. Consider the lattice $\mathcal{L}(\mathbf{A}) = \{\mathbf{y} \in \mathbb{Z}^m \mid \mathbf{y} \equiv \mathbf{As} \bmod q\}$ for some $\mathbf{s} \in \mathbb{Z}_q^n$; in the instances where each e_i are small, the LWE problem is asking an adversary to solve the CVP on the lattice $\mathcal{L}(\mathbf{A})$. Inherently, the value of $\mathbf{s}$ is not uniquely determined but one value is significantly more likely than the rest, thus LWE is a well-defined maximum likelihood problem. This abstraction can be extended to the decision variant of LWE: consider again $\mathcal{L}(\mathbf{A})$ and $\mathbf{b} = \mathbf{As} + \mathbf{e} \bmod q$, since $\mathbf{b}$ is significantly more likely to be decoded as a lattice point than some point $\mathbf{v} \in \mathbb{Z}_q^m$ chosen uniformly at random. The decision-LWE problem can also be made equivalent to a decision bounded distance decoding (BDD) problem, where the bound is the radius of $\mathbf{e}$.

Problems such as LWE and SIS are theoretically sound but lack efficiency in practice due to the need for large unstructured matrices. To alleviate this, the problems have been considered over some polynomial ring, which yields their sister-problems ring-LWE and ring-SIS. Adopting the problems within the ring setting produces a special class of lattices, these being *ideal lattices*, which Micciancio

[2007] shows ameliorates the impracticality of general lattices. Ideal lattices are generally considered in the quotient ring $\mathbb{Z}[x]/f$, for some monic polynomial f of degree n, where n is a power of 2, implying irreducibility over $\mathbb{Z}$. Common examples of such a function f are $f = x^n + 1$ or $f = x^{q-1} + x^{q-2} + \cdots + 1$ for some prime q.

Ring-SIS$_{q,n,m,\beta}$ is the ring variant of SIS and is defined as, given random vectors $\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m \in \mathcal{R}$, find a short non-trivial vector $\mathbf{s} \in \mathcal{R}$ such that $\|\mathbf{s}\|_\infty \leq \beta$ and $\mathbf{As} \equiv 0 \bmod q$.

Ring-LWE$_{q,n,m,\beta}$ is defined as, given a prime modulus $q \equiv 1 \bmod 2n$, random vectors $\mathbf{s}, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_m \in \mathcal{R}$, where $\mathbf{b}_i = \mathbf{a}_i\mathbf{s} + \mathbf{e}_i \bmod q$ (again $\mathbf{e}_i$ following some small error distribution) find $\mathbf{s}$ given access to pairs $(\mathbf{a}_i, \mathbf{b}_i)$.

The decision variant requires to distinguish between pairs $(\mathbf{a}_i, \mathbf{b}_i)$ and $(\mathbf{a}_i, \mathbf{u}_i)$ where $\mathbf{u}_i$ is chosen uniformly at random. Additionally, as shown by Lyubashevsky et al. [2013b], sampling $\mathbf{s}$ from the error distribution (instead of the uniform distribution) is shown to maintain the hardness assumption of the original ring-LWE, which allows the secret $\mathbf{s}$ to be short.

The reason ideal lattices are preferred is because of their simplified representation and subsequent smaller key-size [Micciancio, 2007] as well as the applicability of number theoretic transforms (NTTs), which improve operational run-times from operating in quadratic to quasi-linear complexity.

2.3 The Paradigms Of Lattice-Based Cryptosystems

The following section introduces LBC. The definitions and previous research of encryption schemes and digital signature schemes are presented. The section begins with an analysis of the different paradigms of these different cryptosystems

(important for DSSs[1]) which then motivates the research for the remaining chapters in this thesis. The cryptoschemes based on those paradigms are then presented, with previous research also discussed. DSSs based on the hardness of lattice problems generally fall into three categories, namely GGH/NTRUSign signatures, hash-and-sign signatures, and Fiat-Shamir signatures.

2.3.1 GGH and NTRUSign Signatures

The GGH cryptoscheme by Goldreich et al. [1996] and the NTRUEncrypt cryptoscheme by Hoffstein et al. [1998] were among the first shown to be based on the hardness of lattice problems, specifically based on solving the approximate closest vector problem. The difference between these schemes is that the latter can somewhat be seen as a special instantiation of the former. The GGH cryptosystem included a DSS, in turn forming the basis of NTRUSign by Hoffstein et al. [2003] which combined almost the entire design of GGH but uses the NTRU lattices employed in NTRUEncrypt. The predecessor to NTRUSign, NSS [Hoffstein et al., 2001a], was broken by Gentry et al. [2001; 2002] and incidently NTRUSign suffered the same fate with works by Nguyen and Regev [2009], which show experimental results recovering the secret-key with 400 signatures. Since Nguyen and Regev categorically show (without perturbation) NTRUSign to be absolutely insecure and Ducas and Nguyen [2012b] even broke further countermeasures and a version with perturbations, the descriptions will not be covered, and also since implementation results currently do not have practical applications.

2.3.2 Hash-and-Sign Signatures

DSSs based on the hash-and-sign paradigm follow seminal work by Diffie and Hellman [1976]. The concept follows the criterion that a message should be

[1]Lattice-based encryption schemes follow a singular, standard paradigm and therefore their description begins in Section 2.4.

hashed before being signed. That is, to sign a message, first hash μ to some point $h = H(\mu)$, which must be in the range of the trapdoor function f, the then acclaimed RSA being such a function. Once the message has been hashed, it is signed $\sigma = f^{-1}(h)$ and a verification algorithm checks that $f(\sigma) = H(\mu)$ to confirm whether (μ, σ) is a valid message/signature pair. This theory, by Bellare and Rogaway [1993], became the foundation for full-domain hash (FDH), with the hash function $H(\cdot)$ being modelled on a random oracle. Where f is a trapdoor permutation, the scheme is shown to be existentially unforgeable under a chosen-message attack.

The relation lattices have to hash-and-sign signatures is the intuition that a short basis for a lattice could provide such a trapdoor function. This led to the first proposal by Gentry et al. [2008] (GPV), showing a DSS based on the hardness of lattice problems. The idea was to use a preimage sampleable (trapdoor) function (PSF) that somewhat behaves like trapdoor permutations. The collision resistance of the trapdoor function proposed is the basis of security for the scheme, which consequently is shown to be as hard as SIVP or GapSVP.

As described in GPV and at a high-level, the public function $f_{\mathbf{B}}$ ($\mathbf{B}$ being the public basis for some lattice $\mathcal{L}$) being evaluated by some random input corresponds to choosing a random lattice point $\mathbf{v} \in \mathcal{L}$ and adding some "noise" via some relatively short error term $\mathbf{e}$, giving a point $\mathbf{y} = \mathbf{v} + \mathbf{e}$. Inverting $\mathbf{y}$ corresponds to *decoding* it to any sufficiently close lattice point $\mathbf{v}' \in \mathcal{L}$, though not necessarily $\mathbf{v}$ itself, whereby the noise term is large enough that many preimages exist. Given the trapdoor basis, it is easy to decode $\mathbf{y}$ using the sampling algorithm. However, with only access to the public basis matrix, it is on average a hard problem. Thus central to the scheme is the construction of trapdoor functions, with the necessary property that every output value has several preimages.

Using this approach, the scheme then similarly follows the generic DSS construction already seen. KeyGen outputs a personal uniformly random public-key

matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and an associated secret-key (trapdoor) matrix (with small coefficients) $\mathbf{S} \in \mathbb{Z}^{m \times m}$ such that $\mathbf{AS} \equiv 0 \bmod q$. Moreover, it also chooses a FDH $H(\cdot) : \{0,1\}^* \to \mathbb{Z}_q^m$. $\text{Sign}_{sk}(\mu)$ takes as input $\mu \in \mathbb{Z}^m$ and outputs a signature σ, independent of $\mathbf{S}$, such that $\mathbf{A}\sigma = H(\mu) \bmod q$. $\text{Verify}_{pk}(\mu, \sigma)$ returns 1 if and only if σ is in the domain and $\mathbf{A}\sigma = H(\mu) \bmod q$, or 0 otherwise. The scheme is proven to be strongly existentially unforgeable under a chosen-message attack since it is complete, that is, for all generated keys $(\mathbf{A}, \mathbf{S})$, all messages μ and all signatures σ, $\mathbf{A}\sigma = H(\mu) \bmod q$.

A more recent scheme by Micciancio and Peikert [2012] also adopts hash-and-sign, introducing a more efficient trapdoor than the one used in GPV. Improvements to the key generation were made by Alwen and Peikert [2011]. However, more noteworthy were the further reductions Micciancio and Peikert [2012] made to both. Comparatively, their contributions affirm simplicity and speed over GPV. The public-key from GPV is the pair $(\mathbf{A}, \mathbf{AS})$ whereas this scheme uses a public-key $(\mathbf{A}, \mathbf{AS} + \mathbf{G})$ for some matrix $\mathbf{G}$. The trapdoor for GPV is the basis matrix $\mathbf{A}$ for a lattice. For Micciancio and Peikert's scheme the trapdoor $\mathbf{A}$ is derived from a transformation of the fixed, public lattice denoted by the 'gadget' $\mathbf{G}$. Using this (fixing) method allows for fast, parallel and even offline calculations of the inversions, which is where many of the improvements are achieved. The scheme then follows the general $(\text{KeyGen}, \text{Sign}_{sk}, \text{Verify}_{pk})$ model.

2.3.3 Fiat-Shamir Signatures

An alternative way of constructing a DSS was proposed by Fiat and Shamir [1986] and Abdalla et al. [2002], which first proposes an identification scheme and then adapts it to a DSS by means of the Fiat-Shamir transformation. Identification schemes are between two parties, where one party (the prover) needs to convince the other party (the verifier) they are whom they claim to be. The technique

can be observed by considering the protocol by Schnorr [1989], a frequently used proof of knowledge protocol based on the intractability of the discrete logarithm problem. Details of the transformation are omitted (see Galbraith [2012] for details) but it suffices to say that the security of the signature scheme follows if the hash function $H(\cdot) : \{0,1\}^* \rightarrow \{0,1\}^k$, for a suitable value of k, is considered as a random oracle.

Lattice-based signature schemes which use the Fiat-Shamir transformation are mainly due to research by Lyubashevsky et al. [Lyubashevsky, 2009; Lyubashevsky, 2012; Güneysu et al., 2012; Abdalla et al., 2012; Bai and Galbraith, 2014b; Ducas et al., 2013]. The procedures in the first publication by Lyubashevsky [2009] are shown to be based on SIS. That is, if a solution is found for the DSS then a solution is also found for SIS. The initial step taken in this scheme is to first construct a lattice-based identification scheme whereby the challenge is treated as a polynomial in $\mathcal{R}$. The security of the identification scheme is based on the hardness of finding the approximate shortest vector in the standard model as well as the random oracle model. The identification scheme is then transformed into a DSS where optimisations are made to the tight parameter settings, improving elements such as the length of the signature and making it computationally infeasible to find collisions in the hash function family $\mathcal{H}$. For a complete description of the scheme see Section 3.2 in the research by Lyubashevsky [2009].

The security of the scheme is dependent on the hardness of finding collisions in certain hash function families. An adversary who is able to forge a signature can then use this to find a collision in a hash function chosen randomly from $\mathcal{H}$, meaning that if the DSS is not strongly unforgeable then there exists a polynomial time algorithm that can solve SVP_γ for $\gamma = \tilde{\mathcal{O}}(n^2)$ in the ideal $\mathcal{R}$. Therefore, forging a signature and furthermore finding a collision in a randomly chosen $h \leftarrow \mathcal{H}$ is equivalent to finding short vectors in a lattice over $\mathcal{R}$, that is, the ring-SIS problem.

The subsequent improvements made by Lyubashevsky [2012] (LYU) were two-fold. The most significant change is that of the hardness assumption used, adapting from ring-SIS to ring-LWE, which is shown to significantly decrease the sizes of the signature and the keys, thereby improving efficiency. The second improvement is during the signing procedure, which involves asymptotically shorter signatures. This stage requires more complicated rejection sampling, so that the signatures are independent from the secret-key, and sampling from the normal distribution, wherein highly accurate computations are needed (see Section 4 in the research by Lyubashevsky [2012]). The scheme, as in the previous scheme, is shown to be strongly unforgeable and is based on the worst-case hardness of finding short vectors in a lattice.

The general structure of these DSSs by Lyubashevsky [2009, 2012] are as follows. Consider the secret-key as an $m \times n$ matrix $\mathbf{S}$ with small coefficients, and the public-key as the pair $(\mathbf{A}, \mathbf{T})$ where $\mathbf{A}$ is an $n \times m$ matrix with entries chosen uniformly at random from $\mathbb{Z}_q$ and $\mathbf{T} \equiv \mathbf{AS} \bmod q$. Also an essential part of the scheme, as already discussed, is the hash function, which is considered as a random oracle outputting elements in $\mathbb{Z}^m$ with small norms. In order to sign a message μ, the signing algorithm first chooses a random $\mathbf{y}$ from some discrete Gaussian distribution, then computes $\mathbf{c} = H(\mathbf{Ay} \bmod q, \mu)$ where the (potential) signature is the pair $(\mathbf{z}, \mathbf{c})$ such that $\mathbf{z} = \mathbf{Sc} + \mathbf{y}$, which is sent to the verifier should it pass the rejection stage. Finally, the verification algorithm checks that $\|\mathbf{z}\|$ is small and that $\mathbf{c} = H(\mathbf{Az} - \mathbf{Tc} \bmod q, \mu)$. The details of the discrete Gaussian stage and the importance of the restrictions on $\mathbf{z}$ will become evident later.

In many cases (such as research by Güneysu et al. [2012], Ducas et al. [2013], and Bai and Galbraith [2014b]) the signature is considered as $(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$, allowing the shortening of $\mathbf{z}_1$ or $\mathbf{z}_2$, which motivates various compression techniques. The scheme by Bai and Galbraith [2014b] (BG) is based on the LWE signature scheme LYU, whereby $\mathbf{z}_2$ is actually omitted from the scheme entirely. This is essentially

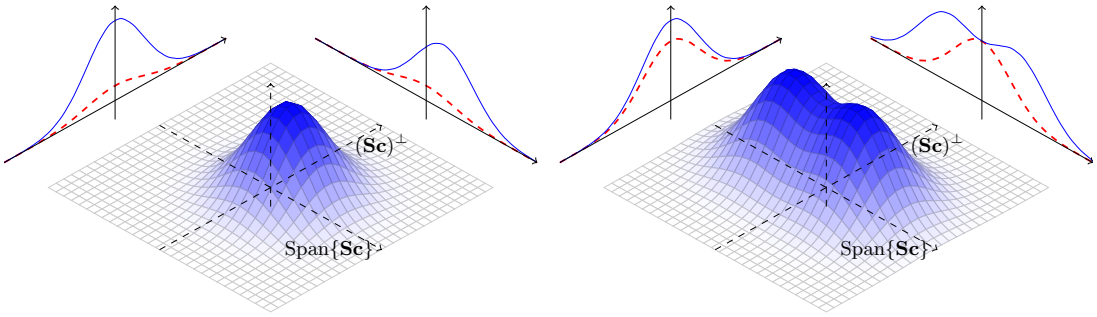


Fig. 2.1 The graphs show the improvement bimodal Gaussians have to the rejection sampling stage. The left (a) showing the LYU [Lyubashevsky, 2012] scheme and the right (b) showing the BLISS [Ducas et al., 2013] scheme. The distribution of $\mathbf{z}$ is shown in blue, fixing $\mathbf{Sc}$ and over the space of all $\mathbf{y}$ in (a) and all $(b, \mathbf{y})$ in (b), before the rejection step and its decomposition as a Cartesian product. The dashed red curves represent the scaled $(1/M)$ target distribution. Notice that the likeliness of acceptance is much greater for (b) than (a).

achieved by adapting the scheme so that proof of knowledge of the pair $(\mathbf{s}, \mathbf{e})$ for the LWE public-key $(\mathbf{A}, \mathbf{b} = \mathbf{As} + \mathbf{e} \mod q)$ only requires knowledge of $\mathbf{s}$. Algorithmic improvements (in particular within key generation) to the BG signature scheme were made by Dagdelen et al. [2014], which was followed by security improvements by Alkim et al. [2015]. The DSS by Alkim et al. [2015] is named TESLA and its hardness is based on standard lattice problems. TESLA is adapted to ideal lattices in the research by Akleylek et al. [2016], named Ring-TESLA, which is the bases of the research in Chapter 5.

The current state-of-the-art in lattice-based DSSs is the proposed schemes by Ducas et al. [2013] named BLISS and Akleylek et al. [2016] named Ring-TESLA. The main contribution of BLISS is the significant improvement in the rejection sampling stage, the major reduction in parameter sizes, and hence the substantial reduction in key/signature size (see Table 2.5). As a consequence, this scheme presents an important bridge between theoretical and practical lattice-based DSSs.

To illustrate the importance of the rejection sampling stage to security, consider the following DSS. The signer has a (short) secret-key pair $\mathbf{s}_1, \mathbf{s}_2 \in \mathcal{R}$ and a public-key pair $(\mathbf{a}, \mathbf{t})$ where $\mathbf{a} \in \mathcal{R}$ is chosen at random and $\mathbf{t} = \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$. The signer then randomly chooses $\mathbf{y}_1, \mathbf{y}_2 \in D_\sigma^m$ and sends $\mathbf{u} = \mathbf{a}\mathbf{y}_1 + \mathbf{y}_2$ to the verifier who returns a sparse $\mathbf{c} \in \mathcal{R}$. The signer then calculates $\mathbf{z}_i = \mathbf{y}_i + \mathbf{s}_i\mathbf{c}$ (for $i \in \{1, 2\}$) and sends $\mathbf{z}_1, \mathbf{z}_2$ to the verifier where it is checked that $\|\mathbf{z}_i\|$ are small and $\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2 - \mathbf{t}\mathbf{c} = \mathbf{u}$.

Using this scheme as it is, there is an inherent vulnerability in the values $\mathbf{z}_i$ sent to the verifier. As stated, $\mathbf{y}_i$ is chosen from the distribution D_σ^m, therefore an adversary knows the distribution of $\mathbf{z}_i$ since they will follow the distribution of $\mathbf{y}_i$ skewed by the addition of $\mathbf{s}_i\mathbf{c}$, which is where the secret-key becomes susceptible. This can be rectified by adapting the distribution of $\mathbf{z}_i$ from $D_{\mathbf{S}\mathbf{c},\sigma}^m$ so that it follows the same distribution as $\mathbf{y}_i \sim D_\sigma^m$. This is achieved through rejection sampling, the idea is to find a value M such that for all (or all but a negligible) x, $f(x) \leq M \cdot g(x)$. Values for x are then drawn from $g(x)$ and accepted with probability $\frac{f(x)}{M \cdot g(x)} \leq 1$, otherwise the process is restarted. If the previous condition is satisfied $\forall x$, then the method will produce exactly the distribution $f(x)$.

This technique can be used for the above scheme, however this imposes a slight hindrance. Since both distributions follow a Gaussian-like distribution, M must be quite large (for instance in LYU, $M = 7.4$) to satisfy the condition $D_\sigma^m(\mathbf{x}) \leq M \cdot D_{\mathbf{S}\mathbf{c},\sigma}^m(\mathbf{x})$ for all $\mathbf{x}$, where the problem exists in the tails of the distributions. The effect of having a large M is that the probability of acceptance is significantly smaller, therefore requiring more samples, incurring inefficiency. By virtue of their novel use of bimodal Gaussians, BLISS currently show the most optimal value for this stage in Fiat-Shamir inspired DSSs, presenting a value $M = 1.6$. A juxtaposition of this is shown in Figure 2.1[2], illustrating how the probability of acceptance significantly increases with the introduction of bimodal Gaussians. Further comparing the repetition rates in other schemes (see Table 2.2

[2]This figure is in the BLISS publication and used with the authors' permission.

for GLP, Table 2.3 for BLISS and given that the repetition rate in GPV ≈ 10) it becomes evident just how significant BLISS is as a practical lattice-based DSS.

To be able to generate the discrete bimodal Gaussian, the scheme is slightly modified in the following way. Choose a bit $b \in \{0, 1\}$ uniformly at random and change the calculation of $\mathbf{z}_i$ such that $\mathbf{z}_i = \mathbf{y}_i + (-1)^b \mathbf{s}_i \mathbf{c}$. Therefore, $\mathbf{z}_i$ will follow the discrete bimodal Gaussian $\frac{1}{2} D^m_{\mathbf{Sc},\sigma}(\mathbf{x}) + \frac{1}{2} D^m_{-\mathbf{Sc},\sigma}(\mathbf{x})$. With some other small alterations in the signing and verifying stages, a completely practical and secure lattice-based DSS is achieved.

An improvement to BLISS was proposed by Ducas [2014a] (named BLISS-B), which changes the geometric bound from BLISS to a more tight and natural bound, and proposes a more efficient way to calculate the products of the secret-key polynomials and hash output polynomial. The algorithmic change improves upon BLISS by at least a factor of 1.2 in terms of speed-up.

The Ring-TESLA DSS by Akleylek et al. [2016] is an ideal lattice-based equivalent of TESLA, the standard lattice-based DSS by Alkim et al. [2015], both of which focus on tightly secure and provably secure instantiations. Post-publication of Ring-TESLA, a flaw in the security reduction was found. The flaw does not lead to an actual attack nor does it affect the security of the scheme, however the specific instantiations are affected[3]. The fix for this is ongoing, with attempts made by Barreto et al. [2016] and Chopra [2016].

Algorithmically, the schemes are based on the BG DSS [Bai and Galbraith, 2014b] and the optimisations of the BG scheme by Dagdelen et al. [2014], which will now be discussed further. The BG scheme was inspired by the seminal works of Lyubashevsky et al. [Lyubashevsky, 2009, 2012, Güneysu et al., 2012, Abdalla et al., 2012, Ducas et al., 2013], and its intent was a reduction in signature size

[3]The security flaw is also discussed on the TESLA homepage (https://tesla.informatik.tu-darmstadt.de/de/tesla/).

(by omitting the requirement for $\mathbf{z}_2$), and also importantly removing the necessity for costly discrete Gaussian sampling during Sign and Verify operations[4].

At a high-level, the LYU signature schemes act like a proof of knowledge of the secret pair $(\mathbf{s}, \mathbf{e})$, however the authors adapt this paradigm to create a scheme where proof of knowledge of $\mathbf{s}$ suffices. This change provides a significant reduction in signature size. Additionally, the discrete Gaussian requirements are now a part of KeyGen, whereby the public-key is of the form $(\mathbf{A}, \mathbf{T} \equiv \mathbf{AS} + \mathbf{E} \mod q)$, where $\mathbf{E}$ follows the discrete Gaussian distribution, and is a LWE instance. A BG signature is computed by choosing a uniform vector $\mathbf{y}$ and using the product of $\mathbf{v} \equiv \mathbf{Ay} \mod q$ in the hash with the message μ to produce a vector $\mathbf{c}$. The signature is then returned as the pair $(\mathbf{z} = \mathbf{y} + \mathbf{Sc}, c)$ so long as the signature is found to be independent of the secret. Verifying the signature requires the lower bits of $\mathbf{w} = \mathbf{Az} - \mathbf{Tc} \equiv \mathbf{Ay} - \mathbf{Ec} \mod q$, where if $\mathbf{Ec}$ is small enough, the MSB of $\mathbf{w}$ will match those of $\mathbf{v}$ and therefore the MSBs of the hash output using $\mathbf{w}$ will match those that used $\mathbf{v}$.

The optimisations by Dagdelen et al. [2014] include the incorporation of the public-key $\mathbf{A}$ as a *global constant* and a rearranging of the Sign algorithm. The rearrangement of Sign has changed the calculation of $\mathbf{w} \equiv \mathbf{Az} - \mathbf{Tc} \mod q$ in BG to $\mathbf{w} \equiv \mathbf{Ay} - \mathbf{Ec} \mod q$, which saves on operational costs since $\mathbf{v} \equiv \mathbf{Ay} \mod q$ is already calculated beforehand. Also, calculating $\mathbf{Ec}$ means the dense key $\mathbf{T}$ is no longer required during Sign, and instead the smaller keys $(\mathbf{S}, \mathbf{E})$ are used.

The TESLA DSS by Alkim et al. [2015] is essentially the cryptoscheme by Dagdelen et al. [2014], with improvements to security such as minimising the underlying assumptions, removing the requirement of a secure PRNG for signing, and improving the performance of its implementation in software. The provably secure ideal lattice-based version of TESLA, Ring-TESLA, whilst not requiring

[4]Removing such a costly discrete Gaussian sampling stage was also the motivation for the design of GLP.

on-device discrete Gaussian sampling, also assures provable security, parameters are chosen according to its provided security reduction (as opposed to GLP and BLISS), and offers an efficient lattice-based DSS based on Ring-LWE which competes with GLP and BLISS in software.

2.4 Lattice-Based Encryption Schemes

Lattice-based encryption schemes have been proposed in works by Ajtai and Dwork [1997] and Regev [2004,2005] and were improved with chosen-ciphertext security by Peikert and Waters [2008] and Peikert [2008]. However, in seminal work by Lindner and Peikert [2011] (now referred to as LP), an encryption scheme with concrete parameters and security estimates is provided, which is shown to be significantly more efficient than previous proposals. The ring-variant of the LP scheme is described by Lindner and Peikert [2011] and Lyubashevsky et al. [2010], which is extended by Lyubashevsky et al. [2013a] (now referred to as LPR), and shows improvements in shrinking the key sizes by a factor of at least 200, which then allows computational improvements in the encryption algorithm. The LP and LPR algorithms will now be described, both are extremely similar and only differ in their use of matrices and vectors (for LP) or polynomials (for LPR). Additionally, for 128-bit security, the medium parameter set $(n, q, \sigma) = (256, 4093, 3.33)$ is taken from the work by Lindner and Peikert [2011], and is applied to both LP and LPR. Brakerski et al. [2013] show that the modulus $q = 4093$ can be simplified to $2^{12} = 4096$, without an effect on the scheme's security (more specifically it states the modulus does not need to be a prime number).

Algorithm 2.1 defines the key generation, encryption, and decryption matrix-vector operations in LP, and Algorithm 2.2 defines the key generation, encryption, and decryption polynomial operations in LPR.

Algorithm 2.1 The LWE Encryption scheme by Lindner and Peikert [2011] (LP)

procedure KeyGen($\mathbf{A}, 1^\ell$)
  $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times n}$
  $\mathbf{R}_1, \mathbf{R}_2 \leftarrow D_\sigma^{n \times \ell}$
  $\mathbf{P} \equiv \mathbf{R}_1 - \mathbf{A} \cdot \mathbf{R}_2 \mod q$
end procedure
procedure Enc($pk = (\mathbf{A}, \mathbf{P}), \mathbf{m} \in \Sigma^\ell$)
  $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \leftarrow D_\sigma^n \times D_\sigma^n \times D_\sigma^\ell$
  $\bar{\mathbf{m}} = \text{encode}(\mathbf{m})$
  $\mathbf{c}_1 \equiv \mathbf{e}_1^t \mathbf{A} + \mathbf{e}_2^t \mod q$
  $\mathbf{c}_2 \equiv \mathbf{e}_1^t \mathbf{P} + \mathbf{e}_3^t + \bar{\mathbf{m}}^t \mod q$
end procedure
procedure Dec($sk = (\mathbf{R}_2), \mathbf{c}_1^t, \mathbf{c}_2^t$)
  $\mathbf{m} = \text{decode}(\mathbf{c}_1^t \mathbf{R}_2 + \mathbf{c}_2)$
end procedure

Since Algorithm 2.1 is essentially equivalent to Algorithm 2.2 (with the latter using polynomials over the ring $\mathcal{R}_q$, instead of vectors and matrices), only the former will be described. The idea is to essentially hide the secret-key information within a LWE sample, where this then becomes the public-key information, and where the message data is also hidden with a LWE sample. Therefore, the public-key and message data appear to a passive adversary as uniformly random, which achieves the scheme's semantic security.

KeyGen derives a uniformly random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times n}$, two matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ drawn from the discrete Gaussian distribution with parameter σ, where the public-key is of the form $pk = (\mathbf{A}, \mathbf{P} \equiv \mathbf{R}_1 - \mathbf{A} \cdot \mathbf{R}_2 \mod q)$ and the secret-key is of the form $sk = \mathbf{R}_2$. The matrix $\mathbf{R}_1$ is only used during KeyGen. Discovering the secret information ($\mathbf{R}_2$) from the publicly available $\mathbf{P}$ is equivalent to solving the search variant of LWE.

Encryption has inputs $pk, \mathbf{m} \in \Sigma^\ell$ and first generates three error vectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ from the discrete Gaussian distribution, which are used to mask the encoded message data, $\bar{\mathbf{m}} = \text{encode}(\mathbf{m})$, in the ciphertext pair $\mathbf{c}_1 \equiv \mathbf{e}_1^t \mathbf{A} + \mathbf{e}_2^t$

Algorithm 2.2 The Ring-LWE Encryption scheme by Lyubashevsky et al. [2013a] (LPR)

procedure KEY GENERATION($\mathbf{a}, 1^{\ell}$)
  $\mathbf{a} \leftarrow \mathcal{R}_q^n$
  $\mathbf{r}_1, \mathbf{r}_2 \leftarrow D_\sigma^n$
  $\mathbf{p} \equiv \mathbf{r}_1 - \mathbf{a} \cdot \mathbf{r}_2 \in \mathcal{R}_q^n$
end procedure
procedure ENCRYPTION($\mathbf{a}, \mathbf{p}, \mathbf{m} \in \Sigma^n$)
  $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \leftarrow D_\sigma^n$
  $\bar{\mathbf{m}} = \text{encode}(\mathbf{m}) \in \mathcal{R}_q^n$
  $\mathbf{c}_1 \equiv \mathbf{e}_1 \cdot \mathbf{a} + \mathbf{e}_2 \in \mathcal{R}_q^n$
  $\mathbf{c}_2 \equiv \mathbf{e}_1 \cdot \mathbf{p} + \mathbf{e}_3 + \bar{\mathbf{m}} \in \mathcal{R}_q^n$
end procedure
procedure DECRYPTION($\mathbf{c}_1, \mathbf{c}_2, \mathbf{r}_2$)
  $\mathbf{m} = \text{decode}(\mathbf{c}_1 \cdot \mathbf{r}_2 + \mathbf{c}_2) \in \Sigma^n$
end procedure

mod q and $\mathbf{c}_2 \equiv \mathbf{e}_1^t\mathbf{P} + \mathbf{e}_3^t + \bar{\mathbf{m}}^t \mod q$. Decryption requires no number generation, just the decoding of the computation $\mathbf{c}_1^t\mathbf{R}_2 + \mathbf{c}_2$ given inputs $sk, \mathbf{c}_1, \mathbf{c}_2$.

The function used to encode/decode was proposed by Lindner and Peikert [2011] since some small noise is still present after decryption, resulting in an erroneous message retrieval. The proposed threshold encoding function maps the individual bits of the message data $m \in \{0, 1\}$ to integers of the form $\text{encode}(m) = m \cdot \lfloor \frac{q}{2} \rfloor$, where decoding returns a bit 1 if and only if $m \in [-\lfloor \frac{q}{4}, \rfloor, \lfloor \frac{q}{4} \rfloor)$, and 0 otherwise.

2.4.1 Summary and Evaluation of Lattice-Based Encryption Designs

In this section, a summary is provided of implementation results for practical lattice-based encryption schemes. The lattice-based scheme investigated here for which implementation results are available is LPR by Lyubashevsky et al. [2013a]. Table 2.1 shows results of previous hardware designs of LPR, however these benchmarks are not all on the same platform, hence they are not all directly

comparable. The LP standard lattice-based encryption scheme by Lindner and Peikert [2011] is the focus of Chapter 4, which presents the first practical results for this scheme.

Lattice-based encryption schemes are shown to be more efficient for the ideal lattice-based scheme by Lyubashevsky et al. [2013a] (built on previous work by Lyubashevsky et al. [2010]), which is essentially a ring-LWE variant of the original scheme based on standard LWE by Lindner and Peikert [2011]. However, there are many arguments to employing an encryption scheme based on standard lattices, instead of ideal lattices, and designs for standard lattice-based encryption can be currently seen as understudied. With regards to key and ciphertext sizes; the standard lattice-based encryption scheme LP has a public-key $(\mathbf{A}, \mathbf{P})$ of size 1180 kb, a secret-key $(\mathbf{R}_2)$ of size 394 kb, and a ciphertext size of 4.6 kb, whereas the ideal lattice-based encryption scheme LPR has a public-key $(\mathbf{a}, \mathbf{p})$ of size 13.3 kb, a secret-key $(\mathbf{r}_2)$ of size 6.7 kb, and a ciphertext size of 6.7 kb, both for 128-bit parameters. From a software point-of-view, Göttert et al. [2012] give a comprehensive evaluation of LP and LPR encryption schemes benchmarked on an Intel Core 2 Duo CPU running at 3 GHz with 4 Gb of RAM as well as Du et al. [2015] which benchmark LPR on a Core i7-4771 CPU running at 3.5 GHz and 8 GB of RAM. Other software-based implementations of the encryption schemes all target microcontrollers; Pöppelmann et al. [2015] target an ATxmega128A1 microcontroller clocked at 32 MHz, which produces 36 signing and 148 verification operations per second. Liu et al. [2015] also target an ATxmega128A1 microcontroller and produces 48 signing and 106 verifying operations per second. Other platforms have also been targeted; Boorghany et al. [2015] target a ATmega64 clocked at 8 MHz producing 2 signing and 5 verifying operations per second, and Boorghany and Jalili [2014b] which target a ATxmega64A3 clocked at 32 MHz, producing 6 signing and 13 verifying operations per second.

Table 2.1 Post-place and route results of ring-LWE (RLWE) encryption/decryption; for balanced designs by Göttert et al. [2012] (GFSBH) and Pöppelmann and Güneysu [2013] (PG13), and for compact/low-area designs by Roy et al. [2014b] (RVMCV) and Pöppelmann and Güneysu [2014] (PG14) on FPGA. Results are also provided for NTRUEncrypt by Kamal and Youssef [2009b] (KaYo), as well as for elliptic-curve encryption by Güneysu and Paar [2008] (GP) and Rebeiro et al. [2012] (RRM).

Operation & Algorithm	**Device**	**LUT/FF/SLICE**	**BRAM/DSP**	**MHz**	**Cycles**	**Ops/s**
RLWE Encrypt (GFSBH)	V6LX240T	298016/ – /143396	–/–	–	–	< 18200
RLWE Decrypt (GFSBH)	V6LX240T	124158/ – /65174	–/–	–	–	< 29540
RLWE Encrypt (PG13)	S6LX16	4121/3513/–	14/1	160	6861	23321
RLWE Decrypt (PG13)	S6LX16	4121/3513/–	14/1	160	4404	36331
RLWE Encrypt (PG13)	V6LX75T	4549/3624/1506	12/1	262	6861	38187
RLWE Decrypt (PG13)	V6LX75T	4549/3624/1506	12/1	262	4404	59492
RLWE Encrypt (PG14)	S6LX9	282/238/95	2/1	144	136212	1057
RLWE Decrypt (PG14)	S6LX9	94/87/32	1/1	189	66338	2849
RLWE Encrypt (RVMCV)	V6LX75T	1349/860/–	2/1	313	6300	49751
RLWE Decrypt (RVMCV)	V6LX75T	1349/860/–	2/1	313	2800	109890
NTRU [Enc/Dec] (KaYo)	XCV1600E	27292/5160/14352	–/–	62	–	–
ECC-P224 (GP)	XC4VFX12	1825/1892/1580	11/26	487	178000	2740
ECC-B233 (RRM)	XC5VLX85T	18097/ – /5644	–/–	156	1919	81300

The results presented in Table 2.1 show results for hardware designs of the ring-LWE encryption scheme LPR. The results shown are for ring-LWE LPR hardware designs [Göttert et al., 2012, Pöppelmann and Güneysu, 2013] and low-area ring-LWE LPR designs [Roy et al., 2014b, Pöppelmann and Güneysu, 2014] for comparison.

The first LPR hardware design by Göttert et al. [2012] is highly impractical in comparison to later designs, possibly due to the use of a fast Fourier transform (FFT) instead of the more efficient (in this case) number theoretic transform (NTT). Pöppelmann and Güneysu [2013] propose an efficient hardware architecture for the LPR encryption scheme, providing results for Spartan-6 and Virtex-6 FPGA platforms. The hardware design utilises the NTT polynomial multiplier by Pöppelmann and Güneysu [2012], a table-based discrete Gaussian sampler, and exploits block-RAM for importing and exporting all keys, message data, and ciphertext data. This is furthered for the design goal of low area [Pöppelmann and Güneysu, 2014], in which a small schoolbook multiplier is used (instead of

a large NTT multiplier) which uses the FPGA DSP. The hardware results show significant reduction in area consumption, at the cost of increased latency.

The main contribution of the hardware design by Roy et al. [2014b] is improvements within the NTT and discrete Gaussian sampling components. Firstly, pre-computed values stored in the design by Pöppelmann and Güneysu [2012] are instead computed on-the-fly, reducing BRAM accesses. Secondly, a high-speed discrete Gaussian sampler is proposed which reduces the clock cycle count previously achieved. The results show a much higher throughput in comparison to other LPR hardware designs.

2.5 Lattice-Based Digital Signature Schemes

In this section the practical lattice-based signature schemes in software and hardware are examined and discussed.

2.5.1 On the Instantiation of GPV Hash-and-Sign Signatures

The most significant instantiation of the schemes based on GPV is the work by Bansarkhani and Buchmann [2013], which amalgamates the scheme by Gentry et al. [2008] with the efficient trapdoor construction proposed by Micciancio and Peikert [2012]. Two variants are presented; a matrix version operating in the general setting and a more efficient ring version. The matrix version incorporates the preimage sampling algorithm of Micciancio and Peikert [2012] and a technique to add perturbation. Generating the perturbation vectors is essential so that the preimages do not give any information away about the secret-key. This is achieved by Peikert's [2010] convolution technique which also requires efficient square root computation. This component is the most time consuming of the

signing procedure, consuming over 60% of the overall runtime. The ring scheme [Bansarkhani and Buchmann, 2013] is shown to be based on the ring-LWE problem, adopting similar constructions and discrete Gaussian sampling is optimised by adopting the inversion transform method, rather than using rejection sampling. The actual implementation of the ring scheme (see Table 2.5) provides 100-bit security and uses the FLINT and GSL libraries[5] for basic arithmetic.

2.5.2 Practical Instantiations of Ideal Lattice-Based Fiat-Shamir Signatures

This section introduces the ideal lattice-based Fiat-Shamir signature schemes by Güneysu et al. [2012] (GLP) and Ducas et al. [2013] (BLISS) in more detail, whilst also examining the computational efficiency of each of their components. The reasons for the discussion of GLP and BLISS and common building blocks are that both schemes have been extensively analysed and currently offer the best trade-off between signature and key sizes as well as security. Thus they are currently considered to be the most practical lattice-based signature schemes.

GLP

Table 2.2 The Parameters of the GLP Signature Scheme by Güneysu et al. [2012].

Name of the scheme	GLP-I	GLP-II
Security	80-bits	$\geq$ 256-bits
(n, q)	$(512, 8383489)$	$(1024, 16760833)$
k	2^{14}	2^{15}
Repetition rate	7	7

The instantiation based on ideal-lattices by Güneysu et al. [2012] (GLP) follows the signature scheme of Lyubashevsky [2012] and specifically targets reconfigurable hardware and constrained devices. This is done by favouring uniformly random

[5]The fast library for number theory (FLINT) is available at http://www.flintlib.org/ and the GNU scientific library (GSL) is available at https://www.gnu.org/software/gsl/.

Algorithm 2.3 Key Generation for GLP

procedure KEYGEN(1^λ)
Generate sk where $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} \mathcal{R}_1$
Generate pk where $\mathbf{a} \xleftarrow{\$} \mathcal{R}$ and $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$
Return $(pk = (\mathbf{a}, \mathbf{t}), sk = (\mathbf{s}_1, \mathbf{s}_2))$
end procedure

Algorithm 2.4 Signing Algorithm for GLP

procedure SIGN($\mu, \mathbf{a}, \mathbf{s}_1, \mathbf{s}_2$)
$\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$

$\mathbf{c} \leftarrow \mathrm{H}\left((\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2)^{(1)}, \mu\right)$

$\mathbf{z}_1 \leftarrow \mathbf{s}_1\mathbf{c} + \mathbf{y}_1, \mathbf{z}_2 \leftarrow \mathbf{s}_2\mathbf{c} + \mathbf{y}_2$

if $\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-32}$, then go to step 1

$\mathbf{z}_2' \leftarrow$ Compress $(\mathbf{a}\mathbf{z}_1 - \mathbf{t}\mathbf{c}, \mathbf{z}_2, q, k - 32)$

if $\mathbf{z}_2' = \perp$, then go to step 1
Return $(\mathbf{z}_1, \mathbf{z}_2', \mathbf{c})$
end procedure

Algorithm 2.5 Verification Algorithm for GLP

procedure VERIFY($\mu, \mathbf{z}_1, \mathbf{z}_2', \mathbf{c}, \mathbf{a}, \mathbf{t}$)
Accept iff
$\mathbf{z}_1, \mathbf{z}_2' \in \mathcal{R}_{k-32}$ and
$\mathbf{c} = \mathrm{H}\left((\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2' - \mathbf{t}\mathbf{c})^{(1)}, \mu\right)$
end procedure

distributed noise over Gaussian noise for secret-keys and masking values, and by basing the hardness assumption on an 'aggressive' version of the decisional ring-LWE problem. This assumption is called the Decisional Compact Knapsack ($\mathrm{DCK}_{q,n}$) problem, whereby an adversary must distinguish between the uniform distribution over $\mathcal{R} \times \mathcal{R}$ and the distribution $(\mathbf{a}, \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2)$, where $\mathbf{a} \xleftarrow{\$} \mathcal{R}$ and $\mathbf{s}_1, \mathbf{s}_2 \xleftarrow{\$} \mathcal{R}_1$. However, the aggressive compression is a source of insecurity and in

the full version by Ducas et al. [2013] it has been shown that the security of the scheme is around 80-bits instead of the 100-bits claimed by Güneysu et al. [2012].

For reference, the GLP algorithms for KeyGen (Algorithm 2.3), Sign (Algorithm 2.4), and Verify (Algorithm 2.5) are provided as well as its parameters in Table 2.2 and key and signature sizes listed in Table 2.5. The secret-keys of the scheme are the random polynomials $\mathbf{s}_1, \mathbf{s}_2$ and the public-key is $(\mathbf{a}, \mathbf{t})$, where $\mathbf{a} \xleftarrow{\$} \mathcal{R}$ and $\mathbf{t} \leftarrow \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$. To sign a message μ, two 'masking' polynomials $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{\$} \mathcal{R}_k$ are chosen uniformly at random and $\mathbf{c} \leftarrow \mathrm{H}(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2, \mu)^{(1)}$ is computed where $\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2$ is the most expensive operation during the signing procedure. Note that the hash function is only evaluated on the most significant bits of the coefficients of the input, denoted by the $^{(1)}$ notation. The actual signature $\mathbf{z}_1, \mathbf{z}_2$ is computed as $\mathbf{z}_i \leftarrow \mathbf{s}_i\mathbf{c} + \mathbf{y}_i$ where the polynomial $\mathbf{c}$ is generated from 160 bits of the output of the random oracle (instantiated as a hash function) and just contains 32 coefficients which are ± 1. Since $\mathbf{c}$, $\mathbf{s}_1$ and $\mathbf{s}_2$ are small and contain a lot of zeros (that is, they are sparse) no modular reduction modulo q is necessary when computing $\mathbf{s}_1\mathbf{c}$ and $\mathbf{s}_2\mathbf{c}$. But before sending the signature, the low-cost rejection sampling step must be performed where the signature is only sent if and only if $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{R}_{k-32}$. The parameter k, which first appears in line 1 of the signing algorithm, controls the trade-off between the security and the runtime of the scheme. The smaller k is, the more secure the scheme becomes (and the shorter the signatures get), but the time to sign will increase. The $\mathrm{Compress}(\mathbf{a}\mathbf{z}_1 - \mathbf{t}\mathbf{c}, \mathbf{z}_2, q, k - 32)$ function is the reason for the relatively short signatures as a large amount of $\mathbf{z}_2$ is removed (the scheme also works without the compression). It encodes the carries that would have been caused by $\mathbf{z}_2$ into $\mathbf{z}_2'$ and ensures correctness so that $(\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2)^{(1)} = (\mathbf{a}\mathbf{z}_1 + \mathbf{z}_2' - \mathbf{t}\mathbf{c}^{(1)})$.

The GLP scheme has currently been implemented on reconfigurable hardware by Güneysu et al. [2012], CPUs by Güneysu et al. [2013], and microcontrollers by Boorghany and Jalili [2014a] (see Section 2.5.3 for further discussions).

Table 2.3 The Parameters of the BLISS Signature Scheme by Ducas et al. [2013].

Name of the scheme	BLISS-I	BLISS-II	BLISS-III	BLISS-IV
Security	128-bits	128-bits	160-bits	192-bits
(n, q)	(512,12289)	(512,12289)	(512,12289)	(512,12289)
Secret-key densities δ_1, δ_2	0.3 , 0	0.3 , 0	0.42 , 0.03	0.45, 0.06
Gaussian std. dev. σ	215.73	107.86	250.54	271.93
Weight of the challenge κ	23	23	30	39
Dropped bits d in $\mathbf{z}_2$	10	10	9	8
Verif. thresholds B_2, B_∞	12872, 2100	11074, 1563	10206,1760	9901, 1613
Repetition rate	1.6	7.4	2.8	5.2

BLISS

The most efficient instantiation of the BLISS signature scheme is based on ideal-lattices (as used by Lyubashevsky et al. [2010, 2013a]) with the BLISS KeyGen, Sign, and Verify algorithms given in Algorithms 2.6, 2.7, and 2.8. Parameters are listed in Table 2.3 with key and signature sizes given in Table 2.5.

The generation of keys involves uniform sampling of two small, sparse polynomials $\mathbf{f}$ and $\mathbf{g}$, computation of the rejection condition $N_\kappa(\mathbf{S})$, and the computation of $\mathbf{f}^{-1}$. Inverting $\mathbf{f}$ makes KeyGen significantly more complex, particularly in comparison to the equivalent stage in GLP.

For the signing stage, two polynomials $\mathbf{y}_1, \mathbf{y}_2$ are sampled from the discrete Gaussian distribution D_σ^n (instead of uniformly random as in GLP). Sampling from a Gaussian distribution is computationally very expensive due to the complex operations (such as calculation of the exponential function) or large tables (as shown by Dwarakanath and Galbraith [2014]). Section 2.6.2 describes the ongoing research in this area.

The calculation of $\mathbf{u}$ is simplified since the computation of $\mathbf{a}_1\mathbf{y}_1$ is performed in an FFT-enabled ring using modulus q, instead of $2q$. The computation of $\mathbf{u}$ is then finalised by multiplying the constant ζ with $\mathbf{a}_1\mathbf{y}_1$, and adding $\mathbf{y}_2$. The d most significant bits of $\mathbf{u}$ are then hashed with the message μ with the output being

Algorithm 2.6 Key Generation for BLISS

procedure KENGEN(1^λ)
Choose $\mathbf{f}, \mathbf{g}$ as uniform polynomials with exactly $d_1 = \lceil \delta_1 n \rceil$ entries in $\{\pm 1\}$ and $d_2 = \lceil \delta_2 n \rceil$ entries in $\{\pm 2\}$
$\mathbf{S} = (\mathbf{s}_1, \mathbf{s}_2)^t \leftarrow (\mathbf{f}, 2\mathbf{g}+1)^t$
Compute rejection condition $N_\kappa(\mathbf{S})$ that accepts approx. 25% of all keys
$\mathbf{a}_q = (2\mathbf{g}+1)/\mathbf{f} \bmod q$ (restart if $\mathbf{f}$ is not invertible)
Return $(pk = \mathbf{A}, sk = \mathbf{S})$ where $\mathbf{A} = (\mathbf{a}_1 = 2\mathbf{a}_q, q-2) \bmod 2q$
end procedure

Algorithm 2.7 Signing Algorithm for BLISS

procedure SIGN($\mu, pk = \mathbf{A}, sk = \mathbf{S}$)
$\mathbf{y}_1, \mathbf{y}_2 \leftarrow D_{\mathbb{Z}^n, \sigma}$
$\mathbf{u} = \zeta \cdot \mathbf{a}_1 \cdot \mathbf{y}_1 + \mathbf{y}_2 \bmod 2q$
$\mathbf{c} \leftarrow H(\lfloor \mathbf{u} \rceil_d \bmod p, \mu)$
Choose a random bit b
$\mathbf{z}_1 \leftarrow \mathbf{y}_1 + (-1)^b \mathbf{s}_1 \mathbf{c}$
$\mathbf{z}_2 \leftarrow \mathbf{y}_2 + (-1)^b \mathbf{s}_2 \mathbf{c}$
Continue with probability $1 \Big/ \left(M \exp\left(-\frac{\|\mathbf{Sc}\|^2}{2\sigma^2}\right) \cosh\left(\frac{\langle \mathbf{z}, \mathbf{Sc} \rangle}{\sigma^2}\right)\right)$ otherwise restart
$\mathbf{z}_2^\dagger \leftarrow (\lfloor \mathbf{u} \rceil_d - \lfloor \mathbf{u} - \mathbf{z}_2 \rceil_d) \bmod p$
Return $(\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$
end procedure

Algorithm 2.8 Verification for BLISS

procedure VERIFY($\mu, pk = \mathbf{A}, (\mathbf{z}_1, \mathbf{z}_2^\dagger, \mathbf{c})$)
if $\|(\mathbf{z}_1 | 2^d \cdot \mathbf{z}_2^\dagger)\|_2 > B_2$ then reject
if $\|(\mathbf{z}_1 | 2^d \cdot \mathbf{z}_2^\dagger)\|_\infty > B_\infty$ then reject
Accept iff $\mathbf{c} = H\Big(\Big\lfloor \zeta \cdot \mathbf{a}_1 \cdot \mathbf{z}_1 + \zeta \cdot q \cdot \mathbf{c} \Big\rceil_d + \mathbf{z}_2^\dagger \bmod p, \mu\Big)$
end procedure

interpreted as a polynomial $\mathbf{c}$. Similar to GLP, $\mathbf{c}$ is also sparse and small but is generated differently and more efficiently from the output of the hash function. The polynomial $\mathbf{c}$ is then multiplied by the secret-key polynomials $\mathbf{s}_1, \mathbf{s}_2$, where the polynomials $\mathbf{y}_1, \mathbf{y}_2$ are used to 'mask' the secret-key inside of the signature. Rejection sampling is then performed so that no information is leaked about the secret-key, whereby Sign may restart. The signature $\mathbf{z}_2$ is then compressed and $(\mathbf{z}_1, \mathbf{z}_2^{\dagger}, \mathbf{c})$ is returned.

The verification stage first validates the Euclidean and infinity norms of the signature, then the input to the hash function is reconstructed and it is checked whether the corresponding hash output matches $\mathbf{c}$ from the signature. It should be evident from this description that the most costly computational components in BLISS are the dense polynomial multiplication, discrete Gaussian sampling, and sparse multiplication stages (as described by Pöppelmann et al. [2014]).

The main optimisations made in the improved BLISS-B are found in the signing algorithm, which adapt the calculations of the signature polynomials. This is achieved by proposing an algorithm which reduces the norm of $\|\mathbf{Sc}\|$ by using ternary representation of the challenge vectors modulo 2, using the binary vector $\mathbf{c}$. The result produces a tighter and more natural bound. The verification algorithm is unchanged from BLISS.

The software results for BLISS and BLISS-B are shown in Table 2.5, which shows a speed-up of 1.2x for BLISS-B compared to BLISS, for the main 128-bit parameters. Hardware results for BLISS are shown in Table 2.6.

Ring-TESLA

There are several lattice-based DSSs which follow the original structure by Bai and Galbraith [2014b]. The most practical of these is Ring-TESLA, since it is based on ideal lattices. The parameters for Ring-TESLA are shown in Table 2.4 with KeyGen, Sign, and Verify algorithms given in Algorithms 5.1, 5.2, and

5.3, respectively. Currently, the only practical results for Ring-TESLA, as well as the BG and TESLA DSS (standard lattices), are provided in software (see Table 2.5). Hardware designs of Ring-TESLA are the basis for the research in Chapter 5.

The generation of the Ring-TESLA keys firstly assumes the generation of the global constant polynomials $\mathbf{a}_1, \mathbf{a}_2 \xleftarrow{\$} \mathcal{R}_q^{\times}$, which are publicly known in advance and can be shared amongst arbitrarily many signers. As already described, the signing procedure for Ring-TESLA does not require discrete Gaussian sampling since this is performed during key generation instead. This is achieved by generating the secret-key sk as $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow D_\sigma^n$, with the error polynomials $\mathbf{e}_1, \mathbf{e}_2$ going through a validity check to ensure the signatures are short and verify correctly. This check is defined as $\text{check}_{\text{E}}(\cdot)$, which uses a function $\max_k(\cdot)$, takes as input a vector and returns the k^{th} largest entries. The error polynomials $\mathbf{e}_1, \mathbf{e}_2$ are rejected if $\sum_{k=1}^{\omega} \max_k(\mathbf{e}_i)$ is greater than a threshold, for either $\mathbf{e}_1$ or $\mathbf{e}_2$. Otherwise the error polynomials are accepted, where the secret-key is output as $sk = (\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2)$ and the public-key is $pk = (\mathbf{t}_1, \mathbf{t}_2) = (\mathbf{a}_1\mathbf{s} + \mathbf{e}_1 \mod q, \mathbf{a}_2\mathbf{s} + \mathbf{e}_2 \mod q)$.

To sign a message μ, a uniform polynomial $\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$ is generated for use in the calculation of the signature and well as for validity checks. Firstly, it is used to calculate intermediate polynomials $\mathbf{v}_1 \equiv \mathbf{a}_1\mathbf{y} \mod q$ and $\mathbf{v}_2 \equiv \mathbf{a}_2\mathbf{y} \mod q$, which are input into the hash function $H(\cdot)$ along with the message data μ to output the bit-string c. An encoding function $F : \{0,1\}^{\kappa} \rightarrow \mathbb{B}_{n,\omega}$ (as described by Bai and Galbraith [2014b]) then maps this bit-string to a low-Hamming weight (LHW) polynomial $\mathbf{c}$ with ω 1 values and $n - \omega$ 0 values, where F is required to be close to an injective function. The LHW polynomial $\mathbf{c}$ is then used to calculate the signature $\mathbf{z} \equiv \mathbf{y} + \mathbf{sc}$ as well as the polynomials $\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$ and $\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$ which are used to check the validity of a signature.

The verification algorithm is essentially equivalent to the signing algorithm without the need for uniform polynomial generation. The intermediate polynomials

are calculated as $\mathbf{w}'_1 \equiv \mathbf{a}_1\mathbf{z} - \mathbf{t}_1\mathbf{c} \mod q$ and $\mathbf{w}'_2 \equiv \mathbf{a}_1\mathbf{z} - \mathbf{t}_2\mathbf{c} \mod q$, which are then input into the hash function to output c'. Should the signature size be valid as well as $c = c'$, the signature is verified.

Both sign and verify algorithms for Ring-TESLA incorporate full (and dense) polynomial multiplication computations ($\mathbf{a}_1\mathbf{y} \mod q$, $\mathbf{a}_2\mathbf{y} \mod q$ for sign and $\mathbf{a}_1\mathbf{z} \mod q$, $\mathbf{a}_2\mathbf{z} \mod q$ for verify) in the ring $\mathcal{R}$ in which an optimised multiplier can be used. They can also both exploit the use of a LHW polynomial multiplier for multiplications with the LHW polynomial $\mathbf{c}$.

Implementations of the BG inspired schemes, currently only available in software, are shown in Table 2.5. These include the scheme by Akleylek et al. [2016] (Ring-TESLA, Intel Core i7-5820K) which, compared to BLISS and BLISS-B implementations, illustrates the speed BLISS gains by employing NTRU assumptions. The standard lattice-based DSSs by Dagdelen et al. [2014] (BG, Intel Core i7-4770K) and Alkim et al. [2015] (TESLA, Intel Core i7-4770K) are also given in Table 2.5, but due to their much larger key sizes they cannot be considered practical, unless for highly secure applications.

2.5.3 Performance Evaluation

In this section, a summary is provided of implementation results for practical lattice-based DSSs, with some comparative results from classical schemes from the literature. As discussed in Section 2.3.1, there are currently no practical instantiations of the GGH signature scheme by Goldreich et al. [1996] and implementations of NTRUSign by Hoffstein et al. [2003], like Driessen et al. [2008] are vulnerable to cryptanalysis, so they will not be considered further. Lattice-based schemes investigated here for which implementation results are available are GPV by Gentry et al. [2008], LYU by Lyubashevsky [2012], GLP by Güneysu et al. [2012], BLISS by Ducas et al. [2013], TESLA by Dagdelen et al. [2014] and

Table 2.4 The Parameters of the Ring-TESLA Signature Scheme by Akleylek et al. [2016].

Name of the scheme	Ring-TESLA-I	Ring-TESLA-II
Security	80-bits	128-bits
(n, q)	(512,8399873)	(512,39960577)
Weight of the challenge ω	11	19
Gaussian std. dev. σ	30	52
Dropped bits d in $\mathbf{z}_2$	21	23
Error threshold L	814	2766
Sign/Verify thresholds B, U	$2^{21} - 1,993$	$2^{22} - 1,3173$
Repetition rate	4.3	2.9

Alkim et al. [2015], and Ring-TESLA by Akleylek et al. [2016] for DSSs. For a quick overview, all DSSs considered for evaluation, their secret-key, public-key and signatures sizes as well as available software (CPU) results are summarised in Table 2.5.

The software results compared in Table 2.5 are the ideal lattice-based DSSs designs by Ducas et al. [2013] (BLISS, Intel Core i7 3.4 GHz), Ducas [2014a] (BLISS-B, Intel Core i7 3.4 GHz), Akleylek et al. [2016] (Ring-TESLA, Intel Core i7-5820K), Güneysu et al. [2013] (GLP-I, Intel Core i5-3210M), Weiden et al. [2013] (LYU-ring, AMD Opteron 2.3 GHz), and Bansarkhani and Buchmann [2013] (GPV-ring, AMD Opteron 2.3 GHz) are also compared to standard lattice-based DSSs by Dagdelen et al. [2014] (BG, Intel Core i7-4770K) and Alkim et al. [2015] (TESLA, Intel Core i7-4770K). Since these benchmarks are not all on the same platform they are not all directly comparable, which is similar for the hardware results in Table 2.6.

The fastest DSS with regard to signing and also with the smallest signature (5.6 kb) is currently BLISS, and BLISS-B, (implemented in plain C) due to the low amount of rejections, fast Gaussian sampling using a large CDT table, and small parameters for n and q. The structural disadvantage of GLP (more rejections, larger n and q) is almost compensated by the optimised design by

Algorithm 2.9 Key Generation for Ring-TESLA

procedure KeyGen(1^λ, $\mathbf{a}_1$, $\mathbf{a}_2$)
  $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow D_\sigma^n$
  if $\text{check}_E(\mathbf{e}_1) = 0 \bigvee \text{check}_E(\mathbf{e}_2) = 0$ **then**
    Restart
  end if
  $\mathbf{t}_1 \equiv \mathbf{a}_1\mathbf{s} + \mathbf{e}_1 \mod q$
  $\mathbf{t}_2 \equiv \mathbf{a}_2\mathbf{s} + \mathbf{e}_2 \mod q$
  $sk \leftarrow (\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2)$, $pk \leftarrow (\mathbf{t}_1, \mathbf{t}_2)$
  return (sk, pk)
end procedure

Algorithm 2.10 Signing Algorithm for Ring-TESLA

procedure Sign(μ, $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{s}$, $\mathbf{e}_1$, $\mathbf{e}_2$)
  $\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$
  $\mathbf{v}_1 \equiv \mathbf{a}_1\mathbf{y} \mod q$
  $\mathbf{v}_2 \equiv \mathbf{a}_2\mathbf{y} \mod q$
  $c = H(\lfloor \mathbf{v}_1 \rceil_{d,q}, \lfloor \mathbf{v}_2 \rceil_{d,q}, \mu)$
  $\mathbf{c} = F(c)$
  $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{sc}$
  $\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$
  $\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$
  if $[\mathbf{w}_1]_{2^d}, [\mathbf{w}_2]_{2^d} \notin \mathcal{R}_{2^d-L} \bigvee \mathbf{z} \notin \mathcal{R}_{B-U}$ **then**
    Restart
  end if
  return $(\mathbf{z}, c)$
end procedure

Algorithm 2.11 Verification for Ring-TESLA

procedure Verify(μ, $\mathbf{z}$, c, $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{t}_1$, $\mathbf{t}_2$)
  $\mathbf{c} = F(c)$
  $\mathbf{w}'_1 \equiv \mathbf{a}_1\mathbf{z} - \mathbf{t}_1\mathbf{c} \mod q$
  $\mathbf{w}'_2 \equiv \mathbf{a}_2\mathbf{z} - \mathbf{t}_2\mathbf{c} \mod q$
  $c' = H(\lfloor \mathbf{w}'_1 \rceil_{d,q}, \lfloor \mathbf{w}'_2 \rceil_{d,q}, \mu)$
  if $c = c' \bigwedge \mathbf{z} \in \mathcal{R}_{B-U}$ **then**
    return 1
  else
    return 0
  end if
end procedure

Güneysu et al. [2013] using assembly optimisation and vectorisation (that is, AVX extensions). As verification almost only requires polynomial multiplication, the vectorised GLP design is twice as fast as BLISS.

Thus in the future, it is expected that improvements regarding BLISS, similar to the vectorisation ideas of Güneysu et al. [2013], could also be applied. Moreover, for the signing procedure of BLISS, the impact of higher security levels on performance is moderate as n and q stay the same, with the significant changes being in the discrete Gaussian sampler and number of rejections. As discrete Gaussian sampling is not needed for verification, the runtime of verification is basically independent of the security level. The LYU design by Weiden et al. [2013] is not competitive, mainly due to larger parameters and also because the implementation uses slow rejection sampling and relies on the NTL library for basic arithmetic. For GPV [Gentry et al., 2008], initial outputs and key sizes were many megabits long and even with improvements by Bansarkhani and Buchmann [2013], signature and key sizes are still large in practice, around 250 kb for security of around 100 bits. With the improvements proposed by Micciancio and Peikert [2012], their scheme alleviates the sizes of the signatures and keys to roughly 100 kb, a drastic improvement over GPV; however for practical applications this is still significantly large and the implementation cannot compete with GLP, BLISS, or Ring-TESLA.

Ring-TESLA is fairly appealing in comparison to GLP and BLISS, due to only requiring uniform random noise - therefore no need for a discrete Gaussian sampler - and signatures being uniformly distributed - meaning Huffman coding for shorter signatures is not required, which is used by Pöppelmann et al. [2014], and consumes significant computational resources. Ring-TESLA also has the attractive characteristic of being based on ring-LWE, as opposed to the NTRU assumptions of BLISS. This quality may seem attractive to practitioners since many NTRU-based schemes have been broken (see Section 2.3.1) as well as

the patenting issues surrounding NTRU. However, Ring-TESLA does suffer in comparison to BLISS by having significantly larger parameter sizes (bit sizes of polynomial coefficients are more than double versus BLISS), as well as an increase in key and signature sizes which can be seen in Table 2.5, although showing competitiveness against signing/verifying operations per second. Also, as already discussed, there is a security issue with Ring-TESLA.

Table 2.5 A summary of ideal and standard lattice-based DSSs and schemes based on classical assumptions. Results have been benchmarked on an Intel Core i7 at 3.4 GHz, 32GB RAM with openssl 1.0.1c, where performance has been scaled to 3.4 GHz based on cycle counts.

Scheme	Security	Sign. Size	*sk* Size	*pk* Size	Sign./s	Ver./s
BG	128-bits	12 kb	870 kb	1540 kb	2800	10000
TESLA-416	128-bits	10 kb	1540 kb	10650 kb	4600	13800
TESLA-768	>128-bits	19 kb	26346 kb	33817 kb	1430	3900
LYU-ring	100-bits	103 kb	103 kb	65 kb	36	260
GPV-ring	100-bits	240 kb	191 kb	300 kb	48	370
GLP-I	80-bits	9.5 kb	2 kb	12 kb	5300	75500
BLISS-I	128-bits	5.6 kb	2 kb	7 kb	8000	33000
BLISS-II	128-bits	5 kb	2 kb	7 kb	2000	33000
BLISS-III	160-bits	6 kb	3 kb	7 kb	5000	32000
BLISS-IV	192-bits	6.5 kb	3 kb	7 kb	2500	31000
BLISS-BI	128-bits	5.6 kb	2 kb	7 kb	9600	33000
BLISS-BII	128-bits	5 kb	2 kb	7 kb	5600	33000
BLISS-BIII	160-bits	6 kb	3 kb	7 kb	8000	32000
BLISS-BIV	192-bits	6.5 kb	3 kb	7 kb	6250	31000
Ring-TESLA-I	80-bits	11.4 kb	12.5 kb	23.5 kb	6600	20000
Ring-TESLA-II	128-bits	11.9 kb	13.7 kb	26 kb	6600	20000
RSA-2048	112-bits	2 kb	2 kb	2 kb	800	27000
RSA-4096	128-bits	4 kb	4 kb	4 kb	100	7500
ECDSA-256	128-bits	0.5 kb	0.25 kb	0.25 kb	9500	2500
ECDSA-384	192-bits	0.75 kb	0.37 kb	0.37 kb	5000	100

Regarding these implementations on constrained devices or microcontrollers; Oder et al. [2014] target an ARM Cortex-M4F microcontroller, which compares different samplers (Bernoulli, Knuth-Yao, and Discrete Ziggurat) and operates at

168 MHz. The device produces 28 signing, 167 verification and 0.46 key generation operations per second. Boorghany et al. [2015] and Boorghany and Jalili [2014a] provide an implementation of GLP and BLISS used as an identification scheme on 8-bit microcontroller architectures (Atmega and ATxmega), showing that lattice-based DSSs perform well even on very constrained devices. The Gaussian sampler is based on the CDT and the table currently fills a large part of the flash. However, the techniques of Pöppelmann et al. [2014] should be directly applicable to reduce the table size with a hopefully moderate impact on runtime. As the signature schemes are implemented as identification schemes their runtimes are not discussed.

For reconfigurable hardware, results are available for GLP and BLISS and are summarised in Table 2.6. While the speed of the GLP implementation, with roughly 1000 signing and verification operations per second, is good in comparison with classical schemes, the implementation by Güneysu et al. [2012] and particularly the usage of schoolbook multiplication is suboptimal given works on fast multiplication like Roy et al. [2013a], despite the larger area comsumption. The BLISS implementation by Pöppelmann et al. [2014] uses the NTT multiplier proposed by Pöppelmann and Güneysu [2012] and achieves high throughput for signing and verification. The resource consumption is also reasonable and the design fits on low-cost Spartan-6 devices. Usage of the improved NTT multiplier design by Roy et al. [2013a] might even give a further reduction of the resource consumption. For BLISS, two variants are given; one implementing the improved CDT approach and another one using the Bernoulli techniques of Ducas et al. [2013].

Table 2.6 A summary of hardware instantiations of DSSs on Virtex-5 (V5) and Spartan-6 (S6), comparing those based on lattice problems (GLP by Güneysu et al. [2012] and BLISS-I by Pöppelmann et al. [2014]) with those of RSA and ECDSA (results taken from Pöppelmann et al. [2014]).

Scheme	Security	Description	Device	Resources	Ops/s
GLP-I (Sign)	80-bits	$q = 8383489, n = 512$	S6 LX16	7,465 LUT/ 8,993 FF/ 28 DSP/ 29.5 BRAM18	931
GLP-I (Ver)	80-bits	$q = 8383489, n = 512$	S6 LX16	6,225 LUT/ 6,663 FF/ 8 DSP/ 15 BRAM18	998
BLISS-I (Sign)	128-bits	CDT sampler	S6 LX25	7,491 LUT/ 7,033 FF/ 6 DSP/ 7.5 BRAM18	7,958
BLISS-I (Sign)	128-bits	Bernoulli sampler	S6 LX25	9,029 LUT/ 8,562 FF/ 8 DSP/ 6.5 BRAM18	8,081
BLISS-I (Ver)	128-bits	-	S6 LX25	5,275 LUT/ 4,488 FF/ 3 DSP/ 4.5 BRAM18	14,438
RSA (Sign)	103-bits	RSA-2048; private key	V5 LX30	3,237 LS/ 17 DSPs	89
ECDSA (Sign)	128-bits	Full ECDSA; secp256r1	V5 LX110	32,299 LUT/FF pairs	139
ECDSA (Ver)	128-bits	Full ECDSA; secp256r1	V5 LX110	32,299 LUT/FF pairs	110

2.6 Building Blocks

The research on common building blocks used within the presented lattice-based encryption and signature schemes are examined. The reason is because these building blocks are essential for the understanding of the performance of LBC. For example, an improvement in one of these components would have a knock-on effect into improving lattice-based encryption or digital signatures.

2.6.1 Polynomial Multiplication

Comparable to point multiplication for ECC and exponentiation for RSA, matrix-vector multiplication is the basic operation in standard LBC and polynomial multiplication is the basic operation in ideal LBC. As a result, it has been polynomial multiplication that has been the subject of various optimisation efforts within LBC. While addition and subtraction in $\mathcal{R}$ are easy to realise with $\mathcal{O}(n)$

primitive operations in $\mathbb{Z}_q$, polynomial multiplication is much more complicated[6]. When computing the product $\mathbf{c} = \mathbf{a} \cdot \mathbf{b}$ in $\mathcal{R}$ the trick that $\mathbf{x}^n \equiv -1$ can be used for instant reduction $\mod \langle \mathbf{x}^n + 1 \rangle$. This leads to the obvious schoolbook approach

$$\mathbf{ab} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (-1)^{\left\lfloor \frac{i+j}{n} \right\rfloor} \mathbf{a}[i]\mathbf{b}[j]\mathbf{x}^{i+j \mod n} \mod q,$$

which generally requires n^2 multiplications and $(n-1)^2$ additions or subtractions. Schoolbook multiplication can either be performed row-wise, column-wise, or using a hybrid approach [Gura et al., 2004]. For row-wise multiplication, a multiplicand $\mathbf{b}[j]$ is fixed and the row, that is, the inner products $\mathbf{c}[i+j \mod n] = \mathbf{c}[i+j \mod n] + (-1)^{\left\lfloor \frac{i+j}{n} \right\rfloor} \mathbf{a}[i] \cdot \mathbf{b}[j] \mod p$ for $i, j \in \{0, \ldots, n-1\}$ are computed. Once a row is completed, the next $\mathbf{b}[j+1]$ is fixed. Another approach is column-wise multiplication, where partial products are used to sum up columns of $\mathbf{c}$. Thus, for column ℓ it is then necessary to compute $\mathbf{c}[\ell] = \sum_{i=0}^{n-1} (-1)^{1-\left\lfloor \frac{n+\ell-i}{n} \right\rfloor} \mathbf{a}[i]\mathbf{b}[n+\ell-i \mod n] \mod p$.

A first attempt to implement GLP on reconfigurable hardware was shown by Güneysu et al. [2012] where an array of fast schoolbook multipliers were used to compute $\mathbf{ay}_1 + \mathbf{y}_2$. The idea was that schoolbook multipliers are simple to realise and they can profit from different operand sizes as $\mathbf{a} \xleftarrow{\$} \mathcal{R}$ and $\mathbf{y} \xleftarrow{\$} \mathcal{R}_k$.

In the case where polynomials are sparse (polynomials with a lot of zero coefficients like $\mathbf{c}$ from GLP or BLISS), or have only many small coefficients (for example, just $-1/0/1$ coefficients like $\mathbf{s}_1, \mathbf{z}_2$ of GLP and BLISS), or both, schoolbook multiplication is an option. Güneysu et al. [2012] adopt column-wise multiplication to allow immediate rejection of out of bound z coefficients ($\mathbf{z}_1$ or $\mathbf{z}_2 \notin \mathcal{R}_{k-32}$). Exploiting the spareness of polynomials already played a role in

[6]Note that for encryption and signature schemes the number of polynomial coefficients n is usually in the range 256 to 1024 (see Table 2.2 and Table 2.3).

implementations of the NTRU public-key encryption schemes (see [Kamal and Youssef, 2009a]).

In order to achieve higher speed for polynomial-multiplication, namely quasi-linear runtime with $\mathcal{O}(n \log n)$ multiplications in $\mathbb{Z}_q$, the Fast Fourier Transform (FFT) or more specifically the Number Theoretic Transform (NTT) [Nussbaumer, 1980, Winkler, 1996, Blahut, 2010a] can be used to implement the negative wrapped convolution. The NTT is defined in a finite field or ring for a given primitive n^{th} root of unity ω and exists if n is a power of two and q is a prime satisfying $q \equiv 1 \mod 2n$[7]. The generic forward $\text{NTT}_\omega(\mathbf{a})$ of a sequence $\{\mathbf{a}[0], \ldots, \mathbf{a}[n-1]\}$ to $\{\mathbf{A}[0], \ldots, \mathbf{A}[n-1]\}$ with elements in $\mathbb{Z}_q$ and length n is defined as $\mathbf{A}[i] = \sum_{j=0}^{n-1} \mathbf{a}[j]\omega^{ij} \mod q$, for $i \in \{0, 1, \ldots, n-1\}$, with the inverse $\text{NTT}_\omega^{-1}(\mathbf{A})$ using ω^{-1} instead of ω [Winkler, 1996]. Thus the reduction by $x^n + 1$ is basically free and it is possible to work with a transform length equal to the number of polynomial coefficients. The NTT itself can be implemented using the common Cooley–Tukey radix-2 decimation-in-time approach [Cormen et al., 2009, Blahut, 2010a] where the main component is the butterfly structure computing $a' = a + \omega^l \cdot b \bmod q$ and $b' = a - \omega^l \cdot b \bmod q$ for $l \in [0, n/2 - 1]$. Naturally, the NTT has been studied before it has been applied to lattice-based cryptography with early works like McClellan [1976]. A recent example is by Emeliyanenko [2009] where the author provides an implementation of polynomial multiplication on graphics hardware (GPUs).

The first work [Göttert et al., 2012] proposing a polynomial multiplier specifically targeting lattice-based cryptography used the multiplier to implement an ideal lattice-based encryption scheme [Lyubashevsky et al., 2013a]. They compute every stage of the NTT in parallel and thus achieve high throughput, but as a consequence their design requires a large amount of device resources. An

[7] The NTT can also be defined for composite moduli, but only prime moduli are considered in this case.

Table 2.7 Post-place and route results of NTT multiplication components used within lattice-based cryptography, where possible[8], for dimension sizes $n = 256$ and 512. Results are by Pöppelmann and Güneysu [2012] (PG) and [Roy et al., 2013a] (RVMCV). The results are not provided by [Aysu et al., 2013] (APS), but are reported in a comparative work by Du and Bai [2016] (DB). Results by Du and Bai [2016] do not provide the modulus used, but it is assumed to be $q = 65537$.

Mult. Type (n, q)	**Device**	**LUT/FF/SLICE**	**BRAM/DSP**	**MHz**	**Cycles**	**Ops/s**
NTT-PG (256,65537)	S6LX100	1438/1123/520	3/1	209	4774	43778
NTT-PG (256,1049089)	S6LX100	1637/1507/640	7/4	218	4806	45359
NTT-PG (512,65537)	S6LX100	1585/1205/615	4/1	196	10014	19572
NTT-PG (512,5941249)	S6LX100	3228/2263/1145	7/4	193	10174	18969
NTT-RVMCV (256,1049089)	S6LX100	−/−/297	1/−	37	4683	787
NTT-BLISS (512,12289)	S6LX25	2557/2707/835	5/1	145	9307	15579
NTT-APS (256,65537)	S6LX100	528/463/208	1/2	185	> 5888	< 31419
NTT-APS (256,65537)	S6LX100	608/527/247	1/3	230	> 5888	< 39062
NTT-APS (512,65537)	S6LX100	490/532/211	2/2	184	> 12800	< 14375
NTT-APS (512,65537)	S6LX100	632/535/256	2/3	224	> 12800	< 17500
NTT-DB (256,65537)	S6LX100	533/538/214	2/1	233	4066	57304
NTT-DB (512,65537)	S6LX100	562/562/239	4/1	196	10014	19572

iterative approach to polynomial multiplication was proposed by Pöppelmann and Güneysu [2012] with the goal of achieving a reasonable area consumption and still high speed. Reasons for better area utilisation compared to the research by Göttert et al. [2012] are that multiplication in $\mathbb{Z}_q$ is performed using an embedded multiplier (via a DSP) and block memory (BRAM) to store polynomials.

Pöppelmann and Güneysu [2013] use the multiplier of Pöppelmann and Güneysu [2012] as a basis for a microcode engine/reconfigurable processor which also supports addition, subtraction, and random sampling of polynomials and allows the programmer fine-grained access to instructions realising the NTT; like NTT() realising the forward NTT, INTT() for the backward NTT, and PW_MUL() for point-wise multiplication. Improvements to the stand-alone polynomial multiplier design of Pöppelmann and Güneysu [2012] were proposed by Aysu et al. [2013] with similar performance, but a reduction of up to 67% of occupied slices and 80% of used BRAMs. This was achieved by better memory organisation and

[8]Separate multiplier results are not available for the research by Roy et al. [2014b], Du et al. [2016], Göttert et al. [2012], or Güneysu et al. [2012] (GLP)

concatenated storage of multiple coefficients in one memory address. The required powers of the twiddle factors, ω and ψ, were generated on-the-fly in an efficient manner and the design can be configured to use multiple dedicated multipliers (DSP). A parallel FFT multiplier ($n = 64$ and $q = 257$) targeting the lattice-based hash function by Lyubashevsky et al. [2008] (SWIFFT) was provided by Györfi et al. [2013] but no larger parameter sets, relevant for signature schemes, were evaluated. A microcode engine with a similar instruction set used by Pöppelmann and Güneysu [2013] and further improved multiplier was introduced by Roy et al. [2013a] to realise lattice-based encryption. The address generation of the NTT algorithm was rearranged and thus repeated multiplications to generate twiddle factors were eliminated and pre-computation like the multiplication with powers of ψ were avoided. Moreover, more efficient memory usage further reduced the amount of required BRAMs. Thus, this design[9], shown in Table 2.7, currently represents the state-of-the-art for polynomial multiplication for ideal lattice-based cryptography.

An optimisation of a NTT multiplier for larger parameter sets supporting somewhat homomorphic cryptography can be found in work by Chen et al. [2014]. Their design goal is high speed and low latency which is achieved by using two processing elements (PE), two additional integer modular multipliers and a very regular constant geometry NTT/FFT algorithm (seen in research by Pease [1968]).

The prime (q) used within the NTT has a significant impact on its hardware performance. As already stated, for NTT to be considered, the prime must be of the form $q \equiv 1 \mod 2n$. However, certain primes, such as Fermat [Agarwal and Burrus, 1974] or Mersenne [Rader, 1972] primes, allow for hardware simplifications; such as shifting in the butterfly unit (instead of multiplication), no storage requirements for twiddle factors, and simpler modular reduction [Baktir and Sunar, 2006, Blahut, 2010b]. This allows for low-area hardware designs, which

[9]This multiplier was also used in the hardware design for BLISS [Pöppelmann et al., 2014]

is investigated in the research by Aysu et al. [2013] and Du and Bai [2016], for the Fermat prime $q = 2^{16} + 1 = 65537$, with results shown in Table 2.7. Aysu et al. [2013] improve on previous research [Pöppelmann and Güneysu, 2012] by simplifying the memory usage, which is achieved by concatenating the coefficients of both polynomials (which have small sizes set as $\log_2(q) \approx 16$), which is achieved since the max width for 512 coefficients is 36-bits [Xilinx, 2011]. Aysu et al. [2013] also reduce hardware usage by computing square roots and twiddle factors on-the-fly. Du and Bai [2016] further these designs by increasing the throughput via optimising the bit-reverse operation of forward and backward NTT.

Targeting desktop CPUs, Güneysu et al. [2013] provide an optimised software implementation of the GLP signature scheme and also implement the NTT. The implementation is optimised for Intel's Sandy Bridge and Ivy Bridge in particular and targets the Advanced Vector Extensions (AVX) providing support for Single Instruction, Multiple Data (SIMD) operations. The C-implementation features storing of parameters in NTT representation, lazy reduction and representation of 512-coefficient polynomials as a 512 double-precision array of floating-point values. By utilising the AVX instruction set, that implementation can perform up to 4 multiplications and 4 additions of coefficients in each cycle from which also the NTT profits. Works such as those by Oder et al. [2014] and Boorghany and Jalili [2014a] also use the NTT as a building block but do not provide specific optimisations besides pre-computation and optimisation in assembly. Some optimisations of the implementation of the NTT on the Cortex-M4F microcontroller were recently proposed by Clercq et al. [2014] which mainly address parallelization, reduction of load and stores, as well as rearranging of the NTT algorithm as proposed by Roy et al. [2013a].

For dimensions $n \in \{256, 512, 1024\}$ and prime $q = 12289$, Longa and Naehrig [2016] investigate the software performance of the NTT. These parameters are chosen because they can be applied to the BLISS signature scheme and the

key-exchange proposal by Alkim et al. [2016]. The main contribution of the research by Longa and Naehrig [2016] is an ad-hoc modular reduction technique for the prime $q = 12289$, which minimises the use of multiplications. Results are provided for C and AVX2 implementations for key-exchange, which reports a speed-up of 1.49x in C and 1.13x faster for AVX2, in comparison to Alkim et al. [2016]

2.6.2 Discrete Gaussian Sampling

Sampling from the one dimensional discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ is an important building block for all LBC, but also a source of inefficiency in concrete implementations and thus the reason why GLP relies on uniform noise. The distribution D_σ is defined such that a value $x \in \mathbb{Z}$ is sampled from D_σ with the probability $\rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$ where $\rho_\sigma(x) = \exp\left(\frac{-x^2}{2\sigma^2}\right)$ and $\rho_\sigma(\mathbb{Z}) = \sum_{k=-\infty}^{\infty} \rho_\sigma(k)$. Conceptually, the simplest algorithm to sample from a Gaussian distribution is rejection sampling. One chooses a uniformly random $u \in \{-\tau\sigma, \ldots, \tau\sigma\}$ (in this case τ is denoted as a tail-cut) and accepts with a probability proportional to $\exp(-x^2/2\sigma^2)$. However, a straightforward implementation would require the costly computation of the $\exp(\cdot)$ function with high precision $\lambda \approx 128$-bits, a large number of random bits, and still result in ≈ 10 trials per sample. Research by Weiden et al. [2013] shows results in software, with some optimisations in the research by Ducas and Nguyen [2012a].

However, the approach can be optimised in order to reduce the amount of rejections. In the scheme by Ducas et al. [2013], the authors also make use of Bernoulli distributed variables. A Bernoulli distributed variable $\mathcal{B}_c$ outputs one with probability c, and zero otherwise. Sampling from this distribution is easy by evaluating if $y < c$ for a uniformly random $y \in [0, 1)$ and pre-computed c. The general idea of the proposed sampler is to reduce the probability of rejections by

sampling first from an intermediate and easily sampleable distribution, called the binary Gaussian distribution, and then from the target distribution. The rejection rate is thus decreased to ≈ 1.47 (compared to 10 for classical rejection sampling) and no computations of the exponentiation function $\exp(\cdot)$ or large pre-computed tables are necessary. The required table is small and just grows logarithmically. It has been used in the hardware design of Pöppelmann and Güneysu [2014] for the small standard deviation necessary for lattice-based public-key encryption. The sampler is also used in the BLISS hardware design by Pöppelmann et al. [2014] and the BLISS microcontroller implementation by Oder et al. [2014].

Another interesting approach to reduce the performance impact of rejections is the discrete Ziggurat [Buchmann et al., 2013]. The algorithm requires the computation of m same-area rectangles with the left corners on the y-axis and the right corners on the graph of the probability distribution function. The entire area under the graph is then covered by rectangles and a rectangle R_i can efficiently be stored by just storing the coordinates (x_i, y_i) of the lower right corner. To sample a value, a rectangle R_i is first sampled uniformly at random. The next step is to uniformly choose a value x within the sampled rectangle. If this x value is smaller or equal to the x coordinate of the previous rectangle, x is accepted, because all points $(x_j, y_j) \in R_i$ with $x_j \leq x_{i-1}$ definitively lie within the area covered by the graph. Otherwise, one has to sample a value y and compute the $\exp(\cdot)$ function to determine whether a value gets rejected or accepted [Oder et al., 2014]. The biggest disadvantage of the Ziggurat algorithm seems to be the necessity to perform rejection sampling (although infrequently). However, Buchmann et al. [2013] show that the performance in software is good compared to other algorithms and also on microcontrollers [Oder et al., 2014] the performance impact of rejections is acceptable. As of yet, there are no published results for hardware implementations.

Rejections can be avoided completely by using table based samplers. One option is the Knuth-Yao algorithm [Dwarakanath and Galbraith, 2014] which constructs a binary tree from the probability matrix and then a random walk is used to sample an element. The probability matrix consists of the binary expansion of the probabilities of all $x \in [0, \tau\sigma]$ ignoring leading zero digits. The matrix determines a rooted binary tree with internal nodes that always have two successors, as well as terminal leaves. The leaves are labelled with the value that is returned if this leaf is reached during the random walk through the tree. The number of leaves at level n is equal to the number of 1s in column n of the probability matrix (starting with column 0). The row in which a 1 appears is used as a label for one of the leaves. All remaining nodes become internal nodes with two successors that get labelled the same way. An implementation of the Knuth-Yao algorithm on reconfigurable hardware for small standard deviations is given by Roy et al. [2013b] (see [Roy et al., 2014a] for an extended version) and for microcontrollers by Oder et al. [2014] and Clercq et al. [2014].

Another rejection-less method to sample from a Gaussian distribution is the cumulative distribution table (CDT) by Peikert [2010]. For this method, a table of cumulative probabilities $p_z = \Pr(x \leqslant z : x \leftarrow D_\sigma)$ are computed for integers $z \in [-\tau\sigma, \ldots, \tau\sigma]$ with a precision of λ bits. For a uniformly random value x chosen from the interval $[0, 1)$, the integer $y \in \mathbb{Z}$ is then returned for which it holds that $p_{z-1} \leq x < p_z$. The comparisons can be performed efficiently without using floating point numbers. The CDT approach is compared to other software samplers by Buchmann et al. [2013] and is also used in the software implementation of BLISS [Ducas et al., 2013]. The performance of the sampler and the whole BLISS scheme is very good and supported by optimisations, like the usage of a set of guide tables to narrow the search radius of the binary search to find the x for which it holds that $p_{z-1} \leq x < p_z$. The disadvantage of the CDT approach is clearly large tables which are acceptable in software, but too expensive for a

hardware implementation. This problem is addressed by Pöppelmann et al. [2014], where an optimised floating point representation and Kullback-Leibler divergence is used to further reduce the table size. The most significant improvement is an application of the Gaussian convolution lemma which states that, under some smoothness condition for $x_1, x_2 \leftarrow D_{\mathbb{Z},\sigma'}$, the value $x = x_1 + k^2 x_2$ is distributed according to $D_{\sqrt{\sigma'^2 + k\sigma'^2}}$. Thus the size of the pre-computed table is massively reduced, as instead of a sampler for $\sigma \approx 215$, two samples from $\sigma' \approx 19.3$ are needed for $k = 11$. Also, the impact on speed is not too high, as the guide tables further reduce the number of required comparisons due to the smaller σ'. Compared with an implementation of the Bernoulli sampling, the CDT requires roughly half of the device resources for comparable throughput. It should further be noted that the usage of the convolution lemma is not restricted to the CDT sampler. The evaluation of the impact of the convolution for others samplers is currently not available, but is investigated in Chapter 3. The performance of the CDT sampler, as well as the Bernoulli sampler, are investigated by Boorghany et al. [2015] and Boorghany and Jalili [2014a] in implementations of GLP and BLISS as identification protocols on constrained devices.

2.7 Side-Channel Analysis

Although side-channel analysis (SCA) is not the main focus of this thesis, it is one of the motivations, and hence contributions, of the research in Chapter 3. Therefore, SCA will now be introduced.

A number of side-channel attacks are feasible for an implementation of a cryptographic algorithm. For a cryptographic hardware and software module, it is often useful to carry out a systematic review of the attacks vectors which are exposed to a potential adversary. These attack vectors are described by Anderson et al. [2006], and are divided into the following classes.

2.7.1 Invasive Attacks

An invasive attack involves de-packaging, which gains direct access to the internal components of a cryptographic module or device. A typical example of this is where an attacker may remove the passivation layer for microprobing, with a needle placed on a data bus the attacker is able to see the data transfer during a cryptographic operation. To counteract this, tamper resistant or responsive mechanisms are usually implemented in hardware. An example of this, for some cryptographic modules of higher security, is to zero all memories when tampering is detected.

2.7.2 Semi-invasive Attacks

The semi-invasive attack, developed first by Skorobogatov and Anderson [2002], involves access to the device but without interfering with the passivation layer or any electrical contact other than with the authorised surface. An example of this is in a fault attack [Skorobogatov and Anderson, 2002], where the attacker can use a laser beam to ionise a device, which can change some of the device's memories and therefore change its output.

2.7.3 Non-Invasive Attacks

A non-invasive attack follows close observation or manipulation of the operations of the device. The attack only uses information available externally, such as outputs which can sometimes leak information. A typical example of this is timing analysis [Kocher, 1996], which observes the time taken by the device to execute its operations, which can then be correlated to a particular operation and to secret information. Such an attack is totally undetectable, and is usually lower in cost compared to invasive and semi-invasive attacks. A susceptible device to this type

of attack is a smart card, where timing analysis can be used at a low cost, even at a large scale, to find secret information.

2.8 A Discrete Gaussian Testing Suite

This section describes GLITCH, a discrete Gaussian sampler testing suite for LBC. An incorrectly operating sampler, for example due to hardware or software errors, has the potential to leak secret-key information and could thus be a potential attack vector for an adversary. Moreover, statistical test suites are already common for use in pseudo-random number generators (PRNGs), and as lattice-based cryptography becomes more prevalent, it is important to develop a method to test the correctness and randomness for discrete Gaussian sampler designs. Additionally, due to the theoretical requirements for the discrete Gaussian distribution within lattice-based cryptography, certain statistical tests for distribution correctness become unsuitable, therefore a number of tests are surveyed. The final GLITCH test suite provides 11 adaptable statistical analysis tests that assess the exactness of a discrete Gaussian sampler, and which can be used to verify any software or hardware sampler design.

For LBC, there has yet been a proposal for testing the outputs of discrete Gaussian samplers, that is, if the samplers are *actually* producing the distribution required for specific values for (σ, τ, λ). The specifications for the discrete Gaussian sampling within lattice-based cryptography are very precise. The statistical distance between the theoretical discrete Gaussian distribution and the one observed in practice should be overwhelmingly small [Peikert, 2010], usually at least as small as $2^{-\lambda}$ for $\lambda \in \{64, \ldots, 128\}$. Providing guidelines to test implementations of discrete Gaussian samplers is therefore necessary for real-world applications in order to prevent attacks exploiting biased samplers, which can even occur via operational errors or bugs within sampler software or hardware designs. Thus, an

erroneously operating sampler could affect the target security level of the overall lattice-based cryptoscheme.

Additionally, the test suite is applicable for lattice-based cryptoschemes whose outputs are also distributed via the discrete Gaussian distribution, such as lattice-based encryption schemes [Lindner and Peikert, 2011, Lyubashevsky et al., 2013a] and digital signatures [Gentry et al., 2008, Ducas et al., 2013]. Indeed, a deviation from the target discrete Gaussian distribution for a lattice-based encryption or signature scheme could lead to a potential attack by an adversary.

2.8.1 Introduction to Statistical Testing in Lattice-Based Cryptography

Statistical testing is used to estimate the likelihood of a hypothesis given a set of data. For example, in cryptanalysis, statistical testing is commonly used to detect non-randomness in data, that is to distinguish the output of a PRNG from a truly random bitstream or to find the correctly decrypted message. The need for random and pseudorandom numbers arises in many cryptographic applications. For example, common cryptosystems employ keys that must be generated in a random fashion. Many cryptographic protocols also require random or pseudorandom inputs at various points, for example, for auxiliary quantities used in generating digital signatures, or for generating challenges in authentication protocols.

Moreover, the inclusion of statistical tests is paramount when implementing cryptography in practice. For example, to test a PRNG for cryptographically adequate randomness, the test suites DIEHARD [Marsaglia, 1985, 1993, 1996] and NIST SP 800-22 Rev. 1a [Bassham III et al., 2010] were made to check for insecure randomness, that is, to test a PRNG for weaknesses which an adversary could exploit.

To exploit or attack a PRNG, an algorithm could determine the deviation of its output from that of a truly uniformly random deviation. This is especially important for the discrete Gaussian distribution within lattice-based cryptography, since these values hide secret information. Normality tests can be used to determine if, and how well, a data set follows the required normally structured distribution. More specifically, statistical hypothesis testing is used, which under the null hypothesis (H_0), states that the data is normally distributed. The alternative hypothesis (H_a), states that the data is not normally distributed. All of the methods proposed for testing the correctness of a discrete Gaussian sampler design only require an input of histogram values output from the sampler.

For the test suite, two normality tests are adopted, each using the same statistics of the discrete Gaussian samples, by producing two important (and somewhat distinct) results. Both also follow the same hypotheses; the null hypothesis that the sample data is normally distributed, and the alternative hypothesis that they are not normally distributed.

The first test considered is the Jarque-Bera [Jarque and Bera, 1980, Bera and Jarque, 1981, Jarque and Bera, 1987] goodness-of-fit test, which takes the skewness and kurtosis from the sample data, and matches it with the discrete Gaussian distribution. It tests the shape of the sampled distribution, rather than dealing with expected values, which makes the test significantly simpler than, say, a χ^2 test. Interestingly, if the sample data is normally distributed, the test statistic from the Jarque-Bera test asymptotically follows a χ^2 distribution with two degrees of freedom, which is then used in the hypothesis test.

The second test is the D'Agostino-Pearson K^2 omnibus test [D'Agostino et al., 1990], and is another goodness-of-fit test using the sample skewness and kurtosis. This test however is an omnibus test, which tests whether the explained deviation in the sample data is significantly greater than the overall unexplained deviation. The test also has the same asymptotic property as the Jarque-Bera test.

D'Agostino et al. [1990] analyse the asymptotic performances of more commonly used normality tests; those being the χ^2 test, Kolmogorov test [Kolmogorov, 1956], and the Shapiro-Wilk W-test [Shapiro and Wilk, 1965]. These are important results, since the sample sizes required are far beyond those used in typical applications, in say, medicine or econometrics. Additionally it is recommended not to use the χ^2 test and Kolmogorov test, due to their poor power[10] properties. That is, for a large sample size, the probability of making a Type II error (that is, incorrectly retaining a false null hypothesis) significantly increases. Furthermore, for sample sizes $N > 50$, D'Agostino et al. state the Shapiro-Wilk W-test is no longer available, and even with the test extended ($N \leq 2000$) [Royston, 1982], it still falls below the required sample size (for example, for parameters by Pöppelmann et al. [2014], $\#\text{bins} \in \{2874, 5748, 7464, 8874\}$). The final major test for normality is the Anderson-Darling test [Anderson and Darling, 1952, 1954]. However, the D'Agostino-Pearson K^2 omnibus test is preferred since the Anderson-Darling test is biased towards the tails of the distribution [Razali et al., 2011].

The overall final tests are graphical, the first simply plot the observed histogram data (in blue) versus that which is expected (in red). The second graphic is a quantile-quantile (QQ) plot. This test illustrates how strongly the histogram data follows a discrete Gaussian distribution, providing a QQ-plot and coefficient of determination (R^2). The QQ-plot is supplementary to the numerical assessment of normality and is a graphical method for comparing two probability distributions. In this case, these two probability distributions are the observed and expected quantiles of the discrete Gaussian distribution. This test is essentially the same as a probability-probability (PP) plot, wherein a data set is plotted against its target theoretical distribution. However, QQ-plots have the ability to arbitrarily

[10]The power of an hypothesis test is the probability that the test rejects the null hypothesis when the alternative hypothesis is true.

Table 2.8 Details of the GLITCH software test suite.

Test No.	Test Description	Test Formula
Test 1	Sample Mean ($\bar{x}$)	$\bar{x} = (\sum_{i=1}^{N} x_i h_i)/N$
	Standard Error of $\bar{x}$	$\text{SE}_{\bar{x}} = s/\sqrt{N}$
	Confidence Interval of $\bar{x}$	$\bar{x} \pm t_{\alpha/2}\text{SE}_{\bar{x}}$
	Accept Null Hypothesis?	Accept if $\lvert\mu\rvert \in \{0, \ldots, \bar{x} + t_{\alpha/2}\text{SE}_{\bar{x}}\}$
Test 2	Sample Standard Deviation (s)	$s = \sqrt{(\sum_{i=1}^{N}(x_i - \bar{\mu}_1)^2 h_i)/N}$
	Standard Error of s	$\text{SE}_s = s/\sqrt{2(N-1)}$
	Confidence Interval of s	$s \pm t_{\alpha/2}\text{SE}_s$
	Accept Null Hypothesis?	Accept if $\lvert\sigma\rvert \in \{0, \ldots, s + t_{\alpha/2}\text{SE}_s\}$
Test 3	Sample Tail-Cut ($\bar{\tau}$)	$\bar{\tau} = \max(x_i)/s$
Test 4	Sample Skewness (ω)	$\omega = m_3\sqrt{N(N-1)}/(N-2)$
	Standard Error of ω	$\text{SE}_\omega = \sqrt{\frac{6N(N-1)}{(N-2)(N+1)(N+3)}}$
Test 5	Sample Excess Kurtosis (κ)	$\kappa = (m_4/s^4) - 3$
	Standard Error of κ	$\text{SE}_\kappa = 2\text{SE}_\omega\sqrt{\frac{N^2-1}{(N-3)(N+5)}}$
Test 6	Sample Hyperskewness	$\omega_* = m_5/s^5$
Test 7	Sample Excess Hyperkurtosis	$\kappa_* = m_6/s^6$
Test 8	Jarque-Bera Test For Normality	$\text{JB} = (N/6)(\omega^2 + ((\kappa-3)^2)/4)$
	Accept Null Hypothesis?	Accept if $\text{JB} < \chi^2_\alpha$
Test 9	D'Agostino-Pearson Omnibus Test	$\text{K}^2 = Z_1(\omega)^2 + Z_2(\kappa)^2$
	Accept Null Hypothesis?	Accept if $\text{K}^2 < \chi^2_\alpha$
Test 10	Histogram Plot	-
Test 11	Coefficient of Determination	$R^2 = 1 - (\sum_{i=1} e_i^2 / \sum_{i=1}(y_i - \hat{y})^2)$

choose the precision (to equal that of λ, say 128-bits) as well as being easier to interpret in the case of large sample sizes, hence its inclusion over PP-plots.

The R^2 value complements this plot, analysing how well the linear reference line approximates the expected data points. The output $R^2 \in [0, 1]$ is a measure of the proportion of total variance of the outcomes, which is explained by the model. Therefore, the higher the R^2 value, the better the model fits the data.

2.8.2 The GLITCH Test Suite

The test suite is designed so that it takes, as input, a histogram of discrete Gaussian samples. This is seen as advantageous over an input of listed samples, as calculations are significantly simplified as well as offering a decrease in storage.

The tests were specifically chosen so that each parameter in the discrete Gaussian sampling stage is tested. The main parameters under test are the mean and standard deviation of the discrete Gaussian distribution (μ, σ), with additional tests added to check the tail-cut (τ), and the shape of the distribution via the statistics skewness and kurtosis. Additional tests are included to check the normality of the distribution. Precision is also adaptable and set to 128-bits as per most lattice-based cryptoschemes.

The GLITCH test suite is provided in Python and is made publicly available online[11]. Additionally, discrete Gaussian data sets are provided. Concise details for GLITCH are given in Table 2.8.

Tests (1-3): Testing for Normality

The first set of tests are to approximate the main statistical parameters μ and σ, producing values for sample mean $(\bar{x})$ and sample standard deviation (s). This is done by using adapted formulas for the first (m_1) and second (m_2) moments, taking as input a histogram of values (x_i, h_i), where $m_1 = \bar{x} = (\sum_{i=1}^{N} x_i h_i)/N$ corresponding to the sample mean, and $m_2 = s^2 = (\sum_{i=1}^{N}(x_i - \bar{x})^2 h_i)/N$ corresponding to the sample variance, for a sample size N. The subsequent moments are then $m_k = (\sum_{i=1}^{N}(x_i - \bar{x})^k h_i/N)/\sigma^k$, using sample standard deviation $s = \sqrt{m_2}$.

Next, the standard error (SE) is calculated for the sampling distribution. This statistic measures the reliability of a given sample's descriptive statistics with respect to the population's target values, that is, the mean and standard deviation. Additionally, the standard error is used in measuring the confidence in the sample mean and sample standard deviation. For this, a two-tail t-test is constructed, given the null hypothesis $\mu = 0$ (similarly for σ), with the alternate hypothesis that they are not equal. So, if the null hypothesis is accepted, it is concluded that a $100(1-\alpha)\%$ confidence interval (C.I.) is $\bar{x} \pm \epsilon_{\bar{x}}$ and $s \pm \epsilon_s$, where $\epsilon_{\bar{x}} = t_{\alpha/2}\mathrm{SE}_{\bar{x}}$

[11]GLITCH software test suite available at https://github.com/jameshoweee/glitch

and $\epsilon_s = t_{\alpha/2}\text{SE}_s$. Since the aim of these tests is for the highest confidence (99.9%), $t_{\alpha/2} = 3.29$.

Tests (4-7): Testing the Distribution's Shape

The next set of tests deal with statistical descriptors of the shape of the probability distribution.

The first descriptor is the skewness; which is a measure of symmetry of the probability distribution and is adapted from the third moment. The skewness for a normally shaped distribution, or any symmetric distribution, is zero. Moreover, a negative skewness implies the left-tail is long, relative to the right-tail, and a positive skewness implies a long right-tail, relative to the left-tail. The population skewness is simply m_3/s^3, however the sample skewness must be adapted to $\omega = m_3\sqrt{N(N-1)}/N - 2$ to account for bias [Joanes and Gill, 1998]. Also SE_ω is calculated, to show the relationship between the expected skewness and ω.

The forth moment is kurtosis; and describes the peakedness of a distribution. For a normally shaped distribution, the target sampled kurtosis is three, and is calculated as m_4/s^4. More commonly, the sampled excess kurtosis is used and is defined as $\kappa = (m_4/s^4) - 3$. A positive kurtosis indicates a peaked distribution, similarly a negative kurtosis indicates a flat distribution. It can also be seen, given an increase in kurtosis, that probability mass has moved from the shoulders of the distribution, to its centre and tails [Balanda and MacGillivray, 1988]. Similarly, SE_κ is calculated to show the relationship between the expected excess kurtosis and κ.

An appropriate test for these statistical descriptors would be a z-test, where confidence intervals could also be calculated for some confidence level α. However, under a null hypothesis of normality, z-tests tend to be easily rejected for larger samples ($N > 300$) taken from a not substantially different normal distribution [Kim, 2013].

Higher-order moments, specifically the fifth and sixth, are used in the last two tests on the distribution's shape. The first of these tests hyper-skewness $\omega_* = m_5/s^5$, which still measures symmetry but is more sensitive to extreme values [Hinton, 2014, p.97]. Likewise, the second of these tests is for excess hyper-kurtosis $\kappa_* = m_6/s^6$, which tests for peakedness with greater sensitivity towards more-than-expected weight in the tails [Hinton, 2014, p.100].

Tests (8-9): Normality Testing

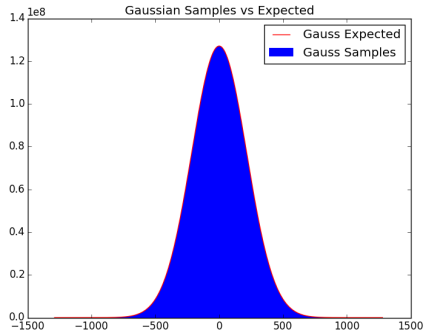
These tests calculate the test statistic and p-value for the two normality tests described in Section 2.8, these are the Jarque-Bera [Jarque and Bera, 1987] and D'Agostino-Pearson [D'Agostino et al., 1990] omnibus tests. The Jarque-Bera test statistic is calculated as $\text{JB} = (N/6)(\omega^2 + ((\kappa - 3)^2/4))$, where its p-value is taken from a χ^2 distribution with two degrees of freedom. The null hypothesis (of normality) is rejected if the test statistic is greater than the χ^2 p-value.

The D'Agostino-Pearson omnibus test is based on transformations of the sample skewness ($Z_1(\omega)$) and sample kurtosis ($Z_2(\kappa)$), which are combined to produce an omnibus test. This statistic detects deviations from normality due to either skewness or kurtosis and is defined as $\text{K}^2 = Z_1(\omega)^2 + Z_2(\kappa)^2$.

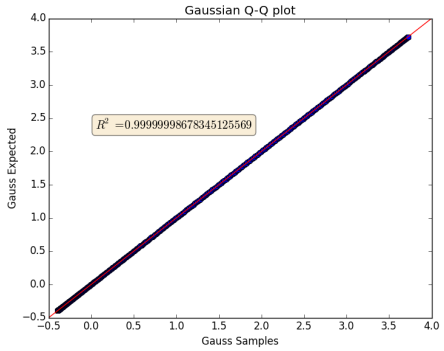
Tests (10-11): Illustrating Normality

D'Agostino et al. [1990] recommend, as well as test statistics for normality, graphical representations of normality are also provided. Hence, the final two tests are illustrative tests on the discrete Gaussian samples. The first graphic, shown in Figures 2.2a and 2.2c, plots the histogram of discrete Gaussian observed values (in blue) alongside the expected values (in red).

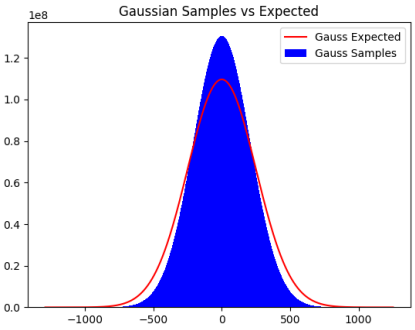
The second graphic is a quantile-quantile (QQ) plot, shown in Figures 2.2b and 2.2d. For this test, the calculated z-scores are plotted against the expected z-scores, where if the data is normally distributed, the result will be a straight



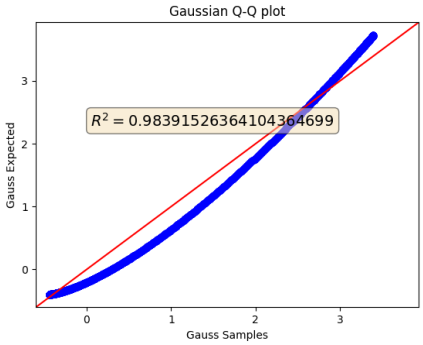
(a) Histogram of observed (blue) discrete Gaussian samples versus expected (red). The observed data matches the expected values.



(b) QQ-plot of the observed discrete Gaussian samples with the coefficient of determination (R^2) value. Both red and blue lines overlap meaning the observed data matches the expected values.



(c) Histogram of observed (blue) discrete Gaussian samples versus expected (red). The observed data does not match the expected values.



(d) QQ-plot of the observed discrete Gaussian samples with the coefficient of determination (R^2) value. Red and blue lines do not overlap meaning the observed data does not match the expected values.

Fig. 2.2 The graphical outputs of GLITCH tests 10 and 11.

diagonal line [Field, 2009, p.145-148]. A 45-degree reference line is plotted, which will overlap with the QQ-plot if the distribution follows the required distribution.

The coefficient of determination (R^2) value is calculated as $R^2 = 1-(SS_{\text{res}}/SS_{\text{tot}})$, where $SS_{\text{res}} = \sum_i (y_i - f_i)^2$ is the residual sum of squares and $SS_{\text{tot}} = \sum_i (y_i - \bar{x})^2$ is the total sum of squares, y_i is the observed data set and f_i is the expected values.

2.8.3 Results

Example results are provided in Appendix A. The results used are from a Bernoulli sampler. The first data set passes all tests, as shown in Table A.1. A second data set, used in Table A.2, is generated with an incorrect standard deviation and fails test 2, showing that GLITCH detects errors in discrete Gaussian samplers. Additionally, this failure is illustrated in the histogram plot and QQ-plot seen in Table A.2, with expected and observed values not matching. Data sets provided are of size 2^{36}. In general, the sample size for GLITCH should be large enough so that extreme values in the discrete Gaussian tails are likely to be filled.

The tests chosen are powerful and operate well on large sample sizes, with each analysing differing aspects within the discrete Gaussian distribution. Failure in any of these tests indicates a deviation from the target distribution, which is therefore evidence of an incorrectly performing discrete Gaussian sampler (or lattice-based cryptoscheme).

2.9 Conclusion

In this chapter, an introduction is given into the theory of lattices, lattice-based cryptography, and lattice-based PKE and DSSs, to provide context to the research presented in this thesis.

With respect to encryption schemes, there exists one predominately used scheme for ideal lattices (LPR by Lyubashevsky et al. [2013a]) and standard lattices (LP by Lindner and Peikert [2011]), and due to its better performance the ring-variant LPR is the most preferred. The hardware performance of LPR has been well-studied, with results (seen in Table 2.1) for balanced and low-area hardware designs. However, there could be more improvements on CPU-based implementations (with the only previous work by Göttert et al. [2012]) as well as the hardware performance of LP since there is no previous research in this area.

With respect to the digital signature schemes, the first lattice-based DSSs were GGH/NTRUSign, however these are known not to be secure hence they are excluded from formal description. The next class of schemes, hash-and-sign signatures (such as GPV signatures), once held promise for practical instantiations but have recently departed from inclusion in future research. This is mainly due to the signatures showing almost complete infeasibility for applications on constrained devices, a feature that is clearly essential given the diversity and scope of future technology.

With the addition of more favourable results shown by Fiat-Shamir signatures, it is no surprise that this has significantly shifted the research focus, giving predominance to the more modernised schemes in this area, such as BLISS. Due to the very exciting results shown in recent instantiations of BLISS on FPGAs by Pöppelmann et al. [2014] ($\approx$ 8000 signatures per second) and on microcontrollers by Oder et al. [2014] (28 signatures per second), in both instances outperforming RSA and ECC for comparable security levels, lattice-based digital signature schemes are now at a stage to be considered for real-world applications. However, no results currently exist for other ideal Fiat-Shamir lattice-based signatures which could compare with BLISS in hardware at 128-bit security level.

Currently there have been no hardware implementations of Ring-TESLA. Despite the larger arithmetic sizes, the scheme is fairly attractive in comparison

to GLP and BLISS, due to only requiring uniform random noise - therefore no need for a discrete Gaussian sampler - and signatures being uniformly distributed - meaning Huffman coding for shorter signatures is not required, which is used in BLISS, with both consuming significant hardware resources. Although, due to Ring-TESLA requiring two full polynomial multiplications, with GLP and BLISS only computing one, it is expected to have a higher latency. This is investigated in Chapter 5.

The research on statistical testing for discrete Gaussian outputs reapplies well established statistical testing techniques to LBC, taking into consideration the stringent requirements within lattices. This was completed by conducting a full survey on a number of different testing techniques. The first number of tests are for calculating the main discrete Gaussian parameters (μ, σ, τ) from the observed data, giving standard error and confidence intervals of each with the highest level of confidence (99.9%). The next set of tests verifies the shape of the distribution, that is whether there is any bias towards the positive or negative side of the distribution (that is, the x-axis) and whether the distribution has a bias towards the peak of the distribution (that is, the y-axis). For these tests and for the following tests on normality, the tests which allow for samples sizes large enough for lattice constraints were chosen. The last tests illustrate the difference between the observed distribution and the expected distribution's shape. Example results are provided in Appendix A which show results for a correctly and incorrectly performing sampler; in both cases the test suite deduces this.

From the background research conducted in this chapter, it is clear that the discrete Gaussian sampler is one of the most important components in practical LBC and is required within almost all lattice-based cryptoschemes. Considering current discrete Gaussian samplers are expensive in hardware, time consuming, and also susceptible to SCA attacks, it is important to design samplers which appease all of these issues. The next chapter proposes novel and efficient hardware designs for

the major techniques for discrete Gaussian sampling (as discussed in Section 2.6.2). The chapter also contributes recommendations for which sampling techniques operate most efficiently for lattice-based encryption and digital signature schemes.

CHAPTER 3

Practical Discrete Gaussian Samplers For Lattice-Based Cryptography

As mentioned in Chapter 2, the discrete Gaussian component is one of the most important modules within lattice-based cryptography (LBC). The aim of this chapter is to improve the performance of the discrete Gaussian sampler component.

In this chapter, novel hardware designs of discrete Gaussian samplers, that operate in independent time, are proposed. Discrete Gaussian samplers are a core building block in most, if not all, lattice-based cryptosystems, and thus optimised samplers are desirable both for high-speed and low area applications. This contribution contains the first comprehensive evaluation of discrete Gaussian samplers in hardware (targeting FPGA devices), proposing novel optimised discrete Gaussian sampler hardware architectures for the main sampling techniques, which include independent-time designs for each of the samplers and offering security against side-channel timing attacks. The proposed designs are compared against the state-of-the-art and recommendations for the selection of suitable sampling techniques as per the application and constraints at hand are given. This research appears in the publications by Khalid et al. [2016] and Howe et al. [2016a]. The

main publication for this chapter [Howe et al., 2016a] also includes the first hardware design of the discrete Ziggurat sampler, however since it is less practical than the other hardware designs, it has been omitted from this thesis.

3.1 Introduction

As LBC starts to become feasible for deployment in the real world, suitable countermeasures against *physical cryptanalysis*, including side-channel analysis (SCA), become a requirement. This is echoed by a recent call by NIST for optimised, quantum-resistant algorithms resistant to SCA attacks [NIST, 2016]. A core component to almost all LBC is the discrete Gaussian sampler, which is arguably the most vulnerable module within modern lattice-based constructions, due to its inherent structure. If the discrete Gaussian sampler is successfully attacked, it would render the whole cryptosystem broken. To date, there has been little research into the SCA-resilience of lattice-based cryptographic implementations (with the only preface by Roy et al. [2014a] and Reparaz et al. [2015a]).

As described in Chapter 3.2, the discrete Gaussian distribution is preferred to other probability distributions as it allows for more efficient and more secure implementations, with smaller output sizes such as ciphertexts or signatures[1]. Moreover, the samplers are used to add "noise" onto secret information, that would otherwise give away secret information in LWE and SIS cryptosystems via Gaussian elimination. The inherent structure of the distribution, however, is a potential attack vector for an adversary, making it susceptible to timing analysis attacks due to its normalised structure.

This chapter investigates and proposes *SCA-resilient hardware implementations* of recently proposed practical discrete Gaussian sampling techniques used within

[1] As opposed to using an alternative distribution as noise, say the uniform distribution.

lattice-based encryption and signature schemes[2], that is, the Bernoulli sampler [Ducas et al., 2013], the cumulative distribution table (CDT) sampler [Peikert, 2010], and the Knuth-Yao sampler [Knuth and Yao, 1976]. The major contributions of this research are as follows:

1. The first comprehensive evaluation of all aforementioned discrete Gaussian samplers, which are a core component of lattice-based encryption and signature schemes, with mathematical descriptions of all prominent sampling techniques, discussions of their inherent limitations and strengths, as well as a comparison with previous discrete Gaussian sampler hardware implementations.

2. Practical hardware FPGA designs of timing-attack resilient discrete Gaussian samplers are presented, for appropriate practical parameters, for throughput (operations per second), memory consumption, and resource count. Novel optimisation strategies are proposed for the hardware architectures of the sampling techniques, where the results at least compete with, and in many cases, significantly outperform, previous implementations.

3. The first constant-time Bernoulli sampler is proposed, as well as the first reported results on a time-independent Knuth-Yao sampler in hardware.

4. Based on the performance results, recommendations are given as to the most appropriate sampler to use for particular applications.

This chapter begins by reintroducing the discrete Gaussian distribution and the discrete Gaussian sampling techniques considered in this research. Section 3.2.2 then discusses SCA and timing analysis, as well as the design goals and objectives of this research. This discussion sets up the proposed research in Section

[2]With the Binomial sampler used by Alkim et al. [2016] currently only applicable for key-exchange protocols.

3.3 on the time-independent discrete Gaussian samplers, with the results given in Section 3.4, recommendations given in Section 3.5, and conclusions in Section 3.6.

3.2 Discrete Gaussian & Side Channel Analysis Introduction

The rationale for employing discrete Gaussian samplers (as opposed to another probability distribution) is that they allow more efficient, secure implementations, with smaller output sizes such as ciphertexts or signatures. Moreover, they are used to add "noise" onto values that would otherwise give away secret information in LWE and SIS constructed cryptosystems via Gaussian elimination.

For applications in this thesis, the following assumptions are made: the discrete Gaussian distribution is considered over the integers, with fixed parameters mean $\mu = 0$ and standard deviation σ, known in advanced and shown in Table 3.1. Due to the nature of the discrete Gaussian distribution, that being its infinitely long tails and infinitely high precision, it is essential to make practical compromises which do not, at the same time, hinder the integrity of the scheme. The two parameters needed here are (λ, τ), which represent the sampler's precision and tail-cut, respectively. Including the mean and standard deviation, the parameters make a tuple $(\mu, \sigma, \lambda, \tau)$ of discrete Gaussian parameters, which, in more detail, operate as follows:

- *The mean* (μ) is the centre of a normalised distribution. Almost always, within lattice-based cryptography, the mean is set to $\mu = 0$.

- *The standard deviation* (σ) is conventional when using the Gaussian distribution in general, and it controls the distribution's shape by quantifying the dispersion of data from the mean. Variance (s) can sometimes be used to describe the discrete Gaussian's shape, which is defined as $s = \sqrt{2\pi}\sigma$. In

general, the standard deviation's value depends on the modulus used within the LWE or SIS scheme. Where, for instance in LWE, should σ be too small the hardness assumption may become easier than expected, and if σ is too large the problem may not be as well-defined as required.

- *The precision parameter* (λ) governs the level of precision required for a specific implementation, exacting the statistical distance between the "perfect" theoretical discrete Gaussian distribution and the "practical" to be no greater than $2^{-\lambda}$, which corresponds directly to the security level of the scheme. However, Saarinen [2015, 2017] recommends that for a scheme with target security level λ-bit, precision need be no greater than $\lambda/2$, arguing that there exists *no algorithm* that can distinguish between a "perfect" sampler and one with statistical distance $2^{-\lambda/2}$.

- *The tail-cut parameter* (τ) administers how much of the less-heavy tails can be excluded in the practical implementation, for a given security level. That is, given a target security level of s-bits, the target distance from "perfect" need be no less than 2^{-s}. Therefore, instead of considering samples in the range $|x| \in \{0, \infty\}$, they can be considered in the range $|x| \in \{0, \sigma\tau\}$. Applying the aforestated reduction in precision also affects the tail-cut parameter, which is calculated as $\tau = \sqrt{\lambda \times 2\ln(2)}$.

A straightforward trick to save on execution time and table space is to only consider the distribution over $\mathbb{Z}^+$, proportional to $\rho_\sigma(x)$ for $\forall x > 0$, where for $x = 0$ $\rho_\sigma(0)$ must be halved since this will otherwise be counted twice. The entire distribution over $\mathbb{Z}$ can be easily recovered by incorporating a random sign bit at the end of sampling.

Further reductions in memory are possible by virtue of Peikert's convolution lemma [Micciancio and Peikert, 2013], which was adapted by Pöppelmann et al.

[2014] using the Kullback-Leibler divergence. Using the lemma implies a significantly smaller standard deviation can be used, which therein impacts on the scheme's area cost. Referring to Peikert [2010] and Pöppelmann et al. [2014] for the formal definitions of the *smoothing parameter* η and Kullback-Leibler divergence respectively, the adaption in Pöppelmann et al. [2014] states the following,

Lemma 1. *Let* $x_1 \leftarrow D_{\mathbb{Z},\sigma_1}$, $x_2 \leftarrow D_{k\mathbb{Z},\sigma_2}$ *for some positive real* σ_1, σ_2 *and let* $\sigma_3^{-2} = \sigma_1^{-2} + \sigma_2^{-2}$ *and* $\sigma^2 = \sigma_1^2 + \sigma_2^2$. *For any* $\epsilon \in (0, \frac{1}{2})$ *if* $\sigma_1 \geq \eta_\epsilon(\mathbb{Z})/\sqrt{2\pi}$ *and* $\sigma_3 \geq \eta_\epsilon(k\mathbb{Z})/\sqrt{2\pi}$, *then ("perfect") distribution* $\mathcal{P}$ *of* $x_1 + x_2$ *verifies*

$$D_{KL}(\mathcal{P}||D_{\mathbb{Z},\sigma}) \leq 2\left(1 - \left(\frac{1+\epsilon}{1-\epsilon}\right)^2\right)^2 \approx 32\epsilon^2.$$

Proof. The proof of this lemma is referred to in Pöppelmann et al. [2014]. □

Utilising Lemma 1 is ideal for minimising the standard deviation $\sigma_{\text{BLISS}} = 215$ in BLISS. The equations in Lemma 1 are satisfied by setting $k = 11$, which means $\sigma' = \sigma/\sqrt{1+k^2} \approx 19.47$, and by sampling twice $x'_1, x'_2 \leftarrow D_{\mathbb{Z},\sigma'}$ a value $x \leftarrow D_{\mathbb{Z},\sigma}$ can be built as $x = x'_1 + kx'_1$. The usage of the significantly smaller σ', as opposed to σ, means that sizes of precomputed tables within table-based samplers is reduced to around 11x smaller, with the main drawback being the requirement of sampling twice.

3.2.1 Sampling Techniques and Previous Work

Conceptually, the simplest algorithm to sample from a Gaussian distribution is rejection sampling or the *acceptance-rejection method* (introduced by Von Neumann [1951]). Using rejection sampling it is possible to sample from an arbitrary target distribution f when also given access to a bounded probability distribution g. In order to sample from f, a sample from g is accepted with probability $f(x)/(M \cdot g(x))$, where M is some positive real. When it is satisfied that $f(x) \leq M$

for all x, then the sampler produces the exact distribution f. In order to use rejection sampling to draw values from the positive half of a discrete Gaussian distribution, a uniformly random integer $x \in \{0, \ldots, \tau\sigma\}$ is chosen and then accepted with a probability proportional to $\rho_\sigma(x) = \exp(-x^2/2\sigma^2)$ (in this case $M = 1$). This is achieved by selecting a uniformly random value x, a random value u chosen from $[0, 1)$ and it is checked whether $u < \rho_\sigma(x)$ and thus whether a random u is under (acceptance) or above the curve (rejection) of the probability mass function of the Gaussian distribution. Sampling from the full range of the discrete Gaussian distribution finally involves rejection of an accepted $x = 0$ with probability $\frac{1}{2}$ and sampling of a sign bit. On average the method requires $2\tau/\sqrt{2\pi}$ trials until a sample is accepted.

Since this method requires many rejections in order to achieve an accepted value (an average of ≈ 8 trials) and the costly computation of the exponential function to high precision, the rejection sampling method is considered inefficient for practical hardware instantiations.

Parameters	$(\sigma, \quad \lambda, \ \tau \quad)$
LP Encryption [Lindner and Peikert, 2011]	$(3.33, 64, 9.42)$
BLISS Signatures [Ducas et al., 2013]	$(215, \ 64, 9.42)$

Table 3.1 Secure 128-bit discrete Gaussian parameters.

Bernoulli Sampling

The rejection sampling technique can be optimised in order to reduce the amount of rejections. This is achieved in the scheme by Ducas et al. [2013] (Algorithm 3.1, 3.2, and 3.3), where the authors make use of Bernoulli distributed variables $\mathcal{B}_c$, outputting one with probability $c \in [0, 1]$, and zero otherwise. Additionally, the number of rejections are reduced by using a distribution g called binary Gaussian distribution where x is proportional to $\rho_{\sigma_{\text{bin}}}(x) = 2^{-x^2}$ with parameter

$\sigma_{\text{bin}} = \sqrt{1/(2\ln 2)} \approx 0.849$ (see Algorithm 3.1) which can simply be constructed on-the-fly.

Algorithm 3.1 Sampling $D_{\mathbb{Z}^+,\sigma_{\text{bin}}}$

Ensure: An integer $x \in \mathbb{Z}^+$ according to $D^+_{\sigma_{\text{bin}}}$
  Generate a bit $b \leftarrow \mathcal{B}_{1/2}$
  if $b = 0$ **then** return 0
  for $i = 1$ to ∞ **do**
    draw random bits $b_1 \dots b_k$ for $k = 2i - 1$
    if $b_1 \dots b_{k-1} \neq 0 \dots 0$ **then** restart
    if $b_k = 0$ **then** return i
  end for

Algorithm 3.2 Sampling $D_{\mathbb{Z}^+,k\sigma_{\text{bin}}}$ for $k \in \mathbb{Z}$

Require: An integer $k \in \mathbb{Z}$ ($\sigma = k\sigma_{\text{bin}}$)
Ensure: An integer $z \in \mathbb{Z}^+$ according to D^+_σ
  sample $x \in \mathbb{Z}$ according to $D^+_{\sigma_{\text{bin}}}$
  sample $y \in \mathbb{Z}$ uniformly in $\{0, \dots, k-1\}$
  $z \leftarrow kx + y$
  sample $b \leftarrow \mathcal{B}_{\exp(-y(y+2kx)/(2\sigma^2))}$
  if $\neg b$ **then** restart
  return z

Algorithm 3.3 Sampling $D_{\mathbb{Z},k\sigma_{\text{bin}}}$ for $k \in \mathbb{Z}$

  Generate an integer $z \leftarrow D^+_{k\sigma_{\text{bin}}}$
  if $z = 0$ restart with probability $1/2$
  Generate a bit $b \leftarrow \mathcal{B}_{1/2}$ and return $(-1)^b z$

Using the binary Gaussian, an intermediate distribution $k \cdot D_{\mathbb{Z}^+,\sigma_{\text{bin}}} + \mathcal{U}(\{0 \dots k-1\})$ is constructed and the correct Gaussian distribution $D_{\mathbb{Z}^+,k\sigma_{\text{bin}}}$ is shaped using rejection sampling (see Algorithm 3.2), which reduces the number of rejections to ≈ 1.47 (compared to 8 for classical rejection sampling). In general, during rejection sampling, a value is accepted with probability $f(x)/(M \cdot g(x))$ which usually requires explicit computation of $f(x)$. However, it holds for an integer $x = \sum_{i=0}^{\ell-1} x_i 2^i$ with $x_i \in \{0, 1\}$ that:

$$\mathcal{B}_{\exp(-x/f)} = \mathcal{B}_{\exp\left(-\sum_i x_i 2^i/f\right)} = \mathcal{B}_{\prod_i \exp(-x_i 2^i/f)} = \bigwedge_{i \text{ s.t. } x_i=1} \mathcal{B}_{\exp(-2^i/f)},$$

meaning the final rejection step is performed independently, by evaluation of Bernoulli trials, which is efficient given precomputed biases c_i for every x_i. The method therefore only requires $\lambda \log_2(2.4\tau\sigma^2)$ bits of storage, and it is shown in Ducas et al. [2013] that Algorithm 3.2 requires less than $1.47 + 1/k$ trials. Algorithm 3.3 then corrects z to fit the final target distribution $D_{\mathbb{Z},k\sigma_{\text{bin}}}$.

One slight caveat to considering the Bernoulli technique is that the standard deviation must be a multiple of the binary Gaussian standard deviation; that is $\sigma = k\sigma_{\text{bin}}$ where $k = 4$ for LP, $k = 254$ for BLISS, and $\sigma_{\text{bin}} = \sqrt{1/2 \ln 2}$. Therefore, the standard deviations seen in Table 3.1 become $\sigma_{\text{LP}} = 3.39$, $\sigma'_{\text{BLISS}} = 19.53$, and $\sigma_{\text{BLISS}} = 215.73$.

Bernoulli sampling has been implemented without using the binary Gaussian distribution by Pöppelmann and Güneysu [2014] for the small standard deviation necessary for lattice-based public-key encryption. A complete implementation of the sampler can be found in the full version of the hardware implementation of the BLISS signature scheme by Pöppelmann et al. [2014], whose design is as follows: the binary Gaussian distribution $\rho_{\sigma_{\text{bin}}}(\{0, 1, \ldots, j\}) = \sum_{i=0}^{j} 2^{-i^2} = 1.1001000010000001\ldots$ is not constructed on-the-fly, instead they use two 64-bit shift registers to store the expansion precomputed up to a precision of 128 bits, which outputs an $x \in D_{\sigma_{\text{bin}}}$. Uniformly random values $y \in \{0, 1, \ldots, k-1\}$ are then sampled which may require rejection sampling (for instance, the probability of a rejection for $k = 254$ is $2/256$). The pipelined Bernoulli calculation stage takes a (y, x) tuple as input and computes $t = kx$ and outputs $z = t + y$ as well as $j = y(y + 2t)$.

While z is retained in a register, the Bernoulli evaluation module evaluates the Bernoulli distribution of $b \leftarrow \mathcal{B}_{\exp(j/2\sigma^2)}$. If and only if $b = 1$ the value z is passed to the output, and discarded otherwise. The evaluation of $\mathcal{B}_{\exp(x/f)}$ requires independent evaluations of Bernoulli variables. Sampling from $\mathcal{B}_c$ is done by just evaluating $s < c$ for a uniformly random $s \in [0, 1)$ and a precomputed c. The precomputed tables comprise of values $c_i = \exp(2i/f)$, for $0 \leq i \leq l$, $f = 2\sigma^2$ where $l = \lceil \log_2(\max(j)) \rceil$, which are implemented in distributed RAM. The $\mathcal{B}_{\exp(x/f)}$ module then searches for one-bit positions u in j and evaluates the Bernoulli variable $\mathcal{B}_{c_u}$.

They do this in a "lazy" manner so that the evaluation aborts when the first bit has been found that differs between a random s and c. The technique saves randomness and run-time but also incurs non-constant run-time. The chance of rejection is larger for the most significant bits, therefore they scan them first in order to abort as quickly as possible. The last step is to use the random one-bit to "sign" the samples and reject half of the samples where $z = 0$.

The Bernoulli sampler is suitable for hardware implementations as most operations work on single bits (mostly comparisons) only. However, due to the non-constant time behavior of rejection sampling, buffers were introduced in Pöppelmann et al. [2014] between each element to allow parallel execution and maximum utilisation of every component. To date, no time-independent implementations of the Bernoulli sampler have been reported. Section 3.3.1 outlines the proposed constant-time Bernoulli sampler design.

Cumulative Distribution Table (CDT) Sampling

The CDT technique requires a precomputed table of discrete Gaussian cumulative distribution function (CDF) values. As discussed in Section 3.2, the symmetry of the distribution is exploited to sample only from $\mathbb{Z}^+$, saving 50% of required table storage space. The first and last samples of the table are kept 0 and 1 respectively,

in keeping with the definition of a standard CDT and a total of $N = \tau \times \sigma$ samples ($0 = S[0] < S[1] < \cdots < S[N-3] = 1$) are required overall. Once the CDF values $S[\cdot]$ are computed, sampling happens as follows: a sample r is drawn uniformly, $r \in [0, 1)$, with λ bits of precision, where the desired sample, x, is found satisfying interval $S[x] \leq r < S[x+1]$, occurring with probability $\rho[x] = S[x+1] - S[x]$. Consequently, for the discrete Gaussian distribution, initial table values (close to $x = 0$) are more probable than values near the end.

Since the discrete Gaussian CDF is a sorted table in descending order, the binary search algorithm can be used to find the position of the target value, as shown in Algorithm 3.4. The search space, comprising initially of the entire CDT (N samples), is dichotomously exhausted in every iteration of the algorithm. Pointers *min* and *cur* point to the first and the middle of the search space, respectively, while *jmp* maintains the number of search space samples reduced by half. In every iteration of the while loop, r is compared to the middle value (*cur*) of search spaces, whose upper or lower half is discarded depending on the comparison result. Consequently, for a bounded interval, the number of comparisons required before a match is found does not exceed $\lceil \log_2(N) \rceil$. A uniformly sampled bit b is used to dictate the sign of result x.

The use of discrete Gaussian sampling based on large pre-computed CDTs was first proposed by Peikert [2010], and adapted by Ducas et al. [2013] for use in their signature scheme BLISS. Several reductions to the table size were suggested by Pöppelmann et al. [2014] for a reconfigurable hardware implementation of BLISS. Firstly, the most significant m bits of r could be hashed to reduce the search space according to some precomputed guide tables. Choosing $m = 8$ cuts the 2891 entries initially required for BLISS into 2^m intervals requiring guide tables holding the *min* and *cur* pointers. As a consequence, the complexity of the binary search is reduced from ~ 11.5 searches to ~ 1.5 searches on average. Secondly, the adoption of Lemma 1 on BLISS parameters significantly reduces

Algorithm 3.4 Sampling $D_{\mathbb{Z},\sigma}$ using the Cumulative Distribution Table (CDT) and binary searches

Require: Three Integers min, cur, and jmp
Discrete Gaussian CDT Samples: $N = \tau \times \sigma$, $0 = S[0] < S[1] < \cdots < S[N-3] = 1$
Sample a bit b uniformly in $\{0, 1\}$
Sample r uniformly in $\{0, \ldots, (2^{\lambda} - 1)\}$
Ensure: $min \leftarrow 0; cur \leftarrow (N/2); jmp \leftarrow cur;$
while $(jmp > 0)$ **do**
$cur \leftarrow min + jmp;$
if $(r \geq S[cur])$ **then**
$min \leftarrow cur;$
end if
$jmp \leftarrow jmp >> 1;$
end while
return $x = (-1)^b min$

the standard deviation for CDT, accompanied by a table size reduction by a factor of 11. Thirdly, a floating point representation of CDT samples with a variable mantissa size skips leading zero storage, reducing the table size further by a factor of 2.

These optimisations effectively reduce the precomputed CDT sizes, consequently reducing search space and improving design throughput. However, hashing divides the search intervals into irregular sizes, meaning the binary search within intervals has non-constant bounds, making it susceptible to timing analysis attacks. The only CDT-based discrete Gaussian sampler promising a constant-time throughput for small standard deviations relies on a array of N parallel comparators, of λ-bits each, comparing against the uniform number r [Pöppelmann and Güneysu, 2012]. Each comparator returns a binary answer, the first comparator x satisfying $S[x] \leq r < S[x+1]$ is the required result. This fully pipelined design results in a single cycle per sample throughput but the large number of parallel comparisons renders it inefficient in terms of hardware resources. Du and Bai [2015] optimise the hardware area by employing a piecewise comparison instead of

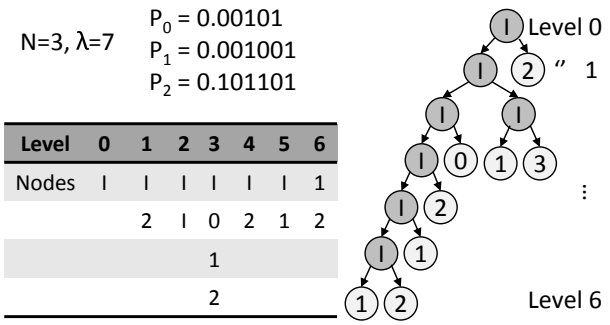
a full λ-bit comparison in the binary search algorithm. To compare any λ-bit table sample and a uniform value of the same size, comparing the first m bits can give a definite result of being greater or smaller with probability $(2^m - 1)/2^m$ (99.6% for $m = 8$) and increases for larger m. In the case of an equality, m is increased and a larger comparison must be carried out. This lazy comparison scheme not only reduces the need for large comparators, but also economises the need for uniform sample bits required per output. Hashing is used to further narrow down the binary search time resulting in an area-efficient and yet high average throughput performance. Section 3.3.2 outlines the proposed constant-time CDT sampler design.

Knuth-Yao Sampling

The Knuth-Yao sampling technique was proposed by Knuth and Yao [1976], which presents a tree based algorithm for sampling from non-uniform distributions by using a minimal number of input uniform bits, that is, close to the entropy of the probability distribution. Given a probability distribution represented by $p_0, p_1, \ldots, p_N$, while each sample is represented by a maximum of λ bits, the probability matrix P can be represented as a $N \times \lambda$ binary matrix (after adding leading/trailing zeros). A binary tree, called the discrete distribution generating (DDG) tree, can be constructed as an equivalent representation, having $\lambda-$levels, comprising two types of nodes: internal nodes (called I nodes) having two children each or terminal nodes (called T nodes) without children.

The DDG is constructed to ensure that the number of terminal nodes at the i^{th} level of the tree equals the number of non-zero bits in the i^{th} column of the probability matrix P. Each of these terminal nodes is marked by the row number of P. The sampling process is then a random walk starting from the tree root moving down, consuming a single uniform bit at each step, taking the left node of the next level in case of encountering a 0 and right node otherwise. Hitting

a terminal node outputs the integer label associated with it, culminating in the generation of a successful sample.



Level	0	1	2	3	4	5	6
Nodes	I	I	I	I	I	I	1
		2	I	0	2	1	2
				1			
				2			

Fig. 3.1 The Knuth-Yao based discrete Gaussian sampler for a toy example: the probability matrix and the DDG tree (right) with its table based representation.

Figure 3.1 illustrates a construction of a DDG tree, for a toy example. The tree levels equal the number of columns of P, while the number of terminal nodes match the non-zeros in P. Hence, a DDG tree can have at most $(N \times \lambda)$ terminal nodes and $(N \times \lambda) - 1$ internal vertices. Figure 3.1 (right) shows an equivalent tree and matrix representation of P, where each of the i^{th} columns of the table has no more than $(2 \times i) \leq (N \times \lambda)$ entries, since that is the upper bound on the number of terminal nodes. To run the sampling algorithm, the entire table does not need to be stored, instead P and the local information of one column suffices so that the information of the next column can be constructed from it at runtime.

Roy et al. [2013b] proposed a hardware friendly random walk for the discrete Gaussian sampler, based on this exploitation of i^{th} column/tree level information to interpret the next. At any level i of the tree traversal, let d denote the distance between the visited node and the right most node of that tree level. Therefore, at the DDG tree root (the 0^{th} level), $d = 0$. Depending on a 0 or a 1 being encountered, the distance becomes $2 \times d + 1$ or $2 \times d$, respectively, after consuming one uniform bit. Next, they exploit the fact that in the construction of a DDG tree, all the I nodes are on the left and all the T nodes (that are non-zero and contribute to the Hamming weight of that particular M column) are on the right.

Hence, subtracting the Hamming weight of the current column from the distance d at the i^{th} level of the tree will show if the current node is an I node or a T node, based on whether the result is positive or negative respectively.

Algorithm 3.5 Sampling $D_{\mathbb{Z},\sigma}$ using the Knuth-Yao technique

Require: Three Integers d, hit and ctr;
1: Discrete samples of Gaussian distribution as matrix P with $N \times \lambda$ dimension and $N = \tau \times \sigma$;
2: Sample bits uniformly in $\{0, 1\}$, store in array r;
3: Column-wise Hamming distance of P, i.e., $h\_dist[j] = \sum_{i=0}^{N} P[i][j]$;
Ensure: $d \leftarrow 0; hit \leftarrow 0; ctr \leftarrow 0;$
4: **for** (int $col \leftarrow 0 : col < \lambda; col \leftarrow col + 1$) **do**
5: $d \leftarrow 2d + (!r[ctr++]) - h\_dist[col]$
6: **if** $(d < 0)$ **then**
7: **for** (int $row \leftarrow 0 : row < N; row \leftarrow row + 1$) **do**
8: $d \leftarrow d + P[row][col];$
9: **if** $(d == 0)$ **then**
10: $hit \leftarrow 1;$
11: **break;**
12: **end if**
13: **end for**
14: **end if**
15: **if** (hit) **then**
16: **break;**
17: **end if**
18: **end for**
19: **return** $(-1)^{r[ctr++]}.row$:

Algorithm 3.5 gives the procedural details of the Knuth-Yao sampling algorithm. Assertion of the `hit` signal indicates a successful sampling completion and `ctr` is the iteration integer over the random binary samples array `r`. `h_dist` is a λ-element vector holding the column-wise Hamming distances of the matrix `P`. The loop at line 4 iterates over the columns, starting from the most significant bit position, and checks if a terminal node is hit in that level of the DDG tree, which is done by checking the sign bit of `d`; each iteration of this loop consumes one uniformly sampled bit. In case a column is localised for a hit, the loop in line 7 iterates over all the rows in that column, until a `row` is localised where a `hit` is

found, terminating the two loops. No uniform random bits are consumed during these iterations. A signed bit is attached to the sample, based on the uniformly sampled bit.

Devroye [1986] shows that in general the required uniformly generated bits for a sample generation is at most two more than the entropy of the distribution. Hence, for the test case $\sigma_{\text{LP}} = 3.33$, the number of uniform bits required per sample is 5.3. The implementation, seen in Algorithm 3.5, requires minimal resources for storing the $N \times \lambda$ dimensional binary matrix `P`, a λ element `h_dist` vector of $\lceil \log_2(N) \rceil$ bits each, and several pointers for holding the `row`, `col`, and `d`, as well as a sampling termination signal `hit`.

Dwarakanath and Galbraith [2014] suggest a compression of the probability matrix, since most of the distribution values close to the tail have an increasingly dominant number of leading zeros. A block variant of the Knuth-Yao algorithm was proposed that divides consecutive probabilities into different blocks with roughly the same number of leading zeros resulting in a reasonable saving in the storage requirement of P. Roy et al. [2013b] exploited this block based compression for the Knuth-Yao sampler in an FPGA implementation with $\sigma_{\text{LP}} = 3.33$. Instead of keeping a `h_dist` vector for column Hamming weight, they iterate over the columns of the `P` matrix, bit by bit. Additionally, the iteration proceeds from bottom to top, consuming one cycle for each bit of the probability matrix. Since the discrete Gaussian values close to the centre are much more likely, working from top to bottom instead significantly accelerates the algorithm's average speed (as done in Algorithm 3.5). Consequently, the implementation Roy et al. [2013b], though very lightweight, requires on average 17 clock cycles to generate one discrete Gaussian sample. Subsequent work by Roy et al. [2014a] improved the speed up to 2.5 cycles per sample, by virtue of table hashing in multiple stages. Since these implementations were non-constant time in nature and susceptible to timing analysis based attacks, a random shuffle method is used to protect the discrete

Gaussian distributed polynomial against side-channel attacks, at the cost of additional FPGA slices. De Clercq et al. [2015] present a software implementation of ring-LWE encryption on a 32-bit ARM Cortex-M4F microcontroller using a Knuth-Yao based fast discrete Gaussian sampler, bettering all existing encryption implementations in software, where discrete Gaussian sampling requires an average 28.5 cycles per sample. Section 3.3.3 outlines the proposed independent-time Knuth-Yao sampler design.

3.2.2 Timing Analysis Of Discrete Gaussian Samplers

Side-channel analysis was introduced in Section 2.7, this section will further discuss timing analysis with respect to the proposed research in this chapter.

Timing attacks were proposed for the first time by Kocher [1996]. These attacks exploit the time difference between operations to gain information about the secret key. Exploitable timing differences can be caused by routines whose execution time depends on the secret data or by the difference in data access patterns caused by memory hierarchy. These attacks have been successfully applied against several cryptographic schemes, including lattice-based schemes (in particular NTRU by Silverman and Whyte [2007]). Thus in the context of LBC, it is crucial to implement schemes resistant against timing attacks.

The need for a discrete Gaussian sampler resistant against timing attacks has been called for by NIST [Moody, 2016, NIST, 2016], which require implementations of encryption and signature schemes (of which discrete Gaussian samplers are core components) explicitly resistant against side-channel attacks, at 128-bit classical security. As shown by Saarinen [2017], software implementations of discrete Gaussian samplers have been shown to leak information through the timing channel. Bruinderink et al. [2016], in particular, presented a timing-based side-channel attack, exploiting the cache patterns on the CDT sampler. Also

the majority of hardware designs and implementations reported so far, such as those by Roy et al. [2013b], Pöppelmann et al. [2014], Howe et al. [2016b], are susceptible to timing attacks.

Resistance against timing attacks is achieved when all the operations involving secret information are executed in a time which is independent from the secret data. Time independence can be defined in the following way:

Definition 1 (Time Independence)**.** *The property of* ***independent-time*** *is achieved when no information about the secret value(s) is leaked by the time required for its execution.*

This property can be achieved in several ways; the most common methods are ensuring **constant execution time** or **random shuffling** of the secret values.

The discrete Gaussian sampler proposed by Pöppelmann and Güneysu [2013], which targets constant execution time, implements a time-independent CDT sampler for encryption. The area overhead of that design is however too high to be practical. Roy et al. [2014a] also presented a discrete Gaussian sampler resistant against timing attacks. The design implements a fast Knuth-Yao based *non-constant time* discrete Gaussian sampler which generates a batch of samples, which is subsequently *shuffled* to disassociate the related timing information.

The primary objective of this chapter is to propose hardware designs of discrete Gaussian samplers which achieve timing independence as in Definition 1. However, time-independence is not the only desired feature; implementations of discrete Gaussian samplers should also ensure fast response time (for high throughput) and require minimal area occupation (for constrained devices). In this research therefore, the design goals are considered and the limitations of previous work are overcome by proposing *independent-time* and *practical* implementations of the three major discrete Gaussian sampling schemes currently used in lattice based cryptography. However, it should be noted that high throughput is considered

as a secondary objective and low area is considered a tertiary objective to be optimised, since the main objective is to ensure time-independence of the proposed designs.

3.3 Efficient Time-Independent Discrete Gaussian Samplers

The following section discusses the novel discrete Gaussian sampler architectures proposed in this research.

3.3.1 Time-Independent Bernoulli Sampling

Discrete Gaussian sampling using the Bernoulli technique is performed in constant-time when the table comparison is always completely evaluated. This means that a comparison would not be aborted if one mismatch between the randomly generated bit r and the table value c is found. However, if a rejection of the Bernoulli variable leads to the rejection of a complete sample, is $r \geq c$ an abort can be tolerated. For acceptance ($r < c$) instead, the full comparison has to be made. The dependency on the input value of x would in fact leak information. The leakage can be mitigated by evaluating the whole table in a pipelined fashion, so that no secret addresses are used (the whole table has to be read). This requires a large number of random bits, which are generated using Trivium x32 as a PRNG. The PRNG produces 32 random bits, sufficient for the whole table comparison, and they are stored in a register, ready to be used when $j = y(y + 2kx)$ is calculated. In the case of a rejected value, the evaluation can abort early since rejection does not leak any secret information.

Figure 3.2 illustrates the high-level architecture of the proposed constant-time Bernoulli sampler. The binary Gaussian module includes the reduced precision

$\lambda = 64$-bits, as proposed by Saarinen [2015, 2017], (instead of 128-bits) and uses a 64-bit shift register (implemented in a LUTs on the target FPGA device) to store the precomputed expansion. This operation, as well as the uniformly random value $y \in \{0, 1, \ldots, k-1\}$, are also improved by incorporating an unrolled x8 Trivium as a PRNG. Previous designs by Pöppelmann et al. [2014] required 12 bits of randomness for the binary Gaussian component and 8-bits of randomness for the uniform value k. The design proposed in this research combines an x8 unrolled Trivium with a reduced precision, meaning the computation can be completed in two clock cycles instead of ≈ 20 clock cycles using a single bit per clock PRNG. The values of j and z are then used in the evaluation, j is used to evaluate $b \leftarrow \mathcal{B}_{\exp(-j/2\sigma^2)}$ where if $b = 1$ the value z is accepted and output, otherwise it is rejected.

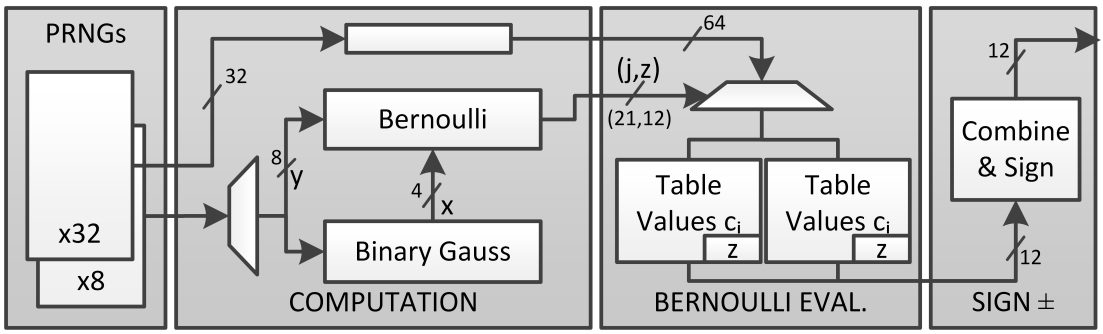


Fig. 3.2 High level architecture of the proposed constant-time Bernoulli sampler with variables and their bit lengths given, which uses a x8 and x32 Trivium as a PRNG.

For the table evaluation, since Gaussian convolutions (that is Lemma 1) are integrated; two table evaluation components are employed since this significantly decreases clock cycle count per sample. Moreover, each evaluation is able to compute a 128-bit comparison on every clock cycle, meaning per clock cycle two rows can be compared. Once the pre-calculation components are completed and the pipeline is full, input values to the tables will be ready after every clock cycle, meaning a discrete Gaussian sample takes exactly $\lceil \lceil \log_2 (\max(j)) \rceil / 2 \rceil + 2$ cycles,

which includes $\lceil\lceil\log_2(\max(j))\rceil/2\rceil$ for $x_1 \leftarrow D_{\sigma'}$, +1 more clock cycle for $x_2 \leftarrow D_{\sigma'}$, and +1 more clock cycle for them to be combined as $x = x_1 + 11x_2 \leftarrow D_\sigma$.

The sampler is designed generically such that it can operate for both standard deviations, $\sigma_{\text{LP}} = 3.39$ and $\sigma_{\text{BLISS}} = 215.73$, only by adapting the lookup table and the constants. Precomputed tables are stored in distributed RAM, of size $\lambda \log_2 2.4\tau\sigma^2$, which for $\sigma_{\text{BLISS}} = 215.73$ requires 784 bits and improves upon the work of Pöppelmann et al. [2014] by $\approx 48\%$ in table space, and for $\sigma_{\text{LP}} = 3.39$ requires 400 bits. The performance of this hardware design is presented in Section 3.4.1.

3.3.2 Time-Independent Cumulative Distribution Table Sampling

The binary search for a desired sample can result in an early termination, in the case where the equality comparison of a uniformly sampled r and the $S[cur]$ results in an exact match, although this early termination is only likely with a very small probability. Algorithm 3.4 for CDT sampling computes the inequality (instead of the 64-bit equality comparison), and hence avoids early termination bounding the algorithm for $\left\lfloor \lfloor\log_2(N)\rfloor, \lceil\log_2(N)\rceil \right\rfloor$ search iterations before generating a result. For CDT with N as a power of two, the algorithm performs inherently in constant-time. For all other N, the algorithm is tweaked to occasionally perform an extra iteration to ensure the algorithm complexity is fixed to $\lceil\log_2(N)\rceil$.

For the generation of discrete Gaussian samples with $\sigma_{\text{LP}} = 3.33$, the CDF table S consists of $N = 32$ ($\lceil\tau \times \sigma_{\text{LP}}\rceil$) entries, each consisting of $\lambda = 64$ bits of precision (refer to Table 3.1). Hence, a single ported BRAM, having a 5-bit address and 64-bit data ports suffices for the design. A 64× unrolled Trivium is used as a PRNG for the generation of the uniform samples r, where the initialisation of the module is handled externally at startup and then controlled by the binary search

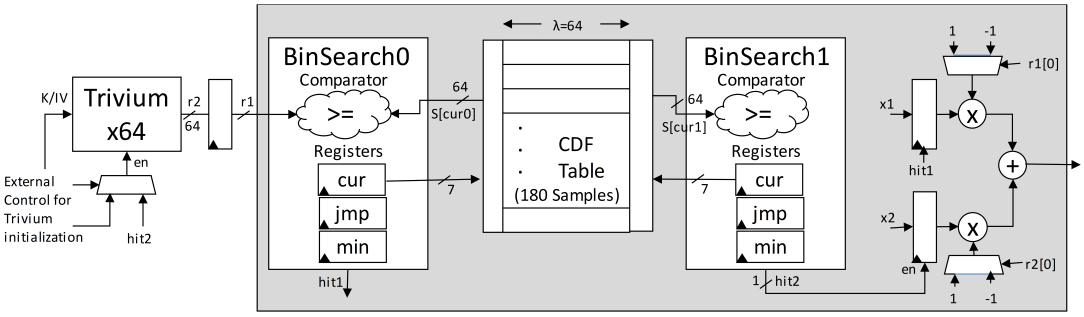


Fig. 3.3 The CDT discrete Gaussian sampler for $\sigma_{\text{BLISS}} = 215$, using two `BinSearch` state machines each accessing the CDF table for $\sigma'_{\text{BLISS}} = 19.47$.

state machine (referred to as `BinSearch`). The Trivium module is activated by the `enable` output pulse signal from `BinSearch`, so that the uniform samples are only generated when required. For $\sigma'_{\text{BLISS}} = 19.47$, the CDT table S is larger ($N = 184$ holding $N \times \lambda \approx 11$K bits).

According to Lemma 1, for each valid discrete Gaussian sample, two samples of $\sigma'_{\text{BLISS}} = 19.47$ are required. A single `BinSearch` state-machine instance would take $2 \times \lceil \log_2(184) \rceil = 16$ cycles for generation of a single valid sample. However, to improve throughput, two instances of a state-machine are used to parallelise two independent searches (seen in Figure 3.3). The two state machines `BinSearch0` and `BinSearch1` each get a 64-bit uniformly random number from the Trivium-based PRNGs, in two consecutive clock cycles.

The `BinSearch` state-machines initiate their processing from the `START` state, resetting three pointers, namely, *min*, *cur*, and *jmp*, with initial values, as given in Algorithm 3.4. They transition unconditionally to the `SEARCH` state in the next clock cycle that updates their three pointers as per the result from their 64-bit comparison operation. After exactly 8 cycles an appropriate x is found, and the state generates a single bit `hit` to signal this occurrence. This hit also activates the Trivium module to request a new uniformly random 64-bit value.

`BinSearch0` and `BinSearch1` then share the same dual ported BRAM, each port having 8-bit address and 64-bit data ports. The state machines work inde-

pendently to generate two random samples $x_1, x_2 \leftarrow D_{\sigma'}$ in 8 clock cycles; the two samples are buffered on their respective *hit* signals into registers. The samples are then combined as $x = x_1 + 11x_2$ where a sign bit is also attached. The performance results of this hardware design are presented in Section 3.4.3.

3.3.3 Time-Independent Knuth-Yao Sampling

Inherently, the Knuth-Yao sampler operates in non-constant time. Considering the column and row localisation phases take one clock cycle to jump from one column/row to the other, successful sampling requires at least 2 cycles if the 0^{th} row and 0^{th} column are being localised for a hit, with the maximum being $(\lambda \times N)$ when the last row and last column get a hit. The average response time is evaluated to be ≈ 10.6 cycles since the column and row localisation each require ≈ 5.3 cycles.

Figure 3.4 gives an architectural description of the proposed Knuth-Yao sampler in hardware, given a probability matrix P. Trivium is used as a PRNG for the generation of the uniformly sampled bit `r` that is inverted to get `r_n`. The initialisation of Trivium is handled externally at startup, which afterwards is controlled via the state-machine. An `en` signal is used to activate the PRNG to produce randomness only when required. The state-machine initiates its processing from the `START` state, resetting output registers; `row`, `col`, `d`, `en` and `hit` with initial values, as given in Algorithm 3.5. It transitions unconditionally to the `SEARCH` state that updates the output registers as per the state of the d pointer. As long as d is a non-zero positive number, the state-machine keeps changing columns, consuming random bits and updating d accordingly. If `d` is a negative number, then the column for a hit is already localised and the row search starts without changing column values. Since no new random bits are required, `en` is kept low and `d` is updated by adding the `P` matrix values until `d` reduces to 0.

Assertion of a `hit` ends the sampling process, where the row number is assigned a sign bit based on a uniformly sampled bit before completing a successful discrete Gaussian sample generation.

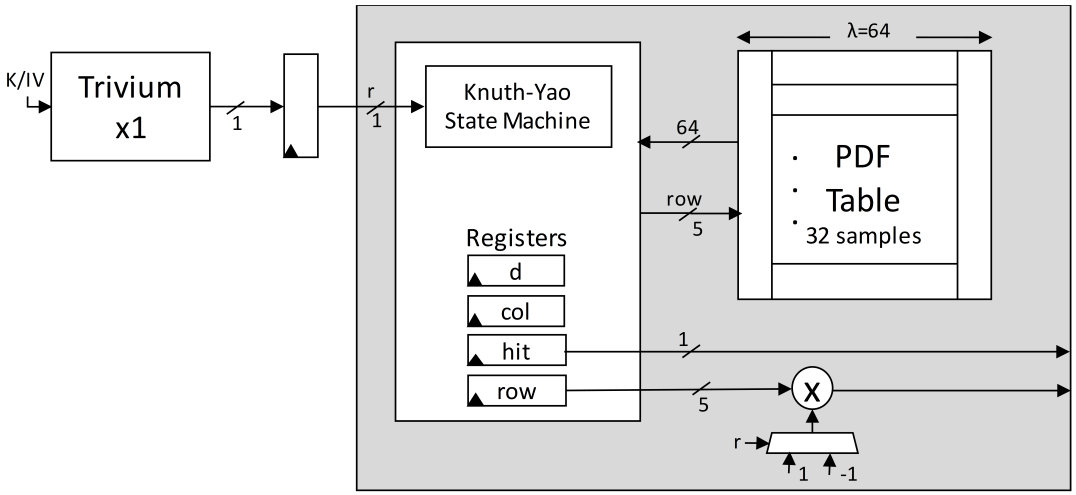


Fig. 3.4 A Knuth-Yao based discrete Gaussian sampler for $\sigma_{\text{LP}} = 3.33$

The transpose of the matrix P is stored in distributed ROM (or in a BRAM) in the proposed hardware design. When using a BRAM, storing a mere 64×32-bit P matrix is excessive. For better resource utilisation, the h_dist vector of Hamming distances is also stored in the same BRAM. Additionally, hashing is employed to boost the throughput, since the response time is nevertheless non-constant and the same BRAM holding the P matrix can also store the hash tables. To consider 8 bits of uniform random numbers at a time, a hash table with 256 entries is easily accommodated in BRAM, for the case where $\sigma_{\text{LP}} = 3.33$, 249 out of these 256 entries result in a hit (97.26%). Otherwise, the hash table entries keep the `d` value to be loaded in the `KYSearch` state-machine directly and scanning of the first 8 columns is skipped, improving throughput.

To make the sampler run in independent-time, operations can be made constant-time, that is the largest possible cycle count ($\lambda \times N$) must be ensured before a discrete Gaussian sample is generated as an output. For $\sigma_{\text{LP}} = 3.33$, it turns out to be too slow to be practical (2000 cycles per valid sample generation, and worse

for signature parameters). Hence, to achieve timing independence, the discrete Gaussian samples are instead shuffled after the generation of a complete block. Considering a block size of n samples, if n_1 are generated by hashing then the remaining $n_2 = n - n_1$ are generated by the `KYSearch` state-machine. The samples are stored in another BRAM such that n_1 samples are accommodated at the top of the BRAM with the remaining at the bottom.

A simplistic shuffling algorithm is implemented, that is the Fisher and Yates [1948] shuffle (also used by Roy et al. [2014a]), which goes over each of the n_2 samples, swapping these within randomly generated locations $[0, n_1 - 1]$. A simple `shuffler` state-machine enables each swap in two clock cycles considering a dual-ported BRAM. Due to the small percentage of n_2 samples in BRAM (2.7%), shuffling does not significantly exacerbate the sample generation throughput. However, the state-machine and the BRAM require extra FPGA resources.

The Knuth-Yao implementation relates the number of random bits required (and consequently the running time of the algorithm) to the distribution entropy (or standard deviation). For $\sigma_{\text{BLISS}} = 215$, even after using Lemma 1, the running time will be more than 20 clock cycles per sample, which is far too slow for practical purposes. Hashing would also require much larger tables to be effective. Consequently, no further work is undertaken on the Knuth-Yao sampler for signature parameters. The performance results of this hardware design are presented in Section 3.4.2.

3.4 Comparison and Results of Sampling Hardware Architectures

In this section the performance results are described and compared against the same implementations for encryption (for $\sigma_{\text{LP}} = 3.33$) and signatures (for $\sigma_{\text{BLISS}} = 215$),

using the parameters stated in Table 3.1. The designs are implemented on either the Spartan-6 LX25-3, Virtex-5 LX30, or Virtex-6 LX75 FPGA devices to compare with previous research, where the results obtained are all post place-and-route (PAR) using Xilinx ISE 14.7. Throughput and throughput per area have been evaluated for all schemes in terms of sampling operations per second (Ops/s) and sampling operations per second per slice of FPGA (Ops/s/S), respectively, in Tables 3.2-3.4.

Significantly more random bits are required to guarantee constant computation time. Random bits are produced by Trivium x8 and x32 designs. The resource needed for these, when compared with the standard Trivium x1, are rather negligible: 28 additional LUTs, 63 additional flip-flops, and 15 additional slices for Trivium x8 and 26 additional LUTs, 147 additional flip-flops, and 21 additional slices for Trivium x32.

3.4.1 Bernoulli Results

Table 3.2 shows the performance of the constant-time Bernoulli sampler in hardware. The results are compared to other implementations which target the same lattice-based cryptosystem but are not protected against timing attacks.

To achieve constant computation time, the proposed Bernoulli samplers need to perform many more comparisons than the designs previously proposed in the literature [Pöppelmann et al., 2014, Howe et al., 2016b]. For this reason, the number of flip-flops in this design is larger than others. However, these comparisons are done in an optimal way by performing them in a single clock cycle. Additionally, incorporating x8 and x32 unrolled Trivium components (see Figure 3.2) has significantly improved the overall design and alleviated the need for excess data buffers, in comparison to previous work. The halving of the precision parameter also contributed to the reduction of the LUT and slice usage.

Table 3.2 Post-place and route results of the Bernoulli sampler for encryption ($\sigma_{\text{LP}} = 3.39$) and signatures ($\sigma_{\text{BLISS}} = 215.73$), in comparison to those targeting the same discrete Gaussian parameters with non-constant operating time.

Op.	**Implementation**	**Device**	λ	**LUT/FF /Slice**	**BRAM /DSP**	**Freq. (MHz)**	**Clock Cycles**	**Rand. Bits**	**Ops/s ($\times 10^6$)**	**Ops/s/S ($\times 10^6$/S)**	**Time Ind.**
$\sigma_{\text{LP}} = 3.33$	This work	XC6SLX25-3	64	679/1580/279	0/0	133	7	85	19	0.06	✓
				516/1475/201	2/0	167	7	85	23.86	0.12	✓
	Howe et al. [2016b]	XC6SLX25-3	64	846/934/297	0/0	129	≈ 12	≈ 26	10.75	0.03	✗
	Pöppelmann and Güneysu [2014]	XC6SLX9-2	80	132/40/37	0/0	136	≈ 144	≈ 96	0.944	0.02	✗
$\sigma_{\text{BLISS}} = 215.73$	This work	XC6SLX25-3	64	1001/1842/356	0/0	139	13	85	10.69	0.03	✓
				571/1480/167	3/0	171	13	85	13.15	0.08	✓
	Pöppelmann et al. [2014]	XC6SLX25-3	128	1231/1134/452	0/1	137	≈ 17.95	≈ 33	7.63	0.01	✗

Table 3.3 Post-place and route results of the Knuth-Yao sampler for encryption ($\sigma_{\text{LP}} = 3.33$), in comparison to existing work with same discrete Gaussian parameters.

Op.	Implementation	Device	λ	LUT/FF /Slice	BRAM /DSP	Freq. (MHz)	Clock Cycles	Rand. Bits	Ops/s ($\times 10^6$)	Ops/s/S ($\times 10^6$/S)	Time Ind.
$\sigma_{\text{LP}} = 3.33$	Roy et al. [2013b]	5VLX30	90	140/66/47	0/0	333	≈17	≈5.3	19.61	0.42	✗
				149/69/53	0/0	303	≈16	≈5.3	18.94	0.36	✗
	Roy et al. [2014a]	5VLX30-3	90	101/81/38	0/0	344	≈17	≈5.3	20.28	0.53	✗
				105/60/32	0/0	400	≈17	≈5.3	23.53	0.74	✗
				102/48/30	0/0	384	≈17	≈5.3	22.62	0.75	✗
				118/48/35	0/0	333	≈13	≈5.3	133.33	3.81	✗
	This work	5VLX30-3	64	99/21/35	0/0	310	≈10	≈5.3	31.02	0.89	✗
				59/25/22	1/0	212	≈1.16	≈8.3	183.02	8.32	✗
				133/52/48	2/0	212	≈1.23	≈8.3	172.60	3.60	✓

The proposed Bernoulli sampler can generate at least 10 samples per second for BLISS signature parameters and 19 samples per second for LP encryption parameters, which as well as being the first Bernoulli sampler to operate in constant-time, also betters the speed of the previous hardware implementations by Pöppelmann et al. [2014]. In general it can be seen from Table 3.2 that overall, despite the increase in flip-flop consumption, the Bernoulli samplers for both standard deviations improve on previous designs.

3.4.2 Knuth-Yao Results

Table 3.3 compares the proposed Knuth-Yao samplers (where $\sigma_{\text{LP}} = 3.33$) with the state-of-the-art. Roy et al. [2013b] present a $\sigma_{\text{LP}} = 3.33$ discrete Gaussian sampler design requiring around 5.3 random bits and 17 clock cycles per sample. The slow throughput was primarily due to the bit-by-bit scanning of the probability matrix `P`, while introducing multiple bit scanning per clock after column localisation does slightly improve throughput, there is an additional resource cost. Additionally, Roy et al. [2014a] present various low latency variants of the same architecture, changing the width and height of the matrix `P`, each requiring 17 clock cycles per sample. They employ multi-stage table hashing to improve throughput up to ≈ 2.5 clock cycles per sample and subsequently a *shuffler* to make the sampler time-independent. However, the results for this hardware design, with a shuffler, are not provided by Roy et al. [2014a].

The first set of results for the proposed hardware designs is undertaken without BRAMs and hashing, in order to fairly compare to the existing work by Roy et al. [2014a] (without hashing). In the second set of results hashing is utilised, wherein BRAM is used to store the P matrix, the h_dist, and the hashing table. At the cost of one additional BRAM, the results of this design improves upon the throughput performance of the best hardware design by Roy et al. [2014a]

by more than a factor of 2. The final set of results is for the Knuth-Yao sampler which includes a shuffler for time independence; hence another BRAM is used for the generated discrete Gaussian samples, meaning the slice budget increases to accommodate the shuffler. As a consequence, the throughput decreases.

3.4.3 Cumulative Distribution Table Results

For the CDT sampler, FPGA implementations with and without BRAMs are proposed. For $\sigma_{\text{LP}} = 3.33$, a single port distributed ROM is used. The number of slices can be significantly reduced if BRAMs are utilised, as shown in Table 3.4, where one instance of `RAMB36` is used in a 64×32 configuration. However, in this configuration the maximum operable frequency is halved. The reduction in slices is higher for $\sigma_{\text{BLISS}} = 215$ due to the larger distribution table being shifted to BRAM. In this case, the critical path is not significantly increased.

Table 3.4 compares the proposed CDT samplers with the state-of-the-art. The only other constant-time CDT implementation for $\sigma_{\text{LP}} = 3.33$, proposed by Pöppelmann and Güneysu [2013], operates at a frequency of more than 4× lower. The slice count is also 5x larger, contributed primarily by as many parallel comparators as the S table words. Hence, despite the reported CDT implementation generating a single sample per cycle [Pöppelmann and Güneysu, 2013], the proposed design in this research proves to be a very lightweight and yet a constant-time alternative, outperforming it by a factor of around 5× in terms of Ops/s/S. The CDT sampler design by Du and Bai [2015] is lightweight in resource consumption and achieves good average throughput with a small number of average uniform random bits required per sample, by virtue of hashing and piecewise lazy comparison. However, the design by Du and Bai [2015] is not an independent time design.

Table 3.4 Post-place and route results of the Cumulative Distribution Table (CDT) sampler for encryption ($\sigma_{\text{LP}} = 3.33$) and signatures ($\sigma_{\text{BLISS}} = 215$), in comparison to existing results with same discrete Gaussian parameters.

Op.	**Implementation**	**Device**	λ	**LUT/FF /Slice**	**BRAM /DSP**	**Freq. (MHz)**	**Clock Cycles**	**Rand. Bits**	**Ops/s ($\times 10^6$)**	**Ops/s/S ($\times 10^6$/S)**	**Time Ind.**
$\sigma_{\text{LP}} = 3.33$	Pöppelmann & Güneysu [2013]	6VLX75T-2	80	863/6/231	0/0	61	1	85	61	0.26	✓
			100	1157/6/314	0/0	58	1	85	58	0.18	✓
	This work	6VLX75T-2	64	112/19/43	0/0	297	5	64	59.4	1.38	✓
				53/17/15	1/0	193	5	64	38.6	2.57	✓
	Du and Bai [2015]	5VLX30	90	43/33/17	1/0	259	≈2.28	≈9.44	113.6	6.68	✗
				85/65/39	1/0	256	≈1.14	≈9.44	224.6	5.76	✗
$\sigma_{\text{BLISS}} = 215$	Pöppelmann et al. [2014]	6SLX25-3	128	928/1121/299	1/0	129	≈7.5	≈21	17.2	0.06	✗
	This work	6SLX25-3	64	577/64/179	0/0	130	8	64	16.3	0.09	✓
				130/48/44	2/0	126	8	64	15.8	0.36	✓

For $\sigma_{\text{BLISS}} = 215$, the implementation by Pöppelmann et al. [2014] compared to this research operates in non-constant time, is costlier in terms of slice consumption, and slower in terms of throughput per slice. Since their reported design consumes BRAM, when compared to this implementation (with BRAM), their design remains $\approx 5\times$ inferior in terms of Ops/s/S. However it requires around $3\times$ less uniform random bits per sample, compared to the proposed constant-time designs.

3.5 Recommendations

Figure 3.5 plots the post-PAR results for CDT, Knuth-Yao (KY) and Bernoulli (Ber) samplers, both with and without the use of RAMs, targeted to the same FPGA Spartan-6 LX25-3 device. As stated at the beginning of this chapter, results for the discrete Ziggurat samplers are not included, due to their inefficiency.

For encryption, the RAM-free CDT (CDT_Enc) sampler surpasses all others in terms of an overall balanced performance with area, throughput, and with timing independence. If the use of additional BRAMs is considered, the Knuth-Yao time-independent implementation (KY_Enc_RAM) has the best overall performance in terms of low area and high throughput, whilst also requiring the lowest number of random bits per sample.

For signatures, the RAM-free CDT implementation (CDT_Sign) proves to be an overall winner, followed by the Bernoulli sampler (Ber_Sign), being around 2x more expensive in terms of slices. The CDT implementation with use of RAM (CDT_Sign_RAM) is also the preferred option, compared to its nearest competitor (Ber_Sign_RAM) it uses significantly less LUTs, FFs, and slices, and also requires less clock cycles per generated sample.

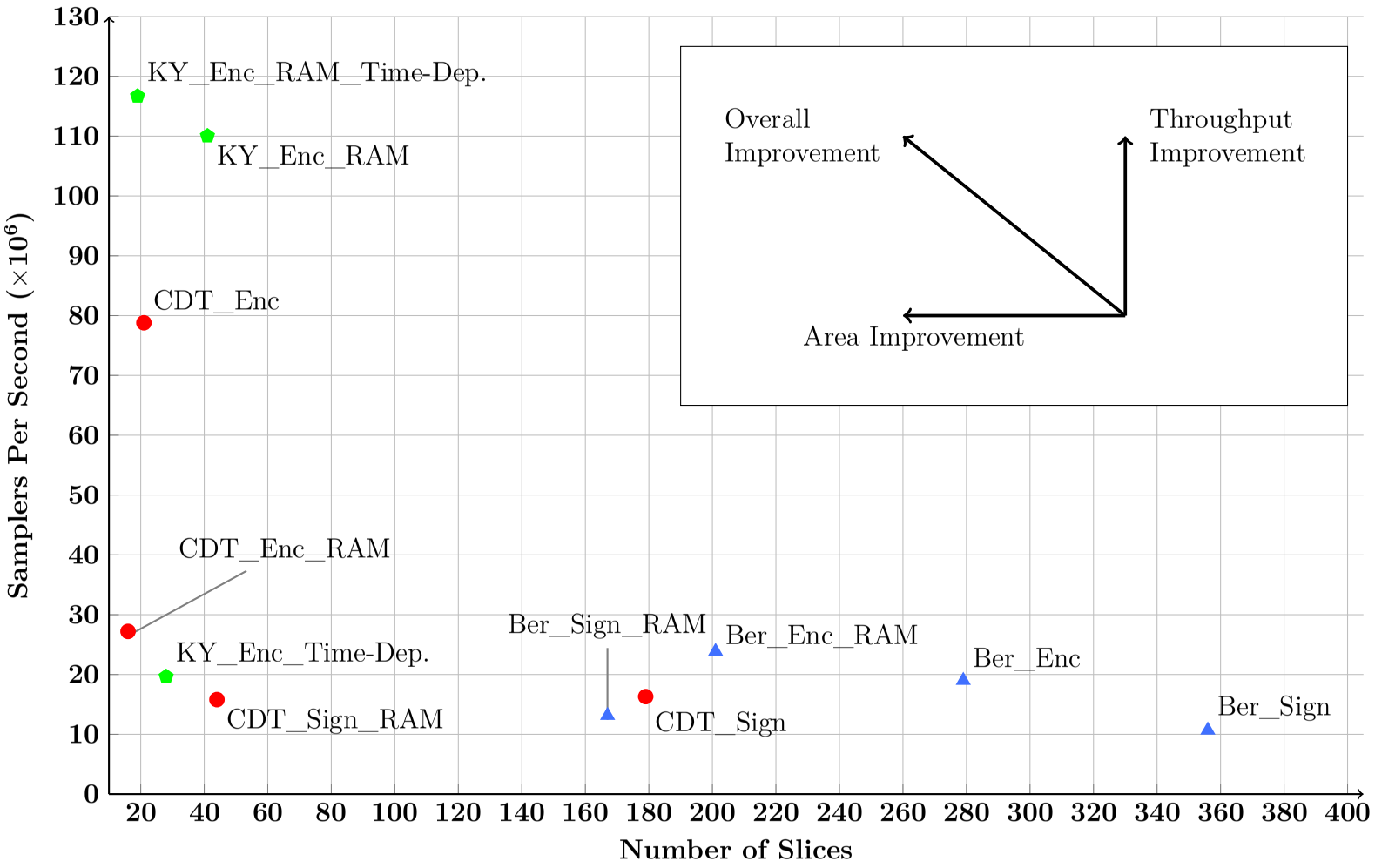


Fig. 3.5 Graphical performance results of the proposed discrete Gaussian samplers, on the Spartan-6 LX25-3 FPGA, with and without RAM use. All results are time-independent unless otherwise stated (Time-Dep.).

3.6 Conclusion

The research on sampling hardware designs provides a thorough investigation of all the practical discrete Gaussian samplers (CDT, Knuth-Yao, and Bernoulli) used in lattice-based cryptosystems. Novel time-independent hardware designs are presented, as well as their subsequent implementation results, which ensure resistance against timing attacks. In addition to timing attack resilience, the designs also focus high throughput with low-area. A survey of all discrete Gaussian sampling techniques as well as all FPGA-based designs reported to date are presented and analysed. In addition to resistance against timing attacks, the proposed hardware sampler designs clearly outperform most of the previously proposed sampler architectures. The research concludes by giving recommendations for the best performing sampling schemes, which should be considered when implementing any lattice-based cryptosystems, especially for public-key encryption (PKE) schemes or digital signature schemes (DSS), which require either high throughput, low area, or an overall balanced performance.

The recommendation for both encryption and digital signatures was the CDT sampler. In both cases, the CDT sampler outperforms the other proposed time-independent hardware designs of the Bernoulli and Knuth-Yao samplers, in terms of FPGA area consumption. For encryption, the CDT and Knuth-Yao samplers closely compete, and although the CDT sampler is in general the recommendation, for high throughput the Knuth-Yao sampler should be considered. For signatures, the CDT sampler outperforms the Bernoulli sampler, both in terms of FPGA area consumption and throughput.

With the improved assurances in samplers provided by the research in this chapter, these optimised hardware designs are now ready to be integrated into lattice-based cryptoschemes. Further research is now required into hardware

designs of cryptographic primitives, such as lattice-based PKE and DSSs. There has been previous research into both lattice-based PKE and DSSs, however with regards to hardware, all previous research focuses on ideal lattices. Additionally, there are security concerns and better performing cryptanalysis regarding the ideal lattice paradigm, such as the research by Ishiguro et al. [2014].

Currently, the standard lattice assumption has yet to be explored in hardware. Standard lattices are viable for hardware designs and provide more confidence in security. The next chapter focuses, for the first time, on the performance of standard lattice-based PKE in hardware. This hardware design also utilises the recommended discrete Gaussian sampler, the CDT sampler, proposed in this chapter.

CHAPTER 4

Lattice-Based Encryption Over Standard Lattices in Hardware

In Chapter 3, hardware optimised designs of discrete Gaussian samplers, a major component within lattice-based cryptography, are presented. This chapter furthers this research by considering an entire lattice-based cryptoscheme. This chapter presents the first hardware design of a lattice-based cryptoscheme based over standard lattices. The research in this chapter was presented in a publication by Howe et al. [2016b].

The overall design goal of the scheme is for a balance between hardware resource consumption and encryption speed. This also means the hardware design results can fairly compare with previous hardware designs of ring-LWE encryption. When compared to previous work for ideal lattice-based encryption hardware designs, the proposed standard-LWE encryption hardware designs perform significantly better than expected. The proposed hardware designs illustrate, for the first time, that standard lattice-based assumptions are applicable for real-world applications, especially for instances where security is paramount.

4.1 Introduction

As a cryptographic primitive, public-key encryption (PKE) is fundamental for use within security protocols, and is used in applications such as secure communications and message verification. As described in Chapter 2, PKE is a tuple of algorithms performing KeyGen(1^n), Enc_{pk}, Dec_{sk}, which uses the publicly available public-key (pk) and a secret-key (sk).

As also discussed in Chapter 2 (Section 2.4.1), previous hardware designs for lattice-based encryption schemes have all focused on the ideal lattice assumption, which are designed upon the encryption scheme by Lyubashevsky et al. [2013a] (LPR). The LPR scheme bases its hardness on the ring-LWE problem, with the scheme actually the ideal equivalent of the standard-LWE encrytion scheme by Lindner and Peikert [2011] (LP), which in terms of software and hardware designs has been significantly understudied.

Inherently, the major difference between standard and ideal lattices is the use of keys and the computations performed using these keys. Standard lattice keys are represented by (usually square) matrices and have computational complexity $\mathcal{O}(n^2)$, whereas the ideal lattice assumption allows for the first column of a (structured) matrix to represent the entire matrix over a specific polynomial ring. This means that a standard-LWE matrix-vector computation with large dimensions, is made equivalent in ring-LWE to a polynomial calculation, which has the potential of turning a quadratic calculation into a quasi-linear one (see further discussions in Section 4.2.2). Although the efficiencies made by the standard-to-ideal transition may significantly alleviate computational resources, ideal lattices may be more susceptible to attacks than standard, due to the use of structured matrices. For instance, the additional cyclic structure of ideal lattices is a potential exploit for an attacker. Indeed, recent research has begun to investigate

the security implications of weak instances and algebraic attacks of ring-LWE [Chen et al., 2015, Elias et al., 2015, Bernstein, 2014, Cramer et al., 2015] as well as improved sieving algorithms for finding the shortest vector in an ideal lattice [Schneider, 2013, Ishiguro et al., 2014, Bos et al., 2014, Becker and Laarhoven, 2015].

A Gauss sieving algorithm [Micciancio and Voulgaris, 2010] consists of a list of vectors in the lattice and a reduction algorithm, that eventually outputs the shortest nonzero vector that can be found in that list. These cryptanalytic algorithms on ideal lattices are also part of a shortest vector challenge (http://www.latticechallenge.org/) For example, the Gauss sieve algorithm of Ishiguro et al. [2014] reports a 600x speed-up when finding the shortest vector in an ideal lattice in comparison to a random, standard lattice of dimension 128. Finding the shortest vectors in a lattice is equivalent to discovering the secret information and thus breaking an entire cryptoscheme.

Critically, if the practical intention is that of cryptographic long-term security, serious considerations are needed to evaluate if the benefits of ideal lattice-based security outweigh the weakened confidence in security. For instance, fixed long-term security such as those on satellites, where re-programming in the field is generally not possible, may benefit from the use of standard lattice-based cryptography as it provides stronger long-term security assurances. Indeed, a scheme with stronger security assumptions is much more appropriate for these types of applications, especially considering that any compromise of the key material and/or algorithms would have devastating consequences to the security of data and/or the satellite itself.

The goal of this chapter is to investigate the potential of performing encryption over standard lattices in practice. For this purpose, a hardware architecture of a standard-LWE encryption scheme is proposed. The target platform is a Spartan-6 FPGA. As previously discussed in Chapter 1, FPGAs are highly suit-

able for use in cryptography as they are reconfigurable and allow fast prototyping. Moreover, FPGAs can be used for satellite applications with Xilinx [2015] producing radiation-hardened versions. Prior ring-LWE hardware designs [Göttert et al., 2012, Pöppelmann and Güneysu, 2013, Roy et al., 2014b, Pöppelmann and Güneysu, 2014] have usually targeted Spartan-6 FPGAs, thus targeting the Spartan-6 platform allows for fair performance comparison.

Additionally, previous research in this area initially proposed balanced hardware designs (that is, a trade-off between area and throughput) [Göttert et al., 2012, Pöppelmann and Güneysu, 2013] which were then extended to low-area, compact designs [Roy et al., 2014b, Pöppelmann and Güneysu, 2014]. Therefore, since this is the first investigation into standard LBC in hardware, a balanced hardware design is thus undertaken. This is also favourable as this allows for a fairer comparison versus ring-LWE encryption. The main contributions of this chapter are outlined as follows:

- The first hardware design of a standard lattice-based cryptoscheme is presented; implementing standard-LWE encryption on a Spartan-6 FPGA, which is designed to balance area and performance.
- A suitably fast discrete Gaussian sampler is incorporated from Chapter 3, that is the CDT sampler, such that samples can be readily produced in parallel and without delay to the critical path.
- Considering that ring-LWE requires only $\mathcal{O}(n)$ elements of $\mathbb{Z}$, as opposed to $\mathcal{O}(n^2)$, to represent n vectors [Regev, 2010], that is significantly larger keys and therefore many more operations required, the encryption scheme performs better than expected, even competing with balanced hardware designs for ring-LWE encryption [Göttert et al., 2012, Pöppelmann and Güneysu, 2013] and outperforming software designs [Göttert et al., 2012].

- The overall encryption and decryption modules operate in constant-time, due to the regularly structured matrix-vector multiplication unit using the FPGA's DSP slice. Hence, with all arithmetic and discrete Gaussian sampling operating in constant-time, the overall encryption and decryption schemes are protected against timing-analysis side-channel attacks.

The structure of this chapter is as follows: firstly, the LWE problem is briefly revisited (full descriptions can be found in Section 2.2.7) as well as the LWE encryption scheme (LP) by Lindner and Peikert [2011] (discussed in Section 2.4). Next, an assessment on adopting standard or ring-LWE cryptography is presented. Following this, a hardware architecture of the standard-LWE encryption scheme is presented. Finally, the implementation results are presented and a discussion on the performance of the proposed standard-LWE design in comparison to alternative ring-LWE hardware designs is given.

4.2 Learning With Errors

The learning with errors (LWE) problem, introduced by Regev [2005], is to find the secret-key $\mathbf{s}$, given access to $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{n \times n} \times \mathbb{Z}_q^n$. The public-key is generated uniformly given the lattice security parameter n and modulus q, and $\mathbf{b} \equiv \mathbf{As} + \mathbf{e} \mod q$ where $\mathbf{e}$ is noise added from an error distribution χ. Usually, χ is defined as the discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$, which is defined with standard deviation $\sigma \in \mathbb{R}$, where a value $x \in \mathbb{Z}$, sampled from D_σ, is output with probability proportional to $\rho_\sigma(x)/\rho_\sigma(\mathbb{Z})$ where $\rho_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$ and $\rho_\sigma(\mathbb{Z}) = \sum_{i=-\infty}^{\infty} \rho_\sigma(i)$.

4.2.1 LWE Encryption

Algorithm 4.1 defines the key generation, encryption, and decryption steps in the LWE encryption scheme (LP) by Lindner and Peikert [2011].

Algorithm 4.1 The LWE Encryption scheme by Lindner and Peikert [2011] (LP)

procedure KeyGen($\mathbf{A}, 1^\ell$)
  $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times n}$
  $\mathbf{R}_1, \mathbf{R}_2 \leftarrow D_\sigma^{n \times \ell}$
  $\mathbf{P} \equiv \mathbf{R}_1 - \mathbf{A} \cdot \mathbf{R}_2 \mod q$
end procedure
procedure Enc($pk = (\mathbf{A}, \mathbf{P}), \mathbf{m} \in \Sigma^\ell$)
  $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3 \leftarrow D_\sigma^n \times D_\sigma^n \times D_\sigma^\ell$
  $\bar{\mathbf{m}} = \text{encode}(\mathbf{m})$
  $\mathbf{c}_1 \equiv \mathbf{e}_1^t \mathbf{A} + \mathbf{e}_2^t \mod q$
  $\mathbf{c}_2 \equiv \mathbf{e}_1^t \mathbf{P} + \mathbf{e}_3^t + \bar{\mathbf{m}}^t \mod q$
end procedure
procedure Dec($sk = (\mathbf{R}_2), \mathbf{c}_1^t, \mathbf{c}_2^t$)
  $\mathbf{m} = \text{decode}(\mathbf{c}_1^t \mathbf{R}_2 + \mathbf{c}_2)$
end procedure

Table 4.1 A table of the main parameters and key sizes for LP, as proposed by Lindner and Peikert [2011].

Parameters	"Low"	"Medium"	"High"
Lattice dimension n	192	256	320
Modulus q, ($\lceil \log_2(q) \rceil$)	4096,(12)	4096,(12)	4096,(12)
Gaussian std. dev. σ	3.54	3.33	3.19
Public-key (pk) size	442.4 kb	786.4 kb	1229 kb
Secret-key (sk) size	1 kb	1.4 kb	1.7 kb
Ciphertext ($\mathbf{c}$) size	4.6 kb	6.1 kb	7.7 kb

Table 4.1 provides the parameter sets proposed by Lindner and Peikert [2011], with additional information on the public-key sizes, secret-key sizes, and ciphertext sizes. The hardware designs proposed are built around the 128-bit "medium" security parameter set $(n, q, \sigma) = (256, 4093, 3.33)$, but are adaptable for other parameter sets. The research by Brakerski et al. [2013] is used to simplify the modulus from $q = 4093$, to $q = 2^{12} = 4096$, without weakening the security.

This simplification allows for a more efficient hardware design, since the modular reduction component can be simplified to take the $\log_2(4096) = 12$ least significant bits.

4.2.2 The Pros and Cons of Ideal Lattice-based Cryptography

Ideal lattices are a subset of standard lattices, with the computational property of being related to polynomials via matrices of a certain form. That is, instead of having a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ that is independently and identically distributed, the matrix is structured in such a way that one column $\mathbf{a}_1 \in \mathbb{Z}_q^n$ is chosen, with the other $n-1$ columns derived as the coefficient representation of the polynomial $\mathbf{a}_1\mathbf{x}^i$ in the ring $\mathbb{Z}_q[\mathbf{x}]/\langle f \rangle$, for some univariate polynomial [Lyubashevsky, 2012].

With this construction of matrices, the matrix-vector multiplication $\mathbf{A} \cdot \mathbf{x}$ corresponds to polynomial multiplications and additions in the ring $\mathbb{Z}_q^n$. This means that the n-dimensional vector-matrix product costs $O(n \log n)$ arithmetic operations instead of $O(n^2)$, wherein optimised components such as NTTs [Pollard, 1971b, Emmart and Weems, 2011] can be used. This also means that key sizes are also much smaller; where $\mathcal{O}(n)$ elements of $\mathbb{Z}$ are able to represent n vectors, as opposed to $\mathcal{O}(n^2)$ [Regev, 2010].

As described in Section 2.2.7, the ring-LWE problem is defined much like the LWE problem. Given a prime modulus $q \equiv 1 \mod 2n$, random polynomials $\mathbf{s}, \mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n \in \mathbb{Z}_q[\mathbf{x}]/\langle \mathbf{x}^n + 1 \rangle$, where $\mathbf{b}_i \equiv \mathbf{a}_i\mathbf{s} + \mathbf{e}_i \mod q$, with $\mathbf{e}_i$ following some small error distribution, the goal of the ring-LWE problem is to find $\mathbf{s}$, given access to pairs $(\mathbf{a}_i, \mathbf{b}_i)$. Observe here the constraint on the prime modulus q, which is dependent on the lattice dimension n, causing significant restrictiveness when choosing these parameters.

For lattice-based cryptography in general, it can be seen that the shift to ideal lattices is desirable for efficient implementations. However, this move does not allow for a fine-grained parameter selection; essentially this is due to the restriction that the monic polynomial $f \in \mathbb{Z}$ in the quotient ring $\mathbb{Z}_q[\mathbf{x}]/\langle f \rangle$ needs to be irreducible, for instance $f = \langle \mathbf{x}^n + 1 \rangle$ if and only if n is a power of two.

As a consequence, many of the proposed parameter sets for ideal lattices are actually derived from its analogous problem in standard lattices. For instance, the parameters used by Pöppelmann and Güneysu [2014] in their hardware design of the LPR ring-LWE encryption by Lyubashevsky et al. [2013a] are taken from another encryption scheme (the LP encryption scheme) by Lindner and Peikert [2011]. Moreover, some of these parameter sets provide more security than required, which has adverse effects on performance.

Parameter selection for standard lattices is better understood and, although there are practical advantages to adopting an ideal lattice-based scheme, there are a number of reasons for choosing a scheme based on standard lattice problems. These reasons are mainly seen in the relative ease in parameter selection and avoiding the assumption that the security level associated with ideal lattices is equivalent to that of standard lattices. As previously mentioned, the concern that structured matrices weaken the security assumptions of ideal lattice-based cryptography is demonstrated in recent examples of weak ring-LWE instances [Eisenträger et al., 2014, Elias et al., 2015, Chen et al., 2015] and improved cryptanalysis of ideal lattices [Bos et al., 2014, Becker and Laarhoven, 2015, Cramer et al., 2015, Ishiguro et al., 2014, Schneider, 2013].

4.3 Standard-LWE Encryption in Hardware

Despite the recent research into both software and hardware designs for ideal LBC, there has not yet been an investigation into the performance of standard LBC in hardware. The only previous research that investigates the practical performance of standard-LWE encryption is by Göttert et al. [2012]. The research provides software results for both ring-LWE and standard-LWE encryption, however they only consider ring-LWE encryption in hardware due to its superior performance in software. In terms of software, for the same parameter set considered in this

research, encryption requires 11.01 milliseconds and decryption requires 2.37 milliseconds. These results are equivalent to achieving around 90 encryptions per second and 422 decryptions per second. These software results are improved upon significantly in the proposed hardware design.

The aim of this chapter is to assess the practicality of implementing a standard lattice-based PKE scheme in hardware. In Section 4.4, the optimised hardware design of the LWE LP scheme is presented. The main efficiencies in hardware are gained from algorithmic optimisations, meaning the encryption scheme operates recursively and repeatedly. More efficiencies are also gained from device-level optimisations, such as adapting the modulus parameter for hardware-friendly modular reduction and exploiting the DSPs on the FPGA platform.

The key generation stage is computed off-line in Python, with the keys being stored on-device in BRAM. The key storage is optimised such that the public-key is expressed as $\mathbf{A} = \mathbf{A}_0 \| \mathbf{A}_1$, which means all matrices $(\mathbf{A}_0, \mathbf{A}_1, \mathbf{P})$ are exactly the same size, allowing for repetitive matrix-vector computations on-device. The global constant $A \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$ is uniformly generated using the PyCrypto [Litzenberger et al., 2013] package `Crypto.Random.random`, a cryptographically strong version of Python's standard random module. The off-line Gaussian noise needed for matrices $\mathbf{R}_1, \mathbf{R}_2 \leftarrow D_\sigma$ is generated using Sage's discrete Gaussian sampler [Albrecht, 2014], where the matrices are used to calculate the public-key, $\mathbf{P} \equiv \mathbf{R}_1 - \mathbf{A} \cdot \mathbf{R}_2 \mod q$, with the secret-key, $\mathbf{R}_2$, being stored for use during decryption.

4.4 Hardware Optimised Architecture of Standard-LWE Encryption

In this section, the components within the proposed hardware architecture are detailed. The optimised LWE encryption scheme is described in Algorithm 4.2,

where the *SUM* operations, in lines 10 and 18, are multiply and accumulate (MAC) components. Figure 4.1 illustrates the high-level architecture of the LWE encryption scheme. A one-time initialisation stage is required to read in keys and, in parallel, generate an initial set of 256 discrete Gaussian samples for encryption. The core components of the hardware encryption unit are the discrete Gaussian sampler and the arithmetic unit, which are described in Section 4.4.1 and 4.4.3 respectively. For decryption, the hardware architecture is similar to encryption, with one exception, that it does not require a discrete Gaussian sampler module, and thus a hardware decryption diagram is omitted.

Encoding and decoding, outlined in Section 4.4.2, are relatively low cost operations in hardware, performed using simple bit shifting and a comparison. Therefore this is emitted from the high level architecture and the message data is presumed to be encoded in Algorithm 4.2.

4.4.1 Sampling the Discrete Gaussian Distribution

The preliminaries of the discrete Gaussian distribution are comprehensively detailed in Sections 2.2.2 and 2.6.2, and in this section they are briefly revisited. Due to the nature of the discrete Gaussian distribution, that is, in theory, its infinitely long tails and infinitely high precision, there are a number of compromises that need to be made in order to produce a practical (yet secure) implementation. Two parameters (λ, τ) are added to the other main parameter σ, to make a tuple of Gaussian parameters for a practical implementation of a discrete Gaussian sampler:

- *The standard deviation* (σ) is conventional when using the Gaussian distribution in general, and it governs the distribution's shape. When using the CDT approach, the standard deviation is set to $\sigma = 3.33$, however using the

Algorithm 4.2 Enc$(pk = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{P}), \mathbf{m} \in \{0,1\}^{\ell})$

for $i = 0$ to $n-1$ **do**
  $\mathbf{e}_1(i) \leftarrow D_\sigma^n$ ▷ Computed on-the-fly after first encryption.
end for
for $k = 0$ to 2 **do** ▷ i.e., $\mathbf{A}_0$, $\mathbf{A}_1$, or $\mathbf{P}$.
  $\mathit{SUM} = 0$
  if $k \in \{0,1\}$ **then**
    for $j = 0$ to $\ell - 1$ **do**
      $e \leftarrow D_\sigma$ ▷ Computed in parallel to MAC operations.
      for $i = 0$ to $n-1$ **do**
        $\mathit{SUM} := \mathit{SUM} + \mathbf{e}_1(i) \times \mathbf{A}_k(i,j) \mod q$
      end for
      $\mathbf{c}_i(j) = \mathit{SUM} + e \mod q$
    end for
  else
    for $j = 0$ to $\ell - 1$ **do**
      $e \leftarrow D_\sigma$
      for $i = 0$ to $n-1$ **do**
        $\mathit{SUM} := \mathit{SUM} + \mathbf{e}_1(i) \times \mathbf{P}(i,j) \mod q$
      end for
      $\mathbf{c}_i(j) = \mathit{SUM} + e + \bar{\mathbf{m}}(j) \mod q$
    end for
  end if
end for

Bernoulli approach the standard deviation is required to be $\sigma = 3.39$ (see Section 3.2.1 for more details).

- *The precision parameter* (λ) determines the level of precision required in a practical design, where the statistical distance between the "perfect" theoretical distribution and the one chosen for practice must be no greater than $2^{-\lambda}$. As described in Section 3.2, Saarinen [2015, 2017] recommends that for a target security level of λ-bit, the precision does not need to be any greater than $\lambda/2$, arguing that there exists *no algorithm* that can distinguish between a "perfect" sampler and one with statistical distance $2^{-\lambda/2}$.

- *The tail-cut parameter* (τ) is used in calculating how much of the less-heavy tails can be excluded from the practical design, for a given security level. That is, with a target security level of λ-bits, the portion of the tails that are undesired is less than $2^{-\lambda}$. Therefore, the tail-cut parameter can be defined as $\tau = 13.4$, for target security level $\lambda = 128$.

The discrete Gaussian sampler architectures used in this research therefore consider the three practical Gaussian parameters (σ, λ, τ). Conventionally, the sampled values are considered over the positive and negative axis; however due to the symmetrical property of the discrete Gaussian distribution, only the positive is required, that is $x \in \{0, \ldots, \tau\sigma\}$, with the probability $x = 0$ being halved (due to it being counted twice), where a sign-bit $b \in \{0, 1\}$ is sampled to recover the negative axis. In Section 3.2, this straightforward optimisation is also used, which mainly benefits the samplers which employ tables.

In this research, two techniques are considered for discrete Gaussian noise generation: these are the Bernoulli sampling technique [Ducas et al., 2013] and the cumulative distribution table (CDT) sampling technique [Peikert, 2010], which are fully described in Sections 3.2.1 and 3.2.1, respectively. The sampling techniques will now be briefly revisited.

Bernoulli Sampling

Previous research within ideal LBC has shown the Bernoulli technique to perform best on small devices Oder et al. [2014] and for small standard deviations Pöppelmann and Güneysu [2014], thus the Bernoulli technique is consider for this research.

The Bernoulli sampler is also employed due to its simplicity and adaptability in hardware, and does not require large pre-computations and long-integer arithmetic. The approach combines the inversion method by Peikert [2010] and the original discrete Gaussian sampling technique of rejection sampling (Section 3.2.1) by Gentry et al. [2008]. The calculation of transcendental functions[1], seen in the rejection sampling technique, is bypassed by the use of Bernoulli variables. This is then combined with the inversion method, to sample from the so-called binary discrete Gaussian distribution $D_{\sigma_{\text{bin}}}$, where $\sigma_{\text{bin}} = \sqrt{1/(2\ln(2))}$, whose cumulative probabilities are of a special form in binary, meaning that they can be computed on-the-fly whilst also reducing the probability of rejection. The Bernoulli technique is described in Algorithm 3.1, 3.2, and 3.3, which, when sampling according to the Bernoulli distribution $\mathcal{B}_{(-x/f)}$ using a small pre-computed table of $\lceil \log_2(\max(x)) \rceil$ entries, is calculated with λ-bits of precision.

Cumulative Distribution Table (CDT) Sampling

In Chapter 3, the Cumulative Distribution Table (CDT) sampling technique was found to have the best performance on FPGAs for encryption parameters [Howe et al., 2016a], therefore the CDT hardware design is also incorporated in the design proposed in this chapter.

[1]These types of functions are not expressible by a finite sequence of algebraic operations, such as addition or multiplication. In essence they "transcend" algebra. Example of these types of functions are the sine function, the cosine function, and the exponential function.

The CDT sampler is a table-based sampler and is described in Algorithm 3.4. The technique stores precomputed discrete Gaussian cumulative distribution function (CDF) values in a table $S[\cdot]$. The CDF values are then accessed by sampling a uniformly random value $r \in [0, 1)$, where the desired sample x is found satisfying interval $S[x] \leq r < S[x + 1]$, occurring with probability $\rho[x] = S[x + 1] - S[x]$.

Fast Discrete Gaussian Sampling

Importantly, these samplers were designed such that their cycles per sample (throughput) performance is high enough, ensuring reductions in latency for the encryption architecture. This is further discussed in Section 4.4.3. More specifically, in order to not delay the multiplier-accumulator (MAC) products seen in Lines 10 and 18 of Algorithm 4.2, one discrete Gaussian sample is required, on average, in less than 128 clock cycles.

Previous research on Bernoulli samplers used within LBC produced a discrete Gaussian sample, on average, every 144 clock cycles [Pöppelmann and Güneysu, 2014]. Therefore, a new Bernoulli design is proposed, which gains in throughput by incorporating additional uniform randomness via unrolled stream ciphers (Trivium [De Cannière, 2006]). The Bernoulli design consumes slightly more area in comparison to the one proposed by Pöppelmann and Güneysu [2014] (the samplers are compared in Table 3.2), but allows for faster sampling and is designed to match the performance of the arithmetic unit, detailed in Section 4.4.3.

The CDT sampler design in Section 3.3.2 also matches the performance requirements for this encryption design, with a significant reduction in area consumption. Additionally, the constant run-time of the sampler means the entire encryption scheme operates in constant-time, therefore being resilient to side-channel timing analysis.

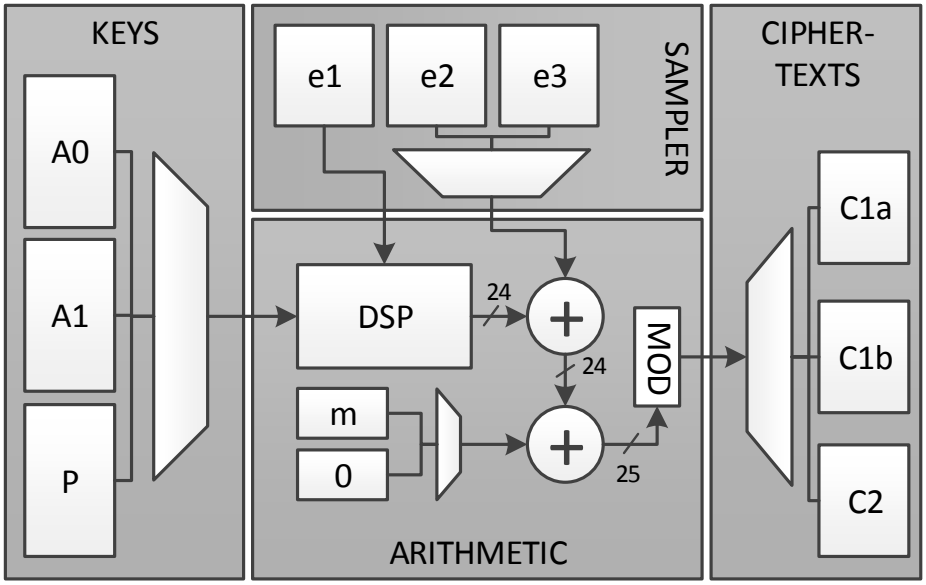


Fig. 4.1 High level architecture of LWE encryption scheme. Lengths are 12 bits unless otherwise stated.

The discrete Gaussian sampling component (either Bernoulli or CDT) can be seen as the `SAMPLER` component of Figure 4.1.

4.4.2 Encoding/Decoding

This stage of the encryption scheme encodes the message $\mathbf{m} \in \{0,1\}^n$. Encoding the message is necessary due to the small noise terms, $\mathbf{e}_1^t\mathbf{R}_1 + \mathbf{e}_2^t\mathbf{R}_2 + \mathbf{e}_3^t$, being present after decryption. The encoding of the message is defined such that, for each bit m of the message $\mathbf{m}$, encode$(m) = \bar{m} := m\lfloor\frac{q}{2}\rfloor \in \mathbb{Z}_q^\ell$. Decoding is adapted from Lindner and Peikert [2011] and is optimised to work with unsigned integers, hence for this research decode$(\bar{m}) := 1$ if $\frac{1}{4}q \leq \bar{m} < \frac{3}{4}q$, and 0 otherwise.

4.4.3 Arithmetic

The computations required for the encryption scheme are computed in the arithmetic module, `ARITHMETIC`, shown in Figure 4.1. The arithmetic subsumes almost the entirety of Algorithm 4.2, that is, the for-loop from Lines 4 to 23. Within this loop, the value of k refers to the operation currently taking place, that is;

$$\begin{aligned}
&\text{for } k = 0 \text{ the operation } \mathbf{c}_{1a} \equiv \mathbf{e}_1^t \times \mathbf{A}_0 + \mathbf{e}_2^t \mod q,\\
&\text{for } k = 1 \text{ the operation } \mathbf{c}_{1b} \equiv \mathbf{e}_1^t \times \mathbf{A}_1 + \mathbf{e}_2^t \mod q,\\
&\text{for } k = 2 \text{ the operation } \mathbf{c}_2 \equiv \mathbf{e}_1^t \times \mathbf{P} + \mathbf{e}_3^t + \bar{\mathbf{m}}^t \mod q.
\end{aligned}$$

A DSP (DSP48A1) slice on the target Spartan-6 FPGA [Xilinx, 2011] is used to calculate the vector-matrix MAC products as well as the addition of the error vectors and message. Each entry in the ciphertext is computed individually, using two clocked counters corresponding to the row and column addresses of the matrices $\mathbf{A}_0$, $\mathbf{A}_1$, and $\mathbf{P}$. Additionally, the structural decomposition of the two vector-matrix products into three identical operations allows for ease of repetition, meaning that the component can be re-used for computing $\mathbf{e}_1^t\mathbf{A}_0$, $\mathbf{e}_1^t\mathbf{A}_1$, or $\mathbf{e}_1^t\mathbf{P}$, since the matrices ($\mathbf{A}_0, \mathbf{A}_1, \mathbf{P} \in \mathbb{Z}_q^{n\times\ell}$) are exactly the same size. Moreover, since

the arithmetic is the bottleneck of the encryption scheme, the discrete Gaussian random number generator and message encoding can be implemented in parallel to the MAC operations.

The MAC operations for which the DSP is used are seen in Lines 10 and 18 of Algorithm 4.2, where 256 clock cycles are required for each accumulation computation per index of j. During which, two discrete Gaussian samples are needed; one for the addition of the discrete Gaussian noise (seen as e in Lines 12 and 20), and the other for the error-vector store $\mathbf{e}_1$ for the next encryption. For this, a double-buffered store (sometimes called the page-flip method or ping-pong buffering) for the discrete Gaussian noise vector $\mathbf{e}_1$ is used within the design. The first buffer is used to provide values for the current encryption, where the second is used to accumulate new values.

The buffers are then swapped at the end of the encryption cycle. This buffer minimises latency since an initial generation of $\mathbf{e}_1 \leftarrow D_\sigma^n$ is only needed once (and can be completed in parallel to the key read-in), instead of per encryption. This double-buffered store switch and parallel discrete Gaussian sampling save 3072 clock cycles per encryption using the Bernoulli sampler [Howe et al., 2016b] and 1280 clock cycles per encryption using the CDT sampler[2]. The increase in area consumption using the double-buffered store is outweighed by the reduction gains in clock cycles.

As stated in Section 4.2.1, the modulus is changed from $q = 4093$ to $q = 2^{12}$. This is undertaken to allow for a more optimal hardware design for encryption, as this significantly simplifies the modular reduction calculation to selecting the 12 least significant bits from the final result. This "Modulus-Dimension Switching" is shown by Brakerski et al. [2013] to have insignificant affect on the security of LWE.

[2]These savings differ due to the CDT sampler operating with a higher throughput than the Bernoulli sampler.

4.5 Results

Table 4.2 shows the performance of the proposed hardware design of standard-LWE encryption and decryption. Results are also provided for all previous research on ring-LWE hardware designs [Göttert et al., 2012, Pöppelmann and Güneysu, 2013] and low-area ring-LWE encryption designs [Roy et al., 2014b, Pöppelmann and Güneysu, 2014] for comparison.

The proposed hardware architectures were designed to balance area consumption with throughput performance. The proposed balanced design can be seen by the fact that only one DSP of the FPGA is utilised, but a double-buffered store for the error-vector is used as well as two PRNGs for the Bernoulli sampler, both of which significantly decrease latency.

Since the design was balanced, the comparisons of standard-LWE encryption versus ring-LWE encryption [Göttert et al., 2012, Pöppelmann and Güneysu, 2013] are the most fair, meaning that the security trade-off for adopting standard-LWE encryption, as discussed in Section 4.2.2, can be equally explored. The choice of comparing against the balanced ring-LWE encryption designs [Göttert et al., 2012, Pöppelmann and Güneysu, 2013], as opposed to low-area ring-LWE encryption designs [Roy et al., 2014b, Pöppelmann and Güneysu, 2014], was made due to the already large area consumption required in theory. This large area consumption is for key storage and is bounded to $(3/2)n^2 \times \log_2(q)$-bits (1180 kilo-bits (kb)), which is 196x larger (6 kb) than the key storage requirements for ring-LWE encryption.

Göttert et al. [2012] present the only previous research on software designs for standard-LWE encryption, which on a Intel Core 2 Duo CPU running at 3 GHz with 4Gb of RAM, produces $1/0.01101 \approx 90$ encryptions per second and 422 decryptions per second. Hence, the proposed hardware design outperforms

software by more than 14x in terms of encryptions per second and 10x in terms of decryptions per second.

Currently there exists no other hardware designs of standard-LWE in the literature. Thus, the results are compared to existing results achieved for hardware designs of ring-LWE encryption and decryption. The comparisons are used as a benchmark to measure the practicality of the proposed hardware designs for standard-LWE. It must be noted that a direct comparison of ring-LWE and standard-LWE designs is somewhat incongruous; ring-LWE schemes inherently require much smaller key sizes and therefore are expected to consume less hardware resources than a hardware design for standard-LWE. For instance, 73 Block RAMs are used in this design, which includes key storage, message storage, and ciphertext storage. Consequently, larger key sizes also have an effect on the LUTs, FFs, and slices used. However, the results compete with many of the ring-LWE hardware designs in terms of speed.

For cycle count and operations per second results, it should be considered that in general 192x more calculations are needed for standard-LWE compared to ring-LWE. Despite this, the clock cycle count of 98304 is much closer than expected to Pöppelmann and Güneysu [2013] and Roy et al. [2014b], whilst bettering clock cycles counts for the hardware designs by Pöppelmann and Güneysu [2014]. Achieving 1272 encryptions per second is at most around 40x less than for the balanced designs by Pöppelmann and Güneysu [2013] and Roy et al. [2014b], but outperforms the throughput of the low-area design by Pöppelmann and Güneysu [2014].

Comparing results for standard-LWE decryption is difficult, since most implementations assimilate these with encryption results. However, the results compare very well to the ring-LWE decryption results of Pöppelmann and Güneysu [2014], whilst significantly improving upon those by Göttert et al. [2012].

From the results, it is clear that the CDT sampler (described in Section 3.3.2) provides the most optimal standard-LWE encryption hardware design. This is contrary to a previous survey [Howe et al., 2015], which concluded that the Bernoulli sampler would be the best suited for smaller designs such as encryption. However, the encryption results with the CDT sampler show a significant improvement from the original approach using the Bernoulli sampler [Howe et al., 2016b], with the results competing closer to the other ring-LWE encryption hardware designs seen in Table 4.2.

With regards to area consumption, the size of the encryption module is significantly affected by the size of the public-keys, that is, the matrices $\mathbf{A}_0, \mathbf{A}_1$, and $\mathbf{P}$. The secret-key size is 196.6 kb in the decryption module, compared to 3 kb for ring-LWE; however since keys are 6x smaller, with decryption using 80-times less slices and computing more than 3 decryptions per encryption, it has less significance. The ciphertext size for standard-LWE is less than that of ring-LWE, with standard-LWE being 4.6 kb whereas a ring-LWE ciphertext is 6.1 kb.

The area consumed by the proposed hardware architecture of standard-LWE encryption is greater than the other low-area ring-LWE hardware designs; however standard-LWE does not require the additional security assumptions associated with using structured ideal lattices and still fits on a lightweight device. Thus, there exists a trade-off in terms of security and performance when choosing between standard-LWE and ring-LWE schemes.

Table 4.2 Post-place and route results of standard-LWE (LWE) and ring-LWE (RLWE) encryption/decryption, using the main parameter set $(256, 4096, 3.39)$, except Göttert et al. [2012], Pöppelmann and Güneysu [2013], Roy et al. [2014b] where $q = 4093$.

Operation & Algorithm	**Device**	**LUT/FF/SLICE**	**BRAM/DSP**	**MHz**	**Cycles**	**Ops/s**
LWE Encrypt (Bernoulli Sampler) [Howe et al., 2016b]	S6LX45	6078/4676/1811	73/1	125	98304	1272
LWE Encrypt (CDT Sampler)	S6LX45	4857/3172/2203	73/1	125	98304	1272
LWE Decrypt [Howe et al., 2016b]	S6LX45	63/58/32	13/1	144	32768	4395
RLWE Encrypt (CDT-like Sampler) [Göttert et al., 2012]	V6LX240T	298016/ – /143396	–/–	–	–	< 18200
RLWE Decrypt [Göttert et al., 2012]	V6LX240T	124158/ – /65174	–/–	–	–	< 29540
RLWE Encrypt (CDT Sampler) [Pöppelmann and Güneysu, 2013]	S6LX16	4121/3513/–	14/1	160	6861	23321
RLWE Decrypt [Pöppelmann and Güneysu, 2013]	S6LX16	4121/3513/–	14/1	160	4404	36331
RLWE Encrypt (CDT Sampler) [Pöppelmann and Güneysu, 2013]	V6LX75T	4549/3624/1506	12/1	262	6861	38187
RLWE Decrypt [Pöppelmann and Güneysu, 2013]	V6LX75T	4549/3624/1506	12/1	262	4404	59492
RLWE Encrypt (Bernoulli Sampler) [Pöppelmann and Güneysu, 2014]	S6LX9	282/238/95	2/1	144	136212	1057
RLWE Decrypt [Pöppelmann and Güneysu, 2014]	S6LX9	94/87/32	1/1	189	66338	2849
RLWE Encrypt (Knuth-Yao Sampler) [Roy et al., 2014b]	V6LX75T	1349/860/–	2/1	313	6300	49751
RLWE Decrypt [Roy et al., 2014b]	V6LX75T	1349/860/–	2/1	313	2800	109890
NTRU [Enc/Dec] (KaYo)	XCV1600E	27292/5160/14352	–/–	62	–	–
ECC-P224 (GP)	XC4VFX12	1825/1892/1580	11/26	487	178000	2740
ECC-B233 (RRM)	XC5VLX85T	18097/ – /5644	–/–	156	1919	81300

4.6 Conclusions

In this chapter, the first hardware design of a standard-LWE encryption scheme is proposed. Since this research is the first to consider standard lattices in hardware, the performance results are compared with equivalent ring-LWE encryption designs. The results for standard-LWE encryption compete with ring-LWE encryption results despite significantly larger keys, meaning many more operations required. Larger keys means more area consumption and a slower throughput, but considering the keys are 192x larger than compared to ring-LWE encryption, the results for standard-LWE encryption better expected results for both area consumption and throughput.

The proposed standard-LWE results closely competes with the ring-LWE encryption hardware designs by Göttert et al. [2012] in terms of area consumption. In terms of encryptions per second, the proposed hardware design also betters the ring-LWE encryption design by Pöppelmann and Güneysu [2014]. The proposed hardware design also outperforms the only software design of standard-LWE encryption [Göttert et al., 2012] by more than 14x in terms of encryptions per second and more than 10x in terms of decryptions per second.

In addition to the relative ease of parameter selection and security benefits associated with standard-LWE (as discussed in Section 4.2.2), the design fits comfortably on a lightweight Spartan-6 FPGA and offers comparable performance to existing ring-LWE schemes, despite having larger key sizes and more multiplication operations. Thus, due to potential security risks associated with ring-LWE schemes, and the practical performance of the standard-LWE hardware encryption and decryption designs illustrated in this research, standard-LWE schemes should be considered for applications requiring long term security assurances.

Long term security applications, such as satellites, would not be able to be reprogrammed in the field. As such, ring-LWE encryption may not be suitable if vulnerabilities are found after the satellite has been deployed. Therefore, standard-LWE encryption would be more suited for these types of applications.

Another key component in ensuring secure communications is a digital signature, which is used for applications such as message authentication and message integrity. Moreover, the use of digital signatures in hardware is increasing, due to the use of secure contact-less bank payments [Murdoch et al., 2010] and smart cards [Rankl and Effing, 2010, p.174-178], which both require signatures for authentication. Furthermore, hardware designs of lattice-based signatures are currently understudied, and it is therefore prudent to investigate alternatives to the current research for efficiencies in area, speed, or both. Therefore a highly-secure, lattice-based digital signature scheme is investigated in Chapter 5.

CHAPTER 5

Ideal Lattice-Based Digital Signatures in Hardware

In this chapter, the hardware architectures of the ring-LWE digital signature scheme Ring-TESLA are presented. This research demonstrates the efficiency of a lattice-based digital signature scheme based on the ring-LWE problem (previous hardware designs rely on NTRU and ring-SIS hardness assumptions). Ring-LWE is described by Peikert [2014] as "a solid foundation on which to design cryptosystems", which also benefits from ease of parameter selection. Moreover, NTRU currently has a number of active patents [Hoffstein et al., 2000, 2001b, Hoffstein and Silverman, 2006, Hoffstein et al., 2007], which could cause issues when considering a scheme in practice.

The proposed designs are compact, targeting long-term security, and low area applications. The results in most cases significantly better the area consumption of all previous lattice-based DSSs in hardware, whilst also remaining competitive with regards to throughput performance. This research also investigates parameter selection for this signature scheme, thus hardware-friendly parameters are proposed which reduce the area consumption in comparison to the parameters proposed by the authors. Ring-TESLA is shown to be provably secure with a tight security

reduction, and it does not require on-device discrete Gaussian sampling; these qualities make it preferable over the state-of-the-art.

5.1 Introduction

The previous chapter investigated a hardware design of a lattice-based encryption scheme. A digital signature scheme (DSS) is another important primitive for building secure systems and is used in most real world security protocols. Almost all popular digital signature schemes are either based on the factoring assumption (RSA) or the hardness of the discrete logarithm problem (DSA/ECDSA). Recently, hardware designs of DSSs have become more widespread, with applications in a variety of areas such as vehicle-to-everything (V2X), the Internet of Things (IoT), and e-commerce technologies [Rupani et al., 2016, Intel, January 2017]. Intel [January 2017] state that "FPGAs are ideally suited to performing these tasks. They operate quickly and can be reconfigured when necessary to upgrade security settings." An example of this is reported by Glas et al. [2011], who propose a FPGA V2X communication accelerator based on an ECDSA signature over 256-bit prime fields. FPGAs are generally considered more tamper resistant than software, and recently hackers showed vulnerabilities in Land Rover software which allowed them to control the car's break and engine systems [Greenberg, July 2015].

In the case of classical cryptanalytic advances or progress on the development of quantum computers, the hardness of these closely related cryptoschemes (RSA/DSA/ ECDSA) are seriously weakened. A potential alternative approach is the construction of signature schemes based on the hardness of certain lattice problems [Lyubashevsky, 2009, 2012, Güneysu et al., 2012, Dagdelen et al., 2014, Ducas et al., 2013, Bai and Galbraith, 2014b, Alkim et al., 2015, Akleylek et al., 2016] which are assumed to be intractable by quantum computers. As discussed

in Chapter 1, in recent years lattice-based schemes have become practical and appear to be a very viable alternative to number-theoretic cryptography.

Previous hardware designs of lattice-based DSSs, such as the GLP scheme by Güneysu et al. [2012] and the BLISS scheme by Pöppelmann et al. [2014], demonstrate good performance and outperform ECDSA and RSA hardware designs. However, a significant number of hardware resources are consumed by the costly discrete Gaussian sampler in the BLISS scheme. Additionally, these schemes rely on extra security assumptions; GLP is based on the decisional knapsack problem and only provides 80-bit classical security (Section 2.5.2) and BLISS is based on a combination of NTRU and ring-SIS hardness assumptions. These assumptions do not offer the very appealing average-case to worst-case hardness property offered by ring-LWE. This quality renders *all* cryptographic constructions based on it secure, under the assumption that worst-case lattice problems are hard. Additionally, for both GLP and BLISS, their parameters are not chosen directly from their security reduction, meaning their *instantiations* are not provably secure.

An alternative lattice-based DSS was recently proposed by Akleylek et al. [2016]. The proposed scheme, named Ring-TESLA, is shown to be based on the appealing ring-LWE problem, which also does not require discrete Gaussian sampling on-device. This DSS competes well with GLP and BLISS in software [Akleylek et al., 2016] in terms of Sign and Verify cycle counts.

Currently, no hardware designs exists of Ring-TESLA. Applications for hardware designs of DSSs already discussed, such as V2X and IoT, require strong security and a high performance rate, which makes a hardware design of Ring-TESLA on a FPGA ideal. The growth of FPGA use is also discussed in Section 1.1, for example being deployed in data centres [Freund, December 2016, Metz, December 2016] due to the hardware and energy savings they provide, as well as for their use in encryption and compression.

This chapter focuses on hardware designs of the Ring-TESLA signature and verification algorithms. The results compete well with GLP and BLISS, showing that for applications requiring compact or highly secure instantiations with worse-case to average-case security, Ring-TESLA should be preferred. The proposed architecture makes use of parallelised schoolbook polynomial multipliers with Barrett reduction for a low-area modular multiplication component. The hardware designs also make use of Trivium x32 (as a PRNG), a state-of-the-art hash function (SHA-3), and a low-area multiplier specifically for low-Hamming weight (LHW) polynomial multiplication. The overall hardware designs outperform all existing lattice-based DSSs in hardware in terms of area consumption, and compete well in terms of throughput.

The chapter is structured as follows: the Ring-TESLA scheme is detailed in Section 5.2, with parameter selection discussed in Section 5.3. Section 5.4 discusses the design goals of these hardware designs, Section 5.5 then outlines the proposed hardware designs of the signature scheme, with the results of the hardware designs given in Section 5.6.

5.2 Ideal Lattice-Based Signatures

As previously discussed, LBC is emerging as a promising quantum-resistant alternative to ECC or RSA, and offers efficient performance for both PKE and DSSs. As discussed in Sections 2.2.7 and 4.2.2, for significant efficiency gains, ideal lattices are usually used instead of standard lattices, as they allow for smaller key sizes and faster computations by computing over a specific algebraic structure. The ring-LWE problem [Lyubashevsky et al., 2013a], commonly used in LBC, is well studied and demonstrates strong computational hardness.

As described in Sections 2.3 and 2.5.3, the most practical lattice-based DSSs are based upon the Fiat-Shamir paradigm [Howe et al., 2015], such as the state-of-the-

art BLISS scheme by Ducas et al. [2013], which is based upon ideal lattices with NTRU assumptions. NTRU cryptoschemes have existed for a significant period of time, with the only current serious break in NTRU-based schemes targeting NTRUSign [Ducas and Nguyen, 2012b]. However, the hardness assumptions of NTRU are not related to the hardness of worst-case lattice problems, a useful property of the ring-LWE problem [Stehlé and Steinfeld, 2011]. Accordingly, an ideal lattice-based DSS based on the ring-LWE problem has been proposed by Akleylek et al. [2016], named Ring-TESLA, which provides three appealing properties:

Firstly, the scheme competes well in terms of throughput performance with other lattice-based signatures (GLP and BLISS) in software, despite having larger input and output sizes (see Table 2.5 in Chapter 2).

Secondly, Ring-TESLA provides a tight security reduction, a provably secure instantiation, and worst-case hardness, which is not provided by GLP or BLISS. If a reduction proving security is "loose," the efficiency of the scheme is impacted, due to a larger parameters. Moreover, cryptoschemes that have provably secure instantiations are considered more theoretically secure [Bellare and Rogaway, 1996], especially as non-tight cryptoschemes have been shown to provide weaker security assurances [Chatterjee et al., 2011].

Secondly, Ring-TESLA provides worst-case hardness since it is based on the ring-LWE problem. As discussed in Section 2.2.7, this worst-case hardness quality renders *all* cryptographic constructions based on it secure, under the assumption that worst-case lattice problems are hard. In other words, breaking the cryptographic construction implies an efficient algorithm for solving any instance of some underlying lattice problem, such as SVP or CVP. This is not offered by GLP or BLISS.

Thirdly, the Ring-TESLA Sign and Verify algorithms do not require on-line discrete Gaussian sampling, and instead this can be computed off-device during

key generation. This minimises resource consumption, as the discrete Gaussian component is very costly in hardware.

For example, in the BLISS hardware design [Pöppelmann et al., 2014], the discrete Gaussian sampling module consumes $\sim$ 15% of the overall hardware resources. The discrete Gaussian module is also known to be susceptible to side-channel analysis (as discussed in Section 3.2.2), especially low-cost timing attacks, since the samplers operate in non-constant time due to the inherent normalised structure of the distribution.

The Ring-TESLA signature scheme is a ring-based version of the TESLA signature scheme by Alkim et al. [2015], which originated from a signature scheme by Bai and Galbraith [2014a] with optimisations by Dagdelen et al. [2014]. Table 5.1 shows the 128-bit parameter set provided by Akleylek et al. [2016]. The modulus is increased from $q = 39960577$ to $q = 51750913$, as the security reduction with the original modulus caused a slightly bigger security gap than expected[1]. As discussed in Section 2.3.3, a flaw in the security reduction of Ring-TESLA was found, post-publication. The flaw does not lead to an actual attack nor does it affect the security of the scheme, however the specific instantiations are affected[2]. The fix for this is ongoing, with attempts made by Barreto et al. [2016] and Chopra [2016]. The authors have so far addressed these issues [Alkim et al., 2017] for the standard lattice-based signature scheme TESLA (the standard equivalent to the ideal lattice-based Ring-TESLA), and plan to address Ring-TESLA in a similar fashion.

The Ring-TESLA algorithms KeyGen(1^n), Sign$_{sk}$, and Verify$_{pk}$ are outlined in Algorithms 5.1, 5.2, and 5.3. Key generation is assumed to be pre-computed off-line. The required arithmetic components for Sign and Verify are polynomial multiplication and addition, modular reduction, and hashing. The global parame-

[1]The updated modulus parameter is found on the TESLA homepage (https://tesla.informatik.tu-darmstadt.de/de/tesla/) and was verified with the authors.

[2]The security flaw is also discussed on the TESLA homepage.

Table 5.1 The parameter sets for RING-TESLA, as proposed by Akleylek et al. [2016].

Parameters	RING-TESLA-I	RING-TESLA-II
Security	80-bits	128-bits
Lattice dimension n	512	512
Modulus q, ($\lceil \log_2(q) \rceil$)	8399873,(23)	51750913,(26)
Weight of the challenge ω	11	19
Gaussian std. dev. σ	30	52
Dropped bits d	21	23
Error threshold L	814	2766
Sign/Verify thresholds B, U	$2^{21} - 1,993$	$2^{22} - 1,3173$
Repetition rate	4.35	2.94
Secret-key size (kilo-bits)	13.82 kb	15.36 kb
Public-key size (kilo-bits)	24.58 kb	26.62 kb
Signature size (kilo-bits)	11.39 kb	18.56 kb

ters are the polynomials $\mathbf{a}_1, \mathbf{a}_2 \xleftarrow{\$} \mathcal{R}_q^{\times}$, ($\mathcal{R}_q^{\times}$ being the set of the units of $\mathcal{R}_q$) which are required for all steps in the signature scheme.

The key generation procedure is shown in Algorithm 5.1. Firstly, three discrete Gaussian distributed polynomials are generated $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow D_{\sigma}^{n}$, with $\mathbf{e}_1$ and $\mathbf{e}_2$ checked for validity [Akleylek et al., 2016]. Two ring-LWE polynomials are then calculated $\mathbf{t}_1 \equiv \mathbf{a}_1\mathbf{s} + \mathbf{e}_1 \mod q$ and $\mathbf{t}_2 \equiv \mathbf{a}_2\mathbf{s} + \mathbf{e}_2 \mod q$ as the scheme's public-key (pk), with the polynomials $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2$ being the scheme's secret-key (sk).

To sign a message μ, a uniform polynomial $\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$ is generated for use in the calculation of the signature and for validity checks. Firstly, it is used to calculate intermediate polynomials $\mathbf{v}_1 \equiv \mathbf{a}_1\mathbf{y} \mod q$ and $\mathbf{v}_2 \equiv \mathbf{a}_2\mathbf{y} \mod q$. Along with the message data μ, the polynomials $\mathbf{v}_1$ and $\mathbf{v}_2$ are input into the hash function $H(\cdot)$, which outputs the bit-string c. This bit-string c, of length n and Hamming weight ω, is then transformed into a low-Hamming weight polynomial $\mathbf{c}$. This is achieved by extracting the ω positions of c which are one, meaning the $n - \omega$ zero positions are bypassed, improving latency.

Algorithm 5.1 Key Generation Algorithm for Ring-TESLA

procedure KeyGen(1^λ, $\mathbf{a}_1$, $\mathbf{a}_2$)
  $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow D_\sigma^n$
  if $\text{check}_E(\mathbf{e}_1) = 0 \bigvee \text{check}_E(\mathbf{e}_2) = 0$ **then**
    Restart
  end if
  $\mathbf{t}_1 \equiv \mathbf{a}_1\mathbf{s} + \mathbf{e}_1 \mod q$, $\mathbf{t}_2 \equiv \mathbf{a}_2\mathbf{s} + \mathbf{e}_2 \mod q$
  $sk \leftarrow (\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2)$, $pk \leftarrow (\mathbf{t}_1, \mathbf{t}_2)$
  return (sk, pk)
end procedure

Algorithm 5.2 Signing Algorithm for Ring-TESLA

1: **procedure** Sign(μ, $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{s}$, $\mathbf{e}_1$, $\mathbf{e}_2$)
2: $\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$
3: $\mathbf{v}_1 \equiv \mathbf{a}_1\mathbf{y} \mod q$, $\mathbf{v}_2 \equiv \mathbf{a}_2\mathbf{y} \mod q$
4: $c = H(\lfloor \mathbf{v}_1 \rceil_{d,q}, \lfloor \mathbf{v}_2 \rceil_{d,q}, \mu)$
5: $\mathbf{c} = F(c)$
6: $\mathbf{z} \leftarrow \mathbf{y} + \mathbf{sc}$
7: $\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$, $\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$
8: **if** $[\mathbf{w}_1]_{2^d}, [\mathbf{w}_2]_{2^d} \notin \mathcal{R}_{2^d-L}$ **or** $\mathbf{z} \notin \mathcal{R}_{B-U}$ **then**
9: Restart
10: **end if**
11: **return** $(\mathbf{z}, c)$
12: **end procedure**

The LHW polynomial $\mathbf{c}$ is then used to calculate the signature $\mathbf{z} \equiv \mathbf{y} + \mathbf{sc}$ as well as the polynomials $\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$ and $\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$ which are used to check the validity of a signature.

The verification algorithm, shown in Algorithm 5.3, is similar to the signing algorithm without uniform polynomial generation and the calculation of the signature. The differences between Sign and Verify are illustrated in Table 5.2. The intermediate polynomials are calculated as $\mathbf{w}_1' \equiv \mathbf{a}_1\mathbf{z} - \mathbf{t}_1\mathbf{c} \mod q$ and $\mathbf{w}_2' \equiv \mathbf{a}_1\mathbf{z} - \mathbf{t}_2\mathbf{c} \mod q$, with $\mathbf{a}_1\mathbf{z} \mod q$ and $\mathbf{a}_2\mathbf{z} \mod q$ input into the polynomial multiplication module and $\mathbf{t}_1\mathbf{c}$ and $\mathbf{t}_2\mathbf{c}$ calculated using a LHW multiplier. These polynomials are then input into the hash function which outputs c'. Should the signature size be valid, as well as $c = c'$, the signature is verified.

Algorithm 5.3 Verification Algorithm for RING-TESLA

procedure VERIFY(μ, $\mathbf{z}$, c, $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{t}_1$, $\mathbf{t}_2$)
  $\mathbf{c} = F(c)$
  $\mathbf{w}'_1 \equiv \mathbf{a}_1\mathbf{z} - \mathbf{t}_1\mathbf{c} \mod q$, $\mathbf{w}'_2 \equiv \mathbf{a}_2\mathbf{z} - \mathbf{t}_2\mathbf{c} \mod q$
  $c' = H(\lfloor \mathbf{w}'_1 \rceil_{d,q}, \lfloor \mathbf{w}'_2 \rceil_{d,q}, \mu)$
  if $c = c'$ **and** $\mathbf{z} \in \mathcal{R}_{B-U}$ **then**
    return 1
  else
    return 0
  end if
end procedure

Table 5.2 A comparison of the similarities between signature and verification operations in RING-TESLA. Polynomial multiplication and the hash function run on the same number of operations for the same input sizes, where the LHW multiplier requires three computations for signing and only two for verification.

RING-TESLA Modules	Sign	Verify
Polynomial Multiplication	$\mathbf{a}_1\mathbf{y} \mod q$ $\mathbf{a}_2\mathbf{y} \mod q$	$\mathbf{a}_1\mathbf{z} \mod q$ $\mathbf{a}_2\mathbf{z} \mod q$
Keccak Hash Function	$c' \leftarrow H(\cdot)$	$c'' \leftarrow H(\cdot)$
LHW Multiplication	**sc** $\mathbf{e}_1\mathbf{c} \mod q$ $\mathbf{e}_2\mathbf{c} \mod q$	- $\mathbf{t}_2\mathbf{c} \mod q$ $\mathbf{t}_1\mathbf{c} \mod q$

5.3 Parameter Selection for Ring-TESLA

The parameters for the Ring-TESLA signature scheme are already provided in Table 5.1 and were derived by the authors. The way in which the Ring-TESLA parameters are derived is described by Akleylek et al. [2016], Essentially, since the signature scheme is proven to be as hard as ring-LWE, the bit-security of the parameters can be calculated by LWE attacks[3]. The most efficient practical approaches to solving LWE are the embedding approach and the decoding attack. This will now be briefly described.

For the decoding attack, a LWE instance $(\mathbf{A}, \mathbf{As} + \mathbf{e})$ is considered as an instance of the bounded distance decoding problem (BDD). The attack first utilises a lattice reduction algorithm, such as the BKZ algorithm [Chen and Nguyen, 2011], and then uses the nearest plane algorithm proposed by Lindner and Peikert [2011] to find the closest vector $(\mathbf{As})$ to a target vector.

The embedding attack reduces a LWE instance to an instance of the shortest vector problem (SVP). The technique by Albrecht et al. [2013] is then used to define a lattice that contains the error term of a LWE instance. The short error term (a shortest vector) of the constructed lattice in then found via basis reduction techniques such as BKZ.

The embedding attack and decoding attack analysis is shown in Figure 5.1. Parameter selection analysis was undertaken to find hardware-friendly parameters for Ring-TESLA. The inputs to these lattice attack algorithms are the parameters (n, q, σ), with the remaining parameters derived from these. For parameter selection in this case, the value of n is fixed to $n = 512$, the value of σ is looped

[3]The ring-LWE problem is an instantiation of the LWE problem, hence the hardness of ring-LWE (and thus Ring-TESLA) can be calculated by attacks against LWE

over the values $\sigma \in \{1, 2, \ldots, 150\}$, and finally the prime modulus is output[4] with the achieved bit-security versus the two attacks.

The parameter selection analysis results are shown in Figure 5.1, which shows plots of the bit-security levels from the embedding attack (blue) and the decoding attack (red), for all values of σ. A 128-bit reference line is also added, as this is the target security level in this research, which also matches the security of the proposed parameters by Akleylek et al. [2016]. A modulus (q) is output which is associated with a given σ, the plot shows the Hamming weight of this modulus (in black). Interestingly, there is a correlation between the LWE bit-security and the Hamming weight of the modulus. As σ increases, the bit-security of the two attacks also expectedly increases, however there are four major peaks on these plots. These peaks correspond to prime number changes, however as seen by the black Hamming weight line, these peaks are also low Hamming weight primes. At the same time as a drop in the black Hamming weight line, there is also a sharp rise in the bound B, which bounds the size of q. Due to the rise in B, the modulus q is permitted to be smaller and close to a power of 2. This correlation allows for smaller parameters in comparison to those proposed by Akleylek et al. [2016]. The proposed hardware-friendly parameter set (Ring-TESLA-HW) achieves 128-bit security *and* low Hamming weight modulus, using the LWE parameters $n = 512, \sigma = 35$, and $q = 16780289$, and is shown in Table 5.3.

Using low Hamming weight parameters within cryptography is not new; it is used within RSA and ECC [Hoffstein and Silverman, 2003], the NTRU cryptoscheme [Hoffstein and Silverman, 2001], and fully homomorphic encryption [Cao et al., 2016]. Low Hamming weight primes are also used within lattice-based digital signature schemes such as BLISS ($q = 12289 = 11000000000001_2$). The proposed hardware-friendly parameters are shown in Table 5.3. Compared to the

[4]The modulus output is based on a theoretical bound by Akleylek et al. [2016], which depends on σ and the target security level, in this case 128-bits.

Table 5.3 The hardware-friendly parameter set derived for Ring-TESLA, as well as the proposed parameters by Akleylek et al. [2016]. Both provide 128-bit classical security.

Parameters	Ring-TESLA-II	Ring-TESLA-HW
Security	128-bits	128-bits
Lattice dimension n	512	512
Modulus q, ($\lceil \log_2(q) \rceil$)	51750913,(26)	16780289,(25)
Weight of the challenge ω	19	19
Gaussian std. dev. σ	52	35
Dropped bits d	23	22
Error threshold L	2766	1862
Sign/Verify thresholds B, U	$2^{22} - 1,3173$	$2^{22} - 1,2136$
Repetition rate	2.94	2.5
Secret-key size (kilo-bits)	15.36 kb	13.82 kb
Public-key size (kilo-bits)	26.62 kb	25.6 kb
Signature size (kilo-bits)	18.56 kb	18.56 kb

Ring-TESLA-II parameters proposed by Akleylek et al. [2016], the modulus bit length is reduced, which reduces all operand sizes whilst retaining the same 128-bit security level. The next significant changes are in reducing the key and signature sizes, as well as bettering the repetition rate. The implications of the hardware-friendly parameters are shown in Tables 5.4 and 5.5.

5.4 Design Goals and Multipliers

As described in Section 5.1, the main design aim of this research is for compactness, that is consuming less area in comparison to the state-of-the-art. Low-area is a common hardware design goal, which has also been merged into other hardware designs of LBC [Roy et al., 2013a, 2014b, Aysu et al., 2013, Pöppelmann and Güneysu, 2014]. For ideal LBC in general, the polynomial multiplication technique almost always employed is the number theoretic transform (NTT), which is essentially a fast Fourier transform (FFT) over the integers modulo q. A background on NTT polynomial multiplication used within LBC is given in Section 2.6.1. In general, the reason NTTs are preferred is due to the gain in operational run-times

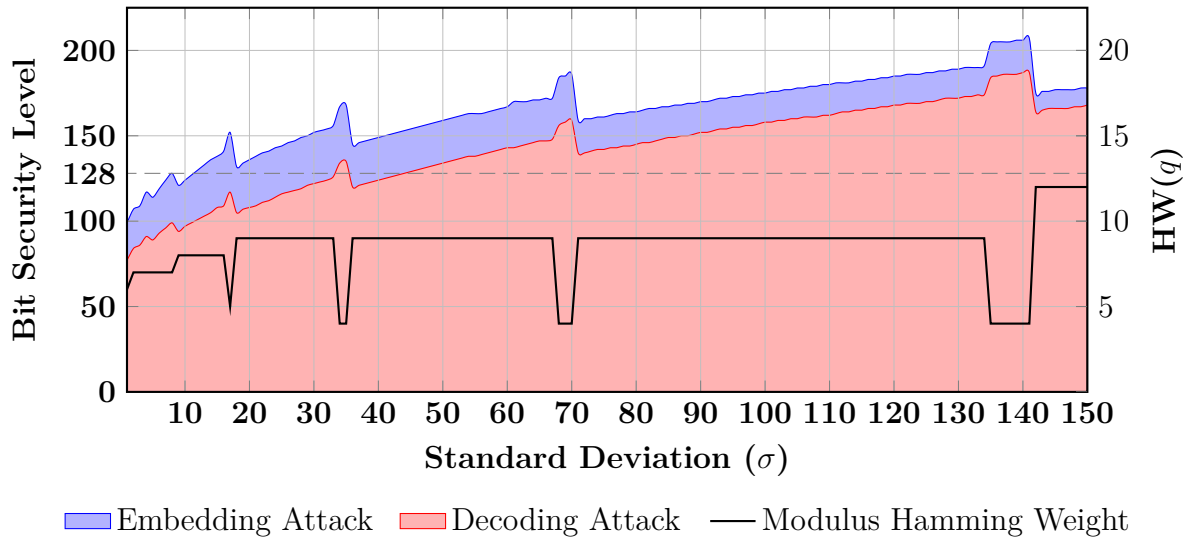


Fig. 5.1 Graphical representation of parameter selection for Ring-TESLA. Security levels provided are via the outputs of the embedding attack and the decoding attack. Also added is a reference line for 128-bits, as well as the hamming weight of the modulus associated with the standard deviation along the x-axis.

from the improvement from quadratic to quasi-linear computational complexity. Additionally, the NTT component also includes the modulo reduction calculation.

However, the gains made in efficiency do come with a number of conditions. Firstly, the NTT multipliers work over a NTT domain, meaning values need to be converted in and out of this domain for calculations. This is an issue for components such as the discrete Gaussian sampler (or even addition/subtraction), as well as issues for public-key or secret-key inputs to the multiplier, which do not work in this domain.

Secondly, the NTT causes problems when considering parameter selection for LBC. Parameter selection is still an open problem in LBC, and NTT is only applicable if n is a power of two and q is a prime satisfying $q \equiv 1 \mod 2n$, meaning two of the three security parameters (n, σ, q) are already fixed, giving significant restrictions on parameter selection. Moreover, it is still an open problem if the choice of multiplicative ring affects the security of a LBC scheme [Lyubashevsky, 2016].

Thirdly, designing a NTT multiplier is non-trivial, and it is usually designed to a specific parameter set to give the best performance. Should a certain parameter set become insecure or a practitioner require scalability, the NTT becomes impractical. This is echoed by Bernstein et al. [2016], arguing alternatives to NTTs should be investigated.

Fourthly, it is not known if the NTT multiplication is susceptible to timing side-channel analysis. Since the run-time of the NTT is usually non-constant, simple timing analysis could give away secret-key information.

Finally, the NTT is the most area consuming hardware component in LBC. For example, in the hardware design of BLISS, the NTT multiplier consumes on average 42% of the hardware resources utilised for signing, similarly 62% for verifying.

Exploring an alternative for polynomial multiplication is therefore important and is thus investigated in this research. The main polynomial multiplication in this research is calculated using Comba multiplication [Comba, 1990] and Barrett modular reduction [Barrett, 1986]. Overall the combination produces a complete modular multiplication component, applicable for polynomial multiplication over $\mathcal{R}_q$ for LBC. In comparison to the NTT, it utilises significantly less hardware resources, meaning a number of parallel multipliers were able to be used to produce several speed variants. The modular multiplication component also offers versatility, meaning the security parameters (n, σ, q) no longer need to be fixed. Moreover, the design is scalable for use with many other parameter sets with little area overhead. For the large area increase however, the NTTs do benefit from higher throughput performance. Additionally, the structured, constant run-time of the Comba multiplier would bypasses any timing analysis. The results of the proposed modular multiplication modules are compared to the NTT in Table 5.4.

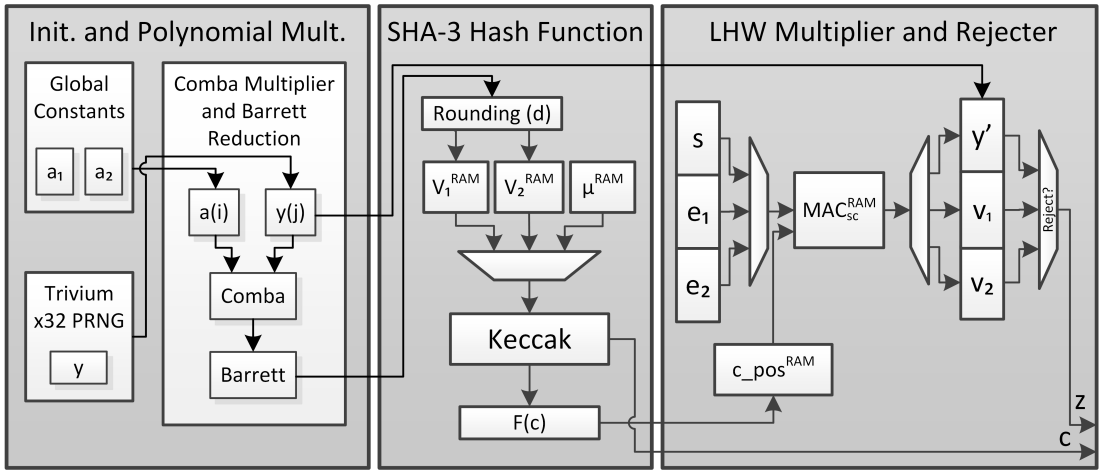


Fig. 5.2 A block diagram of the proposed hardware design for Ring-TESLA Sign.

5.5 Design of Hardware Modules for Ring-TESLA Signing and Verifying

In this section, hardware designs for the Sign and Verify algorithms of the Ring-TESLA DSS are proposed. At a high level and as shown in Table 5.2, the main algorithmic components for both signing and verifying are a polynomial multiplier, a hash function, and a low Hamming weight (LHW) multiplier. Figure 5.2 illustrates the proposed hardware design of the Ring-TESLA signing algorithm. The hardware design for verification is shown in Figure 5.6 and is adapted from the proposed Sign hardware design.

The designs of the Sign and Verify hardware architectures of the Ring-TESLA lattice-based DSS target the Xilinx Spartan-6 FPGA. The research in this chapter is a contribution of separate signing and verifying hardware designs, which both offer compact designs with a variety of performance results. Figure 5.3 shows a high-level overview of the finite-state machine (FSM) of the Ring-TESLA signing hardware design, with details of the polynomials output from those states. Similarly, the Ring-TESLA verification FSM overview is shown in Figure 5.4.

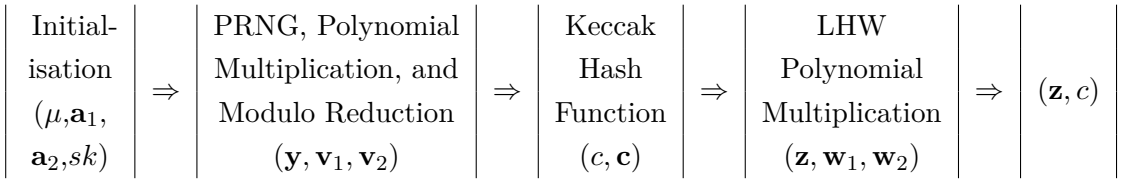


Fig. 5.3 A high-level overview of the hardware design of the signing algorithm of Ring-TESLA, showing the main stages of the finite state machine.

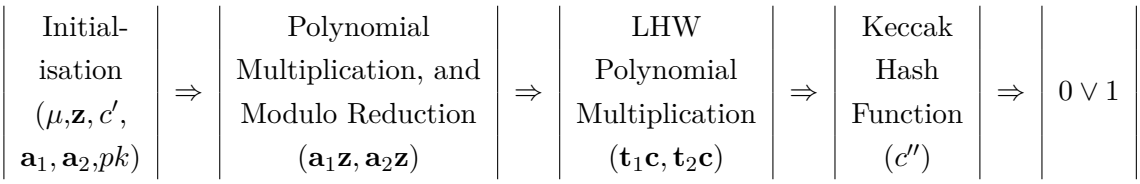


Fig. 5.4 A high-level overview of the hardware design of the verification algorithm of Ring-TESLA, showing the main stages of the finite state machine.

As well as polynomial multiplication, the hardware designs also include the use of a post-quantum secure hash function, a low-area LHW polynomial multiplier, and fast rejection components. Both architectures operate in a pipelined manner (Figure 5.5), such that the critical path is reduced to only the initial polynomial multiplications of $\mathbf{v}_1$ and $\mathbf{v}_2$ (similarly, $\mathbf{a}_1\mathbf{z}$ and $\mathbf{a}_2\mathbf{z}$ for verify). Section 5.5.1 describes the hardware modules that uphold the Sign and Verify architectures, with these overall hardware designs outlined in Section 5.5.2.

5.5.1 Hardware components

Polynomial multiplication is the most expensive module required in the proposed designs in terms of clock cycle count. The most commonly chosen method for modular polynomial multiplication is the number theoretic transform (NTT), which offers fast performance and incorporates the reduction modulo q. However, as discussed in Section 5.4, it is costly in terms of hardware resource usage and there are significant restrictions on the parameter selection when a NTT multiplier is used. Alternatively, traditional multiplication techniques can be employed to carry out the polynomial multiplication operations, which are more compact but

incur an additional latency cost. An additional modular reduction module is also then required.

For the proposed designs, the polynomial multiplication is carried out using a variant of schoolbook multiplication, known as Comba multiplication [Comba, 1990], which achieves an improvement in performance by combining carry handling and reducing write access to memory. The Comba multiplier is particularly suitable for FPGA devices and can exploit the fast arithmetic within the DSP units [Güneysu, 2011]. The multiply-and-accumulate (MAC) operations are computed within each DSP slice until the inner products of the schoolbook multiplier are complete. This approach was particularly attractive since the technique is not restricted by the parameters of Ring-TESLA.

For the modular reduction operation, Barrett modular reduction is employed [Barrett, 1986]. Any generic modulus can be used within Barrett reduction, with only one pre-computation required. Two multiplication units with a maximum of two subtractions are needed in Barrett reduction [Dhem, 1998]. In the proposed designs, an individual Comba multiplier is reused a number of times to carry out the modular reduction. The combination of Comba multiplication and Barrett reduction create an overall generic modular multiplication module, which multiplies polynomials over the ring $\mathcal{R}_q$.

The SHA3 hash function, Keccak, is chosen for the hashing in the proposed Ring-TESLA designs, due to its speed in hardware as well as its post-quantum security [Amy et al., 2016].

A LHW polynomial multiplication module is also designed to compute on the LHW output of the hash function. The LHW multiplier, used in both the Sign and Verify architectures, uses the column-wise schoolbook technique, since the LHW calculations only require a small amount of shift and adds. Column-wise is preferred over row-wise as it requires less storage; this is due to column-wise concentrating on a single output of the resulting polynomial rather than *all*

outputs simultaneously. Hence, incorporating column-wise complements the overall compact design. For LHW calculations, the authors of Ring-TESLA suggest a hybrid NTT/sparse polynomial multiplier, but this would incur a higher cost in hardware resource consumption, and since the LHW calculations are independent of the critical path a low-area (column-wise schoolbook) LHW polynomial multiplier is used.

5.5.2 Signing and Verifying Hardware Designs

Figure 5.2 shows the hardware architecture of the Ring-TESLA signing algorithm. Two dual-port 18 Kb block RAMs (BRAM18) are used to store the global constants $\mathbf{a}_1$ and $\mathbf{a}_2$, which are read-in during the initialisation stage (seen in Figure 5.3). Also during initialisation, an unrolled x32 Trivium component is used to generate uniform random bits for the polynomial $\mathbf{y} \xleftarrow{\$} \mathcal{R}_{q,[B]}$.

For polynomial multiplication, the input of the polynomial $\mathbf{y}$, as well as the input of the polynomials $\mathbf{a}_1$ and $\mathbf{a}_2$, is controlled by two counters which increment on the ready signal of the modular multiplication component. The global constant polynomial selection (that is, $\mathbf{a}_1$ or $\mathbf{a}_2$) is controlled by the FSM. Once each element of $\mathbf{v}_1$ and $\mathbf{v}_2$ is output from the modular multiplication module, the d least significant bits of each are stored in RAM for use in the hash function, where the full values are also stored for use in the rejection stage.

The encoding function $F(c)$ uses the same technique as used by Güneysu et al. [2012], which is also suggested by the authors of Ring-TESLA. The 160-bit string output (c) of the hash function is considered 5-bits at a time; $c_1c_2c_3c_4c_5$. If the leading bit c_1 is 1, then a value of 1 is put in position $c_2c_3c_4c_5$ (this 4-bit value is read as a number between 0 and 15) of a 16-digit string. This converts the 160-bit hash output into a 512-bit string. The 512-bit string is then converted to a LHW polynomial by storing the positions which are 1. This prior knowledge

of a LHW polynomial allows the use of a LHW polynomial multiplier, which only computes for values which are non-zero. The discrete Gaussian distributed (D_σ) secret-keys $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2$ are also LHW since, for instance, the probability a sample $x \leftarrow D_\sigma$ also satisfies $x \in \{-100, 100\}$ (thus 6-7 bits in length) is around 95%.

The LHW polynomial $\mathbf{c}$ is an input into every LHW computation. This includes the calculation of the ciphertext $\mathbf{z} \equiv \mathbf{y} + \mathbf{sc}$, and the variables used to accept or reject the signature, $\mathbf{w}_1 \equiv \mathbf{v}_1 - \mathbf{e}_1\mathbf{c} \mod q$ and $\mathbf{w}_2 \equiv \mathbf{v}_2 - \mathbf{e}_2\mathbf{c} \mod q$. The LHW polynomial multiplier exploits the fact that the hash output polynomial $\mathbf{c}$ has only $\omega = 19$ elements equal to a one, and $(n - \omega) = 493$ zero elements. For simplicity, the LHW polynomial $\mathbf{c}$ is expressed by the positions with an element equal to one. This saves on run-time by automatically ignoring all zero coefficient, which for these parameters make up 96% of $\mathbf{c}$. Once the LHW multiplier has calculated the signature polynomial $\mathbf{z}$, the polynomials $\mathbf{w}_1$ and $\mathbf{w}_2$ are then calculated sequentially.

Once a coefficient is output from the LHW multiplier, it is processed by the rejecter, which checks whether its size is valid for output. Rejecting a signature element-wise is preferable to minimise run-time.

The secret-key values ($\mathbf{s}$, $\mathbf{e}_1$, and $\mathbf{e}_2$) are stored in a single-port distributed RAM. Then, for each element in $\mathbf{s}$, $\mathbf{e}_1$, and $\mathbf{e}_2$, the multiplier accumulates the inner products using a MAC unit. Once each column-wise element is calculated, it is added/subtracted to/from the corresponding values in $\mathbf{v}_1$, $\mathbf{v}_2$, or $\mathbf{y}'$, and the final values are checked with their rejection conditions seen in line 8 in Algorithm 5.2 (similarly line 5 in Algorithm 5.3), where only the signature $\mathbf{z}$ and hash-string c are stored. This approach is advantageous since the rejection validity is calculated instantly and in parallel to the next LHW element calculation.

Both the Sign and Verify architectures operate in two pipelined stages (pipeline illustrated in Figure 5.5), due to the latency of the calculations of $\mathbf{v}_1$ and $\mathbf{v}_2$. For the first signature, $\mathbf{y}$ is generated during initialisation when the global constants

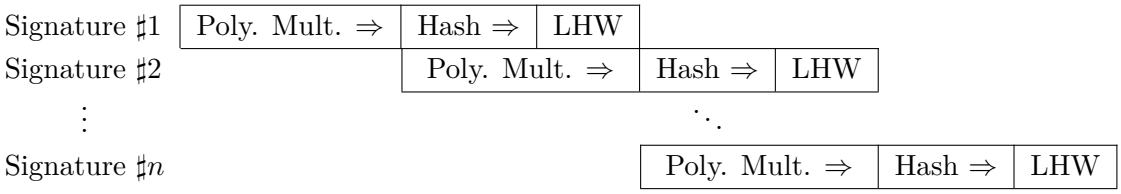


Fig. 5.5 A high-level overview of the pipeline incorporated within the sign (and verify) algorithm of Ring-TESLA.

are read in, with an additional $\mathbf{y}'$ polynomial (see Figure 5.2) generated during the calculation of $\mathbf{v}_1$ and $\mathbf{v}_2$. Once these calculations have finished, $\mathbf{y}$ is swapped with $\mathbf{y}'$, so the calculations of $\mathbf{v}_1$ and $\mathbf{v}_2$ for the next signature can begin again. This removes the hashing and LHW calculations from the critical path and cycle count, as well as savings for generating a new polynomial $\mathbf{y}$.

Verify

Verification (shown in Figure 5.6) operates in a similar fashion to signing, with differences mainly seen in the FSM, seen in Figure 5.4. Verification, as in signing, includes (for the same bit lengths) polynomial multiplication and modular reduction, a hash function, and LHW polynomial multiplication. The Comba/Barrett polynomial multiplier calculates $\mathbf{a}_1\mathbf{z} \mod q$ and $\mathbf{a}_2\mathbf{z} \mod q$, with $\mathbf{a}_1, \mathbf{a}_2$, and $\mathbf{z}$ being stored in BRAM. As with signing, the resulting values are stored in single-port distributed RAM and are updated after subtraction with the LHW calculations of $\mathbf{t}_1\mathbf{c}$ and $\mathbf{t}_2\mathbf{c}$. The final results are input into the hash function, where the signature is validated if the hash value matches the hash-string input from the signature. The similarities between signing and verifying for Ring-TESLA can be seen more explicitly in Table 5.2.

5.5.3 Parallelised multipliers for accelerated performance

A number of hardware designs are proposed for the Ring-TESLA Sign and Verify algorithms that offer a trade-off between area and performance. The lowest area

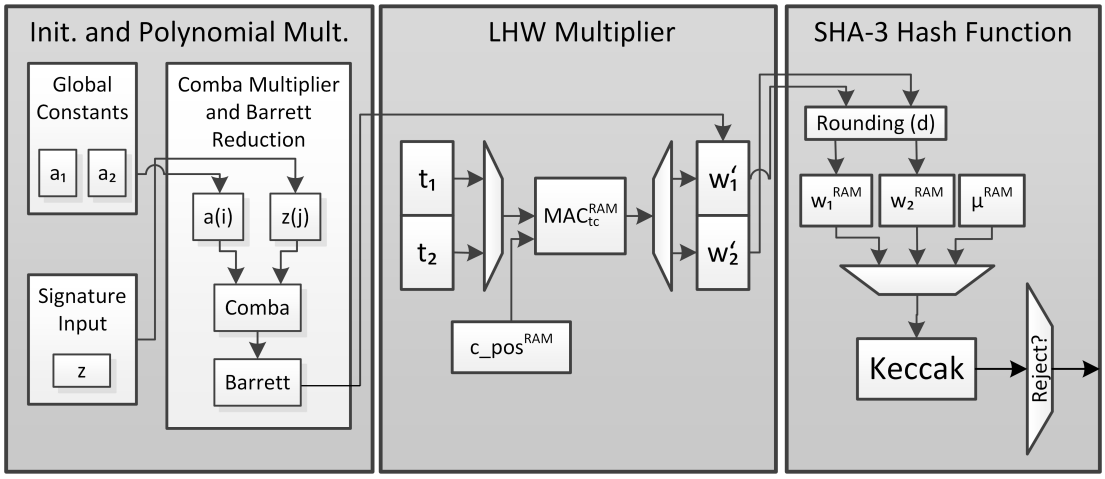


Fig. 5.6 A block diagram of the proposed hardware design for Ring-TESLA Verify. Global parameters are stored in BRAM18.

designs, named SB-I Sign and SB-I Verify, offer reduced area consumption at the cost of additional latency. In these designs, a standard Comba multiplier with one Barrett modular reduction unit is employed. The Barrett modular reduction unit is carried out in parallel while the Comba multiplier is used to minimise latency of the modular multiplication unit. These designs target low area applications, where a slower performance may be acceptable.

The bottleneck in Ring-TESLA is polynomial multiplication. Three additional designs are proposed to enhance the practicality, whereby the polynomial multiplication unit is optimised. To improve latency multiple Comba multipliers are employed in parallel, exploiting the FPGA DSP slices, which increases area consumption. These are named SB-II, SB-IV, and SB-VIII, which reflect the number of parallel Comba multipliers within each design (that is, two, four, and eight parallel multipliers). To minimise latency, in all proposed designs only one modular reduction is employed, running in parallel with the multiple Comba multiplier units. Overall, each polynomial multiplier requires 2 DSPs, with the modular reduction component requiring 4 DSPs.

Table 5.4 shows the hardware resource usage and clock cycle count of the proposed modular multipliers for polynomial multiplication within Ring-TESLA.

Table 5.4 Post-place and route results of the proposed modular multiplication unit, targeting a Spartan-6 (S6) LX25 FPGA. Parameters used are (n, q), with two results for the same multiplier with the original Ring-TESLA parameters (with $q = 51750913$) and for the hardware-friendly set (with $q = 16780289$). BLISS NTT results provided for comparison, GLP multiplier results are not available. Results are also provided from the polynomial multipliers by Pöppelmann and Güneysu [2012] (PG), Aysu et al. [2013] (APS), and Du and Bai [2016] (DB). Results by Du and Bai [2016] do not provide the modulus used, but it is assumed to be 65537. Results with $^{\mathrm{F}}$ indicate multipliers that use Fermat primes.

Mult. Type	**Parameters** (n, q)	**Device**	**LUT/FF/SLICE**	**BRAM/DSP**	**MHz**	**Cycles**	**Ops/s**
SB-I	(512,51750913)	S6LX25	1016/790/317	0/6	192	917504	210
SB-I	(512,16780289)	S6LX25	990/772/316	0/6	192	917504	210
SB-II	(512,51750913)	S6LX25	1357/1162/395	0/8	196	458752	428
SB-II	(512,16780289)	S6LX25	1251/1103/373	0/8	196	458752	428
SB-IV	(512,51750913)	S6LX25	2374/1731/560	0/12	190	229376	829
SB-IV	(512,16780289)	S6LX25	2226/1664/516	0/12	190	229376	829
SB-VIII	(512,51750913)	S6LX25	3488/2637/880	0/20	188	114688	1640
SB-VIII	(512,16780289)	S6LX25	3216/2489/804	0/20	188	114688	1640
$\mathrm{NTT^F}$-PG	(512,65537)	S6LX100	1585/1205/615	4/1	196	10014	19572
NTT-PG	(512,5941249)	S6LX100	3228/2263/1145	7/4	193	10174	18969
NTT-BLISS	(512,12289)	S6LX25	2557/2707/835	5/1	145	9307	15579
$\mathrm{NTT^F}$-APS	(512,65537)	S6LX100	490/532/211	2/2	184	> 12800	< 14375
$\mathrm{NTT^F}$-APS	(512,65537)	S6LX100	632/535/256	2/3	224	> 12800	< 17500
$\mathrm{NTT^F}$-DB	(512,65537)	S6LX100	562/562/239	4/1	196	10014	19572

Clock cycle counts are given for the multiplication of one polynomial, with $n = 512$. Table 5.4 also provides NTT results, where available, for comparison. It should be noted that NTT multipliers are significantly more efficient when Fermat ($\mathrm{NTT^F}$) [Agarwal and Burrus, 1974] or Mersenne [Rader, 1972] primes are used, in comparison to other prime numbers. This is due to simplifications, such as shifting used in the butterfly unit (instead of multiplication), no storage requirements for twiddle factors, and simpler modular reduction [Baktir and Sunar, 2006, Blahut, 2010b].

5.6 Results

The proposed architectures are implemented using the Xilinx ISE Design Suite 14.7 synthesis tool. The target device is a Xilinx Spartan-6 FPGA (S6 LX25). Table 5.5 shows the post-place and route (PAR) results for the proposed hardware

designs of both Sign and Verify of the Ring-TESLA signature scheme. These designs fit comfortably on the low end FPGA. As expected, the optimised designs (SB-II, SB-IV, and SB-VIII) have reduced latency in comparison to SB-I, at the cost of additional area usage. Results indicate that up to 785 operations per second for Sign and 776 operations per second for Verify can be achieved by the proposed designs, with an associated low area cost.

The proposed designs are compared to the hardware designs of existing DSSs; currently used in practice (RSA/ECDSA) and alternative lattice-based algorithms (GLP and BLISS), as seen in Table 5.5. The variety of results for the proposed hardware designs stems from changes in the FSM (Figure 5.3), scheduling, and the integration of multiple polynomial multiplication modules.

Table 5.5 Results of the proposed hardware designs for Ring-TESLA with 128-bit parameters proposed by [Akleylek et al., 2016] (Ring-TESLA-II) and the ones generated in this research (Ring-TESLA-HW). Also added is a summary of other DSSs, including GLP-I [Güneysu et al., 2012], BLISS-I [Pöppelmann et al., 2014], RSA, and ECDSA [Glas et al., 2011] (results taken from [Pöppelmann et al., 2014]).

Operation, Configuration	**Security**	**Device**	**LUT/FF/SLICE**	**BRAM/DSP**	**MHz**	**Cycles**	**Ops/sec**	**Ops/sec/S**
Ring-TESLA-II (Sign, SB-I)	128-bits	S6 LX25	4447/3345/1257	3/6	190	1835540	104	0.08
Ring-TESLA-II (Sign, SB-II)	128-bits	S6 LX25	4828/3790/1513	4/8	196	917771	214	0.14
Ring-TESLA-II (Sign, SB-IV)	128-bits	S6 LX25	5071/3851/1503	4/12	187	458891	408	0.27
Ring-TESLA-II (Sign, SB-VIII)	128-bits	S6 LX25	6848/5457/2254	4/20	180	229446	785	0.35
Ring-TESLA-II (Verify, SB-I)	128-bits	S6 LX25	3714/3023/1172	3/6	188	1835540	102	0.09
Ring-TESLA-II (Verify, SB-II)	128-bits	S6 LX25	3917/3253/1238	3/8	194	917771	212	0.16
Ring-TESLA-II (Verify, SB-IV)	128-bits	S6 LX25	4793/3939/1551	3/12	186	458891	406	0.25
Ring-TESLA-II (Verify, SB-VIII)	128-bits	S6 LX25	6473/5582/2103	3/20	178	229446	776	0.37
Ring-TESLA-HW (Sign, SB-I)	128-bits	S6 LX25	3860/2820/1220	3/6	190	1835540	104	0.08
Ring-TESLA-HW (Sign, SB-II)	128-bits	S6 LX25	4185/3175/1147	4/8	196	917771	214	0.19
Ring-TESLA-HW (Sign, SB-IV)	128-bits	S6 LX25	4722/3729/1438	4/12	187	458891	408	0.28
Ring-TESLA-HW (Sign, SB-VIII)	128-bits	S6 LX25	6331/5230/2155	4/20	180	229446	785	0.36
Ring-TESLA-HW (Verify, SB-I)	128-bits	S6 LX25	3457/2593/1146	3/6	188	1835540	102	0.09
Ring-TESLA-HW (Verify, SB-II)	128-bits	S6 LX25	3663/2796/1210	3/8	194	917771	212	0.16
Ring-TESLA-HW (Verify, SB-IV)	128-bits	S6 LX25	4394/3352/1473	3/12	186	458891	406	0.27
Ring-TESLA-HW (Verify, SB-VIII)	128-bits	S6 LX25	6113/4761/2103	3/12	178	229446	776	0.37
GLP (Sign, Schoolbook x3)	80-bits	S6 LX16	7465/8993/2273	30/28	162	-	931	0.41
GLP (Verify, Schoolbook x3)	80-bits	S6 LX16	6225/6663/2263	15/8	158	-	998	0.44
BLISS (Sign, NTT)	128-bits	S6 LX25	7193/6420/2291	6/5	139	15864	8761	3.82
BLISS (Verify NTT)	128-bits	S6 LX25	5065/4312/1687	4/3	166	16346	17101	10.14
RSA (Sign)	103-bits	V5 LX30	3237 slices	7/17	200	-	89	0.03
ECDSA (Sign)	128-bits	V5 LX110	32299 LUT/FF pairs	10/37	139	-	-	-
ECDSA (Verify)	128-bits	V5 LX110	32299 LUT/FF pairs	10/37	110	-	-	-

The Ring-TESLA results significantly outperform the comparable classical results for RSA and ECDSA. The RSA hardware design is bettered in terms of operations per second, with Ring-TESLA providing significantly more security (103-bits vs. 128-bits). In comparison to ECDSA, the proposed hardware designs are significantly lower in FPGA area consumption, using less BRAMs and DSPs, and also operating at a much higher frequency.

The proposed designs also compete with existing lattice-based DSSs. The GLP Sign hardware design is significantly larger than even the largest design for Ring-TESLA Sign (SB-VIII), additionally consuming more than 7x more BRAMs and 8 more DSPs, whilst operating at a lower frequency. These results are also echoed when comparing the Verify results for GLP and Ring-TESLA, with Ring-TESLA consuming less FPGA area resources, 5x less BRAMs, and operating with a higher frequency. Ring-TESLA also provides more bit security than GLP, with GLP only providing 80-bit security compared to 128-bit with Ring-TESLA. Operations per second results are in general better for GLP compared to Ring-TESLA.

Compared to the BLISS Sign architecture, all proposed Ring-TESLA hardware designs offer more compact hardware designs. Most of the Verify designs for Ring-TESLA also compete with BLISS in terms of area consumption (all except SB-VIII). This is despite the much larger parameters for Ring-TESLA in comparison to BLISS; with the lower bit-lengths in BLISS (14-bits) the hardware designs are able to exploit DSPs (maximum 18-bits) on the Spartan 6 [Xilinx, 2011]. Additionally, BLISS requires one polynomial multiplication calculation per signature or verification, with Ring-TESLA requiring two. After using 8 parallel multipliers for Ring-TESLA SB-VIII, the area saving benefits are lost and NTT becomes favourable in terms of area *and* throughput performance. Due to the usage of the NTT in BLISS, the operations per second is significantly higher in

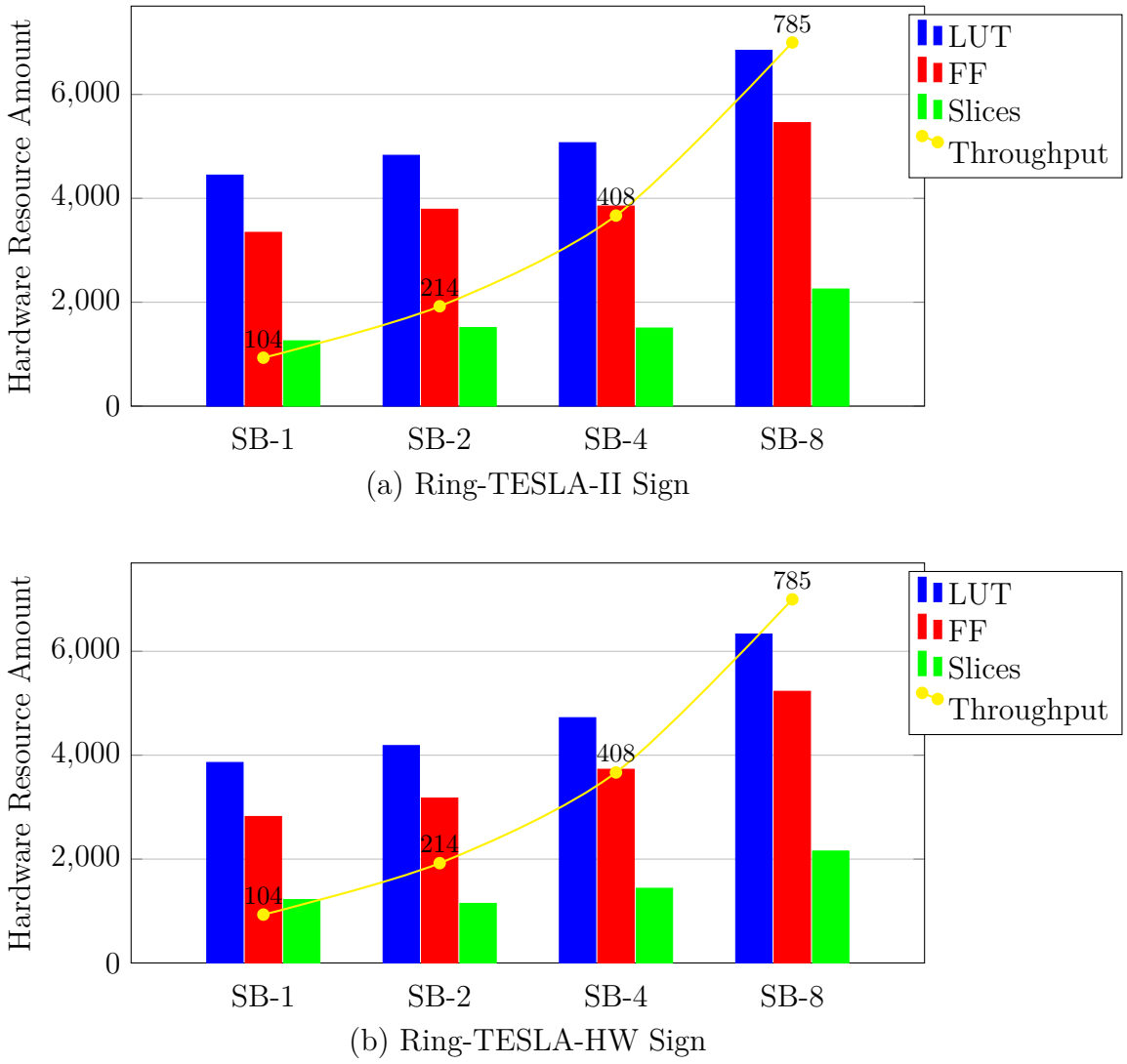
comparison to Ring-TESLA, this can also be seen in the much lower clock cycle counts. Both schemes offer 128-bit security.

Generating hardware-friendly parameters was a option due to Ring-TESLA being based on ring-LWE. The hardware-friendly parameters generated meant that the architecture is more compact in comparison to the original parameter set. These savings were mainly gained by a change in modulus, which is more than 3x smaller and importantly 25-bits instead of 26-bits. This has meant operands sizes are slightly smaller throughout the entire design, thus hardware resource savings are seen in all areas of the architecture. This is illustrated in Figures 5.7 and 5.8, which show the FPGA hardware resource consumption for the original parameter set (Ring-TESLA-II) and the hardware friendly parameter set (Ring-TESLA-HW). The results also show that, as hardware resource consumption increases with additional multipliers, throughput similarly improves.

5.7 Conclusion

In this chapter, low area hardware designs of signing and verifying for a lattice-based DSS is proposed. The DSS is Ring-TESLA, a more secure lattice-based scheme compared to previous research (GLP/BLISS), at the cost however of larger parameters and twice as many multiplications.

Two separate hardware designs have been presented which perform the Sign and Verify operations of Ring-TESLA. Furthermore, each hardware design offers variations in performance as they exploit one, two, four, or eight parallel polynomial multipliers. The design goal of these hardware designs are for low-area, with even the most area expensive design (which utilises 8 multipliers) lower in area consumption than the state-of-the-art. Indeed, the area resource consumption of the proposed designs is less than the state-of-the-art with regards to LUTs, FFs, and slices as well as BRAM and DSP usage. This is despite the larger parameter



(a) Ring-TESLA-II Sign

(b) Ring-TESLA-HW Sign

Fig. 5.7 A graphical depiction of the hardware resource consumption of the proposed Ring-TESLA Sign architecture, provided for all parallel multiplier options.

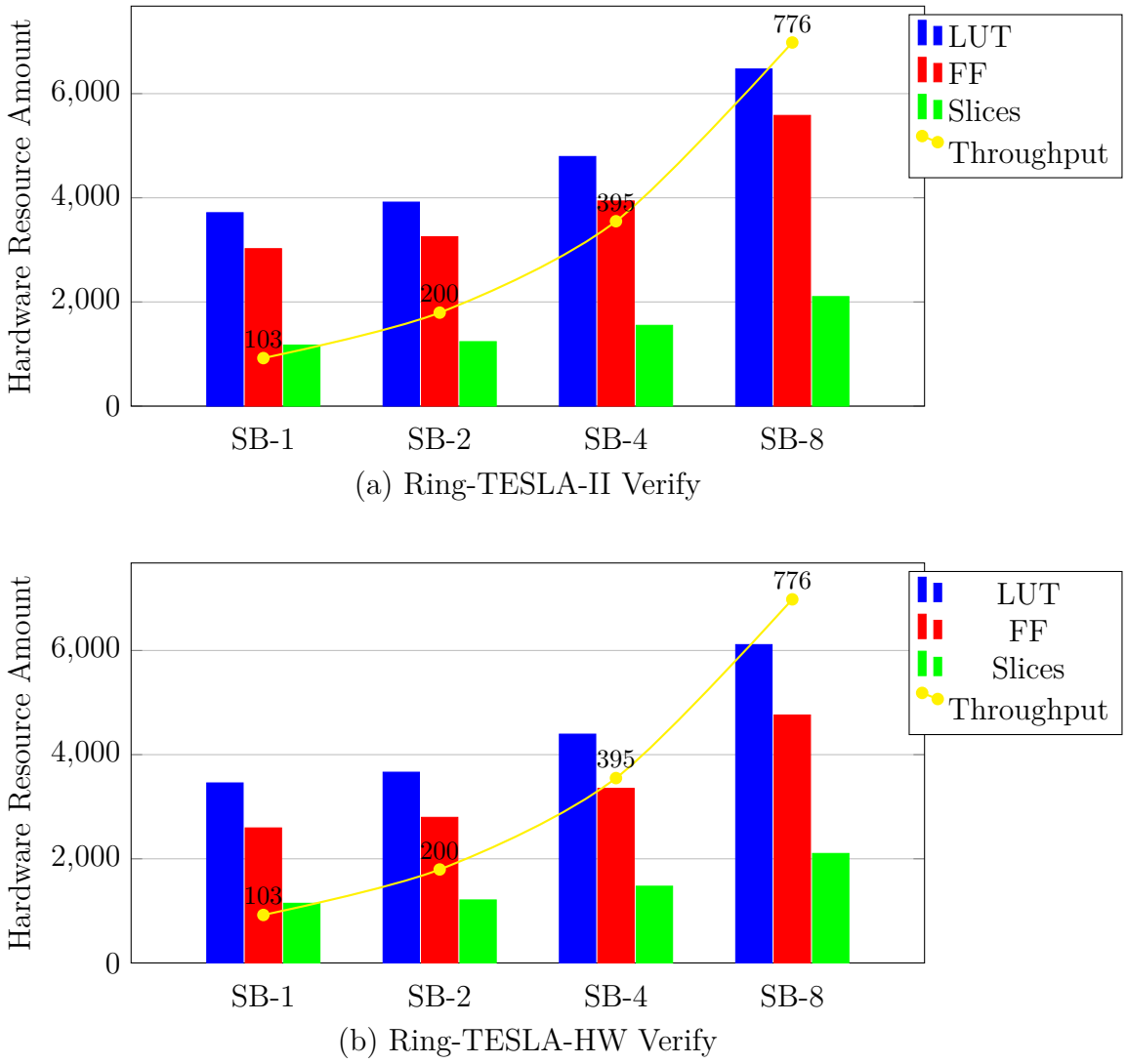


Fig. 5.8 A graphical depiction of the hardware resource consumption of the proposed Ring-TESLA Verify architecture, provided for all parallel multiplier options.

sizes of Ring-TESLA in comparison to alternative lattice-based DSSs, GLP and BLISS. Additionally, Ring-TESLA has the appealing qualities of worst-case hardness. Hardware-friendly parameters were also generated to further reduce the area consumption of the Ring-TESLA hardware designs.

Post-publication, a security flaw was found in the Ring-TESLA signature scheme. As already discussed, this does not lead to an actual attack on the scheme. The proposed hardware design is significantly more generic than other lattice-based signature hardware designs, mainly due to the use of a generic polynomial multiplier. Considering that this flaw will likely be fixed by the authors, a generic hardware design will mean a significantly faster turn-around for updating the hardware design and generating new results.

The results show that lattice-based cryptosystems can be reduced and optimised to fit comfortably on a low-end Spartan 6 FPGA. Furthermore, the performance achieved is practical for low area or high security applications, where tighter security reductions are desirable.

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion Summary

This thesis presents research that advances the field of practical lattice-based cryptography, by considering highly secure encryption and signature schemes, and efficient discrete Gaussian samplers protected against timing analysis with a test suite for its correctness. Furthermore, the research aims to bring lattice-based cryptography closer to being deployed in the real-world, creating alternative hardware designs for discrete Gaussian samplers, an encryption scheme, and a signature scheme, which at least competes with or in most cases outperforms previous research. With the additional protection against quantum computers, the research also provides an interesting alternative to classical cryptography currently used, such as RSA and ECC.

For all the research in this chapter, the proposed hardware designs are optimised for FPGAs and improve over previous research in lattice-based cryptography, elliptic-curve cryptography, and RSA. Moreover, the proposed hardware designs outperform previous research either in terms of a reduction in the area consumption, an improvement in throughput (operations per second), a reduction in latency, or a combination of these. Lattice-based cryptography is indeed practical, which

has been demonstrated in this thesis, and should now be considered as a viable replacement or alternative to RSA and ECC.

The performance improvements have been achieved by considering alternatives to previous research, such as considering alternative cryptoschemes and optimisations at an algorithmic, architectural, and device-specific level.

The conclusions from Chapter 3, Chapter 4, and Chapter 5, are now presented in the following sections, with Section 6.2 detailing several future research avenues for lattice-based cryptography.

6.1.1 Chapter 3

In this chapter, the first comprehensive evaluation of the discrete Gaussian sampling component, with respect to hardware, is undertaken. Novel FPGA hardware designs are proposed for all the major techniques for sampling over the discrete Gaussian distribution. The proposed samplers are also designed to operate in independent time, and hence can protect against timing side-channel analysis. Additionally, the samplers are designed to operate for both encryption and signature parameters, with results given in which BRAMs are used or not used. Recommendations are then provided, based on the results of the sampler designs, for encryption and signature purposes.

In general, the results for Bernoulli sampler [Ducas et al., 2013] hardware designs are lower in area consumption, achieve a high frequency, and have a higher throughput rate than previous research. Only Pöppelmann and Güneysu [2014] achieve lower area for encryption parameters, but the results are hindered by a 20x larger latency. The hardware designs are also the first to demonstrate a time-independent Bernoulli sampler.

The results for the Knuth-Yao sampler [Knuth and Yao, 1976] hardware designs compete with the area consumption of previous research [Roy et al., 2013b, 2014a]

and better them in terms of throughput. The first time-independent hardware architecture for Knuth-Yao is also proposed, which also betters all previous research in terms of its performance.

The cumulative distribution table (CDT) [Peikert, 2010] design provides the best overall performance compared to all the proposed samplers, for both encryption and signatures, and operates in constant time. The encryption and signature results significantly better previous research in terms of FPGA area consumption. For encryption parameters, the frequency is also drastically improved over all previous research. Du and Bai [2015] have better throughput results in comparison to the proposed hardware designs, but do not operate in independent-time. For signature parameters, the area consumption is significantly reduced compared to previous research whilst being the first to operate in independent-time for signatures.

6.1.2 Chapter 4

In this chapter, the first hardware evaluation of standard lattices in undertaken. As described in Section 4.2.2, the standard lattice assumption has better security assurances compared to the ideal lattice assumption, and was believed to be infeasible for hardware. However, the research in this chapter shows that standard lattice-based encryption is feasible for the FPGA platform, and competes well with the equivalent encryption scheme over ideal lattices. The algorithm of the encryption scheme is adapted, so that all the operations run generically and therefore components can be reused. For arithmetic, the hardware design exploits the DSP on the FPGA, to efficiently compute the 12-bit multiplication. The results compete with results for ring-LWE encryption (that is, encryption over ideal lattices) by Göttert et al. [2012]. The results also compete with other ring-LWE encryption hardware designs by Pöppelmann and Güneysu [2013], in

terms of area consumption, and by Pöppelmann and Güneysu [2014], in terms of throughput.

The results illustrate that standard lattices could be considered for real-world implementations. Previously, standard lattices were thought to be too impractical for the real-world. However, the proposed hardware designs for standard-LWE encryption compete with ring-LWE encryption, whilst offering more confidence in security. The results therefore suggest that they could be considered for real-world applications, especially for high security purposes.

6.1.3 Chapter 5

In this chapter, the first hardware designs of the Ring-TESLA [Akleylek et al., 2016] lattice-based digital signature scheme are proposed. These are also the first results which compare to the only other state-of-the-art lattice-based digital signature scheme in hardware at 128-bit security level [Pöppelmann et al., 2014]. These comparative results by Pöppelmann et al. [2014] are hardware designs of the BLISS [Ducas et al., 2013] signature scheme. The Ring-TESLA signature scheme, in comparison to BLISS, is considered more secure since the parameters for BLISS are not chosen directly from their security reduction, and also offers worst-case hardness. Post-publication, a security flaw was found in the Ring-TESLA signature scheme. As already discussed, this does not lead to an actual attack on the scheme. Improvements to Ring-TESLA are currently in progress.

The main design goal was to provide compact/low-area consumption, thus alternatives to NTT multiplication were required, as the NTT consumes a large amount of FPGA area resources whilst also restricting theoretical parameter selection. Instead, an optimised schoolbook multiplier [Comba, 1990] is employed which exploits the FPGA DSPs. The overall Sign and Verify hardware architectures operate generically and can be used for different parameter sets, that do not need

to be NTT-friendly. The area consumption of the proposed Ring-TESLA Sign and Verify hardware designs is significantly less than previous research [Güneysu et al., 2012, Pöppelmann et al., 2014] in LBC. The Ring-TESLA results with the highest throughput also compete with the throughput of GLP [Güneysu et al., 2012], despite its low area design goal. The results also outperform classical hardware designs of RSA and ECDSA [Howe et al., 2015].

The results illustrate that alternative lattice-based signature schemes are available to compete with the state-of-the-art hardware designs of BLISS [Pöppelmann et al., 2014]. Additionally, the proposed designs could be considered in cases requiring low-area and/or higher security situations.

6.2 Topics For Future Research

Despite lattice-based cryptography being around for over 20 years, there is still many areas of future research that need to be investigated. The following sections discuss possible topics for future research in the area of lattice-based cryptography.

6.2.1 Side-Channel Analysis of Lattice-Based Cryptosystems

As lattice-based DSSs become more practical and publicly available, further attack vectors like side-channel analysis (SCA) [Kocher et al., 1999] have to be considered. As described in Section 2.7, attacks such as timing and fault injection attacks, power, electro-magnetic analysis, and advanced machine learning-based attacks are serious threats to many real-world implementations. Highly secure algorithms such as the Advanced Encryption Standard (AES) or ECC are easily breakable if an attacker has physical access to the security device, unless appropriate countermeasures are built in.

So far there has been very little research conducted on the vulnerabilities of lattice-based cryptographic implementations to physical attacks; some have focused on SCA attack results [Bruinderink et al., 2016, Pessl, 2016] whereas others have focused on building lattice-based cryptoschemes [Reparaz et al., 2015b, 2016] or its components [Howe et al., 2016a, Khalid et al., 2016] that protect against SCA attacks.

It is anticipated that there may be a particular vulnerability with respect to algorithmic parts with variable runtime. In particular, the most important components within lattice-based cryptography; the NTT multiplier, the discrete Gaussian sampler, and any rejection sampling used, could be susceptible to side-channel analysis.

6.2.2 Alternatives in Lattice-Based Cryptography

An interesting route for future research for standard lattice-based encryption, seen in Chapter 4, would be to exploit more of the FPGA and to increase the throughput of the encryption scheme. More specifically, parallelising the multiplication operations and therefore utilising more DSP slices would significantly increase throughput, roughly n-times using n DSP slices. Additionally, other standard lattice-based cryptoschemes also exist, such as digital signature schemes, and it would be interesting to see how these compare against their equivalent ideal lattice-based cryptoschemes.

One of the most time consuming components for hardware implementations of lattice-based cryptography is currently polynomial multiplication. Making this stage efficient has been well studied [Pöppelmann and Güneysu, 2012, Göttert et al., 2012, Aysu et al., 2013, Roy et al., 2013a, Ducas et al., 2013], with most instantiations adopting the NTT multiplier. However, alternatives to the NTT are viable, which would not restrict parameter selection, from specialist techniques by

Karatsuba and Ofman [1963] ($\mathcal{O}(n^{\log 3})$), Cooley and Tukey [1965] ($\mathcal{O}(n \log n)$), Pollard [1971a] ($\mathcal{O}(n \log n)$), and Moenck [1976] ($\mathcal{O}(n \log n)$). Optimising such a stage is arguably the most critical in hardware due to the computationally intensive operations, as such, this is still an important focus for research for implementations on larger devices [Pöppelmann and Güneysu, 2012, Pöppelmann and Güneysu, 2013, Göttert et al., 2012, Aysu et al., 2013, Ducas et al., 2013] and on lightweight devices [Boorghany and Jalili, 2014a, Pöppelmann and Güneysu, 2014, Oder et al., 2014, Pöppelmann et al., 2015].

Considering the extensions that are required for IoT technologies, lattice-based cryptography should be considered for a variety of different platforms. BLISS shows very good performance and is thus a candidate for integration into other constrained systems and devices like smart cards and microcontrollers [Boorghany and Jalili, 2014a]. Integrating BLISS, as well as ring-LWE encryption [Lyubashevsky et al., 2013a], with respect to highly-optimised software is also a possible area for future work (similar to the research by Güneysu et al. [2013]).

An interesting area of future research would be to evaluate the practical implications of the compression algorithms and side-channel countermeasures by Saarinen [2017] or optimisations to BLISS by Ducas [2014b].

Recently, there have been proposals for a discrete Gaussian sampler by Micciancio and Walter [2017] and a digital signature scheme by Ducas et al. [2017], both of which show promise in software to compete with the current state-of-the-art. An exciting future research field will be to investigate their performance in hardware, especially in a combined design.

6.2.3 Theoretical Extensions Within Lattice-Based Cryptography

A module which is one of the more computationally expensive in hardware is the discrete Gaussian sampling stage. This was extensively researched in Chapter 3. Due to theoretical restrictions, the discrete Gaussian distribution is essential for most lattice-based cryptoschemes. However, for the "New Hope" key exchange protocol [Alkim et al., 2016], a much simpler distribution, the binomial distribution, is required which would alleviate most of the hardware resources required for discrete Gaussian samplers.

Further theoretical research is also needed for parameters (and security analyses) of these lattice-based cryptoschemes. Rückert and Schneider [2010] investigate this but it is still very much an open problem within lattice-based cryptography. Further investigations into making parameter selection more explicit in lattice-based cryptography would make implementations easier for practitioners as well as increasing the security confidence for cryptanalysts.

6.2.4 The Quantum Random Oracle Model

An interesting area of theoretical research looks into the security of DSSs in the quantum world. Specifically, relating to the DSSs that use random oracle constructions and whether they are still secure to a quantum adversary. Although making the DSSs less efficient, schemes by Gentry et al. [2008] and Lyubashevsky [2012] are respectively shown by Boneh and Zhandry [2013] and Dagdelen et al. [2013] to be secure to such an adversary, creating the quantum random oracle model. This could also motivate an important area for future research, such as proving security for more DSSs to a quantum adversary or possibly creating a

generic technique, that could turn a DSS secure in the random oracle model to that in the quantum random oracle model.

References

Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In *EUROCRYPT*, pages 418–433, 2002.

Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In *EUROCRYPT*, pages 572–590, 2012.

RC Agarwal and C Burrus. Fast convolution using fermat number transforms with applications to digital filtering. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 22(2):87–97, 1974.

Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *STOC*, pages 99–108, 1996.

Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 284–293, New York, NY, USA, 1997. ACM. ISBN 0-89791-888-6. doi: 10.1145/258533.258604. URL http://doi.acm.org/10.1145/258533.258604.

Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001. ISBN 1-58113-349-9.

Sedat Akleylek, Nina Bindel, Johannes A. Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. An efficient lattice-based signature scheme with provably secure instantiation. In *AFRICACRYPT*, pages 44–60, 2016. doi: 10.1007/978-3-319-31517-1_3. URL http://dx.doi.org/10.1007/978-3-319-31517-1_3.

Martin Albrecht. Discrete Gaussian samplers over lattices. http://doc.sagemath.org/html/en/reference/\penalty\z@stats/sage/stats/distributions/discrete_gaussian_lattice.html, 2014. Accessed: 21.07.2015.

Martin R Albrecht, Robert Fitzpatrick, and Florian Göpfert. On the efficacy of solving LWE by reduction to unique-SVP. In *International Conference on Information Security and Cryptology*, pages 293–310. Springer, 2013.

Erdem Alkim, Nina Bindel, Johannes A. Buchmann, and Özgür Dagdelen. TESLA: tightly-secure efficient signatures from standard lattices. *IACR Cryptology ePrint Archive*, 2015:755, 2015. URL http://eprint.iacr.org/2015/755.

Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343, 2016. URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim.

Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. Revisiting tesla in the quantum random oracle model. In *International Workshop on Post-Quantum Cryptography*, pages 143–162. Springer, Cham, 2017.

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

Matthew Amy, Olivia Di Matteo, Vlad Gheorghiu, Michele Mosca, Alex Parent, and John Schanck. Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3. *IACR Cryptology ePrint Archive*, 2016:992, 2016. URL http://eprint.iacr.org/2016/992.pdf.

Ross Anderson, Mike Bond, Jolyon Clulow, and Sergei Skorobogatov. Cryptographic processors-a survey. *Proceedings of the IEEE*, 94(2):357–369, 2006.

Theodore W. Anderson and Donald A. Darling. Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes. *The Annals of Mathematical Statistics*, 23(2):193–212, 06 1952.

Theodore W. Anderson and Donald A. Darling. A test of goodness of fit. *Journal of the American Statistical Association*, 49(268):765–769, 1954.

Aydin Aysu, Cameron Patterson, and Patrick Schaumont. Low-cost and area-efficient fpga implementations of lattice-based cryptography. In *HOST*, pages 81–86, 2013.

Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. pages 28–47, 2014a.

Shi Bai and Steven D. Galbraith. An improved compression technique for signatures based on learning with errors. In *CT-RSA*, pages 28–47, 2014b.

Selçuk Baktir and Berk Sunar. Achieving efficient polynomial multiplication in fermat fields using the fast fourier transform. In *Proceedings of the 44th annual Southeast regional conference*, pages 549–554. ACM, 2006.

Kevin P Balanda and HL MacGillivray. Kurtosis: a critical review. *The American Statistician*, 42(2):111–119, 1988.

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

Rachid El Bansarkhani and Johannes A. Buchmann. High performance lattice-based cca-secure encryption. *IACR Cryptology ePrint Archive*, 2015:42, 2015. URL http://eprint.iacr.org/2015/042.

Paulo S. L. M. Barreto, Patrick Longa, Michael Naehrig, Jefferson E. Ricardini, and Gustavo Zanon. Sharper Ring-LWE Signatures. *IACR Cryptology ePrint Archive*, 2016:1026, 2016. URL http://eprint.iacr.org/2016/1026.

Paul Barrett. Implementing the Rivest, Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *CRYPTO*, pages 311–323, 1986.

Lawrence E. Bassham III, Andrew L. Rukhin, Juan Soto, James R. Nechvatal, Miles E. Smid, Elaine B. Barker, Stefan D. Leigh, Mark Levenson, Mark Vangel, David L. Banks, Nathanael Alan Heckert, James F. Dray, and San Vo. SP 800-22 Rev. 1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical report, Gaithersburg, MD, United States, 2010.

Anja Becker and Thijs Laarhoven. Efficient (ideal) lattice sieving using cross-polytope LSH. *IACR Cryptology ePrint Archive*, 2015:823, 2015. URL http://eprint.iacr.org/2015/823.

Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pages 62–73, 1993.

Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 399–416. Springer, 1996.

Anil K. Bera and Carlos M. Jarque. Efficient tests for normality, homoscedasticity and serial independence of regression residuals: Monte carlo evidence. *Economics Letters*, 7(4):313–318, 1981.

Daniel J. Bernstein. A subfield-logarithm attack against ideal lattices, 2014. http://blog.cr.yp.to/20140213-ideal.html. Feb. 2014. Accessed: 21.10.2015.

Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, pages 368–397, 2015. doi: 10.1007/978-3-662-46800-5_15. URL http://dx.doi.org/10.1007/978-3-662-46800-5_15.

Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. Ntru prime. Cryptology ePrint Archive, Report 2016/461, 2016. http://eprint.iacr.org/2016/461.

Richard E. Blahut. *Fast Algorithms for Signal Processing*. Cambridge University Press, 2010a. ISBN 9780521190497.

Richard E Blahut. *Fast algorithms for signal processing*. Cambridge University Press, 2010b.

Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In *CRYPTO (2)*, pages 361–379, 2013.

Ahmad Boorghany and Rasool Jalili. Implementation and comparison of lattice-based identification protocols on smart cards and microcontrollers. *IACR Cryptology ePrint Archive*, 2014:78, 2014a.

Ahmad Boorghany and Rasool Jalili. Implementation and Comparison of Lattice-based Identification Protocols on Smart Cards and Microcontrollers. *IACR Cryptology ePrint Archive*, 2014:514, 2014b.

Ahmad Boorghany, Siavash Bayat Sarmadi, and Rasool Jalili. On constrained implementation of lattice-based cryptographic primitives and schemes on smart cards. *ACM Trans. Embed. Comput. Syst.*, 14(3):42:1–42:25, April 2015. ISSN 1539-9087. doi: 10.1145/2700078. URL http://doi.acm.org/10.1145/2700078.

Joppe W. Bos, Michael Naehrig, and Joop van de Pol. Sieving for shortest vectors in ideal lattices: a practical perspective. *IACR Cryptology ePrint Archive*, 2014: 880, 2014. URL http://eprint.iacr.org/2014/880.

Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In *STOC*, pages 575–584, 2013. ISBN 978-1-4503-2029-0.

Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. Flush, gauss, and reload - A cache attack on the BLISS lattice-based signature scheme. In *CHES*, pages 323–345, 2016. doi: 10.1007/978-3-662-53140-2_16. URL http://dx.doi.org/10.1007/978-3-662-53140-2_16.

Johannes Buchmann, Daniel Cabarcas, Florian Göpfert, Andreas Hülsing, and Patrick Weiden. Discrete Ziggurat: A time-memory trade-off for sampling from a Gaussian distribution over the integers. In *SAC*, pages 402–417, 2013.

Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, pages 117–129, 2011.

M Campagna, L Chen, Ö Dagdelen, J Ding, JK Fernick, N Gisin, D Hayford, T Jennewein, N Lütkenhaus, M Mosca, et al. Quantum safe cryptography and security. *ETSI White Paper*, (8), 2015.

Xiaolin Cao, Ciara Moore, Máire O'Neill, Elizabeth O'Sullivan, and Neil Hanley. Optimised Multiplication Architectures for Accelerating Fully Homomorphic Encryption. *IEEE Transactions on Computers*, 65(9):2794–2806, 2016.

CESG. Quantum key distribution: A CESG white paper, February 2016. URL https://www.cesg.gov.uk/white-papers/quantum-key-distribution.

Sanjit Chatterjee, Alfred Menezes, and Palash Sarkar. Another look at tightness. In *Selected Areas in Cryptography*, pages 293–319. Springer-Verlag, 2011.

Donald Donglong Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray C. C. Cheung, Derek Pao, and Ingrid Verbauwhede. High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems. *IACR Cryptology ePrint Archive*, 2014:646, 2014.

Hao Chen, Kristin Lauter, and Katherine E. Stange. Attacks on search RLWE. Cryptology ePrint Archive, Report 2015/971, 2015.

Yuanmi Chen and Phong Nguyen. BKZ 2.0: Better Lattice Security Estimates. *Advances in Cryptology–ASIACRYPT 2011*, pages 1–20, 2011.

Arjun Chopra. Improved Parameters for the Ring-TESLA Digital Signature Scheme. *IACR Cryptology ePrint Archive*, 2016:1099, 2016. URL http://eprint.iacr.org/2016/1099.

Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Efficient software implementation of ring-LWE encryption. *IACR Cryptology ePrint Archive*, 2014:725, 2014.

CNSS. Use of public standards for the secure sharing of information among national security systems. Committee on National Security Systems: CNSS Advisory Memorandum, Information Assurance 02-15, July 2015.

Paul G. Comba. Exponentiation cryptosystems on the IBM PC. *IBM Systems Journal*, 29(4):526–538, 1990.

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

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, third edition edition, 7 2009. ISBN 9780262033848.

Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. *IACR Cryptology ePrint Archive*, 2015:313, 2015. URL http://eprint.iacr.org/2015/313.

Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat-Shamir transformation in a quantum world. In *ASIACRYPT (2)*, pages 62–81, 2013.

Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Sánchez, and Peter Schwabe. High-speed signatures from standard lattices. In *LATINCRYPT*, pages 84–103, 2014. doi: 10.1007/978-3-319-16295-9_5. URL http://dx.doi.org/10.1007/978-3-319-16295-9_5.

Ralph B D'Agostino, Albert Belanger, and Ralph B D'Agostino Jr. A suggestion for using powerful and informative tests of normality. *The American Statistician*, 44(4):316–321, 1990.

Barb Darrow. Official At Last: Intel Completes $16.7 Billion Buy of Altera. *Fortune*, February 2016.

Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In *Information Security*, pages 171–186. Springer, 2006.

Ruan De Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Efficient software implementation of ring-LWE encryption. In *DATE*, pages 339–344. EDA Consortium, 2015.

Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology*, 8(3):209–247, 2014.

Luc Devroye. Sample-based non-uniform random variate generation. In *WSC*, pages 260–265. ACM, 1986.

J. F. Dhem. *Design of an efficient public-key cryptographic library for RISC-based smart cards*. PhD thesis, 1998. http://users.belgacom.net/dhem/these/these_public.pdf.

Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

Jintai Ding and Dieter Schmidt. Rainbow, a new multivariable polynomial signature scheme. In *ACNS*, volume 5, pages 164–175. Springer, 2005.

Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating cvp to within almost-polynomial factors is np-hard. *Combinatorica*, 23(2):205–243, April 2003. ISSN 0209-9683.

Benedikt Driessen, Axel Poschmann, and Christof Paar. Comparison of innovative signature algorithms for WSNs. In *WISEC*, pages 30–35, 2008.

Chaohui Du and Guoqiang Bai. Towards efficient discrete Gaussian sampling for lattice-based cryptography. In *FPL '15*, pages 1–6. IEEE, 2015.

Chaohui Du and Guoqiang Bai. Towards efficient polynomial multiplication for lattice-based cryptography. In *IEEE International Symposium on Circuits and Systems, ISCAS 2016, Montréal, QC, Canada, May 22-25, 2016*, pages 1178–1181, 2016. doi: 10.1109/ISCAS.2016.7527456. URL http://dx.doi.org/10.1109/ISCAS.2016.7527456.

Chaohui Du, Guoqiang Bai, and Hongyi Chen. Towards Efficient Implementation of Lattice-Based Public-Key Encryption on Modern CPUs. In *2015 IEEE TrustCom/BigDataSE/ISPA, Helsinki, Finland, August 20-22, 2015, Volume 1*, pages 1230–1236, 2015. doi: 10.1109/Trustcom.2015.510. URL http://dx.doi.org/10.1109/Trustcom.2015.510.

Chaohui Du, Guoqiang Bai, and Xingjun Wu. High-Speed Polynomial Multiplier Architecture for Ring-LWE Based Public Key Cryptosystems. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI, GLVLSI 2016, Boston, MA, USA, May 18-20, 2016*, pages 9–14, 2016. doi: 10.1145/2902961.2902969. URL http://doi.acm.org/10.1145/2902961.2902969.

Léo Ducas. Accelerating Bliss: the geometry of ternary polynomials. *IACR Cryptology ePrint Archive*, 2014:874, 2014a. URL http://eprint.iacr.org/2014/874.

Léo Ducas. Accelerating Bliss: the geometry of ternary polynomials. *IACR Cryptology ePrint Archive*, 2014:874, 2014b.

Léo Ducas and Phong Q. Nguyen. Faster Gaussian lattice sampling using lazy floating-point arithmetic. In *ASIACRYPT*, pages 415–432, 2012a.

Léo Ducas and Phong Q. Nguyen. Learning a zonotope and more: Cryptanalysis of NTRUSign countermeasures. In *ASIACRYPT*, pages 433–450, 2012b.

Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In *CRYPTO (1)*, pages 40–56, 2013. Full version: https://eprint.iacr.org/2013/383.pdf.

Leo Ducas, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehle. Crystals–dilithium: Digital signatures from module lattices. *IACR Cryptology ePrint Archive*, 2017:633, 2017.

Nagarjun C. Dwarakanath and Steven D. Galbraith. Sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Appl. Algebra Eng. Commun. Comput.*, pages 159–180, 2014.

Kirsten Eisenträger, Sean Hallgren, and Kristin Lauter. Weak instances of PLWE. In *Selected Areas in Cryptography–SAC 2014*, pages 183–194. Springer, 2014.

Yara Elias, KristinE. Lauter, Ekin Ozman, and KatherineE. Stange. Provably weak instances of ring-LWE. In *CRYPTO*, volume 9215, pages 63–92. 2015. ISBN 978-3-662-47988-9. doi: 10.1007/978-3-662-47989-6_4. URL http://dx.doi.org/10.1007/978-3-662-47989-6_4.

Pavel Emeliyanenko. Efficient multiplication of polynomials on graphics hardware. In *APPT*, pages 134–149, 2009.

Niall Emmart and Charles C. Weems. High precision integer multiplication with a GPU using Strassen's algorithm with multiple FFT sizes. *Parallel Processing Letters*, 21(3):359–375, 2011.

Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

Andy Field. *Discovering statistics using SPSS*. Sage publications, 2009.

Ronald Aylmer Fisher and Frank Yates. Statistical tables for biological, agricultural and medical research. *Statistical tables for biological, agricultural and medical research.*, pages 25–27, 1948. Third Ed.

Karl Freund. Amazon's Xilinx FPGA Cloud: Why This May Be A Significant Milestone. *Forbes*, December 2016.

Steven D. Galbraith. *Mathematics of Public-Key Cryptography*. Cambridge: Cambridge University Press. xiv, 2012.

Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. On the security of supersingular isogeny cryptosystems. Cryptology ePrint Archive, Report 2016/859, 2016. http://eprint.iacr.org/2016/859.

Craig Gentry and Michael Szydlo. Cryptanalysis of the revised NTRU signature scheme. In *EUROCRYPT*, pages 299–320, 2002. ISBN 3-540-43553-0.

Craig Gentry, Jakob Jonsson, Jacques Stern, and Michael Szydlo. Cryptanalysis of the NTRU signature scheme (NSS). In *ASIACRYPT*, pages 1–20, 2001.

Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

Benjamin Glas, Oliver Sander, Vitali Stuckert, Klaus D Müller-Glaser, and Jürgen Becker. Prime field ecdsa signature processing for reconfigurable embedded systems. *International Journal of Reconfigurable Computing*, 2011:5, 2011.

Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(56), 1996.

Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2): 281–308, apr 1988. ISSN 0097-5397.

Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin A. Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In *CHES*, pages 512–529, 2012.

Andy Greenberg. Hackers Remotely Kill a Jeep on the Highway — With Me in It. *Wired*, July 2015.

Lov K Grover. A fast quantum mechanical algorithm for database search. In *STOC*, pages 212–219. ACM, 1996.

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

Tim Güneysu. Utilizing hard cores of modern FPGA devices for high-performance cryptography. *J. Cryptographic Engineering*, 1(1):37–55, 2011.

Tim Güneysu and Christof Paar. Ultra high performance ECC over NIST primes on commercial FPGAs. In *CHES*, pages 62–78, 2008.

Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. Practical lattice-based cryptography: A signature scheme for embedded systems. In *CHES*, pages 530–547, 2012.

Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. Software speed records for lattice-based signatures. In *PQCrypto*, pages 67–82, 2013.

Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *CHES*, pages 119–132, 2004.

Tamas Györfi, Octavian Cret, and Zalan Borsos. Implementing modular FFTs in FPGAs - a basic block for lattice-based cryptography. In *DSD*, pages 305–308, 2013.

Bettina Helfrich. Algorithms to construct minkowski reduced and hermite reduced lattice bases. *Theor. Comput. Sci.*, 41(2-3):125–139, December 1985. ISSN 0304-3975.

Perry R Hinton. *Statistics explained*. Routledge, 2014.

Jeffrey Hoffstein and Joseph Silverman. Optimizations for NTRU. *Public-Key Cryptography and Computational Number Theory, Warsaw*, pages 77–88, 2001.

Jeffrey Hoffstein and Joseph H Silverman. Random small hamming weight products with applications to cryptography. *Discrete Applied Mathematics*, 130(1):37–49, 2003.

Jeffrey Hoffstein and Joseph H Silverman. Speed enhanced cryptographic method and apparatus, April 18 2006. US Patent 7,031,468.

Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Public key cryptosystem method and apparatus, June 27 2000. US Patent 6,081,597.

Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: An NTRU lattice-based signature scheme. In *EUROCRYPT*, pages 211–228, 2001a. ISBN 3-540-42070-3.

Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. Ring-based public key cryptosystem method, October 2 2001b. US Patent 6,298,137.

Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSign: Digital signatures using the NTRU lattice. In *CT-RSA*, pages 122–140, 2003. ISBN 3-540-00847-0.

Jeffrey Hoffstein, Nicholas A Howgrave-Graham, Jill C Pipher, Joseph H Silverman, and William J Whyte. Digital signature and authentication method and apparatus, December 11 2007. US Patent 7,308,097.

James Howe and Máire O'Neill. GLITCH: A Discrete Gaussian Testing Suite For Lattice-Based Cryptography. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017): SECRYPT, Madrid, Spain, July 24-26, 2017.*, 2017.

James Howe, Ciara Rafferty, Ayesha Khalid, and Máire O'Neill. Compact and Provably Secure Lattice-Based Signatures in Hardware. In *IEEE International Symposium on Circuits and Systems, ISCAS 2017, Baltimore, Maryland, USA, May 28-31, 2017.*

James Howe, Thomas Pöppelmann, Máire O'Neill, Elizabeth O'Sullivan, and Tim Güneysu. Practical lattice-based digital signature schemes. *ACM Transactions on Embedded Computing Systems*, 14(3):24, 2015.

James Howe, Ayesha Khalid, Ciara Rafferty, Francesco Regazzoni, and Máire O'Neill. On Practical Discrete Gaussian Samplers For Lattice-Based Cryptography. *IEEE Transactions on Computers*, 2016a.

James Howe, Ciara Moore, Máire O'Neill, Francesco Regazzoni, Tim Güneysu, and Kevin Beeden. Lattice-based Encryption Over Standard Lattices in Hardware. In *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*, 2016b.

James Howe, Ciara Moore, Máire O'Neill, Francesco Regazzoni, Tim Güneysu, and Kevin Beeden. Lattice-based Encryption Over Standard Lattices in Hardware. In *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*, pages 162:1–162:6. ACM, 2016c.

Intel. The Coming Data Avalanche – And How We'll Handle It. *Forbes Insights*, January 2017.

Tsukasa Ishiguro, Shinsaku Kiyomoto, Yutaka Miyake, and Tsuyoshi Takagi. Parallel Gauss sieve algorithm: Solving the SVP challenge over a 128-dimensional ideal lattice. In *PKC*, pages 411–428, 2014. URL http://dx.doi.org/10.1007/978-3-642-54631-0_24.

David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, pages 19–34, 2011. doi: 10.1007/978-3-642-25405-5_2. URL http://dx.doi.org/10.1007/978-3-642-25405-5_2.

Carlos M. Jarque and Anil K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3):255–259, 1980.

Carlos M. Jarque and Anil K. Bera. A test for normality of observations and regression residuals. *International Statistical Review*, pages 163–172, 1987.

D. N. Joanes and C. A. Gill. Comparing measures of sample skewness and kurtosis. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(1), 1998. ISSN 1467-9884.

Abdel Alim Kamal and Amr M. Youssef. An FPGA implementation of the NTRUEncrypt cryptosystem. In *ICM*, pages 209–212, Dec 2009a.

Abdel Alim Kamal and Amr M Youssef. An FPGA implementation of the NTRUEncrypt cryptosystem. In *Microelectronics (ICM), 2009 International Conference on*, pages 209–212. IEEE, 2009b.

Anatoly A. Karatsuba and Yuri Petrovich Ofman. Multiplication of Multidigit Numbers on Automata. *Soviet Physics Doklady*, 7:595–596, 1963. ISSN 0038–5689.

Ayesha Khalid, James Howe, Ciara Rafferty, and Máire O'Neill. Time-independent discrete gaussian sampling for post-quantum cryptography. In *2016 International Conference on Field-Programmable Technology, FPT 2016, Xi'an, China, December 7-9, 2016*, pages 241–244, 2016. doi: 10.1109/FPT.2016.7929543. URL https://doi.org/10.1109/FPT.2016.7929543.

Hae-Young Kim. Statistical notes for clinical researchers: assessing normal distribution (2) using skewness and kurtosis. *Restorative dentistry & endodontics*, 38(1):52–54, 2013.

Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced oil and vinegar signature schemes. In *Advances in Cryptology—EUROCRYPT'99*, pages 206–222. Springer, 1999.

Donald E Knuth and Andrew C Yao. The complexity of nonuniform random number generation. *Algorithms and complexity: new directions and recent results*, pages 357–428, 1976.

Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 1987.

Paul C Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *CRYPTO*, pages 104–113. Springer, 1996.

Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, pages 388–397, 1999.

A. N. Kolmogorov. Foundations of the theory of probability (2nd ed.). 1956. Chelsea Publishing Co., New York.

Brian Koziel, Reza Azarderakhsh, Mehran Mozaffari Kermani, and David Jao. Post-quantum cryptography on FPGA based on isogenies on elliptic curves. *IEEE Trans. on Circuits and Systems*, 64-I(1):86–99, 2017. doi: 10.1109/TCSI.2016.2611561. URL http://dx.doi.org/10.1109/TCSI.2016.2611561.

Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography*, 2014. ISSN 0925-1022.

Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In *CT-RSA*, pages 319–339, 2011.

Dwayne C. Litzenberger et al. PyCrypto: Cryptographic modules for python, 2013.

Zhe Liu, Hwajeong Seo, Sujoy Sinha Roy, Johann Großschädl, Howon Kim, and Ingrid Verbauwhede. Efficient Ring-LWE encryption on 8-bit AVR processors. In *CHES*, pages 663–682. Springer, 2015.

Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography. In *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, pages 124–139, 2016. doi: 10.1007/978-3-319-48965-0_8. URL http://dx.doi.org/10.1007/978-3-319-48965-0_8.

Vadim Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT*, pages 598–616, 2009.

Vadim Lyubashevsky. Lattice signatures without trapdoors. In *EUROCRYPT*, pages 738–755, 2012.

Vadim Lyubashevsky. Digital signatures based on the hardness of ideal lattice problems in all rings. In *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II 22*, pages 196–214. Springer, 2016.

Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *CRYPTO*, pages 577–594, 2009. ISBN 978-3-642-03355-1.

Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In *FSE*, pages 54–72, 2008.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

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

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. A toolkit for ring-LWE cryptography. In *EUROCRYPT*, pages 35–54, 2013b.

George Marsaglia. A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface. Elsevier Science Publishers, North-Holland, Amsterdam*, pages 3–10, 1985.

George Marsaglia. A current view of random numbers. In L. Billard, editor, *Computer Science and Statistics: Proceedings of the 16th Symposium on the Interface*, volume 36, pages 105–110. Elsevier Science Publishers B. V., 7 1993.

George Marsaglia. DIEHARD: A battery of tests of randomness. http://www.stat.fsu.edu/pub/diehard/, 1996.

George Marsaglia, Wai Wan Tsang, et al. The Ziggurat method for generating random variables. *Journal of statistical software*, 5(8):1–7, 2000.

Tsutomu Matsumoto and Hideki Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *EUROCRYPT*, pages 419–453. Springer, 1988.

James H. McClellan. Hardware realization of a fermat number transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(3):216–225, Jun 1976. ISSN 0096-3518.

Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *DSN progress report*, 42(44):114–116, 1978.

Ralph C Merkle. A certified digital signature. In *CRYPTO*, pages 218–238. Springer, 1990.

Cade Metz. Microsoft Bets Its Future on a Reprogrammable Computer Chip. *Wired*, December 2016.

Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Comput. Complex.*, 16(4):365–411, December 2007. ISSN 1016-3328.

Daniele Micciancio. Efficient reductions among lattice problems. In *SODA*, pages 84–93, 2008.

Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *CRYPTO*, pages 465–484, 2011.

Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

Daniele Micciancio and Chris Peikert. Hardness of SIS and LWE with small parameters. In *CRYPTO (1)*, pages 21–39, 2013.

Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1468–1480. SIAM, 2010.

Daniele Micciancio and Michael Walter. Gaussian sampling over the integers: Efficient, generic, constant-time. *IACR Cryptology ePrint Archive*, 2017:259, 2017.

Victor S Miller. Use of elliptic curves in cryptography. In *CRYPTO*, pages 417–426, 1986. ISBN 0-387-16463-4.

Robert T. Moenck. Practical fast polynomial multiplication. In *SYMSACC*, pages 136–148, 1976.

Dustin Moody. Post-quantum cryptography: NIST's plan for the future. Talk given at PQCrypto '16 Conference, 23-26 February 2016, Fukuoka, Japan, February 2016. URL https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf.

Steven J Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. Chip and pin is broken. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 433–446. IEEE, 2010.

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

NIST. Post-quantum crypto project. http://csrc.nist.gov/groups/ST/post-quantum-crypto/, 2016. Accessed: 27.07.2017.

Henri Nussbaumer. *Fast Fourier transform and convolution algorithms*. Springer-Verlag, 1980. ISBN 978-3540118251.

Tobias Oder, Thomas Pöppelmann, and Tim Güneysu. Beyond ECDSA and RSA: lattice-based digital signatures on constrained devices. In *DAC*, pages 1–6, 2014.

Tobias Oder, Tim Güneysu, Felipe Valencia, Ayesha Khalid, Maire O'Neill, and Francesco Regazzoni. Lattice-based cryptography: From reconfigurable hardware to ASIC. In *Integrated Circuits (ISIC), 2016 International Symposium on*, pages 1–4. IEEE, 2016.

Máire O'Neill, Elizabeth O'Sullivan, Gavin McWilliams, Markku-Juhani Saarinen, Ciara Moore, Ayesha Khalid, James Howe, Rafaël Del Pino, Michel Abdalla, Francesco Regazzoni, Felipe Valencia, Tim Güneysu, Tobias Oder, Adrian Waller, Glyn Jones, Anthony Barnett, Robert Griffin, Andrew Byrne, Bassem Ammar, and David Lund. Secure Architectures of Future Emerging Cryptography *SAFEcrypto*. In *Proceedings of the ACM International Conference on Computing Frontiers, CF'16, Como, Italy, May 16-19, 2016*, pages 315–322, 2016.

Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In *Post-Quantum Cryptography*, pages 95–145. 2009. ISBN 978-3-540-88701-0.

Marshall C. Pease. An adaptation of the fast Fourier transform for parallel processing. *J. ACM*, 15(2):252–264, April 1968. ISSN 0004-5411.

Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. *Electronic Colloquium on Computational Complexity (ECCC)*, 15 (100), 2008.

Chris Peikert. An efficient and parallel Gaussian sampler for lattices. In *CRYPTO*, pages 80–97, 2010.

Chris Peikert. Lattice cryptography for the internet. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, pages 197–219, 2014. doi: 10.1007/978-3-319-11659-4_12. URL http://dx.doi.org/10.1007/978-3-319-11659-4_12.

Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 187–196. ACM, 2008.

Peter Pessl. Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, pages 153–170, 2016. doi: 10.1007/978-3-319-49890-4_9. URL http://dx.doi.org/10.1007/978-3-319-49890-4_9.

John M Pollard. The fast fourier transform in a finite field. *Mathematics of Computation*, 25(114):365–374, 1971a.

John M Pollard. The fast Fourier transform in a finite field. *Mathematics of computation*, 25(114):365–374, 1971b.

Thomas Pöppelmann and Tim Güneysu. Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware. In *LATINCRYPT*, pages 139–158, 2012.

Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In *SAC*, pages 68–85, 2013.

Thomas Pöppelmann and Tim Güneysu. Area optimization of lightweight lattice-based encryption on reconfigurable hardware. In *ISCAS*, pages 2796–2799, 2014.

Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In *CHES*, pages 353–370, 2014. Full version: https://eprint.iacr.org/2014/254.pdf.

Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. High-Performance Ideal Lattice-Based Cryptography on 8-bit ATxmega Microcontrollers. In *International Conference on Cryptology and Information Security in Latin America*, pages 346–365. Springer, 2015.

Charles M Rader. Discrete Convolutions via Mersenne Transforms. *IEEE Transactions on Computers*, 100(12):1269–1273, 1972.

Wolfgang Rankl and Wolfgang Effing. *Smart Card Handbook*. John Wiley & Sons, fourth edition edition, 2010. ISBN 978-0-470-74367-6. http://pdf.th7.cn/down/files/1312/Smart%20Card%20Handbook,%204th%20Edition.pdf.

Nornadiah Mohd Razali, Yap Bee Wah, et al. Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of statistical modeling and analytics*, 2(1):21–33, 2011.

Chester Rebeiro, Sujoy Sinha Roy, and Debdeep Mukhopadhyay. Pushing the limits of high-speed GF(2m) elliptic curve scalar multiplication on FPGAs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 494–511. Springer, 2012.

Oded Regev. New lattice-based cryptographic constructions. *Journal of the ACM (JACM)*, 51(6):899–942, 2004.

Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005. ISBN 1-58113-960-8.

Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

Oded Regev. The learning with errors problem. *Invited survey in CCC*, 2010.

Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-LWE implementation. In *CHES*, pages 683–702, 2015a.

Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A Masked Ring-LWE Implementation. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 683–702, 2015b. doi: 10.1007/978-3-662-48324-4_34. URL http://dx.doi.org/10.1007/978-3-662-48324-4_34.

Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-lwe. *J. Cryptographic Engineering*, 6(2): 139–153, 2016. doi: 10.1007/s13389-016-0126-5. URL http://dx.doi.org/10.1007/s13389-016-0126-5.

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

Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact Hardware Implementation of Ring-LWE Cryptosystems. *IACR Cryptology ePrint Archive*, 2013:866, 2013a. URL http://eprint.iacr.org/2013/866.

Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. High Precision Discrete Gaussian Sampling on FPGAs. In *SAC*, pages 1–39, 2013b.

Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. Compact and side channel secure discrete Gaussian sampling. *IACR Cryptology ePrint Archive*, 2014:591, 2014a.

Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact Ring-LWE Cryptoprocessor. In *CHES*, pages 371–391. Springer, 2014b.

J. P. Royston. An extension of Shapiro and Wilk's W test for normality to large samples. *Applied Statistics*, pages 115–124, 1982.

Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. *IACR Cryptology ePrint Archive*, 2010:137, 2010.

Ajay Rupani, Dikshant Pandey, and Gajendra Sujediya. Review and study of fpga implementation of internet of things. *IJSTE - International Journal of Science Technology & Engineering*, 3(2), 2016.

Markku-Juhani O. Saarinen. Gaussian sampling precision and information leakage in lattice cryptography. Cryptology ePrint Archive, Report 2015/953, 2015. Full version: https://link.springer.com/article/10.1007/s13389-017-0149-6.

Markku-Juhani O Saarinen. Arithmetic Coding And Blinding Countermeasures For Lattice Signatures. *Journal of Cryptographic Engineering*, pages 1–14, 2017.

Michael Schneider. Sieving for shortest vectors in ideal lattices. In *AFRICACRYPT*, pages 375–391, 2013. doi: 10.1007/978-3-642-38553-7_22. URL http://dx.doi.org/10.1007/978-3-642-38553-7_22.

Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989.

Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, pages 591–611, 1965.

Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997. ISSN 0097-5397.

Joseph H Silverman and William Whyte. Timing attacks on NTRUEncrypt via variation in the number of hash calls. In *CT-RSA*, pages 208–224. Springer, 2007.

Sergei P Skorobogatov and Ross J Anderson. Optical fault induction attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 2–12. Springer, 2002.

Damien Stehlé and Ron Steinfeld. Making NTRU as secure as worst-case problems over ideal lattices. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 27–47. Springer, 2011.

Daisuke Suzuki and Tsutomu Matsumoto. How to maximize the potential of FPGA-based DSPs for modular exponentiation. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 94(1):211–222, 2011.

John Von Neumann. Various techniques used in connection with random digits. *Applied Math Series*, 12(36-38):1, 1951.

Patrick Weiden, Andreas Hülsing, Daniel Cabarcas, and Johannes Buchmann. Instantiating treeless signature schemes. *IACR Cryptology ePrint Archive*, 2013: 65, 2013.

Franz Winkler. *Polynomial Algorithms in Computer Algebra (Texts and Monographs in Symbolic Computation)*. Springer, 1 edition, 8 1996. ISBN 9783211827598.

Nicky Woolf. DDoS attack that disrupted internet was largest of its kind in history, experts say. *The Guardian*, October 2016.

Xilinx. Spartan-6 family overview. *Xilinx Product Specification, DS160 (v2. 0)*, 2011.

Xilinx. Virtex-5qv fpga electrical characteristics. *Xilinx Product Specification, DS692 (v1.3.1)*, 2015.

APPENDIX A

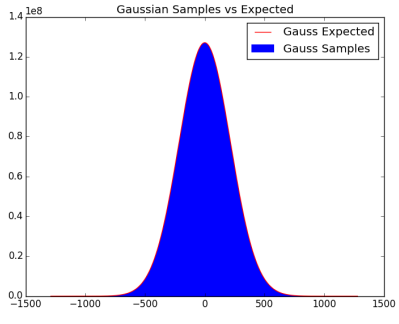
Example Results from the Discrete Gaussian Test Suite

Tables A.1 and A.2 show the performance of an operational discrete Gaussian sampler, a Bernoulli sampler, with 2^{36} samples. Table A.1 shows results for a sampler providing outputs as expected and which passes all proposed tests. Table A.2 shows results for a sampler with all correct values except for the standard deviation, which is in this case set to $\sigma = 210$. The results expectedly pass for most of the tests, but from Test 2 in Table A.2 in can be seen that the calculated σ is lower than required.

Table A.1 Discrete Gaussian sampling test results with target standard deviation $\sigma = 215.727\ldots$ using the Bernoulli sampling with sample size 2^{36}.

#	Test Description	Test Output
(1)	Sample Mean:	0.00137119684950448572635650634 8
	Standard Error Of The Mean:	0.00082293340362298911039881399 99
	C.I. Of The Sample Mean =	0.00137119684950448572635650634 8
		+/- 0.00270745089791963420244856565 8
		with 99.9% confidence
(2)	Sample Standard Deviation:	215.72705415934485735638669 72
	Standard Error Of The Standard Deviation:	0.00058190179017097564940570533 63
	C.I. Of The Sample Standard Deviation =	215.72705415934485735638669 72
		+/- 0.00191445688966250990721807505 3
		with 99.9% confidence
(3)	Sample Tail-Cut Parameter (τ):	13.00250044549569511934255936
	Distance From Target Tail-Cut:	0.31837333302746789195775270 41
(4)	Sample Skewness:	-0.00001763835979933123583199835026
	Standard Error Of The Sample Skewness:	0.00000934406182376751307512264 6283
(5)	Sample Excess Kurtosis:	-0.00002450163588799744963112 6
	Standard Error Of The Sample Kurtosis:	0.00001868812364726307815918276254
(6)	Sample Hyperskewness:	-0.00009572213233407114802715387328
(7)	Sample Excess Hyperkurtosis:	-0.00025338061051584529784336
(8)	Jarque-Bera Test For Normality:	4.31216648721667127493653916 6E-7
	(test statistic, p-value)	0.999999784392
(9)	D'Agostino-Pearson K^2 Omnibus Test:	0.00293403472934388690551443145 7
	(test statistic, p-value)	0.998534058179

(10)-(11) Histogram and Quantile-Quantile (QQ) plots:



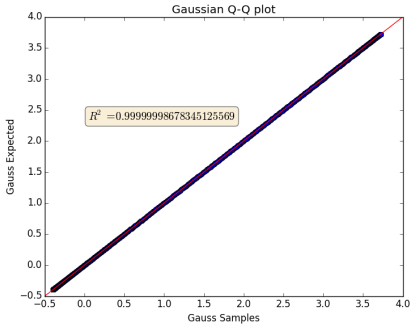
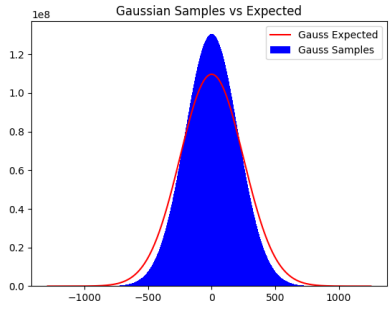
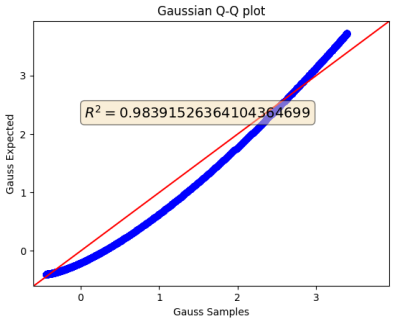


Table A.2 Discrete Gaussian sampling test results with target standard deviation $\sigma = 215.727\ldots$, but generated with standard deviation $\sigma = 210$, using the Bernoulli sampling with sample size 2^{36}.

#	Test Description	Test Output
(1)	Sample Mean:	0.000415780508774332702159881591 8
	Standard Error Of The Mean:	0.000801084797793829643742633526 6
	C.I. Of The Sample Mean =	0.000415780508774332702159881591 8
		+/- 0.002635568984741699556373513493
		with 99.9% confidence
(2)	Sample Standard Deviation:	209.99957323286567812926892 32
	Standard Error Of The Standard Deviation:	0.000566452492829592651241111943 7
	C.I. Of The Sample Standard Deviation =	209.99957323286567812926892 32
		+/- 0.001863628701409359842707693491
		with 99.9% confidence
(3)	Sample Tail-Cut Parameter (τ):	12.65484000577655888620505777
	Distance From Target Tail-Cut:	0.666033772746604125095254294 1
(4)	Sample Skewness:	-0.000008109373516717886834916069369
	Standard Error Of The Sample Skewness:	0.000009344061823767513075122646283
(5)	Sample Excess Kurtosis:	-0.000028095808185055005256698
	Standard Error Of The Sample Kurtosis:	0.000018688123647263078159182762 54
(6)	Sample Hyperskewness:	-0.000103651468047191992233509974
(7)	Sample Excess Hyperkurtosis:	-0.000519099829468152675819 23
(8)	Jarque-Bera Test For Normality:	2.39426048886111709764460353 6E-7
	(test statistic, p-value)	0.999999880287
(9)	D'Agostino-Pearson K^2 Omnibus Test:	0.003004329115990208729071137285
	(test statistic, p-value)	0.998498963126

(10)-(11) Histogram and Quantile-Quantile (QQ) plots:





APPENDIX B

Author's Publications

1. James Howe, Thomas Pöppelmann, Máire O'Neill, Elizabeth O'Sullivan, and Tim Güneysu. Practical lattice-based digital signature schemes. *ACM Transactions on Embedded Computing Systems*, 14(3):24, 2015.

2. James Howe, Ciara Moore, Máire O'Neill, Francesco Regazzoni, Tim Güneysu, and Kevin Beeden. Lattice-based Encryption Over Standard Lattices in Hardware. In *Proceedings of the 53rd Annual Design Automation Conference, DAC 2016, Austin, TX, USA, June 5-9, 2016*, pages 162:1–162:6. ACM, 2016c.

3. Máire O'Neill, Elizabeth O'Sullivan, Gavin McWilliams, Markku-Juhani Saarinen, Ciara Moore, Ayesha Khalid, James Howe, Rafaël Del Pino, Michel Abdalla, Francesco Regazzoni, Felipe Valencia, Tim Güneysu, Tobias Oder, Adrian Waller, Glyn Jones, Anthony Barnett, Robert Griffin, Andrew Byrne, Bassem Ammar, and David Lund. Secure Architectures of Future Emerging Cryptography *SAFEcrypto*. In *Proceedings of the ACM International Conference on Computing Frontiers, CF'16, Como, Italy, May 16-19, 2016*, pages 315–322, 2016.

4. Ayesha Khalid, James Howe, Ciara Rafferty, and Máire O'Neill. Time-independent discrete gaussian sampling for post-quantum cryptography. In *2016 International Conference on Field-Programmable Technology, FPT 2016, Xi'an, China, December 7-9, 2016*, pages 241–244, 2016. doi: 10.1109/FPT.2016.7929543. URL https://doi.org/10.1109/FPT.2016.7929543.

5. James Howe, Ayesha Khalid, Ciara Rafferty, Francesco Regazzoni, and Máire O'Neill. On Practical Discrete Gaussian Samplers For Lattice-Based Cryptography. *IEEE Transactions on Computers*, 2016a.

6. James Howe, Ciara Rafferty, Ayesha Khalid, and Máire O'Neill. Compact and Provably Secure Lattice-Based Signatures in Hardware. In *IEEE International Symposium on Circuits and Systems, ISCAS 2017, Baltimore, Maryland, USA, May 28-31, 2017*.