A

Practical Lattice-based Digital Signature Schemes

JAMES HOWE, Centre for Secure Information Technologies (CSIT), Queen's University Belfast, UK
THOMAS PÖPPELMANN, Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany
MÁIRE O'NEILL, Centre for Secure Information Technologies (CSIT), Queen's University Belfast, UK
ELIZABETH O'SULLIVAN, Centre for Secure Information Technologies (CSIT), Queen's University Belfast, UK
TIM GÜNEYSU, Horst Görtz Institute for IT-Security, Ruhr-University Bochum, Germany

Digital signatures are an important primitive for building secure systems and are used in most real world security protocols. However, almost all popular signature schemes are either based on the factoring assumption (RSA) or the hardness of the discrete logarithm problem (DSA/ECDSA). In the case of classical cryptanalytic advances or progress on the development of quantum computers the hardness of these closely related problems might be seriously weakened. A potential alternative approach is the construction of signature schemes based on the hardness of certain lattices problems which are assumed to be intractable by quantum computers. Due to significant research advancements in recent years, lattice-based schemes have now become practical and appear to be a very viable alternative to number-theoretic cryptography. In this paper we focus on recent developments and the current state-of-the-art in lattice-based digital signatures and provide a comprehensive survey discussing signature schemes with respect to practicality. Additionally, we discuss future research areas that are essential for the continued development of lattice-based cryptography.

Categories and Subject Descriptors: E.3 [**Data Encryption**]

General Terms: Performance, Security, Theory

Additional Key Words and Phrases: Digital signatures, lattices, post-quantum cryptography.

ACM Reference Format:
James Howe and Thomas Pöppelmann and Máire O'Neill and Elizabeth O'Sullivan and Tim Güneysu. 2014. Practical Lattice-based Digital Signature Schemes. *ACM Trans. Embedd. Comput. Syst.* V, N, Article A (January YYYY), 25 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

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 [1997] algorithm. 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

Author's address: J. Howe, Centre for Secure Information Technologies, Queen's University Belfast, Queen's Road, Queen's Island, Belfast, BT3 9DT; Thomas Pöppelmann, Hardware Security Group, Ruhr-University Bochum, Universitaetsstr. 150, 44801 Bochum, Germany; M. O'Neill, Centre for Secure Information Technologies, Queen's University Belfast, Queen's Road, Queen's Island, Belfast, BT3 9DT; Tim Güneysu, Hardware Security Group, Ruhr-University Bochum, Universitaetsstr. 150, 44801 Bochum, Germany.

DOI: http://dx.doi.org/10.1145/0000000.0000000

cryptography (ECC), RSA and DSA would be vulnerable. Accordingly, this has motivated the era of post-quantum cryptography (PQC), which refers to the construction of cryptographic algorithms to withstand quantum reductions. Amongst many important areas in post-quantum research (such as multivariate, code or hash-based) lattice-based cryptography is disputably the most auspicious. Its main advantage over other post-quantum cryptosystems is that it allows for extended functionality and is, at the same time, more 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 a 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 and practicability, for replacing current asymmetric schemes that would be susceptible to attacks in a post-quantum world.

In recent years there has been a tremendous growth in lattice-based cryptography as a research field. As a result, concepts such as functional encryption [Boneh et al. 2011], identity-based encryption [Agrawal et al. 2010; Ducas et al. 2014], attribute-based encryption [Boyen 2013], group signature schemes [Gordon et al. 2010; Camenisch et al. 2012; Laguillaumie et al. 2013] and fully homomorphic encryption [Gentry 2009a; 2009b] are now available.

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. The most recent implementation of a lattice-based encryption scheme in hardware is shown by Roy et al. [2014] (which improves on Pöppelmann and Güneysu [2013] and Göttert et al. [2012]) with results outperforming those of RSA. More specifically, the implementation is an order of magnitude faster than a comparable RSA implementation, provides a higher security level, and consumes less device resources. With regards to digital signature schemes, the two most notable hardware implementations by Güneysu et al. [2012] and Pöppelmann et al. [2014] also show a speed improvement compared to an equivalent RSA design. Additionally, scaling the latter implementation up to security levels higher than 128 bits has proven to incur only a moderate penalty in performance.

The first use of lattices as a cryptographic primitive is due to Ajtai [1996], proposing 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 as well as collision resistant hash functions and oblivious transfers, 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 protocols, 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 they 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 existing in today's digital and pervasive environment.

This motivates the theme of this paper; evaluating digital signature schemes: an essential component of a cryptosystem. Digital signature schemes have a more specific purpose than the conventional encryption paradigm, and are used in a variety of areas; from legal issues such as document integrity to those that support the world's economy through electronic commerce. This paper will concentrate on lattice-based digital signature schemes with the prevailing theme of practicality; beginning in Section 2 with theoretical prerequisites, then in Section 3 discussing the digital signature schemes based on lattice problems, discussing each with respect to its building blocks in Section 4. The paper will then summarise all current practical instantiations of lattice-based digital signature schemes in Section 5, conclude in Sections 6 and in Section 7 propose future research areas essential to the development of lattice-based cryptography.

2. PRELIMINARIES

2.1. Notation

Throughout this paper, 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 $\boldsymbol{M} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n)$, 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 $\boldsymbol{v}$ is denoted as $\|\boldsymbol{v}\|_p$, where for the Euclidean length ($p = 2$) it is simplified to $\|\boldsymbol{v}\|$. 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 $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and $\boldsymbol{x} \in \mathbb{Z}^m$, results in the element $\boldsymbol{y} \in \mathbb{Z}_q^n$. For an element s chosen uniformly at random from the set $\mathcal{S}$, the notation $s \xleftarrow{\$} \mathcal{S}$ is used.

2.2. Discrete Gaussians and Polynomial Rings

The Gaussian distribution, with standard deviation $\sigma \in \mathbb{R}$, centre $\boldsymbol{c} \in \mathbb{R}^n$, and evaluated at $\boldsymbol{x} \in \mathbb{R}^n$ is defined by a weight proportional to $\rho_{\boldsymbol{c},\sigma}(\boldsymbol{x}) = \exp(\frac{-\|\boldsymbol{x}-\boldsymbol{c}\|^2}{2\sigma^2})$. When the centre $\boldsymbol{c} = \boldsymbol{0}$ the notation is simply $\rho_\sigma(\boldsymbol{x})$. The centred discrete Gaussian distribution over $\mathbb{Z}^m$ is defined as $D_\sigma^m = \rho_\sigma(\boldsymbol{x})/\rho_\sigma(\mathbb{Z})^m$. The polynomial ring $\mathcal{R} = \mathbb{Z}_q[\boldsymbol{x}]/\langle \boldsymbol{x}^n + 1 \rangle$ 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 $[-k, k]$.

2.3. Digital Signatures

Formally, a digital signature scheme (DSS) is a tuple $(\text{KeyGen}, \text{Sign}_{sk}, \text{Verify}_{pk})$ where $\text{KeyGen}(n)$ 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]. Meaning that an adversary wins if, given access to the verification key and signing oracle O_{Sign} (i.e. 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.

2.4. The Theory of Lattices

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 the shortest (nonzero) vector, is defined as $\lambda_1(\mathcal{L}) = \min\{\|\boldsymbol{v}\| : \boldsymbol{v} \in \mathcal{L}\backslash\{\boldsymbol{0}\}\}$. 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 $\mathcal{B}(\boldsymbol{0}, r) = \{\boldsymbol{v} : \|\boldsymbol{v}\| \leq r\}$, of radius r and centre $\boldsymbol{0}$, contains at least k linearly independent vectors.

2.5. Computationally Hard Lattice Problems

Certain classes of optimisation problems for lattices, such as SVP and its inhomogeneous counterpart 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 $\boldsymbol{B}$, find a non-zero vector $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ such that $\|\boldsymbol{v}\| \leq \gamma\lambda_1(\mathcal{L}(\boldsymbol{B}))$. The second, GapSVP_γ, an approximate decision version, states that given a lattice basis $\boldsymbol{B}$ and integer d, output 1 if $\lambda_1(\mathcal{L}(\boldsymbol{B})) \leq d$ or 0 if $\lambda_1(\mathcal{L}(\boldsymbol{B})) > \gamma d$. 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 BDD) 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 $\boldsymbol{t} \in \mathbb{R}^m$. More formally, CVP_γ states that, given a lattice $\mathcal{L}$ and a vector $\boldsymbol{t} \in \mathbb{R}^m$, find a lattice point $\boldsymbol{v} \in \mathcal{L}$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leq \gamma \mathrm{d}(\boldsymbol{t}, \mathcal{L})$, where $\mathrm{d}(\boldsymbol{t}, \mathcal{L})$ 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 $\boldsymbol{B}$, a vector $\boldsymbol{t} \in \mathbb{R}^m$ and $d \in \mathbb{R}$, output 1 if $\mathrm{d}(\boldsymbol{t}, \mathcal{L}) \leq d$ or 0 if $\mathrm{d}(\boldsymbol{t}, \mathcal{L}) > \gamma d$. 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.

2.6. 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 lattices problems (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 $s_1\boldsymbol{a}_1 + s_2\boldsymbol{a}_2 + \cdots + s_m\boldsymbol{a}_m \equiv \boldsymbol{0} \bmod q$ (alternatively $\boldsymbol{A}\boldsymbol{s} \equiv \boldsymbol{0} \bmod q$). Restricting the shortness of the solution, such that $0 \leq \|\boldsymbol{s}\| \leq \beta < q$, 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 $\boldsymbol{s} = (s_1, s_2, \ldots, s_m)$, then S is a lattice implying the solution to the SIS problem is simply to find a short vector in S. More formally, as shown by Micciancio and Regev [2004; 2007], for any $q > \text{poly}(n)$ solving SIS also implies a solution to standard lattice problems such as the shortest independent vector problem (SIVP). Thereafter, Langlois and Stehlé [2014] showed reductions from module-SIVP to module-SIS as well as module-SIVP to module-LWE, the use here of module lattices bridges SIS and LWE with their respective ring variants. 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 uses for DSSs which are discussed in the next section.

The LWE problem, first proposed by Regev [2005], has many applications but can be predominantly attributed to uses in public-key cryptography and CCA-secure cryptosystems. The definition of the problem is as follows: given some uniformly distributed $\boldsymbol{a}_i \in \mathbb{Z}_q^n$, integers n and q, and $b_i \equiv \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + e_i \bmod q$, where the secret-key $\boldsymbol{s}$ is chosen uniformly at random from $\mathbb{Z}_q^n$ and each e_i follow some small error distribution, find $\boldsymbol{s}$ given access to pairs $(\boldsymbol{a}_i, b_i)$. The problem naturally produces two forms, namely its search and decision variants. The search variant asks an adversary to find $\boldsymbol{s} \in \mathbb{Z}_q^n$ given $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and $\boldsymbol{b} \equiv \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e} \bmod q$, whereas the decision variant asks an adversary to distinguish between $(\boldsymbol{a}_i, b_i)$ and $(\boldsymbol{a}_i, u_i)$ 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 αq, 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. 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 $\mathcal{L}(\boldsymbol{A}) = \{\boldsymbol{y} \in \mathbb{Z}^m \mid \boldsymbol{y} \equiv \boldsymbol{A}\boldsymbol{s} \bmod q\}$ for some $\boldsymbol{s} \in \mathbb{Z}_q^n$; in the instances where each e_i are small, the LWE problem is asking an adversary to solve the CVP on the lattice $\mathcal{L}(\boldsymbol{A})$. Inherently, the value of $\boldsymbol{s}$ is not uniquely determined, however one value is significantly more likely than the rest, ergo LWE is a well-defined maximum likelihood problem. This abstraction can be extended to the decision variant of LWE: consider again $\mathcal{L}(\boldsymbol{A})$ and $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{e} \bmod q$, since $\boldsymbol{b}$ is significantly more likely to be decoded as a lattice point than some point $\boldsymbol{v} \in \mathbb{Z}_q^m$ 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 $\boldsymbol{e}$.

Problems such as LWE and SIS are theoretically sound but lack some 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. The ring variant of SIS, Ring-SIS$_{q,n,m,\beta}$ 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 $\boldsymbol{s} \in \mathcal{R}$ such that $\|\boldsymbol{s}\|_\infty \leq \beta$ and $\boldsymbol{A}\boldsymbol{s} \equiv 0 \bmod q$. Ring-LWE$_{q,n,m,\beta}$ is defined as, given a prime modulus $q \equiv 1 \bmod 2n$, random vectors $\boldsymbol{s}, \boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_m, \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m \in \mathcal{R}$, where $\boldsymbol{b}_i = \boldsymbol{a}_i\boldsymbol{s} + \boldsymbol{e}_i \bmod q$ (again $\boldsymbol{e}_i$ following some small error distribution) find $\boldsymbol{s}$ given access to pairs $(\boldsymbol{a}_i, \boldsymbol{b}_i)$. The decision variant requires to distinguish between pairs $(\boldsymbol{a}_i, \boldsymbol{b}_i)$ and $(\boldsymbol{a}_i, \boldsymbol{u}_i)$ where $\boldsymbol{u}_i$ is chosen uniformly at random. Additionally, as shown by Lyubashevsky et al. [2013b], sampling $\boldsymbol{s}$ 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 $\boldsymbol{s}$ 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, which improve operational run-times from having quadratic to quasi-linear complexity.

3. LATTICE-BASED DIGITAL SIGNATURE SCHEMES

In this section important research on lattice-based DSSs will be presented, showing how they have become practical in terms of both hardware and software implementations. DSSs based on the hardness of lattice problems generally fall into three

categories[1], namely GGH/NTRUSign signatures, hash-and-sign signatures and Fiat-Shamir signatures.

3.1. GGH and NTRUSign Signatures

The GGH [Goldreich et al. 1996] and NTRUEncrypt [Hoffstein et al. 1998] cryptosystems 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 [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. 2001], was broken by Gentry et al. [2001; 2002] and incidently NTRUSign suffered the same fate with works by Nguyen and Regev [2009], which shows 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 since implementation results currently do not have practical applications. However, interested readers are referred to a survey by Buchmann et al. [2009]. Recent research such as Melchor et al. [2014] hold some promise for the future of these DSSs, such that someday the security and efficiency issues of NTRUSign may be amended.

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 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 became the foundation for full-domain hash (FDH) [Bellare and Rogaway 1993], 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 preimage sampleable (trapdoor) functions (PSFs) that somewhat behave 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 $f_{\boldsymbol{B}}$ ($\boldsymbol{B}$ being the public basis for some lattice $\mathcal{L}$) being evaluated by some random input corresponds to choosing a random lattice point $\boldsymbol{v} \in \mathcal{L}$ and adding some "noise" via some relatively short error term $\boldsymbol{e}$, giving a point $\boldsymbol{y} = \boldsymbol{v} + \boldsymbol{e}$. Inverting $\boldsymbol{y}$ corresponds to *decoding* it to any sufficiently close lattice point $\boldsymbol{v}' \in \mathcal{L}$, though not necessarily $\boldsymbol{v}$ itself, whereby the noise term is large enough that many preimages exist. Given the trapdoor basis, it is easy to

[1]Also note, the "vanishing trapdoor" technique by Boyen [2010], which is improved upon by Ducas and Micciancio [2014], producing a DSS in the *standard* model. However, since their goal is asymptotic optimisation, the practical is currently unclear and may not be competitive.

decode $\boldsymbol{y}$ 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. Additionally, the Gaussian sampling algorithm, that samples from a discrete Gaussian distribution, and the use of modular lattices.

Using this approach, the scheme then similarly follows the generic DSS construction already seen. KeyGen outputs a personal uniformly random public-key matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and an associated secret-key (trapdoor) matrix (with small coefficients) $\boldsymbol{S} \in \mathbb{Z}^{m \times m}$ such that $\boldsymbol{AS} \equiv 0 \bmod q$. Moreover, it also chooses a FDH $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{Z}_q^m$. $\text{Sign}_{sk}(\boldsymbol{\mu})$ takes as input $\boldsymbol{\mu} \in \mathbb{Z}^m$ and outputs a signature $\boldsymbol{\sigma}$, independent of $\boldsymbol{S}$, such that $\boldsymbol{A\sigma} = H(\boldsymbol{\mu}) \bmod q$. $\text{Verify}_{pk}(\boldsymbol{\mu}, \boldsymbol{\sigma})$ returns 1 if and only if $\boldsymbol{\sigma}$ is in the domain and $\boldsymbol{A\sigma} = H(\boldsymbol{\mu}) \bmod q$, 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 $(\boldsymbol{A}, \boldsymbol{S})$, all messages $\boldsymbol{\mu}$ and all signatures $\boldsymbol{\sigma}$, $\boldsymbol{A\sigma} = H(\boldsymbol{\mu}) \bmod q$.

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 had been 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 $(\boldsymbol{A}, \boldsymbol{AS})$ whereas this scheme uses a public-key $(\boldsymbol{A}, \boldsymbol{AS} + \boldsymbol{G})$ for some matrix $\boldsymbol{G}$. The trapdoor for GPV is the basis matrix $\boldsymbol{A}$ for a lattice. However, for Micciancio and Peikert's scheme the trapdoor $\boldsymbol{A}$ is derived from a transformation of the fixed, public lattice denoted by the 'gadget' $\boldsymbol{G}$. 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 and an interested reader should refer to Section 6.2.2 in [Micciancio and Peikert 2012] for a full description.

3.3. Fiat-Shamir Signatures

An alternative way of constructing a DSS is to first build an identification scheme of a certain form, then converting it into a DSS by means of the Fiat-Shamir transformation [Fiat and Shamir 1986; Abdalla et al. 2002]. 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 Schnorr's protocol [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; 2012; Güneysu et al. 2012; Abdalla et al. 2012; Bai and Galbraith 2014; 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 [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 [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 $\boldsymbol{S}$ with small coefficients, and the public-key as the pair $(\boldsymbol{A}, \boldsymbol{T})$ where $\boldsymbol{A}$ is an $n \times m$ matrix with entries chosen uniformly at random from $\mathbb{Z}_q$ and $\boldsymbol{T} \equiv \boldsymbol{AS} \bmod q$. 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 $\boldsymbol{\mu}$ the signing algorithm first chooses a random $\boldsymbol{y}$ from some discrete Gaussian distribution, then computes $\boldsymbol{c} = H(\boldsymbol{Ay} \bmod q, \boldsymbol{\mu})$ where the (potential) signature is the pair $(\boldsymbol{z}, \boldsymbol{c})$ such that $\boldsymbol{z} = \boldsymbol{Sc} + \boldsymbol{y}$, which is sent to the verifier should it pass the rejection stage. Finally, the verification algorithm checks that $\|\boldsymbol{z}\|$ is small and that $\boldsymbol{c} = H(\boldsymbol{Az} - \boldsymbol{Tc} \bmod q, \boldsymbol{\mu})$. The details of the discrete Gaussian stage and the importance of the restrictions on $\boldsymbol{z}$ will become evident later.

In many cases (such as [Güneysu et al. 2012; Ducas et al. 2013; Bai and Galbraith 2014]) the signature is considered as $(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{c})$, allowing the shortening of $\boldsymbol{z}_1$ or $\boldsymbol{z}_2$, which motivates various compression techniques. The scheme by Bai and Galbraith [2014] is based on the LWE signature scheme LYU, whereby $\boldsymbol{z}_2$ is actually omitted from the scheme entirely. This is essentially achieved by adapting the scheme so that proof of knowledge of the pair $(\boldsymbol{s}, \boldsymbol{e})$ for the LWE public-key $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{As} + \boldsymbol{e} \bmod q)$ only requires knowledge of $\boldsymbol{s}$.

The current state-of-the-art in lattice-based DSSs is the proposed scheme by Ducas et al. [2013] named BLISS. The main contribution of this work is the significant improvement in the rejection sampling stage. 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 $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{R}$ and a public-key pair $(\boldsymbol{a}, \boldsymbol{t})$ where $\boldsymbol{a} \in \mathcal{R}$ is chosen at random and $\boldsymbol{t} = \boldsymbol{as}_1 + \boldsymbol{s}_2$. The signer then randomly chooses $\boldsymbol{y}_1, \boldsymbol{y}_2 \in D_\sigma^m$ and sends $\boldsymbol{u} = \boldsymbol{ay}_1 + \boldsymbol{y}_2$ to the verifier who returns a sparse $\boldsymbol{c} \in \mathcal{R}$. The signer then calculates $\boldsymbol{z}_i = \boldsymbol{y}_i + \boldsymbol{s}_i\boldsymbol{c}$ (for $i \in \{1, 2\}$) and sends $\boldsymbol{z}_1, \boldsymbol{z}_2$ to the verifier where it is checked that $\|\boldsymbol{z}_i\|$ are small and $\boldsymbol{az}_1 + \boldsymbol{z}_2 - \boldsymbol{tc} = \boldsymbol{u}$.

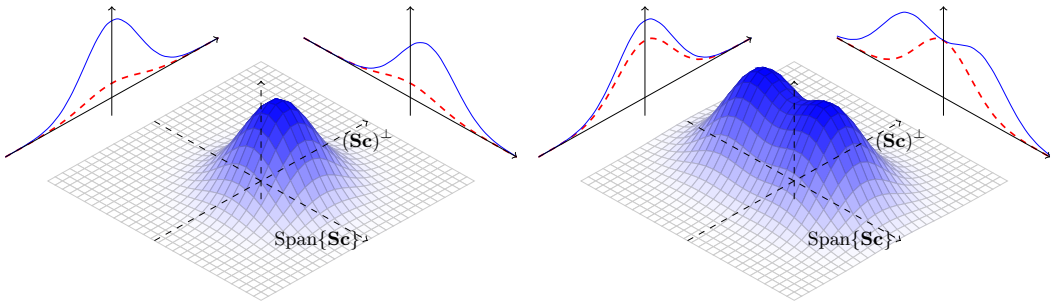

Fig. 1: The graphs show the improvement bimodal Gaussians have to the rejection sampling stage. The left (a) showing the LYU [Lyubashevsky 2012] scheme and the right (b) showing the BLISS [Ducas et al. 2013] scheme. The distribution of $\boldsymbol{z}$ is shown in blue, fixing $\boldsymbol{Sc}$ and over the space of all $\boldsymbol{y}$ in (a) and all $(b, \boldsymbol{y})$ 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).

Using this scheme as it is, there is an inherent vulnerability in the values $\boldsymbol{z}_i$ sent to the verifier. As stated, $\boldsymbol{y}_i$ is chosen from the distribution D^m_σ, therefore an adversary knows the distribution of $\boldsymbol{z}_i$ since they will follow the distribution of $\boldsymbol{y}_i$ skewed by the addition of $\boldsymbol{s}_i\boldsymbol{c}$, which is where the secret-key becomes susceptible. This can be rectified by adapting the distribution of $\boldsymbol{z}_i$ from $D^m_{\boldsymbol{Sc},\sigma}$ so that it follows the same distribution as $\boldsymbol{y}_i \sim D^m_\sigma$. 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 $D^m_\sigma(\boldsymbol{x}) \leq M \cdot D^m_{\boldsymbol{Sc},\sigma}(\boldsymbol{x})$ for all $\boldsymbol{x}$, 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 1, illustrating how the probability of acceptance significantly increases with the introduction of bimodal Gaussians. Further comparing the repetition rates in other schemes (see Table I for GLP, Table II 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 $\boldsymbol{z}_i$ such that $\boldsymbol{z}_i = \boldsymbol{y}_i + (-1)^b \boldsymbol{s}_i\boldsymbol{c}$. Therefore, $\boldsymbol{z}_i$ will follow the discrete bimodal Gaussian $\frac{1}{2}D^m_{\boldsymbol{Sc},\sigma}(\boldsymbol{x}) + \frac{1}{2}D^m_{-\boldsymbol{Sc},\sigma}(\boldsymbol{x})$. With some other small alterations in the

signing and verifying stages, a completely practical and secure lattice-based DSS is achieved.

4. IMPLEMENTATION OF LATTICE-BASED SIGNATURES

In this section the implementation challenges and common building blocks necessary to realise practical lattice-based signature schemes are examined and discussed.

4.1. On the Instantiation of GPV-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 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 III) provides 100-bit security and uses the FLINT and GSL libraries for basic arithmetic.

4.2. 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 by implementers 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.

4.2.1. GLP. 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 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 $(\boldsymbol{a}, \boldsymbol{a}\boldsymbol{s}_1 + \boldsymbol{s}_2)$, where $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}$ and $\boldsymbol{s}_1, \boldsymbol{s}_2 \xleftarrow{\$} \mathcal{R}_1$. However, the aggressive compression is a source of insecurity and in the full version of [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, Figure 2 shows the full description of the KeyGen, Sign and Verify algorithms of GLP and parameters are given in Table I with key and signature sizes being listed in Table III. The secret-keys of the scheme are the random polynomials $\boldsymbol{s}_1, \boldsymbol{s}_2$ and the public-key is $(\boldsymbol{a}, \boldsymbol{t})$, where $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}$ and $\boldsymbol{t} \leftarrow \boldsymbol{a}\boldsymbol{s}_1 + \boldsymbol{s}_2$. To sign a message μ, two 'masking' polynomials $\boldsymbol{y}_1, \boldsymbol{y}_2 \xleftarrow{\$} \mathcal{R}_k$ are chosen uniformly at random and $\boldsymbol{c} \leftarrow \mathrm{H}(\boldsymbol{a}\boldsymbol{y}_1 + \boldsymbol{y}_2, \mu)^{(1)}$ is computed where $\boldsymbol{a}\boldsymbol{y}_1 + \boldsymbol{y}_2$ is the most expensive operation during the

Table I: The GLP [Güneysu et al. 2012] Signature Scheme Parameters.

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

Algorithm KeyGen()

1: Return $(pk = (\boldsymbol{a}, \boldsymbol{t}), sk = (\boldsymbol{s}_1, \boldsymbol{s}_2))$ where $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}$, $\boldsymbol{s}_1, \boldsymbol{s}_2 \xleftarrow{\$} \mathcal{R}_1$ and $\boldsymbol{t} \leftarrow \boldsymbol{a}\boldsymbol{s}_1 + \boldsymbol{s}_2$

Alg. Sign$(\mu, \boldsymbol{a}, \boldsymbol{s}_1, \boldsymbol{s}_2)$

1: $\boldsymbol{y}_1, \boldsymbol{y}_2 \xleftarrow{\$} \mathcal{R}_k$
2: $\boldsymbol{c} \leftarrow \mathrm{H}\left((\boldsymbol{a}\boldsymbol{y}_1 + \boldsymbol{y}_2)^{(1)}, \mu\right)$
3: $\boldsymbol{z}_1 \leftarrow \boldsymbol{s}_1\boldsymbol{c} + \boldsymbol{y}_1, \boldsymbol{z}_2 \leftarrow \boldsymbol{s}_2\boldsymbol{c} + \boldsymbol{y}_2$
4: if $\boldsymbol{z}_1$ or $\boldsymbol{z}_2 \notin \mathcal{R}_{k-32}$, then go to step 1
5: $\boldsymbol{z}'_2 \leftarrow$ Compress $(\boldsymbol{a}\boldsymbol{z}_1 - \boldsymbol{t}\boldsymbol{c}, \boldsymbol{z}_2, q, k-32)$
6: if $\boldsymbol{z}'_2 = \perp$, then go to step 1
7: Return $(\boldsymbol{z}_1, \boldsymbol{z}'_2, \boldsymbol{c})$

Alg. Verify$(\mu, \boldsymbol{z}_1, \boldsymbol{z}'_2, \boldsymbol{c}, \boldsymbol{a}, \boldsymbol{t})$

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

Fig. 2: The GLP Signature Scheme.

signing procedure. Note that the hash function is only evaluated on the "higher order" bits[2] of the coefficients of the input, denoted by the $^{(1)}$ notation. The actual signature $\boldsymbol{z}_1, \boldsymbol{z}_2$ is computed as $\boldsymbol{z}_i \leftarrow \boldsymbol{s}_i\boldsymbol{c} + \boldsymbol{y}_i$ where the polynomial $\boldsymbol{c}$ 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 $\boldsymbol{c}$, $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are small and contain a lot of zeros (i.e., they are sparse) no modular reduction modulo q is necessary when computing $\boldsymbol{s}_1\boldsymbol{c}$ and $\boldsymbol{s}_2\boldsymbol{c}$. But before sending the signature, the low-cost rejection sampling step must be performed where the signature is only sent if and only if $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathcal{R}_{k-32}$. There 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 Compress$(\boldsymbol{a}\boldsymbol{z}_1 - \boldsymbol{t}\boldsymbol{c}, \boldsymbol{z}_2, q, k-32)$ function is the reason for the relatively short signatures as a large amount of $\boldsymbol{z}_2$ is removed (the scheme also works without the compression). It encodes the carries that would have been caused by $\boldsymbol{z}_2$ into $\boldsymbol{z'}_2$ and ensures correctness so that $(\boldsymbol{a}\boldsymbol{y}_1 + \boldsymbol{y}_2)^{(1)} = (\boldsymbol{a}\boldsymbol{z}_1 + \boldsymbol{z'}_2\boldsymbol{t}\boldsymbol{c})^{(1)}$.

The GLP scheme has currently been implemented on reconfigurable hardware [Güneysu et al. 2012], CPUs [Güneysu et al. 2013] and microcontrollers [Boorghany and Jalili 2014] (see Section 5 for further discussions).

4.2.2. BLISS. The most efficient instantiation of the BLISS signature scheme [Ducas et al. 2013] is based on ideal-lattices [Lyubashevsky et al. 2010; 2013a] with the BLISS KeyGen, Sign and Verify algorithms given in Figure 3. Parameters are listed in Table II and key and signature sizes are given in Table III.

The generation of keys involves uniform sampling of two small, sparse polynomials $\boldsymbol{f}$ and $\boldsymbol{g}$, computation of the rejection condition $N_\kappa(\boldsymbol{S})$, and the computation of $\boldsymbol{f}^{-1}$.

[2]In this context, higher-order bits means roughly the most significant bits.

Table II: The BLISS [Ducas et al. 2013] Signature Scheme Parameters.

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
Dropped bits d in $\boldsymbol{z}_2$	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

Algorithm KeyGen()

1: Choose $\boldsymbol{f}, \boldsymbol{g}$ 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 $\{\pm 2\}$
2: $\boldsymbol{S} = (\boldsymbol{s}_1, \boldsymbol{s}_2)^t \leftarrow (\boldsymbol{f}, 2\boldsymbol{g} + 1)^t$
3: Compute rejection condition $N_\kappa(\boldsymbol{S})$ that accepts approx. 25% of all keys
4: $\boldsymbol{a}_q = (2\boldsymbol{g} + 1)/\boldsymbol{f} \bmod q$ (restart if $\boldsymbol{f}$ is not invertible)
5: Return $(pk = \boldsymbol{A}, sk = \boldsymbol{S})$ where $\boldsymbol{A} = (\boldsymbol{a}_1 = 2\boldsymbol{a}_q, q - 2) \bmod 2q$

Alg. Sign(μ,$\boldsymbol{A}$,$\boldsymbol{S}$)

1: $\boldsymbol{y}_1, \boldsymbol{y}_2 \leftarrow D_{\mathbb{Z}^n, \sigma}$
2: $\boldsymbol{u} = \zeta \cdot \boldsymbol{a}_1 \cdot \boldsymbol{y}_1 + \boldsymbol{y}_2 \bmod 2q$
3: $\boldsymbol{c} \leftarrow H(\lfloor \boldsymbol{u} \rceil_d \bmod p, \mu)$
4: Choose a random bit b
5: $\boldsymbol{z}_1 \leftarrow \boldsymbol{y}_1 + (-1)^b \boldsymbol{s}_1 \boldsymbol{c}$
6: $\boldsymbol{z}_2 \leftarrow \boldsymbol{y}_2 + (-1)^b \boldsymbol{s}_2 \boldsymbol{c}$
7: Continue with probability $1 \Big/ \left(M \exp\left(-\frac{\|\boldsymbol{S}\boldsymbol{c}\|^2}{2\sigma^2} \right) \cosh\left(\frac{\langle \boldsymbol{z}, \boldsymbol{S}\boldsymbol{c} \rangle}{\sigma^2} \right) \right)$ otherwise restart
8: $\boldsymbol{z}_2^\dagger \leftarrow (\lfloor \boldsymbol{u} \rceil_d - \lfloor \boldsymbol{u} - \boldsymbol{z}_2 \rceil_d) \bmod p$
9: Return $(\boldsymbol{z}_1, \boldsymbol{z}_2^\dagger, \boldsymbol{c})$

Alg. Verify(μ,$\boldsymbol{A}$,($\boldsymbol{z}_1$,$\boldsymbol{z}_2^\dagger$,$\boldsymbol{c}$))

1: if $\|(\boldsymbol{z}_1 | 2^d \cdot \boldsymbol{z}_2^\dagger)\|_2 > B_2$ then reject
2: if $\|(\boldsymbol{z}_1 | 2^d \cdot \boldsymbol{z}_2^\dagger)\|_\infty > B_\infty$ then reject
3: Accept iff $\boldsymbol{c} = H\big(\lfloor \zeta \cdot \boldsymbol{a}_1 \cdot \boldsymbol{z}_1 + \zeta \cdot q \cdot \boldsymbol{c} \rceil_d + \boldsymbol{z}_2^\dagger \bmod p, \mu \big)$

Fig. 3: The Bimodal Lattice Signature Scheme [Ducas et al. 2013].

Inverting $\boldsymbol{f}$ makes KeyGen significantly more complex, particularly in comparison to the equivalent stage in GLP.

For the signing stage, two polynomials $\boldsymbol{y}_1, \boldsymbol{y}_2$ 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 [Dwarakanath and Galbraith 2014]. Section 4.3.2 describes the ongoing research in this area. The calculation of $\boldsymbol{u}$ is simplified since the computation of $\boldsymbol{a}\boldsymbol{y}_1$ is performed in an FFT-enabled ring using modulus q, instead of $2q$. The d most significant bits of $\boldsymbol{u}$ are then hashed with the message μ with the output being interpreted as a polynomial $\boldsymbol{c}$. Such as in GLP, $\boldsymbol{c}$ is also sparse and small but generated differently and more efficiently from the output of the hash function. The polynomial $\boldsymbol{c}$ is then multiplied by the secret-key polynomials $\boldsymbol{s}_1, \boldsymbol{s}_2$, where the polynomials $\boldsymbol{y}_1, \boldsymbol{y}_2$ 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 $\boldsymbol{z}_2$ is then compressed and $(\boldsymbol{z}_1, \boldsymbol{z}_2^{\dagger}, \boldsymbol{c})$ 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 $\boldsymbol{c}$ from the signature. It should be evident from this description that the most costly computational components to BLISS are the dense polynomial multiplication, discrete Gaussian sampling, and sparse multiplication stages [Pöppelmann et al. 2014].

4.3. Building Blocks

GLP and BLISS have been introduced as the most promising proposals for lattice-based signatures. Before the full performances of published implementations are compared in Section 5, research on common building blocks used in GLP or BLISS are examined. The reasons are that these building blocks as essential for the understanding of the performance of the signature schemes and that they could also be used as a basis for the implementation of existing or new Fiat-Shamir style lattice-based signatures (e.g. a ring instantiation of [Bai and Galbraith 2014]).

4.3.1. Polynomial Multiplication. Comparable to point multiplication for ECC and exponentiation for RSA, polynomial multiplication is the basic operation in ideal lattice-based cryptography and thus subject of various optimisation efforts. While addition and subtraction in $\mathbb{Z}_q[\boldsymbol{x}]/\langle x^n + 1\rangle$ are easy to realise with $\mathcal{O}(n)$ primitive operations in $\mathbb{Z}_q$, polynomial multiplication is much more complicated[3]. When computing the product $\boldsymbol{c} = \boldsymbol{a}\cdot\boldsymbol{b}$ in $\mathbb{Z}_q[\boldsymbol{x}]/\langle x^n + 1\rangle$ the trick that $\boldsymbol{x}^n \equiv -1$ can be used for instant reduction mod $\langle \boldsymbol{x}^n + 1\rangle$. This leads to the obvious schoolbook approach

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

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 $\boldsymbol{b}[j]$ is fixed and the row, i.e., the inner products $\boldsymbol{c}[i+j \bmod n] = \boldsymbol{c}[i+j \bmod n] + (-1)^{\left\lfloor \frac{i+j}{n} \right\rfloor}\boldsymbol{a}[i]\cdot \boldsymbol{b}[j] \bmod q$ for $i, j \in \{0, \ldots, n-1\}$ are computed. Once a row is completed, the next $\boldsymbol{b}[j+1]$ is fixed. Another approach is column-wise multiplication, where partial products are used to sum up columns of $\boldsymbol{c}$. Thus, for column ℓ it is then necessary to compute $\boldsymbol{c}[\ell] = \sum_{i=0}^{n-1} (-1)^{1-\left\lfloor \frac{n+\ell-i}{n} \right\rfloor} \boldsymbol{a}[i]\boldsymbol{b}[n+\ell-i \bmod n] \bmod q$.

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 $\boldsymbol{ay}_1 + \boldsymbol{y}_2$. The idea was that schoolbook multipliers are simple to realise and that they can profit from different operand sizes as $\boldsymbol{a} \xleftarrow{\$} \mathcal{R}$ and $\boldsymbol{y} \xleftarrow{\$} \mathcal{R}_k$. When implemented they basically consist of a block memory (BRAM), a DSP for multiplication, and a reduction circuit and can thus run within a separate and high-frequency clock domain. However, the number of required multipliers is quite large for high-speed requirements (see also [Pöppelmann and Güneysu 2012] for instantiations of schoolbook multipliers for different parameters n and q).

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

In the case where polynomials are sparse (polynomials with a lot of zero coefficients like $\boldsymbol{c}$ from GLP or BLISS), or have only many small coefficients (e.g., just -1/0/1 coefficients like $\boldsymbol{s}_1, \boldsymbol{s}_2$ 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 ($\boldsymbol{z}_1$ or $\boldsymbol{z}_2 \notin \mathcal{R}_{k-32}$). Pöppelmann et al. [2014] further optimise and parallelise the column-wise schoolbook multiplier. Even though row-wise multiplication would allow to profit from the sparsity of both operands ($\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{c}$), more memory accesses would be necessary to add and store inner products. As all memory beyond dual-port RAMs are extremely expensive this also prevents efficient configurable parallelisation. As a consequence, the implementation consists of a configurable number of cores C which perform column-wise multiplication. Each core stores the secret-key (either $\boldsymbol{s}_1$ or $\boldsymbol{s}_2$) efficiently in a distributed RAM and accumulates inner products in a small multiply-accumulate unit. Positions of $\boldsymbol{c}$ are fed simultaneously into the cores. Note also that exploiting the spareness of polynomials already played a role in implementations of the NTRU public-key encryption schemes (see [Kamal and Youssef 2009]).

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 2010] 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 $q \equiv 1 \bmod 2n$[4]. The generic forward $\text{NTT}_\omega(\boldsymbol{a})$ of a sequence $\{\boldsymbol{a}[0], .., \boldsymbol{a}[n-1]\}$ to $\{\boldsymbol{A}[0], \ldots, \boldsymbol{A}[n-1]\}$ with elements in $\mathbb{Z}_q$ and length n is defined as $\boldsymbol{A}[i] = \sum_{j=0}^{n-1} \boldsymbol{a}[j]\omega^{ij} \bmod q,\ i = 0, 1, \ldots, n-1$ with the inverse $\text{NTT}_\omega^{-1}(\boldsymbol{A})$ using ω^{-1} instead of ω [Winkler 1996]. Thus the reduction by $x^n + 1$ is basically for 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 2010] 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 Emeliyanenko [2009] where the author provides an implementation of polynomial multiplication on graphics hardware (GPUs). An FPGA implementation of the NTT for polynomial multiplication is described by Cheng et al. [2005].

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 [Lindner and Peikert 2011]. 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 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 [Göttert et al. 2012] are that multiplication in $\mathbb{Z}_q$ is performed using an embedded multiplier (DSP) and block memory (BRAM) to store polynomials. In this case the block memory access patterns become an issue, as the pipelined NTT butterfly requires two read operations from the memory and two write operations into the memory for best utilisation. As common Xilinx BRAMs have only two ports, a polynomial is stored in two BRAMs where all coefficients with an equal parity address are placed into one BRAM and all coefficients with unequal parity into another BRAM to prevent access conflicts and thus four ports. The twiddle factors (ω, ψ) required for the trans-

[4]The NTT can also be defined for composite moduli, but we only consider the prime case in this survey.

formation are stored in a table and not computed on-the-fly. 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 addition. 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 concatenated storage of multiple coefficients in one memory address. The required powers of ω and ψ were generated on-the-fly in an efficient manner and the design can be configured to use one (2DSP) or two dedicated multipliers (3DSP). 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. [2014] to realise lattice-based encryption. The address generation of the NTT algorithm was rearranged and thus repeated multiplications to generated 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 currently represents the state-of-the-art for polynomial multiplication for ideal lattice-based cryptography. An optimisation of an 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 (see Pease [1968]).

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 [2014] also use the NTT as 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 in Roy et al. [2014].

4.3.2. Sampling. Sampling from the one dimensional discrete Gaussian distribution $D_{\mathbb{Z},\sigma}$ is an important building blocks for BLISS and other signature schemes 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 $\rho_\sigma(x) = \exp(\frac{-x^2}{2\sigma^2})$ 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 $u \in \{-\tau\sigma, ..., \tau\sigma\}$ (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. See [Weiden et al. 2013] for an implementation in software and some optimisations in [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 a probability of $c \in [0, 1]$ and zero otherwise. Sampling from this distribution is easy by lazily evaluation if $y < c$ for a uniformly random $y \in [0, 1)$ and pre-computed c. The general idea (for more details see [Ducas et al. 2013]) 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 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 full implementation of the sampler in hardware can be found in the full version of [Pöppelmann et al. 2014] and a microcontroller implementation of BLISS is provided by Oder et al. [2014]. Boorghany et al. [2014] and Boorghany and Jalili [2014] implement GLP and BLISS as identification protocols on constrained devices using the CDT and Bernoulli sampling.

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 regular 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, but an instantiation might require a special purpose circuit for the rejection sampling (probably computing exponentiations up to a high precision) which is supposed to be very costly in terms of area consumption.

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 1's in column n of the probability matrix (starting with column 0). The row in which a 1 appears is used as 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. [2013] (see [Roy et al. 2014] 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) [Peikert 2010]. For this method, a table of cumulative probabilities $p_z = \Pr(x \leqslant z : x \leftarrow D_\sigma)$ are computed for integers $z \in [-\tau\sigma, ..., \tau\sigma]$ 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 and in a lazy manner. The CDT approach is compared to other software samplers by Buchmann et al. [2013] and 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 and 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, and evaluation of the impact for others samplers is not available. However, for the CDT sampler it seems particularly suitable as the table size was reduced while still maintaining high speed.

5. SUMMARY AND EVALUATION

In this section a summary is provided of implementation results for practical lattice-based signature schemes and also results for classical schemes from the literature. As discussed in Section 3.1, there are currently no practical instantiations of the GGH [Goldreich et al. 1996] signature schemes and implementations of NTRUSign [Hoffstein et al. 2003] like [Driessen et al. 2008] are vulnerable to cryptanalysis, so they will not be consider further. Lattice-based schemes investigated here for which implementation results are available are GPV [Gentry et al. 2008; Micciancio and Peikert 2012], LYU [Lyubashevsky 2012], GLP [Güneysu et al. 2012] and BLISS [Ducas et al. 2013]. For a quick overview, all schemes considered for evaluation; their secret-key, public-key and signatures sizes as well as available software (CPU) results are summarised in Table III, however since these benchmarks are not all on the same platform they are not all directly comparable (similarly in Table IV). The fastest scheme with regard to signing and also with the smallest signature (5.6 kb) is currently BLISS (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 implementation by Güneysu et al. [2013] using assembly optimisation and vectorisation (i.e. AVX extensions). As verification almost only requires polynomial multiplication, the vectorised GLP implementation is twice as fast as BLISS.

Table III: A summary of lattice-based DSSs and schemes based on classical assumptions. Most results are taken from Ducas et al. [2013] and were benchmarked on an Intel Core i7 at 3.4 GHz, 32GB RAM with openssl 1.0.1c. The GLP-I [Güneysu et al. 2013] (Intel Core i5-3210M), LYU [Weiden et al. 2013], GPV [Bansarkhani and Buchmann 2013] (both originally AMD Opteron 2.3 GHz) performance has been scaled to 3.4 GHz based on cycle counts.

Scheme	Security	Sign. Size	sk Size	pk Size	Sign./s	Ver./s
GPV	100-bits	240 kb	191 kb	300 kb	48	370
LYU	100-bits	103 kb	103 kb	65 kb	36	260
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
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

Thus in the future, it is expected that improvements regarding BLISS as 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 Gaussian sampler and number of rejections. As Gaussian sampling is not needed for verification, the runtime of verification is basically independent of the security level. The LYU implementation 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 or BLISS.

Regarding those 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 running at 168 MHz; the device produces 28 signing, 167 verification and 0.46 key generation operations per second. Boorghany et al. [2014] and Boorghany and Jalili [2014] provide an implementation of GLP and BLISS used as identification scheme on 8-bit 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 IV. While the speed of the GLP implementation, with roughly 1000 signing and verification operations per second, is good in comparison with classical

Table IV: A summary of hardware instantiations of DSSs on Virtex-5(V5) and Spartan-6 (S6), comparing those based on lattice problems (GLP [Güneysu et al. 2012] and BLISS-I [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

schemes, the implementation in [Güneysu et al. 2012] and particularly the usage of schoolbook multiplication is suboptimal given works on fast multiplication like [Roy et al. 2014]. 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. [2014] 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].

6. CONCLUSION

In this paper, motivations for the future development of cryptography in a post-quantum world, specifically developments in lattice-based cryptography, are discussed. Furthering this, introductions are given to the theory of lattices, lattice-based cryptography and digital signature schemes so that a thorough survey of lattice-based digital signature schemes can be presented.

With respect to the digital signature schemes, the first shown to be based on the hardness of lattice problems are the GGH/NTRUSign signatures and known not to be secure hence their exclusion from formal description. The next class of schemes, hash-and-sign 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] (≈ 8000 signatures per second) and on microcontrollers by Oder et al. [2014] (28 signatures per second), in both instances outperforming RSA and ECC for compa-

rable security levels, lattice-based digital signature schemes are now at a stage to be considered for real-world applications.

7. FUTURE WORK

Based on the survey of DSSs conducted in this paper, the main area of future research for lattice-based DSSs seems to be the optimisation and implementation of schemes based on the Fiat-Shamir model. Specifically BLISS, which 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 2014]. Integrating the scheme with respect to highly-optimised software is also a possible area for future work (similar to [Güneysu et al. 2013]).

An interesting area of future research would be to evaluate the practical implications of the compression algorithms by Bai and Galbraith [2014] or optimisations to BLISS by Ducas [2014]. Moreover, it is not clear whether theoretical improvements to GPV-based schemes can make them competitive.

Additionally, further research is needed into the parameters (and security analyses) of these schemes. This would build upon research such as [Rückert and Schneider 2010], which would mean parameter selection becoming much more explicit in lattice-based cryptography.

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. 2014], with most instantiations adapting from specialist techniques by Karatsuba and Ofman [1963] $(\mathcal{O}(n^{\log 3}))$, Cooley and Tukey [1965] $(\mathcal{O}(n \log n))$, Pollard [1971] $(\mathcal{O}(n \log n))$ or 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; 2013; Göttert et al. 2012; Aysu et al. 2013] and on lightweight devices [Boorghany and Jalili 2014; Pöppelmann and Güneysu 2014; Oder et al. 2014].

Another module pertaining to one of the more computationally expensive in hardware is the Gaussian sampling stage. Dwarakanath and Galbraith [2014] and Roy et al. [2013] look into different approaches to efficiently compute such a stage for constrained devices. As shown by [Pöppelmann et al. 2014] and [Oder et al. 2014] the CDT approach is best suited for larger devices with the Bernoulli approach showing efficiencies on smaller devices. Due to its computational importance, further research into making this stage more efficient could result in significant improvements overall.

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. 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 (a first work is [Roy et al. 2014]). It is anticipated that there may be a particular vulnerability with respect to algorithmic parts with variable runtime, for instance Gaussian and rejection sampling, which are major components of many lattice-based algorithms.

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.

ACKNOWLEDGMENTS

The authors wish to thank Léo Ducas for the constructive advice, Tancrède Lepoint for permission to use the diagrams seen in Figure 1, and the anonymous reviewers for their careful reading of the paper and their diligent comments. This work was partially funded by the European Union H2020 SAFEcrypto project (grant no. 644729), German Research Foundation (DFG), and DFG Research Training Group GRK 1817/1.

REFERENCES

Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. 2002. From Identification to Signatures via the Fiat-Shamir Transform: Minimizing Assumptions for Security and Forward-Security. In *EUROCRYPT*. 418–433.

Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. 2012. Tightly-Secure Signatures from Lossy Identification Schemes. In *EUROCRYPT*. 572–590.

Shweta Agrawal, Dan Boneh, and Xavier Boyen. 2010. Efficient Lattice (H)IBE in the Standard Model. In *EUROCRYPT*. 553–572.

Miklós Ajtai. 1996. Generating Hard Instances of Lattice Problems (Extended Abstract). In *STOC*. 99–108.

Miklós Ajtai, Ravi Kumar, and D. Sivakumar. 2001. A Sieve Algorithm for the Shortest Lattice Vector Problem. In *STOC*. 601–610.

Joël Alwen and Chris Peikert. 2011. Generating Shorter Bases for Hard Random Lattices. *Theory Comput. Syst.* 48, 3 (2011), 535–553.

Aydin Aysu, Cameron Patterson, and Patrick Schaumont. 2013. Low-Cost and Area-Efficient FPGA Implementations of Lattice-based Cryptography. In *HOST*. 81–86.

Shi Bai and Steven D. Galbraith. 2014. An Improved Compression Technique for Signatures Based on Learning with Errors. In *CT-RSA*. 28–47.

Rachid El Bansarkhani and Johannes Buchmann. 2013. Improvement and Efficient Implementation of a Lattice-Based Signature Scheme. In *Selected Areas in Cryptography*. 48–67.

Mihir Bellare and Phillip Rogaway. 1993. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM CCS*. 62–73.

Richard E. Blahut. 2010. *Fast Algorithms for Signal Processing*. Cambridge University Press.

Dan Boneh, Amit Sahai, and Brent Waters. 2011. Functional Encryption: Definitions and Challenges. In *TCC*. Vol. 6597.

Dan Boneh and Mark Zhandry. 2013. Secure Signatures and Chosen Ciphertext Security in a Quantum Computing World. In *CRYPTO (2)*. 361–379.

Ahmad Boorghany and Rasool Jalili. 2014. Implementation and Comparison of Lattice-based Identification Protocols on Smart Cards and Microcontrollers. *IACR Cryptology ePrint Archive* 2014 (2014), 78.

Ahmad Boorghany, Siavash Bayat Sarmadi, and Rasool Jalili. 2014. On Constrained Implementation of Lattice-based Cryptographic Primitives and Schemes on Smart Cards. *IACR Cryptology ePrint Archive* 2014 (2014), 514.

Xavier Boyen. 2010. Lattice Mixing and Vanishing Trapdoors: A Framework for Fully Secure Short Signatures and More. In *PKC*. 499–517.

Xavier Boyen. 2013. Attribute-Based Functional Encryption on Lattices. In *TCC*. 122–142.

Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. 2013. Classical Hardness of Learning with Errors. In *STOC*. 575–584.

Johannes Buchmann, Daniel Cabarcas, Florian Göpfert, Andreas Hülsing, and Patrick Weiden. 2013. Discrete Ziggurat: A Time-Memory Trade-Off for Sampling from a Gaussian Distribution over the Integers. In *Selected Areas in Cryptography*. 402–417.

Johannes Buchmann, Richard Lindner, Markus Rückert, and Michael Schneider. 2009. Post-Quantum Cryptography: Lattice Signatures. *Computing* 85, 1-2 (2009), 105–125.

Jan Camenisch, Gregory Neven, and Markus Rückert. 2012. Fully Anonymous Attribute Tokens from Lattices. In *SCN*. 57–75.

Donald Donglong Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray C. C. Cheung, Derek Pao, and Ingrid Verbauwhede. 2014. High-speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems. *IACR Cryptology ePrint Archive* 2014 (2014), 646.

Lo Sing Cheng, Ali Miri, and Tet Hin Yeap. 2005. Efficient FPGA implementation of FFT based multipliers. In *Electrical and Computer Engineering, 2005*. 1300–1303.

Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2014. Efficient Software Implementation of Ring-LWE Encryption. *IACR Cryptology ePrint Archive* 2014 (2014), 725.

James Cooley and John Tukey. 1965. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comp.* 19, 90 (1965), 297–301.

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

Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. 2013. The Fiat-Shamir Transformation in a Quantum World. In *ASIACRYPT (2)*. 62–81.

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

Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. 2003. Approximating CVP to Within Almost-Polynomial Factors is NP-Hard. *Combinatorica* 23, 2 (April 2003), 205–243.

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

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

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

Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. 2014. Efficient Identity-Based Encryption over NTRU Lattices. In *ASIACRYPT*. 22–41.

Léo Ducas and Daniele Micciancio. 2014. Improved Short Lattice Signatures in the Standard Model. In *CRYPTO*. 335–352.

Léo Ducas and Phong Q. Nguyen. 2012a. Faster Gaussian Lattice Sampling Using Lazy Floating-Point Arithmetic. In *ASIACRYPT*. 415–432.

Léo Ducas and Phong Q. Nguyen. 2012b. Learning a Zonotope and More: Cryptanalysis of NTRUSign Countermeasures. In *ASIACRYPT*. 433–450.

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

Pavel Emeliyanenko. 2009. Efficient Multiplication of Polynomials on Graphics Hardware. In *APPT*. 134–149.

Amos Fiat and Adi Shamir. 1986. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *CRYPTO*. 186–194.

Steven D. Galbraith. 2012. *Mathematics of Public-Key Cryptography*. Cambridge: Cambridge University Press. xiv. 452–459 pages.

Craig Gentry. 2009a. *A Fully Homomorphic Encryption Scheme*. Ph.D. Dissertation. Stanford University.

Craig Gentry. 2009b. Fully homomorphic encryption using ideal lattices. In *STOC*. 169–178.

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

Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors For Hard Lattices And New Cryptographic Constructions. In *STOC*. 197–206.

Craig Gentry and Michael Szydlo. 2002. Cryptanalysis of the Revised NTRU Signature Scheme. In *EUROCRYPT*. 299–320.

Oded Goldreich, Shafi Goldwasser, and Shai Halevi. 1996. Public-Key Cryptosystems from Lattice Reduction Problems. *Electronic Colloquium on Computational Complexity (ECCC)* 3, 56 (1996).

Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. 1988. A Digital Signature Scheme Secure Against Adaptive Chosen-message Attacks. *SIAM J. Comput.* 17, 2 (apr 1988), 281–308.

Samuel Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. 2010. A Group Signature Scheme from Lattice Assumptions. In *ASIACRYPT*. 395–412.

Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin A. Huss. 2012. On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes. In *CHES*. 512–529.

Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. 2012. Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems. In *CHES*. 530–547.

Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. 2013. Software Speed Records for Lattice-Based Signatures. In *PQCrypto*. 67–82.

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

Tamas Györfi, Octavian Cret, and Zalan Borsos. 2013. Implementing Modular FFTs in FPGAs - A Basic Block for Lattice-Based Cryptography. In *DSD*. 305–308.

Bettina Helfrich. 1985. Algorithms to Construct Minkowski Reduced and Hermite Reduced Lattice Bases. *Theor. Comput. Sci.* 41, 2-3 (Dec. 1985), 125–139.

Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. 2003. NTRUSign: Digital Signatures Using the NTRU Lattice. In *CT-RSA*. 122–140.

Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 1998. NTRU: A Ring-Based Public Key Cryptosystem. In *ANTS*. 267–288.

Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 2001. NSS: An NTRU Lattice-Based Signature Scheme. In *EUROCRYPT*. 211–228.

Abdel Alim Kamal and Amr M. Youssef. 2009. An FPGA implementation of the NTRUEncrypt cryptosystem. In *ICM*. 209–212.

Anatoly A. Karatsuba and Yuri Petrovich Ofman. 1963. Multiplication of Multidigit Numbers on Automata. *Soviet Physics Doklady* 7 (1963), 595–596.

Neal Koblitz. 1987. Elliptic Curve Cryptosystems. *Math. Comp.* (1987).

Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *CRYPTO*. 388–397.

Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. 2013. Lattice-Based Group Signatures with Logarithmic Signature Size. In *ASIACRYPT (2)*. 41–61.

Adeline Langlois and Damien Stehlé. 2014. Worst-case to average-case reductions for module lattices. *Designs, Codes and Cryptography* (2014).

Richard Lindner and Chris Peikert. 2011. Better Key Sizes (and Attacks) for LWE-Based Encryption. In *CT-RSA*. 319–339.

Vadim Lyubashevsky. 2009. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *ASIACRYPT*. 598–616.

Vadim Lyubashevsky. 2012. Lattice Signatures without Trapdoors. In *EUROCRYPT*. 738–755.

Vadim Lyubashevsky and Daniele Micciancio. 2009. On Bounded Distance Decoding, Unique Shortest Vectors, and the Minimum Distance Problem. In *CRYPTO*. 577–594.

Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. 2008. SWIFFT: A Modest Proposal for FFT Hashing. In *FSE*. 54–72.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors over Rings. In *EUROCRYPT*. 1–23.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013a. On Ideal Lattices and Learning with Errors over Rings. *J. ACM* 60, 6 (2013), 43.

Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2013b. A Toolkit for Ring-LWE Cryptography. In *EUROCRYPT*. 35–54.

James H. McClellan. 1976. Hardware realization of a Fermat number transform. *IEEE Transactions on Acoustics, Speech and Signal Processing* 24, 3 (Jun 1976), 216–225.

Carlos Aguilar Melchor, Xavier Boyen, Jean-Christophe Deneuville, and Philippe Gaborit. 2014. Sealing the Leak on Classical NTRU Signatures. In *PQCrypto 2014*. 1–21.

Daniele Micciancio. 2007. Generalized Compact Knapsacks, Cyclic Lattices, and Efficient One-Way Functions. *Comput. Complex.* 16, 4 (Dec. 2007), 365–411.

Daniele Micciancio. 2008. Efficient Reductions Among Lattice Problems. In *SODA*. 84–93.

Daniele Micciancio and Petros Mol. 2011. Pseudorandom Knapsacks and the Sample Complexity of LWE Search-to-Decision Reductions. In *CRYPTO*. 465–484.

Daniele Micciancio and Chris Peikert. 2012. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *EUROCRYPT*. 700–718.

Daniele Micciancio and Chris Peikert. 2013. Hardness of SIS and LWE with Small Parameters. In *CRYPTO (1)*. 21–39.

Daniele Micciancio and Oded Regev. 2004. Worst-Case to Average-Case Reductions Based on Gaussian Measures. In *FOCS*. 372–381.

Daniele Micciancio and Oded Regev. 2007. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.* 37, 1 (2007), 267–302.

Victor S Miller. 1986. Use of Elliptic Curves in Cryptography. In *CRYPTO*. 417–426.

Robert T. Moenck. 1976. Practical Fast Polynomial Multiplication. In *SYMSACC*. 136–148.

Phong Q. Nguyen and Oded Regev. 2009. Learning a Parallelepiped: Cryptanalysis of GGH and NTRU Signatures. *J. Cryptology* 22, 2 (2009), 139–160.

Henri Nussbaumer. 1980. *Fast Fourier transform and convolution algorithms*. Springer-Verlag.

Tobias Oder, Thomas Pöppelmann, and Tim Güneysu. 2014. Beyond ECDSA and RSA: Lattice-based Digital Signatures on Constrained Devices. In *DAC*. 1–6.

Marshall C. Pease. 1968. An Adaptation of the Fast Fourier Transform for Parallel Processing. *J. ACM* 15, 2 (April 1968), 252–264.

Chris Peikert. 2008. Public-Key Cryptosystems from the Worst-Case Shortest Vector Problem. *Electronic Colloquium on Computational Complexity (ECCC)* 15, 100 (2008).

Chris Peikert. 2010. An Efficient and Parallel Gaussian Sampler for Lattices. In *CRYPTO*. 80–97.

John M Pollard. 1971. The Fast Fourier Transform in a Finite Field. *Math. Comp.* 25, 114 (1971), 365–374.

Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. 2014. Enhanced Lattice-Based Signatures on Reconfigurable Hardware. In *CHES*. 353–370. Full version: https://eprint.iacr.org/2014/254.pdf.

Thomas Pöppelmann and Tim Güneysu. 2012. Towards Efficient Arithmetic for Lattice-Based Cryptography on Reconfigurable Hardware. In *LATINCRYPT*. 139–158.

Thomas Pöppelmann and Tim Güneysu. 2013. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. In *Selected Areas in Cryptography*. 68–85.

Thomas Pöppelmann and Tim Güneysu. 2014. Area Optimization of Lightweight Lattice-Based Encryption on Reconfigurable Hardware. In *ISCAS*. 2796–2799.

Oded Regev. 2005. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *STOC*. 84–93.

Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (2009).

Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. 2014. Compact and Side Channel Secure Discrete Gaussian Sampling. *IACR Cryptology ePrint Archive* 2014 (2014), 591.

Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. 2014. Compact Hardware Implementation of Ring-LWE Cryptosystems. In *CHES*. 371–391.

Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2013. High Precision Discrete Gaussian Sampling on FPGAs. In *Selected Areas in Cryptography*. 1–39.

Markus Rückert and Michael Schneider. 2010. Estimating the Security of Lattice-based Cryptosystems. *IACR Cryptology ePrint Archive* 2010 (2010), 137.

Claus-Peter Schnorr. 1989. Efficient Identification and Signatures for Smart Cards. In *CRYPTO*. 239–252.

Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* 26, 5 (Oct. 1997), 1484–1509.

Patrick Weiden, Andreas Hülsing, Daniel Cabarcas, and Johannes Buchmann. 2013. Instantiating Treeless Signature Schemes. *IACR Cryptology ePrint Archive* 2013 (2013), 65.

Franz Winkler. 1996. *Polynomial Algorithms in Computer Algebra (Texts and Monographs in Symbolic Computation)* (1 ed.). Springer.