CECS

CENTER FOR EMBEDDED & CYBER-PHYSICAL SYSTEMS

Software and Hardware Implementation of Lattice-Cased Cryptography Schemes

Hamid Nejatollahi, Nikil Dutt, Sandip Ray,

Francesco Regazzoni, Indranil Banerjee and Rosario Cammarota

Center for Embedded and Cyber-Physical Systems

University of California, Irvine

Irvine, CA 92697-2620, USA

{hnejatol}@uci.edu

CECS Technical Report <17-04>

<11> <09>, <2017>

Software and Hardware Implementation of Lattice-Cased Cryptography Schemes

HAMID NEJATOLLAHI, University of California Irvine
NIKIL DUTT, University of California Irvine
SANDIP RAY, NXP
FRANCESCO REGAZZONI, ALaRi
INDRANIL BANERJEE, Qualcomm Research
ROSARIO CAMMAROTA, Qualcomm Research

Advances in computing steadily erode computer security at its foundation, and call for fundamental innovations to strengthen current practices in computer security, specifically in applied cryptography, from theory to standardization to actual implementations. At the same time, the emergence of new computing paradigms, such as cloud computing, software defined networks and Internet of Everything, demands devices to adopt an increasing number of security standards - with a diverse set of cryptographic primitives. This in turn calls for innovations in security beyond the foundational aspects, down to the actual design and deployment of such primitives and protocols while satisfying conflicting design constraints such as latency, compactness, and energy efficiency.

The advent of Quantum computing threatens to break many classical cryptographic schemes, leading to innovations in public key cryptography that focus on post-quantum cryptography primitives and their protocols that are resistant to quantum computing threats. In particular, lattice-based cryptography is a promising post-quantum cryptography family, both in terms of foundational properties, as well as its application to both traditional and emerging security problems such as encryption (asymmetric, but also symmetric), digital signature, key exchange, homomorphic encryption etc. While such techniques provide guarantees in theory, their realization on contemporary computing platforms requires careful design choices and tradeoffs to manage both the diversity of computing platforms (e.g., high-performance to resource constrained), as well as agility for deployment in the face of emerging and changing standards.

In this work we survey trends in lattice-based cryptographic schemes, some fundamental recent proposals for the use of lattices in computer security, challenges for their implementation in software and hardware, and emerging needs for their adoption.

Additional Key Words and Phrases: Post-quantum cryptography; Public-key encryption; Lattice based cryptography; Ideal lattices; Ring-LWE

ACM Reference Format:
Hamid Nejatollahi, Nikil Dutt, Sandip Ray, Francesco Regazzoni, Indranil Banerjee, and Rosario Cammarota. 2017. Software and Hardware Implementation of Lattice-Cased Cryptography Schemes. *ACM Comput. Surv.* 1, 1, Article 0 (2017), 35 pages. https://doi.org/0000001.0000001

This work was supported in part with a gift from Qualcomm Research.
Authors' addresses: Hamid Nejatollahi, University of California Irvine, Irvine, California, 92697-3435, hnejatol@uci.edu; Nikil Dutt, University of California Irvine, Irvine, California, 92697-3435, dutt@ics.uci.edu; Sandip Ray, NXP, , sandip.ray@nxp.com; Francesco Regazzoni, ALaRi, regazzoni@alari.ch; Indranil Banerjee, Qualcomm Research, San Diego, CA, 92121-1714, ; Rosario Cammarota, Qualcomm Research, San Diego, CA, 92121-1714, ro.c@qti.qualcomm.com.

University of California Irvine, CECS TR 17-04

1 INTRODUCTION

Advances in computing efficiency steadily erode computer security at its foundation. These enable prospective attackers to use ever more powerful computing systems and the best cryptanalysis algorithms to increase the attack speed. One aspect of this arises from Moore's Law[1], that enables traditional computing systems to become more powerful than ever before, allowing increasingly larger brute-force attacks.

Another aspect of concern is the rise of alternative computing paradigms, such as Quantum computing and its algorithms [84, 123] - that are an imminent reality,[2] [3], which in turn promises to further weaken the strength of current, standardized cryptography and its applications. Indeed in the face of quantum computing, doubling of the key size for symmetric key cryptography can only be an interim solution that applies to non resource-constrained devices,[4]. For resource constrained devices, and specifically devices expected to operate in a span of ten years or more, doubling the key size (e.g., for AES) is not an acceptable choice. Against quantum computers traditional public key cryptography is ineffective for any key length algorithm. The Shor's algorithm for quantum computers is designed to solve prime factorization of large primes and the discrete logarithm problem in polynomial time - e.g., solving for an RSA private key given the domain parameter becomes as easy as executing RSA.

The emergence of new computing paradigms, such as cloud Computing, software defined networks and Internet of Everything, demands the adoption of an increasing number of security standards, which in turn requires the implementation of a diverse set of cryptographic primitives, but this is only part of the story. At the computing platform level, we are seeing a diversity of computing capability, ranging from high-performance (real-time) virtualized environments, such as cloud computing resources and software defined networks, to highly resource-constrained IoT platforms to realize the vision of Internet of Everything. This poses tremendous challenges in the design and implementation of emerging standards for cryptography in a single embodiment, since the computing platforms exact diverging goals and constraints. On one end of the spectrum, in the cloud computing and software defined network space, applications demand high-performance, and energy efficiency of cryptographic implementations. This calls for the development of programmable hardware capable of running not only individual cryptographic algorithms, but full protocols efficiently, with the resulting challenge of designing for agility, e.g., designing computing engines that achieve the efficiency of Application-Specific Integrated Circuits (ASICs), while retaining some level of programmability. On the other end of the spectrum, in the IoT space, implementations of standardized cryptography to handle increased key sizes become too expensive in terms cost, speed, and energy, but are necessary, e.g., in the case of long lived systems such as medical implants. In part, this demands the development of new and strong lightweight alternatives to symmetric key cryptography as well. Furthermore, given the variety of applications and their interplay with the cloud, even in this case agility in the implementation becomes a key requirement.

As a result of the trends in technology, the need to strengthen current practices in computer security, including strengthening and adding more variety in the cryptographic primitives in use, has become a widely accepted fact. The examples above, to name a few, form a compelling argument to call for innovation in the computer security space, including and beyond the foundations, i.e., down to the actual implementation and deployment of primitives and protocols to satisfy the emerging business models and their design constraints - latency, compactness, energy efficiency, tamper resistance and, more importantly, agility.

[1] http://www.telegraph.co.uk/technology/2017/01/05/ces-2017-moores-law-not-dead-says-intel-boss
[2] http://spectrum.ieee.org/tech-talk/computing/software/rigetti-launches-fullstack-quantum-computing-service-and-quantum-ic-fab
[3] https://newatlas.com/ibm-next-quantum-processors/49590/
[4] See NIST and NSA recommendations on Suite B Cryptography: https://www.nsa.gov/what-we-do/information-assurance/

Innovation in public key cryptography focuses on the standardization of the so called post-quantum cryptography primitives and their protocols. Among the post-quantum cryptography families, the family of lattice-based cryptography (LBC) appears to be gaining acceptance. Its applications are proliferating for both traditional security problems (e.g., key exchange and digital signature), as well as emerging security problems (e.g., homomorphic schemes, identity-based encryption and even symmetric encryption). Lattice-based cryptographic primitives and protocols provides a rich set of primitives which can be used to tackle the challenges posed by deployment across diverse computing platforms, e.g., Cloud vs. Internet-of-Things (IoT) ecosystem, as well as for diverse use cases, including the ability to perform computation on encrypted data, providing strong (much better understood than before) foundations for protocols based on asymmetric key cryptography against powerful attackers (using Quantum computers and algorithms), and to offer protection beyond the span of traditional cryptography. Indeed, lattice-based cryptography promises to enhance security for long-lived systems, e.g., critical infrastructures, as well as for safety-critical devices such as smart medical implants [57].

In this manuscript, we review the foundations of lattice-based cryptography, some of the more adopted instances of lattices in security, their implementations in software and hardware, and their applications to authentication, key exchange and digital signatures. The purpose of this survey is to focus on the essential ideas and mechanics of the crypto-systems based on lattices and the corresponding implementation aspects. We believe – to the best of our knowledge – that this survey is not only unique, but also important for guidance in the selection of standards, as well as in the analysis and evaluation of candidate implementations that represent state-of-the-art. In particular, we being by describing briefly different lattice-based schemes in Section 2.5. Section 3.1 discusses related art on improvements for arithmetic computation, i.e., polynomial/matrix multiplication or/and noise sampler. Sections 3.3 and 3.2 present hardware and software implementations of different lattice-based schemes, respectively. Sections 4 concludes with an outlook.

2 BACKGROUND

2.1 Standard lattices

A lattice is defined as a infinite set (discrete) of points in n-dimensional Euclidean space with a periodic structure [113]. Let b_1, b_2,..., $b_n \in \mathbb{R}^{m \times n}$ to be n-linearly independent vectors in $\mathbb{R}^m$ as the basis (B) of the lattice. B is a $m \times n$ matrix in which i^{th} column is b_i vector such that $B = [b_1, b_2, ..., b_n]$. Integer n and m are rank and dimension of the lattice $\mathcal{L}(B)$. If $n = m$, $\mathcal{L}(B)$ is a full rank (or dimension) lattice in $\mathbb{R}^m$. Lattice $\mathcal{L}$ is defined as the set of all integer combinations that is generated by the basis matrix B (with integer or rational entries) as below:

$$\mathcal{L}(B) = \{Bx : x \in \mathbb{Z}^n\} = \mathcal{L}(b_0, ..., b_n) = \left\{ \sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z}^n, 1 \leq i \leq n \right\} \quad (1)$$

where **Bx** is matrix-vector multiplication. It should be mentioned that B is not unique and multiple basis vectors exist for a specific lattice. Mathematically, basis matrices B and B' produce the same lattice, $\mathcal{L}(B) = \mathcal{L}(B')$, if and only if there is a an invertible matrix U such that $B = B'U$. It is easy to prove that any invertible matrix U is unimodular; which means the absolute value of its determinant is one ($|det(U)| = 1$, for $U \in \mathbb{Z}^{n \times n}$).

Let det($\mathcal{L}_B$) be the determinant of lattice $\mathcal{L}(B)$ which is equal to absolute value of the determinant of the basis matrix B such that $det(\mathcal{L}_B) = \{|det(B)| : B \in \mathbb{Z}^{n \times n}\}$. The determinant of a lattice geometrically corresponds to the inverse of the density of the lattice points; a lattice which consists of denser set of points has the higher determinant.

The bases B and B' generate the same lattice if and only if $|det(B)| = |det(B')|$. In other words, $\det(\mathcal{L}_B)$ is independent of the basis and multiplication of a basis B with a unimodular matrix U (with determinant of +1 or -1) generates new basis ($B' = BU$) that produces the same lattice.

The dual of lattice $\mathcal{L}$ in $\mathbb{R}^n$ is defined as $\mathcal{L}^*$, which is the set of all vectors $y \in \mathbb{R}^n$ whose inner products with the vectors $x \in \mathcal{L}$ results in $z \in \mathbb{Z}$. It is proven that for each $B \in \mathbb{R}^{n \times n}$, $\mathcal{L}(\mathcal{B})^* = \mathcal{L}((B^{-1})^T)$, where T transposes a matrix. In other words, $\mathcal{L}(B)$ is generated by columns of nonsingular matrix B, while $\mathcal{L}(\mathcal{B})^*$ is generated by rows of matrix B^{-1}.

If A is generated by the columns of the nonsingular matrix B, then A' is the lattice generated by the rows of B -

$$\mathcal{L}^* = \{y : \langle x, y \rangle \in \mathbb{Z}, x \in \mathcal{L}\} \tag{2}$$

A `q-ary` lattice is a lattice $\mathcal{L}$ that satisfies $q\mathbb{Z}^n \subseteq \mathcal{L} \subseteq \mathbb{Z}^n$ which means membership of a vector x in lattice $\mathcal{L}$ is determined by x mod q with integer q much smaller than $\det(\mathcal{L})$. Most Lattice-based cryptographic schemes employ `q-ary` lattices as the hard on average problem.

Let n (rank), m (dimension) and q (preferably a prime integer) be positive integers and A is a matrix in $\mathbb{Z}_q^{n \times m}$. Prime p is a small integer with $O(\log n)$ bits and m is a small multiple of n such that $m = O(n)$. Two forms of `q-ary` lattices are common in practical lattice-based cryptographic systems [94].

$\Lambda_q(A)$ is lattice that is generated by the rows of matrix A (mod q) and is closely related to Learning With Error (`LWE`) problem.

$$\Lambda_q(A) = \{y \in \mathbb{Z}^m : y = A^T s (\mathrm{mod} q), \mathbf{s} \in \mathbb{Z}^n\} \tag{3}$$

$\Lambda_q^\perp(A)$ (orthogonal Λ_q form) is the lattice whose vectors are orthogonal modulo q to the rows of matrix A which corresponds to the code whose parity check matrix is A mod q. In other words, for vectors x,$y \in \Lambda_q^\perp$, $x + y$ is a member of lattice $\Lambda_q^\perp$. It should be mentioned that $\Lambda_q^\perp(A)$ is used to solve the Short Integer Solution (`SIS`) problem.

$$\Lambda_q^\perp(A) = \{y \in \mathbb{Z}^m : Ay = 0 (\mathrm{mod} q), \mathbf{s} \in \mathbb{Z}^n\} \tag{4}$$

For short integer solution (`SIS`) lattice problems $\Lambda_q^\perp$ and for learning with error (`LWE`) lattice problems Λ_q is used.

From definition, lattice Λ_q is dual of lattice $\Lambda_q^\perp$ such that $\Lambda_q^\perp(A) = q.\Lambda_q(A)^*$ and $\Lambda_q(A) = q.\Lambda_q^\perp(A)^*$.

2.2 Ideal lattices

An ideal lattice is defined over a ring $R = \mathbb{Z}_q[x]/(f(x))$, where $f(x) = x^n + f_n x^{n-1} + \ldots + f_1 \in Z[x]$ (cyclotomic polynomial) and R contains all the polynomials with modulo q integer coefficients. In the case where $f(x)$ is monic (leading coefficient is 1) irreducible polynomial degree-n, $R = \mathbb{Z}_q[x]/(f(x))$ includes polynomials of degree less that equal $n - 1$. For instance, $R = \mathbb{Z}_q[x]/(x^n + 1)$ is an ideal lattice when n is a power of 2; however, $R = \mathbb{Z}_q[x]/(x^n - 1)$ is not an ideal lattice since $(x^n - 1)$ is a reducible polynomial.

2.3 Lattice problems and their applications to cryptography

The breakthrough work of Ajtai [4] provides confidence for adopting lattice based schemes in cryptography. Specifically, Ajtai proves that solving some `NP-hard` lattice problems, for instance Shortest Vector Problem (`SVP`), in average-case is as hard as solving them in the worst case assumption. In other words, in order to solve `SVP` problem, an algorithm should solve the problem for any input basis B (all instances of `SVP` problem should be solved).

It is conjectured there is no polynomial time algorithm that can approximate lattice problems to within polynomial factors [94]. The mentioned conjecture is the basis for security of lattice-based cryptography schemes. The fastest algorithm to solve the SVP problem has the time and memory complexity of $2^{O(n)}$ [5]. In a n-dimensional lattice $\mathcal{L}$, **successive minima** is defined as sequence of $\lambda = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ where the i^{th} success minima, λ_i, is the radius of smallest ball that contains i linearly independent lattice vectors. We take the below definitions from [91]:

2.3.1 Shortest Vector Problem (SVP). SVP is the most basic problem over lattices on which lattice-based cryptographic schemes are built. Three variants of the SVP exist in the literature, all of which can be reduced to each other. The first is to find the shortest nonzero vector; the second is to find length of the shortest nonzero vector; and the third determines if the shortest nonzero vector is shorter than a given real number.

Given a lattice basis B, the shortest nonzero vector in lattice $\mathcal{L}(B)$ is defined as $v \in \mathcal{L}(B)\backslash\{0\}\}$ such that $\|v\| = \lambda_1(\mathcal{L}(B))$. Output of the SVP problem is **a** shortest nonzero vector in the lattice which is unique. It should be mentioned that SVP can be defined to an arbitrary norm (we use norm 2 here as the Euclidean norm). In γ-Approximate Shortest Vector Problem (SVP_γ), for $\gamma \geq 1$, the goal is to find the shortest nonzero vector $v \in \mathcal{L}(B)\backslash\{0\}$ where $\|v\| \leq \lambda_1(\mathcal{L}(B))$. The special case of $\gamma = 1$ is equivalent to the exact SVP. The decision version of the Shortest Vector Problem ($G_{AP}SVP_\gamma$) is defined as determining if $d < \lambda_1(\mathcal{L}(B)) \leq \gamma.d$ where d is a positive real number.

To date, there is no known polynomial time solution to solve SVP and its approximate version. Ajtai proved that SVP is a NP-hard problem and solving SVP in average case is as hard as solving SVP in the worst case.

2.3.2 Closest Vector Problem (CVP). Given a lattice basis B and target point t (might not be a member of the lattice), CVP is defined as finding vector $v \in \mathcal{L}$ where its distance ($\|v - t\|$) to a target point is minimized. In γ-Approximate Closest Vector Problem (CVP_γ), for $\gamma \geq 1$, the goal is to find vector $v \in \mathcal{L}(B)$ such that $\|v - t\| \leq \gamma.dist(t, \mathcal{L}(B))$ where $dist(t, \mathcal{L}(\mathcal{B})) = inf\{\|v - t\| : v \in \mathcal{L}\}$ is the distance of target point t to the lattice $\mathcal{L}$.

2.3.3 Shortest Independent Vectors Problem (SIVP). Given lattice basis B and prime integer q, the shortest independent vector problem (SIVP) is defined as finding n linearly independent lattice vectors $\{v = v_1, ..., v_n : v_i \in \mathcal{L}(B) for 1 \leq i \leq n\}$ that minimizes $\|v\|$=max$_i\|v_i\|$. Given an approximate factor $\gamma \geq 1$, the $\gamma - approximate$ Shortest Independent Vector Problem ($SIVP_\gamma$) is defined as finding n-linearly independent vectors lattice vectors $\{v = v_1, ..., v_n : v_i \in \mathcal{L}(B)$ such that max$_i\|v_i\| \leq \lambda_n(\mathcal{L}(B)$ where λ_n denotes the n^{th} success minima. For a n-dimensional lattice, λ_i, i^{th} success minima, is the radius of the smallest ball that contains i linearly independent lattice vectors. The decision version of SIVP is called ($G_{AP}SIVP_\gamma$) and is defined as determined if $d < \lambda_n(\mathcal{L}(B)) \leq \gamma.d$ where d is a positive real number.

In problems described above, the approximate factor is a function of the lattice dimension. By increasing the lattice dimension, solving computationally hard lattice problems become harder.

There is a close relationship between lattice hard problems (like CVP and SVP) and two common average-case lattice-based problems, Learning With Error (LWE) and Shortest Integer Solution (SIS). In the following we introduce SIS and LWE problems with their complexity.

2.3.4 Shortest Integer Solution (SIS). Let a_1, a_2,..., $a_n \in \mathbb{Z}^{m\times n}$ be an arbitrary vector and q is a integer prime number. The SIS problem is defined as finding the vector $x \in \mathbb{Z}^{m\times n}$ such that

$$x_1.a_1 + x_2.a_2 + ... + x_n.a_n = 0 \pmod q \tag{5}$$

Short is usually translates as $z_i \in \{-1, 0, +1\}$. Considering q-ary lattices, let A=(a_1, a_2,..., a_n) be a vector in $\mathbb{Z}^{m\times n}$ and $\Lambda_q^\perp(A) = \{z \in \mathbb{Z}^m : Az = 0 \pmod q\}$; SIS problem is to find the shortest vector problem for the lattice $\Lambda_q^\perp$. Ajtai proved

that if a lattice problem like SVP is hard to solve in the worst case, an average case one-way function exists [4]. Based on the Ajtai's theorem, if algorithm A can solve the SIS problem in polynomial time, a polynomial time algorithm ,B, exists that can solve SVP (or SVIP) problem for any lattice of dimension n.

Ring-SIS. Let A=(a_1, a_2,..., a_n) be a vector with $a_i \in \mathbb{Z}_q[x]/(x^n+1)$ and $\Lambda_q^{\perp}(A) = \{z \in \mathbb{Z}_q[x]/(x^n+1) : Az = 0 \pmod q\}$; Ring-SIS problem is to find the shortest vector problem for the lattice $\Lambda_q^{\perp}$.

2.3.5 Learning With Error (LWE). Let a be the polynomial with coefficients sampled uniformly at random in $\mathbb{Z}_q^n$, where n and q are degrees of lattice and modulus (prime integer), respectively. Recovering the (**unique**) random secret s (uniformly at random in $\mathbb{Z}_q^n$) from m ($m \geq n$) samples of the form $(a, a.s + e \bmod q)$ is known as the Learning With Error (LWE) problem where e is the error that is sampled from error distribution χ. Based on [13], if secret s is sampled from the same error distribution as e, the hardness assumption of the learning with error problem is still valid.

Similar to the SIS problem, solving an average case LWE problem is as hard as solving worst-case lattice problems like SVP.

Ring-LWE. let $R_q = \mathbb{Z}_q[x]/(x^n+1)$ to be the ring of polynomials where n is power of 2 and q is an integer. Recovering random secret s with uniform coefficients in R from $m \geq 1$ samples of the form $(a, a.s + e \bmod q)$ is known as ring learning with error problem (Ring-LWE) where $e \in R$ is the error with coefficients sampled from error distribution χ.

Module-LWE. Let n and d be dimensions of R (degree of ring R_q) and rank of module $M \in R^d$. The hardness of a scheme with $R_q = \mathbb{Z}_q[x]/(x^n+1)$ and $d = 1$ is based on Ring-LWE and Ring-SIS problems; however $R_q = \mathbb{Z}_q$ and $d > 1$ end up with LWE and SIS problems. Module-LWE/SIS is a generalization of LWE/SIS and Ring-LWE/SIS in which the parameters are $R = \mathbb{Z}_q[x]/(x^n+1)$ and $d \geq 1$. Security reduction of lattice problems in module M depends on $N = n \times d$ (dimension of the corresponding module lattice) [81]. Suppose A is a $d \times d$ random matrix; security reduction of LWE/SIS problem for $d = 1$ (Ring-SIS/LWE) and ring of dimension n is the same as $d = i$ and a ring of dimension n/i. In the former case, matrix A contains n elements in Z_q; however, in the latter case, A contains $i^2 \times n$ elements in Z_q.

Let $R_q = \mathbb{Z}_q[x]/(x^n+1)$ and $R = \mathbb{Z}[x]/(x^n+1)$ be rings of polynomials where n is power of 2 and $q \in Z$. Vector a is uniformly sampled in R_q^d. Recovering the random secret s with coefficients sampled from error distribution χ^d in R from $m \geq 1$ samples of the form $(a_i, a_i.s + e_i \bmod q) \leftarrow R_q^d \times R_q$ is known as module learning with error (Module-LWE) problem where $e_i \in R$ is the error with coefficients sampled from error distribution χ [81].

2.4 Arithmetic and Components of Lattices

In this section we provide an analysis of the components in a lattice-based cryptosystem that guide the actual implementation. Discrete Gaussian sampling is used to sample noise in order to hide the secret information which is expensive especially on resource constrained devices. Besides, matrix multiplication for standard lattice, and polynomial multiplication for ideal lattices are the main speedup bottlenecks. There are various algorithms for the sampler and multiplier in the literature, with providing the designer a specific goal [98]. We briefly review different algorithms and outline their practical implementations in Section 3.1

There exist two main classes of lattice-based algorithms used in cryptography, namely NTRU and Learning with Error (LWE). The security of NTRU is based on hardness not-provably reducible to solving the Closest Vector Problem (CVP) in a lattice, whereas the security of LWE relies on provably reducible solving the Shortest-Vector Problem (SVP) in a lattice. Consequently, NTRU suffers from security guarantees, but in practice provides more flexibility and efficiency in

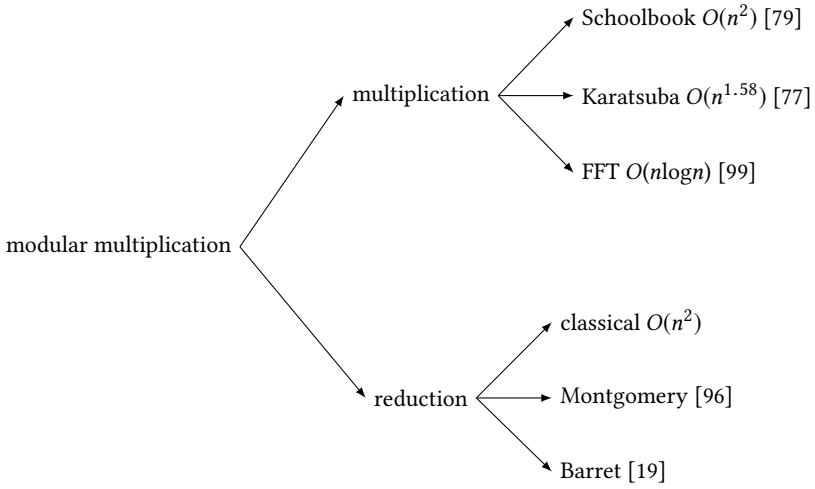

Fig. 1. Modular Multiplication

the implementation. On the contrary, LWE problems are resistant to quantum attacks, while their relatively inefficient nature led researchers to devise more efficient formulations, e.g., over rings - Ring Learning with Errors (Ring-LWE).

Researchers have striven to achieve efficient and secure implementations of lattice-based schemes, from resource constrained embedded systems, to high-performance servers. Implementations are broadly classified in pure software, pure hardware, and hardware/software co-design cryptographic engines [98].

The implementation of the modulo arithmetic (multiplication and addition of big numbers) is important, as the arithmetic clearly is one of the most time consuming parts in lattice-based schemes. For Standard LWE schemes, matrix multiplication algorithms are adopted, whereas number theoretic transform (NTT) is a better choice for polynomial multiplication in Ring-LWE. A summary of modular arithmetic is presented in Figure 1.

In addition to the arithmetic portion, a bottleneck in lattice-based schemes is the extraction of the random term, which usually is implemented with a discrete noise sampler (from a discrete Gaussian distribution) and can be done with rejection, inversion, Ziggurat, or Knuth-Yao sampling with moderate standard deviation for key exchange and public key encryption, and small standard deviation for digital signature to achieve a compact and secure signature.

When implemented, Standard LWE-based schemes exhibit a relatively large memory footprint due to large key size - hundreds of kilobyte for public key - which render a straightforward implementation of standard LWE-based schemes impractical on resource constrained devices. The adoption of specific ring structures, e.g., Ring-LWE, offers key size reduction by a factor of n compared to Standard LWE [89], making Ring-LWE an excellent candidate for resource constrained devices, such as Wi-Fi capable smart devices, including medical implants. Another avenue to address resource constrained devices is that memory footprint can be traded off with security assurance, which improves both efficiency and memory consumption.

High-performance Intel/AMD processors, which are notoriously equipped with AVX vector instructions, and ARM/AVR micro-controllers are popular platforms for software implementations, as we will see in more detail in the coming sections. Practical software implementations of standard lattices, encryption scheme [71] and key exchange [26], have been published recently.

For hardware implementations, FPGAs provide flexibility and customization, but not programmability. Hardware implementations of lattice-based schemes are about an order of magnitude faster than RSA implementation. Besides, lattice-based implementation provides a higher security level, and consumes fewer device resources [72]. Furthermore, another candidate platform to implement lattice-based schemes are application specific integrated circuits (ASICs) of which there appear to be no such implementations in the literature at this time. However, the main advantages and challenges for ASIC design of lattice-based schemes are presented in [100].

2.4.1 The multiplier component. Standard lattice arithmetic operations involve calculations over matrices; however, ideal lattices benefit from their specific structure and operate on polynomials. Matrix multiplication is used for the standard lattices, while polynomial multiplication is employed for ideal lattices.

Arithmetic operations for a `Ring-LWE` based scheme are performed over a ring of polynomials. The most time and memory consuming part is the polynomial multiplication. The easiest way to multiply two polynomials is to use the Schoolbook algorithm with the time complexity of $O(n^2)$ [79]. Let n and p be degree of the lattice (n is a power of 2) and a prime number ($p = 1 \bmod 2n$), respectively. Z_p denotes the ring of integers module p and x^{n+1} is an irreducible polynomial of degree n. The quotient ring R_p, contains all polynomials with the degree less that n in Z_p, that defines as $R_p = Z_p/[x^{n+1}]$ in which coefficients of polynomials are in the range [0,p).

Let $a(x)$ and $b(x)$ be two polynomials in the ring Z_p. Schoolbook algorithms solve multiplication of $a(x)$ and $b(x)$ as below:

$$a(x).b(x) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} a_i b_j x^{i+j} mod(x^n + 1) \tag{6}$$

The number theoretic transform (`NTT`) is a generalization of fast Fourier transform (`FFT`), which is carried out in a finite field instead of complex numbers. The latter could achieve time complexity of $O(n\log n)$. In other words, $\exp(-2\pi j/N)$ with n^{th} primitive root of unity ω_n which is defined as the smallest element in the ring that $\omega_n^n = 1 \bmod p$ and ${\omega_n}^i \neq 1 \bmod p$ for i ≠n. The main idea behind this is to convert polynomials from coefficient representation to point value representation by applying NTT in $O(n\log n)$; thereafter performing point-wise multiplication in $O(n)$ and finally converting the result to coefficient representation by applying Inverse NTT (`INTT`) in $O(n\log n)$.

$$a(x).b(x) = NTT^{-1}(NTT(a) \odot NTT(b)) \tag{7}$$

Where $\odot$ is point-wise multiplication of the coefficients. If NTT is applied to $a(x)$ with $(a_0, ..., a_{n-1})$ as coefficients, we would have:

$$(\hat{a_0}, ..., \hat{a_{n-1}}) = NTT(a_0, ..., a_{n-1}) \tag{8}$$

$$\hat{a_i} = \sum_{j=0}^{n-1} a_i \omega^{ij} mod(p), i = 0, 1, ..., n-1 \tag{9}$$

In order to retrieve the answer from point value representation using NTT^{-1}, it is sufficient to apply NTT function with a slight change as indicated below:

$$a_i = \sum_{j=0}^{n-1} \hat{a}_i \omega^{-ij} mod(p), i = 0, 1, ..., n-1 \tag{10}$$

As can be seen, if instead of ω we use $-\omega$ and divide all the coefficients by n, NTT^{-1} is computed.

In order to computer NTT(a), we pad the vector of coefficients with n zeros which culminates in doubling the input size. By employing negative wrapped convolution technique [129], there is no need to double the input size.

To improve efficiency of polynomial multiplications with NTT, combining multiplications of powers of ω with powers of ψ and ψ^{-1} ($\psi^2 = \omega$) can be beneficial which requires storage memory for precomputed powers of ω and ψ^{-1} in bit-reversed order. The combining idea is used in [85, 110, 116]. It should be mentioned that NTT is applicable only for the $p = 1 \pmod{2n}$ case where p is a prime number. Suppose $a' = (a_0, \psi a_1, ..., \psi^{n-1} a_{n-1})$, $b' = (b_0, \psi b_1, ..., \psi^{n-1} b_{n-1})$, $c' = (c_0, \psi c_1, ..., \psi^{n-1} c_{n-1})$ to be coefficient vectors of the a, b, c that are multiplied component-wise by $(1, \psi^1, ..., \psi^{n-1})$. Based on the negative wrapped convolution theorem, modulo $(x^n + 1)$ is eliminated and the degree of NTT and NTT^{-1} is reduced from $2n$ to n.

$$c' = NTT^{-1}(NTT(a') \odot NTT(b')) \tag{11}$$

where $c(x)$ is the polynomial multiplication of $a(x)$ and $b(x)$.

$$c = (\psi^0 {c'}_0, \psi^{-1} {c'}_1, ..., \psi^{-n+1} {c'}_{n-1}) \tag{12}$$

Other popular multiplication algorithms in literature are the Karatsuba algorithm (with time complexity of $O(n^{log3/log2})$ [77]), and subsequent variants of it [40]. Schönhage-Strassen with time complexity of $O(n.\log n.\log\log n)$ [122] outperforms the Karatsuba algorithm [34].

An extensive analysis and comparison for hardware complexity of various modular multiplications, including Schoolbook (classical), Karatsuba, and FFT, with different operand size are presented in [40]. Authors calculate hardware complexity of each multiplier by decomposing it into smaller units such as full adder, half adder, multiplexer, and gate. Rafferty et al. [112] adopt the same approach to analyze large integer multiplications of both combined and individual multipliers. Karatsuba multiplier outperforms for operands greater of equal to 32 bits. Schoolbook imposes large memory footprint in order to store partial products which negatively impact performance that is mitigated by Comba [38] with the same time complexity but relaxing memory addressing by optimizing the sequence of partial products. Rafferty et al. compare hardware complexity, in terms of +, −, and ∗ units, of different combinations of classical (Comba), Karatsuba, and FFT (NTT for integers) for up to multipliers with 65536-bit operands. However, they evaluate the latency and clock frequency by implementing in hardware (Xilinx Virtex-7 FPGA) for up to 256-bits for combination of NTT+Comba and 4096-bit for Karatsuba+Comba which are not in a good range for lattice-based cryptoghaphy (e.g. $1024 \times 14 = 14336$ bits are used for Newhope key exchange). Based on their (analytical) hardware complexity analysis, combination of Karatsuba-Schoolbook is the best choice for operands under 64 bits. Karatsuba-Comba is preferable for operands for 64 bits to 256 bits. For larger operands, the lowest hardware complexity is achieved by combined multiplier NTT-Karatsuba-Schoolbook. It should be mentioned that results are for a single multiplication. Authors make some assumption for the sake of simplicity such as the shift operation is free and inputs are ready.

2.4.2 The Sampler Component. The quality of a discrete Gaussian sampler is determined by a tuple of three parameters: (σ,λ,τ). In such a tuple σ is the standard deviation (adjusts dispersal of data from the mean), λ is the precision parameter (controls statistical difference between a perfect and implemented discrete Gaussian sampler), and τ is the distribution tail-cut (determines amount of the distribution that we would like to ignore).

Each of these parameters affects the security and efficiency of the sampler. For instance, smaller standard deviation decreases the memory footprint required to store precomputed tables. For encryption/decryption schemes σ=3.33 [82]

is suggested. Digital signature sampling from a Gaussian sampler involves large σ=215 [49]. However, by employing Peikert's convolution lemma [102], standard deviation can be reduced by an order of magnitude which is a remarkable improvement on the precomputed table size. Speed and memory footprint is λ dependent, i.e. higher λ results in more secure but a slower and bigger sampler. The tail of the Gaussian distribution touches the x-axis at x=+∞ (considering only the positive side due to the symmetry) with negligible probability. Tail-cut parameter (τ) determines the amount of the distribution that we would like to ignore, hence random number e is sampled in $|e| \in \{0, \sigma \times \tau\}$ instead $|e| \in \{0, \infty\}$.

Sampling the discrete Gaussian distribution is one the most time and memory hungry parts of lattice-based cryptosystems, due to the demands of high precision, numerous random bits, and huge lookup tables. To be more specific, obligatory negligible statistical distance between the implementation of a Gaussian sampler and its theoretical discrete Gaussian distribution imposes expensive precise floating point arithmetic (to calculate the exponential function) or large memory footprint (to store precomputed probabilities). To keep statistical distance less than $2^{-\lambda}$, floating point precision with more than standard double-precision is obligatory which is not natively supported by the underlying platform; thus software libraries should be used to perform higher floating-point arithmetic. It is impractical to sample from a perfect Gaussian sampler; hence an λ-bit uniform random integer is used to approximate the discrete sampler. Fortunately, it is proven that the Gaussian sampler could be used to achieve $\lambda/2$ security level (approximated sampler) instead of λ level (perfect sampler), since (to date) there is no algorithm that can distinguish between a perfect sampler (λ bits) and an approximate sampler ($\lambda/2$ bits) [118]. In other words, we can cut half of the bits in the sampler which results in a smaller and faster sampler [18, 71, 119]. Reduction in precision parameter (from λ to $\lambda/2$) changes tail-cut parameter (τ) as $\tau = \sqrt{\lambda \times 2.ln(2)}$ [70]. Sampling from a Gaussian distribution may lead to a timing side channel attack, which can be avoided by using a generic constant time Gaussian Sampling over integers [95]. Folláth provides a survey on different Gaussian samplers in lattice based cryptography schemes with a more mathematical outlook [56]. Gaussian samples are classified into six categories and guidelines provided for choosing the best candidate on different platforms for specific parameter ranges. However, we organize Gaussian samplers into the following types discussed below. A Summary of advantage and disadvantages of each sampler are listed in Table 1.

Rejection Sampler. Numerous truly random bits are required for sampling which is extremely time consuming. Firstly, variable x is chosen in (-$\tau\sigma$,$\tau\sigma$) uniformly at random where τ and σ are tail-cut and standard deviation of Gaussian distribution. Thereafter, x is accepted with the probability proportional to exp(-$x^2/2\sigma^2$). High rejection rate of samples (on average 8 trials to reach acceptance) along with expensive calculation of `exp()` are the main reasons for the inefficiency [128].

Rejection sampler was employed for the first time within lattice-based cryptosystems in [59]. Remarkable speed and area improvement could be achieved by accomplishing rejection operation using lazy floating-point arithmetic [51].

Bernoulli Sampler. Bernoulli is an optimized version of rejection sampling in order to reduce average required attempts for a successful sampler from 10 to around 1.47 with no need to calculate `exp()` function or precomputed tables [105]. The main idea is to sample from an intermediate distribution, binary Gaussian distribution; thereafter, second sampling is done from the final distribution. Bernoulli is introduced in [49] for lattice-based cryptography realm and used widely in the research community [71, 106, 107]. For standard deviation $\tau = 16$, on average $\frac{2\tau}{\sqrt{2\pi}} = 10$ trials are needed for rejection and CDT sampling until reaching a sample that has negligible statistical difference to the perfect distribution. Bernoulli distribution has much less statistical difference, resulting in 1.47 average number of rejections that in turn leads to a reduction in sampling time which is achieved by sampling twice, first from an easy to sample intermediate distribution (binary Gaussian sampler $D_{\mathbb{Z}^+,\sigma_2}$) and second from the target distribution [49].

The main idea behind Bernoulli sampling is to approximate sampling from $D_{\mathbb{Z},k\sigma_2}$ using the distribution $k \cdot D_{\mathbb{Z}^+,\sigma_2} + \mathbb{U}(\{0,...,k-1\})$ where $\mathbb{U}$ is uniform distribution. The procedure for Bernoulli sampler is shown below in 5 steps.

(1) sample $x \in \mathbb{Z}$ according to $D_{\mathbb{Z}^+,\sigma_2}$ with probability density of $\rho_{\sigma_2} = e^{-x^2/(2\sigma_2^2)}$
(2) sample $y \in \mathbb{Z}$ uniformly in to $\{0,...,k-1\}$ and calculate $z \leftarrow kx + y$, $j \leftarrow y(y+2kx)$
(3) sample $b \leftarrow \mathcal{B}_{-j/2\sigma^2}$ where $\sigma = k\sigma_2$ and $\mathcal{B}$ is Bernoulli distribution. To sample from $\mathcal{B}_c$ where c is a precomputed constant value, a uniform number $u \in [0,1)$ with λ-bit precision is sampled and one is returned if $u < c$, otherwise zero should be returned.
(4) if $b = 0$ goto to step (1)
(5) if $z = 0$ go to step (1), otherwise generate $b \leftarrow \mathcal{B}_{1/2}$ and return $(-1)^b z$ as the output

The standard deviation of target Gaussian sampler $D_{\mathbb{Z},k\sigma_2}$, equals $k\sigma_2$ where $\sigma_2 = \sqrt{\frac{1}{2ln2}} \approx 0.849$ is standard deviation of the binary Gaussian sampler $D_{\mathbb{Z},\sigma_2}$ and $k \in \mathbb{Z}^+$ is the uniform distribution parameter. For instance, to sample a noise from Gaussian distribution for `BLISS-I` digital signature with $\sigma = 215$ and `LP` encryption with $\sigma = 3.33$, k equals 4 [70]. For schemes with small standard deviation (e.g. public key encryption) sampling from binary Gaussian distribution can be eliminated [107], while for digital signatures with large standard deviation, using Gaussian distribution is mandatory [106].

The Bernoulli approach eliminates long integer calculation; instead, single bit operations are the most frequent operation, hence it is a good candidate for hardware implementation. However, the time dependency of Bernoulli makes it vulnerable to timing attacks which is resolved in hardware implementation of BLISS in [106]. Precomputed tables in Bernoulli sampling are small, besides binary Gaussian distribution (easy to sample intermediate sampler) is independent of σ hence the Peikert convolution lemma (smaller σ) does not have a considerable effect on the area. However, convolution lemma reduces area of precomputed tables in CDT sampler by a factor of 23× for `BLISS` (reduces σ from 215 to 19.47).

Binomial Sampler. Centered Binomial distribution (ψ_k) is a close approximation of rounded Gaussian sampler (ξ_σ) which eliminate the need both computing $exp()$ and precomputed large tables. Let $\sigma = \sqrt{8}$ be standard deviation of ξ_σ and a binomial distribution is parameterized with $k = 2\sigma^2$; choosing ψ_k as the sampling distribution has negligible statistical difference with rounded Gaussian sampler with $\sigma = \sqrt{8}$ [10]. Centered Binomial distribution(ψ_k) for integer $k \geq 0$ is defined as sampling $2 \cdot k$ random numbers from $\{0,1\}$ as $(a_1,...,a_k,b1,...,b_k)$ and output $\sum_{i=1}^{k}(a_i,b_i)$ as the random sample [10]. Since k scales with power 2 of σ, it is practical to use binomial sampling for digital signatures with large standard deviation.

Binomial sampler has been employed inside software implementation of NewHope [11, 12, 125], hardware implementation of NewHope [127] and software implementation of Kyber (CCA-secure encryption, key exchange and authenticated key exchange) [27].

Ziggurat Sampler. Ziggurat sampler is a variation of rejection sampler introduced in [90] for a continuous Gaussian sampler. Discrete version of Ziggurat sampler is proposed in [30] which is suitable for schemes with large standard deviation. The area under the PDF curve is divided into n rectangles with the same area whose size is proportional to the probability of sampling a point in each rectangle. The left and right corner of each rectangle is on the y-axis and Gaussian distribution curve, respectively. Each rectangle is stored using its lower right coordinates. Firstly, rectangle R_i and point x_i inside the rectangle is chosen uniformly at random. Since we are considering positive x_i, a random sign bit, s, is also required. If $x_i \leq x_{i-1}$, x_i resides below the curve and is accepted. Otherwise, a uniformly random y_i is

sampled and if $y_i \leq exp(x_i)$, the random point (x_i, y_i) is accepted; otherwise new random x should be sampled and the process is repeated.

More rectangles reduce the number of rejections and hence results in higher performance and precision. Due to its flexibility offers a trade-off between memory consumption and performance, Ziggurat sampler is a suitable candidate for resource constrained embedded devices. More precision and performance require more precomputed rectangles which impose remarkable memory overhead; however the increase in number of precomputed rectangles could drop the performance if the cache is fully occupied. Consequently, software implementation is preferred to any hardware implementation. There is only one hardware implementation of Zigguart in the literature which hints at the impracticality of realizing the Zigguart sampler in hardware [70].

Cumulative Distribution Table (CDT) Sampling. CDT is also known as the inversion sampling method which introduced in [102]. CDT is faster than rejection and Ziggurat samplers due to elimination of expensive floating-point arithmetic [126]. Since in cumulative distribution all the numbers are less than 1, it is sufficient to use binary expansion of the fraction. CDT requires a large precomputed table of discrete Gaussian cumulative distribution function (CDF)(highest memory footprint [56]) for which their size is a function of distribution tail-cut (τ) and Gaussian parameter (σ). Variable r is sampled uniformly at random in the range [0,1) with λ bits of precision. The goal is to find an x whose probability is $\rho(x) = S[x+1] - S[x]$ where $S[x]$ equals the value of CDF at x. CDF sampler performs a search, usually binary search, on the CDF, which is precomputed and stored in the memory, to find the corresponded x. A standard CDT needs table of size at least $\sigma\tau\lambda$ bits; for instance `BLISS-IV` ($\sigma = 13.4, \tau = 215, \lambda = 128$) and `BLISS-I` ($\sigma = 13.4, \tau = 19.54, \lambda = 128$) need at least pre-computed tables of size 630 kbits and 370 kbits for 192 and 128 bit post quantum security, which is not practical for resource constrained embedded devices [49]. By employing Peikert's convolution lemma [102], the size of precomputed tables are dropped by the factor of 11 by sampling twice from $\tau' = \frac{\tau}{\sqrt{1+k^2}} = 19.47$ instead of sampling from a $D_{Z,\tau}$ with $\tau = 215$. Further improvements on table size are presented in [105] by employing adaptive mantissa size which halves the table size.

A hardware implementation of CDT sampler is proposed in [109]. Further improvements of hardware implementations are accomplished in [43, 106, 108]. An efficient software implementation of CDT is proposed in [42]; however it is vulnerable to timing attack which is resolved in [78] by suggesting a time independent CDT sampler. Authors of [107] combine Cumulative Distribution Function (CDF) and rejection sampling to achieve a compact and reasonably fast Gaussian sampler.

Knuth-Yao sampler. The Knuth-Yao sampler provides a near optimal sampler because the number of random bits required by the sampling algorithm is close to the entropy, the distribution is suitable for high precision sampling [80]. Assume n to be number of possible values for each random variable r with the probability of p_r. The probability matrix is constructed based on the binary expansion of each variable whose r_{th} row denotes the binary expansion of p_r. According to probability matrix, a discrete distribution generating binary tree (DDG) is built whose i_{th} level corresponds to the i_{th} column of the probability matrix. Sampling is the procedure of walking through the DDG tree until a leaf is reached and its value returned as the sampling value. At each level, a uniformly random bit indicates whether the left child or right child of the current node should be visited in the future.

The Knuth-Yao sampler is suitable for schemes with small standard deviations; thus Knuth-Yao is not suitable for digital signature because of its slow sampling caused by high number of random bits. Knuth-Yao also requires large memory to store probability of sample points with high precision, necessary to keep the statistical distance between the true Gaussian distribution and its approximation small, which is a an issue on resource constrained platforms.

Combination of Knuth-Yao and CDT results in about halving the table sizes, which is still prohibitively large; however, the Bernoulli sampler offers the best precomputed table size [49].

De Clercq et al. introduce an efficient software implementation of Ring-LWE based cryptosystem by using Knuth-Yao as a Gaussian sampler [41]. First hardware implementation of Knuth-Yao sampling with small standard deviation ()using a column-wise method for sampling) resulted in faster sampling as introduced in [117]. Same authors improve their implementation in [115].

Based on the presented results on [30], CDT method outperforms discrete Ziggurat and rejection sampling with the same memory budget. For larger values of standard deviation the Ziggurat algorithm beats both CDT and rejection sampling. Compared to Knuth-Yao, Ziggurat achieves almost the same speed but reduces the memory consumption by a factor of more than 400. As stated earlier, due to the large standard deviation necessary for digital signature, Knuth-Yao sampler is not suitable for digital signatures. Inside an encryption scheme with $\sigma_{LP} = 3.3$ [82], with the same security level ($\lambda = 64$), Knuth-Yao sampler beats CDT in terms of number of operations performed in one second per slice of FPGA (Op/s/S) for time independent implementation [78]. However, for time dependent Gaussian sampler with the same security level ($\lambda = 94$), the CDT sampler proposed by Du and Bai [43] outperforms Knuth-Yao implementation of [115] in term of Op/s/S.

The Size of precomputed tables in a Bernoulli sampler is two orders of magnitude smaller than that of CDT and Knuth-Yao sampler [49], however CDT has three times more throughput than Bernoulli for hardware implantation of BLISS in [106].

Table 1. Comparison of different Gaussian samplers; partially extracted from [49]

Sampler	Speed	FP exp()	Table Size	Table Lookup	Entropy	Features
Rejection	slow	10	0	0	$45+10log_2\sigma$	suitable for constrained devices
Ziggurat	flexible	flexible	flexible	flexible	flexible	suitable for encryption requires high precision FP arithmetic not suitable for HW implementation
CDT	fast	0	$\sigma\tau\lambda$	$log_2(\tau\sigma)$	$2.1+log_2\sigma$	suitable for digital signature easy to implement
Knuth-Yao	fastest	0	$1/2\sigma\tau\lambda$	$log_2(\sqrt{2\pi e}\sigma)$	$2.1+log_2\sigma$	not suitable for digital signature
Bernoulli	fast	0	$\lambda log_2(2.4\tau\sigma^2)$	$\approx log_2\sigma$	$\approx 6+3log_2\sigma$	suitable for all schemes
Binomial	fast	0	0	0	$4\sigma^2$	not suitable for digital signature

2.5 Lattice-based schemes

Security of the lattice-based cryptography schemes are based on hardness of solving two average-case problems, a.k.a short integer solution (SIS) [93] and the learning with errors (LWE) problem [113].

The Learning With Errors problem proposed by Regev can be reduced to a worst-case lattice problem like the Shortest Independent Vectors Problem (SIVP) [113]. In the proposed scheme, the ciphertext is $O(n\log n)$ times bigger than plaintext; however, in [104] ciphertext has the same length order compared to the plaintext. A smaller key size for LWE-based encryption is introduced as the LP scheme in [82]. Another difference between Regev's encryption scheme and LP scheme is that the former uses discrete Gaussian sampler in the key generation step, while LP employs Gaussian Sampler in encryption in addition to the key generation step.

As mentioned earlier, the disadvantages of standard lattices are their large key size and expensive matrix arithmetic, although ideal lattices have become popular due to their efficiency. Recently, promising designs for encryption [71] and key exchange [26] have been published.

The most popular lattice-based schemes existed in the literature are listed in Table 2.

Details on security level, public key, secret key and ciphertext size of lattice-based key exchange and public key encryption schemes can be seen in Table 3.

Furthermore, details on security level, public key, secret key and digital signature size of lattice-based digital signature schemes are listed in Table 4.

Table 2. Popular lattice-based schemes

Lattice Type	**Schemes**		
	Public Key Encryption	**Digital Signature**	**Key Exchange**
Standard Lattices	LP [82] Lizard [35] NTRUEnrypt [66]	GGH [1] NTRUSign[1][65] GPV [58] Lyubashevsky [87] BG [17] TESLA [9]	Frodo [26] spKEX[23]
Ideal Lattices	NTRU [67] NTRU Prime[21] Ring-Lizard [35] trunc8 [121] HILA5 [120]	Lyubashevsky [88] GLP [61] GPV [53] BLISS [49] BLISS-B[48] Ring-TESLA [36] TESLA# [18] BLZZRD [119] GLYPH[2][37]	NewHope [10] NewHope-Simple [11] JARJAR [10] BCNS [28] HILA5 [120] NTRU KEM [74]
Module Lattices	Kyber PKE [27]	Dilithium [50] pqNTRUSign [68]	Kyber KEM[27]

[1] adapted from GGH digital signature scheme
[2] adapted from GLP digital signature scheme

2.5.1 Public Key Encryption schemes. Public key encryption is employed to encrypt and decrypt messages between two parties. Additionally, it is a basis for other cryptography schemes such as digital signatures. It consists of three steps including key generation, encryption, and decryption. The first party (Alice) generates two set of keys and keeps one of them private (sk_{Alice}) and distributes the other key (pk_{Alice}) to other party (Bob). Let assume Bob wants to send message M to Alice. Bob encrypts the message as $S = Enc(M, pk_{Alice})$ where pk_{Alice} is Alice's public and Enc is encryption cipher. Thereafter, Alice decrypts the message as $M = Dec(S, sk_{Alice})$ where Dec is the decryption cipher and sk_{Alice} is Alice's private key. Similarly, if Alice wants to send message to Bob, she should encrypt the message with Bob's public key (pk_{Bob}). Encryption and decryption ciphers are one-way functions and are known publicly; hence the only secret data are private keys of the recipients which should be impossible to uncover using their corresponding public keys.

2.5.2 Digital Signatures. Digitally signing a document involves sending the signature and document separately. In order to verify the authenticity of the message, the recipient should perform verification on both the signature

Table 3. Comparison of popular lattice-based key exchange and public key encryption schemes

Scheme	PQ security		Failure Probablity[1]	Size		
	SVP	CCA		Secret Key	Public Key	Ciphertext
BCNS	78	-	?	4096	4096	4224
JarJar	118	-	55	896	928	1024
NewHope	255	-	61	1792	1824	2048
Frodo rec.[2]	130	-	36	1280	11296	11288
Kyber light	102	169	169	832	736	832
Kyber rec.[2]	161	142	142	1248	1088	1184
Kyber paranoid	218	145	145	1664	1440	1536
NTRU KEM	123	∞	∞	1422	1140	1281
NTRU Prime	129	∞	∞	1417	1232	1141
HILA5	255	135	135	1792	1824	2012
trunc8	131	-	45	128	1024	1024
NTRU ees743ep1	159	-	112	1120	1027	980

[1] Failure is $-log_2$ of the failure probability;

[2] For recommended parameters;

Table 4. Comparison of popular lattice-based key digital signature schemes

Scheme	Security		Size		
	PreQ	PostQ	Secret Key	Public Key	Signature
GPV	100	?	256 B	1.5 kB	1.186 kB
BG	128	?	0.87 MB	1.54 MB	1.46 kB
TESLA–128	128	?	1.01 MB	1.33 MB	1.280 kB
TESLA–256	256	128	1.057 MB	2.2 MB	1.688 kB
GLP	100	<80	256 B	1.5 kB	1.186 kB
BLISS-I[2] BLISS-BI[1]	128	<66	256 B	896 B	700 B
TESLA#–I	128	64	2.112 kB	3.328 kB	1.616 kB
TESLA#–II	256	128	4.608 kB	7.168 kB	3.488 kB
Ring-TESLA-II	118	64	1.92 kB	3.328 kB	1.568 kB
Dilithium rec.[3]	138	125	?	1.568 kB	2.435 kB
RSA-4096	128	?	512 B	512 B	512 B
ECDSA-256	128	?	64 B	64 B	128 B

[1] BLISS-BI speeds up BLISS-I by factor of 1.2×;

[2] Speed optimized;

[3] For recommended parameters;

and the document. Digital signature consists of three steps including key generation, Sign_{sk}, and Verify_{pk}. In the first step, a pair of keys, secret key (sk) and public key (pk), are generated; signer keeps the secret key and all verifier parties have the public key of signer. In the sign step, signer applies the encryption algorithm on input message M with its private key and produces output S as $S = \text{Sign}_{sk}(M, sk_{signer})$. Signer sends the tuple (M,S) to the Verifier who applies Verify_{pk} (M,S) and outputs 1 if M and S are a valid message and signature pair; otherwise, S is rejected as the signature of message M. As an example of hash and sign procedure: in the sign step, message M is hashed as $D = h(M)$ where D is the digest. Signer applies the encryption algorithm on D with its private key with the output of

$S = Enc(D, sk_{signer})$. Afterwards, signer sends the pair of (M,S) to the verifier. Subsequently, verifier uses public key to decrypt S as $D' = Dec(S, pk_{signer})$. Then verifier compares $D = h(M)$ (same hash function is shared between signer and verifier) and D'; signature is verified if $D' = D$; otherwise, the signature is rejected. The steps outlined for sign and verify are just an example (used in RSA); hence sign and verify steps might be slightly different for various signature schemes, but follow the same idea.

Lattice-based signature schemes belong to one of three classes including GGH [1], hash-and-sign (e.g. GPV [58]), and Fiat-Shamir signatures (e.g. BG [17], GLP [61], and BLISS [49]). Because of the high standard deviation of Gaussian sampler, implementation of lattice-based digital signatures is more difficult than lattice-based encryption schemes. Besides, the need for hash function components and rejection step makes digital signature more complex. Lattice-based signature schemes which adopt Fiat-Shamir transformation are presented by Lyubashevsky which are based on the Short Integer Solution (SIS) problem [87]. Lyubashevsky improves this scheme by establishing it in an efficient manner for Ring-SIS and Ring-LWE which culminates in smaller signature and key size [88]. BLISS signature is an optimized version of Lyubashevsky's signature where a binomial Gaussian sampler is used in the rejection sampler component, resulting in a remarkable reduction in the standard deviation of the Gaussian distribution. However, GLP and TESLA scheme employs uniform sampling instead of using Gaussian sampling.

Bai and Galbraith proposed a provably secure scheme (BG signature) that can be implemented using uniform distributions which leads to small signature based on standard LWE and SIS problems [17]. Security of BG signature scheme in the random oracle model, is based on standard worst-case computational assumptions on lattices.

Software and hardware implementations of digital signature schemes are given in 3.2 and 3.3, respectively.

2.5.3 key exchange mechanism. Key exchange is the process of exchanging keys between two parties in the presence of adversaries. If parties use symmetric keys, the same key is shared between them; otherwise, public and private key of parties should be exchanged. There are numerous public key exchange methods reported in the literature. For classical cryptography, Diffie–Hellman is a practical public key exchange that has been used widely. For lattice-based cryptography, a key exchange protocol based on ring learning with error problem is proposed by Peikert [103] that has four steps, including (*Setup*, *Gen*, *Encaps* ,*Decaps*), which parties will share an ephemeral secret key that can be employed in future connections.

A practical constant-time software implementation of the Peikert's Ring-LWE key exchange protocol is proposed in [28], namely BCNS, which could be added as the key exchange protocol to the transport layer security (TLS) protocol in OpenSSL along with RSA as the authentication and key SHA-256 as the hashing method. One year later, NewHope key exchange was introduced which uses a simpler noise distribution instead of Gaussian sampler [10]. NewHope has been at the center of the attention of research and industry communities such that Google released the Chrome Canary which uses NewHope as the key exchange protocol along with elliptic curve Diffie–Hellman as the authentication protocol [29]. Key exchange scheme based on standard lattices is proposed in [26], namely Frodo, that can be integrated into TLS which sacrifices a server's throughput in order to archive post-quantum security.

Practical implementations of key exchange mechanism are given in Section 3.2.

3 IMPLEMENTATION CHALLENGES

This section surveys implementation of lattice-based cryptographic schemes using software, hardware, software/hardware codesign and DSP techniques. First we focus on the implementation of key arithmetic modules such as Gaussian sampler and matrix/polynomial multiplication in Section 3.1. Software and hardware implementations of lattice-based

cryptographic schemes are described in Section 3.2 and Section 3.3, respectively. Implementations of lattice-based schemes using hardware/software codesign techniques are described in Section 3.4. The only implementation of a lattice-based scheme using DPS implementation is described in Section 3.5.

Table 5 presents a birds-eye view of popular implementation schemes, and can be helpful as a visual/organizational reference during the discussion of various implementation schemes.

Table 5. Popular implementation of lattice-based schemes

Lattice Type	Schemes		
	Software	**Hardware**	**Hardware/Software**
Standard Lattices	public key encryption: [35] digital signature: [53][39] key exchange: [23]	public key encryption: [71]	-
Ideal Lattices	public key encryption: [41] [83] [41] [110] [32] digital signature: [63] [53] [101] [106] [25] [110, 111] [7] key exchange: [76] [11] [12] [60]	public key encryption: [59] [108] [116] [107] [114] digital signature: [61] [106] [62] [73] key exchange: [127]	digital signature [16] [15]
Module Lattices	-	-	-

3.1 Implementation of Arithmetic Modules

In this section, practical implementations of Gaussian sampler and polynomial multiplication on both hardware and software platform are presented. There is only one hardware implementation of matrix multiplication (for standard lattices) available in the literature which we detail in Section 3.1.2.

3.1.1 Gaussian Sampler. Authors of [52] provide a survey on different algorithms of efficiently computing the exponential function (Gaussian distribution curve) on resource constrained devices in terms of memory capacity. In order to decrease memory footprint, pipelining the sampling algorithm is offered. To be more specific, authors divide distribution curve into rectangles with the same probability and choose the rectangles based on the Knuth-Yao method. Subsequently, the Knuth-Yao method is employed another round for rectangle itself. Tail probabilities in discrete distribution are relatively small which provide the chance of approximating them with lower precision arithmetic. Consequently, on the fly tail construction using standard double-precision floating-point precision is suggested. Although the offered idea could significantly reduce memory (lookup table) footprint since lookup tables just store non-tail probabilities; however, considerable floating point arithmetic overhead is imposed on the system.

Software Implementation. Buchmann et al. [30] design and implement (C++) a flexible Gaussian sampler named Ziggurat which sets a trade-off between memory footprint, precision, and execution time of noise sampling from discrete Gaussian distribution. The area under the probability density function (PDF) is divided into rectangles with the same area which are employed to minimize calculation of expensive exponential function. More rectangles (more

memory footprint) results in higher precision and better performance. The Ziggurat sampler is attractive because of its potential flexibility that makes it a good candidate to use in cryptoengines of either high performance servers (allocate more memory to reach better performance and precision) or low speed resource constraint embedded devices (use few number of rectangles to minimize memory consumption). As a reminder, two points are chosen and crossed line from those points is used as an approximation of the PDF curve. In order to increase performance of Ziggurat sampler, more rectangles are required which imposes significant memory overhead since it needs to re-compute all the rectangles and save them in the memory. A better way to improve Ziggurat sampler is to increase number of approximation lines (by adding two extra points) which culminates in minimizing number of calculating the exponential function [97]. In other words, more approximation lines decreases probability of rejection by providing a more precise approximation of the curve. Consequently, performance and memory occupation are improved by employing more approximate lines.

Hardware Implementation. Roy et al. [117] implement the first high precision and low area hardware implementation of Knuth-Yao sampler with small standard deviation on a Xilinx Virtex 5 FPGA. Knuth-Yao involves a random walk tree traversal which imposes expensive sequential bit scanning and wide ROM footprint. To improve performance, authors traverse the discrete distribution generating (DDG) tree by employing relative distance of intermediate nodes. Column-wised, instead of row-wised, storing of the samples probability in ROM remarkably improves performance of the sampler. Numerous zeros in probability matrix is well compressed by applying one-step compression which culminates in near-optimal number of random bits to generate a sample point. The main drawback of the presented Knuth-Yao sampler is its vulnerability to timing and power attacks due to the non-constant time random walk which is solved by a random shuffle approach in order to eliminate leaking the timing information [115]. It should be mentioned that authors just offer the solution (random shuffle) and do not evaluate hardware implementation of the shuffler. In the new implementation, performance of the sampler is improved by employing small LUTs with the input of the random bits and output of the sample point with high probability or an intermediate node positioned in the DDG tree. Employing a lookup table with 8-bit input results in hitting a sample point (eliminate need of expensive bit-scanning procedure) with probability of 97% by eight random bits. Additionally, a more compact sampler is achieved by reducing the width of ROM and random bit generator.

A highly precise (large tail bound) and area efficient cumulative distribution function (CDF) inversion variant of discrete Gaussian sampler is implemented on Spartan-6 FPGA [43]. Authors reduce area occupation by employing piecewise comparison (results in 90% saving of the random bits) and avoiding comparison of large numbers which is an improvement on [108]. Further improvement is achieved by employing small lookup tables with high hit rate which culminates in performance improvement. Performance of the proposed Gaussian sampler is improved twofold by the same authors with a software implementation (Intel Core i7-4771) [42]. The main challenge to improve performance on a general purpose processor is the large number of random bits that are consumed by the sampler to generate a random number. This obstacle is alleviated (around 30%) by employing multi-level fast lookup table (using 8 smaller lookup table instead of one). Further speed improvement of the sampler is gained by applying multithreading technique. The main security drawback of both hardware and software implementation of discrete Gaussian sampler by Du and Bai is its vulnerability to timing attacks due to the fact that traversal of binary tree is not accomplished in constant time [70].

In [70], a comprehensive evaluation of various practical hardware implementation of time-independent discrete Gaussian samplers, including Bernoulli, discrete Ziggurat (First hardware design on FPGA), CDT, and Knuth-Yao, along with their weaknesses/strengths and comparison with state-of-the-are designs in terms of memory footprint, performance, and FPGA resource consumption is presented. Authors analyze different quantum secure parameters for

public key encryption scheme and digital signature. In case of digital signature, CDT Gaussian sampler provides higher throughput with lower area. Due to the disappointing performance of hardware Ziggurat implementation, authors prohibit use of Ziggurat sampler for digital signature schemes. Similarly, CDT sampler achieves better balanced area and throughput for public key encryption. However, allowing use of BRAMs, makes Knuth-Yao variant a much superior design in terms of area and throughput. Authors use BRAMs in order to decrease occupied slices in FPGA and to save precomputed values which significantly improves performance.

An area optimized hardware implementation of Bernoulli sampler that employs Bernoulli evaluation instead of evaluation of `exp()` is presented in [107]. Rejection probability is high due to the absence of binary Gaussian distribution (easy to sample intermediate sampler) which results in increasing the entropy consumption and runtime. Although proposed sampler is suitable for encryption schemes, digital signatures cannot benefit from it. Hardware implementation of Bernoulli sampler with binary Gaussian distribution is presented in [106] for `BLISS` scheme on Xilinx Spartan-6 FPGA.

Göttert et al. [59] propose the first hardware implementation of discrete Gaussian sampler. Authors use rejection sampling in their software implementation; however, because of the obligatory floating point arithmetic, authors prefer to employ lookup tables in their hardware implementation of implement Gaussian which uses a pseudo random bit generator to index the Gaussian distributed values in the stored array. It should be mentioned that proposed sampler has unsatisfactory precision which has far distance from golden discrete Gaussian distribution due to the small tail bound. Authors employ a fully parallel architecture to design a polynomial multiplier which provides high throughput but makes the design extremely big which cannot be fitted into the largest Virtex-7 FPGA family.

A compact Ring-LWE cryptoprocessor is implemented on [116] with the main goal optimizing NNT multiplication which is accomplished by the reduction in fixed computation (4 NTT instead of 5 NTT) and in reducing the pre-scaling overhead. Besides, NNT Memory access is minimized by storing two coefficients in a single word, processing two pairs of coefficients together, and eliminating idle cycles. Authors avoid using ROM to store the twiddle factors and compute them on demand. Besides, they reduce security by limiting coefficients of secret key to be binary, instead of Gaussian distributed, which gives the opportunity to replace multiply with addition operations. Small lookup tables are used in the Knuth-Yao discrete Gaussian sampler to avoid expensive bit scanning [117] to improve speedup; additionally, ROM widths are reduced which results in a more compact and faster sampler than Bernoulli [107].

3.1.2 Multiplier.

Software Implementation. In order to achieve high throughput polynomial multiplications using NTT algorithm on Nvidia GeForce GTX 280 GPU, an efficient 24-bit modular multiplication is proposed in [54]. Employing CUDA FFT kernel and Chinese Remainder Theorem (CRT), proposed method provides better speedup compared to GMP and NTL libraries for moderate coefficient bit-length.

Sparse polynomial multiplication for lattice-based schemes is proposed in [8] for the first time which improve the performance of digital signature proposed in [62] by 34%. Same authors further improve the sparse polynomial multiplication (a modified FFT algorithm) in [6]. Authors implement the Schönhage-Strassen polynomial multiplication on NVIDIA Geforce GT 555M GPU and compare its performance with existed multiplication schemes, including iterative NTT, parallel NTT and CUDA-based FFT (cuFFT) for different integer size.

Akleylek et al. [6] propse a software implementation of sparse polynomial multiplication with sliding window that bears around 80% of speed improvement compared to NTT [63] on an Intel Core i5-3210M processor . Authors assume to have polynomials with coefficients with three possible values including -1,0 and +1. Multiplications by zero are

avoided and for +1 and -1 cases addition and subtraction are used, respectively. It should be mentioned that their idea can be used polynomials with arbitrary coefficients by performing substituting a multiplication with a loop of additions.Performance of the proposed method depends on high number zeros and numerous identical patterns which make the system prone to timing attacks.

`FNLlib` [3] is a scalable, efficient, and open source C++ library contains optimized arithmetic operations on polynomials for ideal lattice-based cryptography schemes. Authors employ algorithm optimizations, including fixed sized Chinese Remainder Theorem (CRT), scalar modular multiplication and NTT algorithm, and programming level optimizations, such as SSE and AVX2 SIMD instructions, to achieve several orders of magnitude improvement of speed compared to the generic libraries for polynomial arithmetic operations. Authors use NFLlib for the RLWE encryption scheme and homomorphic encryption and compare their efficiency with classical cryptographic schemes (like RSA) and libraries (like NTL).

Recently, an efficient modular reduction is introduced which can be used as a alternative to modular reduction algorithms [86]. The main idea is to limit coefficient length to 32 bits. Consequently, by employing the new technique in NTT, reduction is only required after multiplication. Combined with lazy reduction in NTT, speed improvement of factor 1.9 for C implementation (on 64 bit platform) and 1.25 for AVX2 vector implementation compared to NewHope (tolerant against timing attacks) is achieved. However, due to lack of 64-bit register, proposed reduction technique does not provide any speed up on 32-bit microcontrollers [12]. Additionally, authors use signed integer arithmetic which optimizes number of add operations in both sampling and polynomial multiplication.

Hardware Implementation. The only hardware implementation of standard lattice-based encryption scheme based on LWE problem performs the multiply-accumulate (MAC) operations of matrices in the encryption scheme by utilizing a dedicated DSP48A1 unit of the Spartan-6 FPGA to achieve an area optimized hardware implementation of standard LWE based encryption engine [71].

Pöppelmann et al. [109] propose the first hardware optimization of polynomial multiplication (NTT) in the ideal Ring-LWE lattice-based encryption is implemented on Spartan-6 Xilinx FPGA with the main goal of minimizing the area. Authors design a sequential NTT (one butterfly operator) that stores twiddle factors in a dedicated ROM which imposes memory overhead but achieves decent performance. Twiddle factors refer to different powers of the multiplicative operands. An optimized version of presented NTT polynomial multiplier is employed in [108] to design in a Ring-LWE encryption engine. With acceptable runtime, authors of [14, 116] present an optimized hardware implementation of NTT introduced in [109] to compute polynomial multiplication in a Ring-LWE on the smallest Spartan-3 Xilinx FPGA. The main idea is to compute twiddle factors on the demand instead of storing them in the ROM. By replacing the modulus with a Fermat number, modular reduction is improved and shift operation could be used instead to polynomial exponentiation. However, these proposed optimization could not take advantage of inherent parallelism in NTT. It should be mentioned that authors of [14] do not provide implementation of the whole cryptoengine.

Chen et al. [33] present a high performance polynomial multiplication for Ring-LWE encryption cryptosystems is implemented in hardware on a Spartan-6 FPGA by exploiting the parallel property of the NTT. Authors provide different secure set of parameters by which efficient Ring-LWE encryption and SHE could be achieved. They prove that polynomial multiplication can be done by computing the negative wrapped convolution by which there is no need to compute the modular reduction. Besides, length of FFT/ inverse-FFT and point-wise multiplication is halved compared to the zero padding method. To be more specific, the proposed architecture for polynomial multiplication consists of

two butterflies and two point-wise modulo p multiplier (p has flexible length) which produce outputs (equivalent to two parallel NTT) that could be used to perform the inverse-FFT.

Du and Bai [45] propose a scalable and efficient polynomial multiplier architecture that take advantage of NTT's parallelism (implemented on Xilinx Spartan-6 FPGA) which provides a speed and area trade-off. In [109] and [14, 116] one and two butterfly operators are employed, respectively; however, [45] use b (power of 2) butterfly operators to improve speed of the polynomial multiplier which perform multiplication of two n-degree polynomials in $(1.5n + 1.5n\log n)/b$ cycles. To improve area (minimize required area), authors employ the cancellation lemma to minimized number of constant factors. Butterfly operation takes two coefficients (x,y) and one constant factor (ω) and makes two new coefficients ($[x+\omega y] \bmod p$, $[x-\omega y] \bmod p$). Butterfly can be used as a modulo p multiplier (by setting $x = 0$). Besides, in the first stage of inverse NTT the constant factor (ω) is 1, hence new coefficients are ($x+y$] mod p, [x-y] mod p). Employing presented facts about butterfly operation, necessary clock cycles to calculate a sequential polynomial multiplication is ($1.5n + 1.5n\log n$) which reduces $3.5n$ of required cycles.

Györfi et al. [64] perform a thorough evaluation of various candidates to implement modular FFT is perform on Xilinx Kintex-7 FPGA. Authors study three architectures in the diminished-one number system (computations over Z_{2k+1}) for different parameters in order to meet various factors such as run-time execution, throughput, occupation, and scalability. The first architecture yields the best performance, using pipelined modular butterfly-based FFT in which FFT core is throughput optimized. The other two architectures are serial distributed arithmetic-based and nested multiplication which occupy less area than butterfly-based FFT.

By performing on-the-fly performing bit-reversal step along with a new memory access scheme, (to load/store coefficients in calculating NTT) Du and Bai [46] demonstrate 30% savings in time and space (compared to [109]) on a Spartan-6 FPGA. The idea is to load/store at address bit-reverse(i) instead of load/store at address i in memory, which means ith coefficient of NTT's output is located in the $bit-reverse(i)$th memory location. Consequently,the bit-reversal step in the inverse-NTT is eliminated. Authors employ two Block RAMs on FPGA which provide interleaving, hence two parallel NTT can be interleaved. Authors apply their optimization to the NTT of a RLWE base public key cryptosystem [44]. In addition, it is assumed uniformly random polynomial in the public key scheme to be fixed, hence precomputing the NTT of it offline improves the performance. In the above two articles, only one butterfly operator is used; however, by adapting bit-reversal and memory saving methods of above articles, authors propose a high-speed polynomial multiplication architecture with four butterfly operators that achieve on average 2.2 speedup improvement [47]. Two butterfly operators are used to calculate the ith level and other two perform $(i + 1)$th level calculations in pipeline by using results of the ith stage.

3.2 Software Implementations

3.2.1 Public Key Encryption. An efficient software implementation of Ring-LWE based encryption was proposed for ARM Cortex-M4F micro-controller [41]. The main concern was maximizing the speed (assembly level optimization) and minimizing memory footprint (by storing two coefficients in one word). They employed the Knuth-Yao algorithm to achieve fast discrete Gaussian sampling, and use the platform's True Random Number Generator (`TRNG`) to generate random numbers. Authors employ optimization of the paper [116] including parallelization and reduction of load and stores by integrating multiple coefficients into one large word. Besides, the negative wrapped convolution is unrolled twice to reach efficient polynomial multiplication by performing load/store of coefficients with a single instruction.

Along with applying the optimizations of Knuth-Yao algorithm in [116], a byte-wise scanning method improves speedup of Ring-LWE encryption scheme on a 8-bit AVR processor [83]. By applying sophisticated memory alignments

for storing coefficients, about 20 percent decrease in RAM usage is achieved. For NTT computation a couple of optimization techniques are employed including approximation based reduction and negative wrapped convolution.

By replacing the Gaussian noise distribution with a uniform binary error distribution, a high performance and lightweight public key encryption scheme implemented on small and 8-bit ATXmega128 and 32-bit Cortex-M0 micro-controllers [32]. By evaluating hardness of binary LWE against hybrid attack, security of scheme is determined. The main advantage of this scheme over Lindner-Peikert's proposal (LP) [82] is its smaller key and chipertext size. In terms of the speed, it is beaten by the scheme in [83] with slightly higher memory footprint. Similarly, proposed design in [110], uses NTT with precomputed twiddle factors and eliminating the bit reversal step which results in twofold performance improvement.

Yuan et al. [130] provide a portable JavaScript implementation of lattice-base cryptography schemes on PC web browsers, Tessel (an embedded system for IoT applications), and Android devices. To compute polynomial multiplication in Ring-LWE schemes NTT is used, while Karatsuba algorithms [77] is employed for NTRU schemes. In order to reduce the execution time, inverse transform sampling is employed in which possible values are precomputed and stored in a lookup table.

Reparaz et al. [114] implement a masked ring-LWE scheme on a Virtex-II FPGA and 32-bit ARM Cortex-M4F which is Differential power analysis (DPA) resistant. In order to be resilient to first-order side-channel attacks, a constant time masked decoder with high success probability is implemented. Entire computation is done in the masked domain by employing a dedicated masked decoder which imposes considerable time and area overhead compared with unprotected design . Cheon et al. exploit a variant of learning with error problem called learning with rounding (LWR) problem [2] and represent Lizard and its ring variant (Ring-Lizard) [35]. By employing LWR, a smaller ciphertext is achieved. Discrete Gaussian error distribution is replaced with an efficient rounding process with smaller modulus. Based on the results, Lizard beats NTRU and RSA encryption schemes by factors of 3 and 5, respectively. The main idea behind the Lizard is to eliminate the least significant bits of the ciphertext rather than integrating the message with some error.

3.2.2 Digital Signature. Pöppelmann et al. [110, 111] compare implementations of Ring-LWE encryption scheme and the bimodal lattice signature scheme (BLISS) on an 8-bit AVR micro-controller. They review various NTT algorithms (written in C) and optimize (using assembly language) polynomial multiplication for ideal lattice-based cryptography schemes. To be more precise, employing column-wise multiplication and ignoring zero coefficients lead to an efficient polynomial multiplier. However, using precomputed twiddle factors in NTT computations requires more memory footprint. Official release code of Keccak [22] for AVR is used for the random oracle which is needed for signing and verification. Other optimization offered by the authors are removing bit reversal step during polynomial multiplication and applying the Gaussian sampling in a lazy manner. Compared to RLWE encryption, BLISS needs larger standard deviation for sampling from Gaussian distribution; candidates for Gaussian sampling are CDT sampling [24] with binary search , which results in large tables, and Bernoulli [25], that impose remarkable performance overhead. Consequently, for Gaussian sampler, KL-convolution sampler [106] is used which consume less flash memory compared with the CDT and Bernoulli samplers. The BLISS implementation consumes the least flash footprint on AVR and has the lowest published runtime through 2015

BLISS-B [48] (with the same security level) improves performance of original BLISS 2.8 times by employing ternary representation of polynomials in order to shorten length of the random numbers. During key generation, keys will not be rejected which leads to 5-10 times enhancement of key generation step runtime. Generated signatures by BLISS

and BLISS-B are compatible with each other, allowing signatures generated by one to be valid for the other. Although generated keys of BLISS-B could not be used in BLISS, BLISS generated keys are compatible with BLISS-B.

Oder et al. [101] present an efficient software implementation of BLISS on ARM Cortex-M4F to optimize the throughput along with minimizing memory footprint. Authors evaluate efficiency of variety of Gaussian samplers including the Bernoulli, Knuth-Yao, and Ziggurat [30]. In order to improve NTT computation, assembly level optimization along with precomputed coefficients are employed. They conclude that Knuth-Yao sampler is the best candidate for large devices, while for constrained devices Bernoulli is more favorable.

Güneysu et al. [63] presents a highly optimized SIMD implementation of lattice-based digital signature introduced in [61] (GLP signature) and implement it on Intel's Sandy and Ivy Bridge processors. To exploit SIMD support of Advanced Vector Extensions (AVX), each 512 double-precision floating-point array of coefficients is aligned on a 32-byte limit. In addition, Gaussian sampler is replaced with the uniformly sampling from {-1,0,+1}; besides, modular reduction of coefficient is done in a lazy manner. However, signature size and security level of implemented scheme is inferior to that in BLISS.

El Bansarkhani and Buchmann [53] implement the first software implementation (space and speed optimized) of GPV signature scheme [58] by employing the Micciancio and Peikert (MP) trapdoors [92] on the Sun XFire 4400 server equipped with 16 Quad-Core AMD Opteron. Besides the matrix version, a much faster Ring-LWE based variant of the scheme is provided which has around 3-6 and 3-9 times better speed than matrix version for sign and verification steps, respectively. Due to the small number of stored entries, instead of rejection sampling, the inversion transform method is used for discrete Gaussian sampling during integer key generation; however, in the randomize rounding, rejection sampling [58] is used. It worth mentioning that for random oracle SHA256 and a pseudo random number generator from [31] are used.

Dagdelen et al. [39] propose a high-speed software implementation BG signature on Intel with AVX instructions and ARMv7 with Neon vector instructions. In comparison with BG signature, authors proposed an optimized rejection sampling. Based on the results, small performance degradation is observed by employing standard lattices instead of ideal lattices which is a great achievement because there is no quasi-logarithmic arithmetic scheme like NTT for standard lattices.

Boorghany and Jalili [24, 25] propose efficient software efficient lattice-based GLP and BLISS authentication protocols for resource constrained smart cards and micro-controllers (ARM and AVR). Authors perform a design space exploration by choosing different parameter sets for FFT and Gaussian Sampler (Knuth-Yao and Bernoulli) along with various pubic key encryption schemes. They conclude that lattice-based schemes are efficient enough to be implemented on constrained devices.

Alkim et al. [9] introduce TESLA (a tightly secure signature in random oracle model resulted) by a tight reduction to LWE-based problems on standard lattices and implement it on Intel Core-i7 4770K (Haswell) . Their proposed design is adopted from [17] which is faster and smaller than the same scheme in [39] due to employing parallel matrix vector multiplication and lazy reduction. Authors propose two variants TESLA-128 and TESLA-256; the former one, TELSA-I, is not quantum resistant, while the latter, TELSA-II, provides the first lattice based digital signature with 128 bit security against quantum computers. The main drawback of TESLA is large public key size of about 1 MB. A fast, small, and provably secure Ring-LWE based software implementation of TESLA [9] is presented in [7] by the same authors on the same platform which reduce the key size about 3 order of magnitude. To generate signatures, uniform sampling is employed instead of Gaussian sampler which BLISS uses. The propose Ring-TESLA benefits from the AVX2 instructions, which has a one cycle throughput for eight doubles integers. To instantiate from the Ring-TESLA there is a problem in

parameter selection, which leads to rejection of valid signatures in verification stage. The problem is solved in [36] by new parameter selection method and in [18] by altering the algorithm. Based on the claims in [18], TESLA and Ring-TESLA are using global parameters which results in employing a fixed lattice for all the signatures that could weaken the signature scheme. The most recent version of TESLA [9] fixes the problem by adding a new condition to the signing step which results in dropping the speedup, creating a more complex signature scheme, with less success in signing. Barreto et al. introduce TESLA#, a high performance version of Ring-TESLA [18], on Intel Core i7-4770 Haswell processor, which resolves the security problems of TESLA. Further improvement is achieved by designing a more efficient Gaussian which accelerates the key generation step along with avoiding to store all 32 bits of coefficients of polynomial.

BLZZRD [119], a side-channel attack resistant lattice-based signature based on BLISS-B [48], is implemented on a Intel Core-i7 Haswell processor with small signature size but with costly signing step . Authors achieve optimal compression for discrete Gaussian distribution by using Binary Arithmetic Coding (BAC) which leads to more compact signatures compared to advanced Huffman-based signature compressors. Further security improvement is gained by prohibiting leak of information about execution time and power consumption of arithmetic operation by applying randomization which make the signature resistant to timing and power attacks. Additionally, masking properties of Gaussian samples is achieved by randomizing and combining multiple number the of sample vectors.

Dilithium [50] a simple and efficient digital signature scheme resistant to lattice reduction attacks (with same conservative security parameter in [10]) is adapted from the designs in [61] and [17] that uses Fiat-Shamir Abort Framework [87] which is secure in random oracle model (no security proof in quantum random oracle model is presented). Authors implemented Dilithium and its variant Dilithium-G on Intel Core-i7 4770k with comparable efficiency to BLISS. In [61], *hints* are generated (by signer to help verifier to verify the signature) to make the signature smaller; Dilithium improves hint-generation and halves public key size with less than 5% increase in signature size. In fact, authors set *total size* = *signature size* + *public key size* as their size parameter. Dilithium over ring of $Z_q[x]/[x^n + 1]$ ($n = 256$,$q = 2^{23} - s^{23} + 1 = 8380417$) has slightly bigger *total size* than BLISS (over ring of $Z_q[x]/[x^{1024} + 1]$) with the same security level. Dilithium samples polynomial noises from uniform distribution S_η in $[-\eta, +\eta]$ where η is in the range of 3 to 7 for very high secure to weakly secure scheme, respectively. However, Dilithium-G extract noises from Gaussian sampler which results in better security but is vulnerable to timing attacks. Rejection sampling is same for both schemes as if individual coefficients of a signature is not within a certain range, signing procedure must be restarted. Dilithium employs the standard NNT-based polynomial multiplication, however in vectorized version, Dilithium uses integer instructions instead of floating point vector instructions [10]. For recommended security parameters, *signature size* is 2700 bytes, *public key size* is 1472 bytes and (*KeyGen, Sign, Verify*) takes (522992, 2253378, 625968) cycles, respectively.

3.2.3 Key Exchange. A provably secure key exchange protocol based on the ring learning with errors problem is proposed in [76]; the protocol is not a passively secure key exchange scheme because it produces biased keys. Peikert improves the protocol by using a new reconciliation method which generate unbiased keys [103]. A practical constant-time software implementation of the Peikert's Ring-LWE key exchange protocol is proposed in [28], namely BCNS, which can be added as the key exchange protocol to the transport layer security (TLS) protocol in OpenSSL along with RSA as the authentication and key SHA-256 as the hashing method. The most time consuming part of the protocol is Gaussian sampler, which is done by employing a constant-time search on the Cumulative Distribution Table (CDT). For polynomial arithmetic, authors adapt the FFT from Nussbaumer's method [99], in cyclotomic rings whose degree is

power of two which provides efficient modular reduction. BCNS employs a fixed polynomial as the system parameter which could be a potential weak link of the protocol. In addition, selection of large modulus results in lower efficiency and security level, 78-bit quantum security, than expected from a Ring-LWE scheme. Authors of [10], NewHope key exchange, claim that in contrast to the digital signature and encryption schemes, key exchange scheme does not need a high quality Gaussian sampler, which BCNS uses; consequently, a simpler noise distribution is used in NewHope instead of Gaussian sampler. BCNS caches keys ,which could be very dangerous to security of the protocol because of shared-key reused attacks [55], which is solved in the NewHope.

Alkim et al. [10] introduce NewHope, a portable C and highly optimized SIMD implementation (AVX2) of unauthenticated key exchange scheme, that solves the inefficiency (10 times better performance) and security drawbacks of BCNS (increase quantum security level from 78-bit to 128-bit) by optimizing key exchange scheme and better parameter selection. A better analysis of failure probability which results in smaller modulus, on the fly generation of polynomial system parameter, efficient polynomial arithmetic (combining Montgomery and Barret reduction and employing polynomial encoding), and using centered binomial instead of the of discrete Gaussian distribution are the main improvements of NewHope over the BCNS. NewHope has attracted attention of research and industry communities such that Google released the Chrome Canary which uses NewHope as the key exchange protocol along with elliptic curve Diffie–Hellman as the authentication protocol [29].

Alkim et al. [11] propose `NewHope-Simple`, a simpler variant of the Newhope with the same performance and security level. Simplicity in this design is achieved by eliminating error-reconciliation mechanism [76] with 6% message size overhead. Authors discard the least significant bits of each coefficient due to their negligible impact on the successful plaintext recovery. Additionally, authors encode a single key bit into 4 coefficients that results in reduction of the ciphertext length. In `NewHope-Simple`, polynomial a can be a fixed, while the original `NewHope` generates a on the fly for every single run of the scheme. Software implementation of NewHope on ARM Cortex-M family, low power Cortex-M0 and high performance Cortex-M4, is presented in [12] which is the first key exchange scheme with security level of 128-bit on constrained embedded devices. Authors optimize all hot regions of protocol in assembly, including error reconciliation, the uniform noise generation by ChaCha20 stream cipher [20], and NTT/NTT^{-1}. For NTT, authors set a memory-time trade-off for precomputing powers of constants (design parameters) by which only a subset of the powers of constants are precomputed and stored in the table. Gueron and Schlieker further optimize the NewHope by putting the focus on pseudorandom generation part which results in 1.5 times better performance on the Intel skylake processors [60]. Authors improve the sampling step by lowering the rejection rate (from 25% to 6%) and exploit the parallelism in pseudorandom generation (replace SHAKE-128 with parallelized SHA-256 or AES block cipher) and rejection sampling (employing AVX vector instructions). However, the decrease rejection rate requires 4 subtraction of modulus from the sample. In [86], by employing novel reduction technique, during NTT calculation, modular reduction after additions of two polynomials is eliminated which results in speed improvement of 1.9 and 1.25 for C and AVX implementations, respectively.

`HILA5` [120], a side channel resistant Ring-LWE-based key exchange scheme (and also public key encryption) with the same security parameters ($n = 1024, q = 12289$) and sampler (binomial sampler ψ_{16}) as NewHope is presented by Saarinen [120] and has been tested on Intel Core i7-6700 CPU; also it has been integrated into OQS and OpenSSL. HILA5 uses SafeBits, improved version of Peikert reconciliation mechanism [103], to reach slightly smaller messages than NewHope (36 bytes which is 0.9%) at the same security level by generating unbiased secret bits and hence less randomness in secret bits. HILA5 employs an efficient constant time error correction block to correct 5 bits of error

which results in decryption failure of 2^{-128} compared to NewHope's failure rate of 2^{-64} by sacrificing less than 4% of performance. Considering the higher reliability of HILA5, it can be employed as a public key encryption scheme.

Frodo [26], the first practical implementation of key exchange scheme based on standard lattices, original LWE problem [113], is secure against cache-timing attacks. Like BCNS, Frodo could be integrated into OpenSSL such that Google has announced that Frodo is used in 1% of Chrome web browsers. Matrix arithmetic compared to polynomial arithmetic imposes considerable overheads on bandwidth (4.7 times more than NewHope), throughput (1.2 less throughput than NewHope), and performance (8 times slower than NewHope). Huge memory overhead is imposed if matrix variant should be saved in the memory. By generating and afterwards discarding the matrix variant (on-the-fly), memory overhead is alleviated. Besides, authors use an efficient Gaussian sampler, inversion sampling method, which employs precomputed tables. Based on the authors's claim, integrating Frodo (LWE-based post-quantum key exchange protocol) into TLS halves the server throughput. To tackle the ring related attack, NTRUPrime is proposed as a more secure scheme by using a combination Karatsuba, schoolbook, and Toom's multiplier [21]. Consequently, NTT-friendly prime and polynomial are not crucial which results in negligible drop in performance compared to [10].

Open Quantum Safe (OQS) [124] software platform is designed to evaluate proposed quantum-resistant schemes which has an open-source library (contains C implementation of BCNS, NewHope, and Frodo) of post-quantum cryptographic schemes. More importantly, OQS offers the chance to integrate the quantum resistant schemes into classical applications and protocols with the goal of minimizing the software change; besides, OQS provide opportunity to compare post-quantum schemes with each other or with classical cryptographic algorithms.

Jin and Zhao [75] present symmetric (OKCN) and asymmetric (AKCN) LWE and Ring-LWE based key exchange mechanisms; however, NewHope and Frodo are Ring-LWE and LWE-based symmetric key exchange schemes, . OKCN, optimally-balanced key consensus with noise, could be used for key transport and encryption, while AKCN, asymmetric key consensus with noise, only could be employed for key transport. In the proposed scheme, the server sets the session key before starting the key exchange mechanism. Consequently, it provides opportunity to encrypt the message offline which provides higher security and better workload balance. Compares with Frodo, OKCN-LWE produces a much smaller matrix by eliminating the least significant bits of each LWE sample which results in less computation for matrix arithmetic; smaller matrix also results in faster generation and sampling of the matrix. OKCN could achieve higher security and efficiency than Frodo. With the same set of parameters (same security level), OKCN-LWE consumes negligibly more bandwidth (30% increase) than that of Frodo, while its failure probability is remarkably lower. Employing the same optimization techniques, Ring-LWE based version of OKCN, which adopts the same noise distribution and parameters of NewHope, provides a more computationally efficient scheme than NewHope. It should be mentioned that authors integrate the OKCN-LWE scheme into the open safe project platform [124].

Kyber [27] is a highly optimized chosen ciphertext attack (CCA)-secure module lattice-based key encapsulation mechanism whose security is based on module learning with error problem (Modulue-LWE) [81]. Classically ideal lattices with their ring structure decrease public key and ciphertext size of standard lattices schemes by sacrificing security assumption. Module lattices proposed to fill the gap by believing that full ring structure is excessive [81]. Authors define IND-CPA [5] public key encryption scheme under Modulue-LWE (M-LWE) hardness assumption and apply a variant of Fujisaki-Okamoto transform KEM [69]to build a CCA-secure key encapsulation mechanism. Employing CCA-secure KEM, they design CCA-secure key exchange and authenticated key exchange under hardness assumption in the classical and quantum random-oracle models. Centered Binomial distribution B_η for integer $\eta \geq 0$ is defined as

[5]Indistinguishability under chosen-plaintext attack

sampling 2η random numbers from $\{0, 1\}$ as $(a_1, ..., a_\eta, b1, ..., b_\eta)$ and output $\sum_{i=1}^{\eta}(a_i, b_i)$ as the random sample [27]. Let v to be a vector in $R = \mathbb{Z}[x]/(x^n + 1)$ whose coefficients are sampled from B_η. We define V as a d-dimensional vector of polynomials $A \in R^d$ that can be generated from binomial distribution B_η^d. Let $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ and $R = \mathbb{Z}[x]/(x^n + 1)$ to be rings of polynomials where $n = 256$ $q = 7681$. Vector a is uniformly sampled in R_q^d. `M-LWE` recovers random secret s with coefficients sampled from B_η^d from $m \geq 1$ samples of the form $(a_i, a_i.s + e_i \bmod q) \leftarrow R_q^d \times R_q$ where e_i is the error with coefficients sampled from B_η. Kyber works over only one ring, $R_q = \mathbb{Z}_{7681}[x]/(x^{256} + 1)$, which provides flexibility (e.g. performing polynomial multiplication) to sacrifice security (from 128-bit to 102-bit) to improve performance and communication size (33%) (by only changing k from 3 to 2). Kyber is released at two security levels as Paranoid ($k = 4$,$\eta = 3$) with post quantum security of 218 bits and Light ($k = 2$,$\eta = 5$) with post quantum security of 102 bits. Public key and ciphertext size of the Light (832 and 736 bytes, respectively) is 2× smaller than those of the Paranoid. This flexibility is exclusive to Kyber (Module-LWE schemes); in Ring-LWE schemes ($d = 1$), changing the security parameters results in building a new ring R_q and ring operations.

`NTRU-KEM` [74], a IND-CCA2-secure key encapsulation mechanism based on NTRU cryptosystem, with 128-bit post-quantum security in the quantum random oracle model is the first timing attack resistant NTRU software thanks to its constant-time noise sampler. NTRU-based KEM has active security which allows parties to cache the ephemeral keys, however passive secure key exchange mechanisms like NewHope and Kyber must not use cached values. Compared to NewHope (255-bit PQ security), NTRU-KEM (123-bit PQ security) improves (secret key size, public key size,cipher text size) by (20%, 37%, 37%) and halves the required clock cycles for encryption/encapsulation step. However, it increases required clock cycles for key generation and decryption/encapsulation by a factor of 3.47.

3.3 Hardware Implementations

3.3.1 Public Key Encryption. Göttert et al. [59] propose the first hardware implementation of Ring-LWE based encryption scheme in which only LWE-polynomial variants are chosen to be implemented on a Xilinx Virtex-7 FPGA, since the area occupied by full Ring-LWE encryption scheme is bigger that the largest FPGA of the Virtex-6 family. Proposed implementations are based on LP lattice-based encryption scheme [82] which achieve 316 times higher throughput compared to software implementation. Using a fully parallel architecture (which makes their design extremely big) they optimize the number of clock cycles for computations of the NTT to generate high throughput. The main optimization metric is performance for which they show speedups for encryption and decryption schemes by factors of 200 and 70, respectively, in comparison to the software implementation with the same level of security.

In [108] authors provide a flexible RLWE encryption engine in which one core is used for key generation, encryption, and decryption. In this article, the main optimization metric is throughput per unit of area. In addition, by applying optimizations including different encoding technique and removing some LSBs of the ciphertext coefficnents, enryption engine, which is 60 times smaller than the design in [59], could be fit on a low cost Xilinx Spartan-6 FPGA; however, the encryption engine is 3 times slower. They employ a Gaussian sampler with relatively low precision, using the CDT sampling method that compares random probabilities with a cumulative distribution table. The proposed Gaussian Sampler is fast (one sampler per cycle at 60 MHz) with the cost of numerous random bits (85) in order to generate a random sample. The Gaussian sampler is time independent with the cost of an array of parallel comparators, one per each word of the table. Sampling process is finished when one of the comparators finds a match.

Roy et al. [116] implement a compact Ring-LWE crypto processor is implemented on Virtex 6 FPGA where they optimize NNT multiplication by reducing the fixed computation and pre-scaling overhead. Authors suggest to combine pre-computation stage and NTT computation. Besides, NNT Memory access is minimized by storing two coefficients in

a single word, processing two pairs of coefficients together, and eliminating idle cycles. Small lookup tables are used in the Knuth-Yao discrete Gaussian sampler [117] which leads to more compact and faster sampler than [107].

The smallest lattice-based encryption engine is implemented on Spartan-6 and Virtex-5 in [107]. Compared with the high speed implementation of [108], this is one order of magnitude slower due to non-applicability of using NTT (using DSP-enabled schoolbook polynomial multiplier). Besides, considerable area is saved by using special modulus, power of 2, by which modular reduction is almost cost free. Further area saving is achieved by using a Bernoulli distribution [49] with small precomputed tables in order to optimized simple rejection sampling by eliminating computing of $exp()$ function.

Howe et al. [71] present the first, and the only, hardware implementation of lattice-based encryption engine based on learning with error problem over standard lattices on the lightweight Spartan-6 FPGA. The main concern in is optimizing area, while the scheme maintains balance between area and performance by using a larger Gaussian sampler. The proposed encryption engine is smaller in comparison to the design of [59]; besides, it could closely compete with the encryption scheme of [108]. To maximize performance, authors use a larger Gaussian sampler, Bernoulli sampler, which generates samples in parallel with no adverse effect on the critical path. Although the area consumed by standard-LWE encryption scheme is greater than (ideal) ring-LWE architectures, no extra security is required to compensate security loss. The main drawback of implantation of standard lattice-based implementation is large key size which results in remarkable increase use of FPGA memory elements.

Reparaz et al. [114] implement a masked ring-LWE scheme on a Virtex-II FPGA and 32-bit ARM Cortex-M4F which is Differential power analysis (DPA) resistant. In order to be resilient to first-order side-channel attacks, a constant time masked decoder with high success probability is implemented. The entire computation is done in the masked domain by employing a dedicated masked decoder which imposes considerable time and area overhead compared with an unprotected design.

3.3.2 Digital Signature. Howe et al. [72] provide evaluation and summary of practical instantiations of digital signature schemes based on lattice problems on different platforms. Evaluation metrics are for secret key, public key, and signature size. Additionally, they give a survey on various implementations of basic blocks including NTT and sampling. Authors evaluate Bernoulli, Ziggurat, Knuth-Yao, and cumulative distribution table (CDT) variants of Gaussian sampler.

An efficient lattice-based signature scheme (GLP signature) is implemented on Xilinx Virtex 6 FPGA in [61] which is the first practical lattice-based signature scheme that could resist transcript collision attacks. Author removes the need for Gaussian noise sampling by using rejection sampling which lead to hiding the secret key contained in each signature. It should be mentioned that hardness assumption is based on Decisional Compact Knapsack problem, which lowers the security of the scheme compared with standard lattice-based signatures. Because of the regular structure of the schoolbook algorithm, authors achieve high speed and small size for implementation of polynomial multiplier. Compared with BLISS, the proposed signature is sub-optimal in terms of signature size and security level.

A high throughput hardware implementation of BLISS [49] is introduced in [106] on Xilinx Spartan-6 FPGA. Authors improve and parallelize the column-wise schoolbook multiplier presented in [61]. An efficient Cumulative Distribution Table (CDT) based Gaussian sampler is employed which uses large tables. To improve performance of the CDT sampler, author deploy an decrease number of comparisons by improving the binary search and reducing size of precomputed large tables by using an optimized floating-point representation (adaptive mantissa size) with negligible effect on performance. Authors provides enhanced CDT which uses two smaller samples (Peikert convolution theorem [102]). A standard CDT needs table of size at least $\eta \times \tau \times \lambda = 215.73 \times 13.4 \times 128 = 370kb$ while enhanced CDT needs around 23×

smaller table. Authors evaluate performance and resource consumption of BLISS-I ($n = 512, q = 12289$) by employing the CDT and two parallel Bernoulli samplers and conclude that CDT consumes less FPGA resources than Bernoulli; besides, enhanced CDT achieves 17.4 Million Operation per Seconds (MOPS) which is 2.3× more than that of Bernoulli sampler. Based on the results, performance of enhanced CDT is almost the same for BLISS-I ($\sigma = 215$), BLISS-III ($\sigma = 250$) and BLISS-IV ($\sigma = 271$).

An optimized and flexible lattice-based digital signature is implemented on Xilinx Spartan-6 and Virtex-6 FPGAs [62] which has the same theoretical basis as [61] but with major improvements. Compared with [61], instead of schoolbook multiplier, authors employ a parallelized NTT for polynomial multiplications (the most time consuming part of the digital signature), which leads to smaller and faster signing/verification engines. Authors develop a flexible processing core with VHDL that could be configured as either signing or/and verification engine which does not impose any overhead to provide flexibility. The signing step is broken into three separate blocks, including lattice processing engine, random oracle, and, sparse multiplication along with compression unit, which are running in parallel. The digital signature processor is adapted from the lattice processor for the public key encryption scheme in [108].

3.3.3 Key Exchange. An area optimized constant time implementation of NewHope-Simple [11], on Xilinx Artix-7 FPGA , with decent performance level is proposed in [127]. With the same post-quantum security level as NewHope (128-bit), server and client work with clock frequency of 125 and 117 MHz, respectively. Security of NewHope-Simple depends on lattice dimension ($n = 1024$), modulus ($q = 12289$) and binomial sampler parameter ($k = 16$). Authors design two separate modules for client and server side which forces an embedded system to be only either a server or a client, hence results in lack of re-usability as a huge disadvantage. For the sake of area optimization, 512 butterfly operations are performed serially, while they can be performed in parallel [10].

3.4 Hardware/Software Codesign

Because of the probabilistic inherent feature of rejection sampling in the lattice-based schemes, the probability of generation of an invalid signature exists. Leveraging precompution, failure possibility of signature generation could be minimized. Aysu et al. divide the signature scheme in hash-based cryptographic signatures into two separate phases in order to minimize the runtime energy and latency of signature generation [15]. During the offline phase, input (message) independent computations, for instance key and random number generation, are performed and results are stored in a memory buffer as coupons. Subsequently, output is generated using the precomputed coupons and the input (message) during the online phase. Employing the same idea, Aysu et al. implement a latency optimized lattice based signature with hardware/software co-design technique [16]. The main objective is to optimize latency which is achieved by focusing on the signature generation step that is done on the embedded device. On the other hand, verification step is performed on high performance platform servers. Signature generation scheme is divided into two separate phases including offline and online phases which are performed on NIOS soft-core as software, and Altera Cyclone-IV FPGA as hardware, respectively. Hardware is responsible for low latency hash function and polynomial multiplication; while, the software part computes and stores polynomials.

3.5 DSP Implementation

In order to perform the multiply-accumulate (MAC) operations of matrices in the encryption scheme, authors utilize a dedicated DSP48A1 unit of the Spartan-6 FPGA to achieve an area optimized hardware implementation of standard

LWE based encryption engine [71]. The main goal is optimizing area, while the scheme maintains balance between area and performance by using a larger Gaussian sampler.

4 CONCLUSION

Lattice-based cryptographic algorithms and protocols promise to tackle the challenges posed by deployment across diverse computing platforms, as well as for diverse use cases within reasonable security, performance and energy efficiency guarantees.

Numerous schemes and implementations tackle different trade-offs, such as memory footprint, security, performance, and energy, are mapped on a variety of platforms and are applicable to specific use cases. However, current designs are still deficient in addressing the need for agility, which is paramount to tackle the needs of emerging business models at the computing platform level. In addition, securing such platforms against physical attacks is a topic that needs to be researched.

In this manuscript, we provided a review of lattice-based cryptography, some of the proposals for lattices in computer security, their implementations in software and hardware, and their applications to key exchange and digital signatures.

5 ACKNOWLEDGEMENT

This work was supported in part with a gift from Qualcomm Research.

REFERENCES

[1] 1997. Public-Key Cryptosystems from Lattice Reduction Problems. In *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '97)*.

[2] 2012. Pseudorandom Functions and Lattices. In *Proceedings of the Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT '12)*.

[3] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrede Lepoint. 2016. NFLlib: NTT-based Fast Lattice Library. In *Proceedings of the RSA Conference on The Cryptographers' Track (CT-RSA '16)*.

[4] Miklós Ajtai. 1996. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC '96)*.

[5] Miklós Ajtai, Ravi Kumar, and Dandapani Sivakumar. 2001. A Sieve Algorithm for the Shortest Lattice Vector Problem. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC '01)*.

[6] Sedat Akleylek, Erdem Alkım, and Zaliha Yüce Tok. 2016. Sparse Polynomial Multiplication for Lattice-Based Cryptography with Small Complexity. *The Journal of Supercomputing* (2016).

[7] Sedat Akleylek, Nina Bindel, Johannes Buchmann, Juliane Krämer, and Giorgia Azzurra Marson. 2016. An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation. In *Proceedings of the International Conference on Cryptology in Africa (AFRICACRYPT '16)*.

[8] Sedat Akleylek, Özgur Dağdelen, and Zaliha Yüce Tok. 2015. On the Efficiency of Polynomial Multiplication for Lattice-Based Cryptography on GPUs Using CUDA. In *Proceedings of the International Conference on Cryptography and Information Security in the Balkans*.

[9] Erdem Alkim, Nina Bindel, Johannes Buchmann, Özgür Dagdelen, Edward Eaton, Gus Gutoski, Juliane Krämer, and Filip Pawlega. 2015. Revisiting TESLA in The Quantum Random Oracle Model. Cryptology ePrint Archive, Report 2015/755. (2015).

[10] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2015. Post-Quantum Key Exchange - a New Hope. Cryptology ePrint Archive, Report 2015/1092. (2015).

[11] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. NewHope Without Reconciliation. Cryptology ePrint Archive, Report 2016/1157. (2016).

[12] Erdem Alkim, Philipp Jakubeit, and Peter Schwabe. 2016. NewHope on ARM Cortex-M. In *Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering (SPACE '16)*.

[13] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. 2009. Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '09)*.

[14] Aydin Aysu, Cameron Patterson, and Patrick Schaumont. 2013. Low-Cost and Area-Efficient FPGA Implementations of Lattice-Based Cryptography. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST '13)*.

[15] A. Aysu and P. Schaumont. 2016. Precomputation Methods for Hash-Based Signatures on Energy-Harvesting Platforms. *IEEE Trans. Comput.* (2016).

[16] Aydin Aysu, Bilgiday Yuce, and Patrick Schaumont. 2015. The Future of Real-Time Security: Latency-Optimized Lattice-Based Digital Signatures. *ACM Transactions on Embedded Computing Systems* (2015).

[17] Shi Bai and Steven D Galbraith. 2014. An Improved Compression Technique for Signatures Based on Learning with Errors. In *Proceedings of topics in Cryptology (CT-RSA '14)*.

[18] Paulo S. L. M. Barreto, Patrick Longa, Michael Naehrig, Jefferson E. Ricardini, and Gustavo Zanon. 2016. Sharper Ring-LWE Signatures. Cryptology ePrint Archive, Report 2016/1026. (2016).

[19] Paul Barrett. 1986. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '86)*.

[20] Daniel J Bernstein. 2008. ChaCha, a variant of Salsa20. In *Workshop Record of SASC 2008: The State of the Art of Stream Ciphers*.

[21] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. 2016. NTRU Prime: Reducing Attack Surface at Low Cost. Cryptology ePrint Archive, Report 2016/461. (2016).

[22] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. 2013. (2013).

[23] Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Rietman, and Ludo Tolhuizen. 2017. spKEX: An Optimized Lattice-Based Key Exchange. Cryptology ePrint Archive, Report 2017/709. (2017).

[24] Ahmad Boorghany and Rasool Jalili. 2014. Implementation and Comparison of Lattice-based Identification Protocols on Smart Cards and Microcontrollers. Cryptology ePrint Archive, Report 2014/078. (2014).

[25] Ahmad Boorghany, Siavash Bayat Sarmadi, and Rasool Jalili. 2015. On Constrained Implementation of Lattice-Based Cryptographic Primitives and Schemes on Smart Cards. *ACM Transactions on Embedded Computing Systems* (2015).

[26] Joppe Bos, Craig Costello, Leo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. 2016. Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*.

[27] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. 2017. CRYSTALS – Kyber: a CCA-secure Module-Lattice-Based KEM. Cryptology ePrint Archive, Report 2017/634. (2017).

[28] Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas Stebila. 2015. Post-Quantum Key Exchange for the TLS Protocol from the Ring Learning with Errors Problem. In *Proceedings of the IEEE Symposium on Security and Privacy (SP '15)*.

[29] Matt Braithwaite. 2016. Experimenting with post-quantum cryptography. (2016). https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html

[30] 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 *Proceedings of the International Conference on Selected Areas in Cryptography (SAC '13)*.

[31] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. [n. d.]. Merkle Signatures with Virtually Unlimited Signature Capacity. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS '07)*.

[32] Johannes Buchmann, Florian Göpfert, Tim Güneysu, Tobias Oder, and Thomas Pöppelmann. 2016. High-Performance and Lightweight Lattice-Based Public-Key Encryption. In *Proceedings of the ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '16)*.

[33] Donald Donglong Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray CC Cheung, Derek Pao, and Ingrid Verbauwhede. 2015. High-Speed Polynomial Multiplication Architecture for Ring-LWE and SHE Cryptosystems. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2015).

[34] Donald Donglong Chen, Gavin Xiaoxu Yao, Ray CC Cheung, Derek Pao, and Cetin Kaya Koç. 2016. Parameter Space for the Architecture of FFT-Based Montgomery Modular Multiplication. *IEEE Trans. Comput.* (2016).

[35] Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. 2016. Lizard: Cut off the Tail! Practical Post-Quantum Public-Key Encryption from LWE and LWR. Cryptology ePrint Archive, Report 2016/1126. (2016).

[36] Arjun Chopra. 2016. Improved Parameters for the Ring-TESLA Digital Signature Scheme. Cryptology ePrint Archive, Report 2016/1099. (2016).

[37] Arjun Chopra. 2017. GLYPH: A New Insantiation of the GLP Digital Signature Scheme. Cryptology ePrint Archive, Report 2017/766. (2017).

[38] Paul G. Comba. 1990. Exponentiation cryptosystems on the IBM PC. *IBM systems journal* (1990).

[39] Özgür Dagdelen, Rachid El Bansarkhani, Florian Göpfert, Tim Güneysu, Tobias Oder, Thomas Pöppelmann, Ana Helena Sánchez, and Peter Schwabe. 2014. High-Speed Signatures from Standard Lattices. In *Proceedings of the International Conference on Cryptology and Information Security in Latin America (LATINCRYPT '14)*.

[40] Jean Pierre David, Kassem Kalach, and Nicolas Tittley. 2007. Hardware Complexity of Modular Multiplication and Exponentiation. *IEEE Trans. Comput.* (2007).

[41] Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2015. Efficient Software Implementation of Ring-LWE Encryption. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE '15)*.

[42] C. Du and G. Ba. 2016. High-Performance Software Implementation of Discrete Gaussian Sampling for Lattice-Based Cryptography. In *Proceedings of the IEEE Information Technology, Networking, Electronic and Automation Control Conference*.

[43] Chaohui Du and Guoqiang Bai. 2015. Towards Efficient Discrete Gaussian Sampling for Lattice-Based Cryptography. In *Proceeding of the International Conference on Field Programmable Logic and Applications (FPL '15)*.

[44] Chaohui Du and Guoqiang Bai. 2016. Efficient polynomial multiplier architecture for Ring-LWE based public key cryptosystems. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS' 16)*.

[45] Chaohui Du and Guoqiang Bai. 2016. A Family of Scalable Polynomial Multiplier Architectures for Ring-LWE Based Cryptosystems. (2016).

[46] Chaohui Du and Guoqiang Bai. 2016. Towards Efficient Polynomial Multiplication for Battice-Based Cryptography. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS' 16)*.

[47] Chaohui Du, Guoqiang Bai, and Xingjun Wu. 2016. High-Speed Polynomial Multiplier Architecture for Ring-LWE Based Public Key Cryptosystems. In *Proceedings of the International Great Lakes Symposium on VLSI (GLSVLSI '16)*.

[48] Léo Ducas. 2014. Accelerating Bliss: The Geometry of Ternary Polynomials. Cryptology ePrint Archive, Report 2014/874. (2014).

[49] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. 2013. Lattice Signatures and Bimodal Gaussians. In *Proceedings of the Annual International Cryptology Conference on Advances in Cryptology*.

[50] Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2017. CRYSTALS – Dilithium: Digital Signatures from Module Lattices. Cryptology ePrint Archive, Report 2017/633. (2017).

[51] Léo Ducas and Phong Q. Nguyen. 2012. Faster Gaussian Lattice Sampling Using Lazy Floating-point Arithmetic. In *Proceedings of the International Conference on The Theory and Application of Cryptology and Information Security (ASIACRYPT '12)*.

[52] Nagarjun C. Dwarakanath and Steven D. Galbraith. 2014. Sampling from Discrete Gaussians for Lattice-based Cryptography on a Constrained Device. *Applicable Algebra in Engineering, Communication and Computing* (2014).

[53] Rachid El Bansarkhani and Johannes Buchmann. 2013. Improvement and Efficient Implementation of a Lattice-Based Signature Scheme. In *Proceedings of the International Conference on Selected Areas in Cryptography (SAC '13)*.

[54] Pavel Emeliyanenko. 2009. Efficient Multiplication of Polynomials on Graphics Hardware. In *Proceedings of the International Symposium on Advanced Parallel Processing Technologies (APPT '09)*.

[55] Scott Fluhrer. 2016. Cryptanalysis of Ring-LWE Based Key Exchange with Key Share Reuse. Cryptology ePrint Archive, Report 2016/085. (2016).

[56] János Folláth. 2014. Gaussian Sampling in Lattice Based Cryptography. *Tatra Mountains Mathematical Publications* (2014).

[57] Oscar Garcia-Morchon, Ronald Rietman, Sahil Sharma, Ludo Tolhuizen, and Jose Luis Torre-Arce. [n. d.]. DTLS-HIMMO: Achieving DTLS Certificate Security with Symmetric Key Overhead. In *Proceedings of the European Symposium on Computer Security*.

[58] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. 2008. Trapdoors for Hard Lattices and New Cryptographic Constructions. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC '08)*.

[59] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin Huss. 2012. On the Design of Hardware Building Blocks for Modern Lattice-based Encryption Schemes. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES '12)*.

[60] Shay Gueron and Fabian Schlieker. 2016. Speeding up R-LWE Post-Quantum Key Exchange. Cryptology ePrint Archive, Report 2016/467. (2016).

[61] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. 2012. Practical Lattice-based Cryptography: A Signature Scheme for Embedded Systems. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES '12)*.

[62] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. 2015. Lattice-Sased Signatures: Optimization and Implementation on Reconfigurable Hardware. *IEEE Trans. Comput.* (2015).

[63] Tim Güneysu, Tobias Oder, Thomas Pöppelmann, and Peter Schwabe. 2013. Software Speed Records for Lattice-Based Signatures. In *Proceedings of the International Workshop on Post-Quantum Cryptography (PQCrypto '13)*.

[64] Tamás Györfi, Octavian Cret, and Zalán Borsos. 2013. Implementing Modular FFTs in FPGAs – A Basic Block for Lattice-Based Cryptography. In *Proceedings of the Euromicro Conference on Digital System Design (DSD '13)*.

[65] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. 2003. NTRUSign: Digital Signatures Using the NTRU Lattice. In *Proceedings of the RSA Conference on The Cryptographers' Track (CT-RSA '03)*.

[66] Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. 2015. Choosing Parameters for NTRUEncrypt. Cryptology ePrint Archive, Report 2015/708. (2015).

[67] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. 1998. NTRU: A Ring-Based Public Key Cryptosystem. In *Proceedings of the International Symposium on Algorithmic Number Theory (ANTS-III)*.

[68] Jeffrey Hoffstein, Jill Pipher, William Whyte, and Zhenfei Zhang. 2017. A signature scheme from Learning with Truncation. Cryptology ePrint Archive, Report 2017/995. (2017).

[69] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. 2017. A Modular Analysis of the Fujisaki-Okamoto Transformation. Cryptology ePrint Archive, Report 2017/604. (2017).

[70] J. Howe, A. Khalid, C. Rafferty, F. Regazzoni, and M. O'Neill. 2016. On Practical Discrete Gaussian Samplers For Lattice-Based Cryptography. *IEEE Trans. Comput.* (2016).

[71] J. Howe, C. Moore, M. O'Neill, F. Regazzoni, T. Güneysu, and K. Beeden. 2016. Lattice-based Encryption Over Standard Lattices In Hardware. In *Proceedings of the Annual Design Automation Conference (DAC '16)*.

[72] James Howe, Thomas Pöppelmann, Máire O'neill, Elizabeth O'sullivan, and Tim Güneysu. 2015. Practical Lattice-Based Digital Signature Schemes. *ACM Transactions on Embedded Computing Systems* (2015).

[73] J. Howe, C. Rafferty, A. Khalid, and M. O'Neill. 2017. Compact and Provably Secure Lattice-Based Signatures in Hardware. (2017).

[74] Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe. [n. d.]. High-Speed Key Encapsulation from NTRU. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES '17)*.

[75] Zhengzhong Jin and Yunlei Zhao. 2017. Optimal Key Consensus in Presence of Noise. Cryptology ePrint Archive, Report 2017/1058. (2017).

[76] Xiaodong Lin Jintai Ding, Xiang Xie. 2012. A Simple Provably Secure Key Exchange Scheme Based on the Learning with Errors Problem. Cryptology ePrint Archive, Report 2012/688. (2012).

[77] Anatolii Karatsuba and Yu Ofman. 1963. Multiplication of Many-Digital Numbers by Automatic Computers. In *Proceedings of the USSR Academy of Sciences*.

[78] A Khalid, J Howe, C Rafferty, and M O'Neill. 2016. Time-Independent Discrete Gaussian Sampling For Post-Quantum Cryptography. In *Proceedings of the International Conference on Field-Programmable Technology (FPT '16)*.

[79] Donald E. Knuth. 1997. *The Art of Computer Programming, Volume 2 (3rd Ed.): Seminumerical Algorithms*.

[80] Donald E Knuth and Andrew C Yao. 1976. The Complexity of Nonuniform Random Number Generation. *Algorithms and complexity: new directions and recent results* (1976).

[81] Adeline Langlois and Damien Stehle. 2012. Worst-Case to Average-Case Reductions for Module Lattices. Cryptology ePrint Archive, Report 2012/090. (2012).

[82] Richard Lindner and Chris Peikert. 2011. Better Key Sizes (and Attacks) for LWE-based Encryption. In *Proceedings of the International Conference on Topics in Cryptology (CT-RSA'11)*.

[83] Zhe Liu, Hwajeong Seo, Sujoy Sinha Roy, Johann Großschädl, Howon Kim, and Ingrid Verbauwhede. 2015. Efficient Ring-LWE Encryption on 8-bit AVR Processors. (2015).

[84] Gui-Lu Long. 2001. Grover Algorithm with Zero Theoretical Failure Rate. *Physical Review A* (2001).

[85] Patrick Longa and Michael Naehrig. 2016. Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography. Cryptology ePrint Archive, Report 2016/504. (2016).

[86] Patrick Longa and Michael Naehrig. 2016. Speeding up the Number Theoretic Transform for Faster Ideal Lattice-Based Cryptography. In *Proceedings of the International Conference on Cryptology and Network Security (CANS '16)*.

[87] Vadim Lyubashevsky. 2009. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '09)*.

[88] Vadim Lyubashevsky. 2012. Lattice Signatures Without Trapdoors. In *Proceedings of the Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT '12)*.

[89] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. 2010. On Ideal Lattices and Learning with Errors over Rings. In *Proceedings of the Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'10)*.

[90] George Marsaglia, Wai Wan Tsang, et al. 2000. The Ziggurat Method for Generating Random Variables. *Journal of statistical software* (2000).

[91] Daniele Micciancio. 2010. *Cryptographic Functions from Worst-Case Complexity Assumptions*.

[92] Daniele Micciancio and Chris Peikert. 2012. Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In *Proceedings of the Annual International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT '12)*.

[93] Daniele Micciancio and Oded Regev. 2007. Worst-Case to Average-Case Reductions Based on Gaussian Measures. *SIAM J. Comput.* (2007).

[94] Daniele Micciancio and Oded Regev. 2009. *Lattice-based cryptography*.

[95] Daniele Micciancio and Michael Walter. 2017. Gaussian Sampling over the Integers: Efficient, Generic, Constant-Time. Cryptology ePrint Archive, Report 2017/259. (2017).

[96] Peter L Montgomery. 1985. Modular Multiplication Without Trial Division. *Mathematics of computation* (1985).

[97] Shruti More and Raj Katti. 2015. Discrete Gaussian Sampling for Low-Power Devices. In *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '15)*.

[98] Hamid Nejatollahi, Nikil Dutt, and Rosario Cammarota. 2017. Trends, Challenges and Needs for Lattice-based Cryptography Implementations: Special Session. In *Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis Companion (CODES '17)*.

[99] Henri Nussbaumer. 1980. Fast Polynomial Transform Algorithms for Digital Convolution. *IEEE Transactions on Acoustics, Speech, and Signal Processing* (1980).

[100] Tobias Oder, Tim Güneysu, Felipe Valencia, Ayesha Khalid, Maire O'Neill, and Francesco Regazzoni. 2016. Lattice-Based Cryptography: From Reconfigurable Hardware to ASIC. In *Proceedings of the International Symposium on Integrated Circuits (ISIC '16)*.

[101] Tobias Oder, Thomas Pöppelmann, and Tim Güneysu. 2014. Beyond ECDSA and RSA: Lattice-based digital signatures on constrained devices. In *Proceedings of the Annual Design Automation Conference (DAC '14)*.

[102] Chris Peikert. 2010. An Efficient and Parallel Gaussian Sampler for Lattices. In *Proceedings of the Annual Conference on Advances in Cryptology (CRYPTO'10)*.

[103] Chris Peikert. 2014. Lattice Cryptography for the Internet. In *Proceedings of the International Workshop on Post-Quantum Cryptography (PQCrypto '14)*.

[104] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. 2008. A Framework for Efficient and Composable Oblivious Transfer. In *Proceedings of the Annual Conference on Cryptology: Advances in Cryptology (CRYPTO '08)*.

[105] Thomas Pöppelmann. 2016. *Efficient Implementation of Ideal Lattice-Based Cryptography*. Ruhr-Universität Bochum.

[106] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. 2014. Enhanced Lattice-Based Signatures on Reconfigurable Hardware. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES '14)*.

[107] Thomas Pöppelmann and Tim Güneysu. [n. d.]. Area Optimization of Lightweight Lattice-Based Encryption on Reconfigurable Hardware. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '14)*.

[108] Thomas Pöppelmann and Tim Güneysu. [n. d.]. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. In *Proceedings of the Revised Selected Papers on Selected Areas in Cryptography (SAC '13)*.

[109] Thomas Pöppelmann and Tim Güneysu. 2012. Towards Efficient Arithmetic for Lattice-Based Cryptography on Reconfigurable Hardware. In *Proceedings of the International Conference on Cryptology and Information Security in Latin America (LATINCRYPT '12)*.

[110] Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. 2015. High-Performance Ideal Lattice-Based Cryptography on 8-Bit ATxmega Microcontrollers. In *Proceedings of the International Conference on Cryptology and Information Security in Latin America (LATINCRYPT '15)*.

[111] Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. 2015. High-Performance Ideal Lattice-Based Cryptography on 8-bit ATxmega Microcontrollers. Cryptology ePrint Archive, Report 2015/382. (2015).

[112] Ciara Rafferty, Maire O'Neill, and Neil Hanley. 2017. Evaluation of Large Integer Multiplication Methods on Hardware. *IEEE Trans. Comput.* (2017).

[113] Oded Regev. 2005. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. (2005).

[114] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. 2016. Masking Ring-LWE. *Journal of Cryptographic Engineering* (2016).

[115] Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. 2014. Compact and Side Channel Secure Discrete Gaussian Sampling. Cryptology ePrint Archive, Report 2014/591. (2014).

[116] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. 2014. Compact Ring-LWE Cryptoprocessor. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems (CHES'14)*.

[117] Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2013. High Precision Discrete Gaussian Sampling on FPGAs. In *Proceedings of the International Conference on Selected Areas in Cryptography (SAC '13)*.

[118] Markku-Juhani O. Saarinen. 2015. Gaussian Sampling Precision in Lattice Cryptography. Cryptology ePrint Archive, Report 2015/953. (2015).

[119] Markku-Juhani O Saarinen. 2017. Arithmetic Coding and Blinding Countermeasures for Lattice Signatures. *Journal of Cryptographic Engineering* (2017).

[120] Markku-Juhani O. Saarinen. 2017. HILA5: On Reliability, Reconciliation, and Error Correction for Ring-LWE Encryption. Cryptology ePrint Archive, Report 2017/424. (2017).

[121] Markku-Juhani Olavi Saarinen. 2017. Ring-LWE Ciphertext Compression and Error Correction: Tools for Lightweight Post-Quantum Cryptography. In *Proceedings of the ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS '17)*.

[122] Arnold Schönhage and Volker Strassen. 1971. Schnelle Multiplikation Grosser Zahlen. *Computing* (1971).

[123] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* (1997).

[124] Douglas Stebila and Michele Mosca. 2016. Post-Quantum Key Exchange for the Internet and the Open Quantum Safe Project. Cryptology ePrint Archive, Report 2016/1017. (2016).

[125] Silvan Streit and Fabrizio De Santis. 2017. Post-Quantum Key Exchange on ARMv8-A – A New Hope for NEON made Simple. Cryptology ePrint Archive, Report 2017/388. (2017).

[126] David B Thomas, Wayne Luk, Philip HW Leong, and John D Villasenor. 2007. Gaussian Random Number Generators. *Comput. Surveys* (2007).

[127] Oder Tobias and Güneysu Tim. 2017. Implementing the NewHope-Simple Key Exchange on Low-Cost FPGAs. In *Proceedings of the International Conference on Cryptology and Information Security in Latin America (LATINCRYPT '17)*.

[128] John Von Neumann. 1951. Various Techniques Used in Connection With Random Digits. *National Bureau of Standards Applied Mathematics Series* (1951).

[129] Franz Winkler. 1996. *Polynomial Algorithms in Computer Algebra*. Springer.

[130] Ye Yuan, Chen-Mou Cheng, Shinsaku Kiyomoto, Yutaka Miyake, and Tsuyoshi Takagi. 2016. Portable Implementation of Lattice-Based Cryptography Using JavaScript. *Proceedings of the International Symposium on Computing and Networking*.