Lattice-based Key-sharing Schemes: A Survey

PRASANNA RAVI, Nanyang Technological University, Singapore
JAMES HOWE, PQShield, UK
ANUPAM CHATTOPADHYAY and SHIVAM BHASIN, Nanyang Technological University, Singapore

Public-key cryptography is an indispensable component used in almost all of our present-day digital infrastructure. However, most if not all of it is predominantly built upon hardness guarantees of number theoretic problems that can be broken by large-scale quantum computers in the future. Sensing the imminent threat from continued advances in quantum computing, NIST has recently initiated a global-level standardization process for quantum resistant public-key cryptographic primitives such as public-key encryption, digital signatures, and key encapsulation mechanisms. While the process received proposals from various categories of post-quantum cryptography, lattice-based cryptography features most prominently among all the submissions. Lattice-based cryptography offers a very attractive alternative to traditional public-key cryptography mainly due to the variety of lattice-based schemes offering varying flavors of security and efficiency guarantees. In this article, we survey the evolution of lattice-based key-sharing schemes (public-key encryption and key encapsulation schemes) and cover various aspects ranging from theoretical security guarantees, general algorithmic frameworks, practical implementation aspects, and physical attack security, with special focus on lattice-based key-sharing schemes competing in the NIST's standardization process.

CCS Concepts: • **Security and privacy → Public key (asymmetric) techniques**;

Additional Key Words and Phrases: Lattice-based cryptography, public-key encryption schemes, key encapsulation mechanisms, key-exchange schemes

ACM Reference format:
Prasanna Ravi, James Howe, Anupam Chattopadhyay, and Shivam Bhasin. 2020. Lattice-based Key-sharing Schemes: A Survey. *ACM Comput. Surv.* 54, 1, Article 9 (December 2020), 39 pages.
https://doi.org/10.1145/3422178

1 INTRODUCTION

Public/Asymmetric key cryptography is one of the most important components of today's digital infrastructure. Most secure communication protocols and applications such as TLS, IPSec, and DNSSEC heavily rely on the following public-key cryptographic primitives: Public-key Encryption (PKE), Digital Signatures (DS), Key Exchange (KEX), and Key Encapsulation Mechanisms (KEM)

We gratefully acknowledge partial support of DSO Singapore for this work under Grant No. DSOCL19218.
Authors' addresses: P. Ravi, A. Chattopadhyay, and S. Bhasin, 21 Nanyang Link, 04-01, Temasek Labs and School of Computer Science and Engineering, Nanyang Technological University, Singapore 637371; emails: {PRASANNA.RAVI, anupam, sbhasin}@ntu.edu.sg; J. Howe, 267 (Prama House) Banbury Road, Summertown, Oxford, OX2 7HT, PQShield, UK; email: jameshoweee@gmail.com.

0360-0300/2020/12-ART9 $15.00
https://doi.org/10.1145/3422178

Table 1. NIST Security Classification for Post-quantum Cryptographic Schemes

NIST security level	Equivalent symmetric-key security
Category 1	Key search on a block cipher with 128-bit key (AES-128)
Category 2	Collision search on a 256-bit hash function (SHA256/SHA3-256)
Category 3	Key search on a block cipher with 192-bit key (AES-192)
Category 4	Collision search on a 384-bit hash function (SHA384/SHA3-384)
Category 5	Key search on a block cipher with 256-bit key (AES-256)

based on either Rivest-Shamir-Adleman (RSA) or Elliptic Curve Cryptography (ECC). These classical public-key cryptographic schemes are built upon hard number theoretic assumptions, such as integer factorization and discrete logarithms, which are considered computationally intractable by classical computers. However, Peter Shor in 1994 devised a "quantum" algorithm that can solve the integer factorization problem in polynomial time, provided there is a sufficiently large-scale quantum computer [154]. This sent shockwaves through the cryptographic community and effectively rendered all public-key cryptographic systems in our digital infrastructure insecure in a probable future with large-scale quantum computers. Moreover, continuous advancements in the field of quantum computing toward realizing large-scale quantum computers [119] have prompted the cryptographic community toward developing quantum attack resistant public-key cryptographic primitives leading to a new area of cryptographic research called *post-quantum cryptography* or *quantum-safe cryptography*.

Due to the looming threat of quantum computers, various government agencies, companies, and standards agencies are looking to migrate toward post-quantum algorithms [47, 52]. As a first significant step toward adopting *post-quantum cryptography*, NIST called for proposals for standardization of post-quantum cryptographic schemes and, in particular, PKE, DS, and KEM [124]. They had defined five levels of security to categorize the post-quantum secure schemes, based on the computational resources required to break an equivalent NIST standardized symmetric cryptographic primitive (refer to Table 1). The first round started in December 2017 with 69 submissions (49 PKE/KEMs and 40 DS schemes) and after intense scrutiny and public feedback from the cryptographic community, NIST selected about 26 algorithms (17 PKE/KEMs and 9 DS schemes), which can be broadly classified into five main categories: (1) Lattice-based, (2) Code-based, (3) Hash-based, (4) Multivariate Quadratic-based, and (5) Supersingular Isogeny-based cryptography. The main selection criteria in the first round were theoretical post-quantum security guarantees and uniqueness of the scheme. However, the second round will additionally focus on implementation aspects such as speed, efficiency, bandwidth requirement along with protection against physical attacks. Among the second round candidates, lattice-based cryptographic schemes form the largest category with nine PKE/KEMs and three DS schemes.

Lattice-based cryptographic schemes form a very attractive alternative to current public-key cryptographic schemes owing to a good balance of efficiency and security guarantees, with the performance of some of the schemes being on par with or sometimes even better than the current public-key cryptographic schemes. There exists a long line of ongoing work related to various aspects of lattice-based cryptography such as development of practical schemes [10, 33, 35, 79], theoretical cryptanalysis [55, 101], practical implementations on a variety of implementation platforms [122], physical attacks [138, 139], misuse attacks [69], and even early adoption and prototype testing of lattice-based schemes in real world secure protocols and applications [40, 105, 106]. Thus, lattice-based cryptographic schemes are well understood from a variety of aspects right from theory until practice. Howe et al. [84] proposed a comprehensive survey of lattice-based

digital signature schemes in 2015 and Nejatollahi et al. [122] in 2018 reported a survey solely focussing on the implementation aspects of lattice-based schemes. However, in this article, we focus on lattice-based key-sharing schemes and present a comprehensive and up-to-date survey, which include several candidates currently competing in round 2 of the NIST standardization process. Though each of these schemes can also be used for both message encryption and sharing keys, we will denote them together as key-sharing schemes for simplicity. We cover multiple aspects of key-sharing schemes ranging from theoretical security, practical implementation considerations, physical attacks and a comparative evaluation of performance of the schemes based on the state-of-the-art results on various implementation platforms.

Our survey on lattice-based key-sharing schemes is organized as follows. Section 2 contains background details about hard problems governing lattice-based PKE/KEX/KEMs. Section 3 provides a detailed overview and evolution of various lattice-based schemes reported in literature with separate focus on the second round NIST candidates. Section 4 reviews and surveys the various techniques known to efficiently implement the important sub-blocks of lattice-based schemes such as the polynomial/matrix multiplier and discrete distribution sampler. Section 5 provides a detailed survey of physical attacks (side-channel and fault injection) and associated countermeasures for lattice-based schemes. Section 6 performs a comparative evaluation of the second round lattice-based NIST candidates based on the state-of-the art implementation results on a number of hardware and software platforms. Finally, Section 7 provides a high level overview and comparison of lattice-based schemes against other post-quantum schemes and concludes the article.

2 BACKGROUND

2.1 Notations and Preliminaries

We establish some notations that we will use throughout the article. Let $\mathbb{Z}_q$ denote the residue class ring of integers modulo q and $\mathbf{R}_{q,\phi(x)} = \mathbb{Z}_q[x]/\phi(x)$ denote the polynomial ring, with $\phi(x)$ being the modular polynomial (for simplicitly, sometimes also denoted as $\mathbf{R}_q$). Any letter in capital and bold ($\mathbf{A}$) denotes vector/matrix with elements in $\mathbb{Z}_q$, while letters in small case and bold ($\mathbf{a}$) denote polynomials or matrices/vectors of polynomials. Single elements in $\mathbb{Z}_q$ are denoted in small case. The operation $\lfloor a \rceil_{q->p} = \lfloor (p/q) \cdot a \rceil$ denotes rounding an element a from a modulus q to p. The operation $||\cdot||_\infty$ denotes the l_∞-norm. Conventional multiplication is denoted using $(\cdot)$ or $(\times)$ and point-wise multiplication is denoted using $(*)$. Sampling an element $x = (x_1, x_2, \ldots, x_{k-1})$ in a multi-dimensional space $\mathcal{K}$ from a distribution $\mathcal{D}_\sigma$ with standard deviation σ is denoted as $x \leftarrow \mathcal{D}_\sigma(\mathcal{K})$ with each element $\mathbf{x}_i$ is sampled from $\mathcal{D}_\sigma$. Uniform distribution is denoted as $\mathcal{U}$. The set of all prime numbers is denoted as $\mathcal{P}$ and a prime number x is denoted as $x \in \mathcal{P}$. We denote the space of integers formed using n bits or $m = n/8$ bytes as $\mathcal{B}^m$ or $\{0,1\}^n$.

2.2 Lattice Preliminaries

A lattice is a geometric structure periodic in n-dimensional space that can be defined using a set of k linearly independent vectors $\mathbf{B} := (v_1, v_2, \ldots, v_k) \in \mathbb{R}^n$ together known as a *basis* of the lattice. The lattice generated by a basis $\mathbf{B}$ with rank k is the set of vectors

$$\mathcal{L}(\mathbf{B}) = \mathcal{L}(v_1, v_2, \ldots, v_k) := \left\{ \sum_{i=1}^{i=k} \alpha_i v_i \middle| \alpha_i \in \mathbb{Z} \right\}.$$

Any lattice with dimension greater than 2 can have infinitely many bases. Two bases $\mathbf{B}_1$ and $\mathbf{B}_2$ generate the same lattice $\mathcal{L}$ if $\mathbf{B}_1 = \mathbf{U}\mathbf{B}_2$ where $\mathbf{U}$ is a unimodular matrix with $|det(\mathbf{U})| = \pm 1$. The fundamental domain of a lattice $\mathcal{L}$ is the *fundamental parallelepiped* $\mathcal{P}(\mathbf{B}) = \mathbf{B} \cdot [-\frac{1}{2}, \frac{1}{2})^n$ centered around the origin. Cosets of the *fundamental parallelepiped* are disjoint and cover the

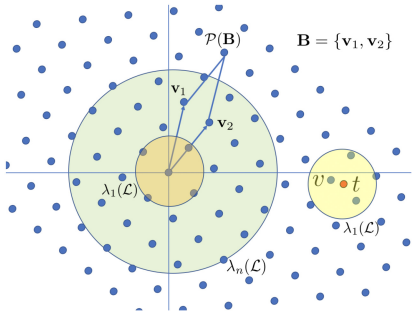

Fig. 1. Pictorial representation of a 2D lattice.

entire lattice. The length of the shortest non zero vector is defined as the minimum distance $\lambda_1(\mathcal{L}) := (min_{v \in \mathcal{L}/\{0\}}||v||)$. Extending the same idea, $\lambda_i(\mathcal{L})$ denotes the ith successive minimum and the radius of the smallest origin-centered circle $d > 0$ consisting of i linearly independent vectors v such that $max_{v \in \mathcal{L}/\{0\}}||v|| < d$. We informally define some worst-case hard problems over this lattice (Refer Figure 1 for pictorial illustration).

- Exact Shortest Vector Problem (Exact SVP): For a given lattice $\mathcal{L}(\mathbf{B})$, find the shortest vector $v \in \mathcal{L}$ such that $||v|| = \lambda_1(\mathcal{L})$.
- Approximate SVP (ASVP$_\gamma$): For an approximation factor γ, find a solution $v \in \mathcal{L}$ such that $||v|| \leq \gamma(n) \cdot \lambda_1(\mathcal{L})$.
- GapSVP$_\gamma$: For a given lattice $\mathcal{L}$, decide whether $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma(n)$.
- Shortest Independent Vectors problem (SIVP$_\gamma$): Find n linearly independent vectors $(v_1, \ldots, v_n)$ such that $max_{v \in \mathcal{L}/\{0\}}||v_i|| \leq \gamma(n)\lambda_n(\mathcal{L})$.
- Bounded Distance Decoding Problem (BDD$_\gamma$): Find $v \in \mathcal{L}$ for a given target point $t \in \mathbb{R}_n$ and an approximation factor $\gamma(n)$ such that $||t - v|| < \gamma(n) \cdot \lambda_1(\mathcal{L})$.

The best-known algorithms to solve the above-mentioned problems only yield sub-exponential approximation factors ($\gamma = 2^{\theta(n\, log\, log(n)/log(n))}$) in polynomial time [110, 152], while algorithms that can solve for *poly*(n) approximation factors require exponential time or space ($2^{\theta(n)}$) [3]. Cryptographic schemes built upon these problems construct their parameter sets, which correspond to polynomial approximation factors of the aforementioned problems, for which there are presently no known quantum/classical algorithm that can solve them in polynomial time.

2.3 Foundations of Practical Lattice-based Cryptography

Practical cryptographic schemes can only be built upon average-case hard problems, but the aforementioned problems are only provably hard in the worst case. Ajtai [4] in 1996 proposed the *Short Integer Solutions* (SIS) problem and proved that its average case instance is provably as hard as the worst case instances of certain lattice problems. Concurrently, Hoffstein, Pipher, and Silverman [79] proposed the *Nth order Truncated Polynomial Ring Unit* (NTRU) public-key encryption scheme (NTRUEncrypt), which is built upon hard problems over algebraically structured lattices better known as "NTRU" lattices, but does not have any such provable security guarantees. Similar to the SIS problem, Regev [140] in 2005 proposed the *Learning with Errors* (LWE) problem, which also enjoys a worst-case to average-case hardness reduction to related lattice-problems. Over the years, several variants of the LWE and the NTRU problem have been developed and all lattice-based PKE/KEM/KEXs in the NIST process are based on either of these two problems.

2.3.1 The Learning with Errors Problem [140]. LWE is an average-case problem proposed by Regev [140] related to the SIVP_γ and BDD_γ problem on standard euclidean lattices. Informally, the LWE problem can be seen as solving an over-defined system of noisy linear equations. For a given dimension $n \geq 1$, elements in $\mathbb{Z}_q$ with $q > 2$ and a Gaussian error distribution $\mathcal{D}_\sigma$, an LWE instance is nothing but an ordered pair $(\mathbf{A}, T) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where $\mathbf{A} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ is public and $T = \mathbf{A} \cdot \mathbf{S} + E$ with $\mathbf{S} \leftarrow \mathcal{D}_\sigma(\mathbb{Z}_q^n)$ and $\mathbf{E} \leftarrow \mathcal{D}_\sigma(\mathbb{Z}_q)$. There are two versions of the LWE problem, namely, the *decision* LWE and *search* LWE problem. The search LWE problem requires to solve for $\mathbf{S}$ given polynomially many LWE instances $(\mathbf{A}, T)$. The decision LWE problem requires to distinguish between samples drawn from the LWE distribution $(\mathbf{A}, T)$ and random samples drawn from $\mathcal{U}(\mathbb{Z}_q^n \times \mathbb{Z}_q)$. Regev [140] showed a quantum reduction from GapSVP_γ and SIVP_γ to the Decisional LWE problem for $\gamma = O(n/\alpha)$, modulus $q \leq 2^{poly(n)}$ and error distribution of standard deviation $(\alpha \cdot q)$. Subsequent works [42, 129] also proposed classical reductions for polynomial sized modulus $q = poly(n)$. There are other proven variants of the LWE problem based on binary secrets by Brakerski et al. [42] and sparse fixed-weight ternary secrets by Cheon et al. [50].

Cryptographic schemes built upon the standard LWE problem suffered from quadratic key sizes and computational times in the dimension of the lattice [140]. Inspired by the NTRU cryptosystem, Lyubashevsky, Peikert, and Regev [116] also proposed a ring-based analogue of the LWE problem known as the Ring-LWE problem (RLWE), whose security guarantees are based on hard problems on the more algebraically structured "ideal" lattices, as opposed to standard euclidean lattices in the case of standard LWE. They proved that $\text{SIVP}\gamma \leq \text{RLWE}$ over arbitrary ideal lattices for $\gamma = poly(n)/\alpha$. Computations were mapped from matrix-vector to polynomials in the ring $\mathbf{R}_q$ and an RLWE instance consists of ordered pairs $(\mathbf{a}, \mathbf{t}) \in (\mathbf{R}_q \times \mathbf{R}_q)$ where $\mathbf{a} \leftarrow \mathcal{U}(\mathbf{R}_q)$ and $\mathbf{t} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e} \in \mathbf{R}_q$ with $\mathbf{s}, \mathbf{e} \leftarrow \mathcal{D}_\sigma(\mathbf{R}_q)$. Brakerski, Gentry, and Vaikuntanathan [41] proposed a generalized-LWE problem (GLWE), which interpolates LWE and the RLWE problem. A GLWE instance is defined as $(\mathbf{a}, \mathbf{t}) = \mathbf{a} * \mathbf{s} + \mathbf{e} \in (\mathbf{R}_q^k \times \mathbf{R}_q)$, where $\mathbf{a} \in \mathcal{U}(\mathbf{R}_q^k)$, $\mathbf{s} \in \mathcal{D}_\sigma(\mathbf{R}_q^k)$ and $\mathbf{e} \in \mathcal{D}_\sigma(\mathbf{R}_q)$. While the choice of $R = \mathbb{Z}$ yields an LWE instance in $(\mathbb{Z}_q^n \times \mathbb{Z}_q)$, choosing $k = 1$ yields an RLWE instance in $(\mathbf{R}_q \times \mathbf{R}_q)$. Further, Langlois and Stehlé [107] quantumly approximated the hardness of the GLWE problem to worst-case problems over *module* lattices, proposing the Module-LWE (MLWE) problem defined over $\mathbf{R}_q^k$ with $k > 1$. The MLWE problem provides flexibility, since security can be scaled by simply increasing the rank k of the module without altering the underlying ring $\mathbf{R}_q$. MLWE schemes are computationally as efficient as RLWE schemes, since they also involve computations over polynomials in rings, modulo a small constant factor increase in run-time (with the rank k). MLWE instances also inherently possess lesser algebraic structure compared to RLWE instances [107] and thus MLWE offer an attractive trade-off between security (standard LWE) and efficiency (RLWE).

Another variant of the LWE problem proposed by Banerjee et al. [19] is the Learning with Rounding (LWR) problem, which works with deterministic errors by rounding to a smaller modulus. An LWR instance $(\mathbf{A}, T) \in \mathbb{Z}_q^n \times \mathbb{Z}_p$ can be constructed as $T = \lfloor \mathbf{A} \cdot \mathbf{S} \rceil_{q \to p}$ with $q > p \in \mathbb{Z}^+$. The search and decisional variants of the LWR problem are also as hard as the corresponding LWE problems, given the same number of samples and there are no known attacks that specifically exploit the LWR structure. Moreover, the LWR problem can also be refined to its ring and module variants denoted as RLWR and MLWR problems, respectively. One of their main advantages is compactness, since they use rounded keys and ciphertexts. Moreover, LWR-based schemes eliminate need for error sampling, since errors are deterministic through rounding. Chu [76] proposed another interesting integer variant of the RLWE problem (I-RLWE) where the polynomial ring is simply replaced with a big integer ring, by substituting the indeterminate x in $\mathbf{R}_q$ with $q \in \mathbb{Z}^+$. Thus, the instance operates over the ring $\mathbb{Z}/N$, where $N = \phi(q)$. Chu also shows that the I-RLWE problem is as hard as the RLWE problem.

2.3.2 NTRU Problem [79]. The hardness of the first practical lattice-based encryption scheme NTRUEncrypt proposed by Hoffstein, Pipher, and Silverman [79] in 1996 is based on its own *NTRU assumption* or *NTRU one-way function*, which involves factorization of polynomials in polynomial rings. The NTRU problem or one-way function can be defined as follows: For a given small and invertible $\mathbf{p} \leftarrow \mathcal{D}_\sigma(\mathbf{R}_q)$, one is required to distinguish between structured samples $\mathbf{g} \cdot \mathbf{p}^{-1} \in \mathbf{R}_q$ from uniformly random samples in $\mathcal{U}(\mathbf{R}_q)$ with $\mathbf{g} \leftarrow \mathcal{D}_\sigma(\mathbf{R}_q)$. This problem was shown to be reducible to an SVP over a special class of lattices known as *NTRU lattices* [54]. It is still not known if SVP over NTRU lattices is as hard as SVP over general lattices. However, the NTRU problem gained a lot of traction ever since its discovery in 1996 and has since then enabled to build asymptotically fast and compact PKE schemes due to its use of arithmetic over efficient polynomial rings. One main advantage of NTRU cryptosystem is that it has survived cryptanalysis for almost 24 years now and hence instills a lot of confidence in its security claims even while lacking provable security proofs similar to LWE/R-based schemes. There is a provably secure variant of NTRU proposed by Stehlé and Steinfeld [158], but is seldom used in practice owing to poor performance.

3 ALGORITHMS FOR LATTICE-BASED KEY SHARING

There are three different approaches to securely exchange a key between two untrusted parties. They are PKE, KEM, and KEX. While KEX schemes are "participatory," where both parties together determine the shared secret key (similar to the Diffie-Hellman KEX scheme), PKE and KEM schemes typically generate the shared key at one end and securely transport it to the other end. Thus, PKE and KEM schemes can together be referred to as "Key Transport" schemes.

A PKE scheme consists of a triple of procedures (KeyGen, Encrypt, Decrypt). First, the KeyGen procedure is used to generate a public-private key pair (pk, sk) and subsequently, the generated public key pk is used by the Encrypt procedure to encrypt a given message m to generate the ciphertext ct. The Decrypt procedure retrieves the message m from the ciphertext ct using the secret key sk. A semantically secure encryption scheme PKE should ensure that $\forall m \in \mathcal{M}$, $\text{Prob}(m' = m|\text{Decrypt}(\text{Encrypt}(pk, m), sk) = m') = 1 - \epsilon$ where $\epsilon \gtrapprox 0$. The message m is nothing but the shared secret key when a PKE scheme is used for key sharing. A KEM is also associated with three corresponding procedures KeyGen, Encaps (encapsulation), Decaps (decapsulation), respectively. But, a subtle difference between PKE and KEM arises from the way the shared secret key is generated. In a PKE scheme, the shared secret key is an explicit input to the Encrypt procedure, while in a KEM scheme it is an output (not a chosen input) of the Encaps procedure. A KEX scheme is very similar to the Diffie-Hellman key-exchange scheme wherein two parties exchange their public-key shares and subsequently utilize their secret keys to convert the public-key shares to derive a shared secret key. In the following discussion, we will trace the evolution of the various lattice-based PKE, KEX and KEMs proposed in literature leading up to the current NIST candidates. We broadly categorize them into two categories: (1) LWE/R-based and (2) NTRU-based schemes.

3.1 LWE/R-based Schemes

The first practical lattice-based encryption scheme based on *standard* LWE was proposed by Regev [140] in 2005, which subsequently underwent various improvements mainly to reduce key and ciphertext sizes through a number of works [75, 118]. Subsequently, Lyubashevsky, Peikert, and Regev in their seminal work on RLWE [116] in 2010 adapted the same scheme to the RLWE problem, proposing the now famously known "LPR encryption scheme" (referred to as LPREncrypt). Subsequently, Lindner and Peikert [111] considerably improved the parameter sets for LWE and RLWE-based PKE scheme, together forming the first works that laid the foundation for practical LWE-based PKE schemes. Most of the subsequently proposed LWE/R-based PKE/KEMs, including several NIST candidates, can be considered to be either tightly/loosely based

on the framework of the LPREncrypt scheme. We describe LPREncrypt scheme in detail as it captures the essence of multiple LWE/R-based PKE and KEMs, but generalize its description so that specific parameterizations such as structure of the underlying ring ($\mathbf{R}_q$, $\mathbf{R}_q^{k\times k}$, $\mathbb{Z}_q^{n\times n}$), relative sizes of the rounding moduli and distribution of errors can be used to describe specific schemes utilizing this framework.

3.1.1 LPREncrypt PKE Scheme [116]. We define gen to be an extendable output function that takes an input seed say $\rho \in \{0,1\}^*$ and outputs a matrix $\mathbf{a} \in \mathbf{R}_q^{k\times k}$. Message encoding (respectively, decoding) functions denoted as enc (respectively, dec) are used to convert binary messages ($\in \mathbb{Z}_2^n$) into elements in the underlying ring and vice versa. The generalized version of the LPREncrypt framework can be defined as follows:

- LPR.KeyGen(): Generate the public parameter $\mathbf{a} = \text{gen}(publicseed) \in \mathbf{R}_q^{k\times k}$ and $\mathbf{s}, \mathbf{e} \leftarrow \mathcal{D}_\sigma(\mathbf{R}_q^k)$. The LWE instance is calculated as $\mathbf{t} = \mathbf{a}\cdot\mathbf{s} + \mathbf{e} \in \mathbf{R}_q$ and is subsequently rounded as $\mathbf{t} = \lfloor \mathbf{t} \rceil_{q\to p} \in \mathbf{R}_p$. While the tuple $(\mathbf{a}, \mathbf{t})$ forms the public key pk, $(\mathbf{s}, \mathbf{e})$ forms the secret key sk.
- LPR.Encrypt($pk, m \in \mathcal{B}^*$): Generate three components $\mathbf{s}', \mathbf{e}' \leftarrow \mathcal{D}_\sigma(\mathbf{R}_q^k)$ and $\mathbf{e}'' \leftarrow \mathcal{D}_\sigma(\mathbf{R}_q)$. The message m to encrypted is first encoded to $\bar{\mathbf{m}} = \text{enc}(m) \in \mathbf{R}_q$. Subsequently, calculate $\mathbf{u} = \mathbf{a}\cdot\mathbf{s}' + \mathbf{e}'$ and $\mathbf{v} = \mathbf{t}\cdot\mathbf{s}' + \mathbf{e}'' + \bar{\mathbf{m}}$. Subsequently, both $(\mathbf{u}, \mathbf{v})$ are rounded as $\mathbf{u} = \lfloor \mathbf{u} \rceil_{q\to p} \in \mathbf{R}_p^k$ and $\mathbf{v} = \lfloor \mathbf{v} \rceil_{q\to t} \in \mathbf{R}_t$ and published as the ciphertext $ct = (\mathbf{u}, \mathbf{v})$.
- LPR.Decrypt($\mathbf{u}, \mathbf{v}, sk$): Decompress $(\mathbf{u}, \mathbf{v})$ as $\mathbf{u}' = \lfloor \mathbf{u} \rceil_{p\to q}$ and $\mathbf{v}' = \lfloor \mathbf{v} \rceil_{t\to q}$ and calculate $\mathbf{r} = (\mathbf{v}' - \mathbf{u}'\cdot\mathbf{s}) \in R_q$, which when decoded as $\text{dec}(\mathbf{r})$ yields the message $\mathbf{m}'$.

Security and Correctness of LPREncrypt *scheme*: While we briefly explain its security and correctness based on LWE, the same also holds for LWR or a combination of the LWE and LWR problem. The LPR.KeyGen procedure just involves generation of the LWE instance $(\mathbf{a}, \mathbf{t})$. The computational hardness of retrieving the private key from the public key directly comes from the search-LWE problem. The LPR.Encrypt procedure generates two LWE instances, $\mathbf{u}$ (with $\mathbf{a}$ as public parameter) and $\mathbf{v}$ (using $\mathbf{t}$). The message m is first encoded using a standard approach, where each bit is encoded into a coefficient as follows: If bit is 1, then coefficient is $\lceil q/2 \rceil$, else 0. The encoded message is added to $\mathbf{v}$ to essentially hide it within the LWE instance. Both $\mathbf{u}$ and $\mathbf{v}$ are rounded (for LWR or for compactness) and output as the ciphertext ct. Subsequently, the decryption procedure computes $\mathbf{r} = \mathbf{v}' - \mathbf{u}'\cdot\mathbf{s}$ where $\mathbf{v}'$ and $\mathbf{u}'$ are decompressed values of $\mathbf{v}$ and $\mathbf{u}$, respectively. Let the errors introduced due to rounding $\mathbf{x}$ be denoted as $\mathbf{e}_\mathbf{x}$. Then,

$$\begin{aligned}
\mathbf{r} &= \mathbf{v}' - \mathbf{u}'\cdot\mathbf{s} \\
&= \mathbf{t}\cdot\mathbf{s}' + \mathbf{e}'' + \text{enc}(m) + \mathbf{e}_\mathbf{v} - (\mathbf{a}\cdot\mathbf{s}' + \mathbf{e}' + \mathbf{e}_\mathbf{u})\cdot\mathbf{s} \\
&= \mathbf{a}\cdot\mathbf{s}\cdot\mathbf{s}' + \mathbf{e}\cdot\mathbf{s}' + \mathbf{e}'' + \text{enc}(m) + \mathbf{e}_\mathbf{v} - (\mathbf{a}\cdot\mathbf{s}' + \mathbf{e}' + \mathbf{e}_\mathbf{u})\cdot\mathbf{s} \\
&= \text{enc}(m) + \mathbf{e}\cdot\mathbf{s}' + \mathbf{e}'' + \mathbf{e}_\mathbf{v} - \mathbf{e}'\cdot\mathbf{s} - \mathbf{e}_\mathbf{u}\cdot\mathbf{s} \\
&= \text{enc}(m) + \hat{\mathbf{e}}. \qquad (1)
\end{aligned}$$

The computed element $\mathbf{r}$ is approximately equal to the encoded form of the message m as can be seen from Equation (1). As long as the total error $\hat{\mathbf{e}}$ is less than a certain threshold q_t, decryption will result in successful retrieval of m. This PKE scheme can be converted into a KEM in a straightforward manner and the standard approach for conversion is to compute a one-way function (hash $\mathcal{H}$) over the message input m, whose result $\mathcal{H}(m)$ is subsequently used as the message in the LPR.Encrypt procedure, thus preventing the user from directly selecting the shared secret key.

3.1.2 Noisy Diffie Hellman Key Exchange. Key-exchange schemes similar to the Diffie-Hellman KEX scheme but built upon the hardness of the LWE/R problem are referred to as "Noisy Diffie-Hellman" KEX (Noisy − DH) schemes. The first such scheme based on LWE was proposed by Ding, Xie, and Lin [62] and can be informally described as follows: Alice generates a public-key share $\mathbf{b} \in \mathbf{R}_q = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$ and sends it across to Bob. Bob similarly generates his own public-key share $\acute{\mathbf{b}} = \mathbf{a} \cdot \acute{\mathbf{s}} + \acute{\mathbf{e}}$ and also computes $\acute{\mathbf{v}} = \mathbf{b} \cdot \acute{\mathbf{s}} = (\mathbf{a} \cdot \mathbf{s} \cdot \acute{\mathbf{s}} + \mathbf{e} \cdot \acute{\mathbf{s}})$. Bob sends the public-key share to Alice who computes $\mathbf{v} = \acute{\mathbf{b}} \cdot \mathbf{s} = (\mathbf{a} \cdot \mathbf{s} \cdot \acute{\mathbf{s}} + \acute{\mathbf{e}} \cdot \mathbf{s})$. We can see that $\mathbf{v}$ and $\acute{\mathbf{v}}$ are approximately equal to $\mathbf{a} \cdot \mathbf{s} \cdot \acute{\mathbf{s}}$ albeit with some errors. Subsequently, an "error reconciliation" procedure is performed to turn this approximate key agreement into an exact key agreement. Hints are provided by Bob to Alice (typically one bit per coefficient), which will help Alice generate a shared secret key k from the approximate shared secret value.

Reconciliation Techniques: The first reconciliation technique proposed by Ding et al. [62] agreed upon the least significant bits of the shared value but resulted in biased keys, unless post-processed for uniform randomness. However, Peikert [131] proposed an improved technique that enables to agree upon the most significant bit of the shared value to create unbiased keys. The basic idea is as follows: the interval $[0, q]$ is split into four quadrants and two pairs of consecutive quadrants are mapped to 0 and 1 (or vice versa) such that there is maximum leeway for errors (so that every coefficient lies far away from the partition boundaries). As long as the errors are bounded by a certain value, Alice and Bob can correctly agree on a shared secret. Saarinen [148] proposed an improved technique with lesser error probability by selecting only m out of the n coefficients that are close to the center of the partitions within a certain bound b. While the above techniques extract one bit from a single reconciliation in $\mathbb{Z}_q$, Bos et al. [33] generalized it to extract multiple bits from a single reconciliation, albeit at the cost of increased failure probability. Alkim et al. [10] proposed another method of reconciliation based on lattice codes and quantizers that also featured much higher error resilience. But, this technique was fairly complex and hence is not usually preferred, since it requires solving CVP over very small lattices. In general, reconciliation techniques considerably reduce the size of the ciphertext. While PKE schemes require to send two components, i.e., $2n(\lceil \log_2(q) \rceil)$ bits as part of the ciphertext, KEX schemes using reconciliation only send one component and a hint (typically a bit vector) that together yields about $n(1 + \lceil \log_2(q) \rceil)$ bits for the ciphertext. The security and correctness properties of the Noisy − DH KEX scheme can be inferred in a similar manner as the LPREncrypt scheme. A KEX scheme can be converted into a PKE scheme and vice versa. With reference to description of the Noisy − DH scheme, to convert a KEX to a PKE scheme, Bob can simply choose a random secret m and compute $c' = m \oplus k$ and send it to Alice along with the ciphertext ct. Alice can subsequently retrieve k from ct and compute $c' \oplus k$ to retrieve m.

Bos et al. [36] proposed the first practical instantiation of the noisy − DH scheme based on RLWE with concrete parameters ($n = 1024, q = 2^{32} - 1$) achieving 128-bit post-quantum security (referred as BCNS KEX scheme). They also demonstrated its practical performance when implemented in a custom post-quantum ciphersuite in the OpenSSL implementation of transport layer security (TLS) protocol. Subsequently, Alkim et al. [10] enhanced the BCNS scheme to propose the now well-known "NewHope" KEX scheme (referred to as NewHopeUSENIX). It incorporated a number of improvements such as improved security analysis leading to efficient parameter sets ($n = 1{,}024$), provably secure use of a Centered Binomial distribution (CBD) for sampling (instead of discrete Gaussian) and improved error reconciliation with higher error margin. The same authors proposed another variant of NewHopeUSENIX (referred as NewHopeSIMPLE) [9], which does not utilize reconciliation but adopts the "encryption" approach of the LPREncrypt schemes yielding similar performance, but with a slight increase of 6.25% in bandwidth requirement.

Bos et al. [33] concurrently proposed the "Frodo" KEX scheme (referred to as Frodo_CCS), which is the only known KEX scheme based on standard LWE. Subsequently, Sauvik et al. [31] proposed the *spKEX* scheme very similar to Frodo_CCS, but based on LWR with sparse and ternary secrets. Usage of the LWR problem resulted in ciphertexts that were 30% smaller compared to Frodo_CCS. Cheon et al. [50] also proposed another KEX scheme very similar to Frodo_CCS, but based on the sparse LWE problem.

3.2 Security Against Chosen Ciphertext Attacks

All the aforementioned LWE/R-based schemes are only secure in the chosen-plaintext model (IND-CPA secure) and are not secure against an adversary who can adaptively decrypt arbitrary chosen-ciphertexts. A scheme secure against such an adversary is said to be secure under the *indistinguishability under adaptive chosen ciphertext attack* (IND-CCA secure) model. While IND-CCA security is not really a strict requirement in ephemeral key exchanges, it is, however, mandatory when using long term public-private key pairs, which is common in most real world applications.

One of the main requirements for concrete parameterizations of PKE schemes to undergo IND-CCA conversion is to have negligible decryption failure rate (Typically lower than 2^{-128}). But, the parameter sets of the aforementioned schemes were not designed for negligible failure probability [122]. Oder et al. [126] proposed the first IND-CCA secure parameter sets for the NewHopeSIMPLE PKE scheme, which reduced the failure rate from 2^{-60} to 2^{-216} at the cost of 23 bits of post-quantum security (from 256 to 233 bits) by simply reducing the standard deviation of the error distribution. Subsequently, they proposed to use the post-quantum variant of the well-known Fujisaki-Okamoto transformation [73] (FO-transformation) by Targhi and Unruh [159] (TU-variant) to convert an IND-CPA secure PKE scheme into an IND-CCA secure PKE scheme. The main idea of the FO-transformation is to check for malformed ciphertexts and immediately reject them upon detection. Thus, the adversary can only decrypt valid ciphertexts, while invalid chosen ciphertexts always lead to a failure. It primarily works through usage of random oracles (instantiated as hash functions). While its classical variant (secure in the classical random oracle model (ROM)) utilizes two random oracles ($\mathcal{H}, \mathcal{G}$), its post-quantum variant [159] requires three oracles to guarantee post-quantum security ($\mathcal{H}, \mathcal{G}, \mathcal{H}'$). The post-quantum (TU) variant of the FO-transformation is briefly described as follows:

- **Enc(pk,m_{cca})**: Let $(\mathbf{c}_1, \mathbf{c}_2)$ be the ciphertexts of the IND-CPA secure LPREncrypt scheme for the message m_{cca} such that $(\mathbf{u}, \mathbf{v}) = \mathsf{LPREncrypt}(pk, \nu, \mathcal{H}(\nu||m_{cca}))$, where $\nu \in \{0,1\}^*$ and $\mathcal{H}(\nu||m_{cca})$ seeds a secure PRNG to deterministically generate secrets and errors. Define $c_1 = \mathcal{G}(\nu) \oplus m_{cca}$ and $c_2 = \mathcal{H}'(\nu)$. The tuple $(\mathbf{u}, \mathbf{v}, c_1, c_2)$ is output as the ciphertext.
- **Dec(pk,u, v, c_1, c_2, s)**: Compute $\nu' = \mathsf{LPRDecrypt}(\mathbf{u}, \mathbf{v}, \mathbf{s})$ and $m'_{cca} = G(\nu') \oplus c_3$, and verify if $(\mathbf{u}, \mathbf{v}) = \mathsf{LPREncrypt}(pk, \nu, H(\nu'||m'_{cca}))$ and if $c_4 = H'(\nu)$. If so, then output m'_{cca}, else $\perp$.

One noticeable change in the FO-transformed decryption procedure is that a re-encryption is performed after decryption, which adds considerable performance overheads. Oder et al. [126] reported a prohibitive increase of 27 times in the number of clock cycles for the decryption procedure on the ARM Cortex-M4 microcontroller. While the original FO-transformation specified to output a failure symbol upon decapsulation failure (explicit rejection), there exists a slightly modified variant that proposed to output pseudorandom keys upon failure (implicit rejection). This relieves the higher level protocol to check for decapsulation failures. While the above transformation yields an IND-CCA secure PKE scheme, it can also be altered to yield an IND-CCA secure KEM and it is worth noting that FO-transformation in the context of lattice-based schemes has only been used to generate IND-CCA secure PKE and KEMs, but not KEX schemes.

Bernstein and Perischetti [28] propose the concept of "rigidity" for IND-CPA secure PKE schemes. A PKE scheme is said to be rigid if $\mathsf{Encrypt}(\mathsf{pk}, \mathsf{m}) = ct \iff \mathsf{Decrypt}(\mathsf{ct}, \mathsf{sk}) = m$. They state that a rigid IND-CPA secure scheme can be converted to achieve IND-CCA security without re-encryption. While correctness of the scheme implies the forward direction, re-encryption is additionally employed during FO transformation to ensure the reverse direction holds as well. The aforementioned LWE/R-based schemes are not rigid and hence require re-encryption for IND-CCA security. As we will see later in Section 3.6, there are certain IND-CPA secure schemes achieve IND-CCA security by avoiding re-encryption through rigidity [48].

3.3 NTRU-based Schemes

The NTRUEncrypt PKE scheme proposed by Hoffstein, Pipher, and Silverman [79] in 1998 is the first practical lattice-based PKE scheme proposed in literature. The NTRUEncrypt scheme is based on the NTRU one-way function and operates over the polynomial ring $\mathbb{Z}_q[x]/\phi_t$ with $\phi_t = (x^n - 1) = \phi_1\phi_n$ where $\phi_1 = (x - 1)$ and $\phi_n = (x^{n-1} + x^{n-2} + x^{n-3} + \cdots + 1)$. We define (n, p, q) to be the tuple of co-prime integers and polynomials $\mathbf{f}, \mathbf{g}, \mathbf{m}, \mathbf{r}$ as small and ternary polynomials sampled from spaces $\mathcal{L}_f, \mathcal{L}_g, \mathcal{L}_r, \mathcal{L}_m$, respectively. We also denote $C(m, \phi_e)$ to be the set of invertible polynomials in the ring $\mathbb{Z}_m[x]/\phi_e(x)$. Please find below a brief algorithmic description of the probabilistic IND-CPA secure NTRUEncrypt scheme.

- NTRU.KeyGen($seed$): Generate a small polynomial $\mathbf{f} \in \mathcal{L}_f$ such that $\mathbf{f} \in C(2, \phi_1\phi_n)$ and $\mathbf{f} \in C(p, \phi_1\phi_n)$ and sample another polynomial $\mathbf{g} \leftarrow \mathcal{L}_g$. Subsequently, compute $\mathbf{h} = (p \cdot \mathbf{g}/\mathbf{f}) \bmod (q, \phi_1\phi_n)$ and $\mathbf{f}_p = 1/\mathbf{f} \bmod (p, \phi_1\phi_n)$. The quotient polynomial $\mathbf{h}$ is the public key, while the tuple $(\mathbf{f}, \mathbf{f}_p)$ forms the secret key.
- NTRUPKE.Encrypt(pk, $m \in \mathcal{L}_m$, $rand \in \{0,1\}^\ell$): Sample a small random polynomial $\mathbf{r} \leftarrow \mathcal{L}_r$ and compute $\mathbf{c} = (\mathbf{r} \cdot \mathbf{h} + \mathbf{m}) \bmod (q, \phi_1\phi_n)$ where $\mathbf{m}$ is the message polynomial.
- NTRUPKE.Decrypt(sk, $\mathbf{c}$): Compute $\mathbf{a} = (\mathbf{c} \cdot \mathbf{f}) \bmod (q, \phi_1\phi_n)$ and subsequently the message polynomial $\mathbf{m}' = (\mathbf{a} \cdot \mathbf{f}_p) \bmod (p, \phi_1\phi_n)$.

The key generation procedure KeyGen involves generation of the NTRU instance $\mathbf{h} = p \cdot \mathbf{g}/\mathbf{f}$, which is the public key, while $\mathbf{f}, \mathbf{g}$ together form the secret key. The hardness of retrieving the secret key from the public key directly comes from the hardness of inverting the NTRU one-way function. KeyGen involves computationally intensive operations such as sampling an invertible polynomial and computing its inverse, thus making it the costliest procedure in the NTRUEncrypt scheme. The encryption procedure Encrypt generates a random $\mathbf{r}$ and hides a message polynomial $\mathbf{m}$ within the NTRU instance to generate the ciphertext as $(\mathbf{r} \cdot \mathbf{h} + \mathbf{m})$. Subsequently, the decryption procedure utilizes $\mathbf{f}$ and its inverses to retrieve $\mathbf{m}$. One can also retrieve $\mathbf{r}$ as $\mathbf{h}$ has an inverse in the ring. The correctness of the scheme is ensured by the Decrypt procedure as shown in Equation (2).

$$\begin{aligned} \mathbf{a} &= ((\mathbf{c} \cdot \mathbf{f}) \bmod (q, \phi_1\phi_n)) \\ &= (p \cdot \mathbf{g} \cdot \mathbf{r} + \mathbf{m} \cdot \mathbf{f}) \bmod (q, \phi_1\phi_n) \\ \mathbf{m}' &= (p \cdot \mathbf{g} \cdot \mathbf{r} \cdot \mathbf{f}_p + \mathbf{m} \cdot \mathbf{f} \cdot \mathbf{f}_p) \bmod (p, \phi_1\phi_n) \\ &= \mathbf{m} \end{aligned} \tag{2}$$

The encryption and decryption procedures are considerably much faster than KeyGen, since they only involve addition and multiplication of polynomials. In fact, the encryption procedure of NTRUEncrypt only involves a single polynomial multiplication as opposed to two multiplications in the LPREncrypt scheme. The original NTRUEncrypt scheme is also not perfectly correct and is known to be susceptible to attacks exploiting decryption failures [86] and is also susceptible to chosen ciphertext attacks [91]. Thus, a number of padding schemes were proposed, but were

all subsequently broken [123] until Howgrave-Graham et al. [87] proposed the "NAEP" padding scheme considered secure even in the presence of decryption failures and is unbroken till date. In fact, an instantiation of the NTRUEncrypt scheme implemented with the NAEP padding scheme known as SVES-3 (Short Vector Encryption Scheme) is already part of the IEEE standard 1363.1 [163]. The NAEP padding scheme involves additively masking the message $\mathbf{m}$ with a deterministically generated ternary polynomial $\mathbf{t}$ as $\mathbf{m}' = \mathbf{m} - \mathbf{t}$, which is subsequently used as the message to be encrypted. The decryption procedure can recover $\mathbf{m}'$ and further deterministically generate $\mathbf{t}$ and subtract it from $\mathbf{m}'$ to retrieve the message $\mathbf{m}$.

Bernstein et al. [26] proposed NTRUPrime, an IND-CCA secure NTRU-based KEM, which operates over a modified ring $\mathbb{Z}[x]/(x^n - x - 1)$ with $n \in \mathcal{P}$, with the main motivation to avoid usage of algebraically structured rings such as ($\mathbb{Z}_q[x]/(x^n - 1)$ with $n \in \mathcal{P}$) and ($\mathbb{Z}_q[x]/(x^n + 1)$ with $n = 2^k$). Some noticeable departures from the NTRUEncrypt PKE scheme are as follows. First, NTRUPrime is a perfectly correct KEM (zero failure rate). It utilizes rounded ciphertexts, which implicitly generate the message m by rounding each coefficient of $\mathbf{h} \cdot \mathbf{r}$ to the closest multiple of 3. Moreover, the public key in NTRUPrime is computed as $\mathbf{g}/3 \cdot \mathbf{f}$ instead of $3 \cdot \mathbf{g}/\mathbf{f}$ in NTRUEncrypt. IND-CCA security is achieved using an FO-like transformation avoiding use of padding techniques.

Hülsing et al. [89] also proposed an NTRU-based IND-CCA secure KEM (referred to as NTRUHRSS), which is a more simplified variant of the probabilistic NTRUEncrypt PKE scheme. The scheme operates over two rings $\mathbb{Z}_q[x]/[\phi_t(x)]$ and $\mathbf{S} = \mathbb{Z}_p[x]/[\phi_n(x)]$ (ϕ_t and ϕ_n follow from our description of the NTRUEncrypt scheme). Both rings are utilized in such a way to avoid invertibility tests during KeyGen. Moreover, the scheme does not restrict to sampling fixed weight polynomials and also avoids message masking. The scheme is perfectly correct and achieves IND-CCA security through the TU variant of the FO-transformation. Subsequently, Saito-Xagawa-Yamakawa [150] proposed a slight variant of NTRUHRSS KEM (referred as NTRU-SXY), but based on the deterministic variant of NTRUEncrypt PKE and consists of one less hash output compared to NTRUHRSS KEM. Lyubashevsky and Seiler [117] recently proposed NTTRU (after the start of the NIST process), an IND-CCA secure KEM based on the NTRUEncrypt scheme, which operates over a different ring $\mathbb{Z}_q[x]/\phi(x)$ where $\phi(x) = (x^n - x^{n/2} + 1)$ is the nth cyclotomic polynomial with $n = 2^k \cdot 3^\ell$. Their proposed parameters allowed use of the efficient Number Theoretic Transform (NTT) for polynomial multiplication, which is a first for an NTRU-based scheme. It is worth noting that the AVX2 implementation of NTTRU is faster by several factors (refer to Table 4) compared to all lattice-based round 2 NIST candidate PKE/KEMs.

3.4 Use of Error Correcting Codes in Lattice-based PKE/KEMs

As seen earlier, all LWE/R-based schemes are associated with a certain decryption failure probability. Designers strive to achieve a zero or atleast negligible failure probability as it is mandatory to achieve IND-CCA security but also allows for a simpler and robust security analysis without unnecessary caveats. It also reduces the attacker's chances of triggering decryption failures, which could lead to possible attacks as shown in References [23, 57, 69]. However, one obvious disadvantage is the performance overhead incurred due to possible *over-designing* of the parameter set for zero or negligible decryption failures. For instance, reducing the error sizes in LWE instances (or increasing the rounding modulus in LWR) could lead to reduction in decryption failures but also reduces security against classical/quantum attacks. This security loss is usually offset by increasing the lattice dimension, which comes with a certain cost on performance.

Alternatively, one could use certain other approaches such as improved message encoding techniques (encoding one bit in multiple coefficients as used in NewHopeUSENIX [10]) or improved

reconciliation techniques [148, 160] to correct larger errors. But, one generic technique that is gaining traction is utilization of Error Correcting Codes (ECCs) to artificially correct errors induced in the message after decryption. The idea is to encode the message $m \in \{0,1\}^n$ into a codeword $c \in \{0,1\}^m$ (i.e) $c = \mathsf{ECCEncode}(m)$, which is used as a modified input to be encrypted. The decryption procedure then retrieves the codeword $c' \in \{0,1\}^m$ and utilizes the ECC's decoding procedure to retrieve $m' \in \{0,1\}^n$. Correctness can be ensured as long as the error rate of the scheme is less than the error correcting capability. An advantage of utilizing ECCs is that decryption failure rate can be improved without affecting theoretical security guarantees of the scheme.

A systematic study of the use of ECCs for lattice-based schemes was first done by Fritzmann et al. [70] who investigated usage of modern and classical ECCs such as LDPC (Low Density Parity Codes) [74] and BCH (Bose Chaudhuri Hocquenghem) codes [51], respectively, to reduce decryption failure rates for the NewHopeSIMPLE PKE scheme. Usage of ECCs come with an advantage of reduced ciphertext sizes along with leeway to increase span of error distribution for increased security. However, there is also an added disadvantage of performance penalty especially when heavy error correcting codes such as BCH are used. Thus, it might better for schemes to design fairly robust parameter sets with a low enough failure rate, which only requires light error correction to achieve negligible failure rate, to not adversely impact performance. Three second round NIST candidates utilize ECCs to achieve negligible decryption failure rate, about which we will discuss in Section 3.6.

3.5 Key Reuse in LWE/R-based Schemes

Key-reuse or Key-caching is a commonly adopted technique in secure protocols such as the TLS and IKE (Internet Key Exchange). In fact, the draft version of TLS 1.3 [1] allowed key reuse of the client, but finally replaced it with a pre-shared key identity in the final standards [2]. Thus, security analysis in practical scenarios such as key-reuse is thus important toward understanding issues with practical deployment of lattice-based schemes. We consider a scenario where Alice and Bob run an IND-CPA secure key-sharing procedure where Alice uses a long term public key to initiate multiple key-exchange sessions with Bob.

Scott Fluhrer [69] proposed the first attack over IND-CPA secure noisy-DH KEX schemes such as the BCNS [36] and Frodo_CCS [34] schemes exploiting key-share reuse. The main idea of the attack is as follows: Bob constructs "handcrafted" public-key shares and reconciliation hints and uses it to perform multiple key exchanges with Alice. The attacker assumes the presence of a *key mismatch* oracle that provides a binary response (true or false) for the attacker's guesses of the shared secret key, which is a weaker assumption compared to a classical *decryption* oracle. The attacker can correlate the oracle's responses for his specially chosen ciphertexts to retrieve the complete secret key. Ding et al. [59] further proposed a variant known as the "signal leakage attack" that works when the protocol is initiated by Bob (opposite direction). The adversary does not assume an oracle but simply observes changes in the reconciliation hints based on the handcrafted public-key shares to recover the secret key. Thus, KEX schemes are attackable through chosen-ciphertext attacks, which work irrespective of who initiates the protocol, as long as there is key-share reuse by atleast one of the involved parties. While the initial key-reuse attacks were reported on "reconciliation-based" KEX schemes [59, 61], *key mismatch* or *plaintext checking* oracle attacks were subsequently generalized and extended to IND-CPA secure "encryption-based" PKE/KEMs by a number of works [17, 23, 60, 137], thus showing that all IND-CPA secure LWE/R-based schemes are vulnerable in key-reuse scenarios. These attacks do not work on IND-CCA secure schemes, since they always return a decapsulation failure for invalid ciphertexts and thus the attacker cannot get any information about the output of the encapsulated decryption procedure. However, Bauer et al. [23] suggested the possibility that side-channel leakage from the decapsulation

procedure can be exploited to instantiate a practical *plaintext checking* oracle to attack IND-CCA secure schemes. In fact, concrete side-channel attacks were demonstrated over IND-CCA secure schemes by a couple of works [64, 139], about which we will discuss in Section 5.2.3.

This line of work brought to light a subtle but important caveat with respect to replacing classical DH schemes with lattice-based PKE/KEM/KEXs for key exchange. Classical DH schemes are participatory and symmetric and allow caching of public-key shares at both ends (non susceptible to key-share reuse attacks). But as we can see, long term keys can only be used with IND-CCA secure lattice-based schemes that are non-participatory PKE/KEMs, which allow key caching only at the decapsulator's end. This thus restricts use of certain message flows in implicitly authenticated key-exchange procedures common in several real-world security protocols as shown in Reference [43], which highlights a critical issue that lattice-based schemes cannot serve as direct drop-in replacements for classical DH schemes in all possible scenarios.

3.6 Round 2 NIST Candidates

In this section, we will briefly review the nine lattice-based PKE/KEMs (seven LWE/R-based and two NTRU-based) currently competing in the second round of the NIST standardization process. Since these schemes undergo constant minor modifications to their specifications, we would like to clarify that the discussions and comparisons presented below are based on details from the latest available (dated) specification documents of the respective schemes (according to the second round) cited in the references section.

3.6.1 LWE/R-based Candidates. All NIST candidate PKE/KEMs based on LWE/R contain in their core an IND-CPA secure PKE scheme based on the framework of the LPREncrypt PKE scheme, unless otherwise specified. None of the underlying PKE schemes are participatory as the shared key is generated at the encapsulator. Moreover, all the candidates offer IND-CCA security through variants of the FO transformation [73, 159]. These candidates mainly differ based on multiple parameters including but not limited to the structure of the underlying ring, relative sizes of the rounding moduli, error distribution and choice of error correction code. All schemes provide three parameter sets at NIST defined security levels 1, 3, and 5, unless specified otherwise.

Frodo: Frodo is a family of IND-CCA secure KEMs and the only candidate based on standard LWE (referred as FrodoKEM) [8]. It utilizes a power of 2 modulus ($q \leq 2^{16}$) and is the only KEM that samples errors from a rounded-Gaussian distribution. The scheme offers two main advantages (1) security based on standard LWE instills confidence in long-term security, (2) simplistic design reducing potential for errors. But, the obvious disadvantage is the order of magnitude difference in speed and bandwidth requirement compared to its structured counterparts.

NewHope: NewHope is a suite of IND-CCA secure KEMs based on RLWE and very similar to the NewHopeSIMPLE PKE scheme (referred as NewHopeKEM) [7]. NewHope operates over $\mathbb{Z}_q[x]/(x^n + 1)$ (power of 2 cyclotomic ring) and parameters are chosen to admit use of the efficient NTT-based polynomial multiplication, while secrets and errors are sampled from a narrow CBD. Though NewHopeNIST offers competitive performance, one subtle disadvantage is that NewHopeNIST only offers two parameter sets with NIST security level 1 ($n = 512$) and 5 ($n = 1{,}024$) due to restrictions on parameters ($n = 2^k$) to allow usage of NTT operation.

Kyber: Kyber is a suite of IND-CCA secure KEMs (referred as KyberKEM) based on MLWE (KyberKEM) [12]. It operates over the module $\mathbf{R}_q^k$, wherein the underlying ring $\mathbf{R}_q$ is same as that of NewHopeNIST and also utilizes NTT for polynomial multiplication. Secrets and errors are sampled from a simple CBD distribution and ciphertexts are compressed for compactness. The design of KyberKEM is very simple and its security is also very scalable (with dimension k of the module).

One main disadvantage is the size of the public module $\mathbf{a}$ with k^2 polynomials, which is k times more compared to an RLWE scheme with equivalent security.

Saber: Saber is a suite of IND-CCA secure KEMs very similar to KyberKEM, but based on MLWR (referred as SaberKEM) [56]. SaberKEM operates in the same ring as KyberKEM, but utilizes a power of 2 modulus, which simplifies modular arithmetic and accelerates sampling operations. SaberKEM offers very competitive speeds along with bandwidth performance owing to use of rounded/compressed elements.

LAC: LAC is a suite of cryptographic algorithms based on RLWE offering IND-CCA secure PKE, KEM and KEX schemes. We refer to the latest specification of LAC (LAC-v3) [115], which we denote as LACKEM. It is the only candidate that offers byte level moduli (q = 256/251) facilitating for bandwidth efficiency and highly vectorized implementations. They operate over a similar cyclotomic ring as NewHopeKEM and sample secrets and errors from a narrow CBD with fixed hamming weight. The use of byte-size modulus comes with high decryption failures and a combination of BCH and D2 error correcting codes are used to achieve negligible failure rates.

Round5: Round5 is a suite of cryptographic algorithms offering IND-CPA secure KEMs and IND-CCA secure PKE based on the GLWR problem with sparse ternary secrets, which can be instantiated as both an LWR/RLWR/MLWR problem (referred as Round5NIST) [16]. Round5 is based on a unified design with the same code base and thus offers flexibility. The operating ring is $\mathbb{Z}_q[x]/(\phi(x))$ with $\phi(x) = x^n + x^{n-1} + \cdots + 1$ (prime cyclotomic ring). The operating modulus is a power of 2 (less than 2^{16}) and the scheme utilizes a lightweight linear error correcting code known as "XEf" for negligible failure rates [148]. Round5KEM also offers the best bandwidth performance among lattice-based schemes mainly owing to use of sparse and ternary secrets along with rounded public keys and ciphertexts.

ThreeBears: ThreeBears is a family of IND-CCA secure KEMs based on the Integer version of the MLWE (I-MLWE) problem (referred as ThreeBearsNIST) [78]. This is the only LWE-based candidate that is built upon a "reconciliation-based" noisy − DH scheme that is subsequently converted to a KEM, unlike other schemes that directly build upon the LPREncrypt style PKE scheme. The scheme mainly involves computation with big integers and a simple Melas-type BCH ECC is used to achieve negligible decryption failure rates. Three parameter sets are offered at NIST security levels 2, 3, and 4.

3.6.2 NTRU-based Candidates. Two schemes NTRUKEM and NTRUPrimeKEM are based on hardness of the NTRU assumption and the framework of the NTRUEncrypt PKE scheme. An attractive feature of both these schemes is that these KEMs are perfectly correct, devoid of any decryption failures. Both the NTRU-based schemes offer very competitive performance for both encapsulation and decapsulation along with compact keys and ciphertexts.

NTRU: NTRU is a family of perfectly correct IND-CCA secure KEMs (referred as NTRUKEM) [48] and is a merger of two first round candidates NTRUEncrypt and NTRUHRSS KEMs. The scheme is a variant of the NTRU-SXY KEM that supports parameter sets of both its aforementioned parent schemes. This underlying PKE has the ability to check for invalid ciphertexts and hence is rigid. Thus, it does not rely on re-encryption for IND-CCA security and is in fact the only scheme that does not utilize re-encryption for IND-CCA security [28].

NTRUPrime: NTRUPrime is a family of perfectly correct IND-CCA secure KEMs (referred as NTRUPrimeKEM), which come in two flavours—StreamlinedNTRUPrime and NTRULPRime [25]. The StreamlinedNTRUPrime is directly based on the NTRUPrime KEM [26], where the public key

is in a quotient form commonly used in NTRU-based schemes and ciphertexts in a rounded product form ($\mathbf{h} \cdot \mathbf{r}$) with an implicit message. But, NTRULPRime is based on the "encryption-based" LPREncrypt PKE scheme where the public key is a rounded product similar to an LWR instance. The designers opted to go for both the approaches, since it is not clear which of the quotient NTRU or product NTRU is a safer option. Both the variants use rounded ciphertext components and hence have very competitive bandwidth performance.

3.7 Concrete Security Analysis

In this section, we very briefly touch upon well-known techniques to estimate concrete hardness of lattice-based schemes, which also offer guidance to choose appropriate parameters to meet the desired security levels. There are primarily two generic classes of attacks against lattice-based schemes: (1) primal attack and (2) dual attack. Primal attack typically works by solving the search-LWE problem as a unique SVP on the same lattice. The dual attack tries to solve the decisional-LWE problem by reducing it to a SIS problem and in-turn into a problem of finding short vectors in a related dual lattice. While primal attacks are considered for both LWE and NTRU schemes, dual attacks are only considered for LWE schemes. It is also worth noting that both attacks do not exploit any algebraic structure in any of the variants of the LWE/R or NTRU problem.

The best-known algorithms to solve such problems typically rely on lattice-reduction algorithms and BKZ is the most common algorithm used, given the limited number of LWE/NTRU samples available to the adversary (samples are constructed from the public key/ciphertext) [152]. The BKZ algorithm in turn involves polynomially many calls to an SVP oracle operating over a small dimension β. Schemes, however, ignore the polynomial factor and conservatively only evaluate the cost for a single SVP call in dimension β. Two well-known techniques for solving SVP are (1) sieving and (2) enumeration. Enumeration requires super exponential time but very limited memory and is typically more efficient in lower dimensions, but sieving, which consumes exponential time and memory space, is widely considered to be better for high dimensions used in lattice-based schemes. The cost of sieving in the RAM model in a dimension β is estimated to be $\beta \cdot 2^{c\beta} + o(\beta)$ and one of the best-known and commonly agreed upon heuristic complexity to solve it classically is $2^{0.292\beta}$ while it is $2^{0.265\beta}$ assuming a quantum speedup due to the Grover's algorithm (β factor is usually ignored for conservatism). Concrete security estimates done using this cost model are well known as core-sieve (classical) or Q-core-sieve (quantum), respectively. While this is the most commonly used cost model, several other cost models exist in literature, each associated with a certain set of assumptions about the operation and memory access costs. We refer to the work of Albrecht et al. [6] who perform a concrete and detailed security analysis of all the round 1 lattice-based NIST candidates based on both the primal and dual attacks considering a wide range of cost models available in literature.

However, there exists another class of attacks called hybrid attacks, which use a combination of lattice-reduction and combinatorial techniques sometimes yielding better attacks than primal and dual attacks [85]. Among the nine NIST candidates, hybrid attacks have shown to be performing better on NTRU-based schemes NTRUKEM, NTRUPrimeKEM, and Round5PKE with sparse secrets and some variants of LACKEM. It is worth noting that the estimator of Albrecht et al. [6] does not take hybrid attacks into consideration. Please refer Table 2 for some representative parameter sets of the all the second round NIST candidate PKE/KEMs secure in the IND-CCA security model.

4 BUILDING BLOCKS FOR PRACTICAL LATTICE-BASED CRYPTOGRAPHY

The two most computationally intensive operations used in lattice-based cryptography are polynomial/matrix multiplication and discrete distribution sampling. Thus, the performance and efficiency of any lattice-based scheme mainly hinges on the way these operations are implemented. In

Table 2. Parameter Sets of the Second Round NIST Candidate PKE/KEMs Based on the LWE/R and the NTRU Problem, Which Are Grouped Together Based on NIST Security Levels 1, 3, and 5

Scheme	Parameter	Hardness	(Dimension, Modulus)	Attack	C/PQ security (bits)	Failure Prob.	Size(bytes)		
						pk	ct	sk[1]	
NIST Security Category 1									
FrodoKEM	FrodoKEM – 640	LWE	640, 2^{15}	Dual	148/108	$2^{-138.7}$	9,616	9,720	19,888
NewHopeKEM	NewHope512	RLWE	512, 12,289	Dual	112/101	2^{-213}	928	1,120	1,888
LACKEM	LAC – 128 – v3a	RLWE	512, 251	Dual	148/141	2^{-151}	544	704	512
Round5PKE	R5ND_1PKE_5d	RLWR	508, 2^{10}	Hybrid	132/120	2^{-142}	461	636	493
Round5PKE	R5N1_1PKE_0d	LWR	636, 2^{12}	Hybrid	145/133	2^{-146}	5,740	5,804	5,772
KyberKEM	Kyber512	MLWE	256×2, 3,329	Dual	111/100	2^{-178}	800	736	1,632
SaberKEM	LightSaber	MLWR	256×2, 2^{13}	Primal	125/114	2^{-120}	672	736	2,304
ThreeBearsKEM (2)	BabyBear	I-MLWE	312×2, 2^{10}	Primal	154/140	2^{-156}	804	917	40[2]
NTRUKEM	ntruhps2048677	NTRU	677, 2,048	Hybrid	-/144	0	931	931	1,235
NTRUKEM	ntruhrss701	NTRU	701, 8,192	Hybrid	-/134	0	1,138	1,138	1,452
NTRUPrimeKEM (2)	sntrup653	NTRU	653, 4,621	Hybrid	129/117	0	994	897	1,518
NTRUPrimeKEM (2)	ntrulpr653	NTRU	653, 4,621	Hybrid	130/118	0	897	1,025	1,125
NIST Security Category 3									
FrodoKEM	FrodoKEM – 976	LWE	976, 2^{16}	Dual	214/154	$2^{-199.6}$	15,632	15,744	31,296
LACKEM	LAC – 192 – v3a	RLWE	1,024, 251	Hybrid	278/267	2^{-324}	1,056	1,352	1,024
Round5PKE	R5ND_3PKE_5d	RLWR	756, 2^{12}	Hybrid	194/176	2^{-256}	780	950	828
Round5PKE	R5N1_3PKE_0d	LWR	876, 2^{15}	Hybrid	192/175	2^{-144}	9,660	9,732	9,708
KyberKEM	Kyber768	MLWE	256×3, 3,329	Dual	181/164	2^{-164}	1,184	1,088	2,400
SaberKEM	Saber	MLWR	256×3, 2^{13}	Primal	203/185	2^{-136}	992	1,088	2,304
ThreeBearsKEM (4)	MamaBear	I-MLWE	312×3, 2^{10}	Primal	235/213	2^{-206}	1,194	1,307	40[2]
NTRUKEM	ntruhps4096821	NTRU	821, 4,096	Hybrid	-/178	0	1,230	1,230	1,592
NTRUPrimeKEM	sntrup761	NTRU	761, 4,591	Hybrid	153/139	0	1,158	1,039	1,763
NTRUPrimeKEM	ntrulpr761	NTRU	761, 4,591	Hybrid	155/140	0	1,039	1,167	1,294
NTRUPrimeKEM (4)	sntrup857	NTRU	857, 5,167	Hybrid	175/159	0	1,184	1,312	1,463
NTRUPrimeKEM (4)	ntrulpr857	NTRU	857, 5,167	Hybrid	176/160	0	1,322	1,184	1,999
NIST Security Category 5									
FrodoKEM	FrodoKEM – 1344	LWE	1,344, 2^{16}	Dual	279/201	$2^{-252.5}$	21,520	21,632	43,088
NewHopeKEM	NewHope1024	RLWE	1,024, 12,289	Dual	257/233	2^{-216}	1,824	2,208	3,680
LACKEM	LAC – 256 – v3a	RLWE	1,024, 251	Dual	316/301	2^{-302}	1,056	1,464	1,024
Round5PKE	R5ND_5PKE_5d	RLWR	946, 2^{11}	Hybrid	256/232	2^{-227}	978	1,301	1,042
Round5PKE	R5N1_5PKE_0d	LWR	1217, 2^{15}	Hybrid	257/234	2^{-144}	14,636	14,724	14,700
KyberKEM	Kyber1024	MLWE	256×5, 3,329	Dual	254/230	2^{-174}	1,568	1,568	3,168
SaberKEM	FireSaber	MLWR	256×5, 2^{13}	Primal	283/257	2^{-165}	1,312	1,472	3,040
ThreeBearsKEM	PapaBear	I-MLWE	312×4, 2^{10}	Primal	314/280	2^{-256}	1,584	1,697	40[2]

However, schemes that do not belong to same security level are explicitly mentioned in brackets. unless otherwise specified explicitly in brackets. C denotes classical while PQ denotes Post-quantum.

this section, we will review in detail the various implementation aspects (algorithmic and design optimizations) of these sub-blocks used in lattice-based key-sharing schemes.

4.1 Polynomial/Matrix Multiplication

Polynomial multipliers are utilized in almost all lattice-based NIST candidates in the second round except FrodoKEM, which operates over matrices and vectors. We consider four main parameters/factors that heavily influence the adopted technique to perform polynomial multiplication,

[1]The secret key of ThreeBears is chosen to be a short random seed (40 bytes) for compactness. Thus, this seed needs to be expanded into a pseudorandom element using a deterministic function before being used for computation.

[2]We do not directly compare the secret key sizes as some schemes such as ThreeBears choose to use a short random seed as the secret key, which is pseudonrandomly expanded before being used for computation, while other schemes choose to directly denote the pseudorandom element as the secret key.

(1) Operating modulus: q; (2) Degree of polynomial: n; (3) Polynomial Ring: $\mathbf{R}_q$; (4) Nature of operands: e.g., sparse/non-sparse, binary/ternary/large. We together refer to them as the *Impl_factors*. We can also orthogonally decompose the operation of polynomial multiplication into two layers (1) High-level Polynomial Arithmetic and (2) Low-level Modular Arithmetic. Some of the standard high-level algorithmic approaches toward polynomial multiplication include School-book method [102], Karatsuba [99], Toom-Cook [161], and NTT [53]. The choice of a particular technique is based on the consideration of all the aforementioned *Impl_factors*.

4.1.1 Number Theoretic Transform. The NTT approach for polynomial multiplication is typically utilized in polynomial rings such as $\mathbb{Z}_q[x]/(x^n + 1)$ with $n = 2^k$ and $q \in \mathcal{P}$ satisfying $q \equiv 1 \mod 2n$. Such a value of q ensures the presence the $2n^{th}$ root of unity in $[0, q]$, which facilitates full factorization of $(x^n + 1)$ in the ring into linear factors (such q is referred to as *max split modulus*). In such a case, the polynomial multiplication operation (along with reduction) can be implemented as "negacyclic" convolution using the NTT, which is a ring-based analogue of the efficient Fast Fourier Transform (FFT). Modular multiplication of two polynomials $\mathbf{a}$ and $\mathbf{b}$ in the ring can be computed as $\mathbf{a} \cdot \mathbf{b} = \text{NTT}^{-1}(\text{NTT(a)} * \text{NTT(b)})$, which involves two forward NTTs and one inverse NTT (2 NTT and 1 NTT^{-1}). Both NTT and NTT^{-1} can be computed in $O(nlog_2(n))$ time and the point-wise multiplication operation $(*)$ in $O(n)$ time and thus NTT-based polynomial multiplication can be computed in $O(nlog_2(n))$ time. One advantage of NTT is that it is homomorphic with respect to addition/subtraction, i.e., $\text{NTT}(\mathbf{a} \pm \mathbf{b}) = \text{NTT}(\mathbf{a}) \pm \text{NTT}(\mathbf{b})$. This saves un-necessary back and forth conversions and also enables schemes to generate ciphertexts in the NTT domain [7]. But, NTT is not homomorphic with respect to the rounding operation and hence is generally not attractive for LWR-based schemes, e.g., SaberKEM and Round5PKE have opted not to use NTT. Moreover, NTT is also an isomorphism (bijective mapping onto itself) and thus uniformly random polynomials or vectors of polynomials can be directly sampled in the NTT domain, saving about k^2 NTT operations (k is rank of the module).

Algorithmic Description: The NTT uses the power of nth and $2n$th roots of unity in $\mathbb{Z}_q$ as pre-computed constants (referred together as twiddle factors (Γ)). The NTT is computed in $log_2(n)$ stages with each stage consisting of $n/2$ independent "butterfly" operations. A "butterfly" operation takes three inputs (a, b, Γ) and involves one multiplication, one subtraction and one addition to generate an output pair (c, d). One can opt between two types of butterfly operations: (1) Cooley-Tukey butterfly: $(c, d) = (a + \Gamma \cdot b, a - \Gamma \cdot b)$; and (2) Gentleman-Sande butterfly: $(c, d) = (a + b, (a - b) \cdot \Gamma)$. The pairs of coefficients and the twiddle factors used in a butterfly operation depend upon the stage of the NTT. Apart from the NTTs, NTT-based multiplication also involves some overhead steps such as pre-scaling, post-scaling and reordering of the inputs. However, intelligent algorithmic optimizations involving use of alternate butterfly structures and modified twiddle factors suggested by Roy et al. [144] and Pöppelmann et al. [135] eliminated all the overhead steps by integrating them into the NTT transform.

Implementation Aspects: The NTT operation is an in-place computation and thus does not consume extra stack space, but is memory intensive involving lot of load and store operations. Thus, reducing reducing memory operations tremendously improves speed of the NTT operation. Well-known techniques for improved memory accesses include: (1) Merging multiple stages of NTT using vectorized load and store instructions and computation through effective utilization of available registers [11, 39]; (2) packing multiple coefficients of a butterfly operation in a single memory location and dynamically changing the relative location of the coefficients after each stage such that coefficient pair required for next stage are stored together [144]; and (3) address-based partitioning of the coefficients in multiple memory modules, which can be retrieved simultaneously enabling

single cycle butterfly operations in hardware [20, 133]. It is worth noting that there is also an out-of-place variant of NTT known as constant-geometry NTT. The in-place NTT involves accessing coefficients in different order depending upon the stage of the NTT, while the constant-geometry NTT accesses the coefficients in the same order in every stage of the computation, which heavily simplifies the read-write control logic, but requires additional memory. While in-place NTT is the popular choice, constant-geometry NTT has been utilized in hardware implementations to simplify control logic [20].

Another aspect with respect to implementing NTT is generation and utilization of the twiddle factors. Most implementations (mostly SW) adopt the standard approach of pre-computing the twiddle factors [58, 135], which can be memory hungry on embedded platforms. One can also completely generate all the twiddle factors online in conjunction with the NTT operation using a dedicated multiplier in HW [14], but could affect performance on SW platforms. However, one could trade performance for memory by pre-computing a portion of the twiddle factors and generating the others on the fly, as done by Alkim et al. [11, 20]. NTT can also be heavily parallelized in hardware by computing multiple butterfly operations in parallel, in pursuit for very fast NTT implementations, however, it also complicates memory access scheme with complex control logic as shown in References [49, 104, 156].

NTT over alternate rings: Lyubashevsky and Seiler [153] recently proposed possibility of performing efficient NTT operations in a new ring $\mathbb{Z}_q[x]/(x^n - x^{n/2} + 1)$ where the modular polynomial is an nth cyclotomic polynomial with $n = 2^k \cdot 3^\ell$. For $(n, q) = (768, 7681)$, they showed that $(x^n - x^{n/2} + 1)$ can be factorized in the ring up to cubic factors, leaving $n/3$ polynomials each with degree 3. Thus, pointwise multiplication in the case of $\mathbb{Z}_q[x]/(x^n + 1)$ is replaced with multiplication of $n/3$ cubic polynomials modulo $(x^3 - \gamma_i)$ for (i = $0, \ldots, n/3 - 1$), each of which requires 11 multiplications and 6 additions. The AVX2 implementation of this NTT with $(n, q) = (768, 7681)$ was shown to be twice as fast as NTT in the ring $\mathbb{Z}_q[x]/(x^n + 1)$ with $(n, q) = (256, 7681)$. This could open doors to attractive possibilities such as having NewHopeKEM in dimension $n = 768$ offering NIST level 3 security and also possibly identifying several other rings where efficient NTT is possible.

4.1.2 Karatsuba and Toom-Cook Algorithm. Other standard well-known approaches toward polynomial multiplication are Karatsuba [99] and Toom-Cook [161] (a generalization of Karatsuba), both of which are asymptotically faster than the schoolbook approach and typically used when application of NTT is not possible. Karatsuba multiplication is a divide-and-conquer approach that involves splitting multiplication of two n-degree polynomials into three (instead of four) multiplications of $n/2$-degree polynomials. This splitting is done recursively until the schoolbook approach outperforms Karatsuba over small degree polynomials. Toom-Cook involves splitting polynomials into more than two parts (Toom-3 with five multiplications of $n/3$ length polynomials or Toom-4 with seven multiplication of $n/4$ length polynomials). The efficiency comes from trading multiplications with the cheaper additions. While modular reduction after multiplication is integrated into the NTT, it has to be done separately when using Karatsuba or Toom-Cook.

Both Karatsuba and Toom-Cook are out-of-place computations and recursive in nature and hence consume extra stack space. Karmakar et al. [100] proposed to utilize an in-place variant of the Karatsuba multiplication algorithm to implement SaberKEM on the memory constrained ARM Cortex-M0 microcontroller, which, however, comes at a cost on performance ($\approx 2x$). They are also highly memory intensive with numerous loads and stores. Thus, several works have proposed efficient techniques to minimize time due to load-store operations by making use of vectorized load-store instructions and efficiently utilizing all the available registers for computation [97, 100]. Kannwischer et al. [97] recently proposed fast polynomial multiplication routines utilizing

an optimal combination of Karatsuba, Toom-Cook, and schoolbook multiplication for polynomial rings $\mathbb{Z}_q[x]/(\phi(x))$, where $q = 2^k$. They perform an automated exploration for the optimal combination of the three techniques for different polynomial degrees. They also utilized several low assembly level optimizations to achieve the fastest multiplication routines for two NIST candidates SaberKEM and NTRUKEM on the ARM Cortex-M4. It is worth noting that the fastest AVX2 optimized polynomial multplication in the ring $\mathbb{Z}_q[x]/(x^n - 1)$ with $q = 2^k, n \in \mathcal{P}$ was reported by Hülsing et al. [89] who also used an optimal (but adhoc) combination of Karatsuba, Toom-Cook and schoolbook techniques coupled with low-level assembly optimizations.

4.1.3 Other Specialized Routines. While the Karatsuba and Toom-Cook multiplication algorithms are efficient for a general case, it might be possible in some cases to exploit certain special structures within the operand polynomials. For example, when one of the operand polynomials is small (binary or ternary) and sparse, multiplication can simply be done using adders and shifters (eg. LAC, NTRUKEM, and NTRUPrimeKEM). In some cases, this was found to be more efficient than utilizing asymptotically better techniques such as the NTT, Karatsuba, or Toom-Cook [46, 134]. Several works have also addressed the performance issues of polynomial multiplication in the ring $\mathbb{Z}_q[x]/(x^n - 1)$ (NTRUEncrypt scheme) by considering it as a convolution operation. Several hardware implementations for performing fast convolution of sparse and binary polynomials have been proposed [18, 94, 108]. Liu et al. [112] proposed a generic technique to perform truncated polynomial multiplication in the same ring using low-cost LFSR (Linear Feedback Shift Register) and the same technique has been adopted in several recent works to perform high speed polynomial multiplication [67, 68]. In fact, this technique has shown to be adaptable to multiply in several other rings and can (with high resource utilization) perform n-degree truncated polynomial multiplication in about n clock cycles (linear time).

4.1.4 Low-level Modular Arithmetic. Utilizing efficient modular reduction techniques is integral to achieving high performance, since it is the most utilized atomic computation in the scheme and in particular during polynomial multiplication, to limit the width of the product. The operating modulus q is crucial in choosing the modular reduction technique. Some standard choices are (1) $q = 2^k$ used in schemes such as SaberKEM, NTRUKEM, and Round5PKE (2) $q \in \mathcal{P}$ used in schemes such as NewHopeKEM, KyberKEM, NTRUPrimeKEM, and LACKEM. When $q = 2^k$, reduction is almost free, since it can be simply done by ignoring the MSBs of the product. However, when q is a prime, there are several efficient (division-less) techniques known to perform reduction such as naïve schoolbook reduction, Montgomery reduction [120] and Barrett reduction [21]. Naïve schoolbook reduction is typically used when coefficients are represented as single/double-precision floating point numbers (e.g., in floating-point-based AVX2 accelerated NTT implementations on Intel x-64 [7, 10]). It simply involves multiplication with an approximate precomputed inverse of the modulus q yielding an approximate quotient, which is subsequently multiplied with the modulus and subtracted from the input to yield the result. This is convenient, since vectorized floating-point multiplications are faster than integer multiplication on AVX2 Intel processors.

However, on embedded platforms where floating point arithmetic is not possible, well-known integer reduction techniques such as Barrett and Montgomery reduction are utilized. Barrett reduction is used to perform one-time reductions of numbers, which can typically fit in one word (not too large) eg. output of addition. It is an optimized and more efficient form of the naïve schoolbook reduction technique, which involves smaller multiplications and arithmetic shift operations [21]. While the Barrett reduction is used for general primes, specialized routines can be used if the prime has a certain special structure. For instance, NewHopeKEM utilizes the prime $q = 2^{13} + 2^{12} + 1$ while KyberKEM works with $q = 2^{13} - 2^9 + 1$. Several reported works have demonstrated that

modular reduction using these special primes can be done be efficiently by trading multiplication operations with shift and add operations [114, 153].

However, when back to back multiplications are performed, Montgomery reduction is the most common technique used to typically reduce large double-word integers. Let us consider computation of s mod q where $s = mq + t$ and $\beta = 2^{\ell}$ is the minimum word size that can fit q. It involves calculating Hensel remainder $t' = s\beta^{-1}$ mod q (instead of s mod q) where $s = mq + t'\beta$, which can be done using cheap multiplications modulo β. In an NTT computation, the other operand is always a constant twiddle factor Γ, thus the factor β^{-1} can be easily removed using $\Gamma\beta$ mod q as a modified constant. Naïve calculation of Hensel remainder involves performing two double word multiplications, but Seiler [153] proposed an efficient technique that only requires two cheaper single-word multiplications, which independently operates over the high and low words of $\mathbf{s}$. This allowed to vectorize up to 16 integer reductions as compared to just 4 floating point reductions on an AVX2 supported platform, subsequently yielding the fastest NTT on AVX2 (4.2× faster than the fastest floating-point-based NTT of Reference [7]). Subsequently, Lyubashevsky and Seiler [117] proposed another improvement that saved one more multiplication with q^{-1} through precomputing additional modified twiddle factors. However, the same techniques do not apply to the ARM Cortex-M4 platform, since high and low words of the product cannot be separately computed [97].

4.2 Polynomial Inversion

Finding inverses of polynomials in the ring is another crucial operation for NTRU-based schemes, which forms a major bottleneck in the key-generation procedure. A well-known technique to compute inverses is the extension of Euclid's GCD algorithm [92], but it consists of several conditional branches dependent on the input (which is usually secret). Thus, the common technique used in cryptography (e.g., ECC scalar multiplication) is to use Fermat's little theorem, implemented as a chain of multiplications and multi-squarings [24]. NTRU-based schemes require to compute inverses in the ring modulo $q = 3$, $q = 2^k$, and $q \in \mathcal{P}$ and different techniques are used depending on the type of modulus. A standard technique to perform inversion modulo $q = 2^k$ was proposed by Silverman [155], which works by first computing inverse modulo 2 and subsequently a variant of the Newton's iteration to compute modulo 2^k. The first step of inversion modulo 2 is done using Fermat's little theorem and Hülsing et al. [89] identified that it can effectively be computed as a series of bit permutations. While this technique is fast, inversion modulo $q = 3$ or $q \in \mathcal{P}$ is computed using the slower almost-inverse algorithm [92], which involves a series of multiplications and divisions by x until the degree of the input is zero. Recently, Bernstein and Bo-Yin Yang [29] proposed an improvement of the Euclid's gcd algorithm whose AVX2 implementation could yield a 1.7× speed-up compared to the AVX2 implementation of the almost-inverse algorithm in the NTRU-HRSS KEM [89]. This inverse procedure to date is the fastest for a generic prime and has also been used to optimize inversions in implementations of NTRUKEM and NTRUPrimeKEM.

4.3 Discrete Distribution Samplers

Research on efficient polynomial multiplication has matured so much that the sampling/generation of the long polynomials/matrices is now the major bottleneck in most lattice-based schemes. We can broadly classify the sampling operation into two types: (1) sampling uniformly random components in $[0, q]$ and (2) sampling from a small distribution. Sampling in both these cases requires a large pool of randomness, which is usually gained from PRNGs.

4.3.1 Uniform Sampling in $[0, q]$. Rejection sampling is a simple technique used to sample uniformly from a given range $[0, q]$. It involves uniformly sampling in the range $[0, 2^k]$ where $k = \lceil log_2(q) \rceil$ and selecting those less than q as valid samples. This technique though fairly simple, suffers from very high rejection rates $(1 - q/2^k)$, as high as 50% for certain primes. Gueron

and Schlieker [77] proposed to accept samples from a wider range $[0, 5q]$ for $q = 12289$ and subsequently reduce them to $[0, q]$, which brought down the rejection rate from 25% to 6%. On the contrary, for a simple choice of $q = 2^k$ completely eliminates the need for rejection sampling as all samples can be accepted by sampling the appropriate number of bits [100]. This technique is used to sample the public element $\mathbf{A} \in \mathbb{Z}_q^{n \times n}/\mathbf{a} \in \mathbf{R}_q^{k \times \ell}$ in most if not all lattice-based schemes.

4.3.2 Error Sampling. Initial instantiations of the LPREncrypt or the Noisy-DH scheme specified to sample both the secrets and errors from short discrete Gaussian distributions. This subsequently lead to a line of works related to efficient implementations of discrete Gaussian samplers on both HW and SW platforms [143, 145]. Some of the most efficient proposals for discrete Gaussian samplers are the cumulative distribution table (CDT) sampler [130], Bernoulli sampler [63], the Knuth-Yao sampler [103], and discrete Ziggurat sampler [45]. However, the notion of using discrete Gaussian distribution for PKE/KEMs was challenged by Alkim et al. in Reference [10] who argued that high-quality Gaussian distribution is only required for schemes relying on zero-knowledge proofs (identification and digital signature schemes) and subsequently proved that sampling from a simpler CBD with sufficient standard deviation can also yield secure LWE instances for PKE/KEMs. Moreover, Gaussian samplers were also shown to be vulnerable to side-channel and fault attacks by a number of reported works [44, 65, 66]. We refer the reader to Reference [81] for more details on the various Gaussian samplers used in lattice-based schemes. Subsequently, most lattice-based PKE/KEM schemes resorted to sampling from simpler distributions such as CBD, uniform and binary/ternary polynomials with fixed/variable hamming weight. Samples from a CBD with standard deviation $\sigma = \sqrt{k/2}$ can be easily computed as $\sum_{i=0}^{k}(b_i - \acute{b}_i)$ where $b_i, \acute{b}_i$ are uniform independent bits [10, 115]. Naïve rejection sampling is used to sample from short uniform distributions [35]. However, FrodoKEM is the only KEM that still relies on sampling from a rounded Gaussian distribution, which is done using a constant-time inversion sampling based on look-up tables [8].

Though most lattice-based schemes sample from relatively simple distributions, discrete distribution samplers still form the main bottleneck mainly due to a high requirement of randomness for sampling very long polynomials or large matrices. Most NIST candidates have resorted to using NIST standardized constructions, such as SHAKE [30], SHA-256/SHA-512, and AES in counter mode as cryptographically secure PRNGs to generate randomness. In fact, implementations of most schemes spend a majority of time in just operation of the PRNGs [67, 98] and this is especially true in schemes based on standard LWE or MLWE with a very large public parameter $\mathbf{A}$. Several works demonstrate that use of simpler cryptographic constructions such as Trivium stream cipher in Reference [82] and SNEIK lightweight block cipher in Reference [146] or simple statistical randomness sources such as xoshiro-128** PRNG [38] result in several factor improvement in overall speed of the scheme.

5 PHYSICAL ATTACKS ON LATTICE-BASED CRYPTOGRAPHY

There also exists a large body of work done on physical security of lattice-based schemes such as Side-channel Attacks (SCA) and Fault Attacks (FA). The adversary is assumed to have physical access to the device computing the cryptographic algorithm and this attacker model is especially relevant to embedded devices deployed in cyber-physical systems (CPS). In this section, we review the various physical security aspects of lattice-based schemes against side-channel and fault attacks.

5.1 SCA Over NTRU-based Schemes

The polynomial multiplier within NTRU-based schemes has been the main target of SCA over NTRU-based schemes. Lee et al. [109] proposed the first SPA (Simple Power Analysis) and DPA

(Differential Power Analysis) style attacks on the sparse binary polynomial multiplier used in the NTRUEncrypt scheme. The attack mainly targeted the accumulator, which is updated only when the partial product is non-zero, denoting multiplication with $\mathbf{s}[i] = 1$. Three countermeasures were proposed (1) random initialization of the accumulator, (2) additive masking of the secret (with random integer or polynomial), and (3) shuffling the order of access of secret coefficients. The first countermeasure succumbed to second order DPA and second countermeasure was claimed to have weakness against SPA. The authors only considered the combination of the first and third countermeasure to be reasonably secure. Wang et al. [164] subsequently broke Lee et al.'s combined countermeasure using a first-order collision attack. They proposed a new "Random Key Rotation" technique (multiplication of operands with x^i and $x^{(N-i)}$, respectively, in the polynomial ring modulo $(X^n - 1)$) as a potential countermeasure against their attack. Schamberger et al. [151] extended the attack of Lee et al. [109] to sparse trinary secrets and also demonstrated second order CPA attacks over the additive masking (with polynomials) countermeasure. They also evaluated the combination of random key rotation with shuffling countermeasure and found no second order leakage even with 2 million measurements. Huang et al. [88] proposed a range of Vertical CPA (VCPA), Horizontal In-depth CPA (HIDCPA), and Online Template Attacks (OTA) against a school-book polynomial multiplier used in NTRUPrimeKEM, where each coefficient of the product is computed individually. They also analyze the strength of the random initialization countermeasure and shuffling countermeasure, but demonstrate their susceptibility to CISPA (Chosen Input SPA) attacks and show that the countermeasure of additively masking both the ciphertext and the secret thwarted their attacks and hence can considered to be secure.

5.2 SCA Over LWE-based Schemes

5.2.1 SCA Over Polynomial/matrix-vector Multiplier. Primas et al. [136] reported the first attack on the NTT-based polynomial multiplier used in the RLWE-based LPREncrypt scheme implemented on the ARM Cortex-M4 microcontroller. The attack only requires a single trace and works by matching templates with all the modular reduction operations within NTT. Subsequently, the probabilities of intermediates from template matching are combined into an NTT-like graph structure and a belief propagation (BP) algorithm [128] is applied to reveal the secret key. The attack also leveraged upon information from non-constant time DIV operations and required about 100 million templates. Subsequently, Pessl et al. [132] improved the attack that required just 233 templates to attack a constant-time implementation of KyberKEM. The improvements were mainly due to (1) templating based on hamming weights of loads and stores instead of actual value of operands and (2) use of improved belief propagation algorithm. The only concrete countermeasure against this attack is to shuffle the order of butterfly operations within NTT.

Aysu et al. [13] reported standard SPA and DPA style attacks over a HW implementation of a binary schoolbook polynomial multiplier, which mainly targeted the accumulator, which adds up partial products. Their attack uses standard DPA to target the multiplier using an "extend and prune" strategy, wherein the $(i + 1)$th coefficient is recovered assuming all previous i coefficients are retrieved/known. They proposed random initialization of the accumulator as a potential countermeasure against first order DPA, but demonstrated its susceptibility to second order DPA. Aysu et al. [15] subsequently proposed single trace horizontal attacks on generic schoolbook multipliers used within HW designs of the NewHopeUSENIX and FrodoCCS KEX schemes on FPGAs. Their attack, however, utilizes "horizontal" DPA and the same "extend and prune" strategy in a greedy fashion, resulting in a full key-recovery using just a single trace over FrodoCCS. Bos et al. [37] investigated feasibility of similar single trace attacks over software implementations of the FrodoCCS KEX scheme. They rely on matching template of the accumulator values instead of DPA and subsequently apply two variants of the "Extend and Prune" strategy: (1) greedy pruning

(always pick single best candidate for each coefficient) and (2) relaxed pruning with b candidates (Selecting b best candidates for each coefficient). They showed that relaxed pruning with a large "b" results in better success rates across all parameter sets of FrodoCCS and FrodoKEM. They also draw an interesting inference that schemes working in larger dimensions (theoretically more secure) are more susceptible to single trace attacks than those working in smaller dimensions. Standard countermeasures such as shuffling the order of accumulation and insertion of dummy operations protect against such attacks. Zijlstra et al. [165] perform a comprehensive evaluation of the cost of a range of SCA countermeasures for the polynomial multiplier in FPGAs. They propose a number of countermeasures, including (1) low-cost novel masker decoder (20% lesser area compared to Reference [142]) and (2) redundant representation of secret coefficients as masking countermeasure (adding random multiples of modulus q as masks). They also implement known countermeasures such as (1) random key rotation [164], (2) blinding countermeasure proposed by Saarinen et al. [147] (multiplication of both operands with random integers), (3) shuffling of the order of butterfly operations within NTT.

5.2.2 Masking LWE-based Schemes. Reparaz et al. [142] proposed the first masking scheme for the decryption procedure of the RLWE-based LPREncrypt scheme. The idea was to additively split the secret key into two shares and perform the decryption procedure separately with the two shares, which are later combined using a novel bespoked masked decoder to retrieve the key bit. Reparaz et al. [141] subsequently proposed a simpler masking approach exploiting the additively homomorphic property of the LPREncrypt scheme, i.e., $\text{Decrypt}(\mathbf{c}_1 + \mathbf{c}'_1, \mathbf{c}_2 + \mathbf{c}'_2) = \text{Decrypt}(\mathbf{c}_1, \mathbf{c}_2) \oplus \text{Decrypt}(\mathbf{c}'_1, \mathbf{c}'_2)$. The ciphertext is masked in the decryption procedure using another ciphertext corresponding to a random message m_r. This masked ciphertext is subsequently decrypted to a masked message, which is xored with m_r to retrieve the correct message. But, both the masking schemes incur a heavy performance penalty and suffer from high decryption failures, which also do not make them suitable for IND-CCA security. Subsequently, Oder et al. [126] proposed an improved holistic masking scheme with negligible decryption failure rate for the RLWE-based LPREncrypt IND-CCA secure PKE scheme. The central idea of their masking scheme is to encrypt two message shares (m, m') separately into two ciphertexts, which can be separately decrypted and subsequently combined to retrieve the message $m \oplus m'$. They also propose a masked decryption procedure with a perfectly correct masked decoder, which performs an arithmetic to boolean masking before decoding. The polynomial multiplier is additionally protected with the random key rotation and multiplicative masking countermeasure [147]. Multiple other operations identified as vulnerable such as the binomial sampler, XOFs and the final ciphertext comparator were also masked. The resulting masked decryption procedure incurred a heavy performance penalty of about 5.7 times compared to the unmasked case on the ARM Cortex-M4 microcontroller.

5.2.3 SCA Over Error Correcting Codes. Some recent works have shown that error correcting codes implemented within LWE-schemes contain exploitable SCA vulnerabilities. D'Anvers et al. [64] demonstrated practical key-recovery attacks against the IND-CCA secure LACKEM scheme by utilizing difference in execution times of error correcting procedures based on the validity of the codewords. They showed that the timing leakage can be used to instantiate a *plaintext checking* oracle and a chosen ciphertext attack can lead to full key recovery. Ravi et al. [139] subsequently reported similar side-channel assisted chosen-ciphertext attacks over several LWE/R-based NIST candidates. They could get information about validity of codewords for chosen ciphertexts from EM-leakage leading to key-recovery over Round5KEM and LACKEM schemes. They also extended their attack to schemes that do not use ECCs by utilizing leakage about the output of the encapsulated IND-CPA secure decryption procedure from the FO transformation. They claim that the only concrete countermeasure against their attack is to mask all

the vulnerable operations [126], but masking schemes for ECC is currently not known and thus creates an interesting research direction.

5.3 FA Over NTRU-based Schemes

Kamal and Youssef [93] proposed the first Differential Fault Analysis (DFA) of the NTRUEncrypt scheme, which works by faulting some coefficients (in unknown locations) of the input to the final multiplication in its decryption procedure. The attacker compares the faulty output from the correct output to retrieve the secret key. This attack works with success probability $(1 - 1/p)$ where NTRUEncrypt is instantiated with parameters (n, p, q). The attacker assumes that he can get direct access to decryption output and hence this attack does not work in an IND-CCA secure setting. Subsequently, the same authors reported a protected implementation of the NTRUEncrypt scheme against their attack using redundancy and checksum-based countermeasures [95].

5.4 FA Over LWE/R-based Schemes

Valencia et al. [162] perform a fault vulnerability analysis of RLWE-based PKE schemes for key/message recovery based on various fault models such as bit flips, zeroing and instruction skips. However, most of their proposed attacks required high number of faults assuming very powerful fault models. However, Ravi et al. [138] performed the first practical electromagnetic injection fault attack on LWE-based PKE/KEMs inducing a very small number of faults that resulted in a scenario where $\mathbf{s} = \mathbf{e}$ resulting in trivially solvable LWE instances leading to key/message recovery. Thus, a simple countermeasure against their attack would be to check the equality of the secret $\mathbf{s}$ and error $\mathbf{e}$. Howe et al. [80] proposed to perform a number of statistical tests such as t-tests and chi-squared tests to test the validity of the generated secrets and errors. They also reported an efficient hardware implementation of a protected CDT sampler with reasonable performance overhead. Fritzmann et al. [72] reported an implementation of a fault-attack resistant NTT accelerator protected using the Dual Modular Redundancy (DMR) countermeasure. They perform simulated fault injection and show that the DMR countermeasure enables to achieve fault detection rates as high as 80%. Arpan et al. [90] presented SPQCop, the first side-channel protected lattice-based cryptoprocessor on the Zynq 7000 SoC and Zynq UltraScale+ FPGA integrated with a number of low-cost fault propagation and DMR countermeasures to protect critical signals and logic circuitry against fault attacks, coupled with SCA countermeasures such as instruction shuffling and random delays, which only introduced an additional 5% increase in resource utilization.

6 RESULTS AND COMPARISON

In this section, we attempt to perform a comparative evaluation of speed, efficiency and bandwidth performance of the second round NIST candidates on both hardware (HW) and software (SW) platforms. NIST recommended the following platforms for fair evaluation of the candidates [5]: (1) High End SW: Intel x64 running Windows or Linux; (2) Embedded SW: ARM Cortex-M4 32-bit microcontroller; and (3) Embedded HW: Artix-7 FPGA.

There are a number of notable benchmarking efforts for implementation of PQC schemes. pqm4 [98] is one such benchmarking and testing framework containing state-of-the-art implementations of PQC schemes on the ARM Cortex-M4 microcontroller. Open Quantum Safe (OQS) [121] is another project with C implementations of PQC schemes whose main aim is to prototype and subsequently enable integration of PQC schemes into practical protocols and applications such as TLS and IPSec. It is worth noting that OpenOQS has integrated PQC primitives into a post-quantum secure fork of the OpenSSL library [157]. eBATS (ECRYPT Benchmarking of Asymmetric Systems) is another well-known benchmarking platform for public-key cryptographic schemes, which now also includes results for efficient implementations of PQC schemes benchmarked

on a variety of high end software platforms [27]. In-spite of existence of several benchmarking frameworks, we identify some reasons why it is too early to directly rank the schemes based on performance: (1) Benchmarking is an ongoing exercise and hence not all schemes are fully optimized for the considered implementations platforms; (2) most schemes spend a significant amount of time in generating randomness using different techniques and there is no agreed upon common technique used by all schemes; and (3) certain optimization techniques though applicable for multiple schemes, have not yet been utilized in implementations of all the relevant schemes.

6.1 Analysis of Communication Bandwidth

We perform a comparative evaluation of the bandwidth performance based on the parameter set table in Table 2. We only focus on public-key and ciphertext sizes, since the secret key is not exchanged/communicated during the protocol. As shown in practical experiments of PQC schemes over TLS [96, 127], communication bandwidth serves as the main bottleneck over communication links where packet loss rates are just more than 3–5%. When comparing LWE and LWR, LWR in general offers better sizes owing to the use of rounded keys and ciphertexts. We can also see that the standard LWE-based FrodoKEM has an order of magnitude higher public-key and ciphertext sizes compared to the structured lattice-based schemes; however, the standard GLWR variant of the Round5PKE scheme has much lower bandwidth (by $\approx 1.3 - 1.6\times$) owing to the use of sparse ternary secrets and rounded public-key and ciphertext components. Bandwidth requirements for all structured lattice-based schemes are similar ranging from 0.6–2.2 KB across all security levels. Among them, Round5PKE parameters based on RLWR offer the most compact keys and ciphertexts, while all other structured lattice-based schemes also offer very similar sizes for equivalent security. Round5PKE is followed by SaberKEM (MLWR) and LACKEM (RLWE). LACKEM offers very compact sizes mainly due to the use of a byte-size modulus. KyberKEM (MLWE) and ThreeBearsKEM (I-MLWE) offer the next best key and ciphertext sizes (although very comparable to SaberKEM). NewHopeKEM based on RLWE offers the largest sizes among structured lattice-based schemes at NIST security level 5, mainly due to large n. As far as NTRU-based schemes are concerned, they offer slightly larger sizes compared to structured LWE/R-based schemes.

6.2 Implementation Results on Software Platforms

In this discussion, we review some recent results from implementation of lattice-based schemes on the NIST recommended embedded software platform, the ARM Cortex-M4 microcontroller. Karmakar et al. [100] proposed a speed optimized implementation of SaberKEM on the ARM Cortex-M4 microcontroller through SIMD accelerated Karatsuba and Toom-Cook polynomial multiplication. Their implementation also adopted a "just-in-time" approach to generate the public parameter ($\mathbf{a}$ with k^2 polynomials) to reduce memory usage by generating and multiplying $\mathbf{a}$ in parts. Subsequently, Kannwischer et al. [97] proposed the most optimized polynomial multiplication routines in the ring $\mathbb{Z}_q[x]$ with $q = 2^k$ on the ARM Cortex-M4 (applicable to SaberKEM and NTRUKEM), which improved speeds of SaberKEM by about 20%. The fastest implementation of Round5KEM was proposed by Saarinen et al. [149] through use of techniques such as (1) sparse ternary polynomial multiplication only using additions and subtractions and (2) efficient sampling of sparse ternary polynomial with reduced rejection rates. Botros et al. [39] proposed a high-speed and memory-optimized implementation of KyberKEM on the ARM Cortex-M4 utilizing techniques such as (1) merge multiple levels of NTT to reduce number of loads and stores, (2) utilization of SIMD instructions, and (3) "just-in-time" approach for multiplication with public parameter $\mathbf{a}$.

Howe et al. [83] proposed an efficient implementation of FrodoKEM (only for parameters with NIST security levels 1 and 3) on the ARM Cortex-M4, which addressed the main challenge of memory usage given the very large matrices used in FrodoKEM. This was achieved using techniques

Table 3. Performance Evaluation of the Second Round Lattice-based KEMs on the ARM Cortex-M4 Microcontroller Obtained from State-of-the-Implementations in the *pqm4* Library [98]

Scheme	Parameter Set	Impl.	KeyGen		Encaps		Decaps	
			Cycles ($\times 10^3$)	Stack (bytes)	Cycles ($\times 10^3$)	Stack (bytes)	Cycles ($\times 10^3$)	Stack (bytes)
NewHope	NewHope1024 (5)	m4	1,162	2,852	1,696	5052	1,620	5,052
LAC	LAC – 192 – v3a	opt	1,377	14,620	2,215	17,492	3,030	20,220
Round5	R5ND_3KEMCCA_5d	m4	723	5,620	1,082	6,212	1,359	7,148
	R5N1_3KEMCCA_0d	m4	722	6,132	1,067	6,652	1,296	7,756
Kyber	Kyber768	m4	893	3,284	1,050	2,980	987	3,012
Saber	Saber	m4	895	13,244	1,162	15,524	1,204	16,620
ThreeBears	MamaBear (4)	opt	1,274	3,580	1,493	3,452	2,125	6,060
NTRU	ntruhps4096821	m4	211,767	34,520	1,218	24,912	1,066	23,968
NTRUPrime	sntrup761	m4	827	28,444	1,311	34,732	1,494	39,692
	ntrulpr761	m4	11,350	28,671	789	28,604	742	27,756
Frodo	FrodoKEM – 640 (1)	m4	79,332	26,596	79,746	51,972	79,227	72,596
	FrodoKEM – 976	m4	187,071	33,800	253,736	57,968	254,195	58,112

All the considered parameter sets in the table offer NIST security level 3. Cycles are specified in units of $\times 10^3$ and stack usage in bytes.

such as (1) "just-in-time" approach combined with on-the-fly computations to optimize stack usage for matrix-matrix multiplication and (2) efficient usage of available registers for reduced loads and stores. Subsequently, Bos et al. [38] improved the speed of the FrodoKEM-640 variant on the ARM Cortex-M4 by about 30–40% through better memory access techniques coupled with utilization of SIMD instructions for matrix-vector multiplication. They also showed that using AES as PRNG doubles the speed, while use of a SR-PRNG xoshiro-128** [32] improved the speed by about 4×. This clearly demonstrated that the sampling operation is the bottleneck in case of FrodoKEM. The most optimized implementation of NewHopeKEM on the ARM Cortex-M4 is based on the optimized implementation of NewHopeUSENIX by Alkim et al. [11] with full assembly-based optimized implementation of the NTT transform utilizing techniques such as unrolling and merging multiple layers of the NTT operation. State-of-the-art implementations of other schemes on the ARM Cortex-M4 are just adaptations of the reference implementations of the respective schemes.

6.2.1 Comparative Evaluation on ARM Cortex-M4. We compare the two main performance parameters on the ARM Cortex-M4 microcontroller (1) speed and (2) stack utilization, in the same order. For a fair, simple and objective comparison, we compare the performance of schemes only based on parameters that offer NIST level 3 security. This choice is motivated by the following reasons: (1) parameter sets with NIST level 3 security offer good balance between security (Level 5) and efficiency (Level 1) and hence appear to be the most widely usable, unless either security or efficiency is prioritized and (2) schemes should compare in a similar way in other security levels also, barring a few discrepancies. Since NewHopeKEM and ThreeBearsKEM do not offer parameter sets for level 3 security, we conservatively utilize numbers from NewHope1024 (level 5) and MamaBear (level 4), respectively. Please refer to Table 3 for the speed and stack usage of the schemes on the ARM Cortex-M4 microcontroller. Apart from FrodoKEM – 976 whose numbers are taken from Reference [83], all others are taken from implementations in the *pqm4* library The library contains three types of implementations: (1) ref—reference implementation in C, (2) opt—optimized in C, and (3) m4—optimized with Cortex-M4 assembly instructions.

With respect to KeyGen, StreamlinedNTRUPrimeKEM is the slowest among all lattice-based schemes primarily due to costly polynomial inversions and invertibility tests, but the performance gap might not reflect an accurate picture, since it is only a reference implementation (unoptimized). Moreover, KeyGen of NTRULPRimeKEM is 10× faster than StreamlinedNTRUPrimeKEM and 3× faster than NTRUKEM (even on using optimized routine from Reference [97]), since it generates an LWR-like instance devoid of any inversions or invertibility checks. RLWR-based Round5KEM provides the fastest KeyGen routine followed by KyberKEM and SaberKEM, which also have comparable runtimes. NewHopeKEM at security level 5 is about 5× slower than Round5KEM mainly due to its larger degree. As expected, KeyGen in Frodo is one to two orders of magnitude slower than structured lattice-based schemes, but it is notable that standard LWR-based variant of Round5KEM is comparatively much faster (≈ 9×). With respect to Encaps and Decaps, KyberKEM, SaberKEM along with NTRUKEM offer the best speeds compared to the other schemes. In particular, KyberKEM and SaberKEM provide the best times for Encaps, while NTRUKEM has the fastest time for Decaps, mainly because it does not compute re-encryption for IND-CCA security. Round5KEM and ThreeBearsKEM have the next best runtimes with costlier Decaps routines. In particular, Decaps of RLWR-based Round5KEM in particular is about 1.7× slower than NTRUKEM. Numbers for both the variants of NTRUPrimeKEM are an order of magnitude higher, but should provide much better runtimes with dedicated optimizations. FrodoKEM has the slowest speeds for Encaps and Decaps (68–70× worse than NTRUKEM) but the standard LWR variant of Round5KEM is about 10× faster than FrodoKEM.

With respect to stack usage, KyberKEM consumes the least stack space (≈ 3 KB) among all the NIST candidates owing to the use of memory optimized in-place NTT computation [39]. Encaps of ThreeBears has comparable stack usage, but stack usage doubles for Decaps, probably due to the decoding procedure of Melas BCH ECC. RLWR-based Round5KEM and LACKEM offer the next best stack usage numbers (though 2.3–2.8× worse than KyberKEM). It is interesting to note that stack usage of LACKEM for Encaps and Decaps is almost twice as that of KeyGen, probably due to the usage of BCH error correcting code. SaberKEM though fast, consumes almost 5× more stack space than KyberKEM mainly due to the use of the Karatsuba and Toom-Cook algorithm for polynomial multiplication. Stack utilization of NewHopeKEM is also high (5–6× worse than Encaps and Decaps of KyberKEM) mainly owing to the use of a very large degree polynomial ($n = 1{,}024$). The stack usage of StreamlinedNTRUPrimeKEM is comparable with SaberKEM (16–19 KB), while that of NTRULPRimeKEM and NTRUKEM are much higher (22–26 KB). Finally, as expected, FrodoKEM consumes the largest stack space among all the considered NIST candidates owing to the use of very large matrices and vectors. In fact, FrodoKEM at security level 5 does not even fit on the ARM Cortex-M4.

6.2.2 Comparative Evaluation on Intel x-64. We also review results on the high-end Intel-x64 CPUs in Table 4. Most schemes have reported numbers for AVX2 vectorized implementations, while some have optimized non-vectorized implementations, and others only have reference implementations. For a simple and objective comparison, we only tabulate implementation numbers for parameter sets with NIST level 3 security, unless specified otherwise. We only focus on stark differences from results on ARM Cortex-M4 (Table 3). In that respect, we can see that NTRULPRimeKEM and LACKEM offer the fastest time for KeyGen among all the schemes, followed by KyberKEM, ThreeBearsKEM, RLWR-based Round5PKE, SaberKEM, and NewHopeKEM (level 5) in the same order. Moreover, the performance gap between runtime of KeyGen of NTRUKEM and StreamlinedNTRUPrimeKEM compared with other structured lattice-based schemes has been drastically reduced owing to the use of fast modular inversion of Reference [29] and AVX2 accelerated multiplications. Moreover, optimized implementations

Table 4. Comparative Performance Evaluation of the Second Round Lattice-based PKE/KEMs on the Intel x64 Platform

Scheme	Parameter	Device	Impl.	Time (kcycles)		
				KeyGen	Encaps	Decaps
NewHope	NewHope1024 (5)	Core i7-4770K (Haswell)	AVX2	132	212	220
LAC	LAC – 192 – v3a	Core i7-4770S (Haswell)	AVX2	29	46	64
Round5	R5ND_3KEM_5d	Core i7	opt	72	119	171
	R5N1_3KEM_0d	Core i7	AVX2	4,181	4,585	4,816
Kyber	Kyber768	Core i7-4770K (Haswell)	AVX2	62	83	70
Saber	Saber	Core i7-6600U	AVX2	104	122	120
ThreeBears	MamaBear (4)	Core i3-6100U (Skylake)	opt	79	96	156
NTRU	ntruhps4096821	Core i7-4770K (Haswell)	ref.	31,835	1,856	4,920
	ntruhps2048677 (1)	NA	AVX2	277	35	69
NTRUPrime	sntrup761	Xeon E3-1220 v3 (Haswell)	opt	810	49	59
NTRUPrime	ntrulpr761	Xeon E3-1220 v3 (Haswell)	opt	44	72	86
Frodo	FrodoKEM – 976	Core i7-6700 (Skylake)	AVX2	8,579	9,302	9,143
NTTRU (Post-NIST)	NTTRU (1)	Core i7-6600U (Skylake)	AVX2	6.4	6.1	7.9

All the considered parameter sets in the table offer NIST security level 3, unless specified otherwise. Cycles are specified in units of $\times 10^3$.

of NTRUPrimeKEM provide one of the best runtimes for Encaps and Decaps routines, followed by AVX2 accelerated LACKEM and KyberKEM. ThreeBearsKEM also provides very comparable runtimes for KeyGen and Encaps but a slower ($\approx 2\times$) Decaps routine (possibly due to Melas BCH decoder), similar to LACKEM with the heavy BCH decoder. Moreover, better numbers for NewHopeKEM can be expected across all three procedures with the use of the fastest AVX2 NTT implementation of Reference [153].

6.3 Implementation Results on FPGA and ASICs

In this section, we will review some recent and relevant works on fully HW and HW/SW co-designed implementations of lattice-based schemes and its sub-blocks on FPGAs, ASICs, and heterogenous SoCs. Oder et al. [125] reported the first HW design for the IND-CPA secure NewHopeSIMPLE PKE scheme, on the Artix-7 FPGA with the main design objective of low-area utilization. While this implementation worked with a sequential NTT module, Kuo et al. [104] subsequently proposed a faster design of the IND-CPA secure NewHopeUSENIX KEX scheme utilizing a highly parallel NTT module with four parallel butterfly operators, which yielded a 19× speedup, but at the cost of 4× higher area utilization on the same FPGA. Arpan et al. [90] presented SPQCOP, the first side-channel resistant fully HW-based cryptoprocessor for NewHopeUSENIX, NewHopeSIMPLE and Hila5 on Zynq 7000 SoC and Zynq UltraScale+ FPGA. Their design is heavily pipelined and also maximally utilizes the available DSPs and BRAMs to yield high frequency and high performance designs. Though they were able to achieve 1.4× higher frequency than Reference [104] for NewHopeUSENIX on the Zynq 7000 SoC, they were 2.2× slower than Reference [104] due to the high latency incurred due to heavy pipelining. Howe et al. [83] proposed the first HW implementation of FrodoKEM on the Artix-7 FPGA. Their design was based on an efficient kernel for the matrix-vector multiplier, which loops over the individual rows and uses a "just-in-time" approach to perform matrix-matrix multiplication. There are no existing dedicated

FPGA implementations for the other lattice-based NIST candidates. Banerjee et al. [20] proposed "Sapphire," the first and only ASIC-based crypto processor for several lattice-based KEMs such as NewHopeKEM, FrodoKEM and KyberKEM, with the main design objective of energy efficiency. Their design is based on (1) configurable NTT module based on the constant-geometry NTT architecture working with multiple single port RAM modules enabling single cycle butterfly operations and (2) configurable discrete distribution sampler.

There were a number of works that focussed on accelerating certain sub-blocks within lattice-based schemes. Fritzmann et al. [72] proposed the first HW/SW co-design for lattice-based PKE schemes with a dedicated NTT and SHAKE accelerator hardened against fault attacks using the DMR countermeasure for the RISC-V architecture. They observe a 13× improvement in the total run time of IND-CPA secure NewHopeKEM PKE scheme compared to a pure SW implementation on RISC-V. Fritzmann et al. [71] designed a power efficient NTT on ASIC using optimization techniques such as clock-gating and operand-isolation and observe about 33–35% reduction in power consumption of the NTT operation. Song et al. [156] subsequently reported an implementation of a high performance and highly configurable accelerator based on highly parallelized NTT operations prototyped in a 2.05 mm^2 40 nm CMOS ASIC test chip.

Basu et al. [22] utilized a High-level Synthesis (HLS) framework to perform HW benchmarking of the several first round NIST candidates such as FrodoKEM, NewHopeKEM, SaberKEM, and NTRUHRSS KEM on both FPGA and ASIC. We do not go into specific details, but highlight a few takeaways from their exercise: (1) At security levels 1 and 3, KyberKEM and SaberKEM on FPGA are ideal for low latency applications on FPGAs; (2) at security level 1, NTRUHRSSKEM offers the best latency-power tradeoff on FPGA; and (3) at security level 1, NTRUHRSSKEM and NewHopeKEM offer the lowest latencies and FrodoKEM consumes the least power on ASICs.

Farahmand et al. [68] reported HW/SW co-designed implementations for Round 1 NTRU-based candidates such as NTRUEncrypt, NTRU – HRSS, StreamlinedNTRUPrimeKEM and NTRULPRimeKEM on the Zynq UltraScale+ FPGA. They only offloaded the polynomial multiplication to HW and implemented them using fast LFSR structures [112, 113]. Subsequently, the same authors also performed detailed benchmarking of HW/SW co-designed implementations of about five Round 2 NIST candidates, which include FrodoKEM, SaberKEM, Round5PKE, NTRUKEM, and NTRUPrime on the same platform [67]. However, they offloaded multiple operations to HW chosen based on two criteria: (1) Operations that consume more than 90% of the total time and (2) operations that can be highly parallelized in HW. They evaluated the performance of different schemes and propose concrete rankings for the same. For encapsulation: SaberKEM, Round5PKE, NTRUKEM, StreamlinedNTRUPrimeKEM, FrodoKEM, and NTRULPRimeKEM. For decapsulation: SaberKEM, Round5PKE, NTRUKEM, StreamlinedNTRUPrimeKEM, NTRULPRimeKEM, and FrodoKEM. They also observed that FrodoKEM consumed the least resources (FF and LUTs on FPGA) among all the schemes, owing to its simpler design. The new bottleneck in most schemes after the HW/SW partitioning is either *hashing* or generating randomness through a custom randombytes() function, with each scheme utilizing varying proportions of the same. Thus, these rankings are not absolute and could possibly be subjected to changes if different PRNGs are used. We do not perform a comparative evaluation of the HW results mainly due to the diversity in the reported implementations, which were done on different target platforms aimed to satisfy different design objectives. Moreover, factors such as use of different operations for randomness generation does not offer a level ground for fair comparison of the schemes.

7 OVERVIEW AND GENERAL RECOMMENDATIONS

In this section, we perform a high-level comparative evaluation of lattice-based PKE/KEMs depending upon security and performance from reported state-of-the-art implementations on

various platforms. We also briefly state advantages and disadvantages of lattice-based PKE/KEMs and compare them against schemes from other types of post-quantum cryptography.

Security: First, unstructured lattice-based schemes have stronger theoretical security guarantees compared to structured lattice-based schemes. Thus, standard LWE-based FrodoKEM and variants of Round5KEM based on standard-LWR could be utilized for very sensitive applications requiring long term-security. Among structured lattice-based schemes, module lattice-based schemes (KyberKEM and SaberKEM) seem to possess lesser algebraic structure and hence more confidence in security than ideal lattice-based schemes (NewHopeKEM and LACKEM). However, when comparing NTRU against LWE/R, there is no definitive distinction in terms of security, but one could choose to be conservative by going for schemes based on polynomials rings with less algebraic structure. In that respect, NTRUPrimeKEM and RLWR-based Round5KEM are based on less structured non-cyclotomic rings compared to the prime cyclotomic rings used in NTRUKEM or power of 2 cyclotomic rings with max-split modulus in NewHopeKEM and KyberKEM and min-split modulus in SaberKEM and LACKEM. In fact, ThreeBearsKEM also relies on a non-cyclotomic polynomial but is based on a very recent variant of I-MLWE and hence requires more cryptanalytic efforts.

Performance: It is very clear that performance of structured lattice-based schemes is far more superior compared to unstrucutred lattice-based schemes, which is also confirmed by experimental results from practical deployment of lattice-based schemes in TLS [40, 105, 106]. However, it is much harder to make a general distinction between structured lattice-based schemes just on the basis of performance, as different platforms seem to favour different schemes. For example, NTRU-based schemes seem to perform very well on AVX2 platforms barring the KeyGen procedure, which involves costly inversions and invertibility tests. Thus, they might be very well be suitable for static IND-CCA PKE/KEMs on high-end servers where the key-generation costs are amortized. In general, schemes such as Round5KEM, SaberKEM, and KyberKEM provide high performance across all embedded hardware and software platforms. Moreover, the scalable security feature also comes as an added advantage for module lattice-based schemes such as KyberKEM and SaberKEM. While NewHopeKEM does not compare well in security level 3, it has competitive parameter settings in security level 5. If I-MLWE instills good confidence in its long term security, then ThreeBears also offers a very attractive alternative in terms of migration, since it can be adapted to work with existing big-integer arithmetic HW and SW modules within smart cards and crypto accelerators. As for FrodoKEM, it yields very low footprint in HW designs owing to its very simple design and hence could be suitable for low speed and resource constrained HW platforms [22, 67].

Advantages:

- Lattice-based cryptography offers a wide range of PKE/KEM/KEXs with varying trade-offs between security and efficiency guarantees, i.e., plain LWE, R/MLWE, ILWE and NTRU.
- Theoretical security guarantees of lattice-based schemes are well studied and in-fact, concrete security estimates of lattice-based schemes are very conservative keeping in mind the scope for improved cryptanalysis in the future (refer to Section 3.7).
- Practical performance of lattice-based schemes are well studied on a variety of hardware and software platforms and are second to none when compared against other post-quantum cryptographic schemes. In fact, some variants of NTRU are standardized [163] and a few others lattice-based schemes have also been deployed as part of real-world test trials in secure protocols such as the TLS [40, 105, 106].
- Physical attacks and countermeasures over lattice-based schemes are also very well studied when compared to other post-quantum cryptographic schemes.

Disadvantages:

- There is an unknown security gap in lattice-based cryptography between structured and unstructured lattice-based schemes. Though it appears to be less likely to be bridged in the near future [55], improved attacks exploiting algebraic structures could significantly impact security of structured lattice-based schemes, given that the best-known attacks against several schemes do not currently exploit any algebraic structure.
- A study by Brendel et al. [43] showed that lattice-based schemes cannot come in as simple drop-in-replacements to classical "symmeteric" Diffie-Hellman style key-exchange schemes due to the inherent asymmetry in IND-CCA secure lattice-based PKE/KEMs. Such issues related to practical deployment requires more attention.
- Passively secure lattice-based PKE/KEMs are vulnerable to chosen ciphertext attacks in the presence of a *plaintext checking* oracle [61, 69] and such attacks have also been extended to IND-CCA secure schemes through exploitation of side-channel information [64, 139]. These works demonstrate easy susceptibility of lattice-based PKE/KEMs to side-channel attacks thus highlighting the need for concrete and efficient masking countermeasures.

Comparison with other post-quantum PKE/KEMs: Apart from lattice-based PKE/KEMs, there are seven other candidate PKE/KEMs from code-based cryptography and one candidate from Supersingular Isogeny-based cryptography (SI) in the second round of the NIST standardization process. SIKE is the only SI-based IND-CCA secure KEM in the NIST process. It offers the lowest key and ciphertext sizes among all post-quantum PKE/KEMs (100–500 bytes). But, the speed of SIKE is about 1 to 2 orders of magnitude worse than all lattice-based PKE/KEMs across several implementation platforms. Practical experiments in Reference [106] showed that TLS connections using SIKE for key exchanges were only performing better than structured lattice-based schemes for the slowest 5% of the connections marred by high packet loss rate, clearly demonstrating that the slow performance of SIKE clearly outweighs the advantages of small key and ciphertext sizes in the context of TLS.

There are two broad classes of code-based KEMs: (1) classical McEliece or Niederreiter schemes based on binary Goppa codes and (2) schemes based on more structured quasi-cyclic (QC) codes. ClassicMcEliece and NTS-KEM are two candidates based on the more conservative binary Goppa codes characterized by very large public-key sizes (200 KB up to 1.5 MB) and very slow key generation times (2 orders of magnitude slower than FrodoKEM on AVX2). However, they offer very small ciphertext sizes (comparable with SIKE) and much faster encapsulation/decapsulation procedures that are comparable but still slower than structured lattice-based schemes. There are five other schemes based on the the less conservative and structured quasi-cyclic codes, which compare much better in terms of performance against lattice-based schemes. In particular, rank-metric-based schemes such as ROLLO built upon quasi cyclic low rank parity check codes (QC-LRPC codes) offers the smallest key and ciphertext sizes (comparable with SIKE) followed by other schemes such as RQC, LEDAcrypt (based on QC-LDPC), and BIKE (based on QC-MDPC), which offer key and ciphertext sizes that compare very favorably with structured lattice-based schemes such as KyberKEM, SaberKEM, and Round5PKE. With respect to performance (based on numbers from AVX2 implementations), these structured code-based schemes have comparable yet slightly slower performance numbers compared to the aforementioned structured lattice-based schemes. Moreover, structured lattice-based schemes currently have very optimized implementations with superior numbers on the embedded ARM Cortex-M4 compared to code-based schemes [98].

8 CONCLUSION

We present a detailed survey of post-quantum secure lattice-based key-sharing schemes categorized into PKE, KEX, and KEM schemes. We have covered various aspects starting from the underlying hardness guarantees, generic algorithmic frameworks, building blocks of implementations of lattice-based schemes, security against physical attacks, and a comparative evaluation of the different lattice-based schemes on a variety of hardware and software implementation platforms with main focus on the schemes competing in the second round of the NIST standardization process. The field of lattice-based cryptography has matured immensely with respect to all the aforementioned aspects and given the current state-of-the-art results, lattice-based cryptographic schemes offer a very attractive alternative for public-key cryptography in the era of quantum computers. Please note that our work is focussed on the results available from the second round of the NIST's standardization process while the standardization process has progressed to the third and final round at the time of publishing this document.

REFERENCES

[1] Eric Rescorla. 2015. The Transport Layer Security (TLS) Protocol Version 1.3 draft-ietf-tls-tls13-07. Retrieved from https://tools.ietf.org/html/draft-ietf-tls-tls13-07.

[2] Eric Rescorla. 2016. The Transport Layer Security (TLS) Protocol Version 1.3 draft-ietf-tls-tls13-13. Retrieved from https://tools.ietf.org/html/draft-ietf-tls-tls13-13.

[3] Divesh Aggarwal, Daniel Dadush, Oded Regev, and Noah Stephens-Davidowitz. 2015. Solving the shortest vector problem in 2 n time using discrete Gaussian sampling. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing*. ACM, 733–742.

[4] Miklós Ajtai. 1996. Generating hard instances of lattice problems. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. ACM, 99–108.

[5] Gorjan Alagic, Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta et al. 2019. *Status Report on the First Round of the NIST Post-quantum Cryptography Standardization Process*. U.S. Department of Commerce, National Institute of Standards and Technology.

[6] Martin R. Albrecht, Benjamin R. Curtis, Amit Deo, Alex Davidson, Rachel Player, Eamonn W. Postlethwaite, Fernando Virdia, and Thomas Wunderer. 2018. Estimate all the {LWE, NTRU} schemes! In *Proceedings of the International Conference on Security and Cryptography for Networks*. Springer, 351–367.

[7] Erdem Alkim, Roberto Avanzi, Joppe W. Bos, Leo Ducas, Antonio de la Piedra, Thomas Poppelmann, Peter Schwabe, and Douglas Stebila [n.d.]. NewHope (Version 1.1): Algorithm specifications and supporting documentation. Retrieved from https://newhopecrypto.org/data/NewHope_2020_04_10.pdf.

[8] Erdem Alkim, Joppe W. Bos, Leo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila [n.d.]. Frodo: Algorithm specifications and supporting documentation. Retrieved from https://frodokem.org/files/FrodoKEM-specification-20200325.pdf.

[9] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Newhope without reconciliation. *IACR ePrint*. Retrieved from https://eprint.iacr.org/2016/1157.

[10] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. 2016. Post-quantum key exchange—A new hope. In *Proceedings of the USENIX Security Symposium*. 327–343.

[11] 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*. Springer, 332–349.

[12] Roberto Avanzi, Joppe Bos, Leo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé [n.d.]. CRYSTALS-Kyber (version 2.0) - Algorithm specifications and supporting documentation. Retrieved from https://pq-crystals.org/kyber/data/kyber-specification-round2.pdf.

[13] Aydin Aysu, Michael Orshansky, and Mohit Tiwari. 2018. Binary Ring-LWE hardware with power side-channel countermeasures. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'18)*. IEEE, 1253–1258.

[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] Aydin Aysu, Youssef Tobah, Mohit Tiwari, Andreas Gerstlauer, and Michael Orshansky. 2018. Horizontal side-channel vulnerabilities of post-quantum key-exchange protocols. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST'18)*. IEEE, 81–88.

[16] Hayo Baan, Sauvik Bhattacharya, Scott Fluhrer, Oscar Garcia-Morchon Garcia-Morchon, Thijs Laarhoven, Rachel Player, Ronald Rietman, Markku-Juhani O. Saarinen, Ludo Tolhuizen, Jos'e Luis Torre-Arce, and Zhenfei Zhang. [n.d.]. Round5: Algorithm specifications and supporting documentation. Retrieved from https://round5.org/doc/Round5_Submission042020.pdf.

[17] Ciprian Băetu, F. Betül Durak, Loïs Huguenin-Dumittan, Abdullah Talayhan, and Serge Vaudenay. 2019. Misuse attacks on post-quantum cryptosystems. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 747–776.

[18] Daniel V. Bailey, Daniel Coffin, Adam Elbirt, Joseph H. Silverman, and Adam D. Woodbury. 2001. NTRU in constrained devices. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 262–272.

[19] Abhishek Banerjee, Chris Peikert, and Alon Rosen. 2012. Pseudorandom functions and lattices. In *Proceedings of the Conference on Advances in Cryptology (EUROCRYPT'12)*. 719–737.

[20] Utsav Banerjee, Tenzin S. Ukyab, and Anantha P. Chandrakasan. 2019. Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols. Retrieved from https://arXiv:1910.07557.

[21] Paul Barrett. 1986. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques*. Springer, 311–323.

[22] Kanad Basu, Deepraj Soni, Mohammed Nabeel, and Ramesh Karri. 2019. NIST post-quantum cryptography-A hardware evaluation study. *IACR ePrint Archive*. https://eprint.iacr.org/2019/047.

[23] Aurélie Bauer, Henri Gilbert, Guénaël Renault, and Mélissa Rossi. 2019. Assessment of the key-reuse resilience of NewHope. In *Proceedings of the Cryptographers' Track at the RSA Conference*. Springer, 272–292.

[24] Daniel J. Bernstein. 2006. Curve25519: New Diffie-Hellman speed records. In *Proceedings of the International Workshop on Public Key Cryptography*. Springer, 207–228.

[25] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. [n.d.]. NTRU Prime: Algorithm specifications and supporting documentation. Retrieved from https://ntruprime.cr.yp.to/nist/ntruprime-20190330.pdf.

[26] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. 2017. NTRU prime: Reducing attack surface at low cost. In *Proceedings of the International Conference on Selected Areas in Cryptography*. Springer, 235–260.

[27] Daniel J. Bernstein, Tanja Lange, and Dan Page. [n.d.]. eBATS. ECRYPT Benchmarking of Asymmetric Systems: Performing Benchmarks (technical report).

[28] Daniel J. Bernstein and Edoardo Persichetti. 2018. Towards KEM unification. *IACR ePrint Archive*. https://eprint.iacr.org/2018/526.

[29] Daniel J. Bernstein and Bo-Yin Yang. 2019. Fast constant-time gcd computation and modular inversion. *IACR Trans. Cryptogr. Hardware Embed. Syst.* 2019 (2019), 340–398. https://doi.org/10.13154/tches.v2019.i3.340-398.

[30] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. 2009. Keccak specifications. *Submission to NIST (Round 2)* (2009), 320–337.

[31] Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Rietman, and Ludo Tolhuizen. 2017. spKEX: An optimized lattice-based key exchange. *IACR EPrint Archive* (2017). Retrieved from https://eprint.iacr.org/2017/709.

[32] David Blackman and Sebastiano Vigna. 2018. Scrambled linear pseudorandom number generators. Retrieved from https://arXiv:1805.01407.

[33] Joppe Bos, Craig Costello, Léo 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*. ACM, 1006–1018.

[34] Joppe Bos, Craig Costello, Léo 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*. ACM, 1006–1018.

[35] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2018. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P'18)*. IEEE, 353–367.

[36] 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)*. IEEE.

[37] Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. 2018. Assessing the feasibility of single trace power analysis of Frodo. In *Proceedings of the International Conference on Selected Areas in Cryptography*. Springer.

[38] Joppe W. Bos, Simon Friedberger, Marco Martinoli, Elisabeth Oswald, and Martijn Stam. 2018. Fly, you fool! Faster Frodo for the ARM Cortex-M4.*IACR ePrint Archive*. Retrieved from https://eprint.iacr.org/2018/1116.

[39] Leon Botros, Matthias J. Kannwischer, and Peter Schwabe. 2019. Memory-efficient high-speed implementation of Kyber on Cortex-M4. In *Proceedings of the International Conference on Cryptology in Africa*. Springer, 209–228.

[40] Matt Braithwaite. 2016. Experimenting with post-quantum cryptography. *Google Security Blog* 7 (2016).

[41] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory* 6, 3 (2014), 13.

[42] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. 2013. Classical hardness of learning with errors. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*. ACM, 575–584.

[43] Jacqueline Brendel, Marc Fischlin, Felix Günther, Christian Janson, and Douglas Stebila. 2019. Challenges in proving post-quantum key exchanges based on key encapsulation mechanisms.

[44] Leon Groot Bruinderink, Andreas Hülsing, Tanja Lange, and Yuval Yarom. 2016. Flush, Gauss, and Reload–a cache attack on the BLISS lattice-based signature scheme. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 323–345.

[45] 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*. Springer, 402–417.

[46] 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 2nd ACM International Workshop on IoT Privacy, Trust, and Security*. ACM, 2–9.

[47] CESG. 2016. Quantum Key Distribution. Retrieved from https://www.cesg.gov.uk/white-papers/quantum-key-distribution.

[48] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M Schanck, Peter Schwabe, William Whyte, and Zhenfei Zhang. [n.d.]. NTRU: Algorithm specifications and supporting documentation. Retrieved from https://ntru.org/f/ntru-20190330.pdf.

[49] Donald Donglong Chen, Nele Mentens, Frederik Vercauteren, Sujoy Sinha Roy, Ray C. C. Cheung, Derek Pao, and Ingrid Verbauwhede. 2015. High-speed polynomial multiplication architecture for ring-LWE and SHE cryptosystems. *IEEE Trans. Circ. Syst. I: Reg. Papers* 62, 1 (2015), 157–166.

[50] Jung Hee Cheon, Kyoohyung Han, Jinsu Kim, Changmin Lee, and Yongha Son. 2016. A practical post-quantum public-key cryptosystem based on spLWE. In *Proceedings of the International Conference on Information Security and Cryptology*. Springer, 51–74.

[51] Robert Chien. 1964. Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. *IEEE Trans. Info. Theory* 10, 4 (1964), 357–363.

[52] CNSS. 2015. Use of Public Standards for the Secure Sharing of Information Among National Security Systems. *Committee on National Security Systems: CNSS Advisory Memorandum, Information Assurance 02-15*.

[53] James Cooley and John Tukey. 1965. An algorithm for the machine calculation of complex fourier series. *Math. Comp.* 19, 90 (1965), 297–301.

[54] Don Coppersmith and Adi Shamir. 1997. Lattice attacks on NTRU. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 52–61.

[55] Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. 2017. Short stickelberger class relations and application to Ideal-SVP. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer.

[56] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. [n.d.]. Saber: Algorithm specifications and supporting documentation (round 2). Retrieved from https://www.esat.kuleuven.be/cosic/pqcrypto/saber/resources.html.

[57] Jan-Pieter D'Anvers, Frederik Vercauteren, and Ingrid Verbauwhede. 2018. On the impact of decryption failures on the security of LWE/LWR based schemes. *IACR ePrint Archive* (2018). Retrieved from https://eprint.iacr.org/2018/1089.

[58] 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)*. IEEE.

[59] Jintai Ding, Saed Alsayigh, R. V. Saraswathy, Scott Fluhrer, and Xiaodong Lin. 2017. Leakage of signal function with reused keys in RLWE key exchange. In *Proceedings of the IEEE International Conference on Communications (ICC'17)*. IEEE, 1–6.

[60] Jintai Ding, Chi Cheng, and Yue Qin. 2019. A simple key reuse attack on LWE and ring LWE encryption schemes as key encapsulation mechanisms (KEMs). *IACR ePrint Archive*. Retrieved from https://eprint.iacr.org/2019/271.

[61] Jintai Ding, Scott Fluhrer, and R. V. Saraswathy. 2018. Complete attack on RLWE key exchange with reused keys, without signal leakage. In *Proceedings of the Australasian Conference on Information Security and Privacy*. Springer, 467–486.

[62] Jintai Ding, Xiang Xie, and Xiaodong Lin. 2012. A simple provably secure key-exchange scheme based on the learning with errors problem. *IACR EPrint Archive*. Retrieved from https://eprint.iacr.org/2012/688.

[63] Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. 2013. Lattice signatures and bimodal Gaussians. In *Proceedings of the Conference on Advances in Cryptology (CRYPTO'13)*. Springer, 40–56.

[64] Jan-Pieter D'Anvers, Marcel Tiepelt, Frederik Vercauteren, and Ingrid Verbauwhede. 2019. Timing attacks on error correcting codes in post-quantum schemes. In *Proceedings of the ACM Workshop on Theory of Implementation Security Workshop*. ACM, 2–9.

[65] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. 2016. Loop abort faults on lattice-based Fiat-Shamir & Hash'n sign signatures. *IACR ePrint Archive*. Retrieved from https://eprint.iacr.org/2016/449.

[66] Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. 2017. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongSwan and electromagnetic emanations in microcontrollers. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*.

[67] Viet Ba Dang, Farnoud Farahmand, Michal Andrzejczak, Kamyar Mohajerani, Duc Tri Nguyen, and Kris Gaj. 2008. Implementation and benchmarking of round 2 candidates in the NIST post-quantum cryptography standardization process using hardware and software/hardware Co-design approaches. *IACR EPrint Archive* (2008). Retrieved from https://eprint.iacr.org/2020/795/20200627:185511.

[68] Farnoud Farahmand, Viet B. Dang, Duc Tri Nguyen, and Kris Gaj. 2019. Evaluating the potential for hardware acceleration of four NTRU-based key encapsulation mechanisms using software/hardware codesign. In *Proceedings of the International Conference on Post-Quantum Cryptography*. Springer, 23–43.

[69] Scott R. Fluhrer. 2016. Cryptanalysis of ring-LWE based key exchange with key share reuse. *IACR ePrint Archive* (2016). Retrieved from https://eprint.iacr.org/2016/085.

[70] Tim Fritzmann, Thomas Pöppelmann, and Johanna Sepulveda. 2018. Analysis of error-correcting codes for lattice-based key exchange. In *Proceedings of the International Conference on Selected Areas in Cryptography*. Springer, 369–390.

[71] Tim Fritzmann and Johanna Sepúlveda. 2019. Efficient and flexible low-power NTT for lattice-based cryptography. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST'19)*.

[72] Tim Fritzmann, Uzair Sharif, Daniel Müller-Gritschneder, Cezar Reinbrecht, Ulf Schlichtmann, and Johanna Sepulveda. 2019. Towards reliable and secure post-quantum co-processors based on RISC-V. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE'19)*. IEEE, 1148–1153.

[73] Eiichiro Fujisaki and Tatsuaki Okamoto. 1999. Secure integration of asymmetric and symmetric encryption schemes. In *Proceedings of the Annual International Cryptology Conference*. Springer, 537–554.

[74] Robert Gallager. 1962. Low-density parity-check codes. *IRE Trans. Info. Theory* 8, 1 (1962), 21–28.

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

[76] Chunsheng Gu. 2019. Integer version of ring-LWE and its applications. In *Proceedings of the International Symposium on Security and Privacy in Social Networks and Big Data*. Springer, 110–122.

[77] Shay Gueron and Fabian Schlieker. 2016. Speeding up R-LWE post-quantum key exchange. In *Proceedings of the Nordic Conference on Secure IT Systems*. Springer, 187–198.

[78] Mike Hamburg. [n.d.]. ThreeBears: Algorithm specifications and supporting documentation. Retrieved from https://www.shiftleft.org/papers/threebears/threebears-july2019.pdf.

[79] Jeffrey Hoffstein, Jill Pipher, and Joseph Silverman. 1998. NTRU: A ring-based public key cryptosystem. *Algor. Number Theory* (1998), 267–288.

[80] James Howe, Ayesha Khalid, Marco Martinoli, Francesco Regazzoni, and Elisabeth Oswald. 2019. Fault attack countermeasures for error samplers in lattice-based cryptography. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'19)*. IEEE, 1–5.

[81] James Howe, Ayesha Khalid, Ciara Rafferty, Francesco Regazzoni, and Máire O'Neill. 2016. On practical discrete Gaussian samplers for lattice-based cryptography. *IEEE Trans. Comput.* (2016).

[82] James Howe, Ciara Moore, Máire O'Neill, Francesco Regazzoni, Tim Güneysu, and Kevin Beeden. 2016. Lattice-based encryption over standard lattices in hardware. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM.

[83] James Howe, Tobias Oder, Markus Krausz, and Tim Güneysu. 2018. Standard lattice-based key encapsulation on embedded devices. *IACR Trans. Cryptogr. Hardware Embed. Syst.* 2018, 3 (2018), 372–393. https://doi.org/10.13154/tches.v2018.i3.372-393

[84] James Howe, Thomas Pöppelmann, Máire O'Neill, Elizabeth O'Sullivan, and Tim Güneysu. 2015. Practical lattice-based digital signature schemes. *ACM Trans. Embed. Comput. Syst.* 14, 3 (2015), 41.

[85] Nick Howgrave-Graham. 2007. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In *Proceedings of the Annual International Cryptology Conference*. Springer, 150–169.
[86] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. 2003. The impact of decryption failures on the security of NTRU encryption. In *Proceedings of the Annual International Cryptology Conference*. Springer, 226–246.
[87] Nick Howgrave-Graham, Joseph H. Silverman, Ari Singer, William Whyte, and NTRU Cryptosystems. 2003. NAEP: Provable security in the presence of decryption failures. *IACR ePrint Archive*. Retrieved from https://eprint.iacr.org/2003/172.
[88] Wei-Lun Huang, Jiun-Peng Chen, and Bo-Yin Yang. 2020. Power analysis on NTRU prime. *IACR Trans. Cryptogr. Hardware Embed. Syst.* 2020, 1 (2020), 123–151. https://doi.org/10.13154/tches.v2020.i1.123-151
[89] Andreas Hülsing, Joost Rijneveld, John Schanck, and Peter Schwabe. 2017. High-speed key encapsulation from NTRU. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 232–252.
[90] Arpan Jati, Naina Gupta, Somitra Kumar Sanadhya, and Anupam Chattopadhyay. 2019. SPQCop: Side-channel protected post-quantum cryptoprocessor. *IACR ePrint Archive*. Retrieved from https://eprint.iacr.org/2019/765.
[91] Éliane Jaulmes and Antoine Joux. 2000. A chosen-ciphertext attack against NTRU. In *Proceedings of the Annual International Cryptology Conference*. Springer, 20–35.
[92] Burton S. Kaliski. 1995. The Montgomery inverse and its applications. *IEEE Trans. Comput.* 44, 8 (1995).
[93] Abdel Alim Kamal and Amr Youssef. 2011. Fault analysis of the NTRUEncrypt cryptosystem. *IEICE Trans. Fund. Electr. Commun. Comput. Sci.* 94, 4 (2011), 1156–1158.
[94] Abdel Alim Kamal and Amr M. Youssef. 2009. An FPGA implementation of the NTRUEncrypt cryptosystem. In *Proceedings of the International Conference on Microelectronics (ICM'09)*. IEEE, 209–212.
[95] Abdel Alim Kamal and Amr M. Youssef. 2013. Strengthening hardware implementations of NTRUEncrypt against fault analysis attacks. *J. Cryptogr. Eng.* 3, 4 (2013), 227–240.
[96] Panos Kampanakis and Dimitrios Sikeridis. 2019. Two post-quantum signature use-cases: Non-issues, challenges and potential solutions. *IACR ePrint Archive (2019)*. Retrieved from https://eprint.iacr.org/2019/1276.
[97] Matthias J. Kannwischer, Joost Rijneveld, and Peter Schwabe. 2018. Faster multiplication in Z2m [x] on cortex-M4 to speed up NIST PQC candidates. *IACR ePrint Archive*. Retrieved from https://eprint.iacr.org/2018/1018
[98] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. 2019. pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4. Retrieved from https://github.com/mupq/pqm4/tree/c32bcd017b202d418c9135e2df77be73a69044a0.
[99] Anatolii Karatsuba. 1963. Multiplication of multidigit numbers on automata. In *Sov. Phys. Dokl.*, Vol. 7. 595–596.
[100] Angshuman Karmakar, Jose Maria Bermudo Mera, Sujoy Sinha Roy, and Ingrid Verbauwhede. 2018. Saber on ARM. CCA-secure module lattice-based key encapsulation on ARM. *IACR Trans. Cryptogr. Hardware Embed. Syst.* 2018, 3 (2018), 243–266. https://doi.org/10.13154/tches.v2018.i3.243-266
[101] Paul Kirchner and Pierre-Alain Fouque. 2017. Revisiting lattice attacks on overstretched NTRU parameters. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 3–26.
[102] Donald Ervin Knuth. 1998. *The Art of Computer Programming: Sorting and Searching*. Vol. 3. Pearson Education.
[103] Donald E. Knuth and Andrew C. Yao. 1976. The complexity of nonuniform random number generation. In *Algorithms and Complexity: New Directions and Recent Results*. Academic Press, 357–428.
[104] Po-Chun Kuo, Wen-Ding Li, Yu-Wei Chen, Yuan-Che Hsu, Bo-Yuan Peng, Chen-Mou Cheng, and Bo-Yin Yang. 2017. Post-quantum key exchange on FPGAs. *IACR ePrint Archive* (2017). Retrieved from https://eprint.iacr.org/2017/690.
[105] Adam Langley. [n.d.]. Post-quantum confidentiality for TLS. Retrieved from https://www.imperialviolet.org/2018/04/11/pqconftls.html.
[106] Adam Langley. [n.d.]. Real-world measurements of structured-lattices and supersingular isogenies in TLS. Retrieved from https://www.imperialviolet.org/.
[107] Adeline Langlois and Damien Stehlé. 2015. Worst-case to average-case reductions for module lattices. *Designs, Codes Cryptogr.* 75, 3 (2015), 565–599.
[108] Mun-Kyu Lee, Jung Woo Kim, Jeong Eun Song, and Kunsoo Park. 2007. Sliding window method for NTRU. In *Applied Cryptography and Network Security*. Springer, 432–442.
[109] Mun-Kyu Lee, Jeong Eun Song, Dooho Choi, and Dong-Guk Han. 2010. Countermeasures against power analysis attacks for the NTRU public key cryptosystem. *IEICE Trans. Fund. Electron. Commun. Comput. Sci.* 93, 1 (2010), 153–163.
[110] Arjen Klaas Lenstra, Hendrik Willem Lenstra, and László Lovász. 1982. Factoring polynomials with rational coefficients. *Math. Ann.* 261, 4 (1982), 515–534.

[111] Richard Lindner and Chris Peikert. 2011. Better key sizes (and attacks) for LWE-based encryption. In *Proceedings of the Cryptographer's Track at RSA Conference (CT-RSA'11)*.
[112] Bingxin Liu and Huapeng Wu. 2015. Efficient architecture and implementation for NTRUEncrypt system. In *Proceedings of the IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS'15)*. IEEE, 1–4.
[113] Bingxin Liu and Huapeng Wu. 2016. Efficient multiplication architecture over truncated polynomial ring for NTRUEncrypt system. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'16)*. IEEE, 1174–1177.
[114] Zhe Liu and Johann Großschädl. 2014. New speed records for Montgomery modular multiplication on 8-bit AVR microcontrollers. In *Proceedings of the International Conference on Cryptology in Africa*. Springer, 215–234.
[115] Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. 2018. LAC: Practical ring-LWE based public-key encryption with byte-level modulus. *IACR ePrint Archive* (2018). Retrieved from https://eprint.iacr.org/2018/1009.
[116] 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 the Theory and Applications of Cryptographic Techniques (EUROCRYPT'10)*. 1–23.
[117] Vadim Lyubashevsky and Gregor Seiler. 2019. NTTRU: Truly fast NTRU using NTT. *IACR Trans. Cryptogr. Hardware Embed. Syst.* 2019, 3 (2019), 180–201. https://doi.org/10.13154/tches.v2019.i3.180-201
[118] Daniele Micciancio. 2010. Duality in lattice cryptography. In *Public Key Cryptography*. Springer, 2.
[119] Masoud Mohseni, Peter Read, Hartmut Neven, Sergio Boixo, Vasil Denchev, Ryan Babbush, Austin Fowler, Vadim Smelyanskiy, and John Martinis. 2017. Commercialize quantum technologies in five years. *Nature News* 543, 7644 (2017), 171.
[120] Peter L Montgomery. 1985. Modular multiplication without trial division. *Math. Comput.* 44, 170 (1985).
[121] Michele Mosca and Douglas Stebila. 2017. Open quantum safe. *Software for Prototyping Quantum-resistant Cryptography. Open Quantum Safe*.
[122] Hamid Nejatollahi, Nikil Dutt, Sandip Ray, Francesco Regazzoni, Indranil Banerjee, and Rosario Cammarota. 2019. Post-quantum lattice-based cryptography implementations: A survey. *ACM Comput. Surveys* 51, 6 (2019).
[123] Phong Q. Nguyen and David Pointcheval. 2002. Analysis and improvements of NTRU encryption paddings. In *Proceedings of the Annual International Cryptology Conference*. Springer, 210–225.
[124] NIST. 2016. Post-Quantum Crypto Project. Retrieved from http://csrc.nist.gov/groups/ST/post-quantum-crypto/.
[125] Tobias Oder and Tim Güneysu. 2017. Implementing the NewHope-simple key exchange on low-cost FPGAs. In *Proceedings of the Conference on Progress in Cryptology (LATINCRYPT'17)*.
[126] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. 2018. Practical CCA2-secure and masked ring-LWE implementation. *IACR Trans. Cryptogr. Hardware Embed. Syst.* 2018, 1 (2018), 142–174. https://doi.org/10.13154/tches.v2018.i1.142-174
[127] Christian Paquin, Douglas Stebila, and Goutam Tamvada. 2019. Benchmarking post-quantum cryptography in TLS. *IACR ePrint Archive* (2019). Retrieved from https://eprint.iacr.org/2019/1447.
[128] Judea Pearl. 1986. Fusion, propagation, and structuring in belief networks. *Artific. Intell.* 29, 3 (1986), 241–288.
[129] Chris Peikert. 2008. Public-key cryptosystems from the worst-case shortest vector problem. *Electr. Colloq. Comput. Complex.* 15, 100 (2008).
[130] Chris Peikert. 2010. An efficient and parallel Gaussian sampler for lattices. In *Proceedings of the Annual Cryptology Conference*. Springer.
[131] Chris Peikert. 2014. Lattice cryptography for the Internet. In *Proceedings of the International Workshop on Post-Quantum Cryptography*. Springer, 197–219.
[132] Peter Pessl and Robert Primas. 2019. More practical single-trace attacks on the number theoretic transform. In *Proceedings of the International Conference on Cryptology and Information Security in Latin America*. Springer, 130–149.
[133] 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*. Springer.
[134] Thomas Pöppelmann and Tim Güneysu. 2014. Area optimization of lightweight lattice-based encryption on reconfigurable hardware. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'14)*. IEEE, 2796–2799.
[135] Thomas Pöppelmann, Tobias Oder, and Tim Güneysu. 2015. High-performance ideal lattice-based cryptography on 8-Bit ATxmega microcontrollers. In *Proceedings of the 4th International Conference on Cryptology and Information Security in Latin America (LATINCRYPT'15)*. 346–365.
[136] Robert Primas, Peter Pessl, and Stefan Mangard. 2017. Single-trace side-channel attacks on masked lattice-based encryption. In *Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 513–533.

[137] Yue Qin, Chi Cheng, and Jintai Ding. 2019. A complete and optimized key mismatch attack on NIST candidate NewHope. *IACR ePrint Archive* (2019). Retrieved from https://eprint.iacr.org/2019/435.

[138] Prasanna Ravi, Debapriya Basu Roy, Shivam Bhasin, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2019. Number "not used" once-practical fault attack on pqm4 implementations of NIST candidates. In *Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 232–250.

[139] Prasanna Ravi, Sujoy Sinha Roy, Anupam Chattopadhyay, and Shivam Bhasin. 2019. Generic side-channel attacks on CCA-secure lattice-based PKE and KEM schemes. *IACR ePrint Archive* (2019). Retrieved from https://eprint.iacr.org/2019/948.

[140] Oded Regev. 2009. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* 56, 6 (2009), 34.

[141] Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2016. Additively homomorphic ring-LWE masking. In *Proceedings of the International Workshop on Post-Quantum Cryptography*. Springer.

[142] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. 2015. A masked ring-LWE implementation. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 683–702.

[143] Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. 2014. Compact and side channel secure discrete Gaussian sampling. *IACR ePrint Archive*. Retrieved from https://eprint.iacr.org/2014/591.

[144] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. 2014. Compact ring-LWE cryptoprocessor. In *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 371–391.

[145] 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*. Springer, 383–401.

[146] Markku-Juhani O. Saarinen. 2019. Exploring NIST LWC/PQC Synergy R5Sneik: How SNEIK 1.1 algorithms were designed to support round5. *IACR ePrint Archive* (2019). Retrieved from https://eprint.iacr.org/2019/685.

[147] Markku-Juhani O. Saarinen. 2016. Arithmetic coding and blinding countermeasures for ring-LWE. *IACR ePrint Archive* (2016). Retrieved from https://eprint.iacr.org/2016/276.

[148] Markku-Juhani O. Saarinen. 2017. HILA5: On reliability, reconciliation, and error correction for Ring-LWE encryption. In *Proceedings of the International Conference on Selected Areas in Cryptography*. Springer, 192–212.

[149] Markku-Juhani O. Saarinen, Sauvik Bhattacharya, Oscar Garcia-Morchon, Ronald Rietman, Ludo Tolhuizen, and Zhenfei Zhang. 2018. Shorter messages and faster post-quantum encryption with Round5 on Cortex M. In *Proceedings of the International Conference on Smart Card Research and Advanced Applications*. Springer, 95–110.

[150] Tsunekazu Saito, Keita Xagawa, and Takashi Yamakawa. 2018. Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 520–551.

[151] Thomas Schamberger, Oliver Mischke, and Johanna Sepulveda. 2019. Practical evaluation of masking for NTRUEncrypt on ARM Cortex-M4. In *Proceedings of the International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer.

[152] Claus-Peter Schnorr and Martin Euchner. 1994. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Math. Program.* 66, 1–3 (1994), 181–199.

[153] Gregor Seiler. 2018. Faster AVX2 optimized NTT multiplication for Ring-LWE lattice cryptography. *IACR ePrint Archive* (2018). Retrieved from https://eprint.iacr.org/2018/039.

[154] Peter W. Shor. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE, 124–134.

[155] Joseph H. Silverman. 1999. Almost inverses and fast NTRU key creation. *NTRU Cryptosyst.* Technical Report #014. Retrieved from https://ntru.org/f/tr/tr014v1.pdf.

[156] Shiming Song, Wei Tang, Thomas Chen, and Zhengya Zhang. 2018. LEIA: A 2.05 mm 2 140mW lattice encryption instruction accelerator in 40nm CMOS. In *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC'18)*. IEEE, 1–4.

[157] Douglas Stebila, Michele Mosca, Christian Paquin, Dimitris Sikeridis, and Goutam Tamvada. [n.d.]. OQS-OpenSSL_1_1_1-Fork of OpenSSL by OpenOQS project. Retrieved from https://github.com/open-quantum-safe/openssl.

[158] Damien Stehlé and Ron Steinfeld. 2011. Making NTRU as secure as worst-case problems over ideal lattices. In *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 27–47.

[159] Ehsan Ebrahimi Targhi and Dominique Unruh. 2016. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In *Proceedings of the 14th International Conference on Theory of Cryptography (TCC'16-B)*. Springer, Berlin, 192–216.

[160] Ludo Tolhuizen, Ronald Rietman, and Oscar Garcia-Morchon. 2017. Improved key-reconciliation method. *IACR ePrint Archive* (2017). Retrieved from https://eprint.iacr.org/2017/295.

[161] Andrei L. Toom. 1963. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, Vol. 3. 714–716.

[162] Felipe Valencia, Tobias Oder, Tim Güneysu, and Francesco Regazzoni. 2018. Exploring the vulnerability of R-LWE encryption to fault attacks. In *Proceedings of the 5th Workshop on Cryptography and Security in Computing Systems*. ACM.

[163] William Whyte, Nick Howgrave-Graham, Jeffrey Hoffstein, Jill Pipher, Joseph H. Silverman, and Philip S. Hirschhorn. 2008. IEEE P1363. 1 Draft 10: Draft standard for public key cryptographic techniques based on hard problems over lattices. *IACR EPrint Archive* (2008). Retrieved from https://eprint.iacr.org/2008/361.

[164] Xuexin Zheng, An Wang, and Wei Wei. 2013. First-order collision attack on protected NTRU cryptosystem. *Microprocess. Microsyst.* 37, 6–7 (2013), 601–609.

[165] Timo Zijlstra, Karim Bigou, and Arnaud Tisserand. 2019. FPGA implementation and comparison of protections against SCAs for RLWE. In *Proceedings of the International Conference on Cryptology in India*. Springer, 535–555.

Received January 2018; revised June 2020; accepted August 2020