Overview of Threshold PQC Schemes

Joon Kim, Filip Rezabek*, Dr. Holger Kinkelin†
Chair of Network Architectures and Services
School of Computation, Information and Technology, Technical University of Munich, Germany
*Email: joon.kim@tum.de, *rezabek@net.in.tum.de, †kinkelin@net.in.tum.de*

***Abstract*—Threshold schemes distribute a signing key among multiple parties, requiring their collaboration to perform cryptographic tasks, thereby mitigating the risk of key compromise. As quantum computing advances, recent research has increasingly focused on combining these schemes with post-quantum secure cryptographic primitives such as lattices. This paper analyzes three significant advancements on post-quantum secure threshold schemes. First, Boneh et al.'s universal thresholdizer converts any cryptographic scheme into a threshold version, offering great flexibility, but it suffers from inefficiencies from homomorphically evaluating entire circuits. Second, Kamil et al. improve upon this by selectively applying homomorphic evaluation to an existing (n, n)-threshold scheme, extending it to a (t, N)-scheme. Lastly, Cozzo et al. thresholdize FALCON using multiparty computation techniques, but the mixture of linear and non-linear operations in FALCON results in relatively long signing times.**

***Index Terms*—Threshold Cryptography, Lattice-based**

1. Introduction

The rapid development of quantum computing poses a significant threat to the security of classical cryptographic systems [1] [2]. Traditional encryption schemes, such as RSA and ECC, rely on the computational hardness of problems, which, however, can be solved in polynomial time by quantum computers [3]. As a result, these cryptographic systems, widely deployed e.g. in securing the internet and financial transactions are no longer considered future-proof in the face of advancing quantum technology. This pressing concern has led to the emergence of post-quantum cryptography (PQC), which is based on mathematical problems, that are believed to be hard even for quantum computers [3].

In parallel with the need for PQC, there is a growing interest in threshold cryptography. This technique strengthens security in distributed systems by decentralizing control across multiple participants [4] [3]. This approach has a wide range of applications, including cloud computing and blockchains [5].

As organizations work to secure data against quantum and other emerging threats, the combination of PQC and threshold cryptography offers a compelling solution for achieving both quantum resilience and enhanced fault-tolerant security in critical systems. In light of this, this paper provides an overview of lattice-based cryptography and threshold cryptography, followed by a survey of recently proposed threshold PQC schemes. It focuses on the works of Boneh et al. [6], Kamil et al. [7], and Cozzo et al. [8].

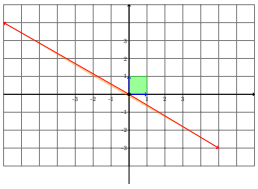

Figure 1: Two different basis vector pairs $\in \mathbb{R}^2$ build the fundamental domain $\mathbb{Z}^2$ [10].

2. Background

This section provides a brief overview of lattices and threshold cryptography.

2.1. Lattices

Lattices are as a key component of PQC due to their mathematical structure and the computational hardness of the problems they pose, which will be introduced in the following [9].

2.1.1. Lattice Fundamentals

Lattices are algebraic structures composed of points in n-dimensional space that are formed by the integer combinations of a set of basis vectors [9].

Definition 1. *A lattice $\mathcal{L}(B)$ is the span of its basis vectors $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$ of $\mathbb{R}^n$ [9], so that*

$$\mathcal{L}(B) = \left\{ \sum a_i \mathbf{b}_i : a_i \in \mathbb{Z} \right\} \tag{1}$$

The cryptographic significance of lattices stems from the fact that a given lattice L can be represented by multiple bases [10]. While a "good" basis can simplify certain computational tasks, a "bad" basis can make these tasks exceedingly difficult [10]. For instance, in Figure 1, the red pair of long vectors and the blue pair of short vectors provide valid alternative bases for the same domain $\mathbb{Z}^2$. However, answering mathematical questions like "Is $(\mathbf{1}, \mathbf{0})^{\mathbf{T}} \in \mathcal{L}(B)$?" would be more challenging when using the long and nearly parallel vectors as B in Figure 1 (the "bad" basis). This is in contrast to the shorter, orthogonal vectors (the "good" basis). Beyond this, there exist other computational problems related to lattices that are considered hard and make lattices appealing for use in PQC, which will be presented in the following.

Definition 2. ***Shortest Vector Problem (SVP):*** *Given a lattice basis B and some norm $\|\cdot\|$, find a*

 doi: 10.2313/NET-2025-05-1_06

(nonzero) vector $\mathbf{v} \in \mathcal{L}(B)$ *such that* $\|\mathbf{v}\| = \lambda_{min}(\mathcal{L}(B))$*, where* λ_{min} *is the minimum distance in the lattice [9].*

Definition 3. ***Closest Vector Problem (CVP):*** *Given a lattice basis* B*, some norm* $\|\cdot\|$*, and an arbitrary vector* $q \in \mathbb{R}^n$*, find a lattice-point* $l \in L$ *such that* $\|l - q\|$ *is minimal [9].*

These problems can also serve as the foundation for other more practical, equation-based challenges. Two examples of such challenges follow.

2.1.2. Learning with Errors

The Learning With Errors (LWE) problem serves as the foundation for many PQC schemes [11]. It starts with a simple system of linear equations $A \cdot s = b$, where $A \in \mathbb{Z}^{n \times m}, s \in \mathbb{Z}^m, b \in \mathbb{Z}^n$. Solving this system can be done efficiently using standard techniques, such as Gaussian elimination. LWE complicates this by adding a small noise vector $e \in \mathbb{Z}^n$, leading to the system $A \cdot s = b + e$. The challenge now is to recover the secret vector s despite the added noise, which makes the problem computationally hard [11]. A formal definition of LWE follows.

Definition 4. ***Learning with Errors (LWE):*** *Let* $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ *be the ring of integers modulo* q*. Given are* A *and* b*, where:*

$$A \sim \mathbb{Z}_q^{n \times m} \quad \text{is a matrix sampled uniformly,}$$
$$b = A \cdot s + e \quad \text{a vector in } \mathbb{Z}_q^n \text{ with added noise } e$$

with $e \in \mathbb{Z}_q^n$ *as a small error vector. Recover the secret* $s \in \mathbb{Z}_q^m$ *[11].*

The hardness of LWE stems from its close connection to SVP and CVP [11]. For example, in an LWE instance of the form $A \cdot s = b + e$, where $A \in \mathbb{Z}_q^{n \times m}$, $s \in \mathbb{Z}_q^m$, $b \in \mathbb{Z}_q^n$, the matrix A can be viewed as a lattice basis, with each column representing a basis vector. As a lattice point is a linear combination of these basis vectors, $A \cdot s$ can be considered a lattice point, while b is a random vector in $\mathbb{Z}_q^n$. Given that the error vector e is small, solving an LWE instance to recover the secret vector s can almost always be done by solving CVP, searching for s with minimal $\|\mathbf{As} - \mathbf{b}\|$. A formal proof of LWE's hardness can be found in [11].

Moreover, the presented LWE-problem can be extended by introducing specific algebraic structures. For instance, the R-LWE uses a polynomial ring rather than a ring of integers as in standard LWE and uses polynomial multiplications, for which efficient algorithms that are similar to Fast-Fourier-Transformation can be used [11] [12]. A formal definition of R-LWE follows.

Definition 5. ***Ring Learning with Errors (R-LWE):*** *Fix a polynomial* $f(x)$*, consider the polynomial ring modulo* $f(x)$*, i.e.,* $\mathbb{Z}_q[x]/f(x)$*. Given are noisy samples* $(a_i(x), b_i(x))$*, where:*

$$a_i(x) \sim \mathbb{Z}_q[x]/f(x), \quad b_i(x) \leftarrow a_i(x) \cdot s(x) + e_i(x)$$

with $e_i(x) \in \mathbb{Z}_q[x]/f(x)$ *as a small error polynomial. Recover the secret* $s(x) \in \mathbb{Z}_q[x]/f(x)$ *[11].*

2.1.3. Inhomogeneous Short Integer Solution

Another practical hard problem is the Inhomogeneous Short Integer Solution Problem (ISIS) [13], which shares structural similarities with LWE but includes an additional constraint of a size bound. In fact, ISIS can also be reduced to CVP and SVP [13]. A formal definition of ISIS follows.

Definition 6. ***Inhomogeneous Short Integer Solution:*** *Given* $A \in \mathbb{Z}_q^{n \times m}$*,* $b \in \mathbb{Z}_q^n$*, and* $\beta \in \mathbb{R}$*, find* $s \in \mathbb{Z}_q^m$ *satisfying* $A \cdot s = y \mod q$ *with* $\|\mathbf{s}\|_2 \leq \beta$ *[13].*

2.2. Threshold Cryptography

In addition to lattices, threshold schemes can enhance security by distributing cryptographic operations or secrets across multiple participants [14]. The general concept of threshold cryptography will be introduced in the following.

2.2.1. Threshold Cryptography Fundamentals

In most large companies with hierarchical structures, significant decisions, such as signing major contracts, are typically made only after a majority of the board members reach an agreement. Threshold cryptography follows a similar principle of collaboration. In a (t, N)-threshold scheme, a secret key sk is split into N shares $(sk_1, ..., sk_N)$, with each share distributed to different participants. In cryptographic scenarios where the complete secret key sk is required, at least t participants must collaborate, combining their shares sk_i to reconstruct the key sk and complete their threshold task with it. This ensures that if an attacker compromises fewer than t participants or servers (e.g., $t-1$), they cannot reconstruct the full key sk and the system still remains secure [14] [15]. Main threshold tasks are introduced in the following.

Key generation (KGen): Two methods exist for KGen:

- *Generation by a trusted authority*: A trusted third party generates the public and private key pair (pk, sk). The secret key sk is then distributed among n participants using methods like (t, N) - Shamir Secret Sharing (SSS) [14]. This sharing method divides sk into N shares sk_i using Lagrange interpolation polynomials λ_i, ensuring that any $t-1$ shares reveal no information about the secret sk.
- *Distributed key generation (DKGen)*: Multiple participants jointly compute the public key pk and secret shares sk_i, which ensures that no single party has access to the complete secret key sk [16].

Threshold signatures: Any subset of t participants can collaborate to generate a signature, ensuring that the signature remains independent of the specific subset of t parties involved. Moreover, the signature size should be independent of t and N [15].

Threshold decryption: Any subset of t parties can decrypt a ciphertext ctx [15].

3. Analysis

This section presents and analyzes three influential contributions to lattice-based threshold PQC schemes.

3.1. Boneh et al. (2017)

The work by Boneh et al. [6] provides two primary contributions. First, they construct a threshold fully-homomorphic encryption (TFHE) scheme based on the LWE problem. Building on this framework, they con-

 doi: 10.2313/NET-2025-05-1_06

struct an "universal thresholdizer" [6], a tool capable of converting non-threshold cryptographic schemes into their threshold variant. These two key contributions are presented and analyzed in the following.

3.1.1. Threshold Fully Homomorphic Encryption

Boneh et al. construct a TFHE scheme, building on the existing FHE scheme developed by Gentry, Sahai, and Waters (GSW) [17], which is presented below.

Simplified GSW-FHE scheme [17]: Fix the message μ and the matrix G.

- **FHE.Setup** $\rightarrow (pk, sk)$: Sample a random matrix $\mathbf{A}$, a random vector $\mathbf{s}$, and a noise vector $\mathbf{e}$. Set $pk = \begin{pmatrix} \mathbf{A} \\ \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \end{pmatrix}$ and $\text{sk} = \begin{pmatrix} -\mathbf{s} & 1 \end{pmatrix}$.
- **FHE.Encrypt**$(pk, \mu) \rightarrow ctx$: Return ciphertext $ctx = \mathbf{A} \cdot R + \mu \cdot G$, where R is a random matrix with entries in $\{0, 1\}$.
- **FHE.Decrypt**$(pk, sk, ctx) \rightarrow \mu$: Compute the linear product $y = \langle sk, ctx^k \rangle$, where ctx^k is the kth column of the matrix ctx, and return 0 if y is small and 1 otherwise.

Using this scheme, one can encrypt the message with pk from FHE.Setup and evaluate an arbitrary cryptographic algorithm directly on the encrypted message, which explains its fully homomorphic capability. The decrypted result can be retrieved through FHE.Decrypt. The security of FHE.Encrypt relies on the hardness of the LWE problem, as a slightly modified LWE instance is constructed during FHE.Setup, with the introduction of a random matrix R during encryption. An example python implementation is available in [18].

The (t, N)-threshold variant of this scheme (TFHE) by Boneh et al. [6] consists of following steps: **TFHE.Setup**, **TFHE.Encrypt**, **TFHE.PartDec**, and **TFHE.FinDec**. In TFHE.Setup, instead of using a single secret key sk as in FHE.Setup, the secret is split into N secret shares sk_i, which are distributed by a trusted third party. For decryption, each party computes a partial decryption $p_i = \langle sk_i, ctx^k \rangle$ using their secret share sk_i during TFHE.PartDec. These partial decryptions p_i are then combined to reconstruct the full decryption p in TFHE.FinDec. For example, by employing (t, N)-SSS in TFHE.Setup, p can be successfully reconstructed in TFHE.FinDec using any t partial decryptions. However, directly combining the partial decryptions could potentially leak information about the secret shares due to the simple operations involved, such as the linear product. To address this, Boneh et al. introduce noise during TFHE.PartDec, modifying the decryption process to $p_i = \langle sk_i, ctx^k \rangle + noise$, which resembles the structure of a LWE instance $b = A \cdot s + e$.

3.1.2. Universal Thresholdizer

Using the constructed TFHE scheme and non-interactive zero-knowledge proofs (NIZK), Boneh et al. develop the universal thresholdizer [6], which is presentend in the following.

UT: Fix a circuit C of a cryptographic scheme and a subset U of t parties.

- **UT.Setup**$(\mu) \rightarrow (pp, sk_1, \ldots, sk_n)$: Generates the public key pk and secret shares sk_i from TFHE.Setup, computes the ciphertext ctx with TFHE.Encrypt(μ), and the commitment $\text{com}_i = \text{Com}(sk_i)$. The public parameters pp are defined as $(pk, ctx, \{\text{com}_i\}_{i \in [N]})$.
- **UT.Eval**$(pp, sk_i, C) \rightarrow (ctx', p_i, \pi_i)$: Evaluates the given circuit C on the ciphertext ctx, producing the evaluated ciphertext ctx' and computes the partial decryption $p_i = \text{TFHE.PartDec}(pk, ctx', sk_i)$. It then generates a NIZK proof π_i regarding the correctness of pp and sk_i.
- **UT.Verify**(pp, C, p_i, π_i): Checks the NIZK proof π_i.
- **UT.Combine**$(pp, \{p_j\}_{j \in U}) \rightarrow p$: Combines partial decryptions p_i and reconstruct p using TFHE.FinDec.

Using this scheme, any cryptographic protocol, such as a signature scheme, can be transformed into its threshold version. First, the cryptographic protocol is encoded into a circuit C composed of gates. After UT.Setup, each party generates a partial signature during UT.Eval by using the encoded circuit as input. The partial signatures are then verified and combined using UT.Verify and UT.Combine, respectively.

Although this approach of UT offers flexibilities with its capability to thresholdize non-threshold schemes, it also presents certain limitations. First, it relies on a trusted third party in UT.Setup, lacking DKGen, which is considered more secure. Second, executing an entire circuit in TFHE can be computationally expensive, particularly when the circuit involves heavy steps like challenges or rejection sampling. Moreover, the noise flooding in TFHE.PartDec leads to inefficient scaling complexities, with $\Omega(N \log N)$ for sizes of secret key shares and $\Omega(\lambda^3)$ for signatures, where λ indicates the security parameter [6]. These drawbacks are addressed in subsequent work.

3.2. Kamil et al. (2023)

Kamil et al. [7] address restrictions of UT [6] with the following solutions:

1) Instead of TFHE, they utilized a threshold linearly homomorphic encryption (THE) scheme, which is selectively applied only to the relevant steps of the protocol, rather than to the entire circuit. Additionally, the constructed THE scheme incorporates DKGen.
2) They combine the state-of-the-art (n, n) threshold pq-signature scheme by Damgård et al. [19] with the constructed THE and DKGen, extending it to a (t, N)-threshold scheme.

3.2.1. THE With DKGen

Kamil et al. construct a THE scheme as a threshold variant of the R-LWE-based HE scheme by Brakerski et al. [20] and extend this with DKGen based on (t, N)-SSS [14]. Morover, they take care to add noise when combining partial decryptions p_i, following the same rationale as Boneh et al. [6]. The simplified version of their built scheme is presented below.

 doi: 10.2313/NET-2025-05-1_06

- **THE.DKGen [7]:** Fix a ring element $a_E \in R_q$, a small prime number k, and a subset U of t parties.
  1) Every party P_i samples $s_i, e_i \in R_q$ and computes $b_i = a_E \cdot s_i + k \cdot e_i$ and its (t, N)-Shamir secret shares $s_{i,j}$ of s_i.
  2) P_i sends $(b_i, s_{i,j})$ to every other party P_j
  3) Every party P_i computes its public key $pk = (a_E, b_E = \sum b_j)$ and secret share $sk_i = \sum s_{j,i}$.
- **THE.Encrypt**$(pk, \mu) \rightarrow (ctx)$: Samples $r, e', e'' \in R_q$ and outputs $ctx = (u, v) = (a_E \cdot r + k \cdot e', b_E \cdot r + k \cdot e'' + \mu)$
- **THE.PartDec**$(ctx, sk_i) \rightarrow (p_i)$: Samples noise E_i and output partial decryption $p_i = \lambda_i \cdot sk_i \cdot u + k \cdot E_i$ with λ_i being the Lagrange multiplier for party P_i.
- **THE.FinDec**$(ctx, \{p_j\}_{j \in U}) \rightarrow (p)$: Outputs the complete decryption $p = (v - \sum_{j \in U} p_j) \bmod q \mod k$.

In essence, Kamil et al. extend the base HE scheme [20] to THE by including SSS and generating individual R-LWE instances, each with its error e_i. Accordingly, the security of the scheme relies on the R-LWE assumption from the base scheme [20]. C++ implementations of the base scheme and a similar version of THE are available in [21].

3.2.2. Combining THE with scheme by Damgård et al.

Based on the existing (n, n)-signature scheme by Damgård et al. [19], Kamil et al. utilize steps from the constructed THE to extend this to an arbitrary threshold scheme. The following presents the simplified protocols, with the modified steps highlighted in bold.

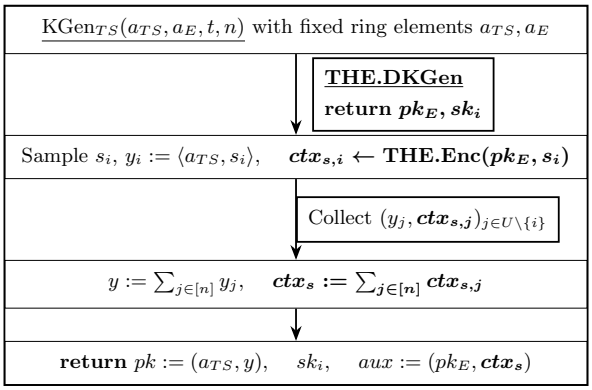

Figure 2: Passively secure (t, N) KGen protocol [22].

- **KGen:** The protocol first triggers THE.DKGen to generate the public key pk_E and the secret key share sk_i. Building on the existing protocol by Damgård et al. [19], it homomorphically encrypts the randomly generated s_i. The combined encrypted randomness, ctx_s, is then computed and returned as auxiliary information for subsequent operations.
- **Signing:** During signing, the protocol homomorphically encrypts the randomness r_i and computes the encrypted signature ctx_z by combining the encrypted random values $ctx_{r,j}$ from other parties, the auxiliary value ctx_s from KGen, and the challenge c. A subset of t parties then cooperatively decrypt ctx_z homomorphically, yielding the final signature z after combining the partial decryptions p_j.

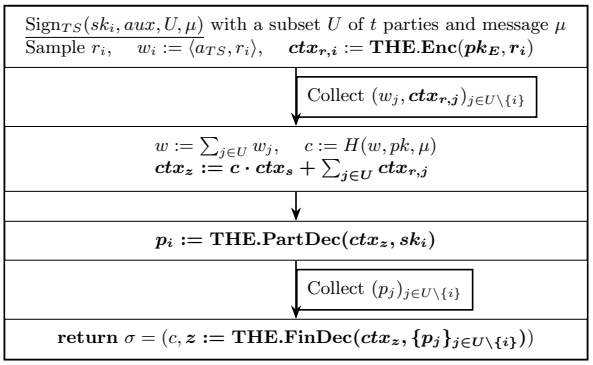

Figure 3: Passively secure (t, N) signing protocol [22].

In short, Kamil et al.'s modified protocol enhances the existing (n, n)-signature scheme by introducing randomness into KGen and generating the encrypted signature ctx_z using THE during signing, instead of the unencrypted signature z. This modification successfully extends the original protocol into a (t, N)-threshold variant. Furthermore, in contrast to UT, which requires a non-threshold scheme as input, this scheme is inherently a threshold scheme and does not depend on any external scheme. At the 128-bit security level with $t = 3$ and $N = 5$, this scheme produces signatures of size 46.6 kB of size 13.6 kB [7], offering a significant improvement in efficiency compared to other recent protocols such as Threshold raccoon [23].

3.3. Cozzo et al. (2019)

In contrast to the two presented works that utilize HE [6] [7], Cozzo et al. [8] discuss possibilities to thresholdize promising PQC schemes such as FALCON [24] based on secure multiparty computation (MPC) [25]. Non-threshold version of FALCON will be presented first.

Input : μ, $sk = B \in R_q^{2\times 2}$ with small $B_{i,j}$, $pk \in R_q^2$ with $B \cdot pk = \mathbf{0}$
Output: σ
1. $c \leftarrow H(r \parallel \mu)$ with random $r \in \{0,1\}^{320}$
2. Sample $t \in R_q^2$ such that $pk^\top \cdot t = c$.
3. $z \leftarrow \text{round}(t, B)$ such that $z \in \mathcal{L}(B)$
4. $s \leftarrow (t - z) \cdot B$
5. If s is not short, return to 3
6. Return $\sigma = (r, s)$

Figure 4: Simplified FALCON signing protocol [24]

FALCON follows the concept of hash-and-sign. It begins by computing a hash value c and search for t from an ISIS instance $pk^\top \cdot t = c$, which is possible by computing $t \leftarrow (c, 0) \cdot B^{-1}$ [24]. Afterwards, t is rounded to a close lattice-point $z \in \mathcal{L}(B)$ and the signature s is computed with the difference between t and z. Plus, a rejection sampling with the condition checking the shortness of s is included to enhance security of B. FALCON is fairly runtime efficient and compact, requiring only 0.3 miliseconds for signing and prodzcing signatures of 1.3 kB [24].

To thresholdize FALCON, Cozzo et al. [8] suggest utilizing MPC, enabling multiple parties to jointly perform computations using their individual inputs while ensuring the privacy of those inputs [25]. First, a linear secret sharing scheme (LSSS) can be used to distribute sk and pk, enabling linear operations to be performed

 doi: 10.2313/NET-2025-05-1_06

on the secret shares rather than directly on the secret key [26]. Furthermore, considering that FALCON involves both linear operations (e.g. step 2 in Figure 4) and non-linear ones (e.g. rejection sampling), LSSS-based MPC schemes are well suited for the linear operations, while garbled circuits (GC)-based MPC can handle the non-linear components [8].

This separation of needed MPC techniques requires costly conversions between LSSS and GC representations, leading to a major bottleneck and a longer signing time of 5.7 seconds [8]. Moreover, this threshold-FALCON possesses further limitations such as the absence of DKGen.

4. Conclusion

In conclusion, the presented three papers show recent advancements on threshold PQC schemes, presenting different ways to design threshold schemes. Boneh et al. [6] introduced the "universal thresholdizer," a tool capable of transforming any cryptographic scheme into its threshold variant through a black-box execution. Kamil et al. [7] identified remaining inefficiencies in the "universal thresholdizer" and proposed a new scheme using THE instead of TFHE. Cozzo et al. [8] utilize LSSS and MPC techniques, rather than HE, to thresholdize FALCON. Future work could focus on further optimizing the protocol proposed by Kamil et al. [7], e.g. by introducing compression techniques used in schemes like FALCON [24] or DILITHIUM [27] [28]. Moreover, exploring PQC schemes based on other mathematical primitives beyond lattices [29], such as hash functions or isogenies, presents an avenue for further research.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, p. 1484–1509, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[2] B. Brubaker, "Thirty years later, a speed boost for quantum factoring," 2023, quanta Magazine, October 17, 2023. [Online]. Available: https://www.quantamagazine.org/thirty-years-later-a-speed-boost-for-quantum-factoring-20231017/

[3] K. Sedghighadikolaei and A. A. Yavuz, "A comprehensive survey of threshold signatures: Nist standards, post-quantum cryptography, exotic techniques, and real-world applications," in *Proceedings of the arXiv Conference*, no. 2311.05514, 2024, pp. 1–2. [Online]. Available: https://arxiv.org/abs/2311.05514

[4] L. T. A. N. Brandão and R. Peralta, "Nist first call for multi-party threshold schemes (initial public draft)," National Institute of Standards and Technology (NIST), Tech. Rep. NIST IR 8214C IPD, 2023, january 2023. [Online]. Available: https://doi.org/10.6028/NIST.IR.8214C.ipd

[5] M. E. Manaa and Z. G. Hadi, "Scalable and robust cryptography approach using cloud computing," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, no. 7, pp. 1439–1445, 2020.

[6] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," vol. 10, no. 1, 2024.

[7] K. D. Gur, J. Katz, and T. Silde, "Two-round threshold lattice-based signatures from threshold homomorphic encryption," in *Proceedings of the Cryptology ePrint Archive*, ser. Lecture Notes in Computer Science (LNCS), vol. 1318. Springer, 2023. [Online]. Available: https://eprint.iacr.org/2023/1318

[8] D. Cozzo and N. P. smart, "Sharing the LUOV: Threshold post-quantum signatures," Cryptology ePrint Archive, Paper 2019/1060, 2019. [Online]. Available: https://eprint.iacr.org/2019/1060

[9] J. H. Silverman, "An introduction to lattices, lattice reduction, and lattice-based cryptography," *IAS/Park City Mathematics Series*, pp. 1–5, 2023.

[10] T. Laarhoven, J. van de Pol, and B. de Weger, "Solving hard lattice problems and the security of lattice-based cryptosystems," in *Proceedings of the Cryptology ePrint Archive*, no. 533. International Association for Cryptologic Research (IACR), 2012, pp. 2–4. [Online]. Available: https://eprint.iacr.org/2012/533

[11] O. Regev, "The learning with errors problem," *Survey on Learning with Errors (LWE)*, pp. 1–23, 2005. [Online]. Available: https://cims.nyu.edu/~regev/papers/lwesurvey.pdf

[12] C. Peikert, O. Regev, and N. Stephens-Davidowitz, "Pseudorandomness of ring-LWE for any ring and modulus," in *Proceedings of the Cryptology ePrint Archive*, no. 258. International Association for Cryptologic Research (IACR), 2017. [Online]. Available: https://eprint.iacr.org/2017/258

[13] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," Cryptology ePrint Archive, Paper 2007/432, 2007. [Online]. Available: https://eprint.iacr.org/2007/432

[14] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] Yvo Desmedt and Yair Frankel, "Threshold Cryptosystems," *Advances in Cryptology*, pp. 305–315, 1989, cRYPTO '89.

[16] C. Komlo, I. Goldberg, and D. Stebila, "A formal treatment of distributed key generation, and new constructions," in *Proceedings of the Cryptology ePrint Archive*, no. 292. International Association for Cryptologic Research (IACR), 2023. [Online]. Available: https://eprint.iacr.org/2023/292

[17] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," in *Proceedings of the Annual International Cryptology Conference (CRYPTO)*. Springer, 2013.

[18] K. Teranishi, "ECLib: An open-source homomorphic encryption library," 2024, accessed: 2024-09-22. [Online]. Available: https://github.com/KaoruTeranishi/EncryptedControl

[19] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi, "Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices," in *PKC 2021: 24th International Conference on Theory and Practice of Public Key Cryptography, Part I*, ser. Lecture Notes in Computer Science (LNCS), J. Garay, Ed., vol. 12710. Virtual Event, May 10–13: Springer, Heidelberg, 2021, pp. 99–130.

[20] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, S. Goldwasser, Ed. Cambridge, MA, USA, January 8–10: Association for Computing Machinery, 2012, pp. 309–325.

[21] A. A. Badawi, A. Alexandru, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, C. Pascoe, Y. Polyakov, I. Quah, S. R.V., K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "OpenFHE: An open-source fully homomorphic encryption library," 2022, cryptology ePrint Archive, Paper 2022/915, Accessed: 2024-09-22. [Online]. Available: https://eprint.iacr.org/2022/915

[22] K. D. Gur, J. Katz, and T. Silde, "Two-round threshold lattice-based signatures from threshold homomorphic encryption," in *Proceedings of the Cryptology ePrint Archive*, ser. Lecture Notes in Computer Science (LNCS), vol. 1318. Springer, 2023, pp. 18–19. [Online]. Available: https://eprint.iacr.org/2023/1318

[23] R. del Pino, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, and M.-J. Saarinen, "Threshold raccoon: Practical threshold signatures from standard lattice assumptions," Cryptology ePrint Archive, Paper 2024/184, 2024. [Online]. Available: https://eprint.iacr.org/2024/184

[24] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast-fourier lattice-based compact signatures over ntru," Falcon Project, Tech. Rep., 2020, specification v1.2 — 01/10/2020. [Online]. Available: https://falcon-sign.info/

[25] Y. Lindell, “Secure multiparty computation (mpc),” *Unbound Tech and Bar-Ilan University*, 2019, accessed: 2024-10-18. [Online]. Available: https://eprint.iacr.org/2020/300.pdf

[26] R. Cramer, I. B. Damgård, N. Döttling, S. Fehr, and G. Spini, “Linear secret sharing schemes from error correcting codes and universal hash functions,” in *Proceedings of Eurocrypt 2019*. Springer, 2019. [Online]. Available: https://www.iacr.org/archive/eurocrypt2015/90560182/90560182.pdf

[27] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, “Crystals-dilithium: A lattice-based digital signature scheme,” *Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, Issue 1, pp. 238–268, 2018. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/839

[28] C.-D. Team, “CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme (GitHub Repository),” 2017, accessed: 17 September 2024. [Online]. Available: https://github.com/pq-crystals/dilithium

[29] M. Buser, R. Dowsley, M. F. Esgin, C. Gritti, S. K. Kermanshahi, V. Kuchta, J. T. Legrow, J. K. Liu, R. C.-W. Phan, A. Sakzad, R. Steinfeld, and J. Yu, “A survey on exotic signatures for post-quantum blockchain: Challenges & research directions,” *Cryptology ePrint Archive*, vol. 1, no. 1, pp. 4–7, 2022. [Online]. Available: https://eprint.iacr.org/2022/1151.pdf