https://doi.org/

A Detailed Analysis of Two Lattice-Based Aggregate Signature Schemes

Stuti Kumari[1] and Kunal Dey[2]

[1]Department of mathematics, National Institute of Technology, Jamshedpur 831014, India
[2]Department of Mathematics, University of Calgary, Canada, Alberta, T2N 1N4
Corresponding author: kunaldey3@gmail.com

Abstract. Post-quantum cryptography has emerged as a major research area, with increasing focus on advanced digital signature primitives that address both efficiency and scalability. This article presents a comprehensive survey of one aggregate and one sequential aggregate signature scheme based on lattices, highlighting their underlying constructions, design principles, and security assumptions. By synthesizing the current state-of-the-art, this survey aims to provide researchers and practitioners with a clear roadmap for developing scalable, quantum-resistant aggregate and sequential aggregate signature solutions suitable for next generation distributed systems.

Key words: Lattices; Aggregate Signature; Sequential Aggregate Signature
Received: 31 October 2025 **Revised:** 26 January 2026 **Accepted:** 27 January 2026

1. Introduction

Digital signatures are fundamental cryptographic primitives that provide authenticity, integrity, and non-repudiation for digital messages. As the scale of digital communication grows, the need for efficient signature schemes that can handle multiple signers and large volumes of messages has become increasingly important. To address this issue, aggregate signature schemes allow distinct signers to generate signatures on different messages that can subsequently be merged into a single, concise signature. This significantly reduces storage and communication overhead while preserving the security guarantees of the individual signatures. Lattice-based cryptography has emerged as a promising foundation for post-quantum secure digital signatures due to its strong security assumptions and resistance to attacks from quantum computers. In particular, lattice-based aggregate signature schemes leverage the algebraic structure of lattices, enabling the linear combination of individual signatures into a single aggregate signature. Sequential aggregate signatures extend the concept of aggregation by allowing signers to sequentially add their signatures to an evolving aggregate. This model is particularly useful in applications such as blockchain systems, multi-hop authentication, and collaborative protocols, where the order of signing carries semantic importance. Recent research has explored lattice-based sequential aggregate signatures that maintain post-quantum security while offering efficient verification and aggregation.
Despite the promising advances, lattice-based aggregate and sequential aggregate signatures face several practical challenges, including signature size and computational efficiency. Early

constructions often produce aggregate signatures whose sizes exceed that of a simple concatenation of individual signatures, highlighting the trade-off between compactness and provable security.
We discuss foundational schemes, highlight practical limitations, and outline open problems, aiming to provide a comprehensive overview for researchers and practitioners interested in post-quantum signature aggregation. This review paper surveys two recent lattice-based aggregate and sequential aggregate signatures, analyzing their construction, security, and efficiency.

2. Prelminaries

Notation. Throughout this article, vectors are expressed using bold lowercase letters, while matrices are indicated by bold uppercase letters.

Lattices [8] Let $\mathbb{R}^n$ denote the n-dimensional Euclidean space. A *lattice* in $\mathbb{R}^n$ is defined as

$$\mathcal{L}(\mathbf{b}_1, \ldots, \mathbf{b}_m) = \left\{ \sum_{i=1}^{m} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z} \right\},$$

that is, the set of all integer linear combinations of m linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m \in \mathbb{R}^n$ (with $m \leq n$). The integer m is called the *rank* of the lattice, while n is its *dimension*. The generating vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$ form a *lattice basis*.

For any $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, we define

$$\mathcal{L}_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \bmod q\}.$$
$$\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{u} \bmod q\}$$

The lattice $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A})$ is a coset of $\mathcal{L}_q^{\perp}(\mathbf{A})$; that is, $\mathcal{L}_q^{\mathbf{u}}(\mathbf{A}) = \mathcal{L}_q^{\perp}(\mathbf{A}) + \mathbf{t}$ for any vector $\mathbf{t}$ satisfying $\mathbf{A}.\mathbf{t} = \mathbf{u} \bmod q$.

Module Short Integer Solution problem (M-SIS) [13] The Module Short Integer Solution problem (M-SIS) is defined by parameters $n, m, d \in \mathbb{N}$. Given a uniformly random matrix $\mathbf{A} \in R_q^{n \times m}$, the objective is to find a non-zero vector $\mathsf{s} \in R_q^{m+n}$ such that:

(i) The Euclidean norm of s is at most d (i.e., $\|\mathsf{s}\| \leq d$)

(ii) s satisfies the linear equation $[\mathbf{A} \mid I_n]\mathsf{s} = \mathbf{0}$ over the ring R_q^n

Module Learning With Error problem (M-LWE) [13] For natural numbers k, ℓ, η. The M-LWE$_{k,\ell,\eta}$, is defined in the following manner. Given a public matrix $\mathbf{A} \in R_q^{k \times \ell}$ and a public vector $\mathbf{t} \in R_q^k$, the task is to distinguish whether $\mathbf{t}$ is uniformly random in R_q^k or if it is of the form $\mathbf{t} = [\mathbf{A} \mid I_k] \cdot \mathbf{s}$, for some secret vector $\mathbf{s} \in S_\eta^{l+k}$.

2.1. Digital Signature [16]

A digital signature scheme DS = (KeyGen, Sign, Verify) is made up of three core algorithms, as shown below.

(i) **KeyGen**(1^λ)**:** Upon receiving the security parameter λ, the key generation algorithm produces an associated pair of keys: a public key pk and a secret key sk.

(ii) **Sign**(sk, m)**:** Using the secret key, the algorithm signs the message m to produces a signature σ.

(iii) **Verify**(pk, σ)**:** The verification algorithm takes the public key and the signature as inputs and outputs a result that confirms or denies the signature's authenticity.

2.2. Fiat-Shamir with Abort Signature [11]

The Fiat-Shamir with abort signature scheme, denoted as FSwA-S = (Setup, Gen, Sign, Verify), is composed of four main algorithms, which are outlined below.

(i) **Setup**(1^λ)**:** The setup algorithm samples a random matrix $\mathbf{A} \in R_q^{k\times\ell}$. It then extends this matrix to $\overline{\mathbf{A}} = [\mathbf{A} \mid \mathbf{I}_k]$, and produces $\overline{\mathbf{A}}$, which is designated as the public parameter.

(ii) **Gen**$(\overline{\mathbf{A}})$**:** The key generation algorithm samples a vector $\mathbf{s} \in S_\eta^{\ell+k}$, computes $\mathbf{t} = \overline{\mathbf{A}}\mathbf{s} \bmod q$. The secret key is defined as sk $= \mathbf{s}$, and the public key as pk $= \mathbf{t}$ and returns (sk,pk).

(iii) **Sign**$(\mathbf{sk}, m)$**:** The signing algorithm

- Set the secret signing key component $\mathbf{s} =$ sk.
- Compute $\mathbf{t} = \overline{\mathbf{A}}\mathbf{s} \bmod q$,

Initially, the response is set to $\mathbf{z} = \perp$. While $\mathbf{z} = \perp$, the algorithm performs the following steps:

- Sample $\mathbf{y}$ from the distribution $\mathcal{D}^{\ell+k}$.
- Compute the masked commitment $\mathbf{u} = \overline{\mathbf{A}}\mathbf{y} \bmod q$.
- Derive the challenge $c = H_1(m, \mathbf{t}, \mathbf{u})$, where $h : \{0,1\}^* \to S_\eta^{\ell+k}$ be a hash function.
- Compute the response $\mathbf{z} = c\mathbf{s} + \mathbf{y}$.
- Apply rejection sampling: $\mathbf{z} = \text{RejSamp}(\mathbf{z}, c \cdot \mathbf{s})$.
- If the response is invalid, reset $\mathbf{z} = \perp$ and repeat.

Once a valid response $\mathbf{z}$ is obtained, the algorithm outputs the new signature $\sigma = (\mathbf{u}, \mathbf{z})$.

(iv) **Verify**$(\mathbf{pk}, m, \sigma)$**:** Given a signature $\sigma = (\mathbf{u}, \mathbf{z})$, public key pk $= \mathbf{t}$, the verifier reconstructs the challenge $c = H_1(m, \mathbf{t}, \mathbf{u})$ and checks if $\|z\|_\infty \leq \mathsf{B}$, for some appropriate bound B and $\overline{\mathbf{A}}\mathbf{z} = c \cdot \mathbf{t} + \mathbf{u}$, the output is 1 for success and 0 for failure.

2.3. The GLP Signature Scheme

[10] Güneysu *et al.* introduced a signature scheme, defined by three core algorithms: KeyGen, Sign, and Verify, described below.

(i) **KeyGen:** The key generation algorithm samples two polynomials $\mathbf{s}_1$ and $\mathbf{s}_2$ from the ring $R_1^{p^n} = \mathbb{Z}_p[x]/(x^n + 1)$, along with a polynomial $\mathbf{a}$ from R^{p^n}. It then computes $\mathbf{t} = \mathbf{a}\mathbf{s}_1 + \mathbf{s}_2$, and outputs the secret key as sk $= (\mathbf{s}_1, \mathbf{s}_2)$ together with the public key pk $= (\mathbf{a}, \mathbf{t})$.

(ii) **Sign:** The signing algorithm begins by sampling masking vectors $\mathbf{y}_1$ and $\mathbf{y}_2$ from $R_k^{p^n}$. It then computes the challenge $\mathbf{c} = H\big((\mathbf{a}\mathbf{y}_1 + \mathbf{y}_2)^{(1)}, m\big)$. Subsequently, the response vectors are derived as $\mathbf{z}_1 = \mathbf{s}_1 \cdot \mathbf{c} + \mathbf{y}_1$ and $\mathbf{z}_2 = \mathbf{s}_2 \cdot \mathbf{c} + \mathbf{y}_2$. The algorithm proceeds

only if $\mathbf{z}_1$ and $\mathbf{z}_2$ lie within the required bounds. Next, $\mathbf{z}_2$ is compressed using a suitable compression function to obtain $\mathbf{z}'_2$, such that

$$(\mathbf{az}_1 - \mathbf{ct} + \mathbf{z}_2)^{(1)} = (\mathbf{az}_1 - \mathbf{ct} + \mathbf{z}'_2)^{(1)}.$$

Finally, the algorithm outputs the signature on message m as $(\mathbf{z}_1, \mathbf{z}'_2, \mathbf{c})$.

(iii) **Verify:** The verification algorithm first ensures that $\mathbf{z}_1$ and $\mathbf{z}'_2$ belong to the ring $R_{k-32}^{p^n}$. It then verifies that the challenge $\mathbf{c}$ satisfies

$$\mathbf{c} = H\Big((\mathbf{az}_1 + \mathbf{z}'_2 - \mathbf{c} \cdot \mathbf{t})^{(1)},\, m\Big).$$

If both conditions hold, the signature is accepted as valid.

2.4. Aggregate Signature

In an aggregate signature scheme, individual users generate signatures that can later be combined into a single aggregate signature by an aggregator, who does not need to be one of the users or trusted by them. Following the framework of Boneh *et al.* [7], such a scheme is typically defined using six core algorithms, which collectively handle setup, key generation, signing, verification, aggregation, and aggregate verification.

Setup(n)**:** With the security parameter n as input, the setup algorithm outputs the public parameters pp.

KeyExtract(n, pp)**:** Given parameters n and pp, the KGC produces and distributes to each signer a corresponding public/secret key pair (pk, sk).

Sign(sk_i, μ_i)**:** Using its private key sk_i, the i-th signer produces a signature v_i that authenticates the message μ_i.

Verify(pk_i, μ_i, v_i)**:** Provided the public key pk_i belonging to the signer i along with the message–signature pair (μ_i, v_i). If (μ_i, v_i) meets certain verification criteria, then output 1 and accept; otherwise, output 0 and reject.

Aggregate$(\{pk_i, \mu_i, v_i\}_{i=1}^N)$**:** The aggregation algorithm takes all message-signature pairs (μ_i, v_i) and their corresponding public keys pk_i for $i = 1, \ldots, N$, and produces a single aggregate signature v.

AggregateVerify$(v, \{pk_i, \mu_i\}_{i=1}^N)$**:** Provided the aggregate signature v together with the public keys pk_i and messages μ_i for all $i = 1, \ldots, N$. If the tuple $(v, \{pk_i, \mu_i\}_{i=1}^N)$ meets the specified verification conditions, output 1 (accept); otherwise, output 0 (reject).

2.5. Sequential Aggregate Signature

In a sequential aggregate signature scheme, aggregation occurs only during the signing process. Each signer adds their signature in sequence to the existing aggregate, enforcing a specific order. As a result, the signers must interact with one another throughout the aggregation procedure. Following the framework of Boudgoust *et al.* [6] we present a sequential aggregate signature scheme defined by four core algorithms. These algorithms, which together specify Setup, Gen, SeqSign, and SeqVerify, are summarized below.

Setup(n)**:** The public parameters pp are generated by the setup algorithm, which is initialized with a security parameter n.

Gen(pp)**:** The key generation algorithm uses the public parameters pp to create a pair of keys: a secret key sk for signing and a public key pk for verification.

SeqSign$(sk_i, \mu_i, \mathfrak{L}_{i-1}, \Sigma_{i-1}) \longrightarrow \Sigma_i$**:** The sequential signing algorithm takes as input a signer's secret key sk_i, their message μ_i, the current aggregate list $\mathfrak{L}_{i-1}$ of previous public key-message pairs with $\mathfrak{L}_{i-1} = (pk_1, \mu_1)||\ldots||(pk_{i-1}, \mu_{i-1})$, and the aggregate signature Σ_{i-1}, it produces an updated aggregate signature Σ_i.

SeqVerify$(\mathfrak{L}_N, \Sigma_N) \longrightarrow (0, 1)$**:** Upon receiving the list $\mathfrak{L}_N$ containing N public keys, each with their respective message along with the sequential aggregate signature Σ_N, the sequential verification algorithm determines validity by outputting 1 (accept) or 0 (reject).

3. Some Insecure Lattice-based Aggregate Signature Schemes

In this section, we provide two insecure lattice-based aggregate signature schemes [18,19], the security defect was first identified by XIUHUA LU *et al.* [17].
In both of the schemes [18, 19], all users share the same public key while having different private keys, completely breaking user authentication and enabling any user to impersonate all other users.

4. Some Post Quantum Aggregate Signature

Several post-quantum aggregate signature (PQAS) schemes have been developed in the past few years, differing in terms of their underlying mathematical assumptions, construction techniques, and efficiency trade-offs. Table 1 presents a concise summary of these schemes, highlighting their post-quantum category, security models, underlying hard problems, and the key limitations identified in each work.
In the subsequent sections, we present a detailed study of two representative lattice-based aggregate signature schemes. The first focuses on a general lattice-based aggregate signature, while the second examines a lattice-based sequential aggregate signature scheme. These two constructions are analyzed in terms of their design principles, underlying assumptions, and security properties.

5. Lattice-Based Aggregate Signature

5.1. Overfull: Too Large Aggregate Signatures Based on Lattices

In this paper, Boudgoust *et al.* [4] investigates the adaptation of aggregation techniques from Schnorr signatures to the lattice-based, post-quantum setting. The authors construct a signature scheme that supports public aggregation and prove its security under the hardness of the Module Learning with Errors (MLWE) and Module Short Integer Solution (MSIS) problems. Although the scheme achieves provable security, its aggregated signatures are larger than simply concatenating individual signatures, due to structural limitations inherent to lattice-based cryptography. As a result, the work is primarily pedagogical, highlighting the challenges and subtleties of designing lattice-based aggregate signatures with rigorous security proofs.

5.2. Construction

The construction extends the signature scheme originally presented by Güneysu *et al.* [10], described in section 2.3, serving as the underlying scheme with slight modifications. Suppose the signature scheme is denoted by O_{AS}. Moreover, let N denote the total number of signers,

Table 1: Post-Quantum Aggregate Signature: A Summary

Aggregate Signature Schemes	**PQC Type**	**Security Model**	**Underlying Hard Problem**	**Key Limitation / Observation**
Bansarkhani *et al.* [5]	Lattice	ROM	SIS	Aggregate signature size increases proportionally with the number of signers, limiting scalability for large groups.
Boudgoust *et al.* [3]	Lattice	ROM	M-SIS, M-LWE	Aggregate signature size grows linearly with the number of signers, making compression less efficient.
Katharina *et al.* [4]	Lattice	ROM	M-SIS, M-LWE	Proposed "Overfull" scheme produces aggregate signatures larger than concatenated individual ones, reducing space benefits.
Akira *et al.* [6]	Lattice	ROM	M-SIS, M-LWE	Sequential half aggregation offers minimal improvement only around 1% size reduction compared to concatenation.
Bennian *et al.* [20]	Code	ROM	Permuted Goppa Code Bounded Decoding (PGBD)	Signature size increases linearly with the number of signers, limiting practical scalability.
Khaburzaniya *et al.* [21]	Hash	ROM	Pre-image and collision resistance of the hash function	Individual signatures are large; overall size remains impractical for constrained environments.
Meneghetti *et al.* [22]	Hash	ROM	One-wayness and pre-image sampling indistinguishability	Constant-size aggregation is not achieved; size grows with number of signatures.

$\{S_1, S_2, \ldots, S_N\}$, each of whom intends to sign their respective messages, $\{m_1, m_2, \ldots, m_N\}$. The signature scheme comprises five algorithms O_{AS} =(KeyGen, Sign, Verify, AggSign, AggVerify) as follows.

(i) **KeyGen**: The KeyGen algorithm generates a secret key vector $SK_i = s_i$, for the signer S_i for each $i = 1, \ldots, N$, where each entry lies in the ring $R = \frac{\mathbb{Z}[x]}{\langle x^n+1 \rangle}$ and has coefficients bounded in absolute value by β. The corresponding public key is defined as $PK_i = t_i = [\mathbf{A} \mid I_k] \cdot s_i \in R_q^k$, where $\mathbf{A} \in R_q^{k \times l}$ and $R_q = \frac{\mathbb{Z}_q[x]}{\langle x^n+1 \rangle}$. The algorithm outputs the key pair (SK, PK).

(ii) **Sign**: For each $i = 1, \ldots, N$, the signing algorithm begins by sampling a masking vector $\mathbf{y_i}$ according to a prescribed distribution D. Using this vector, the signer calculates a commitment vector $\mathbf{u_i}$ by computing the product $[\mathbf{A}, |, I_k] \cdot \mathbf{y_i}$. This value $\mathbf{u_i}$, along with the public key $\mathbf{t_i}$ and the message m_i, is then passed to the random oracle H_c to produce a challenge polynomial c_i. This polynomial has exactly d coefficients set to ± 1, with the rest being zero.
The algorithm computes the second signature component as $\mathbf{z_i} = \mathbf{y_i} + \mathbf{s_i} \cdot c_i$. To prevent $\mathbf{z_i}$ from leaking information about the secret key $\mathbf{s_i}$, the potential signature $\sigma_i = (\mathbf{u_i}, \mathbf{z_i})$ is only output with a probability Pr_{rej}, a rejection sampling step that ensures its final distribution is decoupled from $\mathbf{s_i}$.

(iii) **Verify**: To verify the individual signature $\sigma_i = (\mathbf{u_i}, \mathbf{z_i})$, the verifier starts by reconstructing the hash value $c_i = H_c(m_i, \mathbf{t_i}, \mathbf{u_i})$. The verifier then checks two conditions: that the norm of $\mathbf{z_i}$ is smaller than a prescribed bound B, and that $[\mathbf{A} \mid I_k] \cdot \mathbf{z_i} = \mathbf{t_i} \cdot c_i + \mathbf{u_i}$. If both criteria are satisfied, it returns 1 on success and 0 on failure.

(iv) **AggSign**: To compress N signatures $\{\sigma_j\}_{j=1}^{N}$, the AggSign algorithm begins by reconstructing the individual challenges $c_j = H_C(m_j, \mathbf{t}_j, \mathbf{u}_j)$ for each $j = 1, \ldots, N$. Subsequently, the algorithm queries the random oracle $H_e : \{0,1\}^* \to C$ to compute $e_j = H_e(j, c_1, \ldots, c_N)$ for each $j = 1, \ldots, N$. It then evaluates the sum $z = \sum_{j=1}^{N} e_j z_j$ and produces $\sigma = ((u_1, \ldots, u_N), z)$ as an aggregate signature, as long as $\|z\|_2 \leq B'$. If this condition is not satisfied, the algorithm outputs $\perp$.

(v) **AggVerify**: In the verification phase, AggVerify reconstructs each challenge c_j ($j = 1, \ldots, N$) from the commitments u_j, public keys t_j, and messages m_j. It computes $e_j = H_e(j, c_1, \ldots, c_N)$, checks whether the Euclidean norm $\|z\|$ lies within the prescribed bound, and confirms that $[\mathbf{A} \mid I_k] \cdot z = \sum_{j=1}^{N}(t_j \cdot c_j + u_j)$ holds. If all verifications succeed, the algorithm returns 1; otherwise, it returns 0.

5.3. Correctness

The individual signature in the scheme O_{AS} is valid according to the signature scheme proposed by Güneysu *et al.* [10]. Furthermore, the correctness of the aggregated signature σ is a consequence of the linearity property over R_q combined with the validity of each individual signature.

5.4. Security

The security of the signature scheme O_{AS} against existential forgery under an adaptive chosen-key attack in the aggregate setting is proven in the random oracle model, relying on the computational intractability of M-$\mathrm{SIS}_{k,l+1,b}$ and M-$\mathrm{LWE}_{k,l,\beta}$ with appropriately chosen parameters k, l, β, and b.

5.5. Discussion

What is particularly notable about this construction is that the signature scheme that builds upon the signature framework of Güneysu *et al.* [10]. While their construction preserves the security properties of the underlying scheme, it suffers from a significant practical limitation: the resulting signature size is excessively large. Specifically, the size of the aggregate signature in their construction exceeds that of the trivial concatenation of all individual signatures. This observation highlights an open problem in the field: designing an aggregate signature scheme whose overall size is smaller than the straightforward concatenation of individual signatures, while still maintaining provable security in the context of aggregate chosen-key model. Furthermore, Boudgoust *et al.* critically analyze linear compression techniques, which were employed in earlier iterations of their approach, and demonstrate why such methods are fundamentally unsuitable. Linear compression fails to preserve the necessary security guarantees, making it an unreliable strategy for reducing aggregate signature sizes. These observations indicate that while lattice-based aggregation has strong theoretical foundations, there remains a significant practical challenge in reducing signature size without compromising provable security. This motivates further research into efficient, secure aggregation methods that strike an optimal balance between compactness and rigorous security guarantees.

6. Lattice-based Sequential Aggregate Signature

6.1. Sequential Half Aggregation of Lattice-Based Signatures

In this work, Boudgoust *et al.* [6] investigates sequentially aggregating lattice-based digital signatures (SAS) in the context of post-quantum cryptography. The study begins with a review of the current state of the art in signature aggregation and highlights critical security issues in the known Falcon-based SAS constructions by Wang and Wu [9]. The insecurity is demonstrated by generalizing attack strategies shifting from cryptographic constructions based on the discrete logarithm setting to those based on lattice framework.
A new lattice-based SAS scheme is then proposed. However, as is often the case when transitioning from pre-quantum to post-quantum constructions, significant efficiency challenges arise. In particular, the scheme suffers from limitations that appear to be inherent to lattice-based signatures, leading to small compression rates that fall short of expectations.
The proposed construction is compared with other SAS schemes based on lattices, all following the GPV framework. The comparative analysis shows that, to date, no known lattice-based approaches achieves satisfactory compression rates.

6.2. Construction

This signature scheme follows the Fiat-Shamir with Aborts technique [11, 12] described in section 2.2 and can be viewed as a module-based analogue of [10] or a "vanilla" variant of Dilithium. It is referred to as the FSwA-SAS scheme. The scheme comprises four algorithms (Setup, Gen, SeqSign, SeqVerify) which are specified below.

(i) **Setup:** The setup algorithm samples a random matrix $\mathbf{A} \in R_q^{k \times \ell}$. It then extends this matrix to $\overline{\mathbf{A}} = [\mathbf{A} \mid \mathbf{I}_k]$, and outputs $\overline{\mathbf{A}}$ as the public parameter.

(ii) **Gen:** The key generation algorithm repeatedly samples a vector $\mathbf{s} \in R_q^{\ell+k}$ until all its coefficients are invertible. It then computes $\mathbf{t} = \overline{\mathbf{A}}\mathbf{s} \bmod q$. The secret key is defined as $\mathrm{sk} = \mathbf{s}$, and the public key as $\mathrm{pk} = \mathbf{t}$.

(iii) **SeqSign:** The sequential signing algorithm $\mathrm{SeqSign}(m_i, sk_i, \mathsf{L_{i-1}}, \rho_{\mathsf{i-1}})$ generates a new signature ρ_i on the message m_i, given the signer's secret key sk_i, the current transcript $\mathsf{L_{i-1}} = (\mathsf{m_1}, \mathbf{t_1})||\ldots||(\mathsf{m_{i-1}}, \mathbf{t_{i-1}})$, and the previous signature ρ_{i-1}, as follows:

- Parse the previous signature as $(\tilde{\mathbf{u}}_{i-1}, \mathbf{z}_1, \ldots, \mathbf{z}_{i-1}) = \rho_{i-1}$.
- Set the secret signing key component $\mathbf{s}_i = sk_i$.
- Compute $\mathbf{t}_i = \overline{\mathbf{A}}\mathbf{s}_i \bmod q$, and update the transcript $\mathsf{L_i} = \mathsf{L_{i-1}} \parallel (\mathsf{m_i}, \mathsf{t_i})$.

Initially, the response is set to $\mathbf{z}_i = \perp$. While $\mathbf{z}_i = \perp$, the algorithm performs the following steps:

- Sample $\mathbf{y}_i$ from the distribution $\mathcal{D}^{\ell+k}$.
- Compute the masked commitment $\mathbf{u}_i = \overline{\mathbf{A}}\mathbf{y}_i \bmod q$.
- Update the accumulator $\tilde{\mathbf{u}}_i = \tilde{\mathbf{u}}_{i-1} + \mathbf{u}_i \bmod q$.
- Derive the challenge $c_i = H(\mathbf{z}_{i-1}, \tilde{\mathbf{u}}_i, \mathsf{L_i})$.
- Compute the response $\mathbf{z}_i = c_i\mathbf{s}_i + \mathbf{y}_i$.
- Apply rejection sampling: $\mathbf{z}_i = \mathrm{RejSamp}(\mathbf{z}_i, c_i, \mathbf{s}_i)$.
- If the response is invalid, reset $\mathbf{z}_i = \perp$ and repeat.

Once a valid response $\mathbf{z}_i$ is obtained, the algorithm outputs the new signature $\rho_i = (\tilde{\mathbf{u}}_1, \mathbf{z}_1, \ldots, \mathbf{z}_i)$.

(iv) **SeqVerify:** Given a transcript $\mathsf{L_N}$ and a signature ρ_N, the verifier checks that each $\mathbf{t}_i$ is invertible and that every response $\mathbf{z}_i$ lies within the required bound. It then recomputes the transcript $\mathsf{L_i} = (\mathsf{m_1}, \mathbf{t_1})||\ldots||(\mathsf{m_i}, \mathbf{t_i})$, challenge $c_i = H(\mathbf{z}_{i-1}, \tilde{\mathbf{u}}_i, \mathsf{L_i})$, and responses $u_i = \overline{\mathbf{A}}\mathbf{z}_i - c_i.\mathbf{t}_i \bmod q$, and $\tilde{\mathbf{u}}_{i-1} = \tilde{\mathbf{u}}_i - \mathbf{u_i} \bmod q$ and checks if $\tilde{\mathbf{u}}_1 = \mathbf{u}_1$ then return 1 and accept. Otherwise, return 0 and reject.

6.3. Correctness

The correctness of the FSwA-SAS scheme is demonstrated via induction. For $i = 1$, correctness can be directly inferred from the fact that matrix–vector multiplication over R_q is linear.

6.4. Security

The FSwA-SAS scheme achieves full-history unforgeability against chosen-message attacks (FH-UF-CMA) and its security is provably based on the Module-LWE (M-LWE) assumption.

6.5. Discussion

In this paper, the authors constructed a signature scheme using the Fiat–Shamir with Abort paradigm. The correctness of the proposed scheme is established via an induction hypothesis. The security of the scheme is proven in the full-history model as well as in a new partial-signature history-free model. Additionally, the paper identifies vulnerabilities in two existing signature schemes:

1. Falcon-based SAS [9]: The scheme does not uphold the stated security properties because it is vulnerable to a forgery attack.

2. Dilithium-based interactive multi-signature [14]: The scheme reveals a portion of the secret key as a result of improper use of the Bai–Galbraith HighBits optimization [15].

The authors also provided a comparative analysis of their proposed scheme with other SAS schemes based on lattices following the GPV framework. Finally, the paper discusses several open problems and potential directions for future research.

7. Conclusion

In this survey, we reviewed one aggregate signature and one sequential aggregate signature scheme based on lattices. We observed that lattice-based schemes remain promising candidates for post-quantum cryptography due to their strong security foundations, but they often face challenges related to parameter selection, signature size, and efficiency trade-offs. Overall, lattice-based aggregate signature schemes represent an exciting area of post-quantum cryptography, but further research is needed to bridge the gap between theoretical constructions and practical deployment.

References

[1] Boudgoust, K., and Roux-Langlois, A. (2021). Non-interactive half-aggregate signatures based on module lattices-a first attempt. Cryptology ePrint Archive.

[2] Boneh, Dan, and Sam Kim. "One-time and interactive aggregate signatures from lattices." preprint 4 (2020): 3.

[3] Boudgoust, Katharina, and Adeline Roux-Langlois. "Compressed Linear Aggregate Signatures Based on Module Lattices." IACR Cryptol. ePrint Arch. 2021 (2021): 263.

[4] Boudgoust, Katharina, and Adeline Roux-Langlois. "Overfull: Too large aggregate signatures based on lattices." The Computer Journal 67, no. 2 (2024): 719-727.

[5] El Bansarkhani, Rachid, and Johannes Buchmann. "Towards lattice based aggregate signatures." In International Conference on Cryptology in Africa, pp. 336-355. Cham: Springer International Publishing, 2014.

[6] Boudgoust, Katharina, and Akira Takahashi. "Sequential half-aggregation of lattice-based signatures." In European Symposium on Research in Computer Security, pp. 270-289. Cham: Springer Nature Switzerland, 2023.

[7] Boneh, Dan, Craig Gentry, Ben Lynn, and Hovav Shacham. "Aggregate and verifiably encrypted signatures from bilinear maps." In International conference on the theory and applications of cryptographic techniques, pp. 416-432. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[8] Micciancio, Daniele, and Shafi Goldwasser. Complexity of lattice problems: a cryptographic perspective. Vol. 671. Springer Science and Business Media, 2002.

[9] Wang, Zhipeng, and Qianhong Wu. "A practical lattice-based sequential aggregate signature." In International Conference on Provable Security, pp. 94-109. Cham: Springer International Publishing, 2019.

[10] Güneysu, Tim, Vadim Lyubashevsky, and Thomas Pöppelmann. "Practical lattice-based cryptography: A signature scheme for embedded systems." In International Workshop on Cryptographic Hardware and Embedded Systems, pp. 530-547. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[11] Lyubashevsky, Vadim. "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures." In International conference on the theory and application of cryptology and information security, pp. 598-616. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[12] Lyubashevsky, Vadim. "Lattice signatures without trapdoors." In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 738-755. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[13] Langlois, A., and D. Stehlé. Worst-case to average-case reductions for module lattices. Des. Codes Crypt. 75 (3), 565–599 (2014).

[14] Fukumitsu, Masayuki, and Shingo Hasegawa. "A lattice-based provably secure multisignature scheme in quantum random oracle model." In International Conference on Provable Security, pp. 45-64. Cham: Springer International Publishing, 2020.

[15] Bai, Shi, and Steven D. Galbraith. "An improved compression technique for signatures based on learning with errors." In Cryptographers' Track at the RSA Conference, pp. 28-47. Cham: Springer International Publishing, 2014.

[16] Pointcheval, David, and Jacques Stern. "Security arguments for digital signatures and blind signatures." Journal of cryptology 13, no. 3 (2000): 361-396.

[17] Lu, Xiuhua, Wei Yin, Qiaoyan Wen, Zhengping Jin, and Wenmin Li. "A lattice-based unordered aggregate signature scheme based on the intersection method." IEEE Access 6 (2018): 33986-33994.

[18] Zhang, Peng, Yu Jianping, and W. A. N. G. Ting. "A homomorphic aggregate signature scheme based on lattice." Chinese Journal of Electronics 21, no. 4 (2012): 701-704.

[19] Jing, Zhengjun. "An efficient homomorphic aggregate signature scheme based on lattice." Mathematical Problems in Engineering 2014, no. 1 (2014): 536527.

[20] Dou, Bennian, Lei Xu, Xiaoling Yu, Lin Mei, and Cong Zuo. "Code-based Sequential Aggregate Signature Scheme." Computers, Materials and Continua 73, no. 3 (2022).

[21] Khaburzaniya, Irakliy, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. "Aggregating and thresholdizing hash-based signatures using STARKs." In Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, pp. 393-407. 2022.

[22] Meneghetti, Alessio, and Edoardo Signorini. "History-free sequential aggregation of hash-and-sign signatures." In Cryptographers' Track at the RSA Conference, pp. 187-223. Cham: Springer Nature Switzerland, 2024.