

Lemur+: Compact Post-Quantum Synchronized Multi-Signatures and Multi-Hop Aggregation

Yini Lin^{†,1}, Hongxiao Wang^{†,1}, Muhammed F. Esgin[†], Amin Sakzad[†], and Ron Steinfeld[†]

[†]*Monash University*

Abstract

In this work, we introduce Lemur+, a compact post-quantum synchronized multi-signature scheme based on standard lattice assumptions (namely, Module LWE and Module SIS). Lemur+ builds on Lemur (CCS 2026) and targets large-scale, long-term distributed applications such as blockchain consensus while supporting non-interactive aggregation. Lemur+ maintains an almost constant signature size of about 56 KB even for one million signers and a 42-year key lifetime, compared to 491 KB in Lemur, yielding about $8.8\times$ reduction in communication/storage. Our end-to-end implementation demonstrates practical runtime performance: 7.38 s for signature aggregation and 0.29 s for verification with 2^{13} signers. These are roughly within $2\times$ of the corresponding Lemur runtimes.

The key to Lemur+ is our proposed Homomorphic Vector Commitment with Succinct Opening Proof (HVC-SOP), a new primitive that augments standard HVC with succinct opening proofs. We show that HVC-SOP can be generically combined with key-homomorphic one-time signatures (KOTS) to construct non-interactive synchronized multi-signatures, and prove the security of the resulting construction against rogue-key attacks. We then instantiate HVC-SOP using the lattice-based LaBRADOR proof system (CRYPTO 2023), yielding Lemur+. Finally, we extend Lemur+ to support multi-hop aggregation, allowing signatures to be recursively aggregated across distributed network trees. Beyond Lemur+, our HVC-SOP may be broadly applicable and of independent interest.

1 Introduction

Multi-signatures [21, 24] allow a group of signers to authenticate a common message with a single aggregate signature. They are particularly useful in consensus protocols, distributed ledgers, and certificate systems, where many parties repeatedly attest to the same statement. In proof-of-

stake blockchains such as Ethereum and DFINITY [16], multi-signatures reduce the communication required to disseminate votes and the cost of verifying them.

Constructing practical post-quantum multi-signatures remains challenging. Many existing proposals [3, 7, 10, 18, 19] require interaction among signers, which complicates deployment in unreliable or geographically distributed networks. *Non-interactive* constructions [1, 6, 14–17, 22, 23, 26] avoid this interaction, but face a difficult trade-off between signature size and verification cost. As summarized in Table 1, schemes with compact aggregate signatures often require expensive verification and high memory consumption during aggregation, while schemes with faster verification may produce signatures hundreds or thousands of kilobytes in size. At 2^{15} signers, for example, LaZer [23] and Orthus [6] produce aggregate signatures below approximately 100 KB, but their reported verification times are $19\times$ and $4\times$ slower than Lemur [22], respectively.

In this work, we focus on *synchronized* multi-signatures, where each signer signs at most one message per time slot, and only signatures for the same message and time slot are aggregated. This setting naturally captures many consensus applications, where validator attestations are associated with a specific slot. Our goal is to construct a post-quantum synchronized multi-signature that improves both the aggregate signature size and verification time over the state-of-the-art.

Our starting point is Lemur [22], which combines a key-homomorphic one-time signature (KOTS) with a homomorphic vector commitment (HVC) to transform a one-time primitive into a many-time synchronized multi-signature. The homomorphic structure allows individual signatures to be linearly aggregated, while verification remains efficient. The main limitation of Lemur is the size of the HVC opening. To sign at time t , a signer reveals an opening proving that the one-time public key at time t is consistent with its commitment. The opening contains a path from the root to the leaf, which grows with the Merkle tree depth and, unfortunately, becomes large in practice. As shown in Table 2, for a 42-year key life-

¹Yini Lin and Hongxiao Wang are co-first authors.

Scheme	N	$ \sigma_{\text{agg}} $	Sig. Agg. Time	Agg. Vrfy Time	Sig. Agg. RAM
Chipmunk [16]	2^{13}	479	–	–	–
	2^{15}	>516	–	–	–
LaZer [23]	2^{13}	≈ 75	5.56	3.20	2.03
	2^{15}	≈ 75	21.70	12.80	8.31
Orthus [6]	2^{13}	≈ 100	5.00	1.40	1.34
	2^{15}	≈ 100	16.80	2.90	4.94
HAPPIER [26]	2^{10}	2396	220.60	0.14	<16.00
	2^{16}	2959	795.00	0.16	<16.00
Lemur [22]	2^{13}	307	6.95	0.16	0.12
	2^{15}	341	27.65	0.66	0.12
Lemur+ (this work)	2^{13}	53	7.75	0.29	0.20
	2^{15}	54	30.91	0.80	0.20

Table 1: Performance comparison of state-of-the-art post-quantum non-interactive multi-signature schemes. Aggregate signature size $|\sigma_{\text{agg}}|$ is reported in KB, RAM usage during signature aggregation in GB, and all runtimes (signature aggregation and verification) in seconds. Note that the reported runtimes may not be directly comparable due to varying implementation-specific optimizations, such as multi-threading or AVX acceleration. “–”: not (explicitly) reported.

time¹ and 2^{20} signers, the HVC opening alone accounts for over 98% of Lemur’s aggregate signature size.

We address this bottleneck by introducing a new primitive, Homomorphic Vector Commitment with Succinct Opening Proof (HVC-SOP). HVC-SOP retains the homomorphic opening of an HVC, which is essential for linear aggregation, but augments it with a succinct proof that certifies the aggregated opening without transmitting the full opening. A key property is that a succinct opening proof can be generated for a random linear combination of individual openings. This allows us to preserve Lemur’s efficient aggregation structure while replacing its large final opening with a short proof. We give a generic construction of synchronized multi-signatures from HVC-SOP and KOTS and prove security against rogue-key attacks. We then instantiate HVC-SOP using the lattice-based LaBRADOR proof system [5], obtaining Lemur+.

1.1 Our Contributions

Our contributions are summarized as follows.

1. **New Primitive: HVC-SOP.** We introduce Homomorphic Vector Commitment with Succinct Opening Proof (HVC-SOP), a homomorphic vector commitment equipped with succinct opening proofs. In addition to the homomorphism and binding properties required by a standard HVC, HVC-SOP supports succinct opening proofs for individual and

¹ Assuming a one-time key is consumed every 10 seconds as in [16].

τ	Key lifetime	Lemur		Lemur+		Improvement	
		HVC	agg. sig.	HVC-SOP	agg. sig.	HVC	agg. sig.
24	5 years	435	443	46	56	$9.5\times$	$7.9\times$
25	11 years	451	459	46	57	$9.8\times$	$8.1\times$
26	21 years	467	475	46	57	$10.2\times$	$8.3\times$
27	42 years	483	491	46	56	$10.5\times$	$8.8\times$

Table 2: HVC opening / HVC-SOP opening proof and aggregate signature size of Lemur and Lemur+ (KB) for $N = 2^{20}$ under Rice-encoding. The symbol τ denotes the Merkle tree depth allowing 2^τ signing operations.

aggregated openings. In our instantiation, HVC-SOP opening proof is roughly $10\times$ smaller than an HVC opening of Lemur (Table 2).

2. **Generic synchronized multi-signature construction.** We show how to combine an HVC-SOP with a KOTS to obtain a synchronized multi-signature scheme in Figure 3, and formalize its security in Section 4.2.
3. **Lemur+: an efficient lattice-based instantiation.** We instantiate an HVC-SOP using LaBRADOR [5] and combine it with Lemur’s KOTS to obtain Lemur+. The resulting construction relies on standard Module-LWE and Module-SIS assumptions. We implement Lemur+ using the LaZer library [23]. For $N = 2^{20}$ signers and $\tau = 27$, corresponding to a 42-year key lifetime, Lemur+ reduces aggregate signature size from 491 KB to approximately 56 KB, an $8.8\times$ reduction over Lemur. The signature aggregation and aggregate verification times remain comparable to those of Lemur, with both incurring less than a $2\times$ overhead (Table 1). As an additional contribution, we extend Lemur+ to support multi-hop aggregation. Discussions are provided in Section 6.

The runtime comparison in Table 1 should be interpreted with care: the Lemur implementation exploits multi-threading and scheme-specific optimizations, whereas Lemur+ implementation is constrained by the current Labrador implementation in LaZer library [23]. LaZer provides AVX-512 acceleration, but is a single-threaded, general-purpose lattice proof library rather than an implementation specialized for our purposes. Therefore, our Lemur+ implementation is constrained by settings used in LaZer such as not being able to exploit a single NTT-friendly prime modulus.

1.2 Technical Overview

Homomorphic Vector Commitment with Succinct Opening Proof. Our main technical contribution is addressing the HVC bottleneck in Lemur [22]. Our key insight is that the HVC opening is auxiliary information to verify the committed value, but is not otherwise for further computation. To

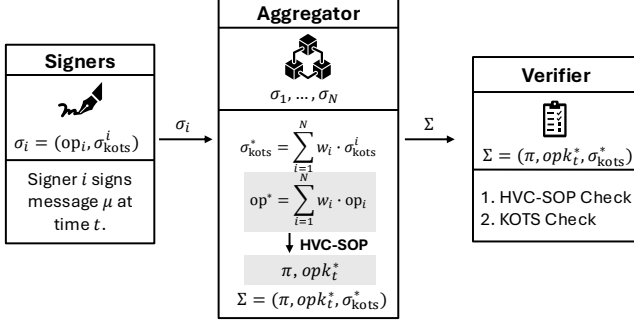


Figure 1: High Level Overview of Lemur+.

this end, we introduce Homomorphic Vector Commitment with Succinct Opening Proof (HVC-SOP). The core idea is to treat the HVC opening as a witness to the HVC verification relation and prove this relation using a succinct proof system. In our lattice-based construction, the relation captures both the underlying algebraic consistency checks and the norm constraints. The key advantage is that the opening proof size is only polylogarithmic in the original explicit opening. We highlight that zero-knowledge is not required for the witness, which also simplifies proof.

To preserve the homomorphism, we retain explicit individual opening for HVC-SOP, and replace only the aggregated opening with a succinct opening proof. This makes it particularly suited to our multi-signature setting: the algebraic homomorphism is used to combine commitments and openings, while the succinct opening proof is used to replace the final expensive opening, which also provides a verification cost independent of the number of signers. We stress that HVC-SOP is a general primitive of independent interest: in applications that do not require homomorphic aggregation, any explicit opening can be directly replaced by its succinct opening proof to generically reduce communication.

A Generic Synchronized Multi-Signature Construction Based on HVC-SOP and KOTS. To demonstrate the applicability of HVC-SOP, we give a generic construction of synchronized multi-signatures from HVC-SOP and KOTS. As illustrated in Figure 1, each signer produces an individual KOTS signature σ_{kots}^i alongside an HVC opening op_i . The aggregator linearly aggregates individual components and applies HVC-SOP to compress the aggregated opening op^* into a succinct opening proof π (shaded part). The final aggregate signature Σ consists of the HVC-SOP opening proof π , the aggregated one-time public key opk_t^* , and the aggregated KOTS signature σ_{kots}^* . Finally, the verifier performs the HVC-SOP and KOTS check.

The main challenge is proving unforgeability. In Lemur [22], the security reduction extracts an alternative HVC opening from a forged aggregate signature and reduces the forgery to either a violation of HVC position binding or KOTS un-

forgeability. In our setting, the explicit opening is replaced by a succinct opening proof and is therefore not directly available to the reduction. We overcome this by leveraging the knowledge-extractability of HVC-SOP: from a successful forgery, we extract the underlying HVC opening and then apply the same reduction strategy as in Lemur. The detailed construction and security proof are given in Figure 3 and Section 4.2, respectively.

A Concrete Instantiation: Lemur+. We instantiate HVC-SOP using lattice-based proof system LaBRADOR [5] and combine it with the KOTS construction underlying Lemur [22], resulting in Lemur+. However, there are two main problems with this approach. First, Lemur’s HVC encoding is incompatible with LaBRADOR’s proof relation. Specifically, Lemur reduces the HVC size by encoding the opening and enforcing a norm-bound check on its projection, whose norm can exceed $q/2$, where q is the HVC modulus. In contrast, LaBRADOR natively supports proof relations over small-norm witnesses. This mismatch makes the original encoding unsuitable for our instantiation. We therefore remove the encoding layer from our HVC-SOP construction. This modification not only makes the relation compatible with LaBRADOR’s small-norm witness model, but also simplifies proof generation by removing the additional encoding constraints and the corresponding unsuitable norm checks.

Second, Lemur uses separate moduli for its HVC and KOTS, which enables it to optimize the aggregate signature size. LaBRADOR, however, is most efficient when the modulus has a bit length that is a multiple of 8. Supporting separate moduli would therefore introduce additional modulus-switching operations across HVC-SOP, KOTS, and LaBRADOR. In Lemur+, we instead use a common modulus across all three components, choosing the smallest feasible LaBRADOR modulus that is also closest in size to the moduli used by Lemur. This design eliminates modulus switching and substantially simplifies both the analysis of the construction.

These modifications yield a conceptually simple and modular construction. However, instantiating it with concrete parameters and achieving an efficient implementation requires careful optimization across the three underlying components. We defer the detailed parameter selection to Section 7 and the implementation details to Section 8.

1.3 Related Work

Post-quantum non-interactive multi-signatures have attracted significant attention in recent years. Existing constructions can be broadly categorized into two directions: generic SNARK-based approaches and specialized synchronized schemes.

Generic SNARK-Based Constructions. A line of work combines hash-based signatures with SNARK proof system to

obtain compact aggregate signatures. Representative schemes include DKKW25 [14] and its successor LeanSig [15] and HAPPIER [26]. These constructions naturally support aggregation of signatures on distinct messages and benefit from the fast verification time. However, their aggregate signature size is large, typically ranging from 2 to 3 MB.

Another line of work combines lattice-based signatures with lattice-compatible proof systems. Representative examples include the LaBRADOR-based [5] construction of Aardal et al. [1], LaZer [23], and the recent proposal Orthus [6]. All these constructions aggregate the NIST-standardized signature scheme Falcon [25], achieving significantly smaller aggregate signatures than previous hash-based approaches. Nevertheless, verification complexity is dominated by the underlying proof system and scales with the size of the proof system. As a result, existing implementations incur substantially higher verification time and also RAM requirement.

Specialized Synchronized Constructions. Rather than relying on generic proof systems, synchronized schemes aim to aggregate signatures on the same message during the same time period. This line includes Squirrel [17], further improved by Chipmunk [16] and Lemur [22]. By exploiting the homomorphic property of the underlying building blocks, these schemes avoid the computational overhead of general-purpose proof systems. Consequently, verification after public-key aggregation is independent of the number of signers, leading to significantly lower verification time.

2 Preliminaries

Notations. For the security parameter λ , let $\text{poly}(\lambda)$ denote an arbitrary function bounded by a polynomial in λ . A function $\text{negl} : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ is said to be negligible if, for every polynomial p , there exists $\lambda_0 \in \mathbb{N}$ such that $\text{negl}(\lambda) < 1/p(\lambda)$ for all $\lambda \geq \lambda_0$. For $n \in \mathbb{N}^+$, we identify $\mathbb{Z}_n$ with the representative set $\mathbb{Z} \cap (-n/2, n/2]$. Let $[n] = \{1, \dots, n\}$ and $[n] = \{0, \dots, n-1\}$. For a bit string $s = (s_1, \dots, s_\tau) \in \{0, 1\}^\tau$, s_i denotes the i -th bit and $s_{<i} := (s_1, \dots, s_{i-1})$ denotes the prefix of length $i-1$. For $i \in [\tau]$, $\text{bin}_\tau(i) \in \{0, 1\}^\tau$ denotes the big-endian binary decomposition of i . We use bold lower-case letters for column vectors (e.g., $\mathbf{a}$) and bold upper-case for matrices (e.g., $\mathbf{A}$). For matrices $\mathbf{A} = (a_{ij}) \in R^{m \times n}$ and $\mathbf{B} \in R^{\ell \times k}$, their Kronecker product is denoted as $\mathbf{A} \otimes \mathbf{B} \in R^{m\ell \times nk}$.

Let d be a power of 2 and q be an integer modulus. Denote $R = \mathbb{Z}[X]/(X^d + 1)$ and $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. Multiplication between elements of R and R_q is performed modulo q by default. For a polynomial $f = \sum_{i=0}^{d-1} f_i X^i \in R$, the ℓ^p ($p \geq 1$) and ℓ^∞ norms are defined as:

$$\|f\|_p := \left(\sum_{i=0}^{d-1} |f_i|^p \right)^{1/p}, \quad \|f\|_\infty := \max_{i \in [d]} |f_i|.$$

For a vector $\mathbf{f} = (f_1, \dots, f_n) \in R^n$, we extend these norms as:

$$\|\mathbf{f}\|_p := \left(\sum_{i=1}^n \|f_i\|_p^p \right)^{1/p}, \quad \|\mathbf{f}\|_\infty := \max_{i \in [n]} \|f_i\|_\infty.$$

Norms over R_q are computed coefficient-wise using centered representatives. Finally, we define the set of ternary polynomials (coefficients in $\{-1, 0, 1\}$) with weight α_w as $\mathcal{T}_{\alpha_w} = \{f \in R : \|f\|_\infty = 1 \wedge \|f\|_1 = \alpha_w\}$.

Lemma 2.1 ([16, Lemma 4]). *Let $d, \alpha_w, N, \beta \in \mathbb{N}^+$ with d a power of 2. Then for any $x_1, \dots, x_N \in R$ with $\|x_i\|_\infty \leq \beta$ for all $i \in [N]$, and for any $B \geq \beta$, it holds that*

$$\Pr_{w_1, \dots, w_N \leftarrow \mathcal{T}_{\alpha_w}} \left[\left\| \sum_{i=1}^N w_i x_i \right\|_\infty > B \right] \leq 2d \cdot \exp\left(-\frac{B^2}{2\beta^2 \alpha_w N}\right).$$

Algorithm $F_{\mathcal{A}}(x)$

1. Pick coins ρ for $\mathcal{A}$ at random
2. $h_1, \dots, h_q \leftarrow H$
3. $(I, \sigma) \leftarrow \mathcal{A}(x, h_1, \dots, h_q; \rho)$
4. **If** $I = 0$ **then return** $(0, \varepsilon, \varepsilon)$
5. $h'_1, \dots, h'_q \leftarrow H$
6. $(I', \sigma') \leftarrow \mathcal{A}(x, h_1, \dots, h_{I-1}, h'_I, \dots, h'_q; \rho)$
7. **If** $(I = I' \text{ and } h_I \neq h'_I)$ **then return** $(1, \sigma, \sigma')$
8. **Else return** $(0, \varepsilon, \varepsilon)$

Figure 2: The forking algorithm $F_{\mathcal{A}}$.

Lemma 2.2 (General Forking Lemma, [4]). *Fix an integer $q \geq 1$ and a set H of size $h \geq 2$. Let $\mathcal{A}$ be a randomized algorithm that on input $x, h_1, \dots, h_q$ returns a pair, the first element of which is an integer in the range $0, \dots, q$ and the second element of which we refer to as a side output. Let IG be a randomized algorithm that we call the input generator. The accepting probability of $\mathcal{A}$, denoted acc , is defined as the probability that $J \geq 1$ in the experiment*

$$x \leftarrow \text{IG}; h_1, \dots, h_q \leftarrow H; (J, \sigma) \leftarrow \mathcal{A}(x, h_1, \dots, h_q).$$

The forking algorithm $F_{\mathcal{A}}$ associated to $\mathcal{A}$ is the randomized algorithm that takes input x proceeds as Figure 2. Let

$$\text{frk} = \Pr[b = 1 : x \leftarrow \text{IG}; (b, \sigma, \sigma') \leftarrow F_{\mathcal{A}}(x)].$$

Then

$$\text{frk} \geq \text{acc} \cdot \left(\frac{\text{acc}}{q} - \frac{1}{h} \right).$$

Alternatively,

$$\text{acc} \leq \frac{q}{h} + \sqrt{q \cdot \text{frk}}.$$

2.1 Key-Homomorphic One-Time Signature

Definition 2.1 (Key-Homomorphic One-Time Signature (KOTS), [16, 17, 22]). Let $\mathcal{S}_{\text{pk}}$ and $\mathcal{S}_{\text{sig}}$ be R -modules representing the public key and signature spaces, respectively. A key-homomorphic one-time signature scheme over R is a tuple of PPT algorithms $\text{KOTS} = (\text{Setup}, \text{KGen}, \text{Sign}, \text{iVrfy}, \text{sVrfy}, \text{wVrfy})$ defined as:

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: On input the security parameter λ , outputs public parameters pp .
- $(\text{osk}, \text{opk}) \leftarrow \text{KGen}(\text{pp})$: On input public parameters pp , outputs a one-time secret / public key pair (osk, opk) with $\text{opk} \in \mathcal{S}_{\text{pk}}$.
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{osk}, \mu)$: On input public parameters pp , a secret key osk , and a message $\mu \in \{0, 1\}^*$, outputs a signature $\sigma \in \mathcal{S}_{\text{sig}}$.
- $b \leftarrow \text{Vrfy}(\text{pp}, \text{opk}, \mu, \sigma)$: For $\text{Vrfy} \in \{\text{iVrfy}, \text{sVrfy}, \text{wVrfy}\}$, takes as input public parameters pp , a public key opk , a message μ , and a signature σ , and outputs a bit b indicating acceptance or rejection.

A KOTS scheme is required to satisfy the following properties.

Definition 2.2 (Individual Correctness). Let $\epsilon_{\text{cor}} \in [0, 1)$. A KOTS scheme KOTS is ϵ_{cor} -individually correct if, for every $\lambda \in \mathbb{N}^+$ and $\mu \in \{0, 1\}^*$, it holds that

$$\Pr \left[\text{iVrfy}(\text{pp}, \text{opk}, \mu, \sigma) = 1 \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ (\text{osk}, \text{opk}) \leftarrow \text{KGen}(\text{pp}), \\ \sigma \leftarrow \text{Sign}(\text{pp}, \text{osk}, \mu) \end{array} \right] \geq 1 - \epsilon_{\text{cor}}.$$

Definition 2.3 (Probabilistic Homomorphism). Let $N \in \mathbb{N}^+$, $\epsilon_{\text{homK}} \in [0, 1)$, and $W \subseteq R$ be a finite set. A KOTS scheme KOTS is $(N, W, \epsilon_{\text{homK}})$ -probabilistically homomorphic if, for every $\lambda \in \mathbb{N}^+$, $\mu \in \{0, 1\}^*$, $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, and any tuple list $\{(\text{opk}_i, \sigma_i)\}_{i \in [N]} \subset \mathcal{S}_{\text{pk}} \times \mathcal{S}_{\text{sig}}$ satisfying $\text{iVrfy}(\text{pp}, \text{opk}_i, \mu, \sigma_i) = 1$ for all $i \in [N]$, it holds that

$$\Pr_{w_i \leftarrow W} \left[\text{sVrfy} \left(\text{pp}, \sum_{i=1}^N w_i \cdot \text{opk}_i, \mu, \sum_{i=1}^N w_i \cdot \sigma_i \right) = 1 \right] \geq 1 - \epsilon_{\text{homK}}.$$

Definition 2.4 (Robust Homomorphism). A KOTS scheme KOTS is robustly homomorphic if, for every $\lambda \in \mathbb{N}^+$, $\mu \in \{0, 1\}^*$, $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, and any pair $(\text{opk}_b, \sigma_b) \in \mathcal{S}_{\text{pk}} \times \mathcal{S}_{\text{sig}}$ for $b \in \{0, 1\}$ satisfying $\text{sVrfy}(\text{pp}, \text{opk}_b, \mu, \sigma_b) = 1$, it holds that $\text{wVrfy}(\text{pp}, \text{opk}_0 - \text{opk}_1, \mu, \sigma_0 - \sigma_1) = 1$.

Definition 2.5 (Multi-User Existential Unforgeability under Rerandomized Keys). Let $W' := \{w_0 - w_1 : w_0, w_1 \in W \wedge w_0 \neq w_1\}$. A KOTS scheme KOTS is W' -existentially

unforgeable under rerandomized keys (EUF-RK) if, for every $\lambda, N \in \mathbb{N}^+$, and PPT adversary $\mathcal{A}$, it holds that

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda); \\ (\text{osk}_i, \text{opk}_i) \leftarrow \text{KGen}(\text{pp}) \text{ for all } i \in [N]; \\ (i^*, \mu^*, \sigma^*, w^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Sign}}}(\text{pp}, \{\text{opk}_i\}_{i \in [N]}); \\ i^* \in [N] \wedge i^* \notin Q_{\text{keys}} \wedge \mu^* \notin Q_{i^*} \wedge \\ w^* \in W' \wedge \text{wVrfy}(\text{pp}, w^* \text{opk}_{i^*}, \mu^*, \sigma^*) = 1 \end{array} \right] \leq \text{negl}(\lambda),$$

where $\mathcal{O}_{\text{Sign}}(i, \mu)$ maintains a set Q_{keys} , initially empty. On query (i, μ) , if $i \in [N]$ and $i \notin Q_{\text{keys}}$, it sets $Q_{\text{keys}} \leftarrow Q_{\text{keys}} \cup \{i\}$ and returns $\text{Sign}(\text{pp}, \text{osk}_i, \mu)$; otherwise, it returns $\perp$.

2.2 Homomorphic Vector Commitment

A homomorphic vector commitment (HVC) supports committing to a vector of messages and opening an individual position. Its homomorphic property additionally allows commitments and openings to be linearly combined.

Definition 2.6 (Homomorphic Vector Commitment (HVC), [16, 17, 22]). Let $\tau, d, \eta, \beta_{\text{agg}} \in \mathbb{N}^+$, where d is a power of two, and let $R = \mathbb{Z}[X]/(X^d + 1)$. Let $\mathcal{S}_{\text{dom}}$, $\mathcal{S}_{\text{com}}$, and $\mathcal{S}_{\text{open}}$ be R -modules. A homomorphic vector commitment scheme for vectors of length 2^τ over $\mathcal{S}_{\text{dom}}$ is a tuple of PPT algorithms $\text{HVC} = (\text{Setup}, \text{Com}, \text{Open}, \text{iVrfy}, \text{sVrfy}, \text{wVrfy})$ defined as follows.

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: On input the security parameter λ , the algorithm outputs public parameters pp .
- $\mathbf{c} \leftarrow \text{Com}(\text{pp}, \mathbf{m})$: On input public parameters pp and a vector $\mathbf{m} = (\mathbf{m}_1, \dots, \mathbf{m}_{2^\tau}) \in \mathcal{S}_{\text{dom}}^{2^\tau}$, the algorithm outputs a commitment $\mathbf{c} \in \mathcal{S}_{\text{com}}$.
- $\text{op} \leftarrow \text{Open}(\text{pp}, \mathbf{c}, \mathbf{m}, t)$: On input public parameters pp , a commitment $\mathbf{c}$, a committed vector $\mathbf{m}$, and an index $t \in [2^\tau]$, the algorithm outputs an opening $\text{op} \in \mathcal{S}_{\text{open}}$.
- $\mathbf{m}_t / \perp \leftarrow \text{iVrfy}(\text{pp}, \mathbf{c}, t, \text{op}, \eta)$: On input public parameters pp , a commitment $\mathbf{c}$, an index t , an opening op , and a norm bound η , the algorithm outputs a message $\mathbf{m}_t \in \mathcal{S}_{\text{dom}}$ or $\perp$.
- $\mathbf{m}_t / \perp \leftarrow \text{sVrfy}(\text{pp}, \mathbf{c}, t, \text{op}, \beta_{\text{agg}})$: On input public parameters pp , a commitment $\mathbf{c}$, an index t , an opening op , and a norm bound β_{agg} , the algorithm outputs a message $\mathbf{m}_t \in \mathcal{S}_{\text{dom}}$ or $\perp$.
- $\mathbf{m}_t / \perp \leftarrow \text{wVrfy}(\text{pp}, \mathbf{c}, t, \text{op}, 2\beta_{\text{agg}})$: On input public parameters pp , a commitment $\mathbf{c}$, an index t , an opening op , and a norm bound $2\beta_{\text{agg}}$, the algorithm outputs a message $\mathbf{m}_t \in \mathcal{S}_{\text{dom}}$ or $\perp$.

An HVC is required to satisfy the following properties.

Definition 2.7 (Individual Correctness). An HVC scheme HVC is individually correct if, for every $\lambda \in \mathbb{N}^+$, $\mathbf{m} \in \mathcal{S}_{\text{dom}}^{2^\tau}$, and $t \in [2^\tau]$, it holds that

$$\Pr \left[\text{iVrfy}(\text{pp}, \mathbf{c}, t, \text{op}, \eta) = \mathbf{m}_t \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ \mathbf{c} \leftarrow \text{Com}(\text{pp}, \mathbf{m}), \\ \text{op} \leftarrow \text{Open}(\text{pp}, \mathbf{c}, \mathbf{m}, t) \end{array} \right] = 1.$$

Definition 2.8 (Probabilistic Homomorphism). Let $N \in \mathbb{N}^+$, $\epsilon_{\text{homH}} \in [0, 1)$, and $W \subseteq R$ be a finite set. An HVC scheme HVC is $(N, W, \epsilon_{\text{homH}})$ -probabilistically homomorphic if, for every $\lambda \in \mathbb{N}^+$, $t \in [2^\tau]$, $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, and any tuple list $\{(\mathbf{c}_i, \text{op}_i)\}_{i \in [N]} \subset \mathcal{S}_{\text{com}} \times \mathcal{S}_{\text{open}}$ satisfying $\text{iVrfy}(\text{pp}, \mathbf{c}_i, t, \text{op}_i, \eta) = \mathbf{m}_t^i \neq \perp$ for all $i \in [N]$, it holds that

$$\Pr_{w_i \leftarrow W} \left[\text{sVrfy} \left(\text{pp}, \sum_{i=1}^N w_i \cdot \mathbf{c}_i, t, \sum_{i=1}^N w_i \cdot \text{op}_i, \beta_{\text{agg}} \right) = \sum_{i=1}^N w_i \cdot \mathbf{m}_t^i \right] \geq 1 - \epsilon_{\text{homH}},$$

Definition 2.9 (Robust Homomorphism). Let $\delta \geq 1$. An HVC scheme HVC is δ -robustly homomorphic if, for every $\lambda \in \mathbb{N}^+$, $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, $t \in [2^\tau]$, any $(\mathbf{c}_b, \text{op}_b) \in \mathcal{S}_{\text{com}} \times \mathcal{S}_{\text{open}}$ satisfying $\text{sVrfy}(\text{pp}, \mathbf{c}_b, t, \text{op}_b, \delta \beta_{\text{agg}}) = \mathbf{m}_t^b \neq \perp$ for $b \in \{0, 1\}$, it holds that $\text{wVrfy}(\text{pp}, \mathbf{c}_0 - \mathbf{c}_1, t, \text{op}_0 - \text{op}_1, 2\delta \beta_{\text{agg}}) = \mathbf{m}_t^0 - \mathbf{m}_t^1$.

Definition 2.10 (Position Binding). Let $\delta \geq 1$. An HVC scheme HVC is δ -position binding if, for every PPT adversary $\mathcal{A}$, it holds that

$$\Pr \left[\begin{array}{l} \mathbf{m}_t^0 \neq \mathbf{m}_t^1 \wedge \\ \mathbf{m}_t^0 \neq \perp \wedge \mathbf{m}_t^1 \neq \perp \wedge \\ \mathbf{m}_t^0 := \text{wVrfy}(\text{pp}, \mathbf{c}, t, \text{op}_0, 2\delta \beta_{\text{agg}}) \wedge \\ \mathbf{m}_t^1 := \text{wVrfy}(\text{pp}, \mathbf{c}, t, \text{op}_1, 2\delta \beta_{\text{agg}}) \end{array} \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ (\mathbf{c}, t, \text{op}_0, \text{op}_1) \leftarrow \mathcal{A}(\text{pp}) \end{array} \right] \leq \text{negl}(\lambda).$$

2.3 Synchronized Multi-Signatures

Definition 2.11 (Synchronized Multi-Signatures). A synchronized multi-signature scheme for 2^τ time steps is a tuple of PPT algorithms $\text{MSIG} = (\text{Setup}, \text{KGen}, \text{Sign}, \text{SigAgg}, \text{iVrfy}, \text{aVrfy})$ defined as:

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: On input the security parameter λ , the algorithm outputs public parameters pp .
- $(\text{sk}, \text{pk}) \leftarrow \text{KGen}(\text{pp})$: On input public parameters pp , the algorithm outputs a secret / public key pair (sk, pk) .
- $\sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, t, \mu)$: On input public parameters pp , a secret key sk , a time step $t \in [2^\tau]$, and a message $\mu \in \{0, 1\}^*$, the algorithm outputs an individual signature σ .
- $\Sigma \leftarrow \text{SigAgg}(\text{pp}, P, t, \mu, S)$: On input public parameters pp , a list of public keys $P = (\text{pk}_1, \dots, \text{pk}_N)$, a time step $t \in [2^\tau]$, a message μ , and individual signatures $S = (\sigma_1, \dots, \sigma_N)$, the algorithm outputs an aggregate signature Σ .

- $b \leftarrow \text{iVrfy}(\text{pp}, \text{pk}, t, \mu, \sigma)$: On input public parameters pp , a public key pk , a time step $t \in [2^\tau]$, a message μ , and an individual signature σ , the algorithm outputs a bit b indicating acceptance or rejection.

- $b \leftarrow \text{aVrfy}(\text{pp}, P, t, \mu, \Sigma)$: On input public parameters pp , a list of public keys P , a time step $t \in [2^\tau]$, a message μ , and an aggregate signature Σ , the algorithm outputs a bit b indicating acceptance or rejection.

A synchronized multi-signature scheme is required to satisfy the following properties.

Definition 2.12 (Individual Correctness). Let $\epsilon_{\text{cor}} \in [0, 1)$. A synchronized multi-signature scheme MSIG is ϵ_{cor} -individually correct if, for every $\lambda \in \mathbb{N}^+$, $t \in [2^\tau]$, and $\mu \in \{0, 1\}^*$, it holds that

$$\Pr \left[\text{iVrfy}(\text{pp}, \text{pk}, t, \mu, \sigma) = 1 \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ (\text{sk}, \text{pk}) \leftarrow \text{KGen}(\text{pp}), \\ \sigma \leftarrow \text{Sign}(\text{pp}, \text{sk}, t, \mu) \end{array} \right] \geq 1 - \epsilon_{\text{cor}}.$$

Definition 2.13 (Aggregation Correctness). A synchronized multi-signature scheme MSIG has correct aggregation if, for every $\lambda \in \mathbb{N}^+$, $t \in [2^\tau]$, $\mu \in \{0, 1\}^*$, $P = (\text{pk}_1, \dots, \text{pk}_N)$ and $S = (\sigma_1, \dots, \sigma_N)$ satisfying $\text{iVrfy}(\text{pp}, \text{pk}_i, t, \mu, \sigma_i) = 1$ for all $i \in [N]$, it holds that

$$\Pr \left[\text{aVrfy}(\text{pp}, P, t, \mu, \Sigma) = 1 \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ \Sigma \leftarrow \text{SigAgg}(\text{pp}, P, t, \mu, S) \end{array} \right] \geq 1 - \text{negl}(\lambda).$$

Definition 2.14 (Unforgeability). A synchronized multi-signature scheme MSIG is unforgeable if, for every $\lambda \in \mathbb{N}^+$ and every PPT adversary $\mathcal{A}$, it holds that

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ (\text{sk}^\dagger, \text{pk}^\dagger) \leftarrow \text{KGen}(\text{pp}), \\ (P^*, t^*, \mu^*, \Sigma^*) \leftarrow \mathcal{A}^{\text{OSign}(\text{pp}, \text{pk}^\dagger)} : \\ \text{aVrfy}(\text{pp}, P^*, t^*, \mu^*, \Sigma^*) = 1 \\ \wedge \text{pk}^\dagger \in P^* \wedge t^* \notin Q_{\text{time}} \end{array} \right] \leq \text{negl}(\lambda),$$

where $\text{OSign}(t, \mu)$ returns $\text{Sign}(\text{pp}, \text{sk}^\dagger, t, \mu)$ and records t in Q_{time} provided that t has not been queried previously; otherwise, it returns $\perp$.

2.4 SNARKs and the LaBRADOR Proof System

Definition 2.15 (Succinct Arguments of Knowledge [11]). Let $\mathcal{R} \subseteq \{0, 1\}^* \times \{0, 1\}^*$ be a polynomial-time decidable relation, where $(x, w) \in \mathcal{R}$ denotes a statement-witness pair. A non-interactive argument system for $\mathcal{R}$ is a tuple of PPT algorithms $\Pi = (\text{Setup}, \text{Prove}, \text{Verify})$ with the following syntax.

- $\text{pp} \leftarrow \text{Setup}(1^\lambda)$: On input the security parameter λ , the algorithm outputs public parameters pp .
- $\pi \leftarrow \text{Prove}(\text{pp}, x, w)$: On input pp , a statement x , and a witness w , the algorithm outputs a proof π .
- $b \leftarrow \text{Verify}(\text{pp}, x, \pi)$: On input pp , a statement x , and a proof π , the algorithm outputs a bit b indicating acceptance or rejection.

Definition 2.16 (Completeness). Let $\epsilon_{\text{comp}} \in [0, 1)$. A non-interactive argument system Π is ϵ_{comp} -complete if, for every $(x, w) \in \mathcal{R}$, it holds that

$$\Pr \left[\text{Verify}(\text{pp}, x, \pi) = 1 \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ \pi \leftarrow \text{Prove}(\text{pp}, x, w) \end{array} \right] \geq 1 - \epsilon_{\text{comp}}.$$

Definition 2.17 (Knowledge Soundness). Let $\epsilon_{\text{ext}} \in [0, 1)$. A non-interactive argument system Π is ϵ_{ext} -knowledge sound if, for every PPT prover $\mathcal{P}^*$, there exists a PPT extractor $\mathcal{E}$ with access to the state of $\mathcal{P}^*$ such that

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{pp}, x, \pi) = 1 \\ \wedge (x, w^*) \notin \mathcal{R} \end{array} \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ (x, \pi) \leftarrow \mathcal{P}^*(\text{pp}), \\ w^* \leftarrow \mathcal{E}^{\mathcal{P}^*}(\text{pp}, x, \pi) \end{array} \right] \leq \epsilon_{\text{ext}}.$$

Definition 2.18 (Succinctness). A non-interactive argument system Π is succinct if there exists a polynomial p such that, for every $(x, w) \in \mathcal{R}$, both its proof length and verification time are bounded by $\text{poly}(\lambda) \cdot \text{poly}(\log |x| + \log |w|)$.

A non-interactive argument system satisfying completeness, knowledge soundness, and succinctness is a succinct non-interactive argument of knowledge (SNARK).

Definition 2.19 (LaBRADOR Proof System [5]). Let $q \in \mathbb{N}^+$. LaBRADOR is a Module-SIS-based succinct argument of knowledge for short solutions to systems of dot-product constraints over R_q . For our application, we restrict its principal relation to full ring equalities and use its non-interactive Fiat-Shamir variant.

For rank n , multiplicity r , and norm bound B , a witness is $\mathbf{w} = (\mathbf{w}_1, \dots, \mathbf{w}_r) \in (R_q^n)^r$, and the associated public statement is $x = (\mathcal{F}, B)$, where $\mathcal{F}$ is a family of public quadratic dot-product functions of the form

$$f(\mathbf{w}) = \sum_{i,j \in [r]} a_{i,j} \langle \mathbf{w}_i, \mathbf{w}_j \rangle + \sum_{i \in [r]} \langle \phi_i, \mathbf{w}_i \rangle - b,$$

with public $a_{i,j}, b \in R_q$ and $\phi_i \in R_q^n$. The corresponding relation $\mathcal{R}_{\text{LB}}$ contains all pairs $(x, \mathbf{w})$ such that $f(\mathbf{w}) = 0$ in R_q for every $f \in \mathcal{F}$ and $\sum_{i=1}^r \|\mathbf{w}_i\|_2^2 \leq (\delta_{\text{LB}} B)^2$, where δ_{LB} is a relaxation factor. Its syntax is defined as follows.

- $\text{pp}_{\text{LB}} \leftarrow \text{LB.Setup}(1^\lambda)$: Outputs the public parameters pp_{LB} .
- $\pi \leftarrow \text{LB.Prove}(\text{pp}_{\text{LB}}, x, \mathbf{w})$: Outputs a proof π for $(x, \mathbf{w}) \in \mathcal{R}_{\text{LB}}$.

- $b_{\text{ver}} \leftarrow \text{LB.Verify}(\text{pp}_{\text{LB}}, x, \pi)$: Outputs a bit b_{ver} indicating acceptance or rejection.

The completeness, knowledge soundness, and succinctness of LaBRADOR are analyzed in detail in [5]. In this work, we use LaBRADOR as a black box and omit its internal details.

3 Homomorphic Vector Commitments with Succinct Opening Proofs

We extend an HVC with a succinct proof of knowledge of a valid opening. The resulting primitive allows a prover to convince a verifier that it knows an opening of a commitment at a specified position, without transmitting the full opening path or the corresponding opened value.

3.1 Definition

Definition 3.1 (Homomorphic Vector Commitment with Succinct Opening Proofs (HVC-SOP)). Let $\text{HVC} = (\text{Setup}, \text{Com}, \text{Open}, \text{iVrfy}, \text{sVrfy}, \text{wVrfy})$ be an HVC as defined in Definition 2.6. A homomorphic vector commitment with succinct opening proofs (HVC-SOP) consists of the HVC together with two PPT algorithms $\text{HVC-SOP} = (\text{Setup}, \text{Com}, \text{Open}, \text{iVrfy}, \text{sVrfy}, \text{wVrfy}, \text{pOpen}, \text{pVrfy})$ defined as follows.

- $\pi \leftarrow \text{pOpen}(\text{pp}, \mathbf{c}, t, \text{op})$: On input public parameters pp , a commitment $\mathbf{c}$, an index $t \in [2^\tau]$, and an opening op , the algorithm outputs an opening proof π .
- $b \leftarrow \text{pVrfy}(\text{pp}, \mathbf{c}, t, \mathbf{m}_t, \pi)$: On input public parameters pp , a commitment $\mathbf{c}$, an index $t \in [2^\tau]$, a target value $\mathbf{m}_t$, and an opening proof π , the algorithm outputs a bit $b \in \{0, 1\}$ indicating if π is an accepting proof of knowledge of a valid opening such that $\mathbf{c}$ opens to value $\mathbf{m}_t$ at position t .

In addition to the properties of the underlying HVC, an HVC-SOP satisfies opening proofs succinctness, completeness, knowledge soundness, and aggregate opening compatibility.

Definition 3.2 (Opening Proofs Succinctness). For every fixed $\lambda \in \mathbb{N}^+$, $t \in [2^\tau]$, $\mathbf{m} \in \mathcal{S}_{\text{dom}}^{2^\tau}$, $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, $\mathbf{c} \in \mathcal{S}_{\text{com}}$, $\text{op} \in \mathcal{S}_{\text{open}}$, and $\pi \leftarrow \text{pOpen}(\text{pp}, \mathbf{c}, t, \text{op})$, define the maximum opening and opening proof lengths by

$$\begin{aligned} \ell_{\text{op}}(\lambda, \tau) &:= \max \{ |\text{op}| : \text{sVrfy}(\text{pp}, \mathbf{c}, t, \text{op}, \beta_{\text{agg}}) = \mathbf{m}_t \}, \\ \ell_{\text{prf}}(\lambda, \tau) &:= \max \{ |\pi| : \text{pVrfy}(\text{pp}, \mathbf{c}, t, \mathbf{m}_t, \pi) = 1 \}, \end{aligned}$$

where the maximum is taken over the randomness of all the algorithms and the input t . An HVC-SOP scheme HVC-SOP has ℓ_{prf} -succinct opening proofs if for every fixed λ , there exists $\tau_0 \in \mathbb{N}^+$ such that $\ell_{\text{prf}}(\lambda, \tau) < \ell_{\text{op}}(\lambda, \tau)$ for all $\tau \geq \tau_0$.

Definition 3.3 (Opening Proofs Completeness). Let $\epsilon_{\text{comp}} \in [0, 1)$. An HVC-SOP scheme HVC-SOP has ϵ_{comp} -complete opening proofs if, for every $\lambda \in \mathbb{N}^+$, $\mathbf{m} \in \mathcal{S}_{\text{dom}}^{2^\tau}$, and $t \in [2^\tau]$, it holds that

$$\Pr \left[\text{pVrfy}(\text{pp}, \mathbf{c}, t, \mathbf{m}_t, \pi) = 1 \mid \begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda), \\ \mathbf{c} \leftarrow \text{Com}(\text{pp}, \mathbf{m}), \\ \text{op} \leftarrow \text{Open}(\text{pp}, \mathbf{c}, \mathbf{m}, t), \\ \pi \leftarrow \text{pOpen}(\text{pp}, \mathbf{c}, t, \text{op}) \end{array} \right] \geq 1 - \epsilon_{\text{comp}}.$$

Definition 3.4 (Opening Knowledge Soundness). Let $\lambda \in \mathbb{N}^+$, $\delta \geq 1$, and $\epsilon_{\text{ext}} \in [0, 1)$. An HVC-SOP scheme HVC-SOP is $(\epsilon_{\text{ext}}, \delta)$ -opening knowledge sound if, for every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}^{\mathcal{A}}$ such that

$$\Pr \left[\begin{array}{l} \text{pVrfy}(\text{pp}, \mathbf{c}, t, \mathbf{m}_t, \pi) = 1 \wedge \\ \text{sVrfy}(\text{pp}, \mathbf{c}, t, \text{op}, \delta \cdot \beta_{\text{agg}}) \neq \mathbf{m}_t \end{array} \right] \leq \epsilon_{\text{ext}}.$$

where $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, $(\mathbf{c}, t, \mathbf{m}_t, \pi) \leftarrow \mathcal{A}(\text{pp})$, and $\text{op} \leftarrow \mathcal{E}^{\mathcal{A}}(\text{pp})$.

Definition 3.5 (Aggregate Opening Compatibility). Let $\epsilon_{\text{agg}} \in [0, 1)$. An HVC-SOP scheme HVC-SOP is ϵ_{agg} -aggregate opening compatible if, for every $N \in \mathbb{N}^+$, $t \in [2^\tau]$, $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, a finite set $W \subseteq R$, and tuple set $\{(\mathbf{c}_i, \mathbf{m}_i, \text{op}_i)\}_{i \in [N]}$ satisfying $\text{iVrfy}(\text{pp}, \mathbf{c}_i, t, \text{op}_i, \eta) = \mathbf{m}_i^i \neq \perp$ for all $i \in [N]$, defining $\mathbf{c}^* := \sum_{i=1}^N w_i \cdot \mathbf{c}_i$, $\text{op}^* := \sum_{i=1}^N w_i \cdot \text{op}_i$, and $\mathbf{m}_t^* := \sum_{i=1}^N w_i \cdot \mathbf{m}_t^i$ and the event $E := \text{sVrfy}(\text{pp}, \mathbf{c}^*, t, \text{op}^*, \beta_{\text{agg}}) = \mathbf{m}_t^*$, it holds that

$$\Pr_{w_i \leftarrow W} \left[\text{pVrfy}(\text{pp}, \mathbf{c}^*, t, \mathbf{m}_t^*, \text{pOpen}(\text{pp}, \mathbf{c}^*, t, \text{op}^*)) = 1 \mid E \right] \geq 1 - \epsilon_{\text{agg}}.$$

4 Generic Construction of Synchronized Multi-Signature Based On HVC-SOP and KOTS

In this section, we build a synchronized multi-signature scheme from HVC-SOP and KOTS and provide a formal security analysis.

4.1 A Generic Construction

We provide a formal description of our generic synchronized multi-signature in Figure 3. Note that the individual signature includes an explicit HVC-SOP opening, while the aggregate signature produces the more compact HVC-SOP opening proof. Thus, the construction preserves the algebraic structure needed for efficient aggregation while obtaining a substantially more compact aggregate signature.

4.2 Security Analysis

Theorem 4.1. Let $\lambda, d, q, N, n, k, \xi, \gamma, \tau, \eta, \beta_{\text{agg}}, \alpha_w \in \mathbb{N}^+$, with d being a power of two, q being prime. Let $\delta \geq 1$. Let

$\epsilon_{\text{cor}}, \epsilon_{\text{homK}}, \epsilon_{\text{homH}}, \epsilon_{\text{ext}}, \epsilon_{\text{agg}} \in [0, 1)$ such that $(\epsilon_{\text{homH}} + \epsilon_{\text{agg}} + \epsilon_{\text{homK}})^\gamma \leq 2^{-\lambda}$. Let $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. Let $W = \mathcal{T}_{\alpha_w} \subseteq R$ be a set such that $|W| > 2^\lambda$ and let $W' := \{w_0 - w_1 \mid w_0, w_1 \in W\}$. Let $H_{\text{agg}} : \{0, 1\}^* \rightarrow W^N$ be a random oracle. Let KOTS be a key-homomorphic one-time signature scheme with public keys in $R_q^{k \times n}$ as defined in Definition 2.1 and let HVC-SOP be a homomorphic vector commitment for domain R_q^ξ as defined in Definition 3.1.

If KOTS is ϵ_{cor} -individually correct, $(N, W, \epsilon_{\text{homK}})$ -probabilistically homomorphic, robustly homomorphic, and W' -multi-user existentially unforgeable under rerandomized keys and HVC-SOP is individually correct, $(N, W, \epsilon_{\text{homH}})$ -probabilistically homomorphic, δ -robustly homomorphic, δ -position binding, is $(\epsilon_{\text{ext}}, \delta)$ -opening knowledge sound, and is ϵ_{agg} -aggregate opening compatible, then the synchronized multi-signature construction in Figure 3 is an unforgeable synchronized multi-signature scheme that is ϵ_{cor} -individually correct and has correct aggregation in the presence of rogue keys and signatures.

Proof. Individual correctness is proved in Lemma 4.1. Aggregation correctness is proved in Lemma 4.2. Unforgeability is proved in Lemma 4.3. $\square$

Lemma 4.1 (Individual correctness). Let $\epsilon_{\text{cor}} \in [0, 1)$. If HVC-SOP is individually correct and KOTS is ϵ_{cor} -individually correct, then the synchronized multi-signature construction MSIG from Figure 3 is ϵ_{cor} -individually correct.

Proof. Let $\text{pp} = (\text{pp}_{\text{kots}}, \text{pp}_{\text{hvc sop}}) \leftarrow \text{Setup}(1^\lambda)$, $(\text{sk}, \text{pk}) = ((\text{OSK}, \text{OPK}), \mathbf{c}) \leftarrow \text{KGen}(\text{pp})$, $t \in [2^\tau]$, $\mu \in \{0, 1\}^*$, and $\sigma = (\text{op}, \sigma_{\text{kots}}) = \text{Sign}(\text{pp}, \text{sk}, t, \mu)$, where

$$\sigma_{\text{kots}} \leftarrow \text{KOTS.Sign}(\text{pp}_{\text{kots}}, \text{osk}_t, \mu), \quad (1)$$

$$\text{op} \leftarrow \text{HVC-SOP.Open}(\text{pp}_{\text{hvc sop}}, \mathbf{c}, \text{OPK}, t). \quad (2)$$

Now, consider the execution of $\text{iVrfy}(\text{pp}, \text{pk}, t, \mu, \sigma)$. The algorithm proceeds as follows:

- **HVC-SOP Verification.** $\text{HVC-SOP.iVrfy}(\text{pp}_{\text{hvc sop}}, \mathbf{c}, t, \text{op}, \eta)$ is called first. By the individual correctness of HVC-SOP, for any honestly generated commitment $\mathbf{c}$ to the vector $\text{OPK} = (\text{opk}_1, \dots, \text{opk}_{2^\tau})$ and opening op for index t , the individual verification returns the original committed value at position t : opk_t . Since $t \in [2^\tau]$ and $\text{opk}_t \neq \perp$, the algorithm does not return 0.
- **KOTS Verification.** The algorithm finally returns the result of $\text{KOTS.iVrfy}(\text{pp}_{\text{kots}}, \text{opk}_t, \mu, \sigma_{\text{kots}})$. By the ϵ_{cor} -individual correctness of KOTS, since σ_{kots} was generated using KOTS.Sign with the secret key osk_t corresponding to the public key opk_t , individual verification algorithm returns 1 with probability $\geq 1 - \epsilon_{\text{cor}}$.

Thus, with probability $\geq 1 - \epsilon_{\text{cor}}$, MSIG.iVrfy returns 1. $\square$

Setup(1^λ) 1. $\text{pp}_{\text{KOTS}} \leftarrow \text{KOTS.Setup}(1^\lambda)$ 2. $\text{pp}_{\text{HVC-SOP}} \leftarrow \text{HVC-SOP.Setup}(1^\lambda)$ 3. return $\text{pp} := (\text{pp}_{\text{KOTS}}, \text{pp}_{\text{HVC-SOP}})$ KGen(pp) 1. for $t \in [2^\tau]$ do $(\text{osk}_t, \text{opk}_t) \leftarrow \text{KOTS.KGen}(\text{pp}_{\text{KOTS}})$ 2. $\text{OSK} := (\text{osk}_1, \dots, \text{osk}_{2^\tau})$ and $\text{OPK} := (\text{opk}_1, \dots, \text{opk}_{2^\tau})$ 3. $\mathbf{c} \leftarrow \text{HVC-SOP.Com}(\text{pp}_{\text{HVC-SOP}}, \text{OPK})$ 4. return $(\text{sk}, \text{pk}) := ((\text{OSK}, \text{OPK}), \mathbf{c})$ Sign($\text{pp}, \text{sk}, t, \mu$) 1. Parse $\text{sk} = (\text{OSK}, \text{OPK})$ 2. if $t \notin [2^\tau]$ then return $\perp$ 3. $\sigma_{\text{kots}} \leftarrow \text{KOTS.Sign}(\text{pp}_{\text{KOTS}}, \text{osk}_t, \mu)$ 4. $\text{op} \leftarrow \text{HVC-SOP.Open}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}, \text{OPK}, t)$ 5. return $\sigma := (\text{op}, \sigma_{\text{kots}})$ iVrfy($\text{pp}, \text{pk}, t, \mu, \sigma$) 1. Parse $\text{pk} = \mathbf{c}$ and $\sigma = (\text{op}, \sigma_{\text{kots}})$ 2. if $t \notin [2^\tau]$ then return 0 3. $\text{opk}_t = \text{HVC-SOP.iVrfy}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}, t, \text{op}, \eta)$ 4. if $\text{opk}_t = \perp$ then return 0 5. return $\text{KOTS.iVrfy}(\text{pp}_{\text{KOTS}}, \text{opk}_t, \mu, \sigma_{\text{kots}})$	SigAgg(pp, P, t, μ, S) 1. Parse $P = (\mathbf{c}_1, \dots, \mathbf{c}_N)$ and $S = (\sigma_1, \dots, \sigma_N)$ with $\sigma_i = (\text{op}_i, \sigma_{\text{kots}}^i)$ 2. if $P \neq N$ or $S \neq N$ then return $\perp$ 3. for $i \in [N]$ do a. if $\text{iVrfy}(\text{pp}, \mathbf{c}_i, t, \mu, \sigma_i) = 0$ then return $\perp$ b. else internally record opk_t^i. 4. for $\text{cnt} = 1, \dots, \gamma$ do a. $(w_1, \dots, w_N) := \text{Hagg}(P, t, \mu, \text{cnt})$ b. $\mathbf{c}^* := \sum_{i=1}^N w_i \cdot \mathbf{c}_i$, $\text{op}^* := \sum_{i=1}^N w_i \cdot \text{op}_i$, $\text{opk}_t^* := \sum_{i=1}^N w_i \cdot \text{opk}_t^i$, $\sigma_{\text{kots}}^* := \sum_{i=1}^N w_i \cdot \sigma_{\text{kots}}^i$ c. if $\text{HVC-SOP.sVrfy}(\text{pp}, \mathbf{c}^*, t, \text{op}^*, \beta_{\text{agg}}) \neq \text{opk}_t^*$ continue d. $\pi \leftarrow \text{HVC-SOP.pOpen}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}^*, t, \text{op}^*)$ e. $\Sigma := (\pi, \text{opk}_t^*, \sigma_{\text{kots}}^*, \text{cnt})$ f. if $\text{aVrfy}(\text{pp}, P, t, \mu, \Sigma) = 0$ continue g. else return Σ 5. return $\perp$ aVrfy($\text{pp}, P, t, \mu, \Sigma$) 1. Parse $\Sigma = (\pi, \text{opk}_t^*, \sigma_{\text{kots}}^*, \text{cnt})$ and $P = (\mathbf{c}_1, \dots, \mathbf{c}_N)$ 2. if $t \notin [2^\tau]$ or $\text{cnt} \notin [\gamma]$ then return 0 3. $(w_1, \dots, w_N) := \text{Hagg}(P, t, \mu, \text{cnt})$ 4. $\mathbf{c}^* := \sum_{i=1}^N w_i \cdot \mathbf{c}_i$ 5. if $\text{HVC-SOP.pVrfy}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}^*, t, \text{opk}_t^*, \pi) = 0$ then return 0 6. return $\text{KOTS.sVrfy}(\text{pp}_{\text{KOTS}}, \text{opk}_t^*, \mu, \sigma_{\text{kots}}^*)$
---	--

Figure 3: A generic synchronized multi-signature construction MSIG based on HVC-SOP and KOTS.

Lemma 4.2 (Aggregation correctness). *Let $\alpha_w, N \in \mathbb{N}^+$. Let $\epsilon_{\text{homK}}, \epsilon_{\text{homH}}, \epsilon_{\text{agg}} \in [0, 1)$ such that $(\epsilon_{\text{agg}} + \epsilon_{\text{homH}} + \epsilon_{\text{homK}})^\gamma \leq 2^{-\lambda}$. Let $W = \mathcal{T}_{\alpha_w} \subseteq R$. If HVC-SOP is $(N, W, \epsilon_{\text{homH}})$ -probabilistically homomorphic and has ϵ_{agg} -aggregate opening compatibility, and KOTS is $(N, W, \epsilon_{\text{homK}})$ -probabilistically homomorphic, then the synchronized multi-signature construction MSIG from Figure 3 has correct aggregation in the presence of rogue keys and signatures.*

Proof. Let $\text{pp} = (\text{pp}_{\text{KOTS}}, \text{pp}_{\text{HVC-SOP}})$, and let (pk_i, σ_i) be pairs such that $\text{iVrfy}(\text{pp}, \text{pk}_i, t, \mu, \sigma_i) = 1$. By the definition of iVrfy in Figure 3, for each $i \in [N]$ there exists a one-time public key opk_t^i such that

$$\text{opk}_t^i = \text{HVC-SOP.iVrfy}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}_i, t, \text{op}_i, \eta), \quad (3)$$

$$\text{KOTS.iVrfy}(\text{pp}_{\text{KOTS}}, \text{opk}_t^i, \mu, \sigma_{\text{kots}}^i) = 1, \quad (4)$$

where $\text{pk}_i = \mathbf{c}_i$ and $\sigma_i = (\text{op}_i, \sigma_{\text{kots}}^i)$. The aggregation algorithm **SigAgg** performs at most γ attempts to find a counter cnt such that the aggregate signature $\Sigma = (\pi, \text{opk}_t^*, \sigma_{\text{kots}}^*, \text{cnt})$ is accepted by **aVrfy**. In iteration cnt , the algorithm samples weights $(w_1, \dots, w_N) \leftarrow \text{Hagg}(P, t, \mu, \text{cnt})$ and computes $\mathbf{c}^* = \sum_{i=1}^N w_i \cdot \mathbf{c}_i$, $\text{op}^* = \sum_{i=1}^N w_i \cdot \text{op}_i$, $\text{opk}_t^* = \sum_{i=1}^N w_i \cdot \text{opk}_t^i$, $\sigma_{\text{kots}}^* = \sum_{i=1}^N w_i \cdot \sigma_{\text{kots}}^i$. If $\text{HVC-SOP.sVrfy}(\text{pp}, \mathbf{c}^*, t, \text{op}^*, \beta_{\text{agg}}) = \text{opk}_t^*$, it computes $\pi \leftarrow \text{HVC-SOP.pOpen}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}^*, t, \text{op}^*)$. According to **aVrfy**, the signature is accepted if:

- $\text{HVC-SOP.pVrfy}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}^*, t, \text{opk}_t^*, \pi) = 1$. By the $(N, W, \epsilon_{\text{homH}})$ -probabilistic homomorphism of HVC-SOP,

since op_i are valid openings for $\mathbf{c}_i$ at position t with values opk_t^i , then $\text{op}^* = \sum w_i \cdot \text{op}_i$ is a valid opening for $\mathbf{c}^* = \sum w_i \cdot \mathbf{c}_i$ at position t with value $\text{opk}_t^* = \sum w_i \cdot \text{opk}_t^i$, except with probability ϵ_{homH} . Conditioned on $\text{HVC-SOP.sVrfy}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}^*, t, \text{op}^*, \beta_{\text{agg}}) = \text{opk}_t^*$, since $\pi \leftarrow \text{HVC-SOP.pOpen}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}^*, t, \text{op}^*)$, by the aggregate opening compatibility of HVC-SOP, it holds that $\text{HVC-SOP.pVrfy}(\text{pp}_{\text{HVC-SOP}}, \mathbf{c}^*, t, \text{opk}_t^*, \pi) = 1$ except with probability ϵ_{agg} .

- $\text{KOTS.sVrfy}(\text{pp}_{\text{KOTS}}, \text{opk}_t^*, \mu, \sigma_{\text{kots}}^*) = 1$. From above, we learn that opk_t^* is a valid aggregated one-time public key of $\mathbf{c}^*$ at position t . By the $(N, W, \epsilon_{\text{homK}})$ -probabilistic homomorphism of KOTS, the linear combination of signatures $\sigma_{\text{kots}}^* = \sum w_i \cdot \sigma_{\text{kots}}^i$ is a valid signature for message μ under the aggregated key opk_t^* , except with probability ϵ_{homK} .

By a union bound, the probability that a single iteration cnt fails to satisfy both conditions is at most $\epsilon_{\text{agg}} + \epsilon_{\text{homH}} + \epsilon_{\text{homK}}$. Since the algorithm makes γ independent attempts with fresh counter inputs cnt to the random oracle, the total failure probability across all γ attempts fail is at most $(\epsilon_{\text{agg}} + \epsilon_{\text{homH}} + \epsilon_{\text{homK}})^\gamma$. Since $(\epsilon_{\text{agg}} + \epsilon_{\text{homH}} + \epsilon_{\text{homK}})^\gamma \leq 2^{-\lambda}$, the failure probability is negligible in the security parameter. Thus, the aggregation algorithm succeeds with probability $1 - \text{negl}(\lambda)$. $\square$

Lemma 4.3. *Let $\epsilon_{\text{ext}} \in [0, 1)$ and $\delta \geq 1$. If KOTS is W' -multi-user existentially unforgeable under rerandomized keys and*

HVC-SOP is δ -position binding, $(\epsilon_{\text{ext}}, \delta)$ -opening knowledge sound, and δ -robustly homomorphic, then the synchronized multi-signature construction MSIG from Figure 3 is existentially unforgeable. Specifically, for any PPT adversary $\mathcal{A}$ making at most q_H queries to H_{agg} , there exist PPT adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\text{Adv}_{\text{MSIG}, \mathcal{A}}^{\text{euf}}(\lambda) \leq \epsilon_{\text{ext}} + \frac{q_H}{|W|} + \sqrt{q_H \cdot \left(\text{Adv}_{\text{HVC-SOP}, \mathcal{B}_1}^{\text{bind}}(\lambda) + \text{Adv}_{\text{KOTS}, \mathcal{B}_2}^{\text{EUF-RK}}(\lambda) \right)}.$$

Proof. The proof proceeds via a sequence of games. Let E_i denote the event that the adversary $\mathcal{A}$ succeeds in Game i .

Game 0. This is the standard unforgeability game (Definition 2.14) played against a PPT adversary $\mathcal{A}$.

1. The challenger generates public parameters $\text{pp} = (\text{pp}_{\text{hvc sop}}, \text{pp}_{\text{kots}}) \leftarrow \text{Setup}(1^\lambda)$ and a target key pair $(\text{sk}^\dagger, \text{pk}^\dagger) \leftarrow \text{KGen}(\text{pp})$, where $\text{pk}^\dagger = \mathbf{c}^\dagger$ commits to $\text{OPK}^\dagger = (\text{opk}_1^\dagger, \dots, \text{opk}_{2^\tau}^\dagger)$.
2. $\mathcal{A}$ is given $(\text{pp}, \text{pk}^\dagger)$ and access to a signing oracle $\mathcal{O}_{\text{Sign}}(t, \mu)$ which returns $\text{Sign}(\text{pp}, \text{sk}^\dagger, t, \mu)$ and random oracle H_{agg} .
3. $\mathcal{A}$ outputs a forgery $(P^*, t^*, \mu^*, \text{cnt}^*, \Sigma^*)$, where $\Sigma^* = (\pi^*, \text{opk}_t^*, \sigma_{\text{kots}}^*, \text{cnt}^*)$, and $P^* = (\mathbf{c}_1, \dots, \mathbf{c}_N)$.

$\mathcal{A}$ wins if: (1) $\text{pk}^\dagger \in P^*$ with $|P^*| = N$; (2) $\mathcal{A}$ made no signing query for time step t^* ; and (3) $\text{aVrfy}(\text{pp}, P^*, t^*, \mu^*, \Sigma^*) = 1$. By definition, $\Pr[E_0] = \text{Adv}_{\text{MSIG}, \mathcal{A}}^{\text{euf}}(\lambda)$.

Game 1. We modify the simulation of the random oracle H_{agg} . Without loss of generality, assume $\mathcal{A}$ queries $H_{\text{agg}}(P^*, t^*, \mu^*, \text{cnt}^*)$ prior to outputting its forgery. For each hash query $i \in [q_H]$, $\mathcal{B}$ lazily samples $\mathbf{w}_i = (w_{i,1}, \dots, w_{i,N}) \leftarrow W^N$. If query i contains $\text{pk}^\dagger$ at index ℓ , $\mathcal{B}$ reorders the elements of $\mathbf{w}_i$ such that the weight assigned to $\text{pk}^\dagger$ is placed at position N . Because weights are sampled independently and uniformly from W , this reordering does not alter the response distribution. Thus, $\Pr[E_1] = \Pr[E_0]$.

Game 2. In this game, we modify adversary the extractor $\mathcal{E}^{\mathcal{A}}$ from Definition 3.4 is invoked upon $\mathcal{A}$'s forgery. When $\mathcal{A}$ outputs a winning forgery $(P^*, t^*, \mu^*, \text{cnt}^*, \Sigma^* = (\pi^*, \text{opk}_t^*, \sigma_{\text{kots}}^*, \text{cnt}^*))$, aVrfy guarantees that

$$\begin{aligned} \text{HVC-SOP.pVrfy}(\text{pp}_{\text{hvc sop}}, \mathbf{c}^*, t^*, \text{opk}_t^*, \pi^*) &= 1, \\ \text{KOTS.sVrfy}(\text{pp}_{\text{kots}}, \text{opk}_t^*, \mu^*, \sigma_{\text{kots}}^*) &= 1, \end{aligned}$$

where $(w_1, \dots, w_N) \leftarrow H_{\text{agg}}(P^*, t^*, \mu^*, \text{cnt}^*)$ and $\mathbf{c}^* = \sum_{i=1}^N w_i \cdot \mathbf{c}_i$. We run $\text{opk}^* \leftarrow \mathcal{E}^{\mathcal{A}}(\text{pp}_{\text{hvc sop}})$, where the extractor

uses $(\mathbf{c}^*, t^*, \text{opk}_t^*, \pi^*)$ output by $\mathcal{A}$. By opening knowledge soundness of HVC-SOP, except with probability at most ϵ_{ext} ,

$$\text{HVC-SOP.sVrfy}(\text{pp}_{\text{hvc sop}}, \mathbf{c}^*, t^*, \text{op}^*, \delta \cdot \beta_{\text{agg}}) = \text{opk}_t^*.$$

Therefore, $|\Pr[E_2] - \Pr[E_1]| \leq \epsilon_{\text{ext}}$.

General Forking. We construct a randomized forking algorithm $\mathcal{F}_{\mathcal{B}}$ that runs $\mathcal{B}$ (incorporating $\mathcal{E}^{\mathcal{A}}$) twice over forked random choices:

1. Sample random coins r for $\mathcal{B}$ and random vectors $\mathbf{w}_1, \dots, \mathbf{w}_{q_H} \leftarrow W^N$, where $\mathbf{w}_i = (w_{i,1}, \dots, w_{i,N})$.
2. Execute $\mathcal{B}(\text{pp}, \text{pk}^\dagger; r)$ with $\mathbf{w}_1, \dots, \mathbf{w}_{q_H}$.
3. If $\mathcal{B}$ outputs a valid forgery, return $(1, i^*, (P^*, t^*, \mu^*, \text{cnt}^*, \Sigma^{(0)} = (\pi^{(0)}, \text{opk}^{(0)}, \sigma_{\text{kots}}^{(0)}, \text{cnt}^*)))$, where $i^* \in [q_H]$ is the query index corresponding to the forged tuple $(P^*, t^*, \mu^*, \text{cnt}^*)$. Otherwise, abort and output $(0, 0, \perp)$.
4. Re-run $\mathcal{B}(\text{pp}, \text{pk}^\dagger; r)$ with identical randomness r and identical oracle responses $\mathbf{w}_1, \dots, \mathbf{w}_{i^*-1}$, but with fresh, independently sampled responses $\mathbf{w}'_{i^*+1}, \dots, \mathbf{w}'_{q_H} \leftarrow W^N$. For the i^* -th response, the first $N-1$ -th components are the same as $\mathbf{w}_{i^*}$, but the last component is freshly sampled. That is, $\mathbf{w}'_{i^*} = (w_{i^*,1}, \dots, w_{i^*,N-1}, w'_{i^*,N})$.
5. If the second run yields a valid forgery for the same tuple $(P^*, t^*, \mu^*, \text{cnt}^*)$ at query index i^* , return $(1, i^*, (P^*, t^*, \mu^*, \text{cnt}^*, \Sigma^{(1)} = (\pi^{(1)}, \text{opk}^{(1)}, \sigma_{\text{kots}}^{(1)}, \text{cnt}^*)))$. Otherwise, it returns $(0, \perp, \perp)$.

Applying Lemma 2.2, we obtain:

$$\Pr[E_2] \leq \frac{q_H}{|W|} + \sqrt{q_H \cdot \Pr[\mathcal{F}_{\mathcal{B}} \text{ succeeds}]}$$

When $\mathcal{F}_{\mathcal{B}}$ succeeds, both executions share identical public key lists P^* , target position t^* , message μ^* . Recall that $\mathbf{w}_{i^*}$ and $\mathbf{w}'_{i^*}$ only differ at index N . Define $\Delta \mathbf{w} := \mathbf{w}_{i^*,N} - \mathbf{w}'_{i^*,N} \in W'$. By the extraction guarantees of Game 2:

$$\begin{aligned} \text{HVC-SOP.sVrfy}(\text{pp}_{\text{hvc sop}}, \mathbf{c}^{(0)}, t^*, \text{op}^{(0)}, \delta \cdot \beta_{\text{agg}}) &= \text{opk}^{(0)}, \\ \text{HVC-SOP.sVrfy}(\text{pp}_{\text{hvc sop}}, \mathbf{c}^{(1)}, t^*, \text{op}^{(1)}, \delta \cdot \beta_{\text{agg}}) &= \text{opk}^{(1)}. \end{aligned}$$

where $\mathbf{c}^{(0)} = \sum_{j=1}^{N-1} w_{i^*,j} \mathbf{c}_j + w_{i^*,N} \mathbf{c}^\dagger$, and $\mathbf{c}^{(1)} = \sum_{j=1}^{N-1} w_{i^*,j} \mathbf{c}_j + w'_{i^*,N} \mathbf{c}^\dagger$. By the robust homomorphism, define opk to be

$$\begin{aligned} \text{HVC-SOP.wVrfy}(\text{pp}_{\text{hvc sop}}, \Delta \mathbf{w} \cdot \mathbf{c}^\dagger, t^*, \text{op}^{(0)} - \text{op}^{(1)}, 2\delta \beta_{\text{agg}}) \\ = \text{opk}^{(0)} - \text{opk}^{(1)} \neq \perp. \end{aligned}$$

Similarly, by the robust homomorphism of KOTS, the signature difference $\Delta \sigma_{\text{kots}} := \sigma_{\text{kots}}^{(0)} - \sigma_{\text{kots}}^{(1)}$ satisfies:

$$\text{KOTS.wVrfy}(\text{pp}_{\text{kots}}, \overline{\text{opk}}, \mu^*, \Delta \sigma_{\text{kots}}) = 1.$$

We partition the success event of $\mathcal{F}_{\mathcal{B}}$ into two mutually exclusive cases:

$\eta \sqrt{2\alpha_w N \ln \left(\frac{2d\kappa(2\tau\omega + \xi)}{\epsilon_{\text{homH}}} \right)}$. If $\text{MSIS}_{q,\omega,2\omega\kappa,4\delta_{\text{LB}}\beta_{\text{agg}}\sqrt{2\omega\kappa d}}^{\ell_2}$ and $\text{MSIS}_{q,\omega,\xi\kappa,4\delta_{\text{LB}}\beta_{\text{agg}}\sqrt{\xi\kappa d}}^{\ell_2}$ problems are hard, then the HVC-SOP construction HVC-SOP in Figure 4 is an individually correct, $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\text{homH}})$ -probabilistically homomorphic, δ_{LB} -robustly homomorphic, δ_{LB} -position binding, ℓ_{prf} -succinct opening proofs, ϵ_{comp} -complete, $(\epsilon_{\text{ext}}, \delta_{\text{LB}})$ -opening knowledge sound, ϵ_{agg} -aggregate opening compatible for domain R_q^ξ and vector length 2^τ .

Proof. The proof of the theorem follows from Lemma 5.1, Lemma 5.2, Lemma 5.3, Lemma 5.4, Lemma 5.5, Lemma 5.6, Lemma 5.7, and Lemma 5.8 below. $\square$

Lemma 5.1 (Individual Correctness). *Let $d, q, \xi, \omega, \eta, \tau \in \mathbb{N}^+$ such that d is a power of 2 and q is a prime. Then the HVC-SOP construction HVC-SOP in Figure 4 is an individually correct for domain R_q^ξ and vector length 2^τ as per Definition 2.7.*

Proof. Fix public parameters $\text{pp} = (\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1, \mathbf{G}_0, \mathbf{G}_1) \leftarrow \text{Setup}(1^\lambda)$, a message vector $\mathbf{m} = (\mathbf{m}_1, \dots, \mathbf{m}_{2^\tau}) \in (R_q^\xi)^{2^\tau}$, an index $t \in [2^\tau]$, and let $\mathbf{c} \leftarrow \text{Com}(\text{pp}, \mathbf{m})$ and $\text{op} \leftarrow \text{Open}(\text{pp}, \mathbf{c}, \mathbf{m}, t)$. Write $\tilde{t} = \text{bin}_\tau(t - 1)$ and parse $\text{op} = ((\mathbf{p}_j, \mathbf{s}_j)_{j=1}^\tau, \mathbf{u})$. We first observe that for all $j \in [\tau]$ it holds in R_q^ω that

$$\mathbf{G}_0 \cdot \mathbf{p}_{j-1} \quad (10)$$

$$= \mathbf{G}_0 \cdot (\text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t}_{<j})) \quad (11)$$

$$= \mathbf{G}_0 \cdot \left(\mathbf{G}_0^{-1} \cdot \left(\mathbf{A}_0 \cdot \text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t}_{<j} \| 0) + \mathbf{A}_1 \cdot \text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t}_{<j} \| 1) \right) \right) \quad (12)$$

$$= \mathbf{A}_0 \cdot \text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t}_{<j} \| 0) + \mathbf{A}_1 \cdot \text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t}_{<j} \| 1) \quad (13)$$

$$= \mathbf{A}_{\tilde{t}_j} \cdot \text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t}_{<j} \| \tilde{t}_j) + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t}_{<j} \| (\tilde{t}_j \oplus 1)) \quad (14)$$

$$= \mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j \pmod{q}. \quad (15)$$

The first equality comes from the definition of Com and Open . The second equality comes from label , $|\tilde{t}_{<j}| < \tau$. The final equality comes from the definition of Open . Observe that for $j = 1$, this gives $\mathbf{c} = \mathbf{A}_{\tilde{t}_1} \cdot \mathbf{p}_1 + \mathbf{A}_{\tilde{t}_1 \oplus 1} \cdot \mathbf{s}_1 \pmod{q}$. Further it holds that

$$\mathbf{G}_0 \cdot \mathbf{p}_\tau = \mathbf{G}_0 \cdot (\text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}(\mathbf{m}, \tilde{t})) \quad (16)$$

$$= \mathbf{G}_0 \cdot \left(\mathbf{G}_0^{-1} \cdot (\mathbf{B} \cdot \mathbf{G}_1^{-1} \cdot (\mathbf{m}_t)) \right) \quad (17)$$

$$= \mathbf{B} \cdot \mathbf{G}_1^{-1} \cdot (\mathbf{m}_t) \quad (18)$$

$$= \mathbf{B} \cdot \mathbf{u} \pmod{q}. \quad (19)$$

It remains only to check the norm bounds. For $j \in [\tau]$, $\mathbf{p}_j$ and $\mathbf{s}_j$ are outputs of the $\text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}$ function and thus, by definition of $\text{label}_{\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1}$, balanced $(2\eta + 1)$ -ary decompositions

(which is denoted as the map $\mathbf{G}_0^{-1}$) of ω many R_q elements. Similarly, $\mathbf{u}$ is the decomposition of ξ R_q elements. By design, this implies that the coefficients of every honestly generated $\mathbf{p}_j, \mathbf{s}_j$, and $\mathbf{u}$ lie in $[-\eta, \eta]$. Hence:

$$\|\mathbf{p}_j\|_\infty, \|\mathbf{s}_j\|_\infty, \|\mathbf{u}\|_\infty \leq \eta \quad \text{for all } j \in [\tau]. \quad (20)$$

Thus, all checks in iVrfy pass, and the verifier outputs $\mathbf{G}_1 \cdot \mathbf{u} = \mathbf{m}_t \pmod{q}$. Therefore, $\text{iVrfy}(\text{pp}, \mathbf{c}, t, \text{op}, \eta) = \mathbf{m}_t$, as required. $\square$

Lemma 5.2 (Probabilistic Homomorphism). *Let $d, q, \alpha_w, N, \eta, \tau, \xi, \omega, \beta_{\text{agg}} \in \mathbb{N}^+$ and $\epsilon_{\text{homH}} \in (0, 1)$ such that d is a power of 2 and q is a prime. Set $\kappa = \lceil \log_{2\eta+1}(q) \rceil$. If $\beta_{\text{agg}} \geq \eta \sqrt{2\alpha_w N \ln \left(\frac{2d\kappa(2\tau\omega + \xi)}{\epsilon_{\text{homH}}} \right)}$, then the HVC-SOP construction HVC-SOP in Figure 4 is $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\text{homH}})$ -probabilistically homomorphic for domain R_q^ξ and vector length 2^τ as per Definition 2.8.*

Proof. Let $\text{pp} \leftarrow \text{Setup}(1^\lambda)$ and $t \in [2^\tau]$ with $\tilde{t} = \text{bin}_\tau(t - 1)$. Let $\mathbf{c}_i \in R_q^\omega$, $\text{op}_i = (\mathbf{p}_1^i, \dots, \mathbf{p}_\tau^i, \mathbf{s}_1^i, \dots, \mathbf{s}_\tau^i, \mathbf{u}^i) \in (R^{\omega\kappa})^{2^\tau} \times R^{\xi\kappa}$ for $i \in [N]$ such that $\text{iVrfy}(\text{pp}, \mathbf{c}_i, t, \text{op}_i, \eta) = \mathbf{m}_t^i \neq \perp$. Let $w_1, \dots, w_N \leftarrow \mathcal{T}_{\alpha_w}$ be arbitrary weights. For $2 \leq j \leq \tau$, it holds that

$$\mathbf{G}_0 \cdot \left(\sum_{i=1}^N w_i \cdot \mathbf{p}_{j-1}^i \right) \quad (21)$$

$$= \sum_{i=1}^N w_i \cdot \mathbf{G}_0 \cdot \mathbf{p}_{j-1}^i \quad (22)$$

$$= \sum_{i=1}^N w_i \cdot (\mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j^i + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j^i) \quad (23)$$

$$= \mathbf{A}_{\tilde{t}_j} \left(\sum_{i=1}^N w_i \cdot \mathbf{p}_j^i \right) + \mathbf{A}_{\tilde{t}_j \oplus 1} \left(\sum_{i=1}^N w_i \cdot \mathbf{s}_j^i \right) \pmod{q}. \quad (24)$$

Similarly,

$$\mathbf{G}_0 \cdot \left(\sum_{i=1}^N w_i \cdot \mathbf{p}_\tau^i \right) = \sum_{i=1}^N w_i \cdot \mathbf{G}_0 \cdot \mathbf{p}_\tau^i \quad (25)$$

$$= \sum_{i=1}^N w_i \cdot (\mathbf{B} \cdot \mathbf{u}^i) \quad (26)$$

$$= \mathbf{B} \cdot \sum_{i=1}^N w_i \cdot \mathbf{u}^i \pmod{q}, \quad (27)$$

and

$$\sum_{i=1}^N w_i \cdot \mathbf{c}_i = \sum_{i=1}^N w_i \cdot \mathbf{G}_0 \cdot \mathbf{p}_0^i \quad (28)$$

$$= \sum_{i=1}^N w_i \cdot (\mathbf{A}_{\tilde{t}_1} \cdot \mathbf{p}_1^i + \mathbf{A}_{\tilde{t}_1 \oplus 1} \cdot \mathbf{s}_1^i) \quad (29)$$

$$= \mathbf{A}_{\tilde{t}_1} \cdot \sum_{i=1}^N w_i \cdot \mathbf{p}_1^i + \mathbf{A}_{\tilde{t}_1 \oplus 1} \cdot \sum_{i=1}^N w_i \cdot \mathbf{s}_1^i \pmod{q}. \quad (30)$$

Next, we bound the failure probability of the norm checks. Writing out the conditions,

$$P := \Pr \left[w_1, \dots, w_N \leftarrow \mathcal{T}_{\alpha_w} : \exists j \in [\tau] \text{ s.t.} \right. \quad (31)$$

$$\left. \left\| \sum_{i=1}^N w_i \cdot \mathbf{p}_j^i \right\|_{\infty} > \beta_{\text{agg}} \right. \quad (32)$$

$$\vee \left\| \sum_{i=1}^N w_i \cdot \mathbf{s}_j^i \right\|_{\infty} > \beta_{\text{agg}} \quad (33)$$

$$\left. \vee \left\| \sum_{i=1}^N w_i \cdot \mathbf{u}^i \right\|_{\infty} > \beta_{\text{agg}} \right]. \quad (34)$$

Observe that this is an ℓ_{∞} bound for a total of

$$N_{\text{bounds}} = 2\tau\omega\kappa + \xi\kappa$$

many ring elements. For each of these N_{bounds} ring elements, we apply Lemma 2.1 with $\beta = \eta$, and $B = \beta_{\text{agg}}$. Taking an N_{bounds} -fold union bound gives

$$P \leq N_{\text{bounds}} \cdot 2d \cdot \exp\left(-\frac{\beta_{\text{agg}}^2}{2\eta^2\alpha_w N}\right).$$

By assumption, we have

$$\frac{\beta_{\text{agg}}^2}{2\eta^2\alpha_w N} \geq \ln\left(\frac{2d \cdot N_{\text{bounds}}}{\epsilon_{\text{homH}}}\right),$$

and hence $P \leq \epsilon_{\text{homH}}$. It follows that with probability at least $1 - \epsilon_{\text{homH}}$, the strong verification algorithm outputs

$$\mathbf{G}_1 \cdot \left(\sum_{i=1}^N w_i \cdot \mathbf{u}^i \right) = \sum_{i=1}^N w_i \cdot \mathbf{G}_1 \cdot \mathbf{u}^i \quad (35)$$

$$= \sum_{i=1}^N w_i \cdot \text{iVrfy}(\text{pp}, \mathbf{c}_i, t, \text{op}_i, \eta) \quad (36)$$

$$= \sum_{i=1}^N w_i \cdot \mathbf{m}_t^i, \quad (37)$$

as required. $\square$

Lemma 5.3 (Robust Homomorphism). *Let $d, q, \alpha_w, N, \eta, \tau, \xi, \omega, \beta_{\text{agg}} \in \mathbb{N}^+$ such that d is a power of 2 and q is a prime. Then the HVC-SOP construction HVC-SOP in Figure 4 is robustly homomorphic as per Definition 2.9.*

Proof. Let $t \in [2^\tau]$ with $\tilde{t} := \text{bin}_\tau(t-1)$. Let $\mathbf{c}_0, \mathbf{c}_1 \in R_q^\omega$ and $\text{op}_0, \text{op}_1 \in \mathcal{S}_{\text{open}}$ be arbitrary such that

$$\text{sVrfy}(\text{pp}, \mathbf{c}_0, t, \text{op}_0, \beta_{\text{agg}}) = \mathbf{m}_t^0, \quad (38)$$

$$\text{sVrfy}(\text{pp}, \mathbf{c}_1, t, \text{op}_1, \beta_{\text{agg}}) = \mathbf{m}_t^1, \quad (39)$$

with $\mathbf{m}_t^0, \mathbf{m}_t^1 \neq \perp$. We need to show that

$$\text{wVrfy}(\text{pp}, \mathbf{c}_0 - \mathbf{c}_1, t, \text{op}_0 - \text{op}_1, 2\beta_{\text{agg}}) = \mathbf{m}_t^0 - \mathbf{m}_t^1. \quad (40)$$

For $i \in \{0, 1\}$, parse $\text{op}_i = (\mathbf{p}_1^i, \dots, \mathbf{p}_\tau^i, \mathbf{s}_1^i, \dots, \mathbf{s}_\tau^i, \mathbf{u}^i)$. Let $\mathbf{c}_i = \mathbf{G}_0 \cdot \mathbf{p}_0 \pmod{q}$. Because sVrfy accepts, for each $i \in \{0, 1\}$ and every $j \in [\tau]$ we have:

$$\|\mathbf{p}_j^i\|_{\infty}, \|\mathbf{s}_j^i\|_{\infty}, \|\mathbf{u}^i\|_{\infty} \leq \beta_{\text{agg}}, \quad (41)$$

as well as

$$\mathbf{G}_0 \cdot \mathbf{p}_{j-1}^i = \mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j^i + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j^i \pmod{q}, \quad (42)$$

$$\mathbf{G}_0 \cdot \mathbf{p}_\tau^i = \mathbf{B} \cdot \mathbf{u}^i, \quad (43)$$

$$\mathbf{G}_1 \cdot \mathbf{u}^i = \mathbf{m}_t^i. \quad (44)$$

From (41), it follows that for all $j \in \tau$, we have

$$\|\mathbf{p}_j^0 - \mathbf{p}_j^1\|_{\infty} \leq 2\beta_{\text{agg}}, \quad (45)$$

$$\|\mathbf{s}_j^0 - \mathbf{s}_j^1\|_{\infty} \leq 2\beta_{\text{agg}}, \quad (46)$$

$$\|\mathbf{u}^0 - \mathbf{u}^1\|_{\infty} \leq 2\beta_{\text{agg}}. \quad (47)$$

From (42), it follows that

$$\mathbf{G}_0 \cdot (\mathbf{p}_{j-1}^0 - \mathbf{p}_{j-1}^1) \quad (48)$$

$$= \mathbf{G}_0 \cdot \mathbf{p}_{j-1}^0 - \mathbf{G}_0 \cdot \mathbf{p}_{j-1}^1 \quad (49)$$

$$= (\mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j^0 + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j^0) - (\mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j^1 + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j^1) \quad (50)$$

$$= \mathbf{A}_{\tilde{t}_j} \cdot (\mathbf{p}_j^0 - \mathbf{p}_j^1) + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot (\mathbf{s}_j^0 - \mathbf{s}_j^1) \pmod{q}. \quad (51)$$

Similarly, from (43) and (44), it follows that

$$\mathbf{G}_0 \cdot (\mathbf{p}_\tau^0 - \mathbf{p}_\tau^1) = \mathbf{G}_0 \cdot \mathbf{p}_\tau^0 - \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \quad (52)$$

$$= \mathbf{B} \cdot \mathbf{u}^0 - \mathbf{B} \cdot \mathbf{u}^1 \quad (53)$$

$$= \mathbf{B} \cdot (\mathbf{u}^0 - \mathbf{u}^1). \quad (54)$$

$$\mathbf{G}_1 \cdot (\mathbf{u}^0 - \mathbf{u}^1) = \mathbf{m}_t^0 - \mathbf{m}_t^1. \quad (55)$$

Thus, all checks in wVrfy(pp, $\mathbf{c}_0 - \mathbf{c}_1, t, \text{op}_0 - \text{op}_1, 2\beta_{\text{agg}}$) go through. $\square$

Lemma 5.4 (Position Binding). *Let $d, q, \alpha_w, N, \eta, \tau, \xi, \omega, \beta_{\text{agg}} \in \mathbb{N}^+$ and $\epsilon_{\text{homH}} \in (0, 1)$ such that d is a power of 2 and q is a prime. If $\text{MSIS}_{q, \omega, 2\omega\kappa, 4\beta_{\text{agg}}\sqrt{2\omega\kappa d}}^{\ell_2}$ and $\text{MSIS}_{q, \omega, \xi\kappa, 4\beta_{\text{agg}}\sqrt{\xi\kappa d}}^{\ell_2}$ are hard, then the HVC-SOP construction HVC-SOP in Figure 4 is position binding as per Definition 2.10.*

Proof. Let $\mathcal{A}$ be an arbitrary PPT adversary against position binding. By the law of total probability it holds that

$$\Pr \left[\begin{array}{c} \text{pp} \leftarrow \text{Setup}(1^\lambda) \\ (c, t, \text{op}_0, \text{op}_1) \leftarrow \mathcal{A}(\text{pp}) \\ \mathbf{m}_0 \leftarrow \text{wVrfy}(\text{pp}, \mathbf{c}, t, \text{op}_0, 2\beta_{\text{agg}}) \\ \mathbf{m}_1 \leftarrow \text{wVrfy}(\text{pp}, \mathbf{c}, t, \text{op}_1, 2\beta_{\text{agg}}) \end{array} : \begin{array}{l} \mathbf{m}_0 \neq \mathbf{m}_1 \wedge \\ \perp \notin \{\mathbf{m}_0, \mathbf{m}_1\} \end{array} \right]$$

$$= \Pr[\text{Bad} \wedge \mathbf{G}_0 \cdot \mathbf{p}_\tau^0 = \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}] + \Pr[\text{Bad} \wedge \mathbf{G}_0 \cdot \mathbf{p}_\tau^0 \neq \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}],$$

where Bad denotes the event $\mathbf{m}_0 \neq \mathbf{m}_1$ and $\perp \notin \{\mathbf{m}_0, \mathbf{m}_1\}$, and $\text{op}^b = (\mathbf{p}_1^b, \dots, \mathbf{p}_\tau^b, \mathbf{s}_1^b, \dots, \mathbf{s}_\tau^b, \mathbf{u}^b)$ for $b \in \{0, 1\}$. We now bound the two probabilities separately.

Case 1: $\mathbf{G}_0 \cdot \mathbf{p}_\tau^0 = \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}$. By definition of wVrfy , we have $\mathbf{m}_i := \mathbf{G}_1 \cdot \mathbf{u}^i \pmod{q}$, and

$$\begin{aligned} & \Pr[\text{Bad} \wedge \mathbf{G}_0 \cdot \mathbf{p}_\tau^0 = \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}] \\ & \leq \Pr[\mathbf{G}_1 \cdot \mathbf{u}^0 \neq \mathbf{G}_1 \cdot \mathbf{u}^1 \pmod{q} \wedge \mathbf{B} \cdot \mathbf{u}^0 = \mathbf{B} \cdot \mathbf{u}^1 \pmod{q} \\ & \quad \wedge \|\mathbf{u}^0\|_\infty, \|\mathbf{u}^1\|_\infty \leq 2\beta_{\text{agg}}] \\ & \leq \Pr[\mathbf{u}^0 \neq \mathbf{u}^1 \wedge \mathbf{B} \cdot (\mathbf{u}^0 - \mathbf{u}^1) = \mathbf{0} \pmod{q}] \\ & \leq \Pr[\|\mathbf{u}^0 - \mathbf{u}^1\|_\infty \leq 4\beta_{\text{agg}} \sqrt{\xi \kappa d}] \\ & \leq \text{negl}(\lambda), \end{aligned}$$

where the last inequality follows from hardness of $\text{MSIS}_{q, \omega, \xi \kappa, 4\beta_{\text{agg}} \sqrt{\xi \kappa d}}^{\ell_2}$.

Case 2: $\mathbf{G}_0 \cdot \mathbf{p}_\tau^0 \neq \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}$. We construct a PPT algorithm $\bar{\mathcal{A}}$ that solves $\text{MSIS}_{q, \omega, 2\omega \kappa, 4\beta_{\text{agg}} \sqrt{2\omega \kappa d}}^{\ell_2}$ as follows. On input a uniformly random matrix $\mathbf{A} \in R_q^{\omega \times 2\omega \kappa}$, the algorithm $\bar{\mathcal{A}}$ parses $\mathbf{A}$ as a horizontal concatenation $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$ with $\mathbf{A}_0, \mathbf{A}_1 \in R_q^{\omega \times \omega \kappa}$, samples $\mathbf{B} \leftarrow R_q^{\omega \times \xi \kappa}$, sets $\text{pp} := (\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1)$, and runs $(\mathbf{c}, t, \text{op}_0, \text{op}_1) \leftarrow \mathcal{A}(\text{pp})$. For $b \in \{0, 1\}$, let $\mathbf{m}_b := \text{wVrfy}(\text{pp}, \mathbf{c}, t, \text{op}_b, 2\beta_{\text{agg}})$. If $\mathbf{m}_0 = \mathbf{m}_1$, or $\perp \in \{\mathbf{m}_0, \mathbf{m}_1\}$, or $\mathbf{G}_0 \cdot \mathbf{p}_\tau^0 = \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}$, then $\bar{\mathcal{A}}$ aborts. Otherwise, let $j^* \in [\tau]$ such that

$$j^* = \min\{\mathbf{G}_0 \cdot \mathbf{p}_{j^*}^0 \neq \mathbf{G}_0 \cdot \mathbf{p}_{j^*}^1 \pmod{q}\}, \quad (56)$$

which exists since $\mathbf{G}_0 \cdot (\mathbf{p}_\tau^0) \neq \mathbf{G}_0 \cdot (\mathbf{p}_\tau^1) \pmod{q}$ by assumption, and satisfies $j^* \geq 1$ as $\mathbf{G} \cdot \mathbf{p}_0^0 = \mathbf{G} \cdot \mathbf{p}_0^1 = \mathbf{c} \pmod{q}$. Therefore, we have $\mathbf{G}_0 \cdot (\mathbf{p}_{j^*-1}^0) = \mathbf{G}_0 \cdot (\mathbf{p}_{j^*-1}^1) \pmod{q}$.

Define the vector $\mathbf{z} \in R^{2\omega \kappa}$ by

$$\mathbf{z} := \begin{cases} (\mathbf{p}_{j^*}^0 - \mathbf{p}_{j^*}^1, \mathbf{s}_{j^*}^0 - \mathbf{s}_{j^*}^1) & \text{if } \tilde{t}_{j^*} = 0, \\ (\mathbf{s}_{j^*}^0 - \mathbf{s}_{j^*}^1, \mathbf{p}_{j^*}^0 - \mathbf{p}_{j^*}^1) & \text{if } \tilde{t}_{j^*} = 1. \end{cases} \quad (57)$$

Since $\mathbf{G}_0 \cdot \mathbf{p}_{j^*-1}^0 = \mathbf{G}_0 \cdot \mathbf{p}_{j^*-1}^1 \pmod{q}$ and both openings pass wVrfy , the linear consistency check at level j^* gives

$$\mathbf{G}_0 \cdot \mathbf{p}_{j^*-1}^b = \mathbf{A}_{\tilde{t}_{j^*}} \cdot \mathbf{p}_{j^*}^b + \mathbf{A}_{\tilde{t}_{j^*} \oplus 1} \cdot \mathbf{s}_{j^*}^b \pmod{q} \quad (58)$$

Subtracting the two equalities ($b \in \{0, 1\}$) from (58) yields

$$\mathbf{A}_{\tilde{t}_{j^*}} \cdot (\mathbf{p}_{j^*}^0 - \mathbf{p}_{j^*}^1) + \mathbf{A}_{\tilde{t}_{j^*} \oplus 1} \cdot (\mathbf{s}_{j^*}^0 - \mathbf{s}_{j^*}^1) = \mathbf{0} \pmod{q}. \quad (59)$$

By construction of $\mathbf{z}$ (swapping the two blocks depending on $\tilde{t}_{j^*}$) this is exactly $\mathbf{A} \cdot \mathbf{z} = \mathbf{0} \pmod{q}$. By definition of wVrfy , both openings satisfy $\|\mathbf{p}_{j^*}^b\|_\infty \leq 2\beta_{\text{agg}}$ and $\|\mathbf{s}_{j^*}^b\|_\infty \leq 2\beta_{\text{agg}}$ for $b \in \{0, 1\}$. Hence,

$$\|\mathbf{z}\|_\infty \leq \max\{\|\mathbf{p}_{j^*}^0\|_\infty, \|\mathbf{s}_{j^*}^0\|_\infty\} + \max\{\|\mathbf{p}_{j^*}^1\|_\infty, \|\mathbf{s}_{j^*}^1\|_\infty\} \quad (60)$$

$$\leq 4\beta_{\text{agg}}. \quad (61)$$

By choice of j^* we have $\mathbf{G}_0 \cdot \mathbf{p}_{j^*}^0 \neq \mathbf{G}_0 \cdot \mathbf{p}_{j^*}^1 \pmod{q}$, and thus $\mathbf{p}_{j^*}^0 \neq \mathbf{p}_{j^*}^1$, implying $\mathbf{z} \neq \mathbf{0}$.

Therefore, whenever $\bar{\mathcal{A}}$ succeeds in event $\text{Bad} \wedge \mathbf{G}_0 \cdot \mathbf{p}_\tau^0 \neq \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}$, the reduction $\bar{\mathcal{A}}$ outputs a nonzero $\mathbf{z}$ with $\|\mathbf{z}\|_2 \leq 4\beta_{\text{agg}} \sqrt{2\omega \kappa d}$ and $\mathbf{A} \cdot \mathbf{z} = \mathbf{0} \pmod{q}$, solving $\text{MSIS}_{q, \omega, 2\omega \kappa, 4\beta_{\text{agg}} \sqrt{2\omega \kappa d}}^{\ell_2}$. Hence,

$$\Pr[\text{Bad} \wedge \mathbf{G}_0 \cdot \mathbf{p}_\tau^0 \neq \mathbf{G}_0 \cdot \mathbf{p}_\tau^1 \pmod{q}] \leq \text{negl}(\lambda). \quad (62)$$

Combining the two cases, we conclude that $\Pr[\text{Bad}] \leq \text{negl}(\lambda)$, as required. $\square$

Lemma 5.5 (Opening Proofs Succinctness). *If the LaBRADOR proof system LB is succinct, then the HVC-SOP scheme HVC-SOP in Figure 4 has succinct opening proofs as per Definition 3.2.*

Proof. Let $\lambda \in \mathbb{N}^+$, $\mathbf{m} \in \mathcal{S}_{\text{dom}}^{2^\tau}$, $t \in [2^\tau]$, $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, $\mathbf{c} \leftarrow \text{Com}(\text{pp}, \mathbf{m})$, $\text{op} \leftarrow \text{Open}(\text{pp}, \mathbf{c}, \mathbf{m}, t)$, and $\pi \leftarrow \text{pOpen}(\text{pp}, \mathbf{c}, t, \text{op})$.

By definition of pOpen , the proof is generated via $\pi \leftarrow \text{LB.Prove}(\text{pp}_{\text{LB}}, x, w)$, where the statement is $x = (\mathbf{c}, t, \mathbf{m}_t)$ and the witness is $w = \text{op}$. By the succinctness property of the LaBRADOR proof system, there exists a polynomial P such that the proof length is bounded by:

$$|\pi| \leq P(\lambda, |x|, \log |w|).$$

Note that the statement length $|x| = |\mathbf{c}| + |t| + |\mathbf{m}_t|$ depends only on the public commitment, index, and target message, whereas the witness length $|w| = |\text{op}|$ scales with the full opening vector size.

For fixed parameters λ and $|x|$, any polynomial in $\log |w|$ grows strictly sublinearly relative to $|w|$, which implies:

$$\lim_{|w| \rightarrow \infty} \frac{P(\lambda, |x|, \log |w|)}{|w|} = 0.$$

Consequently, there exists a threshold length $\ell_m \in \mathbb{N}^+$ such that for all $\mathbf{m} \in \mathcal{S}_{\text{dom}}^{2^\tau}$ with $|\mathbf{m}| > \ell_m$, we have $P(\lambda, |x|, \log |\text{op}|) < |\text{op}|$. Thus, $\ell_{\text{prf}}(\lambda, \mathbf{m}, t) < \ell_{\text{op}}(\lambda, \mathbf{m}, t)$, completing the proof. $\square$

Lemma 5.6 (Opening Proofs Completeness). *If the LaBRADOR proof system LB is ϵ_{comp} -complete, then the HVC-SOP scheme HVC-SOP in Figure 4 has ϵ_{comp} -complete opening proofs as per Definition 3.3.*

Proof. Let $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, $\mathbf{m} \in (R_q^\xi)^{2^\tau}$, $t \in [2^\tau]$, $\mathbf{c} \leftarrow \text{Com}(\text{pp}, \mathbf{m})$, and $\text{op} \leftarrow \text{Open}(\text{pp}, \mathbf{c}, \mathbf{m}, t)$. Parse $\text{op} = ((\mathbf{p}_j, \mathbf{s}_j)_{j=1}^\tau, \mathbf{u})$ and let $\mathbf{m}_t$ be the t -th entry of $\mathbf{m}$. By Lemma 5.1, the honestly generated opening op satisfies

$$\begin{aligned} \mathbf{B} \cdot \mathbf{u} &= \mathbf{G}_0 \cdot \mathbf{p}_\tau \pmod{q}, \\ \mathbf{A}_{\tilde{t}_1} \cdot \mathbf{p}_1 + \mathbf{A}_{\tilde{t}_1 \oplus 1} \cdot \mathbf{s}_1 &= \mathbf{c} \pmod{q}, \\ \mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j &= \mathbf{G}_0 \cdot \mathbf{p}_{j-1} \pmod{q} \quad \text{for all } 2 \leq j \leq \tau, \\ \mathbf{G}_1 \cdot \mathbf{u} &= \mathbf{m}_t \pmod{q}, \end{aligned}$$

along with the norm bounds:

$$\|\mathbf{p}_j\|_\infty, \|\mathbf{s}_j\|_\infty, \|\mathbf{u}\|_\infty \leq \eta \leq \beta_{\text{agg}} \quad \text{for all } j \in [\tau].$$

Thus, $w = \text{op}$ is a valid witness for the statement $x = (\mathbf{c}, t, \mathbf{m}_t)$ under relation $\mathcal{R}_{\text{LB}}$ as specified in (5)–(9). By the ϵ_{comp} -completeness of LB, the prover algorithm $\pi \leftarrow \text{LB.Prove}(\text{pp}_{\text{LB}}, x, w)$ outputs an accepting proof π such that $\text{LB.Verify}(\text{pp}_{\text{LB}}, x, \pi) = 1$ with probability at least $1 - \epsilon_{\text{comp}}$. Consequently, $\text{pVrfy}(\text{pp}, \mathbf{c}, t, \mathbf{m}_t, \pi) = 1$ with probability $1 - \epsilon_{\text{comp}}$. $\square$

Lemma 5.7 (Opening Knowledge Soundness). *If the LaBRADOR proof system LB is ϵ_{ext} -knowledge sound for relation $\mathcal{R}_{\text{LB}}$, then the HVC-SOP scheme HVC-SOP in Figure 4 is ϵ_{ext} -opening knowledge sound as per Definition 3.4.*

Proof. Let $\text{pp} \leftarrow \text{Setup}(1^\lambda)$, and $\mathbf{m} \leftarrow \mathcal{S}_{\text{dom}}^{2^\tau}$. Let $\mathcal{A}$ be a PPT adversary that outputs $(\mathbf{c}, t, \mathbf{m}_t, \pi)$ such that $\text{pVrfy}(\text{pp}, \mathbf{c}, t, \pi) = 1$. By definition of pVrfy , this implies $\text{LB.Verify}(\text{pp}_{\text{LB}}, x, \pi) = 1$ for the statement $x = (\mathbf{c}, t, \mathbf{m}_t)$.

By the knowledge soundness of LB, there exists a PPT extractor $\mathcal{E}_{\text{LB}}$ that extracts a witness $w^* = \text{op}^* = ((\mathbf{p}_j^*, \mathbf{s}_j^*)_{j=1}^\tau, \mathbf{u}^*)$ from $\mathcal{A}$ such that $(x, w^*) \in \mathcal{R}_{\text{LB}}$, except with probability ϵ_{ext} . If $(x, w^*) \in \mathcal{R}_{\text{LB}}$, the extracted witness op^* satisfies $\mathbf{m}_t = \text{sVrfy}(\text{pp}, \mathbf{c}, t, \text{op}^*, \delta_{\text{LB}} \cdot \beta_{\text{agg}})$ with $\delta_{\text{ext}} = \delta_{\text{LB}}$. Thus, we complete the proof. $\square$

Lemma 5.8 (Aggregate Opening Compatibility). *Let $\epsilon_{\text{agg}} \geq \epsilon_{\text{homH}} + \epsilon_{\text{comp}}$. If HVC-SOP is $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\text{homH}})$ -probabilistically homomorphic and LB is complete, then the HVC-SOP scheme HVC-SOP in Figure 4 is aggregate opening compatible as per Definition 3.5.*

Proof. Let $\{(\mathbf{c}_i, \mathbf{m}_i^i, \text{op}_i)\}_{i \in [N]}$ be N valid triples such that $\text{iVrfy}(\text{pp}, \mathbf{c}_i, t, \text{op}_i, \eta) = \mathbf{m}_i^i \neq \perp$. Let $w_1, \dots, w_N \leftarrow \mathcal{T}_{\alpha_w}$ be sampled weights, and compute the aggregated values:

$$\mathbf{c}^* := \sum_{i=1}^N w_i \mathbf{c}_i, \quad \text{op}^* := \sum_{i=1}^N w_i \text{op}_i, \quad \mathbf{m}_t^* := \sum_{i=1}^N w_i \mathbf{m}_i^i.$$

By Lemma 5.2, with probability at least $1 - \epsilon_{\text{homH}}$, the linear combination op^* satisfies $\text{sVrfy}(\text{pp}, \mathbf{c}^*, t, \text{op}^*, \beta_{\text{agg}}) = \mathbf{m}_t^*$. By the definition of sVrfy , this implies that op^* satisfies all linear relations and norm bounds required by $\mathcal{R}_{\text{LB}}$ as specified in (5)–(9) for the statement $x^* = (\mathbf{c}^*, t, \mathbf{m}_t^*)$. Consequently, $w^* = \text{op}^*$ is a valid witness for x^* .

Running $\pi^* \leftarrow \text{pOpen}(\text{pp}, \mathbf{c}^*, t, \text{op}^*)$ invokes $\text{LB.Prove}(\text{pp}_{\text{LB}}, x^*, \text{op}^*)$. By the completeness of LB, we have $\text{LB.Verify}(\text{pp}_{\text{LB}}, x^*, \pi^*) = 1$ with probability $1 - \epsilon_{\text{comp}}$. Combining above, we obtain

$$\Pr[\text{pVrfy}(\text{pp}, \mathbf{c}^*, t, \mathbf{m}_t^*, \pi^*) = 1] \geq 1 - \epsilon_{\text{homH}} - \epsilon_{\text{comp}},$$

which completes the proof of aggregate opening compatibility. $\square$

Theorem 5.2 (Security of Lemur+). *Let $\lambda, d, q, N, n, k, \xi, \gamma, \tau, \alpha_w \in \mathbb{N}^+$, with d being a power of two, q being prime. Let $\epsilon_{\text{cor}}, \epsilon_{\text{homH}}, \epsilon_{\text{ext}}, \epsilon_{\text{agg}} \in [0, 1)$ such that $(\epsilon_{\text{homH}} + \epsilon_{\text{agg}} + \epsilon_{\text{homK}})^\gamma \leq 2^{-\lambda}$. Let $\delta_{\text{LB}} \geq 1$. Let $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. Let $W = \mathcal{T}_{\alpha_w} \subseteq R$ be a set such that $|W| > 2^\lambda$ and let $W' := \{w_0 - w_1 \mid w_0, w_1 \in W\}$. Let $H_{\text{agg}} : \{0, 1\}^* \rightarrow W^N$ be a random oracle. Let KOTS be a key-homomorphic one-time signature scheme with public keys in $R_q^{k \times n}$ as defined in Figure 6 and let HVC-SOP be a homomorphic vector commitment for domain R_q^ξ as defined in Figure 4, then the synchronized multi-signature construction in Figure 3 is an unforgeable synchronized multi-signature scheme that is ϵ_{cor} -individually correct and has correct aggregation in the presence of rogue keys and signatures.*

Proof. The theorem directly follows from applying Theorems C.1, 4.1 and 5.1. $\square$

6 Multi-Hop Aggregation

In this section, we consider a hierarchical distributed setting in which local aggregators first combine signatures from subsets of signers, and higher-level aggregators recursively combine those intermediate results. This is particularly relevant to voting in Proof-of-Stake blockchains [20]. Existing multi-hop constructions either rely on verification hints such as signer multiplicities and aggregation topology, or depend on highly impractical primitives like indistinguishability obfuscation [20]. We show that Lemur+ is a natural fit for multi-hop aggregation. Although it also requires the aggregation topology as a verification hint, it is significantly more lightweight than prior topology-reliant constructions (such as batch-argument schemes [12, 13] or mergeable SNARGs [8]) by exploiting the inherent algebraic linearity of our underlying HVC-SOP and KOTS.

High Level Overview. We define a “plain” Lemur+ aggregate signature as the following procedure: it runs SigAgg in Figure 3, but the output replaces the opening proof with an explicit aggregated HVC-SOP opening. Consider a collection of disjoint signer groups $P_1, \dots, P_M$, where $P_r = (\mathbf{c}_{r,1}, \dots, \mathbf{c}_{r,N_r})$ is the ordered public-key list of group r . All signers authenticate the same message μ at the same time step t . The two-hop aggregation proceeds in two stages:

1. Each local aggregator computes a plain Lemur+ aggregate signature, the aggregated public key, and the aggregated one-time public keys

$$\Sigma_r^{(1)} = (\text{op}_r^{(1)}, \sigma_{\text{kots}, r}^{(1)}, j_r^{(1)}), \quad (63)$$

$$\mathbf{c}_r^{(1)} := \sum_{i=1}^{N_r} w_{r,i} \cdot \mathbf{c}_{r,i}, \quad (64)$$

$$\mathbf{t}_r^{(1)} \leftarrow \text{HVC-SOP.sVrfy}^{(1)}(\text{pp}_{\text{hvc sop}}, \mathbf{c}_r^{(1)}, t, \text{op}_r^{(1)}) \quad (65)$$

for the signer list P_r , where $\text{op}_r^{(1)}$ is the aggregated HVC-SOP opening (not the opening *proof*), $\sigma_{\text{kots},r}^{(1)}$ is the aggregated KOTS signature, and $j_r^{(1)}$ is counter.

2. A higher-level aggregator verifies each local aggregate signature $\Sigma_r^{(1)}$, derives fresh weights ρ_r using a random oracle, and computes

$$\mathbf{c}^{(2)} := \sum_{r=1}^M \rho_r \cdot \mathbf{c}_r^{(1)}, \quad (66)$$

$$\mathbf{t}^{(2)} := \sum_{r=1}^M \rho_r \cdot \mathbf{t}_r^{(1)}, \quad (67)$$

$$\text{op}^{(2)} := \sum_{r=1}^M \rho_r \cdot \text{op}_r^{(1)}, \quad (68)$$

$$\sigma_{\text{kots}}^{(2)} := \sum_{r=1}^M \rho_r \cdot \sigma_{\text{kots},r}^{(1)}, \quad (69)$$

$$\pi \leftarrow \text{HVC-SOP.pOpen}(\text{pp}_{\text{hvc sop}}, \mathbf{c}^{(2)}, t, \text{op}^{(2)}). \quad (70)$$

The final aggregate is $\Sigma_r^{(2)} = (\pi, \mathbf{t}^{(2)}, \sigma_{\text{kots},r}^{(2)}, j_r^{(2)})$.

The idea extends naturally to more than two hops. Intermediate nodes output plain Lemur+ aggregates, while the root of the aggregation tree replaces the final aggregated opening with a Lemur+ opening proof. We show more detail in the following subsections.

6.1 First-Hop Aggregation

For every group $r \in [M]$, the local aggregator receives individually valid signatures

$$S_r = (\sigma_{r,1}, \dots, \sigma_{r,N_r}), \quad \sigma_{r,i} = (\text{op}_{r,i}, \sigma_{\text{kots},r}^i). \quad (71)$$

It first pre-verifies every individual signature. It then derives

$$(w_{r,1}, \dots, w_{r,N_r}) := H_{\text{agg}}(P_r, t, \mu, 1, j_r^{(1)}) \quad (72)$$

and computes

$$\mathbf{c}_r^{(1)} := \sum_{i=1}^{N_r} w_{r,i} \cdot \mathbf{c}_{r,i}, \quad (73)$$

$$\text{op}_r^{(1)} := \sum_{i=1}^{N_r} w_{r,i} \cdot \text{op}_{r,i}, \quad (74)$$

$$\sigma_{\text{kots},r}^{(1)} := \sum_{i=1}^{N_r} w_{r,i} \cdot \sigma_{\text{kots},r}^i. \quad (75)$$

The additional input 1 to H_{agg} identifies the aggregation depth. It prevents a weight derived for one hop from being reused at another hop. The first-hop aggregate signature is

$$\Sigma_r^{(1)} := (\text{op}_r^{(1)}, \sigma_{\text{kots},r}^{(1)}, j_r^{(1)}). \quad (76)$$

It is verified under the commitment $\mathbf{c}_r^{(1)}$ by checking

$$\mathbf{t}_r^{(1)} \leftarrow \text{HVC-SOP.sVrfy}^{(1)}(\text{pp}_{\text{hvc sop}}, \mathbf{c}_r^{(1)}, t, \text{op}_r^{(1)}), \quad (77)$$

$$\text{KOTS.sVrfy}^{(1)}(\text{pp}_{\text{kots}}, \mathbf{t}_r^{(1)}, \mu, \sigma_{\text{kots},r}^{(1)}). \quad (78)$$

The superscript (1) indicates that the verification algorithms use first-hop norm bounds

$$\beta_{\text{agg}}^{(1)} \quad \text{and} \quad \beta_{\sigma}^{(1)}. \quad (79)$$

We show how to derive these bounds in Section 6.4. At this point, every group outputs one plain Lemur+ aggregate.

6.2 Second-Hop Aggregation

Let $\Sigma_1^{(1)}, \dots, \Sigma_M^{(1)}$ be valid first-hop aggregates for the same pair (t, μ) . Before combining them, the second-hop aggregator verifies every aggregate under the commitment $\mathbf{c}_r^{(1)}$. The complete ordered public key list is $P := P_1 \parallel \dots \parallel P_M$. The second-hop aggregator derives fresh weights

$$(\rho_1, \dots, \rho_M) := H_{\text{agg}}(P, t, \mu, 2, (\mathbf{c}_r^{(1)}, P_r, j_r^{(1)})_{r \in [M]}, j^{(2)}). \quad (80)$$

The hash input binds the weights to the full signer list P , time step t , message μ , aggregation depth 2, level 1 commitment for each group $\mathbf{c}_r^{(1)}$, each group's signer set P_r and first-hop aggregation counter $j_r^{(1)}$, and the second-hop aggregation counter $j^{(2)}$. The second-hop aggregator computes

$$\mathbf{c}^{(2)} := \sum_{r=1}^M \rho_r \cdot \mathbf{c}_r^{(1)}, \quad (81)$$

$$\text{op}^{(2)} := \sum_{r=1}^M \rho_r \cdot \text{op}_r^{(1)}, \quad (82)$$

$$\sigma_{\text{kots}}^{(2)} := \sum_{r=1}^M \rho_r \cdot \sigma_{\text{kots},r}^{(1)}. \quad (83)$$

The second-hop aggregate signature is

$$\Sigma^{(2)} := (\text{op}^{(2)}, \sigma_{\text{kots}}^{(2)}, j^{(2)}). \quad (84)$$

How the verification works? For child aggregate r , let

$$\mathbf{t}_r^{(1)} = \text{HVC-SOP.sVrfy}^{(1)}(\text{pp}_{\text{hvc sop}}, \mathbf{c}_r^{(1)}, t, \text{op}_r^{(1)}) \quad (85)$$

be its aggregated one-time public key. Because HVC-SOP strong verification algebraic checks are linear in the commitment and opening, it holds that

$$\mathbf{t}^{(2)} := \sum_{r=1}^M \rho_r \cdot \mathbf{t}_r^{(1)} \quad (86)$$

$$= \sum_{r=1}^M \rho_r \cdot \text{HVC-SOP.sVrfy}^{(1)}(\text{pp}_{\text{hvc sop}}, \mathbf{c}_r^{(1)}, t, \text{op}_r^{(1)}) \quad (87)$$

$$= \text{HVC-SOP.sVrfy}^{(2)}(\text{pp}_{\text{hvc sop}}, \mathbf{c}^{(2)}, t, \text{op}^{(2)}), \quad (88)$$

provided that the second-hop opening satisfies its prescribed norm bounds. The KOTS equations are linear as well. According to Figure 6, it holds that

$$(\sigma_{\text{kots},r}^{(1)})^\top \mathbf{A} = \mathbf{h}^\top \mathbf{T}_r^{(1)} \pmod{q} \quad (89)$$

for every child aggregate, where $\mathbf{T}_r^{(1)}$ is the matrix form of $\mathbf{t}_r^{(1)}$. Summing the equations with weights ρ_r gives

$$\left(\sum_{r=1}^M \rho_r \cdot \sigma_{\text{kots},r}^{(1)} \right)^\top \mathbf{A} = \mathbf{h}^\top \left(\sum_{r=1}^M \rho_r \mathbf{T}_r^{(1)} \right) \quad (90)$$

$$(\sigma_{\text{kots}}^{(2)})^\top \mathbf{A} = \mathbf{h}^\top \mathbf{T}^{(2)} \pmod{q}, \quad (91)$$

where $\mathbf{T}^{(2)} := \sum_{r=1}^M \rho_r \mathbf{T}_r^{(1)}$ and $\mathbf{T}^{(2)}$ is the matrix form $\mathbf{t}^{(2)}$. Thus, the second-hop aggregate satisfies HVC-SOP and KOTS strong verification linear relation as the one-hop aggregate. Only their norm bounds differ.

6.3 Multi-Hop Aggregation

The two-hop aggregate can be written directly as a weighted sum of the individual signatures. Substituting (73)–(75) into (81)–(83) gives

$$\mathbf{c}^{(2)} = \sum_{r=1}^M \sum_{i=1}^{N_r} \hat{w}_{r,i}^{(2)} \cdot \mathbf{c}_{r,i}, \quad (92)$$

$$\text{op}^{(2)} = \sum_{r=1}^M \sum_{i=1}^{N_r} \hat{w}_{r,i}^{(2)} \cdot \text{op}_{r,i}, \quad (93)$$

$$\sigma_{\text{kots}}^{(2)} = \sum_{r=1}^M \sum_{i=1}^{N_r} \hat{w}_{r,i}^{(2)} \cdot \sigma_{\text{kots},r}^i, \quad (94)$$

where

$$\hat{w}_{r,i}^{(2)} := \rho_r \cdot w_{r,i}. \quad (95)$$

Therefore, a two-hop Lemur+ aggregate is algebraically equivalent to a flat aggregate whose weights are products of first-hop and second-hop weights. More generally, consider a signer i whose contribution passes through h aggregation hops. If

$$w_i^{(1)}, w_i^{(2)}, \dots, w_i^{(h)} \quad (96)$$

are the weights assigned along the path from signer i to the final aggregator, then the effective weight of signer i is

$$\hat{w}_i^{(h)} := \prod_{\ell=1}^h w_i^{(\ell)}. \quad (97)$$

Compute

$$\mathbf{c}^{(h)} = \sum_i \hat{w}_i^{(h)} \mathbf{c}_i, \quad (98)$$

$$\mathbf{t}^{(h)} = \sum_i \hat{w}_i^{(h)} \mathbf{t}_i, \quad (99)$$

$$\text{op}^{(h)} = \sum_i \hat{w}_i^{(h)} \text{op}_i, \quad (100)$$

$$\sigma_{\text{kots}}^{(h)} = \sum_i \hat{w}_i^{(h)} \sigma_{\text{kots}}^i, \quad (101)$$

$$\pi \leftarrow \text{HVC-SOP.pOpen}(\text{pp}_{\text{hvcSOP}}, \mathbf{c}^{(h)}, t, \text{op}^{(h)}). \quad (102)$$

The above equations also reveal the principal cost of additional hops: the random weights are products rather than single samples from $\mathcal{W}$, and hence the HVC-SOP and KOTS norm bounds grow with the aggregation depth. The final Lemur+ multi-hop aggregate is $(\pi, \mathbf{t}^{(h)}, \sigma_{\text{kots}}^{(h)}, j^{(h)})$.

6.4 Hop-Dependent Norm Bounds

The one-hop Lemur+ bounds cannot be reused unchanged at every level. A second-hop aggregator combines signatures that are already aggregated and are therefore larger than individual signatures. For every hop $\ell \geq 1$, let

$$\beta_{\text{agg}}^{(\ell)} \quad \text{and} \quad \beta_{\sigma}^{(\ell)} \quad (103)$$

denote the strong-verification bounds for an ℓ -hop HVC-SOP opening and KOTS signature, respectively. The first hop Lemur+ bounds are

$$\beta_{\text{agg}}^{(1)} := \beta_{\text{agg}}, \quad \beta_{\sigma}^{(1)} := \beta_{\sigma}. \quad (104)$$

To derive the recursive relation between bounds, let $\mathcal{W} = \mathcal{T}_{\alpha_w}$ denote the challenge set from which the weights $(w_{r,i})$ and (ρ_r) are sampled, applying the triangle inequality to the second-hop combination in (82) and (83), we obtain:

$$\|\text{op}^{(2)}\|_{\infty} = \left\| \sum_{r=1}^M \rho_r \text{op}_r^{(1)} \right\|_{\infty} \leq \sum_{r=1}^M \alpha_w \|\text{op}_r^{(1)}\|_{\infty} \quad (105)$$

$$\leq M \cdot \alpha_w \cdot \beta_{\text{agg}}^{(1)}, \quad (106)$$

$$\|\sigma_{\text{kots}}^{(2)}\|_{\infty} = \left\| \sum_{r=1}^M \rho_r \sigma_{\text{kots},r}^{(1)} \right\|_{\infty} \leq \sum_{r=1}^M \alpha_w \|\sigma_{\text{kots},r}^{(1)}\|_{\infty} \quad (107)$$

$$\leq M \cdot \alpha_w \cdot \beta_{\sigma}^{(1)}. \quad (108)$$

Therefore, the minimal valid second-hop norm bounds are:

$$\beta_{\text{agg}}^{(2)} := M \cdot \alpha_w \cdot \beta_{\text{agg}}^{(1)}, \quad \beta_{\sigma}^{(2)} := M \cdot \alpha_w \cdot \beta_{\sigma}^{(1)}. \quad (109)$$

By induction, for an ℓ -hop aggregation step combining M_{ℓ} child aggregates from depth $\ell - 1$, the bounds scale recursively

#	Source	Constraint
1	Theorem 5.1	$\beta_{\text{agg}} \geq \eta \sqrt{2\alpha_w N \ln \left(\frac{2d\kappa(2\tau\omega + \xi)}{\epsilon_{\text{homH}}} \right)}$
2	Theorem 5.1	$\text{MSIS}_{q,\omega,2\omega\kappa,4\delta_{\text{LB}}\beta_{\text{agg}}\sqrt{2\omega\kappa d}}^{\ell_2}$ is hard
3	Theorem 5.1	$\text{MSIS}_{q,\omega,\xi\kappa,4\delta_{\text{LB}}\beta_{\text{agg}}\sqrt{\xi\kappa d}}^{\ell_2}$ is hard
4	Theorem 5.1	$\epsilon_{\text{agg}} \geq \epsilon_{\text{homH}} + \epsilon_{\text{comp}}$
5	Theorem 5.2	$(\epsilon_{\text{homH}} + \epsilon_{\text{agg}} + \epsilon_{\text{homK}})^\gamma \leq 2^{-\lambda}$
6	Theorem 5.2	$ \mathcal{T}_{\alpha_w} = \binom{d}{\alpha_w} \cdot 2^{\alpha_w} \geq 2^\lambda$
7	Theorem C.1	$\alpha \geq \sqrt{2}\eta_{\epsilon_1}(\mathbb{Z})$
8	Theorem C.1	$\beta_z \geq 6\alpha\sqrt{1 + (k-1)\alpha_H}$
9	Theorem C.1	$\epsilon_{\text{cor}} \geq md \cdot \left(2^{-162} + 2(k-1)\alpha_H\epsilon_1 \right)$
10	Theorem C.1	$\beta_\sigma \geq \beta_z \cdot \sqrt{2\alpha_w N \cdot \ln \left(\frac{2md}{\epsilon_{\text{homK}}} \right)}$
11	Theorem C.1	$d > t_1 > 1$ and $d > t_2 > 1$ are powers of 2
12	Theorem C.1	$p \equiv 2t_1 + 1 \pmod{4t_1}$
13	Theorem C.1	$q \equiv 2t_2 + 1 \pmod{4t_2}$
14	Theorem C.1	$k \geq 4$
15	Theorem C.1	$\alpha \geq p\sqrt{2\alpha_H \ln(2(k-3)d(1+1/\epsilon_3))}/\pi$
16	Theorem C.1	$\left(\frac{1+\epsilon_3}{1-\epsilon_3} \right)^{2m} \cdot p^{-dm/t_1} < 2^{-\lambda}$
17	Theorem C.1	$ \mathcal{T}_{\alpha_H} ^{k-3} \geq 2^{2\lambda}$
18	Theorem C.1	$\text{DualHintMLWE}_{q,m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$ is hard
19	Theorem C.1	$\text{MSIS}_{q,n,m,2\beta_\sigma+2\alpha_w\beta_z}^{\ell_\infty}$ is hard

Table 3: Parameter constraints of Lemur+.

as:

$$\beta_{\text{agg}}^{(\ell)} := M_\ell \cdot \alpha_w \cdot \beta_{\text{agg}}^{(\ell-1)} = \left(\prod_{k=1}^{\ell} M_k \right) \cdot \alpha_w^\ell \cdot \beta_{\text{agg}}, \quad (110)$$

$$\beta_\sigma^{(\ell)} := M_\ell \cdot \alpha_w \cdot \beta_\sigma^{(\ell-1)} = \left(\prod_{k=1}^{\ell} M_k \right) \cdot \alpha_w^\ell \cdot \beta_\sigma, \quad (111)$$

where $M_1 = N_r$ is the group size, M_ℓ for $\ell > 1$ is the group size of higher levels, and $\beta_{\text{agg}}, \beta_\sigma$ are the baseline single aggregate bounds.

7 Parameter Setting

We now describe the concrete parameter selection for Lemur+. We provide a parameter generation script under the `/params` folder in our artifact. The script is built on the parameter generation script of Lemur, where MSIS hardness is estimated using the state-of-the-art Dilithium security estimation framework², and MLWE hardness is evaluated using the lattice estimator³ [2]. Our goal is to minimize the multi-signature size while maintaining practical aggregation and verification

²<https://github.com/pq-crystals/security-estimates>

³<https://github.com/malb/lattice-estimator>

time at 128-bit security level. In Appendix A, Table 3 lists all the parameter constraints of Lemur+. Table 5 provides the parameter setting that we find using the parameter generation script.

Global parameters. We consider Merkle tree heights τ ranging from 12 to 27 (i.e., approximately 11 hours to 42 years at a 10-second signing interval) and signer populations N ranging from 2^{10} to 2^{20} . Unlike Lemur, which instantiates KOTS and HVC over distinct moduli, Lemur+ uses a common modulus. The LaBRADOR backend is parameterized by byte-aligned modulus sizes, and therefore we choose its native 32-bit modulus $q = 2^{32} - 99$ for both KOTS and HVC-SOP. The two components and the LaBRADOR proof relation are then defined over the same ring $R_q = \mathbb{Z}_q[X]/(X^d + 1)$. This common-modulus design removes the need for cross-modulus conversion and lifting constraints when expressing the cryptographic relations in the proof system.

We then fix $\lambda = 128$, $d = 256$, $k = 4$, and set the probabilistic homomorphism error to $\epsilon_{\text{homH}} = 2^{-15}$, $\epsilon_{\text{homK}} = 2^{-15}$, the completeness error $\epsilon_{\text{comp}} = 2^{-30}$, and the aggregate openings compatibility error $\epsilon_{\text{agg}} = 2^{-14}$, which satisfy Constraint 4. The number of aggregation attempts γ is then determined by Constraint 5, which is set $\gamma = 10$. The aggregation weights are sampled from the ternary set $\mathcal{T}_{\alpha_w}$. We choose the smallest α_w for which $|\mathcal{T}_{\alpha_w}| \geq 2^\lambda$ (Constraint 6), which gives $\alpha_w = 23$. The KOTS sampling parameters are inherited from the concrete analysis of Lemur and are fixed to $\alpha = 83$, $\alpha_H = 60$, $\beta_z = 6700$.

KOTS parameters. We first determine KOTS parameters. The KOTS construction and its security theorem are reviewed in Appendix C. For each candidate pair (n, m) , the KOTS parameter generator evaluates the hardness of the underlying MSIS problem $\text{MSIS}_{q,n,m,\beta_{\text{kots}}}^{\ell_\infty}$, where $\beta_{\text{kots}} = 2\beta_\sigma + 2\alpha_w\beta_z$ and the MLWE problem $\text{DualHintMLWE}_{q,n,m,\alpha}$ (Constraints 18 and 19). We retain only the candidates that meet the target security level and minimize the aggregated KOTS size, and use them in the subsequent search for the HVC-SOP parameters.

HVC-SOP parameters. We next select the HVC-SOP parameters. For each candidate (ω, κ) , we choose the smallest η such that $(2\eta + 1)^\kappa \geq q$. The aggregated HVC-SOP opening has ℓ_∞ norm bounded by β_{agg} (Constraint 1). According to Theorem 5.1, the underlying security assumptions of HVC-SOP are $\text{MSIS}_{q,\omega,2\omega\kappa,4\delta_{\text{LB}}\beta_{\text{agg}}\sqrt{2\omega\kappa d}}^{\ell_2}$ and $\text{MSIS}_{q,\omega,\xi\kappa,4\delta_{\text{LB}}\beta_{\text{agg}}\sqrt{\xi\kappa d}}^{\ell_2}$, where δ_{LB} is the relaxed factor in LaBRADOR proofs.

LaBRADOR parameters. For each feasible HVC-SOP candidate, we jointly optimize its LaBRADOR witness representation. For every opening component $X \in \{\mathbf{p}_1, \dots, \mathbf{p}_\tau, \mathbf{s}_1, \dots, \mathbf{s}_\tau, \mathbf{u}\}$, the honest prover applies the coefficient-wise integer decomposition $X = X_0 + bX_1$, where b is an even decomposition base. LaBRADOR commits directly to X_0 and X_1 , and the HVC-SOP relations are expressed

using the public base b . To reduce the proof cost, we concatenate all decomposed opening components into a single vector and prove one global ℓ_2 -norm constraint. Following [5], let $\delta_{\text{LB}} = \sqrt{128/30}$ denote the relaxation factor. Consequently, the reconstructed extracted opening has ℓ_2 norm at most

$$\beta_{\text{ext}} = \delta_{\text{LB}} \sqrt{d(2\tau\omega + \xi)\kappa(1 + b^2)((b/2)^2 + (\beta_{\text{agg}}/b)^2)}.$$

The corresponding position-binding collision bound is $\beta_{\text{HVC}} = 2\beta_{\text{ext}}$, which is used in both concrete ℓ_2 -MSIS estimates. For each HVC-SOP candidate, we select the feasible base b that minimizes β_{HVC} .

We note that our generic construction treats $\mathbf{m}_t$ as an explicit statement input to the HVC-SOP proof. In our implementation, instead, we keep $\mathbf{m}_t$ as part of the LaBRADOR witness and incorporate the aggregate KOTS verification relation into the same proof, thereby avoiding transmitting $\mathbf{m}_t$.

In the end, following Lemur, we use a canonical Rice-based encoding for the short components of the LaBRADOR proof. Thus, the aggregate signature length is $|\sigma_{\text{KOTS}}| + |\pi| + \lceil \log_2 \gamma \rceil$. For each pair (τ, N) , we select the feasible parameter set minimizing this aggregate signature length.

8 Implementation and Benchmarks

We implement the complete Lemur+ construction and evaluate its performance across multiple parameter sets selected in the preceding section. Our implementation⁴ combines an optimized Rust implementation of Lemur+ with the C reference implementation of LaBRADOR [5]. The Rust component implements the complete KOTS and HVC-SOP pipelines, while a C adapter passes the decomposed opening witness and the corresponding public linear constraints to LaBRADOR through its Chihuahua front end [23]. We do not modify the LaBRADOR proof backend. We also optimize the implementation by parallelizing signer-linear work, fusing individual verification with opening aggregation, caching public matrices in NTT form, and using BDS traversal [9] to maintain signer state.

We compare against Lemur built on the same machine using the same optimizations for the shared Rust primitives. Experiments were conducted on an Intel Core i7-11850H CPU with 8 cores and 16 hardware threads, using rustc 1.92.0 and GCC 13.3.0. Communication is measured from canonical serialized outputs. All reported running times are averaged over 100 repetitions.

Figure 5 compares the Rice-encoded aggregate signature sizes of Lemur and Lemur+ for $N \in \{2^{10}, 2^{15}, 2^{20}\}$ across different Merkle tree heights τ . From the figure, Lemur+ remains an almost constant aggregate signature size of roughly 50 KB across different τ and N , while Lemur scales linear with τ and $\log N$. These results demonstrate that Lemur+ is particularly suitable for large signer populations and long key lifetimes.

⁴<https://github.com/dingdangdangwhx/Lemur-plus>

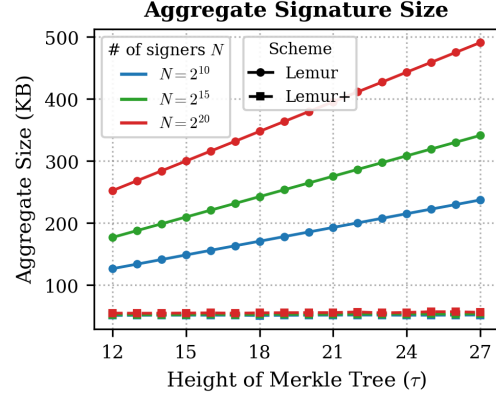


Figure 5: Aggregate signature size of Lemur and Lemur+ under Rice encoding across different τ and N .

Table 4 reports representative end-to-end runtimes for signature aggregation and aggregate verification in Lemur and Lemur+. For Lemur+, we additionally report the runtime of the proof layer in brackets. Overall, the results show that incorporating the proof mechanism introduces only modest overhead relative to the underlying aggregation and verification procedures. Here, to reduce aggregator memory, both implementations process inputs in streaming batches of 64, avoiding memory growth linear in N . Smaller batches can further reduce peak memory, but increase aggregation time.

An important implementation consideration is the arithmetic supported by the current LaBRADOR backend. Lemur+ uses the common modulus $q = 2^{32} - 99$, which is fixed by the current backend. Unlike the modulus used by Lemur’s HVC implementation, this modulus does not support a native degree-256 NTT. We therefore perform polynomial multiplication over auxiliary NTT-friendly primes and reconstruct the result modulo q using the CRT. This additional arithmetic introduces overhead specific to the current proof backend rather than to the proof construction itself. A lattice-based proof system supporting an NTT-friendly common modulus could further reduce this computational cost. We therefore view the reported runtimes as representative of a concrete and deployable implementation, while noting that they leave room for additional optimization at the proof-backend level.

Other costs. Across all evaluated parameter sets, individual signature takes 7.48 to 26.80 milliseconds. Key generation runtime spans from 0.37 seconds to 6.79 hours. Importantly, this computation is performed only once for each long-lived signing key and can be executed entirely offline. It therefore does not affect the latency of online signing or verification. A more significant practical concern is signer-side storage. Explicitly storing the full Merkle tree is intractable for large τ , as materializing 2^τ nodes would demand multi-GB or TB storage allocations. Lemur+ mitigates this by using BDS traversal method, maintaining active tree hash stacks and caching only

τ	N	Signature Aggregation		Aggregate Verification	
		Lemur	Lemur+ [Proof]	Lemur	Lemur+ [Proof]
12	2 ¹³	3.71	3.26 [0.07]	0.16	0.27 [0.06]
12	2 ¹⁵	14.71	16.35 [0.08]	0.65	0.73 [0.07]
16	2 ¹³	4.50	3.98 [0.08]	0.15	0.27 [0.07]
16	2 ¹⁵	18.12	20.36 [0.10]	0.63	0.75 [0.08]
20	2 ¹³	5.55	4.83 [0.10]	0.16	0.29 [0.08]
20	2 ¹⁵	21.56	24.11 [0.12]	0.63	0.79 [0.10]

Table 4: Representative end-to-end runtimes (seconds). Values in brackets [·] indicate the proof-layer runtime, which is included in the total Lemur+ runtime.

$O(\tau)$ authentication path nodes in memory. Consequently, the persistent state serialized to non-volatile storage stays strictly under 299 KB across all profiles. Complete results for all parameter sets are provided with the artifact.

References

- [1] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. Aggregating falcon signatures with labrador. In *CRYPTO 2024*, LNCS, pages 71–106. Springer, 2024.
- [2] Martin R. Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *J. Math. Cryptol.*, 9(3):169–203, 2015.
- [3] Rachid El Bansarkhani and Jan Sturm. An efficient lattice-based multisignature scheme with applications to bitcoins. In *CANS 2016*, volume 10052 of LNCS, pages 140–155, 2016.
- [4] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *CCS 2006*, pages 390–399. ACM, 2006.
- [5] Ward Beullens and Gregor Seiler. LaBRADOR: Compact proofs for R1CS from Module-SIS. In *CRYPTO 2023*, LNCS, pages 518–548. Springer, 2023.
- [6] Madalina Bolboceanu, Jonathan Bootle, Vadim Lyubashevsky, Antonio Merino-Gallardo, and Gregor Seiler. Orthus: Practical sublinear batch-verification of lattice relations from standard assumptions. To appear in *CRYPTO 2026*, ia.cr/2026/398.
- [7] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. Musig-l: Lattice-based multi-signature with single-round online phase. In *CRYPTO 2022*, volume 13508 of LNCS, pages 276–305. Springer, 2022.
- [8] Zvika Brakerski, Maya Farber Brodsky, Omer Paneth, and Tomer Solomon. Mergeable snargs for trapdoor languages and their applications. *IACR Cryptol. ePrint Arch.*, 2026:678, 2026. URL: <https://eprint.iacr.org/2026/678>.
- [9] Johannes Buchmann, Erik Dahmen, and Michael Schneider. Merkle tree traversal revisited. In *Post-Quantum Cryptography (PQCrypto) 2008*, volume 5299 of LNCS, pages 63–78. Springer, 2008.
- [10] Yanbo Chen. Dualms: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In *Crypto 2023*, pages 716–747. Springer, 2023.
- [11] Valerio Cini, Russell WF Lai, and Giulio Malavolta. Lattice-based succinct arguments from vanishing polynomials. In *Annual International Cryptology Conference*, pages 72–105. Springer, 2023.
- [12] Lalita Devadas, Rishab Goyal, Yael Kalai, and Vinod Vaikuntanathan. Rate-1 non-interactive arguments for batch-np and applications. In *FOCS 2022*, pages 1057–1068. IEEE, 2022.
- [13] Nico Döttling, Giulio Malavolta, and Omer Paneth. Batch arguments with optimal communication. *IACR Cryptol. ePrint Arch.*, 2026:3, 2026. URL: <https://eprint.iacr.org/2026/003>.
- [14] Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. Hash-based multi-signatures for post-quantum ethereum. *CIC 2025*, 2(1):13, 2025.
- [15] Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. Technical note: Leansig for post-quantum ethereum. *IACR Cryptol. ePrint Arch.*, 2025:1332, 2025.
- [16] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. Chipmunk: Better synchronized multi-signatures from lattices. In *CCS 2023*, pages 386–400. ACM, 2023.
- [17] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. Squirrel: Efficient synchronized multi-signatures from lattices. In *CCS 2022*, pages 1109–1123. ACM, 2022.
- [18] Masayuki Fukumitsu and Shingo Hasegawa. A lattice-based provably secure multisignature scheme in quantum random oracle model. In *ProvSec 2020*, volume 12505 of LNCS, pages 45–64. Springer, 2020.
- [19] Calvin Abou Haidar, Dipayan Das, Anja Lehmann, Cavit Özbay, and Octavio Perez-Kempner. Privacy-preserving multi-signatures: Generic techniques and constructions without pairings. In *PKC 2025*, volume 15675 of LNCS, pages 66–98. Springer, 2025.

- [20] Dennis Hofheinz, Michael Reichle, and Benedikt Wagner. Hint-free multi-signatures. *IACR Cryptol. ePrint Arch.*, 2026:723, 2026.
- [21] Kazuharu Itakura and Katsuhiko Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, (71):1–8, 1983.
- [22] Yini Lin, Muhammed F. Esgin, Amin Sakzad, Ron Steinfield, and Markku-Juhani O. Saarinen. Lemur: Scalable post-quantum synchronized multi-signatures. To appear at CCS 2026, ia.cr/2026/1161.
- [23] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. The lazer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. In *CCS 2024*, pages 3125–3137. ACM, 2024.
- [24] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: extended abstract. In *CCS 2001*, pages 245–254. ACM, 2001.
- [25] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. Technical report, Technical report, National Institute of Standards and Technology, 2020.
- [26] Arda Saygan, Muhammed Said Gündogan, Atakan Arslan, and Mehmet Emin Gönen. HAPPIER: hash-based, aggregatable, practical post-quantum signatures implemented efficiently with risc0. In *Lightweight Cryptography for Security and Privacy - 6th International Workshop, LightSec 2025, Istanbul, Türkiye, September 1-2, 2025, Revised Selected Papers*, LNCS, pages 3–22. Springer, 2025.

A Concrete Parameter Setting

We provide a concrete parameter setting in Table 5.

B Related Work

Post-quantum non-interactive multi-signatures have attracted significant attention in recent years. Existing constructions can be broadly categorized into two directions: generic SNARK-based approaches and specialized synchronized schemes.

Generic SNARK-Based Constructions. A line of work combines hash-based signatures with SNARK proof system to obtain compact aggregate signatures. Representative schemes include DKKW25 [14] and its successor LeanSig [15] and

HAPPIER [26]. These constructions naturally support aggregation of signatures on distinct messages and benefit from the fast verification time. However, their aggregate signature size is large, typically ranging from 2 to 3 MB.

Another line of work combines lattice-based signatures with lattice-compatible proof systems. Representative examples include the LaBRADOR-based [5] construction of Aardal et al. [1], LaZer [23], and the recent proposal Orthus [6]. All these constructions aggregates the NIST-standardized signature scheme Falcon [25], achieving significantly smaller aggregate signatures than previous hash-based approaches. Nevertheless, verification complexity is dominated by the underlying proof system and scales with the size of the proof system. As a result, existing implementations incur substantially higher verification time and also RAM requirement.

Specialized Synchronized Constructions. Rather than relying on generic proof systems, synchronized schemes aim to aggregate signatures on the same message during the same time period. This line includes Squirrel [17], further improved by Chipmunk [16] and Lemur [22]. By exploiting the homomorphic property of the underlying building blocks, these schemes avoids the computational overhead of general-purpose proof systems. Consequently, verification after public-key aggregation is independent of the number of signers, leading to significantly lower verification time.

C Details of KOTS

Gaussian Distributions. Let $\Sigma \in \mathbb{R}^{n \times n}$ be positive definite and $\mathbf{c} \in \mathbb{R}^n$. Define the (unnormalized) elliptical Gaussian function $\rho_{\mathbf{c}, \sqrt{\Sigma}} : \mathbb{R}^n \rightarrow (0, 1]$ as

$$\rho_{\mathbf{c}, \sqrt{\Sigma}}(\mathbf{x}) := \exp\left(-\pi(\mathbf{x} - \mathbf{c})^\top \Sigma^{-1}(\mathbf{x} - \mathbf{c})\right).$$

Let $\Lambda \subseteq \mathbb{R}^n$ be a lattice and let $\mathbf{v} \in \mathbb{R}^n$. The discrete Gaussian distribution over the coset $\mathbf{v} + \Lambda$ is defined by

$$\mathcal{D}_{\mathbf{v} + \Lambda, \mathbf{c}, \sqrt{\Sigma}}(\mathbf{x}) := \frac{\rho_{\mathbf{c}, \sqrt{\Sigma}}(\mathbf{x})}{\rho_{\mathbf{c}, \sqrt{\Sigma}}(\mathbf{v} + \Lambda)},$$

where $\mathbf{x} \in \mathbf{v} + \Lambda$ and $\rho_{\mathbf{c}, \sqrt{\Sigma}}(\mathbf{v} + \Lambda) := \sum_{\mathbf{y} \in \mathbf{v} + \Lambda} \rho_{\mathbf{c}, \sqrt{\Sigma}}(\mathbf{y})$. When $\mathbf{c} = \mathbf{0}$, we omit $\mathbf{c}$ from subscripts. When $\Sigma = \alpha^2 \mathbf{I}_n$, we have $\sqrt{\Sigma} = \alpha \mathbf{I}_n$, and we may write the distribution as $\mathcal{D}_{\mathbf{v} + \Lambda, \mathbf{c}, \alpha}$. The scalar $\alpha > 0$ is called the width parameter.

Definition C.1 (Dual Hint-MLWE). Let $q, d, m, n, k \in \mathbb{N}^+$ such that $k > 1$ and χ, C be distributions over R . Let $\mathbf{A} = [\mathbf{I}_n; \mathbf{A}_2] \in R_q^{m \times n}$ for $\mathbf{A}_2 \leftarrow R_q^{(m-n) \times n}$, $\mathbf{S} \leftarrow \chi^{k \times m}$, and $\mathbf{h} = [1, \mathbf{h}'] \in R^k$ for $\mathbf{h}' \leftarrow C^{k-1}$. The Dual Hint-MLWE problem is to distinguish between $D_0 := (\mathbf{A}, \mathbf{S}\mathbf{A}, \mathbf{h}, \mathbf{h}^\top \mathbf{S})$ and $D_1 := (\mathbf{A}, (\mathbf{S} + \mathbf{U})\mathbf{A}, \mathbf{h}, \mathbf{h}^\top \mathbf{S})$, where $\mathbf{U} \in R_q^{k \times m}$ with each column $\mathbf{u}_i \leftarrow \ker(\phi_{\mathbf{h}, q})$ sampled independently. The advantage

τ	N	KOTS Parameters			HVC Parameters						Size (KB)		
		n	m	β_σ	ω	η	κ	$\beta_{P/S, L_2}^{\text{honest}}$	$\beta_{U, L_2}^{\text{honest}}$	$\beta_{\text{agg}, L_2}^{\text{ext}}$	σ_{kots}	π	total agg.
12	2^{10}	4	9	6,310,455	5	813	3	52,329,521	93,609,892	545,836,728	6.75	49.34	56.11
12	2^{13}	5	10	17,898,519	5	813	3	148,082,806	296,165,612	1,567,455,898	8.12	44.22	52.36
12	2^{15}	5	10	35,797,038	4	42	5	17,753,378	39,697,760	191,214,675	8.44	48.34	56.80
12	2^{16}	5	10	50,624,657	4	42	5	25,107,073	56,141,120	270,414,542	8.44	44.62	53.07
12	2^{17}	5	10	71,594,076	5	128	4	108,224,617	216,449,234	1,145,475,322	8.75	48.98	57.75
12	2^{20}	6	11	203,007,545	5	42	5	112,807,440	247,148,719	1,210,901,901	9.97	50.00	59.99
16	2^{10}	4	9	6,310,455	5	813	3	52,631,056	94,149,294	624,527,106	6.75	49.11	55.88
16	2^{13}	5	10	17,898,519	5	813	3	148,918,565	297,837,129	1,787,558,044	8.12	44.63	52.77
16	2^{15}	5	10	35,797,038	4	42	5	17,848,975	39,911,520	217,154,450	8.44	49.18	57.63
16	2^{16}	5	10	50,624,657	4	42	5	25,242,238	56,443,360	307,087,886	8.44	44.82	53.27
16	2^{17}	5	10	71,594,076	5	128	4	108,825,386	217,650,772	1,306,029,398	8.75	50.13	58.90
16	2^{20}	6	11	203,007,545	5	42	5	113,412,400	248,474,120	1,376,275,606	9.97	50.51	60.50
20	2^{10}	4	9	6,310,455	5	813	3	52,868,020	94,573,189	694,993,391	6.75	50.13	56.90
20	2^{13}	5	10	17,898,519	5	813	3	149,578,149	299,156,298	1,985,032,028	8.12	44.78	52.93
20	2^{15}	5	10	35,797,038	4	42	5	17,925,037	40,081,600	240,497,476	8.44	49.87	58.33
20	2^{16}	5	10	50,624,657	5	128	4	77,286,953	154,573,906	1,025,423,974	8.44	44.84	53.30
20	2^{17}	5	10	71,594,076	5	128	4	109,300,291	218,600,582	1,450,333,461	8.75	50.40	59.17
20	2^{20}	6	11	203,007,545	5	42	5	113,893,120	249,527,324	1,525,224,132	9.97	51.13	61.12
24	2^{10}	4	9	6,310,455	5	813	3	53,063,095	94,922,149	759,391,311	6.75	50.15	56.92
24	2^{13}	5	10	17,898,519	4	42	5	8,994,074	20,111,360	130,957,042	8.12	44.95	53.09
24	2^{15}	5	10	35,797,038	4	42	5	17,988,148	40,222,720	261,919,719	8.44	50.68	59.14
24	2^{16}	5	10	50,624,657	5	128	4	77,564,440	155,128,880	1,118,823,294	8.44	45.26	53.72
24	2^{17}	5	10	71,594,076	5	128	4	109,692,623	219,385,245	1,582,590,051	8.75	51.02	59.79
24	2^{20}	6	11	203,007,545	5	42	5	114,291,840	250,400,876	1,661,728,114	9.97	51.67	61.65
26	2^{10}	4	9	6,310,455	5	813	3	53,149,044	95,075,900	789,793,855	6.75	50.45	57.22
26	2^{13}	5	10	17,898,519	4	42	5	9,008,027	20,142,560	136,022,358	8.12	44.65	52.80
26	2^{15}	5	10	35,797,038	4	42	5	18,016,054	40,285,120	272,040,628	8.44	49.36	57.82
26	2^{16}	5	10	50,624,657	5	128	4	77,686,941	155,373,882	1,162,952,014	8.44	45.57	54.03
26	2^{17}	5	10	71,594,076	5	128	4	109,865,927	219,731,853	1,645,081,018	8.75	51.35	60.12
26	2^{20}	7	12	203,471,040	5	42	5	114,496,320	270,947,746	1,738,819,883	10.88	51.97	62.86
27	2^{10}	4	9	6,310,455	5	813	3	53,189,633	95,148,507	804,599,237	6.75	50.61	57.38
27	2^{13}	5	10	17,898,519	4	42	5	9,014,682	20,157,440	138,491,943	8.12	44.78	52.92
27	2^{15}	5	10	35,797,038	4	42	5	18,029,291	40,314,720	276,983,150	8.44	51.21	59.66
27	2^{16}	5	10	50,624,657	5	128	4	77,744,900	155,489,799	1,184,418,432	8.44	45.65	54.10
27	2^{17}	5	10	71,594,076	5	128	4	109,947,857	219,895,713	1,675,486,317	8.75	51.42	60.19
27	2^{20}	6	11	203,007,545	5	42	5	114,551,680	250,970,157	1,757,657,784	9.97	52.01	62.00

Table 5: Lemur+ Parameter Sets ($k = 4$, $d = 256$, $q = 2^{32} - 99$).

of a PPT adversary $\mathcal{A}$ is defined as:

$$\text{Adv}_{q,m,n,k,\chi,C}^{\text{DualHintMLWE}}(\mathcal{A}) := |\Pr[\mathcal{A}(D_0) = 1] - \Pr[\mathcal{A}(D_1) = 1]|.$$

If $\chi = \mathcal{D}_{R,\alpha}$, we denote the problem by $\text{DualHintMLWE}_{q,m,n,k,\alpha,C}$. The Dual Hint-MLWE states that the advantage of $\mathcal{A}$ is negligible.

Figure 6 shows the construction of KOTS in Lemur [22]. Theorem C.1 states the security of the KOTS construction.

Theorem C.1 ([22, Theorem 4.1]). *Let $\lambda, t_1, t_2, p, q, d, m, n, k, \beta_z, \beta_\sigma, \alpha_H, \alpha_w, N \in \mathbb{N}^+$, where t_1, t_2, d are powers of 2 with $d > t_1 > 1$ and $d > t_2 > 1$, $p \equiv 2t_1 + 1 \pmod{4t_1}$ is prime, $q \equiv 2t_2 + 1 \pmod{4t_2}$ is prime, $(\frac{1+\epsilon_3}{1-\epsilon_3})^{2m} p^{-dm/t_1} < 2^{-\lambda}$, and $k \geq 4$. Let $\alpha, \epsilon_{\text{cor}}, \epsilon_{\text{homK}}, \epsilon_1, \epsilon_3 \in \mathbb{R}$, such that $\alpha \geq \sqrt{2\eta_{\epsilon_1}(\mathbb{Z})}$, $\alpha \geq p\sqrt{2\alpha_H \ln(2(k-1)d(1+1/\epsilon_3))}/\pi$, $\beta_z \geq$*

$\text{pp} \leftarrow \text{Setup}(1^\lambda)$ 1. $\mathbf{A}_2 \leftarrow R_q^{(m-n) \times n}$ 2. return $\text{pp} = \mathbf{A} = [\mathbf{I}_n; \mathbf{A}_2]$ $(\text{osk}, \text{opk}) \leftarrow \text{KGen}(\text{pp})$ 1. $\mathbf{S} \leftarrow \mathcal{D}_{R^k \times m, \alpha}$ 2. $\mathbf{T} = \mathbf{S}\mathbf{A} \pmod{q} \in R_q^{k \times n}$ 3. return $(\text{osk} = \mathbf{S}, \text{opk} = \mathbf{T})$ $\sigma \leftarrow \text{Sign}(\text{pp}, \text{osk}, \mu)$ 1. $\mathbf{h}' = \text{H}(\mu) \in (\mathcal{T}_{\alpha_H})^{k-1}$ 2. $\mathbf{h} = [1, \mathbf{h}'] \in R^k$ 3. $\mathbf{z}^\top = \mathbf{h}^\top \mathbf{S}$ 4. return $\sigma = \mathbf{z} \in R^m$	 $b \leftarrow \text{iVrfy}(\text{pp}, \text{opk}, \mu, \sigma)$ 1. return $\text{Vrfy}(\text{pp}, \text{opk}, \mu, \sigma, \beta_z)$ $b \leftarrow \text{sVrfy}(\text{pp}, \text{opk}, \mu, \sigma)$ 1. return $\text{Vrfy}(\text{pp}, \text{opk}, \mu, \sigma, \beta_\sigma)$ $b \leftarrow \text{wVrfy}(\text{pp}, \text{opk}, \mu, \sigma)$ 1. return $\text{Vrfy}(\text{pp}, \text{opk}, \mu, \sigma, 2\beta_\sigma)$ $b \leftarrow \text{Vrfy}(\text{pp}, \text{opk}, \mu, \sigma, \beta)$ 1. Parse $\text{pp} = \mathbf{A}$, $\text{opk} = \mathbf{T}$ and $\sigma = \mathbf{z}$ 2. $\mathbf{h}' = \text{H}(\mu)$, and $\mathbf{h} = [1, \mathbf{h}']$ 3. if $\ \mathbf{z}\ _\infty > \beta$ or $\mathbf{z}^\top \mathbf{A} \neq \mathbf{h}^\top \mathbf{T} \pmod{q}$, re- turn 0, else return 1
--	--

Figure 6: KOTS Construction in Lemur.

$6\alpha\sqrt{1+(k-1)\alpha_H}$, and $\epsilon_{\text{cor}} \geq md \cdot (2^{-162} + 2(k-1)\alpha_H\epsilon_1)$, $\beta_\sigma \geq \beta_z \cdot \sqrt{2\alpha_w N \cdot \ln\left(\frac{2md}{\epsilon_{\text{homK}}}\right)}$. Let $H: \mathcal{M} \rightarrow (\mathcal{T}_{\alpha_H})^{k-1}$ be a random oracle with $|\mathcal{T}_{\alpha_H}|^{k-3} \geq 2^{2\lambda}$. Let $W' = \{w_0 - w_1 \mid w_0, w_1 \in \mathcal{T}_{\alpha_w} \wedge w_0 \neq w_1\}$. If the $\text{DualHintMLWE}_{q,m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$ and $\text{MSIS}_{q,n,m,2\beta_\sigma+2\alpha_w\beta_z}^\infty$ problems are hard, and any $w^* \in W'$ is a unit in R_q , then the construction KOTS in Figure 6 is an ϵ_{cor} -individually correct, $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\text{homK}})$ -probabilistically homomorphic, robustly homomorphic KOTS that is W' -multi-user existentially unforgeable under rerandomized keys.