Lemur: Scalable Post-Quantum Synchronized Multi-Signatures

Yini Lin
yini.lin@monash.edu
Monash University
Melbourne, Victoria, Australia

Muhammed F. Esgin
muhammed.esgin@monash.edu
Monash University
Melbourne, Victoria, Australia

Amin Sakzad
amin.sakzad@monash.edu
Monash University
Melbourne, Victoria, Australia

Ron Steinfeld
ron.steinfeld@monash.edu
Monash University
Melbourne, Victoria, Australia

Markku-Juhani O. Saarinen
markku-juhani.saarinen@tuni.fi
Tampere University
Tampere, Finland

Abstract

Synchronized multi-signatures allow for *non-interactive* aggregation of signatures generated within the same time step. This primitive is particularly well-suited for high-throughput blockchain protocols like Ethereum, where many distributed signers must validate the same block within a synchronized slot. In this work, we present Lemur, a post-quantum synchronized multi-signature from (module) lattices that improves upon the state-of-the-art in efficiency, scalability, and flexibility. Lemur follows the blueprint of Squirrel/Chipmunk (CCS 2022/2023) but introduces a fundamental redesign of the foundations of the overall framework. Our revisit of the framework is also motivated by the fact that our evaluation of Chipmunk's parameter sets, using the state-of-the-art lattice security estimation methods, suggests a substantially lower concrete security level (approximately 30 bits rather than the claimed 112 bits).

First, we revisit the underlying building block of key-homomorphic one-time signature (KOTS) and introduce a novel security reduction based on a new lattice problem: the *Dual Hint-MLWE* assumption, which may be of independent interest. We then provide a formal reduction from the standard Module-LWE problem to Dual Hint-MLWE, which overall enables us to base the security of Lemur on the standard Module-LWE and Module-SIS assumptions. By shifting from a statistical security argument to a computational one, our KOTS design enjoys much better compactness and scalability.

Second, we optimize the underlying homomorphic vector commitment (HVC) by transitioning from Ring-SIS to the Module-SIS setting and extending the commitment domain from vectors to matrices. This generalization reduces opening size and improves aggregation efficiency.

After rectifying the parameters of Chipmunk for a fair comparison, our results show that Lemur's KOTS size achieves up to an order of magnitude improvement over Chipmunk's KOTS. In particular, aggregating 1 million one-time signatures requires under 8 KB. For the total multi-signature size, Lemur demonstrates around 2× improvement over Chipmunk. To showcase our design, we provide a full-fledged Rust implementation. Our benchmarks demonstrate an aggregate signature size of 380 KB for 2^{20} signers. Stateful signing takes roughly 4.2 ms for a Merkle tree of height 20, while batch verification for an aggregate of 1024 signers completes in 15.0 ms ($\approx$ 14.6 μs per signer).

Keywords

Post-Quantum, Lattice, Multi-Signatures, Key-Homomorphic One-Time Signature, Homomorphic Vector Commitment

1 Introduction

Multi-signatures allow many signers to authenticate the same message by producing a single, compact aggregated signature [21, 29]. This primitive is a key building block in distributed systems, especially in proof-of-stake blockchains such as Ethereum and Dfinity, where large validator sets must repeatedly reach agreement on the validity of proposed blocks. By compressing many signatures into one short object, multi-signatures substantially reduce communication, storage, and verification costs in the consensus layer.

For such applications, *non-interactive* aggregation is crucial. In a public blockchain, validators may be online at different times, may not know the identities of others, and often communicate only indirectly through the network. Interactive protocols [4, 8, 11, 12, 19, 20, 22] are therefore a poor fit: they require additional rounds of communication, introduce coordination bottlenecks, and increase latency in already time-sensitive consensus pipelines. By contrast, *non-interactive* aggregation allows any untrusted aggregator to collect independently generated signatures and combine them publicly, without requiring any further coordination among the signers themselves.

This functionality is already central to real-world blockchain deployments. Today, major proof-of-stake systems such as Ethereum and Dfinity achieve non-interactive aggregation using the BLS signature scheme [7]. While highly compact, the security of BLS rests on discrete-logarithm assumptions in pairing-friendly groups and is therefore vulnerable to quantum attacks. As a result, replacing BLS with a post-quantum alternative that preserves public, non-interactive aggregation has become a pressing challenge for blockchain consensus.

Despite the practical importance of this problem, designing such a replacement is difficult. One possible direction is hash-based cryptography. However, hash-based signatures do not naturally support compact aggregation, since hash functions lack the algebraic structure that enables compact aggregation. Existing proposals [14] therefore achieve aggregation only by invoking succinct arguments, such as SNARK-like proof systems, to prove knowledge of many valid signatures. While conceptually appealing, this approach introduces heavy proof machinery and substantial computational

N (signers)	Chipmunk (KB)		Lemur (KB)	
	One-Time Sig	Multi-Sig	One-Time Sig	Multi-Sig
2^{10}	26	237	5.6	185
2^{15}	>90	>411	6.9	265
2^{20}	>110	>728	7.8	380

Table 1: Aggregated signature sizes at 128-bit security (τ=20). Chipmunk: theoretical, from their scripts. Lemur: measured in implementation for N=2^{10}; Rice-encoded on the corresponding cells of Table 6 for $N \in \{2^{15}, 2^{20}\}$.

Operation	$N = 2^{10}$	$N = 2^{15}$	$N = 2^{20}$
Signing (BDS08)	4.2 ms	4.2 ms	4.2 ms
Signing (Tree-in-memory)	1.6 ms	1.6 ms	1.6 ms
Aggregation	1.16 s	$\approx$ 11.2 s†	$\approx$ 6.0 min†
Batch verification	15.0 ms	$\approx$ 442 ms†	$\approx$ 13.4 s†
Agg. signature size	185.5 KB	264.5 KB‡	379.6 KB‡

†Aggregation values for $N > 2^{10}$ are linear extrapolations from the bench `-fast` $N = 2^{13}$ run (2.79 s), the largest aggregation cell measured with the implemented constants; the large-N batch-verification entries are measured with `bench_verify -zero-fixture`. ‡Predicted Rice-encoded sizes on the $\tau = 20$ rows of Table 6; the artifact only has the $N = 2^{10}$ cell.

Table 2: Representative Rust implementation performance using (τ=20, N=2^{10}) parameter setting. All measurements were timed on the laptop system described in Section 7.2.

overhead, which makes it difficult to deploy in latency-sensitive consensus settings.

A promising alternative to the hash-based approach is to build non-interactive aggregation from lattices. In this direction, Squirrel [18] and its successor Chipmunk [16] constructed lattice-based non-interactive multi-signatures in the *synchronized* setting, where a common message is signed at a particular time step. This setting is particularly natural for proof-of-stake blockchains: validators are already synchronized by the protocol, sign the same candidate block within a common slot, and are expected to sign at most one message per time step. Thus, synchronized multi-signatures capture the needs of blockchain consensus well.

However, upon closer inspection, we find that Chipmunk [16] does not achieve its claimed concrete security level. In particular, its parameter selection relies simplified heuristic estimates of the Ring-SIS hardness. We reevaluate these parameters using the state-of-the-art Dilithium security estimation framework. Our analysis suggests that they correspond to substantially lower concrete security (approximately 30-bit instead of the claimed 112-bit). This discrepancy has important concrete implications: adjusting Chipmunk's parameters to meet a standard 128-bit security level increases their aggregate signature size by a factor of around 2×, significantly undermining its reported efficiency [16]. Moreover, Chipmunk's reliance on the Ring-SIS assumption limits its parameter flexibility compared to Module-SIS, further complicating the trade-off between security and signature compactness.

Taken together, the current landscape shows that post-quantum non-interactive aggregation is advancing rapidly, but existing approaches still leave substantial room for improvement. The central challenge is therefore to design post-quantum synchronized multi-signatures that achieve strong concrete security while maintaining scalable and bandwidth-efficient for blockchain-scale systems.

1.1 Our Contributions

In this work, we present Lemur, a concretely efficient, non-interactive lattice-based multi-signature scheme in the synchronized setting. Lemur builds upon the non-interactive framework in Chipmunk [16] but redesigns its underlying building blocks to overcome bandwidth and scalability bottlenecks. Specifically, by shifting away from the statistical security arguments that forced previous schemes to use large parameters, Lemur achieves significantly smaller aggregated signature sizes for large sets of signers while maintaining security against rogue-key attacks in the random oracle model.

Our improvements stem from two major technical contributions regarding the underlying cryptographic primitives. First, we revisit the key-homomorphic one-time signature (KOTS) used in Chipmunk (arising from [6, 26]). Unlike the KOTS in Chipmunk, which relies on a statistical security argument, our construction introduces and relies upon a computational assumption: the Dual Hint-MLWE assumption. To establish confidence in this new assumption, we provide a formal, rigorous reduction from the standard Module-LWE problem to our Dual Hint-MLWE assumption. By transitioning to a computational argument, we significantly reduce the state-of-the-art key-homomorphic one-time signature sizes while ultimately reducing security to the standard Module-LWE. Concretely, we can aggregate 1 million KOTS signatures into 7.8 KB while Chipmunk requires more than 110 KB (see Table 1).

In addition to being key-homomorphic, our KOTS design also provides *unique* signatures. As shown in [10], a unique one-time signature can be extended to a *many-time* verifiable random function (VRF) using a Merkle tree. Furthermore, the signing procedure is completely linear, which facilitates masking against side-channel attacks when needed.

Second, we extend the underlying homomorphic vector commitment (HVC) scheme used to compress the KOTS public keys. We transition the underlying hardness assumption from Ring-SIS to Module-SIS. Furthermore, we extend the commitment domain from vectors to matrices. This generalization allows for more compact aggregated openings, reducing the bandwidth required to support large signer sets without affecting correctness or aggregation success probability.

Combined, these theoretical optimizations allow Lemur to outperform prior synchronized lattice-based multi-signatures [16, 18]. Our experimental results demonstrate that Lemur achieves significantly smaller aggregate signatures while maintaining scalability in settings involving thousands and even millions of signers. We refer to Tables 1 and 6 for a size comparison and Table 2 for the concrete Lemur runtimes. Since Chipmunk's implemented parameters are at a substantially lower security level (around 30 bits) than ours (128 bits), a direct runtime comparison would not be meaningful.

Implementation and Benchmarks. To validate Lemur's deployability, we provide two interoperable open-source reference implementations: an optimized **Rust** crate and a readable **Python** prototype, both covering the full pipeline (KOTS, HVC, aggregation, and bit-packed encoding) and cross-validated against shared

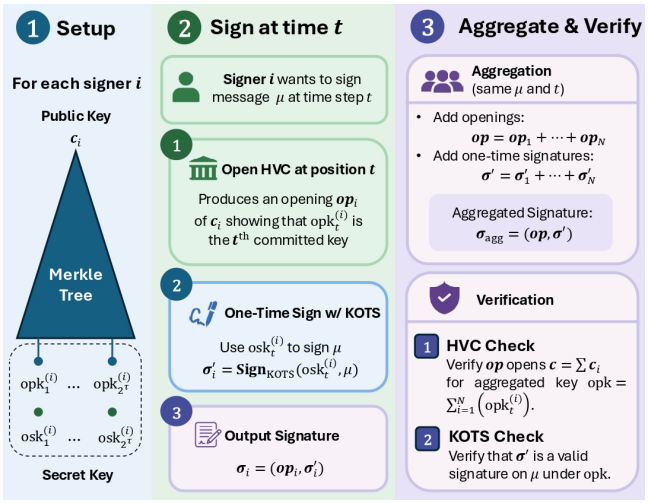

Figure 1: High Level Idea of Lemur / Chipmunk / Squirrel.

byte-level test vectors. Our default signer uses the BDS08 Merkle traversal algorithm [9], replacing a multi-GB materialized tree with a roughly 110 KB authentication-path cache against a 32-byte master seed; this brings stateful signing to roughly 4 ms on commodity hardware. At $\lambda = 128$ and $\tau = 20$, batch verification of an aggregated signature runs in 15 ms for $N = 1024$, and the aggregated signature itself is 185.5 KB at $N = 2^{10}$ (see Table 6 for larger N). The artifact is released at https://github.com/lemur-sig/lemur; further implementation notes appear in Section 7.2.

Open question. Our work leaves an important open question for future work. As discussed above, our KOTS achieves highly compact signatures, requiring under 8 KB for 1 million signers. However, the opening of HVC, needed for many-time signing, remains relatively large (refer to Table 6). An interesting question here is if we can find an alternative way to extend from one-time signing to many-time signing via other means, for example a solution that exploits the higher-level application layer.

1.2 Technical Overview

At a high level, Lemur follows the blueprint of Chipmunk [16]. We first recall the common high-level framework, illustrated in Figure 1, and then highlight the main technical improvements introduced by Lemur.

Chipmunk Recap. In Chipmunk, signatures are aggregated only for the same message and time step, and each signer supports at most 2^τ signatures before key refresh. As shown in Figure 1, during Setup (left), each signer generates 2^τ one-time key pairs and commits to the vector of one-time public keys using a homomorphic vector commitment (HVC), yielding a compact public key. To sign a message μ at time t (middle), the signer opens the commitment at position t and produces a one-time signature under the corresponding secret key, forming an individual signature consisting of the opening and the one-time signature. Aggregation and verification (right) exploit the homomorphic properties of both primitives: openings and one-time signatures are combined via sums, producing a single aggregate signature that verifies for the same message and time step under the aggregated public key. To achieve security against rogue-key attacks, aggregation is performed using weighted sums, where the weights are derived from a random oracle. We omit these weights here for simplicity.

Key-Homomorphic One-Time Signatures. Chipmunk's KOTS closely follows the line of work of Lyubashevsky-Micciancio [26] and Boneh-Kim [6], and is conceptually very simple. However, earlier unforgeability proofs rely on statistical arguments. As a result, the parameters must be chosen so that the secret key retains sufficiently large entropy even after revealing the public key and one signature. Our Lemur design avoids this bottleneck by replacing the statistical step with a computational argument via our new computational assumption: Dual Hint-MLWE.

At a high level, Lemur's KOTS works as follows. Each signer samples their secret key as a short (i.e., small-norm) matrix $\mathbf{S}$ from a discrete Gaussian distribution and publish $\mathbf{T} = \mathbf{SA} \pmod{q'}$ as the public key, where q' a system modulus parameter. To sign a message μ, the signer first maps it to a short vector $\mathbf{h}$ via a random oracle, and outputs the signature $\mathbf{z}^\top = \mathbf{h}^\top\mathbf{S}$. Verification then consists of two checks: a norm bound on $\mathbf{z}$, and the algebraic condition $\mathbf{z}^\top\mathbf{A} = \mathbf{h}^\top\mathbf{T} \pmod{q'}$. This preserves homomorphism under aggregation.

The main challenge is security. Revealing a linear image of the secret key (i.e., $\mathbf{h}^\top\mathbf{S}$) appears, at first glance, to leak substantial information about $\mathbf{S}$. Lemur relies on the *Dual Hint-MLWE* assumption to control this leakage. Informally, this assumption states that even given the hint $(\mathbf{h}, \mathbf{h}^\top\mathbf{S})$, the MLWE instance $(\mathbf{A}, \mathbf{SA})$ is computationally indistinguishable from a rerandomized instance $(\mathbf{A}, (\mathbf{S}+\mathbf{U})\mathbf{A})$, where each column of $\mathbf{U}$ is sampled uniformly random from the right kernel of $\mathbf{h}^\top \pmod{q'}$. We formally prove the unforgeability of Lemur's KOTS under the Dual Hint-MLWE and MSIS assumption in Lemma 4.4.

Security of Dual Hint-MLWE. Kim et al. [24] introduced the Hint-MLWE assumption, which states that an MLWE sample $(\mathbf{A}, \mathbf{As})$ remains indistinguishable from uniform even when given a bounded number of noisy linear hints of the form $\mathbf{z}_i = c_i \cdot \mathbf{s} + \mathbf{y}_i$. Unfortunately, this assumption is not directly applicable to Lemur's KOTS unforgeability proof for two reasons. First, Lemur's KOTS reveals *noise-free* linear images of the secret key, rather than perturbed hints. Second, the secret key appears on opposite sides of the public key and the signature computation: the public key is of the form $\mathbf{T} = \mathbf{SA}$, while the signature reveals $\mathbf{z}^\top = \mathbf{h}^\top\mathbf{S}$. This structural mismatch necessitates a new assumption.

We therefore introduce the *Dual Hint-MLWE* problem. In this setting, the adversary is asked to distinguish between two distributions: the real sample $(\mathbf{A}, \mathbf{SA}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$, and the "uniform" sample $(\mathbf{A}, (\mathbf{S}+\mathbf{U})\mathbf{A}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$ where $\mathbf{U}$ is a mask term uniformly chosen to stay in the right kernel of $\mathbf{h}^\top \pmod{q'}$. We prove that the Dual Hint-MLWE is at least as hard as the standard MLWE by a sequence of hybrids in Theorem 3.1. A key technical challenge in the proof is sampling from discrete Gaussians over *non-full-rank* lattice, a setting not directly covered by existing literature. We establish a suitable sampleability lemma (Lemma 2.11) and provide a rigorous analysis, which may be of independent interest.

1.3 Related Works

Lattice-based Paradigm. Aardal et. al [1] and Lazer [28] demonstrate a non-interactive method for aggregating Falcon-512 signatures [32] using the LaBRADOR proof system [5], achieving an aggregate signature size of around 73 KB for 1024 signers. Such LaBRADOR-based systems rely on complicated SNARK machinery and therefore are not easy to implement, while Lemur employs easy-to-implement building blocks only. The main performance advantage of Lemur against these LaBRADOR-based aggregation systems is the verification time. For 1024 signers, Lemur verification time is only 15 ms compared to roughly 400 ms via LaZer [28]. Lemur also supports public key aggregation and our verification time is only a couple of milliseconds (independent of the number of signers) under an aggregated public key. Additionally, Lemur requires a lot less RAM compared to these works where [28] states that about 25 GB RAM is needed for aggregating 100K signatures, compared to 110 KB in Lemur. We note that no concrete implementation is provided in [1].

Anadel et al. [3] proposed the first lattice-based synchronized aggregate signature scheme in the standard model, following the framework of Squirrel [18] and Chipmunk [16]. While their construction achieves a tight security reduction in the standard model, it is restricted to the certified-key model, making it susceptible to rogue-key attacks in more general settings. Furthermore, they do not provide a concrete implementation or empirical performance benchmarks to demonstrate the practical efficiency of their scheme.

Hash-Based Paradigm. Recent research into post-quantum Ethereum has produced a family of hash-based signature schemes that rely on the "aggregation-via-SNARK" paradigm. [14] introduces a unified framework for generalized XMSS variants that provides a rigorous security analysis in the standard model. However, they do not provide concrete benchmarks for signature aggregation. Their estimates suggest that the aggregate signature size using Plonky3 ranges from 2 MB to 3 MB. [23] explores the theoretical tradeoffs between signature size and verification efficiency by mapping incomparable encodings into hypercube structures. LeanSig [15] serves as bridge by instantiating the framework of [14] with the hypercube-based encodings of [23], specifically tailored for Ethereum's concensus layer. However, LeanSig does not provide concrete performance numbers for the aggregate signature. HAPPIER [33] focuses on practicality and scalability, demonstrating the first implementation of multi-level aggregation using standardized XMSS and the RISC Zero zkVM on standard laptop hardware. However, its aggregate signature size is still constrained by the SNARK paradigm, reaching 2.89 MB for 2^{16} signatures. In summary, while these hash-based works offer minimal security assumptions and standardized foundations, they are all ultimately constrained by the bandwidth and proving overhead of the SNARK-based aggregation required for large committees.

2 Preliminaries

This section introduces the notations, definitions, and important lemmas used throughout the paper. Finally, we summarize all notations used in the paper in Table 3.

Notations. For $n \in \mathbb{N}^+$, we identify $\mathbb{Z}_n$ with the representative set $\mathbb{Z} \cap (-n/2, n/2]$. Let $[n] = \{1, 2, \ldots, n\}$ and $[\![n]\!] = \{0, 1, \ldots, n-1\}$. For a bit string $s = (s_1, \ldots, s_\tau) \in \{0,1\}^\tau$, s_i denotes the i-th bit and $s_{<i} := (s_1, \ldots, s_{i-1})$ denotes the prefix of length $i-1$. For $i \in [\![2^\tau]\!]$, $\mathrm{bin}_\tau(i) \in \{0,1\}^\tau$ 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 any n-dimensional vector $\mathbf{a} = (a_1, \ldots, a_n)$, a_i is its i-th entry and $\mathbf{a}_{<i} := (a_1, \ldots, a_{i-1})$ is its prefix. We write $\mathbf{a}^\top$ for the transpose (treated as a row vector). For an $m \times n$ matrix $\mathbf{A}$, the entry in row i and column j is $a_{i,j}$ for $i \in [m]$, $j \in [n]$. Let $\mathbf{I}_n$ denote the $n \times n$ identity matrix. For two vectors of the same length $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$, define $\mathrm{zip}(\mathbf{a}, \mathbf{b}) = ((a_1, b_1), \ldots, (a_n, b_n))$. The operator $\mathrm{vec}(\mathbf{M})$ denotes the vectorization of matrix $\mathbf{M}$ by stacking its columns, and $\mathrm{vec}^{-1}(\cdot)$ denotes its inverse operation. We use $[\mathbf{A}; \mathbf{B}]$ to denote the vertical concatenation of matrices and $[\mathbf{A} \mid \mathbf{B}]$ for horizontal concatenation.

Let d be a power of 2 and q be an integer modulus. Denote $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q[X]/(X^d + 1)$. Multiplication between elements of $\mathcal{R}$ and $\mathcal{R}_q$ is performed modulo q by default. For a polynomial $f = \sum_{i=0}^{d-1} f_i X^i \in \mathcal{R}$, 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 $\mathbf{f} = (f_1, \ldots, f_n) \in \mathcal{R}^n$, we extend these norms as:

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

The spectral norm $\|\mathbf{A}\|_2$ of a matrix $\mathbf{A} \in \mathcal{R}^{m \times n}$ is defined as the largest singular value of the matrix $\sigma(\mathbf{A}) \in \mathbb{Z}^{md \times nd}$, where $\sigma(\cdot)$ maps each polynomial in $\mathcal{R}$ to its $d \times d$ negacyclic convolution matrix. Norms over $\mathcal{R}_q$ 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 $\mathcal{T}_{\alpha_w} = \{f \in \mathcal{R} : \|f\|_\infty = 1 \wedge \|f\|_1 = \alpha_w\}$.

Lemma 2.1. *For $a, b \in \mathcal{R}$, it holds that*

(1) $\|a\|_\infty \leq \|a\|_2 \leq \|a\|_1$,
(2) $\|a + b\|_\infty \leq \|a\|_\infty + \|b\|_\infty$,
(3) $\|a \cdot b\|_\infty \leq \|a\|_1 \cdot \|b\|_\infty$.

Lemma 2.2 ([27, Corollary 1.2]). *Let $d \geq t > 1$ be powers of 2 and let $p \equiv 2t + 1 \pmod{4t}$ be a prime. Then any $y \in \mathcal{R}_p$ satisfying $0 < \|y\|_\infty < \frac{1}{\sqrt{t}} \cdot p^{1/t}$ is invertible in $\mathcal{R}_p$.*

Lemma 2.3 ([16, Lemma 4]). *Let $d, \alpha_w, N, \beta \in \mathbb{N}^+$ with d a power of 2. Then for any $x_1, \ldots, x_N \in \mathcal{R}$ with $\|x_i\|_\infty \leq \beta$ for all $i \in [N]$, and for any scaling factor $\zeta \geq 1$, it holds that*

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

Definition 2.1 (Labeled Full Binary Tree). *For a depth $\tau \in \mathbb{N}^+$, we identify the nodes of a full binary tree with the set of bit strings $\mathcal{V} = \bigcup_{i=0}^{\tau} \{0,1\}^i$. The root is the empty string ε. For any internal node $v \in \{0,1\}^{<\tau}$, its left and right children are $v\|0$ and $v\|1$, respectively. The set of leaves is $\mathcal{L} = \{0,1\}^\tau$. A labeled full binary tree is defined*

by a function label : $\mathcal{V} \to \mathcal{X}$*, which assigns a value from a set* $\mathcal{X}$ *to each node in the tree.*

Definition 2.2 (*R*-module). *Let* $(R, +, \times)$ *be a commutative ring. We say that a set* M *is an* R*-module if* $(M, +)$ *is an abelian group, and there exists a scalar multiplication operation* $\cdot$ *such that for all* $r, s \in R$ *and* $x, y \in M$*, it holds that*

(1) $1_R \cdot x = x$,
(2) $(r + s) \cdot x = r \cdot x + s \cdot x$,
(3) $r \cdot (x + y) = r \cdot x + r \cdot y$,
(4) $(r \times s) \cdot x = r \cdot (s \cdot x)$.

2.1 Probability Distribution

For a finite set S, let $x \leftarrow S$ denote sampling x uniformly at random from S. More generally, for a distribution D, $x \leftarrow D$ denotes sampling x according to D. Let D_1 and D_2 be two distributions over a countable set S. Their statistical distance is defined as $\Delta(D_1, D_2) := \frac{1}{2} \sum_{x \in S} |D_1(x) - D_2(x)|$.

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

$$\rho_{\mathbf{c},\sqrt{\Sigma}} : \mathbb{R}^n \to (0, 1], \quad \rho_{\mathbf{c},\sqrt{\Sigma}}(\mathbf{x}) := \exp\Big(-\pi\,(\mathbf{x} - \mathbf{c})^\top \Sigma^{-1} (\mathbf{x} - \mathbf{c})\Big).$$

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 $\rho_{\mathbf{c},\sqrt{\Sigma}}(\mathbf{v} + \Lambda) := \sum_{\mathbf{y} \in \mathbf{v}+\Lambda} \rho_{\mathbf{c},\sqrt{\Sigma}}(\mathbf{y})$. Note that coset sampling from $\mathcal{D}_{\mathbf{v}+\Lambda,\mathbf{0},\sqrt{\Sigma}}$ is equivalent to sampling from $\mathcal{D}_{\Lambda,-\mathbf{v},\sqrt{\Sigma}}$ and then adding the shift $\mathbf{v}$. 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 $\mathcal{D}_{\mathbf{v}+\Lambda,\mathbf{c},\alpha}$ (or $\mathcal{D}_{\Lambda,\mathbf{c},\alpha}$ when $\mathbf{v} = \mathbf{0}$). The scalar $\alpha > 0$ is called the width parameter.

Let r, n be positive integers with $1 \le r \le n$, and let $\mathbf{B} \in \mathbb{R}^{n \times r}$ have $\mathbb{R}$-rank r. We recall that $\Lambda := \mathbf{B} \cdot \mathbb{Z}^r$ is called a rank r lattice in $\mathbb{R}^n$ with column $\mathbb{Z}$-basis $\mathbf{B}$. If $r = n$, we say that Λ is a full-rank lattice.

Definition 2.3 (Smoothing Parameter [30, Def. 3.1]). *For any lattice* $\Lambda \subset \mathbb{R}^n$*, the smoothing parameter* $\eta_\varepsilon(\Lambda)$ *is the smallest* $s > 0$ *such that* $\rho_{1/s}(\Lambda^* \setminus \{\mathbf{0}\}) \le \varepsilon$*, where* $\Lambda^* := \{\mathbf{x} \in \mathrm{Span}_{\mathbb{R}}(\Lambda) : \forall \mathbf{y} \in \Lambda, \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}\}$ *is the dual lattice of* Λ.

We now recall several standard results used in the security analysis of lattice-based schemes.

Lemma 2.4 ([24, Lemma 3], [31, Theorem 3.1]). *Let* $k \in \mathbb{N}^+$ *and* $\varepsilon \in (0, \frac{1}{2})$ *such that* $\alpha \ge \sqrt{2}\eta_\varepsilon(\mathbb{Z})$*. If* $x_1, \ldots, x_k$ *are independent samples from* $\mathcal{D}_{\mathbb{Z},\alpha}$*, then*

$$\Delta\left(\sum_{i=1}^{k} x_i,\ \mathcal{D}_{\mathbb{Z},\sqrt{k}\alpha}\right) \le 2(k-1)\varepsilon.$$

Lemma 2.5 ([25, Lemma 4.4], [24, Lemma 1]). *For any* $\alpha \in \mathbb{R}_{>0}$ *and* $z \leftarrow \mathcal{D}_{\mathbb{Z},\alpha}$*, it holds that* $\Pr[|z| \ge 6\alpha] \le 2^{-162}$.

Lemma 2.6 ([31, Theorem 3.1]). *Let* $\Sigma_1, \Sigma_2 > \mathbf{0}$ *be positive definite matrices, with* $\Sigma = \Sigma_1 + \Sigma_2 > \mathbf{0}$*. Let* Λ_1, Λ_2 *be lattices such that* $\sqrt{\Sigma_1} \succeq \eta_\varepsilon(\Lambda_1)$ *for some* $\varepsilon \in (0, 1/2]$*. Let* $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{R}^n$*. Consider the following experiment:*

(1) Sample $\mathbf{x}_2 \leftarrow \mathcal{D}_{\Lambda_2+\mathbf{c}_2,\sqrt{\Sigma_2}}$.
(2) Sample $\mathbf{e} \leftarrow \mathcal{D}_{\Lambda_1+\mathbf{c}_1-\mathbf{x}_2,\sqrt{\Sigma_1}}$ *and set* $\mathbf{x}_1 = \mathbf{x}_2 + \mathbf{e}$.

The marginal distribution of $\mathbf{x}_1$ *is within statistical distance* 8ε *of* $\mathcal{D}_{\Lambda_1+\mathbf{c}_1,\sqrt{\Sigma}}$.

Lemma 2.7 ([31, Lemma 2.4]). *For any lattice* $\Lambda \subset \mathbb{R}^n$*,* $\varepsilon \in (0, 1)$*,* $\Sigma > \mathbf{0}$ *such that* $\sqrt{\Sigma} \ge \eta_\varepsilon(\Lambda)$*, and any* $\mathbf{v} \in \mathbb{R}^n$*:*

$$\rho_{\sqrt{\Sigma}}(\Lambda + \mathbf{v}) \in \left[\frac{1-\varepsilon}{1+\varepsilon}, 1\right] \cdot \rho_{\sqrt{\Sigma}}(\Lambda).$$

Lemma 2.8 ([30, Lemma 4.4]). *For any* n*-dimensional lattice* Λ*, vector* $\mathbf{c} \in \mathbb{R}^n$*, and reals* $\varepsilon \in (0, 1)$*,* $\alpha \ge \eta_\varepsilon(\Lambda)$*, it holds that*

$$(1 - \varepsilon)\alpha^n \det(\Lambda)^{-1} \le \rho(\Lambda) \le (1 + \varepsilon)\alpha^n \det(\Lambda)^{-1}.$$

Lemma 2.9 ([30, Lemma 3.3]). *For a rank* r *lattice* $\Lambda \subset \mathbb{R}^n$ *with* $1 \le r \le n$ *and* $\varepsilon > 0$*,*

$$\eta_\varepsilon(\Lambda) \le \sqrt{\frac{\ln\big(2r(1 + 1/\varepsilon)\big)}{\pi}} \cdot \lambda_r(\Lambda) \le \sqrt{\frac{\ln\big(2r(1 + 1/\varepsilon)\big)}{\pi}} \cdot \max_{i \in [r]} \|\mathbf{b}_i\|_2,$$

where $\lambda_r(\Lambda)$ *is the smallest radius of an* n*-dimensional ball that contains* r *linearly independent vectors in* Λ*, and* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_r]$ *is a basis of the lattice* Λ.

Lemma 2.10 ([13, Lemma 4]). *There is a probabilistic polynomial time algorithm that, given a basis* $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ *of a full-rank* n*-dimensional lattice* Λ*, a positive definite symmetric matrix* Σ *and* $\mathbf{c} \in \mathbb{R}^n$ *returns a sample from* $\mathcal{D}_{\Lambda,\mathbf{c},\sqrt{\Sigma}}$*, assuming that*

$$\sqrt{\ln(2n + 4)/\pi} \cdot \max_i \|\Sigma^{-1/2}\mathbf{b}_i\|_2 \le 1.$$

The above lemma works for *full-rank* lattices, while our work needs an analogous result for general rank-r lattices in $\mathbb{R}^n$ with $r \le n$. We show in the lemma below the extension to the general rank-r case.

Lemma 2.11 (Reduction to full-rank sampler). *Let* r *and* n *be positive integers with* $1 \le r \le n$*. Suppose there is an algorithm* $\mathcal{A}$ *that, given as input a full-rank column* $\mathbb{Z}$*-basis* $\mathbf{B}' = [\mathbf{b}_1, \ldots, \mathbf{b}_r] \in \mathbb{R}^{r \times r}$ *for a full-rank lattice* $\Lambda' := \mathbf{B}'\mathbb{Z}^r \subseteq \mathbb{R}^r$*, a vector* $\mathbf{t} \in \mathbb{R}^r$ *and a parameter* $s \ge \eta(r) \cdot \max_i \|\mathbf{b}_i\|_2$ *(for some function* $\eta : \mathbb{R} \mapsto \mathbb{R}$*), outputs a sample from the discrete Gaussian* $\mathcal{D}_{\Lambda'+\mathbf{t},s}$ *in time* $T(r, \ell(\mathbf{B}'))$*, where* $\ell(\mathbf{B}')$ *denotes the bit-length of* $\mathbf{B}'$*. Then there is an algorithm* $\mathcal{B}$ *that, given a column* $\mathbb{Z}$*-basis* $\mathbf{B}_0 = [\mathbf{b}_{0,1}, \ldots, \mathbf{b}_{0,r}] \in \mathbb{R}^{n \times r}$ *for a rank* r *lattice* $\Lambda_0 := \mathbf{B}_0 \cdot \mathbb{Z}^r \subseteq \mathbb{R}^n$*, a vector* $\mathbf{c}_0 \in \mathbb{R}^n$ *and a positive definite symmetric matrix* $\Sigma_0 \in \mathbb{R}^{n \times n}$*, returns a sample from* $\mathcal{D}_{\Lambda_0+\mathbf{c}_0,\sqrt{\Sigma_0}}$*, assuming that*

$$\eta(r) \cdot \max_{i \in [r]} \|\Sigma_0^{-1/2}\mathbf{b}_{0,i}\|_2 \le 1, \tag{1}$$

in time $\mathrm{poly}(\ell(\mathbf{B}_0)) + T\big(r, \ell(\mathbf{B}_0)\big)$*. In particular, if* $\mathcal{A}$ *runs in polynomial time, then so does* $\mathcal{B}$.

Proof. The algorithm $\mathcal{B}$ runs as follows on input the basis $\mathbf{B}_0 = [\mathbf{b}_{0,1}, \ldots, \mathbf{b}_{0,r}]$ for a rank r lattice $\Lambda_0 := \mathbf{B}_0 \cdot \mathbb{Z}^r \subseteq \mathbb{R}^n$, a vector $\mathbf{c}_0 \in \mathbb{R}^n$ and a positive definite symmetric matrix $\Sigma_0 \in \mathbb{R}^{n \times n}$:

(1) Let $\mathbf{S}_0 := \sqrt{\Sigma_0} \in \mathbb{R}^{n \times n}$ (i.e. $\mathbf{S}_0\mathbf{S}_0^\top = \Sigma_0$).
(2) Let lattice $\Lambda := \mathbf{S}_0^{-1} \cdot \Lambda_0$ with $\mathbb{Z}$-basis $\mathbf{B} := \mathbf{S}_0^{-1} \cdot \mathbf{B}_0 \in \mathbb{R}^{n \times r}$, and vector $\mathbf{c} := \mathbf{S}_0^{-1} \cdot \mathbf{c}_0 \in \mathbb{R}^n$.

(3) Let $V := \mathrm{span}_{\mathbb{R}}(\mathbf{B})$ denote the rank r vector space over $\mathbb{R}$ spanned by the columns of $\mathbf{B}$, and compute an orthonormal column $\mathbb{R}$-basis $\mathbf{U} \in \mathbb{R}^{n\times r}$ for V (e.g. $\mathbf{U}$ can be computed as the normalized Gram-Schmidt Orthogonalization of $\mathbf{B}$).
(4) Let $\mathbf{c}_{\|} := \mathbf{U}\mathbf{U}^\top \mathbf{c} \in V$ denote the projection of $\mathbf{c}$ onto V, and let $\mathbf{c}_{\perp} := \mathbf{c} - \mathbf{c}_{\|} \in V^{\perp}$ denote the projection of $\mathbf{c}$ onto the orthogonal complement $V^{\perp} := \{\mathbf{v}_{\perp} \in \mathbb{R}^n : \mathbf{v}_{\perp}^\top \cdot V = \mathbf{0}\}$ of V.
(5) Let $\mathbf{B}' := \mathbf{U}^\top \cdot \mathbf{B} \in \mathbb{R}^{r\times r}$ denote a basis for the full rank lattice $\Lambda' := \mathbf{U}^\top \cdot \Lambda \subseteq \mathbb{R}^r$, let $\mathbf{t} := \mathbf{U}^\top \cdot \mathbf{c} \in \mathbb{R}^r$, and $s := 1$.
(6) Call algorithm $\mathcal{A}$ on input $(\mathbf{B}', \mathbf{t}, s)$ to sample $\mathbf{y} \leftarrow \mathcal{D}_{\Lambda'+\mathbf{t},1}$.
(7) Let $\mathbf{x} := \mathbf{U} \cdot \mathbf{y} + \mathbf{c}_{\perp}$ (remark: $\mathbf{x}$ is distributed as $\mathcal{D}_{\Lambda+\mathbf{c},1}$).
(8) Let $\mathbf{x}_0 := \mathbf{S}_0 \cdot \mathbf{x}$ (remark: $\mathbf{x}_0$ is distributed as $\mathcal{D}_{\Lambda_0+\mathbf{c}_0,\sqrt{\Sigma_0}}$).
(9) Return $\mathbf{x}_0$.

To analyze the output distribution of $\mathbf{x}_0$, we first observe that in the call to algorithm $\mathcal{A}$ in line (6), the required parameter condition $s \geq \eta(r) \cdot \max_i \|\mathbf{b}'_i\|_2$ is satisfied. Indeed, the assumed condition (1) and the setting $s = 1$ implies that $s \geq \eta(r) \cdot \max_{i\in[r]} \|\mathbf{S}_0^{-1}\mathbf{b}_{0,i}\|_2 \geq \eta(r)\cdot\max_{i\in[r]} \|\mathbf{U}^\top\cdot\mathbf{S}_0^{-1}\mathbf{b}_{0,i}\|_2 = \eta(r)\cdot\max_i \|\mathbf{b}'_i\|_2$, as required, where in the inequality we used the fact that the mapping $\mathbf{v} \mapsto \mathbf{U}^\top\mathbf{v}$ is a projection, and hence cannot increase the norm. It follows that $\mathbf{y}$ is distributed as $\mathcal{D}_{\Lambda'+\mathbf{t},1}$.

Next, we show that $\mathbf{x}$ computed in line (7) is distributed as $\mathcal{D}_{\Lambda+\mathbf{c},1}$. Indeed, since $\mathbf{y} \in \Lambda' + \mathbf{t}$, $\Lambda' = \mathbf{U}^\top \cdot \Lambda$ and $\mathbf{t} := \mathbf{U}^\top \cdot \mathbf{c}$, we have $\mathbf{y} = \mathbf{U}^\top \cdot \mathbf{v} + \mathbf{U}^\top \cdot \mathbf{c}$ for some $\mathbf{v} \in \Lambda$. Hence $\mathbf{x} := \mathbf{U} \cdot \mathbf{y} + \mathbf{c}_{\perp} = \mathbf{U}\mathbf{U}^\top \cdot \mathbf{v} + \mathbf{U}\mathbf{U}^\top \cdot \mathbf{c} + \mathbf{c}_{\perp}$. Since $\mathbf{v} \in V$, the projection $\mathbf{U}\mathbf{U}^\top$ onto V acts as the identity on $\mathbf{v}$ (indeed, we have $\mathbf{v} = \mathbf{U}\mathbf{w}$ for some $\mathbf{w}$ and hence $\mathbf{U}\mathbf{U}^\top\mathbf{v} = \mathbf{U}\mathbf{U}^\top\mathbf{U}\mathbf{w} = \mathbf{U}\mathbf{w} = \mathbf{v}$ since $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ due to the orthonormal columns of $\mathbf{U}$). Furthermore, $\mathbf{U}\mathbf{U}^\top \cdot \mathbf{c} = \mathbf{c}_{\|}$. So we get $\mathbf{x} = \mathbf{v} + \mathbf{c}_{\|} + \mathbf{c}_{\perp} = \mathbf{v} + \mathbf{c}$ and hence the support of $\mathbf{x}$ is $\Lambda + \mathbf{c}$. Furthermore, for each $\overline{\mathbf{x}} = \mathbf{v} + \mathbf{c}_{\|} + \mathbf{c}_{\perp} \in \Lambda + \mathbf{c}$ with $\mathbf{v} \in \Lambda$, we have

$$\begin{aligned}
\Pr[\mathbf{x} = \overline{\mathbf{x}}] &= \Pr[\mathbf{U}\mathbf{y} + \mathbf{c}_{\perp} = \mathbf{v} + \mathbf{c}_{\|} + \mathbf{c}_{\perp}] && (2)\\
&= \Pr[\mathbf{U}\mathbf{y} = \mathbf{v} + \mathbf{c}_{\|}] && (3)\\
&= \Pr[\mathbf{y} = \mathbf{U}^\top \cdot (\mathbf{v} + \mathbf{c}_{\|})] && (4)\\
&= \mathcal{D}_{\Lambda'+\mathbf{t},1}\left(\mathbf{U}^\top \cdot (\mathbf{v} + \mathbf{c}_{\|})\right) && (5)\\
&= \alpha \cdot \exp\left(-\pi\|\mathbf{U}^\top \cdot (\mathbf{v} + \mathbf{c}_{\|})\|^2\right) \quad \text{(normaliz. factor } \alpha) && (6)\\
&= \alpha \cdot \exp\left(-\pi\|\mathbf{v} + \mathbf{c}_{\|}\|^2\right) \quad (\mathbf{U}^\top \text{ preserves norm on V}) && (7)\\
&= \alpha \exp(\pi\|\mathbf{c}_{\perp}\|^2) \cdot \exp\left(-\pi\|\mathbf{v} + \mathbf{c}_{\|}\|^2\right) \cdot \exp(-\pi\|\mathbf{c}_{\perp}\|^2) && (8)\\
&= \alpha \exp(\pi\|\mathbf{c}_{\perp}\|^2) \cdot \exp\left(-\pi\|\mathbf{v} + \mathbf{c}_{\|} + \mathbf{c}_{\perp}\|^2\right) && (9)\\
&= \mathcal{D}_{\Lambda+\mathbf{c},1}(\overline{\mathbf{x}}), && (10)
\end{aligned}$$

as required, where in the second last equality above we used the orthogonality of $\mathbf{v} + \mathbf{c}_{\|}$ and $\mathbf{c}_{\perp}$, and in the last equality, the fact that $\alpha \exp(\pi\|\mathbf{c}_{\perp}\|^2)$ is a normalization factor independent of $\mathbf{v}$. Finally, we show that $\mathbf{x}_0$ computed in step (8) of the above algorithm is distributed as $\mathcal{D}_{\Lambda_0+\mathbf{c}_0,\sqrt{\Sigma_0}}$. Indeed, since $\mathbf{x} = \mathbf{v} + \mathbf{c} \in \Lambda + \mathbf{c}$ for some $\mathbf{v} \in \Lambda$, we have $\mathbf{x}_0 := \mathbf{S}_0 \cdot (\mathbf{v} + \mathbf{c}) \in \mathbf{S}_0 \cdot (\Lambda + \mathbf{c}) = \Lambda_0 + \mathbf{c}_0$, as required. Furthermore, for each $\overline{\mathbf{x}}_0 = \overline{\mathbf{v}}_0 + \mathbf{c}_0 \in \Lambda_0 + \mathbf{c}_0$ with $\overline{\mathbf{v}}_0 \in \Lambda_0$, we have

$$\begin{aligned}
\Pr[\mathbf{x}_0 = \overline{\mathbf{x}}_0] &= \Pr[\mathbf{S}_0 \cdot \mathbf{x} = \overline{\mathbf{v}}_0 + \mathbf{c}_0] && (11)\\
&= \Pr[\mathbf{x} = \mathbf{S}_0^{-1} \cdot (\overline{\mathbf{v}}_0 + \mathbf{c}_0)] && (12)\\
&= \mathcal{D}_{\Lambda+\mathbf{c},1}\left(\mathbf{S}_0^{-1} \cdot \overline{\mathbf{x}}_0\right) && (13)\\
&= \alpha' \cdot \exp\left(-\pi\overline{\mathbf{x}}_0^\top \cdot (\mathbf{S}_0\mathbf{S}_0^\top)^{-1} \cdot \overline{\mathbf{x}}_0\right) \text{ (normaliz. factor } \alpha') && (14)\\
&= \mathcal{D}_{\Lambda_0+\mathbf{c}_0,\sqrt{\Sigma_0}}(\overline{\mathbf{x}}_0), && (15)
\end{aligned}$$

as required. The time bound follows from the fact that all the linear algebra operations run in polynomial time. □

2.2 Linear Maps

Let $k \in \mathbb{N}^+$ and $\mathbf{h} \in \mathcal{R}^k$. We define the $\mathcal{R}$-module and $\mathcal{R}_q$-module homomorphisms as:

$$\varphi_{\mathbf{h}} : \mathcal{R}^k \to \mathcal{R}, \quad \varphi_{\mathbf{h}}(\mathbf{s}) = \mathbf{h}^\top\mathbf{s},$$
$$\varphi_{\mathbf{h},q} : \mathcal{R}_q^k \to \mathcal{R}_q, \quad \varphi_{\mathbf{h},q}(\mathbf{s}) = \mathbf{h}^\top\mathbf{s} \bmod q.$$

Their respective kernels are denoted by:

$$\ker(\varphi_{\mathbf{h}}) = \{\mathbf{s} \in \mathcal{R}^k : \mathbf{h}^\top\mathbf{s} = 0\},$$
$$\ker(\varphi_{\mathbf{h},q}) = \{\mathbf{s} \in \mathcal{R}_q^k : \mathbf{h}^\top\mathbf{s} = 0 \pmod q\}.$$

Let $\pi : \ker(\varphi_{\mathbf{h}}) \to \ker(\varphi_{\mathbf{h},q})$ be the natural reduction map defined by $\pi(\mathbf{x}) = \mathbf{x} \pmod q$.

Lemma 2.12 (Coset Resampling). *Let $\mathbf{h} \in \mathcal{R}^k$. For $\alpha > 0$, the following two distributions over $\mathcal{R}^k \times \mathcal{R}$ are identical:*
(1) $D_1 = \{(\mathbf{s}, z) : \mathbf{s} \leftarrow \mathcal{D}_{\mathcal{R}^k,\alpha}, z = \mathbf{h}^\top\mathbf{s}\}$.
(2) $D_2 = \{(\tilde{\mathbf{s}}, z) : \mathbf{s} \leftarrow \mathcal{D}_{\mathcal{R}^k,\alpha}, z = \mathbf{h}^\top\mathbf{s}, \tilde{\mathbf{s}} \leftarrow \mathcal{D}_{\ker(\varphi_{\mathbf{h}})+\mathbf{s},\alpha}\}$.

Proof. Fix $(\hat{\mathbf{s}}, \hat{z}) \in \mathcal{R}^k \times \mathcal{R}$. Let $C(\hat{z}) = \{\mathbf{x} \in \mathcal{R}^k : \mathbf{h}^\top\mathbf{x} = \hat{z}\}$. Note that if $\hat{\mathbf{s}} \in C(\hat{z})$, then $C(\hat{z}) = \ker(\varphi_{\mathbf{h}}) + \hat{\mathbf{s}}$. Under D_1, $\Pr[(\hat{\mathbf{s}}, \hat{z})] = \mathcal{D}_{\mathcal{R}^k,\alpha}(\hat{\mathbf{s}})$ if $\mathbf{h}^\top\hat{\mathbf{s}} = \hat{z}$, and 0 otherwise. Under D_2, the probability is:

$$\Pr[z = \hat{z}] \cdot \Pr[\tilde{\mathbf{s}} = \hat{\mathbf{s}} \mid z = \hat{z}] = \frac{\rho_\alpha(C(\hat{z}))}{\rho_\alpha(\mathcal{R}^k)} \cdot \frac{\rho_\alpha(\hat{\mathbf{s}})}{\rho_\alpha(C(\hat{z}))} = \frac{\rho_\alpha(\hat{\mathbf{s}})}{\rho_\alpha(\mathcal{R}^k)},$$

which is exactly $\mathcal{D}_{\mathcal{R}^k,\alpha}(\hat{\mathbf{s}})$ if $\hat{\mathbf{s}} \in C(\hat{z})$, and 0 otherwise. □

Lemma 2.13. *Let $\mathbf{h} \in \mathcal{R}^k$. If there exists $\mathbf{u} \in \mathcal{R}^k$ such that $\mathbf{h}^\top\mathbf{u} = 1$, then $\pi : \ker(\varphi_{\mathbf{h}}) \to \ker(\varphi_{\mathbf{h},q})$ defined by $\pi(\mathbf{x}) = \mathbf{x} \bmod q$ is surjective.*

Proof. Let $\mathbf{t}_q \in \ker(\varphi_{\mathbf{h},q})$ and let $\mathbf{t} \in \mathcal{R}^k$ be any lift such that $\mathbf{t} \bmod q = \mathbf{t}_q$. Since $\mathbf{h}^\top\mathbf{t} = 0 \pmod q$, we have $\mathbf{h}^\top\mathbf{t} = qr$ for some $r \in \mathcal{R}$. Set $\mathbf{s} = \mathbf{t} - qr\mathbf{u}$. Then $\mathbf{h}^\top\mathbf{s} = \mathbf{h}^\top\mathbf{t} - q(\mathbf{h}^\top\mathbf{u})r = qr - qr = 0$. Thus $\mathbf{s} \in \ker(\varphi_{\mathbf{h}})$. Since $\mathbf{s} \equiv \mathbf{t} \equiv \mathbf{t}_q \pmod q$, the map π is surjective. □

Lemma 2.14. *Let $k \in \mathbb{N}^+$ such that $k > 1$. Let $\mathbf{h}^\top = [1 \mid (\mathbf{h}')^\top] \in \mathcal{R}^k$ where $\mathbf{h}' \in \mathcal{R}^{k-1}$. Then, $\mathbf{B} = \begin{pmatrix} -(\mathbf{h}')^\top \\ \mathbf{I}_{k-1} \end{pmatrix} \in \mathcal{R}^{k\times(k-1)}$ is an $\mathcal{R}$-basis for $\ker(\varphi_{\mathbf{h}})$, and $\mathbf{B}_q = \mathbf{B} \bmod q$ is an $\mathcal{R}_q$-basis for $\ker(\varphi_{\mathbf{h},q})$.*

Proof. We first prove that $\mathbf{B}$ is a basis of $\ker(\varphi_{\mathbf{h}})$. Let $\mathbf{x} \in \ker(\varphi_{\mathbf{h}}) \subseteq \mathcal{R}^k$ be arbitrary. Partition $\mathbf{x}$ into $x_1 \in \mathcal{R}$ and $\mathbf{x}_2 \in \mathcal{R}^{k-1}$. Then $\mathbf{h}^\top\mathbf{x} = 0$ implies $1 \cdot x_1 + (\mathbf{h}')^\top\mathbf{x}_2 = 0$. We can therefore write $\mathbf{x} = \begin{pmatrix} -(\mathbf{h}')^\top\mathbf{x}_2 \\ \mathbf{x}_2 \end{pmatrix} = \mathbf{B}\mathbf{x}_2$. The columns of $\mathbf{B}$ are linearly independent because the bottom $k-1$ rows form $\mathbf{I}_{k-1}$. If $\mathbf{B}\mathbf{c} = \mathbf{0}$, then $\mathbf{I}_{k-1}\mathbf{c} = \mathbf{0}$, implying $\mathbf{c} = \mathbf{0}$. Thus, $\mathbf{B}$ is an $\mathcal{R}$-basis for $\ker(\varphi_{\mathbf{h}})$.

Second, we show that $\mathbf{B}_q$ is an $\mathcal{R}_q$-basis for $\ker(\varphi_{\mathbf{h},q})$. Define $\mathbf{v} = (1, 0, \ldots, 0)^\top \in \mathcal{R}^k$. Since $\mathbf{h}^\top\mathbf{v} = 1$, Lemma 2.13 implies that the reduction map $\pi : \ker(\varphi_{\mathbf{h}}) \to \ker(\varphi_{\mathbf{h},q})$ is surjective. Let $\mathbf{u}_q \in \ker(\varphi_{\mathbf{h},q})$. By the surjectivity of π, there exists $\mathbf{u} \in \ker(\varphi_{\mathbf{h}})$ such that $\mathbf{u} \bmod q = \mathbf{u}_q$. Since $\mathbf{B}$ spans $\ker(\varphi_{\mathbf{h}})$, we have $\mathbf{u} = \mathbf{B}\mathbf{r}$ for some $\mathbf{r} \in \mathcal{R}^{k-1}$. Then $\mathbf{u}_q = \mathbf{B}\mathbf{r} \bmod q = \mathbf{B}_q(\mathbf{r} \bmod q)$, so $\mathbf{B}_q$ spans $\ker(\varphi_{\mathbf{h},q})$. Suppose $\mathbf{B}_q\mathbf{c}_q = \mathbf{0} \bmod q$ for $\mathbf{c}_q \in \mathcal{R}_q^{k-1}$. The last $k-1$ rows of this system give $\mathbf{I}_{k-1}\mathbf{c}_q = \mathbf{0} \bmod q$, so $\mathbf{c}_q = \mathbf{0} \in \mathcal{R}_q^{k-1}$. Thus, $\mathbf{B}_q$ is a basis for $\ker(\varphi_{\mathbf{h},q})$, which is a free module of rank $k-1$. □

Symbol	Description
λ	security parameter
d	a power of 2 to set $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$
q, q'	moduli for HVC and KOTS, respectively
τ	depth of the labelled full binary tree
N	number of signers in the multi-signature
η	HVC employs $(2\eta + 1)$-ary decomposition
κ, κ'	$\kappa := \lceil \log_{2\eta+1} q \rceil$ and $\kappa' := \lceil \log_{2\eta+1} q' \rceil$
ω	dimension of an HVC commitment $c \in \mathcal{R}_q^{\omega}$
ρ, ν	message domain of the HVC is $(\mathcal{R}_{q'}^{\rho \times \nu})^{2^\tau}$
$\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1$	HVC public parameter: $\mathbf{B} \in \mathcal{R}_q^{\omega \times \rho\nu\kappa'}$ and $\mathbf{A}_0, \mathbf{A}_1 \in \mathcal{R}_q^{\omega \times \omega\kappa}$
β_{agg}	norm bound used by HVC strong verification
β_{encode}	norm bound for encoded HVC openings
W, α_w	weight set used in HVC / KOTS with $W = \mathcal{T}_{\alpha_w}$
α_H	hash output in KOTS: $\mathsf{H}(\mu) \in \mathcal{T}_{\alpha_H}^{k-1}$
m, n	KOTS public parameter: $\mathbf{A} \in \mathcal{R}_{q'}^{m \times n}$
k	KOTS secret $\mathbf{S} \in \mathcal{R}^{k \times m}$ and public key $\mathbf{T} \in \mathcal{R}_{q'}^{k \times n}$
α	width parameter of KOTS secret
β_z	norm bound for individual signature
β_σ	norm bound for weighted sum of individual signatures
ϵ_{cor}	individual correctness error for KOTS
ϵ_{hom}	probabilistic homomorphism error (KOTS / HVC)
γ	maximum number of aggregation attempts

Table 3: Symbols used in this paper.

2.3 Lattice Assumptions

Definition 2.4 ($\mathsf{MLWE}_{q,m,n,k,\chi}$). *Let χ be a distribution over $\mathcal{R}$. Sample $\mathbf{A} \leftarrow \mathcal{R}_q^{m \times n}$, $\mathbf{S} \leftarrow \chi^{k \times m}$, $\mathbf{E} \leftarrow \chi^{k \times n}$. The $\mathsf{MLWE}_{q,m,n,k,\chi}$ problem asks an adversary $\mathcal{A}$ to distinguish between the pairs $(\mathbf{A}, \mathbf{SA}+\mathbf{E})$ and $(\mathbf{A}, \mathbf{U})$, where $\mathbf{U} \leftarrow \mathcal{R}_q^{k \times n}$. The advantage of $\mathcal{A}$ is defined as*

$$\mathsf{Adv}^{\mathsf{MLWE}}_{q,m,n,k,\chi}(\mathcal{A}) := \left| \Pr[\mathcal{A}(\mathbf{A}, \mathbf{SA} + \mathbf{E}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{U}) = 1] \right|,$$

where the probability is taken over the randomness of $\mathbf{A}$, $\mathbf{S}$, $\mathbf{E}$, $\mathbf{U}$, and the internal coins of $\mathcal{A}$.

Definition 2.5 ($\mathsf{MSIS}^{\infty}_{q,m,n,\beta}$). *Let $m, n, \beta, q \in \mathbb{N}^+$ such that $0 < \beta < q$. Then, for a given matrix $\mathbf{A} \leftarrow \mathcal{R}_q^{m \times n}$, the goal of the $\mathsf{MSIS}^{\infty}_{q,m,n,\beta}$ problem is to find $\mathbf{x} \in \mathcal{R}^n$ such that $\mathbf{Ax} = \mathbf{0} \pmod q$ and $0 < \|\mathbf{x}\|_\infty \le \beta$. We say that a PPT adversary $\mathcal{A}$ has advantage ϵ in solving $\mathsf{MSIS}^{\infty}_{q,m,n,\beta}$ if*

$$\Pr[0 < \|\mathbf{x}\|_\infty \le \beta \wedge \mathbf{Ax} = \mathbf{0} \mid \mathcal{A}(\mathbf{A}) = \mathbf{x} \pmod q] \ge \epsilon,$$

where the probability is taken over the randomness of $\mathbf{A}$ and the internal coins of $\mathcal{A}$.

In this paper, we also consider a variant of the MSIS assumption in which the matrix $\mathbf{A}$ is of the form $\mathbf{A} = [\mathbf{I} \mid \mathbf{A}']$. We note that this variant is as hard as the standard MSIS above.

3 Dual Hint-MLWE Problem

We start with the problem definition.

Definition 3.1 (Dual Hint-MLWE). *Let $q', d, m, n, k \in \mathbb{N}^+$ such that $k > 1$ and χ, C be distributions over $\mathcal{R}$. Let $\mathbf{A} = [\mathbf{I}_n; \mathbf{A}_2] \in \mathcal{R}_{q'}^{m \times n}$ for $\mathbf{A}_2 \leftarrow \mathcal{R}_{q'}^{(m-n) \times n}$, $\mathbf{S} \leftarrow \chi^{k \times m}$, and $\mathbf{h} = [1, \mathbf{h}'] \in \mathcal{R}^k$ for $\mathbf{h}' \leftarrow C^{k-1}$. The Dual Hint-MLWE problem is to distinguish between $D_0 := (\mathbf{A}, \mathbf{SA}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$ and $D_1 := (\mathbf{A}, (\mathbf{S}+\mathbf{U})\mathbf{A}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$, where $\mathbf{U} \in \mathcal{R}_{q'}^{k \times m}$ with each column $\mathbf{u}_i \leftarrow \ker(\varphi_{\mathbf{h},q'})$ sampled independently. The advantage of a PPT adversary $\mathcal{A}$ is defined as:*

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

If $\chi = \mathcal{D}_{\mathcal{R},\alpha}$, we denote the problem by $\mathsf{DualHintMLWE}_{q',m,n,k,\alpha,C}$.

We prove the hardness of the Dual Hint-MLWE problem under the MLWE assumption when $\chi = \mathcal{D}_{\mathcal{R},\alpha}$ and $C = \mathcal{T}_{\alpha_H}$, i.e., the problem $\mathsf{DualHintMLWE}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$.

Theorem 3.1 (Hardness of Dual Hint–MLWE). *Let $q', d, m, n, k, \alpha_H \in \mathbb{N}^+$ with $k > 1$, and let $\alpha_0, \alpha, \varepsilon_2 \in \mathbb{R}_{>0}$. Let $\mathbf{h}' \leftarrow (\mathcal{T}_{\alpha_H})^{k-1}$ and define $\mathbf{h} = [1; \mathbf{h}']$, $\mathbf{B} = \begin{pmatrix} -(\mathbf{h}')^\top \\ \mathbf{I}_{k-1} \end{pmatrix} \in \mathcal{R}^{k \times (k-1)}$, and $\Sigma_1 = \alpha^2 \mathbf{I}_k - \alpha_0^2 \mathbf{B}\mathbf{B}^\top$. If $\alpha^2 \ge \alpha_0^2 \|\mathbf{B}\|_2^2 + (1 + \alpha_H) \frac{\ln(2d(k-1)(1+1/\varepsilon_2))}{\pi}$, then there exists a reduction $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathsf{DualHintMLWE}}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{MLWE}}_{q',m-n,n,k-1,\alpha_0}(\mathcal{B}) + 8m\varepsilon_2.$$

Proof. Let $\mathbf{A}_2 \leftarrow \mathcal{R}_{q'}^{(m-n) \times n}$, $\mathbf{A} := [\mathbf{I}_n; \mathbf{A}_2] \in \mathcal{R}_{q'}^{m \times n}$, $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_m] \leftarrow \mathcal{D}_{\mathcal{R}^{k \times m},\alpha}$, and $\mathbf{h} = [1, \mathbf{h}']$ with $\mathbf{h}' \leftarrow (\mathcal{T}_{\alpha_H})^{k-1}$. Consider the following sequence of hybrids:

- G_0: Output $(\mathbf{A}, \mathbf{SA}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$.
- G_1: For each $i \in [m]$, sample $\tilde{\mathbf{s}}_i \leftarrow \mathcal{D}_{\ker(\varphi_\mathbf{h}) + \mathbf{s}_i,\alpha}$. Let $\tilde{\mathbf{S}} = [\tilde{\mathbf{s}}_1, \ldots, \tilde{\mathbf{s}}_m]$ and output $(\mathbf{A}, \tilde{\mathbf{S}}\mathbf{A}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$.
- G_2: Sample $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$ with $\mathbf{u}_i \leftarrow \ker(\varphi_{\mathbf{h},q'})$ and output $(\mathbf{A}, (\mathbf{S} + \mathbf{U})\mathbf{A}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$.

By definition, G_0 and G_2 correspond to the Definition 3.1 distributions. We now establish the statistical equivalence of G_0 and G_1.

Lemma 3.1. *The output distributions of G_0 and G_1 are identical.*

Proof. For each column $i \in [m]$, $\mathbf{s}_i \leftarrow \mathcal{D}_{\mathcal{R}^k,\alpha}$. By Lemma 2.12, the joint distributions $(\mathbf{s}_i, \mathbf{h}^\top\mathbf{s}_i)$ and $(\tilde{\mathbf{s}}_i, \mathbf{h}^\top\mathbf{s}_i)$ are identical, where $\tilde{\mathbf{s}}_i \leftarrow \mathcal{D}_{\ker(\varphi_\mathbf{h}) + \mathbf{s}_i,\alpha}$. Since columns are independent and $\mathbf{h}$ is sampled identically in both hybrids, $(\mathbf{S}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$ in G_0 is distributed identically to $(\tilde{\mathbf{S}}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$ in G_1. Finally, since $\mathbf{A}$ is a public parameter independent of the secrets and sampled identically in both games, the map $(\mathbf{S}, \mathbf{h}^\top\mathbf{S}, \mathbf{A}) \mapsto (\mathbf{SA}, \mathbf{h}^\top\mathbf{S}, \mathbf{A})$ preserves identical distribution. Thus, $\mathrm{G}_0 \equiv \mathrm{G}_1$. □

Let $(\mathbf{A}', \mathbf{T}') \in \mathcal{R}_{q'}^{(m-n) \times n} \times \mathcal{R}_{q'}^{(k-1) \times n}$ be given $\mathsf{MLWE}_{q',m-n,n,k-1,\alpha_0}$ instance and $\mathbf{A} = [\mathbf{I}_n; \mathbf{A}']$. Our reduction $\mathcal{B}$ works as follows.

(1) Sample $\mathbf{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_m] \leftarrow \mathcal{D}_{\mathcal{R}^{k \times m},\alpha}$.
(2) For $i \in [m]$, sample $\mathbf{r}_i \leftarrow \mathcal{D}_{\ker(\varphi_\mathbf{h}) + \mathbf{s}_i, \sqrt{\Sigma_1}}$, relying on the $\mathcal{R}$-basis $\mathbf{B}$ for $\ker(\varphi_\mathbf{h})$ by Lemma 3.5. Let $\mathbf{R} := [\mathbf{r}_1, \ldots, \mathbf{r}_m]$.
(3) Compute $\mathbf{T} := \mathbf{B}\,\mathbf{T}' + \mathbf{R}\,\mathbf{A} \pmod{q'} \in \mathcal{R}_{q'}^{k \times n}$.
(4) Send $(\mathbf{A}, \mathbf{T}, \mathbf{h}, \mathbf{h}^\top\mathbf{S})$ to $\mathcal{A}$ and output its response.

Lemmas 3.3 and 3.4 imply that $\mathcal{B}$'s output is $8m\varepsilon_2$-close to G_1 if $\mathbf{T}' = \mathbf{S}'\mathbf{A}$ for $\mathbf{S}' \leftarrow \mathcal{D}_{\mathcal{R}^{(k-1) \times m},\alpha_0}$ and distributed as G_2 if $\mathbf{T}'$ is uniform. Consequently, any Dual Hint-MLWE distinguishing advantage reduces to MLWE with at most $8m\varepsilon_2$ loss. □

Lemma 3.2. *The matrix $\Sigma_1 = \alpha^2\mathbf{I}_k - \alpha_0^2\mathbf{B}\mathbf{B}^\top$ is symmetric positive definite, and $\sqrt{\Sigma_1} \succeq \eta_{\varepsilon_2}(\ker(\varphi_\mathbf{h}))$.*

Proof. Symmetry is immediate: $\Sigma_1^\top = (\alpha^2\mathbf{I}_k)^\top - \alpha_0^2(\mathbf{B}\mathbf{B}^\top)^\top = \Sigma_1$. To show positive definiteness, observe that:

$$\lambda_{\min}(\Sigma_1) = \alpha^2 - \alpha_0^2\lambda_{\max}(\mathbf{B}\mathbf{B}^\top) = \alpha^2 - \alpha_0^2\|\mathbf{B}\|_2^2 \tag{16}$$

$$\geq (1+\alpha_H)\frac{\ln(2d(k-1)(1+1/\varepsilon_2))}{\pi} > 0, \tag{17}$$

making Σ_1 strictly positive definite. Next, we show that $\sqrt{\Sigma_1} \succeq \eta_{\varepsilon_2}(\ker(\varphi_{\mathbf{h}}))$. Recall that $\ker(\varphi_{\mathbf{h}}) \subseteq \mathcal{R}^k$. By Lemma 2.14, the matrix $\mathbf{B} = \begin{pmatrix} -(\mathbf{h}')^\top \\ \mathbf{I}_{k-1} \end{pmatrix} \in \mathcal{R}^{k\times(k-1)}$ is an $\mathcal{R}$-basis of $\ker(\varphi_{\mathbf{h}})$. Let $n = (k-1)d$ be the rank of $\ker(\varphi_{\mathbf{h}})$. By Lemma 2.9,

$$\eta_{\varepsilon_2}(\ker(\varphi_{\mathbf{h}})) \leq \sqrt{\frac{\ln(2n(1+1/\varepsilon_2))}{\pi}} \cdot \max_i \|\mathbf{b}_i\|_2 \tag{18}$$

$$= \sqrt{\frac{\ln(2n(1+1/\varepsilon_2))}{\pi}} \cdot \sqrt{1+\alpha_H}, \tag{19}$$

where $\mathbf{b}_i$ is the column vector of $\mathbf{B}$. Hence $\lambda_{\min}(\Sigma_1) \geq \eta_{\varepsilon_2}(\ker(\varphi_{\mathbf{h}}))^2$ implying $\sqrt{\Sigma_1} \succeq \eta_{\varepsilon_2}(\ker(\varphi_{\mathbf{h}}))$, which completes the proof. □

Lemma 3.3. *If $\mathbf{T}' = \mathbf{S}'\mathbf{A}$ with $\mathbf{S}' \leftarrow \mathcal{D}_{\mathcal{R}^{(k-1)\times m},\alpha_0}$, then the output of $\mathcal{B}$ is within statistical distance $8m\varepsilon_2$ of the distribution of G_1.*

Proof. We only need to show that $\mathbf{T} = (\mathbf{B}\mathbf{S}' + \mathbf{R})\mathbf{A}$ output by the reduction $\mathcal{B}$ is within $8m\varepsilon_2$ of $\tilde{\mathbf{S}}\mathbf{A}$ output in G_1.

Let $\Sigma_2 = \alpha_0^2\mathbf{B}\mathbf{B}^\top$ and $\Sigma = \Sigma_1 + \Sigma_2 = \alpha^2\mathbf{I}_k$. For each $i \in [m]$, let $\mathbf{x}_{2,i} = \mathbf{B}\mathbf{s}'_i \in \ker(\varphi_{\mathbf{h}})$, where $\mathbf{s}'_i$ denotes the i-th column of $\mathbf{S}'$. Since $\mathbf{s}'_i \leftarrow \mathcal{D}_{\mathcal{R}^{k-1},\alpha_0}$, we have $\mathbf{x}_{2,i} \sim \mathcal{D}_{\ker(\varphi_{\mathbf{h}}),\sqrt{\Sigma_2}}$. By Lemma 3.2, $\sqrt{\Sigma_1} \succeq \eta_{\varepsilon_2}(\ker(\varphi_{\mathbf{h}}))$. Applying Lemma 2.6 with $\Lambda_1 = \Lambda_2 = \ker(\varphi_{\mathbf{h}})$, $\mathbf{c}_1 = \mathbf{s}_i$, $\mathbf{c}_2 = \mathbf{0}$, $\mathbf{x}_2 = \mathbf{x}_{2,i}$ and $\mathbf{e} = \mathbf{r}_i$, and using the fact that $\Lambda_1 + \mathbf{c}_1 - \mathbf{x}_2 = \Lambda_1 + \mathbf{c}_1$ since $\mathbf{x}_2 \in \Lambda_1$, the sum $\mathbf{r}_i + \mathbf{x}_{2,i}$ where $\mathbf{r}_i \leftarrow \mathcal{D}_{\ker(\varphi_{\mathbf{h}})+\mathbf{s}_i,\sqrt{\Sigma_1}}$ is within statistical distance $8\varepsilon_2$ of:

$$\mathcal{D}_{\ker(\varphi_{\mathbf{h}})+\mathbf{s}_i,\sqrt{\Sigma_1+\Sigma_2}} = \mathcal{D}_{\ker(\varphi_{\mathbf{h}})+\mathbf{s}_i,\alpha}.$$

This matches the definition of $\tilde{\mathbf{s}}_i$ in G_1. Summing the statistical distance over the m columns using the independence of the columns of $\mathbf{S}'$ and $\mathbf{R}$, the total statistical distance is at most $8m\varepsilon_2$. □

Lemma 3.4. *If $\mathbf{T}' \leftarrow \mathcal{R}_{q'}^{(k-1)\times n}$, then the output of $\mathcal{B}$ is distributed exactly as G_2.*

Proof. The reduction $\mathcal{B}$ outputs $\mathbf{T} = \mathbf{B}\mathbf{T}' + \mathbf{R}\mathbf{A} \pmod{q'}$. Let $\mathbf{t}_j \in \mathcal{R}_{q'}^k$ denote the j-th column of $\mathbf{T}$ for $j \in [n]$. By definition of matrix multiplication, the j-th column is $\mathbf{t}_j = \mathbf{B}\mathbf{t}'_j + \sum_{i=1}^m a_{i,j}\mathbf{r}_i \pmod{q'}$, where $\mathbf{t}'_j \in \mathcal{R}_{q'}^{k-1}$ is the j-th column of $\mathbf{T}'$ and $a_{i,j}$ are entries of $\mathbf{A}$. Recall that each $\mathbf{r}_i$ was sampled from $\mathcal{D}_{\ker(\varphi_{\mathbf{h}})+\mathbf{s}_i,\sqrt{\Sigma_1}}$. Thus, we can write $\mathbf{r}_i = \mathbf{s}_i + \mathbf{w}_i$ for some $\mathbf{w}_i \in \ker(\varphi_{\mathbf{h}})$. Substituting this into the expression for $\mathbf{t}_j$:

$$\mathbf{t}_j = \mathbf{B}\mathbf{t}'_j + \sum_{i=1}^m a_{i,j}(\mathbf{s}_i + \mathbf{w}_i) \tag{20}$$

$$= \underbrace{\sum_{i=1}^m a_{i,j}\mathbf{s}_i}_{(\mathbf{S}\mathbf{A})_j} + \underbrace{\left(\mathbf{B}\mathbf{t}'_j + \sum_{i=1}^m a_{i,j}\mathbf{w}_i\right)}_{\mathbf{u}_j} \pmod{q'}. \tag{21}$$

By Lemma 2.14, $\mathbf{B}_{q'} = \mathbf{B} \bmod q'$ is a $\mathcal{R}_{q'}$-basis for $\ker(\varphi_{\mathbf{h},q'})$. Since $\mathbf{t}'_j$ is sampled uniformly from $\mathcal{R}_{q'}^{k-1}$, the term $\mathbf{B}\mathbf{t}'_j$ is distributed uniformly over $\ker(\varphi_{\mathbf{h},q'})$. For each j, the term $\mathbf{w}'_j = \sum_{i=1}^m a_{i,j}\mathbf{w}_i \pmod{q'}$ is an element of $\ker(\varphi_{\mathbf{h},q'})$ because each $\mathbf{w}_i \bmod q' \in \ker(\varphi_{\mathbf{h},q'})$. Thus, $\mathbf{u}_j = \mathbf{B}\mathbf{t}'_j + \mathbf{w}'_j$ is uniformly distributed over $\ker(\varphi_{\mathbf{h},q'})$. Furthermore, since the columns $\mathbf{t}'_j$ are independent for each j, the columns $\mathbf{u}_j$ are also independent. We conclude $\mathbf{T} = \mathbf{S}\mathbf{A} + \mathbf{U} \pmod{q'}$, where $\mathbf{U} \leftarrow [\ker(\varphi_{\mathbf{h},q'})]^m$. Recall that $\mathbf{A} = [\mathbf{I}_n; \mathbf{A}_2]$. Consider the $\mathcal{R}_{q'}$-linear map $f : [\ker(\varphi_{\mathbf{h},q'})]^m \to [\ker(\varphi_{\mathbf{h},q'})]^n$, with $\mathbf{U}' \mapsto \mathbf{U}'\mathbf{A}$. For any $\mathbf{V} \in [\ker(\varphi_{\mathbf{h},q'})]^n$, choosing $\mathbf{U}' = [\mathbf{V} \mid \mathbf{0}]$ yields $f(\mathbf{U}') = \mathbf{V}$, hence f is surjective. Therefore, if $\mathbf{U}' \leftarrow [\ker(\varphi_{\mathbf{h},q'})]^m$ is uniform, then $\mathbf{U}'\mathbf{A} \leftarrow [\ker(\varphi_{\mathbf{h},q'})]^n$ is uniform. Thus, $\mathbf{T} = \mathbf{S}\mathbf{A} + \mathbf{U}'\mathbf{A} \pmod{q'}$ is distributed exactly as in G_2. □

Lemma 3.5 (Sampleability). *Let $q', k, \alpha_H \in \mathbb{N}^+$ such that $k > 1$ and let $\alpha, \alpha_0 \in \mathbb{R}$. Let $\mathbf{h} = [1; \mathbf{h}']$ with $\mathbf{h}' \in (\mathcal{T}_{\alpha_H})^{k-1}$. By Lemma 2.14, $\mathbf{B} := \begin{pmatrix} -(\mathbf{h}')^\top \\ \mathbf{I}_{k-1} \end{pmatrix} \in \mathcal{R}^{k\times(k-1)}$ is an $\mathcal{R}$-basis of $\Lambda = \ker(\varphi_{\mathbf{h}})$. If $\alpha^2 \geq \alpha_0^2\|\mathbf{B}\|_2^2 + (1+\alpha_H)\cdot\frac{\ln(2(k-1)d+4)}{\pi}$, then there exists a PPT algorithm to sample $\mathbf{r}_i \leftarrow \mathcal{D}_{\Lambda+\mathbf{s}_i,\sqrt{\Sigma_1}}$ for $\mathbf{s}_i \in \mathcal{R}^k$.*

Proof. The rank of the lattice Λ is $n = (k-1)d$. Let $L = \frac{\ln(2n+4)}{\pi}$. We now verify the sampleability condition. By the Rayleigh quotient and (17), $L\max_j \|\Sigma_1^{-1/2}\mathbf{b}_j\|_2^2 = L\max_j\left(\mathbf{b}_j^\top\Sigma_1^{-1}\mathbf{b}_j\right) \leq L\frac{\max_j\|\mathbf{b}_j\|_2^2}{\lambda_{\min}(\Sigma_1)} \leq 1$, which implies efficient sampleability by Lemma 2.11. □

4 Key-Homomorphic One-Time Signature

In this section, we first provide the syntax and definition of Key-Homomorphic One-Time Signature (KOTS) and its properties. We then present our construction and analyze its properties.

4.1 Definition

Definition 4.1 (Key-Homomorphic One-Time Signature (KOTS) [16]). *Let $\mathcal{M}$ be a message space and $\mathcal{S}_{\mathsf{sk}}$ be a secret key space. Let $\mathcal{S}_{\mathsf{pk}}$ and $\mathcal{S}_{\mathsf{sig}}$ be $\mathcal{R}$-modules representing the public key and signature spaces. A key-homomorphic one-time signature (KOTS) scheme Π_{KOTS} over $\mathcal{R}$ consists of six PPT algorithms:*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: On input a security parameter λ, the setup algorithm outputs public parameters* pp.
- $(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*: On input public parameters* pp*, the key generation algorithm outputs a one-time signing key* $\mathsf{osk} \in \mathcal{S}_{\mathsf{sk}}$ *and the corresponding public key* $\mathsf{opk} \in \mathcal{S}_{\mathsf{pk}}$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{osk}, \mu)$*: On input public parameters* pp*, a signing key* $\mathsf{osk} \in \mathcal{S}_{\mathsf{sk}}$*, and a message $\mu \in \mathcal{M}$, the signing algorithm outputs a signature $\sigma \in \mathcal{S}_{\mathsf{sig}}$.*
- $b \leftarrow \mathsf{iVrfy}/\mathsf{sVrfy}/\mathsf{wVrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma)$*: On input public parameters* pp*, a verification key* $\mathsf{opk} \in \mathcal{S}_{\mathsf{pk}}$*, a message $\mu \in \mathcal{M}$, and a signature $\sigma \in \mathcal{S}_{\mathsf{sig}}$, the individual/strong/weak verification algorithm outputs a bit b indicating acceptance or rejection.*

Definition 4.2 (Individual Correctness). *Let $\epsilon_{\mathsf{cor}} \in (0,1)$. We say that Π_{KOTS} is ϵ_{cor}-individually correct, if for all $\lambda \in \mathbb{N}^+$ and $\mu \in \mathcal{M}$, it holds that $\Pr\left[\mathsf{iVrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma) = 1\right] \geq 1 - \epsilon_{\mathsf{cor}}$, where $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$, and $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{osk}, \mu)$.*

Definition 4.3 (Probabilistic Homomorphism). *Let $N \in \mathbb{N}^+$, $\epsilon_{\mathsf{hom}} \in (0,1)$, and $W \subseteq \mathcal{R}$. We say that Π_{KOTS} is $(N, W, \epsilon_{\mathsf{hom}})$-probabilistically homomorphic, if for all $\lambda \in \mathbb{N}^+$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mu \in \mathcal{M}$, and $(\mathsf{opk}_i, \sigma_i)_{i=1}^N \in \mathcal{S}_{\mathsf{pk}} \times \mathcal{S}_{\mathsf{sig}}$ with $\mathsf{iVrfy}(\mathsf{pp}, \mathsf{opk}_i, \mu, \sigma_i) = 1$,*

Game $\mathsf{EUF\text{-}RK}^{W'}_{\Pi_{\mathsf{KOTS}},\mathcal{A}}(\lambda)$

(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$
(2) $\forall i \in [N]$: $(\mathsf{osk}_i, \mathsf{opk}_i) \leftarrow \mathsf{KGen}(\mathsf{pp})$ and set $Q_i \leftarrow \emptyset$
(3) $(i^*, \mu^*, \sigma^*, w^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(\cdot,\cdot)}(\mathsf{pp}, \mathsf{opk}_1, \ldots, \mathsf{opk}_N)$
(4) Output 1 iff $i^* \in [N] \wedge \mu^* \notin Q_{i^*} \wedge w^* \in W' \wedge \mathsf{wVrfy}\left(\mathsf{pp}, w^* \cdot \mathsf{opk}_{i^*}, \mu^*, \sigma^*\right) = 1$

Oracle $\mathsf{Sign}(i, \mu)$

(1) If $i \notin [N]$ or $|Q_i| = 1$ then return $\perp$
(2) Set $Q_i \leftarrow Q_i \cup \{\mu\}$
(3) Return $\mathsf{Sign}(\mathsf{pp}, \mathsf{osk}_i, \mu)$

Figure 2: EUF-RK Game for a KOTS Scheme.

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

Definition 4.4 (Robust Homomorphism). *Let* Π_{KOTS} *be a* KOTS *scheme. We say that* Π_{KOTS} *is robustly homomorphic, if for all* $\lambda \in \mathbb{N}^+$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mu \in \mathcal{M}$, *and any* $\mathsf{opk}_b \in \mathcal{S}_{\mathsf{pk}}$ *and* $\sigma_b \in \mathcal{S}_{\mathsf{sig}}$ $(b \in \{0,1\})$ *satisfying* $\mathsf{sVrfy}(\mathsf{pp}, \mathsf{opk}_b, \mu, \sigma_b) = 1$, *it holds that* $\mathsf{wVrfy}(\mathsf{pp}, \mathsf{opk}_0 - \mathsf{opk}_1, \mu, \sigma_0 - \sigma_1) = 1$.

Definition 4.5 (Multi-User Existential Unforgeability under Rerandomized Keys). *Let* $W' := \{w_0 - w_1 : w_0, w_1 \in W \wedge w_0 \neq w_1\}$. *Consider the game in Figure 2. We say that* Π_{KOTS} *is* W'*-existentially unforgeable under rerandomized keys (EUF-RK), if for all* $\lambda \in \mathbb{N}^+$ *and PPT adversaries* $\mathcal{A}$, *it holds that* $\mathsf{Adv}^{\mathsf{EUF\text{-}RK}}_{\Pi_{\mathsf{KOTS}}}(\mathcal{A}) := \Pr[\mathsf{EUF\text{-}RK}^{W'}_{\Pi_{\mathsf{KOTS}},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

4.2 KOTS Construction

We now present our KOTS construction in Figure 3.

Definition 4.6. *Let* $q', d, m, n, k, \alpha_H, \beta_z, \beta_\sigma \in \mathbb{N}^+$ *where* d *is a power of* 2 *and* $k > 1$. *Let* $\alpha \in \mathbb{R}$. *Let* $\mathsf{H} : \mathcal{M} \to (\mathcal{T}_{\alpha_H})^{k-1}$ *be a collision-resistant hash function. We define* $\Pi_{\mathsf{KOTS}} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{iVrfy}, \mathsf{sVrfy}, \mathsf{wVrfy})$ *as the key-homomorphic one-time signature scheme over* $\mathcal{R}$ *as in Figure 3. Its secret key space is* $\mathcal{S}_{\mathsf{sk}} \subseteq \mathcal{R}^{k \times m}$, *public key space is* $\mathcal{S}_{\mathsf{pk}} \subseteq \mathcal{R}_{q'}^{k \times n}$, *and its signature space is* $\mathcal{S}_{\mathsf{sig}} \subseteq \mathcal{R}^m$.

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,* $(\frac{1+\varepsilon_3}{1-\varepsilon_3})^{2m} p^{-dm/t_2} < 2^{-\lambda}$, *and* $k \geq 4$. *Let* $\alpha, \epsilon_{\mathsf{cor}}, \epsilon_{\mathsf{hom}}, \varepsilon_1, \varepsilon_3 \in \mathbb{R}$, *such that* $\alpha \geq \sqrt{2}\eta_{\varepsilon_1}(\mathbb{Z})$, $\alpha \geq p\sqrt{2\alpha_H \ln(2(k-1)d(1+1/\varepsilon_3))/\pi}$, $\beta_z \geq 6\alpha\sqrt{1+(k-1)\alpha_H}$, *and* $\epsilon_{\mathsf{cor}} \geq md \cdot \left(2^{-162} + 2(k-1)\alpha_H\varepsilon_1\right)$, $\beta_\sigma \geq \beta_z \cdot \sqrt{2\alpha_w N \cdot \ln\left(\frac{2md}{\epsilon_{\mathsf{hom}}}\right)}$. *Let* $\mathsf{H}\colon \mathcal{M} \to (\mathcal{T}_{\alpha_H})^{k-1}$ *be a random oracle with* $\left|\mathcal{T}_{\alpha_H}\right|^{k-3} \geq 2^{2\lambda}$. *Let* $W' = \{w_0 - w_1 \mid w_0, w_1 \in \mathcal{T}_{\alpha_w} \wedge w_0 \neq w_1\}$. *If the* $\mathsf{DualHintMLWE}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$ *and* $\mathsf{MSIS}^\infty_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}$ *problems are hard, and any* $w^* \in W'$ *is a unit in* $\mathcal{R}_{q'}$, *then the construction* Π_{KOTS} *in Figure 3 is an* ϵ_{cor}*-individually correct,* $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\mathsf{hom}})$*-probabilistically homomorphic, robustly homomorphic KOTS that is* W'*-multi-user existentially unforgeable under rerandomized keys.*

Proof. We prove the theorem via Lemmas 4.1, 4.2, 4.3 and 4.4. □

$\underline{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)}$
(1) $\mathbf{A}_2 \leftarrow \mathcal{R}_{q'}^{(m-n)\times n}$
(2) **return** $\mathsf{pp} = \mathbf{A} = [\mathbf{I}_n; \mathbf{A}_2]$

$\underline{(\mathsf{osk}, \mathsf{opk}) \leftarrow \mathsf{KGen}(\mathsf{pp})}$
(1) $\mathbf{S} \leftarrow \mathcal{D}_{\mathcal{R}^{k\times m},\alpha}$
(2) $\mathbf{T} = \mathbf{SA} \pmod{q'} \in \mathcal{R}_{q'}^{k\times n}$
(3) **return** $(\mathsf{osk} = \mathbf{S}, \mathsf{opk} = \mathbf{T})$

$\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{osk}, \mu)}$
(1) $\mathbf{h}' = \mathsf{H}(\mu) \in (\mathcal{T}_{\alpha_H})^{k-1}$
(2) $\mathbf{h} = [1, \mathbf{h}'] \in \mathcal{R}^k$
(3) $\mathbf{z}^\top = \mathbf{h}^\top\mathbf{S}$
(4) **return** $\sigma = \mathbf{z} \in \mathcal{R}^m$

$\underline{b \leftarrow \mathsf{iVrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma)}$
(1) **return** $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma, \beta_z)$

$\underline{b \leftarrow \mathsf{sVrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma)}$
(1) **return** $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma, \beta_\sigma)$

$\underline{b \leftarrow \mathsf{wVrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma)}$
(1) **return** $\mathsf{Vrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma, 2\beta_\sigma)$

$\underline{b \leftarrow \mathsf{Vrfy}(\mathsf{pp}, \mathsf{opk}, \mu, \sigma, \beta)}$
(1) Parse $\mathsf{pp} = \mathbf{A}$, $\mathsf{opk} = \mathbf{T}$ and $\sigma = \mathbf{z}$
(2) $\mathbf{h}' = \mathsf{H}(\mu)$, and $\mathbf{h} = [1; \mathbf{h}']$
(3) **if** $\|\mathbf{z}\|_\infty > \beta$ **or** $\mathbf{z}^\top\mathbf{A} \neq \mathbf{h}^\top\mathbf{T} \pmod{q'}$, **return** 0, **else return** 1

Figure 3: Our KOTS Construction Π_{KOTS}.

Lemma 4.1. *Let* $\lambda, q', d, m, n, k, \alpha_H, \beta_z \in \mathbb{N}^+$ *where* d *is a power of* 2 *and* $k > 1$. *Let* $\alpha, \epsilon_{\mathsf{cor}} \in \mathbb{R}$ *and* $\varepsilon_1 \in (0, \frac{1}{2})$, *such that* $\alpha \geq \sqrt{2}\eta_{\varepsilon_1}(\mathbb{Z})$, $\beta_z \geq 6\alpha\sqrt{1+(k-1)\alpha_H}$, *and* $\epsilon_{\mathsf{cor}} \geq md \cdot \left(2^{-162} + 2(k-1)\alpha_H\varepsilon_1\right)$. *Let* $\mathsf{H} : \mathcal{M} \to (\mathcal{T}_{\alpha_H})^{k-1}$ *be a hash function. Then* Π_{KOTS} *in Figure 3 is* ϵ_{cor}*-individually correct.*

Proof. Let $\mathsf{pp} = \mathbf{A} \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{osk}, \mathsf{opk}) = (\mathbf{S}, \mathbf{T}) \leftarrow \mathsf{KGen}(\mathsf{pp})$, so that $\mathbf{T} = \mathbf{SA}$. For any $\mu \in \mathcal{M}$, the signer computes $\mathbf{h}' = \mathsf{H}(\mu)$, forms $\mathbf{h} = [1, \mathbf{h}']$, and outputs $\sigma^\top = \mathbf{z}^\top = \mathbf{h}^\top\mathbf{S}$. The algebraic check always holds since $\mathbf{z}^\top\mathbf{A} = (\mathbf{h}^\top\mathbf{S})\mathbf{A} = \mathbf{h}^\top\mathbf{T} \pmod{q'}$.

We show that $\Pr[\|\mathbf{z}\|_\infty > \beta_z] \leq \epsilon_{\mathsf{cor}}$. Fix indices $(c, i) \in [m] \times [\![d]\!]$. Since $\mathbf{z}^\top = \mathbf{h}^\top\mathbf{S}$, the i-th coefficient of the c-th entry z_c expands as $z_c^{(i)} = s_{1,c}^{(i)} + \sum_{j=1}^{k-1} (h'_j s_{1+j,c})^{(i)}$. Since $h'_j \in \mathcal{T}_{\alpha_H}$, the coefficient $(h'_j s_{1+j,c})^{(i)}$ is a signed sum of exactly α_H coefficients of $s_{1+j,c}$. Since the coefficients of $\mathbf{S}$ are sampled independently from $\mathcal{D}_{\mathbb{Z},\alpha}$, we conclude that $z_c^{(i)}$ is distributed as the sum of $1 + (k-1)\alpha_H$ independent samples from $\mathcal{D}_{\mathbb{Z},\alpha}$.

Since $\alpha \geq \sqrt{2}\eta_{\varepsilon_1}(\mathbb{Z})$, Lemma 2.4 bounds the statistical distance between $z_c^{(i)}$ and $\mathcal{D}_{\mathbb{Z},\alpha\sqrt{1+(k-1)\alpha_H}}$ by $2(k-1)\alpha_H\varepsilon_1$. Furthermore, because $\beta_z \geq 6\alpha\sqrt{1+(k-1)\alpha_H}$, Lemma 2.5 ensures a sample from this Gaussian exceeds β_z with probability at most 2^{-162}. Combining these, $\Pr[|z_c^{(i)}| > \beta_z] \leq 2^{-162} + 2(k-1)\alpha_H\varepsilon_1$. Taking a union bound over all md coefficients in $\mathbf{z}$ yields $\Pr[\|\mathbf{z}\|_\infty > \beta_z] \leq md\left(2^{-162} + 2(k-1)\alpha_H\varepsilon_1\right) \leq \epsilon_{\mathsf{cor}}$, as required. □

Lemma 4.2. *Let* $\lambda, q', d, m, n, k, \beta_z, \beta_\sigma, \alpha_H, \alpha_w, N \in \mathbb{N}^+$, *where* d *is a power of* 2, $k > 1$, *and* β_z *be as in Lemma 4.1. Let* $\epsilon_{\mathsf{hom}} \in (0,1)$ *such that* $\beta_\sigma \geq \beta_z\sqrt{2\alpha_w N \ln(\frac{2md}{\epsilon_{\mathsf{hom}}})}$. *Let* $\mathsf{H}\colon \mathcal{M} \to (\mathcal{T}_{\alpha_H})^{k-1}$ *be a hash function, and* $W = \mathcal{T}_{\alpha_w} \subseteq \mathcal{R}$. *Then* Π_{KOTS} *in Figure 3 is* $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\mathsf{hom}})$*-probabilistically homomorphic.*

Proof. Let $\mathsf{pp} = \mathbf{A} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mu \in \mathcal{M}$, and for $i \in [N]$, $\mathsf{iVrfy}(\mathsf{pp}, \mathsf{opk}_i, \mu, \sigma_i) = 1$. Let $(\mathsf{opk}_i, \sigma_i) = (\mathbf{T}_i, \mathbf{z}_i)$, then for each i, $\|\mathbf{z}_i\|_\infty \leq \beta_z$ and $\mathbf{z}_i^\top\mathbf{A} = \mathbf{h}^\top\mathbf{T}_i \pmod{q'}$, where $\mathbf{h} = [1, \mathsf{H}(\mu)]$.

Sample weights $w_1, \ldots, w_N \leftarrow W$. We define the aggregated public key $\mathsf{opk}^* = \mathbf{T}^* = \sum_{i=1}^N w_i\mathbf{T}_i$ and the aggregated signature

$\sigma^* = \mathbf{z}^* = \sum_{i=1}^N w_i \mathbf{z}_i$. By linearity,

$$(\mathbf{z}^*)^\top \mathbf{A} = \left(\sum_{i=1}^N w_i \mathbf{z}_i^\top\right)\mathbf{A} = \sum_{i=1}^N w_i(\mathbf{z}_i^\top \mathbf{A}) = \sum_{i=1}^N w_i(\mathbf{h}^\top \mathbf{T}_i) \tag{22}$$

$$= \mathbf{h}^\top \left(\sum_{i=1}^N w_i \mathbf{T}_i\right) = \mathbf{h}^\top \mathbf{T}^* \pmod{q'}. \tag{23}$$

Thus, the algebraic check in $\mathsf{sVrfy}(\mathsf{pp}, \mathsf{opk}^*, \mu, \sigma^*)$ holds.

It remains to bound the probability that $\|\mathbf{z}^*\|_\infty > \beta_\sigma$. Let $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,m})$ where each $z_{i,c} \in \mathcal{R}$. Since $\mathsf{iVrfy}(\mathsf{pp}, \mathsf{opk}_i, \mu, \sigma_i) = 1$, we have $\|z_{i,c}\|_\infty \le \beta_z$ for all $i \in [N]$ and $c \in [m]$. For any fixed c, we treat the coefficients of the polynomial $\sum_{i=1}^N w_i z_{i,c}$ as a weighted sum. Applying Lemma 2.3 with $\zeta = \beta_\sigma/\beta_z$, we have:

$$\Pr_{w_i \leftarrow W}\left[\left\|\sum_{i=1}^N w_i z_{i,c}\right\|_\infty > \beta_\sigma\right] \le 2d \exp\left(-\frac{\beta_\sigma^2}{2\alpha_w N \beta_z^2}\right). \tag{24}$$

By applying a union bound over all m polynomial entries of the vector $\mathbf{z}^*$, the probability that the signature exceeds the norm bound β_σ is at most $2md \exp\left(-\frac{\beta_\sigma^2}{2\alpha_w N \beta_z^2}\right)$. Substituting the assumed lower bound for β_σ, this failure probability is at most ϵ_{hom}, as required. □

Lemma 4.3. *Let $\lambda, q', d, m, n, k, \beta_\sigma, \alpha_H, N \in \mathbb{N}^+$ where d is a power of 2 and $k > 1$. Let $\mathsf{H}\colon \mathcal{M} \to (\mathcal{T}_{\alpha_H})^{k-1}$ be a hash function. Then Π_{KOTS} in Figure 3 is robustly homomorphic.*

Proof. Fix $\lambda \in \mathbb{N}$, $\mathsf{pp} = \mathbf{A} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mu \in \mathcal{M}$, and let the pairs $(\mathsf{opk}_b, \sigma_b) = (\mathbf{T}_b, \mathbf{z}_b)$ for $b \in \{0,1\}$ satisfy the strong verification $\mathsf{sVrfy}(\mathsf{pp}, \mathbf{T}_b, \mu, \mathbf{z}_b) = 1$. Let $\mathbf{h}' = \mathsf{H}(\mu)$ and $\mathbf{h} = [1, \mathbf{h}']$. By the definition of sVrfy, for $b \in \{0,1\}$, it holds that:

$$\mathbf{z}_b^\top \mathbf{A} = \mathbf{h}^\top \mathbf{T}_b \pmod{q'} \quad \text{and} \quad \|\mathbf{z}_b\|_\infty \le \beta_\sigma. \tag{25}$$

By linearity,

$$(\mathbf{z}_0 - \mathbf{z}_1)^\top \mathbf{A} = \mathbf{z}_0^\top \mathbf{A} - \mathbf{z}_1^\top \mathbf{A} \tag{26}$$

$$= \mathbf{h}^\top \mathbf{T}_0 - \mathbf{h}^\top \mathbf{T}_1 \tag{27}$$

$$= \mathbf{h}^\top (\mathbf{T}_0 - \mathbf{T}_1) \pmod{q'}. \tag{28}$$

Furthermore,

$$\|\mathbf{z}_0 - \mathbf{z}_1\|_\infty \le \|\mathbf{z}_0\|_\infty + \|\mathbf{z}_1\|_\infty \le \beta_\sigma + \beta_\sigma = 2\beta_\sigma. \tag{29}$$

Combining (28) and (29), it follows that $\mathsf{wVrfy}(\mathsf{pp}, \mathbf{T}_0 - \mathbf{T}_1, \mu, \mathbf{z}_0 - \mathbf{z}_1) = 1$, as required. □

Lemma 4.4. *Let $\lambda, t_1, t_2, d, p, q', m, n, k, \beta_\sigma, \alpha_H, \alpha_w, N \in \mathbb{N}^+$ and $\alpha, \varepsilon_3 \in \mathbb{R}$ such that $d > t_1 > 1$ and $d > t_2 > 1$ are powers of 2, $p \equiv 2t_1 + 1 \pmod{4t_1}$ is prime, $q' \equiv 2t_2 + 1 \pmod{4t_2}$ is prime, $k \ge 4$, $\alpha \ge p\sqrt{2\alpha_H \ln(2(k-3)d(1+1/\varepsilon_3))/\pi}$, and $(\frac{1+\varepsilon_3}{1-\varepsilon_3})^{2m} p^{-dm/t_1} < 2^{-\lambda}$. Let $\mathsf{H}\colon \mathcal{M} \to (\mathcal{T}_{\alpha_H})^{k-1}$ be a random oracle with $|\mathcal{T}_{\alpha_H}|^{k-3} \ge 2^{2\lambda}$. If the $\mathsf{DualHintMLWE}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$ and $\mathsf{MSIS}^\infty_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}$ problems are hard, and any $w^* \in W'$ is a unit in $\mathcal{R}_{q'}$, then Π_{KOTS} is W'-EUF-RK secure in the ROM. Specifically, for any PPT adversary $\mathcal{A}$ making Q_H hash queries, there exist PPT algorithms $\mathcal{B}$ and $\mathcal{D}$ such that:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}RK}}_{\Pi_{\mathsf{KOTS}}}(\mathcal{A}) \le N(Q+1)^2\Big(\mathsf{Adv}^{\mathsf{DualHintMLWE}}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}(\mathcal{B}) + \mathsf{Adv}^{\mathsf{MSIS}^\infty}_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}(\mathcal{D})\Big) + \frac{Q_H^2}{|\mathcal{T}_{\alpha_H}|^{k-3}} + \mathsf{negl}(\lambda),$$

where $Q = Q_H + N$ is the total number of distinct messages queried.

Proof. We proceed via a sequence of games $\mathrm{G}_0, \mathrm{G}_1, \mathrm{G}_2, \mathrm{G}_3$. Let $p_i = \Pr[\mathrm{G}_i = 1]$ denote the probability that the adversary wins in Game i.

Game G_0. This is the original EUF-RK game. The challenger generates N key pairs, answers hash queries, and provides at most one signature per key. By definition, $p_0 = \mathsf{Adv}^{\mathsf{EUF\text{-}RK}}_{\Pi_{\mathsf{KOTS}}}(\mathcal{A})$.

Game G_1. This game is identical to G_0, except the challenger tracks all Q_H queries made to the random oracle H. For any $\mathbf{h}^{(i)}$, partition it as $[1, h_2^{(i)}, \overline{\mathbf{h}}^{(i)}]$, where $\overline{\mathbf{h}}^{(i)}$ has $k-2$ entries. For $i \in \{1,2\}$, let $\overline{\mathbf{h}}^{(i)} = [\overline{h}_1^{(i)}, \ldots, \overline{h}_{k-2}^{(i)}]$ and define $g_j^{(i)} = \overline{h}_j^{(i)}/\overline{h}_1^{(i)}$ for $j \in \{2, \ldots, k-2\}$. The challenger aborts if there exists any pair of distinct queries with outputs $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$ such that $g_j^{(2)} = g_j^{(1)}$ for all $j \in \{2, \ldots, k-2\}$.

Claim 4.1. $|p_0 - p_1| \le \frac{Q_H^2}{|\mathcal{T}_{\alpha_H}|^{k-3}}$.

Proof. Since H is modeled as a random oracle, each query yields an independent, uniformly random vector in $\mathcal{T}_{\alpha_H}^{k-2}$ for the last $k-2$ entries. For any fixed pair of queries, the relation $g_j^2 = g_j^1$ holds for a specific j with probability at most $1/|\mathcal{T}_{\alpha_H}|$. The probability it holds for entries $j \in \{2, \ldots, k-2\}$ simultaneously is at most $1/|\mathcal{T}_{\alpha_H}|^{k-3}$. Applying a union bound over all $\le Q_H^2$ pairs of queries gives the claimed bound. Conditioned on not aborting, the game is identical to G_0. □

Game G_2. This game is identical to G_1, except that the challenger guesses the target user $i_0 \leftarrow [N]$, the hash query index $j_1 \in [Q]$ corresponding to the signing query, and the index $j_2 \in [Q]$ corresponding to the forgery. It aborts if $\mathcal{A}$'s actions do not match these guesses.

Claim 4.2. $p_2 \ge \frac{1}{N(Q+1)^2}\, p_1$.

Proof. There are at most N choices for the target user i^*, and at most $Q+1$ possible query positions for the signing message and the forged message. With probability $1/(N(Q+1)^2)$, the guesses match the adversary's behavior. Conditioned on a correct guess, the simulation is perfectly identical to G_1. □

Game G_3. This game is identical to G_2, except that the challenger alters how the target key $\mathbf{T}^{i_0}$ is generated. It samples $\mathbf{A}_2 \leftarrow \mathcal{R}_{q'}^{(m-n)\times n}$, $\mathbf{S} \leftarrow \mathcal{D}_{\mathcal{R}^{k\times m},\alpha}$, sets $\mathbf{A} = [\mathbf{I}_n; \mathbf{A}_2]$, and programs the hashes $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}$ at j_1, j_2. Using Lemma 4.6, it samples $\mathbf{U}$ with $(\mathbf{h}^{(1)})^\top \mathbf{U} = \mathbf{0}$ and $(\mathbf{h}^{(2)})^\top \mathbf{U} = \mathbf{0}$, sets $\mathbf{T} = (\mathbf{S} + \mathbf{U})\mathbf{A}$, and answers the signing query with $\mathbf{z}_1^\top = (\mathbf{h}^{(1)})^\top \mathbf{S}$.

Claim 4.3. *There exists a PPT distinguisher $\mathcal{B}$ such that $|p_2 - p_3| = \mathsf{Adv}^{\mathsf{DualHintMLWE}}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}(\mathcal{B})$.*

Proof. On input a Dual Hint-MLWE challenge $(\widetilde{\mathbf{A}}, \widetilde{\mathbf{T}}, \widetilde{\mathbf{h}}, \widetilde{\mathbf{z}})$, the algorithm $\mathcal{B}$ sets $(\mathbf{A}, \mathbf{T}, \mathbf{h}^{(1)}, \mathbf{z}_1) = (\widetilde{\mathbf{A}}, \widetilde{\mathbf{T}}, \widetilde{\mathbf{h}}, \widetilde{\mathbf{z}})$ and simulates the exact environment described in G_2 or G_3. If $\mathcal{B}$ receives a Case 0 Dual Hint-MLWE challenge ($\mathbf{T} = \mathbf{SA}$), the simulated view corresponds to G_2. If it receives Case 1 challenge ($\mathbf{T} = (\mathbf{S} + \mathbf{U})\mathbf{A}$), the view corresponds to G_3. The distinguishing advantage is exactly $|p_2 - p_3|$. □

Claim 4.4. *Assume w^* is a unit in $\mathcal{R}_{q'}$. There exists a PPT* MSIS *solver $\mathcal{D}$ such that $p_3 \le \mathsf{Adv}^{\mathsf{MSIS}^\infty}_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}(\mathcal{D}) + \mathsf{negl}(\lambda)$.*

Proof. Algorithm $\mathcal{D}$ receives an MSIS challenge $\mathbf{A}' \leftarrow \mathcal{R}_{q'}^{n\times m}$. It initiates the G_3 challenger by setting $\mathbf{A} = (\mathbf{A}')^\top \in \mathcal{R}_{q'}^{m\times n}$. It follows the G_3 procedure: sampling $\mathbf{S}$, programming $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}$, and sampling $\mathbf{U}$ from their joint kernel. If $\mathcal{A}$ outputs a successful forgery $(i^*, \mu^*, \mathbf{z}^*, w^*)$, $\mathcal{D}$ computes $\mathbf{x} = ((\mathbf{z}^*)^\top - w^*(\mathbf{h}^{(2)})^\top\mathbf{S})^\top$. If $\mathbf{x} \neq \mathbf{0}$, then $\mathcal{D}$ outputs $\mathbf{x}$, otherwise it aborts.

First, $\mathcal{D}$'s simulation is perfect. The verification equation $\mathbf{z}_1^\top\mathbf{A} = (\mathbf{h}^{(1)})^\top\mathbf{T} \pmod{q'}$ holds since $(\mathbf{h}^{(1)})^\top\mathbf{U} = \mathbf{0} \pmod{q'}$, making the view identical to G_3. Because $\mathcal{A}$ wins, verification passes: $(\mathbf{z}^*)^\top\mathbf{A} = (\mathbf{h}^{(2)})^\top(w^*\mathbf{T}) \pmod{q'}$ and $\|\mathbf{z}^*\|_\infty \le 2\beta_\sigma$. Since w^* is a scalar and $(\mathbf{h}^{(2)})^\top\mathbf{U} = \mathbf{0} \pmod{q'}$, substituting $\mathbf{T}$ yields:

$$\mathbf{x}^\top\mathbf{A} = ((\mathbf{z}^*)^\top - w^*(\mathbf{h}^{(2)})^\top\mathbf{S})\mathbf{A} \tag{30}$$

$$= (\mathbf{h}^{(2)})^\top(w^*\mathbf{T}) - w^*(\mathbf{h}^{(2)})^\top\mathbf{S}\mathbf{A} = \mathbf{0} \pmod{q'}. \tag{31}$$

Taking the transpose gives $\mathbf{A}'\mathbf{x} = \mathbf{0} \pmod{q'}$. Also,

$$\|\mathbf{x}\|_\infty \le \|\mathbf{z}^*\|_\infty + \|w^*\|_1 \cdot \|(\mathbf{h}^{(2)})^\top\mathbf{S}\|_\infty \le 2\beta_\sigma + 2\alpha_w\beta_z. \tag{32}$$

The remaining failure condition is $\mathbf{x} = \mathbf{0}$. We show this happens with negligible probability. Partition $\mathbf{S} \in \mathcal{R}^{k\times m}$ into $[\mathbf{s}_1^\top; \mathbf{s}_2^\top; \overline{\mathbf{S}}]$ where $\mathbf{s}_1^\top, \mathbf{s}_2^\top \in \mathcal{R}^{1\times m}$ and $\overline{\mathbf{S}} \in \mathcal{R}^{(k-2)\times m}$. Correspondingly, for $i \in \{1,2\}$, we partition the programmed hashes $\mathbf{h}^{(i)} = [1, h_2^{(i)}, \overline{\mathbf{h}}^{(i)}] \in \mathcal{R}^k$ with $h_2^{(i)} \in \mathcal{R}$ and $\overline{\mathbf{h}}^{(i)} \in \mathcal{R}^{k-2}$. The adversary's view includes $\mathbf{T} = (\mathbf{S}+\mathbf{U})\mathbf{A}$ and the signature $\mathbf{z}_1^\top = (\mathbf{h}^{(1)})^\top\mathbf{S}$. By Lemma 4.6, $\mathbf{U}$ is sampled such that it is uniform in the joint right kernels of $(\mathbf{h}^{(1)})^\top$ and $(\mathbf{h}^{(2)})^\top$. Specifically, the last $k-2$ rows of $\mathbf{U}$ are uniformly random from $\mathcal{R}_{q'}^{k-2}$ and provide enough degrees of freedom to statistically mask $\overline{\mathbf{S}}$ given $\mathbf{T}$. Conditioned on the adversary's view, the probability that the forgery satisfies the zero-condition is:

$$\Pr_{\mathbf{S}\leftarrow\mathcal{D}_{\mathcal{R}^{k\times m},\alpha}}[\mathbf{x} = \mathbf{0} \mid (\mathbf{h}^{(1)})^\top\mathbf{S} = \mathbf{z}_1^\top \wedge \mathbf{T} = (\mathbf{U}+\mathbf{S})\mathbf{A}] \tag{33}$$

$$= \Pr_{\mathbf{S}\leftarrow\mathcal{D}_{\mathcal{R}^{k\times m},\alpha}}[w^*(\mathbf{h}^{(2)})^\top\mathbf{S} = (\mathbf{z}^*)^\top \mid (\mathbf{h}^{(1)})^\top\mathbf{S} = \mathbf{z}_1^\top \wedge \mathbf{T} = (\mathbf{U}+\mathbf{S})\mathbf{A}] \tag{34}$$

$$= \Pr_{\mathbf{S}\leftarrow\mathcal{D}_{\mathcal{R}^{k\times m},\alpha}}[(\mathbf{h}^{(2)})^\top\mathbf{S} = (w^*)^{-1}(\mathbf{z}^*)^\top \mid (\mathbf{h}^{(1)})^\top\mathbf{S} = \mathbf{z}_1^\top \wedge \mathbf{T} = (\mathbf{U}+\mathbf{S})\mathbf{A}] \tag{35}$$

$$\le \max_{\mathbf{y}^\top} \Pr_{\mathbf{S}\leftarrow\mathcal{D}_{\mathcal{R}^{k\times m},\alpha}}[(\mathbf{h}^{(2)})^\top\mathbf{S} = \mathbf{y}^\top \mid (\mathbf{h}^{(1)})^\top\mathbf{S} = \mathbf{z}_1^\top \wedge \mathbf{T} = (\mathbf{U}+\mathbf{S})\mathbf{A}] \tag{36}$$

$$\le \max_{\mathbf{y}^\top} \Pr_{\mathbf{S}\leftarrow\mathcal{D}_{\mathcal{R}^{k\times m},\alpha}}[(\mathbf{h}^{(2)})^\top\mathbf{S} = \mathbf{y}^\top \mid (\mathbf{h}^{(1)})^\top\mathbf{S} = \mathbf{z}_1^\top \wedge \mathbf{V} = \mathbf{U}+\mathbf{S}] \tag{37}$$

$$\le \max_{\overline{\mathbf{y}}^\top} \Pr_{\overline{\mathbf{S}}\leftarrow\mathcal{D}_{\mathcal{R}^{(k-2)\times m},\alpha}}[(\overline{\mathbf{h}}^{(2)})^\top\overline{\mathbf{S}} = \overline{\mathbf{y}}^\top \mid (\overline{\mathbf{h}}^{(1)})^\top\overline{\mathbf{S}} = \overline{\mathbf{z}}_1^\top] \tag{38}$$

$$\le \left(\frac{1+\varepsilon_3}{1-\varepsilon_3}\right)^{2m} p^{-dm/t_1} < 2^{-\lambda}. \tag{39}$$

In the above derivation, (35) holds since w^* is a unit. In (37), $\mathbf{V}$ represents the part of $\mathbf{S}$ leaked via $\mathbf{T}$. By Lemma 4.6, the last $k-2$ rows of $\mathbf{U}$ are uniform and mask $\overline{\mathbf{S}}$, leading to (38). Applying Lemma 4.5 with $r = k-2$, we get (39) for some prime p. Thus, with overwhelming probability, $\mathcal{D}$ computes a nonzero $\mathbf{x}$ satisfying $\mathbf{A}'\mathbf{x} = \mathbf{0} \pmod{q'}$ and $\|\mathbf{x}\|_\infty \le 2\beta_\sigma + 2\alpha_w\beta_z$. Therefore, $\mathbf{x}$ is a valid solution to the $\mathsf{MSIS}^\infty_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}$ problem. Thus, $p_3 \le \mathsf{Adv}^{\mathsf{MSIS}^\infty_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}}(\mathcal{D}) + \mathsf{negl}(\lambda)$. □

Putting Claim 4.1, Claim 4.2, Claim 4.3, Claim 4.4 together completes the proof. □

Lemma 4.5. *Let $r, d, m \in \mathbb{N}^+$ such that $r \ge 2$ and d is a power of 2. Let p be prime such that $X^d + 1$ factors modulo p into t_1 irreducible polynomials of degree d/t_1 (e.g., $p \equiv 2t_1 + 1 \pmod{4t_1}$). Let $\varepsilon_3 \in (0,1)$ and $\alpha \ge p\sqrt{2\alpha_H \ln(2(r-1)d(1+1/\varepsilon_3))/\pi}$. For $i \in \{1,2\}$, let $\overline{\mathbf{h}}^{(i)} = [\overline{h}_1^{(i)}, \ldots, \overline{h}_r^{(i)}] \in (\mathcal{T}_{\alpha_H})^r$. Assume $\overline{h}_j^{(i)}$ is invertible in $\mathcal{R}_p$ (which naturally holds for $p = 5$ and $t_1 = 2$ by Lemma 2.2). Define $\overline{g}_j^{(i)} := \overline{h}_j^{(i)}/\overline{h}_1^{(i)}$. Assume there exists $j^* \in \{2, \ldots, r\}$ such that $\delta_{j^*} := \overline{g}_{j^*}^{(2)} - \overline{g}_{j^*}^{(1)} \neq 0$.*

Let $\overline{\Lambda}_1 := \ker(\varphi_{\overline{\mathbf{h}}^{(1)}}) \subseteq \mathcal{R}^r$ and $\overline{\Lambda}_{12} := \{\mathbf{u} \in \overline{\Lambda}_1 : (\overline{\mathbf{h}}^{(2)})^\top\mathbf{u} = 0 \pmod{p}\}$. Let $\overline{\mathbf{S}} \leftarrow \mathcal{D}_{\mathcal{R}^{r\times m},\alpha}$ and define $\overline{\mathbf{z}}_i^\top := (\overline{\mathbf{h}}^{(i)})^\top\overline{\mathbf{S}}$ for $i \in \{1,2\}$. Then for every fixed $\hat{\mathbf{z}}_1^\top$,

$$\max_{\hat{\mathbf{z}}_2^\top} \Pr[\overline{\mathbf{z}}_2^\top = \hat{\mathbf{z}}_2^\top \mid \overline{\mathbf{z}}_1^\top = \hat{\mathbf{z}}_1^\top] \le \left(\frac{1+\varepsilon_3}{1-\varepsilon_3}\right)^{2m} p^{-dm/t_1}. \tag{40}$$

Proof. We first upper bound $\eta_{\varepsilon_3}(\overline{\Lambda}_{12})$ to show $\alpha \ge \eta_{\varepsilon_3}(\overline{\Lambda}_{12})$ and $\alpha \ge \eta_{\varepsilon_3}(\overline{\Lambda}_1)$. Observe that $p \cdot \overline{\Lambda}_1' \subseteq \overline{\Lambda}_{12}$, where $\overline{\Lambda}_1'$ is generated by column basis $\mathbf{B}'$ formed by placing $-\overline{h}_j^{(1)}$ in first row and $\overline{h}_1^{(1)}$ along the diagonal for $j \ge 2$. Under coefficient embedding, all columns of $\mathbf{B}'$ have weight $2\alpha_H$ and non-zero coefficients in $\{-1, 1\}$, giving a Euclidean norm of $\sqrt{2\alpha_H}$. Since $\overline{\Lambda}_1'$ has $\mathbb{Z}$-rank $(r-1)d$, applying Lemma 2.9 gives $\eta_{\varepsilon_3}(\overline{\Lambda}_{12}) \le p \cdot \eta_{\varepsilon_3}(\overline{\Lambda}_1') \le p\sqrt{\frac{2\alpha_H \ln(2(r-1)d(1+1/\varepsilon_3))}{\pi}} \le \alpha$. Since $\overline{\Lambda}_{12} \subseteq \overline{\Lambda}_1$, we get $\alpha \ge \eta_{\varepsilon_3}(\overline{\Lambda}_1)$.

Fix $j \in [m]$ and write $\overline{\mathbf{s}} := \overline{\mathbf{s}}_j \leftarrow \mathcal{D}_{\mathcal{R}^r,\alpha}$, $\overline{z}_1 := (\overline{\mathbf{h}}^{(1)})^\top\overline{\mathbf{s}}$, $\overline{z}_2 := (\overline{\mathbf{h}}^{(2)})^\top\overline{\mathbf{s}}$. Fix $\hat{z}_1 \in \mathcal{R}$ and choose any $\overline{\mathbf{c}}_1 \in \mathcal{R}^r$ with $(\overline{\mathbf{h}}^{(1)})^\top\overline{\mathbf{c}}_1 = \hat{z}_1$. Then $\{\overline{z}_1 = \hat{z}_1\}$ is equivalent to $\overline{\mathbf{s}} \in \overline{\mathbf{c}}_1 + \overline{\Lambda}_1$. If the system $((\overline{\mathbf{h}}^{(1)})^\top\overline{\mathbf{s}} = \hat{z}_1,\ (\overline{\mathbf{h}}^{(2)})^\top\overline{\mathbf{s}} = \hat{z}_2 \pmod{p})$ is solvable, its solution set is a coset $\overline{\mathbf{c}}_{12} + \overline{\Lambda}_{12}$. Thus,

$$\Pr[\overline{z}_2 = \hat{z}_2 \mid \overline{z}_1 = \hat{z}_1] = \frac{\Pr[\overline{z}_1 = \hat{z}_1 \wedge \overline{z}_2 = \hat{z}_2]}{\Pr[\overline{z}_1 = \hat{z}_1]} \tag{41}$$

$$\le \frac{\Pr[\overline{z}_1 = \hat{z}_1 \wedge \overline{z}_2 = \hat{z}_2 \pmod{p}]}{\Pr[\overline{z}_1 = \hat{z}_1]} \tag{42}$$

$$= \frac{\rho_\alpha(\overline{\mathbf{c}}_{12} + \overline{\Lambda}_{12})}{\rho_\alpha(\overline{\mathbf{c}}_1 + \overline{\Lambda}_1)} \tag{43}$$

$$\le \frac{\rho_\alpha(\overline{\Lambda}_{12})}{\rho_\alpha(\overline{\mathbf{c}}_1 + \overline{\Lambda}_1)} \quad \text{(Lemma 2.7).} \tag{44}$$

Since $\alpha \ge \eta_{\varepsilon_3}(\overline{\Lambda}_1)$,

$$\rho_\alpha(\overline{\mathbf{c}}_1 + \overline{\Lambda}_1) \ge \frac{1-\varepsilon_3}{1+\varepsilon_3}\rho_\alpha(\overline{\Lambda}_1) \quad \text{(Lemma 2.7)} \tag{45}$$

$$\ge \frac{(1-\varepsilon_3)^2 \cdot \alpha^{(r-1)d}}{1+\varepsilon_3} \det(\overline{\Lambda}_1)^{-1} \quad \text{(Lemma 2.8).} \tag{46}$$

Since $\alpha \ge \eta_{\varepsilon_3}(\overline{\Lambda}_{12})$, by Lemma 2.8,

$$\rho_\alpha(\overline{\Lambda}_{12}) \le (1+\varepsilon_3) \cdot \alpha^{(r-1)d} \det(\overline{\Lambda}_{12})^{-1}. \tag{47}$$

Combining (44), (46), (47) yields

$$\Pr[\overline{z}_2 = \hat{z}_2 \mid \overline{z}_1 = \hat{z}_1] \le \left(\frac{1+\varepsilon_3}{1-\varepsilon_3}\right)^2 \cdot \frac{\det(\overline{\Lambda}_1)}{\det(\overline{\Lambda}_{12})}. \tag{48}$$

Define $\psi : \overline{\Lambda}_1 \to \mathcal{R}_p$ by $\psi(\mathbf{u}) = (\overline{\mathbf{h}}^{(2)})^\top \mathbf{u} \pmod p$. By definition of $\overline{\Lambda}_{12}$, we have $\ker(\psi) = \overline{\Lambda}_{12}$. By the first isomorphism theorem,

$$\frac{\det(\overline{\Lambda}_1)}{\det(\overline{\Lambda}_{12})} = 1/|\overline{\Lambda}_1/\overline{\Lambda}_{12}| = 1/|\mathrm{Im}(\psi)|, \tag{49}$$

hence it suffices to lower bound $|\mathrm{Im}(\psi)|$. We first observe that because $\overline{\Lambda}'_1 \subseteq \overline{\Lambda}_1$, we have $|\mathrm{Im}(\psi)| \geq |\mathrm{Im}(\psi')|$, where $\psi' : \overline{\Lambda}'_1 \to \mathcal{R}_p$ is the restriction of ψ to the sublattice $\overline{\Lambda}'_1$. We have:

$$|\mathrm{Im}(\psi')| := \left|\{y \in \mathcal{R}_p : \exists \mathbf{u} \in \overline{\Lambda}'_1 \text{ s.t. } (\overline{\mathbf{h}}^{(2)})^\top \mathbf{u} = y\}\right| \tag{50}$$

$$= \left|\{y \in \mathcal{R}_p : \exists \mathbf{v} \in \mathcal{R}_p^{r-1} \text{ s.t. } (\overline{\mathbf{h}}^{(2)})^\top \mathbf{B}'\mathbf{v} = y\}\right| \tag{51}$$

$$= \left|\{y \in \mathcal{R}_p : \exists \mathbf{v} \in \mathcal{R}_p^{r-1} \text{ s.t. } \mathbf{c}^\top \mathbf{v} = y\}\right| \tag{52}$$

$$= \left|\mathbf{c}^\top \mathcal{R}_p^{r-1}\right|, \tag{53}$$

where $\mathbf{c}^\top := (\overline{\mathbf{h}}^{(2)})^\top \mathbf{B}' = (c_2, \ldots, c_r) \in \mathcal{R}_p^{r-1}$ with $c_j = \overline{h}_1^{(1)}\overline{h}_j^{(2)} - \overline{h}_1^{(2)}\overline{h}_j^{(1)} = \overline{h}_1^{(1)}\overline{h}_1^{(2)}\delta^{(j)}$ for $j \in \{2, \ldots, r\}$. Therefore, $|\mathrm{Im}(\psi')|$ is the cardinality of the ideal $\mathcal{I}$ in $\mathcal{R}_p$ generated by $c_2, \ldots, c_r$. From the assumption that there exists some $j^* \in \{2, \ldots, r\}$ such that $\delta^{(j^*)} \neq 0$ and the invertibility of $\overline{h}_1^{(1)}\overline{h}_1^{(2)}$ in $\mathcal{R}_p$, it follows that $c_{j^*} \neq 0$ and hence $\mathcal{I}$ is a non-trivial ideal in $\mathcal{R}_p$. We recall that $\mathcal{R}_p$ is a principal ideal domain, and $X^d + 1$ factors mod p into irreducible factors $\phi_i(X)$ over $\mathbb{Z}_p$ each of degree d/t_1. Hence, any non-trivial proper ideal of $\mathcal{R}_p$ is generated by $\prod_{i \in S} \phi_i(X)$ for some proper subset $S \subset [t_1]$. It follows that the smallest cardinality over all non-trivial proper ideals of $\mathcal{R}_p$ is p^{d/t_1} and we conclude that $|\mathrm{Im}(\psi)| \geq |\mathrm{Im}(\psi')| \geq p^{d/t_1}$.

Then $\frac{\det(\overline{\Lambda}_1)}{\det(\overline{\Lambda}_{12})} = [\overline{\Lambda}_1 : \overline{\Lambda}_{12}]^{-1} \leq p^{-d/t_1}$. Plugging this into (48) gives the single-column bound

$$\max_{\hat{z}_2 \in \mathcal{R}} \Pr[\overline{z}_2 = \hat{z}_2 \mid \overline{z}_1 = \hat{z}_1] \leq \left(\frac{1+\varepsilon_3}{1-\varepsilon_3}\right)^2 p^{-d/t_1}. \tag{54}$$

Finally, conditioning on $\overline{\mathbf{z}}_1^\top = \hat{\mathbf{z}}_1^\top$ fixes each per-column condition $\overline{z}_{1,j} = \hat{z}_{1,j}$ and preserves independence across columns, hence

$$\Pr[\overline{\mathbf{z}}_2^\top = \hat{\mathbf{z}}_2^\top \mid \overline{\mathbf{z}}_1^\top = \hat{\mathbf{z}}_1^\top] \leq \left(\left(\frac{1+\varepsilon_3}{1-\varepsilon_3}\right)^2 p^{-d/t_1}\right)^m, \tag{55}$$

and taking maximum over $\hat{\mathbf{z}}_2^\top$ yields the claim. □

Lemma 4.6. *Let $d \geq t_2 > 1$ be powers of 2 and $q' \equiv 2t_2 + 1 \pmod{4t_2}$ be a prime. Let $k, \alpha_H \in \mathbb{N}^+$ such that $k \geq 4$. Fix vectors $\mathbf{h}^{(i)} = [1, h_2^{(i)}, \overline{\mathbf{h}}^{(i)}] \in \mathcal{R}^k$ for $i \in \{1, 2\}$, where $h_2^{(i)} \in \mathcal{T}_{\alpha_H}$ and $\overline{\mathbf{h}}^{(i)} \in (\mathcal{T}_{\alpha_H})^{k-2}$. Assume $x := h_2^{(1)} - h_2^{(2)} \pmod{q'}$ is invertible in $\mathcal{R}_{q'}$. Define*

$$\mathbf{B}_{q'} := \begin{bmatrix} -h_2^{(1)} x^{-1}(\overline{\mathbf{h}}^{(2)} - \overline{\mathbf{h}}^{(1)})^\top - (\overline{\mathbf{h}}^{(1)})^\top \\ x^{-1}(\overline{\mathbf{h}}^{(2)} - \overline{\mathbf{h}}^{(1)})^\top \\ \mathbf{I}_{k-2} \end{bmatrix} \in \mathcal{R}_{q'}^{k \times (k-2)}.$$

Then the map $\phi : \mathcal{R}_{q'}^{k-2} \to \mathcal{R}_{q'}^k$ defined by $\phi(\mathbf{v}) = \mathbf{B}_{q'}\mathbf{v}$ is a bijection from $\mathcal{R}_{q'}^{k-2}$ onto the joint right kernel of $(\mathbf{h}^{(1)})^\top$ and $(\mathbf{h}^{(2)})^\top$. Thus, sampling m uniform vectors $\mathbf{v} \leftarrow \mathcal{R}_{q'}^{k-2}$ yields a matrix $\mathbf{U} \in \mathcal{R}_{q'}^{k \times m}$ with i.i.d. columns uniform in the joint right kernel of $(\mathbf{h}^{(1)})^\top$ and $(\mathbf{h}^{(2)})^\top$.

Proof. Write any $\mathbf{u} \in \mathcal{R}_{q'}^k$ as $\mathbf{u} = (u_1, u_2, \overline{\mathbf{u}})$ with $u_1, u_2 \in \mathcal{R}_{q'}$ and $\overline{\mathbf{u}} \in \mathcal{R}_{q'}^{k-2}$. The kernel constraints $(\mathbf{h}^{(1)})^\top \mathbf{u} = 0 \pmod{q'}$ and $(\mathbf{h}^{(2)})^\top \mathbf{u} = 0 \pmod{q'}$ expand to

$$u_1 + h_2^{(1)} u_2 + (\overline{\mathbf{h}}^{(1)})^\top \overline{\mathbf{u}} = 0 \pmod{q'}, \tag{56}$$

$$u_1 + h_2^{(2)} u_2 + (\overline{\mathbf{h}}^{(2)})^\top \overline{\mathbf{u}} = 0 \pmod{q'}. \tag{57}$$

Subtracting (57) from (56) gives

$$(h_2^{(1)} - h_2^{(2)})u_2 + (\overline{\mathbf{h}}^{(1)} - \overline{\mathbf{h}}^{(2)})^\top \overline{\mathbf{u}} = 0 \pmod{q'}. \tag{58}$$

Therefore, solving for u_2 using the inverse of x:

$$u_2 = x^{-1}(\overline{\mathbf{h}}^{(2)} - \overline{\mathbf{h}}^{(1)})^\top \overline{\mathbf{u}} \pmod{q'}. \tag{59}$$

Plugging (59) into (56) yields:

$$\begin{aligned} u_1 &= -h_2^{(1)} u_2 - (\overline{\mathbf{h}}^{(1)})^\top \overline{\mathbf{u}} \\ &= -h_2^{(1)} x^{-1}(\overline{\mathbf{h}}^{(2)} - \overline{\mathbf{h}}^{(1)})^\top \overline{\mathbf{u}} - (\overline{\mathbf{h}}^{(1)})^\top \overline{\mathbf{u}} \pmod{q'}. \end{aligned} \tag{60}$$

Since u_1 and u_2 are uniquely determined by the free choice of $\overline{\mathbf{u}} \in \mathcal{R}_{q'}^{k-2}$, every $\mathbf{u}$ such that $(\mathbf{h}^{(1)})^\top \mathbf{u} = 0 \pmod{q'}$ and $(\mathbf{h}^{(2)})^\top \mathbf{u} = 0 \pmod{q'}$ equals $\mathbf{B}_{q'} \cdot \overline{\mathbf{u}}$.

To show that ϕ is a bijection, we observe surjectivity from the derivation above. For injectivity, if $\phi(\mathbf{v}) = \mathbf{B}_{q'}\mathbf{v} = \mathbf{0}$, then looking at the bottom block (the last $k-2$ entries) gives $\mathbf{I}_{k-2}\mathbf{v} = \mathbf{0}$, which implies $\mathbf{v} = \mathbf{0}$. Thus, the kernel of ϕ is trivial. Consequently, a uniform choice of $\mathbf{v}$ yields a uniform distribution over the joint kernel. Independence for m columns follows immediately. □

5 Homomorphic Vector Commitment

In this section, we formally define homomorphic vector commitments (HVCs) and present a concrete construction based on the hardness of the Module-SIS problem. Our construction generalizes and optimizes the Chipmunk HVC framework [16] in several key dimensions. Specifically, we extend the supported commitment domain $\mathcal{S}_{\mathsf{dom}}$ from vectors to matrices, and transition from the Ring-SIS assumption to the more versatile Module-SIS assumption.

In what follows, Section 5.1 provides the formal definition of HVCs and their required security properties. In Section 5.2 and Section 5.3, we introduce the necessary projection and encoding gadgets, "lifting" the original Chipmunk primitives to our matrix-based Module-SIS setting. Section 5.4 presents our concrete construction and formal proofs of security.

5.1 Definition

We formally define homomorphic vector commitments here.

Definition 5.1 (Homomorphic Vector Commitment (HVC), [16]). *Let $\tau, d \in \mathbb{N}^+$ such that d is a power of 2. Let $\mathcal{R} = \mathbb{Z}[X]/(X^d + 1)$. Let $\mathcal{S}_{\mathsf{dom}}, \mathcal{S}_{\mathsf{com}}, \mathcal{S}_{\mathsf{open}}$ be $\mathcal{R}$-modules. A homomorphic vector commitment scheme for domain $\mathcal{S}_{\mathsf{dom}}$ and vectors of length 2^τ consists of six PPT algorithms $\Pi_{\mathrm{HVC}} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{iVrfy}, \mathsf{sVrfy}, \mathsf{wVrfy})$.*

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: On input a security parameter $\lambda \in \mathbb{N}^+$, the setup algorithm outputs public parameters* pp.

- $c \leftarrow \mathsf{Com}(\mathsf{pp}, \widetilde{\mathbf{M}})$*: On input public parameters* pp *and a vector of matrices* $\widetilde{\mathbf{M}} = \{\mathbf{M}_1, \ldots, \mathbf{M}_{2^\tau}\} \in \mathcal{S}_{\mathsf{dom}}^{2^\tau}$*, the commitment algorithm outputs a commitment* $c \in \mathcal{S}_{\mathsf{com}}$.
- $\mathsf{op} \leftarrow \mathsf{Open}(\mathsf{pp}, c, \widetilde{\mathbf{M}}, t)$*: On input public parameters* pp*, a commitment* c*, a committed vector* $\widetilde{\mathbf{M}}$*, and an index* $t \in [2^\tau]$*, the opening algorithm outputs a decommitment* $\mathsf{op} \in \mathcal{S}_{\mathsf{open}}$
- $\mathbf{M}_t / \perp \leftarrow \mathsf{iVrfy/sVrfy/wVrfy}(\mathsf{pp}, c, t, \mathsf{op})$*: On input public parameters* pp*, a commitment* c*, an index* t*, and a decommitment* op*, the individual/strong/weak verification algorithm outputs either* $\mathbf{M}_t \in \mathcal{S}_{\mathsf{dom}}$ *or an error symbol* $\perp$.

Our HVC construction, operates over the message space $\mathcal{S}_{\mathsf{dom}} = \mathcal{R}_{q'}^{\rho \times \nu}$ for $\rho, \nu \in \mathbb{N}^+$ and a prime q'. In contrast, Chipmunk considers the domain $\mathcal{S}_{\mathsf{dom}} = \mathcal{R}_{q'}^{\ell_{\mathsf{dom}}}$. The commitment and opening spaces will be of the form $\mathcal{S}_{\mathsf{com}} = \mathcal{R}_q^{\ell_{\mathsf{com}}}$ and $\mathcal{S}_{\mathsf{open}} = \mathcal{R}^{\ell_{\mathsf{open}}}$ for a prime q as in Chipmunk. We require a HVC to satisfy the following four properties: individual correctness, probabilistic homomorphism, robust homomorphism, and position binding. These properties are formally defined below.

Definition 5.2 (Individual Correctness, [16]). *We say that* Π_{HVC} *is individually correct, if for all* $\lambda \in \mathbb{N}^+$, $\widetilde{\mathbf{M}} \in \mathcal{S}_{\mathsf{dom}}^{2^\tau}$, $t \in [2^\tau]$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $c \leftarrow \mathsf{Com}(\mathsf{pp}, \widetilde{\mathbf{M}})$*, and* $\mathsf{op} \leftarrow \mathsf{Open}(\mathsf{pp}, c, \widetilde{\mathbf{M}}, t)$ *it holds that* $\mathsf{iVrfy}(\mathsf{pp}, c, t, \mathsf{op}) = \mathbf{M}_t$.

Definition 5.3 (Probabilistic Homomorphism, [16]). *Let* $N \in \mathbb{N}^+$, $\epsilon_{\mathsf{hom}} \in (0,1)$*, and* $W \subseteq \mathcal{R}$*. We say that* Π_{HVC} *is* $(N, W, \epsilon_{\mathsf{hom}})$*-probabilistically homomorphic, if for all* $\lambda \in \mathbb{N}^+$, $t \in [2^\tau]$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* $(c_i \in \mathcal{S}_{\mathsf{com}}, \mathsf{op}_i \in \mathcal{S}_{\mathsf{open}})$ *where* $\mathsf{iVrfy}(\mathsf{pp}, c_i, t, \mathsf{op}_i) = \mathbf{M}_i \neq \perp$ *for all* $i \in [N]$*, it holds that:*

$$\Pr_{w_i \leftarrow W}\left[\mathsf{sVrfy}\left(\mathsf{pp}, \sum_{i=1}^{N} w_i \cdot c_i, t, \sum_{i=1}^{N} w_i \cdot \mathsf{op}_i\right) = \sum_{i=1}^{N} w_i \cdot \mathbf{M}_i\right] \geq 1 - \epsilon_{\mathsf{hom}}.$$

Definition 5.4 (Robust Homomorphism, [16]). *We say that* Π_{HVC} *is robustly homomorphic, if for all* $\lambda \in \mathbb{N}^+$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $t \in [2^\tau]$*, any* $c_b \in \mathcal{S}_{\mathsf{com}}$ *and* $\mathsf{op}_b \in \mathcal{S}_{\mathsf{open}}$ *(*$b \in \{0,1\}$*) with* $\mathsf{sVrfy}(\mathsf{pp}, c_b, t, \mathsf{op}_b) = \mathbf{M}_b \neq \perp$*, it holds that* $\mathsf{wVrfy}(\mathsf{pp}, c_0 - c_1, t, \mathsf{op}_0 - \mathsf{op}_1) = \mathbf{M}_0 - \mathbf{M}_1$.

Definition 5.5 (Position Binding, [16]). *We say that* Π_{HVC} *is position binding, if for all* $\lambda \in \mathbb{N}^+$ *and PPT algorithms* $\mathcal{A}$ *it holds that* $\Pr[\mathbf{M}_0 \neq \mathbf{M}_1 \wedge \perp \notin \{\mathbf{M}_0, \mathbf{M}_1\}] \leq \mathsf{negl}(\lambda)$*, where* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(c, t, \mathsf{op}_0, \mathsf{op}_1) \leftarrow \mathcal{A}(\mathsf{pp})$*, and* $\mathbf{M}_b := \mathsf{wVrfy}(\mathsf{pp}, c, t, \mathsf{op}_b)$ *for* $b \in \{0,1\}$.

5.2 Projection and Decomposition

Chipmunk constructs a homomorphic vector commitment (HVC) using a Merkle tree, where homomorphism is achieved by replacing the standard hash function with the Ajtai hash. This design, however, introduces an inherent technical constraint: the Ajtai hash requires inputs to be "small", whereas its outputs are not guaranteed to satisfy this property. To address this issue, Chipmunk introduces a pair of canonical transformations: a decomposition map, which converts an element of $\mathcal{R}_q$ into a short vector over $\mathcal{R}$, and a corresponding projection map, which recombines such a short vector into the original ring element. Informally, decomposition enables "large" elements to be represented in a form suitable for Ajtai hashing, while projection ensures consistency with the underlying algebraic structure. In this subsection, we first recall the projection and decomposition maps used in Chipmunk, along with their key properties. We then extend these maps to Lemur's label space by applying them independently to each block of a tree node.

Projection and Decomposition in Chipmunk. We first provide additional intuition for why projection and decomposition are necessary in the construction of a homomorphic Merkle tree. For clarity, we focus on the core idea and omit some technical details.

In Chipmunk, the internal node of the Merkle tree is computed using an Ajtai hash. Concretely, given two child nodes $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{R}_q^{\ell_{\mathsf{dom}}}$, the parent node is computed as

$$\mathbf{p} = \mathbf{a}_1^\top \mathbf{c}_1 + \mathbf{a}_2^\top \mathbf{c}_2 \pmod q, \tag{61}$$

where $\mathbf{a}_1, \mathbf{a}_2 \in \mathcal{R}_q^{\ell_{\mathsf{dom}}}$ are public parameters. While this operation preserves homomorphism, it creates a mismatch in representation: the resulting $\mathbf{p}$ lies in $\mathcal{R}_q$, whereas the Ajtai hash requires inputs to be vectors of "small" elements over $\mathcal{R}$. To enable iterative hashing across tree levels, Chipmunk employs a balanced $(2\eta+1)$-ary decomposition on $\mathbf{p}$, where coefficients lie in small sets $\{-\eta, \ldots, \eta\}$. The projection map serves as an inverse transformation and ensures that this transformation remains consistent with the original algebraic value, i.e.,

$$\overline{\mathsf{proj}}_q(\mathbf{p}) = \mathbf{a}_1^\top \mathbf{c}_1 + \mathbf{a}_2^\top \mathbf{c}_2 \pmod q. \tag{62}$$

We will see in the following that the projection map enjoys useful homomorphic properties.

(1) **Projection and Decomposition over $\mathcal{R}$.** Let $\eta, \kappa \in \mathbb{N}^+$. Define the projection map $\overline{\mathsf{proj}}_{\eta,\kappa} : \mathcal{R}^\kappa \to \mathcal{R}$ by $\overline{\mathsf{proj}}_{\eta,\kappa}(a_1, \ldots, a_\kappa) := \sum_{i=1}^{\kappa} a_i \cdot (2\eta+1)^{i-1}$. The corresponding decomposition map $\overline{\mathsf{dec}}_{\eta,\kappa} : \mathcal{R} \to \mathcal{R}^\kappa$ is defined as $\overline{\mathsf{dec}}_{\eta,\kappa}(a) = (a_1, \ldots, a_\kappa)$, where $a = \sum_{i=1}^{\kappa} a_i \cdot (2\eta+1)^{i-1}$, and $\|a_i\|_\infty \leq \eta$ for all $1 \leq i < \kappa$.

(2) **Projection and Decomposition over $\mathcal{R}_q$.** Let $\eta \in \mathbb{N}^+$ and q be an odd prime. Define $\kappa := \lceil \log_{2\eta+1} q \rceil$. The projection map $\overline{\mathsf{proj}}_q : \mathcal{R}^\kappa \to \mathcal{R}_q$ is given by $\overline{\mathsf{proj}}_q(\mathbf{a}) := \overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{a}) \pmod q$. The corresponding decomposition map $\overline{\mathsf{dec}}_q : \mathcal{R}_q \to \mathcal{R}^\kappa$ is defined as $\overline{\mathsf{dec}}_q(a) := \overline{\mathsf{dec}}_{\eta,\kappa}(a')$, where $a' \in \mathcal{R}$ is the centered representative of $a \in \mathcal{R}_q$ with coefficients in $\left\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\right\}$.

Proposition 5.1. *The following facts are established in Proposition 13 of Chipmunk [16].*

- $\forall \mathbf{a}_0, \mathbf{a}_1 \in \mathcal{R}^\kappa$ *and* $w_0, w_1 \in \mathcal{R}$,

  $\overline{\mathsf{proj}}_{\eta,\kappa}(w_0 \cdot \mathbf{a}_0 + w_1 \cdot \mathbf{a}_1) = w_0 \cdot \overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{a}_0) + w_1 \cdot \overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{a}_1)$,
  $\overline{\mathsf{proj}}_q(w_0 \cdot \mathbf{a}_0 + w_1 \cdot \mathbf{a}_1) = w_0 \cdot \overline{\mathsf{proj}}_q(\mathbf{a}_0) + w_1 \cdot \overline{\mathsf{proj}}_q(\mathbf{a}_1)$.

- $\forall a \in \mathcal{R}$, $\overline{\mathsf{proj}}_{\eta,\kappa}(\overline{\mathsf{dec}}_{\eta,\kappa}(a)) = a$.
- $\forall a \in \mathcal{R}_q$, $\overline{\mathsf{proj}}_q(\overline{\mathsf{dec}}_q(a)) = a$.
- $\forall a \in \mathcal{R}_q$, $\|\overline{\mathsf{dec}}_q(a)\|_\infty \leq \eta$.

Proposition 5.2 (Proposition 22 of [16]). *Let* $q, \eta \in \mathbb{N}^+$ *with* q *prime. Set* $B := 2\eta + 1$ *and* $\kappa := \lceil \log_B q \rceil$*. Define lattices* $\Lambda_{\mathcal{R}} := \{\mathbf{v} \in$

$\mathcal{R}^\kappa \mid \overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{v}) = 0\}$. *Let*

$$\mathcal{B} := \begin{pmatrix} -B & 0 & \cdots & 0 \\ 1 & -B & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & -B \\ 0 & 0 & \cdots & 1 \end{pmatrix} \in \mathcal{R}^{\kappa\times(\kappa-1)}. \tag{63}$$

Then, $\Lambda_{\mathcal{R}}$ is an $\mathcal{R}$-module lattice. Moreover, the columns of $\mathcal{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1}]$ form an $\mathcal{R}$-basis of $\Lambda_{\mathcal{R}}$.

Projection and Decomposition in Lemur. We now explain why it is necessary to extend the projection and decomposition maps of Chipmunk. In Lemur, the binding property of the Ajtai hash is based on the MSIS assumption, whereas Chipmunk relies on RSIS. Concretely, given two child nodes $\mathbf{c}_1, \mathbf{c}_2 \in \mathcal{R}_q^{\ell_{\mathsf{dom}}}$, the hash computation in Lemur takes the form

$$\mathbf{y} = \mathbf{A}_1\mathbf{c}_1 + \mathbf{A}_2\mathbf{c}_2 \in \mathcal{R}_q^\omega, \tag{64}$$

where $\mathbf{A}_1, \mathbf{A}_2 \in \mathcal{R}_q^{\omega\times\ell_{\mathsf{dom}}}$ are public parameters. Thus, in contrast to Chipmunk, where the hash output is a single element of $\mathcal{R}_q$ element, the output in Lemur is a vector of $\mathcal{R}^\omega$. This distinction leads to a natural employment of the decomposition map entry-wise. Moving forward, after decomposition, Merkle tree node labels in Lemur lie in $\mathcal{R}^{\omega\kappa}$. We may view any vector $\mathbf{v} \in \mathcal{R}^{\omega\kappa}$ as a concatenation of ω blocks, i.e.,

$$\mathbf{v} = \left(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\omega)}\right), \text{ and } \mathbf{v}^{(r)} \in \mathcal{R}^\kappa \text{ for } r \in [\omega]. \tag{65}$$

We extend the maps $\overline{\mathsf{proj}}_{\eta,\kappa}, \overline{\mathsf{proj}}_q, \overline{\mathsf{dec}}_{\eta,\kappa}, \overline{\mathsf{dec}}_q$ to this setting by applying them independently to each block. Specifically, we define

(1) **Projection and Decomposition over $\mathcal{R}$.** Let $\eta, \kappa, \omega \in \mathbb{N}^+$. Define the projection map $\mathsf{proj}_{\eta,\kappa} : \mathcal{R}^{\omega\kappa} \to \mathcal{R}^\omega$, with $\mathsf{proj}_{\eta,\kappa}(\mathbf{v}) := \left(\overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{v}^{(1)}), \ldots, \overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{v}^{(\omega)})\right)$, where $\mathbf{v}^{(r)} \in \mathcal{R}^\kappa$ for $r \in [\omega]$. The corresponding decomposition map $\mathsf{dec}_{\eta,\kappa} : \mathcal{R}^\omega \to \mathcal{R}^{\omega\kappa}$, is given by $\mathsf{dec}_{\eta,\kappa}(\mathbf{y}) := \left(\overline{\mathsf{dec}}_{\eta,\kappa}(y_1), \ldots, \overline{\mathsf{dec}}_{\eta,\kappa}(y_\omega)\right)$.

(2) **Projection and Decomposition over $\mathcal{R}_q$.** Let $\eta \in \mathbb{N}^+$ and q be an odd prime. Define $\kappa := \lceil \log_{2\eta+1} q \rceil$. The projection map is given by $\mathsf{proj}_q : \mathcal{R}^{\omega\kappa} \to \mathcal{R}_q^\omega$, with $\mathsf{proj}_q(\mathbf{v}) := \left(\overline{\mathsf{proj}}_q(\mathbf{v}^{(1)}), \ldots, \overline{\mathsf{proj}}_q(\mathbf{v}^{(\omega)})\right)$. The corresponding decomposition map is defined as $\mathsf{dec}_q : \mathcal{R}_q^\omega \to \mathcal{R}^{\omega\kappa}$, with $\mathsf{dec}_q(\mathbf{y}) := \left(\overline{\mathsf{dec}}_q(y_1), \ldots, \overline{\mathsf{dec}}_q(y_\omega)\right)$.

Proposition 5.3. *The following facts are obtained by independently employing Proposition 5.1 block by block.*

- $\forall \mathbf{a}_0, \mathbf{a}_1 \in \mathcal{R}^{\omega\kappa}$ *and* $w_0, w_1 \in \mathcal{R}$,

$\mathsf{proj}_{\eta,\kappa}(w_0 \cdot \mathbf{a}_0 + w_1 \cdot \mathbf{a}_1) = w_0 \cdot \mathsf{proj}_{\eta,\kappa}(\mathbf{a}_0) + w_1 \cdot \mathsf{proj}_{\eta,\kappa}(\mathbf{a}_1)$,
$\mathsf{proj}_q(w_0 \cdot \mathbf{a}_0 + w_1 \cdot \mathbf{a}_1) = w_0 \cdot \mathsf{proj}_q(\mathbf{a}_0) + w_1 \cdot \mathsf{proj}_q(\mathbf{a}_1)$.

- $\forall \mathbf{a} \in \mathcal{R}_q^\omega$, $\mathsf{proj}_q(\mathsf{dec}_q(\mathbf{a})) = \mathbf{a}$.
- $\forall \mathbf{a} \in \mathcal{R}^\omega$, $\mathsf{proj}_{\eta,\kappa}(\mathsf{dec}_{\eta,\kappa}(\mathbf{a})) = \mathbf{a}$.
- $\forall \mathbf{a} \in \mathcal{R}_q^\omega$, $\|\mathsf{dec}_q(\mathbf{a})\|_\infty \le \eta$.

5.3 Encode / Decode Algorithms

Projection and decomposition give us canonical short representatives for node labels, but they do not by themselves yield small *aggregatable* openings: since dec_q is not additive, aggregated openings cannot be verified from sibling nodes alone without also sending the intermediate path labels. Chipmunk avoids this blow-up using an additional encoding step that makes the missing path information recoverable from a short "hint". In this subsection, we migrate Chipmunk's encoding techinique to our label space $\mathcal{R}^{\omega\kappa}$. The resulting algorithms $\mathsf{Encode}^{\mathcal{B}}_{\eta,q}$ and $\mathsf{Decode}^{\mathcal{B}}_{\eta,q}$ are applied block-wise and allow the verifier to reconstruct each path label from its encoding and the corresponding hint. See Figure 4 for the explicit algorithms.

$\underline{\mathsf{Encode}^{\mathcal{B}}_{\eta,q}(\mathbf{v})}$

(1) Set $\kappa := \left\lceil \log_{2\eta+1} q \right\rceil$.
(2) Parse $\mathbf{v} \in \mathcal{R}^{\omega\kappa}$ as $\mathbf{v} = \left(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\omega)}\right)$, with $\mathbf{v}^{(r)} \in \mathcal{R}^\kappa$.
(3) For each $r \in [\omega]$ do:
  (a) $\mathsf{hint}^{(r)} \leftarrow \overline{\mathsf{proj}}_q(\mathbf{v}^{(r)}) \in \mathcal{R}_q$.
  (b) Represent $\mathsf{hint}^{(r)}$ by $\mathsf{hint}'^{(r)} \in \mathcal{R}$ with coefficients in $\left\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\right\}$.
  (c) $\alpha_*^{(r)} := \dfrac{\overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{v}^{(r)}) - \mathsf{hint}'^{(r)}}{q} \in \mathcal{R}$
  (d) $\boldsymbol{\delta}^{(r)} := \mathbf{v}^{(r)} - \overline{\mathsf{dec}}_{\eta,\kappa}(\overline{\mathsf{proj}}_{\eta,\kappa}(\mathbf{v}^{(r)})) \in \mathcal{R}^\kappa$.
  (e) Find unique $\alpha_1^{(r)}, \ldots, \alpha_{\kappa-1}^{(r)} \in \mathcal{R}$ such that $\boldsymbol{\delta}^{(r)} = \alpha_1^{(r)}\mathbf{b}_1 + \cdots + \alpha_{\kappa-1}^{(r)}\mathbf{b}_{\kappa-1}$, where $\mathbf{b}_1, \ldots, \mathbf{b}_{\kappa-1}$ is the basis of $\Lambda_{\mathcal{R}} = \ker(\overline{\mathsf{proj}}_{\eta,\kappa}) \subset \mathcal{R}^\kappa$.
(4) Return $\overline{\mathbf{v}} := \left(\overline{\mathbf{v}}^{(1)}, \ldots, \overline{\mathbf{v}}^{(\omega)}\right)$, with $\overline{\mathbf{v}}^{(r)} = \left(\alpha_*^{(r)}, \alpha_1^{(r)}, \ldots, \alpha_{\kappa-1}^{(r)}\right)$ for $r \in [\omega]$.

$\underline{\mathsf{Decode}^{\mathcal{B}}_{\eta,q}(\overline{\mathbf{v}}, \mathsf{hint})}$

(1) Set $\kappa := \left\lceil \log_{2\eta+1} q \right\rceil$.
(2) Parse $\overline{\mathbf{v}} \in \mathcal{R}^{\omega\kappa}$ as $\overline{\mathbf{v}} = \left(\overline{\mathbf{v}}^{(1)}, \ldots, \overline{\mathbf{v}}^{(\omega)}\right)$, with $\overline{\mathbf{v}}^{(r)} \in \mathcal{R}^\kappa$. Parse $\mathsf{hint} \in \mathcal{R}_q^\omega$ as $\mathsf{hint} = (\mathsf{hint}^{(1)}, \ldots, \mathsf{hint}^{(\omega)})$.
(3) For each $r \in [\omega]$ do:
  (a) Represent $\mathsf{hint}^{(r)}$ by $\mathsf{hint}'^{(r)} \in \mathcal{R}$ with coefficients in $\left\{-\frac{q-1}{2}, \ldots, \frac{q-1}{2}\right\}$.
  (b) Parse $\overline{\mathbf{v}}^{(r)}$ as $(\alpha_*^{(r)}, \alpha_1^{(r)}, \ldots, \alpha_{\kappa-1}^{(r)})$.
  (c) $h''^{(r)} := \mathsf{hint}'^{(r)} + q \cdot \alpha_*^{(r)} \in \mathcal{R}$.
  (d) $\boldsymbol{\delta}^{(r)} := \alpha_1^{(r)}\mathbf{b}_1 + \cdots + \alpha_{\kappa-1}^{(r)}\mathbf{b}_{\kappa-1} \in \mathcal{R}^\kappa$.
  (e) $\mathbf{v}^{(r)} := \overline{\mathsf{dec}}_{\eta,\kappa}(h''^{(r)}) + \boldsymbol{\delta}^{(r)}) \in \mathcal{R}^\kappa$.
(4) Return $\mathbf{v} := (\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\omega)}) \in \mathcal{R}^{\omega\kappa}$.

Figure 4: Algorithms for encoding and decoding an element $\mathbf{v} \in \mathcal{R}^{\omega\kappa}$, which is viewed as $\mathbf{v} = \left(\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\omega)}\right)$, $\mathbf{v}^{(r)} \in \mathcal{R}^\kappa$.

Lemma 5.1 (Properties of $\mathsf{Encode}^{\mathcal{B}}_{\eta,q}$ and $\mathsf{Decode}^{\mathcal{B}}_{\eta,q}$, adopted from [16]). *Let $q, \eta, \omega, \kappa \in \mathbb{N}^+$ with q prime and $\kappa := \left\lceil \log_{2\eta+1}(q) \right\rceil$. Then the deterministic encoding and decoding algorithms defined in Figure 4 satisfy the following properties:*

(1) For $r \in [\omega]$, coefficients $\left(\alpha_*^{(r)}, \alpha_1^{(r)}, \ldots, \alpha_{\kappa-1}^{(r)}\right)$ as required in $\mathsf{Encode}^{\mathcal{B}}_{\eta,q}$ exist, are unique and can be found in polynomial-time.
(2) For any $\mathbf{v} \in \mathcal{R}^{\omega\kappa}$, $\mathsf{hint} = \mathsf{proj}_q(\mathbf{v})$, and $\overline{\mathbf{v}} \leftarrow \mathsf{Encode}^{\mathcal{B}}_{\eta,q}(\mathbf{v})$, we have $\mathsf{Decode}^{\mathcal{B}}_{\eta,q}(\overline{\mathbf{v}}, \mathsf{hint}) = \mathbf{v}$.
(3) For any $\overline{\mathbf{v}} \in \mathcal{R}^{\omega\kappa}$, $\mathsf{hint} \in \mathcal{R}_q^\omega$, and $\mathbf{v} \leftarrow \mathsf{Decode}^{\mathcal{B}}_{\eta,q}(\overline{\mathbf{v}}, \mathsf{hint})$, we have $\mathsf{proj}_q(\mathbf{v}) = \mathsf{hint}$, and $\mathsf{Encode}^{\mathcal{B}}_{\eta,q}(\mathbf{v}) = \overline{\mathbf{v}}$.
(4) For any $\mathbf{v} \in \mathcal{R}^{\omega\kappa}$ and $\overline{\mathbf{v}} \leftarrow \mathsf{Encode}^{\mathcal{B}}_{\eta,q}(\mathbf{v})$, with $\overline{\mathbf{v}} \leftarrow \left(\overline{\mathbf{v}}^{(1)}, \ldots, \overline{\mathbf{v}}^{(\omega)}\right)$ and $\overline{\mathbf{v}}^{(r)} \leftarrow \left(\alpha_*^{(r)}, \alpha_1^{(r)}, \ldots, \alpha_{\kappa-1}^{(r)}\right) \in \mathcal{R}^\kappa$, we have

$$\left\|\alpha_*^{(r)}\right\|_\infty \le \frac{\left\|\mathsf{proj}_{\eta,\kappa}(\mathbf{v}^{(r)})\right\|_\infty}{q} + \frac{1}{2}, \tag{66}$$

$$\left\|\alpha_i^{(r)}\right\|_\infty \le \frac{\left\|\mathbf{v}^{(r)}\right\|_\infty}{2\eta} + \frac{1}{2}. \tag{67}$$

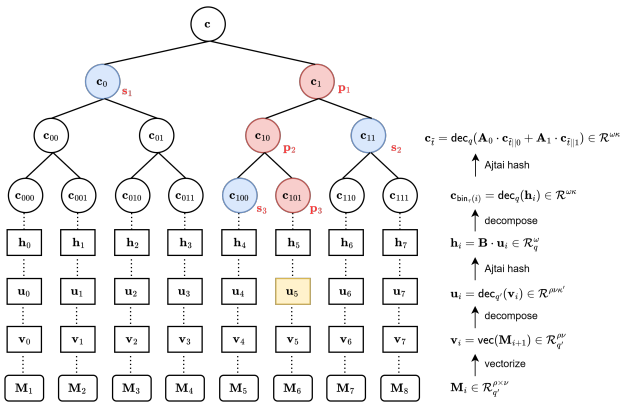

Figure 5: Visualization of an opening for the message $\mathbf{M}_6 \in \mathcal{R}_{q'}^{\rho\times\nu}$ at time slot $t = 6$ ($\tilde{t} = \mathsf{bin}_\tau(t-1) = 101$). Let $\mathbf{B} \in \mathcal{R}_q^{\omega\times\rho\nu\kappa'}, \mathbf{A}_0, \mathbf{A}_1 \in \mathcal{R}_q^{\omega\times\omega\kappa}$ be the public parameters of the Ajtai hash. The opening consists of three distinct components: (i) the path nodes $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ (highlighted in pink) which correspond to the ancestors of the target leaf; (ii) the auxiliary sibling nodes $\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$ (highlighted in blue) required to reconstruct the root commitment c; and (iii) the decomposed message vector $\mathbf{u}_5$ (highlighted in yellow) derived from the target matrix $\mathbf{M}_6$. Note that the path nodes $\mathbf{p}_i$ depicted here are in unencoded form; in the final Lemur HVC construction, these are encoded as $\overline{\mathbf{p}}_i$ to minimize bandwidth.

Proof. By definition, $\mathsf{Encode}_{\eta,q}^{\mathcal{B}}$ and $\mathsf{Decode}_{\eta,q}^{\mathcal{B}}$ act independently on each block $r \in [\omega]$. For each fixed r, we can apply Lemma 23 in Chipmunk [16] to the single-block input $\mathbf{v}^{(r)} \in \mathcal{R}^{\kappa}$ and obtain the stated lemma. □

5.4 HVC Construction

Given the building blocks introduced in the previous two subsections, we are now ready to present our concrete HVC construction. We begin by formally defining a labeled full binary tree.

Definition 5.6 (Labeled Full Binary Tree). *Let $d, q, q', \rho, \nu, \omega, \eta \in \mathbb{N}^+$ with d a power of 2 and q, q' primes. Define $\kappa := \left\lceil \log_{2\eta+1} q \right\rceil$ and $\kappa' := \left\lceil \log_{2\eta+1} q' \right\rceil$. Let $\widetilde{\mathbf{M}} = (\mathbf{M}_1, \ldots, \mathbf{M}_{2^\tau}) \in \left(\mathcal{R}_{q'}^{\rho\times\nu}\right)^{2^\tau}$, $\mathbf{B} \in \mathcal{R}_q^{\omega\times\rho\nu\kappa'}$, and $\mathbf{A}_0, \mathbf{A}_1 \in \mathcal{R}_q^{\omega\times\omega\kappa}$ be fixed. We define the labeling function $\mathsf{label}_{\mathbf{B},\mathbf{A}_0,\mathbf{A}_1} : \left(\mathcal{R}_{q'}^{\rho\times\nu}\right)^{2^\tau} \times \{0,1\}^{\leq\tau} \mapsto \mathcal{R}^{\omega\kappa}$ for a labeled full binary tree of depth τ as $\mathsf{label}_{\mathbf{B},\mathbf{A}_0,\mathbf{A}_1}(\widetilde{\mathbf{M}}, v)$ and define it as:*

$$\begin{cases} \mathsf{dec}_q\big(\mathbf{B}\cdot\mathsf{dec}_{q'}(\mathsf{vec}(\mathbf{M}_v))\big) & \text{if } |v| = \tau, \\ \mathsf{dec}_q\Big(\mathbf{A}_0\cdot\mathsf{label}_{\mathbf{B},\mathbf{A}_0,\mathbf{A}_1}(\widetilde{\mathbf{M}}, v\|0) + \mathbf{A}_1\cdot\mathsf{label}_{\mathbf{B},\mathbf{A}_0,\mathbf{A}_1}(\widetilde{\mathbf{M}}, v\|1)\Big) & \text{if } |v| < \tau. \end{cases}$$

We treat $v \in \{0,1\}^\tau$ as a big-endian integer in $\mathbf{M}_v$.

Figure 5 illustrates the HVC labelling procedure. Starting at the leaves, each matrix $\mathbf{M}_i \in \mathcal{R}_{q'}^{\rho\times\nu}$ is vectorized as $\mathbf{v}_i$ and decomposed into a small-norm representative $\mathbf{u}_i = \mathsf{dec}_{q'}(\mathbf{v}_i)$ to resolve the domain mismatch between $\mathcal{R}_{q'}$ and $\mathcal{R}_q$. Leaf labels are then computed as $\mathbf{c}_i = \mathsf{dec}_q(\mathbf{B}\mathbf{u}_i) \in \mathcal{R}^{\omega\kappa}$ using the Ajtai hash matrix $\mathbf{B}$. Internal nodes are computed recursively: for child labels $\mathbf{c}_{\tilde{i}\|0}, \mathbf{c}_{\tilde{i}\|1}$, the parent label is $\mathbf{c}_{\tilde{i}} = \mathsf{dec}_q(\mathbf{A}_0\cdot\mathbf{c}_{\tilde{i}\|0} + \mathbf{A}_1\cdot\mathbf{c}_{\tilde{i}\|1})$, where $\mathbf{A}_0, \mathbf{A}_1$ are public matrices. This consistent use of decomposition ensures that Ajtai hash inputs always remain small-norm representatives, maintaining concrete efficiency throughout the tree.

Given the definition of Labeled Full Binary Tree, we are now ready to define our HVC construction.

Definition 5.7. *Let $d, q, q', \alpha_w, N, \eta, \tau, \rho, \nu, \omega, \beta_{\mathsf{agg}} \in \mathbb{N}^+$ such that d is a power of 2 and q, q' are primes. We define the homomorphic vector commitment $\Pi_{\mathsf{HVC}} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Open}, \mathsf{iVrfy}, \mathsf{sVrfy}, \mathsf{wVrfy})$ for domain $\mathcal{S}_{\mathsf{dom}} = \mathcal{R}_{q'}^{\rho\times\nu}$, and vectors of length 2^τ by the algorithms given in Figure 6. Its commitments and openings are from $\mathcal{S}_{\mathsf{com}} = \mathcal{R}_q^{\omega}$ and $\mathcal{S}_{\mathsf{open}} = \mathcal{R}^{\tau\kappa} \times \mathcal{R}^{\tau\kappa} \times \mathcal{R}^{\rho\nu\kappa'}$, where $\kappa = \left\lceil \log_{2\eta+1} q \right\rceil$ and $\kappa' = \left\lceil \log_{2\eta+1} q' \right\rceil$.*

Similar to [16], the use of the encoding algorithm (Figure 4) implies that aggregated openings are not $\mathcal{R}$-linear in the standard sense. To recover homomorphic behavior, we define suitable operations under which our HVC is homomorphic.

Definition 5.8 ([16]). *Let $d, \eta, q, q', \omega, \rho, \nu \in \mathbb{N}^+$ with d a power of 2, and q, q' primes. Let $\kappa = \lceil \log_{2\eta+1} q \rceil$ and $\kappa' = \lceil \log_{2\eta+1} q' \rceil$. Consider Π_{HVC} from Figure 6. Let $\mathcal{S}_{\mathsf{open,un}} \subseteq (\mathcal{R}^{\omega\kappa})^{2\tau} \times (\mathcal{R}^{\kappa'})^{\rho\nu}$ be the unencoded opening space. Let $\mathsf{pp} := (\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1) \leftarrow \mathsf{Setup}(1^\lambda)$. We call an unencoded opening $\mathsf{op} = (\mathbf{p}_1, \ldots, \mathbf{p}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u})$ linearly verifying for time slot $t \in [2^\tau]$ iff the following conditions hold*

$$\mathbf{B}\cdot\mathbf{u} = \mathsf{proj}_q(\mathbf{p}_\tau), \tag{68}$$

$$\mathsf{proj}_q(\mathbf{p}_{j-1}) = \mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j \quad \text{for all } 2 \leq j \leq \tau, \tag{69}$$

where $\tilde{t} = \mathsf{bin}_\tau(t-1)$. We define $\mathcal{S}_{\mathsf{open,un}}^t \subset \mathcal{S}_{\mathsf{open,un}}$ to be the subset of all linearly verifying unencoded openings for t.

Definition 5.9 ([16]). *Fix $\mathsf{pp} = (\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1)$ and $t \in [2^\tau]$ with $\tilde{t} = \mathsf{bin}_\tau(t-1)$. Let $\mathsf{op} = (\mathbf{p}_1, \ldots, \mathbf{p}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u}) \in \mathcal{S}_{\mathsf{open,un}}^t$. Define $\mathsf{ENCODE}(\mathsf{pp}, t, \mathsf{op}) := (\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u})$, with $\overline{\mathbf{p}}_j := \mathsf{Encode}_{\eta,q}^{\mathcal{B}}(\mathbf{p}_j)$.*

For $\overline{\mathsf{op}} = (\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u})$, define $\mathsf{DECODE}(\mathsf{pp}, t, \overline{\mathsf{op}})$ as in Vrfy of Figure 6 but without checks: $\mathsf{hint}_\tau := \mathbf{B}\cdot\mathbf{u} \in \mathcal{R}_q^{\omega}$, and for $j = \{\tau, \tau-1, \ldots, 1\}$, recursively compute $\mathbf{p}_j := \mathsf{Decode}_{\eta,q}^{\mathcal{B}}(\overline{\mathbf{p}}_j, \mathsf{hint}_j)$ and $\mathsf{hint}_{j-1} := \mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j \in \mathcal{R}_q^{\omega}$, and finally output $\mathsf{op} = (\mathbf{p}_1, \ldots, \mathbf{p}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u})$.

In the following, we provide a lemma that is derived from item 5 of Thoerem 25 in [16].

Lemma 5.2. *Let $\mathsf{op} = (\mathbf{p}_1, \ldots, \mathbf{p}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u}) \in \mathcal{S}_{\mathsf{open,un}}^t$, and $(\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u}) := \overline{\mathsf{op}} := \mathsf{Encode}_{\mathsf{op}}(\mathsf{pp}, t, \mathsf{op})$. Let $c \in \mathcal{S}_{\mathsf{com}}$ be any (possibly maliciously generated) comment and let Vrfy be as in Figure 6. If $\mathsf{Vrfy}(\mathsf{pp}, c, t, \mathsf{op}, \beta) \neq \perp$ for some β, then $\|\overline{\mathbf{p}}_i\|_\infty < \frac{\beta}{2\eta} + \frac{1}{2}$ for all $i \in [\tau]$.*

Definition 5.10. *Fix pp and t. Let $\mathcal{S}_{\mathsf{open}}^t$ be the set of encoded openings $\overline{\mathsf{op}}$ of the form $\overline{\mathsf{op}} = (\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u})$. Define for $w \in \mathcal{R}$ and $\overline{\mathsf{op}}, \overline{\mathsf{op}}' \in \mathcal{S}_{\mathsf{open}}^t$:*

$$w \odot \overline{\mathsf{op}} := \mathsf{ENCODE}\big(\mathsf{pp}, t,\ w\cdot\mathsf{DECODE}(\mathsf{pp}, t, \overline{\mathsf{op}})\big), \tag{70}$$

$$\overline{\mathsf{op}} \oplus \overline{\mathsf{op}}' := \mathsf{ENCODE}\big(\mathsf{pp}, t, \mathsf{op} + \mathsf{op}'\big), \tag{71}$$

$\underline{\mathsf{Setup}(1^\lambda)}$

(1) sample $\mathbf{B} \leftarrow \mathcal{R}_q^{\omega\times\rho\nu\kappa'}$
(2) sample $\mathbf{A}_0, \mathbf{A}_1 \leftarrow \mathcal{R}_q^{\omega\times\omega\kappa}$
(3) **return** $\mathsf{pp} := (\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1)$

$\underline{\mathsf{Com}(\mathsf{pp}, \widetilde{\mathbf{M}})}$

(1) $\mathbf{p}_0 := \mathsf{label}_{\mathbf{B},\mathbf{A}_0,\mathbf{A}_1}(\widetilde{\mathbf{M}}, \varepsilon)$
(2) $c := \mathsf{proj}_q(\mathbf{p}_0) \in \mathcal{R}_q^\omega$
(3) **return** c

$\underline{\mathsf{Open}(\mathsf{pp}, c, \widetilde{\mathbf{M}}, t)}$

(1) $\tilde{t} := \mathsf{bin}_\tau(t-1)$
(2) **for** $1 \le j \le \tau$ **do**
  (a) $\mathbf{p}_j := \mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j} \,\|\, \tilde{t}_j)$
  (b) $\overline{\mathbf{p}}_j := \mathsf{Encode}^{\mathcal{B}}_{\eta,q}(\mathbf{p}_j)$
  (c) $\mathbf{s}_j := \mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j} \,\|\, (\tilde{t}_j \oplus 1))$
(3) $\mathbf{u} := \mathsf{dec}_{q'}(\mathsf{vec}(\mathbf{M}_t)) \in \mathcal{R}^{\rho\nu\kappa'}$
(4) **return** $\overline{\mathsf{op}} := ((\overline{\mathbf{p}}_j)_j, (\mathbf{s}_j)_j, \mathbf{u})$

$\underline{\mathsf{iVrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}})}$

(1) **return** $\mathsf{Vrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}}, \eta)$

$\underline{\mathsf{sVrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}})}$

(1) **return** $\mathsf{Vrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}}, \beta_{\mathsf{agg}})$

$\underline{\mathsf{wVrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}})}$

(1) **return** $\mathsf{Vrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}}, 2\beta_{\mathsf{agg}})$

$\underline{\mathsf{Vrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}}, \beta)}$

(1) $\tilde{t} := \mathsf{bin}_\tau(t-1)$; parse $\overline{\mathsf{op}}$ as $((\overline{\mathbf{p}}_j)_j, (\mathbf{s}_j)_j, \mathbf{u})$
(2) **if** $\|\mathbf{u}\|_\infty > \beta$ **return** $\perp$
(3) $\mathsf{hint}_\tau \leftarrow \mathbf{B}\cdot\mathbf{u} \in \mathcal{R}_q^\omega$
(4) **for** $j = \tau, \tau-1, \ldots, 1$ **do**
  (a) $\mathbf{p}_j \leftarrow \mathsf{Decode}^{\mathcal{B}}_{\eta,q}(\overline{\mathbf{p}}_j, \mathsf{hint}_j)$
  (b) **if** $\|\mathbf{p}_j\|_\infty > \beta$ **or** $\|\mathbf{s}_j\|_\infty > \beta$ **return** $\perp$
  (c) **if** $\|\mathsf{proj}_{\eta,\kappa}(\mathbf{p}_j)\|_\infty > \frac{q\beta}{2\eta}$ **or** $\|\mathsf{proj}_{\eta,\kappa}(\mathbf{s}_j)\|_\infty > \frac{q\beta}{2\eta}$ **return** $\perp$
  (d) $\mathsf{hint}_{j-1} \leftarrow \mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j$
(5) **if** $c \ne \mathsf{hint}_0$ **return** $\perp$
(6) **return** $\mathsf{vec}^{-1}(\mathsf{proj}_{q'}(\mathbf{u})) \in \mathcal{R}_{q'}^{\rho\times\nu}$

Figure 6: Our HVC Construction Π_{HVC}.

where $\mathsf{op} := \mathsf{DECODE}(\mathsf{pp}, t, \overline{\mathsf{op}})$ *and* $\mathsf{op}' := \mathsf{DECODE}(\mathsf{pp}, t, \overline{\mathsf{op}}')$ *the* $+$ *and scalar multiplication on the right-hand side are taken componentwise on the unencoded tuple* $(\mathbf{p}_1, \ldots, \mathbf{p}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u})$.

Lemma 5.3 (Correctness of DECODE ∘ ENCODE). *For any* pp, t *and any unencoded opening* $\mathsf{op} = (\mathbf{p}_1, \ldots, \mathbf{p}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u})$, *let* $\overline{\mathsf{op}} := \mathsf{ENCODE}(\mathsf{pp}, t, \mathsf{op})$. *Then* $\mathsf{DECODE}(\mathsf{pp}, t, \overline{\mathsf{op}}) = \mathsf{op}$.

Proof. By definition, DECODE computes $\mathsf{hint}_\tau = \mathbf{B}\cdot\mathbf{u}$ and then iterates downward. Assume inductively that at some level j we have $\mathsf{hint}_j = \mathsf{proj}_q(\mathbf{p}_j)$. Since $\overline{\mathbf{p}}_j = \mathsf{Encode}^{\mathcal{B}}_{\eta,q}(\mathbf{p}_j)$, Lemma 5.1 (Item 2) gives $\mathsf{Decode}^{\mathcal{B}}_{\eta,q}(\overline{\mathbf{p}}_j, \mathsf{hint}_j) = \mathbf{p}_j$. Then DECODE sets

$$\mathsf{hint}_{j-1} = \mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j = \mathsf{proj}_q(\mathbf{p}_{j-1}), \tag{72}$$

where the last equality is exactly the projection identity induced by the labeling recursion. The base case $j = \tau$ holds because $\mathsf{proj}_q(\mathbf{p}_\tau) = \mathbf{B}\cdot\mathbf{u}$ for honest leaves. Thus DECODE recovers every $\mathbf{p}_j$ and outputs op. □

We are now ready to show our constructed HVC satisfies all the Definitions in subsection Section 5.1.

Theorem 5.1. *Let* $d, q, q', \alpha_w, N, \eta, \tau, \rho, \nu, \omega, \beta_{\mathsf{agg}} \in \mathbb{N}^+$ *such that* d *is a power of* 2 *and* q, q' *are primes. Let* $\epsilon_{\mathsf{hom}} \in (0,1)$. *Let*

$$\beta_{\mathsf{agg}} \;\ge\; \eta\sqrt{2\alpha_w N\Big(\ln\frac{2d}{\epsilon_{\mathsf{hom}}} + \ln\big(N\cdot(2\tau\omega\kappa + \rho\nu\kappa' + 2\tau\omega)\big)\Big)}.$$

If $\mathsf{MSIS}^\infty_{q,\omega,2\omega\kappa,4\beta_{\mathsf{agg}}}$ *and* $\mathsf{MSIS}^\infty_{q,\omega,\rho\nu\kappa',4\beta_{\mathsf{agg}}}$ *problems are hard, then the HVC construction* Π_{HVC} *in Figure 6 is an individually correct,* $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\mathsf{hom}})$*-probabilistically homomorphic, robustly homomorphic and position binding for domain* $\mathcal{R}_{q'}^{\rho\times\nu}$ *and vector length* 2^τ.

Proof. The proof of the theorem follows from Lemma 5.4, Lemma 5.5, Lemma 5.6, and Lemma 5.7 proven below. □

Lemma 5.4. *Let* $d, q, q', \rho, \nu, \omega, \eta, \tau \in \mathbb{N}^+$ *such that* d *is a power of* 2 *and* q, q' *are primes. Let* $\mathcal{R}_q, \mathcal{R}_{q'}$ *be the polynomial rings* $\mathbb{Z}_q[X]/(X^d+1)$ *and* $\mathbb{Z}_{q'}[X]/(X^d+1)$, *respectively. Then the HVC construction* Π_{HVC} *in Figure 6 is an individually correct HVC for domain* $\mathcal{R}_{q'}^{\rho\times\nu}$ *and vector length* 2^τ *as per Definition 5.2.*

Proof. Let $\widetilde{\mathbf{M}} \in (\mathcal{R}_{q'}^{\rho\times\nu})^{2^\tau}$, $c = \mathsf{Com}(\mathsf{pp}, \widetilde{\mathbf{M}})$, $t \in [2^\tau]$, and $(\overline{\mathbf{p}}_1, \ldots, \overline{\mathbf{p}}_\tau, \mathbf{s}_1, \ldots, \mathbf{s}_\tau, \mathbf{u}) = \mathsf{Open}(\mathsf{pp}, c, \widetilde{\mathbf{M}}, t)$. Let $\tilde{t} = \mathsf{bin}_\tau(t-1)$ be as in the definition of Open. To disambiguate, we temporarily denote the equally named values $\mathbf{p}_j$ appearing in both Open and Vrfy by $\mathbf{p}_j^{\mathsf{Open}}$ and $\mathbf{p}_j^{\mathsf{Vrfy}}$ for $j \in [\tau]$. For notational convenience in the proof, we abuse $\mathbf{p}_0^{\mathsf{Open}}$ to denote the computation of $\mathbf{p}_0$ in Com. We first observe that for all $j \in [\tau]$ it holds in $\mathcal{R}_q^\omega$ that

$$\mathsf{proj}_q(\mathbf{p}_{j-1}^{\mathsf{Open}}) \tag{73}$$
$$= \mathsf{proj}_q\big(\mathsf{label}_{\mathbf{B},\mathbf{A}_0,\mathbf{A}_1}(\widetilde{\mathbf{M}}, \tilde{t}_{<j})\big) \tag{74}$$
$$= \mathsf{proj}_q\Big(\mathsf{dec}_q\big(\mathbf{A}_0\cdot\mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j}\|0) + \mathbf{A}_1\cdot\mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j}\|1)\big)\Big) \tag{75}$$
$$= \mathbf{A}_0\cdot\mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j}\|0) \;+\; \mathbf{A}_1\cdot\mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j}\|1) \tag{76}$$
$$= \mathbf{A}_{\tilde{t}_j}\cdot\mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j}\|\tilde{t}_j) \;+\; \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathsf{label}(\widetilde{\mathbf{M}}, \tilde{t}_{<j}\|(\tilde{t}_j\oplus 1)) \tag{77}$$
$$= \mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j^{\mathsf{Open}} \;+\; \mathbf{A}_{\tilde{t}_{j\oplus 1}}\cdot\mathbf{s}_j. \tag{78}$$

The first equality comes from the definition of Com and Open. The second equality comes from label, $|\tilde{t}_{<j}| < \tau$. The third equality comes from Item 3 of Proposition 5.3. The final equality comes from the definition of Open. Observe that for $j = 1$ this yields

$$c = \mathsf{proj}_q(\mathbf{p}_0^{\mathsf{Open}}) = \mathbf{A}_{\tilde{t}_1}\cdot\mathbf{p}_1^{\mathsf{Open}} + \mathbf{A}_{\tilde{t}_1\oplus 1}\cdot\mathbf{s}_1 \quad \text{in } \mathcal{R}_q^\omega. \tag{79}$$

Further it holds that

$$\mathsf{proj}_q(\mathbf{p}_\tau^{\mathsf{Open}}) = \mathsf{proj}_q\big(\mathsf{label}_{\mathbf{B},\mathbf{A}_0,\mathbf{A}_1}(\mathbf{M}, \tilde{t})\big) \tag{80}$$
$$= \mathsf{proj}_q\Big(\mathsf{dec}_q\big(\mathbf{B}\cdot\mathsf{dec}_{q'}(\mathsf{vec}(\mathbf{M}_t))\big)\Big) \tag{81}$$
$$= \mathbf{B}\cdot\mathsf{dec}_{q'}(\mathsf{vec}(\mathbf{M}_t)) \tag{82}$$
$$= \mathbf{B}\cdot\mathbf{u}. \tag{83}$$

where the first and last equalities hold by definition of Open. The second equality holds true by Def. of label, leaf case and the second last equality holds by Item 3 of Proposition 5.3. We argue that, on an honestly generated opening, the verifier decodes the same vectors $\mathbf{p}_j$ that were used by the opener, i.e., $\mathbf{p}_j^{\mathsf{Vrfy}} = \mathbf{p}_j^{\mathsf{Open}}$ for all $j \in [\tau]$. To this end, we show the following invariant for the hints computed by Vrfy:

$$(\star_j) \qquad \mathsf{hint}_j \;=\; \mathsf{proj}_q(\mathbf{p}_j^{\mathsf{Open}}) \in \mathcal{R}_q^\omega \quad \text{for all } j \in \{0, 1, \ldots, \tau\}.$$

We prove $(\star_j)$ by backward induction on j:

- *Base case* $(j = \tau)$. In Vrfy the verifier sets $\mathsf{hint}_\tau = \mathbf{B}\cdot\mathbf{u}$. By Equation (80), we have $\mathsf{proj}_q(\mathbf{p}_\tau^{\mathsf{Open}}) = \mathbf{B}\cdot\mathbf{u}$, hence $(\star_\tau)$ holds.
- *Inductive step.* Fix $j \in \{1, \ldots, \tau\}$, we assume $(\star_j)$, i.e., $\mathsf{hint}_j = \mathsf{proj}_q(\mathbf{p}_j^{\mathsf{Open}})$, and prove $(\star_{j-1})$, i.e., $\mathsf{hint}_{j-1} = \mathsf{proj}_q(\mathbf{p}_{j-1}^{\mathsf{Open}})$. In Vrfy, we have

$$\mathbf{p}_j^{\mathsf{Vrfy}} = \mathsf{Decode}^{\mathcal{B}}_{\eta,q}(\overline{\mathbf{p}}_j, \mathsf{hint}_j) \tag{84}$$
$$= \mathsf{Decode}^{\mathcal{B}}_{\eta,q}\big(\mathsf{Encode}^{\mathcal{B}}_{\eta,q}(\mathbf{p}_j^{\mathsf{Open}}), \mathsf{proj}_q(\mathbf{p}_j^{\mathsf{Open}})\big) \tag{85}$$
$$= \mathbf{p}_j^{\mathsf{Open}}, \tag{86}$$

where the first equality is satisfied by Def. of Vrfy, the second equality is true by Def. of Open and hypothesis and last equality comes from item 2 of Lemma 5.1. We obtain

$$\mathsf{hint}_{j-1} = \mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j^{\mathsf{Vrfy}} + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j \tag{87}$$

$$= \mathbf{A}_{\tilde{t}_j} \cdot \mathbf{p}_j^{\mathsf{Open}} + \mathbf{A}_{\tilde{t}_j \oplus 1} \cdot \mathbf{s}_j \tag{88}$$

$$= \mathsf{proj}_q(\mathbf{p}_{j-1}^{\mathsf{Open}}), \tag{89}$$

where the first equality is satisfied by Def. of Vrfy and the second and third equalities are held by Equation (86) and Equation (78), respectively. This exactly proves $(\star_{j-1})$. This completes the backward induction and establishes $(\star_j)$ for all j. In particular, $(\star_0)$ implies that on honestly generated openings the verifier obtains

$$\mathsf{hint}_0 \;=\; \mathsf{proj}_q(\mathbf{p}_0^{\mathsf{Open}}) \;=\; c, \tag{90}$$

and hence the check $c \stackrel{?}{=} \mathsf{hint}_0$ succeeds.

It only remains to check that the norm bounds are not violated for iVrfy, i.e. for $\beta = \eta$. For every $j \in [\tau]$, the vectors $\mathbf{p}_j^{\mathsf{Open}}$ and $\mathbf{s}_j$ are outputs of label and thus (depending on the node depth) are obtained by applying dec_q to an element of $\mathcal{R}_q^\omega$. Similarly, $\mathbf{u} = \mathsf{dec}_{q'}(\mathsf{vec}(\mathbf{M}_t))$ is obtained by applying $\mathsf{dec}_{q'}$ to an element of $\mathcal{R}_{q'}^{\rho\nu}$. By design of dec_q and $\mathsf{dec}_{q'}$ (base $2\eta+1$), all resulting coefficients lie in $\{-\eta, \ldots, \eta\}$, and therefore $\|\mathbf{p}_j^{\mathsf{Vrfy}}\|_\infty \le \eta$, $\|\mathbf{s}_j\|_\infty \le \eta$, and $\|\mathbf{u}\|_\infty \le \eta$. Moreover, applying $\mathsf{proj}_{\eta,\kappa}$ to $\mathbf{p}_j^{\mathsf{Vrfy}}$ or $\mathbf{s}_j$ yields the centered representative of the decomposed element, so

$$\left\|\mathsf{proj}_{\eta,\kappa}(\mathbf{p}_j^{\mathsf{Vrfy}})\right\|_\infty,\ \left\|\mathsf{proj}_{\eta,\kappa}(\mathbf{s}_j)\right\|_\infty \;\le\; \frac{q-1}{2}. \tag{91}$$

Thus all checks in $\mathsf{iVrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}}) = \mathsf{Vrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}}, \eta)$ pass, and the algorithm outputs

$$\mathsf{vec}^{-1}\big(\mathsf{proj}_{q'}(\mathbf{u})\big) = \mathsf{vec}^{-1}\big(\mathsf{proj}_{q'}(\mathsf{dec}_{q'}(\mathsf{vec}(\mathbf{M}_t)))\big) = \mathbf{M}_t. \tag{92}$$

This proves individual correctness. □

Lemma 5.5. *Let $d, q, q', \alpha_w, N, \eta, \tau, \rho, \nu, \omega, \beta_{\mathsf{agg}} \in \mathbb{N}^+$ and $\epsilon_{\mathsf{hom}} \in \mathbb{R}$ $(0,1)$ such that d is a power of 2 and q, q' are primes. Let $\mathcal{R}_q, \mathcal{R}_{q'}$ be the polynomial rings $\mathbb{Z}_q[X]/(X^d+1)$ and $\mathbb{Z}_{q'}[X]/(X^d+1)$, respectively. Set $\kappa = \lceil \log_{2\eta+1}(q) \rceil$ and $\kappa' = \lceil \log_{2\eta+1}(q') \rceil$. If $\beta_{\mathsf{agg}} \ge \eta\sqrt{2\alpha_w N\Big(\ln\frac{2d}{\epsilon_{\mathsf{hom}}} + \ln(2\tau\omega\kappa + \rho\nu\kappa' + 2\tau\omega)\Big)}$, then the HVC construction Π_{HVC} in Figure 6 is $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\mathsf{hom}})$-probabilistically homomorphic for domain $\mathcal{R}_{q'}^{\rho\times\nu}$ and vector length 2^τ as per Definition 5.3.*

Proof. Let $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and fix $t \in [2^\tau]$ with $\tilde{t} = \mathsf{bin}_\tau(t-1)$. Let $(c_i, \overline{\mathsf{op}}_i)$ for $i \in [N]$ be such that $\mathsf{iVrfy}(\mathsf{pp}, c_i, t, \overline{\mathsf{op}}_i) = \mathbf{M}_i \neq \perp$. Let $w_1, \ldots, w_N \leftarrow \mathcal{T}_{\alpha_w}$ be arbitrary weights, and define

$$c^* := \sum_{i=1}^N w_i \cdot c_i \in \mathcal{R}_q^\omega, \qquad \overline{\mathsf{op}}^* := \bigoplus_{i=1}^N \big(w_i \odot \overline{\mathsf{op}}_i\big) \in \mathcal{S}_{\mathsf{open}}^t. \tag{93}$$

For each $i \in [N]$, define

$$\mathsf{op}_i := \mathsf{DECODE}(\mathsf{pp}, t, \overline{\mathsf{op}}_i) = (\mathbf{p}_1^i, \ldots, \mathbf{p}_\tau^i, \mathbf{s}_1^i, \ldots, \mathbf{s}_\tau^i, \mathbf{u}^i) \tag{94}$$

and $\mathsf{op}^* := \sum_{i=1}^N w_i \cdot \mathsf{op}_i = (\mathbf{p}_1^*, \ldots, \mathbf{p}_\tau^*, \mathbf{s}_1^*, \ldots, \mathbf{s}_\tau^*, \mathbf{u}^*)$, where

$$\mathbf{p}_j^* = \sum_i w_i \mathbf{p}_j^i, \quad \mathbf{s}_j^* = \sum_i w_i \mathbf{s}_j^i, \quad \mathbf{u}^* = \sum_i w_i \mathbf{u}^i. \tag{95}$$

By Definition 5.10 and Lemma 5.3, $\mathsf{op}^* = \mathsf{DECODE}(\mathsf{pp}, t, \overline{\mathsf{op}}^*)$.

First, we check that the linear relations in sVrfy hold deterministically for the aggregated opening. Since $\mathsf{iVrfy}(\mathsf{pp}, c_i, t, \overline{\mathsf{op}}_i) \neq \perp$ for all $i \in [N]$, the individual verification passes, satisfying the verifier's decoding recursion and producing hints $\mathsf{hint}_j^i \in \mathcal{R}_q^\omega$ such that $\mathsf{hint}_0^i = c_i$, and for all $j \in [\tau]$,

$$\mathsf{hint}_{j-1}^i = \mathbf{A}_{\tilde{t}_j} \mathbf{p}_j^i + \mathbf{A}_{\tilde{t}_j \oplus 1} \mathbf{s}_j^i, \tag{96}$$

$$\mathsf{hint}_\tau^i = \mathbf{B} \cdot \mathbf{u}^i. \tag{97}$$

Define the aggregated hints as $\mathsf{hint}_j^* := \sum_{i=1}^N w_i \cdot \mathsf{hint}_j^i$. Then $\mathsf{hint}_0^* = \sum_{i=1}^N w_i c_i = c^*$. For every $j \in [\tau]$, expanding Equation (96) gives:

$$\mathsf{hint}_{j-1}^* = \sum_{i=1}^N w_i \big(\mathbf{A}_{\tilde{t}_j} \mathbf{p}_j^i + \mathbf{A}_{\tilde{t}_j \oplus 1} \mathbf{s}_j^i\big) \tag{98}$$

$$= \mathbf{A}_{\tilde{t}_j}\Big(\sum_{i=1}^N w_i \mathbf{p}_j^i\Big) + \mathbf{A}_{\tilde{t}_j \oplus 1}\Big(\sum_{i=1}^N w_i \mathbf{s}_j^i\Big) \tag{99}$$

$$= \mathbf{A}_{\tilde{t}_j} \mathbf{p}_j^* + \mathbf{A}_{\tilde{t}_j \oplus 1} \mathbf{s}_j^*, \tag{100}$$

and identically for the final hint using Equation (97):

$$\mathsf{hint}_\tau^* = \sum_{i=1}^N w_i (\mathbf{B} \cdot \mathbf{u}^i) = \mathbf{B} \cdot \mathbf{u}^*. \tag{101}$$

Furthermore, for each individual signature, $\mathbf{M}_i = \mathsf{proj}_{q'}(\mathbf{u}^i)$. The aggregated message checks out as:

$$\mathsf{proj}_{q'}(\mathbf{u}^*) = \mathsf{proj}_{q'}\Big(\sum_{i=1}^N w_i \mathbf{u}^i\Big) = \sum_{i=1}^N w_i \cdot \mathsf{proj}_{q'}(\mathbf{u}^i) \tag{102}$$

$$= \sum_{i=1}^N w_i \mathbf{M}_i = \mathbf{M}^*. \tag{103}$$

Thus, all recursive algebraic checks required by sVrfy hold deterministically.

Next, we bound the failure probability of the norm checks. Writing out the conditions (cf. $\mathsf{sVrfy}(\mathsf{pp}, \cdot, \cdot, \cdot) = \mathsf{Vrfy}(\mathsf{pp}, \cdot, \cdot, \cdot, \beta_{\mathsf{agg}})$),

$$P := \Pr\Big[\, w_1, \ldots, w_N \leftarrow \mathcal{T}_{\alpha_w} :\ \exists j \in [\tau] \text{ s.t.} \tag{104}$$

$$\left\|\sum_{i=1}^N w_i \cdot \mathbf{p}_j^i\right\|_\infty > \beta_{\mathsf{agg}} \tag{105}$$

$$\vee \left\|\sum_{i=1}^N w_i \cdot \mathbf{s}_j^i\right\|_\infty > \beta_{\mathsf{agg}} \tag{106}$$

$$\vee \left\|\sum_{i=1}^N w_i \cdot \mathbf{u}^i\right\|_\infty > \beta_{\mathsf{agg}} \tag{107}$$

$$\vee \left\|\sum_{i=1}^N w_i \cdot \mathsf{proj}_{\eta,\kappa}(\mathbf{p}_j^i)\right\|_\infty > \frac{q\beta_{\mathsf{agg}}}{2\eta} \tag{108}$$

$$\vee \left\|\sum_{i=1}^N w_i \cdot \mathsf{proj}_{\eta,\kappa}(\mathbf{s}_j^i)\right\|_\infty > \frac{q\beta_{\mathsf{agg}}}{2\eta}\ \Big]. \tag{109}$$

Observe that this is an $\|\cdot\|_\infty$-bound for a total of

$$N_{\mathsf{bounds}} \;=\; 2\tau\omega\kappa + \rho\nu\kappa' + 2\tau\omega$$

many ring elements. For each of these N_{bounds} ring elements, we apply Lemma 2.3 with growth factor $\zeta = \beta_{\text{agg}}/\eta$. Taking an N_{bounds}-fold union bound gives

$$P \;\le\; N_{\text{bounds}} \cdot 2d \cdot \exp\Big(-\frac{\beta_{\text{agg}}^2}{2\eta^2\alpha_w N}\Big).$$

By assumption, we have

$$\frac{\beta_{\text{agg}}^2}{2\eta^2\alpha_w N} \;\ge\; \ln\Big(\frac{2d}{\epsilon_{\text{hom}}}\cdot(2\tau\omega\kappa + \rho\nu\kappa' + 2\tau\omega)\Big) = \ln\Big(\frac{2d}{\epsilon_{\text{hom}}}\cdot N_{\text{bounds}}\Big),$$

and hence $P \le \epsilon_{\text{hom}}$. This proves $(N, \mathcal{T}_{\alpha_w}, \epsilon_{\text{hom}})$-probabilistic homomorphism. □

Lemma 5.6. *Let $d, q, q', \alpha_w, N, \eta, \tau, \rho, \nu, \omega, \beta_{\text{agg}} \in \mathbb{N}^+$ such that d is a power of 2 and q, q' are primes. Let $\mathcal{R}_q, \mathcal{R}_{q'}$ be the polynomial rings $\mathbb{Z}_q[X]/(X^d+1)$ and $\mathbb{Z}_{q'}[X]/(X^d+1)$, respectively. Then the HVC construction Π_{HVC} in Figure 6 is robustly homomorphic as per Definition 5.3.*

Proof. Let $t \in [2^\tau]$ with $\tilde{t} := \text{bin}_\tau(t-1)$. Let $c_0, c_1 \in \mathcal{R}_q^\omega$ and $\overline{\text{op}}_0, \overline{\text{op}}_1 \in \mathcal{S}^t_{\text{open}}$ be arbitrary such that

$$\text{sVrfy}(\text{pp}, c_0, t, \overline{\text{op}}_0) = \mathbf{M}_0 \text{ and } \text{sVrfy}(\text{pp}, c_1, t, \overline{\text{op}}_1) = \mathbf{M}_1 \tag{110}$$

with $\mathbf{M}_0, \mathbf{M}_1 \neq \perp$. We need to show that

$$\text{wVrfy}(\text{pp}, c_0 - c_1, t, \overline{\text{op}}_0 \ominus \overline{\text{op}}_1) = \mathbf{M}_0 - \mathbf{M}_1, \tag{111}$$

where $\ominus$ is the subtraction in $\mathcal{S}^t_{\text{open}}$ opposed to $\oplus$. For $i \in \{0,1\}$, parse $\overline{\text{op}}_i = (\overline{\mathbf{p}}_1^i, \ldots, \overline{\mathbf{p}}_\tau^i,\ \mathbf{s}_1^i, \ldots, \mathbf{s}_\tau^i, \mathbf{u}^i)$. Let

$$\text{op}_i := \text{DECODE}(\text{pp}, t, \overline{\text{op}}_i) = (\mathbf{p}_1^i, \ldots, \mathbf{p}_\tau^i, \mathbf{s}_1^i, \ldots, \mathbf{s}_\tau^i, \mathbf{u}^i). \tag{112}$$

Since $\text{sVrfy}(\text{pp}, c_i, t, \overline{\text{op}}_i) \neq \perp$, the verifier's decoding recursion is satisfied and produces hints $\text{hint}_j^i \in \mathcal{R}_q^\omega$ such that:

(1) $\text{hint}_\tau^i = \mathbf{B}\cdot\mathbf{u}^i$,
(2) for all $j \in [\tau]$,

$$\text{hint}_{j-1}^i = \mathbf{A}_{\tilde{t}_j}\cdot\text{Decode}^{\mathcal{B}}_{\eta,q}(\overline{\mathbf{p}}_j, \text{hint}_j) + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j^i \tag{113}$$

$$= \mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j^i + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j^i. \tag{114}$$

(3) $\text{hint}_0^i = c_i$.

Now define

$$\text{op}^* := \text{op}_0 - \text{op}_1 \tag{115}$$

$$(\text{i.e., } \mathbf{p}_j^* = \mathbf{p}_j^0 - \mathbf{p}_j^1,\ \mathbf{s}_j^* = \mathbf{s}_j^0 - \mathbf{s}_j^1,\ \mathbf{u}^* = \mathbf{u}^0 - \mathbf{u}^1), \tag{116}$$

and

$$\overline{\text{op}}^* := \text{ENCODE}(\text{pp}, t, \text{op}^*) = \overline{\text{op}}_0 \ominus \overline{\text{op}}_1 \in \mathcal{S}^t_{\text{open}}. \tag{117}$$

Let $c^* := c_0 - c_1 \in \mathcal{R}_q^\omega$, and define hints $\text{hint}_j^* := \text{hint}_j^0 - \text{hint}_j^1$. Then for every $j \in [\tau]$, by linearity in $\mathcal{R}_q^\omega$ we obtain

$$\text{hint}_{j-1}^* = \text{hint}_{j-1}^0 - \text{hint}_{j-1}^1 \tag{118}$$

$$= \Big(\mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j^0 + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j^0\Big) - \Big(\mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j^1 + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j^1\Big) \tag{119}$$

$$= \mathbf{A}_{\tilde{t}_j}(\mathbf{p}_j^0 - \mathbf{p}_j^1) + \mathbf{A}_{\tilde{t}_j\oplus 1}(\mathbf{s}_j^0 - \mathbf{s}_j^1) \tag{120}$$

$$= \mathbf{A}_{\tilde{t}_j}\cdot\mathbf{p}_j^* + \mathbf{A}_{\tilde{t}_j\oplus 1}\cdot\mathbf{s}_j^*, \tag{121}$$

where the second equality is due to Equation (114). In particular, $\text{hint}_0^* = c_0 - c_1$.

Moreover, Lemma 5.3 implies $\text{DECODE}(\text{pp}, t, \overline{\text{op}}^*) = \text{op}^*$. Hence, when Vrfy runs its decoding loop on input $\overline{\text{op}}^*$, it reconstructs $\mathbf{p}_j^{\text{Vrfy}} = \mathbf{p}_j^*$ for all $j \in [\tau]$ (and similarly $\mathbf{s}_j^{\text{Vrfy}} = \mathbf{s}_j^*$ and $\mathbf{u}^{\text{Vrfy}} = \mathbf{u}^*$), and therefore all linear (mod-q) equalities checked by Vrfy hold deterministically for $(c^*, \overline{\text{op}}^*)$. Because sVrfy accepts, for each $i \in \{0,1\}$ and every $j \in [\tau]$ we have:

$$\|\mathbf{p}_j^i\|_\infty \le \beta_{\text{agg}}, \quad \|\mathbf{s}_j^i\|_\infty \le \beta_{\text{agg}}, \quad \text{and} \quad \|\mathbf{u}^i\|_\infty \le \beta_{\text{agg}}, \tag{122}$$

as well as

$$\big\|\text{proj}_{\eta,\kappa}(\mathbf{p}_j^i)\big\|_\infty \le \frac{q\beta_{\text{agg}}}{2\eta} \text{ and } \big\|\text{proj}_{\eta,\kappa}(\mathbf{s}_j^i)\big\|_\infty \le \frac{q\beta_{\text{agg}}}{2\eta}. \tag{123}$$

From (122) and (123), we obtain

$$\|\mathbf{p}_j^*\|_\infty \le 2\beta_{\text{agg}}, \|\mathbf{s}_j^*\|_\infty \le 2\beta_{\text{agg}}, \|\mathbf{u}^*\|_\infty \le 2\beta_{\text{agg}}. \tag{124}$$

$$\big\|\text{proj}_{\eta,\kappa}(\mathbf{p}_j^*)\big\|_\infty \le \frac{q\beta_{\text{agg}}}{\eta} \text{ and } \big\|\text{proj}_{\eta,\kappa}(\mathbf{s}_j^*)\big\|_\infty \le \frac{q\beta_{\text{agg}}}{\eta}. \tag{125}$$

Thus all checks in $\text{wVrfy}(\text{pp}, c_0 - c_1, t, \overline{\text{op}}_0 \ominus \overline{\text{op}}_1)$ go through and output

$$\text{vec}^{-1}(\text{proj}_{q'}(\mathbf{u}^*)) \tag{126}$$

$$= \text{vec}^{-1}(\text{proj}_{q'}(\mathbf{u}_0)) - \text{vec}^{-1}(\text{proj}_{q'}(\mathbf{u}_1)) \tag{127}$$

$$= \mathbf{M}_0 - \mathbf{M}_1, \tag{128}$$

as required. □

Lemma 5.7. *Let $d, q, q', \alpha_w, N, \eta, \tau, \rho, \nu, \omega, \beta_{\text{agg}} \in \mathbb{N}^+$ and $\epsilon_{\text{hom}} \in (0,1)$ such that d is a power of 2 and q, q' are primes. Let $\mathcal{R}_q, \mathcal{R}_{q'}$ be the polynomial rings $\mathbb{Z}_q[X]/(X^d+1)$ and $\mathbb{Z}_{q'}[X]/(X^d+1)$, respectively. If $\text{MSIS}^\infty_{q,\,\omega,\,2\omega\kappa,\,4\beta_{\text{agg}}}$ and $\text{MSIS}^\infty_{q,\,\omega,\,\rho\nu\kappa',\,4\beta_{\text{agg}}}$ are hard, then the construction in Figure 6 is position binding of Definition Definition 5.5.*

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

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

$$\Pr\big[\text{Bad} \wedge \text{proj}_q(\mathbf{p}_\tau^0) = \text{proj}_q(\mathbf{p}_\tau^1)\big] + \Pr\big[\text{Bad} \wedge \text{proj}_q(\mathbf{p}_\tau^0) \neq \text{proj}_q(\mathbf{p}_\tau^1)\big],$$

where Bad denotes the event $\mathbf{M}_0 \neq \mathbf{M}_1$ and $\perp \notin \{\mathbf{M}_0, \mathbf{M}_1\}$, and $\overline{\text{op}}^b = (\overline{\mathbf{p}}_1^b, \ldots, \overline{\mathbf{p}}_\tau^b,\ \mathbf{s}_1^b, \ldots, \mathbf{s}_\tau^b,\ \mathbf{u}^b)$ for $b \in \{0,1\}$, and $\mathbf{p}_j^b$ for $j \in [\tau]$ denotes the decoded value appearing in $\text{wVrfy}(\text{pp}, c, t, \overline{\text{op}}^b)$. We now bound the two probabilities separately.

Case 1: $\text{proj}_q(\mathbf{p}_\tau^0) = \text{proj}_q(\mathbf{p}_\tau^1)$**.** Using the definition of wVrfy (in particular, its output is $\text{vec}^{-1}(\text{proj}_{q'}(\mathbf{u}^b))$ on acceptance), we have

$$\begin{aligned}
&\Pr\big[\text{Bad} \ \wedge\ \text{proj}_q(\mathbf{p}_\tau^0) = \text{proj}_q(\mathbf{p}_\tau^1)\big] \\
&\le \Pr\Big[\text{proj}_{q'}(\mathbf{u}^0) \neq \text{proj}_{q'}(\mathbf{u}^1) \wedge \mathbf{B}\cdot\mathbf{u}^0 = \mathbf{B}\cdot\mathbf{u}^1 \wedge \|\mathbf{u}^0\|, \|\mathbf{u}^1\| \le 2\beta_{\text{agg}}\Big] \\
&\le \Pr\Big[\mathbf{u}^0 \neq \mathbf{u}^1 \wedge \mathbf{B}\cdot(\mathbf{u}^0 - \mathbf{u}^1) = \mathbf{0} \bmod q \wedge \|\mathbf{u}^0 - \mathbf{u}^1\| \le 4\beta_{\text{agg}}\Big] \\
&\le \text{negl}(\lambda),
\end{aligned}$$

where the last inequality follows from hardness of $\text{MSIS}^\infty_{q,\omega,\rho\nu\kappa',4\beta_{\text{agg}}}$.

Case 2: $\text{proj}_q(\mathbf{p}_\tau^0) \neq \text{proj}_q(\mathbf{p}_\tau^1)$**.** We construct a PPT algorithm $\overline{\mathcal{A}}$ that solves $\text{MSIS}^\infty_{q,2\omega\kappa,4\beta_{\text{agg}}}$ as follows. On input a uniformly random matrix $\mathbf{A} \in \mathcal{R}_q^{\omega\times 2\omega\kappa}$, the algorithm $\overline{\mathcal{A}}$ parses $\mathbf{A}$ as a horizontal

concatenation $\mathbf{A} = [\mathbf{A}_0 \mid \mathbf{A}_1]$ with $\mathbf{A}_0, \mathbf{A}_1 \in \mathcal{R}_q^{\omega \times \omega\kappa}$, samples $\mathbf{B} \leftarrow \mathcal{R}_q^{\omega \times \rho\nu\kappa'}$, sets $\mathsf{pp} := (\mathbf{B}, \mathbf{A}_0, \mathbf{A}_1)$, and runs

$$(c, t, \overline{\mathsf{op}}_0, \overline{\mathsf{op}}_1) \leftarrow \mathcal{A}(\mathsf{pp}).$$

For $b \in \{0,1\}$, let $\mathbf{M}_b := \mathsf{wVrfy}(\mathsf{pp}, c, t, \overline{\mathsf{op}}_b)$. If $\mathbf{M}_0 = \mathbf{M}_1$, or $\perp \in \{\mathbf{M}_0, \mathbf{M}_1\}$, or $\mathsf{proj}_q(\mathbf{p}_\tau^0) = \mathsf{proj}_q(\mathbf{p}_\tau^1)$, then $\overline{\mathcal{A}}$ aborts. Otherwise, let $j^* \in [\tau]$ be the *largest* index such that

$$\mathsf{proj}_q(\mathbf{p}_{j^*}^0) \neq \mathsf{proj}_q(\mathbf{p}_{j^*}^1), \tag{129}$$

which exists since $\mathsf{proj}_q(\mathbf{p}_\tau^0) \neq \mathsf{proj}_q(\mathbf{p}_\tau^1)$ by assumption, and satisfies $j^* \geq 1$ while $\mathsf{proj}_q(\mathbf{p}_{j^*-1}^0) = \mathsf{proj}_q(\mathbf{p}_{j^*-1}^1)$ by maximality.

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

$$\mathbf{z} := \begin{cases} (\mathbf{p}_{j^*}^0 - \mathbf{p}_{j^*}^1,\ \mathbf{s}_{j^*}^0 - \mathbf{s}_{j^*}^1) & \text{if } \tilde{t}_{j^*} = 0, \\ (\mathbf{s}_{j^*}^0 - \mathbf{s}_{j^*}^1,\ \mathbf{p}_{j^*}^0 - \mathbf{p}_{j^*}^1) & \text{if } \tilde{t}_{j^*} = 1. \end{cases} \tag{130}$$

Since $\mathsf{proj}_q(\mathbf{p}_{j^*-1}^0) = \mathsf{proj}_q(\mathbf{p}_{j^*-1}^1)$ and both openings pass wVrfy, the linear consistency check at level j^* gives

$$\mathsf{proj}_q(\mathbf{p}_{j^*-1}^b) = \mathbf{A}_{\tilde{t}_{j^*}} \cdot \mathbf{p}_{j^*}^b + \mathbf{A}_{\tilde{t}_{j^*} \oplus 1} \cdot \mathbf{s}_{j^*}^b \qquad (b \in \{0,1\}). \tag{131}$$

Subtracting the two equalities and using linearity of proj_q yields

$$\mathbf{A}_{\tilde{t}_{j^*}} \cdot (\mathbf{p}_{j^*}^0 - \mathbf{p}_{j^*}^1) + \mathbf{A}_{\tilde{t}_{j^*} \oplus 1} \cdot (\mathbf{s}_{j^*}^0 - \mathbf{s}_{j^*}^1) = \mathbf{0} \bmod q. \tag{132}$$

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 $\|\mathbf{p}_{j^*}^b\|_\infty \leq 2\beta_{\mathsf{agg}}$ and $\|\mathbf{s}_{j^*}^b\|_\infty \leq 2\beta_{\mathsf{agg}}$ for $b \in \{0,1\}$. Hence,

$$\|\mathbf{z}\|_\infty \leq \max\{\|\mathbf{p}_{j^*}^0\|_\infty, \|\mathbf{s}_{j^*}^0\|_\infty\} + \max\{\|\mathbf{p}_{j^*}^1\|_\infty, \|\mathbf{s}_{j^*}^1\|_\infty\} \tag{133}$$

$$\leq 4\beta_{\mathsf{agg}}. \tag{134}$$

By choice of j^* we have $\mathsf{proj}_q(\mathbf{p}_{j^*}^0) \neq \mathsf{proj}_q(\mathbf{p}_{j^*}^1)$, and thus $\mathbf{p}_{j^*}^0 \neq \mathbf{p}_{j^*}^1$, implying $\mathbf{z} \neq \mathbf{0}$.

Therefore, whenever $\mathcal{A}$ succeeds in event $\mathsf{Bad} \wedge \mathsf{proj}_q(\mathbf{p}_\tau^0) \neq \mathsf{proj}_q(\mathbf{p}_\tau^1)$, the reduction $\overline{\mathcal{A}}$ outputs a nonzero $\mathbf{z}$ with $\|\mathbf{z}\|_\infty \leq 4\beta_{\mathsf{agg}}$ and $\mathbf{A} \cdot \mathbf{z} = \mathbf{0} \bmod q$, solving $\mathsf{MSIS}^\infty_{q,\omega,2\omega\kappa,4\beta_{\mathsf{agg}}}$. Hence,

$$\Pr\left[\mathsf{Bad} \ \wedge\ \mathsf{proj}_q(\mathbf{p}_\tau^0) \neq \mathsf{proj}_q(\mathbf{p}_\tau^1)\right] \ \leq\ \mathsf{negl}(\lambda). \tag{135}$$

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

6 Lemur: Synchronized Multi-Signatures

In this section, we describe our synchronized multi-signature scheme, Lemur. At a high level, our construction follows the modular design patterns introduced by Squirrel [18] and Chipmunk [16]. We utilize a Homomorphic Vector Commitment (HVC) to commit to a signer's collection of one-time public keys and a Key-Homomorphic One-Time Signature (KOTS) to enable the non-interactive aggregation of individual signatures. While the structural "blueprint" remains consistent with these prior works, the underlying security foundations of our building blocks (particularly for KOTS) are fundamentally different (as seen in previous sections), leading to significant performance gains. Together, these building blocks support synchronized signing across 2^τ time steps.

6.1 Formal Definitions

We first formally define synchronized multi-signatures, following the modular framework introduced in [16].

Definition 6.1 (Synchronized N-wise Multi-Signatures). *A synchronized N-wise multi-signature scheme for 2^τ time periods is defined by six PPT algorithms* $\Pi_{\mathsf{MSIG}} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Aggregate}, \mathsf{iVrfy}, \mathsf{aVrfy})$.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$*: The setup algorithm takes as input a security parameter $\lambda \in \mathbb{N}^+$ and outputs public parameters* pp.
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$*: The key generation algorithm takes as input public parameters* pp *and outputs a secret key* sk *and the corresponding public key* pk.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, t, \mu)$*: The signing algorithm takes as input public parameters* pp*, a secret key* sk*, a time period $t \in [2^\tau]$, and a message μ, and outputs a signature σ.*
- $\sigma_{\mathsf{agg}} \leftarrow \mathsf{Aggregate}(\mathsf{pp}, P, t, \mu, S)$*: The aggregation algorithm takes as input public parameters* pp*, a tuple list of public keys P, a time period $t \in [2^\tau]$, a message μ, and a list S of signatures such that $|P| = |S| = N$, and outputs an aggregated signature σ_{agg} or an error symbol $\perp$.*
- $b \leftarrow \mathsf{iVrfy}(\mathsf{pp}, \mathsf{pk}, t, \mu, \sigma)$*: The deterministic individual verification algorithm takes as input public parameters* pp*, a public key* pk*, a time period $t \in [2^\tau]$, a message μ, and a signature σ, and outputs a bit b indicating acceptance/rejection.*
- $b \leftarrow \mathsf{aVrfy}(\mathsf{pp}, P, t, \mu, \sigma_{\mathsf{agg}})$*: The deterministic aggregate verification algorithm takes as input public parameters* pp*, a list P of public keys, a time period $t \in [2^\tau]$, a message μ, and a signature σ_{agg}, and outputs a bit b indicating acceptance/rejection.*

We require that a synchronized N-wise multi-signature satisfies the following three properties: individual correctness, aggregation correctness, and unforgeability.

Definition 6.2 (Individual Correctness). *Let $\epsilon_{\mathsf{cor}} \in [0,1)$. We say that Π_{MSIG} is ϵ_{cor}-individually correct, if for all $\lambda \in \mathbb{N}^+$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$, $t \in [2^\tau]$, $\mu \in \{0,1\}^*$, and $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, t, \mu)$, it holds that $\Pr[\mathsf{iVrfy}(\mathsf{pp}, \mathsf{pk}, t, \mu, \sigma) = 1] \geq 1 - \epsilon_{\mathsf{cor}}$.*

Definition 6.3 (Aggregation Correctness). *We say that Π_{MSIG} has correct aggregation in the presence of rogue keys and signatures if for all $\lambda \in \mathbb{N}^+$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $t \in [2^\tau]$, $\mu \in \{0,1\}^*$, $P = (\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ and $S = (\sigma_1, \ldots, \sigma_N)$ such that for all $i \in [N]$ it holds that $\mathsf{iVrfy}(\mathsf{pp}, \mathsf{pk}_i, t, \mu, \sigma_i) = 1$, we have $\Pr\left[\sigma_{\mathsf{agg}} \leftarrow \mathsf{Aggregate}(\mathsf{pp}, P, t, \mu, S) : \mathsf{aVrfy}(\mathsf{pp}, P, t, \mu, \sigma_{\mathsf{agg}}) = 1\right] \geq 1 - \mathsf{negl}(\lambda)$.*

Definition 6.4 (Unforgeability). *Consider $\mathsf{EUF}_{\Pi_{\mathsf{MSIG}},\mathcal{A}}(\lambda)$ defined in Figure 7. We say that Π_{MSIG} is unforgeable if for all $\lambda \in \mathbb{N}^+$ and all PPT adversaries $\mathcal{A}$, it holds that $\Pr\left[\mathsf{EUF}_{\Pi_{\mathsf{MSIG}},\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda)$.*

6.2 Lemur Construction

In this subsection, we formally describe the Lemur construction (see Figure 8). Conceptually, the scheme is built by composing a key-homomorphic one-time signature (KOTS) and a homomorphic vector commitment (HVC), following the blueprint introduced in Squirrel [18] and Chipmunk [16].

Game $\mathsf{EUF}_{\Pi_{\mathsf{MSIG}},\mathcal{A}}(\lambda)$

(1) $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$.
(2) $(\mathsf{sk}^*, \mathsf{pk}^*) \leftarrow \mathsf{KGen}(\mathsf{pp})$.
(3) Initialize $Q \leftarrow \emptyset$ (set of queried time-message pairs).
(4) $(P^*, t^*, \mu^*, \sigma^*_{\mathsf{agg}}) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{pp},\mathsf{sk}^*,\cdot,\cdot)}(\mathsf{pp}, \mathsf{pk}^*)$.
(5) Output 1 iff:
  (a) $\mathsf{aVrfy}(\mathsf{pp}, P^*, t^*, \mu^*, \sigma^*_{\mathsf{agg}}) = 1$,
  (b) $\mathsf{pk}^* \in P^*$, // Target pk is included in signer set
  (c) $(t^*, \mu^*) \notin Q$, // Forgery was not previously queried

Oracle $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}^*, t, \mu)$

(1) If $Q_t \neq \emptyset$, return $\perp$. // $Q_t := \{(t', \mu) \in Q : t' = t\}$
(2) $Q \leftarrow Q \cup \{(t, \mu)\}$.
(3) Return $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}^*, t, \mu)$.

Figure 7: EUF security game for multi-signatures.

$\underline{\mathsf{Setup}(1^\lambda)}$

(1) $\mathsf{pp}_{\mathsf{KOTS}} \leftarrow \Pi_{\mathsf{KOTS}}.\mathsf{Setup}(1^\lambda)$
(2) $\mathsf{pp}_{\mathsf{HVC}} \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{Setup}(1^\lambda)$
(3) **return** $\mathsf{pp} := (\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{pp}_{\mathsf{HVC}})$

$\underline{\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, t, \mu)}$

(1) $\sigma' \leftarrow \Pi_{\mathsf{KOTS}}.\mathsf{Sign}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{osk}, \mu)$
(2) $\mathsf{op} \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{Open}(\mathsf{pp}_{\mathsf{HVC}}, c, \mathsf{OPK}, t)$
(3) **return** $\sigma := (\sigma', \mathsf{op})$

$\underline{\mathsf{Aggregate}(\mathsf{pp}, P, t, \mu, S)}$

(1) **if** $|S| \neq N$ **or** $|P| \neq N$ **return** $\perp$
(2) **for** $(\mathsf{pk}, \sigma) \in \mathsf{zip}(P, S)$ **do**
(3) **if** $\mathsf{iVrfy}(\mathsf{pp}, \mathsf{pk}, t, \mu, \sigma) = 0$ **return** $\perp$
(4) $j \leftarrow 0$
(5) **do**
  (a) $j \leftarrow j + 1$
  (b) $(w_1, \ldots, w_N) \leftarrow H(t, \mu, P, j)$
  (c) $\sigma' \leftarrow \sum_{i=1}^N w_i \cdot \sigma'_i$
  (d) $\mathsf{op} \leftarrow \sum_{i=1}^N w_i \cdot \mathsf{op}_i$
(6) **while** $\mathsf{aVrfy}(\mathsf{pp}, P, t, \mu, (\sigma', \mathsf{op}, j)) = 0$ **and** $j < \gamma$
(7) **return** $\sigma_{\mathsf{agg}} := (\sigma', \mathsf{op}, j)$

$\underline{\mathsf{KGen}(\mathsf{pp})}$

(1) **for** $i \in [2^\tau]$ **do**
(2) $(\mathsf{osk}_i, \mathsf{opk}_i) \leftarrow \Pi_{\mathsf{KOTS}}.\mathsf{KGen}(\mathsf{pp}_{\mathsf{KOTS}})$
(3) $\mathsf{OSK} := (\mathsf{osk}_1, \ldots, \mathsf{osk}_{2^\tau})$
(4) $\mathsf{OPK} := (\mathsf{opk}_1, \ldots, \mathsf{opk}_{2^\tau})$
(5) $c \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{Com}(\mathsf{pp}_{\mathsf{HVC}}, \mathsf{OPK})$
(6) **return** $(\mathsf{sk}, \mathsf{pk}) := ((\mathsf{OSK}, \mathsf{OPK}), c)$

$\underline{\mathsf{iVrfy}(\mathsf{pp}, \mathsf{pk}, t, \mu, \sigma)}$

(1) Parse $\mathsf{pk} = c$ and $\sigma = (\sigma', \mathsf{op})$
(2) $\mathsf{opk} \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{iVrfy}(\mathsf{pp}_{\mathsf{HVC}}, c, t, \mathsf{op})$
(3) **if** $t > 2^\tau$ **or** $\mathsf{opk} = \perp$ **return** 0
(4) **return** $\Pi_{\mathsf{KOTS}}.\mathsf{iVrfy}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{opk}, \mu, \sigma')$

$\underline{\mathsf{aVrfy}(\mathsf{pp}, P, t, \mu, \sigma_{\mathsf{agg}})}$

(1) Parse $P = (c_1, \ldots, c_N)$ and $\sigma_{\mathsf{agg}} = (\sigma', \mathsf{op}, j)$
(2) $(w_1, \ldots, w_N) \leftarrow H(t, \mu, P, j)$
(3) $c \leftarrow \sum_{i=1}^N w_i \cdot c_i$
(4) $\mathsf{opk} \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{sVrfy}(\mathsf{pp}_{\mathsf{HVC}}, c, t, \mathsf{op})$
(5) **if** $|P| \neq N$ **or** $\mathsf{opk} = \perp$ **return** 0
(6) **return** $\Pi_{\mathsf{KOTS}}.\mathsf{sVrfy}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{opk}, \mu, \sigma')$

Figure 8: Lemur Multi-Signature Construction.

Definition 6.5 (The Lemur Scheme). *Let $d, q, q', \eta, \tau, N, \rho, \nu, \alpha_w, m, n, k, \alpha_H \in \mathbb{N}^+$ such that d is a power of 2, and q and q' are primes. Let Π_{KOTS} be a key-homomorphic one-time signature scheme defined in Figure 3 and Π_{HVC} be a homomorphic vector commitment scheme defined in Figure 6. We define the synchronized multi-signature scheme Lemur $= (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Aggregate}, \mathsf{iVrfy}, \mathsf{aVrfy})$ as specified in Figure 8, with $\rho = k$ and $\nu = n$.*

Theorem 6.1. *Let $\lambda, d, q', N, n, k, \gamma, \tau, \alpha_w \in \mathbb{N}^+$, $\epsilon_{\mathsf{cor}} \in (0, 1)$, and $\epsilon_{\mathsf{hom}} \in (0, \frac{1}{2})$ with d being a power of 2, q' being prime, and $\gamma \geq \lambda / \log_2\left(\frac{1}{2\epsilon_{\mathsf{hom}}}\right)$. Let $\mathcal{R}_{q'} = \mathbb{Z}_{q'}[X]/(X^d + 1)$. Let $W = \mathcal{T}_{\alpha_w} \subseteq \mathcal{R}$ be a set such that $|W| > 2^\lambda$ and let $W' := \{w_0 - w_1 \mid w_0, w_1 \in W\}$. Let $H : \{0,1\}^* \to W^N$ be a random oracle. Let Π_{KOTS} be a key-homomorphic one-time signature scheme with public keys in $\mathcal{R}_{q'}^{k \times n}$ as defined in Figure 3 and let Π_{HVC} be a homomorphic vector commitment for domain $\mathcal{R}_{q'}^{k \times n}$ as defined in Figure 6.*

If Π_{KOTS} is ϵ_{cor}-individually correct, $(N, W, \epsilon_{\mathsf{hom}})$-probabilistically homomorphic, robustly homomorphic, and W'-multi-user existentially unforgeable under rerandomized keys and Π_{HVC} is individually correct, $(N, W, \epsilon_{\mathsf{hom}})$-probabilistically homomorphic, robustly homomorphic, and position binding, then the Lemur construction from Figure 8 is an unforgeable synchronized N-wise multi-signature scheme that is ϵ_{cor}-individually correct and has correct aggregation in the presence of rogue keys and signatures.

Proof. The theorem follows from Lemma 6.1, Lemma 6.2, and Lemma 6.3. □

Lemma 6.1. *Let $\lambda, d, q', n, k, \gamma, N, \tau \in \mathbb{N}^+$ with d being a power of 2, q' being prime. Let $\mathcal{R}_{q'} = \mathbb{Z}_{q'}[X]/(X^d + 1)$. Let Π_{KOTS} be a key-homomorphic one-time signature scheme with public keys in $\mathcal{R}_{q'}^{k \times n}$, and let Π_{HVC} be a homomorphic vector commitment for domain $\mathcal{R}_{q'}^{k \times n}$.*

If Π_{KOTS} is ϵ_{cor}-individually correct and Π_{HVC} is individually correct, then the Lemur construction from Figure 8 is ϵ_{cor}-individually correct.

Proof. Let $\mathsf{pp} = (\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{pp}_{\mathsf{HVC}}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk}) = ((\mathsf{OSK}, \mathsf{OPK}), c) \leftarrow \mathsf{KGen}(\mathsf{pp})$, $t \in [2^\tau]$, $\mu \in \{0,1\}^*$, and $\sigma = (\sigma', \mathsf{op}) = \mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, t, \mu)$, where:

$$\sigma' \leftarrow \Pi_{\mathsf{KOTS}}.\mathsf{Sign}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{osk}_t, \mu), \tag{136}$$

$$\mathsf{op} \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{Open}(\mathsf{pp}_{\mathsf{HVC}}, c, \mathsf{OPK}, t). \tag{137}$$

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

- HVC Verification. It computes $\mathsf{opk} \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{iVrfy}(\mathsf{pp}_{\mathsf{HVC}}, c, t, \mathsf{op})$. By the individual correctness of Π_{HVC}, for any c that is a valid commitment to the vector $\mathsf{OPK} = (\mathsf{opk}_1, \ldots, \mathsf{opk}_{2^\tau})$ and any valid opening op for index t, the verification algorithm $\Pi_{\mathsf{HVC}}.\mathsf{iVrfy}$ must return the original committed value at that position. Therefore, we have: $\mathsf{opk} = \mathsf{opk}_t$. Since $t \leq 2^\tau$ and $\mathsf{opk} = \mathsf{opk}_t \neq \perp$, the algorithm does not return 0 at step 3.
- KOTS Verification. The algorithm finally returns the result of $\Pi_{\mathsf{KOTS}}.\mathsf{iVrfy}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{opk}, \mu, \sigma')$. Substituting $\mathsf{opk} = \mathsf{opk}_t$ from the previous step, this is equivalent to $\Pi_{\mathsf{KOTS}}.\mathsf{iVrfy}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{opk}_t, \mu, \sigma')$. By the ϵ_{cor}-individual correctness of Π_{KOTS}, since σ' was generated using $\Pi_{\mathsf{KOTS}}.\mathsf{Sign}$ with the secret key osk_t corresponding to the public key opk_t, this verification algorithm returns 1 with probability $\geq 1 - \epsilon_{\mathsf{cor}}$.

Therefore, with probability $\geq 1 - \epsilon_{\mathsf{cor}}$, the Lemur individual verification algorithm iVrfy returns 1. □

Lemma 6.2. *Let $\lambda, d, q', n, k, \gamma, N, \tau, \alpha_w \in \mathbb{N}^+$ with d being a power of 2, q' being prime. Let $\epsilon_{\mathsf{hom}} \in_{\mathbb{R}} (0, \frac{1}{2})$ such that $\gamma \geq \lambda / \log_2\left(\frac{1}{2\epsilon_{\mathsf{hom}}}\right)$. Let $\mathcal{R}_{q'} = \mathbb{Z}_{q'}[X]/(X^d + 1)$. Let $W = \mathcal{T}_{\alpha_w} \subseteq \mathcal{R}$ be a set such that $|W| > 2^\lambda$ and let $W' := \{w_0 - w_1 \mid w_0, w_1 \in W\}$. Let $H : \{0,1\}^* \to W^N$ be a random oracle. Let Π_{KOTS} be a key-homomorphic one-time signature scheme with public keys in $\mathcal{R}_{q'}^{k \times n}$, and let Π_{HVC} be a homomorphic vector commitment for domain $\mathcal{R}_{q'}^{k \times n}$.*

If both Π_{KOTS} and Π_{HVC} are $(N, W, \epsilon_{\mathsf{hom}})$-probabilistically homomorphic, then the Lemur construction from Figure 8 has correct aggregation in the presence of rogue keys and signatures.

Proof. Let $\mathsf{pp} = (\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{pp}_{\mathsf{HVC}})$, and let $(\mathsf{pk}_i, \sigma_i)$ be pairs such that $\mathsf{iVrfy}(\mathsf{pp}, \mathsf{pk}_i, t, \mu, \sigma_i) = 1$. By the definition of iVrfy in Figure 8, this implies that for each $i \in [N]$, there exists a one-time public key opk_i such that:

$$\mathsf{opk}_i = \Pi_{\mathsf{HVC}}.\mathsf{iVrfy}(\mathsf{pp}_{\mathsf{HVC}}, c_i, t, \mathsf{op}_i), \tag{138}$$

$$\Pi_{\mathsf{KOTS}}.\mathsf{iVrfy}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{opk}_i, \mu, \sigma'_i) = 1, \tag{139}$$

where $\mathsf{pk}_i = c_i$ and $\sigma_i = (\sigma'_i, \mathsf{op}_i)$. The aggregation algorithm Aggregate performs at most γ attempts to find a salt j such that the aggregated signature $(\sigma', \mathsf{op}, j)$ is accepted by aVrfy. In each iteration j, the algorithm samples weights $(w_1, \ldots, w_N) \leftarrow H(t, \mu, P, j)$ and computes:

$$\sigma' = \sum_{i=1}^{N} w_i \cdot \sigma'_i, \quad \mathsf{op} = \sum_{i=1}^{N} w_i \cdot \mathsf{op}_i, \quad c = \sum_{i=1}^{N} w_i \cdot c_i. \tag{140}$$

According to aVrfy, the signature is accepted if:

- $\Pi_{\mathsf{HVC}}.\mathsf{sVrfy}(\mathsf{pp}_{\mathsf{HVC}}, c, t, \mathsf{op}) = \mathsf{opk}$ for some $\mathsf{opk} \neq \perp$. By the $(N, W, \epsilon_{\mathsf{hom}})$-probabilistic homomorphism of Π_{HVC}, since op_i are valid openings for c_i at position t with values opk_i, then $\mathsf{op} = \sum w_i \cdot \mathsf{op}_i$ is a valid opening for $c = \sum w_i \cdot c_i$ at position t with value $\mathsf{opk} = \sum w_i \cdot \mathsf{opk}_i$, except with probability ϵ_{hom}.
- $\Pi_{\mathsf{KOTS}}.\mathsf{sVrfy}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{opk}, \mu, \sigma') = 1$. Given that $\mathsf{opk} = \sum w_i \cdot \mathsf{opk}_i$ from the step above, the $(N, W, \epsilon_{\mathsf{hom}})$-probabilistic homomorphism of Π_{KOTS} ensures that the linear combination of signatures $\sigma' = \sum w_i \cdot \sigma'_i$ is a valid signature for message μ under the aggregated key opk, except with probability ϵ_{hom}.

By a union bound, the probability that a single iteration j fails to satisfy both conditions is at most $2\epsilon_{\mathsf{hom}}$. Since the algorithm makes γ independent attempts (using different salts j for the random oracle), the probability that all γ attempts fail is at most $(2\epsilon_{\mathsf{hom}})^{\gamma}$. Since $(2\epsilon_{\mathsf{hom}})^{\gamma} \leq 2^{-\lambda}$, the failure probability is negligible in the security parameter. Thus, the aggregation algorithm succeeds with probability $1 - \mathsf{negl}(\lambda)$. □

Lemma 6.3. *Let $\lambda, d, q', n, k, \gamma, N, \tau, \alpha_w \in \mathbb{N}^+$ with d being a power of 2, q' being prime. Let $\mathcal{R}_{q'} = \mathbb{Z}_{q'}[X]/(X^d + 1)$. Let $W = \mathcal{T}_{\alpha_w} \subseteq \mathcal{R}$ be a set such that $|W| > 2^{\lambda}$ and let $W' := \{w_0 - w_1 \mid w_0, w_1 \in W\}$. Let $H : \{0,1\}^* \to W^N$ be a random oracle. Let Π_{KOTS} be a key-homomorphic one-time signature scheme with public keys in $\mathcal{R}_{q'}^{k \times n}$, and let Π_{HVC} be a homomorphic vector commitment for domain $\mathcal{R}_{q'}^{k \times n}$.*

If Π_{KOTS} is W'-multi-user existentially unforgeable under rerandomized keys and Π_{HVC} is position binding, then the Lemur construction from Figure 8 is existentially unforgeable. Specifically, for any PPT adversary $\mathcal{A}$ against Lemur, there exist PPT adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\mathsf{Adv}^{\mathsf{euf}}_{\mathsf{Lemur},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{binding}}_{\Pi_{\mathsf{HVC}},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{EUF\text{-}RK}}_{\Pi_{\mathsf{KOTS}},\mathcal{B}_2}(\lambda).$$

Proof. The proof follows the same idea as in [16]. For completeness, we briefly sketch the proof. Suppose an adversary $\mathcal{A}$ outputs a valid forgery $(P^*, t^*, \mu^*, \sigma^*_{\mathsf{agg}})$ where $\sigma^*_{\mathsf{agg}} = (\sigma', \mathsf{op}, j)$. By the winning conditions of the EUF game, the target honest key $\mathsf{pk}^* = c^*$ must be in P^*, and the pair (t^*, μ^*) must not have been queried to the signing oracle. Let $(w_1, \ldots, w_N) \leftarrow H(t^*, \mu^*, P^*, j)$ be the weights derived from the random oracle. We define $\mathsf{opk} \leftarrow \Pi_{\mathsf{HVC}}.\mathsf{sVrfy}(\mathsf{pp}_{\mathsf{HVC}}, c, t^*, \mathsf{op})$, where $c = \sum w_i \cdot c_i$. For the forgery to be valid, aVrfy must return 1, implying $\mathsf{opk} \neq \perp$ and $\Pi_{\mathsf{KOTS}}.\mathsf{sVrfy}(\mathsf{pp}_{\mathsf{KOTS}}, \mathsf{opk}, \mu^*, \sigma') = 1$. We distinguish between two cases based on the behavior of the adversary:

Case 1: Breaking HVC Binding. In this case, the aggregated public key opk extracted from the forgery is different from the intended linear combination of one-time public keys. Let opk_{i,t^*} be the t^*-th one-time public key of the i-th signer. If $\mathsf{opk} \neq \sum_{i=1}^{N} w_i \cdot \mathsf{opk}_{i,t^*}$, then the opening op is a valid opening for commitment c at position t^* to a value other than the sum of the committed values. Since the HVC is homomorphic, $\sum w_i \cdot \mathsf{op}_i$ is a valid opening for $\sum w_i \cdot \mathsf{opk}_{i,t^*}$. Providing a different valid opening op for the same commitment c at the same position t^* directly violates the position binding property of Π_{HVC}. We can construct $\mathcal{B}_1$ to output these two different openings to win the binding game.

Case 2: Breaking KOTS Security. If Case 1 does not occur, then $\mathsf{opk} = \sum_{i=1}^{N} w_i \cdot \mathsf{opk}_{i,t^*}$. This means the adversary has produced a valid signature σ' for the message μ^* under the aggregated key opk, which is a random linear combination of the one-time keys. Since (t^*, μ^*) was never queried, the adversary has never seen a signature for μ^* at time t^* for the honest signer. The W'-multi-user existential unforgeability of Π_{KOTS} ensures that it is hard to forge a signature even under a linear combination of keys (rerandomized keys). Specifically, $\mathcal{B}_2$ can simulate the Lemur environment by embedding its own EUF-RK challenge keys into the HVC commitments. A successful forgery σ' by $\mathcal{A}$ then serves as a valid forgery against the KOTS challenge.

Since $\mathcal{A}$ must succeed in one of these two cases to produce a valid forgery, the advantage of $\mathcal{A}$ is bounded by the sum of the advantages of $\mathcal{B}_1$ and $\mathcal{B}_2$. □

7 Implementation and Benchmarks

In this section, we provide a concrete parameter setting for Lemur and evaluate its performance across various scales of signer sets N and time periods 2^{τ}. We compare our results with Chipmunk [16].

7.1 Parameters and Security Estimates

In this subsection, we describe how Lemur's parameters are chosen in practice. Table 4 summarizes all constraints required by Lemur construction. Table 5 reports the resulting public parameters, public and secret keys, and individual and aggregated signature sizes. We provide a parameter generation script[1] (The specific parameter settings are provided in Table 8). The script is built on top of the Chipmunk codebase, but the underlying parameter-selection logic is different. In particular, MSIS hardness is estimated using the state-of-the-art Dilithium security estimation framework[2], and MLWE hardness is evaluated using the lattice estimator[3] [2].

Optimization Goal. Our primary optimization goal is to minimize the size of the *aggregated signature*, subject to satisfying all constraints in Table 4. The aggregated signature consists of three components (see also Table 5 for a summary):

(1) **Path nodes.** There are $\tau\omega\kappa$ polynomials in $\mathcal{R}$ with ℓ^{∞}-norm bounded by β_{agg}, and an additional $\tau\omega\kappa$ polynomials with norm bounded by β_{encode}. This contributes at most $\tau\omega\kappa d\big(\log(2\beta_{\mathsf{agg}} + 1) + \log(2\beta_{\mathsf{encode}} + 1)\big)$ bits without any entropic encoding.
(2) **Aggregated one-time public keys.** These consist of $kn\kappa'$ polynomials in $\mathcal{R}$ with norm bounded by β_{agg}, contributing (at most) $kn\kappa' d \log(2\beta_{\mathsf{agg}} + 1)$ bits without any entropic encoding.
(3) **Aggregated KOTS signature.** This contains m polynomials in $\mathcal{R}$ with norm bounded by β_{σ}, which is (at most) $md \log(2\beta_{\sigma} + 1)$ bits without any entropic encoding.

[1] https://github.com/lemur-sig/lemur
[2] https://github.com/pq-crystals/security-estimates
[3] https://github.com/malb/lattice-estimator

#	Source	Constraint
1	Lemma 5.5	$\beta_{\text{agg}} \geq \eta\sqrt{2\alpha_w N \ln L}$ $L = \frac{2d}{\epsilon_{\text{hom}}} \cdot (2\tau\omega\kappa + k n\kappa' + 2\tau\omega)$
2	Lemma 5.7	$\text{MSIS}_{q,\omega,2\omega\kappa,4\beta_{\text{agg}}}$ is hard
3	Lemma 5.7	$\text{MSIS}_{q,\omega,kn\kappa',4\beta_{\text{agg}}}$ is hard
4	Lemma 5.2	$\beta_{\text{encode}} < \frac{\beta_{\text{agg}}}{2\eta} + \frac{1}{2}$
5	Lemma 4.1	$\alpha \geq \sqrt{2}\eta_{\varepsilon_1}(\mathbb{Z})$
6	Lemma 4.1	$\beta_z \geq 6\alpha\sqrt{1 + (k-1)\alpha_H}$
7	Lemma 4.1	$\epsilon_{\text{cor}} \geq md \cdot \left(2^{-162} + 2(k-1)\alpha_H\varepsilon_1\right)$
8	Lemma 4.2	$\beta_\sigma \geq \beta_z \cdot \sqrt{2\alpha_w N \cdot \ln\left(\frac{2md}{\epsilon_{\text{hom}}}\right)}$
9	Lemma 4.4	$d > t_1 > 1$ and $d > t_2 > 1$ are powers of 2
10	Lemma 4.4	$p \equiv 2t_1 + 1 \pmod{4t_1}$
11	Lemma 4.4	$q' \equiv 2t_2 + 1 \pmod{4t_2}$
12	Lemma 4.4	$k \geq 4$
13	Lemma 4.4	$\alpha \geq p\sqrt{2\alpha_H \ln(2(k-3)d(1+1/\varepsilon_3))/\pi}$
14	Lemma 4.4	$\left(\frac{1+\varepsilon_3}{1-\varepsilon_3}\right)^{2m} \cdot p^{-dm/t_1} < 2^{-\lambda}$
15	Lemma 4.4	$\left\lvert\mathcal{T}_{\alpha_H}\right\rvert^{k-3} \geq 2^{2\lambda}$
16	Lemma 4.4	$\text{DualHintMLWE}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$ is hard
17	Lemma 4.4	$\text{MSIS}^{\infty}_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}$ is hard
18	Lemma 6.2	$\gamma \geq \frac{\lambda}{\log_2\left(\frac{1}{2\epsilon_{\text{hom}}}\right)}$
19	Lemma 6.3	$\left\lvert\mathcal{T}_{\alpha_w}\right\rvert = \binom{d}{\alpha_w} \cdot 2^{\alpha_w} \geq 2^{\lambda}$

Table 4: Parameter constraints of Lemur.

Contribution	Size in Bits
Public Parameters	(can be generated from a seed instead of the expanded sizes below)
HVC: Ajtai's hash	$d(\omega k n\kappa' + 2\omega^2\kappa)\lceil\log q\rceil$
KOTS	$d(m-n)n\lceil\log q'\rceil$
Public Key	
HVC commitment	$d\omega\lceil\log q\rceil$
Secret Key	
2^τ KOTS keys	may regenerate on the fly
Individual Signature	
KOTS signature	$dm\lceil\log(2\beta_z+1)\rceil$
HVC opening	$(\tau\omega\kappa + k n\kappa')d\lceil\log(2\eta+1)\rceil$
Aggregate Signature	
agg. KOTS signature	$dm\lceil\log(2\beta_\sigma+1)\rceil$
agg. HVC opening	$(\tau\omega\kappa + k n\kappa')d\lceil\log(2\beta_{\text{agg}}+1)\rceil + \tau\omega\kappa d\lceil\log(2\beta_{\text{encode}}+1)\rceil$
index j of attempt	$\lceil\log\gamma\rceil$

Table 5: Sizes of Lemur components without any entropic encoding.

τ	N	Chipmunk ($d = 1024$)			Lemur ($d = 256$)			
		KOTS	HVC	Total	KOTS	HVC	Total	Encoded
12	2^{10}	26	133	159	7	151	157	126
	2^{15}	> 90	201	> 291	8	202	210	177
	2^{17}	> 100	263	> 362	9	229	238	203
	2^{20}	> 110	383	> 493	9	285	294	252
16	2^{10}	26	172	198	7	188	195	156
	2^{15}	> 90	261	> 351	8	254	262	221
	2^{17}	> 100	343	> 442	8	289	298	254
	2^{20}	> 110	498	> 608	9	360	369	316
20	2^{10}	26	211	237	7	226	232	185
	2^{15}	> 90	321	> 411	8	307	315	265
	2^{17}	> 100	423	> 522	9	349	358	304
	2^{20}	> 110	618	> 728	9	435	444	380
24	2^{10}	26	250	276	7	263	270	215
	2^{15}	> 90	381	> 471	8	359	367	308
	2^{17}	> 100	503	> 602	9	409	418	355
	2^{20}	> 110	734	> 844	9	510	519	443

Table 6: Comparison of aggregated KOTS, HVC opening, and total aggregate signature sizes (KB) for $\lambda = 128$ and root Hermite factor $\text{RHF} \leq 1.0045$ – see Section 7.1 for details. For Lemur, we first provide the theoretical worst-case size bounds, followed by the average on-wire length of our Rice-encoded aggregate (Encoded). The Chipmunk implementation [17] declares a Babai-style HVC-opening compression in the paper but its `EncodedPoly` serializer does not implement it.

Parameter selection strategy. We divide the parameters into three categories: parameters shared by KOTS and HVC, KOTS-specific parameters, and HVC-specific parameters. Our strategy is to fix as many global parameters as possible, and then exhaustively search over the remaining degrees of freedom. We target a security level of $\lambda = 128$ bits. We begin by fixing the common parameters: the ring dimension $d = 256$, the number of signers $N \in \{2^{10}, 2^{15}, 2^{17}, 2^{20}\}$, and the Merkle-tree height $\tau \in \{12, 16, 20, 24\}$. For ease of illustration, we describe the parameter-selection process for a representative setting $d = 256$, $N = 2^{20}$, and $\tau = 24$ from now on. Once d and λ are fixed, Constraint 19 (from Table 4) determines $\alpha_w = 23$. We set the probabilistic homomorphism error to $\epsilon_{\text{hom}} = 2^{-15}$, which determines the number of aggregation attempts to $\gamma = 10$ via Constraint 18.

KOTS parameters. We next determine the KOTS parameters. We set $k = 4$ satisfying Constraint 12. Given (k, λ), Constraint 15 fixes $\alpha_H = 60$. We set $t_1 = 2$ and $p = 5$, $\varepsilon_3 = 2^{-0.5}$, under which Constraint 10 and Constraint 14 are automatically satisfied for all valid choices of m. We set $t_2 = 8$ and $q' = 17 \pmod{32}$ by Constraint 11. Since $d = 256$, $t_1 = 2$, and $t_2 = 8$, Constraint 9 is satisfied. The choice of the Gaussian width α is more subtle, as it must simultaneously satisfy Constraints 5, 13, and 16. As discussed later after HVC parameters, we select $\alpha = 83$, which meets all three constraints with sufficient margin. Once α is fixed, Constraint 6 determines the individual-signature norm bound $\beta_z = 6700$. The remaining KOTS parameters (m, n, q') are chosen by enumerating feasible ranges for m and n, and selecting NTT-friendly primes q' such that both $\text{DualHintMLWE}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$ (Constraint 16) and $\text{MSIS}^{\infty}_{q',n,m,2\beta_\sigma+2\alpha_w\beta_z}$ (Constraint 17) meet the target security level. For each candidate, we compute β_σ using Constraint 8 and select the parameters minimizing the aggregated KOTS signature size. Constraint 7 is checked to ensure that the individual correctness error is negligible.

HVC parameters. Finally, we determine the HVC parameters. To reduce the search space, we enumerate candidate bit-lengths for q in the range $[16, 64]$ bits. For each candidate q, we enumerate small integer values of κ and ω and retain those satisfying Constraints 1, 2, 3, and 4. Among all valid candidates, we select the parameter set that

minimizes the HVC opening size. Overall, this systematic parameter selection procedure yields compact aggregated signatures while satisfying all correctness and security constraints.

How to choose α. We convert all the three constraints we require for α in Table 4 into explicit numerical bounds and explain how the final value is chosen.

(1) Constraint 5: Upper bound $\eta_{\varepsilon_1}(\mathbb{Z})$ by Lemma 2.9, we obtain $\alpha \geq \sqrt{\frac{2(1+1/\varepsilon_1)}{\pi}} = 5.77$ for $\varepsilon_1 = 2^{-150}$. Substituting $m = 10$ (see Table 8), $d = 256$, $k = 4$, $\alpha_H = 60$, $\varepsilon_1 = 2^{-150}$ into Constraint 7 gives $\epsilon_{\text{cor}} \approx 2^{-129}$, which is negligible.
(2) Constraint 16: By Theorem 3.1, $\mathsf{DualHintMLWE}_{q',m,n,k,\alpha,\mathcal{T}_{\alpha_H}}$ is at least as hard as $\mathsf{MLWE}_{q',m-n,n,k-1,\alpha_0}$ provided that $\alpha^2 \geq \alpha_0^2 \|\mathbf{B}\|_2^2 + (1+\alpha_H)\frac{\ln(2d(k-1)(1+1/\varepsilon_2))}{\pi}$, where $\mathbf{B} = \begin{pmatrix} -(\mathbf{h}')^\top \\ \mathbf{I}_{k-1} \end{pmatrix}$. To evaluate this bound concretely, we estimate $\|\mathbf{B}\|_2$ numerically by sampling $\mathbf{H}'$ according to $\mathcal{T}_{\alpha_H}$ and computing the spectral norm of $\mathbf{B}$. Across 10^3, 10^4, 10^5, 10^6 trials, the empirical distribution of $\|\mathbf{B}\|_2$ follows a Gaussian distribution. We therefore conservatively bound $\|\mathbf{B}\|_2$ by taking the empirical mean plus 12× standard deviations, capturing the tail behavior. This yields $\|\mathbf{B}\|_2 \approx 43.07$. Setting $\alpha_0 = \sqrt{2\pi/3}$ and $\varepsilon_2 = 2^{-136}$, chosen so that the term $8m\varepsilon_2$ in Lemma 3.3 is negligible, and substituting all parameters into the above inequality gives $\alpha \geq 76.54$.
(3) Constraint 13: Substituting the above concrete parameters, we obtain $\alpha \geq 82.45$.

Taking the maximum over all three bounds, we set $\alpha = 83$. Substituting this value back into the Dual Hint-MLWE sampleability condition yields a revised bound $\alpha_0 = 1.6$, which is used in the standard MLWE estimation for the parameter generation script.

In Table 6, we compare the aggregated KOTS, HVC opening, and total aggregate signature sizes of Lemur against Chipmunk. Overall, Lemur achieves substantial improvements at large scales. In particular, for $N = 2^{20}$, Lemur reduces the total aggregate signature size by nearly 2× across all values of τ. For example, when $(\tau, N) = (24, 2^{20})$, Lemur achieves a total size of 519 KB compared to Chipmunk's > 967 KB.

A key advantage of Lemur is the compactness of its aggregated KOTS component. Across all evaluated parameters, the KOTS size remains nearly constant as τ and N grow, ranging only from 7–9 KB. In contrast, Chipmunk's aggregated KOTS already exceeds 110 KB for $N = 2^{20}$. This yields more than an order-of-magnitude reduction in the KOTS component.

Lemur also achieves smaller HVC openings for larger signer sets. This improvement stems from replacing Ring-SIS with Module-SIS, enabling the use of a significantly smaller ring dimension ($d = 256$ versus $d = 1024$ in Chipmunk) while maintaining the same target security level.

Finally, the "Encoded" column reports the average on-wire size after Rice encoding. The encoded sizes are consistently smaller than the corresponding worst-case bounds, further improving the practical communication cost of Lemur.

Revised Security and Size Estimates for Chipmunk. First, we retain the original Chipmunk parameter generation script and use the `lattice-estimator` to evaluate its resistance against the latest lattice-based attacks. Since Chipmunk relies on three underlying Ring-SIS problems, we evaluate the *rop* (number of word operations) of each Ring-SIS instance separately under the corresponding parameter set, and then identify the minimum security level among the three. This yields an estimated security level of only 23–30 bits.

Second, to determine Chipmunk's actual theoretical signature size, we employ another state-of-the-art framework previously used to estimate Dilithium's security[4]. We again emphasize that we do not modify the internal logic of the original script; instead, we replace its insecure Ring-SIS hardness estimation component with the latest framework. Notably, we had to increase the ring dimension from the original 512 to 1024 in order to obtain a feasible parameter set. However, supporting larger values of N requires relaxing the target root Hermite factor. As shown in Table 6, feasible parameters could not be found for $d = 1024$ with $\mathsf{RHF} \leq 1.0045$. For these cases, we relax the KOTS requirements: $(N, \mathsf{RHF}_{\mathsf{KOTS}}) \in \{(2^{15}, 1.00464), (2^{17}, 1.0048), (2^{20}, 1.00501)\}$. In these cases, we obtain only lower bounds on the aggregate signature size. Lemur uses the fixed $d = 256, k = 4$ parameter set and maintains $\mathsf{RHF} \leq 1.0045$ throughout. Our revised scripts for Chipmunk are included in the artifact.

7.2 Implementation Notes

The main advantage of Lemur over non-aggregated signatures is the smaller signature size. However, its resource utilization is also remarkably modest. The implementations and more complete benchmarks are available at https://github.com/lemur-sig/lemur.

Two Interoperable Implementations. We provide two independent, fully interoperable open-source reference implementations: an optimized **Rust** crate and a readable **Python** prototype. Both cover the full pipeline (KOTS, HVC, aggregation, encoding) and are cross-validated against shared byte-level test vectors, so the specification admits a single concrete realization.

The Rust artifact combines a Montgomery-domain NTT for the HVC ring with a CRT-via-auxiliary-primes backend for the KOTS ring (whose modulus $q' \equiv 17 \pmod{32}$ is not natively NTT-friendly), a discrete-Gaussian CDT sampler whose precision and tailcut are pinned by a Rényi-divergence analysis at $\lambda = 128$, and three interchangeable signing paths—seed-only, BDS08 cache [9], and an optional in-memory materialized tree—that produce byte-identical signatures.

Signer State Optimization. The main deployment challenge is signer state: a naively materialized HVC tree at $\tau = 20$ occupies many GB of RAM, and τ is limited by available RAM (as in the Chipmunk reference implementation). Our default signer instead uses the BDS08 Merkle traversal algorithm [9], which maintains an amortized-$O(\tau)$ authentication-path cache of ≈ 110 KB against the same 32-byte master seed. This brings signing below 10 ms on commodity hardware after a one-time offline key generation, without carrying a multi-GB tree on disk or in RAM. With the entire tree in RAM, consecutive signatures can be generated in 1.6 ms. We consider the use of BDS08 as a significant practical contribution. Signature verification times are close to those of ML-DSA and Falcon at the same security level.

[4]We do not use the lattice estimator here because we did not find an easy way to extract the root Hermite factor.

A materialized tree rebuilt from the seed on demand remains available for callers that prefer sub-millisecond signing; the three signing paths (seed-only, BDS cache, materialized tree) all produce byte-identical signatures.

Serialization Optimizations. Each polynomial in the aggregated signature is bit-packed independently: the aggregated KOTS signature uses a fixed-width encoding tuned to the CLT-predicted standard deviation of $\mathbf{z}$, and the HVC opening uses Golomb-Rice coding for components whose folded-Gaussian model yields a shorter average code than the fixed-width alternative. On the implemented (τ=20, N=2^{10}) parameter setting, the encoder brings the aggregate size from the 232 KB bound (Table 6) down to 185.5 KB; the same encoder yields $\approx$ 15% savings on the larger-N cells (e.g. 379.6 KB vs. 444 KB at $N = 2^{20}$).

Multi-thread Optimization. The Rust crate parallelizes every data-parallel path with `rayon`. HVC key generation and the initial BDS08-cache build run as a divide-and-conquer recursion over the 2^τ leaves, each leaf being an independent KOTS key derivation followed by a hash up the subtree; the optional in-memory tree fills each level in parallel chunks. Aggregation parallelizes the per-signer weighted sums of the KOTS aggregate $\mathbf{z}$, the HVC commitment, and the opening. Aggregate verification likewise parallelizes the public-key aggregation $\sum_i w_i c_i$ over signers, and individual pre-verification `iVrfy` (the rogue-key check on each partial signature before aggregation) runs each signer's check on its own `rayon` task. The remaining serial work is the inner cryptographic core: one KOTS sign, one Babai re-encode, and the final NTT-based bound and equality checks on a single aggregated object.

Arithmetic Techniques. Public matrices and secret rows are independently derivable from counter-mode sub-seeds with explicit domain-separation labels. The KOTS keygen loops are sequential inside one key; the artifact exploits this independence at the leaf, tree, and aggregation levels.

The HVC modulus $q \approx 2^{42}$ satisfies $q \equiv 1 \pmod{2d}$ and uses a Montgomery-domain NTT, with public matrices pre-transformed once at setup and reused across key generation, signing, and verification. NTT butterflies and reductions use a branch-free signed-mask idiom (no external constant-time crate); the u32 backend can be auto-vectorized on AVX2 / NEON, while the shipped u64 path remains straight-line constant-time scalar on typical 64-bit targets.

The KOTS modulus $q' \equiv 17 \pmod{32}$ is not natively NTT-friendly, so KOTS multiplications route through a CRT backend over two pseudo-Mersenne primes

$$p_1 = 2^{48} - 16383 = 281{,}474{,}976{,}694{,}273 \quad \text{and}$$
$$p_2 = 2^{48} - 19967 = 281{,}474{,}976{,}690{,}689.$$

Both satisfy $p_i \equiv 1 \pmod{512}$, supporting native length-d NTT at $d \in \{128, 256\}$; their product $p_1 p_2 \approx 2^{96}$ leaves wide headroom for centered CRT reconstruction. Each $(96 \to 48)$-bit modular reduction is two 48-bit folds via the Mersenne-style identity $x \bmod p_i \equiv (\mathrm{hi} \cdot c_i + \mathrm{lo}) \bmod p_i$ for $x = \mathrm{hi} \cdot 2^{48} + \mathrm{lo}$ and $c_i \in \{16383, 19967\}$, followed by a conditional subtract.

For aggregation, the HVC opening accumulator $\sum_i w_i \cdot \mathrm{op}_i$ runs unconditionally in the NTT domain of the shipped u64 HVC backend, paying a single inverse transform per output polynomial rather than one per signer. The KOTS aggregate $\sum_i w_i \cdot \mathbf{z}_i$ takes the analogous path in the CRT-NTT domain over the two auxiliary primes, then performs the inverse transforms and CRT-combines into the exact centred signed-integer coefficient sum (skipping the final reduction modulo q'). The CRT exactness bound $N \cdot d \cdot ((q'-1)/2)^2 < P/2$ (where $P = p_1 p_2 \approx 2^{96}$) holds with vast margin at every realistic N: for `D256_K4` it permits N up to $\sim 5 \cdot 10^8$, so the fast in-CRT-NTT path covers the paper's full $N \in \{2^{10}, 2^{13}, 2^{15}, 2^{17}, 2^{20}\}$ range without falling back to per-signer scaling.

The discrete-Gaussian CDT sampler consumes one 32-bit word per coefficient: the least significant bit is the sign, and the remaining 31 bits supply the CDT comparison. The analytical minimum precision `prec_re = 28` and tailcut `tc_re = 5` are computed by `parameter/Lemur-DGS-Prec_TailCut.py` from a Rényi-divergence analysis at $\lambda = 128$ over $Q_{\mathrm{bs}} = kmd = 9{,}216$ base-sampler queries per KOTS keygen (the 2^τ slot dimension is absorbed by the EUF-RK reduction rather than multiplied into Q_{bs}). The byte-aligned implementation ships `cdt_bits = 32` and `tailcut = 5`, retaining a 3-bit comparison margin over the analytical minimum. XOF reads are batched per polynomial to amortize per-call overhead.

Benchmarking notes. All benchmarks were obtained with `rustc` version 1.96.0 running on a laptop system with AMD Ryzen AI 9 HX 370 CPU (12 cores with 2-way SMT; 24 threads), 32GB RAM, and Debian forky / Linux 7.0.9 operating system. The CPU has dynamic clock frequency topped at 5.1 GHz.

Table 7 offers timings for each sub-step of `lemur_aggregate` and `lemur_avrfy` in isolation on the implemented (τ=20, d=256, k=4) profile, measured by the artifact's `bench_aggregate` binary on the same hardware as Table 2.

At $N = 2^{10}$ the HVC opening aggregator dominates aggregation at 80% of the wall-clock total; at $N = 2^{13}$ the per-signer pre-verify and HVC opening aggregation are roughly equal, both linear in N. As a sidenote, the Chipmunk implementation [17] does not actually implement the pre-verify step, even though the pseudocode in [16] includes it, and it is required for the security claims of both schemes.

Batch verification is dominated by the public-key serialization and SHAKE128-based randomizer derivation, both also linear in N. We report $N = 2^{13}$ as the largest cell that runs cleanly against the same implementation constants as the declared $D256\_K4$ cell at $N = 2^{10}$ (the paper's $N \in \{2^{15}, 2^{17}, 2^{20}\}$ rows use distinct parameter cells with larger $k, m, \omega, \eta, \beta_{\mathrm{agg}}$). In Table 2, the large-N aggregation entries are linearly extrapolated from the `bench -fast` $N = 2^{13}$ run, while the large-N batch-verification entries are measured with `bench_verify -zero-fixture`. The end-to-end totals here (2.80 s aggregation, 114.0 ms batch verify) are independent `bench_aggregate` measurements, consistent with the `bench -fast` anchor (2.79 s aggregation, 111.2 ms batch verify) to within $\approx$ 1% and $\approx$ 3%, respectively.

Acknowledgments

We thank Thomas Espitau for sharing ideas about an initial version of Lemma 2.11. This work is partly supported by an Ethereum Foundation grant, and Australian Research Council Discovery Grants DP250100229 and DP260100245.

Step	$N = 2^{10}$	$N = 2^{13}$
Aggregation `lemur_aggregate`:		
Individual pre-verify ($\times N$, `rayon`)	177.6 ms	1.31 s
PK serialization (`concat_pk_bytes`)	6.2 ms	59.9 ms
Randomizer derivation (SHAKE128)	5.4 ms	43.6 ms
KOTS aggregate $\sum_i w_i z_i$	9.8 ms	64.1 ms
HVC opening aggregate $\sum_i w_i \cdot \mathrm{op}_i$	937.7 ms	1.32 s
`aVrfy` probe (single attempt)	15.5 ms	113.7 ms
End-to-end `lemur_aggregate`	**1.16 s**	**2.80 s**
Batch verification `lemur_avrfy`:		
PK serialization	6.2 ms	59.9 ms
Randomizer derivation (SHAKE128)	5.3 ms	46.8 ms
Public key aggregate $\sum_i w_i c_i$	0.89 ms	5.63 ms
HVC `sVrfy` (Babai decode + verify)	2.72 ms	1.30 ms
KOTS `sVrfy`	0.56 ms	0.27 ms
End-to-end `lemur_aVrfy`	**15.5 ms**	**114.0 ms**

Table 7: Sub-step breakdown of `lemur_aggregate` and `lemur_aVrfy` on the implemented (τ=20, d=256, k=4) profile, measured by the `bench_aggregate` binary. See Section 7.2 for system details.

References

[1] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. 2024. Aggregating Falcon Signatures with LaBRADOR. In *CRYPTO 2024 (LNCS)*. Springer, 71–106.

[2] Martin R. Albrecht, Rachel Player, and Sam Scott. 2015. On the concrete hardness of Learning with Errors. *J. Math. Cryptol.* 9, 3 (2015), 169–203.

[3] Hiroaki Anada, Masayuki Fukumitsu, and Shingo Hasegawa. 2024. Tightly Secure Lattice-Based Synchronized Aggregate Signature in Standard Model. In *ICISC 2024 (LNCS)*. Springer, 56–70.

[4] Rachid El Bansarkhani and Jan Sturm. 2016. An Efficient Lattice-Based Multisignature Scheme with Applications to Bitcoins. In *CANS 2016 (LNCS, Vol. 10052)*. 140–155.

[5] Ward Beullens and Gregor Seiler. 2023. LaBRADOR: Compact Proofs for R1CS from Module-SIS. In *CRYPTO 2023 (LNCS)*. Springer, 518–548.

[6] Dan Boneh and Sam Kim. 2020. One-Time and Interactive Aggregate Signatures from Lattices. https://api.semanticscholar.org/CorpusID:221364120

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. 2001. Short Signatures from the Weil Pairing. In *ASIACRYPT 2001 (LNCS)*. Springer, 514–532.

[8] Cecilia Boschini, Akira Takahashi, and Mehdi Tibouchi. 2022. MuSig-L: Lattice-Based Multi-signature with Single-Round Online Phase. In *CRYPTO 2022 (LNCS, Vol. 13508)*. Springer, 276–305.

[9] Johannes Buchmann, Erik Dahmen, and Michael Schneider. 2008. Merkle Tree Traversal Revisited. In *Post-Quantum Cryptography (PQCrypto) 2008 (LNCS, Vol. 5299)*. Springer, 63–78.

[10] Maxime Buser, Rafael Dowsley, Muhammed F. Esgin, Shabnam Kasra Kermanshahi, Veronika Kuchta, Joseph K. Liu, Raphaël C.-W. Phan, and Zhenfei Zhang. 2022. Post-Quantum Verifiable Random Function from Symmetric Primitives in PoS Blockchain. In *ESORICS (1) (LNCS)*. Springer, 25–45.

[11] Yanbo Chen. 2023. DualMS: Efficient Lattice-Based Two-Round Multi-signature with Trapdoor-Free Simulation. In *Crypto 2023*. Springer, 716–747.

[12] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. 2021. Two-Round n-out-of-n and Multi-signatures and Trapdoor Commitment from Lattices. In *PKC 2021 (LNCS, Vol. 12710)*. Springer, 99–130.

[13] Julien Devevey, Alain Passelègue, and Damien Stehlé. 2023. G+G: A Fiat-Shamir Lattice Signature Based on Convolved Gaussians. In *ASIACRYPT 2023 (LNCS)*. Springer, 37–64.

[14] Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. 2025. Hash-Based Multi-Signatures for Post-Quantum Ethereum. *CIC 2025* 2, 1 (2025), 13.

[15] Justin Drake, Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. 2025. Technical Note: LeanSig for Post-Quantum Ethereum. *IACR Cryptol. ePrint Arch.* 2025 (2025), 1332.

[16] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. 2023. Chipmunk: Better Synchronized Multi-Signatures from Lattices. In *CCS 2023*. ACM, 386–400.

[17] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. 2023. Reference Implementation – Chipmunk: Better Synchronized Multi-Signatures from Lattices. Public GitHub repository referenced in the Chipmunk CCS 2023 paper. https://github.com/GottfriedHerold/Chipmunk

[18] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. 2022. Squirrel: Efficient Synchronized Multi-Signatures from Lattices. In *CCS 2022*. ACM, 1109–1123.

[19] Masayuki Fukumitsu and Shingo Hasegawa. 2019. A Tightly-Secure Lattice-Based Multisignature. In *6th on ASIA Public-Key Cryptography Workshop*. ACM, 3–11.

[20] Masayuki Fukumitsu and Shingo Hasegawa. 2020. A Lattice-Based Provably Secure Multisignature Scheme in Quantum Random Oracle Model. In *ProvSec 2020 (LNCS, Vol. 12505)*. Springer, 45–64.

[21] Kazuharu Itakura and Katsuhiro Nakamura. 1983. A public-key cryptosystem suitable for digital multisignatures. *NEC Research & Development* 71 (1983), 1–8.

[22] Meenakshi Kansal and Ratna Dutta. 2020. Round Optimal Secure Multisignature Schemes from Lattice with Public Key Aggregation and Signature Compression. In *AFRICACRYPT 2020 (LNCS, Vol. 12174)*. Springer, 281–300.

[23] Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. 2025. At the Top of the Hypercube - Better Size-Time Tradeoffs for Hash-Based Signatures. In *CRYPTO 2025 (LNCS)*. Springer, 93–123.

[24] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. 2023. Toward Practical Lattice-Based Proof of Knowledge from Hint-MLWE. In *CRYPTO 2023 (LNCS, Vol. 14085)*. Springer, 549–580.

[25] Vadim Lyubashevsky. 2012. Lattice Signatures without Trapdoors. In *EUROCRYPT 2012 (LNCS, Vol. 7237)*. Springer, 738–755.

[26] Vadim Lyubashevsky and Daniele Micciancio. 2008. Asymptotically Efficient Lattice-Based Digital Signatures. In *TCC 2008 (LNCS)*. Springer, 37–54.

[27] Vadim Lyubashevsky and Gregor Seiler. 2018. Short, Invertible Elements in Partially Splitting Cyclotomic Rings and Applications to Lattice-Based Zero-Knowledge Proofs. In *EUROCRYPT 2018 (LNCS, Vol. 10820)*. Springer, 204–224.

[28] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. 2024. The LaZer Library: Lattice-Based Zero Knowledge and Succinct Proofs for Quantum-Safe Privacy. In *CCS 2024*. ACM, 3125–3137.

[29] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. 2001. Accountable-subgroup multisignatures: extended abstract. In *CCS 2001*. ACM, 245–254.

[30] Daniele Micciancio and Oded Regev. 2004. Worst-Case to Average-Case Reductions Based on Gaussian Measures. In *FOCS 2004*. IEEE Computer Society, 372–381.

[31] Chris Peikert. 2010. An Efficient and Parallel Gaussian Sampler for Lattices. In *CRYPTO 2010 (LNCS, Vol. 6223)*. Springer, 80–97.

[32] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. 2020. *Falcon*. Technical Report. Technical report, National Institute of Standards and Technology.

[33] Arda Saygan, Muhammed Said Gündogan, Atakan Arslan, and Mehmet Emin Gönen. 2025. 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)*. Springer, 3–22.

A Parameter Setting

Parameter Sets		KOTS Parameters				HVC Parameters							Size (KB)		
τ	N	n	m	β_σ	q'_{bit}	ω	η	β_{agg}	β_{encode}	q_{bit}	κ	κ'	KOTS	HVC	Total
12	1024	4	9	6310455	30	2	169	173980	515	43	5	4	7	151	157
	32768	4	10	35797038	36	4	256	1498795	2928	28	3	4	8	202	210
	131072	5	10	71594076	31	4	1625	19023917	5854	36	3	3	9	229	238
	1048576	5	10	202498625	33	5	203	6756063	16641	27	3	4	9	285	294
16	1024	4	9	6310455	30	2	169	174910	518	43	5	4	7	188	195
	32768	4	10	35797038	36	4	256	1507055	2944	28	3	4	8	254	262
	131072	5	10	71594076	31	4	1625	19129623	5887	36	3	3	9	289	298
	1048576	5	10	202498625	33	5	203	6792932	16732	27	3	4	9	360	369
20	1024	4	9	6310455	30	2	169	175655	520	43	5	4	7	226	232
	32768	4	10	35797038	36	4	256	1513617	2957	28	3	4	8	307	315
	131072	5	10	71594076	31	4	1625	19213475	5912	36	3	3	9	349	358
	1048576	5	10	202498625	33	5	203	6822228	16804	27	3	4	9	435	444
24	1024	4	9	6310455	30	2	169	176277	522	43	5	4	7	263	270
	32768	4	10	35797038	36	4	256	1519059	2967	28	3	4	8	359	367
	131072	5	10	71594076	31	4	1625	19282927	5934	36	3	3	9	409	418
	1048576	5	10	202498625	33	5	203	6846521	16864	27	3	4	9	510	519

Table 8: Parameter sets for Lemur with constant values: $\lambda = 128$, $d = 256$, $\epsilon_{\text{hom}} = 2^{-15}$, $\alpha_w = 23$, $\gamma = 10$, $k = 4$, $\alpha = 83$, $\alpha_{\text{mlwe}} = 1.6$, $\alpha_H = 60$, **and** $\beta_z = 6700$.