

Lattice-Based Synchronous Signatures: Efficiently Aggregatable and Thresholdizable

Dan Boneh¹, Binyi Chen^{*2}, and Trisha Datta¹

¹Stanford University

²Tsinghua University

September 20, 2026

Abstract

Aggregate signatures play an important role in proof-of-stake systems, where many validators sign a block. There is a strong desire to aggregate all these signatures into one short signature that is fast to verify. Since all validators know the block number they are signing, this setting is well suited for synchronous (a.k.a stateful) signatures. Boneh and Kim (2019) showed that lattice-based one-time signatures (OTS) can be aggregated very efficiently. The Chipmunk and Lemur signature schemes extend this to an ℓ -time synchronous scheme using a (homomorphic) Merkle tree of ℓ one-time public keys. Each leaf of the tree is used to sign one message, and these one-time signatures can be aggregated across many signers. Due to the Merkle tree, every aggregate signature includes a Merkle authentication path of length $O_\lambda(\log \ell)$.

We present a different lattice-based approach to constructing a synchronous aggregate signature scheme. The length of an aggregate signature in our scheme is *independent* of ℓ . The resulting signatures are asymptotically shorter than existing schemes, and concretely shorter for some parameter choices. The resulting signature verification algorithm is algebraic, which makes it amendable to efficient threshold signing. Moreover, we are able to prove security in an adaptive corruption model. We thus obtain an efficient lattice-based synchronous threshold signature scheme, where signatures from many signers can be aggregated into a single short signature. The scheme relies on a one time trusted setup to generate the public parameters.

^{*}Work done while at Stanford University.

Contents

1	Introduction	3
1.1	Technical Overview	5
2	Preliminaries	7
2.1	Lattice Basics	8
2.2	Forking Lemma	12
2.3	The Value Guessing Game	13
3	Statistically Sampleable Homomorphic Vector Commitments	15
3.1	Definition	15
4	Constructing a Succinct Hom. Vector Commitment	19
5	Key Homomorphic Synchronous Signature Schemes	28
5.1	Definition	28
5.2	A Generic Construction from Statistically Sampleable HVC	30
6	A Generic Aggregate Synchronous Signature Scheme	38
6.1	Aggregating Synchronous Signatures via Linear Combinations	40
6.2	Aggregating Synchronous Signatures via Proof of Possession	44
7	A Generic Threshold Synchronous Signature Scheme	48
7.1	Secret Sharing Schemes	49
7.2	Statically Secure Threshold Synchronous Signature from Basic Synchronous Signature Scheme	52
7.3	Adaptively Secure Threshold Synchronous Signature from Basic Synchronous Signature Scheme	56
8	Instantiation and Performance	70
9	Conclusions and Open Problems	72
A	Bootstrapping Vector Commitments	85
A.1	Proof of Lemma 3.1	85
B	The Relationship between Different Primitives	90
C	Proof of Lemma 7.5	91

1 Introduction

In a proof-of-stake blockchain, every block is signed by many validators, and these signatures are stored on chain so that anyone can verify the entire chain. Signature aggregation is an important tool used to compress these signatures.

A signature scheme supports *aggregation* if given triples of public keys, messages, and signatures $(\mathbf{pk}_i, m_i, \sigma_i)_{i=1}^n$, an aggregator can produce a single aggregate signature σ that convinces the verifier that every aggregated triple $(\mathbf{pk}_i, m_i, \sigma_i)$ is valid [BGLS03]. When all the messages are the same, namely $m_1 = \dots = m_n$, aggregation is called multisigning [MOR01]. There are several efficient aggregate signatures in the literature [BGLS03; Bol03; Lu+06; BNN07; RY07; Sch11; HKW15; BDN18; HW18; Wei+24; PW24; BW25], but these constructions are not post-quantum secure.

Ethereum’s current post-quantum plan [DKKW25a; DKKW25b] relies on hash-based signatures. Signature aggregation is SNARK-based [KCLM22; Zha+24; Aar+24], meaning that the aggregator will produce a succinct proof that it knows n signatures on the currently proposed block, and validators will verify this succinct proof. However, this adds considerable complexity to the consensus protocol. It would be much preferable if there were a post-quantum signature scheme with a *simple algebraic* aggregation algorithm where anyone can non-interactively aggregate a provided tuple of signatures into a short aggregate signature. Unfortunately, existing lattice-based aggregation schemes do not support this [EB14; WW19; SS22a; TS23; SS22b; BR21; DHSS20; BT23; JRS24].

In the case of proof-of-stake consensus it suffices to construct a simpler aggregation primitive. The key point is that an honest validator will never sign two different blocks for the same slot h on chain. This suggests that block validation can be done using a one-time signature scheme, where the validator uses a different public key for every slot. In particular, each validator pre-commits to a vector of ℓ one-time public keys, for some ℓ , say $\ell = 10^6$. A validator’s signature on block number $h \in [\ell]$ is a pair (σ, π) where σ is its one-time signature with respect to the public key $\mathbf{pk}_h$, and π is a proof that position h in the committed vector is $\mathbf{pk}_h$. To aggregate we need that a set of pairs (σ, π) from distinct validators, all with respect to the same slot $h \in [\ell]$, can be efficiently aggregated.

More generally, for proof-of-stake consensus it suffices to construct a *synchronous* signature scheme [GR06] that is efficiently aggregatable [AGH10; HW18; DGNW20; KS23]. In a synchronous scheme the signing algorithm $\text{Sign}(\mathbf{sk}, m, h)$ takes an additional input $h \in [\ell]$. The scheme is unforgeable as long as the adversary does not obtain two signatures at the same slot h (see Section 5.1). As explained above, a synchronous scheme with ℓ slots can be built from a one-time signature and a vector commitment for vectors of length ℓ . We need that a set of triples $(\mathbf{pk}_i, m_i, \sigma_i)_{i=1}^n$ from distinct signers, all with respect to the same slot h , can be efficiently aggregated.

The goal is to construct such a lattice-based synchronous signature scheme. The starting point is the one-time aggregate signature scheme of Boneh and Kim [BK20], which is based on the one-time signature of Lyubashevsky and Micciancio [LM18]. Squirrel and

Chipmunk [FSZ22; FHSZ23; AFH24] construct a synchronous aggregatable scheme by employing this aggregatable one-time signature along with a homomorphic Merkle commitment to commit to ℓ one-time public keys. Lemur [Lin+26] provides further optimizations based on the Module-LWE and Module-SIS assumptions. Due to the use of Merkle proofs in these schemes, the size of a standalone signature and the size of an aggregate signature scales with $\log_2 \ell$, where ℓ is the total number of slots.

Our results. We construct new synchronous signature schemes that support aggregation and threshold signing, with asymptotically shorter signatures than Squirrel, Chipmunk, and Lemur [FSZ22; FHSZ23; AFH24; Lin+26]. In Section 5 we present a general framework for constructing an ℓ -slot w -way synchronous signature scheme directly from a homomorphic vector commitment, without additionally relying on a one-time signature scheme (w -way means that security is maintained as long as one signs at most w messages per slot). For security, the homomorphic vector commitment must satisfy a new hiding property we call *statistical sampleability*. This property ensures that the commitment, along with several homomorphic opening proofs, reveals nothing about unopened parts of the committed vector. This is necessary to ensure that yet unused secret keys are not compromised by published signatures. Our notion of homomorphic vector commitment extends the homomorphic commitment notion introduced in Squirrel [FSZ22].

In Section 4 we build on the BASIS vector commitment of Wee and Wu [WW23] to construct a homomorphic vector commitment with the required statistical sampleability property. The length of opening commitments in this scheme is independent of the vector length ℓ . We prove security using the standard Module-SIS assumption, using a technique due to Ducas and Micciancio [DM14].

Next, in Section 5 we instantiate our framework using this homomorphic vector commitment. We obtain a synchronous signature scheme where the verification algorithm is homomorphic with respect to the public key. This lets us efficiently aggregate signatures via a simple linear combination algorithm (Section 6) or via proof of knowledge of the secret key (Section 6.2). The length of standalone signatures and aggregate signatures are independent of the total number of slots ℓ . In Section 7 we show that the signing algorithm is homomorphic with respect to the secret key, and this lets us obtain an efficient threshold signature scheme. We prove security of this threshold scheme under an adaptive corruption model. These improvements come at a cost of using a one-time trusted setup to generate the system parameters (see Section 9). One can eliminate the trusted setup using a technique due to Waters, Wee, and Wu [WWW25], but at the cost of increasing the signature size by a factor of $\log_2 \ell$, which we want to avoid.

Table 1 compares the asymptotic behavior of our scheme to prior post-quantum constructions. The main improvement is in the signature size, which is independent of ℓ . In Section 8 we evaluate the concrete aggregate signature size. Our construction outperforms existing schemes when aggregating across a large number of signatures. Nevertheless, the asymptotic improvement suggests that further optimizations could make our scheme

Scheme	$ \text{crs} $	$ \text{pk} $	$ \text{sk} $	$ \text{sig} $	$ \text{aggsig} $
[FHSZ23]	$O(\lambda nd)$	$O(\lambda nd)$	$O(\lambda \ell nd)$	$O(\lambda nd \log \ell)$	$O((\lambda + \log N) nd \log \ell)$
[Lin+26]	$O(\lambda nd)$	$O(\lambda nd)$	$O(\lambda \ell nd)$	$O(\lambda nd \log \ell)$	$O((\lambda + \log N) nd \log \ell)$
Ours	$O(\lambda(\ell nd)^2)$	$O(\lambda nd)$	$O(\lambda \ell nd)$	$O(\lambda nd)$	$O((\lambda + \log N) nd)$

Table 1: Key and signature sizes (in bit) of our scheme vs. Chipmunk [FHSZ23] and Lemur [Lin+26]. ℓ is the number of supported slots, N is the maximum number of signatures being aggregated, and the commitment space used is R_q^n where $R_q = \mathbb{Z}_q[X]/(X^d + 1)$.

concretely competitive even for smaller batches. For example, recent work [Unp26] shows that the commitment, opening proofs, and crs in BASIS commitments can be compressed using NTRU lattices [Hof+03], as in Falcon signatures [Pre+22]. This optimization would shrink the concrete size of our signatures and aggregate signatures, but does not seem to improve the Merkle-based constructions in prior work. We leave this as a promising direction for future work.

Related work. Lattice based aggregate signatures have been explored in [EB14; WW19; SS22a; TS23; SS22b; BR21; DHSS20; BT23; JRS24]. In some schemes the aggregate signature is not succinct, while others only provide half aggregation meaning that the aggregate length is linear in the number of signatures being aggregated. Some works also explore sequential aggregation, where signers sequentially aggregate their signature to a previously aggregated set [LMRS04; Lu+06; Nev08; BGR12; FLS12; LLY13b; LLY13a; GOR18]. This is sufficient for aggregating signatures in a certificate chain, but not for aggregating signatures in a consensus protocol where parties sign concurrently and aggregation is done non-interactively by an untrusted third party. Further work on aggregate signatures is available in [BCJP24; EMP16]. Pre-quantum synchronous signature aggregation was explored in [AGH10; HW18; DGNW20; TT20; TT22; KS23].

1.1 Technical Overview

In the main body we construct our aggregatable signature scheme in a modular way. We first define the concept of a statistically sampleable homomorphic vector commitment and construct such a scheme. Then we derive a general synchronous signature scheme, and finally we build aggregation and thresholding schemes. In this section we put these components together and sketch the complete signature scheme. We present it as a one-time ℓ -slot aggregatable synchronous signature. In the main body we construct a w -way ℓ -slot scheme, meaning that security is maintained as long as no more than w messages are signed per slot. Here $w = 1$.

First, let us define the parameters for the scheme. Let $R_q := \mathbb{Z}_q[X]/(X^d + 1)$ and

let $\mathbf{A} \in R_q^{n \times m}$ and $\mathbf{A}_{[0]}, \mathbf{A}_{[1]} \in R_q^{n \times w}$ be random matrices so that a trapdoor for $\mathbf{A}$ is known during the trusted setup. For $h \in [\ell]$ set $\mathbf{A}_h := [\mathbf{A} \mid \mathbf{A}_{[0]} + \hat{h} \cdot \mathbf{A}_{[1]}]$, where $\hat{h}$ is an encoding of h as an element of R_q . Let $\mathbf{G} \in R_q^{n \times w}$ be a gadget matrix. Finally, let $\mathbf{T}^*$ be a trapdoor for a matrix $\mathbf{D}_\ell$, derived from $\mathbf{A}_1, \dots, \mathbf{A}_\ell$ and $\mathbf{G}$, as defined in the body (Figure 2 eq. (8)). This $\mathbf{T}^*$ is generated using the trapdoor for $\mathbf{A}$. The verifier's parameters are $\mathbf{vp} := (\mathbf{A}, \mathbf{A}_{[0]}, \mathbf{A}_{[1]})$ and the signer's parameters are $\mathbf{cp} := (\mathbf{vp}, \mathbf{T}^*)$. The trapdoor for $\mathbf{A}$ is then erased and never used again.

To generate a key pair, the signer samples two vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in R_q^\ell$ and uses $\mathbf{T}^*$ to compute a commitment $\mathbf{cm}^{(1)}, \mathbf{cm}^{(2)} \in R_q^n$ to them using the BASIS commitment scheme [WW23], as explained in Section 4. The signer obtains low-norm commitment opening proofs $\{\pi_h^{(b)} \in R_q^{m+w}\}_{b \in \{1,2\}, h \in [\ell]}$ for every entry in the vectors $\mathbf{x}^{(1)}, \mathbf{x}^{(2)} \in R_q^\ell$. By construction, each proof $\pi_h^{(b)}$ satisfies

$$\mathbf{A}_h \cdot \pi_h^{(b)} = \mathbf{cm}^{(b)} + \hat{\mathbf{x}}_h^{(b)}, \quad \text{for } b \in \{1, 2\} \text{ and } h \in [\ell],$$

where $\hat{\mathbf{x}}_h^{(b)} \in R_q^n$ is an encoding of $\mathbf{x}_h^{(b)} \in R_q$. The public key is $\mathbf{pk} = (\mathbf{cm}^{(1)}, \mathbf{cm}^{(2)})$ and the signing key is $\mathbf{sk} = (\mathbf{pk}, \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \{\pi_h^{(b)}\})$.

Now, to sign a message m at slot $h \in [\ell]$ the signer employs a collision resistant hash $H(m, h) \rightarrow \alpha_{m,h} \in R_q$, where $\alpha_{m,h}$ is low-norm. Let $\pi_h^{(1)}, \pi_h^{(2)}$ be the opening proofs for $\mathbf{x}_h^{(1)}, \mathbf{x}_h^{(2)} \in R_q$. The signature is simply the pair

$$(\sigma, \tau) := (\mathbf{x}_h^{(1)}, \pi_h^{(1)}) + \alpha_{m,h} \cdot (\mathbf{x}_h^{(2)}, \pi_h^{(2)}) \in R_q \times R_q^{m+w}. \quad (1)$$

Note that the signature size is independent of ℓ . Also $\mathbf{T}^*$ is not used during signing; it is only used in key generation.

To verify the signature (σ, τ) using $(\mathbf{pk}, m, h)$, re-derive $\alpha_{m,h}$ and accept if τ is a valid commitment opening proof for σ . That is, accept if τ is low norm and

$$\mathbf{A}_h \cdot \tau = (\mathbf{cm}^{(1)} + \alpha_{m,h} \cdot \mathbf{cm}^{(2)}) + \hat{\sigma}. \quad (2)$$

Again, $\hat{\sigma} \in R_q^n$ is an encoding of $\sigma \in R_q$. Unlike previous schemes, signature verification requires no Merkle proofs. We prove security of the scheme in the standard model (no random oracles) using the Module-SIS assumption and the collision resistance of H . Here we use the statistical sampleability of the vector commitment scheme. This requires a subtle argument using a tool we call the value guessing game (Section 2.3).

Homomorphic properties. This signature scheme has two important homomorphic properties. First, the verification equation (2) shows that verification is linear with respect to the public key $\mathbf{pk} = (\mathbf{cm}^{(1)}, \mathbf{cm}^{(2)})$ and the signature (σ, τ) . This enables efficient signature aggregation. For $u \in \{1, 2\}$ suppose that the pair (σ_u, τ_u) is a signature on m_u with

respect to $\mathbf{pk}_u$, both using the same slot h . Then $(\sigma_1, \tau_1) + (\sigma_2, \tau_2)$ can be verified as an aggregate signature for $(\mathbf{pk}_1, \mathbf{pk}_2)$ as follows:

$$\mathbf{A}_h \cdot (\tau_1 + \tau_2) = (\mathbf{cm}_1^{(1)} + \mathbf{cm}_2^{(1)}) + (\alpha_{m_1, h} \cdot \mathbf{cm}_1^{(2)} + \alpha_{m_2, h} \cdot \mathbf{cm}_2^{(2)}) + (\hat{\sigma}_1 + \hat{\sigma}_2).$$

Of course, this simple aggregation procedure is not secure against rogue public key attacks [BGLS03]. But it can be made secure using either (i) the technique of derived combinations [BDN18; BK20] (Section 6) or (ii) proof of knowledge of the secret key [Bol03] (Section 6.2).

Second, the signing algorithm in (1) shows that signing is linear with respect to the secret key. This lets us secret share the secret signing key using any linear secret sharing scheme. The signature shares can be combined into a complete signature using a suitable linear combination. We present the threshold signature scheme in Section 7. We prove security in the random oracle model under adaptive corruptions using a masking technique from [Del+24; JWZ26].

2 Preliminaries

Notation. $\lambda \in \mathbb{N}$ is the security parameter. For $l, r \in \mathbb{Z}$, $l < r$, we denote by (l, r) the set $\{l+1, l+2, \dots, r-1\}$, $[l, r)$ the set $\{l, l+1, \dots, r-1\}$, and $[l, r]$ the set $\{l, l+1, \dots, r\}$. We define $[n] := [1, n]$. For a finite set S we use $x \leftarrow \$ S$ to denote a uniform random variable x over the set S . Similarly, for a randomized algorithm $\mathcal{A}$ we use $x \leftarrow \$ \mathcal{A}(y)$ to denote the random variable x that is the output of running $\mathcal{A}(y)$. For a deterministic algorithm $\mathcal{B}$ we use $x \leftarrow \mathcal{B}(y)$ to assign the output of $\mathcal{B}(y)$ to x . A function $f(\lambda)$ is **poly**(λ) if there exists a $c \in \mathbb{N}$ such that $f(\lambda) = O(\lambda^c)$. If $f(\lambda) = o(\lambda^{-c})$ for all $c \in \mathbb{N}$, we say that $f(\lambda)$ is **negligible** and write $f \leq \text{negl}(\lambda)$. A probability that is $1 - \text{negl}(\lambda)$ is **overwhelming**.

The **statistical distance** between two distributions $\mathcal{X}$ and $\mathcal{Y}$ over the same set is defined as $\Delta(\mathcal{X}, \mathcal{Y}) := \sup_A |\Pr_{\mathcal{X}}[A] - \Pr_{\mathcal{Y}}[A]|$, where $\Pr_{\mathcal{X}}[A]$ is the measure of A under the distribution $\mathcal{X}$. We use $\mathcal{X} \approx_s \mathcal{Y}$ to denote that $\mathcal{X}$ and $\mathcal{Y}$ have negligible statistical distance.

Vectors and matrices. We use bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$) to denote matrices, bold lowercase letters (e.g., $\mathbf{u}, \mathbf{v}$) to denote vectors. For a vector $\mathbf{v}$ of length n , we use $\mathbf{v}_i$ or $\mathbf{v}[i]$ ($1 \leq i \leq n$) to denote its i th element. For two (column) vectors $\mathbf{u}, \mathbf{v}$ of the same length we let $\langle \mathbf{u}, \mathbf{v} \rangle$ denote their inner product, and let $(\mathbf{u}, \mathbf{v})$ denote the vertical concatenation of $\mathbf{u}$ and $\mathbf{v}$, and denote by $[\mathbf{u}^t, \mathbf{v}^t]$ the horizontal concatenation of $\mathbf{u}, \mathbf{v}$'s transposes. For any matrix $\mathbf{B}$, we denote by $s_1(\mathbf{B}) = \max_{\mathbf{u}, \|\mathbf{u}\|=1} \|\mathbf{B}\mathbf{u}\|$ the largest singular value of $\mathbf{B}$. Let $\text{rows}(\mathbf{A})$ denote the set of rows of a matrix $\mathbf{A}$.

2.1 Lattice Basics

An n -dimensional **lattice** Λ is a discrete subgroup of $\mathbb{R}^n$. A lattice is **full rank** if $\text{span}_{\mathbb{R}}(\Lambda) = \mathbb{R}^n$. A coset of a lattice $\Lambda \in \mathbb{R}^n$ is a set of the form $A = \Lambda + \mathbf{a}$ for some $\mathbf{a} \in \mathbb{R}^n$. The **dual lattice** of a lattice $\Lambda \in \mathbb{R}^n$ is the lattice $\Lambda^* := \{\mathbf{x} \in \text{span}_{\mathbb{R}}(\Lambda) : \langle \mathbf{x}, \Lambda \rangle \in \mathbb{Z}\}$. A linear subspace $S \subseteq \mathbb{R}^n$ is a Λ -*subspace* if $S = \text{span}_{\mathbb{R}}(\Lambda \cap S)$.

Discrete Gaussian distributions. For $k \geq 1$ let $\mathcal{D}$ be the Gaussian probability measure on $\mathbb{R}^k$ with density measure $\rho(\mathbf{x}) := e^{-\pi\|\mathbf{x}\|^2}$. For a matrix $\mathbf{S} \in \mathbb{R}^{n \times k}$ we denote by $\mathcal{D}_{\mathbf{S}}$ the distribution obtained by sampling $\mathbf{x} \leftarrow \mathcal{D}$ and outputting $\mathbf{S}\mathbf{x} \in \mathbb{R}^n$. The (possibly non-spherical) distribution $\mathcal{D}_{\mathbf{S}}$ has covariance $\mathbf{\Sigma}/2\pi$ where $\mathbf{\Sigma} = \mathbf{S}\mathbf{S}^\top$. When $\mathbf{S} = s\mathbf{I}_k$ we write $\mathcal{D}_{\mathbf{S}}$ as $\mathcal{D}_s$. For a set $A \subseteq \text{span}_{\mathbb{R}}(\mathbf{S}) \subseteq \mathbb{R}^n$, where A has a positive measure under $\mathcal{D}_{\mathbf{S}}$, we define $\mathcal{D}_{A,\mathbf{S}}$ as the distribution $\mathbf{y} \leftarrow \mathcal{D}_{\mathbf{S}}$ conditioned on $\mathbf{y} \in A$. We will often take A to be a lattice coset $\Lambda + \mathbf{c}$ and obtain a discrete Gaussian distribution $\mathcal{D}_{\Lambda+\mathbf{c},\mathbf{S}}$ over $\Lambda + \mathbf{c}$. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, where $m > n$, and a vector $\mathbf{y} \in \mathbb{Z}_q^n$ in the column span of $\mathbf{A}$, we use $(\mathbf{A})_{\mathbf{S}}^{-1}(\mathbf{y})$ to denote the distribution obtained by sampling from $\mathcal{D}_{\mathbb{Z}^m,\mathbf{S}}$ conditioned on $\mathbf{A}\mathbf{x} = \mathbf{y} \bmod q$. As usual, when $\mathbf{S} = s\mathbf{I}_m$ we write $(\mathbf{A})_s^{-1}(\mathbf{y})$.

The smoothing parameter $\eta_\epsilon(\Lambda)$ [MR04]. For $\epsilon > 0$ and a lattice Λ , the smoothing parameter $\eta_\epsilon(\Lambda)$ of Λ is the smallest positive value σ so that $\rho_{1/\sigma}(\Lambda^*) \leq 1 + \epsilon$. For a discrete Gaussian width parameter $s > \eta_\epsilon(\Lambda)$, all cosets of Λ have roughly equal mass under $\mathcal{D}_s$.

Cyclotomic rings. Let $R := \mathbb{Z}[X]/(X^d + 1)$ where d is a power of two. Denote $K_{\mathbb{R}} := \mathbb{R}[X]/(X^d + 1)$. For a prime q we let $R_q := R/qR$. By convention, an algebraic expression that involves elements in both R and R_q outputs an element in R_q , by first projecting the elements in R into R_q . For example, if $\tau \in R$ and $x \in R_q$ then $x + \tau x$ is in R_q . The **coefficient embedding** $\text{cf} : R \rightarrow \mathbb{Z}^d$ maps an element in R to its d -dimensional coefficient vector. This embedding extends naturally to vectors as $\text{cf} : R^n \rightarrow \mathbb{Z}^{dn}$. For a linear map $f : R^m \rightarrow R^n$ we let $\text{cf}(f) : \mathbb{Z}^{md} \rightarrow \mathbb{Z}^{nd}$ be the linear map that satisfies $\text{cf}(f)(x) = \text{cf}(f(x))$ for all $x \in R$. For a real matrix $\mathbf{S}$, the coefficient embedding can be used to pull to distribution $\mathcal{D}_{\mathbb{Z}^{md},\mathbf{S}}$ to $\mathcal{D}_{R^m,\mathbf{S}}$. For $\mathbf{A} \in R_q^{n \times m}$ and a vector $\mathbf{y} \in R_q^n$ in the column span of $\mathbf{A}$, we use $(\mathbf{A})_{\mathbf{S}}^{-1}(\mathbf{y})$ to denote the distribution obtained by sampling from $\mathcal{D}_{R^m,\mathbf{S}}$ conditioned on $\mathbf{A}\mathbf{x} = \mathbf{y} \bmod q$.

Norms. The ℓ_2 , ℓ_∞ , and ℓ_1 -norm of a matrix $\mathbf{M} \in \mathbb{Z}^{n \times m}$ are defined as

$$\|\mathbf{M}\|_\infty := \max_{i \in [n], j \in [m]} (|\mathbf{M}_{i,j}|), \quad \|\mathbf{M}\|_2 := \left(\sum_{i \in [n], j \in [m]} \mathbf{M}_{i,j}^2 \right)^{1/2}, \quad \|\mathbf{M}\|_1 := \sum_{i \in [n], j \in [m]} |\mathbf{M}_{i,j}|,$$

respectively. For $p \in \{1, 2, \infty\}$, the ℓ_p -norm of $\mathbf{M} \in R^{n \times m}$ is defined by $\|\mathbf{M}\|_p := \|\text{cf}(\mathbf{M})\|_p$. For a finite set $S \subseteq R$, the **operator norm** $\|S\|_{\text{op}}$ of S is

$$\|S\|_{\text{op}} := \max_{a \in S} \left\{ \sup_{y \in R} \frac{\|a \cdot y\|_2}{\|y\|_2} \right\}. \quad (3)$$

Definition 2.1 (Strong Sampling Sets). *We say $S \subseteq R$ is a q -strong sampling set if the difference of any two distinct elements in S is invertible over R_q . More generally, for $w \in \mathbb{N}$, we say S is a (w, q) -strong sampling set if the determinant of any full rank matrix in $S^{(w+1) \times (w+1)}$ is invertible over R_q .*

Let $S_\Delta := S - S \subseteq R$ denote the set of differences of distinct elements in S .

Lemma 2.1 (Gaussian Tail Bound, Lemma 3 of [AGHS13]). *For all $\epsilon \in (0, 1)$, lattice $\Lambda \subseteq R^m$, target $\mathbf{c} \in K_{\mathbb{R}}^m$, and $s \geq \eta_\epsilon(\Lambda)$,*

$$\Pr_{\mathbf{x} \leftarrow \mathcal{D}_{\Lambda, \mathbf{c}, s}} \left[\|\mathbf{x} - \mathbf{c}\| \geq s\sqrt{md} \right] \leq \frac{1 + \epsilon}{1 - \epsilon} \cdot 2^{-md}.$$

Gadget trapdoors and Gaussian Sampling. We will use the gadget trapdoor from [MP12], which builds on several works on constructing lattice trapdoors [Ajt96; GPV08; AP08; ABB10a; ABB10b]. We state these results as applied to the ring R .

Theorem 2.2 (Gadget Trapdoor [MP12]). *Let $b \geq 2$ be a base. Let n, k, w, m, q be lattice parameters with $w > 2n \lceil \log_b q \rceil$, $k := n \lceil \log_b q \rceil$ and $m := w + k$. There exist PPT algorithms (TrapGen, SamplePre, DGSamp) where*

- $\text{TrapGen}(1^n, q, m, b) \rightarrow (\mathbf{A}, \mathbf{T})$: *On input the lattice dimension n , the modulus q , the number of samples m , and base b , the trapdoor-generation algorithm outputs a matrix $\mathbf{A} \in R_q^{n \times m}$ together with a trapdoor $\mathbf{T} \in R_q^{m \times k}$.*
- $\text{SamplePre}(\mathbf{A}, \mathbf{T}, y, \mathbf{S}) \rightarrow x$: *On input a matrix $\mathbf{A} \in R_q^{n \times m}$, a trapdoor $\mathbf{T}$, a target vector $y \in R_q^n$, and a covariance matrix $\mathbf{S} \in \mathbb{R}^{m \times m}$, the preimage-sampling algorithm outputs a vector $x \in R^m$. When $\mathbf{S} = s\mathbf{I}_m$ for some $s \in \mathbb{R}$, we simply write $\text{SamplePre}(\mathbf{A}, \mathbf{T}, y, s) \rightarrow x$.*
- $\text{DGSamp}(\mathbf{A}, \mathbf{T}, \mathbf{S}) \rightarrow x$: *This is the same as $\text{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{0}, \mathbf{S})$ with the target vector being $\mathbf{0}$. This special case has a more efficient sampling algorithm compared to the general preimage sampling. When $\mathbf{S} = s\mathbf{I}_m$ for some $s \in \mathbb{R}$, we simply write $\text{DGSamp}(\mathbf{A}, \mathbf{T}, s) \rightarrow x$.*

These algorithms satisfy the following three properties:

- *Trapdoor distribution:* For $(\mathbf{A}, \mathbf{T}) \leftarrow \text{TrapGen}(1^n, q, m, b)$, it holds that $\mathbf{AT} = \mathbf{G}_b$, where $\mathbf{G}_b \in R_q^{n \times k}$ is the base- b gadget matrix¹. Moreover, $\mathbf{A}$ is close to uniformly random over $R_q^{n \times m}$ with negligible statistical distance.
- *Preimage sampling:* For all width parameter $s > 0$, target vector $\mathbf{t} \in R_q^n$ in $\text{span}_{R_q}(\mathbf{A})$, and all trapdoor $\mathbf{T}$, the output $\mathbf{x} \leftarrow \$ \text{SamplePre}(\mathbf{A}, \mathbf{T}, y, s)$ satisfies that $\mathbf{Ax} = \mathbf{y}$.
- *Preimage distribution:* For all $\mathbf{T}$ with $\mathbf{AT} = \mathbf{G}_b$, and all $s \geq \omega(\sqrt{\log(dn)}) \cdot s_1(\mathbf{T}) \cdot \sqrt{1+b^2}$ and target vector $y \in R_q^n$, the following distributions have negligible statistical distance:

$$\{\mathbf{x} \leftarrow \$ \text{SamplePre}(\mathbf{A}, \mathbf{T}, \mathbf{y}, s)\} \quad \text{vs.} \quad \{\mathbf{x} \leftarrow \$ (\mathbf{A})_s^{-1}(y)\}.$$

Remark 2.1. By Lemma 2.1, we assume that the output of SamplePre has norm smaller than $s\sqrt{md}$. This is without loss of generality as we can abort and resample when it is not the case, which happens with negligible probability.

The following lemmas adapts and combines [WWW25, Lemma 3.1, 3.3, 3.5, 3.6]. Since there is a standard isomorphism between R_q and the sub-ring of anticirculant matrices in $\mathbb{Z}_q^{d \times d}$, we adapt the lemmas in the ring setting.

Lemma 2.3 (Adapted from [GPV08]). *Let n, m, q, b, s be lattice parameters where $m \geq n \lceil \log_b(q) \rceil$ and $s \geq \omega(\sqrt{d \log(dm)})$. Then the distributions $\{\mathbf{G}_b \mathbf{x} : \mathbf{x} \leftarrow \mathcal{D}_{R^m, s}\}$ and $\{u : u \leftarrow \$ R_q^n\}$ have negligible statistical distance.*

Lemma 2.4 (Discrete Gaussian Preimages [WW23], adapted). *Let q be a prime and let n, m, k, b be lattice parameters where $m \geq 2n \lceil \log_b(q) \rceil$ and $k := n \lceil \log_b(q) \rceil$. For all $\ell = \text{poly}(\lambda)$, matrices $\mathbf{A}_1, \dots, \mathbf{A}_\ell \in R_q^{n \times m}$ where $\mathbf{A}_i$'s ($1 \leq i \leq \ell$) marginal distribution is close to uniformly random over $R_q^{n \times m}$, any matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell \in R_q^{n \times k}$, and any target vector $\mathbf{t} \in R_q^{n\ell}$, define matrices,*

$$\mathbf{C}_\ell = \begin{bmatrix} \mathbf{A}_1 & & \mathbf{B}_1 \\ & \ddots & \vdots \\ & & \mathbf{A}_\ell & \mathbf{B}_\ell \end{bmatrix} \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_\ell \end{bmatrix}. \quad (4)$$

Then, for all width parameters $s \geq \omega(\sqrt{d \log(\ell md)})$, the following distributions have negligible statistical distance:

$$\{\mathbf{v} : \mathbf{v} \leftarrow \$ (\mathbf{C}_\ell)_s^{-1}(\mathbf{t})\} \quad \text{and} \quad \left\{ (\mathbf{v}_1, \dots, \mathbf{v}_\ell, \mathbf{v}_{\ell+1}) : \begin{array}{l} \mathbf{v}_{\ell+1} \leftarrow \$ \mathcal{D}_{R^k, s}, \\ \mathbf{v}_i \leftarrow \$ (\mathbf{A}_i)_s^{-1}(\mathbf{t}_i - \mathbf{B}_i \mathbf{v}_{\ell+1}). \end{array} \right\}.$$

¹More precisely, $\mathbf{G}_b := \mathbf{I}_n \otimes [1, b, \dots, b^{\lceil \log_b q \rceil - 1}] \in R_q^{n \times k}$, where we can also view it as a $\mathbb{Z}_q$ -matrix $\mathbf{G}'_b := \mathbf{I}_{nd} \otimes [1, b, \dots, b^{\lceil \log_b q \rceil - 1}] \in \mathbb{Z}_q^{nd \times kd}$ by considering $\mathbf{G}_b$'s corresponding anticirculant matrix.

Smoothing Lemma We recall a simplified version of the smoothing lemma over rings [BL25; LPR13; RSW18; SS11], which is a special case of the leftover hash lemma where the hash input is from a discrete Gaussian distribution.

Lemma 2.5 (Smoothing over rings, adapted from [BL25; LPR13; RSW18; SS11]). *Fix parameters n , prime $q > 5$, and $m \geq 2n \log(q)$. Let $\mathbf{A} \leftarrow \$ R_q^{n \times m}$ and $\mathbf{x} \leftarrow \$ \mathcal{D}_{R^m, s}$ for $s \geq \Omega(\sqrt{d \log(dm)})$. The distributions $(\mathbf{A}, \mathbf{Ax})$ and $(\mathbf{A}, \mathbf{u} \leftarrow \$ R_q^n)$ have negligible statistical distance.*

Rerandomizing Trapdoors. Given a matrix $\mathbf{A} \in R_q^{n \times m}$ and parameter $s' \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$, set $k := n \lceil \log_b(q) \rceil$ and let $\mathcal{O}_{\mathbf{A}, s'}$ be an oracle for discrete Gaussian sampling over cosets of $\Lambda^\perp(\mathbf{A})$ with width s' , we define a trapdoor randomization algorithm **DeriveTD** as follows.

DeriveTD $^{\mathcal{O}_{\mathbf{A}, s'}}(\mathbf{A}, s') \rightarrow \mathbf{T}$: Sample each column of $\mathbf{T} \in R^{m \times k}$ independently by calling $\mathcal{O}_{\mathbf{A}, s'}$, so that $\mathbf{AT} = \mathbf{G}_b$. Output $\mathbf{T}$.

Lemma 2.6 (Lemma 5.7 of [MP12], adapted). *For any $\mathbf{A} \in R_q^{n \times m}$, $s' \geq \eta_\epsilon(\Lambda^\perp(\mathbf{A}))$, and any oracle $\mathcal{O}_{\mathbf{A}, s'}$ whose output distribution is statistically close to $(\mathbf{A})_{s'}^{-1}$, the distribution of $\mathbf{T} \in R^{m \times k}$ is statistically close to $\text{DeriveTD}^{(\mathbf{A})_{s'}^{-1}}(\mathbf{A}, s')$, and $s_1(\mathbf{T}) \leq s' \sqrt{d} \cdot O(\sqrt{m} + \sqrt{k})$ except with negligible probability.*

A convolution Lemma. We will need the following lemma about a linear combination of discrete Gaussians over rings. This lemma builds on previous discrete convolution lemmas [Pei10; BF11; MP13; AGHS13; BPMW16; GM18; CGM19; DGPY20; GMPW20]. However, since the existing lemmas do not support a linear combination of discrete Gaussians where the linear coefficients are ring elements, we prove such a lemma here using Theorem 3.1 from [GMPW20]. First, we need some notation.

- For a vector $\alpha = (\alpha_1, \dots, \alpha_w) \in R^w$, let $\mathbf{T}_\alpha \in R^{t \times tw}$ be the matrix s.t.:

$$\mathbf{T}_\alpha \cdot \mathbf{u} = \alpha_1 \mathbf{u}_1 + \dots + \alpha_w \mathbf{u}_w$$

for all $\mathbf{u} = (\mathbf{u}_1, \dots, \mathbf{u}_w) \in (R^t)^w$. Recall that $\text{cf}(\mathbf{T}_\alpha) \in \mathbb{Z}^{td \times twd}$ is the corresponding integer matrix under the coefficient embedding.

- For a full rank matrix $\mathbf{A} \in R_q^{n \times t}$ we let $\mathcal{L}^\perp(\mathbf{A}) = \{\mathbf{v} \in R^t : \mathbf{Av} = \mathbf{0} \bmod q\}$.
- We say that a vector $\alpha \in R^w$ is a **unit** if there is another vector $\beta \in R^w$ such that $\langle \alpha, \beta \rangle = 1$. It is not difficult to see that if $\alpha = (\alpha_1, \dots, \alpha_w)$ is a unit then $\alpha_1 \mathcal{L}^\perp(\mathbf{A}) + \dots + \alpha_w \mathcal{L}^\perp(\mathbf{A}) = \mathcal{L}^\perp(\mathbf{A})$.

Theorem 2.7 (Ring convolution lemma). *Let R be the ring $R := \mathbb{Z}[X]/(X^d + 1)$ and $R_q := R/qR$. Let $\mathbf{A} \in R_q^{n \times t}$ be full rank, $\alpha = (\alpha_1, \dots, \alpha_w) \in R^w$ is a unit, and $\mathbf{v}_1, \dots, \mathbf{v}_w \in R_q^n$.*

For $\epsilon > 0$ let $s \geq \eta_\epsilon(\text{cf}(\mathcal{L}^\perp(\mathbf{A})) \cap \ker(\text{cf}(\mathbf{T}_\alpha)))$ be a discrete Gaussian width parameter. Then

$$\Delta\left(\sum_{i=1}^w \alpha_i \mathbf{A}_s^{-1}(\mathbf{v}_i), \mathbf{A}_{s, \text{cf}(\mathbf{T}_\alpha)}^{-1}(\sum_{i=1}^w \alpha_i \mathbf{v}_i)\right) < \epsilon/(1 - \epsilon)$$

Proof. Let $\Lambda_w := \mathcal{L}^\perp(\mathbf{I}_w \otimes \mathbf{A}) \subseteq R^{tw}$. There is a shift $\mathbf{c} \in R^{tw}$ such that

$$(\mathbf{I}_w \otimes \mathbf{A}) \cdot \mathbf{c} = (\mathbf{v}_1^\top, \dots, \mathbf{v}_w^\top)^\top$$

Then the distribution $\mathbf{T}_\alpha \cdot \mathcal{D}_{\Lambda_w + \mathbf{c}, s}$ is identical to the distribution $\sum_{i=1}^w \alpha_i \mathbf{A}_s^{-1}(\mathbf{v}_i)$. Similarly, the distribution $\mathcal{D}_{\mathbf{T}_\alpha(\Lambda_w + \mathbf{c}), s, \text{cf}(\mathbf{T}_\alpha)}$ is identical to the distribution $\mathbf{A}_{s, \text{cf}(\mathbf{T}_\alpha)}^{-1}(\sum_{i=1}^w \alpha_i \mathbf{v}_i)$. This is because (i) $\mathbf{T}_\alpha \cdot \Lambda_w = \alpha_1 \mathcal{L}(\mathbf{A}) + \dots + \alpha_w \mathcal{L}(\mathbf{A}) = \mathcal{L}(\mathbf{A})$ thanks to the requirement that α is a unit, and (ii) $\mathbf{A}(\mathbf{T}_\alpha \mathbf{c}) = \sum_{i=1}^w \alpha_i \mathbf{v}_i$. Therefore, $\mathbf{T}_\alpha(\Lambda_w + \mathbf{c})$ is the set of $\mathbf{x} \in R^t$ such that $\mathbf{A}\mathbf{x} = \sum_{i=1}^w \alpha_i \mathbf{v}_i$. Our goal then is to show that

$$\Delta\left(\mathbf{T}_\alpha \cdot \mathcal{D}_{\Lambda_w + \mathbf{c}, s}, \mathcal{D}_{\mathbf{T}_\alpha(\Lambda_w + \mathbf{c}), s, \text{cf}(\mathbf{T}_\alpha)}\right) < \epsilon/(1 - \epsilon)$$

This is the topic of Theorem 3.1 from [GMPW20]. The theorem is stated for general integer lattices, but here we apply it to the special case of Λ_w . The theorem requires the following two conditions to hold: (i) $\ker(\text{cf}(\mathbf{T}_\alpha))$ must be a $\text{cf}(\Lambda_w)$ -subspace, and (ii) $\eta_\epsilon(\text{cf}(\mathcal{L}^\perp(\mathbf{A})) \cap \ker(\text{cf}(\mathbf{T}_\alpha))) \leq s$. Condition (i) is satisfied because $\text{cf}(\Lambda_w)$ contains some integer multiple of every vector in $\ker(\text{cf}(\mathbf{T}_\alpha)) \cap \mathbb{Z}^{tw}$. Condition (ii) is satisfied by assumption on s . $\square$

Lattice Assumptions. We recall the module SIS assumption (MSIS) from [LS15].

Definition 2.2 (Module SIS [LS15]). *For integers $q = q(\lambda)$, $n = n(\lambda)$, $m = \Omega(n \log q)$, and norm bound $B = B(\lambda) \in \mathbb{R}$, the MSIS $_{n,m,q,B}$ assumption stipulates that for all PPT adversaries $\mathcal{A}$,*

$$\Pr \left[\begin{array}{l} \mathbf{A} \leftarrow \$ R_q^{n \times m} \\ \mathbf{x} \in R^n \leftarrow \$ \mathcal{A}(\mathbf{A}) \end{array} : (\mathbf{A}\mathbf{x} = \mathbf{0}) \wedge (0 < \|\mathbf{x}\|_2 < B) \right] < \text{negl}(\lambda),$$

where the equality $\mathbf{A}\mathbf{x} = \mathbf{0}$ is over R_q .

2.2 Forking Lemma

In this work, we rely on the forking lemma [PS00; Sch91; BN06; BCJ08] to prove the security of our aggregate constructions. We follow the presentation in [BN06; BK20].

Definition 2.3 (Simulation). *Fix $Q \in \mathbb{N}$ and sets $\mathcal{X}, \mathcal{H}$ where $|\mathcal{H}| > 1$. An algorithm $\mathcal{S}$ is a simulation algorithm if on input $x \in \mathcal{X}$, $h_1, \dots, h_Q \in \mathcal{H}$, it returns either $\perp$ or a pair $(\text{aux}, i) \in \{0, 1\}^* \times [Q]$. Let IG be a distribution over $\mathcal{X}$, the advantage of $\mathcal{S}$ is defined as*

$$\text{Adv}_{\text{IG}, \mathcal{S}}^{\text{Sim}}(\lambda) := \Pr[\mathcal{S}(x, h_1, \dots, h_Q) \neq \perp],$$

where $x \leftarrow \$ \text{IG}$ and $h_1, \dots, h_Q \leftarrow \$ \mathcal{H}$.

Definition 2.4 (Forking Algorithm). Fix $Q \in \mathbb{N}$ and sets $\mathcal{X}, \mathcal{H}$ where $|\mathcal{H}| > 1$. For a simulation algorithm $\mathcal{S}$, the forking algorithm $\mathcal{F}_{\mathcal{S}}(x)$ is defined as follows:

- Sample random coins for $\mathcal{S}$
- $h_1, \dots, h_Q \leftarrow \$ \mathcal{H}$
- Output $\perp$ and abort if $\mathcal{S}(x, h_1, \dots, h_Q) = \perp$, otherwise let (aux, i) be $\mathcal{S}$'s output
- $h'_i, \dots, h'_Q \leftarrow \$ \mathcal{H}$
- Output $\perp$ and abort if $\mathcal{S}(x, h_1, \dots, h_{i-1}, h'_i, \dots, h'_Q) = \perp$, otherwise let (aux', i') be $\mathcal{S}$'s output
- Return $(\text{aux}, \text{aux}')$ if $i = i'$ and $h_i \neq h'_i$, return $\perp$ otherwise

Let IG be a distribution over $\mathcal{X}$, the advantage of $\mathcal{F}_{\mathcal{S}}$ is defined as

$$\text{Adv}_{\text{IG}, \mathcal{F}_{\mathcal{S}}}^{\text{Fork}}(\lambda) := \Pr_{x \leftarrow \text{IG}} [\mathcal{F}_{\mathcal{S}}(x) \neq \perp] .$$

The general forking lemma connects the success probability of the forking algorithm with that of the simulation algorithm.

Lemma 2.8. Fix $Q \in \mathbb{N}$ and sets $\mathcal{X}, \mathcal{H}$ where $|\mathcal{H}| > 1$. Let IG be a distribution over $\mathcal{X}$, $\mathcal{S}$ a simulation algorithm, and $\mathcal{F}_{\mathcal{S}}$ a forking algorithm for $\mathcal{S}$. Then

$$\text{Adv}_{\text{IG}, \mathcal{S}}^{\text{Sim}} \leq \frac{Q}{|\mathcal{H}|} + \sqrt{Q \cdot \text{Adv}_{\text{IG}, \mathcal{F}_{\mathcal{S}}}^{\text{Fork}}(\lambda)} .$$

2.3 The Value Guessing Game

The proof of security for our signature scheme uses an information theoretic lemma about the inability of the adversary to guess the inner product of a secret vector with a chosen vector, given such inner products with other adaptively chosen vectors. We refer to this as the **value guessing game**, defined next.

Definition 2.5 (Value Guessing Game). For a ring R_q and parameter $w \in \mathbb{N}$, let $\mathcal{S} \subseteq R_q$ be a (w, q) -strong sampling set (as in [Definition 2.1](#)). The game $G_{\text{guess}, \mathcal{A}} = G_{\text{guess}, \mathcal{A}}^{(w, \mathcal{S}, R_q)}$ against an adversary $\mathcal{A}$ is defined as follows:

1. The challenger samples $\mathbf{x} := (x_1, \dots, x_{w+1}) \leftarrow \$ R_q^{w+1}$.
2. The (unbounded) adversary $\mathcal{A}$ makes w adaptive queries. For each query $\alpha^{(j)} \in \mathcal{S}^{w+1}$ ($1 \leq j \leq w$), the challenger returns the inner product $y_j = \langle \alpha^{(j)}, \mathbf{x} \rangle \in R_q$.
3. $\mathcal{A}$ outputs $(\alpha^{(w+1)} \in \mathcal{S}^{w+1}, y_{w+1} \in R_q)$.

$\mathcal{A}$ wins if $y_{w+1} = \langle \alpha^{(w+1)}, \mathbf{x} \rangle$ (over R_q) and the matrix

$$\mathbf{V} := [\alpha^{(1)}, \dots, \alpha^{(w+1)}] \in \mathcal{S}^{(w+1) \times (w+1)} \tag{5}$$

satisfies that $\hat{\mathbf{V}} := \mathbf{V} \bmod q \in R_q^{(w+1) \times (w+1)}$ is full rank over R_q .

Next, we bound the advantage of the value guessing game.

Lemma 2.9. *For all $w \in \mathbb{N}$, (w, q) -strong sampling set $\mathcal{S}$, and (possibly unbounded) adversary $\mathcal{A}$,*

$$\Pr[G_{\text{guess}, \mathcal{A}}] \leq (|R_q| - |R_q^\times|)/|R_q|.$$

Proof. Throughout the proof, for matrix $\mathbf{V} \in \mathcal{S}^{(w+1) \times (w+1)}$, we denote by $\hat{\mathbf{V}} := \mathbf{V} \bmod q \in R_q^{(w+1) \times (w+1)}$. We proceed by showing the two statements below:

1. After fixing the adversary's queries, there still exists an index $j^* \in [w+1]$ such that x_{j^*} is uniformly random over R_q .
2. Even after fixing all queries and $\alpha^{(w+1)}$, the correct value of y_{w+1} is unpredictable as long as $\hat{\mathbf{V}}$ is full rank over R_q and x_{j^*} has high min-entropy.

In the following, we denote by $H_\infty(X|\mathbf{y}, \mathbf{V})$ the min-entropy of variable X given that the y -values $\mathcal{A}$ received is $\mathbf{y}$ and the matrix defined in Eq. (5) is $\mathbf{V}$.

Claim 1. *Let X_j ($1 \leq j \leq W+1$) be the variables for x_j . For all $\mathbf{y} \in R_q^w$ and $\mathbf{V} \in \mathcal{S}^{(w+1) \times (w+1)}$ where $\hat{\mathbf{V}}$ is full rank over R_q , there exists $j^* \in [w+1]$ such that $H_\infty(X_{j^*}|\mathbf{y}, \mathbf{V}) \geq \log_2(|R_q|)$, i.e., X_{j^*} is uniformly random over R_q .*

Proof. Since $\hat{\mathbf{V}}$ is full rank, the submatrix $\hat{\mathbf{V}}_{\overline{w+1}}$ (first w columns) has rank w . Thus, some row $\hat{\mathbf{V}}_{j^*}$ ($j^* \in [w+1]$) can be removed from $\hat{\mathbf{V}}_{\overline{w+1}}$ so that the resulting matrix $\hat{\mathbf{V}}_{\overline{j^*, w+1}}$ is full rank and thus also invertible by definition of (w, q) -strong sampling set. Denote by $\mathbf{x} = (X_1, \dots, X_{w+1})$ and $\mathbf{x}_{j^*} = (X_1, \dots, X_{j^*-1}, X_{j^*+1}, \dots, X_{w+1})$. Writing

$$\mathbf{y}^\top = \mathbf{x}^\top \hat{\mathbf{V}}_{\overline{w+1}} = \mathbf{x}_{j^*}^\top \hat{\mathbf{V}}_{\overline{j^*, w+1}} + \hat{\mathbf{V}}_{j^*}^\top \cdot X_{j^*}. \quad (6)$$

For every possible value of X_{j^*} over R_q , there is a unique solution for $\mathbf{x}_{j^*}$ (as $\hat{\mathbf{V}}_{\overline{j^*, w+1}}$ is invertible). Thus X_{j^*} remains uniform over R_q and the claim holds. $\square$

Claim 2. *Let Y be the variable for $y_{w+1} = \langle \alpha^{(w+1)}, \mathbf{x} \rangle \bmod q$. For all $\mathbf{y} \in R_q^w$ and $\mathbf{V} \in \mathcal{S}^{(w+1) \times (w+1)}$ where $\hat{\mathbf{V}}$ is full rank over R_q ,*

$$H_\infty(Y|\mathbf{y}, \mathbf{V}) \geq \log_2(|R_q|/(|R_q| - |R_q^\times|)).$$

Proof. Denote by $\alpha^{(w+1)}$ the $(w+1)$ -th column of $\hat{\mathbf{V}}$. Let j^* and $\hat{\mathbf{V}}_{\overline{j^*, w+1}}$ be the row index and the full-rank matrix as in Claim 1. Since $\hat{\mathbf{V}}_{\overline{j^*, w+1}}$ is full rank, there exists $\beta \in R_q^w$ such that for some $t \in R_q$,

$$\alpha^{(w+1)} - \hat{\mathbf{V}}_{\overline{w+1}} \times \beta = (\mathbf{0}^{j^*-1}, t, \mathbf{0}^{w+1-j^*}). \quad (7)$$

That is, the linear combination $\hat{\mathbf{V}}_{\overline{j^*, w+1}} \times \boldsymbol{\beta}$ matches the subvector of $\boldsymbol{\alpha}^{(w+1)}$ that excludes $\boldsymbol{\alpha}_{j^*}^{(w+1)}$. However, we claim that $\boldsymbol{\alpha}^{(w+1)} \neq \hat{\mathbf{V}}_{\overline{w+1}} \times \boldsymbol{\beta}$ and thus $t \in R_q$ is *nonzero*: this is because $\hat{\mathbf{V}}$ is full rank and $\boldsymbol{\alpha}^{(w+1)}$ is a non-zero vector linearly independent of the first w columns of $\hat{\mathbf{V}}$, namely, $\hat{\mathbf{V}}_{\overline{w+1}}$.

Recall that by definition,

$$Y = \langle \boldsymbol{\alpha}^{(w+1)}, \mathbf{x} \rangle \quad \text{and} \quad \langle \mathbf{y}, \boldsymbol{\beta} \rangle = \langle \hat{\mathbf{V}}_{\overline{w+1}} \times \boldsymbol{\beta}, \mathbf{x} \rangle$$

where the equalities are over R_q . By Eq. (7), we have $Y - \langle \mathbf{y}, \boldsymbol{\beta} \rangle = t \cdot X_{j^*}$ with $t \neq 0$. Or equivalently, $Y = \langle \mathbf{y}, \boldsymbol{\beta} \rangle + t \cdot X_{j^*}$. Thus, the entropy of Y is the same as that of $t \cdot X_{j^*}$.

We now bound the entropy of $t \cdot X_{j^*}$: For every non-zero $z \in R_q$, there is at most one solution of X_{j^*} such that $t \cdot X_{j^*} = z$; and there are at most $|R_q| - |R_q^\times|$ choices of X_{j^*} where $t \cdot X_{j^*} = 0$. By Claim 1, X_{j^*} is uniformly random over R_q , so the min-entropy $H_\infty(Y|\mathbf{y}, \mathbf{V})$ is bounded as claimed. $\square$

Lemma 2.9 now follows from the above claims. $\square$

3 Statistically Sampleable Homomorphic Vector Commitments

In this section, we define and construct *statistically sampleable* homomorphic vector commitments. This notion extends the schemes introduced in [FSZ22] by further requiring the commitment's distribution to be *oblivious* to the committed vector, and a position opening's distribution depends only on the commitment and the opened value at the position, but not the entire committed vector.

3.1 Definition

Definition 3.1 (Homomorphic Vector Commitment, adapted from [FSZ22]). *Let $\bar{R}$ be a ring and $\mathcal{O}$ an $\bar{R}$ -module. Set $\bar{R}_q := \bar{R}/q\bar{R}$ where $q = q(\lambda) \in \mathbb{N}$. A homomorphic vector commitment (HVC) Π_{VC} with message space $\bar{R}_q^{n_m}$, commitment space $\bar{R}_q^{n_c}$ and opening space $\mathcal{O}$ consists of PPT algorithms (Setup, Com, Open, Vf, wVf):*

Setup($1^\lambda, \ell$) $\rightarrow$ (cp, vp): *on security parameter 1^λ , vector length $\ell \in \mathbb{N}$, output committing parameter cp and verification parameter vp.*

Com(cp, $\mathbf{x}$) $\rightarrow$ (cm, st): *on commitment parameter cp, vector $\mathbf{x} \in (\bar{R}_q^{n_m})^\ell$, output commitment cm $\in \bar{R}_q^{n_c}$ and opening state st $\in \mathcal{Z}$.*

Open(st, i) $\rightarrow \pi_i$: *on opening state st $\in \mathcal{Z}$, index $i \in [\ell]$, output opening $\pi_i \in \mathcal{O}$. We require that this algorithm is deterministic.*

$\text{Vf}(\text{vp}, i, \text{cm}, \mathbf{x}_i, \pi_i) \rightarrow 0/1$: on verification parameter vp , index $i \in [\ell]$, commitment cm , value $\mathbf{x}_i \in R_q^{n_m}$, opening $\pi_i \in \mathcal{O}$, outputs a bit b indicating opening's validity.

$\text{wVf}(\text{vp}, i, \text{cm}, \mathbf{x}_i, \pi_i) \rightarrow 0/1$: Same as $\text{Vf}(\cdot)$, except that it may accept a wider range of openings.

The scheme Π_{VC} must satisfy the properties defined below.

Definition 3.2 (Correctness). *The scheme Π_{VC} satisfies correctness if for all $\lambda \in \mathbb{N}$, $\ell = \ell(\lambda)$, vector $\mathbf{x} \in (\bar{R}_q^{n_m})^\ell$, and index $h \in [\ell]$,*

$$\Pr \left[\begin{array}{l} (\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell); \\ (\text{cm}, \text{st}) \leftarrow \text{Com}(\text{cp}, \mathbf{x}); \quad \text{Vf}(\text{vp}, h, \text{cm}, \mathbf{x}_h, \pi_h) = 1 \\ \pi_h \leftarrow \text{Open}(\text{st}, h) \end{array} \right] = 1.$$

The next definition ensures that *certain* linear combinations of valid openings are also valid openings.

Definition 3.3 (Homomorphic Correctness for Valid Openings). *Fix $k \in \mathbb{N}$ and sets $\mathcal{O}, \mathcal{S} \subseteq \bar{R}$ where $1 \in \mathcal{S}$. The scheme Π_{VC} satisfies $(k, \mathcal{S}, \mathcal{O})$ -homomorphic correctness for valid openings if for all $\lambda \in \mathbb{N}$, $\ell = \ell(\lambda)$, $k' \in [1, k]$, vectors $\mathbf{x}^1, \dots, \mathbf{x}^{k'} \in (\bar{R}_q^{n_m})^\ell$, scalars $\alpha^1, \dots, \alpha^{k'} \in \mathcal{S}$, openings $\pi^1, \dots, \pi^{k'} \in \mathcal{O}$, and index $h \in [\ell]$,*

$$\Pr \left[\begin{array}{l} (\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell); \quad \forall i \in [k'] : \text{Vf}(\text{vp}, h, \text{cm}^i, \mathbf{x}_h^i, \pi_h^i) = 1 \Rightarrow \\ \forall i \in [k'] : (\text{cm}^i, \text{st}^i) \leftarrow \text{Com}(\text{cp}, \mathbf{x}^i) \quad \text{Vf}(\text{vp}, h, \sum_{i \in [k']} \alpha^i \cdot (\text{cm}^i, \mathbf{x}_h^i, \pi_h^i)) = 1 \end{array} \right] = 1.$$

Let $\{\text{st}^i\}_i$ be a list of opening states. The next definition ensures that a linear combination of openings from states $\{\text{st}^i\}_i$ is also an opening w.r.t. the linear combination of states $\{\text{st}^i\}_i$.

Definition 3.4 (Linearity of Opening States). *The scheme Π_{VC} satisfies linearity of opening states if for all $\lambda \in \mathbb{N}$, $\ell = \ell(\lambda)$, index $h \in [\ell]$, vector $\mathbf{x} \in (\bar{R}_q^{n_m})^\ell$, $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, $(\text{cm}, \text{st}) \leftarrow \text{Com}(\text{cp}, \mathbf{x})$, and for any $k \in \mathbb{N}$ and $\text{st}^1, \dots, \text{st}^k \in \mathcal{Z}$, scalars $\alpha^1, \dots, \alpha^k \in \bar{R}$, we have that*

$$\left(\text{st} = \sum_{i \in [k]} \alpha^i \cdot \text{st}^i \right) \implies \left(\sum_{i \in [k]} \alpha^i \cdot \Pi_{\text{VC}}.\text{Open}(\text{st}^i, h) = \Pi_{\text{VC}}.\text{Open}(\text{st}, h) \right).$$

Definition 3.5 (Robust Homomorphism). *The scheme Π_{VC} is robustly homomorphic if for all $\lambda \in \mathbb{N}$, $\ell = \ell(\lambda)$, $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, index $h \in [\ell]$, tuples (cm, x, π) , $(\text{cm}', x', \pi') \in R_q^{n_c} \times R_q^{n_m} \times \mathcal{O}$:*

$$\begin{aligned} \text{Vf}(\text{vp}, h, \text{cm}, x, \pi) = \text{Vf}(\text{vp}, h, \text{cm}', x', \pi') = 1 \\ \implies \text{wVf}(\text{vp}, h, \text{cm} - \text{cm}', x - x', \pi - \pi') = 1. \end{aligned}$$

Definition 3.6 (Weak Verifiability). Fix set $\mathcal{W} \subseteq \bar{R}$ where $1 \in \mathcal{W}$. The scheme Π_{VC} is $\mathcal{W}$ -weakly verifiable if for all λ , $\ell = \ell(\lambda)$, $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, index $h \in [\ell]$, tuple (cm, x, π) , and value $\text{aux} \in \mathcal{W}$,

$$(\text{Vf}(\text{vp}, h, \text{cm}, x, \pi) = 1) \implies (\text{wVf}(\text{vp}, h, \text{aux} \cdot (\text{cm}, x, \pi)) = 1).$$

Definition 3.7 (Position Binding). The scheme Π_{VC} is position binding if for all $\lambda \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$,

$$\Pr \left[\begin{array}{l} (\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell); \\ (h, \text{cm}, x, x', \pi, \pi') \leftarrow \mathcal{A}(\text{cp}, \text{vp}) \end{array} : \begin{array}{l} x \neq x' \wedge \\ \text{wVf}(\text{vp}, h, \text{cm}, x, \pi) = 1 \wedge \\ \text{wVf}(\text{vp}, h, \text{cm}, x', \pi') = 1 \end{array} \right] \leq \text{negl}(\lambda).$$

Bootstrapping HVC. For $\ell \in \mathbb{N}$, we say Π_{VC} is an ℓ -homomorphic vector commitment scheme if the **Setup** algorithm of Π_{VC} always fixes the vector length as ℓ . We introduce a simple lemma that bootstraps the supported vector length of a commitment from ℓ to $\ell \cdot \ell'$.

Lemma 3.1 (Bootstrapping HVC). Fix any $\ell, \ell' \in \mathbb{N}$. Let Π_ℓ (resp. $\Pi_{\ell'}$) be an ℓ -homomorphic (resp. ℓ' -homomorphic) vector commitment scheme with message space $\bar{R}_q^{n_m}$ and commitment space $\bar{R}_q^{n_c}$, respectively. Let Π_1 be a 1-homomorphic vector commitment² with message space $\bar{R}_q^{n_c}$ and commitment space $\bar{R}_q^{n_m}$. Then [Construction A.1](#) is an $(\ell \cdot \ell')$ -homomorphic vector commitment scheme $\Pi_{\ell\ell'}$ with message space $\bar{R}_q^{n_m}$ and commitment space $\bar{R}_q^{n_c}$.

Proof. We defer the proof to [Appendix A.1](#). □

Remark 3.1. Π_1 is used to compress a Π_ℓ -commitment in $\bar{R}_q^{n_c}$ to the message space $\bar{R}_q^{n_m}$ of $\Pi_{\ell'}$. We do not need Π_1 if $n_c \leq n_m$.

Remark 3.2. The above bootstrapping lemma leverages the fact that an HVC's commitment space and message space are *both* $\bar{R}_q$ -modules. In particular, a Pedersen commitment whose message space is a prime field while the commitment space is an (elliptic curve) group does not allow for bootstrapping, even though it is linearly homomorphic.

Statistical sampleability. Next, we introduce the notion of *statistical sampleability*. Intuitively, it means that:

1. The commitment's distribution is *oblivious* to the committed vector.
2. For every $h \in [\ell]$, the opening at index h follows a distribution $\mathcal{D}$ that depends only on the commitment and the opened value *at position* h , but not the other values in the committed vector.

²An 1-homomorphic vector commitment is essentially a linearly homomorphic collision resistant hash.

Construction 3.1.

- $\text{Setup}(1^\lambda, \ell \cdot \ell') \rightarrow (\text{cp}, \text{vp})$:
 - $(\text{cp}_{\ell'}, \text{vp}_{\ell'}) \leftarrow \Pi_{\ell'}. \text{Setup}(1^\lambda, \ell')$, $(\text{cp}_\ell, \text{vp}_\ell) \leftarrow \Pi_\ell. \text{Setup}(1^\lambda, \ell)$,
 $(\text{cp}_1, \text{vp}_1) \leftarrow \Pi_1. \text{Setup}(1^\lambda, 1)$
 - output $\text{cp} := (\text{cp}_{\ell'}, \text{cp}_\ell, \text{cp}_1)$ and $\text{vp} := (\text{vp}_{\ell'}, \text{vp}_\ell, \text{vp}_1)$
- $\text{Com}(\text{cp}, \mathbf{x}) \rightarrow (\text{cm}, \text{st})$:
 - parse $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^{\ell'})$ where $\mathbf{x}^i \in (\bar{R}_q^{n_m})^\ell$ for all $i \in [\ell']$
 - $(\text{cm}^i, \text{st}^i) \leftarrow \Pi_\ell. \text{Com}(\text{cp}_\ell, \mathbf{x}^i)$ for all $i \in [\ell']$
 - $(m^i, \text{st}_o^i) \leftarrow \Pi_1. \text{Com}(\text{cp}_1, \text{cm}^i)$ for all $i \in [\ell']$
 - $(\text{cm}, \text{st}') \leftarrow \Pi_{\ell'}. \text{Com}(\text{cp}_{\ell'}, (m_1, \dots, m^{\ell'}))$
 - output cm and $\text{st} := (\text{st}', (\text{st}^i, \text{st}_o^i)_{i \in [\ell']}, (\text{cm}^i, m^i)_{i \in [\ell']})$
- $\text{Open}(\text{st}, h = (i \in [\ell'], j \in [\ell])) \rightarrow \pi_h$:
 - parse $\text{st} := (\text{st}', (\text{st}^k, \text{st}_o^k)_{k \in [\ell']}, (\text{cm}^k, m^k)_{k \in [\ell']})$
 - $\pi_j^i \leftarrow \Pi_\ell. \text{Open}(\text{st}^i, j)$
 - $o^i \leftarrow \Pi_1. \text{Open}(\text{st}_o^i, 1)$
 - $\pi'_i \leftarrow \Pi_{\ell'}. \text{Open}(\text{st}', i)$
 - output $\pi_h := (\pi'_i, o^i, \pi_j^i, \text{cm}^i, m^i)^a$
- $\text{Vf}(\text{vp}, h = (i \in [\ell'], j \in [\ell])) \rightarrow 0/1$:
 - parse $\pi_h := (\pi'_i, o^i, \pi_j^i, \text{cm}^i, m^i)$
 - output 1 if $\Pi_{\ell'}. \text{Vf}(\text{vp}_{\ell'}, i, \text{cm}, m^i, \pi'_i) = 1$ and $\Pi_1. \text{Vf}(\text{vp}_1, 1, m^i, \text{cm}^i, o^i) = 1$
 and $\Pi_\ell. \text{Vf}(\text{vp}_\ell, j, \text{cm}^i, \mathbf{x}_h, \pi_j^i) = 1$, otherwise, output 0
- $\text{wVf}(\text{vp}, h, \text{cm}, \mathbf{x}_h, \pi_h) \rightarrow 0/1$: Same as $\text{Vf}(\cdot)$, except that the calls to Vf in Π_ℓ , $\Pi'_{\ell'}$, Π_1 are replaced with wVf

^aIn many cases, cm^i uniquely determines o^i, m^i , so there's no need to include o^i, m^i in the proof.

Figure 1: The $\ell \cdot \ell'$ -slot homomorphic vector commitment scheme $\Pi_{\ell\ell'}$.

3. For every $h \in [\ell]$, a linear combination of the openings at index h follows a distribution $\mathcal{D}$ that depends only on the *linearly-combined* commitment and the *linearly-combined* opened value at position h , but not the individual commitments and the rest of committed vector values.

Importantly, the distribution $\mathcal{D}$ is not necessarily efficiently sampleable.

Definition 3.8 (Statistical Sampleability). *Fix $k \in \mathbb{N}$ and $\mathcal{S} \subseteq \bar{R}$ where $1 \in \mathcal{S}$. The scheme Π_{VC} is $(k, \mathcal{S})$ -statistically sampleable if for all $\lambda \in \mathbb{N}$, $\ell = \ell(\lambda)$, $k' \in [1, k]$, vector $\alpha := (\alpha^1, \dots, \alpha^{k'}) \in \mathcal{S}^{k'}$ that is a unit in $\bar{R}^{k'}$, with overwhelming probability over $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, there exists a family of distributions $\{\mathcal{D}_{h,x,c}^\alpha\}_{h \in [\ell], x \in R_q^{n_m}, c \in R_q^{n_c}}$ (that may not be efficiently sampleable), such that the following distributions have negligible statistical distance:*

$$\mathcal{D}_0 := \left((\text{cm}^i)_{i \in [k']}, (\pi_h^*)_{h \in [\ell]} : \begin{array}{ll} \forall i \in [k'] : & \mathbf{x}^i \leftarrow_{\$} (R_q^{n_m})^\ell, (\text{cm}^i, \text{st}^i) \leftarrow \text{Com}(\text{cp}, \mathbf{x}^i); \\ \forall h \in [\ell] : & \pi_h^* := \sum_{i \in [k']} \alpha^i \cdot \text{Open}(\text{st}^i, h) \end{array} \right),$$

$$\mathcal{D}_1 := \left((\text{cm}^i)_{i \in [k']}, (\pi_h^*)_{h \in [\ell]} : \begin{array}{ll} \forall i \in [k'] : & \mathbf{x}^i \leftarrow_{\$} (R_q^{n_m})^\ell, \text{cm}^i \leftarrow_{\$} R_q^{n_c}; \\ \forall h \in [\ell] : & \pi_h^* \leftarrow \mathcal{D}_{h, \sum_{i \in [k']} \alpha^i \cdot (\mathbf{x}^i[h], \text{cm}^i)} \end{array} \right).$$

In particular, the regular sampleability is implied by setting $k' = 1$ and $\alpha^1 = 1$.

4 Constructing a Succinct Hom. Vector Commitment

In this section, we present a statistically sampleable homomorphic vector commitment scheme whose opening proof size is *independent* of the vector length. This is quite different from a Merkle vector commitment, where the opening proof size is logarithmic in the length of the vector.

For all $\ell \leq 2^d$ and $i \in [\ell]$, we denote by $\hat{i} := \sum_{i=0}^{d-1} c_i X^i \in R_q$ the ring element whose coefficient vector $(c_0, \dots, c_{d-1}) \in \{0, 1\}^d$ maps to the binary representation of $i - 1$, that is, $i - 1 = \sum_{i=0}^{d-1} c_i 2^i$. We describe [Construction 4.1](#) in [Figure 2](#): Similar to the construction from [\[WWW25; WW23\]](#), given a vector $\mathbf{x}$, the commitment cm and the openings $(\pi_1, \dots, \pi_\ell)$ are jointly sampled as a (low-norm) preimage w.r.t. the matrix $\mathbf{D}_\ell$ defined in [Eq. \(8\)](#), such that $\mathbf{A}_h \pi_h = \text{cm} + (\mathbf{x}_h, \mathbf{0}^{n-n_m})$ for every $h \in [\ell]$. Preimage sampling is made possible by publishing a trapdoor $\mathbf{T}^*$ for $\mathbf{D}_\ell$ in the setup phase. The key difference from [\[WWW25; WW23\]](#) is that we use a matrix $\mathbf{A}_h$ that is $\log(\ell)$ -factor narrower. This is achieved by deriving $\mathbf{A}_h$ from an R_q -linear combination of two narrow matrices $\mathbf{A}_{[0]}, \mathbf{A}_{[1]}$, more precisely, $\mathbf{A}_h := [\mathbf{A} \mid \mathbf{A}_{[0]} + \hat{h} \cdot \mathbf{A}_{[1]}]$.

Statistical sampleability of the openings is proved by combining the techniques of [\[WWW25\]](#) with the convolution lemma ([Theorem 2.7](#)). Position binding is proved via a reduction to MSIS (inspired by techniques from [\[DM14\]](#)). We embed trapdoors $\mathbf{R}_{[0]}, \mathbf{R}_{[1]}$ into $\mathbf{A}_{[0]}$ and

$\mathbf{A}_{[1]}$, respectively, allowing us to obtain trapdoors for all matrices $\{\mathbf{A}_h\}_h$ except for a random index $h^* \in [\ell]$. The MSIS challenge instance is then embedded in $\mathbf{A}_{h^*}$. Consequently, if a position-binding adversary $\mathcal{A}$ attacks slot h^* , which occurs with probability $1/\ell$, the reduction can use $\mathcal{A}$ to break MSIS. However, more care is needed because the reduction must publish a trapdoor $\mathbf{T}^*$ that is indistinguishable from the one used in an honest execution. We achieve this via the trapdoor re-randomization technique of [Lemma 2.6](#), at the cost of moderately degrading the quality of $\mathbf{T}^*$.

Theorem 4.1. *Let $R := \mathbb{Z}[X]/(X^d + 1)$ and $R_q := R/qR$ such that every R_q -element with coefficients in $\{-1, 0, 1\}$ are invertible. Let $(\text{TrapGen}, \text{SamplePre})$ be a gadget trapdoor sampling scheme as in [Theorem 2.2](#), with parameters n, w, m, b as in [Construction 4.1](#). Set*

$$s' \geq \omega(\sqrt{\log(dn)}) \cdot O(\sqrt{dm}) \cdot \sqrt{1 + b^2} \cdot \sqrt{\log \ell}, \quad (10)$$

$$s \geq \omega(\sqrt{\log(\ell dn)}) \cdot O(\sqrt{\ell dm}) \cdot s'. \quad (11)$$

For all $k \in \mathbb{N}$, $\mathcal{S} \subseteq R$, and all $\mathcal{W} \subseteq R$ with operator norm $\|\mathcal{W}\|_{\text{op}}$, let $\mathcal{O} := \{o \in R : \|o\| \leq s \cdot \sqrt{md}\}$ and

$$B := ks\sqrt{md} \cdot \|\mathcal{S}\|_{\text{op}}, \quad B_w := B \cdot \max(2, \|\mathcal{W}\|_{\text{op}}), \quad B_{\text{sis}} := 2B_w \cdot \Theta(d\sqrt{w \log \ell \cdot \log(dm)}).$$

Assume that $\text{MSIS}_{n-n_m, \bar{m}, q, B_{\text{sis}}}$ is hard, then for all $\ell \leq 2^d$, [Construction 4.1](#) is an ℓ -homomorphic vector commitment with statistical sampleability.

Remark 4.1. [Construction 4.1](#) has message space $R_q^{n_m}$, commitment space R_q^n , opening space $\mathcal{O} := \{o \in R^m : \|o\| \leq B\}$, verification parameter size $O(n^2 \log_b(q))$, and committing parameter size $O(\ell^2 n^2 \log_b^2(q))$. The committing parameter size is quadratic in vector length ℓ , but we can improve it to sublinear in ℓ via the bootstrapping lemma ([Lemma 3.1](#)).

Remark 4.2. When the committed vector $\mathbf{x}$ is uniformly random by default, there is a variant of [Construction 4.1](#) that replaces the `SamplePre` algorithm in the committing procedure with a more efficient Gaussian sampling procedure `DGSamp`. The idea is to set $\mathbf{x}$ as the “hash” of the Gaussian samples $(\pi_1, \dots, \pi_\ell)$ (i.e., $\mathbf{x}_i := \langle \mathbf{a}, \pi_i \rangle$ for some uniformly random $\mathbf{a} \in R_q^m$), and $\mathbf{x}$ is close to uniformly random by the Leftover Hash Lemma.

Proof of Theorem 4.1. The theorem follows from [Lemma 4.2](#), [Lemma 4.4](#), [Lemma 4.3](#), [Lemma 4.5](#), [Lemma 4.6](#), [Lemma 4.7](#), and [Lemma 4.8](#) provided below. $\square$

Lemma 4.2 (Correctness). [Construction 4.1](#) is correct ([Definition 3.2](#)) for $B := s \cdot \sqrt{md}$.

Proof. Fix any $\ell \in \mathbb{N}$, vectors $\mathbf{x} \in (R_q^{n_m})^\ell$, and index $h \in [\ell]$. Let $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, $(\text{cm}, \text{st}) \leftarrow \text{Com}(\text{cp}, \mathbf{x})$ and $\pi_h \leftarrow \text{Open}(\text{st}, h)$. By [Remark 2.1](#), we have that $\|\pi_h\| \leq s \cdot \sqrt{md}$ and

$$\mathbf{A}_h \pi_h = \text{cm} + (\mathbf{x}_h, \mathbf{0}^{n-n_m}).$$

Hence $\text{Vf}(\text{vp}, h, \text{cm}, \mathbf{x}_h, \pi_h) = 1$ by construction. $\square$

Construction 4.1. Parameters: $s', s, b \geq 2$, $q = q(\lambda)$, $n = n(\lambda)$, $w := n \lceil \log_b(q) \rceil$, $\bar{m} \geq 2w$, $m := \bar{m} + w$, bounds $B, B_w \in \mathbb{R}$

- $\text{Setup}(1^\lambda, \ell) \rightarrow (\text{cp}, \text{vp})$:

- Sample $(\mathbf{A}, \mathbf{T}) \leftarrow \text{TrapGen}(1^n, q, \bar{m}, b)$, $\mathbf{A}_{[0]}, \mathbf{A}_{[1]} \leftarrow \$ R_q^{n \times w}$
- Define matrix

$$\mathbf{D}_\ell = \begin{bmatrix} \mathbf{A}_1 & & \mathbf{G}_b \\ & \ddots & \vdots \\ & & \mathbf{A}_\ell & \mathbf{G}_b \end{bmatrix} \in R_q^{\ell n \times (\ell m + w)}, \quad (8)$$

where $\mathbf{A}_i := [\mathbf{A} \mid \mathbf{A}_{[0]} + \hat{i} \cdot \mathbf{A}_{[1]}] \in R_q^{n \times m}$ and $\mathbf{G}_b \in R_q^{n \times w}$.

- Define sampling oracle $\mathcal{O}_{\mathbf{D}_\ell, s'}$ that on input $\mathbf{t} \in R_q^{n\ell}$ does the following:
 - * $\mathbf{v}_{\ell+1} \leftarrow \$ \mathcal{D}_{R^w, s'}$
 - * Let $\mathbf{t} = (\mathbf{t}_1, \dots, \mathbf{t}_\ell) \in R_q^{n\ell}$, for all $i \in [\ell]$, set:

$$\mathbf{v}_i \leftarrow \text{SamplePre}(\mathbf{A}_i, \mathbf{T}' := \begin{bmatrix} \mathbf{T} \\ \mathbf{0} \end{bmatrix}, \mathbf{t}_i - \mathbf{G}_b \mathbf{v}_{\ell+1}, s')$$
 - * Output $\mathbf{v} := (\mathbf{v}_1, \dots, \mathbf{v}_\ell, \mathbf{v}_{\ell+1}) \in R^{\ell m + w}$
- Sample $\mathbf{T}^* \leftarrow \text{DeriveTD}^{\mathcal{O}_{\mathbf{D}_\ell, s'}}(\mathbf{D}_\ell, s')$ where DeriveTD is defined above [Lemma 2.6](#).
- Output $\text{vp} := (\mathbf{A}, \mathbf{A}_{[0]}, \mathbf{A}_{[1]})$ and $\text{cp} := (\text{vp}, \mathbf{T}^* \in R^{(\ell m + w) \times \ell w})$

- $\text{Com}(\text{cp}, \mathbf{x} \in (R_q^{n_m})^\ell) \rightarrow (\text{cm}, \text{st})$:

- Parse $\text{cp} = (\mathbf{A}, \mathbf{A}_{[0]}, \mathbf{A}_{[1]}, \mathbf{T}^*)$
- Define $\mathbf{D}_\ell$ as in Eq. (8). For all $i \in [\ell]$, set $\mathbf{t}_i := (\mathbf{x}_i, \mathbf{0}^{n-n_m}) \in R_q^n$, and sample

$$(\pi_1, \dots, \pi_\ell, \hat{c}) \leftarrow \$ \text{SamplePre}(\mathbf{D}_\ell, \mathbf{T}^*, \mathbf{t} := (\mathbf{t}_1, \dots, \mathbf{t}_\ell), s) \quad (9)$$

- Output $\text{cm} := -\mathbf{G}_b \times \hat{c} \in R_q^n$ and $\text{st} := (\pi_1, \dots, \pi_\ell) \in (R^m)^\ell$

- $\text{Open}(\text{st}, h \in [\ell]) \rightarrow \pi_h$: Parse $\text{st} = (\pi_1, \dots, \pi_\ell)$ and output $\pi_h \in R^m$

- $\text{Vf}(\text{vp}, h \in [\ell], \text{cm}, \mathbf{x}_h, \pi_h) \rightarrow b$:

- Parse $\text{vp} = (\mathbf{A}, \mathbf{A}_{[0]}, \mathbf{A}_{[1]})$ and set $\mathbf{A}_h := [\mathbf{A} \mid \mathbf{A}_{[0]} + \hat{h} \cdot \mathbf{A}_{[1]}]$
- Accept and set $b = 1$ if $\|\pi_h\| \stackrel{?}{\leq} B$ and $\mathbf{A}_h \pi_h \stackrel{?}{=} \text{cm} + (\mathbf{x}_h, \mathbf{0}^{n-n_m})$.

- $\text{wVf}(\text{vp}, h \in [\ell], \text{cm}, \mathbf{x}_h, \pi_h) \rightarrow b$: Same as Vf , but change bound B to B_w

Figure 2: The ℓ -slot statistically sampleable hom. vector commitment scheme Π_{VC}^ℓ .

Lemma 4.3 (Homomorphic Correctness for Valid Openings). *Let $\mathcal{S}$, $\mathcal{O}$ and bound B be defined as in [Theorem 4.1](#). [Construction 4.1](#) satisfies $(k, \mathcal{S}, \mathcal{O})$ -homomorphic correctness for valid openings ([Definition 3.3](#)).*

Proof. Fix any $\ell \in \mathbb{N}$, $k' \in [1, k]$, vectors $\mathbf{x}^1, \dots, \mathbf{x}^{k'} \in (R_q^{n_m})^\ell$, scalars $\alpha^1, \dots, \alpha^{k'} \in \mathcal{S}$ and index $h \in [\ell]$. Let $(\mathbf{cp}, \mathbf{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, and for all $i \in [k']$, let $(\mathbf{cm}^i, \mathbf{st}^i) \leftarrow \text{Com}(\mathbf{cp}, \mathbf{x}^i)$. Consider any $\pi_h^1, \dots, \pi_h^{k'} \in \mathcal{O}$ such that $\text{Vf}(\mathbf{vp}, h, \mathbf{cm}^i, \mathbf{x}_h^i, \pi_h^i) = 1$. We claim that:

$$\text{Vf}(\mathbf{vp}, h, \sum_{i \in [k']} \alpha^i \cdot (\mathbf{cm}^i, \mathbf{x}_h^i, \pi_h^i)) = 1. \quad (12)$$

First, for all $i \in [k']$, $\text{Vf}(\mathbf{vp}, h, \mathbf{cm}^i, \mathbf{x}_h^i, \pi_h^i) = 1$ implies that

$$\mathbf{A}_h \pi_h^i = \mathbf{cm}^i + (\mathbf{x}_h^i, \mathbf{0}^{n-n_m}).$$

Define

$$(\mathbf{cm}^*, \mathbf{x}_h^*, \pi_h^*) := \sum_{i \in [k']} \alpha^i \cdot (\mathbf{cm}^i, \mathbf{x}_h^i, \pi_h^i).$$

By linearity of matrix vector multiplications, it holds that $\mathbf{A}_h \pi_h^* = \mathbf{cm}^* + (\mathbf{x}_h^*, \mathbf{0}^{n-n_m})$.

Second, since $\pi_h^1, \dots, \pi_h^{k'} \in \mathcal{O}$, we have that $\|\pi_h^i\| \leq s \cdot \sqrt{md}$ for all $i \in [k']$ and

$$\|\pi_h^*\| \leq (\|\pi_h^1\| + \dots + \|\pi_h^{k'}\|) \cdot \|\mathcal{S}\|_{\text{op}} \leq k' s \sqrt{md} \cdot \|\mathcal{S}\|_{\text{op}} \leq B.$$

Hence Eq. (12) holds, which finishes the proof. $\square$

Lemma 4.4 (Linearity of Opening States). [Construction 4.1](#) satisfies linearity of opening states ([Definition 3.4](#)).

Proof. Fix any $\ell \in \mathbb{N}$, index $h \in [\ell]$, and $\mathbf{x} \in (\bar{R}_q^{n_m})^\ell$. Let $(\mathbf{cp}, \mathbf{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, and let $(\mathbf{cm}, \mathbf{st}) \leftarrow \text{Com}(\mathbf{cp}, \mathbf{x})$. Fix $\mathbf{st}^1, \dots, \mathbf{st}^k \in \mathcal{Z}$ and $\alpha^1, \dots, \alpha^k \in \bar{R}$ such that $\mathbf{st} = \sum_{i \in [k]} \alpha^i \cdot \mathbf{st}^i$. For all $i \in [k]$, parse $\mathbf{st}^i$ as $(\pi_1^i, \dots, \pi_\ell^i)$. Then for all $i \in [k]$, $\text{Open}(\mathbf{st}^i, h) = \pi_h^i$, and:

$$\sum_{i \in [k]} \alpha^i \cdot \text{Open}(\mathbf{st}^i, h) = \sum_{i \in [k]} \alpha^i \cdot \pi_h^i = \text{Open}\left(\sum_{i \in [k]} \alpha^i \cdot \mathbf{st}^i, h\right) = \text{Open}(\mathbf{st}, h).$$

Since $\text{Vf}(\mathbf{vp}, h, \mathbf{cm}, \mathbf{x}_h, \text{Open}(\mathbf{st}, h)) = 1$ by the correctness of [Construction 4.1](#) ([Lemma 4.2](#)), the lemma follows. $\square$

Lemma 4.5 (Robust Homomorphism). [Construction 4.1](#) is robustly homomorphic ([Definition 3.5](#)) for $B_w := 2B$.

Proof. Fix $\ell \in \mathbb{N}$, $h \in [\ell]$, and any $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$. If

$$\text{Vf}(\text{vp}, h, \text{cm}, x, \pi) = \text{Vf}(\text{vp}, h, \text{cm}', x', \pi') = 1$$

for some (cm, x, π) and (cm', x', π') . Then $\|\pi\|, \|\pi'\| \leq B$ and

$$\mathbf{A}_h \pi = \text{cm} + (x, \mathbf{0}^{n-n_m}) \quad \text{and} \quad \mathbf{A}_h \pi' = \text{cm}' + (x', \mathbf{0}^{n-n_m}).$$

By linearity,

$$\mathbf{A}_h(\pi - \pi') = \text{cm} - \text{cm}' + (x - x', \mathbf{0}^{n-n_m})$$

and $\|\pi - \pi'\| \leq \|\pi\| + \|\pi'\| \leq 2B = B_w$, and thus $\text{wVf}(\text{vp}, h, \text{cm} - \text{cm}', x - x', \pi - \pi') = 1$, which finishes the proof. $\square$

Lemma 4.6 (Weak Verifiability). *Construction 4.1 is $\mathcal{W}$ -weakly verifiable (Definition 3.6) for $B_w := B\|\mathcal{W}\|_{\text{op}}$.*

Proof. For all $\ell = \ell(\lambda)$, $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, index $h \in [\ell]$, tuple (cm, x, π) such that $\text{Vf}(\text{vp}, h, \text{cm}, x, \pi) = 1$, it holds that $\|\pi\| \leq B$ and

$$\mathbf{A}_h \pi = \text{cm} + (x, \mathbf{0}^{n-n_m}).$$

Thus for all $\text{aux} \in \mathcal{W}$, let $(\text{cm}^*, x^*, \pi^*) := \text{aux} \cdot (\text{cm}, x, \pi)$, it holds that $\|\pi^*\| \leq B\|\mathcal{W}\|_{\text{op}}$, and by linearity,

$$\mathbf{A}_h \pi^* = \text{cm}^* + (x^*, \mathbf{0}^{n-n_m}),$$

and thus $\text{wVf}(\text{vp}, h, \text{aux} \cdot (\text{cm}, x, \pi)) = 1$, finishing the proof. $\square$

Lemma 4.7 (Statistical Sampleability). *Let $s_{\text{conv}} := \max_{h \in [\ell]} \{\eta_\epsilon(\text{cf}(\mathcal{L}^\perp(\mathbf{A}_h)) \cap \ker(\text{cf}(\mathbf{T}_\alpha)))\}$. Then for all $s > \max(s_{\text{conv}}, \omega(\sqrt{d \log(\ell m d)}))$, Construction 4.1 is $(k, \mathcal{S})$ -statistically sampleable (Definition 3.8).*

Proof. For $\ell \in \mathbb{N}$, $k' \in [1, k]$, vector $\alpha := (\alpha^1, \dots, \alpha^{k'})$ that is a unit in $R^{k'}$, $(\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell)$, index $h \in [\ell]$, target value $x \in R_q^{n_m}$, and target commitment $\mathbf{c} \in R_q^n$, we define distribution

$$\mathcal{D}_{h,x,c}^\alpha := (\mathbf{A}_h)_{\mathbf{S}}^{-1}(\mathbf{c} + (x, \mathbf{0}^{n-n_m})),$$

here the matrix $\mathbf{S}$ is $\mathbf{S} := s \cdot \text{cf}(\mathbf{T}_\alpha)$ where $\mathbf{T}_\alpha$ is the matrix satisfying $\mathbf{T}_\alpha \cdot \mathbf{v} = \langle \alpha, \mathbf{v} \rangle$.

To prove statistical sampleability, we consider the following hybrid distributions:

Hyb₀: Output $(\text{cm}^i)_{i \in [k']}$, $(\pi_h^*)_{h \in [\ell]}$, where $(\mathbf{x}^1, \dots, \mathbf{x}^{k'}) \leftarrow_{\$} ((R_q^{n_m})^\ell)^{k'}$; for all $i \in [k']$, $(\text{cm}^i, \text{st}^i) \leftarrow \text{Com}(\text{cp}, \mathbf{x}^i)$, and for all $h \in [\ell]$, $\pi_h^* := \sum_{i \in [k']} \alpha^i \cdot \text{Open}(\text{st}^i, h)$.

Hyb₁: Almost identical to Hyb₀, but for every $i \in [k']$, the samples $(\pi_1^i, \dots, \pi_\ell^i, \hat{c}^i)$ in Eq. (9) follows the distribution $(\mathbf{D}_\ell)_s^{-1}(\mathbf{t}^i)$.

Hyb₂: Almost identical to **Hyb₁**, but for every $i \in [k']$, $(\pi_1^i, \dots, \pi_\ell^i, \hat{c}^i)$ in Eq. (9) is sampled as follows: $\hat{c}^i \leftarrow \mathcal{D}_{R^w, s}$, for all $h \in [\ell]$, $\pi_h^i \leftarrow (\mathbf{A}_h)_s^{-1}(-\mathbf{G}_b \times \hat{c}^i + (\mathbf{x}_h^i, \mathbf{0}^{n-n_m}))$.

Hyb₃: Almost identical to **Hyb₂**, but for every $i \in [k']$, the commitment cm^i is uniformly random over R_q^n rather than being $\text{cm}^i := -\mathbf{G}_b \times \hat{c}^i$. Moreover, for all $i \in [k']$ and $h \in [\ell]$, we set $\pi_h^i \leftarrow (\mathbf{A}_h)_s^{-1}(\text{cm}^i + (\mathbf{x}_h^i, \mathbf{0}^{n-n_m}))$, and $\pi_h^* := \sum_{i \in [k']} \alpha^i \cdot \pi_h^i$.

Hyb₄: Almost identical to **Hyb₃**, but for all $h \in [\ell]$, the opening π_h^* is sampled as

$$\pi_h^* \leftarrow \mathcal{D}_{h, \sum_{i \in [k']} \alpha^i \cdot (\mathbf{x}_h^i, \text{cm}^i)}^\alpha.$$

Note that **Hyb₀** and **Hyb₄** are identical to the distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ in Definition 3.8.

Claim 3. $\text{Hyb}_0 \approx_s \text{Hyb}_1$.

Proof. The claim follows from the preimage distribution property from Theorem 2.2 and a straightforward hybrid argument. $\square$

Claim 4. $\text{Hyb}_1 \approx_s \text{Hyb}_2$.

Proof. The claim follows from Lemma 2.4 and a straightforward hybrid argument. $\square$

Claim 5. $\text{Hyb}_2 \approx_s \text{Hyb}_3$.

Proof. The claim follows from Lemma 2.3 and a straightforward hybrid argument. $\square$

Claim 6. If $s \geq \max_{h \in [\ell]} \{\eta_\epsilon(\text{cf}(\mathcal{L}^\perp(\mathbf{A}_h)) \cap \ker(\text{cf}(\mathbf{T}_\alpha))\}$, then $\text{Hyb}_3 \approx_s \text{Hyb}_4$.

Proof. Since α is a unit in $R^{k'}$, there exists $\beta \in R^{k'}$ where $\langle \alpha, \beta \rangle = 1$. Thus the claim follows from the convolution lemma (Theorem 2.7) and a straightforward hybrid argument. $\square$

Lemma 4.7 then follows from the claims above. $\square$

Lemma 4.8 (Position Binding). Assume that $\text{MSIS}_{n-n_m, \bar{m}, q, B_{\text{sis}}}$ is hard, then Construction 4.1 is position binding (Definition 3.7).

Proof. For every $\lambda \in \mathbb{N}$, $\ell = \ell(\lambda)$, and PPT adversary $\mathcal{A}$, denote by

$$\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^\ell}^{\text{vc}}(\lambda) := \Pr \left[\begin{array}{ll} h^* \leftarrow \mathcal{S}[\ell]; & x \neq x' \wedge h = h^* \wedge \\ (\text{cp}, \text{vp}) \leftarrow \text{Setup}(1^\lambda, \ell); & : \text{wVf}(\text{vp}, h, \text{cm}, x, \pi) = 1 \wedge \\ (h, \text{cm}, x, x', \pi, \pi') \leftarrow \mathcal{A}(\text{cp}, \text{vp}) & \text{wVf}(\text{vp}, h, \text{cm}, x', \pi') = 1 \end{array} \right].$$

Since h^* is sampled independently, it is clear that the binding error is no more than $\ell \cdot \text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^\ell}^{\text{vc}}(\lambda)$, hence it suffices to show that $\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^\ell}^{\text{vc}}(\lambda) \leq \text{negl}(\lambda)$. Towards this, we first define a trapdoor setup algorithm:

TrapSetup($1^\lambda, \ell, h^* \in [\ell], \mathbf{A} \in R_q^{n \times \bar{m}} \rightarrow (\text{cp}, \text{vp}, \mathbf{R}_{h^*})$):

- Sample $\mathbf{R}_{[0]}, \mathbf{R}_{[1]} \leftarrow \mathcal{D}_{R^{\bar{m} \times w}, s''}$ for $s'' = \Theta(\sqrt{d \log(d\bar{m})})$
- $\mathbf{A}_{[0]} := -\hat{h}^* \cdot \mathbf{G}_b - \mathbf{A}\mathbf{R}_{[0]} \in R_q^{n \times w}$, $\mathbf{A}_{[1]} := \mathbf{G}_b - \mathbf{A}\mathbf{R}_{[1]} \in R_q^{n \times w}$
- For all $i \in [\ell]$,

$$\mathbf{R}_i := \mathbf{R}_{[0]} + \hat{i} \cdot \mathbf{R}_{[1]} \in R^{\bar{m} \times w}, \quad \mathbf{A}_i := [\mathbf{A} \mid \mathbf{A}_{[0]} + \hat{i} \cdot \mathbf{A}_{[1]}] \in R_q^{n \times m}, \quad (13)$$

and define $\mathbf{D}_\ell \in R_q^{\ell n \times (\ell m + w)}$ as in Eq. (8)

- Instantiate the sampling oracle $\mathcal{O}_{\mathbf{D}_\ell, s'}^{\text{trap}}(\mathbf{t}) \rightarrow \mathbf{v}$:
 - $\mathbf{v}_{h^*} \leftarrow \mathcal{D}_{R^m, s'}$ and parse $\mathbf{t} = (\mathbf{t}_1, \dots, \mathbf{t}_\ell) \in R_q^{n\ell}$
 - $\mathbf{v}_{\ell+1} \leftarrow \mathcal{S}(\mathbf{G}_b)_{s'}^{-1}(\mathbf{t}_{h^*} - \mathbf{A}_{h^*} \mathbf{v}_{h^*})$
 - For $i \in [\ell], i \neq h^*$, set $\mathbf{v}_i \leftarrow \text{SamplePre}(\mathbf{A}_i, \mathbf{T}_i := \begin{bmatrix} \mathbf{R}_i \\ \mathbf{I}_w \end{bmatrix}, \mathbf{t}_i - \mathbf{G}_b \mathbf{v}_{\ell+1}, s')$
 - Output $\mathbf{v} := (\mathbf{v}_1, \dots, \mathbf{v}_\ell, \mathbf{v}_{\ell+1}) \in R^{\ell m + w}$
- $\mathbf{T}^* \leftarrow \text{DeriveTD}^{\mathcal{O}_{\mathbf{D}_\ell, s'}^{\text{trap}}}(\mathbf{D}_\ell, s')$ where DeriveTD is defined above [Lemma 2.6](#)
- Output $\text{vp} := (\mathbf{A}, \mathbf{A}_{[0]}, \mathbf{A}_{[1]})$, $\text{cp} := (\text{vp}, \mathbf{T}^*)$ and trapdoor $\mathbf{R}_{h^*}$

Let $\Pi_{\text{VC}}^{\text{trap}, \ell}$ denote the vector commitment scheme that replaces **Setup** with **TrapSetup** in [Construction 4.1](#). Define advantage

$$\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^{\text{trap}, \ell}}^{\text{vc}}(\lambda) := \Pr \left[\begin{array}{l} h^* \leftarrow \mathcal{S}[\ell], \mathbf{A} \leftarrow \mathcal{S} R_q^{n \times \bar{m}} \\ (\text{cp}, \text{vp}, \cdot) \leftarrow \text{TrapSetup}(1^\lambda, \ell, h^*, \mathbf{A}); \\ (h, \text{cm}, x, x', \pi, \pi') \leftarrow \mathcal{A}(\text{cp}, \text{vp}) \end{array} : \begin{array}{l} x \neq x' \wedge h = h^* \wedge \\ \text{wVf}(\text{vp}, h, \text{cm}, x, \pi) = 1 \wedge \\ \text{wVf}(\text{vp}, h, \text{cm}, x', \pi') = 1 \end{array} \right].$$

We first show that $\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^{\text{trap}, \ell}}^{\text{vc}}(\lambda)$ is close to $\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^\ell}^{\text{vc}}(\lambda)$.

Claim 7. $|\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^{\text{trap}, \ell}}^{\text{vc}}(\lambda) - \text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^\ell}^{\text{vc}}(\lambda)| \leq \text{negl}(\lambda)$.

Proof. It suffices to show that the parameters (cp, vp) are indistinguishable in the two experiments. In fact, we show that they are statistically close by considering the following hybrid games.

- H_0 : Output $(\mathbf{A}, \mathbf{A}_{[0]}, \mathbf{A}_{[1]}, \mathbf{T}^*)$ that is generated following the **Setup**($1^\lambda, \ell$) algorithm specified in [Construction 4.1](#).
- H_1 : Same as H_0 except that $\mathbf{T}^* \leftarrow \text{DeriveTD}^{(\mathbf{D}_\ell)_{s'}^{-1}}(\mathbf{D}_\ell, s')$.
- H_2 : Same as H_1 except that $\mathbf{A}$ is uniformly random, that is, $\mathbf{A} \leftarrow \mathcal{S} R_q^{n \times \bar{m}}$.

- H_3 : Same as H_2 except that $\mathbf{A}_{[0]}, \mathbf{A}_{[1]}$ are generated using trapdoors. That is, we sample $\mathbf{R}_{[0]}, \mathbf{R}_{[1]} \leftarrow \$ \mathcal{D}_{R^{\bar{m} \times w}, s''}$ and set

$$\mathbf{A}_{[0]} := -\hat{h}^* \cdot \mathbf{G}_b - \mathbf{A}\mathbf{R}_{[0]} \in R_q^{n \times w}, \quad \mathbf{A}_{[1]} := \mathbf{G}_b - \mathbf{A}\mathbf{R}_{[1]} \in R_q^{n \times w}$$

- H_4 : Same as H_3 except that $\mathbf{T}^* \leftarrow \text{DeriveTD}^{\mathcal{O}_{\mathbf{D}_\ell, s'}}(\mathbf{D}_\ell, s')$. Notably, the output is generated following the `TrapSetup` algorithm specified before.

By definition, H_0 and H_4 match the distributions of (cp, vp) in the two experiments, respectively. Thus, it suffices to argue that each adjacent pair of distributions has negligible statistical distance:

- H_0 and H_1 are statistically close by [Lemma 2.4](#) and [Lemma 2.6](#). In particular, since s' is larger than that specified in [Lemma 2.4](#), the output distribution of the oracle $\mathcal{O}_{\mathbf{D}_\ell, s'}(\cdot)$ in [Construction 4.1](#) is statistically close to that of $(\mathbf{D}_\ell)_{s'}^{-1}(\cdot)$. Then by [Lemma 2.6](#), the distributions of trapdoor $\mathbf{T}^*$ in H_0 and H_1 are statistically close, and $s_1(\mathbf{T}^*) \leq s'\sqrt{d} \cdot O(\sqrt{\ell m})$ with overwhelming probability.
- H_1 and H_2 are statistically close by the trapdoor distribution property of [Theorem 2.2](#).
- H_2 and H_3 are statistically close by the smoothing lemma ([Lemma 2.5](#)). In particular, as $\bar{m} \geq 2n \log_b(q)$ and s'' is larger than that in [Lemma 2.5](#), by a straightforward hybrid argument, $\mathbf{A}_{[0]}, \mathbf{A}_{[1]}$ are close to uniformly random over $R_q^{n \times w}$.
- H_3 and H_4 are statistically close by [Lemma 2.4](#) and [Lemma 2.6](#). In particular, since s' is larger than that specified in [Lemma 2.4](#), the output distribution of the oracle $\mathcal{O}_{\mathbf{D}_\ell, s'}^{\text{trap}}(\cdot)$ is statistically close to that of $(\mathbf{D}_\ell)_{s'}^{-1}(\cdot)$. Then by [Lemma 2.6](#), the distributions of trapdoor $\mathbf{T}^*$ in H_3 and H_4 are statistically close, and $s_1(\mathbf{T}^*) \leq s'\sqrt{d} \cdot O(\sqrt{\ell m})$ with overwhelming probability.

In summary, the distributions of (cp, vp) in H_0 and H_4 are statistically close, and $s_1(\mathbf{T}^*)$ is small enough to support pre-image sampling for width parameter $s \geq \omega(\sqrt{\log(dn)}) \cdot s_1(\mathbf{T}^*) \cdot \sqrt{1+b^2}$ as in [Theorem 2.2](#). This finishes the proof. $\square$

Claim 8. Assume that $\text{MSIS}_{n-n_m, \bar{m}, q, B_{\text{sis}}}$ is hard, then for all $\lambda \in \mathbb{N}$, $\ell = \ell(\lambda)$, and PPT adversary $\mathcal{A}$, it holds that $\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^{\text{trap}, \ell}}^{\text{VC}}(\lambda) \leq \text{negl}(\lambda)$.

Proof. Fix adversary $\mathcal{A}$, we construct an MSIS attacker $\mathcal{B}$ as follows:

- On input $\underline{\mathbf{A}} \in R_q^{(n-n_m) \times \bar{m}}$, sample $\bar{\mathbf{A}} \leftarrow \$ R_q^{n_m \times \bar{m}}$ and set $\mathbf{A} := \begin{bmatrix} \bar{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix} \in R_q^{n \times \bar{m}}$.

- Sample $h^* \leftarrow_{\$} [\ell]$, run $(\text{cp}, \text{vp}, \mathbf{R}_{h^*}) \leftarrow \text{TrapSetup}(1^\lambda, \ell, h^*, \mathbf{A})$ and forward (cp, vp) to adversary $\mathcal{A}$.
- If $\mathcal{A}(\text{cp}, \text{vp})$ outputs $(h, \text{cm}, \mathbf{x}_h, \mathbf{x}'_h, \pi_h, \pi'_h)$ such that $x \neq x'$, $h = h^*$ and

$$\text{wVf}(\text{vp}, h, \text{cm}, \mathbf{x}_h, \pi_h) = \text{wVf}(\text{vp}, h, \text{cm}, \mathbf{x}'_h, \pi'_h) = 1, \quad (14)$$

then parse $\pi_h - \pi'_h = (\tau_1, \tau_2)$ where $\tau_1 \in R^{\bar{m}}$ and $\tau_2 \in R^w$, and output

$$\tau^* := \tau_1 - \mathbf{R}_{h^*} \tau_2 \in R^{\bar{m}}.$$

We argue that $\mathcal{B}$ outputs a valid MSIS solution $\tau^* \in R^m$ for $\underline{\mathbf{A}} \in R_q^{(n-n_m) \times \bar{m}}$ whenever $\mathcal{A}$ succeeds. In particular, by construction of wVf , if Eq. (14) holds, we have that

$$\mathbf{A}_h \pi_h = \text{cm} + (\mathbf{x}_h, \mathbf{0}^{n-n_m}), \quad \mathbf{A}_h \pi'_h = \text{cm} + (\mathbf{x}'_h, \mathbf{0}^{n-n_m}),$$

which implies that

$$\mathbf{A}_h(\pi_h - \pi'_h) = (\mathbf{x}_h - \mathbf{x}'_h, \mathbf{0}^{n-n_m}).$$

Recall that $\pi_h - \pi'_h = (\tau_1, \tau_2)$ and

$$\begin{aligned} \mathbf{A}_h &= [\mathbf{A} \mid \mathbf{A}_{[0]} + \hat{h}^* \cdot \mathbf{A}_{[1]}] \\ &= [\mathbf{A} \mid (-\hat{h}^* \cdot \mathbf{G}_b - \mathbf{A}\mathbf{R}_{[0]}) + \hat{h}^* \cdot (\mathbf{G}_b - \mathbf{A}\mathbf{R}_{[1]})] \\ &= [\mathbf{A} \mid -\mathbf{A}(\mathbf{R}_{[0]} + \hat{h}^* \cdot \mathbf{R}_{[1]})] \\ &= [\mathbf{A} \mid -\mathbf{A}\mathbf{R}_{h^*}] \end{aligned}$$

where the 2nd and the last equality follows by the specification of $\mathbf{A}_{[0]}$, $\mathbf{A}_{[1]}$, $\mathbf{R}_{h^*}$ in TrapSetup . Thus,

$$\begin{aligned} [\mathbf{A} \mid -\mathbf{A}\mathbf{R}_{h^*}](\tau_1, \tau_2) &= \mathbf{A}\tau_1 - \mathbf{A}\mathbf{R}_{h^*}\tau_2 \\ &= \mathbf{A}(\tau_1 - \mathbf{R}_{h^*}\tau_2) \\ &= \mathbf{A}\tau^* = (\mathbf{x}_h - \mathbf{x}'_h, \mathbf{0}^{n-n_m}). \end{aligned}$$

Since $\mathbf{x}_h \neq \mathbf{x}'_h$ and $\mathbf{A} := \begin{bmatrix} \bar{\mathbf{A}} \\ \underline{\mathbf{A}} \end{bmatrix}$, this implies that $\tau^* \neq \mathbf{0}$ and $\underline{\mathbf{A}}\tau^* = \mathbf{0}^{n-n_m}$.

It remains to argue that $\|\tau^*\| \leq B_{\text{sis}}$. Recall that $\|(\tau_1, \tau_2)\| = \|\pi_h - \pi'_h\| \leq 2B_w$. Thus,

$$\|\tau^*\| = \|\tau_1 - \mathbf{R}_{h^*}\tau_2\| \leq \|(\tau_1, \tau_2)\| \cdot (1 + s_1(\mathbf{R}_{h^*})) \leq 2B_w \cdot (1 + s_1(\mathbf{R}_{h^*})) \leq B_{\text{sis}},$$

where the last inequality holds by definition of B_{sis} and because $(1 + s_1(\mathbf{R}_{h^*})) \leq \Theta(\sqrt{dw \log \ell})$. s'' and $s'' = \Theta(\sqrt{d \log(d\bar{m})})$. This finishes the proof. $\square$

By the above claims, $\text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^\ell}^{\text{vc}}(\lambda) \leq \text{Adv}_{\mathcal{A}, \Pi_{\text{VC}}^{\text{trap}, \ell}}^{\text{vc}}(\lambda) + \text{negl}(\lambda) \leq \text{negl}(\lambda)$, finishing the proof of [Lemma 4.8](#). $\square$

5 Key Homomorphic Synchronous Signature Schemes

In this section, we define key homomorphic synchronous signatures and provide a generic construction from statistically sampleable homomorphic vector commitments. Throughout the section we fix the ring $R := \mathbb{Z}[X]/(X^d + 1)$ and set $R_q := R/qR$ for a prime q .

5.1 Definition

In this section we define the notion of a key homomorphic synchronous signature. Our notion is a generalization of key homomorphic one-time signatures defined in [FSZ22]. The signature supports up to ℓ slots, and (multi-instance) existential unforgeability is maintained as long as every slot is used to sign at most w messages. In real-world usage we expect ℓ to be in the millions, while w is small, say $w \leq 4$, or perhaps even $w = 1$.

Definition 5.1. *A **Key Homomorphic Synchronous Signature Scheme** Sig consists of a tuple of PPT algorithms:*

- $\text{Setup}(1^\lambda, \ell, w) \rightarrow \text{pp}$: for positive integers λ, ℓ, w , outputs public parameters pp for a w -way ℓ -slot scheme.
- $\text{KGen}(1^\lambda, \text{pp}) \rightarrow (\text{pk}, \text{sk})$: generates a key pair where $\text{sk} \in \mathcal{K}$ for a w -way ℓ -slot synchronous scheme.
- $\text{Sign}(1^\lambda, \text{sk}, m, h) \rightarrow \sigma$: signs $m \in \mathcal{M}$ at slot $h \in [\ell]$ and outputs a signature σ . In what follows we require that the **Sign** algorithm is deterministic.
- $\text{Vf}(1^\lambda, \text{pp}, \text{pk}, m, h, \sigma) \rightarrow \{0, 1\}$: verifies a signature σ on m with respect to slot $h \in [\ell]$.
- $\text{wVf}(1^\lambda, \text{pp}, \text{pk}, m, h, \sigma) \rightarrow \{0, 1\}$: same as $\text{Vf}(\cdot)$ except that it may accept a wider range of signatures. wVf is only used in our proof of security.

We often drop the input 1^λ from algorithms $\text{KGen}, \text{Sign}, \text{Vf}, \text{wVf}$ and leave it as an implicit input. These algorithms must satisfy the properties defined below.

Definition 5.2 (Multi-Instance Unforgeability). *A synchronous signature scheme Sig is **multi-instance unforgeable** if for all λ , $\kappa = \kappa(\lambda)$, and PPT adversaries $\mathcal{A}$,*

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, \kappa}(\lambda) := \Pr[\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, \kappa}(\lambda) = 1] \leq \text{negl}(\lambda).$$

Moreover, for set $\mathcal{W} \subseteq R$, Sig is **$\mathcal{W}$ -multi-instance unforgeable** if for all λ , $\kappa = \kappa(\lambda)$, and PPT adversaries $\mathcal{A}$,

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{augssig}, \kappa, \mathcal{W}}(\lambda) := \Pr[\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{augssig}, \kappa, \mathcal{W}}(\lambda) = 1] \leq \text{negl}(\lambda).$$

The security games $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, \kappa}(\lambda)$ and $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{augssig}, \kappa, \mathcal{W}}(\lambda)$ are defined in [Figure 3](#).

The Games $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, \kappa}(\lambda)$ and $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{augssig}, \kappa, \mathcal{W}}(\lambda)$	Oracle $\text{OSign}(i, m, h)$
1 : $(\ell, w, \text{state}) \leftarrow \mathcal{A}(1^\lambda)$	1 : if $h \notin [\ell]$: return $\perp$
2 : $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$	2 : if $i \notin [\kappa]$: return $\perp$
3 : $\forall i \in [\kappa] : (\text{pk}^i, \text{sk}^i) \leftarrow \text{KGen}(1^\lambda, \text{pp})$	3 : $S[i][h] \leftarrow S[i][h] \cup \{m\}$
4 : $(\text{aux}, i^*, m, h, \sigma) \leftarrow \mathcal{A}^{\text{OSign}(\cdot, \cdot, \cdot)}(1^\lambda, (\text{pk}^i)_{i \in [\kappa]}, \text{state})$	4 : return $\text{Sign}(1^\lambda, \text{sk}^i, m, h)$
5 : if $ S[i^*][h] > w$ OR $m \in S[i^*][h]$: return 0	
6 : if $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, \kappa}$ AND $\text{aux} \neq 1$: return 0	
7 : if $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{augssig}, \kappa, \mathcal{W}}$ AND $\text{aux} \notin \mathcal{W}$: return 0	
8 : return $\text{wVf}(\text{pp}, \text{aux} \cdot \text{pk}^{i^*}, m, h, \sigma)$	

Figure 3: The multi-instance EUF-CMA security game for an (augmented) synchronous signature scheme Sig , where $\text{Sig} = (\text{Setup}, \text{KGen}, \text{Sign}, \text{Vf}, \text{wVf})$. The adversary fails if it makes more than w queries at the slot where it chooses to forge.

Remark 5.1. If the adversary makes more than w queries at certain slots, then it may be able to produce a forgery at those slots. Even then, [Definition 5.2](#) requires that it cannot produce forgeries at slots where it made at most w queries.

To support thresholdizing signatures, we require the following homomorphic signing correctness property:

Definition 5.3 (Homomorphic Signing Correctness). *Fix $k \in \mathbb{N}$, and let $\mathcal{K} = R^v$ for some v . The scheme Sig satisfies homomorphic signing correctness, if for all $\lambda, \ell, w \in \mathbb{N}$, messages $m \in \mathcal{M}$, $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$, $(\text{pk}, \text{sk}) \leftarrow \text{KGen}(\text{pp})$, and any $\text{sk}^1, \dots, \text{sk}^k \in \mathcal{K}$, scalars $\alpha^1, \dots, \alpha^k \in R$:*

$$\left(\text{sk} = \sum_{i \in [k]} \alpha^i \cdot \text{sk}^i \right) \implies \Pr \left[\text{Vf} \left(\text{pp}, \text{pk}, m, h, \sum_{i \in [k]} \alpha^i \cdot \text{Sign}(\text{sk}^i, m, h) \right) = 1 \right] = 1. \quad (15)$$

To support aggregating signatures, we further require that there exist efficient algorithms DerivePK , $\overline{\text{Vf}}$, and $\overline{\text{wVf}}$ that satisfy the following properties.

Definition 5.4 (Strong Public Key Homomorphic Correctness). *Fix $k \in \mathbb{N}$ and $\mathcal{S} \subseteq R$ where $1 \in \mathcal{S}$. The scheme Sig satisfies $(k, \mathcal{S})$ -strong public key homomorphic correctness, if for all $\lambda, \ell, w \in \mathbb{N}$, ring elements $\alpha^1, \dots, \alpha^k \in \mathcal{S}$, index $h \in [\ell]$, and messages $m_1, \dots, m_k \in \mathcal{M}$:*

$\mathcal{M}$,

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w); \\ \forall i \in [k], \quad (\text{pk}^i, \text{sk}^i) \leftarrow \text{KGen}(\text{pp}); \\ \forall i \in [k], \quad \text{pk}_{m_i, h}^i \leftarrow \text{DerivePK}(\text{pk}^i, m_i, h); \\ \forall i \in [k], \quad \sigma_h^i \leftarrow \text{Sign}(\text{sk}^i, m_i, h) \end{array} : \overline{\text{Vf}}(\text{pp}, \sum_{i \in [k']} \alpha^i \cdot \text{pk}_{m_i, h}^i, h, \sum_{i \in [k']} \alpha^i \cdot \sigma_h^i) = 1 \right] = 1. \quad (16)$$

Definition 5.5 ($\mathcal{W}$ -Weak Verification Consistency). *For set $\mathcal{W} \subseteq R$, the scheme is Sig satisfies $\mathcal{W}$ -weak verification consistency if for all $\lambda, \ell, w \in \mathbb{N}$, $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$, and all $\text{aux} \in \mathcal{W}$, public key pk , message m , index h , signature σ :*

$$\overline{\text{Vf}}(\text{pp}, \text{aux} \cdot \text{DerivePK}(\text{pk}, m, h), h, \sigma) = 1 \implies \text{wVf}(\text{pp}, \text{aux} \cdot \text{pk}, m, h, \sigma) = 1.$$

Definition 5.6 (Strong Robust Homomorphism). *The scheme Sig satisfies strong robust homomorphism if for all $\lambda, \ell, w \in \mathbb{N}$, $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$, public keys pk, pk' , index $h \in [\ell]$, signatures σ, σ' , it holds that*

$$(\overline{\text{Vf}}(\text{pp}, \text{pk}, h, \sigma) = \overline{\text{Vf}}(\text{pp}, \text{pk}', h, \sigma') = 1) \implies \overline{\text{wVf}}(\text{pp}, \text{pk} - \text{pk}', h, \sigma - \sigma') = 1.$$

5.2 A Generic Construction from Statistically Sampleable HVC

We present a generic construction of key homomorphic synchronous signatures based on statistically sampleable homomorphic vector commitments. We refer to Figure 17 for the relationship between the properties of our generic constructions.

Linear independent hash functions. First, we need a hash function for which it is infeasible to find $w + 1$ distinct hash inputs, whose hashes are linearly dependent over R_q . In Lemma 5.1 we show that a hash function with this property can be constructed from a collision resistant hash function.

Definition 5.7. *For $w \in \mathbb{N}$, we say a family of hash functions $\mathcal{H}_w = \{\text{H} : \mathcal{X} \rightarrow R^{w+1}\}$ is w -linear-independent over R_q if for every PPT adversary $\mathcal{A}$,*

$$\Pr \left[\begin{array}{l} \text{H} \leftarrow \$ \mathcal{H}_w \\ (x_1, \dots, x_{w+1}) \leftarrow \$ \mathcal{A}(\text{H}) \\ \forall i \in [w+1] : \mathbf{v}_i \leftarrow \text{H}(x_i) \bmod q \end{array} : \begin{array}{l} \dim(\text{span}_{R_q}(\mathbf{v}_1, \dots, \mathbf{v}_{w+1})) < w+1 \\ \wedge \forall i \neq j \in [w+1] : x_i \neq x_j \end{array} \right] \leq \text{negl}(\lambda).$$

Lemma 5.1. *Let $\mathcal{H} = \{h : \mathcal{X} \rightarrow R\}$ be a family of collision resistant hash functions. For every $w \in \mathbb{N}$, there exists a family of hash functions $\mathcal{H}_w = \{\text{H} : \mathcal{X} \rightarrow R^{w+1}\}$ that is w -linear-independent over R_q .*

Proof. The hash function H is defined as

$$H(x) := \left(1, h(x), h(x)^2, \dots, h(x)^w\right) \in R^{w+1}. \quad (17)$$

Suppose for contradiction that an adversary outputs $w + 1$ *distinct* inputs $(x_1, \dots, x_{w+1})$ whose corresponding hash outputs $\mathbf{v}_1, \dots, \mathbf{v}_{w+1}$ are linearly dependent. Since a square Vandermonde matrix with distinct columns is full rank, it must be that $\mathbf{v}_i = \mathbf{v}_j$ for some $i, j \in [w + 1]$, $i \neq j$. This leads to a hash collision, contradicting the collision resistance of h . Therefore, $\mathcal{H}_w$ is w -linear-independent over R_q . $\square$

Remark 5.2. In practice we expect w to be small, say $0 < w \leq 4$ so that the norm of $H(x)$ is low whenever the norm of $h(x)$ is low. But for large w , even low-norm outputs of h may lead to high-norm outputs of H due to the exponentiations. In this case, $\mathcal{H}_w$ can instead be instantiated directly using a standard hash function that outputs low norm vectors. If the hash function is modeled as a random map, then it still guarantees w -linear-independence with high probability.

The signature scheme. Our w -way ℓ -slot synchronous signature scheme with message space $\mathcal{M}$ is presented in Figure 4 as Construction 5.1. Fix sets $\mathcal{S}, \mathcal{W} \subseteq R$ where $1 \in \mathcal{S}$. Moreover, $\mathcal{S}$ is a (w, q) -strong sampling set (Definition 2.1), and every element in $\mathcal{W}$ is invertible over R_q . The construction uses two primitives:

- $\mathcal{H}_w \subseteq \{h : \mathcal{M} \times [\ell] \rightarrow \mathcal{S}^{w+1}\}$ is a family of w -linear-independent hash functions (as in Definition 5.7).
- Π_{VC} is a homomorphic vector commitment (as in Definition 3.1) with $(k(w+1), \mathcal{S}^*, \mathcal{O}_{\text{VC}})$ -homomorphic correctness for valid openings, where $\mathcal{S}^* := \{xy : x, y \in \mathcal{S}\}$ and $\mathcal{O}_{\text{VC}}$ is the set of possible openings that could be output by $\Pi_{\text{VC}}.\text{Open}$ given an opening state st generated from $\Pi_{\text{VC}}.\text{Com}$. Moreover, Π_{VC} satisfies $\mathcal{W}$ -weak verifiability, linearity of opening states, and $(w + 1, \mathcal{S})$ -statistical sampleability (as in Definition 3.8). The message space is R_q , the commitment space is R_q^n , and the opening state space is $\mathcal{Z}$.

We present Construction 5.1 in Figure 4. Roughly, the public key consists of the commitments to $w + 1$ independently sampled random vectors $\mathbf{x}^1, \dots, \mathbf{x}^{w+1}$; the secret key consists of the secret vectors $\mathbf{x}^1, \dots, \mathbf{x}^{w+1}$ and the corresponding openings. A signature for message m and slot h is a linear combination of the secret values $\mathbf{x}_h^1, \dots, \mathbf{x}_h^{w+1}$ and their openings, where the linear combiner is derived via the w -linear-independent hash function on input (m, h) . Informally, signature verification checks that the linear combination of the commitments (in the public key) is consistent with the same linear combination of the secret values and their openings.

The public key contains $w + 1$ elements in R_q^n , so that, for example, when $w = 1$ the public key contains only two R_q^n elements. The size of the secret key is linear in the number

of slots ℓ , but only a constant portion of the key is used per signature. That portion can be re-computed from a short seed, in which case the secret key is only the short seed. The signature is an element in R_q and a low norm element in $\mathcal{O}$.

We stress that the commitment parameter $\mathbf{cp}$ is only used during key generation, and need not be stored after. Moreover, while the number of slots ℓ is fixed during setup, the per-slot signature arity w can be chosen at key generation time, and is not fixed for all keys.

Security proofs. Before formally proving the properties of the construction, we sketch the proof idea of unforgeability. By the value guessing lemma (Lemma 2.9) and the statistical sampleability of Π_{VC} , with overwhelming probability, no adversary (even an unbounded one) can produce a forgery (z_h, τ_h) for a pair (m, h) such that z_h equals the linear combination of the secret values $\mathbf{x}_h^1, \dots, \mathbf{x}_h^{w+1}$, where the linear combiners are given by the hash of (m, h) . Conditioned on this, (z_h, τ_h) can be used to break the position binding property of the underlying vector commitment scheme: the reduction simply opens the position h with both values z_h and the linear combination of $\mathbf{x}_h^1, \dots, \mathbf{x}_h^{w+1}$.

Next, we formally prove that Construction 5.1 satisfies the properties specified in Definition 5.2, Definition 5.3, Definition 5.4, Definition 5.5, and Definition 5.6.

Theorem 5.2. *Construction 5.1 satisfies k -homomorphic signing correctness (Definition 5.3) if Π_{VC} satisfies linearity of opening states and $(w+1, \mathcal{S}, \mathcal{O}_{\text{VC}})$ -homomorphic correctness for valid openings.*

Proof. Fix $\lambda, \ell, w \in \mathbb{N}$, and message $m \in \mathcal{M}$. Let $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$, $(\text{pk}, \text{sk}) \leftarrow \text{KGen}(\text{pp})$ where $\text{pk} := (\text{H}, \{\text{cm}^j\}_{j \in [w+1]})$ and $\text{sk} = (\text{H}, \{(\text{st}^j, \mathbf{x}^j)\}_{j \in [w+1]})$. Fix any $\beta^1, \dots, \beta^k \in R_q$ and $\text{sk}^1, \dots, \text{sk}^k \in \mathcal{K}$ such that $\text{H}^1 = \dots = \text{H}^k$ and $\text{sk} = \sum_{i \in [k]} \beta^i \cdot \text{sk}^i$. For every $\text{sk}^i \in \mathcal{K}$, we parse

$$\text{sk}^i = (\text{H}^i, \{(\text{st}^{i,j}, \mathbf{x}^{i,j})\}_{j \in [w+1]})$$

and define

$$\sum_{i \in [k]} \beta^i \cdot \text{sk}^i = \left(\text{H}^1, \left\{ \sum_{i \in [k]} \beta^i \cdot \text{st}^{i,j}, \sum_{i \in [k]} \beta^i \cdot \mathbf{x}^{i,j} \right\}_{j \in [w+1]} \right).$$

Since $\text{sk}^1, \dots, \text{sk}^{k'}$ satisfy that $\text{H}^1 = \dots = \text{H}^{k'}$ and $\text{sk} = \sum_{i \in [k']} \beta^i \cdot \text{sk}^i$, then for all $j \in [w+1]$, $\text{st}^j = \sum_{i \in [k']} \beta^i \cdot \text{st}^{i,j}$.

By the linearity of opening states of Π_{VC} (Definition 3.4),

$$\sum_{i \in [k']} \beta^i \cdot \Pi_{\text{VC}}.\text{Open}(\text{st}^{i,j}, h) = \Pi_{\text{VC}}.\text{Open}(\text{st}^j, h) \in \mathcal{O}_{\text{VC}}. \quad (18)$$

Construction 5.1.

- $\text{Setup}(1^\lambda, \ell, w) \rightarrow \text{pp}$: $\quad \parallel \quad \ell$ slots and w -time security per slot
 - $(\text{cp}, \text{vp}) \leftarrow \Pi_{\text{VC}}.\text{Setup}(1^\lambda, \ell)$ and output $\text{pp} := (w, \text{cp}, \text{vp})$
- $\text{KGen}(1^\lambda, \text{pp}) \rightarrow (\text{pk}, \text{sk})$:
 - $\text{H} \leftarrow \mathcal{H}_w \subseteq \{h : \mathcal{M} \times [\ell] \rightarrow \mathcal{S}^{w+1}\}$
 - For $j \in [w+1]$ do: $\quad \parallel \quad$ Commit to $w+1$ random vectors in R_q^ℓ
 $\mathbf{x}^j \leftarrow R_q^\ell, \quad (\text{cm}^j, \text{st}^j) \leftarrow \Pi_{\text{VC}}.\text{Com}(\text{cp}, \mathbf{x}^j)$
 - Output $\text{pk} := (\text{H}, \{\text{cm}^j\}_{j \in [w+1]})$, $\text{sk} := (\text{H}, \{\text{st}^j, \mathbf{x}^j\}_{j \in [w+1]})$
- $\text{Sign}(1^\lambda, \text{sk}, m, h \in [\ell]) \rightarrow \sigma_h$:
 - Parse $\text{sk} := (\text{H}, \{\text{st}^j, \mathbf{x}^j\}_{j \in [w+1]})$
 $\quad \parallel \quad$ build $w+1$ opening proofs for slot h
 - For $j \in [w+1]$ do: $\pi_h^j \leftarrow \Pi_{\text{VC}}.\text{Open}(\text{st}^j, h)$
 - $\alpha \leftarrow \text{H}(m, h) \in \mathcal{S}^{w+1}$
 $\quad \parallel \quad$ signature is a linear combination of $(\mathbf{x}_h^j, \pi_h^j) \in R_q \times \mathcal{O}$
 - $(z_h, \tau_h) \leftarrow \sum_{j=1}^{w+1} \alpha_j \cdot (\mathbf{x}_h^j, \pi_h^j)$
 - Output $\sigma_h := (z_h, \tau_h) \in R_q \times \mathcal{O}$
- $\text{Vf}(\text{pp}, \text{pk}, m, h, \sigma) \rightarrow \{0, 1\}$:
 - $\text{pk}_{m,h} \leftarrow \text{DerivePK}(\text{pk}, m, h) \quad \parallel \quad \text{DerivePK is defined next}$
 - Output $\overline{\text{Vf}}(\text{pp}, \text{pk}_{m,h}, h, \sigma) \quad \parallel \quad \overline{\text{Vf}} \text{ is defined below}$
- $\text{DerivePK}(\text{pk}, m, h) \rightarrow \text{pk}_{m,h}$:
 - Parse $\text{pk} := (\text{H}, \{\text{cm}^j\}_{j \in [w+1]})$
 - $\alpha \leftarrow \text{H}(m, h) \in \mathcal{S}^{w+1}$
 - Output $\text{pk}_{m,h} := \sum_{j \in [w+1]} \alpha_j \cdot \text{cm}^j$
- $\overline{\text{Vf}}(\text{pp}, \text{pk}, h, \sigma) \rightarrow \{0, 1\}$:
 - Parse $\text{pp} = (\cdot, \text{vp})$ and $\text{pk} = \text{cm}^*$ and $\sigma = (z, \tau) \in R_q \times \mathcal{O}$
 - Output $\Pi_{\text{VC}}.\text{Vf}(\text{vp}, h, \text{cm}^*, z, \tau)$
- $\overline{\text{wVf}}(\text{pp}, \text{pk}, h, \sigma) \rightarrow \{0, 1\}$: Same as $\overline{\text{Vf}}$ but call $\Pi_{\text{VC}}.\text{wVf}()$ instead of $\Pi_{\text{VC}}.\text{Vf}()$
- $\text{wVf}(\text{pp}, \text{pk}, m, h, \sigma) \rightarrow \{0, 1\}$: Same as Vf but call $\overline{\text{wVf}}()$ instead of $\overline{\text{Vf}}()$

Figure 4: The basic w -way ℓ -slot synchronous signature scheme $\Pi_{\text{sync}}^{w,\ell}$.

Let $\alpha \leftarrow H(m, h)$. Then:

$$\begin{aligned}
\sigma &:= \sum_{i=1}^{k'} \beta^i \cdot \text{Sign}(\text{sk}^i, m, h) = \sum_{i=1}^{k'} \beta^i \cdot \sum_{j=1}^{w+1} \alpha_j \cdot (\mathbf{x}_h^{i,j}, \Pi_{\text{VC}}.\text{Open}(\text{st}^{i,j}, h)) \\
&= \sum_{j=1}^{w+1} \alpha_j \left(\sum_{i=1}^{k'} \beta^i \cdot \mathbf{x}_h^{i,j}, \sum_{i=1}^{k'} \beta^i \cdot \Pi_{\text{VC}}.\text{Open}(\text{st}^{i,j}, h) \right) \\
&= \sum_{j=1}^{w+1} \alpha_j \left(\mathbf{x}_h^j, \sum_{i=1}^{k'} \beta^i \cdot \Pi_{\text{VC}}.\text{Open}(\text{st}^{i,j}, h) \right) \\
&= \sum_{j=1}^{w+1} \alpha_j \left(\mathbf{x}_h^j, \Pi_{\text{VC}}.\text{Open}(\text{st}^j, h) \right)
\end{aligned}$$

Parse $\sigma = (z, \tau)$, $\text{pp} = (\cdot, \text{vp})$ and $\text{pk} = (H, \{\text{cm}^j\}_{j \in [w+1]})$. By construction of DerivePK , the derived public key $\text{pk}_{m,h} = \sum_{j \in [w+1]} \alpha_j \cdot \text{cm}^j$. Then:

$$\begin{aligned}
\text{Vf}(\text{pp}, \text{pk}, m, h, \sigma) &= \overline{\text{Vf}}(\text{pp}, \text{pk}_{m,h}, h, \sigma) \\
&= \Pi_{\text{VC}}.\text{Vf}(\text{vp}, h, \text{pk}_{m,h}, z, \tau) \\
&= \Pi_{\text{VC}}.\text{Vf} \left(\text{vp}, h, \sum_{j=1}^{w+1} \alpha_j \cdot \text{cm}^j, \sum_{j=1}^{w+1} \alpha_j \cdot \mathbf{x}_h^j, \sum_{j=1}^{w+1} \alpha_j \Pi_{\text{VC}}.\text{Open}(\text{st}^j, h) \right) \\
&= \Pi_{\text{VC}}.\text{Vf} \left(\text{vp}, h, \sum_{j \in [w+1]} \alpha_j \cdot \left(\text{cm}^j, \mathbf{x}_h^j, \Pi_{\text{VC}}.\text{Open}(\text{st}^j, h) \right) \right)
\end{aligned}$$

By the $(w+1, \mathcal{S}, \mathcal{O}_{\text{VC}})$ -homomorphic correctness for valid openings of Π_{VC} ([Definition 3.3](#)), $\text{Vf}(\text{pp}, \text{pk}_{m,h}, m, h, \sigma)$ returns 1. $\square$

Theorem 5.3. *Construction 5.1 satisfies $(k, \mathcal{S})$ -strong public key homomorphic correctness ([Definition 5.4](#)) if Π_{VC} satisfies correctness and $(k(w+1), \mathcal{S}^*, \mathcal{O}_{\text{VC}})$ -homomorphic correctness for valid openings.*

Proof. Fix $\lambda, \ell, w \in \mathbb{N}$, $k' \in [k]$, values $\beta^1, \dots, \beta^{k'} \in \mathcal{S}$, index $h \in [\ell]$, and messages $m_1, \dots, m_{k'} \in \mathcal{M}$, let $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$, and for all $i \in [k']$, let $(\text{pk}^i, \text{sk}^i) \leftarrow \text{KGen}(\text{pp})$, where

$$\text{sk}^i = (H^i, \{(\text{st}^{i,j}, \mathbf{x}_h^{i,j})\}_{j \in [w+1]}),$$

let $\text{pk}_{m^i,h}^i \leftarrow \text{DerivePK}(\text{pk}^i, m^i, h)$ and $\sigma_h^i \leftarrow \text{Sign}(\text{sk}^i, m_i, h)$. We want to show that $\overline{\text{Vf}}(\text{pp}, \sum_{i \in [k']} \beta^i \cdot \text{pk}_{m^i,h}^i, h, \sum_{i \in [k']} \beta^i \cdot \sigma_h^i) = 1$.

Parse $\text{pp} = (\cdot, \text{vp})$ and $\text{pk}^i = (\text{H}^i, \{\text{cm}_i^j\}_{j \in [w+1]})$. By construction of DerivePK , the derived public key $\text{pk}_{m^i, h}^i$ satisfies that

$$\text{pk}_{m^i, h}^i = \sum_{j \in [w+1]} \alpha_j^i \cdot \text{cm}_i^j$$

where $\alpha^i = \text{H}^i(m^i, h)$. Thus,

$$\text{cm}^* := \sum_{i \in [k']} \beta^i \cdot \text{pk}_{m^i, h}^i = \sum_{i \in [k'], j \in [w+1]} \beta^i \cdot \alpha_j^i \cdot \text{cm}_i^j,$$

where $\beta^i \cdot \alpha_j^i \in \mathcal{S}^*$.

Similarly, by construction of Sign , the signature $\sigma_h^i = (z_h^i, \tau_h^i)$ satisfies that

$$(z_h^i, \tau_h^i) = \sum_{j=1}^{w+1} \alpha_j^i \cdot (\mathbf{x}_h^{i,j}, \pi_h^{i,j})$$

where $\pi_h^{i,j} \leftarrow \Pi_{\text{VC}}.\text{Open}(\text{st}^{i,j}, h)$. Thus,

$$(\mathbf{x}_h^*, \pi_h^*) := \sum_{i \in [k']} \beta^i \cdot \sigma_h^i = \sum_{i \in [k'], j \in [w+1]} \beta^i \cdot \alpha_j^i \cdot (\mathbf{x}_h^{i,j}, \pi_h^{i,j}),$$

where $\beta^i \cdot \alpha_j^i \in \mathcal{S}^*$.

Note that $\pi_h^{i,j} \leftarrow \Pi_{\text{VC}}.\text{Open}(\text{st}^{i,j}, h)$ are valid openings for $(\text{cm}_i^j, \mathbf{x}_h^{i,j})$ by the correctness (Definition 3.2) of Π_{VC} , and $\pi_h^{i,j} \in \mathcal{O}_{\text{VC}}$ by definition of $\mathcal{O}_{\text{VC}}$. Then by the $(k(w+1), \mathcal{S}^*, \mathcal{O}_{\text{VC}})$ -homomorphic correctness for valid openings of Π_{VC} (Definition 3.3), it holds that

$$\Pi_{\text{VC}}.\text{Vf}(\text{vp}, h, \text{cm}^*, \mathbf{x}_h^*, \pi_h^*) = 1,$$

which implies that $\overline{\text{Vf}}(\text{pp}, \sum_{i \in [k']} \beta^i \cdot \text{pk}_{m^i, h}^i, h, \sum_{i \in [k']} \beta^i \cdot \sigma_h^i) = 1$ by construction of $\overline{\text{Vf}}$. $\square$

Theorem 5.4. *Construction 5.1 satisfies $\mathcal{W}$ -weak verification consistency (Definition 5.5).*

Proof. Fix $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$, $\text{aux} \in \mathcal{W}$, public key pk , message m , index h , signature σ , let $\text{pk}_{m, h} \leftarrow \text{DerivePK}(\text{pk}, m, h)$. By construction of wVf , the output of $\text{wVf}(\text{pp}, \text{aux} \cdot \text{pk}, m, h, \sigma)$ is exactly $\overline{\text{wVf}}(\text{pp}, \text{DerivePK}(\text{aux} \cdot \text{pk}, m, h), h, \sigma)$ where

$$\text{DerivePK}(\text{aux} \cdot \text{pk}, m, h) = \text{aux} \cdot \text{DerivePK}(\text{pk}, m, h)$$

by linearity of DerivePK . $\square$

Theorem 5.5. *Construction 5.1 satisfies strong robust homomorphism (Definition 5.6).*

Proof. Fix $\lambda, \ell, w \in \mathbb{N}$, $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w)$, public keys pk, pk' , index $h \in [\ell]$, signatures σ, σ' such that

$$\overline{\text{Vf}}(\text{pp}, \text{pk}, h, \sigma) = \overline{\text{Vf}}(\text{pp}, \text{pk}', h, \sigma') = 1.$$

Parse $\text{pp} = (\cdot, \text{vp})$, $\sigma = (z, \tau)$, $\sigma' = (z', \tau')$. Then by construction of $\overline{\text{Vf}}$, it holds that

$$\Pi_{\text{VC}}.\text{Vf}(\text{vp}, h, \text{pk}, z, \tau) = \Pi_{\text{VC}}.\text{Vf}(\text{vp}, h, \text{pk}', z', \tau') = 1.$$

By robust homomorphism of Π_{VC} (Definition 3.5), it holds that

$$\Pi_{\text{VC}}.\text{wVf}(\text{vp}, h, \text{pk} - \text{pk}', z - z', \tau - \tau') = 1.$$

By construction of $\overline{\text{wVf}}$, this implies that $\overline{\text{wVf}}(\text{pp}, \text{pk} - \text{pk}', h, \sigma - \sigma') = 1$, finishing the proof. $\square$

Theorem 5.6. Suppose $R_q \cong \mathbb{F}_{q^t}^{d/t}$ for some $t \mid d$ and $(1 - 1/q^t)^{d/t}$ is overwhelming, and $\mathcal{S} \subseteq R$ is a (w, q) -strong sampling set, and every element in $\mathcal{W}$ is invertible over R_q . Then Construction 5.1 is $\mathcal{W}$ -multi-instance unforgeable (Definition 5.2).

Proof. We prove the theorem via a sequence of hybrid games. Let $\mathcal{A}$ be an efficient adversary in the multi-instance EUF-CMA security game (Figure 3).

Game G_0 . This is the actual multi-instance EUF-CMA security game.

Game G_1 : Rule out linear dependence. This is the same as G_0 , except: If $\mathcal{A}$ outputs $(\text{aux}, i^*, m, h, \sigma)$, let $m_1, \dots, m_k$ be $\mathcal{A}$'s signing queries on the i^* -th instance and slot h . WLOG, we assume that $m_1, \dots, m_k$ and $m_{k+1} := m$ are all distinct and $k \leq w$. Let H^{i^*} denote the hash function used in the i^* -th instance. If the vector $(\text{H}^{i^*}(m_1, h), \dots, \text{H}^{i^*}(m_{k+1}, h))$ are linearly dependent over R_q , the game outputs 0. By definition of w -linear independence (Definition 5.7),

$$\left| \Pr[G_0^{\mathcal{A}} = 1] - \Pr[G_1^{\mathcal{A}} = 1] \right| \leq \text{Adv}_{\mathcal{H}_w}^{\text{lindep}_w}(\lambda) \leq \text{negl}(\lambda).$$

Game G_2 : Guess forgery slot. This is the same as G_1 , except that the challenger samples an index $h \leftarrow_{\$} [\ell]$ and outputs 0 if $\mathcal{A}$'s forgery slot is not h . Since the sampling of h is independent and it matches with $\mathcal{A}$'s successful forgery with probability $1/\ell$, it holds that

$$\Pr[G_2^{\mathcal{A}} = 1] = \Pr[G_1^{\mathcal{A}} = 1]/\ell.$$

Game G_3 : Rule out prediction of secret values. This is the same as G_2 , except that G_3 further outputs 0 if the following event occurs: Let $(\mathbf{aux}, i^*, m, h, \sigma_h = (z_h, \tau_h))$ denotes $\mathcal{A}$'s forgery. Let $\mathbf{x}_h^1, \dots, \mathbf{x}_h^{w+1} \in R_q^{n_m}$ denote the secretly sampled entries in the i^* -th key generation for slot h . Let $\alpha := H^{i^*}(m, h) \in \mathcal{S}^{w+1}$. The game G_3 outputs 0 if

$$z_h = \mathbf{aux} \cdot \sum_{j=1}^{w+1} \alpha_j \cdot \mathbf{x}_h^j. \quad (19)$$

Claim 9. $|\Pr[G_3^{\mathcal{A}} = 1] - \Pr[G_2^{\mathcal{A}} = 1]| \leq \text{negl}(\lambda)$.

Proof. Consider G as the game that simulates the execution of $G_2^{\mathcal{A}}$ and outputs 1 only if $G_2^{\mathcal{A}} = 1$ and Eq. (19) holds. Clearly,

$$|\Pr[G_3^{\mathcal{A}} = 1] - \Pr[G_2^{\mathcal{A}} = 1]| \leq \Pr[G = 1].$$

Thus it suffices to show that $\Pr[G = 1] \leq \text{negl}(\lambda)$.

Recall the value guessing G_{guess} as in Definition 2.5. We consider the following (unbounded) adversary $\mathcal{C}$ that internally simulates the game $G^{\mathcal{A}}$:

- Samples $h^* \leftarrow \$[\ell]$.
- In the setup phase, $\mathcal{C}$ prepares the distributions $\{\mathcal{D}_{h,x,c}^{\alpha}\}_{h \in [\ell], x \in R_q^{n_m}, c \in R_q^{n_c}}$ for Π_{VC} , specified in Definition 3.8. $\mathcal{C}$ aborts if such distributions don't exist.
- In the key generation phase, instead of computing commitments $\text{cm}^1, \dots, \text{cm}^{w+1}$ honestly, $\mathcal{C}$ samples the commitments uniformly over $R_q^{n_c}$.
- Given signing query $(i, m, h \neq h^*)$, let $\text{cm}^1, \dots, \text{cm}^{w+1}$ denote the commitments in pk^i , let $\mathbf{x}_h^1, \dots, \mathbf{x}_h^{w+1} \in R_q^{n_m}$ be the secretly sampled entries in the i -th key generation for slot h . $\mathcal{C}$ computes $\alpha \leftarrow H^i(m, h)$, $\text{cm}^* := \sum_{j=1}^{w+1} \alpha_j \cdot \text{cm}^j$, $z_h := \sum_{j=1}^{w+1} \alpha_j \cdot \mathbf{x}_h^j$, and samples $\tau_h \leftarrow \$\mathcal{D}_{h,z_h,\text{cm}^*}^{\alpha}$. The output is (z_h, τ_h) .
- Given signing query (i, m, h^*) , $\mathcal{C}$ works similarly as above, except that z_h is derived by querying $\alpha \leftarrow H^i(m, h^*)$ to G_{guess} 's challenger.
- If the adversary $\mathcal{A}$ succeeds (in the simulation of G) and outputs forgery

$$(\mathbf{aux}, i^*, m, h^*, \sigma = (z_{h^*}, \tau_{h^*})).$$

$\mathcal{C}$ outputs $H^{i^*}(m, h^*)$ and $\mathbf{aux}^{-1} \cdot z_{h^*}$. (Note $\mathbf{aux} \in \mathcal{W}$ is invertible over R_q .)

By statistical sampleability of the vector commitment scheme Π_{VC} (Definition 3.8), the view of $\mathcal{A}$ in the above simulation is statistically close to the actual view in $G^{\mathcal{A}}$. On the

other hand, $\mathcal{C}$ wins G_{guess} if $\mathcal{A}$ succeeds in the simulated game of G . By the value guessing lemma (Lemma 2.9), it holds that

$$\Pr[G^{\mathcal{A}} = 1] \leq \text{negl}(\lambda) + \Pr[\mathcal{A} \text{ succeeds in the simulated game}] \leq \text{negl}(\lambda) + (|R_q| - |R_q^\times|) / |R_q|.$$

By the condition that $R_q \cong \mathbb{F}_{q^t}^{d/t}$ for some $t \mid d$ and $(1 - 1/q^t)^{d/t}$ is overwhelming, the last term is also negligible and thus $\Pr[G^{\mathcal{A}} = 1] \leq \text{negl}(\lambda)$, finishing the proof. $\square$

Finally, we show that $\Pr[G_3^{\mathcal{A}} = 1] \leq \text{negl}(\lambda)$, finishing the proof of Theorem 5.6.

Claim 10. $\Pr[G_3^{\mathcal{A}} = 1] \leq \text{negl}(\lambda)$.

Proof. Suppose for contradiction that $\Pr[G_3^{\mathcal{A}} = 1]$ is non-negligible. We obtain an adversary $\mathcal{B}$ that breaks the position-binding property of Π_{VC} (Definition 3.7). $\mathcal{B}$ simulates the game $G_3^{\mathcal{A}}$. Suppose $\mathcal{A}$ succeeds and outputs a forgery $(\text{aux}, i^*, m, h, \sigma = (z_h, \tau_h))$ such that

$$z_h \neq x^* := \text{aux} \cdot \sum_{j=1}^{w+1} \alpha_j \cdot \mathbf{x}_h^j,$$

where $\mathbf{x}_h^1, \dots, \mathbf{x}_h^{w+1} \in R_q^{n_m}$ are the sampled entries in the i^* -th key generation for slot h . Let $(\text{cm}^1, \dots, \text{cm}^{w+1})$ be the commitments in pk^{i^*} , and let $\text{st}^1, \dots, \text{st}^{w+1}$ be the opening states generated in the i^* -th key generation. Set $\alpha := H^{i^*}(m, h)$,

$$\text{cm}^* := \text{aux} \cdot \sum_{j=1}^{w+1} \alpha_j \cdot \text{cm}^j, \quad \pi^* := \text{aux} \cdot \sum_{j=1}^{w+1} \alpha_j \cdot \Pi_{\text{VC}}.\text{Open}(\text{st}^j, h). \quad (20)$$

The adversary $\mathcal{B}$ outputs $(h, \text{cm}^*, z_h, x^*, \tau_h, \pi^*)$.

Note that $z_h \neq x^*$ by assumption, and $\Pi_{\text{VC}}.\text{wVf}(\text{vp}, h, \text{cm}^*, z_h, \tau_h) = 1$ as $(\text{aux}, i^*, m, h, \sigma = (z_h, \tau_h))$ is a valid forgery. Moreover, by Eq. (20) and $(w+1, \mathcal{S}, \mathcal{O}_{\text{VC}})$ -homomorphic correctness for valid openings of Π_{VC} (Definition 3.3), $\Pi_{\text{VC}}.\text{Vf}(\text{vp}, h, \frac{1}{\text{aux}} \cdot (\text{cm}^*, x^*, \pi^*)) = 1$, and by $\mathcal{W}$ -weak verifiability of Π_{VC} (Definition 3.6), it implies that $\Pi_{\text{VC}}.\text{wVf}(\text{vp}, h, \text{cm}^*, x^*, \pi^*) = 1$. Therefore, $\mathcal{B}$'s output is a valid collision pair whenever $\mathcal{A}$ succeeds, finishing the proof. $\square$

$\square$

6 A Generic Aggregate Synchronous Signature Scheme

An aggregate synchronous signature scheme supports aggregation of signatures generated with respect to different public keys, but all at the same slot $h \in [\ell]$. The aggregate signature should be about the same length as a single signature. That is, aggregation is a compression mechanism that compresses multiple signatures at slot h into a single signature. We support aggregation of signatures on different messages.

Definition 6.1. An *Aggregate Synchronous Signature Scheme* is a tuple of five PPT algorithms

- $\text{Setup}(1^\lambda, \ell, w, N) \rightarrow \text{pp}$: for positive integers λ, ℓ, w , and an upper bound N on the number of signers in aggregation, outputs public parameters pp . The sizes of parameters and signatures are only proportional to $\log(N)$.
- KGen, Sign are as in [Definition 5.1](#).
- $\text{Aggregate}(\{\text{pk}_i, m_i, \sigma_i\}_{i=1}^n, h) \rightarrow \sigma$: aggregates $n \leq N$ signatures at slot $h \in [\ell]$ into one signature σ .
- $\text{Vf}(\{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma) \rightarrow \{0, 1\}$: verifies the aggregate signature σ on messages $\{m_i\}_{i=1}^n$ with respect to slot $h \in [\ell]$.

Definition 6.2. An aggregate synchronous signature scheme Sig is correct if for all positive integers λ, w, ℓ , and $n \leq N$, and all $\{m_i\}_{i=1}^n \subseteq \mathcal{M}, h \in [\ell]$:

$$\Pr[\text{Vf}(\text{pp}, \{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma) = 1] = 1,$$

where $\text{pp} \leftarrow \$ \text{Setup}(1^\lambda, \ell, w, N)$, and for all $i \in [1, n]$, $(\text{pk}_i, \text{sk}_i) \leftarrow \$ \text{KGen}(1^\lambda, \text{pp})$, $\sigma_i \leftarrow \$ \text{Sign}(\text{sk}_i, m_i, h)$, and finally $\sigma \leftarrow \$ \text{Aggregate}(\{\text{pk}_i, m_i, \sigma_i\}_{i=1}^n, h)$.

Security for an aggregate synchronous scheme requires that an adversary that generates $n - 1 \leq N - 1$ rogue public keys cannot generate an aggregate signature that verifies over a set of public keys with at least one honest public key.

Definition 6.3. An aggregate synchronous signature scheme Sig satisfies the **EUFCMA security** if for every PPT adversary $\mathcal{A}$

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{assig}}(\lambda) := \Pr[\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{assig}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

where the security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{assig}}(\lambda)$ is defined in [Figure 5](#). The definition extends to the random oracle model, where all the algorithms in $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{assig}}(\lambda)$ are given access to a random oracle. We then denote the advantage by $\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{assig-ro}}(\lambda)$.

There are three commonly used techniques for secure aggregation [[BS23](#), §15.5.3]: (i) message augmentation [[BGLS03](#)], (ii) proof of possession [[RY07](#)], and (iii) linear combinations [[BDN18](#)]. In [Construction 6.1](#), we show how to *generically* transform a key-homomorphic synchronous signature scheme into an aggregate signature using the linear combination technique, which in the context of lattices is also used in [[BK20](#); [FSZ22](#); [FHSZ23](#)]. In [Construction 6.2](#), we show how to *generically* transform a key-homomorphic synchronous signature scheme into an aggregate signature using the proof of possession technique. We leave it for future work to explore aggregation using message augmentation.

The Game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{assig}}(\lambda)$	Oracle $\text{OSign}(m, h)$
1 : $(\ell, w, N, \text{state}) \leftarrow \mathcal{A}(1^\lambda)$	1 : if $h \notin [\ell]$: return $\perp$
2 : $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w, N)$	2 : $S[h] \leftarrow S[h] \cup \{m\}$
3 : $(\text{pk}, \text{sk}) \leftarrow \text{KGen}(1^\lambda, \text{pp})$	3 : return $\text{Sign}(\text{sk}, m, h)$
4 : $(\{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma) \leftarrow \mathcal{A}^{\text{OSign}(\cdot, \cdot)}(1^\lambda, \text{pk}, \text{pp}, \text{state})$	
5 : if $ S[h] > w$ OR $m_1 \in S[h]$ OR $\text{pk}_1 \neq \text{pk}$: return 0	
6 : return $(n \leq N)$ AND $\text{Vf}(\text{pp}, \{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma)$	

Figure 5: The EUF-CMA security game for an aggregate synchronous signature scheme Sig , where $\text{Sig} = (\text{Setup}, \text{KGen}, \text{Sign}, \text{Aggregate}, \text{Vf})$.

6.1 Aggregating Synchronous Signatures via Linear Combinations

Here we show how to aggregate signatures by taking a low norm linear combination of the signatures, where the coefficients are generated by a hash function. We describe the construction as one-shot aggregation, but it easily extends to supporting aggregation of aggregates. Our aggregate signature scheme is presented in Figure 6 as Construction 6.1. The construction uses primitives below:

- Π_{sync}^N is a key homomorphic synchronous signature scheme (as in Definition 5.1), with message space $\mathcal{M}$, public key space $\mathcal{P}$. Moreover, there exist efficient algorithms DerivePK , Vf , wVf that satisfy $(N, \mathcal{S})$ -strong public key homomorphic correctness (Definition 5.4), $(\mathcal{S} - \mathcal{S})$ -weak verification consistency (Definition 5.5), and strong robust homomorphism (Definition 5.6).
- $\mathcal{H}_{\text{agg}} = \{h : [\mathcal{P} \times \mathcal{M}]^{\leq N} \times [\ell] \rightarrow \mathcal{S}^N\}$ is a family of hash functions that we model as random oracles.

Theorem 6.1. *Construction 6.1 is correct (Definition 6.2).*

Proof. Fix any λ, w, ℓ , and $n \leq N$, and $\{m_i\}_{i=1}^n \subseteq \mathcal{M}, h \in [\ell]$. Let $\text{pp} = (\text{pp}_{\text{sync}}, \text{H}_{\text{agg}}) \leftarrow \text{Setup}(1^\lambda, \ell, w, N)$, and for all $i \in [1, n]$,

$$(\text{pk}_i, \text{sk}_i) \leftarrow \text{KGen}(1^\lambda, \text{pp}), \quad \text{pk}_{m_i, h}^i \leftarrow \Pi_{\text{sync}}^N.\text{DerivePK}(\text{pk}_i, m_i, h), \quad \sigma_i \leftarrow \text{Sign}(\text{sk}_i, m_i, h).$$

Let $(r_1, \dots, r_N) := \text{H}_{\text{agg}}(\{\text{pk}_i, m_i\}_{i=1}^n, h)$ and set $\text{pk}_{\text{agg}} := \sum_{i=1}^n r_i \cdot \text{pk}_{m_i, h}^i$, $\sigma_{\text{agg}} := \sum_{i=1}^n r_i \cdot \sigma_i$. By the $(N, \mathcal{S})$ -strong public key homomorphic correctness of Π_{sync}^N (Definition 5.4), it holds that

$$\Pi_{\text{sync}}^N.\text{Vf}(\text{pp}_{\text{sync}}, \text{pk}_{\text{agg}}, h, \sigma_{\text{agg}}) = 1.$$

Construction 6.1.

- $\text{Setup}(1^\lambda, \ell, w, N) \rightarrow \text{pp}$:
 - $\text{pp}_{\text{sync}} \leftarrow \Pi_{\text{sync}}^N.\text{Setup}(1^\lambda, \ell, 2w)$; $\text{H}_{\text{agg}} \leftarrow_{\$} \mathcal{H}_{\text{agg}}$
 - Output $\text{pp} := (\text{pp}_{\text{sync}}, \text{H}_{\text{agg}})$
- $\text{KGen}(1^\lambda, \text{pp})$ and $\text{Sign}(\text{sk}, m, h)$ directly call the KGen and Sign algorithms of the basic signature scheme Π_{sync}^N .
- $\text{Aggregate}(\{\text{pk}_i, m_i, \sigma_i\}_{i=1}^n, h) \rightarrow \sigma$:
 - Abort if $n > N$, otherwise, $(r_1, \dots, r_N) \leftarrow \text{H}_{\text{agg}}(\{\text{pk}_i, m_i\}_{i=1}^n, h) \in \mathcal{S}^N$
 - Output $\sigma_{\text{agg}} := \sum_{i=1}^n r_i \cdot \sigma_i$
- $\text{Vf}(\text{pp}, \{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma_{\text{agg}}) \rightarrow \{0, 1\}$
 - Output 0 if $n > N$, otherwise $(r_1, \dots, r_N) \leftarrow \text{H}_{\text{agg}}(\{\text{pk}_i, m_i\}_{i=1}^n, h)$
 - For all $i \in [n]$: $\text{pk}_{m_i, h}^i \leftarrow \Pi_{\text{sync}}^N.\text{DerivePK}(\text{pk}_i, m_i, h)$
 - $\text{pk}_{\text{agg}} \leftarrow \sum_{i=1}^n r_i \cdot \text{pk}_{m_i, h}^i$
 - Output $\Pi_{\text{sync}}^N.\text{Vf}(\text{pp}_{\text{sync}}, \text{pk}_{\text{agg}}, h, \sigma_{\text{agg}})$

Figure 6: A w -way ℓ -slot aggregate synchronous signature scheme $\Pi_{\text{agg}}^{w, \ell, N}$ using linear combinations to aggregate.

By the specification of Aggregate and Vf in [Construction 6.1](#), this implies that

$$\text{Vf}(\text{pp}, \{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma_{\text{agg}}) = 1,$$

and thus correctness holds. $\square$

Theorem 6.2. *[Construction 6.1](#) is EUF-CMA secure. More precisely, let $\mathcal{W} := \mathcal{S} - \mathcal{S}$, let Q_{agg} be the number of oracle queries made to H_{agg} . For every PPT adversary $\mathcal{A}$ to the EUF-CMA security game w.r.t. the aggregate synchronous signature scheme ([Figure 5](#)), there is a PPT adversary $\mathcal{B}$ in the EUF-CMA security game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}(\lambda)$ ([Figure 3](#)) such that*

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{assig-ro}}(\lambda) \leq \frac{Q_{\text{agg}} + 1}{|\mathcal{S}|} + \sqrt{(Q_{\text{agg}} + 1) \cdot \text{Adv}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}(\lambda)}.$$

Proof. Given any adversary $\mathcal{A}'$ w.r.t. the aggregate signature, we slightly modify it so that it always queries the random oracle H_{agg} on its forgery. This increases the number of oracle queries by at most one and does not decrease $\mathcal{A}'$'s success probability. Let $\mathcal{A}$ denote the resulting adversary. We start by considering the following simulation algorithm $\text{Sim}^{\mathcal{A}}$:

- Simulate the game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{assig-ro}}(\lambda)$ where $\text{Sig} := \Pi_{\text{agg}}^{w, \ell, N}$ is specified in [Construction 6.1](#).
- If $\mathcal{A}$ outputs a valid forgery $\mathcal{I} := (\{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma)$, let $q_1 \in [Q_{\text{agg}} + 1]$ denote the index of the first H_{agg} -query with input $((\{\text{pk}_i, m_i\}_{i=1}^n, h)$, the simulator $\text{Sim}^{\mathcal{A}}$ outputs $(\mathcal{I}, q_1)$. Otherwise, $\text{Sim}^{\mathcal{A}}$ outputs $\perp$.

From the simulator $\text{Sim}^{\mathcal{A}}$, we construct an adversary $\mathcal{B}$ for the *augmented* EUF-CMA security game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$ ([Figure 3](#)) as follows:

- Upon receiving ℓ, w, N from $\mathcal{A}$, the adversary $\mathcal{B}$ forwards (ℓ, w) to the challenger and receives the public parameter pp_{sync} and a public key pk^* . $\mathcal{B}$ then forwards³ pp_{sync} and pk^* to $\mathcal{A}$.
- $\mathcal{B}$ continues the simulation of the algorithm $\text{Sim}^{\mathcal{A}}$, where all signing queries from $\mathcal{A}$ are answered by forwarding the query to the challenger in $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$. $\mathcal{B}$ aborts if Sim outputs $\perp$, otherwise, let $(\mathcal{I}, q_1)$ be Sim 's output.
- $\mathcal{B}$ rewinds and simulates the algorithm $\text{Sim}^{\mathcal{A}}$ for the second time. Here before the q_1 -th query to H_{agg} , the simulation is exactly the same as in the first execution of $\text{Sim}^{\mathcal{A}}$. But for the q_1 -th query to H_{agg} , denote by $(r_1, \dots, r_N)$ the output in the first simulation. $\mathcal{B}$ samples $r'_1 \leftarrow \mathcal{S}$ and returns $(r'_1, r_2, \dots, r_N)$. After that, $\mathcal{B}$ continues the simulation of $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$, where for each new H_{agg} -query that hasn't been queried before, the *first* $\mathcal{S}$ -element of the output is sampled using *fresh* randomness. Finally, $\mathcal{B}$ aborts if the simulation outputs $\perp$, otherwise let $(\mathcal{I}', q'_1)$ be the output.

³ $\mathcal{B}$ does not send H_{agg} as a parameter since H_{agg} is modeled as a random oracle.

- Abort if $r_1 = r'_1$ or $q_1 \neq q'_1$ or $\mathcal{I}, \mathcal{I}'$ are *not* of the following form: $\mathcal{I} = (\{\mathbf{pk}_i, m_i\}_{i=1}^n, h, \sigma_{\text{agg}})$ and $\mathcal{I}' = (\{\mathbf{pk}_i, m_i\}_{i=1}^n, h, \sigma'_{\text{agg}})$, where $\mathbf{pk}_1 = \mathbf{pk}^*$. Otherwise, $\mathcal{B}$ returns forgery:

$$(\text{aux} := (r_1 - r'_1), 1, m_1, h, \sigma_h := \sigma_{\text{agg}} - \sigma'_{\text{agg}}). \quad (21)$$

Note that in total, $\mathcal{B}$ made at most $w + w = 2w$ signing queries for slot h . Hence $\mathcal{B}$ is an admissible adversary in game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$. Next, we show that the forgery is valid if $\mathcal{B}$ does not abort.

Claim 11. *If $\mathcal{B}$ does not abort, then its output is a valid forgery in $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$ (Figure 3).*

Proof. If $\mathcal{B}$ does not abort, both $\mathcal{I} = (\{\mathbf{pk}_i, m_i\}_{i=1}^n, h, \sigma_{\text{agg}})$ and $\mathcal{I}' = (\{\mathbf{pk}_i, m_i\}_{i=1}^n, h, \sigma'_{\text{agg}})$ are valid. For all $i \in [n]$, let $\mathbf{pk}_{m_i, h}^i := \Pi_{\text{sync}}^N \cdot \text{DerivePK}(\mathbf{pk}_i, m_i, h)$. Let $\mathbf{pk}_{\text{agg}} := \sum_{i=1}^n r_i \cdot \mathbf{pk}_{m_i, h}^i$ and $\mathbf{pk}'_{\text{agg}} := r'_1 \cdot \mathbf{pk}_{m_1, h}^1 + \sum_{i=2}^n r_i \cdot \mathbf{pk}_{m_i, h}^i$. By construction in Figure 6,

$$\Pi_{\text{sync}}^N \cdot \overline{\text{Vf}}(\mathbf{pp}_{\text{sync}}, \mathbf{pk}_{\text{agg}}, h, \sigma_{\text{agg}}) = \Pi_{\text{sync}}^N \cdot \overline{\text{Vf}}(\mathbf{pp}_{\text{sync}}, \mathbf{pk}'_{\text{agg}}, h, \sigma'_{\text{agg}}) = 1.$$

Then by the strong robust homomorphism of Π_{sync}^N (Definition 5.6), it holds that

$$\Pi_{\text{sync}}^N \cdot \overline{\text{wVf}}(\mathbf{pk}_{\text{agg}} - \mathbf{pk}'_{\text{agg}}, h, \sigma_{\text{agg}} - \sigma'_{\text{agg}}) = 1,$$

where $\mathbf{pk}_{\text{agg}} - \mathbf{pk}'_{\text{agg}} = (r_1 - r'_1) \cdot \mathbf{pk}_{m_1, h}^1 = (r_1 - r'_1) \cdot \Pi_{\text{sync}}^N \cdot \text{DerivePK}(\mathbf{pk}_1, m_1, h)$. By the $\mathcal{W}$ -weak verification consistency of Π_{sync}^N (Definition 5.5), it holds that

$$\Pi_{\text{sync}}^N \cdot \text{wVf}((r_1 - r'_1) \cdot \mathbf{pk}_1, m_1, h, \sigma_{\text{agg}} - \sigma'_{\text{agg}}) = 1,$$

and since $(r_1 - r'_1) \in \mathcal{W}$, $\mathcal{B}$'s output as in Eq. (21) is a valid forgery in game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$. $\square$

It remains to show that $\mathcal{B}$ succeeds with high probability. By the general forking lemma (Lemma 2.8), it holds that

$$\Pr[\text{Sim}^{\mathcal{A}} \neq \perp] \leq \frac{Q_{\text{agg}} + 1}{|\mathcal{S}|} + \sqrt{(Q_{\text{agg}} + 1) \cdot \text{Adv}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}(\lambda)}.$$

Moreover, note that $\Pr[\text{Sim}^{\mathcal{A}} \neq \perp] = \text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{assig-ro}}(\lambda)$, thus the theorem holds. $\square$

Remark 6.1. In Construction 6.1, the underlying synchronous signature scheme Π_{sync}^N is required to be $2w$ -way secure, i.e., every slot is allowed to sign at most $2w$ messages. This is a proof artifact due to the forking lemma. We conjecture that the aggregation scheme is still secure even if we use a w -way signature.

6.2 Aggregating Synchronous Signatures via Proof of Possession

Here we show how to aggregate signatures by providing a proof of possession of the secret key. As in the previous section, we describe the construction as one-shot aggregation, but it naturally extends to supporting aggregation of aggregate signatures. For the construction in this section, we will need the following definitions from [RT25; BCS16] of straight-line extractable non-interactive zero-knowledge proofs in the random oracle model:

Definition 6.4 (Non-Interactive Zero-Knowledge Random Oracle Arguments [RT25]). *Let $\mathcal{R} = \{(x, w)\}$ be a relation, and $\mathcal{L} = \{x : \exists w, (x, w) \in \mathcal{R}\}$ be the language induced by $\mathcal{R}$. A non-interactive random oracle argument for $\mathcal{R}$ is a pair $(\mathcal{P}, \mathcal{V})$, where both $\mathcal{P}$ and verifier $\mathcal{V}$ have oracle access to a random oracle $\mathcal{O}$. $\mathcal{P}$ is a probabilistic oracle-aided algorithm, and $\mathcal{V}$ is a deterministic algorithm. A non-interactive zero-knowledge random oracle argument must satisfy the following properties:*

- Completeness: $(\mathcal{P}, \mathcal{V})$ is complete if for all $(x, w) \in \mathcal{R}$:

$$\Pr[\mathcal{V}^{\mathcal{O}}(1^\lambda, x, \pi) = 1 : \pi \leftarrow \mathcal{P}^{\mathcal{O}}(1^\lambda, x, w)] \geq 1 - \text{negl}(\lambda).$$

- Straight-line Knowledge Soundness: $(\mathcal{P}, \mathcal{V})$ has knowledge error ϵ if there is an efficient algorithm E such that for every x and every malicious prover $\mathcal{P}^*$:

$$\Pr[(x, E^{\mathcal{P}^*}(1^\lambda, x)) \in \mathcal{R}] \geq \Pr[\mathcal{V}^{\mathcal{O}}(1^\lambda, x, \pi) = 1 : \pi \leftarrow (\mathcal{P}^*)^{\mathcal{O}}(1^\lambda, x)] - \epsilon.$$

Here, the notation $E^{\mathcal{P}^*}(1^\lambda, x)$ means that E invokes $\mathcal{P}^*$ as a black-box and simulates the oracle to $\mathcal{P}^*$. E 's running time is polynomial in λ , $|x|$, and the number of random oracle queries issued by $\mathcal{P}^*$ on these inputs. Moreover, E only invokes $\mathcal{P}^*$ once.

We say $(\mathcal{P}, \mathcal{V})$ is straight-line knowledge-sound if it has negligible knowledge error.

- Zero-Knowledge [BCS16]: $(\mathcal{P}, \mathcal{V})$ is zero-knowledge if there is an efficient algorithm S such that for every $(x, w) \in \mathcal{R}$ and efficient distinguisher D , we have:

$$\left| \Pr[D^{\mathcal{O}[\mu]}(\pi) = 1 : (\pi, \mu) \leftarrow S^{\mathcal{O}}(x)] - \Pr[D^{\mathcal{O}}(\pi) = 1 : \pi \leftarrow \mathcal{P}^{\mathcal{O}}(x, w)] \right| = \text{negl}(\lambda),$$

where the function $\mathcal{O}[\mu]$, on input x , outputs $\mu(x)$ if μ is defined on x , otherwise it outputs $\mathcal{O}(x)$.

Our aggregate signature scheme is presented in Figure 7 as Construction 6.2. The construction uses primitives below:

- Π_{sync}^N is a key homomorphic synchronous signature scheme (as in Definition 5.1), with message space $\mathcal{M}$, public key space $\mathcal{P}$ and deterministic signing algorithm Sign . Moreover, there exist efficient algorithms DerivePK , Vf , wVf satisfying $(N, \mathcal{W})$ -strong public key homomorphic correctness (Definition 5.4), $\mathcal{W}$ -weak verification consistency (Definition 5.5), and strong robust homomorphism (Definition 5.6), where $\mathcal{W} = \{1_R\}$.

- $\Pi_{\text{NIZK}} = (\text{Setup}, \mathcal{P}, \mathcal{V})$ is a non-interactive zero-knowledge random oracle argument for the relation:

$$\mathcal{R} = \{(\text{pk}, \text{sk}) : (\text{pk}, \text{sk}) \text{ is a valid public key-secret key pair for } \Pi_{\text{sync}}^N\}$$

with knowledge error ϵ_{NIZK} .

Theorem 6.3. *Construction 6.2 is correct.*

Proof. Fix any λ, w, ℓ , and $n \leq N$, and $\{m_i\}_{i=1}^n \subseteq \mathcal{M}, h \in [\ell]$. Let $\text{pp} \leftarrow \$ \text{Setup}(1^\lambda, \ell, w, N)$, and for all $i \in [1, n]$,

$$(\text{pk}_i, \text{sk}_i) \leftarrow \$ \text{KGen}(1^\lambda, \text{pp}), \quad \text{pk}_{m_i, h}^i \leftarrow \Pi_{\text{sync}}^N.\text{DerivePK}(\text{pk}_i, m_i, h), \quad \sigma_i \leftarrow \$ \text{Sign}(\text{sk}_i, m_i, h).$$

Set $\text{pk}_{\text{agg}} := \sum_{i=1}^n \text{pk}_{m_i, h}^i$, $\sigma_{\text{agg}} := \sum_{i=1}^n \sigma_i$. By the $(N, \mathcal{W})$ -strong public key homomorphic correctness of Π_{sync}^N (Definition 5.4), it holds that

$$\Pi_{\text{sync}}^N.\overline{\text{Vf}}(\text{pp}_{\text{sync}}, \text{pk}_{\text{agg}}, h, \sigma_{\text{agg}}) = 1.$$

By the specification of `Aggregate` and `Vf` in Construction 6.2, this implies that

$$\text{Vf}(\text{pp}, \{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma_{\text{agg}}) = 1,$$

and thus correctness holds. $\square$

Theorem 6.4. *Construction 6.2 is EUF-CMA secure. More precisely, let $\mathcal{W} := \{1_R\}$. For every PPT adversary $\mathcal{A}$ to the EUF-CMA security game w.r.t. the aggregate synchronous signature scheme (Figure 5), there is a PPT adversary $\mathcal{B}$ in the augmented EUF-CMA security game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}(\lambda)$ (Figure 3) such that*

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{assig}}(\lambda) \leq \text{Adv}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}(\lambda) + (n - 1) \cdot \epsilon_{\text{NIZK}}.$$

Proof. Given an adversary $\mathcal{A}$ to the EUF-CMA security game w.r.t. the aggregate synchronous signature scheme, we construct an adversary $\mathcal{B}$ for the *augmented* EUF-CMA security game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$ (Figure 3) as follows:

- Upon receiving ℓ, w, N from $\mathcal{A}$, the adversary $\mathcal{B}$ forwards (ℓ, w) to the challenger and receives the public parameter pp_{sync} and a public key pk^* . $\mathcal{B}$ uses the simulator `Sim` for Π_{NIZK} to simulate a proof $(\pi^*, \mu) \leftarrow \text{Sim}(\text{pk}^*)$. $\mathcal{B}$ forwards pp_{sync} and (pk^*, π^*) to $\mathcal{A}$.
- To answer any random oracle query x from $\mathcal{A}$, $\mathcal{B}$ responds with $\mu(x)$ if μ is defined on x and with a randomly sampled element otherwise; $\mathcal{B}$ stores any randomly sampled elements so it can answer $\mathcal{A}$ random oracle queries consistently.

Construction 6.2.

- $\text{Setup}(1^\lambda, \ell, w, N) \rightarrow \text{pp}$:
 - $\text{pp}_{\text{sync}} \leftarrow \Pi_{\text{sync}}^N.\text{Setup}(1^\lambda, \ell, w)$
 - $\text{pp}_{\text{NIZK}} \leftarrow \Pi_{\text{NIZK}}.\text{Setup}(1^\lambda)$
 - Output $\text{pp} := (\text{pp}_{\text{sync}}, \text{pp}_{\text{NIZK}})$
- $\text{KGen}(1^\lambda, \text{pp}) \rightarrow (\text{pk}, \text{sk})$:
 - $(\overline{\text{pk}}, \overline{\text{sk}}) \leftarrow \Pi_{\text{sync}}^N.\text{KGen}(1^\lambda, \text{pp})$
 - $\pi \leftarrow \Pi_{\text{NIZK}}.\mathcal{P}(\text{pp}_{\text{NIZK}}, \text{pk}, \text{sk})$
 - Output $\text{pk} := (\overline{\text{pk}}, \pi)$ and $\text{sk} := \overline{\text{sk}}$
- $\text{Sign}(\text{sk}, m, h)$ directly calls the Sign algorithm of the basic signature scheme Π_{sync}^N .
- $\text{Aggregate}(\{\text{pk}_i, m_i, \sigma_i\}_{i=1}^n, h) \rightarrow \sigma$:
 - Abort if $n > N$, otherwise output $\sigma_{\text{agg}} := \sum_{i=1}^n \sigma_i$
- $\text{Vf}(\text{pp}, \{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma_{\text{agg}}) \rightarrow \{0, 1\}$
 - Output 0 if $n > N$.
 - For all $i \in [n]$, parse $(\overline{\text{pk}}_i, \pi_i) \leftarrow \text{pk}_i$
 - For all $i \in [n]$, if $\mathcal{V}(\text{pp}_{\text{NIZK}}, \overline{\text{pk}}_i, \pi_i) = 0$, output 0.
 - For all $i \in [n]$: $\text{pk}_{m_i, h}^i \leftarrow \Pi_{\text{sync}}^N.\text{DerivePK}(\overline{\text{pk}}_i, m_i, h)$
 - $\text{pk}_{\text{agg}} \leftarrow \sum_{i=1}^n \text{pk}_{m_i, h}^i$
 - Output $\Pi_{\text{sync}}^N.\overline{\text{Vf}}(\text{pp}_{\text{sync}}, \text{pk}_{\text{agg}}, h, \sigma_{\text{agg}})$

Figure 7: A w -way ℓ -slot aggregate synchronous signature scheme $\Pi_{\text{agg}}^{w, \ell, N}$ using proof of possession to aggregate.

- To answer a signing query (m, h) from $\mathcal{A}$, $\mathcal{B}$ forwards (m, h) to the challenger in $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{sig}, 1}$ and returns challenger's output.
- When $\mathcal{A}$ outputs a valid forgery $\mathcal{I} = (\{\text{pk}_i, m_i\}_{i \in [n]}, h, \sigma_{\text{agg}})$, for $i \in [2, n]$, $\mathcal{B}$ does the following:
 - Parses $(\overline{\text{pk}}_i, \pi_i) \leftarrow \text{pk}_i$
 - Uses the extractor E for Π_{NIZK} to compute $\text{sk}_i \leftarrow E^{\mathcal{A}}(\overline{\text{pk}}_i, \pi_i)$. If the extractor fails on any call, abort
 - Computes $\sigma_i \leftarrow \text{Sign}(\text{sk}_i, m, h)$

$\mathcal{B}$ outputs the forgery:

$$(1, 1, m_1, h, \sigma_h := \sigma_{\text{agg}} - \sum_{i=2}^n \sigma_i). \quad (22)$$

Note that in total, $\mathcal{B}$ makes at most w signing queries for slot h . Hence $\mathcal{B}$ is an admissible adversary in game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{ssig}, 1}$. Next, we show that the forgery is valid if $\mathcal{B}$ does not abort.

Claim 12. *If $\mathcal{B}$ does not abort, then its output is a valid forgery in $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{ssig}, 1}$ (Figure 3).*

Proof. Let $\mathcal{I} = (\{\text{pk}_i, m_i\}_{i=1}^n, h, \sigma_{\text{agg}})$. For all $i \in [n]$, parse $(\overline{\text{pk}}_i, \pi_i) \leftarrow \text{pk}_i$, and let $\text{pk}_{m_i, h}^i := \Pi_{\text{sync}}^N \cdot \text{DerivePK}(\overline{\text{pk}}_i, m_i, h)$. Let $\text{pk}_{\text{agg}} := \sum_{i=1}^n \text{pk}_{m_i, h}^i$. Because $\mathcal{I}$ is a valid forgery, we have that

$$\Pi_{\text{sync}}^N \cdot \overline{\text{Vf}}(\text{pp}_{\text{sync}}, \text{pk}_{\text{agg}}, h, \sigma_{\text{agg}}) = 1. \quad (23)$$

If $\mathcal{B}$ does not abort, then it has valid public key-secret key pairs $\{(\overline{\text{pk}}_i, \text{sk}_i)\}_{i=2}^n$ for Π_{sync}^N . For $i \in [2, n]$, let $\sigma_i \leftarrow \text{Sign}(\text{sk}_i, m_i, h)$. Let $\text{pk}'_{\text{agg}} := \sum_{i=2}^n \text{pk}_{m_i, h}^i$, and let $\sigma'_{\text{agg}} := \sum_{i=2}^n \sigma_i$. By the correctness of the scheme, $\text{Vf}(\text{pp}, \{\text{pk}_i, m_i\}_{i=2}^n, h, \sigma'_{\text{agg}}) = 1$, so:

$$\Pi_{\text{sync}}^N \cdot \overline{\text{Vf}}(\text{pp}_{\text{sync}}, \text{pk}'_{\text{agg}}, h, \sigma'_{\text{agg}}) = 1. \quad (24)$$

Given Eq. (23) and Eq. (24), by the strong robust homomorphism of Π_{sync}^N (Definition 5.6):

$$\Pi_{\text{sync}}^N \cdot \overline{\text{wVf}}(\text{pp}_{\text{sync}}, \text{pk}_{\text{agg}} - \text{pk}'_{\text{agg}}, h, \sigma_{\text{agg}} - \sigma'_{\text{agg}}) = 1,$$

where $\text{pk}_{\text{agg}} - \text{pk}'_{\text{agg}} = \text{pk}_{m_1, h}^1 = \Pi_{\text{sync}}^N \cdot \text{DerivePK}(\overline{\text{pk}}_1, m_1, h)$. By the $\mathcal{W}$ -weak verification consistency of Π_{sync}^N (Definition 5.5):

$$\begin{aligned} \Pi_{\text{sync}}^N \cdot \overline{\text{wVf}}(\text{pp}_{\text{sync}}, \text{DerivePK}(\overline{\text{pk}}_1, m, h), h, \sigma_{\text{agg}} - \sigma'_{\text{agg}}) &= 1 \\ \implies \Pi_{\text{sync}}^N \cdot \overline{\text{wVf}}(\text{pp}_{\text{sync}}, \overline{\text{pk}}_1, m, h, \sigma_{\text{agg}} - \sigma'_{\text{agg}}) &= 1, \end{aligned}$$

and since $\mathcal{W} = \{1_R\}$, $\mathcal{B}$'s output as in Eq. (22) is a valid forgery in game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{augssig}, 1, \mathcal{W}}$. $\square$

It remains to show that $\mathcal{B}$ succeeds with high probability. Because E is a straight-line extractor, it does not need to rewind $\mathcal{A}$ and can extract sk_i using just $\mathcal{A}$'s calls to the random oracle. $\mathcal{B}$ knows all of these calls as it simulates the random oracle for $\mathcal{A}$. $\mathcal{B}$ aborts if the extractor E fails to extract a valid secret key sk_i when run on any member of the set $\{\text{pp}_{\text{NIZK}}, \overline{\text{pk}}_i, \pi_i\}_{i=2}^n$. The extractor has knowledge error ϵ_{NIZK} , so by a union bound, the probability that $\mathcal{B}$ does not abort is $\leq (n-1) \cdot \epsilon_{\text{NIZK}}$. This finishes the proof. $\square$

7 A Generic Threshold Synchronous Signature Scheme

In a T -out-of- N threshold synchronous signature scheme, each party receives a secret key and can produce a partial signature for any message at a given slot. To construct a full signature on a message in a given slot, we need at least T partial signatures on this message at that slot. Existential unforgeability holds as long as an adversary holds fewer than T secret keys.

We present two threshold synchronous signature schemes: a statically secure scheme where the adversary corrupts $T-1$ parties before receiving any partial signatures from honest parties and an adaptively secure scheme where the adversary may receive partial signatures from honest parties when deciding which parties to corrupt.

Definition 7.1. *A Threshold Synchronous Signature Scheme is a tuple of PPT algorithms:*

- $\text{Setup}(1^\lambda, \ell, w, N, T) \rightarrow \text{pp}$: for positive integers λ, ℓ, w, N, T , outputs public parameters for a w -way ℓ -slot scheme with a T -out-of- N threshold.
- $\text{KGen}(1^\lambda, \text{pp}) \rightarrow (\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i \in [N]})$: generates a public key pk , N partial verification keys $\text{pk}_1, \dots, \text{pk}_N$, and N signing key shares $\text{sk}_1, \dots, \text{sk}_N$.
- Either PartialSign or PartialSignQuorum will be defined⁴:
 - $\text{PartialSign}(\text{sk}_i, m, h) \rightarrow \sigma_i$: signs $m \in \mathcal{M}$ at slot $h \in [\ell]$ with signing key share sk_i and outputs signature share σ_i .
 - $\text{PartialSignQuorum}(\text{sk}_i, m, h, S) \rightarrow \sigma_i$: signs $m \in \mathcal{M}$ at slot $h \in [\ell]$ with signing key share sk_i for signing set $S \subseteq [N]$ where $|S| \geq T$ and outputs signature share σ_i .
- $\text{PartialVerify}(\text{pk}_i, m, h, \sigma_i) \rightarrow \{0, 1\}$: verifies a partial signature share σ_i on message m with respect to slot $h \in [\ell]$.
- $\text{Combine}(S, m, h, \{\text{pk}_i, \sigma_i\}_{i \in S}) \rightarrow \sigma / \perp$: takes as input a set S with $|S| \geq T$ and combines $|S|$ signature shares into a signature σ or outputs $\perp$.

⁴ PartialSign enables partial signing without knowing the signing set S beforehand.

- $\text{Vf}(1^\lambda, \text{pp}, \text{pk}, m, h, \sigma) \rightarrow \{0, 1\}$: verifies a signature σ on m with respect to slot $h \in [\ell]$.

Definition 7.2. A threshold synchronous signature scheme Sig is correct if for all positive integers λ, w, ℓ , all $m \in \mathcal{M}$, $h \in [\ell]$, all subsets S of $[N]$ such that $|S| \geq T$:

$$\Pr[\text{Vf}(1^\lambda, \text{pk}, m, h, \sigma) = 1] = 1 - \text{negl}(\lambda),$$

where $\text{pp} \leftarrow \$ \text{Setup}(1^\lambda, \ell, w, N, T)$; $(\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i=1}^N) \leftarrow \$ \text{KGen}(1^\lambda, \text{pp})$; if PartialSign is defined then $\sigma_i \leftarrow \$ \text{PartialSign}(\text{sk}_i, m, h)$ for all $i \in S$ and otherwise $\sigma_i \leftarrow \$ \text{PartialSignQuorum}(\text{sk}_i, m, h, S)$ for all $i \in S$; and $\sigma \leftarrow \$ \text{Combine}(S, m, h, \{\text{pk}_i, \sigma_i\}_{i \in S})$.

Definition 7.3. A threshold synchronous signature scheme Sig is **statically secure** if for every PPT adversary $\mathcal{A}$,

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}(\lambda) := \Pr[\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

where the security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}(\lambda)$ is defined in [Figure 8](#).

Definition 7.4. A threshold synchronous signature scheme Sig is **adaptively secure** if for every PPT adversary $\mathcal{A}$,

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{ad}}(\lambda) := \Pr[\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{ad}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

where the security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{ad}}(\lambda)$ is defined in [Figure 8](#).

Definition 7.5. A threshold synchronous signature scheme Sig is **robust** if for every PPT adversary $\mathcal{A}$,

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{rb}}(\lambda) := \Pr[\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{rb}}(\lambda) = 1] \leq \text{negl}(\lambda)$$

where the security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{rb}}(\lambda)$ is defined in [Figure 8](#).

7.1 Secret Sharing Schemes

We define secret sharing for ring elements and give the construction for Shamir secret sharing for ring elements, which we use to construct our threshold synchronous signature scheme.

Definition 7.6. A **Secret Sharing Scheme** over a ring R is a tuple of two PPT algorithms:

- $\text{Share}(N, T, x \in R) \rightarrow \{x_i\}_{i \in [N]} \in R^N$: produces secret shares for a T -out-of- N secret sharing of x .
- $\text{Combine}(S, \{x_i\}_{i \in [S]}) \rightarrow x$: given $S \subseteq [N]$ where $|S| \geq T$, recovers x from the shares $\{x_i\}_{i \in [N]}$.

The Game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}(\lambda)$ (static) 1 : $(\ell, w, N, T, C, \text{state}) \leftarrow \mathcal{A}(1^\lambda)$ 2 : if $C \not\subseteq [N]$ OR $ C \neq T - 1$: return 0 3 : $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w, N, T)$ 4 : $(\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i=1}^N) \leftarrow \text{KGen}(1^\lambda, \text{pp})$ 5 : $\text{pp}_{\mathcal{A}} \leftarrow (\text{pk}, \{\text{pk}_i\}_{i=1}^N, \{\text{sk}_i\}_{i \in C}, \text{state})$ 6 : $(m, h, \sigma) \leftarrow \mathcal{A}^{\text{OPartialSign}(\cdot, \cdot, \cdot)}(1^\lambda, \text{pp}_{\mathcal{A}})$ 7 : $U := \{m' \in Q[h] : Q[h][m'] \cup C \geq T\}$ 8 : if $ U > w$ OR $ Q[h][m] \cup C \geq T$: return 0 9 : return $\text{Vf}(1^\lambda, \text{pk}, m, h, \sigma)$	Oracle $\text{OCorrupt}(i)$ 1 : if $ C \geq T$: return $\perp$ 2 : $C \leftarrow C \cup \{i\}$ 3 : return sk_i
The Game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{ad}}(\lambda)$ 1 : $(\ell, w, N, T, \text{state}) \leftarrow \mathcal{A}(1^\lambda)$ 2 : $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w, N, T)$ 3 : $C \leftarrow \{\}$ 4 : $(\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i=1}^N) \leftarrow \text{KGen}(1^\lambda, \text{pp})$ 5 : $\text{pp}_{\mathcal{A}} \leftarrow (\text{pk}, \{\text{pk}_i\}_{i=1}^N, \text{state})$ 6 : $(m, h, \sigma) \leftarrow \mathcal{A}^{\text{OCorrupt}(\cdot), \text{OPartialSignQuorum}(\cdot, \cdot, \cdot)}(1^\lambda, \text{pp}_{\mathcal{A}})$ 7 : $\forall (m', S),$ $\text{signers}_{m', S} := \{i : S \in Q[h][m'][i]\}$ 8 : $\text{fullSigs} := \{m' : \exists S \text{ s.t.}$ $ \text{signers}_{m', S} \cup C \geq S \}$ 9 : $\forall i \in [N],$ $\text{partialSigs}_i := \{(m', S) : i \in \text{signers}_{m', S}\}$ 10 : if $ \text{fullSigs} > w$ OR $ \text{signers}_{m, S} \cup C \geq T$ OR $\exists i \in [N] \setminus C, \text{partialSigs}_i > w$: return 0 11 : return $\text{Vf}(1^\lambda, \text{pk}, m, h, \sigma)$	Oracle $\text{OPartialSign}(i, m, h)$ 1 : if $h \notin [\ell]$ OR $i \in C$: return $\perp$ 2 : $Q[h][m] \leftarrow Q[h][m] \cup \{i\}$ 3 : return $\text{PartialSign}(1^\lambda, \text{sk}_i, m, h)$
	Oracle $\text{OPartialSignQuorum}(i, m, h, S)$ 1 : if $h \notin [\ell]$ OR $i \in C$: return $\perp$ 2 : $Q[h][m][i] \leftarrow Q[h][m][i] \cup \{S\}$ 3 : return $\text{PartialSignQuorum}(1^\lambda, \text{sk}_i, m, h, S)$
	The Game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{rb}}(\lambda)$ 1 : $(\ell, w, N, T, \text{state}) \leftarrow \mathcal{A}(1^\lambda)$ 2 : $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w, N, T)$ 3 : $(\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i=1}^N) \leftarrow \text{KGen}(1^\lambda, \text{pp})$ 4 : $\text{pp}_{\mathcal{A}} \leftarrow (\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i=1}^N, \text{state})$ 5 : $(S, m, h, \{\sigma_i\}_{i \in S}) \leftarrow \mathcal{A}(1^\lambda, \text{pp}_{\mathcal{A}})$ 6 : if $ S < T$ OR $\exists i \in S,$ $\text{PartialVerify}(\text{pk}_i, m, h, \sigma_i) = 0$: return 0 7 : $\sigma \leftarrow \text{Combine}(S, m, h, \{\text{pk}_i, \sigma_i\}_{i \in S})$ 8 : if $\text{Vf}(1^\lambda, \text{pk}, m, h, \sigma) = 0$: return 1 9 : return 0

Figure 8: The EUF-CMA adaptive security game and robustness game for a threshold synchronous signature scheme Sig , where $\text{Sig} = (\text{Setup}, \text{KGen}, \text{PartialSign}, \text{PartialSignQuorum}, \text{PartialVerify}, \text{Combine}, \text{Vf})$.

These algorithms must satisfy the following correctness property: for all positive integers $N, T \leq N$, all $x \in R$, every possible output $\{x_i\}_{i \in [N]} \leftarrow \$ \text{Share}(N, T, x)$, and every subset $S \subseteq [N]$ such that $|S| \geq T$, we must have:

$$\text{Combine}(S, \{x_i\}_{i \in S}) = x.$$

A secret sharing scheme is secure if for all positive integers $N, T \leq N$, every $x, x' \in R$, and every set $S \subseteq [N]$ such that $|S| = T - 1$, the following distributions are identical:

$$\begin{aligned} & \{\{x_i\}_{i \in S} : \{x_i\}_{i \in [N]} \leftarrow \$ \text{Share}(N, T, x)\}, \\ & \{\{x'_i\}_{i \in S} : \{x'_i\}_{i \in [N]} \leftarrow \$ \text{Share}(N, T, x')\}. \end{aligned}$$

We now give the Shamir secret sharing scheme for ring elements.

Construction 7.1 (Shamir Secret Sharing Scheme SS for Ring Elements). Let $\mathbf{s}$ be a vector of N distinct elements in R such that for all $i \in [N]$, $\mathbf{s}[i]$ is a unit in R and for all $i, j \in [N], i \neq j$, $\mathbf{s}[i] - \mathbf{s}[j]$ is a unit in R . For any subset $S \subseteq [N]$ and for all $i \in S$, define the Lagrange basis polynomial $\lambda_{S,i}(X) := \prod_{j \in S \setminus \{i\}} \frac{X - \mathbf{s}[j]}{\mathbf{s}[i] - \mathbf{s}[j]}$.

- $\text{Share}(N, T, x) \rightarrow \{x_i\}_{i \in [N]}$:

- Sample random $r_{T-1}, \dots, r_1 \leftarrow \$ R$ and define the *generating polynomial*:

$$f(X) = r_{T-1}X^{T-1} + \dots r_1X + x$$

- For all $i \in [N]$, compute $x_i \leftarrow f(\mathbf{s}[i])$. Output $\{x_i\}_{i \in [N]}$.

- $\text{Combine}(S, \{x_i\}_{i \in [S]}) \rightarrow x$:

- Output $\sum_{i \in S} \lambda_{S,i}(0) \cdot x_i$

We require elements in $\mathbf{s}$ and also non-zero differences between elements in $\mathbf{s}$ to be invertible, so that the Lagrange basis polynomials exist and Combine and DeriveShare succeed. By Theorem 1.1 in [LS18], non-zero polynomials with small coefficients are invertible (over $R = R_q$). Thus, we can choose $\mathbf{s}$ to be the set of non-zero polynomials with small coefficients, so that their non-zero differences also have small coefficients.

In our security proofs for threshold signatures, we use the following alternative but equivalent form of Shamir secret sharing for ring elements.

Construction 7.2 (Alternative Shamir Secret Sharing SS_{alt} for Ring Elements). Define $\mathbf{s}$ and the Lagrange basis polynomials as in [Construction 7.1](#).

- $\text{Share}(N, T, x) \rightarrow \{x_i\}_{i \in [N]}$:

- Set $x_0 := x$, sample $x_1, \dots, x_{T-1} \leftarrow R$, and define the *generating polynomial* f as the unique degree $T - 1$ polynomial that passes through $(0, x)$ and $(s[i], x_i)$ for all $i \in [T - 1]$:

$$f(X) = \sum_{i \in [0, T-1]} \lambda_{[0, T-1], i}(X) \cdot x_i$$

- For all $i \in [T, N]$, compute $x_i \leftarrow f(s[i])$. Output $\{x_i\}_{i \in [N]}$.

- Combine is the same as in [Construction 7.1](#).

Claim 13. *For a fixed N, T, x , the distribution of the output of $\text{Share}(N, T, x)$ from [Construction 7.1](#) is the same as the distribution of the output of $\text{Share}(N, T, x)$ from [Construction 7.2](#).*

Proof. Both Share algorithms define a generating polynomial $f(X) = x + \sum_{i \in [T-1]} r_i \cdot X^i$ and shares $\{x_i\}_{i \in [T-1]}$ such that $x_i = f(s[i])$. We can express this as the following matrix-vector product:

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & s[1] & s[1]^2 & \dots & s[1]^{T-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & s[T-1] & s[T-1]^2 & \dots & s[T-1]^{T-1} \end{bmatrix} \begin{bmatrix} x \\ r_1 \\ \vdots \\ r_{T-1} \end{bmatrix} = \begin{bmatrix} x \\ x_1 \\ \vdots \\ x_{T-1} \end{bmatrix}$$

A matrix of ring elements is invertible if its determinant is a unit in R . The above matrix is a Vandermonde matrix whose determinant is $\prod_{1 \leq j \leq T-1} s[j] \prod_{1 \leq i < j \leq T-1} (s[j] - s[i])$. Because $s[i]$ and $s[j] - s[i]$ for $j \neq i$ are all units, this determinant is also a unit, so the Vandermonde matrix is invertible, and there is a one-to-one mapping between $\{r_i\}_{i \in [T-1]}$ and $\{x_i\}_{i \in [T-1]}$. This means sampling $\{r_i\}_{i \in [T-1]}$ uniformly at random and then solving for $\{x_i\}_{i \in [T-1]}$ is equivalent to sampling $\{x_i\}_{i \in [T-1]}$ uniformly at random and then solving for $\{r_i\}_{i \in [T-1]}$. The former is exactly the Share algorithm in [Construction 7.1](#), and the latter is exactly the Share algorithm in [Construction 7.2](#), so the two methods of sampling are the same. $\square$

7.2 Statically Secure Threshold Synchronous Signature from Basic Synchronous Signature Scheme

[Construction 7.3](#) thresholdizes the basic synchronous signature scheme from [Construction 5.1](#) to obtain a statically secure threshold synchronous signature scheme. The KGen algorithm in the threshold scheme samples a public key-signing key pair using KGen from the basic scheme and then computes signing key shares using Shamir secret sharing of the signing key. For simplicity, we present a scheme that is not robust (i.e., we set $\text{pk}_i := \perp$ and do not define PartialVerify). We can generically transform our scheme into a robust scheme using NIZKs.

Construction 7.3.

- $\text{Setup}(1^\lambda, \ell, w, N, T) \rightarrow \text{pp}$ directly calls the **Setup** algorithm of the basic signature scheme Π_{sync} on $(1^\lambda, \ell, w)$
- $\text{KGen}(1^\lambda, \text{pp})$
 - $(\text{pk}, \text{sk}) \leftarrow \Pi_{\text{sync}}.\text{KGen}(1^\lambda, \text{pp})$
 - $\text{Parse}(\text{H}, \overline{\text{sk}}) \in (\mathcal{H}, R) \leftarrow \text{sk}$
 - $\{\overline{\text{sk}}_i\}_{i \in [N]} \leftarrow \text{SS.Share}(N, T, \overline{\text{sk}})$
 - For $i \in [N]$, $\text{sk}_i \leftarrow (\text{H}, \overline{\text{sk}}_i)$
 - Output $(\text{pk}, \{\perp, \text{sk}_i\}_{i \in [N]})$
- $\text{PartialSign}(\text{sk}_i, m, h)$ directly calls the **Sign** algorithm of the basic signature scheme Π_{sync} on (sk_i, m, h) .
- $\text{Combine}(S, m, h, \{\text{pk}_i, \sigma_i\}_{i \in S})$:
 - If $|S| < T$, output $\perp$
 - Output $\text{SS.Combine}(S, \{\sigma_i\}_{i \in S})$
- $\text{Vf}(\text{pp}, \text{pk}, m, h, \sigma)$ directly calls the **Vf** algorithm of the basic threshold signature scheme Π_{sync}

Figure 9: The w -way ℓ -slot statically secure threshold synchronous signature scheme $\Pi_{\text{thresh}}^{w, \ell, N, T}$.

The statically secure threshold signature scheme. Our statically secure threshold signature scheme is presented in Figure 9 as Construction 7.3. The construction uses the primitives below:

- Π_{sync} is a synchronous signature scheme (as in Definition 5.1) that satisfies homomorphic signing correctness (Definition 5.3) with message space $\mathcal{M}$ and has secret key space $\mathcal{H} \times R$ where $\mathcal{H}$ is a family of hash functions, and R is a ring.
- SS is the Shamir secret sharing scheme for R in Construction 7.1.

Theorem 7.1. *Construction 7.3 is correct (Definition 7.2).*

Proof. Fix any λ, w, ℓ , and $T \leq N$, and $m \in \mathcal{M}, h \in [\ell]$ and any subset $S \subseteq [N]$ such that $|S| \geq T$. Let $\text{pp} \leftarrow \text{Setup}(1^\lambda, \ell, w, N, T)$, $(\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i \in [N]}) \leftarrow \text{KGen}(1^\lambda, \text{pp})$. Let sk

be the corresponding signing key for public key pk produced in KGen . For all $i \in S$, let $\sigma_i \leftarrow \$ \text{PartialSign}(\text{sk}_i, m, h)$, and let $\sigma \leftarrow \$ \text{Combine}(S, m, h, \{\text{pk}_i, \sigma_i\}_{i \in S})$. By the definitions of SS.Combine and σ_i :

$$\sigma = \sum_{i \in S} \lambda_{S,i}(0) \cdot \sigma_i = \sum_{i \in S} \lambda_{S,i}(0) \cdot \Pi_{\text{sync}}.\text{Sign}(\text{sk}_i, m, h).$$

From the correctness of SS , $\text{sk} = \sum_{i \in S} \lambda_{S,i}(0) \cdot \text{sk}_i$, so by the T -homomorphic signing correctness of Π_{sync} (Definition 5.3):

$$\text{Vf}(\text{pp}, \text{pk}, m, h, \sigma) = \Pi_{\text{sync}}.\text{Vf} \left(\text{pp}, \text{pk}, m, h, \sum_{i \in [k']} \lambda_{S,i}(0) \cdot \Pi_{\text{sync}}.\text{Sign}(\text{sk}_i, m, h) \right) = 1.$$

Correctness then follows from correctness of the basic signature scheme Π_{sync} . $\square$

Theorem 7.2. *Construction 7.3 is statically secure. More precisely, for every adversary $\mathcal{A}$ in the static EUF-CMA security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}$ for a threshold synchronous signature scheme (Figure 8), there is a PPT adversary $\mathcal{B}$ that wins the EUF-CMA security game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ for the basic signature scheme Π_{sync} s.t.*

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}(\lambda) \leq \text{Adv}_{\mathcal{B}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$$

Proof. Without loss of generality, we assume that the adversary corrupts parties $1, \dots, T-1$, issues partial signature queries for exactly w *distinct* messages in the forgery slot h , never queries for the partial signature of a corrupted party, and issues only a single partial signature query for each of these w messages⁵. We can fully describe the set of queries made by any such adversary as w partial signature queries, each on a distinct message.

Game G_0 . This is the actual EUF-CMA static security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}$ for a threshold synchronous signature scheme (Figure 8).

Game G_1 . Same as G_0 except that 1) the challenger stores the signing key sk generated in KGen when calling $\Pi_{\text{sync}}.\text{KGen}$ and 2) upon receiving a partial signature query (i, m, h) (where $i \geq T$), instead of computing $\text{PartialSign}(1^\lambda, \text{sk}_i, m, h)$, the challenger does the following:

- For $j \in [T-1]$, $\sigma_j \leftarrow \text{PartialSign}(1^\lambda, \text{sk}_j, m, h)$

⁵We can always convert an adversary $\mathcal{A}$ to $\mathcal{A}'$ that satisfies these properties but has the same winning probability as $\mathcal{A}$. The first, second, and third properties are straightforward to satisfy. For the fourth property, once an adversary has issued a single partial signature query for a given message m , it can derive any other party's partial signature share on m from this single partial signature and the $T-1$ corrupted parties' secret keys by the correctness of Shamir secret sharing.

- $\sigma \leftarrow \Pi_{\text{sync}}.\text{Sign}(\text{sk}, m, h)$
- Let $S = [T - 1] \cup \{i\}$.
- Return $\frac{1}{\lambda_{S,i}(0)} \left(\sigma - \sum_{j \in [T-1]} \lambda_{S,j}(0) \cdot \sigma_j \right)$

By the correctness of the threshold scheme, $\sigma = \sum_{i \in S} \lambda_{S,i}(0) \cdot \sigma_i$, where $\lambda_{S,i}(0)$ exists by construction. This means that first computing σ and then solving for σ_i is equivalent to directly computing σ_i . Thus:

$$\Pr[G_0^{\mathcal{A}} = 1] = \Pr[G_1^{\mathcal{A}} = 1].$$

Game G_2 . G_2 is the same as G_1 except instead of running $\text{SS.Share}(N, T, \overline{\text{sk}})$ in KGen , the challenger runs $\text{SS}_{\text{alt}}.\text{Share}(N, T, \overline{\text{sk}})$ where $(H, \overline{\text{sk}}) \leftarrow \text{sk}$. From Claim 13, $\text{SS.Share}(N, T, \overline{\text{sk}})$ and $\text{SS}_{\text{alt}}.\text{Share}(N, T, \overline{\text{sk}})$ produce identical distributions of secret shares, so:

$$\Pr[G_1^{\mathcal{A}} = 1] = \Pr[G_2^{\mathcal{A}} = 1].$$

Now we show that we can convert any adversary $\mathcal{A}$ for Game G_2 into an adversary $\mathcal{B}$ for EUF-CMA game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ with the same winning probability:

- Upon receiving $(\ell, w, N, T, C, \text{state})$ from $\mathcal{A}$, $\mathcal{B}$ forwards (ℓ, w, state) to the challenger of $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ for the basic scheme.
- Upon receiving pk from the challenger, $\mathcal{B}$ does the following:
 - For all $i \in [T - 1]$, sample $\overline{\text{sk}}_i \leftarrow_{\$} R$.
 - For $i \in [T - 1]$, $\text{sk}_i \leftarrow (H, \overline{\text{sk}}_i)$
 - Send $(\text{pk}, \{\perp\}_{i \in [N]}, \{\text{sk}_i\}_{i \in [T-1]})$ to $\mathcal{A}$
- Upon receiving a partial signature query (i, m, h) where $i \notin [T - 1]$:
 - For $j \in [T - 1]$, $\sigma_j \leftarrow \text{PartialSign}(1^\lambda, \text{sk}_j, m, h)$
 - Query the challenger in $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ for the basic scheme for a signature on (m, h) and let σ be the response.
 - Let $S = [T - 1] \cup \{i\}$.
 - Return $\frac{1}{\lambda_{S,i}(0)} \left(\sigma - \sum_{j \in [T-1]} \lambda_{S,j}(0) \cdot \sigma_j \right)$
- Upon receiving (m, h, σ) from $\mathcal{A}$, return forgery $(1, 1, m, h, \sigma)$ to challenger.

We first argue that $\mathcal{A}$ cannot distinguish between interacting with $\mathcal{B}$ and interacting with the challenger in G_2 . The public key pk received by $\mathcal{A}$ in both interactions is an honestly generated public key for the basic signature scheme Π_{sync} . Moreover, both $\mathcal{B}$ and the challenger in G_2 generate the $T - 1$ secret keys sent to $\mathcal{A}$ by randomly sampling secret keys from R , so these keys are indistinguishable. Lastly, both $\mathcal{B}$ and the challenger in G_2 derive the answer to partial signature query (i, m, h) using the corrupted parties' keys and a full signature on (m, h) . The only difference between $\mathcal{B}$ and the challenger in G_2 when answering partial signature queries is that the challenger in G_2 computes the full signature itself, whereas $\mathcal{B}$ receives the full signature from the challenger in $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$. These full signatures are indistinguishable because they are both honestly generated with respect to the public key given to $\mathcal{A}$ at the beginning of the game. Because the corrupted parties' keys and the full signature on (m, h) have the same distribution in both settings, the partial signature query responses from $\mathcal{B}$ are indistinguishable from the partial signature query responses from the challenger in G_2 .

If $\mathcal{A}$ wins, then by definition $\Pi_{\text{sync}}. \text{Vf}(\text{pk}, m, h, \sigma) = 1$. Moreover, because $\mathcal{A}$ can issue at most w partial signature queries in slot h , $\mathcal{B}$ makes at most w full signature queries to the underlying signature scheme at slot h , so (m, h, σ) is a valid forgery for $\mathcal{B}$ in $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$. $\square$

7.3 Adaptively Secure Threshold Synchronous Signature from Basic Synchronous Signature Scheme

Construction 7.4 thresholdizes the basic synchronous signature scheme from [Construction 5.1](#) to obtain an adaptively secure threshold synchronous signature scheme that uses masks as in [\[Del+24; JWZ26\]](#).

We start by giving some intuition for why [Construction 7.3](#) is not statically secure even for $|C| = T - 2$. If the adversary corrupts only $T - 2$ parties, then it can query up to w signatures for each of the $N - T$ honest parties. This reveals up to $(N - T)w$ linearly independent constraints on the secret keys of honest parties, and we cannot prove security without making additional assumptions about the basic scheme. To achieve adaptive security, each party receives masking key as part of their secret keys and masks any partial signature it produces with a mask derived from this masking key and a hash function $\mathbf{H}_{\text{seed}}$ that we model as a random oracle.

The adaptively secure threshold signature scheme. Our adaptively secure threshold signature scheme is presented in [Figure 10](#) as [Construction 7.4](#). The construction uses the primitives below:

- Π_{sync} is a synchronous signature scheme (as in [Definition 5.1](#)) that satisfies homomorphic signing correctness ([Definition 5.3](#)) with message space $\mathcal{M}$, signature space Σ that is a ring with an identity 1_Σ , and secret key space $\mathcal{H} \times R$ where $\mathcal{H}$ is a family of hash functions and R is a ring.

- A hash function $H_{\text{seed}} : \{0, 1\}^\lambda \times \mathcal{M} \times \mathbb{N} \times Q \rightarrow \Sigma$ where $Q = \{S \subseteq [N] : |S| \geq T\}$
- SS is the Shamir secret sharing scheme for R in [Construction 7.1](#), and $\lambda_{S,i}$ is the Lagrange basis polynomial from [Construction 7.1](#).

Theorem 7.3. *Construction 7.4 is correct (Definition 7.2).*

Proof. Fix any λ, w, ℓ , and $m \in \mathcal{M}, h \in [\ell]$ and any $S \subseteq [N]$ such that $|S| \geq T$. Let $\text{pp} \leftarrow \$ \text{Setup}(1^\lambda, \ell, w, N, T)$, $(\text{pk}, \{\text{pk}_i, \text{sk}_i\}_{i \in [N]}) \leftarrow \text{KGen}(1^\lambda, \text{pp})$. For all $i \in S$, $\sigma_i \leftarrow \text{PartialSignQuorum}(\text{sk}_i, m, h, S)$. Let $\sigma \leftarrow \text{Combine}(S, m, h, \{\text{pk}_i, \sigma_i\}_{i \in S})$. By the definition of Combine and PartialSignQuorum, $\sigma = \sum_{i \in S} \sigma_i$ and

$$\begin{aligned} \sum_{i \in S} \sigma_i &= \sum_{i \in S} \left(\lambda_{S,i}(0) \cdot \Pi_{\text{sync}} \cdot \text{Sign}(\text{sk}_i, m, h) + \sum_{j \in S \setminus \{i\}} H_{\text{seed}}(\text{seed}_{i,j}, m, h, S) - H_{\text{seed}}(\text{seed}_{j,i}, m, h, S) \right) \\ &= \sum_{i \in S} \lambda_{S,i}(0) \cdot \Pi_{\text{sync}} \cdot \text{Sign}(\text{sk}_i, m, h) + \sum_{i \in S} \sum_{j \in S \setminus \{i\}} (H_{\text{seed}}(\text{seed}_{i,j}, m, h, S) - H_{\text{seed}}(\text{seed}_{j,i}, m, h, S)) \\ &= \sum_{i \in S} \lambda_{S,i}(0) \cdot \Pi_{\text{sync}} \cdot \text{Sign}(\text{sk}_i, m, h). \end{aligned}$$

By the correctness of Shamir secret sharing, $\text{sk} = \sum_{i \in S} \lambda_{S,i}(0) \cdot \text{sk}_i$, so by the homomorphic signing correctness of Π_{sync} :

$$\text{Vf} \left(\text{pp}, \text{pk}, m, h, \sum_{i \in [k']} \lambda_{S,i}(0) \cdot \text{Sign}(\text{sk}_i, m, h) \right) = 1.$$

□

Theorem 7.4. *Construction 7.4 is adaptively secure. More precisely, for every adversary $\mathcal{A}$ that makes Q_h queries to the random oracle H_{seed} in the adaptively EUF-CMA security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{ad}}$ for a threshold synchronous signature scheme ([Figure 8](#)), there is a PPT adversary $\mathcal{B}$ that wins the EUF-CMA security game $\mathbf{G}_{\mathcal{B}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ for the basic signature scheme Π_{sync} s.t.*

$$\text{Adv}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}(\lambda) \leq \text{Adv}_{\mathcal{B}, \text{Sig}}^{\text{ssig}, 1}(\lambda) + Q_h(N^2 - N)/2^\lambda$$

Proof. Let C be the current set of corrupted parties. We assume that the adversary does not issue the same query more than once and that it does not issue partial signature queries for parties that it has already corrupted.

Game G_0 . This is the actual EUF-CMA adaptive security game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{tssig}, \text{st}}$ for a threshold synchronous signature scheme ([Figure 8](#)).

Construction 7.4.

- $\text{Setup}(1^\lambda, \ell, w, N, T) \rightarrow \text{pp}$ directly calls the **Setup** algorithm of the basic signature scheme Π_{sync} on $(1^\lambda, \ell, w)$
- $\text{KGen}(1^\lambda, \text{pp})$
 - $(\text{pk}', \text{sk}') \leftarrow \Pi_{\text{sync}}.\text{KGen}(1^\lambda, \text{pp})$
 - $\text{pk} \leftarrow (\text{pk}', \text{H}_{\text{seed}})$
 - $\text{Parse}(\text{H}, \overline{\text{sk}}) \leftarrow \text{sk}'$
 - $\{\overline{\text{sk}}_i\}_{i \in [N]} \leftarrow \text{SS.Share}(N, T, \overline{\text{sk}})$
 - For $i, j \in [N], i \neq j$, sample $\text{seed}_{i,j}, \text{seed}_{j,i} \leftarrow \$ \{0, 1\}^\lambda$
 - For $i \in [N]$, $\text{sk}_i \leftarrow (\text{H}, \overline{\text{sk}}_i, \text{H}_{\text{seed}}, \{\text{seed}_{i,j}, \text{seed}_{j,i}\}_{j \in [N] \setminus \{i\}})$
 - Output $(\text{pk}, \{\perp, \text{sk}_i\}_{i \in [N]})$
- $\text{PartialSignQuorum}(\text{sk}_i, m, h, S)$:
 - $\text{Parse}(\text{H}, \overline{\text{sk}}_i, \text{H}_{\text{seed}}, \{\text{seed}_{i,j}, \text{seed}_{j,i}\}_{j \in [N] \setminus \{i\}}) \leftarrow \text{sk}_i$
 - $\overline{\sigma}_i \leftarrow \Pi_{\text{sync}}.\text{Sign}((\text{H}, \overline{\text{sk}}_i), m, h)$
 - Output

$$\sigma_i \leftarrow \lambda_{S,i}(0) \cdot \overline{\sigma}_i + \sum_{j \in S \setminus \{i\}} (\text{H}_{\text{seed}}(\text{seed}_{i,j}, m, h, S) - \text{H}_{\text{seed}}(\text{seed}_{j,i}, m, h, S))$$
- $\text{Combine}(S, m, h, \{\text{pk}_i, \sigma_i\}_{i \in S})$:
 - If $|S| < T$, output $\perp$
 - Output $\sum_{i \in S} \sigma_i$
- $\text{Vf}(\text{pk}, m, h, \sigma)$:
 - $\text{Parse}(\text{pk}', \text{H}_{\text{seed}}) \leftarrow \text{pk}$
 - Return $\Pi_{\text{sync}}.\text{Vf}(\text{pk}', m, h, \sigma)$

Figure 10: The w -way ℓ -slot adaptively secure threshold synchronous signature scheme $\Pi_{\text{thresh}}^{w, \ell, N, T}$.

G_1 1 : $H \leftarrow []$ $\quad \quad \quad \text{// Rest of game setup is same as in } G_0$ Oracle $\text{OH}_{\text{seed}}(\text{seed}, m, h, S)$ 1 : if $(\text{seed}, m, h, S) \notin H,$ 2 : $H[(\text{seed}, m, h, S)] := r \leftarrow_{\$} \Sigma$ 3 : return $H[(\text{seed}, m, h, S)]$	G_2 Oracle $\text{OH}_{\text{seed}}(\text{seed}, m, h, S)$ 1 : if $\exists i, j \in [N] \text{ s.t. } \text{seed} = \text{seed}_{i,j}$ $\quad \quad \quad \text{AND } \{i, j\} \cap C = \emptyset, \text{ return } 0$ 2 : if $(\text{seed}, m, h, S) \notin H,$ 3 : $H[(\text{seed}, m, h, S)] := r \leftarrow_{\$} \Sigma$ 4 : return $H[(\text{seed}, m, h, S)]$
G_3 Oracle $\text{OPartialSignQuorum}(i, m, h, S)$ 1 : if $h \notin [\ell] \text{ OR } i \in C : \text{ return } \perp$ 2 : $\text{signers} := \{i : S \in Q[h][m][i]\}$ 3 : $Q[h][m][i] \leftarrow Q[h][m][i] \cup \{S\}$ 4 : if $ \text{signers} \cup (S \cap C) < S - 1 :$ 5 : return $\text{PartialSignQuorum}(\text{sk}_i, m, h, S)$ 6 : $\forall j \in S \setminus \{i\} :$ 7 : $\sigma_j \leftarrow \text{PartialSignQuorum}(\text{sk}_j, m, h, S)$ 8 : $\sigma \leftarrow \Pi_{\text{sync}}.\text{Sign}(\text{sk}, m, h)$ 9 : return $\sigma - \sum_{j \in S \setminus \{i\}} \sigma_j$	G_4 1 : $P \leftarrow \{ \}$ $\quad \quad \quad \text{// Rest of game setup is same as in } G_3$ Sample(m, h, S) 1 : $P := P \cup \{(m, h, S)\}$ 2 : $\forall i \in S, j \in S \setminus \{k\} :$ 3 : $H[(\text{seed}_{i,j}, m, h, S)] := r \leftarrow_{\$} \Sigma$ 4 : $H[(\text{seed}_{j,i}, m, h, S)] := r \leftarrow_{\$} \Sigma$ Oracle $\text{OH}_{\text{seed}}(\text{seed}, m, h, S)$ 1 : if $\exists i, j \in [N] \text{ s.t. } \text{seed} = \text{seed}_{i,j}$ $\quad \quad \quad \text{AND } \{i, j\} \cap C = \emptyset, \text{ return } 0$ 2 : if $(m, h, S) \notin P, \text{ Sample}(m, h, S)$ 3 : $\quad \quad \quad \text{// Same as } G_3$ Oracle $\text{OPartialSignQuorum}(i, m, h, S)$ 1 : if $h \notin [\ell] \text{ OR } i \in C : \text{ return } \perp$ 2 : if $(m, h, S) \notin P, \text{ Sample}(m, h, S)$ 3 : $\quad \quad \quad \text{// Same as } G_3$

Figure 11: Hybrid games used in proof of Theorem 7.2. Differences from G_i to G_{i+1} are highlighted. If a function is not explicitly included in G_i , it is the same as in G_{i-1} .

Game G_1 . This is the same as G_0 except that the challenger lazily samples the random oracle entries. The challenger maintains a map H , and upon receiving a random oracle query $H_{\text{seed}}(\text{seed}, m, h, S)$ (either from the adversary or within the challenger's execution of `PartialSignQuorum`), the challenger checks if $(\text{seed}, m, h, S) \in H$. If not, it sets $H[(\text{seed}, m, h, S)] := r \leftarrow \$ \Sigma$. Then it returns $H[(\text{seed}, m, h, S)]$.

Because the challenger randomly samples all H_{seed} responses and returns the same response for a given (seed, m, h, S) , the distribution of H_{seed} responses is the same in G_0 and G_1 , so:

$$\Pr[G_0^A = 1] = \Pr[G_1^A = 1].$$

Game G_2 . This is the same as G_1 except that if the adversary queries the random oracle H_{seed} at some $(\text{seed}_{i,j}, m, h, S)$ before party i or j is corrupted, then G_2 returns 0. Because each of the $N^2 - N$ seed values is sampled uniformly from $\{0, 1\}^\lambda$, the probability that the adversary guesses the seed value of an honest party in a single random oracle query is $\leq (N^2 - N)/2^\lambda$. Because the adversary makes at most Q_h random oracle queries:

$$\left| \Pr[G_1^A = 1] - \Pr[G_2^A = 1] \right| \leq Q_h(N^2 - N)/2^\lambda.$$

Game G_3 . This is the same as G_2 except for how the challenger responds to a partial signature query (i, m, h, S) . Let $\text{signers} \subseteq S$ be the set of parties (excluding i) for which the adversary has issued partial signature queries on (m, h, S) . If the adversary already has $|S| - 1$ partial signatures on (m, h, S) (i.e., $|\text{signers} \cup (S \cap C)| = |S| - 1$), then the challenger computes partial signature $\sigma_j \leftarrow \text{PartialSignQuorum}(i, m, h, S)$ for all $j \in \text{signers} \cup (S \cap C)$ and $\sigma \leftarrow \Pi_{\text{sync}}.\text{Sign}(\text{sk}, m, h)$, and it sends $\sigma - \sum_{j \in S \setminus \{i\}} \sigma_j$ to the adversary. By the correctness of the scheme, this response is equivalent to the response sent by the G_2 challenger, so:

$$\Pr[G_2^A = 1] = \Pr[G_3^A = 1].$$

Game G_4 . This is the same as G_3 except that the challenger samples additional H_{seed} values before the adversary queries for them. Let P be a set maintained by the challenger that keeps track of all (m, h, S) for which the adversary has issued a partial signature query or H_{seed} query. If the adversary issues a partial signature query or H_{seed} query for some $(m, h, S) \notin P$, then the challenger samples all $\{H_{\text{seed}}(\text{seed}_{i,j}, m, h, S), H_{\text{seed}}(\text{seed}_{j,i}, m, h, S)\}_{i \in S, j \in S \setminus \{i\}}$. Because the challenger checks if $(m, h, S) \in P$ before sampling, it never re-samples any H_{seed} values. Sampling extra H_{seed} values does not change the adversary's winning probability, so:

$$\Pr[G_3^A = 1] = \Pr[G_4^A = 1].$$

G_5	
Retrieve ($m, h, S, \mathbf{a}$) 1: return $\mathbf{a}^\top X^{m,h,S}$	Sign (i, m, h, S) 1: return SignWithoutMask (i, m, h, S) + Retrieve ($m, h, S, \mathbf{a}_i^S$)
SignWithoutMask (i, m, h, S) 1: $(H, \overline{\mathbf{sk}}_i, \cdot, \cdot) \leftarrow \mathbf{sk}_i$ 2: $\overline{\sigma}_i \leftarrow \Pi_{\text{sync}}.\text{Sign}((H, \overline{\mathbf{sk}}_i), m, h)$ 3: return $\lambda_{S,i}(0)\overline{\sigma}_i$	Oracle OPartialSignQuorum (i, m, h, S) 1: if $h \notin [\ell]$ OR $i \in C$: return $\perp$ 2: if $(m, h, S) \notin P$, Sample (m, h, S) 3: signers := $\{i : S \in Q[h][m][i]\}$ 4: $Q[h][m][i] \leftarrow Q[h][m][i] \cup \{S\}$ 5: if $ \text{signers} \cup (S \cap C) < S - 1$: 6: return Sign (i, m, h, S) 7: $\forall j \in S \setminus \{i\}$: 8: $\sigma_j \leftarrow \text{Sign}(j, m, h, S)$ 9: $\sigma \leftarrow \Pi_{\text{sync}}.\text{Sign}(\mathbf{sk}, m, h)$ 10: return $\sigma - \sum_{j \in S \setminus \{i\}} \sigma_j$
Oracle OH_{seed} (seed, m, h, S) 1: if $\exists i, j \in [N]$ s.t. $\text{seed} = \text{seed}_{i,j}$ AND $\{i, j\} \cap C = \emptyset$, return 0 2: if $(m, h, S) \notin P$, Sample (m, h, S) 3: if $(\text{seed}, m, h, S) \notin H$, 4: $H[(\text{seed}, m, h, S)] := r \leftarrow \$ \Sigma$ 5: if $\exists i, j \in [N]$ s.t. $\text{seed} = \text{seed}_{i,j}$: return Retrieve ($m, h, S, \mathbf{a}_{i,j}^S$) 6: return $H[(\text{seed}, m, h, S)]$	

Figure 12: Hybrid games used in proof of Theorem 7.2. Differences from G_i to G_{i+1} are highlighted. If a function is not explicitly included in G_i , it is the same as in G_{i-1} .

Game G_5 . For a tuple (m, h, S) where $S = \{s_1, \dots, s_{|S|}\}$, we denote $H_{\text{seed}}(\text{seed}_{i,j}, m, h, S)$ as $H_{i,j}^{m,h,S}$ and let

$$X^{m,h,S} := (H_{s_1,s_2}^{m,h,S}, H_{s_1,s_3}^{m,h,S}, \dots, H_{s_2,s_1}^{m,h,S}, H_{s_2,s_3}^{m,h,S}, \dots, H_{s_{|S|-1},s_{|S|}}^{m,h,S}).$$

Let $\mathbf{a}_{i,j}^S \in \Sigma^{|S|^2-|S|}$ be the standard basis vector that “selects” $H_{i,j}^{m,h,S}$; that is, $(\mathbf{a}_{i,j}^S)^\top X^{m,h,S} = H_{i,j}^{m,h,S}$. Let $\mathbf{a}_i \in \Sigma^{|S|^2-|S|}$ be a *partial signature query vector* defined as $\mathbf{a}_i^S = \sum_{j \in S \setminus \{i\}} \mathbf{a}_{i,j}^S - \mathbf{a}_{j,i}^S$. The game G_5 is defined similarly as in G_4 except for the following:

- When the challenger needs to answer an H_{seed} -query on input $(\text{seed}_{i,j}, m, h, S)$, it returns $(\mathbf{a}_{i,j}^S)^\top X^{m,h,S}$ instead;
- Upon receiving a **PartialSignQuorum** query, the challenger sends $\lambda_{S,i}(0)\bar{\sigma}_i + (\mathbf{a}_i^S)^\top X^{m,h,S}$ where $\bar{\sigma}_i$ is generated as in the first two bullet points of **PartialSignQuorum**. We implement this using the following helper functions: (i) **Retrieve** $(m, h, S, \mathbf{a})$ returns $\mathbf{a}^\top X^{m,h,S}$; (ii) **SignWithoutMask** (i, m, h, S) returns $\lambda_{S,i}(0)\bar{\sigma}_i$; and (iii) **Sign** (i, m, h, S) returns $\lambda_{S,i}(0)\bar{\sigma}_i + (\mathbf{a}_i^S)^\top X^{m,h,S}$ by calling **SignWithoutMask** and **Retrieve**.

When sending the adversary any $H_{\text{seed}}(\text{seed}_{i,j}, m, h, S)$ value, both the G_4 and G_5 challengers respond with $H_{\text{seed}}(\text{seed}_{i,j}, m, h, S) = \mathbf{a}_{i,j}^\top X^{m,h,S}$. By the definition of all partial signature query vectors $\mathbf{a}_i^S$, the G_4 and G_5 challengers give the same responses to partial signature queries, so:

$$\Pr[G_4^{\mathcal{A}} = 1] = \Pr[G_5^{\mathcal{A}} = 1].$$

Game G_6 . This is the same as G_5 except that whenever the challenger adds a new tuple (m, h, S) to P , it initializes an empty matrix $\mathbf{A}^{m,h,S} \in \Sigma^{0 \times |S|^2-|S|}$, and to answer any call **Retrieve** $(m, h, S, \mathbf{a})$, instead of directly returning $\mathbf{a}^\top X^{m,h,S}$ as in G_5 , the challenger adds some rows to $\mathbf{A}^{m,h,S}$ and returns $\mathbf{a}^\top X^{m,h,S}$ only if $\mathbf{a}$ is in the span of $\text{rows}(\mathbf{A}^{m,h,S})$, or, equivalently, if there is some $\mathbf{w}$ such that $\mathbf{w}^\top \mathbf{A}^{m,h,S} = \mathbf{a}^\top$. We thus need to construct each $\mathbf{A}^{m,h,S}$ so that every call to **Retrieve** $(m, h, S, \mathbf{a})$ in G_6 returns $\mathbf{a}^\top X^{m,h,S}$.

The G_6 challenger only makes calls of the form **Retrieve** $(m, h, S, \mathbf{a})$ where $\mathbf{a} = \mathbf{a}_{i,j}^S$ and either i or j is corrupted or $\mathbf{a} = \mathbf{a}_i^S$ where the adversary has either corrupted party i or issued a partial signature query (i, m, h, S) . To ensure that **Retrieve** $(m, h, S, \mathbf{a})$ always returns $\mathbf{a}^\top X^{m,h,S}$, we must have that for all corrupted parties $i \in C$, all $\{\mathbf{a}_{i,j}^S, \mathbf{a}_{j,i}^S\}_{j \in S \setminus \{i\}}$ are in the span of $\text{rows}(\mathbf{A}^{m,h,S})$ and that for all i that are either corrupt or for which the adversary has issued the partial signature query (i, m, h, S) , $\mathbf{a}_i^S$ is in the span of $\text{rows}(\mathbf{A}^{m,h,S})$. We also want to ensure that we do not add new rows to $\mathbf{A}^{m,h,S}$ that are linearly dependent on the rows already in $\mathbf{A}^{m,h,S}$. The challenger does the following:

- Whenever the challenger adds an entry (m, h, S) to P , it initializes an empty matrix $\mathbf{A}^{m,h,S}$, and for all corrupted parties $i \in C$, it adds rows $\{\mathbf{a}_{i,j}^S, \mathbf{a}_{j,i}^S\}_{j \in S \setminus \{i\}}$.

G_6

AddRows($m, h, S, \{\mathbf{a}_i\}_{i \in [n]}$)

1 : **for** $i \in [n]$: Add $\mathbf{a}_i$ as row to $\mathbf{A}^{m,h,S}$

Retrieve($m, h, S, \mathbf{a}$)

1 : $\text{// if } \mathbf{a} \text{ is in the span of } \text{rows}(\mathbf{A}^{m,h,S})$

2 : **if** $\exists \mathbf{w} \text{ s.t. } \mathbf{w}^\top \mathbf{A}^{m,h,S} = \mathbf{a}^\top$:

3 : **return** $\mathbf{w}^\top \mathbf{A}^{m,h,S} X^{m,h,S}$

4 : **else** abort

Oracle OCorrupt(i)

1 : **if** $|C| \geq T$: **return** $\perp$

2 : $C \leftarrow C \cup \{i\}$

3 : $\forall (m, h, S) \in P$:

4 : $\text{rows} := \{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus (C \cup \{i\})}$

5 : $\text{// signers} = \{i \in S : \text{adv issued}\}$
 $\text{// sig query } (m, h, S, i)$

6 : $\text{signers} := \{i : S \in Q[h][m][i]\}$

7 : **if** $i \notin \text{signers}$ OR

$|\text{signers} \cup (S \cap C)| < |S|$:

8 : $\text{AddRows}(m, h, S, \text{rows})$

9 : **else** :

10 : **if** $|S \setminus (\text{signers} \cup C)| = 0$:

$k := \min(S \setminus C)$

11 : **else** $k := \min(S \setminus (\text{signers} \cup C))$

12 : $\text{rows} := \text{rows} \setminus \{\mathbf{a}_{k,i}\}$

13 : $\text{AddRows}(m, h, S, \text{rows})$

14 : **return** sk_i

Sample(m, h, S)

1 : $P := P \cup \{(m, h, S)\}$

2 : $\forall i \in S, j \in S \setminus \{k\}$:

3 : $H[(\text{seed}_{i,j}, m, h, S)] := r \leftarrow \$ \Sigma$

4 : $H[(\text{seed}_{j,i}, m, h, S)] := r \leftarrow \$ \Sigma$

5 : Initialize empty matrix $\mathbf{A}^{m,h,S}$

6 : $\text{AddRows}(m, h, S, \{\mathbf{a}_{i,j}^S,$

$\mathbf{a}_{j,i}^S\}_{i \in S \cap C, j \in S \setminus \{i\})$

Oracle OH_{seed}(seed, m, h, S)

1 : **if** $\exists i, j \in [N] \text{ s.t. } \text{seed} = \text{seed}_{i,j}$
AND $\{i, j\} \cap C = \emptyset$, **return** 0

2 : **if** $(m, h, S) \notin P$, $\text{Sample}(m, h, S)$

3 : **if** $(\text{seed}, m, h, S) \notin H$,

4 : $H[(\text{seed}, m, h, S)] := r \leftarrow \$ \Sigma$

5 : **if** $\exists i, j \in [N] \text{ s.t. } \text{seed} = \text{seed}_{i,j}$
return $\text{Retrieve}(m, h, S, \mathbf{a}_{i,j}^S)$

6 : **return** $H[(\text{seed}, m, h, S)]$

Oracle OPartialSignQuorum(i, m, h, S)

1 : **if** $h \notin [\ell]$ OR $i \in C$: **return** $\perp$

2 : **if** $(m, h, S) \notin P$, $\text{Sample}(m, h, S)$

3 : $\text{signers} := \{i : S \in Q[h][m][i]\}$

4 : $Q[h][m][i] \leftarrow Q[h][m][i] \cup \{S\}$

5 : $\text{// if adv has } < |S| - 1 \text{ partial sigs on}$
 $\text{// } (m, h, S)$

6 : **if** $|\text{signers} \cup (S \cap C)| < |S| - 1$:

7 : $\text{AddRows}(m, h, S, \mathbf{a}_i^S)$

8 : **return** $\text{Sign}(i, m, h, S)$

9 : $\forall j \in S \setminus \{i\}$:

10 : $\sigma_j \leftarrow \text{Sign}(j, m, h, S)$

11 : $\sigma \leftarrow \Pi_{\text{sync}}.\text{Sign}(\text{sk}, m, h)$

12 : **return** $\sigma - \sum_{j \in S \setminus \{i\}} \sigma_j$

Figure 13: Hybrid game G_6 used in proof of Theorem 7.2. Differences from G_i to G_{i+1} are highlighted. If a function is not explicitly included in G_i , it is the same as in G_{i-1} .

- When the adversary makes a partial signature query (i, m, h, S) , we need to ensure that $\mathbf{a}_i^S$ is in the span of $\text{rows}(\mathbf{A}^{m,h,S})$. If the adversary already has $< |S| - 1$ partial signatures on (m, h, S) , the challenger adds $\mathbf{a}_i^S$ as a row to $\mathbf{A}^{m,h,S}$. Else, all $\{\mathbf{a}_j^S\}_{j \in S \setminus \{i\}}$ should be in the span of $\text{rows}(\mathbf{A})$ from prior corruption and partial signature queries, so $\mathbf{a}_i^S$ is already in the span of $\text{rows}(\mathbf{A}^{m,h,S})$ because $\mathbf{a}_i^S = -\sum_{j \in S \setminus \{i\}} \mathbf{a}_j^S$.
- When the adversary corrupts a party i , for every $(m, h, S) \in P$, we want to ensure all $\{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus \{i\}}$ are in the span of $\text{rows}(\mathbf{A}^{m,h,S})$. $\{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \cap C \setminus \{i\}}$ should already be in the span of $\text{rows}(\mathbf{A}^{m,h,S})$. If the adversary has not issued a partial signature query (i, m, h, S) , then the adversary just adds all $\{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus C}$ as rows to $\mathbf{A}^{m,h,S}$. Else, if adversary has issued a partial signature query (i, m, h, S) or it already has $|S| - 1$ partial signatures on (m, h, S) , then $\mathbf{a}_i^S$ should be in the span of $\text{rows}(\mathbf{A}^{m,h,S})$. The challenger cannot just add all $\{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus C}$ as rows to $\mathbf{A}^{m,h,S}$ because there is a linear dependency between $\mathbf{a}_i^S$ and $\{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus \{i\}}$. As such, the challenger picks an honest party k and adds all $\{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus C}$ as rows to $\mathbf{A}^{m,h,S}$ except for $\mathbf{a}_{k,i}$. While $\mathbf{a}_{k,i} \notin \text{rows}(\mathbf{A}^{m,h,S})$, it is clearly in the span of $\text{rows}(\mathbf{A}^{m,h,S})$.

To choose k , the challenger does the following. Let K be the set of honest parties for which the adversary has not issued partial signature queries on (m, h, S) . If $G \neq \emptyset$, the challenger sets $k := \min(K)$. Else, the challenger defines $K := S \setminus C$ and $k := \min(K)$.

- When answering a $H_{\text{seed}}(\text{seed}_{i,j}, m, h, S)$ query, the challenger calls $\text{Retrieve}(m, h, S, \mathbf{a}_{i,j})$. Because we must have that i or j is corrupt, then given the procedures described above, $\mathbf{a}_{i,j}$ is in the span of $\text{rows}(\mathbf{A}^{m,h,S})$, so $\text{Retrieve}(m, h, S, \mathbf{a}_{i,j})$ returns $(\mathbf{a}_{i,j})^\top X^{m,h,S}$.

If we construct every matrix $\mathbf{A}^{m,h,S}$ as described above, then every call made to $\text{Retrieve}(m, h, S, \mathbf{a})$ returns $\mathbf{a}^\top X^{m,h,S}$, so G_6 and G_5 have the same success probability, that is,

$$\Pr[G_5^A = 1] = \Pr[G_6^A = 1].$$

We will use the following lemma in proving that G_6 and G_7 are indistinguishable.

Lemma 7.5. *For every $(m, h, S) \in P$, $\text{rows}(\mathbf{A}^{m,h,S})$ form a partial basis of $\Sigma^{|S|^2 - |S|}$.*

Proof. We defer the proof to [Appendix C](#). □

Game G_7 . This is the same as G_6 except that in addition to storing a matrix $\mathbf{A}^{m,h,S}$ for each $(m, h, S) \in P$, the challenger also stores a column vector $\mathbf{v}^{m,h,S}$. Every time the challenger adds a row to $\mathbf{A}^{m,h,S}$, it samples $r \leftarrow \Sigma$ and appends r to $\mathbf{v}^{m,h,S}$. Whenever the G_7 challenger calls $\text{Retrieve}(m, h, S, \mathbf{a})$, instead of returning $\mathbf{w}^\top \mathbf{A}^{m,h,S} X^{m,h,S}$, the challenger returns $\mathbf{w}^\top \mathbf{v}^{m,h,S}$. Effectively, the challenger is directly sampling $\mathbf{v}^{m,h,S} = \mathbf{A}^{m,h,S} X^{m,h,S}$ rather than first sampling $X^{m,h,S}$ and then computing $\mathbf{A}^{m,h,S} X^{m,h,S}$.

G_7 AddRows($m, h, S, \{\mathbf{a}_i\}_{i \in [n]}$) 1: Add all $\{\mathbf{a}_i\}_{i \in [n]}$ as rows to $\mathbf{A}^{m,h,S}$ 2: for $i \in [n]$, $r_i \leftarrow \\$ \Sigma$ 3: Add $\{r_i\}_{i \in [n]}$ as separate entries to $\mathbf{v}^{m,h,S}$ Retrieve($m, h, S, \mathbf{a}$) 1: if $\exists \mathbf{w}$ s.t. $\mathbf{w}^\top \mathbf{A}^{m,h,S} = \mathbf{a}^\top$: 2: return $\mathbf{w}^\top \mathbf{v}^{m,h,S}$ 3: else abort Sample(m, h, S) 1: $P := P \cup \{(m, h, S)\}$ 2: $\forall i \in S, j \in S \setminus \{k\}$: 3: $H[(\text{seed}_{i,j}, m, h, S)] := r \leftarrow \\$ \Sigma$ 4: $H[(\text{seed}_{j,i}, m, h, S)] := r \leftarrow \\$ \Sigma$ 5: Initialize empty matrix $\mathbf{A}^{m,h,S}$ 6: Initialize empty vector $\mathbf{v}^{m,h,S}$ 7: $\parallel$ Same as G_6	G_8 1: Sigs $\leftarrow []$ 2: $\parallel$ Rest of game setup is same as in G_7 Retrieve($m, h, S, \mathbf{a}$) 1: Find $\mathbf{w}$ s.t. $\mathbf{w}^\top \mathbf{A}^{m,h,S} = \mathbf{a}^\top$ 2: return $\mathbf{w}^\top \mathbf{v}^{m,h,S}$ AddRows($m, h, S, \{\mathbf{a}_i\}_{i \in [n]}$) 1: Add all $\{\mathbf{a}_i\}_{i \in [n]}$ as rows to $\mathbf{A}^{m,h,S}$ 2: for $i \in [n]$: 3: if $\exists j \in S$ s.t. $\mathbf{a}_i = \mathbf{a}_j^S$ 4: $\sigma_i \leftarrow \\$ \Sigma$ 5: Sigs[i, m, h, S] $\leftarrow \sigma_i$ 6: $r_i \leftarrow \sigma_i -$ SignWithoutMask(i, m, h, S) 7: else $r_i \leftarrow \\$ \Sigma$ 8: Add $\{r_i\}_{i \in [n]}$ as separate entries to $\mathbf{v}^{m,h,S}$ Sign(i, m, h, S) 1: return Sigs[i, m, h, S]
--	---

Figure 14: Hybrid games used in proof of Theorem 7.2.

By Lemma 7.5, the rows of $\mathbf{A}^{m,h,S}$ for all $(m, h, S) \in P$ form a partial basis of $\Sigma^{|S|^2 - |S|}$. Lemma 7.6 states that if we have a vector X of n uniformly random variables over some ring R and an $m \times n$ matrix $\mathbf{A} \in R^{m \times n}$, $m \leq n$ whose rows form a partial basis of R^n , then $\mathbf{A}X$ is a vector of m uniformly random variables from R . This directly implies that the manner of responding to the adversary's queries in G_7 gives the same distribution of responses as in G_6 , so:

$$\Pr[G_6^{\mathcal{A}} = 1] = \Pr[G_7^{\mathcal{A}} = 1].$$

Lemma 7.6. *For a ring R and parameter $n \in \mathbb{N}$, define uniformly random variables $X_1, \dots, X_n \leftarrow_{\$} R$, and let $X := (X_1, \dots, X_n)$. Let $\mathbf{A} \in R^{m \times n}$, $m \leq n$ be a matrix whose rows form a partial basis for R^n . Then $\mathbf{A}X$ is identically distributed to $X_1, \dots, X_m$.*

Proof. If the rows of $\mathbf{A}$ form a partial basis of R , then that means we can add $n - m$ rows to $\mathbf{A}$ to get $\mathbf{A}'$ such that the rows of $\mathbf{A}'$ form a basis of R^n . Because the rows of $\mathbf{A}'$ form a basis of R^n , for every standard basis vector $\mathbf{e}_i$, there is some row vector $\mathbf{v}_i \in R^n$ such that $\mathbf{v}_i \mathbf{A}' = \mathbf{e}_i$. Let $\mathbf{B}$ be the matrix whose i -th row is the vector $\mathbf{v}_i$ such that $\mathbf{v}_i \mathbf{A}' = \mathbf{e}_i$. Clearly $\mathbf{B} \mathbf{A}' = I_n$, so $\mathbf{B}$ is the inverse of $\mathbf{A}'$. Let $Y \in R^n = \mathbf{A}' X$, or equivalently, $\mathbf{B} Y = X$. Because $\mathbf{A}'$ is invertible, there is a one-to-one mapping between X and Y , so sampling X uniformly at random is equivalent to sampling Y uniformly at random. $\square$

Game G_8 . This is the same as G_7 except that in $\text{Retrieve}(m, h, S, \mathbf{a})$, we remove the check on whether there exists $\mathbf{w}$ such that $\mathbf{w}^\top \mathbf{A}^{m,h,S} = \mathbf{a}^\top$. As argued earlier, $\mathbf{w}$ always exists and the challenger never aborts when calling Retrieve , so this conditional is unnecessary. Furthermore, the challenger maintains a map Sigs to store partial signatures, and when the challenger answers a partial signature query for party i on (m, h, S) :

- If the adversary has $< |S| - 1$ signatures on (m, h, S) , then instead of appending a random value to $\mathbf{v}^{m,h,S}$ in $\text{AddRows}(m, h, S, \mathbf{a}_i^S)$, the challenger samples a random partial signature $\sigma_i \leftarrow_{\$} R$, stores $\text{Sigs}[i, m, h, S] = \sigma_i$, and appends $\sigma_i - \lambda_{S,i}(0) \bar{\sigma}_i$ to $\mathbf{v}^{m,h,S}$. This sets $\text{Retrieve}(m, h, S, \mathbf{a}_i^S) = \sigma_i - \lambda_{S,i}(0) \bar{\sigma}_i$. The challenger sends σ_i to the adversary as the partial signature for party i .
- Else, the challenger computes $\sigma \leftarrow \Pi_{\text{sync}}.\text{Sign}(\text{sk}, m, h)$ and returns $\sigma - \sum_{j \in S \setminus \{i\}} \sigma_j$ where the σ_j 's are values stored in Sigs .

For the first bullet, sampling σ_i and then deriving r_i gives the same distribution as sampling r_i and then computing σ_i , which is what the G_7 challenger does. For the second bullet, this is exactly what the G_7 challenger does by the correctness of the scheme, so:

$$\Pr[G_7^{\mathcal{A}} = 1] = \Pr[G_8^{\mathcal{A}} = 1].$$

G_9	
1 : $M \leftarrow []$	Oracle $\text{OCorrupt}(i)$
2 : $\quad \text{// Rest same as in } G_8$	1 : if $ C \geq T$: return $\perp$
$\text{AddRows}(m, h, S, \{\mathbf{a}_i\}_{i \in [n]})$	2 : $C \leftarrow C \cup \{i\}$
1 : Add all $\{\mathbf{a}_i\}_{i \in [n]}$ as rows to $\mathbf{A}^{m, h, S}$	3 : $\forall (m, h, S) \in P$:
2 : for $i \in [n]$:	4 : $\text{rows} := \{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus (C \cup \{i\})}$
3 : if $\exists j \in S$ s.t. $\mathbf{a}_i = \mathbf{a}_j^S$	5 : $\text{signers} := \{i : S \in Q[h][m][i]\}$
4 : $\sigma_i \leftarrow \$ \Sigma$	6 : if $i \notin \text{signers}$ OR $ \text{signers} \cup C < S $
5 : $\text{Sigs}[i, m, h, S] \leftarrow \sigma_i$	7 : $\text{AddRows}(m, h, S, \text{rows})$
6 : $r_i \leftarrow 0_\Sigma$	8 : else :
7 : else	9 : if $ S \setminus (\text{signers} \cup C) = 0$:
8 : $r_i \leftarrow \$ \Sigma$	10 : $k := \min(S \setminus C)$
9 : Find $j, k \in S$ s.t. $\mathbf{a}_{j,k}^S = \mathbf{a}_i$	11 : else $k := \min(S \setminus (\text{signers} \cup C))$
10 : $M[m, h, S, \mathbf{a}_{j,k}^S] := r_i$	12 : $\text{rows} := \text{rows} \setminus \{\mathbf{a}_{k,i}\}$
11 : Add $\{r_i\}_{i \in [n]}$ as separate entries to $\mathbf{v}^{m, h, S}$	13 : $\text{AddRows}(m, h, S, \text{rows})$
$\text{Retrieve}(m, h, S, \mathbf{a})$	14 : $x \leftarrow \text{Sigs}[i, m, h, S]$
1 : if $\exists i, j \in S$ such that $\mathbf{a}_{i,j}^S = \mathbf{a}$	$\quad - \text{SignWithoutMask}(i, m, h, S)$
2 : return $M[m, h, S, \mathbf{a}_{i,j}^S]$	15 : $M[m, h, S, \mathbf{a}_{k,i}^S] :=$
3 : return $\perp$	$\quad \text{Retrieve}(m, h, S, \mathbf{a}_{i,k}) - x$
	$\quad + \sum_{j \in S \setminus \{i,k\}} \text{Retrieve}(m, h, S, \mathbf{a}_{i,j})$
	$\quad - \sum_{j \in S \setminus \{i,k\}} \text{Retrieve}(m, h, S, \mathbf{a}_{j,i})$
	16 : return sk_i

Figure 15: Hybrid games used in proof of Theorem 7.2. Differences from G_i to G_{i+1} are highlighted. If a function is not explicitly included in G_i , it is the same as in G_{i-1} .

Game G_9 . This is the same as G_8 except that the challenger maintains a map M to store values returned by `Retrieve`. (Note that `Retrieve` is now only called by OH_{seed} , with inputs of the form $(m, h, S, \mathbf{a}_{i,j}^S)$.) The challenger also “lazily” samples entries of $\mathbf{v}^{m,h,S}$ when calling `AddRows` on partial signature query vectors. The map M and the entries of $\mathbf{v}^{m,h,S}$ are maintained as follows:

- If the challenger adds a standard basis vector $\mathbf{a}_{i,j}^S$ (as a row) to $\mathbf{A}^{m,h,S}$, it appends a random $r \leftarrow_{\$} \Sigma$ to $\mathbf{v}^{m,h,S}$ and sets $M[m, h, S, \mathbf{a}_{i,j}^S] := r$.
- If the challenger adds a partial signature query vector $\mathbf{a}_i^S$ (as a row) to $\mathbf{A}^{m,h,S}$, it appends 0_Σ to $\mathbf{v}^{m,h,S}$.
- If the adversary corrupts a party i , for all $(m, h, S) \in P$, if it does not add $\mathbf{a}_{k,i}$ as a row to $\mathbf{A}^{m,h,S}$, then it computes $x = \text{Sigs}[i, m, h, S] - \text{SignWithoutMask}(i, m, h, S)$ and sets $M[m, h, S, \mathbf{a}_{k,i}^S] := v$, where

$$v = \text{Retrieve}(m, h, S, \mathbf{a}_{i,k}) - x + \sum_{j \in S \setminus \{i,k\}} \text{Retrieve}(m, h, S, \mathbf{a}_{i,j}) - \text{Retrieve}(m, h, S, \mathbf{a}_{j,i})$$

From the adversary’s perspective, the only difference between G_8 and G_9 is on how a query `Retrieve`($m, h, S, \mathbf{a}_{i,j}^S$) (invoked in a OH_{seed} query) is returned. We claim that G_8 and G_9 always return the same values.

Recall from G_2 that when the adversary queries H_{seed} at $(\text{seed}_{i,j}, m, h, S)$, at least one of i or j must be corrupted. This means that when `Retrieve`($m, h, S, \mathbf{a}_{i,j}^S$) is called, at least one of i or j must be corrupted. This means we only need to show that for all corrupted parties $i \in C$, $(m, h, S) \in P$, and $\mathbf{a} \in \{\mathbf{a}_{i,j}^S, \mathbf{a}_{j,i}^S\}_{j \in S \setminus \{i\}}$, the G_8 and G_9 challengers obtain the same value for `Retrieve`($m, h, S, \mathbf{a}$).

We proceed by induction. We first show that after the first query, for all $i \in C$, $(m, h, S) \in P$, and $\mathbf{a} \in \{\mathbf{a}_{i,j}^S, \mathbf{a}_{j,i}^S\}_{j \in S \setminus \{i\}}$, the challengers in G_8 and G_9 obtain the same value for `Retrieve`($m, h, S, \mathbf{a}$). If the first query is a H_{seed} query or a partial signature query, then $C = \emptyset$, so the statement is vacuously true. If the first query is a corruption query, there are no entries in P , so the statement is vacuously true.

We now assume that the inductive hypothesis is true after n queries and prove it holds after the $(n+1)$ -th query.

- If the $(n+1)$ -th query is a partial signature query (i, m, h, S) or H_{seed} query $(\text{seed}_{i,j}, m, h, S)$, then either $(m, h, S) \in P$, and C, P do not change; or $(m, h, S) \notin P$, in which case the G_8 or G_9 challengers both add all $\{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{i \in S \cap C, j \in S \setminus \{i\}}$ as rows to $\mathbf{A}^{m,h,S}$. In these cases, the inductive hypothesis still holds.
- If the $(n+1)$ -th query is a corruption query for party i , we show that after this call, for all $(m, h, S) \in P$ and $\mathbf{a} \in \{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus C}$, the G_8 and G_9 challengers obtain the

same value for $\text{Retrieve}(m, h, S, \mathbf{a})$. Either the G_8 and G_9 challengers add all these vectors as rows to $\mathbf{A}^{m, h, S}$, or the G_8 and G_9 challengers add all these vectors as rows to $\mathbf{A}^{m, h, S}$ except for $\mathbf{a}_{k,i}$ for a specifically chosen honest party k . In the former case, G_8 and G_9 challengers clearly obtain the same value for $\text{Retrieve}(m, h, S, \mathbf{a})$ for all $\mathbf{a} \in \{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus C}$. In the latter case, G_8 and G_9 challengers clearly obtain the same value for $\text{Retrieve}(m, h, S, \mathbf{a})$ for all $\mathbf{a} \in \{\mathbf{a}_{i,j}, \mathbf{a}_{j,i}\}_{j \in S \setminus C}$ except for $\mathbf{a}_{k,i}$. By the definition of $\mathbf{a}_{k,i}$, when the G_8 challenger calls $\text{Retrieve}(m, h, S, \mathbf{a}_{k,i})$ in G_8 , it obtains v where:

$$\begin{aligned} v = & G_8.\text{Retrieve}(m, h, S, \mathbf{a}_{i,k}) - G_8.\text{Retrieve}(m, h, S, \mathbf{a}_i) \\ & + \sum_{j \in S \setminus \{i,k\}} G_8.\text{Retrieve}(m, h, S, \mathbf{a}_{i,j}) - G_8.\text{Retrieve}(m, h, S, \mathbf{a}_{j,i}) \end{aligned}$$

The G_9 challenger manually sets $M[m, h, S, \mathbf{a}_{k,i}] := v'$ such that

$$\begin{aligned} v' = & G_9.\text{Retrieve}(m, h, S, \mathbf{a}_{i,k}) - x \\ & + \sum_{j \in S \setminus \{i,k\}} G_9.\text{Retrieve}(m, h, S, \mathbf{a}_{i,j}) - G_9.\text{Retrieve}(m, h, S, \mathbf{a}_{j,i}). \end{aligned}$$

Here x is exactly the value G_8 challenger returns when calling $\text{Retrieve}(m, h, S, \mathbf{a}_i^S)$. Since all $G_8.\text{Retrieve}$ and $G_9.\text{Retrieve}$ calls made when defining v and v' return the same values, it holds that $v = v'$.

In sum, we have that

$$\Pr[G_8^{\mathcal{A}} = 1] = \Pr[G_9^{\mathcal{A}} = 1].$$

Game G_{10} . This is the same as G_9 except that the challenger uses the alternate Shamir secret sharing for ring elements from [Construction 7.2](#) to sample signing shares. The alternate Shamir secret sharing scheme gives the same distribution as the standard Shamir secret sharing scheme, so:

$$\Pr[G_9^{\mathcal{A}} = 1] = \Pr[G_{10}^{\mathcal{A}} = 1].$$

Reducing to EUF-CMA security. Finally, we convert any adversary $\mathcal{A}$ for Game G_{10} into an adversary $\mathcal{B}$ for EUF-CMA game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ with the same winning probability:

- Upon receiving $(\ell, w, N, T, \text{state})$ from $\mathcal{A}$, $\mathcal{B}$ forwards (ℓ, w, state) to the challenger of $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ for the basic scheme. $\mathcal{B}$ maintains a map U to keep track of the queries it makes to the challenger in $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ for the basic scheme.
- $\mathcal{B}$ answers every query from the challenger just as in G_{10} except:
 - When $\mathcal{A}$ issues a corruption query for party i , $\mathcal{B}$ randomly samples a secret key $\text{sk}_i := (H, \overline{\text{sk}}_i \leftarrow \$ R)$, stores sk_i as the secret key for party i , and returns sk_i to $\mathcal{A}$.

- When the adversary makes a partial signature query (i, m, h, S) where it already has $|S| - 1$ signatures on (m, h, S) , if $(m, h) \notin U$, then $\mathcal{B}$ queries the challenger in $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ for a signature on (m, h) . Let σ be the response. $\mathcal{B}$ stores $U[(m, h)] \leftarrow \sigma$. $\mathcal{B}$ then computes its response to $\mathcal{A}$ using $U[(m, h)]$ as the full signature σ on (m, h, S) .

- Upon receiving (m, h, σ) from $\mathcal{A}$, $\mathcal{B}$ returns forgery $(1, 1, m, h, \sigma)$ to challenger.

We first argue that $\mathcal{A}$ cannot distinguish between interacting with $\mathcal{B}$ and interacting with the challenger in G_{10} . In both games, the (no more than $T - 1$) secret keys returned to the adversary are randomly sampled. The only difference between the challenger in G_{10} and $\mathcal{B}$ is how they answer partial signature queries on (m, h, S) when $\mathcal{A}$ already has $|S| - 1$ partial signatures. The challenger in G_{10} computes the full signature itself, while $\mathcal{B}$ receives the full signature from the basic scheme game's challenger. Both signatures are honestly generated using the basic signature scheme with respect to the public key given to $\mathcal{A}$ at the beginning of the game.

Finally, we argue that if (m, h, σ) is a valid forgery in the adaptive threshold security game, then (m, h, σ) is a valid forgery in the EUF-CMA game $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$, and $\mathcal{B}$ is a valid adversary. For the former point, by definition $\Pi_{\text{sync}} \cdot \text{Vf}(\text{pk}, m, h, \sigma) = 1$. For the latter point, because $\mathcal{A}$ is a valid adversary, it can obtain valid signatures for at most w different (m', h) tuples, which means that $\mathcal{B}$ queries the $\mathbf{G}_{\mathcal{A}, \text{Sig}}^{\text{ssig}, 1}(\lambda)$ challenger at slot h on at most w different m' values and is therefore a valid adversary. $\square$

8 Instantiation and Performance

In this section, we analyze the asymptotic and concrete complexity of our signature schemes.

Asymptotic complexity. Let $\ell = \bar{\ell}^\tau$ denote the total number of slots where $\tau = O(1)$. Let Π_{VC}^ℓ be the ℓ -slot homomorphic vector commitment as in Figure 2. Let Π_{VC}^ℓ be the bootstrapped ℓ -slot vector commitment by applying Construction A.1 for $\tau - 1$ times. Built upon Π_{VC}^ℓ , we obtain a w -way ℓ -slot synchronous signature scheme $\Pi_{\text{sync}}^{w, \ell}$ as in Figure 4 with the following complexity:

- The CRS is mainly a public trapdoor matrix $\mathbf{T}^* \in R^{(\bar{\ell}m+k) \times \bar{\ell}k}$ with width parameter s , where $k = n \lceil \log_b(q) \rceil$. By Theorem 4.1,

$$s \approx \omega(\log(dn)) \sqrt{\bar{\ell} \log \bar{\ell}} \cdot d \cdot m \cdot b.$$

- A public key is dominated by $w + 1$ commitments in R_q^n .
- A secret key consists of $O(\ell w)$ elements in R_q and $O(\ell w)$ Gaussian preimages. Here each Gaussian preimage is over R^m for $m := 3n \log_b(q)$ with width parameter s . Note that secret key values can be generated on the fly.

- A signature consists of $\tau - 1$ commitments⁶ in R_q^n , τ opening proofs in R^m and 1 R_q -element.

To analyze bit complexity of the above components, we need to analyze the size of each Gaussian preimage and opening proof. By the rANS encoding from [ETWY22], an (md) -dimensional Gaussian vector with width s can be encoded using approximately

$$\frac{md}{2}(1 + \log_2(2\pi s^2)) \approx md \cdot \log_2(s)$$

bits. However, each opening proof π is a linear combination of $w + 1$ Gaussian vectors with coefficients in $\mathcal{S} \subseteq R_q$. Here we let $\mathcal{S}$ be the set of R_q -elements with at most $\delta_{\mathcal{S}}$ non-zero coefficients, where each coefficient is in $\{0, \pm 1\}$. Then the entropy (and thus the bit complexity) of π can be approximately bounded by

$$\frac{md}{2}(\log_2(2\pi s^2 \cdot (w + 1) \cdot \delta_{\mathcal{S}})) \approx md \cdot \left(\log_2(s) + \frac{1 + \log_2(\delta_{\mathcal{S}}) + \log_2(w)}{2} \right).$$

Similarly, for an aggregate signature with at most N signers as described in [Construction 6.1](#), an opening proof's entropy is approximately bounded by

$$\frac{md}{2}(1 + \log_2(2\pi s^2 \cdot (w + 1) \cdot \delta_{\mathcal{S}}^2 \cdot N)) \approx md \cdot \left(\log_2(s) + \log_2(\delta_{\mathcal{S}}) + \frac{\log_2(w) + \log_2(N)}{2} \right).$$

And if we use the aggregation scheme as in [Construction 6.2](#), we can further remove an additive term of $md \cdot \log_2(\delta_{\mathcal{S}})/2$.

In sum, recall that $\tau = O(1)$ and we set the base $b = q^{1/c}$ for some $c = O(1)$. Note that

$$\log_2(s) \approx \log_2(\bar{\ell}) + \log_2(nd) + \log_2(q) = O(\lambda)$$

(where λ is the security parameter) and $\log_2(s) + \log_2(\delta_{\mathcal{S}}) + \log_2(w) = O(\lambda)$. Thus, the complexity of $\Pi_{\text{sync}}^{w, \ell}$ is as specified in [Table 1](#). Note that the signature and aggregate signature sizes are $\Theta(\log \ell)$ -factor smaller compared to Chipmunk [FHSZ23] and Lemur [Lin+26], at the cost of a $O(\bar{\ell}^2 nd)$ -factor larger public parameters and a trusted setup.

Concrete complexity. Next, we specify the concrete signature sizes of our schemes and compare with Chipmunk [FHSZ23] and Lemur [Lin+26]. Based on the asymptotic results in [Table 1](#) we know that our scheme will dominate Chipmunk [FHSZ23] and Lemur [Lin+26] for a sufficiently large number of slots ℓ . We show that for a level-1 tree, namely when $\tau = 1$, our schemes produces shorter aggregate signatures when the number of slots is

⁶Recall that in [Construction A.1](#), to vector-commit $\bar{\ell}$ commitments over R_q^n , we first compress each commitment in R_q^n into an R_q -element via the Ajtai hash function, and then commit to the $\bar{\ell}$ R_q -element via $\Pi_{\text{VC}}^{\bar{\ell}}$.

$\ell = 2^{24}$ and one aggregates sufficiently many signatures. For a two-level tree (i.e. $\tau = 2$) the crossover point is somewhere above $\ell = 2^{24}$.

To do the comparison we focus on 1-way synchronous signatures, namely $w = 1$. We set $d = 128$, modulus $q \approx 2^{160}$, gadget base $b = 2^{40}$. The challenge set $\mathcal{S}$ consists of elements with at most $\delta_{\mathcal{S}} = 20$ non-zero coefficients, with each coefficient in $\{0, \pm 1\}$. The randomized rounding parameters γ', γ that map to the terms $\omega(\sqrt{\log(dn)})$ and $\omega(\sqrt{\log(\ell dn)})$ in Eq. (10) and Eq. (11) are set as $\gamma' = 5$ and $\gamma = 10$. Parameters are chosen to achieve 100-bit security according to the lattice estimator⁷ [APS15].

Our results are summarized in tables below. Table 2 shows our signature sizes for supporting $\ell = 2^{24}$ slots and aggregating signatures from up to $N = 2^{20}$ signers, using either a depth-1 or depth-2 tree. Note that $\ell = (\bar{\ell})^\tau$ where τ is the tree height and $\bar{\ell}$ is the number of slots supported by the base scheme (which determines the size of the CRS).

τ	ℓ	$\bar{\ell}$	N	q (bits)	Size (KB) (Construction 6.1)	Size (KB) (Construction 6.2)
1	2^{24}	2^{24}	2^{10}	160	351	326
			2^{15}		396	370
			2^{17}		418	391
			2^{20}		443	415
2	2^{24}	2^{12}	2^{10}	160	634	586
			2^{15}		687	637
			2^{17}		730	678
			2^{20}		778	725

Table 2: Aggregate signature sizes (KB) of our schemes when aggregating N signatures. ℓ is the total number of supported slots, τ is the tree depth (either one or two), and $\bar{\ell}$ is the number of supported slots in each level of the tree, and q is the modulus.

Table 3 compares the signature sizes of our schemes with Chipmunk [FHSZ23] and Lemur [Lin+26]. Our schemes achieve sizes comparable with Lemur, and the signature is even smaller when the number of signers is large (e.g. 2^{17}). If we aim for a smaller CRS size and set $\tau = 2$, the signature size is slightly worse compared to Lemur, as shown in Table 2.

9 Conclusions and Open Problems

We presented a synchronous signature scheme that supports efficient aggregation and thresholding. The construction is very different from existing constructions such as Chipmunk [FHSZ23] and Lemur [Lin+26], and may lead to new directions in this area. For

⁷<https://github.com/malb/lattice-estimator>

ℓ	N	Chipmunk ($d = 1024$)	Lemur ($d = 256$)	Construction 6.1 ($d = 128, \tau = 1$)	Construction 6.2 ($d = 128, \tau = 1$)
	2^{15}	> 471	367	396	370
2^{24}	2^{17}	> 602	418	418	391
	2^{20}	> 844	519	443	415

Table 3: Signature sizes (KB) of our scheme vs. Chipmunk [FHSZ23] and Lemur [Lin+26]. ℓ is the number of supported slots, N is the number of aggregated signatures, and d is the required ring dimensions. The meaning of $\tau = 1$ is explained in Section 8.

certain parameter settings, the resulting aggregate signatures are shorter than those obtained by Chipmunk [FHSZ23] and Lemur [Lin+26].

Our work raises a number of interesting directions for future research. The key question is how to further shorten the signatures generated by our scheme. One reason for the length is the degraded quality of the lattice basis $\mathbf{T}^*$ for $\mathbf{D}_\ell$ defined in (8). The lattice basis $\mathbf{T}^*$ is the result of basis randomization and is not as short as it can be. Is there a way to generate a basis for the lattice defined by $\mathbf{D}_\ell$? A different direction for shortening the signatures is to use NTRU lattices [Hof+03], as in Falcon [Pre+22], or even module-NTRU lattices [Chu+20].

Another important question is how to remove the trusted setup while preserving the asymptotic complexity of the scheme. A technique from [WWW25] lets us generate a public trapdoor for $\mathbf{D}_\ell$, but at the cost of blowing up the dimension of $\mathbf{D}_\ell$ by a factor of $\log_2 \ell$, which increases the size of the signature by $\log_2 \ell$. As a result, the resulting scheme is no better asymptotically than the Merkle-based methods [FHSZ23; Lin+26]. Is there a way to remove the trusted setup without re-introducing the $\log_2 \ell$ factor in the lattice dimension?

Acknowledgments. This work was partially supported by the Stanford Center for Blockchain Research (CBR) and UBRI.

References

- [Aar+24] Marius A. Aardal, Diego F. Aranha, Katharina Boudgoust, Sebastian Kolby, and Akira Takahashi. “Aggregating Falcon Signatures with LaBRADOR”. In: *Advances in Cryptology – CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, 2024, pp. 71–106. DOI: [10.1007/978-3-031-68376-3_3](https://doi.org/10.1007/978-3-031-68376-3_3).

- [ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. “Efficient Lattice (H)IBE in the Standard Model”. In: *Advances in Cryptology – EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. Lecture Notes in Computer Science. French Riviera: Springer Berlin Heidelberg, Germany, 2010, pp. 553–572. DOI: [10.1007/978-3-642-13190-5_28](https://doi.org/10.1007/978-3-642-13190-5_28).
- [ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. “Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE”. In: *Advances in Cryptology – CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, 2010, pp. 98–115. DOI: [10.1007/978-3-642-14623-7_6](https://doi.org/10.1007/978-3-642-14623-7_6).
- [AFH24] Hiroaki Anada, Masayuki Fukumitsu, and Shingo Hasegawa. “Tightly Secure Lattice-Based Synchronized Aggregate Signature in Standard Model”. In: *ICISC 24: 27th International Conference on Information Security and Cryptology*. Ed. by Jongsung Kim, Jungsoo Park, and Wai-Kong Lee. Vol. 15596. Lecture Notes in Computer Science. Seoul, Korea: Springer, Singapore, Singapore, 2024, pp. 56–70. DOI: [10.1007/978-981-96-5566-3_4](https://doi.org/10.1007/978-981-96-5566-3_4).
- [AGH10] Jae Hyun Ahn, Matthew Green, and Susan Hohenberger. “Synchronized aggregate signatures: new definitions, constructions and applications”. In: *ACM CCS 2010: 17th Conference on Computer and Communications Security*. Ed. by Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov. Chicago, Illinois, USA: ACM Press, 2010, pp. 473–484. DOI: [10.1145/1866307.1866360](https://doi.org/10.1145/1866307.1866360).
- [AGHS13] Shweta Agrawal, Craig Gentry, Shai Halevi, and Amit Sahai. “Discrete Gaussian Leftover Hash Lemma over Infinite Domains”. In: *Advances in Cryptology – ASIACRYPT 2013, Part I*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. Lecture Notes in Computer Science. Bangalore, India: Springer Berlin Heidelberg, Germany, 2013, pp. 97–116. DOI: [10.1007/978-3-642-42033-7_6](https://doi.org/10.1007/978-3-642-42033-7_6).
- [Ajt96] Miklós Ajtai. “Generating Hard Instances of Lattice Problems (Extended Abstract)”. In: *28th Annual ACM Symposium on Theory of Computing*. Philadelphia, PA, USA: ACM Press, 1996, pp. 99–108. DOI: [10.1145/237814.237838](https://doi.org/10.1145/237814.237838).
- [AP08] Joël Alwen and Chris Peikert. *Generating Shorter Bases for Hard Random Lattices*. Cryptology ePrint Archive, Report 2008/521. 2008. URL: <https://eprint.iacr.org/2008/521>.
- [APS15] Martin R Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of Learning with Errors.” In: *J. Math. Cryptol.* 9.3 (2015), pp. 169–203.

- [BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. “Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma”. In: *ACM CCS 2008: 15th Conference on Computer and Communications Security*. Ed. by Peng Ning, Paul F. Syverson, and Somesh Jha. Alexandria, Virginia, USA: ACM Press, 2008, pp. 449–458. DOI: [10.1145/1455770.1455827](https://doi.org/10.1145/1455770.1455827).
- [BCJP24] Maya Farber Brodsky, Arka Rai Choudhuri, Abhishek Jain, and Omer Paneth. “Monotone-Policy Aggregate Signatures”. In: *Advances in Cryptology – EUROCRYPT 2024, Part IV*. Ed. by Marc Joye and Gregor Leander. Vol. 14654. Lecture Notes in Computer Science. Zurich, Switzerland: Springer, Cham, Switzerland, 2024, pp. 168–195. DOI: [10.1007/978-3-031-58737-5_7](https://doi.org/10.1007/978-3-031-58737-5_7).
- [BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. “Interactive Oracle Proofs”. In: *TCC 2016-B: 14th Theory of Cryptography Conference, Part II*. Ed. by Martin Hirt and Adam D. Smith. Vol. 9986. Lecture Notes in Computer Science. Beijing, China: Springer Berlin Heidelberg, Germany, 2016, pp. 31–60. DOI: [10.1007/978-3-662-53644-5_2](https://doi.org/10.1007/978-3-662-53644-5_2).
- [BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. “Compact Multi-signatures for Smaller Blockchains”. In: *Advances in Cryptology – ASIACRYPT 2018, Part II*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11273. Lecture Notes in Computer Science. Brisbane, Queensland, Australia: Springer, Cham, Switzerland, 2018, pp. 435–464. DOI: [10.1007/978-3-030-03329-3_15](https://doi.org/10.1007/978-3-030-03329-3_15).
- [BF11] Dan Boneh and David Mandell Freeman. “Linearly Homomorphic Signatures over Binary Fields and New Tools for Lattice-Based Signatures”. In: *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Dario Catalano, Nelly Fazio, Rosario Genaro, and Antonio Nicolosi. Vol. 6571. Lecture Notes in Computer Science. Taormina, Italy: Springer Berlin Heidelberg, Germany, 2011, pp. 1–16. DOI: [10.1007/978-3-642-19379-8_1](https://doi.org/10.1007/978-3-642-19379-8_1).
- [BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. “Aggregate and Verifiably Encrypted Signatures from Bilinear Maps”. In: *Advances in Cryptology – EUROCRYPT 2003*. Ed. by Eli Biham. Vol. 2656. Lecture Notes in Computer Science. Warsaw, Poland: Springer Berlin Heidelberg, Germany, 2003, pp. 416–432. DOI: [10.1007/3-540-39200-9_26](https://doi.org/10.1007/3-540-39200-9_26).
- [BGR12] Kyle Brogle, Sharon Goldberg, and Leonid Reyzin. “Sequential Aggregate Signatures with Lazy Verification from Trapdoor Permutations - (Extended Abstract)”. In: *Advances in Cryptology – ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazuo Sako. Vol. 7658. Lecture Notes in Computer Sci-

- ence. Beijing, China: Springer Berlin Heidelberg, Germany, 2012, pp. 644–662. DOI: [10.1007/978-3-642-34961-4_39](https://doi.org/10.1007/978-3-642-34961-4_39).
- [BK20] Dan Boneh and Sam Kim. “One-Time and Interactive Aggregate Signatures from Lattices”. In: (2020). URL: https://crypto.stanford.edu/~skim13/agg_ots.pdf.
- [BL25] Katharina Boudgoust and Oleksandra Lapiha. *Leftover Hash Lemma(s) Over Cyclotomic Rings*. Cryptology ePrint Archive, Report 2025/1080. 2025. URL: <https://eprint.iacr.org/2025/1080>.
- [BN06] Mihir Bellare and Gregory Neven. “Multi-signatures in the plain public-Key model and a general forking lemma”. In: *ACM CCS 2006: 13th Conference on Computer and Communications Security*. Ed. by Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati. Alexandria, Virginia, USA: ACM Press, 2006, pp. 390–399. DOI: [10.1145/1180405.1180453](https://doi.org/10.1145/1180405.1180453).
- [BNN07] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. “Unrestricted Aggregate Signatures”. In: *ICALP 2007: 34th International Colloquium on Automata, Languages and Programming*. Ed. by Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki. Vol. 4596. Lecture Notes in Computer Science. Wroclaw, Poland: Springer Berlin Heidelberg, Germany, 2007, pp. 411–422. DOI: [10.1007/978-3-540-73420-8_37](https://doi.org/10.1007/978-3-540-73420-8_37).
- [Bol03] Alexandra Boldyreva. “Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme”. In: *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*. Ed. by Yvo Desmedt. Vol. 2567. Lecture Notes in Computer Science. Miami, FL, USA: Springer Berlin Heidelberg, Germany, 2003, pp. 31–46. DOI: [10.1007/3-540-36288-6_3](https://doi.org/10.1007/3-540-36288-6_3).
- [BPMW16] Florian Bourse, Rafaël del Pino, Michele Minelli, and Hoeteck Wee. “FHE Circuit Privacy Almost for Free”. In: *Advances in Cryptology – CRYPTO 2016, Part II*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9815. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, 2016, pp. 62–89. DOI: [10.1007/978-3-662-53008-5_3](https://doi.org/10.1007/978-3-662-53008-5_3).
- [BR21] Katharina Boudgoust and Adeline Roux-Langlois. *Compressed Linear Aggregate Signatures Based on Module Lattices*. Cryptology ePrint Archive, Report 2021/263. 2021. URL: <https://eprint.iacr.org/2021/263>.
- [BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography (version 0.6)*. Available at <https://toc.cryptobook.us/>. 2023.

- [BT23] Katharina Boudgoust and Akira Takahashi. “Sequential Half-Aggregation of Lattice-Based Signatures”. In: *ESORICS 2023: 28th European Symposium on Research in Computer Security, Part I*. Ed. by Gene Tsudik, Mauro Conti, Kaitai Liang, and Georgios Smaragdakis. Vol. 14344. Lecture Notes in Computer Science. The Hague, The Netherlands: Springer, Cham, Switzerland, 2023, pp. 270–289. DOI: [10.1007/978-3-031-50594-2_14](https://doi.org/10.1007/978-3-031-50594-2_14).
- [BW25] Renas Bacho and Benedikt Wagner. “T-Spoon: Tightly Secure Two-Round Multi-signatures with Key Aggregation”. In: *Advances in Cryptology – CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, 2025, pp. 256–290. DOI: [10.1007/978-3-032-01887-8_9](https://doi.org/10.1007/978-3-032-01887-8_9).
- [CGM19] Yilei Chen, Nicholas Genise, and Pratyay Mukherjee. “Approximate Trapdoors for Lattices and Smaller Hash-and-Sign Signatures”. In: *Advances in Cryptology – ASIACRYPT 2019, Part III*. Ed. by Steven D. Galbraith and Shihō Moriai. Vol. 11923. Lecture Notes in Computer Science. Kobe, Japan: Springer, Cham, Switzerland, 2019, pp. 3–32. DOI: [10.1007/978-3-030-34618-8_1](https://doi.org/10.1007/978-3-030-34618-8_1).
- [Chu+20] Chitchanok Chuengsatiansup, Thomas Prest, Damien Stehlé, Alexandre Wallet, and Keita Xagawa. “ModFalcon: Compact Signatures Based On Module-NTRU Lattices”. In: *ASIACCS 20: 15th ACM Symposium on Information, Computer and Communications Security*. Ed. by Hung-Min Sun, Shih-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese. Taipei, Taiwan: ACM Press, 2020, pp. 853–866. DOI: [10.1145/3320269.3384758](https://doi.org/10.1145/3320269.3384758).
- [Del+24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. “Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions”. In: *Advances in Cryptology – EUROCRYPT 2024, Part II*. Ed. by Marc Joye and Gregor Leander. Vol. 14652. Lecture Notes in Computer Science. Zurich, Switzerland: Springer, Cham, Switzerland, 2024, pp. 219–248. DOI: [10.1007/978-3-031-58723-8_8](https://doi.org/10.1007/978-3-031-58723-8_8).
- [DGNW20] Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. “Pixel: Multi-signatures for Consensus”. In: *USENIX Security 2020: 29th USENIX Security Symposium*. Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, 2020, pp. 2093–2110. URL: <https://www.usenix.org/conference/usenixsecurity20/presentation/drijvers>.
- [DGPY20] Léo Ducas, Steven Galbraith, Thomas Prest, and Yang Yu. “Integral Matrix Gram Root and Lattice Gaussian Sampling Without Floats”. In: *Advances in Cryptology – EUROCRYPT 2020, Part II*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12106. Lecture Notes in Computer Science. Zagreb, Croatia: Springer, Cham, Switzerland, 2020, pp. 3–20. DOI: [10.1007/978-3-030-38468-2_1](https://doi.org/10.1007/978-3-030-38468-2_1).

- tia: Springer, Cham, Switzerland, 2020, pp. 608–637. DOI: [10.1007/978-3-030-45724-2_21](https://doi.org/10.1007/978-3-030-45724-2_21).
- [DHSS20] Yarkın Doröz, Jeffrey Hoffstein, Joseph H. Silverman, and Berk Sunar. *MM-SAT: A Scheme for Multimessage Multiuser Signature Aggregation*. Cryptology ePrint Archive, Report 2020/520. 2020. URL: <https://eprint.iacr.org/2020/520>.
- [DKKW25a] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. *Hash-Based Multi-Signatures for Post-Quantum Ethereum*. Cryptology ePrint Archive, Report 2025/055. 2025. URL: <https://eprint.iacr.org/2025/055>.
- [DKKW25b] Justin Drake, Dmitry Khovratovich, Mikhail Kudinov, and Benedikt Wagner. *Technical Note: LeanSig for Post-Quantum Ethereum*. Cryptology ePrint Archive, Report 2025/1332. 2025. URL: <https://eprint.iacr.org/2025/1332>.
- [DM14] Léo Ducas and Daniele Micciancio. “Improved Short Lattice Signatures in the Standard Model”. In: *Advances in Cryptology – CRYPTO 2014, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, 2014, pp. 335–352. DOI: [10.1007/978-3-662-44371-2_19](https://doi.org/10.1007/978-3-662-44371-2_19).
- [EB14] Rachid El Bansarkhani and Johannes Buchmann. “Towards Lattice Based Aggregate Signatures”. In: *AFRICACRYPT 14: 7th International Conference on Cryptology in Africa*. Ed. by David Pointcheval and Damien Vergnaud. Vol. 8469. Lecture Notes in Computer Science. Marrakesh, Morocco: Springer, Cham, Switzerland, 2014, pp. 336–355. DOI: [10.1007/978-3-319-06734-6_21](https://doi.org/10.1007/978-3-319-06734-6_21).
- [EMP16] Rachid El Bansarkhani, Mohamed Saied Emam Mohamed, and Albrecht Petzoldt. *MQSAS - A Multivariate Sequential Aggregate Signature Scheme*. Cryptology ePrint Archive, Report 2016/503. 2016. URL: <https://eprint.iacr.org/2016/503>.
- [ETWY22] Thomas Espitau, Mehdi Tibouchi, Alexandre Wallet, and Yang Yu. “Shorter Hash-and-Sign Lattice-Based Signatures”. In: *Advances in Cryptology – CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, 2022, pp. 245–275. DOI: [10.1007/978-3-031-15979-4_9](https://doi.org/10.1007/978-3-031-15979-4_9).
- [FHSZ23] Nils Fleischhacker, Gottfried Herold, Mark Simkin, and Zhenfei Zhang. “Chipmunk: Better Synchronized Multi-Signatures from Lattices”. In: *ACM CCS 2023: 30th Conference on Computer and Communications Security*. Ed. by Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and En-

- gin Kirda. Copenhagen, Denmark: ACM Press, 2023, pp. 386–400. DOI: [10.1145/3576915.3623219](https://doi.org/10.1145/3576915.3623219).
- [FLS12] Marc Fischlin, Anja Lehmann, and Dominique Schröder. “History-Free Sequential Aggregate Signatures”. In: *SCN 12: 8th International Conference on Security in Communication Networks*. Ed. by Ivan Visconti and Roberto De Prisco. Vol. 7485. Lecture Notes in Computer Science. Amalfi, Italy: Springer Berlin Heidelberg, Germany, 2012, pp. 113–130. DOI: [10.1007/978-3-642-32928-9_7](https://doi.org/10.1007/978-3-642-32928-9_7).
- [FSZ22] Nils Fleischhacker, Mark Simkin, and Zhenfei Zhang. “Squirrel: Efficient Synchronized Multi-Signatures from Lattices”. In: *ACM CCS 2022: 29th Conference on Computer and Communications Security*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. Los Angeles, CA, USA: ACM Press, 2022, pp. 1109–1123. DOI: [10.1145/3548606.3560655](https://doi.org/10.1145/3548606.3560655).
- [GM18] Nicholas Genise and Daniele Micciancio. “Faster Gaussian Sampling for Trapdoor Lattices with Arbitrary Modulus”. In: *Advances in Cryptology – EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Cham, Switzerland, 2018, pp. 174–203. DOI: [10.1007/978-3-319-78381-9_7](https://doi.org/10.1007/978-3-319-78381-9_7).
- [GMPW20] Nicholas Genise, Daniele Micciancio, Chris Peikert, and Michael Walter. “Improved Discrete Gaussian and Subgaussian Analysis for Lattice Cryptography”. In: *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part I*. Ed. by Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas. Vol. 12110. Lecture Notes in Computer Science. Edinburgh, UK: Springer, Cham, Switzerland, 2020, pp. 623–651. DOI: [10.1007/978-3-030-45374-9_21](https://doi.org/10.1007/978-3-030-45374-9_21).
- [GOR18] Craig Gentry, Adam O’Neill, and Leonid Reyzin. “A Unified Framework for Trapdoor-Permutation-Based Sequential Aggregate Signatures”. In: *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*. Ed. by Michel Abdalla and Ricardo Dahab. Vol. 10770. Lecture Notes in Computer Science. Rio de Janeiro, Brazil: Springer, Cham, Switzerland, 2018, pp. 34–57. DOI: [10.1007/978-3-319-76581-5_2](https://doi.org/10.1007/978-3-319-76581-5_2).
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. “Trapdoors for hard lattices and new cryptographic constructions”. In: *40th Annual ACM Symposium on Theory of Computing*. Ed. by Richard E. Ladner and Cynthia Dwork. Victoria, BC, Canada: ACM Press, 2008, pp. 197–206. DOI: [10.1145/1374376.1374407](https://doi.org/10.1145/1374376.1374407).

- [GR06] Craig Gentry and Zulfikar Ramzan. “Identity-Based Aggregate Signatures”. In: *PKC 2006: 9th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin. Vol. 3958. Lecture Notes in Computer Science. New York, NY, USA: Springer Berlin Heidelberg, Germany, 2006, pp. 257–273. DOI: [10.1007/11745853_17](https://doi.org/10.1007/11745853_17).
- [HKW15] Susan Hohenberger, Venkata Koppula, and Brent Waters. “Universal Signature Aggregators”. In: *Advances in Cryptology – EUROCRYPT 2015, Part II*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9057. Lecture Notes in Computer Science. Sofia, Bulgaria: Springer Berlin Heidelberg, Germany, 2015, pp. 3–34. DOI: [10.1007/978-3-662-46803-6_1](https://doi.org/10.1007/978-3-662-46803-6_1).
- [Hof+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. “NTRUSIGN: Digital Signatures Using the NTRU Lattice”. In: *Topics in Cryptology – CT-RSA 2003*. Ed. by Marc Joye. Vol. 2612. Lecture Notes in Computer Science. San Francisco, CA, USA: Springer Berlin Heidelberg, Germany, 2003, pp. 122–140. DOI: [10.1007/3-540-36563-X_9](https://doi.org/10.1007/3-540-36563-X_9).
- [HW18] Susan Hohenberger and Brent Waters. “Synchronized Aggregate Signatures from the RSA Assumption”. In: *Advances in Cryptology – EUROCRYPT 2018, Part II*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10821. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Cham, Switzerland, 2018, pp. 197–229. DOI: [10.1007/978-3-319-78375-8_7](https://doi.org/10.1007/978-3-319-78375-8_7).
- [JRS24] Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. “Phoenix: Hash-and-Sign with Aborts from Lattice Gadgets”. In: *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I*. Ed. by Markku-Juhani Saarinen and Daniel Smith-Tone. Oxford, UK: Springer, Cham, Switzerland, 2024, pp. 265–299. DOI: [10.1007/978-3-031-62743-9_9](https://doi.org/10.1007/978-3-031-62743-9_9).
- [JWZ26] Kaijie Jiang, Hoeteck Wee, and Chenzhi Zhu. *Oriole: Adaptively Secure Partially Non-Interactive Threshold Signatures from Lattices*. Cryptology ePrint Archive, Report 2026/793. 2026. URL: <https://eprint.iacr.org/2026/793>.
- [KCLM22] Irakliy Khaburzaniya, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. “Aggregating and Thresholdizing Hash-based Signatures using STARKs”. In: *ASIACCS 22: 17th ACM Symposium on Information, Computer and Communications Security*. Ed. by Yuji Suga, Kouichi Sakurai, Xuhua Ding, and Kazue Sako. Nagasaki, Japan: ACM Press, 2022, pp. 393–407. DOI: [10.1145/3488932.3524128](https://doi.org/10.1145/3488932.3524128).

- [KS23] R. Kabaleeshwaran and Panuganti Venkata Shanmukh Sai. “Synchronized Aggregate Signature Under Standard Assumption in the Random Oracle Model”. In: *Progress in Cryptology - INDOCRYPT 2023: 24th International Conference in Cryptology in India, Part I*. Ed. by Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro. Vol. 14459. Lecture Notes in Computer Science. Goa, India: Springer, Cham, Switzerland, 2023, pp. 197–220. DOI: [10.1007/978-3-031-56232-7_10](https://doi.org/10.1007/978-3-031-56232-7_10).
- [Lin+26] Yini Lin, Muhammed F. Esgin, Amin Sakzad, Ron Steinfeld, and Markku-Juhani O. Saarinen. *Lemur: Scalable Post-Quantum Synchronized Multi-Signatures*. Cryptology ePrint Archive, Paper 2026/1161. 2026. URL: <https://eprint.iacr.org/2026/1161>.
- [LLY13a] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. “Sequential Aggregate Signatures Made Shorter”. In: *ACNS 2013: 11th International Conference on Applied Cryptography and Network Security*. Ed. by Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini. Vol. 7954. Lecture Notes in Computer Science. Banff, AB, Canada: Springer Berlin Heidelberg, Germany, 2013, pp. 202–217. DOI: [10.1007/978-3-642-38980-1_13](https://doi.org/10.1007/978-3-642-38980-1_13).
- [LLY13b] Kwangsu Lee, Dong Hoon Lee, and Moti Yung. “Sequential Aggregate Signatures with Short Public Keys: Design, Analysis and Implementation Studies”. In: *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. Lecture Notes in Computer Science. Nara, Japan: Springer Berlin Heidelberg, Germany, 2013, pp. 423–442. DOI: [10.1007/978-3-642-36362-7_26](https://doi.org/10.1007/978-3-642-36362-7_26).
- [LM18] Vadim Lyubashevsky and Daniele Micciancio. “Asymptotically Efficient Lattice-Based Digital Signatures”. In: *Journal of Cryptology* 31.3 (July 2018), pp. 774–797. DOI: [10.1007/s00145-017-9270-z](https://doi.org/10.1007/s00145-017-9270-z).
- [LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. “Sequential Aggregate Signatures from Trapdoor Permutations”. In: *Advances in Cryptology – EUROCRYPT 2004*. Ed. by Christian Cachin and Jan Camenisch. Vol. 3027. Lecture Notes in Computer Science. Interlaken, Switzerland: Springer Berlin Heidelberg, Germany, 2004, pp. 74–90. DOI: [10.1007/978-3-540-24676-3_5](https://doi.org/10.1007/978-3-540-24676-3_5).
- [LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “A Toolkit for Ring-LWE Cryptography”. In: *Advances in Cryptology – EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Athens, Greece: Springer Berlin Heidelberg, Germany, 2013, pp. 35–54. DOI: [10.1007/978-3-642-38348-9_3](https://doi.org/10.1007/978-3-642-38348-9_3).

- [LS15] Adeline Langlois and Damien Stehlé. “Worst-case to average-case reductions for module lattices”. In: *Designs, Codes and Cryptography* 75.3 (2015), pp. 565–599. DOI: [10.1007/s10623-014-9938-4](https://doi.org/10.1007/s10623-014-9938-4).
- [LS18] Vadim Lyubashevsky and Gregor Seiler. “Short, Invertible Elements in Partially Splitting Cyclotomic Rings and Applications to Lattice-Based Zero-Knowledge Proofs”. In: *Advances in Cryptology – EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Cham, Switzerland, 2018, pp. 204–224. DOI: [10.1007/978-3-319-78381-9_8](https://doi.org/10.1007/978-3-319-78381-9_8).
- [Lu+06] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. “Sequential Aggregate Signatures and Multisignatures Without Random Oracles”. In: *Advances in Cryptology – EUROCRYPT 2006*. Ed. by Serge Vaudenay. Vol. 4004. Lecture Notes in Computer Science. St. Petersburg, Russia: Springer Berlin Heidelberg, Germany, 2006, pp. 465–485. DOI: [10.1007/11761679_28](https://doi.org/10.1007/11761679_28).
- [MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. “Accountable-Subgroup Multisignatures: Extended Abstract”. In: *ACM CCS 2001: 8th Conference on Computer and Communications Security*. Ed. by Michael K. Reiter and Pierangela Samarati. Philadelphia, PA, USA: ACM Press, 2001, pp. 245–254. DOI: [10.1145/501983.502017](https://doi.org/10.1145/501983.502017).
- [MP12] Daniele Micciancio and Chris Peikert. “Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller”. In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. Lecture Notes in Computer Science. Cambridge, UK: Springer Berlin Heidelberg, Germany, 2012, pp. 700–718. DOI: [10.1007/978-3-642-29011-4_41](https://doi.org/10.1007/978-3-642-29011-4_41).
- [MP13] Daniele Micciancio and Chris Peikert. “Hardness of SIS and LWE with Small Parameters”. In: *Advances in Cryptology – CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, 2013, pp. 21–39. DOI: [10.1007/978-3-642-40041-4_2](https://doi.org/10.1007/978-3-642-40041-4_2).
- [MR04] Daniele Micciancio and Oded Regev. “Worst-Case to Average-Case Reductions Based on Gaussian Measures”. In: *45th Annual Symposium on Foundations of Computer Science*. Rome, Italy: IEEE Computer Society Press, 2004, pp. 372–381. DOI: [10.1109/F0CS.2004.72](https://doi.org/10.1109/F0CS.2004.72).
- [Nev08] Gregory Neven. “Efficient Sequential Aggregate Signed Data”. In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel P. Smart. Vol. 4965. Lecture Notes in Computer Science. Istanbul, Turkey: Springer Berlin Heidelberg, Germany, 2008, pp. 52–69. DOI: [10.1007/978-3-540-78967-3_4](https://doi.org/10.1007/978-3-540-78967-3_4).

- [Pei10] Chris Peikert. “An Efficient and Parallel Gaussian Sampler for Lattices”. In: *Advances in Cryptology – CRYPTO 2010*. Ed. by Tal Rabin. Vol. 6223. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, 2010, pp. 80–97. DOI: [10.1007/978-3-642-14623-7_5](https://doi.org/10.1007/978-3-642-14623-7_5).
- [Pre+22] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. *Falcon*. Technical report, National Institute of Standards and Technology. Available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>. 2022.
- [PS00] David Pointcheval and Jacques Stern. “Security Arguments for Digital Signatures and Blind Signatures”. In: *Journal of Cryptology* 13.3 (June 2000), pp. 361–396. DOI: [10.1007/s001450010003](https://doi.org/10.1007/s001450010003).
- [PW24] Jiaxin Pan and Benedikt Wagner. “Toothpicks: More Efficient Fork-Free Two-Round Multi-signatures”. In: *Advances in Cryptology – EUROCRYPT 2024, Part I*. Ed. by Marc Joye and Gregor Leander. Vol. 14651. Lecture Notes in Computer Science. Zurich, Switzerland: Springer, Cham, Switzerland, 2024, pp. 460–489. DOI: [10.1007/978-3-031-58716-0_16](https://doi.org/10.1007/978-3-031-58716-0_16).
- [RSW18] Miruna Rosca, Damien Stehlé, and Alexandre Wallet. “On the Ring-LWE and Polynomial-LWE Problems”. In: *Advances in Cryptology – EUROCRYPT 2018, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Tel Aviv, Israel: Springer, Cham, Switzerland, 2018, pp. 146–173. DOI: [10.1007/978-3-319-78381-9_6](https://doi.org/10.1007/978-3-319-78381-9_6).
- [RT25] Lior Rotem and Stefano Tessaro. “Straight-Line Knowledge Extraction for Multi-Round Protocols”. In: *Advances in Cryptology – CRYPTO 2025, Part VII*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16006. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, 2025, pp. 95–127. DOI: [10.1007/978-3-032-01907-3_4](https://doi.org/10.1007/978-3-032-01907-3_4).
- [RY07] Thomas Ristenpart and Scott Yilek. “The Power of Proofs-of-Possession: Securing Multiparty Signatures against Rogue-Key Attacks”. In: *Advances in Cryptology – EUROCRYPT 2007*. Ed. by Moni Naor. Vol. 4515. Lecture Notes in Computer Science. Barcelona, Spain: Springer Berlin Heidelberg, Germany, 2007, pp. 228–245. DOI: [10.1007/978-3-540-72540-4_13](https://doi.org/10.1007/978-3-540-72540-4_13).
- [Sch11] Dominique Schröder. “How to Aggregate the CL Signature Scheme”. In: *ESORICS 2011: 16th European Symposium on Research in Computer Security*. Ed. by Vijay Atluri and Claudia Díaz. Vol. 6879. Lecture Notes in Computer Science. Leuven, Belgium: Springer Berlin Heidelberg, Germany, 2011, pp. 298–314. DOI: [10.1007/978-3-642-23822-2_17](https://doi.org/10.1007/978-3-642-23822-2_17).

- [Sch91] Claus-Peter Schnorr. “Efficient Signature Generation by Smart Cards”. In: *Journal of Cryptology* 4.3 (Jan. 1991), pp. 161–174. DOI: [10.1007/BF00196725](https://doi.org/10.1007/BF00196725).
- [SS11] Damien Stehlé and Ron Steinfeld. “Making NTRU as Secure as Worst-Case Problems over Ideal Lattices”. In: *Advances in Cryptology – EUROCRYPT 2011*. Ed. by Kenneth G. Paterson. Vol. 6632. Lecture Notes in Computer Science. Tallinn, Estonia: Springer Berlin Heidelberg, Germany, 2011, pp. 27–47. DOI: [10.1007/978-3-642-20465-4_4](https://doi.org/10.1007/978-3-642-20465-4_4).
- [SS22a] Shingo Sato and Junji Shikata. “Identity-based Interactive Aggregate Signatures from Lattices”. In: *ICISC 22: 25th International Conference on Information Security and Cryptology*. Ed. by Seung-Hyun Seo and Hwajeong Seo. Vol. 13849. Lecture Notes in Computer Science. Seoul, Korea: Springer, Cham, Switzerland, 2022, pp. 408–432. DOI: [10.1007/978-3-031-29371-9_20](https://doi.org/10.1007/978-3-031-29371-9_20).
- [SS22b] Shingo Sato and Junji Shikata. *Quantum-Secure Aggregate One-time Signatures with Detecting Functionality*. Cryptology ePrint Archive, Report 2022/272. 2022. URL: <https://eprint.iacr.org/2022/272>.
- [TS23] Toi Tomita and Junji Shikata. *Compact Aggregate Signature from Module-Lattices*. Cryptology ePrint Archive, Report 2023/471. 2023. URL: <https://eprint.iacr.org/2023/471>.
- [TT20] Masayuki Tezuka and Keisuke Tanaka. “Improved Security Proof for the Camenisch-Lysyanskaya Signature-Based Synchronized Aggregate Signature Scheme”. In: *ACISP 20: 25th Australasian Conference on Information Security and Privacy*. Ed. by Joseph K. Liu and Hui Cui. Vol. 12248. Lecture Notes in Computer Science. Perth, WA, Australia: Springer, Cham, Switzerland, 2020, pp. 225–243. DOI: [10.1007/978-3-030-55304-3_12](https://doi.org/10.1007/978-3-030-55304-3_12).
- [TT22] Masayuki Tezuka and Keisuke Tanaka. “Pointcheval-Sanders Signature-Based Synchronized Aggregate Signature”. In: *ICISC 22: 25th International Conference on Information Security and Cryptology*. Ed. by Seung-Hyun Seo and Hwajeong Seo. Vol. 13849. Lecture Notes in Computer Science. Seoul, Korea: Springer, Cham, Switzerland, 2022, pp. 317–336. DOI: [10.1007/978-3-031-29371-9_16](https://doi.org/10.1007/978-3-031-29371-9_16).
- [Unp26] Unpublished. *A Homomorphic Vector Commitment With a Short CRS*. private communications. 2026.
- [Wei+24] Jianghong Wei, Guohua Tian, Ding Wang, Fuchun Guo, Willy Susilo, and Xiaofeng Chen. “Pixel+ and Pixel++: Compact and Efficient Forward-Secure Multi-Signatures for PoS Blockchain Consensus”. In: *USENIX Security 2024: 33rd USENIX Security Symposium*. Ed. by Davide Balzarotti and Wenyan Xu. Philadelphia, PA, USA: USENIX Association, 2024. URL:

<https://www.usenix.org/conference/usenixsecurity24/presentation/wei-jianghong>.

- [WW19] Zhipeng Wang and Qianhong Wu. “A Practical Lattice-Based Sequential Aggregate Signature”. In: *ProvSec 2019: 13th International Conference on Provable Security*. Ed. by Ron Steinfeld and Tsz Hon Yuen. Vol. 11821. Lecture Notes in Computer Science. Cairns, QLD, Australia: Springer, Cham, Switzerland, 2019, pp. 94–109. DOI: [10.1007/978-3-030-31919-9_6](https://doi.org/10.1007/978-3-030-31919-9_6).
- [WW23] Hoeteck Wee and David J. Wu. “Succinct Vector, Polynomial, and Functional Commitments from Lattices”. In: *Advances in Cryptology – EUROCRYPT 2023, Part III*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14006. Lecture Notes in Computer Science. Lyon, France: Springer, Cham, Switzerland, 2023, pp. 385–416. DOI: [10.1007/978-3-031-30620-4_13](https://doi.org/10.1007/978-3-031-30620-4_13).
- [WWW25] Brent Waters, Hoeteck Wee, and David J. Wu. “New Techniques for Preimage Sampling: Improved NIZKs and More from LWE”. In: *Advances in Cryptology – EUROCRYPT 2025, Part IV*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15604. Lecture Notes in Computer Science. Madrid, Spain: Springer, Cham, Switzerland, 2025, pp. 3–33. DOI: [10.1007/978-3-031-91134-7_1](https://doi.org/10.1007/978-3-031-91134-7_1).
- [Zha+24] Xinyu Zhang, Ron Steinfeld, Muhammed F. Esgin, Joseph K. Liu, Dongxi Liu, and Sushmita Ruj. “Loquat: A SNARK-Friendly Post-quantum Signature Based on the Legendre PRF with Applications in Ring and Aggregate Signatures”. In: *Advances in Cryptology – CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, 2024, pp. 3–38. DOI: [10.1007/978-3-031-68376-3_1](https://doi.org/10.1007/978-3-031-68376-3_1).

A Bootstrapping Vector Commitments

A.1 Proof of [Lemma 3.1](#)

Proof. We prove that [Construction A.1](#) satisfies all of the required properties.

Claim 14. $\Pi_{\ell\ell'}$ satisfies correctness ([Definition 3.2](#)) if Π_1 , Π_ℓ and $\Pi_{\ell'}$ are correct.

Proof. Fix any λ, ℓ , vector $\mathbf{x} \in (\bar{R}_q^{n_m})^\ell$, and index $h \in [\ell]$. Let $(\mathbf{cp}, \mathbf{vp}) \leftarrow \Pi_{\ell\ell'}.Setup(1^\lambda, \ell\ell')$ where $\mathbf{vp} = (\mathbf{vp}_\ell, \mathbf{vp}_{\ell'}, \mathbf{vp}_1)$. Let $(\mathbf{cm}, \mathbf{st}) \leftarrow \Pi_{\ell\ell'}.Com(\mathbf{cp}, \mathbf{x})$ and $\pi_h \leftarrow \Pi_{\ell\ell'}.Open(\mathbf{st}, h)$. Parse $\pi_{h=(i,j)} = (\pi'_i, o^i, \pi_j^i, \mathbf{cm}^i, m^i)$ where π'_i is the i -th opening of the $\Pi_{\ell'}$ -commitment $\mathbf{cm}$ for vector $(m^1, \dots, m^{\ell'})$, and m^i is the Π_1 -commitment to $\mathbf{cm}^i$ with opening o^i , and $\mathbf{cm}^i$ is the Π_ℓ -commitment to vector $(\mathbf{x}_{(i-1)\cdot\ell+1}, \dots, \mathbf{x}_{i\ell})$, π_j^i is the opening of $\mathbf{cm}^i$ at position $j \in [\ell]$.

Construction A.1.

- $\text{Setup}(1^\lambda, \ell \cdot \ell') \rightarrow (\text{cp}, \text{vp})$:
 - $(\text{cp}_{\ell'}, \text{vp}_{\ell'}) \leftarrow \Pi_{\ell'}. \text{Setup}(1^\lambda, \ell')$, $(\text{cp}_\ell, \text{vp}_\ell) \leftarrow \Pi_\ell. \text{Setup}(1^\lambda, \ell)$,
 $(\text{cp}_1, \text{vp}_1) \leftarrow \Pi_1. \text{Setup}(1^\lambda, 1)$
 - output $\text{cp} := (\text{cp}_{\ell'}, \text{cp}_\ell, \text{cp}_1)$ and $\text{vp} := (\text{vp}_{\ell'}, \text{vp}_\ell, \text{vp}_1)$
- $\text{Com}(\text{cp}, \mathbf{x}) \rightarrow (\text{cm}, \text{st})$:
 - parse $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^{\ell'})$ where $\mathbf{x}^i \in (\bar{R}_q^{n_m})^\ell$ for all $i \in [\ell']$
 - $(\text{cm}^i, \text{st}^i) \leftarrow \Pi_\ell. \text{Com}(\text{cp}_\ell, \mathbf{x}^i)$ for all $i \in [\ell']$
 - $(m^i, \text{st}_o^i) \leftarrow \Pi_1. \text{Com}(\text{cp}_1, \text{cm}^i)$ for all $i \in [\ell']$
 - $(\text{cm}, \text{st}') \leftarrow \Pi_{\ell'}. \text{Com}(\text{cp}_{\ell'}, (m_1, \dots, m^{\ell'}))$
 - output cm and $\text{st} := (\text{st}', (\text{st}^i, \text{st}_o^i)_{i \in [\ell']}, (\text{cm}^i, m^i)_{i \in [\ell']})$
- $\text{Open}(\text{st}, h = (i \in [\ell'], j \in [\ell])) \rightarrow \pi_h$:
 - parse $\text{st} := (\text{st}', (\text{st}^k, \text{st}_o^k)_{k \in [\ell']}, (\text{cm}^k, m^k)_{k \in [\ell']})$
 - $\pi_j^i \leftarrow \Pi_\ell. \text{Open}(\text{st}^i, j)$
 - $o^i \leftarrow \Pi_1. \text{Open}(\text{st}_o^i, 1)$
 - $\pi'_i \leftarrow \Pi_{\ell'}. \text{Open}(\text{st}', i)$
 - output $\pi_h := (\pi'_i, o^i, \pi_j^i, \text{cm}^i, m^i)^a$
- $\text{Vf}(\text{vp}, h = (i \in [\ell'], j \in [\ell])), \text{cm}, \mathbf{x}_h, \pi_h) \rightarrow 0/1$:
 - parse $\pi_h := (\pi'_i, o^i, \pi_j^i, \text{cm}^i, m^i)$
 - output 1 if $\Pi_{\ell'}. \text{Vf}(\text{vp}_{\ell'}, i, \text{cm}, m^i, \pi'_i) = 1$ and $\Pi_1. \text{Vf}(\text{vp}_1, 1, m^i, \text{cm}^i, o^i) = 1$
 and $\Pi_\ell. \text{Vf}(\text{vp}_\ell, j, \text{cm}^i, \mathbf{x}_h, \pi_j^i) = 1$, otherwise, output 0
- $\text{wVf}(\text{vp}, h, \text{cm}, \mathbf{x}_h, \pi_h) \rightarrow 0/1$: Same as $\text{Vf}(\cdot)$, except that the calls to Vf in Π_ℓ , $\Pi_{\ell'}$, Π_1 are replaced with wVf

^aIn many cases, cm^i uniquely determines o^i, m^i , so there's no need to include o^i, m^i in the proof.

Figure 16: The $\ell \cdot \ell'$ -slot homomorphic vector commitment scheme $\Pi_{\ell\ell'}$.

Since Π_1 , Π_ℓ and $\Pi_{\ell'}$ are correct, we have that

$$\Pi_{\ell'}. \text{Vf}(\text{vp}_{\ell'}, i, \text{cm}, m^i, \pi'_i) = 1, \quad \Pi_1. \text{Vf}(\text{vp}_1, 1, m^i, \text{cm}^i, o^i) = 1$$

and

$$\Pi_\ell. \text{Vf}(\text{vp}_\ell, j, \text{cm}^i, \mathbf{x}_{(i-1)\ell+j}, \pi_j^i) = 1.$$

By construction of $\Pi_{\ell\ell'}$, this implies that $\Pi_{\ell\ell'}. \text{Vf}(\text{vp}, h, \text{cm}, \mathbf{x}_h, \pi_h) = 1$, finishing the proof. $\square$

Claim 15. Let $k \in \mathbb{N}$, $\mathcal{S} \subseteq \bar{R}$, and let $\mathcal{O}_{\ell'}$, $\mathcal{O}_\ell$, $\mathcal{O}_1$ be some opening sets. Define $\mathcal{O} := \mathcal{O}_{\ell'} \times \mathcal{O}_1 \times \mathcal{O}_\ell \times \mathcal{C} \times \mathcal{M}$ where $\mathcal{C}$, $\mathcal{M}$ are the commitment and message spaces of Π_1 . Then $\Pi_{\ell\ell'}$ is $(k, \mathcal{S}, \mathcal{O})$ -homomorphically correct for valid openings (Definition 3.3) if for all $\mu \in \{1, \ell, \ell'\}$, it holds that Π_μ satisfies $(k, \mathcal{S}, \mathcal{O}_t)$ -homomorphic correctness for valid openings.

Proof. Fix any $k' \in [k]$, vectors $\mathbf{x}^1, \dots, \mathbf{x}^{k'} \in (\bar{R}_q^{n_m})^{\ell\ell'}$, scalars $\alpha^1, \dots, \alpha^{k'} \in \mathcal{S}$, and index $h = (i, j) \in [\ell'] \times [\ell]$. Let $(\text{cp}, \text{vp}) \leftarrow \Pi_{\ell\ell'}. \text{Setup}(1^\lambda, \ell\ell')$ where $\text{vp} = (\text{vp}_\ell, \text{vp}_{\ell'}, \text{vp}_1)$. For every $t \in [k']$, let $(\text{cm}^t, \text{st}^t) \leftarrow \Pi_{\ell\ell'}. \text{Com}(\text{cp}, \mathbf{x}^t)$ and let $\pi_h^t \in \mathcal{O}$ be an opening such that $\Pi_{\ell\ell'}. \text{Vf}(\text{vp}, h, \text{cm}^t, \mathbf{x}_h^t, \pi_h^t) = 1$. Our goal is to check that

$$\Pi_{\ell\ell'}. \text{Vf}(\text{vp}, h, \sum_{t \in [k']} \alpha^t \cdot (\text{cm}^t, \mathbf{x}_h^t, \pi_h^t)) = 1. \quad (25)$$

For every $t \in [k']$, parse $\pi_{h=(i,j)}^t = (\pi_i^{t,i}, o^{t,i}, \pi_j^{t,i}, \text{cm}^{t,i}, m^{t,i}) \in \mathcal{O}_{\ell'} \times \mathcal{O}_1 \times \mathcal{O}_\ell \times \mathcal{C} \times \mathcal{M}$. Since $\Pi_{\ell\ell'}. \text{Vf}(\text{vp}, h, \text{cm}^t, \mathbf{x}_h^t, \pi_h^t) = 1$, we have that

$$\Pi_{\ell'}. \text{Vf}(\text{vp}_{\ell'}, i, \text{cm}^t, m^{t,i}, \pi_i^{t,i}) = 1, \quad \Pi_1. \text{Vf}(\text{vp}_1, 1, m^{t,i}, \text{cm}^{t,i}, o^{t,i}) = 1$$

and

$$\Pi_\ell. \text{Vf}(\text{vp}_\ell, j, \text{cm}^{t,i}, \mathbf{x}_{(i-1)\ell+j}^t, \pi_j^{t,i}) = 1.$$

Recall that for all $\mu \in \{1, \ell, \ell'\}$, Π_μ is $(k, \mathcal{S}, \mathcal{O}_t)$ -homomorphically correct for valid openings, hence

$$\Pi_{\ell'}. \text{Vf}(\text{vp}_{\ell'}, i, \sum_{t \in [k']} \alpha^t \cdot (\text{cm}^t, m^{t,i}, \pi_i^{t,i})) = 1, \quad \Pi_1. \text{Vf}(\text{vp}_1, 1, \sum_{t \in [k']} \alpha^t \cdot (m^{t,i}, \text{cm}^{t,i}, o^{t,i})) = 1$$

and

$$\Pi_\ell. \text{Vf}(\text{vp}_\ell, j, \sum_{t \in [k']} \alpha^t \cdot (\text{cm}^{t,i}, \mathbf{x}_{(i-1)\ell+j}^t, \pi_j^{t,i})) = 1.$$

By construction of $\Pi_{\ell\ell'}$, this implies that Eq. (25) holds, finishing the proof. $\square$

Claim 16. If Π_1 , Π_ℓ and $\Pi_{\ell'}$ satisfy linearity of opening states (Definition 3.4), then $\Pi_{\ell\ell'}$ also satisfies linearity of opening states.

Proof. Fix any index $h = (i, j) \in [\ell'] \times [\ell]$, vector $\mathbf{x} \in (\bar{R}_q^{n_m})^{\ell\ell'}$. Let $(\mathbf{cp}, \mathbf{vp}) \leftarrow \Pi_{\ell\ell'}. \text{Setup}(1^\lambda, \ell\ell')$ and $(\mathbf{cm}, \mathbf{st}) \leftarrow \Pi_{\ell\ell'}. \text{Com}(\mathbf{cp}, \mathbf{x})$. For any $k \in \mathbb{N}$, $\alpha^1, \dots, \alpha^k \in \bar{R}$ and $\mathbf{st}_1, \dots, \mathbf{st}_k \in \mathcal{Z}$ such that $(\mathbf{st} = \sum_{t \in [k]} \alpha^t \cdot \mathbf{st}_t)$, we claim that

$$\sum_{t \in [k]} \alpha^t \cdot \Pi_{\ell\ell'}. \text{Open}(\mathbf{st}_t, h) = \Pi_{\ell\ell'}. \text{Open}(\mathbf{st}, h). \quad (26)$$

Parse $\mathbf{st} := (\mathbf{st}', (\mathbf{st}^i, \mathbf{st}_o^i, \mathbf{cm}^i, m^i), \dots)$ and for all $t \in [k]$, parse $\mathbf{st}_t := (\mathbf{st}'_t, (\mathbf{st}_t^i, \mathbf{st}_{o,t}^i, \mathbf{cm}_t^i, m_t^i), \dots)$. Note that $(\mathbf{st} = \sum_{t \in [k]} \alpha^t \cdot \mathbf{st}_t)$, so it holds that

$$\mathbf{cm}^i = \sum_{t \in [k]} \alpha^t \cdot \mathbf{cm}_t^i, \quad m^i = \sum_{t \in [k]} \alpha^t \cdot m_t^i.$$

Moreover, by the linearity of opening states of Π_ℓ and $\Pi_{\ell'}$, we have that

$$\sum_{t \in [k]} \alpha^t \cdot \Pi_{\ell'}. \text{Open}(\mathbf{st}'_t, i) = \Pi_{\ell'}. \text{Open}(\mathbf{st}', i), \quad \sum_{t \in [k]} \alpha^t \cdot \Pi_1. \text{Open}(\mathbf{st}_{o,t}^i, 1) = \Pi_1. \text{Open}(\mathbf{st}_o^i, 1),$$

and

$$\sum_{t \in [k]} \alpha^t \cdot \Pi_\ell. \text{Open}(\mathbf{st}_t^i, j) = \Pi_\ell. \text{Open}(\mathbf{st}^i, j).$$

By construction of $\Pi_{\ell\ell'}$, this implies that Eq. (26) holds, finishing the proof. $\square$

Claim 17. $\Pi_{\ell\ell'}$ is robustly homomorphic (Definition 3.5) if $\Pi_1, \Pi_\ell, \Pi_{\ell'}$ are.

Proof. Let $(\mathbf{cp}, \mathbf{vp}) \leftarrow \Pi_{\ell\ell'}. \text{Setup}(1^\lambda, \ell\ell')$ where $\mathbf{vp} = (\mathbf{vp}_\ell, \mathbf{vp}_{\ell'}, \mathbf{vp}_1)$. Fix any $h = (i, j) \in [\ell'] \times [\ell]$ and tuples $(\mathbf{cm}, x_h, \pi)$, $(\hat{\mathbf{cm}}, \hat{x}_h, \hat{\pi})$ where

$$\Pi_{\ell\ell'}. \text{Vf}(\mathbf{vp}, h, \mathbf{cm}, x_h, \pi) = \Pi_{\ell\ell'}. \text{Vf}(\mathbf{vp}, h, \hat{\mathbf{cm}}, \hat{x}_h, \hat{\pi}) = 1. \quad (27)$$

Our goal is to check that

$$\Pi_{\ell\ell'}. \text{wVf}(\mathbf{vp}, h, \mathbf{cm} - \hat{\mathbf{cm}}, x_h - \hat{x}_h, \pi - \hat{\pi}) = 1. \quad (28)$$

Denote by $\pi = (\pi_i', o^i, \pi_j^i, \mathbf{cm}^i, m^i)$ and $\hat{\pi} = (\hat{\pi}_i', \hat{o}^i, \hat{\pi}_j^i, \hat{\mathbf{cm}}^i, \hat{m}^i)$. By construction of $\Pi_{\ell\ell'}$, Eq. (27) implies that

$$\Pi_{\ell'}. \text{Vf}(\mathbf{vp}_{\ell'}, i, \mathbf{cm}, m^i, \pi_i') = \Pi_1. \text{Vf}(\mathbf{vp}_1, 1, m^i, \mathbf{cm}^i, o^i) = \Pi_\ell. \text{Vf}(\mathbf{vp}_\ell, j, \mathbf{cm}^i, x_h, \pi_j^i) = 1$$

and

$$\Pi_{\ell'}. \text{Vf}(\mathbf{vp}_{\ell'}, i, \hat{\mathbf{cm}}, \hat{m}^i, \hat{\pi}_i') = \Pi_1. \text{Vf}(\mathbf{vp}_1, 1, \hat{m}^i, \hat{\mathbf{cm}}^i, \hat{o}^i) = \Pi_\ell. \text{Vf}(\mathbf{vp}_\ell, j, \hat{\mathbf{cm}}^i, \hat{x}_h, \hat{\pi}_j^i) = 1.$$

Since $\Pi_1, \Pi_\ell, \Pi_{\ell'}$ are robustly homomorphic, it holds that

$$\begin{aligned}\Pi_{\ell'}.wVf(\mathbf{vp}_{\ell'}, i, \mathbf{cm} - \hat{\mathbf{cm}}, m^i - \hat{m}^i, \pi'_i - \hat{\pi}'_i) &= 1, \\ \Pi_1.wVf(\mathbf{vp}_1, 1, m^i - \hat{m}^i, \mathbf{cm}^i - \hat{\mathbf{cm}}^i, o^i - \hat{o}^i) &= 1, \\ \Pi_\ell.wVf(\mathbf{vp}_\ell, j, \mathbf{cm}^i - \hat{\mathbf{cm}}^i, x_h - \hat{x}_h, \pi_j^i - \hat{\pi}_j^i) &= 1.\end{aligned}$$

By construction of $\Pi_{\ell\ell'}$, this implies that Eq. (28) holds, finishing the proof. $\square$

Claim 18. *If Π_1, Π_ℓ and $\Pi_{\ell'}$ are $\mathcal{W}$ -weakly verifiable for $\mathcal{W} \subseteq \bar{R}$ (Definition 3.6), then $\Pi_{\ell\ell'}$ is also $\mathcal{W}$ -weakly verifiable.*

Proof. Let $(\mathbf{cp}, \mathbf{vp}) \leftarrow \Pi_{\ell\ell'}.Setup(1^\lambda, \ell\ell')$ where $\mathbf{vp} = (\mathbf{vp}_\ell, \mathbf{vp}_{\ell'}, \mathbf{vp}_1)$. Fix any index $h = (i, j) \in [\ell'] \times [\ell]$, value $\mathbf{aux} \in \mathcal{W}$ and tuple $(\mathbf{cm}, x_h, \pi)$ such that $\Pi_{\ell\ell'}.Vf(\mathbf{vp}, h, \mathbf{cm}, x_h, \pi) = 1$. We want to show that

$$\Pi_{\ell\ell'}.wVf(\mathbf{vp}, h, \mathbf{aux} \cdot (\mathbf{cm}, x_h, \pi)) = 1.$$

Denote by $\pi = (\pi'_i, o^i, \pi_j^i, \mathbf{cm}^i, m^i)$. Recall that $\Pi_{\ell\ell'}.Vf(\mathbf{vp}, h, \mathbf{cm}, x_h, \pi) = 1$ if and only if

$$\Pi_{\ell'}.Vf(\mathbf{vp}_{\ell'}, i, \mathbf{cm}, m^i, \pi'_i) = \Pi_1.Vf(\mathbf{vp}_1, 1, m^i, \mathbf{cm}^i, o^i) = \Pi_\ell.Vf(\mathbf{vp}_\ell, j, \mathbf{cm}^i, x_h, \pi_j^i) = 1.$$

Since Π_1, Π_ℓ and $\Pi_{\ell'}$ are $\mathcal{W}$ -weakly verifiable, it holds that

$$\begin{aligned}\Pi_{\ell'}.wVf(\mathbf{vp}_{\ell'}, i, \mathbf{aux} \cdot (\mathbf{cm}, m^i, \pi'_i)) &= 1, \\ \Pi_1.wVf(\mathbf{vp}_1, 1, \mathbf{aux} \cdot (m^i, \mathbf{cm}^i, o^i)) &= 1, \\ \Pi_\ell.wVf(\mathbf{vp}_\ell, j, \mathbf{aux} \cdot (\mathbf{cm}^i, x_h, \pi_j^i)) &= 1.\end{aligned}$$

By construction of $\Pi_{\ell\ell'}$, this implies that $\Pi_{\ell\ell'}.wVf(\mathbf{vp}, h, \mathbf{aux} \cdot (\mathbf{cm}, x_h, \pi)) = 1$ as desired. $\square$

Claim 19. *$\Pi_{\ell\ell'}$ is position binding (Definition 3.7) if $\Pi_1, \Pi_\ell, \Pi_{\ell'}$ are.*

Proof. Let $(\mathbf{cp}, \mathbf{vp}) \leftarrow \Pi_{\ell\ell'}.Setup(1^\lambda, \ell\ell')$ where $\mathbf{vp} = (\mathbf{vp}_\ell, \mathbf{vp}_{\ell'}, \mathbf{vp}_1)$. Suppose for contradiction that there exists an adversary $\mathcal{A}$ such that with non-negligible probability, $\mathcal{A}$ outputs a tuple $(h, \mathbf{cm}, x_h, \hat{x}_h, \pi, \hat{\pi})$ where $x_h \neq \hat{x}_h$ and

$$\Pi_{\ell\ell'}.wVf(\mathbf{vp}, h, \mathbf{cm}, x_h, \pi) = \Pi_{\ell\ell'}.wVf(\mathbf{vp}, h, \mathbf{cm}, \hat{x}_h, \hat{\pi}) = 1. \quad (29)$$

Next, given $(h, \mathbf{cm}, x_h, \hat{x}_h, \pi, \hat{\pi})$, we will show how to obtain a tuple that breaks the position binding property of either $\Pi_\ell, \Pi_{\ell'}$ or Π_1 .

Denote by $h = (i, j) \in [\ell'] \times [\ell]$, opening proofs $\pi = (\pi'_i, o^i, \pi_j^i, \mathbf{cm}^i, m^i)$ and $\hat{\pi} = (\hat{\pi}'_i, \hat{o}^i, \hat{\pi}_j^i, \hat{\mathbf{cm}}^i, \hat{m}^i)$. Eq. (29) implies that

$$\Pi_{\ell'}.wVf(\mathbf{vp}_{\ell'}, i, \mathbf{cm}, m^i, \pi'_i) = \Pi_1.wVf(\mathbf{vp}_1, 1, m^i, \mathbf{cm}^i, o^i) = \Pi_\ell.wVf(\mathbf{vp}_\ell, j, \mathbf{cm}^i, x_h, \pi_j^i) = 1$$

and

$$\Pi_{\ell'}.wVf(vp_{\ell'}, i, cm, \hat{m}^i, \hat{\pi}'_i) = \Pi_1.wVf(vp_1, 1, \hat{m}^i, \hat{cm}^i, \hat{o}^i) = \Pi_{\ell}.wVf(vp_{\ell}, j, \hat{cm}^i, \hat{x}_h, \hat{\pi}'_j) = 1.$$

There are 3 possible cases:

- **Case $cm^i = \hat{cm}^i$:** Since $\Pi_{\ell}.wVf(vp_{\ell}, j, cm^i, x_h, \pi_j^i) = \Pi_{\ell}.wVf(vp_{\ell}, j, \hat{cm}^i, \hat{x}_h, \hat{\pi}_j^i) = 1$ and $x_h \neq \hat{x}_h$, the tuple $(j, cm^i, x_h, \hat{x}_h, \pi_j^i, \hat{\pi}_j^i)$ breaks position binding of Π_{ℓ} .
- **Case $(cm^i \neq \hat{cm}^i) \wedge (m^i = \hat{m}^i)$:** Since $\Pi_1.wVf(vp_1, 1, m^i, cm^i, o^i)$ and $\Pi_1.wVf(vp_1, 1, \hat{m}^i, \hat{cm}^i, \hat{o}^i)$ both equal 1, the tuple $(1, m^i, cm^i, \hat{cm}^i, o^i, \hat{o}^i)$ breaks position binding of Π_1 .
- **Case $(cm^i \neq \hat{cm}^i) \wedge (m^i \neq \hat{m}^i)$:** Since $\Pi_{\ell'}.wVf(vp_{\ell'}, i, cm, m^i, \pi'_i)$ and $\Pi_{\ell'}.wVf(vp_{\ell'}, i, cm, \hat{m}^i, \hat{\pi}'_i)$ both equal 1, the tuple $(i, cm, m^i, \hat{m}^i, \pi'_i, \hat{\pi}'_i)$ breaks position binding of $\Pi_{\ell'}$.

In sum, let ϵ denote $\mathcal{A}$'s success probability w.r.t. $\Pi_{\ell'}$. Then we derive an adversary that breaks the position binding property of one of Π_{ℓ} , $\Pi_{\ell'}$ and Π_1 with probability at least $\epsilon/3$, contradicting with the fact that Π_{ℓ} , $\Pi_{\ell'}$ and Π_1 are all position binding. $\square$

$\square$

B The Relationship between Different Primitives

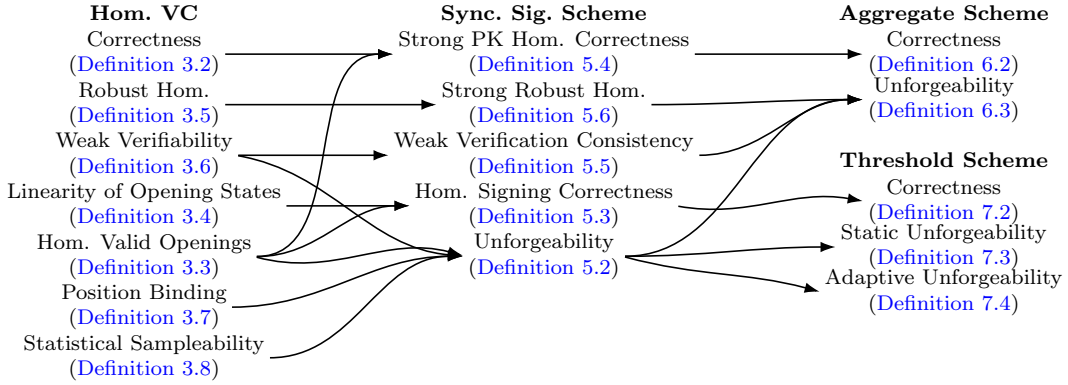


Figure 17: This figure explains the relationship between the properties of our vector commitment scheme, synchronous signature scheme, and aggregate and threshold schemes.

C Proof of Lemma 7.5

Proof. Because the lemma is for a single (m, h, S) , we drop the (m, h, S) and S superscripts for clarity. Say $\mathbf{A} \in \Sigma^{\mu \times (|S|^2 - |S|)}$. We prove that $\text{rows}(\mathbf{A})$ form a partial basis of $\Sigma^{|S|^2 - |S|}$ by showing that we can select μ columns of $\mathbf{A}$ to obtain a $\mu \times \mu$ matrix $\bar{\mathbf{A}}$ whose columns form a basis of Σ^μ . To construct $\bar{\mathbf{A}}$, we iterate through $\text{rows}(\mathbf{A})$ in a particular order. For the i -th iteration, we pick some column j_i in $\mathbf{A}$ to add to $\bar{\mathbf{A}}$. For this added column, the i -th row has a 1_Σ or -1_Σ and every row $i' < i$ has a 0_Σ . If we add columns to $\bar{\mathbf{A}}$ such that this invariant is always satisfied, then the columns of $\bar{\mathbf{A}}$ form a basis of Σ^μ .

For clarity, we introduce some notation: let $\text{idx}(i, j) = \nu$ such that $\mathbf{a}_{i,j}$ has a 1_Σ in position ν and a 0_Σ in all other positions. The rows in $\mathbf{A}$ are either standard basis vectors $\mathbf{a}_{i,j}$ or partial signature query vectors $\mathbf{a}_i = \sum_{j \in S \setminus \{i\}} \mathbf{a}_{i,j} - \mathbf{a}_{j,i}$. For any $i, j \in S$, the possible rows of $\mathbf{A}$ that could have non-zero values at position $\text{idx}(i, j)$ are $\mathbf{a}_{i,j}$, $\mathbf{a}_i$, and $\mathbf{a}_j$. This means that our invariant reduces to:

- If the k -th row in our iteration is a standard basis vector $\mathbf{a}_{i,j}$, we must add column $\text{idx}(i, j)$ to $\bar{\mathbf{A}}$, and we must not have iterated over $\mathbf{a}_i$ or $\mathbf{a}_j$. If the k -th row in our iteration is a partial signature query vector $\mathbf{a}_i$, then for some $j \in S, j \neq i$, we either we add column $\text{idx}(i, j)$ to $\bar{\mathbf{A}}$, and we must have not iterated over $\mathbf{a}_{i,j}$ or $\mathbf{a}_j$, or we add column $\text{idx}(j, i)$ to $\bar{\mathbf{A}}$, and we must have not iterated over $\mathbf{a}_{j,i}$ or $\mathbf{a}_j$.

We first consider the rows in $\mathbf{A}$ that are standard basis vectors $\mathbf{a}_{i,j}$. If $\mathbf{a}_{i,j} \in \text{rows}(\mathbf{A})$, then at least one of i or j must be corrupted. We iterate through the standard basis rows in any order, and for each $\mathbf{a}_{i,j} \in \text{rows}(\mathbf{A})$, we add column $\text{idx}(i, j)$ to $\bar{\mathbf{A}}$. Because we are only iterating over rows that are standard basis vectors, the invariant holds.

We now consider the rest of the rows in $\mathbf{A}$, i.e., the partial signature query vectors. Let signers be the set of parties for which the adversary has issued a partial signature query on (m, h, S) . From the definition of $\text{OPartialSignQuorum}$ in G_6 , there must be at least one party i such that $\mathbf{a}_i \notin \text{rows}(\mathbf{A})$. If i is corrupt, let $c^* = i$; else $c^* = \perp$.

Let $C_{<c^*}$ be the set of $i \in \text{signers} \cap C$ such that i was corrupted before c^* , and let $C_{>c^*}$ be the set of $i \in \text{signers} \cap C$ such that i was corrupted after c^* . If $c^* = \perp$, then $C_{<c^*} = \text{signers} \cap C$, and $C_{>c^*} = \emptyset$. If $c^* \neq \perp$, then all for $i \in \text{signers} \setminus \{c^*\}$, $\mathbf{a}_i$ must be a row in $\mathbf{A}$. This means that for all $i \in C_{<c^*} \cup C_{>c^*}$, $\mathbf{a}_i \in \text{rows}(\mathbf{A})$.

We can partition the partial signature vector rows of $\mathbf{A}$ into three sets: $\{\mathbf{a}_i : i \in C_{<c^*}\}$, $\{\mathbf{a}_i : i \in C_{>c^*}\}$, and $\{\mathbf{a}_i \in \text{rows}(\mathbf{A}) : i \notin C\}$.

For all $i \in C_{<c^*} \cup C_{>c^*}$, we know $i \in \text{signers}$ by definition, so when the adversary corrupted i , the challenger chose a party k_i that was honest at the time of corruption and did not add $\mathbf{a}_{k_i,i}$ as a row to $\mathbf{A}$. If the adversary later corrupted party k_i , then the challenger would not have added $\mathbf{a}_{k_i,i}$ as a row to $\mathbf{A}$ during this corruption because $i \in C$, so $\mathbf{a}_{k_i,i} \notin \text{rows}(\mathbf{A})$.

Let K be the set of these chosen k_i 's for the parties corrupted after c^* and also c^* ; that is, $K = \{k_i : i \in C_{>c^*} \cup \{c^*\}\}$. After c^* is corrupted, $|S \setminus (\text{signers} \cup C)| = 0$, so for all

$i \in C_{>c^*}$, the challenger chooses $k_i := \min(S \setminus C)$. Thus, in consecutive corruption queries for i', i , either $k_{i'} = k_i$, or $k_i > k_{i'}$ and the second query is a corruption of $k_{i'}$. That is, for any $i \in K \setminus \{k_{c^*}\}$, there is some $i' \in K$ such that $k_{i'} = i$ and also $i' < i$.

If $c^* = \perp$, then there must be some honest party j^* for which $\mathbf{a}_j$ is not in the rows of $\mathbf{A}$. Else if $c^* \neq \perp$, let $j^* = h_c$, where c is the last party in $C_{>c^*}$ corrupted. Because c is the last party corrupted, h_c is still honest, so regardless of whether $c^* = \perp$, j^* is honest.

We now iterate over the partial signature vector rows in $\mathbf{A}$:

1. We iterate over rows $\{\mathbf{a}_i : i \in C_{<c^*}\}$ in any order. For the row $\mathbf{a}_i$, we add column $\text{idx}(k_i, i)$ to $\overline{\mathbf{A}}$. We argued earlier that $\mathbf{a}_{k_i, i} \notin \text{rows}(\mathbf{A})$. Because we iterate in order of corruption and we said that party k_i must have been honest at the time that party i was corrupted, if $\mathbf{a}_{k_i} \in \text{rows}(\mathbf{A})$, we have not yet iterated over it yet, and the invariant holds.
2. We next iterate over rows $\{\mathbf{a}_i : i \notin C \cup \{j^*\}\}$. For any $\mathbf{a}_i$, we add column $\text{idx}(i, j^*)$ to $\overline{\mathbf{A}}$. Because i and j^* are both honest by assumption, $\mathbf{a}_{i, j^*} \notin \text{rows}(\mathbf{A})$. Because we exclude j^* from our iteration set and j^* is honest and therefore not in $C_{<c^*}$, we have not yet iterated over $\mathbf{a}_{j^*}$, so the invariant holds.
3. If $c^* = \perp$, then $C_{>c^*} = \emptyset$, and we have finished iterating all rows of $\mathbf{A}$, so this step is unnecessary. Else:
 - We first iterate over $i \in C_{>c^*} \setminus K$ in any order. For every row $\mathbf{a}_i$, we add column $\text{idx}(k_i, i)$ to $\overline{\mathbf{A}}$. We argued earlier that $\mathbf{a}_{k_i, i} \notin \text{rows}(\mathbf{A})$. Because we are iterating over $C_{>c^*} \setminus K$ and $K = \{k_i : i \in C_{>c^*} \cup \{c^*\}\}$, we have also clearly not yet iterated over $\mathbf{a}_{k_i}$, so the invariant holds.
 - We lastly iterate over K from the largest element to smallest element. By definition, the last element in K we iterate over is k_{c^*} . For $i \in K, i \neq k_{c^*}$, we argued earlier that there is some $i' \in K$ such that $k_{i'} = i$ and also $i' < i$. For such k_i , we add column $\text{idx}(i, i') = \text{idx}(k_{i'}, i')$ to $\overline{\mathbf{A}}$. We argued earlier that $\mathbf{a}_{k_{i'}, i'}$ is never a row of $\mathbf{A}$. Because $i' \in K, i' < i$, and we are iterating over K from largest to smallest, we have not yet iterated over $\mathbf{a}_{i'}$, so the invariant holds. For k_{c^*} , we add column $\text{idx}(k_{c^*}, c^*)$ to $\overline{\mathbf{A}}$. $\mathbf{a}_{k_{c^*}, c^*}$ is never a row of $\mathbf{A}$, and by definition $\mathbf{a}_{c^*} \notin \text{rows}(\mathbf{A})$.

□