LaTSEC: Lattice-Based Threshold Signatures with Native Non-Interactive Identifiable Abort and Compact Per-Party Storage

Muhammad Arzaki*, Muhammed F. Esgin*, Markku-Juhani O. Saarinen†, Amin Sakzad*, and Ron Steinfeld*
*Faculty of Information Technology, Monash University, Melbourne, Australia
{muhammad.arzaki, muhammed.esgin, amin.sakzad, ron.steinfeld}@monash.edu
†Faculty of Information Technology and Communication Sciences, Tampere University, Tampere, Finland
markku-juhani.saarinen@tuni.fi

***Abstract*—We present LaTSEC, a three-round lattice-based threshold signature scheme with native non-interactive identifiable abort. Each party holds one small-norm signing share and is associated with one public abort key, against which its partial signature is verified directly, without zero-knowledge proofs or an additional interactive protocol. Prior lattice-based threshold signatures that identify misbehaving parties rely on one of three mechanisms: zero-knowledge proofs, an additional interactive protocol, or an everywhere-short secret sharing scheme in which each party holds multiple signing shares, each with its own public abort key. Compared with schemes that use the third mechanism, LaTSEC substantially reduces per-party storage. The construction uses SEC, a recent ramp everywhere-short secret sharing scheme based on lossless expanders that gives each party a *single* short share. We prove correctness, identifiable-abort security, and threshold unforgeability under the standard MLWE and MSIS assumptions. The unforgeability proof introduces the *Repeated-A Matrix Hint-MLWE* problem, a new variant of MLWE with hints that captures the complete tuple of correlated public abort keys. We rigorously prove that this new variant admits a reduction from the standard MLWE problem. For 60 parties at 128-bit security, LaTSEC tolerates up to 6 corrupted parties, and, empirically, a randomly chosen set of at least 46 parties can sign with probability at least 0.99. Each party holds one signing share of about 14.3 KiB and is associated with one 24-KiB public abort key.**

1. Introduction

A threshold signature [1] distributes a signing key among n parties so that sufficiently large sets can sign jointly, while smaller coalitions cannot forge. Threshold signatures have been built from RSA [2], Schnorr [3], [4], [5], BLS [6], and ECDSA [7], [8], [9]. FROST [3] is a noteworthy example that also provides *native non-interactive identifiable abort*: each partial signature is checked directly against a participant-specific public key, so a failed session exposes the misbehaving parties without further interaction or proofs. Several threshold ECDSA schemes also provide identifiable abort [7], [8], and ROAST turns FROST into a robust, asynchronous threshold signature [4]. However, these schemes rely on mathematical problems that are vulnerable to large-scale quantum computers, and there is a need to develop *post-quantum* threshold signatures with practical efficiency. The NIST call for multi-party threshold schemes [10] has made this a concrete target.

Lattice-based threshold signatures. Lattice-based threshold and distributed signatures use trapdoor sampling [11], hash-and-sign with noise flooding [12], generic multi-party computation (MPC) [13], threshold homomorphic encryption (THE) [14], [15], [16], or Fiat-Shamir signatures, either with aborts in distributed and multi-signature settings [17], [18], [19], [20] or without aborts in the Raccoon style [21], [22]. Trapdoor sampling and noise flooding impose strong parameter constraints that make the resulting schemes costly for large thresholds. For example, the per-party communication of Pelican grows linearly with the threshold [12]. Generic MPC and THE incur high communication costs, reaching 3 MB per party in the THE-based GKS scheme of Gur et al. [16]. In distributed Fiat-Shamir signatures with aborts, a rejection by any signer restarts the signing protocol. Raccoon removes this rejection step [21], and Threshold Raccoon builds on it to obtain practical threshold signatures from standard lattice assumptions [22]. Our scheme LaTSEC follows this Raccoon-style approach. A separate line targets compatibility with ML-DSA, either with more signing rounds [23] or for small committees [24]. For practical lattice-based threshold signatures with low communication and security under standard assumptions, a central remaining question is how to handle malicious signers.

Handling malicious signers. In Raccoon-style signatures, the response $\boldsymbol{z} = c\boldsymbol{s}+\boldsymbol{r}$ must stay short, that is, have small norm [21]—we use *short* in this sense throughout. A malicious signer can cause signing to fail, and how a scheme identifies it depends on its secret sharing: existing schemes use either uniform shares, mostly from Shamir sharing, or everywhere-short sharing, which we discuss next and build on.

With Shamir sharing [25], a partial response $c\lambda_i\boldsymbol{s}_i + \boldsymbol{r}_i$ involves a uniform share $\boldsymbol{s}_i$ and a possibly large Lagrange coefficient λ_i, so it is neither short nor safe to reveal [22], [26]. Threshold Raccoon [22] hides it with one-time additive masks, an approach adopted by later schemes with adaptive security [27], partially non-interactive two-round

signing [28], [29], or two-round signing under standard MLWE [30]. Chairattana-Apirom et al. instead use Benaloh-Leichter sharing [31] with coefficients in $\{0, \pm 1\}$, but still with uniform shares [32]. In all of these schemes, partial responses cannot be checked directly against public data, so identifying misbehaving signers requires an additional mechanism.

TRaccoon-IA runs an interactive protocol after a failed aggregation [33], whereas GKS and Olingo attach non-interactive zero-knowledge proofs to their signing messages, with per-party communication of 3 MB and 852 kB, respectively [16], [34]. Identifiable abort also yields robustness, since the honest signers can exclude the identified parties and retry [4]. *Robust* schemes instead guarantee that the protocol directly outputs a signature whenever sufficiently many honest signers participate: Bendlin et al. give robust protocols for trapdoor sampling [11], while Pelican, built on Plover, relies on an approximate shortness proof in the honest-majority setting [12], [35].

Everywhere-short sharing and its cost. Native non-interactive identifiable abort over lattices becomes possible with *everywhere-short* secret sharing, in which the secret, the shares, and the reconstruction coefficients are all short: each partial response is then short and can be verified against participant-specific public material [26], [36]. The sDKG framework of del Pino et al. instantiates this idea using replicated and random-assignment ramp sharing [36], whereas Hermine uses Vandermonde sharing and adds partially non-interactive signing and proactive security [26], [37], [38]. For n participants, signing threshold t, and security parameter κ, these constructions pay for shortness through the number of short sub-shares held by each participant: $\binom{n-1}{t-1}$ under replicated sharing, one from each of $\mu = \kappa$ parallel sharings under the ramp construction, and $O\big(\binom{t-1+\lceil \lg n \rceil}{t-1}\big)$ under Vandermonde sharing [38]. Because native non-interactive identifiable abort associates public verification material with every sub-share, this multiplicity inflates both the long-term secret-share storage of each participant and the participant-specific public material required to verify its partial responses.

Why storage matters. Post-quantum cryptography must also run on constrained platforms with limited memory, energy, and bandwidth, where use-case-specific memory budgets matter [39]. In the IETF classification of constrained devices [40], Class 1 and Class 2 devices have about 10 and 50 KiB of RAM and 100 and 250 KiB of flash, respectively; Java Card smart cards typically have tens of KiB of persistent memory and a few KiB of RAM [41]; and the PQM4 benchmarking board has 196 KiB of RAM and 1 MiB of flash [42]. Threshold signatures have already been implemented on such platforms, including FROST on Java Card smart cards [43] and the lattice-based n-out-of-n scheme DS2 [17] on a microcontroller with 192 KiB of RAM [44]. Post-quantum protocols on banking smart cards must likewise fit larger objects into limited memory [45].

In a threshold signature with native non-interactive identifiable abort, each participant keeps its secret-share material for the lifetime of the key, while every verifier of partial responses keeps the authenticated participant-specific public material. Thus, a per-participant share of hundreds of KiB or more, as in the everywhere-short baselines of Table 1, already exceeds the flash capacity of a Class 2 device and the persistent memory of a typical Java Card smart card. To the best of our knowledge, prior lattice-based threshold-signature evaluations have not treated these sizes as primary metrics; we report both per participant.

1.1. Our Contributions

LaTSEC. We construct LaTSEC, a three-round lattice-based threshold signature with native non-interactive identifiable abort (Section 3). Its sharing layer is SEC, the ramp everywhere-short secret sharing scheme of Arzaki et al. [46] built from lossless expanders, which gives each participant a single short share. Each participant therefore holds one short signing share and is associated with one public abort key, against which its partial response is verified directly. To the best of our knowledge, LaTSEC is the *first* lattice-based threshold signature with native non-interactive identifiable abort in which this holds, with the number of signing shares and public abort keys per participant independent of n, the thresholds, and κ. The ramp access structure of SEC, with privacy threshold k_p and empirical reconstruction threshold k_c, induces distinct unforgeability and signing thresholds $t_{\mathsf{uf}} = k_p < t_{\mathsf{sig}} = k_c$: up to t_{uf} corrupted participants cannot forge, while a uniformly random set of at least t_{sig} participants can sign except with a small probability. Table 1 compares LaTSEC with prior lattice-based schemes that handle malicious signers.

Security under standard assumptions. We prove correctness, identifiable abort, and threshold unforgeability against static corruptions in the classical random oracle model under standard MLWE and MSIS assumptions (Sections 4 to 6). The unforgeability reduction in our case is particularly challenging since the underlying SEC relies on *computational* privacy, and the analysis must account for the additional leakage from threshold signing: every signing session reveals noisy linear functions $c\boldsymbol{s}_i + \boldsymbol{r}_i$ of the shares. In particular, we must simulate the complete tuple of public abort keys, which are images of correlated shares under one public matrix $\boldsymbol{A}$, jointly with this leakage. To handle this, we introduce Repeated-$\boldsymbol{A}$ Matrix Hint-MLWE (RA-MHMLWE), in which several MLWE targets share the same $\boldsymbol{A}$ matrix and the hints act jointly on all of their secrets. We prove that RA-MHMLWE admits a reduction from the standard MLWE, so our security rests on standard assumptions only. This new problem may also be useful for other threshold schemes whose public keys are images of correlated secrets under one matrix. Our proof thereby extends the multi-query privacy of SEC [46], under which the target $\boldsymbol{As}$ stays pseudorandom given the corrupted shares and the partial responses of all sessions, to the complete abort-key tuple.

Concrete evaluation. We instantiate LaTSEC for a participant set size of $n \in \{40, 60\}$ and security parameter $\kappa \in \{128, 192, 256\}$, and evaluate its security and practical

TABLE 1. LATTICE-BASED THRESHOLD SIGNATURES WITH MECHANISMS FOR HANDLING MALICIOUS SIGNERS AT $\kappa = 128$-BIT SECURITY, IN CHRONOLOGICAL ORDER OF FIRST APPEARANCE

Scheme	Access	Sharing layer	Short partial responses	Storage at $n = 60$[a]		Reported[b]			Malicious signers
				$\lvert\mathsf{sk}_i\rvert$ (KiB)	$\lvert\mathsf{ak}_i\rvert$ (KiB)	n	$\lvert\mathsf{sig}\rvert$ (kB)	Comm_i (kB)	
GKS [16]	threshold	Shamir	×	n/r	n/r	5	46.6[c]	3000[c]	proof IA[c]
Pelican [12]	threshold	Shamir	×	n/r	n/r	48	12.3	26.8+56t	robust
sDKG, replicated [36]	threshold	replicated	✓	$\geq 2.71\times10^8$	$\geq 5.06\times10^8$	16	10.3	20.0	NI-IA
sDKG, ramp [36]	ramp[d]	rand.-assign. ramp	✓	≥ 963	≥ 1920	1024	11.9	25.0	NI-IA
TRaccoon-IA [33]	threshold	Shamir	×	n/r	n/r	1024	12.7	88.2+6.4t	int. IA
Olingo [34]	threshold	Shamir	×	n/r	n/a[e]	1024	11.3	852	proof IA
Hermine [26]	threshold	Vandermonde	✓	≥ 699	≥ 2040	16[f]	11.3	73.2	NI-IA
LaTSEC (this work)	ramp	SEC	✓	14.3	24.0	60	43.6	61.6	NI-IA

KiB and kB: 1024 and 1000 bytes. Threshold and ramp: one threshold (t), or distinct thresholds, for unforgeability and signing. Short partial responses: partial responses are short by construction, since the sharing layer is everywhere-short. Robust: the protocol directly outputs a valid signature whenever sufficiently many honest signers participate. NI-IA, int. IA, and proof IA: partial responses verified directly against participant-specific public material, an additional protocol after a failed aggregation, and non-interactive zero-knowledge proofs, respectively. $\lvert\mathsf{sk}_i\rvert$ and $\lvert\mathsf{ak}_i\rvert$: total long-term secret-share storage held by participant i and total public verification material associated with that participant. $\lvert\mathsf{sig}\rvert$ and Comm_i: aggregate-signature size and per-party signing communication. n/r and n/a: not reported and not applicable. [a]Our estimates (Table 5 for LaTSEC); prior schemes are extrapolated to $n = 60$ with threshold $t = 7$, matching LaTSEC's tolerance of up to $t_{\mathsf{uf}} = k_p = 6$ corrupted participants. These values lie outside the sources' parameter ranges; lower bounds are rounded down, and for Hermine the smallest per-participant value is given; see Appendix O. [b]From the sources, with n the number of participants covered by the parameter set behind the entries; for LaTSEC, Table 5 converted from KiB. [c]As reported in [34, Table 1], which notes identification without a formal IA proof. [d]Thresholds hold with overwhelming probability over a public seed. [e]Proof-based IA has no counterpart to $\lvert\mathsf{ak}_i\rvert$. [f]Parameters only for $n \leq 16$, $t \leq 8$ [26, Section 7]; benchmarks up to $n = t = 64$.

performance (Sections 7 and 8). At $n = 60$ and $\kappa = 128$, each participant stores only one 14.3-KiB signing share and is associated with one 24.0-KiB public abort key; the signature is 42.6 KiB, and the per-party signing communication is 60.1 KiB. Against the everywhere-short baselines extrapolated to $n = 60$ in Table 1, under the size model described in Appendix O, this reduces per-participant secret-share storage by factors of at least 48.9 (Hermine) and 67.5 (ramp sDKG), and per-participant public abort-key storage by factors of at least 85.0 and 80.0, respectively. The interactive and proof-based approaches keep shares small but pay in communication instead. In LaTSEC, each partial response is checked directly against its abort key, so identifying misbehaving signers requires no further messages, and the per-party communication stays at 60.1 KiB (61.6 kB) even when signing fails. The reported per-party communication with identification is $88.2 + 6.4t$ kB for TRaccoon-IA, 852 kB for Olingo, and 3000 kB for GKS (Table 1), each exceeding 61.6 kB for every threshold t. Our Rust prototype (Section 8) corroborates these estimates: at $n = 60$ and $\kappa = 128$, its encoded signing shares average 14.3 KiB, its signatures average 37.8 KiB, below the worst-case 42.6-KiB estimate because the encoding adapts to the actual width of the aggregate response, and one partial verification takes 1.3 ms.

Trade-offs. At $n = 60$ and $\kappa = 128$, the 42.6-KiB (43.6-kB) signature of LaTSEC is larger than those of the more recent schemes in Table 1 for two reasons. First, we do not currently apply bit dropping, which substantially reduces signature size and signing communication in Raccoon-style schemes [21], [22]. A preliminary size calculation suggests that adapting this technique to LaTSEC could reduce the dominant transmitted vectors, and hence its signature and 60.1-KiB per-party signing communication, by up to a factor of two. Realizing this saving is challenging: the resulting rounding errors and hints would further complicate partial-response verification, aggregation, and the identifiable-abort and unforgeability analyses, so we leave it to future work. Second, the abort-key simulation requires wider Gaussians and hence a larger modulus and dimension in our setting. The signature of LaTSEC is nevertheless smaller than the 46.6-kB signature of GKS and the 144.5-kB signature of the two-round line of [29], [32], as reported in [34, Table 1].

Because SEC is a ramp scheme, the signing threshold exceeds the unforgeability threshold: at $n = 60$, we have $t_{\mathsf{uf}} = 6$ and $t_{\mathsf{sig}} = 46$, whereas the threshold schemes in Table 1 use a single threshold. Here, t_{sig} is the minimum signing-set size the protocol enforces, not a bound on which coalitions can reconstruct the key (Section 8.2). LaTSEC also uses three message-dependent rounds and a trusted dealer. Its concrete instantiations currently go up to $n = 60$, the largest party count evaluated for SEC; larger n should be reachable with more computation [46, Section 6].

1.2. Technical Challenges

At first sight, SEC makes native non-interactive identifiable abort immediate: it gives each party one short share, short shares make partial responses directly verifiable, and short reconstruction vectors turn reconstruction into aggregation. We first recall these three properties and then explain why the move from SEC to LaTSEC is nonetheless non-trivial.

One short share per party. SEC shares a short secret $\boldsymbol{s}$ through a public matrix $\mathbf{G} = \begin{bmatrix}\mathbf{g}\ \hat{\mathbf{G}}\end{bmatrix} \in \mathbb{Z}^{n\times k}$, where $\hat{\mathbf{G}} \in \{-1, 0, 1\}^{n\times(k-1)}$ is derived from a lossless bipartite expander and $\mathbf{g}$ is a short secret-bearing column [46]. Party i receives the i-th row $\boldsymbol{s}_i$ of $\mathbf{G}\begin{bmatrix}\boldsymbol{s}^\top \\ \mathbf{R}\end{bmatrix}$ for a short Gaussian masking matrix $\mathbf{R}$, so each share is a short integer combination of $\boldsymbol{s}$ and a few rows of $\mathbf{R}$. A set

T is *certified* if it reconstructs with an integer vector $\boldsymbol{\lambda}_T$ satisfying $\boldsymbol{\lambda}_T^\top \mathbf{G}_T = \mathbf{e}_1^\top$. Signing uses short vectors of this form, and we write $\mathsf{Rec}_{\mathsf{pp}}(T, (\boldsymbol{x}_i)_{i\in T}) := \sum_{i\in T} \lambda_{T,i}\boldsymbol{x}_i$.

Direct verification. LaTSEC follows the Raccoon-style no-abort signature underlying Threshold Raccoon [22], with SEC in place of masked Shamir sharing. Party i holds the share $\boldsymbol{s}_i$ of the signing key $\boldsymbol{s}$, whose verification key is $\boldsymbol{t} := \boldsymbol{A}\boldsymbol{s}$, and is associated with the public abort key $\boldsymbol{t}_i := \boldsymbol{A}\boldsymbol{s}_i$. In three rounds, each active signer commits to $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{r}_i$ for a fresh short mask $\boldsymbol{r}_i$, opens the commitment, and, given the common challenge c, sends $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$. Since $\boldsymbol{s}_i$ is short, $\boldsymbol{z}_i$ is short, and anyone can check $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{w}_i$ and $\|\boldsymbol{z}_i\|_2 \le B_1$; a failed check identifies the signer.

Reconstruction as aggregation. The combiner outputs $\boldsymbol{z} := \mathsf{Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{z}_i)_{i\in\mathsf{act}})$, with the public vector $\boldsymbol{\lambda}_{\mathsf{act}}$ as aggregation weights. Because $\mathsf{Rec}_{\mathsf{pp}}$ is integer-linear and commutes with left multiplication by $\boldsymbol{A}$ [46, Section 4.3],

$$\begin{aligned}\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} &= \boldsymbol{A}\,\mathsf{Rec}_{\mathsf{pp}}\left(\mathsf{act}, (c\boldsymbol{s}_i + \boldsymbol{r}_i)_{i\in\mathsf{act}}\right) - c\boldsymbol{t} \\ &= c\boldsymbol{A}\,\mathsf{Rec}_{\mathsf{pp}}\left(\mathsf{act}, (\boldsymbol{s}_i)_{i\in\mathsf{act}}\right) \\ &\quad + \mathsf{Rec}_{\mathsf{pp}}\left(\mathsf{act}, (\boldsymbol{A}\boldsymbol{r}_i)_{i\in\mathsf{act}}\right) - c\boldsymbol{t} \\ &= c\boldsymbol{t} + \mathsf{Rec}_{\mathsf{pp}}\left(\mathsf{act}, (\boldsymbol{w}_i)_{i\in\mathsf{act}}\right) - c\boldsymbol{t} = \boldsymbol{w}, \end{aligned} \tag{1}$$

so $(c, \boldsymbol{z})$ satisfies the verification relation of the underlying signature, and $\|\boldsymbol{z}\|_2 \le \|\boldsymbol{\lambda}_{\mathsf{act}}\|_1 B_1$ is short because $\boldsymbol{\lambda}_{\mathsf{act}}$ is.

Three challenges. Prior work on everywhere-short sharing shows that removing the masks is not automatic: the sDKG framework distills the properties a sharing must satisfy for identifiable abort [36], and Borin et al. note that security depends on the specific sharing structure [26]. For SEC, this raises three challenges.

Hiding the shares in unmasked partial responses. Direct verification reveals each $\boldsymbol{z}_i = c\boldsymbol{s}_i + \boldsymbol{r}_i$ without one-time masks, so the Gaussian mask $\boldsymbol{r}_i$ alone must hide $c\boldsymbol{s}_i$, while the privacy of SEC is computational. We rely on the multi-query privacy of SEC, which keeps $\boldsymbol{t}$ pseudorandom given the corrupted shares and these responses [46], and choose the mask width to meet its conditions (Section 7).

Simulating the abort keys. LaTSEC also publishes the abort keys $\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{s}_i$, and the reduction must simulate all of them together with the signing leakage. Matrix Hint-MLWE cannot model this view directly: it provides one MLWE target with hints on its secret, whereas the abort keys are several MLWE targets under the same $\boldsymbol{A}$, and the hints act jointly on their correlated secrets. We therefore introduce RA-MHMLWE, prove that it reduces from MLWE (Section 2.2), and reduce the abort-key view to it through an algebraic completion of the corrupted keys and a unimodular change of variables that isolates the hidden coordinates (Section 5).

Min-entropy of the aggregate commitment. Programming the challenge requires the aggregate commitment $\boldsymbol{w}$ to have high min-entropy given the adversary's view. In Threshold Raccoon, $\boldsymbol{w}$ is a plain sum of commitments [22]. In LaTSEC, it is the reconstruction $\sum_{i\in\mathsf{act}} \lambda_{\mathsf{act},i}\boldsymbol{w}_i$, whose entropy the adversary could remove if the honest coefficients vanished, or reduce if they were not invertible modulo q. We prove a min-entropy bound specific to such reconstructed commitments: since no set of at most t_{uf} parties is certified, some honest coefficient is nonzero, and admissibility bounds its magnitude below q, so it is invertible modulo the prime q. Thus, $\boldsymbol{w}$ inherits that signer's min-entropy (Corollary 2).

2. Preliminaries

Besides standard background, this section introduces a new problem, RA-MHMLWE, and states its reduction from MLWE (Section 2.2).

2.1. Notation and Lattice Background

We follow the notation of SEC [46, Section 2 and Appendix A]. Let $[n] := \{1, \dots, n\}$, let $\mathcal{R} := \mathbb{Z}[X]/(X^d + 1)$ for a power of two d, and let $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$. Bold italic letters denote objects over $\mathcal{R}$ or $\mathcal{R}_q$, and bold upright letters denote their coefficient embeddings or objects over $\mathbb{Z}$, $\mathbb{Z}_q$, or $\mathbb{R}$. Thus, $\boldsymbol{a} \in \mathcal{R}^\ell$ is identified with $\mathbf{a} \in \mathbb{Z}^{d\ell}$, and $\boldsymbol{A} \in \mathcal{R}_q^{p\times\ell}$ with $\mathbf{A} \in \mathbb{Z}_q^{dp\times d\ell}$. Throughout, $D := d\ell$. For a matrix $\mathbf{M}$ and $T \subseteq [n]$, $\mathbf{M}_T$ denotes the rows indexed by T. $\mathrm{Row}_{\mathbb{Z}}$, $\mathrm{Col}_{\mathbb{Z}}$, and ker denote integer row and column modules and kernels, $\|\mathbf{M}\|_2$ and $\sigma_{\min}(\mathbf{M})$ the largest and smallest singular values, wt_H the Hamming weight, vec column-wise vectorization, and $\otimes$ the Kronecker product. Norms of ring elements are those of their coefficient embeddings, with representatives in $(-q/2, q/2]$ modulo q. H_∞ denotes min-entropy.

For a countable $S \subseteq \mathbb{R}^N$, a center $\mathbf{c}$, and a positive-definite parameter matrix $\boldsymbol{\Sigma}$, $\mathcal{D}_{S,\mathbf{c},\sqrt{\boldsymbol{\Sigma}}}$ is the discrete Gaussian with mass proportional to $\exp(-\frac{1}{2}(\mathbf{x}-\mathbf{c})^\top\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{c}))$; we write $\mathcal{D}_\sigma$ when $S = \mathbb{Z}^N$, $\mathbf{c} = \mathbf{0}$, and $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. The smoothing parameter η_ϵ is normalized in the same standard-deviation convention, and hence is smaller by a factor $\sqrt{2\pi}$ than in the $\exp(-\pi\|\mathbf{x}\|^2/s^2)$ convention [46, Eq. (2)]. $\mathsf{MLWE}_{n,m,q,d,\sigma}$ denotes decisional Module-LWE in the systematic form of [46, Definition 2]: distinguish $(\boldsymbol{A}, \overline{\boldsymbol{A}}\boldsymbol{s})$ from $(\boldsymbol{A}, \boldsymbol{u})$, where $\overline{\boldsymbol{A}} := \left[\boldsymbol{I}_n \;\; \boldsymbol{A}\right]$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$, $\boldsymbol{s} \leftarrow \mathcal{D}_\sigma^{n+m}$, and $\boldsymbol{u} \xleftarrow{\$} \mathcal{R}_q^n$. For $h \in \mathbb{Z}^+$, $\overline{\boldsymbol{A}}^{\langle h\rangle} := \boldsymbol{I}_h \otimes \overline{\boldsymbol{A}} \in \mathcal{R}_q^{hn\times h(n+m)}$ repeats $\overline{\boldsymbol{A}}$ on the diagonal. MSIS and the hint-free repeated variant of MLWE are recalled in Appendix A.

2.2. Repeated-$\boldsymbol{A}$ Matrix Hint-MLWE

Matrix Hint-MLWE augments an MLWE sample with noisy linear images $\boldsymbol{M}\boldsymbol{s} + \boldsymbol{y}$ of its secret [12], [46], [47], [48]. The abort-key simulation of Section 5 produces several MLWE targets under the same matrix, together with correlated hints on all of their secrets jointly. We capture this with the following problem.

Definition 1 (Repeated-$\boldsymbol{A}$ Matrix Hint-MLWE)**.** Let $n, m, q, d, h \in \mathbb{Z}^+$, $\nu \in \mathbb{Z}_{\ge 0}$, and $(n_j)_{j\in[\nu]} \in (\mathbb{Z}^+)^\nu$. Let $\chi_{\boldsymbol{s}}$ be a distribution over $\mathcal{R}^{h(n+m)}$ and, for $j \in [\nu]$, let $\chi_{\boldsymbol{y}_j}$ be a distribution over $\mathcal{R}^{n_j}$ and $\mathcal{M}_j$ a distribution over $\mathcal{R}^{n_j\times h(n+m)}$. The problem RA-MHMLWE$[h]$

is to distinguish $\left(\boldsymbol{A}, \overline{\boldsymbol{A}}^{\langle h\rangle}\boldsymbol{s}, (\boldsymbol{M}_j)_{j\in[\nu]}, (\boldsymbol{z}_j)_{j\in[\nu]}\right)$ from $\left(\boldsymbol{A}, \boldsymbol{u}, (\boldsymbol{M}_j)_{j\in[\nu]}, (\boldsymbol{z}_j)_{j\in[\nu]}\right)$, where $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$, $\boldsymbol{u} \xleftarrow{\$} \mathcal{R}_q^{hn}$, $\boldsymbol{s} \leftarrow \chi_{\boldsymbol{s}}$, and, independently for $j \in [\nu]$, $\boldsymbol{M}_j \leftarrow \mathcal{M}_j$, $\boldsymbol{y}_j \leftarrow \chi_{\boldsymbol{y}_j}$, and $\boldsymbol{z}_j := \boldsymbol{M}_j\boldsymbol{s} + \boldsymbol{y}_j$ over $\mathcal{R}$. We write $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RA\text{-}MHMLWE}[h]}(\kappa)$ for the distinguishing advantage of $\mathcal{A}$.

The hint matrices act on the whole stacked secret and may mix coordinates of different blocks. For $h = 1$, the problem is Matrix Hint-MLWE with rectangular hints [46], [48]. Without hints, it is the stacked form of multi-secret MLWE [49], [50], which we call RA-MLWE (Appendix A). Other hinted variants of (M)LWE include Leaky LWE [51] and adaptive Hint-MLWE [52]. RA-MHMLWE is a new problem rather than a new assumption: its hardness follows from standard MLWE. Under coefficient embedding, let $N := dh(n+m)$ and $N_j := dn_j$.

Lemma 1 (Hardness of RA-MHMLWE)**.** *In Definition 1, let* $\mathbf{s} \leftarrow \mathcal{D}_{\mathbb{Z}^N, \mathbf{c}_{\mathbf{s}}, \sqrt{\boldsymbol{\Sigma}_{\mathbf{s}}}}$ *and* $\mathbf{y}_j \leftarrow \mathcal{D}_{\mathbb{Z}^{N_j}, \mathbf{c}_{\mathbf{y}_j}, \sqrt{\boldsymbol{\Sigma}_{\mathbf{y}_j}}}$ *with* $\boldsymbol{\Sigma}_{\mathbf{s}}, \boldsymbol{\Sigma}_{\mathbf{y}_j} \succ 0$*. For every tuple of hint matrices in the support, with coefficient embeddings* $\mathbf{M}_j \in \mathbb{Z}^{N_j\times N}$*, let* $\boldsymbol{\Xi} := \boldsymbol{\Sigma}_{\mathbf{s}}^{-1} + \sum_{j=1}^{\nu} \mathbf{M}_j^\top \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1}\mathbf{M}_j$*. Suppose that* $1/\sigma^2 \geq 2\|\boldsymbol{\Xi}\|_2$ *uniformly over all such tuples, and that* $\sigma \geq \sqrt{2}\eta_\epsilon(\mathbb{Z}^N)$ *for some* $0 < \epsilon \leq 1/\sqrt{2N+4}$*. Then, for every PPT adversary* $\mathcal{A}$*, there exists a PPT adversary* $\mathcal{B}$ *such that* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RA\text{-}MHMLWE}[h]}(\kappa) \leq h \cdot \mathsf{Adv}_{\mathcal{B}}^{\mathsf{MLWE}_{n,m,q,d,\sigma}}(\kappa) + 2\epsilon$*. The same holds in the presence of public auxiliary information jointly distributed with the secret, provided that this information can be sampled exactly in polynomial time, that, conditioned on it, the secret is an elliptic discrete Gaussian over* $\mathbb{Z}^N$ *with publicly computable parameters, and that the precision and smoothing conditions hold uniformly over its outcomes.*

Proof sketch. A block-by-block hybrid shows that RA-MLWE$[h]$ is hard under MLWE, with loss h (Lemma 8). The posterior-resampling argument for Matrix Hint-MLWE [12], [47], [48] then applies jointly to the stacked secret: it creates the correlated elliptic secret required by the hints from a spherical source secret, and the repeated target map, being linear and public, does not otherwise affect it. The reduction samples the hint noise as in the problem, and the dummy secret and the correction exactly: for a covariance $\boldsymbol{\Sigma}$ with all eigenvalues at least σ^2, the exact sampler of [49, Lemma 2.3] on the lattice $\boldsymbol{\Sigma}^{-1/2}\mathbb{Z}^N$ applies by the smoothing condition, so the bound has no sampling term. Details are in Appendix B; the full proof is in Appendix G. □

2.3. Self-Target MSIS

As in [22], [36], the final forgery reduction targets Self-Target MSIS [53], [54].

Definition 2 (Self-Target MSIS)**.** Let $\mathsf{G} : \mathcal{R}_q^n \times \{0,1\}^* \to \mathcal{C}$ be a random oracle with $\mathcal{C} \subseteq \mathcal{R}_q$. Given $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$, the problem $\mathsf{SelfTargetMSIS}_{n,m,q,d,\mathcal{C},B_{\mathsf{STMSIS}}}$ asks for $\mathsf{msg} \in \{0,1\}^*$ and $\boldsymbol{z} = \begin{bmatrix}\boldsymbol{z}'\\ c\end{bmatrix} \in \mathcal{R}_q^{n+m}$ with $c \in \mathcal{C}$, $0 < \|\boldsymbol{z}\|_2 \leq B_{\mathsf{STMSIS}}$, and $\mathsf{G}(\begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{A}\end{bmatrix}\boldsymbol{z}, \mathsf{msg}) = c$. We write $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{STMSIS}}(\kappa)$ for the success probability of $\mathcal{A}$.

Lemma 2 (Hardness of SelfTargetMSIS [22, Lemma B.1])**.** *For every adversary* $\mathcal{A}$ *against* $\mathsf{SelfTargetMSIS}_{n,m,q,d,\mathcal{C},B_{\mathsf{STMSIS}}}$ *making at most* Q_H *random-oracle queries, there exists* $\mathcal{B}$ *against* $\mathsf{MSIS}_{n,m,q,d,2B_{\mathsf{STMSIS}}}$ *with* $\mathsf{Time}(\mathcal{B}) \approx 2\,\mathsf{Time}(\mathcal{A})$ *and* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{STMSIS}}(\kappa) \leq \sqrt{Q_H\,\mathsf{Adv}_{\mathcal{B}}^{\mathsf{MSIS}}(\kappa)} + Q_H/|\mathcal{C}|$.

The proof of [22, Lemma B.1] does not depend on the message length, so the lemma holds for $\mathsf{msg} \in \{0,1\}^*$, as in [53].

2.4. The SEC Sharing Scheme

We recall the properties of SEC [46, Definition 11] that LaTSEC uses. $\mathsf{SEC.Setup}(1^\kappa)$ samples a lossless bipartite expander with left degree $c_{\exp}$, sets $\hat{\mathbf{G}} \in \{-1,0,1\}^{n\times(k-1)}$ to its signed bi-adjacency matrix, so that every row of $\hat{\mathbf{G}}$ has $c_{\exp}$ nonzero entries, and appends a secret-bearing column $\mathbf{g} \in \mathbb{Z}^n$ with $\|\mathbf{g}\|_\infty \leq \beta_{\mathbf{g}}$ to form $\mathbf{G} = \begin{bmatrix}\mathbf{g} & \hat{\mathbf{G}}\end{bmatrix}$. It outputs pp, including $\mathbf{G}$, a privacy threshold k_p, and an empirical reconstruction threshold $k_c > k_p$ above which random sets reconstruct except with a small probability ϵ_c. $\mathsf{SEC.Deal}_{\mathsf{pp}}(\mathbf{s})$, for $\mathbf{s} \in \mathbb{Z}^D$, samples $\mathbf{R} \leftarrow \mathcal{D}_{\sigma_{\mathbf{R}}}^{(k-1)\times D}$ and outputs the rows $\mathbf{s}_i$ of

$$\mathbf{S} := \mathbf{G}\begin{bmatrix}\mathbf{s}^\top\\ \mathbf{R}\end{bmatrix} = \mathbf{g}\mathbf{s}^\top + \hat{\mathbf{G}}\mathbf{R} \in \mathbb{Z}^{n\times D}. \tag{2}$$

$\mathsf{SEC.Rec}_{\mathsf{pp}}(T, (\mathbf{x}_i)_{i\in T})$ outputs $\sum_{i\in T}\lambda_{T,i}\mathbf{x}_i$ for a reconstruction vector $\boldsymbol{\lambda}_T \in \mathbb{Z}^{|T|}$ with $\boldsymbol{\lambda}_T^\top\mathbf{G}_T = \mathbf{e}_1^\top$. We use the following properties.

1) *One short share.* Each party holds one share $\mathbf{s}_i \in \mathbb{Z}^D$, and if $\|\mathbf{s}\|_\infty \leq \beta_{\boldsymbol{s}}$ and all entries of $\mathbf{R}$ are bounded by $\beta_{\mathbf{R}}$, then $\|\mathbf{s}_i\|_\infty \leq \beta_{\mathbf{g}}\beta_{\boldsymbol{s}} + c_{\exp}\beta_{\mathbf{R}}$ [46, Lemma 12].
2) *Exact, deterministic reconstruction.* A set T is *certified* if $\mathbf{e}_1^\top \in \mathrm{Row}_{\mathbb{Z}}(\mathbf{G}_T)$, that is, if such a vector exists. For certified T, a deterministic algorithm [46, Algorithm 1] computes a short $\boldsymbol{\lambda}_T$ from $\mathbf{G}_T$ alone; otherwise it returns $\perp$. Reconstruction is exact for every such vector, since $\boldsymbol{\lambda}_T^\top\mathbf{S}_T = \boldsymbol{\lambda}_T^\top\mathbf{G}_T\begin{bmatrix}\mathbf{s} & \mathbf{R}^\top\end{bmatrix}^\top = \mathbf{s}^\top$ [46, Lemma 5].
3) *Public coefficient checks.* Given T and a candidate $\boldsymbol{\lambda}_T$, the identity $\boldsymbol{\lambda}_T^\top\mathbf{G}_T = \mathbf{e}_1^\top$ and the bounds $\|\boldsymbol{\lambda}_T\|_\infty \leq \beta_{\boldsymbol{\lambda}}$ and $wt_H(\boldsymbol{\lambda}_T) \leq \omega_{\boldsymbol{\lambda}}$ are exact integer checks on public data; for the selected instances, the output of [46, Algorithm 1] meets both bounds on all but a small fraction of certified sets of size at least k_c ([46, Section 6.3] and Section 7.1).
4) *Linearity and public-map compatibility.* $\mathsf{SEC.Rec}_{\mathsf{pp}}$ is $\mathbb{Z}$-linear and commutes with left multiplication by any public matrix modulo q [46, Lemmas 17

and 18]. In particular, $\mathsf{SEC.Rec}_{\mathsf{pp}}(T, (\boldsymbol{A}\boldsymbol{x}_i)_{i\in T}) = \boldsymbol{A}\,\mathsf{SEC.Rec}_{\mathsf{pp}}(T, (\boldsymbol{x}_i)_{i\in T})$.

5) *Non-cancellation and unimodular minors.* For every $U \subseteq [n]$ with $1 \le |U| \le k_p$, the expander yields a unimodular $|U| \times |U|$ submatrix $\mathbf{M}_U$ of $\hat{\mathbf{G}}_U$, computed deterministically by peeling [46, Lemma 4 and Algorithm 3]. Hence $\mathrm{Col}_{\mathbb{Z}}(\hat{\mathbf{G}}_U) = \mathbb{Z}^{|U|}$, so $(\mathbf{g})_U \in \mathrm{Col}_{\mathbb{Z}}(\hat{\mathbf{G}}_U)$ and no such U is certified (Lemma 10).
6) *Multi-query privacy.* For $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{I}_p & \boldsymbol{A}'\end{bmatrix}$, the target $\boldsymbol{A}\boldsymbol{s}$ is pseudorandom given $\boldsymbol{A}$, the shares of any U with $|U| \le k_p$, and noisy linear leakage $c_s\boldsymbol{s}_i + \boldsymbol{r}_{i,s}$ from all shares over Q queries, under $\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{MQ}}}$ for the width σ_{MQ} of [46, Eq. (58)], provided its kernel-smoothing and posterior-resampling conditions hold [46, Theorem 4]. Formally, privacy is stated through the experiment HC-Leak [46, Definition 9].

2.5. Threshold Signatures with Identifiable Abort

We use the round-based syntax of [36, Definition 8], which refines that of [22] with identifiable abort in the sense of [55], replacing distributed key generation by a trusted dealer and adding a partial-verification algorithm as in [4], [26]. To accommodate ramp sharing, we separate the *unforgeability threshold* t_{uf}, the maximum number of corrupted parties, from the *signing threshold* t_{sig}, the minimum size of a signing set. An efficiently computable predicate $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) \in \{0,1\}$ defines the *admissible* session inputs, each with $|\mathsf{act}| \ge t_{\mathsf{sig}}$; a session is admissible if its input is.

Definition 3 (TSS with IA)**.** Let $1 \le t_{\mathsf{uf}} < t_{\mathsf{sig}} \le n$. An rnd-round $(t_{\mathsf{uf}}, t_{\mathsf{sig}}, n)$-threshold signature scheme with identifiable abort consists of the following algorithms.

1) $\mathsf{Setup}(1^\kappa, n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$ outputs public parameters tspar.
2) $\mathsf{Keygen}(\mathsf{tspar})$ outputs a tuple $(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in[n]})$, i.e., a verification key vk, public abort information $\mathsf{ak} = (\mathsf{ak}_i)_{i\in[n]}$, and secret key shares $(\mathsf{sk}_i)_{i\in[n]}$.
3) For $k \in [\mathsf{rnd}]$, $\mathsf{ShareSign}_k(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \cdot)$ outputs an updated state and a message $(\mathsf{st}_i, \mathsf{pm}_i^k)$, or an abort $(\mathsf{abort}, S_\perp^i)$ with $S_\perp^i \subseteq \mathsf{act}$; its last input is msg for $k = 1$ and $(\mathsf{pm}_j^{k-1})_{j\in\mathsf{act}}$ otherwise.
4) $\mathsf{PartialVerify}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{PM})$ outputs a bit that checks party i's contribution to the transcript $\mathsf{PM} := (\mathsf{pm}_j^k)_{k\in[\mathsf{rnd}], j\in\mathsf{act}}$, or $\perp$ if another party's message needed for the check is missing or malformed.
5) $\mathsf{Combine}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{PM})$ outputs a signature sig or an abort $(\mathsf{abort}, S_\perp)$.
6) $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ outputs a bit.

Signing is synchronous: messages of one round are delivered consistently to all active parties before the next round, for example over an authenticated broadcast channel. *Unforgeability* (Definition 8) requires that no PPT adversary corrupting at most t_{uf} parties, chosen statically, forges a signature on a message for which no honest party contributed a signing message. *Identifiable abort* (Definition 9) requires that, in every admissible session, either all honest parties abort with the same non-empty blame set contained in the corrupted set, or Combine returns such a set, or the output signature verifies. *Correctness* requires that an honest execution of an admissible session outputs a valid signature; it follows from identifiable abort with $\mathsf{corrupt} = \emptyset$, since no non-empty blame set is then contained in $\mathsf{corrupt}$. The games are given in Appendix C.

TABLE 2. PARAMETERS OF LaTSEC

Parameter	Meaning
d, q	ring degree and prime modulus of $\mathcal{R}_q$
p, ℓ	dimensions of $\boldsymbol{A} \in \mathcal{R}_q^{p\times\ell}$, with $\ell > p$; $D := d\ell$
n, k	number of parties, columns of $\mathbf{G} \in \mathbb{Z}^{n\times k}$
$t_{\mathsf{uf}}, t_{\mathsf{sig}}$	unforgeability and signing thresholds, k_p and k_c of SEC
$\mathcal{C}, \omega_c$	challenges $\{c \in \mathcal{R} : \|c\|_\infty = 1,\ \|c\|_1 = \omega_c\}$
$\sigma_{\boldsymbol{s}}, \sigma_{\mathbf{R}}, \sigma_{\boldsymbol{r}}$	widths of the signing key, of the entries of $\mathbf{R} \in \mathbb{Z}^{(k-1)\times D}$, and of the signing masks
$\beta_{\boldsymbol{s}}, \beta_{\mathbf{R}}, \beta_{\boldsymbol{r}}$	tail bounds on $\|\boldsymbol{s}\|_2$ (hence on $\|\boldsymbol{s}\|_\infty$), $\|\mathbf{R}[i', j']\|$, and $\|\boldsymbol{r}_i\|_2$
$\beta_{\mathbf{g}}, c_{\exp}$	bound on $\|\mathbf{g}\|_\infty$; nonzero entries per row of $\hat{\mathbf{G}}$
$\beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}}$	bounds on $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_\infty$ and $wt_H(\boldsymbol{\lambda}_{\mathsf{act}})$ in admissible sessions
B_1, B_2	norm bounds of partial and final verification

3. The LaTSEC Scheme

3.1. Overview and Parameters

LaTSEC distributes the Raccoon-style no-abort signature underlying Threshold Raccoon [21], [22], without compression or rounding (see Appendix H). Its signing key is $\boldsymbol{s} \leftarrow \mathcal{D}_{\sigma_{\boldsymbol{s}}}^{\ell}$, its verification key is $(\boldsymbol{A}, \boldsymbol{t})$ with $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{I}_p & \boldsymbol{A}'\end{bmatrix} \in \mathcal{R}_q^{p\times\ell}$ and $\boldsymbol{t} := \boldsymbol{A}\boldsymbol{s}$, and a signature $(c, \boldsymbol{z})$ is valid if $c = H_c(\cdots, \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t})$ and $\|\boldsymbol{z}\|_2 \le B_2$. The signing key is shared with SEC under the coefficient embedding $\mathcal{R}^\ell \cong \mathbb{Z}^D$, and we set $t_{\mathsf{uf}} := k_p$ and $t_{\mathsf{sig}} := k_c$. Party i holds $\mathsf{sk}_i := \boldsymbol{s}_i$ and is associated with the public abort key $\mathsf{ak}_i := \boldsymbol{t}_i := \boldsymbol{A}\boldsymbol{s}_i$. Besides the payload, the message msg canonically encodes the session identifier sid, the active set $\mathsf{act} \subseteq [n]$, and a reconstruction vector $\boldsymbol{\lambda}_{\mathsf{act}} \in \mathbb{Z}^{|\mathsf{act}|}$, which $\mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, \cdot)$ uses throughout. A session input is *admissible*, that is, $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1$, if msg encodes $(\mathsf{sid}, \mathsf{act}, \boldsymbol{\lambda}_{\mathsf{act}})$, $|\mathsf{act}| \ge t_{\mathsf{sig}}$, $\boldsymbol{\lambda}_{\mathsf{act}}^\top \mathbf{G}_{\mathsf{act}} = \mathbf{e}_1^\top$, $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_\infty \le \beta_{\boldsymbol{\lambda}}$, and $wt_H(\boldsymbol{\lambda}_{\mathsf{act}}) \le \omega_{\boldsymbol{\lambda}}$. These exact integer checks on public data do not depend on how $\boldsymbol{\lambda}_{\mathsf{act}}$ was computed, so implementations need not share a lattice-reduction routine; the session initiator can take $\boldsymbol{\lambda}_{\mathsf{act}}$ from [46, Algorithm 1]. An inadmissible input is rejected before signing without blaming anyone, which affects availability only (Section 7.1). Table 2 summarizes the parameters; their conditions are given in Section 4.1.

The hash functions $H_{\mathsf{cmt}} : \{0,1\}^* \to \{0,1\}^{2\kappa}$ and $H_c : \{0,1\}^* \to \mathcal{C}$ are modeled as random oracles; H_{cmt}

binds each signer to its commitment before the openings are revealed, and H_c derives the challenge. For implementations, H_c can feed the output stream of an extendable-output function on its input directly into a sample-in-ball sampler as in Dilithium [53]. This is a random oracle into $\mathcal{C}$ whenever the extendable-output function is one (Section 8.1).

3.2. Protocol

Figure 1 gives the protocol. The complete stateful specification, with session states and all assertions, is in Appendix I. A trusted dealer runs Keygen, and the active set of an admissible session signs in three rounds. In the first round, each signer samples a fresh mask $\boldsymbol{r}_i$ and broadcasts a hash commitment to $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{r}_i$. In the second round, it reveals $\boldsymbol{w}_i$. In the third round, it checks every opening against its commitment, aborts with the first inconsistent signer in the canonical order of act, and otherwise reconstructs the aggregate commitment $\boldsymbol{w}$, derives the common challenge c, and sends $\boldsymbol{z}_i = c\boldsymbol{s}_i + \boldsymbol{r}_i$. Since all honest parties receive the same transcript, they compute the same blame set. The combiner runs PartialVerify for every active signer, aborts with the set of signers whose responses fail, and otherwise outputs $(c, \boldsymbol{z})$ with $\boldsymbol{z} = \sum_{j \in \mathsf{act}} \lambda_{\mathsf{act},j} \boldsymbol{z}_j$. Partial verification needs only the transcript, vk, and the abort key ak_i; it requires neither interaction nor zero-knowledge proofs. By Items 2 and 4 of the SEC properties in Section 2.4, $\mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{t}_j)_{j \in \mathsf{act}}) = \boldsymbol{t}$, and the aggregation identity in (1) shows that a non-aborting combination of valid partial responses satisfies $\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} = \boldsymbol{w}$.

4. Correctness and Identifiable Abort

4.1. Parameter Conditions

The following conditions suffice for correctness, identifiable abort, and unforgeability. All dimensions and the query bounds Q_H and Q_{Sign} are polynomial in κ.

1) *Dimensions.* $d \geq \kappa$, $\ell > p$, and $q > 2$ is prime.
2) *Challenges.* $|\mathcal{C}| = 2^{\omega_c}\binom{d}{\omega_c} \geq Q_H 2^{\kappa}$; every $c \in \mathcal{C}$ satisfies $\|c\boldsymbol{u}\|_2 \leq \omega_c \|\boldsymbol{u}\|_2$.
3) *Gaussian tails.* The tail factors $\theta_{\boldsymbol{s}}, \theta_{\mathbf{R}}, \theta_{\boldsymbol{r}}$ satisfy [46, Eqs. (64)–(66)] with query bound $Q := Q_{\mathsf{Sign}}$, and $\beta_{\boldsymbol{s}} := \theta_{\boldsymbol{s}}\sigma_{\boldsymbol{s}}\sqrt{D}$, $\beta_{\mathbf{R}} := \theta_{\mathbf{R}}\sigma_{\mathbf{R}}$, $\beta_{\boldsymbol{r}} := \theta_{\boldsymbol{r}}\sigma_{\boldsymbol{r}}\sqrt{D}$. Let $\mathsf{Good}_{\mathsf{tail}}$ be the event that $\|\boldsymbol{s}\|_2 \leq \beta_{\boldsymbol{s}}$, every entry of $\mathbf{R}$ is at most $\beta_{\mathbf{R}}$ in absolute value, and every signing mask $\boldsymbol{r}_{i,s}$ of an honest signer in any of the Q_{Sign} signing queries satisfies $\|\boldsymbol{r}_{i,s}\|_2 \leq \beta_{\boldsymbol{r}}$. Then $\Pr[\neg\mathsf{Good}_{\mathsf{tail}}] \leq 3 \cdot 2^{-\kappa}$ by the tail bound and union bounds of [46, Lemmas 35 and 36], and under $\mathsf{Good}_{\mathsf{tail}}$, since $\|\boldsymbol{s}\|_\infty \leq \|\boldsymbol{s}\|_2 \leq \beta_{\boldsymbol{s}}$, $\|\boldsymbol{s}_i\|_\infty \leq \beta_{\mathsf{sh}} := \beta_{\mathbf{g}}\beta_{\boldsymbol{s}} + c_{\exp}\beta_{\mathbf{R}}$ by Item 1 of the SEC properties in Section 2.4.
4) *Reconstruction.* Every admissible session input satisfies $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_\infty \leq \beta_{\boldsymbol{\lambda}} < q$ and $wt_H(\boldsymbol{\lambda}_{\mathsf{act}}) \leq \omega_{\boldsymbol{\lambda}}$, so $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_1 \leq \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}$.
5) *Verification bounds.* $B_1 \geq \omega_c\sqrt{D}\beta_{\mathsf{sh}} + \beta_{\boldsymbol{r}}$ and $B_2 \geq \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}B_1$.
6) *Commitment entropy.* $\sigma_{\boldsymbol{r}} > 2d \cdot q^{1/\ell + 2/(d(\ell-p))}$ (Lemma 9).
7) *Key privacy.* The width σ_{MQ} of [46, Eq. (58)] satisfies the kernel-smoothing and posterior-resampling conditions of [46, Theorem 4] with $Q := Q_{\mathsf{Sign}}$, and $\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{MQ}}}$ and $\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\boldsymbol{s}}}$ are hard. Hence, $\boldsymbol{t}$ is pseudorandom given the corrupted shares and all signing responses (Item 6 of the SEC properties in Section 2.4).
8) *Abort keys.* The width σ_{AK} of (9) satisfies $\sigma_{\mathrm{AK}} \geq \sqrt{2}\eta_\epsilon(\mathbb{Z}^{kD})$ for a negligible ϵ, and $\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}$ is hard.
9) *Forgery.* $\mathsf{MSIS}_{p,\ell-p+1,q,d,2B_{\mathsf{STMSIS}}}$ is hard for $B_{\mathsf{STMSIS}} \geq \sqrt{B_2^2 + \omega_c}$.

Item 1 and Items 3 to 5 above give correctness and identifiable abort. Unforgeability uses Items 1, 2, 4, 6, 8 and 9. Item 7 is not a hypothesis of the unforgeability theorem, but guarantees that LaTSEC keeps the key privacy of SEC.

4.2. Identifiable Abort and Correctness

Identifiable abort and correctness both follow from the event $\mathsf{Good}_{\mathsf{tail}}$; the proofs are in Appendix J.

Lemma 3 (Honest partial responses)**.** *Under* $\mathsf{Good}_{\mathsf{tail}}$ *and Item 5 of the parameter conditions in Section 4.1, every honest partial response* $\boldsymbol{z}_i = c\boldsymbol{s}_i + \boldsymbol{r}_i$ *satisfies* $\|\boldsymbol{z}_i\|_2 \leq \omega_c\sqrt{D}\beta_{\mathsf{sh}} + \beta_{\boldsymbol{r}} \leq B_1$ *and* $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{w}_i$.

Theorem 1 (Identifiable abort)**.** *Assume Item 1 and Items 3 to 5 of the parameter conditions in Section 4.1. Then, for every adversary* $\mathcal{A}$ *in* $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$, $\mathsf{Adv}^{\mathsf{TS\text{-}IA}}_{\mathcal{A}}(\kappa) \leq 3 \cdot 2^{-\kappa}$.

Proof sketch. Condition on $\mathsf{Good}_{\mathsf{tail}}$; all honest parties parse the same $\boldsymbol{\lambda}_{\mathsf{act}}$ from msg. An honest signer reveals the opening it committed to, so every signer blamed in $\mathsf{ShareSign}_3$ is corrupted, and consistent delivery makes all honest parties blame the same signer. By Lemma 3, every honest signer passes PartialVerify, so a blame set returned by Combine is non-empty and contained in the corrupted set. If Combine does not abort, every $i \in \mathsf{act}$ satisfies $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{w}_i$ and $\|\boldsymbol{z}_i\|_2 \leq B_1$. By Items 2 and 4 of the SEC properties in Section 2.4, $\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} = \sum_i \lambda_{\mathsf{act},i}(\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i) = \boldsymbol{w}$, and $\|\boldsymbol{z}\|_2 \leq \|\boldsymbol{\lambda}_{\mathsf{act}}\|_1 B_1 \leq \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}B_1 \leq B_2$. Since msg determines sid and act, Verify recomputes the same challenge and accepts. The adversary can therefore win only if $\mathsf{Good}_{\mathsf{tail}}$ fails. The full proof is in Appendix J. □

Corollary 1 (Correctness)**.** *Under the assumptions of Theorem 1, an honest execution of an admissible session outputs a signature accepted by* Verify*, except with probability at most* $3 \cdot 2^{-\kappa}$.

Proof sketch. Apply Theorem 1 with corrupt $= \emptyset$: no non-empty blame set is contained in $\emptyset$, so under $\mathsf{Good}_{\mathsf{tail}}$ the execution neither aborts nor outputs an invalid signature. The full proof is in Appendix J. □

$\mathsf{Setup}(1^\kappa, n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$

1: $\mathsf{pp} \leftarrow \mathsf{SEC.Setup}(1^\kappa)$ for n parties and secret dimension D, with $(k_p, k_c) = (t_{\mathsf{uf}}, t_{\mathsf{sig}})$, or return $\perp$ if no such instance is available
2: Fix $\mathcal{C}, \sigma_{\boldsymbol{s}}, \sigma_{\boldsymbol{r}}, \beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}}, B_1, B_2$ as in Section 4.1.
3: **return** $\mathsf{tspar} := (\mathsf{pp}, \mathcal{C}, B_1, B_2, \sigma_{\boldsymbol{s}}, \sigma_{\boldsymbol{r}}, \beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}})$

$\mathsf{Keygen}(\mathsf{tspar})$ (trusted dealer)

1: $\boldsymbol{A}' \xleftarrow{\$} \mathcal{R}_q^{p\times(\ell-p)}$; $\boldsymbol{A} := [\boldsymbol{I}_p \ \boldsymbol{A}']$; $\boldsymbol{s} \leftarrow \mathcal{D}^\ell_{\sigma_{\boldsymbol{s}}}$
2: $(\boldsymbol{s}_i)_{i\in[n]} \leftarrow \mathsf{SEC.Deal}_{\mathsf{pp}}(\boldsymbol{s})$ ▷ (2)
3: $\boldsymbol{t} := \boldsymbol{A}\boldsymbol{s}$; $\boldsymbol{t}_i := \boldsymbol{A}\boldsymbol{s}_i$ for $i \in [n]$
4: **return** $\mathsf{vk} := (\boldsymbol{A}, \boldsymbol{t})$, $\mathsf{ak} := (\boldsymbol{t}_i)_{i\in[n]}$, $(\mathsf{sk}_i := \boldsymbol{s}_i)_{i\in[n]}$

$\mathsf{PartialVerify}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{PM})$

1: Parse PM for $(\boldsymbol{w}_j)_{j\in\mathsf{act}}$ and $\boldsymbol{z}_i$.
2: $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}})$; $c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$
3: **return** 1 iff $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{w}_i$ and $\|\boldsymbol{z}_i\|_2 \le B_1$

$\mathsf{Combine}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{PM})$

1: Compute $\boldsymbol{w}$ and c as in PartialVerify.
2: $S_\perp := \{i \in \mathsf{act} : \mathsf{PartialVerify}(\dots, \mathsf{ak}_i, \dots) = 0\}$
3: **if** $S_\perp \neq \emptyset$ **then return** $(\mathsf{abort}, S_\perp)$
4: $\boldsymbol{z} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{z}_j)_{j\in\mathsf{act}})$; **return** $\mathsf{sig} := (c, \boldsymbol{z})$

$\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

1: Parse $(\mathsf{sid}, \mathsf{act})$ from msg and $(c, \boldsymbol{z}) := \mathsf{sig}$, returning 0 on failure; $\boldsymbol{w} := \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}$.
2: **return** 1 iff $c = H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ and $\|\boldsymbol{z}\|_2 \le B_2$

$\mathsf{ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$

1: **assert** sid is fresh, $i \in \mathsf{act}$, and $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1$
2: $\boldsymbol{r}_i \leftarrow \mathcal{D}^\ell_{\sigma_{\boldsymbol{r}}}$; $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{r}_i$
3: $\mathsf{cmt}_i := H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, i, \boldsymbol{w}_i)$
4: Store $(\mathsf{act}, \mathsf{msg}, \boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i)$ in $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$.
5: **return** $\mathsf{pm}_i^1 := \mathsf{cmt}_i$

$\mathsf{ShareSign}_2(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{cmt}_j)_{j\in\mathsf{act}})$

1: **assert** the session is in round 1 and its own entry equals cmt_i
2: Store $(\mathsf{cmt}_j)_{j\in\mathsf{act}}$.
3: **return** $\mathsf{pm}_i^2 := \boldsymbol{w}_i$

$\mathsf{ShareSign}_3(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}})$

1: **assert** the session is in round 2 and its own entry equals $\boldsymbol{w}_i$
2: **for** $j \in \mathsf{act}$ in canonical order **do**
3: **if** $\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, j, \boldsymbol{w}_j)$ **then**
4: **return** $(\mathsf{abort}, \{j\})$ ▷ identify signer j
5: **end if**
6: **end for**
7: $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}}) = \sum_{j\in\mathsf{act}} \lambda_{\mathsf{act},j}\boldsymbol{w}_j$
8: $c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$
9: **return** $\mathsf{pm}_i^3 := \boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$

$\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$

1: Parse $(\mathsf{sid}', \mathsf{act}', \boldsymbol{\lambda}_{\mathsf{act}})$ from msg; **if** this fails or $(\mathsf{sid}', \mathsf{act}') \neq (\mathsf{sid}, \mathsf{act})$ **then return** 0.
2: **if** $|\mathsf{act}| < t_{\mathsf{sig}}$ **then return** 0
3: **return** 1 iff $\boldsymbol{\lambda}_{\mathsf{act}}^\top \mathbf{G}_{\mathsf{act}} = \mathbf{e}_1^\top$, $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_\infty \le \beta_{\boldsymbol{\lambda}}$, and $wt_H(\boldsymbol{\lambda}_{\mathsf{act}}) \le \omega_{\boldsymbol{\lambda}}$

Figure 1. The LaTSEC protocol. PM denotes the transcript $(\mathsf{pm}_j^k)_{k\in[3],j\in\mathsf{act}} = ((\mathsf{cmt}_j, \boldsymbol{w}_j, \boldsymbol{z}_j))_{j\in\mathsf{act}}$, and $\mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, \cdot)$ uses the vector $\boldsymbol{\lambda}_{\mathsf{act}}$ encoded in msg. Each st_i holds tspar, vk, sk_i, and a session table, and vk implicitly includes tspar. A failed assertion is a malformed invocation and returns $\perp$ without blame; only the checks in $\mathsf{ShareSign}_3$ and Combine produce identifiable aborts, and a missing or unparsable peer message fails the check that uses it, so its sender is blamed.

For correctness alone, the honest aggregate $\boldsymbol{z} = c\boldsymbol{s} + \sum_i \lambda_{\mathsf{act},i}\boldsymbol{r}_i$ satisfies $\|\boldsymbol{z}\|_2 \le \omega_c\beta_{\boldsymbol{s}} + \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}\beta_{\boldsymbol{r}}$ under $\mathsf{Good}_{\mathsf{tail}}$, so $B_2 \ge \omega_c\beta_{\boldsymbol{s}} + \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}\beta_{\boldsymbol{r}}$ would suffice. We use $B_2 \ge \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}B_1$, which also covers corrupted responses that pass partial verification.

5. Simulating the Public Abort Keys

The unforgeability proof must eventually replace the verification target $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$ by a uniform value, so that a forgery can be turned into a SelfTargetMSIS solution. In LaTSEC, however, the adversary also sees the abort keys $\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{s}_i$ of all parties, which must remain consistent with the corrupted shares and must reconstruct to the replaced target. This section shows how to complete such a tuple from a uniform target (Section 5.1), formalizes the resulting real-or-simulated experiment (Section 5.2), and proves the two views indistinguishable under MLWE (Sections 5.3 and 5.4). Throughout, $\mathbf{A} \in \mathbb{Z}_q^{dp\times D}$ is the coefficient embedding of $\boldsymbol{A}$, and $\mathbf{X} := \begin{bmatrix}\mathbf{s}^\top \\ \mathbf{R}\end{bmatrix} \in \mathbb{Z}^{k\times D}$ is the dealing matrix, so that $\mathbf{S} = \mathbf{G}\mathbf{X}$.

5.1. Abort-Key Completion and Simulation

Let $\mathbf{t} := \mathbf{A}\mathbf{s}$, $\mathbf{t}_i := \mathbf{A}\mathbf{s}_i$, and let $\mathbf{T} \in \mathbb{Z}_q^{n\times dp}$ have rows $\mathbf{t}_i^\top$. Then

$$\mathbf{T} = \mathbf{S}\mathbf{A}^\top = \mathbf{G}\mathbf{Y} = \left(\mathbf{g}\mathbf{t}^\top + \hat{\mathbf{G}}\mathbf{R}_{\mathbf{t}}\right) \bmod q, \qquad (3)$$

where $\mathbf{Y} := \mathbf{X}\mathbf{A}^\top = \begin{bmatrix}\mathbf{t}^\top \\ \mathbf{R}_{\mathbf{t}}\end{bmatrix}$ and $\mathbf{R}_{\mathbf{t}} := \mathbf{R}\mathbf{A}^\top \bmod q$. Thus the abort keys form an SEC-form sharing of $\mathbf{t}$ over $\mathbb{Z}_q$, with the *evaluated* matrix $\mathbf{Y}$ in place of the dealing matrix.

Lemma 4 (Algebraic completion of abort keys)**.** *Let q be prime, let $U \subseteq [n]$ with $|U| \le k_p$, let $\mathbf{T}_U \in \mathbb{Z}_q^{|U|\times dp}$, and let $\breve{\mathbf{t}} \in \mathbb{Z}_q^{dp}$. One can efficiently sample $\breve{\mathbf{R}}_{\mathbf{t}}$ such that $\breve{\mathbf{T}} := (\mathbf{g}\breve{\mathbf{t}}^\top + \hat{\mathbf{G}}\breve{\mathbf{R}}_{\mathbf{t}}) \bmod q$ satisfies $\breve{\mathbf{T}}_U = \mathbf{T}_U$, and the rows of $\breve{\mathbf{T}}_T$ reconstruct to $\breve{\mathbf{t}}$ for every certified set T.*

Proof sketch. For $U \neq \emptyset$, Item 5 of Section 2.4 gives a unimodular submatrix $\mathbf{M}_U$ of $\hat{\mathbf{G}}_U$; placing $\mathbf{M}_U^{-1}$ in the selected rows yields an integer right inverse $\mathbf{K}_U$ of $\hat{\mathbf{G}}_U$. Set $\breve{\mathbf{R}}_{\mathbf{t}} := \mathbf{K}_U(\mathbf{T}_U - (\mathbf{g})_U\breve{\mathbf{t}}^\top) + \mathbf{N}$ for any $\mathbf{N}$ with $\hat{\mathbf{G}}_U\mathbf{N} = \mathbf{0} \bmod q$; then $\breve{\mathbf{T}}_U = \mathbf{T}_U$. For certified T, $\boldsymbol{\lambda}_T^\top(\mathbf{g})_T = 1$ and $\boldsymbol{\lambda}_T^\top\hat{\mathbf{G}}_T = \mathbf{0}^\top$, so $\boldsymbol{\lambda}_T^\top\breve{\mathbf{T}}_T = \breve{\mathbf{t}}^\top$. The full proof is in Appendix K. □

Definition 4 (Abort-key simulator)**.** For prime q, $U \subseteq [n]$ with $|U| \le k_p$, $\breve{\mathbf{t}} \in \mathbb{Z}_q^{dp}$, and $\mathbf{T}_U \in \mathbb{Z}_q^{|U|\times dp}$, the simulator $\mathsf{SimAK}_{\mathsf{pp}}(U, \breve{\mathbf{t}}, \mathbf{T}_U)$ samples $\breve{\mathbf{R}}_{\mathbf{t}}$ uniformly from $\{\mathbf{Z} \in \mathbb{Z}_q^{(k-1)\times dp} : \hat{\mathbf{G}}_U\mathbf{Z} = \mathbf{T}_U - (\mathbf{g})_U\breve{\mathbf{t}}^\top \bmod q\}$ and outputs $\breve{\mathbf{T}} := (\mathbf{g}\breve{\mathbf{t}}^\top + \hat{\mathbf{G}}\breve{\mathbf{R}}_{\mathbf{t}}) \bmod q$.

The particular solution of Lemma 4 plus a uniform element of the modular kernel of $\hat{\mathbf{G}}_U$ gives an efficient uniform sampler. Algebraic consistency does not imply indistinguishability: the real matrix $\mathbf{R}_{\mathbf{t}} = \mathbf{R}\mathbf{A}^\top$ is structured,

whereas SimAK samples uniformly from an affine space. The remainder of this section bridges this gap.

5.2. The HC-Leak-AK Experiment

We extend the multi-query privacy experiment of SEC [46, Definition 9] by the verification target and the complete abort-key tuple. A *leakage pattern* $\mathcal{U} = (U_{\mathrm{raw}}, (V_s)_{s\in[Q_{\mathsf{Sign}}]}) \in \mathcal{U}_{k_p,Q_{\mathsf{Sign}}}$ [46, Section 5.5.2] consists of a set $U_{\mathrm{raw}} \subseteq [n]$ with $|U_{\mathrm{raw}}| \le k_p$, whose raw shares are revealed, and sets $V_s \subseteq [n]$ of parties whose responses are revealed in query s.

Definition 5 (HC-Leak-AK experiment)**.** Fix a leakage pattern $\mathcal{U}$ and challenges $\mathbf{c} = (c_s)_{s\in[Q_{\mathsf{Sign}}]} \in \mathcal{C}^{Q_{\mathsf{Sign}}}$, and let $\mathcal{S} := \{(s,i) : s \in [Q_{\mathsf{Sign}}],\ i \in V_s\}$. The experiment $\mathsf{HC\text{-}Leak\text{-}AK}_{\mathcal{B}}^{(\mathsf{pp},\mathcal{U},\mathbf{c})}(\kappa)$ samples $\boldsymbol{A}' \xleftarrow{\$} \mathcal{R}_q^{p\times(\ell-p)}$, $\boldsymbol{s} \leftarrow \mathcal{D}_{\sigma_{\boldsymbol{s}}}^{\ell}$, $(\boldsymbol{s}_i)_{i\in[n]} \leftarrow \mathsf{SEC.Deal}_{\mathsf{pp}}(\boldsymbol{s})$, and, for $(s,i) \in \mathcal{S}$, $\boldsymbol{r}_{i,s} \leftarrow \mathcal{D}_{\sigma_{\boldsymbol{r}}}^{\ell}$ and $\boldsymbol{z}_{i,s} := c_s\boldsymbol{s}_i + \boldsymbol{r}_{i,s}$. It sets the real view $(\boldsymbol{v}_1, \mathsf{ak}_1) := (\boldsymbol{t}, (\boldsymbol{t}_i)_{i\in[n]})$ with $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$ and $\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{s}_i$, and the simulated view $(\boldsymbol{v}_0, \mathsf{ak}_0) := (\breve{\boldsymbol{t}}, (\breve{\boldsymbol{t}}_i)_{i\in[n]})$, where $\breve{\boldsymbol{t}} \xleftarrow{\$} \mathcal{R}_q^{p}$ and $(\breve{\boldsymbol{t}}_i)_{i\in[n]}$ are the rows of $\mathsf{SimAK}_{\mathsf{pp}}(U_{\mathrm{raw}}, \breve{\mathbf{t}}, \mathbf{T}_{U_{\mathrm{raw}}})$. It samples $b \xleftarrow{\$} \{0,1\}$ and gives $(\mathsf{pp}, \boldsymbol{A}, \mathcal{U}, \mathbf{c}, \boldsymbol{v}_b, (\boldsymbol{s}_i)_{i\in U_{\mathrm{raw}}}, (\boldsymbol{z}_{i,s})_{(s,i)\in\mathcal{S}}, \mathsf{ak}_b)$ to $\mathcal{B}$, which outputs b'. The advantage is $\mathsf{Adv}^{(\mathsf{pp},\mathcal{U},\mathbf{c})}_{\mathcal{B},\mathsf{HC\text{-}Leak\text{-}AK}}(\kappa) := 2|\Pr[b'=b] - 1/2|$.

Both branches reveal the same raw shares and responses; by Lemma 4, the simulated tuple agrees with the real abort keys on U_{raw} and reconstructs to $\breve{\boldsymbol{t}}$.

5.3. Hidden Coordinates

Write $U := U_{\mathrm{raw}}$ and $h_U := k - |U|$. To isolate what the raw shares $\mathbf{S}_U = \mathbf{G}_U\mathbf{X}$ reveal, we change variables in $\mathbf{X}$ unimodularly. For $U \neq \emptyset$, let $\mathbf{M}_U$ be the peeled unimodular submatrix of $\hat{\mathbf{G}}_U$ (Item 5 of Section 2.4), let $\mathbf{P}_U$ be the column permutation with $\mathbf{G}_U\mathbf{P}_U = \begin{bmatrix}\mathbf{M}_U & \mathbf{N}_U\end{bmatrix}$, and define

$$\mathbf{Q}_U := \mathbf{P}_U \begin{bmatrix} \mathbf{M}_U^{-1} & -\mathbf{M}_U^{-1}\mathbf{N}_U \\ \mathbf{O} & \mathbf{I}_{h_U} \end{bmatrix} \in \mathsf{GL}_k(\mathbb{Z}), \tag{4}$$

so that $\mathbf{G}_U\mathbf{Q}_U = \begin{bmatrix}\mathbf{I}_{|U|} & \mathbf{O}\end{bmatrix}$; let $\mathbf{Q}_\emptyset := \mathbf{I}_k$. Then $\mathbf{X}' := \mathbf{Q}_U^{-1}\mathbf{X} = \begin{bmatrix}\mathbf{X}'_{\mathsf{pub}} \\ \mathbf{X}'_{\mathsf{hid}}\end{bmatrix}$ satisfies $\mathbf{X}'_{\mathsf{pub}} = \mathbf{S}_U$: the raw shares reveal exactly the public coordinates, and $\mathbf{X}'_{\mathsf{hid}} \in \mathbb{Z}^{h_U\times D}$ is hidden. Consequently $\mathbf{Y} = \mathbf{Q}_U \begin{bmatrix}\mathbf{S}_U\mathbf{A}^\top \\ \mathbf{X}'_{\mathsf{hid}}\mathbf{A}^\top\end{bmatrix}$, and the only hidden part of the abort-key view is the evaluated matrix $\mathbf{X}'_{\mathsf{hid}}\mathbf{A}^\top$. Replacing it by a uniform matrix yields a uniform element of $\{\mathbf{Z} : \mathbf{G}_U\mathbf{Z} = \mathbf{T}_U\}$, which is exactly the distribution of SimAK.

Let $\mathbf{x}'_{\mathsf{hid}} := \mathrm{vec}(\mathbf{X}'^{\top}_{\mathsf{hid}}) \in \mathbb{Z}^{h_U D}$. After subtracting the contribution of $\mathbf{S}_U$, each response becomes a noisy linear image of $\mathbf{x}'_{\mathsf{hid}}$, and the stacked centered responses are $\widetilde{\mathbf{z}}_{\mathcal{S}} = \mathbf{M}_{\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}} + \mathbf{r}_{\mathcal{S}}$ for a public hint matrix $\mathbf{M}_{\mathsf{hid}}$ built from $\mathbf{G}\mathbf{Q}_U$ and the challenges. With $\mathbf{A}^{\langle h_U\rangle} := \mathbf{I}_{h_U} \otimes \mathbf{A}$, the real and simulated hidden views are

$$\left(\mathbf{A}, \mathbf{A}^{\langle h_U\rangle}\mathbf{x}'_{\mathsf{hid}}, \mathbf{M}_{\mathsf{hid}}, \widetilde{\mathbf{z}}_{\mathcal{S}}\right) \text{ and}$$
$$\left(\mathbf{A}, \mathbf{u}_{\mathsf{hid}}, \mathbf{M}_{\mathsf{hid}}, \widetilde{\mathbf{z}}_{\mathcal{S}}\right), \quad \mathbf{u}_{\mathsf{hid}} \xleftarrow{\$} \mathbb{Z}_q^{h_U dp},$$

which is an instance of RA-MHMLWE$[h_U]$ with one stacked hint (see Appendix D).

It remains to describe the secret $\mathbf{x}'_{\mathsf{hid}}$ given $\mathbf{S}_U$. Each column of $\mathbf{X}$ is a discrete Gaussian with parameter matrix $\mathbf{\Sigma}_{\mathbf{X}} := \mathrm{diag}(\sigma_{\boldsymbol{s}}^2, \sigma_{\mathbf{R}}^2\mathbf{I}_{k-1})$. After the unimodular change of variables and conditioning on $\mathbf{X}'_{\mathsf{pub}} = \mathbf{S}_U$, the hidden coordinates are a discrete Gaussian whose center is determined by $\mathbf{S}_U$ and whose parameter matrix $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}$ is a Schur complement.

Lemma 5 (Conditional precision of the hidden coordinates)**.** *Let $\mathbf{Q}_{U,\mathsf{hid}}$ be the last h_U columns of $\mathbf{Q}_U$. Then $\mathbf{\Sigma}^{-1}_{U,\mathsf{hid}|\mathsf{pub}} = \mathbf{Q}^\top_{U,\mathsf{hid}}\mathbf{\Sigma}^{-1}_{\mathbf{X}}\mathbf{Q}_{U,\mathsf{hid}}$, and hence $\|\mathbf{\Sigma}^{-1}_{U,\mathsf{hid}|\mathsf{pub}}\|_2 \le \|\mathbf{\Sigma}^{-1}_{\mathbf{X}}\|_2\|\mathbf{Q}_U\|_2^2$.*

The proof identifies the conditional precision with the hidden principal block of the transformed precision $\mathbf{Q}_U^\top\mathbf{\Sigma}^{-1}_{\mathbf{X}}\mathbf{Q}_U$ (see Appendix K). To make this bound uniform over all coalitions, note that (4) gives

$$\|\mathbf{Q}_U\|_2 \le 1 + \frac{\sqrt{1+\|\mathbf{N}_U\|_2^2}}{\sigma_{\min}(\mathbf{M}_U)}. \tag{5}$$

Let $\delta_{k_p} := \max_{1\le|U|\le k_p}\|\mathbf{G}_U\|_2$ and $\gamma_{k_p} := \min_{1\le|U|\le k_p}\sigma_{\min}(\mathbf{M}_U) > 0$. Since $\|\mathbf{N}_U\|_2 \le \|\mathbf{G}_U\|_2$, the quantity $\beta^*_{\mathbf{Q}} := \max_{|U|\le k_p}\|\mathbf{Q}_U\|_2$ satisfies

$$\beta^*_{\mathbf{Q}} \le 1 + \frac{\sqrt{1+\delta_{k_p}^2}}{\gamma_{k_p}}. \tag{6}$$

Both δ_{k_p} and γ_{k_p} are computed exhaustively for the selected instances (Table 3). Below, $\beta_{\mathbf{Q}}$ denotes any upper bound on $\beta^*_{\mathbf{Q}}$. The concrete instantiations take the right-hand side of (6).

5.4. Abort-Key Indistinguishability

The responses add precision through the hints. Since the masks are independent with parameter $\sigma_{\boldsymbol{r}}^2\mathbf{I}_D$, every challenge satisfies $\|\mathbf{L}_s\|_2 \le \omega_c$ for its multiplication matrix $\mathbf{L}_s$, and the relevant rows of $\mathbf{G}\mathbf{Q}_U$ have spectral norm at most $\|\mathbf{G}\|_2\beta_{\mathbf{Q}}$, a Kronecker-product calculation (see Appendix K) gives

$$\left\|\mathbf{M}^\top_{\mathsf{hid}}\mathbf{\Sigma}^{-1}_{\mathbf{r}_{\mathcal{S}}}\mathbf{M}_{\mathsf{hid}}\right\|_2 \le \frac{\omega_c^2 Q_{\mathsf{Sign}}\|\mathbf{G}\|_2^2\beta_{\mathbf{Q}}^2}{\sigma_{\boldsymbol{r}}^2}. \tag{7}$$

With $\|\mathbf{\Sigma}^{-1}_{\mathbf{X}}\|_2 = \max\{\sigma_{\boldsymbol{s}}^{-2}, \sigma_{\mathbf{R}}^{-2}\}$ and Lemma 5, the combined precision is at most

$$\beta_{\mathbf{Q}}^2\left(\max\left\{\sigma_{\boldsymbol{s}}^{-2}, \sigma_{\mathbf{R}}^{-2}\right\} + \frac{\omega_c^2 Q_{\mathsf{Sign}}\|\mathbf{G}\|_2^2}{\sigma_{\boldsymbol{r}}^2}\right), \tag{8}$$

uniformly over all leakage patterns and challenge sequences. We therefore define $\sigma_{\mathrm{AK}} > 0$ by

$$\frac{1}{\sigma_{\mathrm{AK}}^2} := 2\beta_{\mathbf{Q}}^2 \left(\max\left\{ \sigma_{\boldsymbol{s}}^{-2}, \sigma_{\mathbf{R}}^{-2} \right\} + \frac{\omega_c^2 Q_{\mathsf{Sign}} \|\mathbf{G}\|_2^2}{\sigma_{\boldsymbol{r}}^2} \right), \quad (9)$$

so that, for every pattern,

$$\left\| \boldsymbol{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1} \otimes \mathbf{I}_D + \mathbf{M}_{\mathsf{hid}}^{\top} \boldsymbol{\Sigma}_{\mathbf{r}_{\mathcal{S}}}^{-1} \mathbf{M}_{\mathsf{hid}} \right\|_2 \leq \frac{1}{2\sigma_{\mathrm{AK}}^2}, \quad (10)$$

which is the precision condition of Lemma 1.

Lemma 6 (Indistinguishability of the abort-key view)**.** *Fix a leakage pattern $\mathcal{U}$ with $h_U := k - |U_{\mathrm{raw}}|$ and challenges $\mathbf{c} \in \mathcal{C}^{Q_{\mathsf{Sign}}}$. If $\sigma_{\mathrm{AK}} \geq \sqrt{2}\eta_\epsilon(\mathbb{Z}^{kD})$ for some $0 < \epsilon \leq 1/\sqrt{2kD+4}$, then for every PPT $\mathcal{B}$ there is a PPT $\mathcal{B}'$ with*

$$\mathsf{Adv}^{(\mathsf{pp},\mathcal{U},\mathbf{c})}_{\mathcal{B},\mathsf{HC\text{-}Leak\text{-}AK}}(\kappa) \leq h_U \cdot \mathsf{Adv}^{\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}}_{\mathcal{B}'}(\kappa) + 2\epsilon. \quad (11)$$

Proof sketch. Condition on $\mathbf{S}_U$. By the representation above, the two branches differ only in whether the hidden target is $\mathbf{A}^{\langle h_U \rangle}\mathbf{x}'_{\mathsf{hid}}$ or uniform, and the representation is a bijection, so it adds no loss. The conditional secret is an elliptic discrete Gaussian whose center is a public function of $\mathbf{S}_U$, and (10) and $h_U \leq k$ give the precision and smoothing conditions of Lemma 1 with $\mathbf{S}_U$ as auxiliary information. That lemma bounds the RA-MHMLWE$[h_U]$ advantage by that against $\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}$, with loss h_U and statistical error 2ϵ. Since the conditional parameter matrix and the precision bound do not depend on the value of $\mathbf{S}_U$, the bound holds for the joint distribution. The full proof is in Appendix K. □

The condition $\sigma_{\mathrm{AK}} \geq \sqrt{2}\eta_\epsilon(\mathbb{Z}^{kD})$ covers every coalition at once, because $h_U D \leq kD$. In the concrete instantiations, this condition and the hardness of $\mathsf{MLWE}_{\sigma_{\mathrm{AK}}}$ determine $\sigma_{\boldsymbol{s}} = \sigma_{\mathbf{R}}$ (Section 7.2).

6. Unforgeability

We prove unforgeability through a sequence of hybrids $\mathsf{Hyb}_0, \ldots, \mathsf{Hyb}_5$, following the strategy of [36], and then reduce a forgery in Hyb_5 to SelfTargetMSIS. Two steps are specific to LaTSEC: the challenge is programmed at an aggregate commitment reconstructed through $\mathsf{SEC.Rec}_{\mathsf{pp}}$, which requires a new min-entropy argument, and the abort keys are replaced by the simulation of Section 5. We write Hyb_j for the j-th hybrid and $\Pr[\mathsf{Hyb}_j]$ for the probability that the adversary wins it. The complete experiments are in Appendix L.

6.1. Hybrid Argument

Hyb_0. This is $\mathsf{Game}_{\mathsf{TS\text{-}UF}}$ (Figure 2) for LaTSEC, so $\Pr[\mathsf{Hyb}_0] = \mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathcal{A}}(\kappa)$.

Hyb_1: random honest commitments. For every honest active signer, the challenger samples $\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$ in round 1 and programs $H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, i, \boldsymbol{w}_i) := \mathsf{cmt}_i$ before revealing $\boldsymbol{w}_i$. The views differ only if the adversary queried that point earlier. By Lemma 9, $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{r}_i$ has min-entropy at least $d-1$ except with probability 2^{-d+1} over $\boldsymbol{A}'$, so a union bound gives

$$|\Pr[\mathsf{Hyb}_0] - \Pr[\mathsf{Hyb}_1]| \leq n(Q_H+1)Q_{\mathsf{Sign}} \cdot 2^{-d+1}. \quad (12)$$

Hyb_2: extracted corrupted openings. In round 2, the challenger looks up openings $\widehat{\boldsymbol{w}}_j$ of the corrupted commitments in the H_{cmt}-table, reconstructs $\boldsymbol{w}$ from the honest and extracted openings, computes $c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$, and precomputes the honest responses, which it returns only in round 3. The views differ only if a corrupted signer later opens to a value that was not extracted, which requires a preimage of a fixed 2κ-bit value or a collision:

$$|\Pr[\mathsf{Hyb}_1] - \Pr[\mathsf{Hyb}_2]| \leq (Q_H^2 + t_{\mathsf{uf}} Q_{\mathsf{Sign}} Q_H) \cdot 2^{-2\kappa}. \quad (13)$$

Hyb_3: programmed challenges. The challenger instead samples $c \xleftarrow{\$} \mathcal{C}$ and programs $H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w}) := c$. This changes the view only if the adversary queried that point before round 2, so $\boldsymbol{w}$ must have high min-entropy given the adversary's view. Unlike in a single signer, $\boldsymbol{w} = \sum_{i \in \mathsf{act}} \lambda_{\mathsf{act},i} \boldsymbol{w}_i$ is a reconstruction in which the corrupted openings are adversarial. Since no set of at most $t_{\mathsf{uf}} = k_p$ parties is certified (Item 5 of Section 2.4), some honest i^* has $\lambda_{\mathsf{act},i^*} \neq 0$ (Lemma 11), and since q is prime and $\beta_{\boldsymbol{\lambda}} < q$, multiplication by $\lambda_{\mathsf{act},i^*}$ is a bijection modulo q. This gives the following corollary of Lemma 13, proved in Appendix E.

Corollary 2 (Min-entropy of the aggregate commitment)**.** *Let q be prime, let $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{A}' \end{bmatrix}$ with $\boldsymbol{A}' \xleftarrow{\$} \mathcal{R}_q^{p \times (\ell-p)}$, and let $\sigma_{\boldsymbol{r}} > 2d \cdot q^{1/\ell + 2/(d(\ell-p))}$. Let act be certified with $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_\infty \leq \beta_{\boldsymbol{\lambda}} < q$, let $|\mathsf{corrupt}| \leq t_{\mathsf{uf}}$, let $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{r}_i$ with $\boldsymbol{r}_i \leftarrow \mathcal{D}^{\ell}_{\sigma_{\boldsymbol{r}}}$ for $i \in \mathsf{act} \setminus \mathsf{corrupt}$, and let the adversary's view be independent of these masks and determine the corrupted openings. Then there is $i^* \in \mathsf{act} \setminus \mathsf{corrupt}$ such that, conditioned on that view and on all randomness except $\boldsymbol{r}_{i^*}$, $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_i)_{i \in \mathsf{act}})$ satisfies $\Pr_{\boldsymbol{A}'}[H_\infty(\boldsymbol{w}) \geq d-1] \geq 1 - 2^{-d+1}$.*

A union bound over the random-oracle and signing queries then gives

$$|\Pr[\mathsf{Hyb}_2] - \Pr[\mathsf{Hyb}_3]| \leq (Q_H+1)Q_{\mathsf{Sign}} \cdot 2^{-d+1}. \quad (14)$$

Hyb_4: openings from responses. The challenger computes each honest opening as $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i$. Since $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{r}_i$, nothing changes, and

$$\Pr[\mathsf{Hyb}_4] = \Pr[\mathsf{Hyb}_3]. \quad (15)$$

This expresses the honest openings through the responses and the abort keys, which prepares the replacement of the keys in Hyb_5.

Hyb_5: simulated keys. The challenger samples $\breve{\boldsymbol{t}} \xleftarrow{\$} \mathcal{R}_q^p$, computes $(\breve{\boldsymbol{t}}_i)_{i\in[n]}$ with $\mathsf{SimAK}_{\mathsf{pp}}(\mathsf{corrupt}, \breve{\mathbf{t}}, \mathbf{T}_{\mathsf{corrupt}})$, publishes $(\boldsymbol{A}, \breve{\boldsymbol{t}})$ and $(\breve{\boldsymbol{t}}_i)_{i\in[n]}$ instead of the real keys, and sets $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c\breve{\boldsymbol{t}}_i$; the honest responses $\boldsymbol{z}_i = c\boldsymbol{s}_i + \boldsymbol{r}_i$ are unchanged. Consider the fixed pattern $\mathcal{U}_{\mathsf{UF}} := (\mathsf{corrupt}, (\mathsf{honest})_{s\in[Q_{\mathsf{Sign}}]})$. Given an HC-Leak-AK challenge for $\mathcal{U}_{\mathsf{UF}}$, a reduction $\mathcal{B}_{\mathrm{AK}}$ keeps, in signing query s, the response blocks of $\mathsf{honest} \cap \mathsf{act}_s$, discards the others, and simulates everything else as in Hyb_4; the challenges are sampled inside the experiment and programmed as in Hyb_3. It never uses honest shares or masks: it assigns the leakage slots to sessions in the order in which their corrupted openings are extracted, uses c_s and the leaked responses as the challenge and $\boldsymbol{z}_i$ of slot s, and sets $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c_s\boldsymbol{t}_i$ with the abort keys it receives. If extraction fails, it opens to $\boldsymbol{A}\boldsymbol{r}_i$ for fresh masks $\boldsymbol{r}_i$, and the session then aborts in round 3 or on $\mathsf{Bad}_{\mathsf{open}}$, so no response is needed. The real branch yields Hyb_4 and the simulated branch yields Hyb_5. Since (8) is uniform over $\mathbf{c} \in \mathcal{C}^{Q_{\mathsf{Sign}}}$ and $h_{\mathsf{corrupt}} \le k$, Lemma 6 gives, under $\sigma_{\mathrm{AK}} \ge \sqrt{2}\eta_\epsilon(\mathbb{Z}^{kD})$,

$$|\Pr[\mathsf{Hyb}_4] - \Pr[\mathsf{Hyb}_5]| \le \mathsf{Adv}^{\mathsf{HC\text{-}Leak\text{-}AK}}_{\mathcal{B}_{\mathrm{AK}}}(\kappa) \tag{16}$$

$$\le k \cdot \mathsf{Adv}^{\mathsf{MLWE}_{\sigma_{\mathrm{AK}}}}_{\mathcal{B}'}(\kappa) + 2\epsilon, \tag{17}$$

where $\mathsf{MLWE}_{\sigma_{\mathrm{AK}}} := \mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}$. By Lemma 4, the simulated keys agree with $\boldsymbol{A}\boldsymbol{s}_i$ for $i \in \mathsf{corrupt}$ and reconstruct to $\breve{\boldsymbol{t}}$ from every certified set, while $\breve{\boldsymbol{t}}$ is uniform and independent of the signing key.

6.2. Reduction and Main Unforgeability Theorem

In Hyb_5, $\boldsymbol{A}_{\mathsf{ST}} := \begin{bmatrix}\boldsymbol{A}' & -\breve{\boldsymbol{t}}\end{bmatrix}$ is uniform. A valid forgery $(c^*, \boldsymbol{z}^*)$ satisfies $\begin{bmatrix}\boldsymbol{I}_p & \boldsymbol{A}_{\mathsf{ST}}\end{bmatrix}\begin{bmatrix}\boldsymbol{z}^* \\ c^*\end{bmatrix} = \boldsymbol{A}\boldsymbol{z}^* - c^*\breve{\boldsymbol{t}} = \boldsymbol{w}^*$, and $c^* = H_c(\mathsf{sid}^*, \mathsf{vk}, \mathsf{act}^*, \mathsf{msg}^*, \boldsymbol{w}^*)$.

Lemma 7 (From Hyb_5 to SelfTargetMSIS)**.** *Let $B_{\mathsf{STMSIS}} \ge \sqrt{B_2^2 + \omega_c}$. For every PPT $\mathcal{A}$, there is a PPT $\mathcal{B}_{\mathsf{STMSIS}}$ against $\mathsf{SelfTargetMSIS}_{p,\ell-p+1,q,d,\mathcal{C},B_{\mathsf{STMSIS}}}$ with $\Pr[\mathsf{Hyb}_5] \le \mathsf{Adv}^{\mathsf{STMSIS}}_{\mathcal{B}_{\mathsf{STMSIS}}}(\kappa)$.*

Proof sketch. $\mathcal{B}_{\mathsf{STMSIS}}$ embeds its challenge as $\boldsymbol{A}_{\mathsf{ST}}$, samples $\boldsymbol{s}$ and the shares itself, completes the abort keys with SimAK, and answers every unprogrammed H_c-query $(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ with $\mathsf{G}(\boldsymbol{w}, \mathsf{Encode}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}))$ for a fixed injective encoding Encode. By freshness, the forgery point was not programmed, so $(\mathsf{Encode}(\mathsf{sid}^*, \mathsf{vk}, \mathsf{act}^*, \mathsf{msg}^*), \begin{bmatrix}\boldsymbol{z}^* \\ c^*\end{bmatrix})$ is a solution with $0 < \|(\boldsymbol{z}^*, c^*)\|_2^2 \le B_2^2 + \omega_c$. The full proof is in Appendix L. □

Theorem 2 (Unforgeability)**.** *Let q be prime, $\beta_{\boldsymbol{\lambda}} < q$, and $\sigma_{\boldsymbol{r}} > 2d \cdot q^{1/\ell + 2/(d(\ell-p))}$. Let $\mathcal{A}$ make at most Q_H random-oracle and Q_{Sign} signing queries and corrupt at most $t_{\mathsf{uf}} = k_p$ parties in $\mathsf{Game}_{\mathsf{TS\text{-}UF}}$, where Q_H also counts the random-oracle evaluations made by the game, such as opening checks and the final verification. Then there are PPT adversaries $\mathcal{B}_{\mathrm{AK}}$ for HC-Leak-AK under $\mathcal{U}_{\mathsf{UF}}$ and $\mathcal{B}_{\mathsf{STMSIS}}$ against $\mathsf{SelfTargetMSIS}_{p,\ell-p+1,q,d,\mathcal{C},B_{\mathsf{STMSIS}}}$ with $B_{\mathsf{STMSIS}} \ge \sqrt{B_2^2 + \omega_c}$, both running in time about $\mathsf{Time}(\mathcal{A})$, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathcal{A}}(\kappa) \le{}& (n+1)(Q_H+1)Q_{\mathsf{Sign}} \cdot 2^{-d+1} \\ &+ (Q_H^2 + t_{\mathsf{uf}}Q_{\mathsf{Sign}}Q_H) \cdot 2^{-2\kappa} \\ &+ \mathsf{Adv}^{\mathsf{HC\text{-}Leak\text{-}AK}}_{\mathcal{B}_{\mathrm{AK}}}(\kappa) + \mathsf{Adv}^{\mathsf{STMSIS}}_{\mathcal{B}_{\mathsf{STMSIS}}}(\kappa).\end{aligned} \tag{18}$$

Proof. Combine (12)–(16) with Lemma 7. □

Corollary 3 (Unforgeability under MLWE and MSIS)**.** *In the setting of Theorem 2, suppose that $\sigma_{\mathrm{AK}} \ge \sqrt{2}\eta_\epsilon(\mathbb{Z}^{kD})$ for some $0 < \epsilon \le 1/\sqrt{2kD+4}$. Let $\mathsf{MSIS}_{2B_{\mathsf{STMSIS}}} := \mathsf{MSIS}_{p,\ell-p+1,q,d,2B_{\mathsf{STMSIS}}}$. Then there are PPT adversaries $\mathcal{B}_{\mathsf{MLWE}}$ and $\mathcal{B}_{\mathsf{MSIS}}$, with $\mathsf{Time}(\mathcal{B}_{\mathsf{MLWE}}) \approx \mathsf{Time}(\mathcal{A})$ and $\mathsf{Time}(\mathcal{B}_{\mathsf{MSIS}}) \approx 2\,\mathsf{Time}(\mathcal{A})$, such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathcal{A}}(\kappa) \le{}& (n+1)(Q_H+1)Q_{\mathsf{Sign}} \cdot 2^{-d+1} \\ &+ (Q_H^2 + t_{\mathsf{uf}}Q_{\mathsf{Sign}}Q_H) \cdot 2^{-2\kappa} + 2\epsilon \\ &+ k \cdot \mathsf{Adv}^{\mathsf{MLWE}_{\sigma_{\mathrm{AK}}}}_{\mathcal{B}_{\mathsf{MLWE}}}(\kappa) \\ &+ \sqrt{Q_H \cdot \mathsf{Adv}^{\mathsf{MSIS}_{2B_{\mathsf{STMSIS}}}}_{\mathcal{B}_{\mathsf{MSIS}}}(\kappa)} + \frac{Q_H}{|\mathcal{C}|}.\end{aligned} \tag{19}$$

In particular, LaTSEC *is unforgeable if these* MLWE *and* MSIS *problems are hard and ϵ is negligible.*

Proof. Substitute (17) and Lemma 2 into (18). □

7. Concrete Instantiations

7.1. Methodology and Requirements

We instantiate LaTSEC from the $(n, k) = (40, 20)$ and $(60, 40)$ SEC instances used in [46, Section 6]. Among the selected matrices with $n \ge 25$, the former has the largest ratio $k_p/k_c \approx 0.19$, while the latter is the largest evaluated configuration. They yield $(t_{\mathsf{uf}}, t_{\mathsf{sig}}) = (5, 26)$ and $(6, 46)$, respectively, and fix $c_{\mathrm{exp}} = 3$, $\beta_{\mathbf{g}} = 1$, and the reconstruction bounds $\beta_{\boldsymbol{\lambda}}$ and $\omega_{\boldsymbol{\lambda}}$ (Table 3). These thresholds and their certification probabilities are inherited from SEC [46, Section 6.2.2]; instances with larger k_p or smaller k_c would correspondingly improve t_{uf} or t_{sig}, subject to re-selecting the parameters below. Moreover, every set of at least k_c^{det} parties is certified, with $k_c^{\mathsf{det}} = 35$ for $n = 40$ and 56 for $n = 60$, by the exhaustive verification of [46, Section 6.2.2]. Our own exhaustive computation shows that, on every set of exactly k_c^{det} parties, the output of [46, Algorithm 1] meets both bounds; since such a vector extends by zeros to any larger set, every active set of at least $t_{\mathsf{sig}}^{\mathsf{det}} := k_c^{\mathsf{det}}$ participants has an admissible reconstruction vector.

The reconstruction bounds lie at or above the maxima observed for the output of [46, Algorithm 1] over the certified sets of size at least k_c tested in [46, Section 6.3], with $3 \cdot 10^4$ sets per size. In our expanded tests of 10^5 sets per size, $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_\infty$ reaches $\beta_{\boldsymbol{\lambda}}$ for both instances, and one certified 27-party set for $n = 40$ has weight $26 > \omega_{\boldsymbol{\lambda}} = 25$

TABLE 3. MATRIX AND RECONSTRUCTION QUANTITIES OF THE TWO SEC INSTANCES USED BY LaTSEC

n	k	$t_{\mathsf{uf}}{=}k_p$	$t_{\mathsf{sig}}{=}k_c$	$t_{\mathsf{sig}}^{\mathsf{det}}$	$\beta_{\boldsymbol{\lambda}}$	$\omega_{\boldsymbol{\lambda}}$	$\lVert\mathbf{G}\rVert_2$	δ_{k_p}	γ_{k_p}	$\beta_{\mathbf{Q}}$
40	20	5	26	35	5	25	5.861	3.646	0.193	20.635
60	40	6	46	56	9	45	7.842	3.788	0.152	26.759

and is rejected. We retain the bounds of [46], since such rejection affects only availability and the rejected set admits many other valid reconstruction vectors that the initiator may supply (Section 8.2).

For the abort-key analysis, we additionally compute $\lVert\mathbf{G}\rVert_2$, δ_{k_p}, and γ_{k_p} exhaustively over $1 \le |U| \le k_p$, where $\mathbf{M}_U$ is returned by the deterministic peeling procedure of [46, Algorithm 3]. Because row restriction cannot increase the spectral norm, δ_{k_p} is attained at $|U| = k_p$. We set $\beta_{\mathbf{Q}}$ to the right-hand side of (6). Table 3 reports all resulting values; profiles for all 17 instances of [46] appear in Appendix M.

We target κ-bit security for query bounds Q_H and Q_{Sign}, with $d = 512$, a prime $q \equiv 1 \pmod{2d}$, and $\ell = p + \ell'$. A parameter set must satisfy the following requirements.

1) *Challenges.* $\omega_c := \min\{\omega : 2^{\omega}\binom{d}{\omega} \ge Q_H 2^{\kappa}\}$, so $Q_H/|\mathcal{C}| \le 2^{-\kappa}$.
2) *Query terms.* $(n+1)(Q_H+1)Q_{\mathsf{Sign}}2^{-d+1} \le 2^{-\kappa}$ and $(Q_H^2 + t_{\mathsf{uf}}Q_{\mathsf{Sign}}Q_H)2^{-2\kappa} \le 2^{-\kappa}$.
3) *Abort keys.* With $\epsilon_{\mathrm{AK}} := 2^{-(\kappa+2)}$, $\sigma_{\mathrm{AK}} \ge \sqrt{2}\eta_{\epsilon_{\mathrm{AK}}}(\mathbb{Z}^{kD})$, and $\mathsf{MLWE}_{p,\ell',q,d,\sigma_{\mathrm{AK}}}$ has at least $\kappa_{\mathsf{MLWE}} := \lceil \kappa + 1 + \lg k \rceil$ bits of security, so the term $k \cdot \mathsf{Adv}^{\mathsf{MLWE}} + 2\epsilon_{\mathrm{AK}}$ is at most $2^{-\kappa}$.
4) *Key privacy.* The requirements of SEC [46, Section 6.4.3] hold with $Q = Q_{\mathsf{Sign}}$ and its smoothing parameters $\epsilon = \epsilon' = 2^{-(\kappa+3)}$. In particular, the kernel-smoothing condition [46, Eq. (56)] holds with margins of at least 4.66 for $n = 40$ and 6.58 for $n = 60$.
5) *Norm bounds.* $B_1 := \omega_c\sqrt{D}(\beta_{\boldsymbol{s}} + c_{\exp}\beta_{\mathbf{R}}) + \beta_{\boldsymbol{r}}$, $B_2 := \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}B_1$, and $B_{\mathsf{STMSIS}} := \sqrt{B_2^2 + \omega_c}$, and $\mathsf{MSIS}_{p,\ell'+1,q,d,2B_{\mathsf{STMSIS}}}$ has at least κ bits of security.
6) *Commitment entropy.* $\sigma_{\boldsymbol{r}} > 2d \cdot q^{1/\ell + 2/(d\ell')}$.

Each statistical term in Corollary 3 is then at most $2^{-\kappa}$, so together they amount to a small constant multiple of $2^{-\kappa}$.

7.2. Security Estimates and Parameter Sets

We estimate every retained tuple at its exact prime modulus with the full LWE attack set of the lattice estimator [56], including primal, BDD, dual, and hybrid attacks, under the reduction-cost model of [57]. For each tuple, q is the largest prime below $2^{\lceil \lg q \rceil}$ with $q \equiv 1 \pmod{2d}$ and $q > 2B_{\mathsf{MSIS}} + 1$, since the estimator at the pinned commit `27a581b` treats SIS instances with larger bounds as trivial. Upstream commit `2f9a6d6` relaxes this check to $B_{\mathsf{MSIS}} < q$; a modulus one bit smaller then meets every target, saving one bit per coefficient of $\mathcal{R}_q$ but raising B_{MSIS}/q from 0.26–0.45 to 0.52–0.91, so we keep the more conservative moduli. As in prior concrete analyses, we do not charge the square-root loss of the forking reduction [26], [28], [58], [59], [60]. Table 4 reports the selected sets. We choose $(\sigma_{\boldsymbol{s}}, \sigma_{\mathbf{R}}, \sigma_{\boldsymbol{r}})$ to minimize B_1, which determines the signature and communication sizes and the modulus. In every set, the abort-key requirement is then binding, with $\sigma_{\mathrm{AK}} = \sqrt{2}\eta_{\epsilon_{\mathrm{AK}}}(\mathbb{Z}^{kD})$ up to rounding, and the optimum has $\sigma_{\boldsymbol{s}} = \sigma_{\mathbf{R}}$ and assigns almost all of its precision budget to the mask term. Shifting part of this budget to $\sigma_{\boldsymbol{s}}$ and $\sigma_{\mathbf{R}}$ would reduce $|\mathsf{sk}_i|$ further at a small cost in B_1. We leave this storage-oriented selection to future work. The parameter-selection notebook, which checks every requirement at 200-bit precision, and its lattice-estimator outputs are in the artifact (https://github.com/latsec-ts/latsec). The requirements of SEC then hold with slack: σ_{MQ} exceeds $\sqrt{2}\eta_{\epsilon}(\mathbb{Z}^D)$ by factors of 4.4 ($n = 40$) and 8.0 ($n = 60$), and $\sigma_{\mathrm{MQ}} > \sigma_{\mathrm{AK}}$, so $\mathsf{MLWE}_{\sigma_{\mathrm{MQ}}}$ is estimated to be at least as hard as $\mathsf{MLWE}_{\sigma_{\mathrm{AK}}}$.

TABLE 4. SELECTED LaTSEC PARAMETER SETS ($d = 512$), WITH $(t_{\mathsf{uf}}, t_{\mathsf{sig}}) = (5, 26)$ FOR $n = 40$ AND $(6, 46)$ FOR $n = 60$

n	κ	$\lg Q_H$	$\lg Q_{\mathsf{Sign}}$	$\lceil \lg q \rceil$	ℓ	p	ω_c	$\lg \sigma_{\boldsymbol{s}} {=} \lg \sigma_{\mathbf{R}}$	$\lg \sigma_{\boldsymbol{r}}$
40	128	63	62	61	12	6	30	15.464	45.013
40	192	64	64	63	16	8	44	15.982	46.827
40	256	64	64	64	20	10	60	16.115	47.465
60	128	63	62	64	12	6	30	15.983	45.813
60	192	64	64	66	16	8	44	16.499	47.625
60	256	64	64	66	20	10	60	16.632	48.263

$\ell' = \ell - p = p$ in every set. Logarithms are base two and computed from full-precision values.

7.3. Size Estimates

Table 5 reports the public storage, long-term secret-share storage, signature sizes, and per-party signing communication. The uniform block $\boldsymbol{A}'$ is represented by a 256-bit seed, and the verification target, the abort keys, and the commitment openings are encoded without bit dropping, at $\lceil \lg q \rceil$ bits per coefficient: $|\mathsf{ak}_i| = pd\lceil \lg q \rceil$, $|\mathsf{vk}| = 256 + |\mathsf{ak}_i|$, and $|\mathsf{ak}| = n|\mathsf{ak}_i|$. As usual, the proofs model $\boldsymbol{A}'$ as uniform, treating its expansion from the public seed as a random oracle. For the Gaussian components, we report ideal entropy-coded sizes [61], using a variance-matched discrete Gaussian [62, Section 5.1]: since each share coefficient is $\mathbf{g}[i]\mathbf{s}[j]$ plus $c_{\exp}$ signed entries of $\mathbf{R}$, $|\mathsf{sk}_i| \approx D(\lg \sigma_{\boldsymbol{s}} + \frac{1}{2}\lg(1 + c_{\exp}) + 2.047)$ bits, as in [46, Appendix I]. The response bounds give $|\mathsf{sig}_i| \approx b_c + \mathsf{Size}(B_1)$ and $|\mathsf{sig}| \approx b_c + \mathsf{Size}(B_2)$, where $\mathsf{Size}(B) := D(\lg(B/\sqrt{D}) + 2.047)$ and $b_c := \lceil \omega_c + \lg\binom{d}{\omega_c} \rceil$ encodes the challenge, and $\mathsf{Comm}_i \approx 2\kappa + pd\lceil \lg q \rceil + \mathsf{Size}(B_1)$ counts the commitment, opening, and response that each signer broadcasts. The estimates exclude session context, coding, framing, and byte-alignment overheads; their derivation is in Appendix N. Since $B_2 = \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}B_1$ is a worst-case bound, the estimate of $|\mathsf{sig}|$ charges $\lg(\omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}})$ extra bits per coeffi-

TABLE 5. ESTIMATED LaTSEC SIZES IN KIB, WITHOUT BIT DROPPING

n	κ	$\lvert\mathsf{vk}\rvert$	$\lvert\mathsf{ak}_i\rvert$	$\lvert\mathsf{ak}\rvert$	$\lvert\mathsf{sk}_i\rvert$	$\lvert\mathsf{sig}_i\rvert$	$\lvert\mathsf{sig}\rvert$	Comm_i
40	128	22.906	22.875	915.0	13.883	35.525	40.750	58.408
40	192	31.531	31.500	1260.0	19.029	49.181	56.147	80.696
40	256	40.031	40.000	1600.0	23.952	62.274	70.981	102.297
60	128	24.031	24.000	1440.0	14.273	36.125	42.622	60.133
60	192	33.031	33.000	1980.0	19.547	49.979	58.641	82.995
60	256	41.281	41.250	2475.0	24.599	63.271	74.098	104.544

cient, whereas an honest aggregate response grows only with $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_2$; an encoding that adapts to the actual width, as in our prototype, recovers much of this gap (Section 8.3). Bit dropping and a tighter verification bound B_2, which must also cover corrupted responses that pass partial verification, remain open.

8. Implementation and Evaluation

We provide two implementations of LaTSEC: a Rust implementation used for all performance measurements, and a Python implementation that follows Figure 1 line by line and serves as a test-vector generator; both are in the artifact. Both are still research prototypes: they are not fully constant-time, key generation uses a trusted dealer, and all parties run in one process with consistent message delivery.

8.1. Implementation Choices

All hashes are SHAKE256 [63] with domain separation and length-framed inputs; H_{cmt} outputs 2κ bits. For H_c, the SHAKE256 output stream drives an unbiased sample-in-ball shuffle directly, as suggested in Section 3.1. The block $\boldsymbol{A}'$ is expanded from a 256-bit seed by rejection sampling, directly in the NTT domain.

The discrete Gaussians are sampled with the algorithm of Canonne et al. [64] in exact integer arithmetic, with bounded statistical error: proposals of magnitude 2^{126} or more are rejected, which puts the output within statistical distance $\exp(-2^{48})$ of $\mathcal{D}_\sigma$ for every width used, and the remaining machine-integer counters overflow with probability below $\exp(-2^{128})$.

The session message msg encodes $(\mathsf{sid}, \mathsf{act}, \boldsymbol{\lambda}_{\mathsf{act}})$ and the payload; the initiator computes $\boldsymbol{\lambda}_{\mathsf{act}}$ with [46, Algorithm 1] using an exact integral LLL, and each signer evaluates $\mathcal{P}$ with exact integer checks.

Residues are encoded at $\lceil \lg q \rceil$ bits, the challenge as the rank of its support followed by its signs (b_c bits, Section 7.3), and all Gaussian components with a Golomb–Rice code [65], [66]. For a component of width σ, we use the largest k with $1.2863 \cdot 2^k \le \sigma$. This is the large-width limit of the length-minimizing parameter. For every component width in our parameter sets, it gives the lowest computed length in a discrete-Gaussian model (each width is at least 4% from a switch point), and for these widths ($\sigma \ge 2^{15}$) the code stays within 0.17 bits per coefficient of the entropy. Since the width of $\boldsymbol{z}$ depends on $\boldsymbol{\lambda}_{\mathsf{act}}$, a signature carries its own parameter.

TABLE 6. MEASURED LaTSEC SIZES IN KIB, WITH THE ESTIMATES OF TABLE 5 IN PARENTHESES

n	κ	$\lvert\mathsf{sk}_i\rvert$	$\lvert\boldsymbol{z}_i\rvert$	$\lvert\mathsf{sig}\rvert$	Comm_i
40	128	13.93 (13.88)	35.36 (35.53)	36.68 (40.75)	58.27 (58.41)
40	192	19.02 (19.03)	48.96 (49.18)	50.71 (56.15)	80.50 (80.70)
40	256	23.95 (23.95)	62.06 (62.27)	64.15 (70.98)	102.13 (102.30)
60	128	14.32 (14.27)	35.96 (36.13)	37.84 (42.62)	59.99 (60.13)
60	192	19.66 (19.55)	49.78 (49.98)	52.24 (58.64)	82.83 (83.00)
60	256	24.71 (24.60)	63.05 (63.27)	66.15 (74.10)	104.37 (104.54)

Means over all parties or 100 sessions. $\lvert\boldsymbol{z}_i\rvert$ is compared with $\lvert\mathsf{sig}_i\rvert$, which also counts the challenge.

8.2. Correctness and Identifiable Abort

For each parameter set, we ran 100 all-honest sessions on random sets of t_{sig} parties. All 600 signed and verified, and no honest partial response was rejected; the largest $\|\boldsymbol{z}_i\|_2/B_1$ was 0.85 and the largest $\|\boldsymbol{z}\|_2/B_2$ was 0.037. Three of the 603 sets drawn were not certified and were refused before the first round; availability rests on [46, Section 6.2.2].

On average, only 10.4 of 26 signers ($n = 40$) and 23.0 of 46 ($n = 60$) had a nonzero coefficient in $\boldsymbol{\lambda}_{\mathsf{act}}$. The others contribute nothing to $\boldsymbol{z}$, but they must stay online for all three rounds and pass PartialVerify, and a deviation by any of them aborts the session with that signer blamed.

Tests with tampered commitments, openings, and partial responses confirm that only deviating signers are blamed.

For one certified 27-party set at $n = 40$, the vector returned by [46, Algorithm 1] has weight $26 > \omega_{\boldsymbol{\lambda}}$ and fails $\mathcal{P}$ (Section 7.1). A search over small combinations of the reduced kernel basis found 11,051 vectors for this set that pass $\mathcal{P}$, the lightest found of weight 12, so the initiator can supply one of them and the set can sign.

8.3. Sizes

Table 6 compares the encoded sizes with Table 5. Shares are within 0.6% of the entropy estimate, and partial responses and per-party communication are slightly smaller than estimated, since honest responses have $\|\boldsymbol{z}_i\|_2 \approx 0.83\, B_1$. Signatures are 9.6–11.2% smaller than in Table 5, which charges the worst-case bound $B_2 = \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}B_1$: the aggregate response grows with $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_2$, on average 3.3 ($n = 40$) and 5.6 ($n = 60$), rather than with $\omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}} = 125$ and 405. This supports the tighter aggregate bound suggested in Section 7.3; verification still uses B_2.

8.4. Running Time

Table 7 in Appendix F gives median single-core running times on an AMD Ryzen AI 9 HX 370, at $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$; the measurement method and further results are in Appendix P. At $n = 60$ and $\kappa = 128$, each signer spends 7.6 ms over the three rounds, Combine takes 33 ms, and Verify takes 0.76 ms. The implementation uses portable 128-bit Montgomery arithmetic without vectorization. The second

round only releases the stored opening. Identifiable abort costs one PartialVerify per signer, 1.0 to 2.8 ms (a standalone call recomputes $\boldsymbol{w}$ and c; Combine computes them once), and Combine, which partially verifies every signer, accounts for most of the combiner's work.

References

[1] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Advances in Cryptology – CRYPTO '89*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. New York, NY: Springer, 1990, pp. 307–315.

[2] R. Gennaro, S. Halevi, H. Krawczyk, and T. Rabin, "Threshold RSA for dynamic and ad-hoc groups," in *Advances in Cryptology – EUROCRYPT 2008*, ser. Lecture Notes in Computer Science, N. P. Smart, Ed., vol. 4965. Berlin, Heidelberg: Springer, 2008, pp. 88–107.

[3] C. Komlo and I. Goldberg, "FROST: Flexible round-optimized Schnorr threshold signatures," in *Selected Areas in Cryptography – SAC 2020*, ser. Lecture Notes in Computer Science, O. Dunkelman, M. J. Jacobson, Jr., and C. O'Flynn, Eds., vol. 12804. Cham: Springer, 2021, pp. 34–65.

[4] T. Ruffing, V. Ronge, E. Jin, J. Schneider-Bensch, and D. Schröder, "ROAST: Robust asynchronous Schnorr threshold signatures," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS 2022)*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. New York, NY: ACM, 2022, pp. 2551–2564.

[5] E. Crites, C. Komlo, and M. Maller, "Fully adaptive Schnorr threshold signatures," in *Advances in Cryptology – CRYPTO 2023, Part I*, ser. Lecture Notes in Computer Science, H. Handschuh and A. Lysyanskaya, Eds., vol. 14081. Cham: Springer, 2023, pp. 678–709.

[6] R. Bacho and J. Loss, "On the adaptive security of the threshold BLS signature scheme," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS 2022)*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. New York, NY: ACM, 2022, pp. 193–207.

[7] R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled, "UC non-interactive, proactive, threshold ECDSA with identifiable aborts," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (CCS 2020)*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. New York, NY: ACM, 2020, pp. 1769–1787.

[8] G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker, "Bandwidth-efficient threshold EC-DSA revisited: Online/offline extensions, identifiable aborts proactive and adaptive security," *Theoretical Computer Science*, vol. 939, pp. 78–104, 2023.

[9] J. Doerner, Y. Kondi, E. Lee, and A. Shelat, "Threshold ECDSA in three rounds," in *2024 IEEE Symposium on Security and Privacy (SP)*. Piscataway, NJ: IEEE, 2024, pp. 3053–3071.

[10] L. T. A. N. Brandão and R. Peralta, "NIST first call for multi-party threshold schemes," National Institute of Standards and Technology, NIST Internal Report 8214C, 2026.

[11] R. Bendlin, S. Krehbiel, and C. Peikert, "How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE," in *Applied Cryptography and Network Security – ACNS 2013*, ser. Lecture Notes in Computer Science, M. J. Jacobson, Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, Eds., vol. 7954. Berlin, Heidelberg: Springer, 2013, pp. 218–236.

[12] T. Espitau, G. Niot, and T. Prest, "Flood and submerse: Distributed key generation and robust threshold signature from lattices," in *Advances in Cryptology – CRYPTO 2024, Part VII*, ser. Lecture Notes in Computer Science, L. Reyzin and D. Stebila, Eds., vol. 14926. Cham: Springer, 2024, pp. 425–458.

[13] G. Tang, B. Pang, L. Chen, and Z. Zhang, "Efficient lattice-based threshold signatures with functional interchangeability," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4173–4187, 2023.

[14] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai, "Threshold cryptosystems from threshold fully homomorphic encryption," in *Advances in Cryptology – CRYPTO 2018, Part I*, ser. Lecture Notes in Computer Science, H. Shacham and A. Boldyreva, Eds., vol. 10991. Cham: Springer, 2018, pp. 565–596.

[15] S. Agrawal, D. Stehlé, and A. Yadav, "Round-optimal lattice-based threshold signatures, revisited," in *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), M. Bojańczyk, E. Merelli, and D. P. Woodruff, Eds., vol. 229. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022, pp. 8:1–8:20.

[16] K. D. Gur, J. Katz, and T. Silde, "Two-round threshold lattice-based signatures from threshold homomorphic encryption," in *Post-Quantum Cryptography – PQCrypto 2024, Part II*, ser. Lecture Notes in Computer Science, M.-J. O. Saarinen and D. Smith-Tone, Eds., vol. 14772. Cham: Springer, 2024, pp. 266–300.

[17] I. Damgård, C. Orlandi, A. Takahashi, and M. Tibouchi, "Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices," in *Public-Key Cryptography – PKC 2021, Part I*, ser. Lecture Notes in Computer Science, J. A. Garay, Ed., vol. 12710. Cham: Springer, 2021, pp. 99–130.

[18] C. Boschini, A. Takahashi, and M. Tibouchi, "MuSig-L: Lattice-based multi-signature with single-round online phase," in *Advances in Cryptology – CRYPTO 2022, Part II*, ser. Lecture Notes in Computer Science, Y. Dodis and T. Shrimpton, Eds., vol. 13508. Cham: Springer, 2022, pp. 276–305.

[19] Y. Chen, "DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation," in *Advances in Cryptology – CRYPTO 2023, Part V*, ser. Lecture Notes in Computer Science, H. Handschuh and A. Lysyanskaya, Eds., vol. 14085. Cham: Springer, 2023, pp. 716–747.

[20] N. A. Alkadri, N. Döttling, and S. Pu, "Practical lattice-based distributed signatures for a small number of signers," in *Applied Cryptography and Network Security – ACNS 2024, Part I*, ser. Lecture Notes in Computer Science, C. Pöpper and L. Batina, Eds., vol. 14583. Cham: Springer, 2024, pp. 376–402.

[21] R. del Pino, S. Katsumata, T. Prest, and M. Rossi, "Raccoon: A masking-friendly signature proven in the probing model," in *Advances in Cryptology – CRYPTO 2024, Part I*, ser. Lecture Notes in Computer Science, L. Reyzin and D. Stebila, Eds., vol. 14920. Cham: Springer, 2024, pp. 409–444.

[22] R. del Pino, S. Katsumata, M. Maller, F. Mouhartem, T. Prest, and M.-J. O. Saarinen, "Threshold Raccoon: Practical threshold signatures from standard lattice assumptions," in *Advances in Cryptology – EUROCRYPT 2024, Part II*, ser. Lecture Notes in Computer Science, M. Joye and G. Leander, Eds., vol. 14652. Cham: Springer, 2024, pp. 219–248.

[23] A. Bienstock, L. de Castro, D. Escudero, A. Polychroniadou, and A. Takahashi, "Quorus: Efficient, scalable threshold ML-DSA signatures from MPC," in *35th USENIX Security Symposium (USENIX Security 26)*. Berkeley, CA: USENIX Association, 2026, pp. 6405–6424, full version: Cryptology ePrint Archive, Paper 2025/1163.

[24] S. Celi, R. del Pino, T. Espitau, G. Niot, and T. Prest, "Efficient threshold ML-DSA," in *35th USENIX Security Symposium (USENIX Security 26)*. Berkeley, CA: USENIX Association, 2026, full version: Cryptology ePrint Archive, Paper 2026/013.

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

[26] G. Borin, S. Celi, R. del Pino, T. Espitau, S. Katsumata, G. Niot, T. Prest, and K. Takemure, "Hermine: An efficient lattice-based FROST-like threshold signature," in *Advances in Cryptology – ASIACRYPT 2026*, 2026, to appear. Full version: Cryptology ePrint Archive, Paper 2026/419.

[27] S. Katsumata, M. Reichle, and K. Takemure, "Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding," in *Advances in Cryptology – CRYPTO 2024, Part VII*, ser. Lecture Notes in Computer Science, L. Reyzin and D. Stebila, Eds., vol. 14926. Cham: Springer, 2024, pp. 459–491.

[28] T. Espitau, S. Katsumata, and K. Takemure, "Two-round threshold signature from algebraic one-more learning with errors," *Journal of Cryptology*, vol. 38, no. 4, p. 31, 2025.

[29] C. Zhu and S. Tessaro, "The algebraic one-more MISIS problem and applications to threshold signatures," in *Advances in Cryptology – CRYPTO 2025, Part I*, ser. Lecture Notes in Computer Science, Y. T. Kalai and S. F. Kamara, Eds., vol. 16000. Cham: Springer, 2025, pp. 548–581.

[30] C. Boschini, D. Kaviani, R. W. F. Lai, G. Malavolta, A. Takahashi, and M. Tibouchi, "Ringtail: Practical two-round threshold signatures from learning with errors," in *2025 IEEE Symposium on Security and Privacy (SP)*. Piscataway, NJ: IEEE, 2025, pp. 149–164.

[31] J. C. Benaloh and J. Leichter, "Generalized secret sharing and monotone functions," in *Advances in Cryptology – CRYPTO '88*, ser. Lecture Notes in Computer Science, S. Goldwasser, Ed., vol. 403. New York, NY: Springer, 1990, pp. 27–35.

[32] R. Chairattana-Apirom, S. Tessaro, and C. Zhu, "Partially non-interactive two-round lattice-based threshold signatures," in *Advances in Cryptology – ASIACRYPT 2024, Part IV*, ser. Lecture Notes in Computer Science, K.-M. Chung and Y. Sasaki, Eds., vol. 15487. Singapore: Springer, 2024, pp. 268–302.

[33] R. del Pino, S. Katsumata, G. Niot, M. Reichle, and K. Takemure, "Unmasking TRaccoon: A lattice-based threshold signature with an efficient identifiable abort protocol," in *Advances in Cryptology – CRYPTO 2025, Part VI*, ser. Lecture Notes in Computer Science, Y. T. Kalai and S. F. Kamara, Eds., vol. 16005. Cham: Springer, 2025, pp. 423–456.

[34] K. D. Gur, P. Hough, J. Katz, C. Sandsbråten, and T. Silde, "Olingo: Threshold lattice signatures with DKG and identifiable abort," Cryptology ePrint Archive, Paper 2025/1789, 2025, to appear at ACM CCS 2026.

[35] M. F. Esgin, T. Espitau, G. Niot, T. Prest, A. Sakzad, and R. Steinfeld, "Plover: Masking-friendly hash-and-sign lattice signatures," in *Advances in Cryptology – EUROCRYPT 2024, Part VII*, ser. Lecture Notes in Computer Science, M. Joye and G. Leander, Eds., vol. 14657. Cham: Springer, 2024, pp. 316–345.

[36] R. del Pino, T. Espitau, G. Niot, and T. Prest, "How to shortly share a short vector: DKG with short shares and application to lattice-based threshold signatures with identifiable aborts," Manuscript, 2025, [Online]. Available: https://gniot.fr/assets/papers/sdkg.pdf.

[37] Y. Desmedt, G. Di Crescenzo, and M. Burmester, "Multiplicative non-abelian sharing schemes and their application to threshold cryptography," in *Advances in Cryptology – ASIACRYPT '94*, ser. Lecture Notes in Computer Science, J. Pieprzyk and R. Safavi-Naini, Eds., vol. 917. Berlin, Heidelberg: Springer, 1995, pp. 19–32.

[38] M. Battagliola, G. Borin, G. Di Crescenzo, A. Meneghetti, and E. Persichetti, "Enhancing threshold group action signature schemes: Adaptive security and scalability improvements," in *Post-Quantum Cryptography – PQCrypto 2025, Part I*, ser. Lecture Notes in Computer Science, R. Niederhagen and M.-J. O. Saarinen, Eds., vol. 15577. Cham: Springer, 2025, pp. 129–161.

[39] N. Puch, M. Pursche, S. N. Peters, and M. P. Heinl, "SoK: The engineer's guide to post-quantum cryptography for embedded devices," in *2026 IEEE 11th European Symposium on Security and Privacy (EuroS&P)*. Piscataway, NJ: IEEE, 2026, pp. 23–42.

[40] C. Bormann, M. Ersue, and A. Keranen, "Terminology for constrained-node networks," RFC 7228, RFC Editor, 2014.

[41] E. van der Laan, E. Poll, J. Rijneveld, J. de Ruiter, P. Schwabe, and J. Verschuren, "Is Java Card ready for hash-based signatures?" in *Advances in Information and Computer Security – IWSEC 2018*, ser. Lecture Notes in Computer Science, A. Inomata and K. Yasuda, Eds., vol. 11049. Cham: Springer, 2018, pp. 127–142.

[42] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4," Cryptology ePrint Archive, Paper 2019/844, 2019.

[43] A. Dufka and P. Švenda, "Enabling efficient threshold signature computation via Java Card API," in *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES 2023)*. New York, NY: ACM, 2023, pp. 2:1–2:10.

[44] S. Ricci, V. Shapoval, P. Dzurenda, P. Roenne, J. Oupicky, and L. Malina, "Lattice-based multisignature optimization for RAM constrained devices," in *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES 2024)*. New York, NY: ACM, 2024.

[45] L. Bettale, M. De Oliveira, and E. Dottax, "Post-quantum protocols for banking applications," in *Smart Card Research and Advanced Applications – CARDIS 2022*, ser. Lecture Notes in Computer Science, I. Buhan and T. Schneider, Eds., vol. 13820. Cham: Springer, 2023, pp. 271–289.

[46] M. Arzaki, M. F. Esgin, A. Sakzad, and R. Steinfeld, "SEC: One-share everywhere-short secret sharing from lossless expanders," Cryptology ePrint Archive, Paper 2026/2841, 2026.

[47] D. Kim, D. Lee, J. Seo, and Y. Song, "Toward practical lattice-based proof of knowledge from Hint-MLWE," in *Advances in Cryptology – CRYPTO 2023, Part V*, ser. Lecture Notes in Computer Science, H. Handschuh and A. Lysyanskaya, Eds., vol. 14085. Cham: Springer, 2023, pp. 549–580.

[48] H. Wang, R. Steinfeld, M.-J. O. Saarinen, M. F. Esgin, and S.-M. Yiu, "mmCipher: Batching post-quantum public key encryption made bandwidth-optimal," in *35th USENIX Security Symposium (USENIX Security 26)*. Berkeley, CA: USENIX Association, 2026, full version: Cryptology ePrint Archive, Paper 2025/1000.

[49] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," in *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013)*, D. Boneh, T. Roughgarden, and J. Feigenbaum, Eds. New York, NY: ACM, 2013, pp. 575–584.

[50] K. Boudgoust, C. Jeudy, A. Roux-Langlois, and W. Wen, "On the hardness of Module-LWE with binary secret," in *Topics in Cryptology – CT-RSA 2021*, ser. Lecture Notes in Computer Science, K. G. Paterson, Ed., vol. 12704. Cham: Springer, 2021, pp. 503–526.

[51] R. Lai, M. Swarnakar, and I. Woo, "Leaky LWE: learning with errors with semi-adaptive secret-and error-leakage," *IACR Communications in Cryptology*, pp. 1–23, 2025.

[52] Y. Hwang, S. Katsumata, S. Min, G. Niot, and Y. Song, "Proving threshold Regev PKE from adaptive Hint-MLWE: Efficient, non-interactive, and CCA secure," Cryptology ePrint Archive, Paper 2026/1585, 2026.

[53] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: A lattice-based digital signature scheme," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 1, pp. 238–268, 2018.

[54] E. Kiltz, V. Lyubashevsky, and C. Schaffner, "A concrete treatment of Fiat–Shamir signatures in the quantum random-oracle model," in *Advances in Cryptology – EUROCRYPT 2018, Part III*, ser. Lecture Notes in Computer Science, J. B. Nielsen and V. Rijmen, Eds., vol. 10822. Cham: Springer, 2018, pp. 552–586.

[55] Y. Ishai, R. Ostrovsky, and V. Zikas, "Secure multi-party computation with identifiable abort," in *Advances in Cryptology – CRYPTO 2014, Part II*, ser. Lecture Notes in Computer Science, J. A. Garay and R. Gennaro, Eds., vol. 8617. Berlin, Heidelberg: Springer, 2014, pp. 369–386.

[56] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of Learning with Errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.

[57] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange – A new hope," in *25th USENIX Security Symposium (USENIX Security 16)*. Berkeley, CA: USENIX Association, 2016, pp. 327–343.

[58] C.-P. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology – CRYPTO '89*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. New York, NY: Springer, 1989, pp. 239–252.

[59] M. Fersch, E. Kiltz, and B. Poettering, "On the provable security of (EC)DSA signatures," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*, E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, Eds. New York, NY: ACM, 2016, pp. 1651–1662.

[60] W. Beullens, T. Kleinjung, and F. Vercauteren, "CSI-FiSh: Efficient isogeny based signatures through class group computations," in *Advances in Cryptology – ASIACRYPT 2019, Part I*, ser. Lecture Notes in Computer Science, S. D. Galbraith and S. Moriai, Eds., vol. 11921. Cham: Springer, 2019, pp. 227–247.

[61] M.-J. O. Saarinen, "Arithmetic coding and blinding countermeasures for lattice signatures: Engineering a side-channel resistant post-quantum signature scheme with compact signatures," *Journal of Cryptographic Engineering*, vol. 8, no. 1, pp. 71–84, 2018.

[62] T. Espitau, M. Tibouchi, A. Wallet, and Y. Yu, "Shorter hash-and-sign lattice-based signatures," in *Advances in Cryptology – CRYPTO 2022, Part II*, ser. Lecture Notes in Computer Science, Y. Dodis and T. Shrimpton, Eds., vol. 13508. Cham: Springer, 2022, pp. 245–275.

[63] NIST, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," Federal Information Processing Standards Publication FIPS 202, August 2015.

[64] C. L. Canonne, G. Kamath, and T. Steinke, "The discrete Gaussian for differential privacy," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[65] S. W. Golomb, "Run-length encodings," *IEEE Transactions on Information Theory*, vol. 12, no. 3, pp. 399–401, 1966.

[66] R. F. Rice, "Some practical universal noiseless coding techniques," Jet Propulsion Laboratory, California Institute of Technology, Tech. Rep. JPL Publication 79-22, 1979.

[67] V. Lyubashevsky, "Lattice signatures without trapdoors," in *Advances in Cryptology – EUROCRYPT 2012*, ser. Lecture Notes in Computer Science, D. Pointcheval and T. Johansson, Eds., vol. 7237. Berlin, Heidelberg: Springer, 2012, pp. 738–755.

[68] V. Lyubashevsky, "Fiat–Shamir with aborts: Applications to lattice and factoring-based signatures," in *Advances in Cryptology – ASIACRYPT 2009*, ser. Lecture Notes in Computer Science, M. Matsui, Ed., vol. 5912. Berlin, Heidelberg: Springer, 2009, pp. 598–616.

[69] S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan, "Robustness of the learning with errors assumption," in *Innovations in Computer Science – ICS 2010*, A. C.-C. Yao, Ed. Beijing: Tsinghua University Press, 2010, pp. 230–240.

Appendix A. Lattice Problems

We use MLWE in the systematic form of [46, Definition 2] (Section 2.1) and the following two problems.

Definition 6 (Module-SIS)**.** Given $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$, the problem $\mathsf{MSIS}_{n,m,q,d,B}$ asks for $\boldsymbol{z} \in \mathcal{R}^{n+m} \setminus \{\mathbf{0}\}$ with $\begin{bmatrix}\boldsymbol{I}_n & \boldsymbol{A}\end{bmatrix}\boldsymbol{z} = \mathbf{0}$ over $\mathcal{R}_q$ and $\|\boldsymbol{z}\|_2 \le B$.

Definition 7 (Repeated-$\boldsymbol{A}$ MLWE)**.** For $h \in \mathbb{Z}^+$ and a distribution ψ over $\mathbb{Z}$, the problem $\mathsf{RA\text{-}MLWE}_{n,m,q,d,\psi}[h]$ is to distinguish $(\boldsymbol{A}, \overline{\boldsymbol{A}}^{\langle h\rangle}\boldsymbol{s})$ from $(\boldsymbol{A}, \boldsymbol{u})$, where $\boldsymbol{A} \xleftarrow{\$} \mathcal{R}_q^{n\times m}$, the coefficients of $\boldsymbol{s} \in \mathcal{R}^{h(n+m)}$ are i.i.d. from ψ, and $\boldsymbol{u} \xleftarrow{\$} \mathcal{R}_q^{hn}$.

Writing $\boldsymbol{s} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_h)$, the target is $(\overline{\boldsymbol{A}}\boldsymbol{s}_1, \ldots, \overline{\boldsymbol{A}}\boldsymbol{s}_h)$ with independent blocks, so RA-MLWE is the stacked presentation of multi-secret MLWE under a common matrix [49], [50], [67], and $\mathsf{RA\text{-}MLWE}[1] = \mathsf{MLWE}$. We use the stacked form because the hints of RA-MHMLWE may couple different blocks.

Lemma 8 (Hardness of RA-MLWE [49], [50])**.** *For every PPT $\mathcal{A}$ against $\mathsf{RA\text{-}MLWE}_{n,m,q,d,\psi}[h]$, there is a PPT $\mathcal{B}$ against $\mathsf{MLWE}_{n,m,q,d,\psi}$ with $\mathsf{Adv}^{\mathsf{RA\text{-}MLWE}[h]}_{\mathcal{A}}(\kappa) \le h \cdot \mathsf{Adv}^{\mathsf{MLWE}}_{\mathcal{B}}(\kappa)$.*

Proof. Replace the h target blocks by independent uniform values one at a time; each step embeds one MLWE challenge and samples the other blocks itself, as in [49, Lemma 4.8] and [50, Lemma 17]. □

Appendix B. Hardness of RA-MHMLWE: Proof Sketch

We sketch the proof of Lemma 1; the full proof is in Appendix G. Let $\mathcal{B}_0$ receive an $\mathsf{RA\text{-}MLWE}_{n,m,q,d,\sigma}[h]$ challenge $(\boldsymbol{A}, \boldsymbol{b})$, where $\boldsymbol{b} = \overline{\boldsymbol{A}}^{\langle h\rangle}\boldsymbol{w}$ for $\mathbf{w} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma}$ or $\boldsymbol{b}$ is uniform. It samples a dummy secret $\mathbf{s}' \leftarrow \mathcal{D}_{\mathbb{Z}^N,\mathbf{c}_\mathbf{s},\sqrt{\boldsymbol{\Sigma}_\mathbf{s}}}$, hint matrices $\boldsymbol{M}_j \leftarrow \mathcal{M}_j$, and perturbations $\mathbf{y}_j$, and sets $\mathbf{z}_j := \mathbf{M}_j\mathbf{s}' + \mathbf{y}_j$. Given the hints, the secret has the posterior distribution $\mathcal{D}_{\mathbb{Z}^N,\mathbf{c}_0,\sqrt{\boldsymbol{\Sigma}_0}}$ with $\boldsymbol{\Sigma}_0 := \boldsymbol{\Xi}^{-1}$ and a center $\mathbf{c}_0$ computable from the hints [47, Lemma 7], [12, Lemma 1], [48, Lemma G.2]. Since $\|\boldsymbol{\Xi}\|_2 \le 1/(2\sigma^2)$, we have $\boldsymbol{\Sigma}_0 - \sigma^2\mathbf{I}_N \succeq \sigma^2\mathbf{I}_N$, so $\mathcal{B}_0$ can sample, exactly as $\mathbf{s}'$, $\mathbf{t} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\mathbf{c}_0,\sqrt{\boldsymbol{\Sigma}_0-\sigma^2\mathbf{I}_N}}$ and give $(\boldsymbol{A}, \boldsymbol{b} + \overline{\boldsymbol{A}}^{\langle h\rangle}\boldsymbol{t}, (\boldsymbol{M}_j, \boldsymbol{z}_j)_j)$ to $\mathcal{A}$. In the real case, $\boldsymbol{b} + \overline{\boldsymbol{A}}^{\langle h\rangle}\boldsymbol{t} = \overline{\boldsymbol{A}}^{\langle h\rangle}(\boldsymbol{w} + \boldsymbol{t})$ by linearity, and the convolution lemma [47, Lemmas 3 and 5], whose hypothesis is $\sigma \ge \sqrt{2}\eta_\epsilon(\mathbb{Z}^N)$ in our normalization [46, Appendix B], puts $\mathbf{w} + \mathbf{t}$ within statistical distance 2ϵ of the posterior; in the uniform case, the target stays uniform. Hence $\mathsf{Adv}^{\mathsf{RA\text{-}MHMLWE}[h]}_{\mathcal{A}} \le \mathsf{Adv}^{\mathsf{RA\text{-}MLWE}[h]}_{\mathcal{B}_0} + 2\epsilon$, and Lemma 8 completes the proof. With auxiliary information, the reduction first samples it and then runs conditionally on the outcome, with the conditional center and parameter matrix. The uniform precision and smoothing conditions keep every sample exact and make the bound hold after averaging.

Appendix C. Security Games

Figure 2 gives the games for a TSS with IA, adapted from [36]. $\mathcal{A}$ statically chooses corrupt and receives the corrupted shares. L_{sid} stores each session's status, round,

active set, message, transcript table PM ($\mathsf{PM}[k,i] = \mathsf{pm}_i^k$), and blame table BL ($\mathsf{BL}[k,i] = S_\perp^i$); L_{sig} records the messages for which honest signing messages were returned; and $H(\mathsf{str}, \cdot)$ is a domain-separated random oracle. OSign_k completes the round-$(k-1)$ messages of the honest parties stored in PM with those supplied by $\mathcal{A}$ for corrupt $\cap$ act, where a missing message is $\perp$; a failed **assert** returns $\perp$.

Definition 8 (Unforgeability)**.** A TSS with IA is *unforgeable* if $\mathsf{Adv}^{\mathsf{TS\text{-}UF}}_{\mathcal{A}}(\kappa) := \Pr[\mathsf{Game}_{\mathsf{TS\text{-}UF}} = 1]$ is negligible for every PPT $\mathcal{A}$, where $\mathsf{Game}_{\mathsf{TS\text{-}UF}}$ is described in Figure 2.

Definition 9 (Identifiable abort)**.** A TSS with IA *identifies aborts* for the admissible session inputs of $\mathcal{P}$ if $\mathsf{Adv}^{\mathsf{TS\text{-}IA}}_{\mathcal{A}}(\kappa) := \Pr[\mathsf{Game}_{\mathsf{TS\text{-}IA}} = 1]$ is negligible for every PPT $\mathcal{A}$, where $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$ is described in Figure 2.

Thus, in every admissible session, either all honest parties abort with the same non-empty blame set contained in corrupt, or Combine returns such a set, or the output signature verifies. Correctness of admissible sessions follows by running $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$ with $\mathsf{corrupt} = \emptyset$ and an honest execution. Inadmissible inputs are rejected before signing; for a random active set of size at least t_{sig} with the vector of [46, Algorithm 1], this happens if the set is not certified, with probability about the coverage error ϵ_c of SEC [46, Definition 11], or, rarely, if the vector exceeds the bounds (Section 7.1). The correctness game is in Appendix I.

Appendix D. The Abort-Key View as an RA-MHMLWE Instance

We detail the representation of Section 5.3. Write $U := U_{\mathrm{raw}}$ and $h_U := k - |U|$. Products involving $\mathbf{A}$ are modulo q.

D.1. Evaluated Matrices and Affine Completions

With $\mathbf{X} = \begin{bmatrix} \mathbf{s}^\top \\ \mathbf{R} \end{bmatrix}$ and $\mathbf{Y} = \mathbf{X}\mathbf{A}^\top = \begin{bmatrix} \mathbf{t}^\top \\ \mathbf{R_t} \end{bmatrix}$, the abort keys are $\mathbf{T} = \mathbf{G}\mathbf{Y} = \mathbf{S}\mathbf{A}^\top$ and $\mathbf{T}_U = \mathbf{G}_U\mathbf{Y} = \mathbf{S}_U\mathbf{A}^\top$. Let $\mathcal{Y}_U(\mathbf{T}_U) := \{\mathbf{Z} \in \mathbb{Z}_q^{k\times dp} : \mathbf{G}_U\mathbf{Z} = \mathbf{T}_U\}$. For $\breve{\mathbf{Y}} = \begin{bmatrix} \breve{\mathbf{t}}^\top \\ \breve{\mathbf{R}}_\mathbf{t} \end{bmatrix}$, the constraint $\mathbf{G}_U\breve{\mathbf{Y}} = \mathbf{T}_U$ is exactly the completion equation $(\mathbf{g})_U\breve{\mathbf{t}}^\top + \hat{\mathbf{G}}_U\breve{\mathbf{R}}_\mathbf{t} = \mathbf{T}_U$ of Definition 4. Since $\breve{\mathbf{t}}$ is uniform and SimAK samples $\breve{\mathbf{R}}_\mathbf{t}$ uniformly from the corresponding affine space, the simulated $\breve{\mathbf{Y}}$ is uniform over $\mathcal{Y}_U(\mathbf{T}_U)$, and both views are deterministic functions of $\mathbf{Y}$ or $\breve{\mathbf{Y}}$: the target is the first row, and the abort keys are $\mathbf{G}\mathbf{Y}$ or $\mathbf{G}\breve{\mathbf{Y}}$.

D.2. Separation of Exposed and Hidden Coordinates

With $\mathbf{Q}_U$ from (4) and $\mathbf{X}' = \mathbf{Q}_U^{-1}\mathbf{X}$, we get $\mathbf{S}_U = \mathbf{G}_U\mathbf{Q}_U\mathbf{X}' = \mathbf{X}'_{\mathsf{pub}}$ and $\mathbf{Y} = \mathbf{Q}_U \begin{bmatrix} \mathbf{S}_U\mathbf{A}^\top \\ \mathbf{X}'_{\mathsf{hid}}\mathbf{A}^\top \end{bmatrix}$. For $\mathbf{U}_{\mathsf{hid}} \xleftarrow{\$} \mathbb{Z}_q^{h_U\times dp}$, the matrix $\breve{\mathbf{Y}} := \mathbf{Q}_U \begin{bmatrix} \mathbf{S}_U\mathbf{A}^\top \\ \mathbf{U}_{\mathsf{hid}} \end{bmatrix}$ satisfies $\mathbf{G}_U\breve{\mathbf{Y}} = \mathbf{S}_U\mathbf{A}^\top$, and since $\mathbf{Q}_U$ is invertible modulo q, the map $\mathbf{U}_{\mathsf{hid}} \mapsto \breve{\mathbf{Y}}$ is a bijection onto $\mathcal{Y}_U(\mathbf{S}_U\mathbf{A}^\top)$. Hence a uniform hidden target induces exactly the distribution of SimAK.

D.3. Centered Response-Leakage Representation

Write $\mathbf{G}\mathbf{Q}_U = \begin{bmatrix}\mathbf{G}'_{\mathsf{pub}} & \mathbf{G}'_{\mathsf{hid}}\end{bmatrix}$, so that $\mathbf{S} = \mathbf{G}'_{\mathsf{pub}}\mathbf{S}_U + \mathbf{G}'_{\mathsf{hid}}\mathbf{X}'_{\mathsf{hid}}$. Let $\mathbf{L}_s \in \mathbb{Z}^{D\times D}$ represent multiplication by c_s, so that $\mathbf{z}_{i,s} = \mathbf{L}_s\mathbf{s}_i + \mathbf{r}_{i,s}$. The centered response $\widetilde{\mathbf{z}}_{i,s} := \mathbf{z}_{i,s} - \mathbf{L}_s(\mathbf{G}'_{\mathsf{pub}}[i,\cdot]\mathbf{S}_U)^\top$ equals $\mathbf{M}_{i,s}\mathbf{x}'_{\mathsf{hid}} + \mathbf{r}_{i,s}$ with $\mathbf{M}_{i,s} := \mathbf{L}_s(\mathbf{G}'_{\mathsf{hid}}[i,\cdot] \otimes \mathbf{I}_D)$ and $\mathbf{x}'_{\mathsf{hid}} = \mathrm{vec}(\mathbf{X}'^{\top}_{\mathsf{hid}})$. Stacking over $\mathcal{S}$ gives $\widetilde{\mathbf{z}}_\mathcal{S} = \mathbf{M}_{\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}} + \mathbf{r}_\mathcal{S}$. The centering depends only on $\mathbf{S}_U$ and public data.

D.4. Identification with RA-MHMLWE

With $\mathbf{A}^{\langle h_U\rangle} = \mathbf{I}_{h_U} \otimes \mathbf{A}$, we have $\mathrm{vec}((\mathbf{X}'_{\mathsf{hid}}\mathbf{A}^\top)^\top) = \mathbf{A}^{\langle h_U\rangle}\mathbf{x}'_{\mathsf{hid}}$, and $\mathbf{u}_{\mathsf{hid}} := \mathrm{vec}(\mathbf{U}_{\mathsf{hid}}^\top)$ is uniform. Conditioned on $\mathbf{X}'_{\mathsf{pub}} = \mathbf{S}_U$, the real and simulated views are therefore $(\mathbf{A}, \mathbf{A}^{\langle h_U\rangle}\mathbf{x}'_{\mathsf{hid}}, \mathbf{M}_{\mathsf{hid}}, \mathbf{M}_{\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}} + \mathbf{r}_\mathcal{S})$ and $(\mathbf{A}, \mathbf{u}_{\mathsf{hid}}, \mathbf{M}_{\mathsf{hid}}, \mathbf{M}_{\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}} + \mathbf{r}_\mathcal{S})$: an RA-MHMLWE$[h_U]$ instance with repeated matrix $\mathbf{A}^{\langle h_U\rangle}$, joint secret $\mathbf{x}'_{\mathsf{hid}}$, one stacked hint $\mathbf{M}_{\mathsf{hid}}$, and perturbation $\mathbf{r}_\mathcal{S}$ of parameter $\sigma_r^2\mathbf{I}$. Since $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{I}_p & \boldsymbol{A}'\end{bmatrix}$, each block is a systematic MLWE target as in Definition 1 with $(n,m) = (p, \ell - p)$. The conditional secret has parameter matrix $\boldsymbol{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}} \otimes \mathbf{I}_D$ (Lemma 5) and a center determined by $\mathbf{S}_U$, which we treat as public auxiliary information.

Appendix E. Min-Entropy of Reconstructed Commitments

Definition 10 (Systematic MLWE samples [22, Definition 3.7])**.** For $\boldsymbol{A}' \in \mathcal{R}_q^{p\times(\ell-p)}$, let $\mathcal{D}^{\mathsf{MLWE}}_{p,\ell-p,q,d,\sigma}(\boldsymbol{A}')$ be the distribution of $\begin{bmatrix}\boldsymbol{I}_p & \boldsymbol{A}'\end{bmatrix}\boldsymbol{r} \bmod q$ for $\boldsymbol{r} \leftarrow \mathcal{D}_\sigma^\ell$.

Lemma 9 (Min-entropy of systematic samples [22, Lemma 3.8])**.** *If* $\sigma > 2d \cdot q^{1/\ell + 2/(d(\ell-p))}$, *then* $\Pr_{\boldsymbol{A}'}[H_\infty(\mathcal{D}^{\mathsf{MLWE}}_{p,\ell-p,q,d,\sigma}(\boldsymbol{A}')) \geq d-1] \geq 1 - 2^{-d+1}$.

This is the no-bit-dropping case of [22, Lemma 3.8].

Lemma 10 (Structural non-cancellation)**.** *Let* $U \subseteq [n]$. *If* $(\mathbf{g})_U \in \mathrm{Col}_\mathbb{Z}(\hat{\mathbf{G}}_U)$, *then* U *is not certified.*

Proof. Suppose, for contradiction, that U is certified, that is, $\boldsymbol{\lambda}_U^\top\mathbf{G}_U = \mathbf{e}_1^\top$ for some $\boldsymbol{\lambda}_U \in \mathbb{Z}^{|U|}$. Since $\mathbf{G}_U = \begin{bmatrix}(\mathbf{g})_U & \hat{\mathbf{G}}_U\end{bmatrix}$, this means $\boldsymbol{\lambda}_U^\top(\mathbf{g})_U = 1$ and $\boldsymbol{\lambda}_U^\top\hat{\mathbf{G}}_U = \mathbf{0}^\top$. By hypothesis, $(\mathbf{g})_U = \hat{\mathbf{G}}_U\mathbf{a}$ for some $\mathbf{a} \in \mathbb{Z}^{k-1}$. Then $1 = \boldsymbol{\lambda}_U^\top(\mathbf{g})_U = (\boldsymbol{\lambda}_U^\top\hat{\mathbf{G}}_U)\mathbf{a} = 0$, a contradiction. □

Lemma 11 (Honest nonzero coefficient)**.** *Suppose that no set of size at most* t_{uf} *is certified, let* act *be certified with* $\boldsymbol{\lambda}_{\mathsf{act}}^\top\mathbf{G}_{\mathsf{act}} = \mathbf{e}_1^\top$, *and let* $|\mathsf{corrupt}| \leq t_{\mathsf{uf}}$. *Then* $\lambda_{\mathsf{act},i^*} \neq 0$ *for some* $i^* \in \mathsf{act} \setminus \mathsf{corrupt}$.

$\mathsf{Game}_{\mathsf{TS\text{-}UF}}$ and $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$

$L_{\mathsf{sid}}, L_H, L_{\mathsf{sig}} := \emptyset$; $\mathsf{tspar} \leftarrow \mathsf{Setup}(1^\kappa, n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$
$\mathsf{corrupt} \leftarrow \mathcal{A}(1^\kappa, \mathsf{tspar})$; **assert** $\mathsf{corrupt} \subseteq [n]$, $|\mathsf{corrupt}| \le t_{\mathsf{uf}}$
$\mathsf{honest} := [n] \setminus \mathsf{corrupt}$; $(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{Keygen}(\mathsf{tspar})$
Give $(\mathsf{sk}_i)_{i\in\mathsf{corrupt}}$ to $\mathcal{A}$; initialize st_i for $i \in \mathsf{honest}$
▷ $\mathsf{Game}_{\mathsf{TS\text{-}UF}}$:
$(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{H, (\mathsf{OSign}_k)_{k\in[\mathsf{rnd}]}}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak})$
return $\mathsf{msg}^* \notin L_{\mathsf{sig}} \wedge \mathsf{Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1$
▷ $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$:
$(\mathsf{sid}^*, (\mathsf{pm}_j^*)_{j\in\mathsf{corrupt}}) \leftarrow \mathcal{A}^{H, (\mathsf{OSign}_k)_{k\in[\mathsf{rnd}]}}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak})$
Fetch $(\mathsf{status}, r, \mathsf{act}^*, \mathsf{msg}^*, \mathsf{PM}^*, \mathsf{BL}^*) := L_{\mathsf{sid}}[\mathsf{sid}^*]$
assert $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}^*, \mathsf{act}^*, \mathsf{msg}^*) = 1$ and ($\mathsf{status} = \mathsf{abort}$ or $r = \mathsf{rnd}$)
if $\mathsf{status} = \mathsf{abort}$ **then**
  Let $S_\perp^i := \mathsf{BL}^*[r, i]$ for $i \in \mathsf{honest} \cap \mathsf{act}^*$
  return 1 iff some $S_\perp^i \neq S_\perp^j$, $S_\perp^i = \emptyset$, or $S_\perp^i \not\subseteq \mathsf{corrupt}$
end if
Record $(\mathsf{pm}_j^*)_{j\in\mathsf{corrupt}\cap\mathsf{act}^*}$ in PM^* as the corrupted round-rnd messages.
$\mathsf{out} \leftarrow \mathsf{Combine}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}, \mathsf{sid}^*, \mathsf{act}^*, \mathsf{msg}^*, \mathsf{PM}^*)$
if $\mathsf{out} = (\mathsf{abort}, S_\perp)$ **then**
  return $S_\perp = \emptyset \vee S_\perp \not\subseteq \mathsf{corrupt}$
end if
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{out}) = 0$

$\mathsf{OSign}_1(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$

assert $\mathsf{sid} \notin L_{\mathsf{sid}}$, $\mathsf{act} \subseteq [n]$, $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1$, $\mathsf{honest} \cap \mathsf{act} \neq \emptyset$
$(\mathsf{st}_i, \mathsf{pm}_i^1) \leftarrow \mathsf{ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$ for $i \in \mathsf{honest} \cap \mathsf{act}$
Record the honest messages and blame sets in PM and BL.
if some honest $\mathsf{pm}_i^1 = (\mathsf{abort}, S_\perp^i)$ **then**
  $L_{\mathsf{sid}}[\mathsf{sid}] := (\mathsf{abort}, 1, \mathsf{act}, \mathsf{msg}, \mathsf{PM}, \mathsf{BL})$; **return** $\perp$
end if
$L_{\mathsf{sig}} := L_{\mathsf{sig}} \cup \{\mathsf{msg}\}$; $L_{\mathsf{sid}}[\mathsf{sid}] := (\mathsf{running}, 1, \mathsf{act}, \mathsf{msg}, \mathsf{PM}, \mathsf{BL})$
return $(\mathsf{pm}_i^1)_{i\in\mathsf{honest}\cap\mathsf{act}}$

$\mathsf{OSign}_k(\mathsf{sid}, (\mathsf{pm}_j^{k-1})_{j\in\mathsf{corrupt}\cap\mathsf{act}})$, $k \in [2, \mathsf{rnd}]$

assert $\mathsf{sid} \in L_{\mathsf{sid}}$; fetch $(\mathsf{status}, r, \mathsf{act}, \mathsf{msg}, \mathsf{PM}, \mathsf{BL})$
assert $\mathsf{status} = \mathsf{running}$ and $r = k - 1$
Record the corrupted messages in PM, completing $(\mathsf{pm}_u^{k-1})_{u\in\mathsf{act}}$.
$(\mathsf{st}_i, \mathsf{pm}_i^k) \leftarrow \mathsf{ShareSign}_k(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{pm}_u^{k-1})_{u\in\mathsf{act}})$ for $i \in \mathsf{honest} \cap \mathsf{act}$
Record the honest messages and blame sets in PM and BL.
if some honest $\mathsf{pm}_i^k = (\mathsf{abort}, S_\perp^i)$ **then**
  $L_{\mathsf{sid}}[\mathsf{sid}] := (\mathsf{abort}, k, \dots)$; **return** $\perp$
end if
$L_{\mathsf{sid}}[\mathsf{sid}] := (\mathsf{running}, k, \dots)$; **return** $(\mathsf{pm}_i^k)_{i\in\mathsf{honest}\cap\mathsf{act}}$

$H(\mathsf{str}, \mathsf{digest})$

if $(\mathsf{str}, \mathsf{digest}) \notin L_H$ **then** sample $L_H[(\mathsf{str}, \mathsf{digest})]$ uniformly from the range
return $L_H[(\mathsf{str}, \mathsf{digest})]$

Figure 2. Unforgeability and identifiable-abort games for a TSS with IA, and their oracles.

Proof. Suppose, for contradiction, that $\lambda_{\mathsf{act},i} = 0$ for every $i \in \mathsf{act} \setminus \mathsf{corrupt}$, and let $U := \mathsf{act} \cap \mathsf{corrupt}$, so that $|U| \le t_{\mathsf{uf}}$. Let $\boldsymbol{\lambda}_U$ be the restriction of $\boldsymbol{\lambda}_{\mathsf{act}}$ to U. Since the omitted coefficients vanish, $\boldsymbol{\lambda}_U^\top \mathbf{G}_U = \boldsymbol{\lambda}_{\mathsf{act}}^\top \mathbf{G}_{\mathsf{act}} = \mathbf{e}_1^\top$, so U is certified. This contradicts the hypothesis; for $U = \emptyset$, the left-hand side is the zero vector, which is also impossible. □

Lemma 12 (Entropy loss of a scalar multiple)**.** *Let $\boldsymbol{X}$ be a random variable over $\mathcal{R}_q^p$, let $\lambda \in \mathbb{Z}$, and let $\boldsymbol{\delta} \in \mathcal{R}_q^p$ be fixed. Then $H_\infty(\lambda \boldsymbol{X} + \boldsymbol{\delta}) \ge H_\infty(\boldsymbol{X}) - dp \log_2 \gcd(\lambda, q)$.*

Proof. Let $g := \gcd(\lambda, q)$ and $\phi(\boldsymbol{x}) := \lambda \boldsymbol{x} + \boldsymbol{\delta}$. The map ϕ acts independently on the dp coefficients in $\mathbb{Z}_q$, and on each coefficient $y \mapsto \lambda y \bmod q$ is a group homomorphism whose kernel $\{y \in \mathbb{Z}_q : (q/g) \mid y\}$ has g elements. Hence every fiber of ϕ has at most g^{dp} elements, and for every $\boldsymbol{v}$, $\Pr[\phi(\boldsymbol{X}) = \boldsymbol{v}] = \sum_{\boldsymbol{x}\in\phi^{-1}(\boldsymbol{v})} \Pr[\boldsymbol{X} = \boldsymbol{x}] \le g^{dp}\, 2^{-H_\infty(\boldsymbol{X})}$. Taking the maximum over $\boldsymbol{v}$ and the negative logarithm gives the claim. □

Lemma 13 (Min-entropy of the aggregate commitment)**.** *Assume the setting of Corollary 2, but with an arbitrary modulus q and without the bound $\beta_{\boldsymbol{\lambda}} < q$, and suppose that no set of size at most t_{uf} is certified. Let $i^* \in \mathsf{act} \setminus \mathsf{corrupt}$ minimize $g := \gcd(\lambda_{\mathsf{act},i^*}, q)$ among the honest active signers with $\lambda_{\mathsf{act},i^*} \neq 0$. Then, except with probability 2^{-d+1} over $\boldsymbol{A}'$, conditioned on the adversary's view and on all randomness except $\boldsymbol{r}_{i^*}$, $H_\infty(\boldsymbol{w}) \ge d - 1 - dp \log_2 g$.*

Proof. By Lemma 11, some honest active signer has a nonzero coefficient, so i^* is well defined. Fix the adversary's view and all randomness except $\boldsymbol{r}_{i^*}$. The corrupted openings are then determined by the view, and the honest openings $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{r}_i$ for $i \neq i^*$ by the fixed randomness, so $\boldsymbol{w} = \sum_{i\in\mathsf{act}} \lambda_{\mathsf{act},i} \boldsymbol{w}_i = \lambda_{\mathsf{act},i^*} \boldsymbol{A}\boldsymbol{r}_{i^*} + \boldsymbol{\delta}$ for a fixed $\boldsymbol{\delta}$. Since the view is independent of $\boldsymbol{r}_{i^*}$, the conditional distribution of $\boldsymbol{A}\boldsymbol{r}_{i^*}$ is still $\mathcal{D}^{\mathsf{MLWE}}_{p,\ell-p,q,d,\sigma_{\boldsymbol{r}}}(\boldsymbol{A}')$. By Lemma 9, except with probability 2^{-d+1} over $\boldsymbol{A}'$, its min-entropy is at least $d-1$. Lemma 12, applied with $\boldsymbol{X} := \boldsymbol{A}\boldsymbol{r}_{i^*}$, $\lambda := \lambda_{\mathsf{act},i^*}$, and $\boldsymbol{\delta}$, gives $H_\infty(\boldsymbol{w}) \ge d - 1 - dp \log_2 g$. □

Proof of Corollary 2. For every U with $|U| \le t_{\mathsf{uf}} = k_p$, Item 5 of Section 2.4 gives $(\mathbf{g})_U \in \mathrm{Col}_{\mathbb{Z}}(\hat{\mathbf{G}}_U)$, so no such U is certified by Lemma 10, and Lemma 13 applies. Let i^* be the signer it provides. Since $0 < |\lambda_{\mathsf{act},i^*}| \le \beta_{\boldsymbol{\lambda}} < q$ and q is prime, $q \nmid \lambda_{\mathsf{act},i^*}$, so $g = \gcd(\lambda_{\mathsf{act},i^*}, q) = 1$, and the bound becomes $H_\infty(\boldsymbol{w}) \ge d - 1$, except with probability 2^{-d+1} over $\boldsymbol{A}'$. □

Appendix F. Running Times

Table 7 gives the median running times discussed in Section 8.4.

Appendix G. Hardness of RA-MHMLWE

G.1. Posterior-Resampling Reduction

We prove the base case of Lemma 1, without public auxiliary information, by adapting the Matrix Hint-MLWE posterior-resampling argument to the repeated target map.

TABLE 7. MEDIAN RUNNING TIMES OF LaTSEC IN MS ON ONE CORE, AT $|\mathsf{act}| = t_{\mathsf{sig}}$

n	κ	Keygen	SS_1	SS_2	SS_3	PV	Comb	Vfy
40	128	63	4.0	<0.01	2.3	1.0	18	0.76
40	192	82	4.3	<0.01	3.2	1.5	29	1.2
40	256	145	5.5	<0.01	4.1	2.2	44	1.7
60	128	109	3.3	<0.01	4.3	1.3	33	0.76
60	192	167	4.3	<0.01	6.0	2.0	52	1.2
60	256	220	7.1	<0.01	7.6	2.8	76	1.8

$|\mathsf{act}| = 26$ for $n = 40$ and 46 for $n = 60$. SS_k, PV, Comb, and Vfy denote $\mathsf{ShareSign}_k$, PartialVerify, Combine, and Verify; signing rounds are per signer, and Keygen covers all n parties.

The source Repeated-$\boldsymbol{A}$ MLWE secret is spherical with independent blocks, whereas the potentially correlated elliptic secret required by the hinted problem is created jointly by posterior resampling.

Proof of Lemma 1 without public auxiliary information. Let $N := dh(n+m)$ and $N_j := dn_j$ for $j \in [\nu]$. We first construct an adversary $\mathcal{B}_0$ against $\mathsf{RA\text{-}MLWE}_{n,m,q,d,\sigma}[h]$. The adversary receives a challenge $(\boldsymbol{A}, \boldsymbol{b})$, where either $\boldsymbol{b} = \overline{\boldsymbol{A}}^{\langle h \rangle}\boldsymbol{w}$ for a spherical secret whose coefficient embedding satisfies $\mathbf{w} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\sigma}$, or $\boldsymbol{b} \xleftarrow{\$} \mathcal{R}_q^{hn}$.

The reduction samples a dummy secret $\mathbf{s}' \leftarrow \mathcal{D}_{\mathbb{Z}^N,\mathbf{c}_\mathbf{s},\sqrt{\boldsymbol{\Sigma}_\mathbf{s}}}$. For every $j \in [\nu]$, it independently samples $\boldsymbol{M}_j \leftarrow \mathcal{M}_j$ and $\mathbf{y}_j \leftarrow \mathcal{D}_{\mathbb{Z}^{N_j},\mathbf{c}_{\mathbf{y}_j},\sqrt{\boldsymbol{\Sigma}_{\mathbf{y}_j}}}$. Let $\mathbf{M}_j \in \mathbb{Z}^{N_j \times N}$ denote the coefficient embedding of $\boldsymbol{M}_j$, and set $\mathbf{z}_j := \mathbf{M}_j\mathbf{s}' + \mathbf{y}_j$. For the sampled tuple, define $\boldsymbol{\Xi} := \boldsymbol{\Sigma}_\mathbf{s}^{-1} + \sum_{j=1}^{\nu} \mathbf{M}_j^\top \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1}\mathbf{M}_j$. By the uniform precision assumption, $\|\boldsymbol{\Xi}\|_2 \le 1/(2\sigma^2)$ for every tuple that may be sampled.

For the sampled hint matrices and values, define $\boldsymbol{\Sigma}_0 := \boldsymbol{\Xi}^{-1}$ and $\mathbf{c}_0 := \boldsymbol{\Sigma}_0\big(\boldsymbol{\Sigma}_\mathbf{s}^{-1}\mathbf{c}_\mathbf{s} + \sum_{j=1}^{\nu} \mathbf{M}_j^\top \boldsymbol{\Sigma}_{\mathbf{y}_j}^{-1}(\mathbf{z}_j - \mathbf{c}_{\mathbf{y}_j})\big)$. By the posterior-resampling identity of [47, Lemma 7], [12, Lemma 1], and [48, Lemma G.2], conditioned on the hint matrices and values, the desired secret has distribution $\mathcal{D}_{\mathbb{Z}^N,\mathbf{c}_0,\sqrt{\boldsymbol{\Sigma}_0}}$.

Since $\|\boldsymbol{\Xi}\|_2 \le 1/(2\sigma^2)$, we have $\lambda_{\min}(\boldsymbol{\Sigma}_0) \ge 2\sigma^2$, so $\boldsymbol{\Sigma}_0 - \sigma^2\mathbf{I}_N \succeq \sigma^2\mathbf{I}_N \succ 0$. The reduction samples $\mathbf{t} \leftarrow \mathcal{D}_{\mathbb{Z}^N,\mathbf{c}_0,\sqrt{\boldsymbol{\Sigma}_0 - \sigma^2\mathbf{I}_N}}$. Let $\boldsymbol{t} \in \mathcal{R}^{h(n+m)}$ denote the corresponding ring vector, reduced coefficient-wise modulo q when used in the principal target. The reduction sets $\boldsymbol{b}' := \boldsymbol{b} + \overline{\boldsymbol{A}}^{\langle h \rangle}\boldsymbol{t}$ and gives $\big(\boldsymbol{A}, \boldsymbol{b}', (\boldsymbol{M}_j)_{j\in[\nu]}, (\boldsymbol{z}_j)_{j\in[\nu]}\big)$ to $\mathcal{A}$.

All samples above are exact. The hint noise is sampled as in the problem. For the dummy secret and the correction, let $\mathbf{c}$ be a center and $\boldsymbol{\Sigma} \succeq \sigma^2\mathbf{I}_N$ a parameter matrix. The exact sampler of [49, Lemma 2.3], run on the coset $\boldsymbol{\Sigma}^{-1/2}(\mathbb{Z}^N - \mathbf{c})$ of the lattice $\boldsymbol{\Sigma}^{-1/2}\mathbb{Z}^N$ with parameter $\sqrt{2\pi}$ in its normalization, outputs $\mathbf{y}$ such that $\boldsymbol{\Sigma}^{1/2}\mathbf{y} + \mathbf{c}$ follows $\mathcal{D}_{\mathbb{Z}^N,\mathbf{c},\sqrt{\boldsymbol{\Sigma}}}$ exactly. The basis $\boldsymbol{\Sigma}^{-1/2}$ has Gram–Schmidt norms at most $\|\boldsymbol{\Sigma}^{-1/2}\|_2 \le 1/\sigma$, so the width condition of that lemma holds when $\sigma^2 \ge \ln(2N+4)/(2\pi^2)$. In our normalization, $\eta_\epsilon(\mathbb{Z}^N) \ge \eta_\epsilon(\mathbb{Z}) \ge \sqrt{\ln(2/\epsilon)}/(\sqrt{2}\pi)$, so $\sigma \ge \sqrt{2}\eta_\epsilon(\mathbb{Z}^N)$ gives $\sigma^2 \ge \ln(2/\epsilon)/\pi^2$, which suffices because $\epsilon \le 1/\sqrt{2N+4}$. Both $\boldsymbol{\Sigma}_\mathbf{s} \succeq \boldsymbol{\Xi}^{-1} \succeq 2\sigma^2\mathbf{I}_N$ and $\boldsymbol{\Sigma}_0 - \sigma^2\mathbf{I}_N \succeq \sigma^2\mathbf{I}_N$ satisfy $\boldsymbol{\Sigma} \succeq \sigma^2\mathbf{I}_N$.

Suppose first that the source challenge is real. By linearity of the repeated target map, $\boldsymbol{b}' = \overline{\boldsymbol{A}}^{\langle h \rangle}(\boldsymbol{w} + \boldsymbol{t})$. Set $\boldsymbol{\Sigma}_1 := \sigma^2\mathbf{I}_N$, $\boldsymbol{\Sigma}_2 := \boldsymbol{\Sigma}_0 - \sigma^2\mathbf{I}_N$, and $\boldsymbol{\Sigma}_3^{-1} := \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}$. Since $\boldsymbol{\Sigma}_2 \succeq \sigma^2\mathbf{I}_N$, we have $\|\boldsymbol{\Sigma}_3^{-1}\|_2 \le 2/\sigma^2 \le 1/\eta_\epsilon(\mathbb{Z}^N)^2$. As in the proof of [46, Corollary 1], this is the hypothesis of the convolution lemma [47, Lemmas 3 and 5] in the normalization [46, Eq. (2)], which therefore shows that $\mathbf{w}+\mathbf{t}$ is within statistical distance at most 2ϵ of $\mathcal{D}_{\mathbb{Z}^N,\mathbf{c}_0,\sqrt{\boldsymbol{\Sigma}_0}}$. Together with the posterior-resampling identity, this implies that the reduction's output is within statistical distance at most 2ϵ of the real Repeated-$\boldsymbol{A}$ Matrix Hint-MLWE distribution.

Suppose instead that $\boldsymbol{b}$ is independently uniform over $\mathcal{R}_q^{hn}$. Then $\boldsymbol{b} + \overline{\boldsymbol{A}}^{\langle h \rangle}\boldsymbol{t}$ remains uniformly distributed over $\mathcal{R}_q^{hn}$, so the reduction produces the random Repeated-$\boldsymbol{A}$ Matrix Hint-MLWE distribution exactly. Hence, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RA\text{-}MHMLWE}[h]}(\kappa) \le \mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{RA\text{-}MLWE}[h]}(\kappa) + 2\epsilon$. Applying Lemma 8 yields a PPT adversary $\mathcal{B}$ against $\mathsf{MLWE}_{n,m,q,d,\sigma}$ such that $\mathsf{Adv}_{\mathcal{B}_0}^{\mathsf{RA\text{-}MLWE}[h]}(\kappa) \le h\,\mathsf{Adv}_{\mathcal{B}}^{\mathsf{MLWE}}(\kappa)$. Combining the two inequalities proves $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RA\text{-}MHMLWE}[h]}(\kappa) \le h\,\mathsf{Adv}_{\mathcal{B}}^{\mathsf{MLWE}}(\kappa) + 2\epsilon$. □

G.2. Extension to Public Auxiliary Information

We now prove the public-auxiliary-information extension in the final claim of Lemma 1. Let aux denote the public auxiliary information jointly distributed with the hinted secret, and generate their joint distribution by first sampling aux from its marginal distribution and then sampling the secret from its conditional distribution given aux. The reduction first samples aux, exactly by assumption, and then runs conditionally on its outcome.

Fix an outcome a of aux. By assumption, conditioned on $\mathsf{aux} = a$, the secret is an elliptic discrete Gaussian over $\mathbb{Z}^N$ with publicly computable center $\mathbf{c}_{\mathbf{s}|a}$ and parameter matrix $\boldsymbol{\Sigma}_{\mathbf{s}|a}$. The reduction in Appendix G.1 therefore applies with these parameters, and its conditional dummy secret and correction are sampled exactly as there, since the uniform precision condition gives $\boldsymbol{\Sigma}_{\mathbf{s}|a} \succeq 2\sigma^2\mathbf{I}_N$ and $\boldsymbol{\Sigma}_0 - \sigma^2\mathbf{I}_N \succeq \sigma^2\mathbf{I}_N$ for every outcome a. Since the precision and smoothing conditions hold uniformly over all auxiliary-information outcomes and supported tuples of hint matrices, the conditional real branch is simulated within statistical distance at most 2ϵ, whereas the conditional random branch is simulated exactly. Averaging over the marginal distribution of aux preserves the same bound, so the reduction and factor-h advantage loss established above continue to hold.

Appendix H. The Underlying No-Abort Signature

We recall the non-threshold signature scheme underlying LaTSEC. The construction follows the Raccoon-style variant

of Lyubashevsky's Fiat-Shamir signature scheme [21], [22], [67], [68], adapted to the notation of this work and without compression or rounding.

The scheme avoids the rejection-sampling step of the original Fiat-Shamir construction. Such rejection sampling is undesirable in a threshold setting because a rejection by any active signer may require the entire signing session to restart in order to preserve the privacy of the shared signing key. Instead, following the noise-flooding approach of [69], the signing mask is sampled from a sufficiently wide distribution so that the response hides the signing key without rejection sampling. The same general approach has also been used in lattice-based threshold signature constructions [15], [16], [22]. Here, "no-abort" refers specifically to the absence of rejection-sampling aborts during signature generation.

Let $\boldsymbol{A} := \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{A}' \end{bmatrix} \in \mathcal{R}_q^{p\times\ell}$, where $\boldsymbol{A}' \xleftarrow{\$} \mathcal{R}_q^{p\times(\ell-p)}$ and $\ell > p$. The signing key is sampled as $\boldsymbol{s} = \begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{s}' \end{bmatrix} \leftarrow \mathcal{D}^{\ell}_{\sigma_s}$, where $\boldsymbol{e} \in \mathcal{R}^p$ and $\boldsymbol{s}' \in \mathcal{R}^{\ell-p}$. The verification target is $\boldsymbol{t} := \boldsymbol{A}\boldsymbol{s} = \boldsymbol{A}'\boldsymbol{s}' + \boldsymbol{e} \pmod q$. Consequently, $(\boldsymbol{A}', \boldsymbol{t})$ has the usual systematic MLWE form with secret $\boldsymbol{s}'$ and error $\boldsymbol{e}$. The complete construction is given in Figure 3.

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Keygen}(1^\kappa)$

1: $\boldsymbol{A}' \xleftarrow{\$} \mathcal{R}_q^{p\times(\ell-p)}$ and set $\boldsymbol{A} := \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{A}' \end{bmatrix}$.
2: $\boldsymbol{s} \leftarrow \mathcal{D}^{\ell}_{\sigma_s}$ ▷ $\boldsymbol{s} = \begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{s}' \end{bmatrix} \in \mathcal{R}^{\ell}$
3: $\boldsymbol{t} := \boldsymbol{A}\boldsymbol{s}$ ▷ $\boldsymbol{t} \in \mathcal{R}_q^p$
4: **return** $\mathsf{sk} := \boldsymbol{s}$; $\mathsf{vk} := (\boldsymbol{A}, \boldsymbol{t})$

$\mathsf{sig} \leftarrow \mathsf{Sign}(\mathsf{vk}, \mathsf{sk}, \mathsf{msg})$

1: $\boldsymbol{r} \leftarrow \mathcal{D}^{\ell}_{\sigma_r}$ ▷ signing mask in $\mathcal{R}^{\ell}$
2: $\boldsymbol{w} := \boldsymbol{A}\boldsymbol{r}$ ▷ commitment in $\mathcal{R}_q^p$
3: $c := H_c(\mathsf{vk}, \mathsf{msg}, \boldsymbol{w})$ ▷ challenge in $\mathcal{C}$
4: $\boldsymbol{z} := c\boldsymbol{s} + \boldsymbol{r}$ ▷ response in $\mathcal{R}^{\ell}$
5: **return** $\mathsf{sig} := (c, \boldsymbol{z})$

$\{0, 1\} \leftarrow \mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

1: $(c, \boldsymbol{z}) := \mathrm{Parse}(\mathsf{sig})$; **if** this fails, **return** 0
2: $\boldsymbol{w}' := \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}$
3: $c' := H_c(\mathsf{vk}, \mathsf{msg}, \boldsymbol{w}')$
4: **if** $c = c'$ **and** $\|\boldsymbol{z}\|_2 \le B_2$ **then**
5: **return** 1 ▷ sig is a valid signature
6: **else**
7: **return** 0 ▷ sig is an invalid signature
8: **end if**

Figure 3. The underlying Raccoon-style no-abort signature scheme, adapted from [22] without compression or rounding.

The commitment and verification relation are interpreted over $\mathcal{R}_q$, whereas the response $\boldsymbol{z}$ is viewed as an element of $\mathcal{R}^{\ell}$ and must satisfy the prescribed norm bound. For an honestly generated signature, $\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} = \boldsymbol{A}(c\boldsymbol{s}+\boldsymbol{r}) - c\boldsymbol{A}\boldsymbol{s} = \boldsymbol{A}\boldsymbol{r} = \boldsymbol{w}$. Hence, verification reconstructs the original commitment and recomputes the same challenge. The threshold construction in Section 3.2 preserves the signature format $(c, \boldsymbol{z})$ and this verification relation.

Appendix I. Full Specification of LaTSEC

I.1. Syntax and Session States

We model LaTSEC as a synchronous three-round threshold signature scheme with identifiable abort. Each signing execution is associated with a unique public session identifier sid, a message msg, and an active set $\mathsf{act} \subseteq [n]$. Messages sent in one round are assumed to be delivered consistently to all active parties before the next round begins, for example, through an authenticated broadcast channel or a public bulletin board. The scheme is defined by the following algorithm interfaces, where $\mathsf{PM} := (\mathsf{pm}_j^k)_{k\in[3], j\in\mathsf{act}}$:

1) $\mathsf{tspar} \leftarrow \mathsf{LaTSEC.Setup}(1^\kappa, n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$;
2) $(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{LaTSEC.Keygen}(\mathsf{tspar})$;
3) $(\mathsf{st}_i, \mathsf{pm}_i^1) \leftarrow \mathsf{LaTSEC.ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$;
4) $(\mathsf{st}_i, \mathsf{pm}_i^2) \leftarrow \mathsf{LaTSEC.ShareSign}_2(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{pm}_j^1)_{j\in\mathsf{act}})$;
5) $(\mathsf{st}_i, \mathsf{pm}_i^3) \mid (\mathsf{abort}, S_\perp^i) \leftarrow \mathsf{LaTSEC.ShareSign}_3(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{pm}_j^2)_{j\in\mathsf{act}})$;
6) $\mathsf{sig} \mid (\mathsf{abort}, S_\perp) \leftarrow \mathsf{LaTSEC.Combine}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{PM})$;
7) $\{0, 1, \perp\} \ni b \leftarrow \mathsf{LaTSEC.PartialVerify}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, \mathsf{PM})$;
8) $\{0, 1\} \ni b \leftarrow \mathsf{LaTSEC.Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$.

Here, $\mathsf{ak} := (\mathsf{ak}_i)_{i\in[n]}$ denotes the public abort information, pm_i^k denotes the message broadcast by party i in round k, and $\mathsf{pm}^k := (\mathsf{pm}_i^k)_{i\in\mathsf{act}}$ denotes the corresponding round transcript. The key-generation algorithm initializes the local state $\mathsf{st}_i := (\mathsf{tspar}, \mathsf{vk}, \mathsf{sk}_i)$ of each party $i \in [n]$, with an empty session table, and vk implicitly includes tspar. Whenever an algorithm returns an identifiable abort, $S_\perp^i$ or $S_\perp$ contains the parties identified as having supplied invalid protocol messages. The parameters and hash functions are those of Section 3.1.

Signer i maintains a long-lived user state containing its index i, its secret signing share sk_i, its public abort key ak_i, the global verification key vk, and the public parameters tspar. To avoid ambiguity, we reserve the notation pp for the public parameters of the underlying secret-sharing scheme. In our construction, tspar includes pp together with the other public parameters of the threshold signature scheme. Each execution of the signing protocol is indexed by a unique public session identifier sid. Signer i's per-session state is stored as $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] = (\mathsf{status}, \mathsf{rnd}, \mathsf{act}, \mathsf{msg}, \mathsf{open}, \mathsf{aux})$, where status records whether the session is running, completed, or aborted; rnd records the current signing round; act and msg record the public signing context (the active signer set and the message, respectively); open stores the signer's local ephemeral opening information, including the ephemeral

randomness, the revealed opening, and the corresponding hash commitment; and aux stores any earlier-round transcript data or other auxiliary values needed in subsequent rounds. Once a session is completed or aborted, its masks and other ephemeral data are erased, but its session identifier stays recorded in st_i, so the same sid is never accepted as fresh again.

I.2. Algorithms

The setup algorithm initializes the underlying SEC instance and collects the public parameters required by the threshold signature scheme. We set $t_{\mathsf{uf}} := k_p$ and $t_{\mathsf{sig}} := k_c$, where k_p and k_c are the privacy and empirical reconstruction thresholds of SEC, respectively. The secret dimension of the SEC instance is set to $d\ell$, corresponding to the coefficient embedding of vectors in $\mathcal{R}^\ell$. For notational simplicity, the globally fixed ring and module parameters are left implicit in tspar. The setup algorithm is given in Algorithm 1.

The trusted key-generation algorithm samples the public matrix and the short signing key of the underlying no-abort signature scheme. In particular, a short secret $\boldsymbol{s} \leftarrow \mathcal{D}^\ell_{\sigma_s}$ is sampled and then shared among the n parties using the dealing algorithm $\mathsf{SEC.Deal}_{\mathsf{pp}}$ from [46, Definition 11], as defined in [46, Eq. (3)]. Under the coefficient embedding, $\boldsymbol{s} \in \mathcal{R}^\ell$ is identified with an integer vector $\mathbf{s} \in \mathbb{Z}^{d\ell}$. The resulting integer shares are identified with ring-vector shares $(\boldsymbol{s}_i)_{i\in[n]}$ in $\mathcal{R}^\ell$. For each party $i \in [n]$, the secret signing-key share and public abort key are defined as $\mathsf{sk}_i := \boldsymbol{s}_i$ and $\mathsf{ak}_i := \boldsymbol{t}_i := \boldsymbol{A}\boldsymbol{s}_i$, respectively. The key-generation algorithm is given in Algorithm 2.

I.2.1. Three-Round Signing Protocol. Let sid identify a fresh signing execution with an admissible session input $(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$, where $\mathsf{act} \subseteq [n]$ and msg encodes $(\mathsf{sid}, \mathsf{act}, \boldsymbol{\lambda}_{\mathsf{act}})$ as in Section 3.1. Throughout, $\mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, \cdot)$ uses this vector $\boldsymbol{\lambda}_{\mathsf{act}}$, which every signer parses from the stored msg. The three round messages are given by $\mathsf{pm}^1_i := \mathsf{cmt}_i$, $\mathsf{pm}^2_i := \boldsymbol{w}_i$, and $\mathsf{pm}^3_i := \boldsymbol{z}_i$. The local session state $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$ stores signer i's ephemeral opening information and the transcript data required in subsequent rounds. Failed assertion checks in the algorithms below represent malformed or inconsistent invocations and are distinct from identifiable aborts caused by invalid protocol messages.

In the first round, each signer $i \in \mathsf{act}$ samples a fresh signing mask $\boldsymbol{r}_i$, computes its local commitment $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{r}_i$, and broadcasts the hash commitment $\mathsf{pm}^1_i := \mathsf{cmt}_i := H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, i, \boldsymbol{w}_i)$. The field open stores signer i's local opening information $(\boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i)$ for the session, while aux is reserved for auxiliary per-session data required in later rounds. The first round is given in Algorithm 3.

In the second round, each signer receives the first-round transcript $(\mathsf{pm}^1_j)_{j\in\mathsf{act}}$, stores it in the auxiliary part of its session state, and reveals its local commitment by broadcasting $\mathsf{pm}^2_i := \boldsymbol{w}_i$. Although the generic TSS syntax permits every round algorithm to output $(\mathsf{abort}, S^i_\perp)$, the concrete algorithm $\mathsf{LaTSEC.ShareSign}_2$ does not abort on well-formed inputs. Verification of the commitment openings is deferred to the third round, where an inconsistent opening can be attributed to the corresponding signer. The second round is given in Algorithm 4.

In the third round, each signer receives the commitment openings $(\mathsf{pm}^2_j)_{j\in\mathsf{act}} = (\boldsymbol{w}_j)_{j\in\mathsf{act}}$ and retrieves the first-round commitments $(\mathsf{cmt}_j)_{j\in\mathsf{act}}$ from its session state. For every $j \in \mathsf{act}$, the signer checks whether $\mathsf{cmt}_j = H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, j, \boldsymbol{w}_j)$. If the check fails for some j, signer i outputs $(\mathsf{abort}, \{j\})$, thereby identifying signer j as having supplied an invalid commitment opening. If all openings are valid, the aggregate commitment is reconstructed using the public reconstruction map of SEC: $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}}) \in \mathcal{R}^p_q$. Equivalently, if $\boldsymbol{W}[j, \cdot] := \boldsymbol{w}_j^\top$ and $\boldsymbol{\lambda}_{\mathsf{act}}$ is the reconstruction vector associated with act, then $\boldsymbol{w} = \boldsymbol{W}^\top \boldsymbol{\lambda}_{\mathsf{act}}$. The common challenge and partial response are then computed as $c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ and $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$, respectively. The third round is given in Algorithm 5.

I.2.2. Combination and Verification. We first define the public partial-verification procedure used to identify invalid partial responses. The combination algorithm applies this procedure to every active signer and either returns an identifiable abort or reconstructs the final response. Final verification then follows the verification relation of the underlying no-abort signature scheme in Appendix H.

Partial verification. The deterministic algorithm LaTSEC.PartialVerify publicly checks the partial response associated with the signer-specific abort key $\mathsf{ak}_i = \boldsymbol{t}_i$. Given the round-2 openings $(\boldsymbol{w}_j)_{j\in\mathsf{act}}$, the algorithm reconstructs the aggregate commitment $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}})$ and recomputes the common challenge $c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$. It accepts the partial response $\boldsymbol{z}_i$ if and only if $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{w}_i$ and $\|\boldsymbol{z}_i\|_2 \le B_1$. Thus, partial verification checks both the algebraic consistency and the shortness of the partial response using only public information. The procedure is given in Algorithm 6.

Combination. The aggregator applies LaTSEC.PartialVerify to every signer in the active set and defines the blame set $S_\perp$ containing all indices $i \in \mathsf{act}$ such that LaTSEC.PartialVerify for an input associated with ak_i returns 0. If $S_\perp \neq \emptyset$, then the algorithm returns $(\mathsf{abort}, S_\perp)$, identifying every signer whose partial response fails verification. Otherwise, the aggregator reconstructs the final response as $\boldsymbol{z} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{z}_j)_{j\in\mathsf{act}})$. Equivalently, if $\boldsymbol{Z}[j, \cdot] := \boldsymbol{z}_j^\top$, then $\boldsymbol{z} = \boldsymbol{Z}^\top \boldsymbol{\lambda}_{\mathsf{act}}$. The algorithm returns the final signature $\mathsf{sig} := (c, \boldsymbol{z})$. The combination procedure is given in Algorithm 7.

Missing and malformed messages. The algorithms above are stated for transcripts in which every message is present and parses. As stated in Figure 1, a missing or unparsable peer message fails the check that uses it; we make the cases explicit. In $\mathsf{LaTSEC.ShareSign}_3$, a signer j whose commitment or opening is missing or unparsable fails the commitment check, and the first such signer in the canonical order of act is returned. LaTSEC.PartialVerify

Algorithm 1 $\mathsf{tspar} \leftarrow \mathsf{LaTSEC.Setup}(1^\kappa, n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$

Require: Security parameter 1^κ, number of parties n, unforgeability threshold t_{uf}, and signing threshold t_{sig}.
Ensure: Public parameters $\mathsf{tspar} = (\mathsf{pp}, \mathcal{C}, B_1, B_2, \sigma_{\boldsymbol{s}}, \sigma_{\boldsymbol{r}}, \beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}})$.

Instantiate SEC for n parties, privacy threshold $k_p = t_{\mathsf{uf}}$, empirical reconstruction threshold $k_c = t_{\mathsf{sig}}$, and secret dimension $d\ell$. If no such instance is available, return $\perp$.
$\mathsf{pp} \leftarrow \mathsf{SEC.Setup}(1^\kappa)$
Fix the challenge set $\mathcal{C} := \{c \in \mathcal{R} : \|c\|_\infty = 1 \wedge \|c\|_1 = \omega_c\}$.
Fix the Gaussian widths $\sigma_{\boldsymbol{s}}$ and $\sigma_{\boldsymbol{r}}$.
Fix the reconstruction bounds $(\beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}})$, which, with pp, determine the admissible session inputs.
Fix the verification bounds B_1 and B_2 subject to the parameter conditions in Section 4.1.
$\mathsf{tspar} := (\mathsf{pp}, \mathcal{C}, B_1, B_2, \sigma_{\boldsymbol{s}}, \sigma_{\boldsymbol{r}}, \beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}})$
return tspar

Algorithm 2 $(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{LaTSEC.Keygen}(\mathsf{tspar})$

Require: Public parameters $\mathsf{tspar} = (\mathsf{pp}, \mathcal{C}, B_1, B_2, \sigma_{\boldsymbol{s}}, \sigma_{\boldsymbol{r}}, \beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}})$.
Ensure: Verification key vk, public abort information ak, and secret key shares $(\mathsf{sk}_i)_{i\in[n]}$.

Parse $\mathsf{tspar} = (\mathsf{pp}, \mathcal{C}, B_1, B_2, \sigma_{\boldsymbol{s}}, \sigma_{\boldsymbol{r}}, \beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}})$.
$\boldsymbol{A}' \xleftarrow{\$} \mathcal{R}_q^{p\times(\ell-p)}$ and set $\boldsymbol{A} := [\boldsymbol{I}_p \ \boldsymbol{A}']$.
$\boldsymbol{s} \leftarrow \mathcal{D}^\ell_{\sigma_{\boldsymbol{s}}}$ ▷ $\boldsymbol{s} \in \mathcal{R}^\ell$, with coefficient embedding $\mathbf{s} \in \mathbb{Z}^{d\ell}$
$(\boldsymbol{s}_i)_{i\in[n]} \leftarrow \mathsf{SEC.Deal}_{\mathsf{pp}}(\boldsymbol{s})$ ▷ Share $\boldsymbol{s}$ using [46, Eq. (3)]
$\boldsymbol{t} := \boldsymbol{A}\boldsymbol{s}$ ▷ $\boldsymbol{t} \in \mathcal{R}_q^p$
$\boldsymbol{t}_i := \boldsymbol{A}\boldsymbol{s}_i$ for every $i \in [n]$ ▷ $\boldsymbol{t}_i \in \mathcal{R}_q^p$
$\mathsf{vk} := (\boldsymbol{A}, \boldsymbol{t})$
$\mathsf{ak} := (\mathsf{ak}_i)_{i\in[n]} := (\boldsymbol{t}_i)_{i\in[n]}$
$\mathsf{sk}_i := \boldsymbol{s}_i$ for every $i \in [n]$
Initialize $\mathsf{st}_i := (\mathsf{tspar}, \mathsf{vk}, \mathsf{sk}_i)$ with an empty session table, for every $i \in [n]$.
return $(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in[n]})$

Algorithm 3 $(\mathsf{st}_i, \mathsf{pm}_i^1) \leftarrow \mathsf{LaTSEC.ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$

Require: Local state st_i, fresh session identifier sid, active set act, and message msg, where st_i contains $\mathsf{sk}_i = \boldsymbol{s}_i$, $\mathsf{vk} = (\boldsymbol{A}, \boldsymbol{t})$, and tspar.
Ensure: Updated state st_i and first-round message $\mathsf{pm}_i^1 = \mathsf{cmt}_i$.

assert $\{\mathsf{sid} \notin \mathsf{st}_i.\mathsf{session}\}$ ▷ session consistency check
assert $\{i \in \mathsf{act}\}$
assert $\{\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1\}$ ▷ admissibility check
$\boldsymbol{r}_i \leftarrow \mathcal{D}^\ell_{\sigma_{\boldsymbol{r}}}$ ▷ $\boldsymbol{r}_i \in \mathcal{R}^\ell$
$\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{r}_i$ ▷ $\boldsymbol{w}_i \in \mathcal{R}_q^p$
$\mathsf{cmt}_i := H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, i, \boldsymbol{w}_i)$ ▷ $\mathsf{cmt}_i \in \{0,1\}^{2\kappa}$
$\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] := (\mathsf{status} = \mathsf{running}, \mathsf{rnd} = 1, \mathsf{act}, \mathsf{msg}, \mathsf{open} = (\boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i), \mathsf{aux} = \emptyset)$
$\mathsf{pm}_i^1 := \mathsf{cmt}_i$
return $(\mathsf{st}_i, \mathsf{pm}_i^1)$

Algorithm 4 $(\mathsf{st}_i, \mathsf{pm}_i^2) \leftarrow \mathsf{LaTSEC.ShareSign}_2(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{pm}_j^1)_{j\in\mathsf{act}})$

Require: Local state st_i, session identifier sid, active set act, and first-round transcript $(\mathsf{pm}_j^1)_{j\in\mathsf{act}}$.
Ensure: Updated state st_i and second-round message $\mathsf{pm}_i^2 = \boldsymbol{w}_i$.

assert $\{\mathsf{sid} \in \mathsf{st}_i.\mathsf{session}\}$ ▷ session consistency check
Fetch $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] = (\mathsf{status}, \mathsf{rnd}, \mathsf{act}', \mathsf{msg}, \mathsf{open}, \mathsf{aux})$.
assert $\{\mathsf{status} = \mathsf{running} \wedge \mathsf{rnd} = 1 \wedge \mathsf{act}' = \mathsf{act}\}$
assert $\{i \in \mathsf{act}\}$
Parse $\mathsf{open} = (\boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i)$.
assert $\{\mathsf{pm}_i^1 = \mathsf{cmt}_i\}$
$\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] := (\mathsf{status} = \mathsf{running}, \mathsf{rnd} = 2, \mathsf{act}, \mathsf{msg}, \mathsf{open} = (\boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i), \mathsf{aux} = (\mathsf{pm}_j^1)_{j\in\mathsf{act}})$
$\mathsf{pm}_i^2 := \boldsymbol{w}_i$
return $(\mathsf{st}_i, \mathsf{pm}_i^2)$

for the signer i associated with ak_i first applies the checks that need no challenge: it returns 0 if $\boldsymbol{w}_i$ is missing or unparsable, or if $\boldsymbol{z}_i$ is missing, unparsable, or satisfies $\|\boldsymbol{z}_i\|_2 > B_1$. Otherwise, if the opening of another signer is missing or unparsable, $\boldsymbol{w}$ and c are undefined; the call is then a malformed invocation and returns $\perp$, which is not

Algorithm 5 $(\mathsf{st}_i, \mathsf{pm}_i^3) \mid (\mathsf{abort}, S_\perp^i) \leftarrow \mathsf{LaTSEC.ShareSign}_3(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{pm}_j^2)_{j\in\mathsf{act}})$

Require: Local state st_i, session identifier sid, active set act, and second-round transcript $(\mathsf{pm}_j^2)_{j\in\mathsf{act}}$.
Ensure: Updated state and third-round message $\mathsf{pm}_i^3 = \boldsymbol{z}_i$, or an identifiable abort $(\mathsf{abort}, S_\perp^i)$.

assert $\{\mathsf{sid} \in \mathsf{st}_i.\mathsf{session}\}$ ▷ session consistency check
Fetch $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] = (\mathsf{status}, \mathsf{rnd}, \mathsf{act}', \mathsf{msg}, \mathsf{open}, \mathsf{aux})$.
assert $\{\mathsf{status} = \mathsf{running} \wedge \mathsf{rnd} = 2 \wedge \mathsf{act}' = \mathsf{act}\}$
assert $\{i \in \mathsf{act}\}$
Parse $\mathsf{open} = (\boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i)$.
Parse $\mathsf{aux} = (\mathsf{pm}_j^1)_{j\in\mathsf{act}} = (\mathsf{cmt}_j)_{j\in\mathsf{act}}$.
Parse $(\mathsf{pm}_j^2)_{j\in\mathsf{act}} = (\boldsymbol{w}_j)_{j\in\mathsf{act}}$.
assert $\{\mathsf{pm}_i^2 = \boldsymbol{w}_i\}$ ▷ own opening delivered unchanged
for $j \in \mathsf{act}$ **do**
  if $\mathsf{cmt}_j \neq H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, j, \boldsymbol{w}_j)$ **then**
    Set $\mathsf{status} := \mathsf{aborted}$ in $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$ and erase $\boldsymbol{r}_i$.
    return $(\mathsf{abort}, \{j\})$ ▷ identify signer j
  end if
end for
$\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}})$ ▷ $\boldsymbol{w} \in \mathcal{R}_q^p$
$c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ ▷ $c \in \mathcal{C} \subseteq \mathcal{R}$
$\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$ ▷ $\boldsymbol{z}_i \in \mathcal{R}^\ell$
$\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] := (\mathsf{status} = \mathsf{completed}, \mathsf{rnd} = 3, \mathsf{act}, \mathsf{msg}, \mathsf{open} = (\boldsymbol{w}_i, \mathsf{cmt}_i), \mathsf{aux} = ((\mathsf{pm}_j^1)_{j\in\mathsf{act}}, (\mathsf{pm}_j^2)_{j\in\mathsf{act}}))$
$\mathsf{pm}_i^3 := \boldsymbol{z}_i$
return $(\mathsf{st}_i, \mathsf{pm}_i^3)$

Algorithm 6 $\{0, 1, \perp\} \leftarrow \mathsf{LaTSEC.PartialVerify}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_j^k)_{k\in[3],\, j\in\mathsf{act}})$

Require: Public parameters $\mathsf{tspar} = (\mathsf{pp}, \mathcal{C}, B_1, B_2, \sigma_{\boldsymbol{s}}, \sigma_{\boldsymbol{r}}, \beta_{\boldsymbol{\lambda}}, \omega_{\boldsymbol{\lambda}})$, verification key $\mathsf{vk} = (\boldsymbol{A}, \boldsymbol{t})$, abort key $\mathsf{ak}_i = \boldsymbol{t}_i$, session identifier sid, active set act, message msg, and signing transcript $(\mathsf{pm}_j^k)_{k\in[3],\, j\in\mathsf{act}}$.
Ensure: 1 if the partial response associated with ak_i is valid, and 0 otherwise.

Parse $\mathsf{pm}_i^2 = \boldsymbol{w}_i$ and $\mathsf{pm}_i^3 = \boldsymbol{z}_i$.
Parse $(\mathsf{pm}_j^2)_{j\in\mathsf{act}} = (\boldsymbol{w}_j)_{j\in\mathsf{act}}$.
$\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}})$ ▷ aggregate commitment
$c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ ▷ common challenge
if $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i \neq \boldsymbol{w}_i$ **or** $\|\boldsymbol{z}_i\|_2 > B_1$ **then**
  return 0
else
  return 1
end if

Algorithm 7 $\mathsf{sig} \mid (\mathsf{abort}, S_\perp) \leftarrow \mathsf{LaTSEC.Combine}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_j^k)_{k\in[3],\, j\in\mathsf{act}})$

Require: Public parameters tspar, verification key $\mathsf{vk} = (\boldsymbol{A}, \boldsymbol{t})$, public abort information $\mathsf{ak} = (\mathsf{ak}_i)_{i\in[n]} = (\boldsymbol{t}_i)_{i\in[n]}$, session identifier sid, active set act, message msg, and signing transcript $(\mathsf{pm}_j^k)_{k\in[3],\, j\in\mathsf{act}}$.
Ensure: A signature $\mathsf{sig} = (c, \boldsymbol{z})$ or an identifiable abort $(\mathsf{abort}, S_\perp)$.

Parse $(\mathsf{pm}_j^2)_{j\in\mathsf{act}} = (\boldsymbol{w}_j)_{j\in\mathsf{act}}$.
Parse $(\mathsf{pm}_j^3)_{j\in\mathsf{act}} = (\boldsymbol{z}_j)_{j\in\mathsf{act}}$.
$\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{w}_j)_{j\in\mathsf{act}})$ ▷ aggregate commitment
$c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ ▷ common challenge
$S_\perp := \left\{i \in \mathsf{act} : \mathsf{LaTSEC.PartialVerify}\left(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_j^k)_{k\in[3],\, j\in\mathsf{act}}\right) = 0\right\}$
if $S_\perp \neq \emptyset$ **then**
  return $(\mathsf{abort}, S_\perp)$
end if
$\boldsymbol{z} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\boldsymbol{z}_j)_{j\in\mathsf{act}})$ ▷ aggregate response
$\mathsf{sig} := (c, \boldsymbol{z})$
return sig

a verdict on i. LaTSEC.Combine first collects the signers whose openings are missing or unparsable. If there are any, it returns $(\mathsf{abort}, S_\perp)$, where $S_\perp$ consists of these signers and of the signers j whose responses are missing, unparsable, or satisfy $\|\boldsymbol{z}_j\|_2 > B_1$; otherwise it proceeds as in Algorithm 7. In each case, a blamed signer sent a message that is missing or unparsable or that fails a public check, and by Lemma 3 no honest signer does so under $\mathsf{Good}_{\mathsf{tail}}$. If signer i finds its own commitment or opening missing from, or altered in, the messages delivered to it, the assertions of

$\mathsf{LaTSEC.ShareSign}_2$ and $\mathsf{LaTSEC.ShareSign}_3$ fail, so the call is a malformed invocation and blames nobody. Both implementations follow these rules.

Verification. Final verification follows the underlying no-abort signature scheme from Appendix H. The public message context canonically determines the session identifier sid and active set act. Equivalently, if m denotes the external payload, then $\mathsf{msg} := \mathrm{Encode}(\mathsf{sid}, \mathsf{act}, \boldsymbol{\lambda}_{\mathsf{act}}, \mathsf{m})$, as in Section 3.1. We retain the expanded challenge-hash notation $H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ throughout the construction, where sid and act are recovered from the public message context. Given $\mathsf{sig} = (c, \boldsymbol{z})$, the verifier computes $\boldsymbol{w} := \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} \in \mathcal{R}_q^p$ and accepts if and only if $c = H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ and $\|\boldsymbol{z}\|_2 \le B_2$. The verification procedure is given in Algorithm 8.

I.3. Correctness Game

Correctness is required only for admissible session inputs. More precisely, for every session input $(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$ with $\mathsf{act} \subseteq [n]$, $|\mathsf{act}| \ge t_{\mathsf{sig}}$, and $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1$, an honest execution of the signing protocol should output a signature sig such that $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}) = 1$. We formalize this through the experiment $\mathsf{Game}_{\mathsf{correct}}$ shown in Figure 4.

In the correctness experiment, the parameters $(n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$ are fixed as part of the scheme instance, while act, sid, and msg describe the honest signing session under consideration. If $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 0$, then the session input is malformed or not supported by the underlying sharing structure, and any resulting failure is attributed to the entity or mechanism responsible for selecting the session input, rather than to the honest signing protocol itself.

In our concrete instantiation, the admissibility predicate is the one described in Section 3.1: $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1$ if and only if msg canonically encodes $(\mathsf{sid}, \mathsf{act}, \boldsymbol{\lambda}_{\mathsf{act}})$ for some $\boldsymbol{\lambda}_{\mathsf{act}} \in \mathbb{Z}^{|\mathsf{act}|}$, $|\mathsf{act}| \ge t_{\mathsf{sig}}$, $\boldsymbol{\lambda}_{\mathsf{act}}^\top \mathbf{G}_{\mathsf{act}} = \mathbf{e}_1^\top$, $\|\boldsymbol{\lambda}_{\mathsf{act}}\|_\infty \le \beta_{\boldsymbol{\lambda}}$, and $wt_H(\boldsymbol{\lambda}_{\mathsf{act}}) \le \omega_{\boldsymbol{\lambda}}$. This predicate is deterministic and consists of exact integer checks on public data, so it does not depend on how $\boldsymbol{\lambda}_{\mathsf{act}}$ was computed. Every admissible input satisfies $\mathbf{e}_1^\top \in \mathrm{Row}_{\mathbb{Z}}(\mathbf{G}_{\mathsf{act}})$, so reconstruction is exact. If the initiator takes $\boldsymbol{\lambda}_{\mathsf{act}}$ from [46, Algorithm 1], admissibility nearly coincides with certification, which holds with estimated probability at least 0.99 for every tested size $t \ge k_c$; the bounds reject only a small fraction of certified sets (Section 7.1).

Definition 11 (Operational correctness of ramp-TSS)**.** Fix $n \in \mathbb{N}$, a signing threshold $t_{\mathsf{sig}} \in [n]$, and an admissibility predicate $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \cdot, \cdot, \cdot)$. Let $\epsilon_{\mathrm{cor}} \in [0, 1]$. We say that an $(n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$-TSS with IA is ϵ_{cor}-*operationally correct with respect to* $\mathcal{P}$ if, for every session input $(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$ with $\mathsf{act} \subseteq [n]$, $|\mathsf{act}| \ge t_{\mathsf{sig}}$, and $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1$, $\Pr[\mathsf{Game}_{\mathsf{correct}} = 1] \ge 1 - \epsilon_{\mathrm{cor}}$, where $\mathsf{Game}_{\mathsf{correct}}$ is defined in Figure 4. The probability is taken over the randomness of TS.Setup, Keygen, the honest signing algorithms $\mathsf{ShareSign}_k$, and Combine, if randomized. If no additional set-dependent restriction is required, we may set $\mathcal{P} \equiv 1$, while retaining the separate requirement $|\mathsf{act}| \ge t_{\mathsf{sig}}$.

Finally, correctness on admissible session inputs follows from identifiable aborts in the all-honest setting. Consider an adversary $\mathcal{B}$ in $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$ that chooses $\mathsf{corrupt} = \emptyset$ and executes the signing protocol honestly on an admissible session input $(\mathsf{sid}, \mathsf{act}, \mathsf{msg})$. Since admissibility includes $\boldsymbol{\lambda}_{\mathsf{act}}^\top \mathbf{G}_{\mathsf{act}} = \mathbf{e}_1^\top$, reconstruction is exact, and any abort by an honest party violates identifiable abort, because a valid blamed abort would require $\emptyset \ne S_\perp \subseteq \mathsf{corrupt} = \emptyset$, which is impossible. The same reasoning applies if Combine returns an abort. If instead Combine outputs a signature rejected by Verify, then the non-aborting branch of $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$ returns 1. Consequently, $\Pr[\mathsf{Game}_{\mathsf{correct}} = 0] \le \mathsf{Adv}_{\mathcal{B}}^{\mathsf{TS\text{-}IA}}(\kappa)$, so the operational correctness error satisfies $\epsilon_{\mathrm{cor}} \le \mathsf{Adv}_{\mathcal{B}}^{\mathsf{TS\text{-}IA}}(\kappa)$. The coverage error ϵ_c of SEC [46, Definition 11] affects availability rather than correctness: it bounds the probability that a random active set of size at least t_{sig} is not certified and hence has no admissible input.

Appendix J. Proofs for Section 4

The tail bounds behind $\mathsf{Good}_{\mathsf{tail}}$ are proved in [46, Section 6.4.1 and Appendix I]; we prove the remaining statements of Section 4.2.

J.1. Honest Partial-Response Bounds

Proof of Lemma 3. Assume that $\mathsf{Good}_{\mathsf{tail}}$ holds and let i be an honest signer participating in any signing query. By [46, Lemma 12], the bounds $\|\boldsymbol{s}\|_\infty \le \|\boldsymbol{s}\|_2 \le \beta_{\boldsymbol{s}}$ and $|\mathbf{R}[i', j']| \le \beta_{\mathbf{R}}$ imply $\|\boldsymbol{s}_i\|_\infty \le \beta_{\mathbf{g}}\beta_{\boldsymbol{s}} + c_{\exp}\beta_{\mathbf{R}}$. Since $\boldsymbol{s}_i \in \mathcal{R}^\ell$ has coefficient dimension $D = d\ell$, $\|\boldsymbol{s}_i\|_2 \le \sqrt{D}\,(\beta_{\mathbf{g}}\beta_{\boldsymbol{s}} + c_{\exp}\beta_{\mathbf{R}})$. The honest partial response is $\boldsymbol{z}_i = c\boldsymbol{s}_i + \boldsymbol{r}_i$. Since $c \in \mathcal{C}$ satisfies $\|c\|_1 = \omega_c$, multiplication by c satisfies $\|c\boldsymbol{s}_i\|_2 \le \omega_c\|\boldsymbol{s}_i\|_2$. Moreover, $\mathsf{Good}_{\mathsf{tail}}$ implies $\|\boldsymbol{r}_i\|_2 \le \beta_{\boldsymbol{r}}$. Consequently,

$$\begin{aligned}\|\boldsymbol{z}_i\|_2 &\le \|c\boldsymbol{s}_i\|_2 + \|\boldsymbol{r}_i\|_2 \\ &\le \omega_c\sqrt{D}\,(\beta_{\mathbf{g}}\beta_{\boldsymbol{s}} + c_{\exp}\beta_{\mathbf{R}}) + \beta_{\boldsymbol{r}} \le B_1.\end{aligned}$$

It remains to verify the algebraic relation checked by LaTSEC.PartialVerify. For an honest signer, $\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{s}_i$ and $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{r}_i$. Therefore,

$$\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{A}\,(c\boldsymbol{s}_i + \boldsymbol{r}_i) - c\boldsymbol{A}\boldsymbol{s}_i = \boldsymbol{A}\boldsymbol{r}_i = \boldsymbol{w}_i.$$

Thus, an honest partial response passes both checks of LaTSEC.PartialVerify, proving Lemma 3. □

J.2. Proof of Identifiable-Abort Security

Proof of Theorem 1. By Item 3 of the parameter conditions in Section 4.1, the event $\mathsf{Good}_{\mathsf{tail}}$ fails with probability at most $3{\cdot}2^{-\kappa}$. It therefore suffices to show that, conditioned on

Algorithm 8 $\{0,1\} \leftarrow \mathsf{LaTSEC.Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

Require: Verification key $\mathsf{vk} = (\boldsymbol{A}, \boldsymbol{t})$, context-bound message msg, and signature $\mathsf{sig} = (c, \boldsymbol{z})$.
Ensure: 1 if the signature is valid and 0 otherwise.

Parse $(\mathsf{sid}, \mathsf{act})$ from msg and $(c, \boldsymbol{z}) := \mathsf{sig}$; **if** either fails, **return** 0.
$\boldsymbol{w} := \boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t}$ ▷ $\boldsymbol{w} \in \mathcal{R}_q^p$
if $c = H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$ **and** $\|\boldsymbol{z}\|_2 \leq B_2$ **then**
    return 1 ▷ sig is valid
else
    return 0 ▷ sig is invalid
end if

$\mathsf{Game}_{\mathsf{correct}}$

$\mathsf{tspar} \leftarrow \mathsf{TS.Setup}(1^\kappa, n, t_{\mathsf{uf}}, t_{\mathsf{sig}})$
$(\mathsf{vk}, \mathsf{ak}, (\mathsf{sk}_i)_{i\in[n]}) \leftarrow \mathsf{Keygen}(\mathsf{tspar})$
assert$\{\mathsf{act} \subseteq [n] \ \wedge \ |\mathsf{act}| \geq t_{\mathsf{sig}} \wedge \mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1\}$
Set $\mathsf{st}_i.\mathsf{sk} := \mathsf{sk}_i$, $\mathsf{st}_i.\mathsf{ak} := \mathsf{ak}_i$, $\mathsf{st}_i.\mathsf{vk} := \mathsf{vk}$, and $\mathsf{st}_i.\mathsf{tspar} := \mathsf{tspar}$ for all $i \in \mathsf{act}$.
for $i \in \mathsf{act}$ **do**
    $(\mathsf{st}_i, \mathsf{pm}_i^1) \leftarrow \mathsf{ShareSign}_1(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, \mathsf{msg})$
    if $\mathsf{pm}_i^1 = (\mathsf{abort}, \cdot)$ **then**
        return 0 ▷ early termination
    end if
end for
for $k = 2$ **to** rnd **do**
    for $i \in \mathsf{act}$ **do**
        $(\mathsf{st}_i, \mathsf{pm}_i^k) \leftarrow \mathsf{ShareSign}_k(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{pm}_j^{k-1})_{j\in\mathsf{act}})$
        if $\mathsf{pm}_i^k = (\mathsf{abort}, \cdot)$ **then**
            return 0 ▷ early termination
        end if
    end for
end for
$\mathsf{out} \leftarrow \mathsf{Combine}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}, (\mathsf{pm}_i^k)_{k\in[\mathsf{rnd}],\, i\in\mathsf{act}})$
if $\mathsf{out} = (\mathsf{abort}, \cdot)$ **then**
    return 0
end if
Parse $\mathsf{out} = \mathsf{sig}$
return $\mathsf{Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$

Figure 4. Correctness game for a threshold signature with IA; the scheme is correct if $\mathsf{Game}_{\mathsf{correct}}$ returns 1 with high probability.

$\mathsf{Good}_{\mathsf{tail}}$, no adversary can make $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$ return 1. Consider a session sid^* selected by the adversary, with active set act^* and context-bound message msg^*. The game proceeds only if $\mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}^*, \mathsf{act}^*, \mathsf{msg}^*) = 1$; in particular, all honest signers parse the same $\boldsymbol{\lambda}_{\mathsf{act}^*}$ from msg^*.

(Consistency of honest commitment openings.) In $\mathsf{LaTSEC.ShareSign}_1$, an honest signer $i \in \mathsf{act}^*$ samples $\boldsymbol{r}_i$, computes $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{r}_i$, and broadcasts $\mathsf{cmt}_i = H_{\mathsf{cmt}}(\mathsf{sid}^*, \mathsf{vk}, \mathsf{act}^*, i, \boldsymbol{w}_i)$. In $\mathsf{LaTSEC.ShareSign}_2$, the signer reveals the same stored value $\boldsymbol{w}_i$. Hence, every honest signer supplies a valid opening of its first-round commitment. The assertion checks in $\mathsf{LaTSEC.ShareSign}_1$ and $\mathsf{LaTSEC.ShareSign}_2$ enforce session consistency and well-formed invocation conditions. They do not produce blamed aborts in $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$.

(Abort during $\mathsf{LaTSEC.ShareSign}_3$***.)*** An abort during $\mathsf{LaTSEC.ShareSign}_3$ occurs only if some $j \in \mathsf{act}^*$ fails $\mathsf{cmt}_j = H_{\mathsf{cmt}}(\mathsf{sid}^*, \mathsf{vk}, \mathsf{act}^*, j, \boldsymbol{w}_j)$. No honest signer can fail this check because an honest signer reveals the same value to which it committed. Thus, every signer blamed in this round is corrupted. By the consistent-delivery assumption, all honest active parties receive the same first- and second-round transcripts. They therefore perform the same deterministic opening checks in the canonical order of act^* and return the same singleton blame set $\{j\}$. Consequently, the common blame set is non-empty and contained in corrupt, so the abort branch of $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$ returns 0.

(Abort returned by $\mathsf{LaTSEC.Combine}$***.)*** Suppose that all active parties complete the third signing round and that $\mathsf{LaTSEC.Combine}(\mathsf{tspar}, \mathsf{vk}, \mathsf{ak}, \mathsf{sid}^*, \mathsf{act}^*, \mathsf{msg}^*, \mathsf{PM}^*)$ returns $(\mathsf{abort}, S_\perp)$. If an opening in PM^* is missing or unparsable, $\mathsf{LaTSEC.Combine}$ blames the senders of such openings and of the responses that are missing, unparsable, or longer than B_1 (Section I). Under consistent delivery, every honest signer's opening and response are present and parse, and by Lemma 3 its response has norm at most B_1, so this blame set is non-empty and contained in corrupt. Otherwise, by Algorithm 7, $S_\perp = \{i \in \mathsf{act}^* : \mathsf{LaTSEC.PartialVerify}(\cdots) = 0\}$, and the algorithm returns an abort only if $S_\perp \neq \emptyset$. Let $i \in \mathsf{honest} \cap \mathsf{act}^*$.

By Lemma 3, the honest response satisfies $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{w}_i$ and $\|\boldsymbol{z}_i\|_2 \le B_1$. Hence, LaTSEC.PartialVerify returns 1 for every honest signer, and therefore $S_\perp \subseteq \mathsf{corrupt}$. Since $S_\perp \neq \emptyset$, the output is a valid identifiable abort, and the game returns 0.

(Signature returned by* LaTSEC.Combine*.) It remains to consider the case in which LaTSEC.Combine returns $\mathsf{sig} = (c, \boldsymbol{z})$. Since the algorithm did not abort, every active signer passed LaTSEC.PartialVerify. Therefore, for every $i \in \mathsf{act}^*$, $\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i = \boldsymbol{w}_i$ and $\|\boldsymbol{z}_i\|_2 \le B_1$. Let $\boldsymbol{\lambda}_{\mathsf{act}^*} = (\lambda_i)_{i\in\mathsf{act}^*}$ be the reconstruction vector encoded in msg^* and used by $\mathsf{SEC.Rec}_{\mathsf{pp}}$. Since the session input is admissible, $\|\boldsymbol{\lambda}_{\mathsf{act}^*}\|_\infty \le \beta_{\boldsymbol{\lambda}}$ and $wt_H(\boldsymbol{\lambda}_{\mathsf{act}^*}) \le \omega_{\boldsymbol{\lambda}}$, so $\|\boldsymbol{\lambda}_{\mathsf{act}^*}\|_1 \le \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}$. By the linearity and matrix-compatibility properties of $\mathsf{SEC.Rec}_{\mathsf{pp}}$ from [46, Lemma 17 and Lemma 18], $\boldsymbol{z} = \sum_{i\in\mathsf{act}^*}\lambda_i\boldsymbol{z}_i$, $\boldsymbol{w} = \sum_{i\in\mathsf{act}^*}\lambda_i\boldsymbol{w}_i$, and $\boldsymbol{t} = \sum_{i\in\mathsf{act}^*}\lambda_i\boldsymbol{t}_i$. It follows that

$$\begin{aligned}\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} &= \sum_{i\in\mathsf{act}^*}\lambda_i\left(\boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i\right)\\ &= \sum_{i\in\mathsf{act}^*}\lambda_i\boldsymbol{w}_i = \boldsymbol{w}.\end{aligned}$$

Moreover,

$$\begin{aligned}\|\boldsymbol{z}\|_2 &= \left\|\sum_{i\in\mathsf{act}^*}\lambda_i\boldsymbol{z}_i\right\|_2 \le \sum_{i\in\mathsf{act}^*}|\lambda_i|\|\boldsymbol{z}_i\|_2\\ &\le \|\boldsymbol{\lambda}_{\mathsf{act}^*}\|_1 B_1 \le \omega_{\boldsymbol{\lambda}}\beta_{\boldsymbol{\lambda}}B_1 \le B_2.\end{aligned}$$

The challenge used by LaTSEC.Combine is $c = H_c(\mathsf{sid}^*, \mathsf{vk}, \mathsf{act}^*, \mathsf{msg}^*, \boldsymbol{w})$. Since the context-bound message msg^* determines sid^* and act^*, the verification algorithm recomputes the same challenge. Together with $\boldsymbol{A}\boldsymbol{z} - c\boldsymbol{t} = \boldsymbol{w}$ and $\|\boldsymbol{z}\|_2 \le B_2$, this gives $\mathsf{LaTSEC.Verify}(\mathsf{vk}, \mathsf{msg}^*, (c, \boldsymbol{z})) = 1$. Therefore, the game also returns 0 in the non-aborting branch.

Conditioned on $\mathsf{Good}_{\mathsf{tail}}$, the adversary cannot make $\mathsf{Game}_{\mathsf{TS\text{-}IA}}$ return 1. Consequently, $\mathsf{Adv}^{\mathsf{TS\text{-}IA}}_{\mathcal{A}}(\kappa) \le \Pr[\neg\mathsf{Good}_{\mathsf{tail}}] \le 3\cdot 2^{-\kappa}$, where the final inequality follows from Item 3 of the parameter conditions in Section 4.1. This proves Theorem 1. □

J.3. Proof of Correctness

Corollary 1 follows from the non-aborting branch of Theorem 1. Conditioned on $\mathsf{Good}_{\mathsf{tail}}$, that branch shows that every signature output by LaTSEC.Combine satisfies both checks of LaTSEC.Verify. The remaining failure probability is at most $3\cdot 2^{-\kappa}$ by Item 3 of the parameter conditions in Section 4.1.

Appendix K. Constructions and Proofs for Section 5

K.1. Algebraic Completion and Uniform Sampling

We give the explicit affine completion underlying SimAK. Recall from (3) that the coefficient-embedded public abort-key tuple satisfies $\mathbf{T} = \left(\mathbf{g}\mathbf{t}^\top + \hat{\mathbf{G}}\mathbf{R}_{\mathbf{t}}\right) \bmod q$, where $\mathbf{R}_{\mathbf{t}} = \mathbf{R}\mathbf{A}^\top$. Since $\mathbf{G} = \left[\mathbf{g}\ \hat{\mathbf{G}}\right]$ and $\mathbf{t}$ are public, the abort-key tuple reveals $\hat{\mathbf{G}}\mathbf{R}_{\mathbf{t}} = \mathbf{T} - \mathbf{g}\mathbf{t}^\top \bmod q$. If $\hat{\mathbf{G}}$ has full column rank over $\mathbb{Z}_q$, this value uniquely determines $\mathbf{R}_{\mathbf{t}}$. The construction below does not require this global rank condition. Instead, for each prescribed coordinate set U within the privacy threshold, it uses a unimodular minor of $\hat{\mathbf{G}}_U$ to construct an integer right inverse.

Let $U \subseteq [n]$ satisfy $1 \le |U| \le k_p$. By [46, Lemma 4], there exists a set $J_U \subseteq [k-1]$ with $|J_U| = |U|$, such that $\hat{\mathbf{G}}_U[\cdot, J_U] \in \mathbb{Z}^{|U|\times|U|}$ is unimodular. In particular, $\det\left(\hat{\mathbf{G}}_U[\cdot, J_U]\right) \in \{-1, 1\}$, so its inverse has integer entries. Define $\mathbf{K}_U \in \mathbb{Z}^{(k-1)\times|U|}$ by placing $\hat{\mathbf{G}}_U[\cdot, J_U]^{-1}$ in the rows indexed by J_U and placing zero rows in all remaining positions. Then $\hat{\mathbf{G}}_U\mathbf{K}_U = \mathbf{I}_{|U|}$. Thus, $\mathbf{K}_U$ is an integer right inverse of $\hat{\mathbf{G}}_U$.

For prescribed abort keys $\mathbf{T}_U \in \mathbb{Z}_q^{|U|\times dp}$ and a target $\breve{\mathbf{t}} \in \mathbb{Z}_q^{dp}$, define the particular completion

$$\breve{\mathbf{R}}_{\mathbf{t},0} := \left(\mathbf{K}_U\left(\mathbf{T}_U - (\mathbf{g})_U\breve{\mathbf{t}}^\top\right)\right) \bmod q, \tag{20}$$

and let $\mathcal{K}_U := \{\mathbf{Z} \in \mathbb{Z}_q^{(k-1)\times dp} : \hat{\mathbf{G}}_U\mathbf{Z} = \mathbf{O} \bmod q\}$. For any $\mathbf{N} \in \mathcal{K}_U$, set

$$\breve{\mathbf{R}}_{\mathbf{t}} := \left(\breve{\mathbf{R}}_{\mathbf{t},0} + \mathbf{N}\right) \bmod q, \tag{21}$$

$$\breve{\mathbf{T}} := \left(\mathbf{g}\breve{\mathbf{t}}^\top + \hat{\mathbf{G}}\breve{\mathbf{R}}_{\mathbf{t}}\right) \bmod q. \tag{22}$$

Proof of Lemma 4. We first consider $U \neq \emptyset$. The construction above gives an efficiently computable integer matrix $\mathbf{K}_U$ satisfying $\hat{\mathbf{G}}_U\mathbf{K}_U = \mathbf{I}_{|U|}$. Therefore, by (20),

$$\begin{aligned}\hat{\mathbf{G}}_U\breve{\mathbf{R}}_{\mathbf{t},0} &= \left(\hat{\mathbf{G}}_U\mathbf{K}_U\left(\mathbf{T}_U - (\mathbf{g})_U\breve{\mathbf{t}}^\top\right)\right) \bmod q\\ &= \left(\mathbf{T}_U - (\mathbf{g})_U\breve{\mathbf{t}}^\top\right) \bmod q.\end{aligned}$$

Moreover, since $\mathbf{N} \in \mathcal{K}_U$, $\hat{\mathbf{G}}_U\mathbf{N} = \mathbf{O}_{|U|\times dp} \bmod q$. It follows from (21) and (22) that

$$\begin{aligned}\breve{\mathbf{T}}_U &= \left((\mathbf{g})_U\breve{\mathbf{t}}^\top + \hat{\mathbf{G}}_U\breve{\mathbf{R}}_{\mathbf{t}}\right) \bmod q\\ &= \left((\mathbf{g})_U\breve{\mathbf{t}}^\top + \hat{\mathbf{G}}_U\left(\breve{\mathbf{R}}_{\mathbf{t},0} + \mathbf{N}\right)\right) \bmod q\\ &= \left((\mathbf{g})_U\breve{\mathbf{t}}^\top + \mathbf{T}_U - (\mathbf{g})_U\breve{\mathbf{t}}^\top\right) \bmod q\\ &= \mathbf{T}_U.\end{aligned}$$

This proves that the prescribed abort keys are preserved. Now let $T \subseteq [n]$ be any certified set, and let $\boldsymbol{\lambda}_T$ be its deterministic reconstruction vector. Since $\boldsymbol{\lambda}_T^\top\mathbf{G}_T = \mathbf{e}_1^\top$, we have $\boldsymbol{\lambda}_T^\top(\mathbf{g})_T = 1$ and $\boldsymbol{\lambda}_T^\top\hat{\mathbf{G}}_T = \mathbf{0}^\top$. Let $\breve{\mathbf{t}}_i^\top := \breve{\mathbf{T}}[i, \cdot]$. Then

$$\begin{aligned}&\mathsf{SEC.Rec}_{\mathsf{pp}}\left(T, (\breve{\mathbf{t}}_i)_{i\in T}\right)^\top = \boldsymbol{\lambda}_T^\top\breve{\mathbf{T}}_T\\ &= \boldsymbol{\lambda}_T^\top\left((\mathbf{g})_T\breve{\mathbf{t}}^\top + \hat{\mathbf{G}}_T\breve{\mathbf{R}}_{\mathbf{t}}\right) \bmod q\\ &= \left(\boldsymbol{\lambda}_T^\top(\mathbf{g})_T\right)\breve{\mathbf{t}}^\top\\ &\quad + \left(\boldsymbol{\lambda}_T^\top\hat{\mathbf{G}}_T\right)\breve{\mathbf{R}}_{\mathbf{t}} \bmod q\\ &= \breve{\mathbf{t}}^\top.\end{aligned}$$

Transposing gives $\mathsf{SEC.Rec}_{\mathsf{pp}}\left(T,(\breve{\mathbf{t}}_i)_{i\in T}\right) = \breve{\mathbf{t}} \bmod q$. If $\mathbf{T}_U$ contains the real abort keys of the parties in U, then $\breve{\mathbf{T}}_U = \mathbf{T}_U$ immediately gives $(\breve{\mathbf{t}}_i)_{i\in U} = (\mathbf{A}\mathbf{s}_i \bmod q)_{i\in U}$. For $U = \emptyset$, the prescribed-coordinate condition is empty. We take $\breve{\mathbf{R}}_{\mathbf{t},0} := \mathbf{O}$ and $\mathcal{K}_U = \mathbb{Z}_q^{(k-1)\times dp}$. The reconstruction argument above remains unchanged. □

We next show how to sample uniformly from the affine completion space in Definition 4. Since q is prime, $\mathbb{Z}_q$ is a field. For $U \neq \emptyset$, let $\mathbf{B}_U \in \mathbb{Z}_q^{(k-1)\times(k-1-|U|)}$ be a matrix whose columns form a basis of $\ker_{\mathbb{Z}_q}(\hat{\mathbf{G}}_U)$. Sample $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^{(k-1-|U|)\times dp}$ and set $\mathbf{N} := \mathbf{B}_U\mathbf{W} \bmod q$. Each column of $\mathbf{N}$ belongs to $\ker_{\mathbb{Z}_q}(\hat{\mathbf{G}}_U)$, so $\mathbf{N} \in \mathcal{K}_U$.

Because the columns of $\mathbf{B}_U$ form a basis, the map $\mathbf{W} \mapsto \mathbf{B}_U\mathbf{W}$ is a bijection from $\mathbb{Z}_q^{(k-1-|U|)\times dp}$ to $\mathcal{K}_U$. Therefore, $\mathbf{N}$ is uniformly distributed over $\mathcal{K}_U$. Since $\breve{\mathbf{R}}_{\mathbf{t}} = \breve{\mathbf{R}}_{\mathbf{t},0} + \mathbf{N} \bmod q$, translation by the fixed particular solution $\breve{\mathbf{R}}_{\mathbf{t},0}$ is a bijection from $\mathcal{K}_U$ onto the affine space $\left\{\mathbf{Z} \in \mathbb{Z}_q^{(k-1)\times dp} : \hat{\mathbf{G}}_U\mathbf{Z} = \mathbf{T}_U - (\mathbf{g})_U\breve{\mathbf{t}}^\top \bmod q\right\}$. Hence, SimAK samples uniformly from exactly the affine space specified in Definition 4.

For $U = \emptyset$, we take $\mathbf{B}_U = \mathbf{I}_{k-1}$ and sample $\mathbf{W} \xleftarrow{\$} \mathbb{Z}_q^{(k-1)\times dp}$, so that $\mathbf{N} = \mathbf{W}$ is uniform over the entire space.

The simulator uses deterministic polynomial-time procedures, with fixed tie-breaking rules, to select the unimodular minor, construct $\mathbf{K}_U$, and compute the modular kernel basis $\mathbf{B}_U$ from (pp, U). Thus, for fixed $(U, \breve{\mathbf{t}}, \mathbf{T}_U)$, the simulator's only internal randomness is the uniform matrix $\mathbf{W}$.

We conclude by recording how the preceding completion is used in Definition 5. Ignoring values that are identically distributed in both branches, the real branch contains $\left(\boldsymbol{A}, \boldsymbol{t}, (\boldsymbol{s}_i)_{i\in U_{\mathrm{raw}}}, (\boldsymbol{z}_{i,s})_{(s,i)\in\mathcal{S}}, (\boldsymbol{t}_i)_{i\in[n]}\right)$, where $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$ and $\boldsymbol{t}_i = \boldsymbol{A}\boldsymbol{s}_i$. The simulated branch contains $\left(\boldsymbol{A}, \breve{\boldsymbol{t}}, (\boldsymbol{s}_i)_{i\in U_{\mathrm{raw}}}, (\boldsymbol{z}_{i,s})_{(s,i)\in\mathcal{S}}, (\breve{\boldsymbol{t}}_i)_{i\in[n]}\right)$, where $\breve{\boldsymbol{t}}$ is uniform and $(\breve{\boldsymbol{t}}_i)_{i\in[n]}$ is generated by SimAK. By Lemma 4, the simulated tuple satisfies $(\breve{\boldsymbol{t}}_i)_{i\in U_{\mathrm{raw}}} = (\boldsymbol{A}\boldsymbol{s}_i)_{i\in U_{\mathrm{raw}}}$ and, for every certified set T, $\mathsf{SEC.Rec}_{\mathsf{pp}}\left(T, (\breve{\boldsymbol{t}}_i)_{i\in T}\right) = \breve{\boldsymbol{t}}$. Thus, the simulated branch preserves the public abort keys determined by the corrupted raw shares and remains globally consistent with the simulated verification target.

Remark 1 (Algebraic versus distributional simulation). Lemma 4 and Definition 4 establish algebraic consistency of the simulated abort-key tuple. They show that the simulation preserves the prescribed corrupted abort keys and reconstructs to the simulated global target from every certified set. These properties do not by themselves imply that the simulated tuple is distributed indistinguishably from the real tuple. The real induced masking matrix has the structured form $\mathbf{R}_{\mathbf{t}} = \mathbf{R}\mathbf{A}^\top$, whereas SimAK samples uniformly from an affine completion space. The remaining analysis therefore proves that these two views are computationally indistinguishable in the presence of the corrupted raw shares and multi-query response leakage. This is established through the RA-MHMLWE representation and reduction developed in the following subsections.

K.2. Conditional Distribution of the Hidden Coordinates

We give the conditioning calculation underlying Lemma 5. The calculation is performed for one column of the extended secret matrix $\mathbf{X}$. Since the D coefficient positions are sampled independently, the vectorized result follows by repeating the resulting Gaussian parameter matrix across the D positions.

Let $\mathbf{x} \in \mathbb{Z}^k$ denote one column of $\mathbf{X}$. Its distribution is a centered discrete Gaussian governed by the positive-definite Gaussian parameter matrix $\mathbf{\Sigma}_{\mathbf{X}} := \mathrm{diag}\left(\sigma_{\boldsymbol{s}}^2, \sigma_{\mathbf{R}}^2\mathbf{I}_{k-1}\right)$. Equivalently, up to normalization, its probability mass function has the form $\Pr[\mathbf{x}] \propto \exp\left(-\frac{1}{2}\mathbf{x}^\top\mathbf{\Sigma}_{\mathbf{X}}^{-1}\mathbf{x}\right)$.

Under the unimodular transformation $\mathbf{x}' := \mathbf{Q}_U^{-1}\mathbf{x}$, we have $\mathbf{x} = \mathbf{Q}_U\mathbf{x}'$. Therefore, the transformed discrete Gaussian is governed by

$$\mathbf{\Sigma}'_U := \mathbf{Q}_U^{-1}\mathbf{\Sigma}_{\mathbf{X}}\mathbf{Q}_U^{-\top}, \tag{23}$$

and its precision matrix is $\mathbf{\Sigma}'^{-1}_U = \mathbf{Q}_U^\top\mathbf{\Sigma}_{\mathbf{X}}^{-1}\mathbf{Q}_U$.

Partition the transformed coordinates as $\mathbf{x}' = \begin{bmatrix}\mathbf{x}'_{\mathsf{pub}}\\ \mathbf{x}'_{\mathsf{hid}}\end{bmatrix}$, where $\mathbf{x}'_{\mathsf{pub}} \in \mathbb{Z}^{|U|}$ and $\mathbf{x}'_{\mathsf{hid}} \in \mathbb{Z}^{h_U}$. Accordingly, partition the transformed Gaussian parameter matrix as

$$\mathbf{\Sigma}'_U = \begin{bmatrix}\mathbf{\Sigma}_{U,\mathsf{pub}} & \mathbf{\Sigma}_{U,\mathsf{pub},\mathsf{hid}}\\ \mathbf{\Sigma}_{U,\mathsf{hid},\mathsf{pub}} & \mathbf{\Sigma}_{U,\mathsf{hid}}\end{bmatrix}. \tag{24}$$

Define the conditional Gaussian parameter matrix $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}} := \mathbf{\Sigma}_{U,\mathsf{hid}} - \mathbf{\Sigma}_{U,\mathsf{hid},\mathsf{pub}}\mathbf{\Sigma}_{U,\mathsf{pub}}^{-1}\mathbf{\Sigma}_{U,\mathsf{pub},\mathsf{hid}}$. This is the Schur complement of the public block in $\mathbf{\Sigma}'_U$.

Lemma 14 (Conditional distribution of the hidden coordinates)**.** *Let* $\mathbf{x}' = \begin{bmatrix}\mathbf{x}'_{\mathsf{pub}}\\ \mathbf{x}'_{\mathsf{hid}}\end{bmatrix} = \mathbf{Q}_U^{-1}\mathbf{x}$ *be distributed as above. For every fixed exposed value* $\mathbf{x}'_{\mathsf{pub}} = \mathbf{a} \in \mathbb{Z}^{|U|}$*, the conditional distribution of* $\mathbf{x}'_{\mathsf{hid}}$ *is a discrete Gaussian over* $\mathbb{Z}^{h_U}$ *with conditional center* $\mu_{U,\mathsf{hid}|\mathsf{pub}}(\mathbf{a}) := \mathbf{\Sigma}_{U,\mathsf{hid},\mathsf{pub}}\mathbf{\Sigma}_{U,\mathsf{pub}}^{-1}\mathbf{a}$ *and Gaussian parameter matrix* $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}} = \mathbf{\Sigma}_{U,\mathsf{hid}} - \mathbf{\Sigma}_{U,\mathsf{hid},\mathsf{pub}}\mathbf{\Sigma}_{U,\mathsf{pub}}^{-1}\mathbf{\Sigma}_{U,\mathsf{pub},\mathsf{hid}}$*. Equivalently, its conditional precision is* $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1} = \left(\mathbf{\Sigma}'^{-1}_U\right)_{\mathsf{hid},\mathsf{hid}}$.

Proof. Write the transformed precision matrix as $\mathbf{\Sigma}'^{-1}_U = \begin{bmatrix}\mathbf{J}_{U,\mathsf{pub}} & \mathbf{J}_{U,\mathsf{pub},\mathsf{hid}}\\ \mathbf{J}_{U,\mathsf{hid},\mathsf{pub}} & \mathbf{J}_{U,\mathsf{hid}}\end{bmatrix}$. Conditioning on $\mathbf{x}'_{\mathsf{pub}} = \mathbf{a}$, the terms in the quadratic form that depend on $\mathbf{x}'_{\mathsf{hid}}$ are

$$\mathbf{x}'^{\top}_{\mathsf{hid}}\mathbf{J}_{U,\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}} + 2\mathbf{a}^\top\mathbf{J}_{U,\mathsf{pub},\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}}.$$

Completing the square gives a discrete Gaussian over $\mathbb{Z}^{h_U}$ whose conditional precision is $\mathbf{J}_{U,\mathsf{hid}} = \left(\mathbf{\Sigma}'^{-1}_U\right)_{\mathsf{hid},\mathsf{hid}}$. By the standard block-inverse identity, $\mathbf{J}_{U,\mathsf{hid}} = \left(\mathbf{\Sigma}_{U,\mathsf{hid}} - \mathbf{\Sigma}_{U,\mathsf{hid},\mathsf{pub}}\mathbf{\Sigma}_{U,\mathsf{pub}}^{-1}\mathbf{\Sigma}_{U,\mathsf{pub},\mathsf{hid}}\right)^{-1}$. Hence, $\mathbf{J}_{U,\mathsf{hid}} = \mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}$. The corresponding center is obtained by completing the square: $\mu_{U,\mathsf{hid}|\mathsf{pub}}(\mathbf{a}) = -\mathbf{J}_{U,\mathsf{hid}}^{-1}\mathbf{J}_{U,\mathsf{hid},\mathsf{pub}}\mathbf{a}$. The block-inverse

identities also give $-\mathbf{J}_{U,\mathsf{hid}}^{-1}\mathbf{J}_{U,\mathsf{hid},\mathsf{pub}} = \mathbf{\Sigma}_{U,\mathsf{hid},\mathsf{pub}}\mathbf{\Sigma}_{U,\mathsf{pub}}^{-1}$. Therefore, $\mu_{U,\mathsf{hid}|\mathsf{pub}}(\mathbf{a}) = \mathbf{\Sigma}_{U,\mathsf{hid},\mathsf{pub}}\mathbf{\Sigma}_{U,\mathsf{pub}}^{-1}\mathbf{a}$. □

Proof of Lemma 5. Partition $\mathbf{Q}_U = \begin{bmatrix}\mathbf{Q}_{U,\mathsf{pub}} & \mathbf{Q}_{U,\mathsf{hid}}\end{bmatrix}$ according to the public and hidden transformed coordinates. By (23), the transformed joint precision is ${\mathbf{\Sigma}'_U}^{-1} = \mathbf{Q}_U^\top \mathbf{\Sigma}_{\mathbf{X}}^{-1}\mathbf{Q}_U$. Its hidden principal block is therefore $\left({\mathbf{\Sigma}'_U}^{-1}\right)_{\mathsf{hid},\mathsf{hid}} = \mathbf{Q}_{U,\mathsf{hid}}^\top \mathbf{\Sigma}_{\mathbf{X}}^{-1}\mathbf{Q}_{U,\mathsf{hid}}$. By Lemma 14, this hidden principal block is exactly the conditional precision: $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1} = \mathbf{Q}_{U,\mathsf{hid}}^\top \mathbf{\Sigma}_{\mathbf{X}}^{-1}\mathbf{Q}_{U,\mathsf{hid}}$. Using submultiplicativity of the spectral norm,

$$\begin{aligned}\left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\right\|_2 &\le \left\|\mathbf{Q}_{U,\mathsf{hid}}^\top\right\|_2 \left\|\mathbf{\Sigma}_{\mathbf{X}}^{-1}\right\|_2 \left\|\mathbf{Q}_{U,\mathsf{hid}}\right\|_2 \\ &= \left\|\mathbf{\Sigma}_{\mathbf{X}}^{-1}\right\|_2 \left\|\mathbf{Q}_{U,\mathsf{hid}}\right\|_2^2.\end{aligned}$$

Since $\mathbf{Q}_{U,\mathsf{hid}}$ is a column submatrix of $\mathbf{Q}_U$, $\|\mathbf{Q}_{U,\mathsf{hid}}\|_2 \le \|\mathbf{Q}_U\|_2$. Therefore, $\left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\right\|_2 \le \left\|\mathbf{\Sigma}_{\mathbf{X}}^{-1}\right\|_2 \|\mathbf{Q}_U\|_2^2$. □

Since the columns of $\mathbf{X}$ are independently distributed according to the same one-column discrete Gaussian, the conditioning calculation applies independently to all D coefficient positions. Under the vectorization $\mathbf{x}'_{\mathsf{hid}} = \mathrm{vec}\left({\mathbf{X}'_{\mathsf{hid}}}^\top\right)$, the conditional Gaussian parameter matrix is $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}} \otimes \mathbf{I}_D$, and the corresponding conditional precision is $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1} \otimes \mathbf{I}_D$. Its spectral norm is $\left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1} \otimes \mathbf{I}_D\right\|_2 = \left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\right\|_2$. Thus, the one-column bound in Lemma 5 also controls the vectorized hidden secret used in the RA-MHMLWE representation.

K.3. Construction of the Hidden-Coordinate Transformation

We give the deterministic construction of the transformation matrix $\mathbf{Q}_U$ used to separate the coordinates exposed by the corrupted raw-share view from the remaining hidden coordinates. For every $U \subseteq [n]$ with $1 \le |U| \le k_p$, the construction produces $\mathbf{Q}_U \in GL_k(\mathbb{Z})$ such that $\mathbf{G}_U\mathbf{Q}_U = \begin{bmatrix}\mathbf{I}_{|U|} & \mathbf{O}_{|U|\times h_U}\end{bmatrix}$, where $h_U := k - |U|$. It also establishes the pattern-dependent estimate in (5) and the uniform estimate in (6). For $U = \emptyset$, we retain the convention $\mathbf{Q}_\emptyset := \mathbf{I}_k$.

Fix a non-empty admissible set $U \subseteq [n]$, and write $r := |U|$. We use the unit-valued unique-neighbor property established in the proof of [46, Lemma 4] to construct an $r \times r$ unimodular submatrix of $\hat{\mathbf{G}}_U$ by peeling. The deterministic procedure is given in [46, Algorithm 3].

By the unit-valued unique-neighbor property, the candidate set C in [46, Algorithm 3] is non-empty at every iteration. Moreover, once a column j_t and its unique active row i_t are selected, the column j_t has no nonzero entry among the remaining active rows. Hence, the selected columns are distinct.

Under the peeling row order $I_U = (i_1, \dots, i_r)$, the matrix $\mathbf{M}_U^\triangle := \hat{\mathbf{G}}_U[I_U, J_U]$ is triangular with diagonal entries in $\{-1, 1\}$. Consequently, $\left|\det\left(\mathbf{M}_U^\triangle\right)\right| = 1$. Since $\mathbf{M}_U$ differs from $\mathbf{M}_U^\triangle$ only by a row permutation, $|\det(\mathbf{M}_U)| = 1$. Thus, $\mathbf{M}_U \in GL_r(\mathbb{Z})$. The deterministic choice $j_t := \min C$ and the fixed ordering conventions make $(\mathbf{M}_U, J_U, I_U)$ uniquely determined by (pp, U).

Let $\mathbf{P}_U \in GL_k(\mathbb{Z})$ be the permutation matrix that moves the columns of $\mathbf{G}_U$ corresponding to J_U to its first r positions, in the peeling order, and places all remaining columns afterward in their original relative order. Since the selected columns belong to $\hat{\mathbf{G}}_U$, we may write $\mathbf{G}_U\mathbf{P}_U = \begin{bmatrix}\mathbf{M}_U & \mathbf{N}_U\end{bmatrix}$, where $\mathbf{N}_U \in \mathbb{Z}^{r\times h_U}$. Define $\mathbf{V}_U := \begin{bmatrix}\mathbf{M}_U^{-1} & -\mathbf{M}_U^{-1}\mathbf{N}_U \\ \mathbf{O}_{h_U\times r} & \mathbf{I}_{h_U}\end{bmatrix}$ and $\mathbf{Q}_U := \mathbf{P}_U\mathbf{V}_U$. Since $\mathbf{M}_U$ is unimodular, $\mathbf{M}_U^{-1}$ has integer entries, and $\det(\mathbf{V}_U) = \det(\mathbf{M}_U^{-1}) \in \{-1, 1\}$. Therefore, $\mathbf{V}_U, \mathbf{Q}_U \in GL_k(\mathbb{Z})$. Furthermore, $\mathbf{G}_U\mathbf{Q}_U = \mathbf{G}_U\mathbf{P}_U\mathbf{V}_U = \begin{bmatrix}\mathbf{M}_U & \mathbf{N}_U\end{bmatrix}\begin{bmatrix}\mathbf{M}_U^{-1} & -\mathbf{M}_U^{-1}\mathbf{N}_U \\ \mathbf{O}_{h_U\times r} & \mathbf{I}_{h_U}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_r & \mathbf{O}_{r\times h_U}\end{bmatrix}$. Thus, this construction gives the deterministic transformation used in the hidden-coordinate analysis.

We now prove (5). Since $\mathbf{P}_U$ is a permutation matrix, $\|\mathbf{Q}_U\|_2 = \|\mathbf{V}_U\|_2$. Write $\mathbf{V}_U = \begin{bmatrix}\mathbf{M}_U^{-1} & -\mathbf{M}_U^{-1}\mathbf{N}_U \\ \mathbf{O} & \mathbf{O}\end{bmatrix} + \begin{bmatrix}\mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{I}_{h_U}\end{bmatrix}$. The triangle inequality and submultiplicativity of the spectral norm give

$$\begin{aligned}\|\mathbf{Q}_U\|_2 &\le 1 + \left\|\mathbf{M}_U^{-1}\begin{bmatrix}\mathbf{I}_r & -\mathbf{N}_U\end{bmatrix}\right\|_2 \\ &\le 1 + \|\mathbf{M}_U^{-1}\|_2 \left\|\begin{bmatrix}\mathbf{I}_r & -\mathbf{N}_U\end{bmatrix}\right\|_2 \\ &= 1 + \frac{\sqrt{1 + \|\mathbf{N}_U\|_2^2}}{\sigma_{\min}(\mathbf{M}_U)}.\end{aligned}$$

The last equality follows from $\left\|\begin{bmatrix}\mathbf{I}_r & -\mathbf{N}_U\end{bmatrix}\right\|_2^2 = 1 + \|\mathbf{N}_U\|_2^2$. This proves (5).

Recall the quantities $\delta_{k_p} := \max_{\substack{U\subseteq[n]\\ 1\le|U|\le k_p}} \|\mathbf{G}_U\|_2$ and $\gamma_{k_p} := \min_{\substack{U\subseteq[n]\\ 1\le|U|\le k_p}} \sigma_{\min}(\mathbf{M}_U)$, where $\mathbf{M}_U$ is the deterministic unimodular submatrix returned by [46, Algorithm 3]. Since every $\mathbf{M}_U$ is non-singular and the family of admissible sets is finite, $\gamma_{k_p} > 0$. The matrix $\mathbf{N}_U$ is a column submatrix of $\mathbf{G}_U\mathbf{P}_U$. Therefore, $\|\mathbf{N}_U\|_2 \le \|\mathbf{G}_U\mathbf{P}_U\|_2 = \|\mathbf{G}_U\|_2 \le \delta_{k_p}$. Substituting this estimate into (5) gives $\|\mathbf{Q}_U\|_2 \le 1 + \frac{\sqrt{1+\delta_{k_p}^2}}{\gamma_{k_p}}$ for every non-empty admissible set U. Recall that $\beta_{\mathbf{Q}}^* := \max_{\substack{U\subseteq[n]\\ |U|\le k_p}} \|\mathbf{Q}_U\|_2$. Since $\|\mathbf{Q}_\emptyset\|_2 = 1$, we obtain $\beta_{\mathbf{Q}}^* \le 1 + \frac{\sqrt{1+\delta_{k_p}^2}}{\gamma_{k_p}}$, which proves (6).

For completeness, one may eliminate γ_{k_p} at the cost of a looser estimate. Define $\beta_{k_p} := \max_{\substack{U\subseteq[n]\\ 1\le|U|\le k_p}} \|\hat{\mathbf{G}}_U\|_2$. Fix a non-empty admissible set U. Since $\mathbf{M}_U$ is unimodular, $1 = |\det(\mathbf{M}_U)| = \prod_{j=1}^{|U|} \sigma_j(\mathbf{M}_U)$. Moreover, because $\mathbf{M}_U$ is a column submatrix of $\hat{\mathbf{G}}_U$, $\sigma_j(\mathbf{M}_U) \le \|\mathbf{M}_U\|_2 \le \|\hat{\mathbf{G}}_U\|_2 \le \beta_{k_p}$. The unimodularity of $\mathbf{M}_U$ also implies $\|\mathbf{M}_U\|_2 \ge 1$, and hence $\beta_{k_p} \ge 1$. Therefore, $\sigma_{\min}(\mathbf{M}_U) \ge \beta_{k_p}^{-(|U|-1)} \ge$

$\beta_{k_p}^{-(k_p-1)}$. Consequently, $\gamma_{k_p} \geq \beta_{k_p}^{-(k_p-1)}$, and hence $\beta_{\mathbf{Q}} \leq 1 + \beta_{k_p}^{k_p-1}\sqrt{1+\delta_{k_p}^2}$.

K.4. Precision Analysis

We give the detailed calculation underlying (7) and (8). Recall that the centered multi-query response view is $\widetilde{\mathbf{z}}_{\mathcal{S}} = \mathbf{M}_{\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}} + \mathbf{r}_{\mathcal{S}}$, where $\mathbf{M}_{\mathsf{hid}} = (\mathbf{M}_{i,s})_{(s,i)\in\mathcal{S}} \in \mathbb{Z}^{|\mathcal{S}|D\times h_U D}$ is formed by block-row stacking, and $\mathbf{M}_{i,s} := \mathbf{L}_s\left(\mathbf{G}'_{\mathsf{hid}}[i,\cdot]\otimes\mathbf{I}_D\right) \in \mathbb{Z}^{D\times h_U D}$.

The response masks $(\mathbf{r}_{i,s})_{(s,i)\in\mathcal{S}}$ are mutually independent, and each is governed by the Gaussian parameter matrix $\sigma_{\boldsymbol{r}}^2\mathbf{I}_D$. Therefore, the stacked perturbation vector $\mathbf{r}_{\mathcal{S}} := (\mathbf{r}_{i,s})_{(s,i)\in\mathcal{S}}$ is governed by $\mathbf{\Sigma}_{\mathbf{r}_{\mathcal{S}}} = \sigma_{\boldsymbol{r}}^2\mathbf{I}_{|\mathcal{S}|D}$, and its precision matrix is $\mathbf{\Sigma}_{\mathbf{r}_{\mathcal{S}}}^{-1} = \frac{1}{\sigma_{\boldsymbol{r}}^2}\mathbf{I}_{|\mathcal{S}|D}$.

Fix $(s,i)\in\mathcal{S}$. By the definition of $\mathbf{M}_{i,s}$,

$$\begin{aligned}
&\mathbf{M}_{i,s}^\top\mathbf{M}_{i,s}\\
&= \left(\mathbf{G}'_{\mathsf{hid}}[i,\cdot]\otimes\mathbf{I}_D\right)^\top\mathbf{L}_s^\top\mathbf{L}_s\left(\mathbf{G}'_{\mathsf{hid}}[i,\cdot]\otimes\mathbf{I}_D\right)\\
&= \left(\mathbf{G}'_{\mathsf{hid}}[i,\cdot]^\top\otimes\mathbf{I}_D\right)\left(\mathbf{I}_1\otimes\mathbf{L}_s^\top\mathbf{L}_s\right)\\
&\quad\cdot\left(\mathbf{G}'_{\mathsf{hid}}[i,\cdot]\otimes\mathbf{I}_D\right)\\
&= \left(\mathbf{G}'_{\mathsf{hid}}[i,\cdot]^\top\mathbf{G}'_{\mathsf{hid}}[i,\cdot]\right)\otimes\left(\mathbf{L}_s^\top\mathbf{L}_s\right).
\end{aligned}$$

The final equality follows from the mixed-product rule for Kronecker products.

Because $\mathbf{M}_{\mathsf{hid}}$ is obtained by vertically stacking the matrices $(\mathbf{M}_{i,s})_{(s,i)\in\mathcal{S}}$, we have

$$\begin{aligned}
\mathbf{M}_{\mathsf{hid}}^\top\mathbf{M}_{\mathsf{hid}} &= \sum_{(s,i)\in\mathcal{S}}\mathbf{M}_{i,s}^\top\mathbf{M}_{i,s}\\
&= \sum_{s\in[Q_{\mathsf{Sign}}]}\left((\mathbf{G}'_{\mathsf{hid}})_{V_s}^\top(\mathbf{G}'_{\mathsf{hid}})_{V_s}\right)\otimes\left(\mathbf{L}_s^\top\mathbf{L}_s\right),
\end{aligned}$$

where $(\mathbf{G}'_{\mathsf{hid}})_{V_s}$ denotes the row restriction of $\mathbf{G}'_{\mathsf{hid}}$ to the response-visibility set V_s. Using the triangle inequality, the multiplicativity of the spectral norm under Kronecker products, and $\|\mathbf{L}_s\|_2 \leq \omega_c$, we obtain

$$\begin{aligned}
\left\|\mathbf{M}_{\mathsf{hid}}^\top\mathbf{M}_{\mathsf{hid}}\right\|_2 &\leq \sum_{s\in[Q_{\mathsf{Sign}}]}\left\|(\mathbf{G}'_{\mathsf{hid}})_{V_s}^\top(\mathbf{G}'_{\mathsf{hid}})_{V_s}\right\|_2\left\|\mathbf{L}_s^\top\mathbf{L}_s\right\|_2\\
&= \sum_{s\in[Q_{\mathsf{Sign}}]}\|(\mathbf{G}'_{\mathsf{hid}})_{V_s}\|_2^2\,\|\mathbf{L}_s\|_2^2\\
&\leq \omega_c^2\sum_{s\in[Q_{\mathsf{Sign}}]}\|(\mathbf{G}'_{\mathsf{hid}})_{V_s}\|_2^2.
\end{aligned}$$

Since $\mathbf{\Sigma}_{\mathbf{r}_{\mathcal{S}}}^{-1} = \frac{1}{\sigma_{\boldsymbol{r}}^2}\mathbf{I}_{|\mathcal{S}|D}$, it follows that

$$\left\|\mathbf{M}_{\mathsf{hid}}^\top\mathbf{\Sigma}_{\mathbf{r}_{\mathcal{S}}}^{-1}\mathbf{M}_{\mathsf{hid}}\right\|_2 \leq \frac{\omega_c^2}{\sigma_{\boldsymbol{r}}^2}\sum_{s\in[Q_{\mathsf{Sign}}]}\|(\mathbf{G}'_{\mathsf{hid}})_{V_s}\|_2^2. \tag{25}$$

We now derive the uniform estimate (7). Recall that $\mathbf{G}\mathbf{Q}_U = \left[\mathbf{G}'_{\mathsf{pub}}\ \mathbf{G}'_{\mathsf{hid}}\right]$. Since row restriction and column restriction cannot increase the spectral norm,

$$\begin{aligned}
\|(\mathbf{G}'_{\mathsf{hid}})_{V_s}\|_2 &\leq \|\mathbf{G}'_{\mathsf{hid}}\|_2 \leq \|\mathbf{G}\mathbf{Q}_U\|_2\\
&\leq \|\mathbf{G}\|_2\|\mathbf{Q}_U\|_2 \leq \|\mathbf{G}\|_2\beta_{\mathbf{Q}}.
\end{aligned}$$

Substituting this estimate into (25) gives

$$\begin{aligned}
\left\|\mathbf{M}_{\mathsf{hid}}^\top\mathbf{\Sigma}_{\mathbf{r}_{\mathcal{S}}}^{-1}\mathbf{M}_{\mathsf{hid}}\right\|_2 &\leq \frac{\omega_c^2}{\sigma_{\boldsymbol{r}}^2}\sum_{s\in[Q_{\mathsf{Sign}}]}\|\mathbf{G}\|_2^2\beta_{\mathbf{Q}}^2\\
&= \frac{\omega_c^2 Q_{\mathsf{Sign}}\|\mathbf{G}\|_2^2\beta_{\mathbf{Q}}^2}{\sigma_{\boldsymbol{r}}^2}.
\end{aligned}$$

This proves (7).

It remains to combine the response-hint precision with the conditional prior precision of the hidden transformed coordinates. By Lemma 5 and (6),

$$\left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\otimes\mathbf{I}_D\right\|_2 = \left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\right\|_2 \leq \left\|\mathbf{\Sigma}_{\mathbf{X}}^{-1}\right\|_2\beta_{\mathbf{Q}}^2.$$

Since $\mathbf{\Sigma}_{\mathbf{X}} = \operatorname{diag}\left(\sigma_{\boldsymbol{s}}^2, \sigma_{\mathbf{R}}^2\mathbf{I}_{k-1}\right)$, we have $\left\|\mathbf{\Sigma}_{\mathbf{X}}^{-1}\right\|_2 = \max\left\{\sigma_{\boldsymbol{s}}^{-2}, \sigma_{\mathbf{R}}^{-2}\right\}$. Therefore, $\left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\otimes\mathbf{I}_D\right\|_2 \leq \beta_{\mathbf{Q}}^2\max\left\{\sigma_{\boldsymbol{s}}^{-2}, \sigma_{\mathbf{R}}^{-2}\right\}$. Using the triangle inequality together with (7), we obtain

$$\begin{aligned}
&\left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\otimes\mathbf{I}_D + \mathbf{M}_{\mathsf{hid}}^\top\mathbf{\Sigma}_{\mathbf{r}_{\mathcal{S}}}^{-1}\mathbf{M}_{\mathsf{hid}}\right\|_2\\
&\leq \left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1}\otimes\mathbf{I}_D\right\|_2 + \left\|\mathbf{M}_{\mathsf{hid}}^\top\mathbf{\Sigma}_{\mathbf{r}_{\mathcal{S}}}^{-1}\mathbf{M}_{\mathsf{hid}}\right\|_2\\
&\leq \beta_{\mathbf{Q}}^2\left(\max\left\{\sigma_{\boldsymbol{s}}^{-2}, \sigma_{\mathbf{R}}^{-2}\right\} + \frac{\omega_c^2 Q_{\mathsf{Sign}}\|\mathbf{G}\|_2^2}{\sigma_{\boldsymbol{r}}^2}\right).
\end{aligned}$$

This proves (8).

The resulting bounds depend only on the public parameter bounds Q_{Sign}, ω_c, $\|\mathbf{G}\|_2$, $\beta_{\mathbf{Q}}$, $\sigma_{\boldsymbol{s}}$, $\sigma_{\mathbf{R}}$, and $\sigma_{\boldsymbol{r}}$. They are therefore uniform over every admissible corrupted set U and every admissible multi-query response-visibility pattern $(V_s)_{s\in[Q_{\mathsf{Sign}}]}$.

K.5. Proof of Lemma 6

We give the full proof of Lemma 6. The reduction proceeds through HC-Leak-AK $\to$ RA-MHMLWE$[h_U]$ $\to$ RA-MLWE$[h_U]$ $\to$ MLWE, where $U := U_{\mathrm{raw}}$, $h_U := k - |U|$, and $D := d\ell$. The first step uses the hidden-coordinate representation and the combined precision bound established in the preceding analyses. The second step follows from the posterior-resampling reduction of Lemma 1, and the final step applies Lemma 8.

Proof of Lemma 6. Fix an admissible multi-query leakage pattern $\mathcal{U} = \left(U_{\mathrm{raw}}, (V_s)_{s\in[Q_{\mathsf{Sign}}]}\right)$ and a public challenge sequence $\mathbf{c} = (c_s)_{s\in[Q_{\mathsf{Sign}}]}$. Write $U := U_{\mathrm{raw}}$, $h_U := k - |U|$, and $D := d\ell$. Fix an arbitrary value $\mathbf{S}_U$ in the support of the exposed raw-share view. By Lemma 14, conditioned on $\mathbf{X}'_{\mathsf{pub}} = \mathbf{S}_U$, the vectorized hidden secret $\mathbf{x}'_{\mathsf{hid}} = \mathrm{vec}\left(\mathbf{X}'^{\top}_{\mathsf{hid}}\right) \in \mathbb{Z}^{h_U D}$ follows a discrete Gaussian governed by the Gaussian parameter matrix $\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}\otimes\mathbf{I}_D$, with a conditional center determined by $\mathbf{S}_U$. After conditioning, this center is fixed public auxiliary information.

As shown in the detailed RA-MHMLWE representation, the centered response leakage is $\widetilde{\mathbf{z}}_{\mathcal{S}} = \mathbf{M}_{\mathsf{hid}}\mathbf{x}'_{\mathsf{hid}} + \mathbf{r}_{\mathcal{S}}$, while the real and simulated hidden evaluated targets are, respectively, $\mathbf{A}^{\langle h_U\rangle}\mathbf{x}'_{\mathsf{hid}}$ and $\mathbf{u}_{\mathsf{hid}} \xleftarrow{\$} \mathbb{Z}_q^{h_U dp}$. Hence, conditioned

on $\mathbf{S}_U$, the real-versus-simulated distinction has the form of an elliptic RA-MHMLWE$[h_U]$ instance with one stacked rectangular hint. When $\mathcal{S} = \emptyset$, there is no hint, which is the case $\nu = 0$ of Definition 1.

By (10), the combined conditional-prior and hint precision satisfies $\left\|\mathbf{\Sigma}_{U,\mathsf{hid}|\mathsf{pub}}^{-1} \otimes \mathbf{I}_D + \mathbf{M}_{\mathsf{hid}}^\top \mathbf{\Sigma}_{\mathbf{r}_\mathcal{S}}^{-1} \mathbf{M}_{\mathsf{hid}}\right\|_2 \le \frac{1}{2\sigma_{\mathrm{AK}}^2}$. Furthermore, since $h_U \le k$, the smoothing assumption $\sigma_{\mathrm{AK}} \ge \sqrt{2}\eta_\epsilon\left(\mathbb{Z}^{kD}\right)$ implies $\sigma_{\mathrm{AK}} \ge \sqrt{2}\eta_\epsilon\left(\mathbb{Z}^{h_U D}\right)$. Therefore, the hypotheses of Lemma 1 are satisfied for the realized hidden-coordinate instance.

Applying Lemma 1 with $\mathbf{S}_U$ and the conditional center as public auxiliary information replaces $\mathbf{A}^{\langle h_U\rangle}\mathbf{x}'_{\mathsf{hid}}$ with $\mathbf{u}_{\mathsf{hid}}$, while preserving the centered response leakage, with statistical loss at most 2ϵ. The conditional Gaussian parameter matrix and the combined-precision bound are independent of the realized value of $\mathbf{S}_U$. Thus, the reduction applies to the joint distribution obtained by first sampling the raw-share view and then sampling the hidden coordinates from their conditional distribution. The raw-share view is sampled exactly by drawing a fresh dealing matrix $\mathbf{X}$, whose rows are spherical, and keeping only $\mathbf{S}_U = \mathbf{G}_U\mathbf{X}$.

The hidden-target replacement induces exactly the simulated abort-key distribution. Indeed, in the real branch, $\mathbf{A}^{\langle h_U\rangle}\mathbf{x}'_{\mathsf{hid}} = \mathrm{vec}\left(\left(\mathbf{X}'_{\mathsf{hid}}\mathbf{A}^\top\right)^\top\right)$. In the uniform branch, write $\mathbf{u}_{\mathsf{hid}} = \mathrm{vec}\left(\mathbf{U}_{\mathsf{hid}}^\top\right)$, where $\mathbf{U}_{\mathsf{hid}} \xleftarrow{\$} \mathbb{Z}_q^{h_U \times dp}$. Then $\breve{\mathbf{Y}} := \mathbf{Q}_U \begin{bmatrix} \mathbf{S}_U\mathbf{A}^\top \\ \mathbf{U}_{\mathsf{hid}} \end{bmatrix}$ is uniform over $\mathcal{Y}_U\left(\mathbf{S}_U\mathbf{A}^\top\right)$. By the affine-completion analysis, extracting the first row of $\breve{\mathbf{Y}}$ and applying the public map $\mathbf{G}$ produces exactly the uniform target and completed abort-key tuple generated by SimAK. Thus, the representation introduces no additional loss. Consequently, writing RA-MLWE$_{\sigma_{\mathrm{AK}}}[h_U] :=$ RA-MLWE$_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}[h_U]$ and MLWE$_{\sigma_{\mathrm{AK}}} :=$ MLWE$_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}$, there exists a PPT adversary $\mathcal{B}$ against RA-MLWE$_{\sigma_{\mathrm{AK}}}[h_U]$ such that

$$\mathsf{Adv}^{(\mathsf{pp},\mathcal{U},\mathbf{c})}_{\mathcal{A},\mathsf{HC\text{-}Leak\text{-}AK}}(\kappa) \le \mathsf{Adv}^{\mathsf{RA\text{-}MLWE}_{\sigma_{\mathrm{AK}}}[h_U]}_{\mathcal{B}}(\kappa) + 2\epsilon. \quad (26)$$

By Lemma 8, there exists a PPT adversary $\mathcal{B}'$ against MLWE$_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}$ such that

$$\mathsf{Adv}^{\mathsf{RA\text{-}MLWE}_{\sigma_{\mathrm{AK}}}[h_U]}_{\mathcal{B}}(\kappa) \le h_U \cdot \mathsf{Adv}^{\mathsf{MLWE}_{\sigma_{\mathrm{AK}}}}_{\mathcal{B}'}(\kappa). \quad (27)$$

Combining (26) and (27) gives (11). □

Appendix L. Hybrid Experiments and Proofs for Section 6

This section gives the complete experiments used in the hybrid argument of Section 6. The main text describes the changes between consecutive hybrids and states the corresponding transition bounds. In each figure below, any oracle or game step not explicitly modified is executed as in the preceding hybrid, and "complete ... as in Hyb_j" runs only the remaining steps of that oracle in Hyb_j, without repeating earlier ones.

L.1. Hyb_1: Programming Honest Commitments

The complete modification defining Hyb_1 is given in Figure 5. In the first signing round, the challenger replaces each honest commitment hash with an independently sampled value in $\{0,1\}^{2\kappa}$ and stores that value together with the corresponding opening $\boldsymbol{w}_i$. In the second signing round, before returning the opening to the adversary, the challenger programs H_{cmt} at the associated input. The random commitment has the same distribution as a fresh output of H_{cmt}. Therefore, the transition changes the adversary's view only if the relevant oracle input was queried before the challenger programs it. This event is recorded as $\mathsf{Bad}_{\mathsf{cmt}}$. All other signing-oracle behavior, including transcript maintenance, blame handling, and the third signing round, remains unchanged from Hyb_0.

Conditioned on $\neg\mathsf{Bad}_{\mathsf{cmt}}$, the programmed random-oracle value is distributed exactly as the value that would have been returned by an honest evaluation of H_{cmt}. Moreover, programming occurs before the opening $\boldsymbol{w}_i$ is revealed, so the subsequent commitment-opening check remains unchanged. Hence, conditioned on $\neg\mathsf{Bad}_{\mathsf{cmt}}$, the adversary's view in Hyb_1 is identically distributed to its view in Hyb_0.

For completeness, the transition bound follows from the entropy of the programmed input. By Lemma 9, each value $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{r}_i$ has min-entropy at least $d-1$, except with probability 2^{-d+1} over the choice of $\boldsymbol{A}'$. Thus, for each oracle query, the probability of guessing the programmed opening is at most 2^{-d+1}. A union bound over at most Q_H oracle queries, n honest active signers per signing session, and Q_{Sign} signing sessions, together with the possible failure of the min-entropy event, gives $\Pr[\mathsf{Bad}_{\mathsf{cmt}}] \le n(Q_H+1)Q_{\mathsf{Sign}} \cdot 2^{-d+1}$. This proves the transition bound in (12).

L.2. Hyb_2: Extraction of Corrupted Openings

The complete modification defining Hyb_2 is given in Figure 6. The challenger searches the H_{cmt}-table during OSign_2 for openings of all commitments supplied by corrupted active signers. If every corrupted opening can be extracted, the challenger combines these values with the stored honest openings, reconstructs the aggregate commitment, computes the common challenge, and precomputes the honest third-round responses. The table Programmed records the extracted corrupted openings, the resulting challenge, and the precomputed honest responses. All of these values remain internal during OSign_2: the only values returned to the adversary are the honest second-round openings, exactly as in Hyb_1. The stored responses are returned only in OSign_3, after the challenger has verified that the corrupted openings agree with the extracted values. Thus, the modification advances the challenger's internal computation without advancing the adversary's view.

We justify that the preceding extraction and precomputation preserve the transcript distribution unless $\mathsf{Bad}_{\mathsf{open}}$ occurs. If every corrupted opening later supplied in OSign_3

Hyb_1: the game is identical to Hyb_0, except for the following modifications to OSign_1 and OSign_2.

$\underline{\mathsf{OSign}_1(\mathsf{sid}, \mathsf{act}, \mathsf{msg})}$

assert $\{\mathsf{sid} \notin L_{\mathsf{sid}} \wedge \mathsf{act} \subseteq [n] \wedge |\mathsf{act}| \geq t_{\mathsf{sig}} \wedge \mathcal{P}(\mathsf{tspar}, \mathsf{vk}, \mathsf{sid}, \mathsf{act}, \mathsf{msg}) = 1 \wedge (\mathsf{honest} \cap \mathsf{act} \neq \emptyset)\}$
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  assert $\{\mathsf{sid} \notin \mathsf{st}_i.\mathsf{session}\}$
  $\boldsymbol{r}_i \leftarrow \mathcal{D}^{\ell}_{\sigma_{\boldsymbol{r}}}$.
  $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{r}_i$.
  $\mathsf{cmt}_i \xleftarrow{\$} \{0,1\}^{2\kappa}$. ▷ replace cmt_i with random value
  $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}] := (\mathsf{status} = \mathsf{running}, \mathsf{rnd} = 1, \mathsf{act}, \mathsf{msg}, \mathsf{open} = (\boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i), \mathsf{aux} = \emptyset)$
  $\mathsf{pm}^1_i := \mathsf{cmt}_i$
end for
Initialize transcript table PM and blame table BL as in Figure 2.
$L_{\mathsf{sig}} := L_{\mathsf{sig}} \cup \{\mathsf{msg}\}$
$L_{\mathsf{sid}}[\mathsf{sid}] := (\mathsf{running}, 1, \mathsf{act}, \mathsf{msg}, \mathsf{PM}, \mathsf{BL})$
return $(\mathsf{pm}^1_i)_{i \in \mathsf{honest} \cap \mathsf{act}}$

$\underline{\mathsf{OSign}_2(\mathsf{sid}, (\mathsf{pm}^1_j)_{j \in \mathsf{corrupt} \cap \mathsf{act}})}$

assert $\{\mathsf{sid} \in L_{\mathsf{sid}}\}$
Fetch $(\mathsf{status}, r, \mathsf{act}, \mathsf{msg}, \mathsf{PM}, \mathsf{BL}) \leftarrow L_{\mathsf{sid}}[\mathsf{sid}]$.
if $\mathsf{status} = \mathsf{abort}$ **then**
  return $\perp$
end if
assert $\{r = 1\}$
Record $(\mathsf{pm}^1_j)_{j \in \mathsf{corrupt} \cap \mathsf{act}}$ in PM.
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  Fetch $\mathsf{open} = (\boldsymbol{r}_i, \boldsymbol{w}_i, \mathsf{cmt}_i)$ from $\mathsf{st}_i.\mathsf{session}[\mathsf{sid}]$.
  Let $x_i := (\mathsf{sid}, \mathsf{vk}, \mathsf{act}, i, \boldsymbol{w}_i)$.
  if x_i has already been queried to H_{cmt} **then**
    $\mathsf{Bad}_{\mathsf{cmt}} := 1$; abort the hybrid.
  else
    Program $H_{\mathsf{cmt}}(x_i) := \mathsf{cmt}_i$. ▷ program random oracle
  end if
end for
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  $(\mathsf{st}_i, \mathsf{pm}^2_i) \leftarrow \mathsf{LaTSEC.ShareSign}_2(\mathsf{st}_i, \mathsf{sid}, \mathsf{act}, (\mathsf{pm}^1_u)_{u \in \mathsf{act}})$.
end for
Set $\mathsf{pm}^2_i := \boldsymbol{w}_i$ for every $i \in \mathsf{honest} \cap \mathsf{act}$.
Update PM, BL, and L_{sid}, including abort handling, exactly as in Figure 2.
return $(\mathsf{pm}^2_i)_{i \in \mathsf{honest} \cap \mathsf{act}}$.

Figure 5. Hyb_1 of the unforgeability proof of our scheme; the differences from the previous hybrid are highlighted. For conciseness, we omit some routine assertions in Algorithm 4.

matches the value extracted during OSign_2, then the reconstructed commitment $\boldsymbol{w}$, the challenge c, and every honest response $\boldsymbol{z}_i = c\boldsymbol{s}_i + \boldsymbol{r}_i$ are exactly the values that would have been computed in Hyb_1. If no opening was extracted for some corrupted commitment and that commitment later fails verification, both hybrids follow the same abort behavior. Consequently, conditioned on $\neg\mathsf{Bad}_{\mathsf{open}}$, the adversary's view is identical in Hyb_1 and Hyb_2.

The bad event can occur in two ways. First, a corrupted signer may later supply a valid opening for a commitment value for which no preimage appeared in the H_{cmt}-table during OSign_2. For a fixed commitment value in $\{0,1\}^{2\kappa}$, each fresh oracle query finds such a preimage with probability $2^{-2\kappa}$. Taking a union bound over at most t_{uf} corrupted active signers, Q_{Sign} signing sessions, and Q_H oracle queries gives a probability of at most $t_{\mathsf{uf}} Q_{\mathsf{Sign}} Q_H \cdot 2^{-2\kappa}$. Second, a corrupted signer may open a commitment to a value different from the one extracted during OSign_2. This requires a collision in H_{cmt}, whose probability among at most Q_H oracle queries is at most $Q_H^2 \cdot 2^{-2\kappa}$. Taking a union bound over these events gives $\Pr[\mathsf{Bad}_{\mathsf{open}}] \leq \left(Q_H^2 + t_{\mathsf{uf}} Q_{\mathsf{Sign}} Q_H\right) 2^{-2\kappa}$. This proves the transition bound in (13).

L.3. Hyb_3: Programming the Challenge Oracle

The complete experiment defining Hyb_3 is given in Figure 7. As in Hyb_2, the challenger extracts the corrupted commitment openings during OSign_2 and combines them with the stored honest openings to reconstruct the aggregate commitment $\boldsymbol{w}$. The challenger then samples $c \xleftarrow{\$} \mathcal{C}$ and programs H_c at $(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$. If that point has al-

Hyb_2: the game is identical to Hyb_1, except that the challenger extracts corrupted openings in OSign_2 and stores precomputed honest third-round responses in Programmed. Programmed is initially empty and we write $\mathsf{Programmed}[\mathsf{sid}, \boldsymbol{w}] = \bot$ when no entry is defined.

$\mathsf{OSign}_2(\mathsf{sid}, (\mathsf{pm}_j^1)_{j \in \mathsf{corrupt} \cap \mathsf{act}})$

Run OSign_2 as in Hyb_1, up to the point after H_{cmt} has been programmed for all honest openings.
for $j \in \mathsf{corrupt} \cap \mathsf{act}$ **do**
  Search the H_{cmt}-table for $\widehat{\boldsymbol{w}}_j \in \mathcal{R}_q^p$ such that $H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, j, \widehat{\boldsymbol{w}}_j) = \mathsf{pm}_j^1$.
  if no such $\widehat{\boldsymbol{w}}_j$ exists **then**
    Complete OSign_2 exactly as in Hyb_1 and return its output.
  end if
end for
Define $\widetilde{\boldsymbol{w}}_u := \begin{cases} \boldsymbol{w}_u, & u \in \mathsf{honest} \cap \mathsf{act}, \\ \widehat{\boldsymbol{w}}_u, & u \in \mathsf{corrupt} \cap \mathsf{act}. \end{cases}$
Compute $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\widetilde{\boldsymbol{w}}_u)_{u \in \mathsf{act}})$.
Set $c := H_c(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$.
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$
end for
$\mathsf{Programmed}[\mathsf{sid}, \boldsymbol{w}] := ((\widehat{\boldsymbol{w}}_j)_{j \in \mathsf{corrupt} \cap \mathsf{act}}, c, (\boldsymbol{z}_i)_{i \in \mathsf{honest} \cap \mathsf{act}})$.
Complete OSign_2 exactly as in Hyb_1 and return its output.

$\mathsf{OSign}_3(\mathsf{sid}, (\mathsf{pm}_j^2)_{j \in \mathsf{corrupt} \cap \mathsf{act}})$

Let $(\mathsf{pm}_u^2)_{u \in \mathsf{act}}$ be obtained from the honest stored openings and the corrupted openings provided to OSign_3.
Verify the supplied openings and update the transcript, blame table, and abort status as in $\mathsf{Game}_{\mathsf{TS\text{-}UF}}$.
if $\exists j \in \mathsf{corrupt} \cap \mathsf{act}$ such that $H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, j, \mathsf{pm}_j^2) \neq \mathsf{pm}_j^1$ **then**
  Follow the ordinary abort behavior of $\mathsf{Game}_{\mathsf{TS\text{-}UF}}$ and return.
end if
Form $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\mathsf{pm}_u^2)_{u \in \mathsf{act}})$.
if $\mathsf{Programmed}[\mathsf{sid}, \boldsymbol{w}] = \bot$ **then**
  $\mathsf{Bad}_{\mathsf{open}} := 1$; abort the hybrid.
end if
Parse $\mathsf{Programmed}[\mathsf{sid}, \boldsymbol{w}] = ((\widehat{\boldsymbol{w}}_j)_{j \in \mathsf{corrupt} \cap \mathsf{act}}, c, (\boldsymbol{z}_i)_{i \in \mathsf{honest} \cap \mathsf{act}})$.
if $(\mathsf{pm}_j^2)_{j \in \mathsf{corrupt} \cap \mathsf{act}} \neq (\widehat{\boldsymbol{w}}_j)_{j \in \mathsf{corrupt} \cap \mathsf{act}}$ **then**
  $\mathsf{Bad}_{\mathsf{open}} := 1$; abort the hybrid.
end if
return $(\boldsymbol{z}_i)_{i \in \mathsf{honest} \cap \mathsf{act}}$, updating the transcript exactly as in $\mathsf{Game}_{\mathsf{TS\text{-}UF}}$.

Figure 6. Hyb_2 of the unforgeability proof. The challenger extracts corrupted openings in OSign_2 and stores precomputed honest third-round responses in Programmed; the differences from Hyb_1 are highlighted.

ready been queried, the challenger records Bad_c and aborts the hybrid. Otherwise, the challenger computes the honest responses $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$ and stores them in Programmed as in Hyb_2. This modification remains internal during OSign_2: only the honest second-round openings are returned at that point, while the stored responses are returned later by OSign_3. The latter oracle is therefore unchanged from Hyb_2.

Conditioned on $\neg\mathsf{Bad}_c$, the programmed value has the same distribution as a fresh evaluation of H_c, and hence the adversary's view is identical in Hyb_2 and Hyb_3. The min-entropy analysis bounding $\Pr[\mathsf{Bad}_c]$ is given in Appendix E, and the resulting transition bound is stated in (14).

L.4. Hyb_4: Rewriting the Honest Openings

The complete experiment defining Hyb_4 is given in Figure 8. The only modification from Hyb_3 is the highlighted rewriting of each honest second-round opening. All other oracle and game steps are executed as in Hyb_3.

L.5. Hyb_5: Simulation of the Public Abort-Key View

The complete experiment defining Hyb_5 is given in Figure 9. At initialization, the challenger replaces the real global verification key and public abort-key tuple by a uniform target and the corresponding completion generated by SimAK. The honest response values remain unchanged, but each honest opening is computed using the simulated abort key as $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c\breve{\boldsymbol{t}}_i$. Since these openings depend on the sampled challenge, programming H_{cmt} at the honest openings is deferred until the challenge, responses, and openings have been determined. All remaining signing-oracle bookkeeping is performed as in Hyb_4.

Hyb_3: the game is identical to Hyb_2, except that the challenge c is sampled uniformly from $\mathcal{C}$ and the random oracle H_c is programmed at $(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$. Programmed is maintained as in Hyb_2.

$\mathsf{OSign}_2(\mathsf{sid}, (\mathsf{pm}_j^1)_{j \in \mathsf{corrupt} \cap \mathsf{act}})$

1: Run OSign_2 as in Hyb_1, up to the point after H_{cmt} has been programmed for all honest openings.
2: **for** $j \in \mathsf{corrupt} \cap \mathsf{act}$ **do**
3: Search the H_{cmt}-table for $\widehat{\boldsymbol{w}}_j \in \mathcal{R}_q^p$ such that $H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, j, \widehat{\boldsymbol{w}}_j) = \mathsf{pm}_j^1$.
4: **if** no such $\widehat{\boldsymbol{w}}_j$ exists **then**
5: Complete OSign_2 exactly as in Hyb_1 and return its output.
6: **end if**
7: **end for**
8: Define $\widetilde{\boldsymbol{w}}_u := \begin{cases} \boldsymbol{w}_u, & u \in \mathsf{honest} \cap \mathsf{act}, \\ \widehat{\boldsymbol{w}}_u, & u \in \mathsf{corrupt} \cap \mathsf{act}. \end{cases}$
9: Compute $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\widetilde{\boldsymbol{w}}_u)_{u \in \mathsf{act}})$.
10: Sample $c \xleftarrow{\$} \mathcal{C}$.
11: Let $y := (\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$.
12: **if** y has already been queried to H_c **then**
13: $\mathsf{Bad}_c := 1$; abort the hybrid.
14: **else**
15: Program $H_c(y) := c$. ▷ program random oracle
16: **end if**
17: **for** $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
18: $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$
19: **end for**
20: $\mathsf{Programmed}[\mathsf{sid}, \boldsymbol{w}] := ((\widehat{\boldsymbol{w}}_j)_{j \in \mathsf{corrupt} \cap \mathsf{act}}, c, (\boldsymbol{z}_i)_{i \in \mathsf{honest} \cap \mathsf{act}})$.
21: Complete OSign_2 exactly as in Hyb_2 and return its output.

$\mathsf{OSign}_3(\mathsf{sid}, (\mathsf{pm}_j^2)_{j \in \mathsf{corrupt} \cap \mathsf{act}})$ is unchanged from Hyb_2 in Figure 6.

Figure 7. Hyb_3 of the unforgeability proof. The challenger samples c uniformly from $\mathcal{C}$ and programs H_c at the reconstructed aggregate commitment; the differences from Hyb_2 are highlighted.

By Lemma 4, the completed tuple preserves the abort keys indexed by corrupt and reconstructs to $\breve{\boldsymbol{t}}$ from every certified set. Thus, the simulated public material is algebraically consistent with both the corrupted raw-share view and the uniform global verification key. The computational indistinguishability of the real and simulated public views is established by Lemma 6, whose full proof is given in Appendix K. The resulting transition bound is stated in (17). Figure 9 specifies the hybrid, whose challenger knows $\boldsymbol{s}_i$ and $\boldsymbol{r}_i$. The reduction $\mathcal{B}_{\mathrm{AK}}$ of Section 6.1 simulates it without them: once extraction succeeds, it takes c and each $\boldsymbol{z}_i$ from the next unused leakage slot, and when extraction fails, it samples a fresh $\boldsymbol{r}_i$. Leakage slots left unused are ignored.

L.6. Reduction to SelfTargetMSIS

We give the complete simulation and extraction argument underlying Lemma 7.

Proof of Lemma 7. The adversary $\mathcal{B}_{\mathsf{STMSIS}}$ receives $\boldsymbol{A}_{\mathsf{ST}} \xleftarrow{\$} \mathcal{R}_q^{p \times (\ell - p + 1)}$ from the SelfTargetMSIS challenger. It parses $\boldsymbol{A}_{\mathsf{ST}} = \left[\boldsymbol{A}' \ \boldsymbol{a}\right]$, where $\boldsymbol{A}' \in \mathcal{R}_q^{p \times (\ell - p)}$ and $\boldsymbol{a} \in \mathcal{R}_q^p$, and sets $\boldsymbol{A} := \left[\boldsymbol{I}_p \ \boldsymbol{A}'\right]$ and $\breve{\boldsymbol{t}} := -\boldsymbol{a}$. Thus, $\boldsymbol{A}_{\mathsf{ST}} = \left[\boldsymbol{A}' \ -\breve{\boldsymbol{t}}\right]$. Since $\boldsymbol{A}_{\mathsf{ST}}$ is uniform, the pair $(\boldsymbol{A}', \breve{\boldsymbol{t}})$ consists of mutually independent uniform values and has exactly the distribution prescribed in Hyb_5.

The reduction samples $\boldsymbol{s} \leftarrow \mathcal{D}_{\boldsymbol{s}}$ and generates $(\boldsymbol{s}_i)_{i \in [n]} \leftarrow \mathsf{SEC.Deal}_{\mathsf{pp}}(\boldsymbol{s})$. Let $U_{\mathrm{raw}} := \mathsf{corrupt}$. For every $i \in U_{\mathrm{raw}}$, it computes $\boldsymbol{t}_i^{\mathrm{raw}} := \boldsymbol{A}\boldsymbol{s}_i$, and lets $\mathbf{T}_{U_{\mathrm{raw}}}$ be the row-stacked coefficient embedding of $(\boldsymbol{t}_i^{\mathrm{raw}})_{i \in U_{\mathrm{raw}}}$. Writing $\breve{\mathbf{t}} \in \mathbb{Z}_q^{dp}$ for the coefficient embedding of $\breve{\boldsymbol{t}}$, the reduction computes $\breve{\mathbf{T}} \leftarrow \mathsf{SimAK}_{\mathsf{pp}}\left(U_{\mathrm{raw}}, \breve{\mathbf{t}}, \mathbf{T}_{U_{\mathrm{raw}}}\right)$ and lets $(\breve{\boldsymbol{t}}_i)_{i \in [n]}$ be the corresponding ring-valued abort-key tuple. By Lemma 4, $(\breve{\boldsymbol{t}}_i)_{i \in U_{\mathrm{raw}}} = (\boldsymbol{A}\boldsymbol{s}_i)_{i \in U_{\mathrm{raw}}}$, and, for every certified set T, $\mathsf{SEC.Rec}_{\mathsf{pp}}\left(T, (\breve{\boldsymbol{t}}_i)_{i \in T}\right) = \breve{\boldsymbol{t}}$. Hence, the global verification target, raw-share view, and public abort-key tuple have exactly the initialization distribution of Hyb_5.

The reduction runs $\mathcal{A}$ with this public material and simulates the signing interaction as in Hyb_5. In particular, for each honest active signer i, it computes $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$ and $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c\breve{\boldsymbol{t}}_i$, programs H_{cmt} and H_c, maintains Programmed, and performs the signing-oracle bookkeeping exactly as prescribed by Hyb_5.

To simulate H_c, the reduction records and returns all values programmed during signing. For every unprogrammed query $(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$, it defines $\mathsf{msg}_{\mathsf{ST}} := \mathsf{Encode}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg})$ for a fixed injective encoding Encode and answers with $\mathsf{G}(\boldsymbol{w}, \mathsf{msg}_{\mathsf{ST}})$. Thus, the simulated challenge oracle has the distribution required in Hyb_5.

Hyb_4: the game is identical to Hyb_3, except that the honest second-round openings are rewritten as $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i$. Programmed is maintained as in Hyb_3.

$\mathsf{OSign}_2(\mathsf{sid}, (\mathsf{pm}_j^1)_{j\in\mathsf{corrupt}\cap\mathsf{act}})$

Run OSign_2 as in Hyb_1, up to the point after H_{cmt} has been programmed for all honest openings.
for $j \in \mathsf{corrupt} \cap \mathsf{act}$ **do**
  Search the H_{cmt}-table for $\widehat{\boldsymbol{w}}_j \in \mathcal{R}_q^p$ such that $H_{\mathsf{cmt}}(\mathsf{sid}, \mathsf{vk}, \mathsf{act}, j, \widehat{\boldsymbol{w}}_j) = \mathsf{pm}_j^1$.
  if no such $\widehat{\boldsymbol{w}}_j$ exists **then**
    Complete OSign_2 exactly as in Hyb_1 and return its output.
  end if
end for
Define $\widetilde{\boldsymbol{w}}_u := \begin{cases} \boldsymbol{w}_u, & u \in \mathsf{honest} \cap \mathsf{act}, \\ \widehat{\boldsymbol{w}}_u, & u \in \mathsf{corrupt} \cap \mathsf{act}. \end{cases}$
Compute $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}(\mathsf{act}, (\widetilde{\boldsymbol{w}}_u)_{u\in\mathsf{act}})$.
Sample $c \xleftarrow{\$} \mathcal{C}$.
Let $y := (\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$.
if y has already been queried to H_c **then**
  $\mathsf{Bad}_c := 1$; abort the hybrid.
else
  Program $H_c(y) := c$. ▷ program random oracle
end if
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$
  Rewrite honest opening $\boldsymbol{w}_i$ as $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i$. ▷ equal to $\boldsymbol{A}\boldsymbol{r}_i$
end for
$\mathsf{Programmed}[\mathsf{sid}, \boldsymbol{w}] := ((\widehat{\boldsymbol{w}}_j)_{j\in\mathsf{corrupt}\cap\mathsf{act}}, c, (\boldsymbol{z}_i)_{i\in\mathsf{honest}\cap\mathsf{act}})$.
Complete OSign_2 exactly as in Hyb_3 and return its output.

$\mathsf{OSign}_3(\mathsf{sid}, (\mathsf{pm}_j^2)_{j\in\mathsf{corrupt}\cap\mathsf{act}})$ is unchanged from Hyb_3 in Figure 7.

Figure 8. Hyb_4 of the unforgeability proof. The honest openings are rewritten as $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{z}_i - c\boldsymbol{t}_i$; the differences from Hyb_3 are highlighted.

Suppose that $\mathcal{A}$ outputs an accepting forgery $(\mathsf{sid}^*, \mathsf{act}^*, \mathsf{msg}^*, c^*, \boldsymbol{z}^*)$. Define $\mathsf{msg}_{\mathsf{ST}}^* := \mathsf{Encode}(\mathsf{sid}^*, \mathsf{vk}, \mathsf{act}^*, \mathsf{msg}^*)$ and $\boldsymbol{z}_{\mathsf{ST}}^* := \begin{bmatrix} \boldsymbol{z}^* \\ c^* \end{bmatrix}$. Verification gives $c^* = H_c(\mathsf{sid}^*, \mathsf{vk}, \mathsf{act}^*, \mathsf{msg}^*, \boldsymbol{w}^*)$, where $\boldsymbol{w}^* = \boldsymbol{A}\boldsymbol{z}^* - c^*\check{\boldsymbol{t}}$. By the relation in Section 6.2, $\boldsymbol{w}^* = \begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{A}_{\mathsf{ST}} \end{bmatrix} \boldsymbol{z}_{\mathsf{ST}}^*$.

By freshness, the forgery point is not one of the signing-query points at which H_c was programmed. Its value is therefore obtained through G, and hence $\mathsf{G}\left(\begin{bmatrix} \boldsymbol{I}_p & \boldsymbol{A}_{\mathsf{ST}} \end{bmatrix} \boldsymbol{z}_{\mathsf{ST}}^*, \mathsf{msg}_{\mathsf{ST}}^*\right) = c^*$.

It remains to verify the norm condition. Since $\|c^*\|_\infty = 1$ and $\|c^*\|_1 = \omega_c$, the coefficients of c^* belong to $\{-1, 0, 1\}$, with exactly ω_c nonzero coefficients. Therefore, $\|c^*\|_2^2 = \omega_c$, and $\|\boldsymbol{z}_{\mathsf{ST}}^*\|_2^2 = \|\boldsymbol{z}^*\|_2^2 + \|c^*\|_2^2 \le B_2^2 + \omega_c \le B_{\mathsf{STMSIS}}^2$. Moreover, $\boldsymbol{z}_{\mathsf{ST}}^* \neq \boldsymbol{0}$ because $\omega_c > 0$ implies $c^* \neq 0$. Thus, $0 < \|\boldsymbol{z}_{\mathsf{ST}}^*\|_2 \le B_{\mathsf{STMSIS}}$.

Consequently, $(\mathsf{msg}_{\mathsf{ST}}^*, \boldsymbol{z}_{\mathsf{ST}}^*)$ is a valid solution to $\mathsf{SelfTargetMSIS}_{p,\ell-p+1,q,d,\mathcal{C},B_{\mathsf{STMSIS}}}$. Whenever $\mathcal{A}$ wins Hyb_5, the reduction therefore solves its SelfTargetMSIS instance. This proves Lemma 7. □

L.7. Detailed Derivation of the Unforgeability Bound

Proof of Theorem 2. By definition, Hyb_0 is the original threshold-unforgeability experiment. Hence, $\Pr[\mathsf{Hyb}_0 = 1] = \mathsf{Adv}_{\mathcal{A}}^{\mathsf{TS\text{-}UF}}(\kappa)$. By (12), $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \le n(Q_H+1)Q_{\mathsf{Sign}} \cdot 2^{-d+1}$. By (13), $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| \le \left(Q_H^2 + t_{\mathsf{uf}}Q_{\mathsf{Sign}}Q_H\right) 2^{-2\kappa}$. By (14), $|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| \le (Q_H+1)Q_{\mathsf{Sign}} \cdot 2^{-d+1}$. By (15), $\Pr[\mathsf{Hyb}_3 = 1] = \Pr[\mathsf{Hyb}_4 = 1]$. By (16), $|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_5 = 1]| \le \mathsf{Adv}_{\mathcal{B}_{\mathsf{AK}}}^{\mathsf{HC\text{-}Leak\text{-}AK}}(\kappa)$. Finally, by Lemma 7, $\Pr[\mathsf{Hyb}_5 = 1] \le \mathsf{Adv}_{\mathcal{B}_{\mathsf{STMSIS}}}^{\mathsf{STMSIS}}(\kappa)$. Combining these bounds by the triangle inequality and collecting the two terms proportional to 2^{-d+1} gives (18). The running-time claims follow because each reduction invokes $\mathcal{A}$ once and performs only polynomially many additional random-oracle simulations, table operations, Gaussian samples, and algebraic computations. □

Proof of Corollary 3. By (17) and the smoothing assumption $\sigma_{\mathrm{AK}} \ge \sqrt{2}\eta_\epsilon\left(\mathbb{Z}^{kD}\right)$, there exists a PPT adversary $\mathcal{B}_{\mathsf{MLWE}}$ against $\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}$ such that $\mathsf{Adv}_{\mathcal{B}_{\mathrm{AK}}}^{\mathsf{HC\text{-}Leak\text{-}AK}}(\kappa) \le k \cdot \mathsf{Adv}_{\mathcal{B}_{\mathsf{MLWE}}}^{\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}}(\kappa) + 2\epsilon$. By Lemma 2, there exists a PPT adversary $\mathcal{B}_{\mathsf{MSIS}}$ against $\mathsf{MSIS}_{p,\ell-p+1,q,d,2B_{\mathsf{STMSIS}}}$ such that $\mathsf{Adv}_{\mathcal{B}_{\mathsf{STMSIS}}}^{\mathsf{STMSIS}}(\kappa) \le \sqrt{Q_H \cdot \mathsf{Adv}_{\mathcal{B}_{\mathsf{MSIS}}}^{\mathsf{MSIS}_{p,\ell-p+1,q,d,2B_{\mathsf{STMSIS}}}}(\kappa)} + \frac{Q_H}{|\mathcal{C}|}$. Substituting these two inequalities into (18) gives (19).

If $\mathsf{MLWE}_{p,\ell-p,q,d,\sigma_{\mathrm{AK}}}$ and $\mathsf{MSIS}_{p,\ell-p+1,q,d,2B_{\mathsf{STMSIS}}}$ are hard, then the corresponding computational advantages are negligible. Moreover, by assumption, $Q_H, Q_{\mathsf{Sign}}, n, k, t_{\mathsf{uf}}$ are

Hyb_5: the game is identical to Hyb_4, except that the real global verification key $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$ and the real public abort-key tuple $(\boldsymbol{t}_i)_{i\in[n]}$ are jointly replaced by a uniform global verification key $\breve{\boldsymbol{t}}$ and a consistent abort-key completion $(\breve{\boldsymbol{t}}_i)_{i\in[n]}$. The response values remain real, while the honest openings are defined by $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c\breve{\boldsymbol{t}}_i$. The table Programmed is maintained as in Hyb_4.

Initialization

Sample $\breve{\boldsymbol{t}} \xleftarrow{\$} \mathcal{R}_q^p$, and let $\breve{\mathbf{t}} \in \mathbb{Z}_q^{dp}$ be its coefficient embedding.
Compute $\breve{\mathbf{T}} \leftarrow \mathsf{SimAK}_{\mathsf{pp}}\left(\mathsf{corrupt}, \breve{\mathbf{t}}, \mathbf{T}_{\mathsf{corrupt}}\right)$ as in Definition 4.
Let $(\breve{\boldsymbol{t}}_i)_{i\in[n]}$ be the ring-valued tuple corresponding to the rows of $\breve{\mathbf{T}}$.
Use $\breve{\boldsymbol{t}}$ as the public global verification key and $(\breve{\boldsymbol{t}}_i)_{i\in[n]}$ as the public abort-key tuple.

$\mathsf{OSign}_2(\mathsf{sid}, (\mathsf{pm}_j^1)_{j\in\mathsf{corrupt}\cap\mathsf{act}})$

Run the common OSign_2 prefix as in Hyb_4, except that programming H_{cmt} at the honest openings is deferred.
Extract the corrupted openings from the H_{cmt}-table as in Hyb_4, obtaining $(\widehat{\boldsymbol{w}}_j)_{j\in\mathsf{corrupt}\cap\mathsf{act}}$; if extraction fails, complete OSign_2 exactly as in Hyb_1 and return its output.
Sample $c \xleftarrow{\$} \mathcal{C}$.
for $i \in \mathsf{honest} \cap \mathsf{act}$ **do**
  $\boldsymbol{z}_i := c\boldsymbol{s}_i + \boldsymbol{r}_i$
  $\boldsymbol{w}_i := \boldsymbol{A}\boldsymbol{z}_i - c\breve{\boldsymbol{t}}_i$
  Store $\boldsymbol{w}_i$ as the honest opening and set $\mathsf{pm}_i^2 := \boldsymbol{w}_i$.
  Let $x_i := (\mathsf{sid}, \mathsf{vk}, \mathsf{act}, i, \boldsymbol{w}_i)$.
  if x_i has already been queried to H_{cmt} **then**
    $\mathsf{Bad}_{\mathsf{cmt}} := 1$; abort the hybrid.
  else
    Program $H_{\mathsf{cmt}}(x_i) := \mathsf{pm}_i^1$.
  end if
end for
Define $\widetilde{\boldsymbol{w}}_u := \begin{cases} \boldsymbol{w}_u, & u \in \mathsf{honest}\cap\mathsf{act}, \\ \widehat{\boldsymbol{w}}_u, & u \in \mathsf{corrupt}\cap\mathsf{act}. \end{cases}$
Compute $\boldsymbol{w} := \mathsf{SEC.Rec}_{\mathsf{pp}}\left(\mathsf{act}, (\widetilde{\boldsymbol{w}}_u)_{u\in\mathsf{act}}\right)$.
Let $y := (\mathsf{sid}, \mathsf{vk}, \mathsf{act}, \mathsf{msg}, \boldsymbol{w})$.
if y has already been queried to H_c **then**
  $\mathsf{Bad}_c := 1$; abort the hybrid.
else
  Program $H_c(y) := c$.
end if
$\mathsf{Programmed}[\mathsf{sid}, \boldsymbol{w}] := ((\widehat{\boldsymbol{w}}_j)_{j\in\mathsf{corrupt}\cap\mathsf{act}}, c, (\boldsymbol{z}_i)_{i\in\mathsf{honest}\cap\mathsf{act}})$.
Complete OSign_2 exactly as in Hyb_4 and return its output.

$\mathsf{OSign}_3(\mathsf{sid}, (\mathsf{pm}_j^2)_{j\in\mathsf{corrupt}\cap\mathsf{act}})$ is unchanged from Hyb_4.

Figure 9. Hyb_5 of the unforgeability proof. The real global verification key and public abort-key tuple are jointly replaced by a uniform target and the corresponding algebraic completion generated by SimAK. The response values remain real, while each honest opening is defined as $\boldsymbol{w}_i = \boldsymbol{A}\boldsymbol{z}_i - c\breve{\boldsymbol{t}}_i$; the differences from Hyb_4 are highlighted.

polynomially bounded in κ, and the parameter choices ensure that 2^{-d+1}, $2^{-2\kappa}$, and $Q_H/|\mathcal{C}|$ are negligible. Thus, the right-hand side of (19) consists of negligible computational and oracle-programming terms, together with the explicit statistical loss 2ϵ. If 2ϵ is also negligible, then $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{TS\text{-}UF}}(\kappa) \le \mathsf{negl}(\kappa)$. The running-time claims follow by composing the reductions from HC-Leak-AK to repeated-$\boldsymbol{A}$ MLWE, from repeated-$\boldsymbol{A}$ MLWE to ordinary MLWE, and from SelfTargetMSIS to MSIS, together with the polynomial-time hybrid simulations. In particular, the factor 2 in $\mathsf{Time}(\mathcal{B}_{\mathsf{MSIS}})$ is inherited from the reduction in Lemma 2. □

Appendix M. Full Empirical Profiles

We provide the complete empirical profiles supporting the concrete-instantiation methodology in Section 7.1. For each selected SEC generator matrix, we evaluate the additional quantities required by the hidden-coordinate and abort-key analyses. For every $U \subseteq [n]$, let $\hat{\mathbf{G}}_U$ and $\mathbf{G}_U$ denote the corresponding row restrictions. We compute $\beta_{k_p} := \max_{\substack{U\subseteq[n]\\ 1\le|U|\le k_p}} \|\hat{\mathbf{G}}_U\|_2$, $\delta_{k_p} := \max_{\substack{U\subseteq[n]\\ 1\le|U|\le k_p}} \|\mathbf{G}_U\|_2$, whose maxima are attained at $|U| = k_p$ because a row restriction cannot increase the spectral norm, and $\gamma_{k_p} := \min_{\substack{U\subseteq[n]\\ 1\le|U|\le k_p}} \sigma_{\min}(\mathbf{M}_U)$, where $\mathbf{M}_U$ is the deterministic

TABLE 8. EXHAUSTIVELY EVALUATED MATRIX PARAMETERS OF THE SEC INSTANCES OF [46, TABLE 4]

No.	n	$k = m+1$	k_p	β_{k_p}	δ_{k_p}	γ_{k_p}	$\beta_{\mathbf{Q}}$
1	10	6	2	1.732	2.236	0.618	4.963
2	10	6	2	2.236	2.646	0.618	5.576
3	15	8	3	2.646	3.059	0.445	8.232
4	20	10	3	2.000	2.646	0.445	7.355
5	20	10	3	2.646	3.162	0.445	8.452
6	25	13	3	2.000	2.646	0.445	7.355
7	25	13	3	2.646	3.059	0.445	8.232
8	30	15	3	2.000	2.646	0.445	7.355
9	30	15	3	2.646	3.059	0.445	8.232
10	30	20	4	2.236	3.000	0.347	10.105
11	30	20	4	2.665	3.317	0.295	12.744
12	35	18	4	2.665	3.245	0.295	12.512
13	40	20	5	2.940	3.646	0.193	20.635
14	45	23	5	2.868	3.635	0.193	20.582
15	50	25	5	2.940	3.580	0.193	20.305
16	55	37	6	2.893	3.788	0.152	26.759
17	60	40	6	2.893	3.788	0.152	26.759

unimodular submatrix returned by [46, Algorithm 3]. For the concrete parameter selection, we use the explicit value $\beta_{\mathbf{Q}} := 1 + \frac{\sqrt{1+\delta_{k_p}^2}}{\gamma_{k_p}}$, which is the right-hand side of the uniform transformation bound in (6). All extrema are evaluated exhaustively over their indicated subset domains. The resulting values for the complete empirical range are reported in Table 8.

Across the evaluated instances, the overall growth of $\beta_{\mathbf{Q}}$ as k_p increases is driven primarily by the deterioration of the minimum singular value γ_{k_p}, with additional variation arising from δ_{k_p}. The concrete parameter sets use the profiles for $n = 40$ and $n = 60$, whose corresponding transformation bounds are 20.635 and 26.759, respectively.

Appendix N. Detailed Size Estimates

We give the calculations underlying the size estimates reported in Table 5. We represent the uniform block of $\boldsymbol{A}$ by a 256-bit seed and do not apply bit dropping to the verification target or public abort keys. Charging $\lceil \lg q \rceil$ bits per uniform coefficient gives $|\mathsf{vk}| = 256 + pd\lceil \lg q \rceil$, $|\mathsf{ak}_i| = pd\lceil \lg q \rceil$, and $|\mathsf{ak}| = n|\mathsf{ak}_i|$. For non-uniform components, we report ideal entropy-coded sizes. Arithmetic coding approaches the entropy of a stationary memoryless source up to a constant coding overhead [61, Theorem 2].

Let $D := d\ell$. Each party stores $\mathsf{sk}_i := \boldsymbol{s}_i \in \mathcal{R}^{\ell}$. By [46, Eq. (7)], each coefficient satisfies $\mathrm{Var}(\mathbf{s}_i[j]) \le \sigma_{\mathbf{s}}^2 + c_{\exp}\sigma_{\mathbf{R}}^2$, because $\beta_{\mathbf{g}} = 1$ and every row of $\hat{\mathbf{G}}$ has $c_{\exp}$ nonzero entries in $\{-1, 1\}$. We therefore use the effective coefficient scale $\sigma_{\mathsf{sk}_i} := \sqrt{\sigma_{\mathbf{s}}^2 + c_{\exp}\sigma_{\mathbf{R}}^2}$. Since $\sigma_{\mathbf{s}} = \sigma_{\mathbf{R}}$ in every selected instance, $\sigma_{\mathsf{sk}_i} = \sigma_{\mathbf{s}}\sqrt{1 + c_{\exp}}$. Using a variance-matched discrete Gaussian as the ideal source model, whose per-coordinate entropy is approximated by $\lg \sigma + \frac{1}{2}\lg(2\pi e)$ bits [62, Section 5.1], we estimate $|\mathsf{sk}_i| \approx D\left(\lg \sigma_{\mathsf{sk}_i} + \frac{1}{2}\lg(2\pi e)\right) \approx D\left(\lg \sigma_{\mathbf{s}} + \frac{1}{2}\lg(1 + c_{\exp}) + 2.047\right)$ bits.

For a response vector in $\mathbb{Z}^D$ with Euclidean bound B, define $\mathsf{Size}(B) := D\left(\lg \frac{B}{\sqrt{D}} + \frac{1}{2}\lg(2\pi e)\right)$. This agrees with the dominant D-linear term in the logarithmic volume of the Euclidean ball, since $\lg \mathrm{Vol}\left(\mathcal{B}_D(B)\right) = \mathsf{Size}(B) - \frac{1}{2}\lg(\pi D) + O(D^{-1})$. Writing $b_c := \left\lceil \omega_c + \lg \binom{d}{\omega_c} \right\rceil$ for the challenge encoding length, the bounds $\|\boldsymbol{z}_i\|_2 \le B_1$ and $\|\boldsymbol{z}\|_2 \le B_2$ give $|\mathsf{sig}_i| \approx b_c + \mathsf{Size}(B_1)$ and $|\mathsf{sig}| \approx b_c + \mathsf{Size}(B_2)$.

The per-party signing communication consists of a 2κ-bit commitment, an opening $\boldsymbol{w}_i \in \mathcal{R}_q^p$, and the encoded response $\boldsymbol{z}_i$. Since the common challenge is not retransmitted by each participant, $\mathsf{Comm}_i \approx 2\kappa + pd\lceil \lg q \rceil + \mathsf{Size}(B_1)$. The public abort key ak_i is long-term storage and is not included in the per-signing communication. All reported values use the underlying full-precision parameters and exclude implementation-specific entropy-coding, framing, finite-block, and byte-alignment overheads.

Appendix O. Storage Comparison with sDKG and Hermine

We compare the per-participant long-term secret-share storage $|\mathsf{sk}_i|$ and the participant-specific public verification material $|\mathsf{ak}_i|$ of LaTSEC (Table 5) with those of the replicated and ramp sDKG constructions [36] and of Hermine [26], all at $\kappa = 128$. None of these works gives parameters for $n \in \{40, 60\}$, so their values are extrapolations of the sharing structure, not security-validated parameter sets. Since LaTSEC is ramp, no single threshold t matches it; as in [46, Appendix J.2], we use the privacy-matched profile $t_{\mathsf{pm}} := k_p + 1$, which tolerates the same number of corruptions, and the quorum-matched profile $t_{\mathsf{qm}} := k_c$, which has the same minimum signing-set size, giving $(n, t) \in \{(40, 6), (40, 26), (60, 7), (60, 46)\}$. Neither is like-for-like: privacy matching lets the baseline sign with fewer parties, and quorum matching gives it a stronger corruption threshold.

Secret-share storage. The sub-share counts and the ideal entropy-coded size of one secret sub-share are those of [46, Appendix J.2]. Replicated sDKG stores $\binom{n-1}{t-1}$ sub-shares of 6.024 KiB each, using its first $\kappa = 128$ set, with $(p, \ell) = (4, 4)$ and $\sigma_{\boldsymbol{s}} = 2^{10}$. Ramp sDKG stores $\mu = \kappa = 128$ sub-shares of 7.529 KiB each, using its second set, with $(p, \ell) = (5, 5)$ and $\sigma_{\mathsf{share}} = 2^{10}$; this estimate ignores the larger dependent bucket and is optimistic. In Hermine, participant i stores $u_i(n, t)$ Vandermonde sub-shares, between $L(n, t)$ and $U(n, t)$, of at most $512 \cdot 4 \cdot (13 + \frac{1}{2}\lg 7 + 2.047)$ bits each, using its $\lceil \lg q \rceil = 48$ family with $(p, \ell) = (4, 4)$. Table 9 reports the resulting storage and its ratio to $|\mathsf{sk}_i|$ of LaTSEC, which is 13.883 KiB for $n = 40$ and 14.273 KiB for $n = 60$. For ramp sDKG, the storage is 963.768 KiB, that is, 69.421 and 67.524 times that of LaTSEC for $n = 40$ and $n = 60$, respectively.

Abort-key storage. Each sub-share in these constructions carries its own public verification value in $\mathcal{R}_q^p$, and, as for LaTSEC, we charge $pd\lceil \lg q \rceil$ bits per value without bit dropping. In sDKG, this is the auxiliary key

TABLE 9. ESTIMATED PER-PARTICIPANT LONG-TERM SECRET-SHARE STORAGE AT $\kappa = 128$, COMPARED WITH LaTSEC

			Replicated sDKG		Hermine (KiB)			
n	t	Profile	$\lvert\mathsf{sk}_i\rvert$	Ratio	Min.	Avg.	Max.	Min. ratio
40	6	Privacy	3.307 GiB	2.50×10^5	353.692	429.365	522.312	25.477
40	26	Quorum	8.665×10^4 GiB	6.54×10^9	1,949.417	2,391.942	2,932.350	140.418
60	7	Privacy	258.831 GiB	1.90×10^7	699.158	1,075.881	1,151.554	48.985
60	46	Quorum	7.639×10^7 GiB	5.61×10^{12}	4,441.709	7,192.278	7,896.371	311.197

Ratios divide by $\lvert\mathsf{sk}_i\rvert$ of LaTSEC (13.883 and 14.273 KiB for $n = 40$ and 60). The sDKG values exclude indexing and framing overheads and the seed-and-tree compression proposed in [36]; the Hermine values measure the initial sharing before any refresh.

$\boldsymbol{b}_{\mathsf{sub}} := \begin{bmatrix}\boldsymbol{A}' & \boldsymbol{I}_p\end{bmatrix}\begin{bmatrix}\boldsymbol{s}_{\mathsf{sub}} \\ \boldsymbol{e}_{\mathsf{sub}}\end{bmatrix}$, of $4 \cdot 512 \cdot 45$ bits $= 11.25$ KiB for the replicated set and $5 \cdot 512 \cdot 48$ bits $= 15$ KiB for the ramp set. In Hermine, it is the indexed partial verification key $\mathsf{vk}_i[\mathsf{idx}] := 2(\boldsymbol{A}'\boldsymbol{s}_{\mathsf{idx}} + \boldsymbol{e}_{\mathsf{idx}})$, which Hermine does not compress [26, Figure 6, line 11], of $4 \cdot 512 \cdot 48$ bits $= 12$ KiB. Participant i is associated with $\binom{n-1}{t-1}$ such values in replicated sDKG, $\mu = 128$ in ramp sDKG, and $u_i(n,t)$ in Hermine. The globally distinct collection has $\binom{n}{t-1}$ values in replicated sDKG and $\nu\mu$ in ramp sDKG; for the latter we choose the smallest bucket count $\nu := k_p + 1$ for which no set of k_p parties covers all buckets of a parallel sharing, which is favorable to sDKG. In Hermine, $\mathsf{vk}_i[\mathsf{idx}]$ depends only on idx, so one copy per distinct index suffices; the number $D(n,t)$ of distinct values satisfies $D(n,1) = 1$, $D(n,n) = n$ (Hermine's $t = n$ optimization), $D(n,t) = 0$ for $t \le 0$ or $t > n$, and, for $2 \le t \le n-1$ with $n_{\mathsf{L}} := \lfloor n/2 \rfloor$ and $n_{\mathsf{R}} := \lceil n/2 \rceil$,

$$D(n,t) = \sum_{t_{\mathsf{L}}=\max\{0,t-n_{\mathsf{R}}\}}^{\min\{t,n_{\mathsf{L}}\}} \left(D(n_{\mathsf{L}},t_{\mathsf{L}}) + D(n_{\mathsf{R}},t-t_{\mathsf{L}})\right).$$

Table 10 reports the results against $\lvert\mathsf{ak}_i\rvert$ and $\lvert\mathsf{ak}\rvert = n\lvert\mathsf{ak}_i\rvert$ of LaTSEC. For ramp sDKG, the per-participant bundle is 1920 KiB, that is, 83.934 and 80.000 times $\lvert\mathsf{ak}_i\rvert$ for $n = 40$ and $n = 60$, and the distinct collection is 11,520 and 13,440 KiB, that is, 12.590 and 9.333 times $\lvert\mathsf{ak}\rvert$.

At $\kappa = 256$, the sDKG sizes scale by $7/4$ for the replicated set and by $14/5$ (secret shares) or $21/8$ (auxiliary keys) for the ramp construction, and the Hermine sizes by 2 (secret shares, $\ell = 8$) and $7/4$ (partial verification keys, $p = 7$); the corresponding LaTSEC sizes grow by about 1.7 (Table 5). In all cases, the multiplicity of sub-shares, rather than the size of an individual sub-share, accounts for the gap: each LaTSEC participant holds exactly one share and one abort key.

Appendix P. Extended Benchmarks

This appendix gives the measurement method behind Section 8.4, the running times of all algorithms with their dispersion, their dependence on the size of the signing set, and the cost of encoding and decoding.

P.1. Method

All measurements use the Rust implementation, built with `rustc` 1.100.0 in release mode (optimization level 3) for the baseline `x86-64` target, that is, without `target-cpu=native` and without vector instructions beyond SSE2. The implementation is single-threaded. The runs use one core of an AMD Ryzen AI 9 HX 370 (a Zen 5 core, at most 5.16 GHz) under Linux 7.2, with the process pinned to that core, the `amd-pstate-epp` driver, the `powersave` governor, and the energy-performance preference `balance_performance`; frequency boosting is enabled. Each run records these settings together with its timings.

For each parameter set and signing-set size, key generation is run 11 times. Then five all-honest signing sessions are run and discarded, and signer sets are drawn uniformly at random until 100 sessions have completed for $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$, or 30 for the larger sizes. Every algorithm is timed with the monotonic clock around its call. The time of $\mathsf{ShareSign}_1$ therefore includes parsing msg and evaluating $\mathcal{P}$, and PartialVerify and Combine evaluate $\mathcal{P}$ again; computing $\boldsymbol{\lambda}_{\mathsf{act}}$ is the initiator's work and is reported separately. The parties exchange values in memory, so encoding and decoding are not part of these times; they are measured separately (Table 13). The time of a signing round is the total over the signers of a session divided by $\lvert\mathsf{act}\rvert$, and a standalone PartialVerify is timed once for every signer of every session. We report medians over the sessions (over the calls, for PartialVerify). At $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$, the interquartile range of the times of $\mathsf{ShareSign}_1$, $\mathsf{ShareSign}_3$, Combine, and Verify is at most 11% of the median, and that of PartialVerify, whose cost varies with the signer set, at most 16%; $\mathsf{ShareSign}_2$, which takes a few microseconds, varies more.

P.2. Running Times

Table 11 gives the medians at $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$. Of the 603 signer sets drawn for it, three were not certified and were refused before the first round; none was certified but outside the reconstruction bounds, and all 600 sessions signed and verified. Computing $\boldsymbol{\lambda}_{\mathsf{act}}$ with the exact integral LLL takes 0.18 ms for 26 of 40 parties and 0.57 ms for 46 of 60, and evaluating $\mathcal{P}$ on a supplied vector takes between 1 and 2.4 μs; neither depends on κ. A signer spends 7.6 ms in total in the three rounds at $n = 60$ and $\kappa = 128$, and 14.7 ms at $\kappa = 256$.

TABLE 10. ESTIMATED PARTICIPANT-SPECIFIC PUBLIC VERIFICATION MATERIAL AT $\kappa = 128$, COMPARED WITH LaTSEC

			Replicated sDKG				Hermine				
n	t	Profile	$\lvert\mathsf{aux}_i\rvert$	Ratio	Total	Ratio	Min. $\lvert\mathsf{vk}_i\rvert$ (MiB)	Ratio	$D(n,t)$	Total (MiB)	Ratio
40	6	Privacy	6.177 GiB	2.83×10^5	7.060 GiB	8.09×10^3	1.008	45.115	2,582	30.258	33.862
40	26	Quorum	158.0 TiB	7.42×10^9	421.5 TiB	4.95×10^8	5.555	248.656	16,888	197.906	221.482
60	7	Privacy	483.414 GiB	2.11×10^7	537.127 GiB	3.91×10^5	1.992	85.000	9,646	113.039	80.383
60	46	Quorum	1.393×10^5 TiB	6.23×10^{12}	5.573×10^5 TiB	4.16×10^{11}	12.656	540.000	76,744	899.344	639.533

Per-participant ratios divide by $\lvert\mathsf{ak}_i\rvert$ of LaTSEC (22.875 and 24.000 KiB for $n = 40$ and 60); total ratios divide by $\lvert\mathsf{ak}\rvert$ (915 and 1440 KiB). Replicated sDKG uses its first $\kappa = 128$ set ($p = 4$, $\lceil \lg q \rceil = 45$); Hermine uses $p = 4$ and $\lceil \lg q \rceil = 48$. Values exclude indexing and framing overheads.

TABLE 11. MEDIAN RUNNING TIMES OF LaTSEC IN MS ON ONE CORE, AT $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$

n	κ	Keygen	SS_1	SS_2	SS_3	PV	Comb	Vfy	$\boldsymbol{\lambda}_{\mathsf{act}}$
40	128	63.3	4.0	<0.01	2.3	1.0	18.1	0.76	0.18
40	192	81.6	4.3	<0.01	3.2	1.5	28.6	1.2	0.18
40	256	145	5.5	<0.01	4.1	2.2	43.5	1.7	0.18
60	128	109	3.3	<0.01	4.3	1.3	32.6	0.76	0.57
60	192	167	4.3	<0.01	6.0	2.0	52.4	1.2	0.56
60	256	220	7.1	<0.01	7.6	2.8	76.2	1.8	0.56

$\lvert\mathsf{act}\rvert = 26$ for $n = 40$ and 46 for $n = 60$. Notation as in Table 7; $\boldsymbol{\lambda}_{\mathsf{act}}$ is the initiator's computation of the reconstruction vector. Keygen covers all n parties and is the median of 11 runs.

P.3. Dependence on the Signing Set

Table 12 varies $\lvert\mathsf{act}\rvert$ from t_{sig} to n for $\kappa = 128$ and 256; the values for $\kappa = 192$ lie in between and are in the artifact. The times of $\mathsf{ShareSign}_1$ and Verify remain approximately constant over the measured sizes. $\mathsf{ShareSign}_3$ grows with it, because a signer checks the commitment of every active signer, hashing $\lvert\mathsf{act}\rvert$ openings of $pd\lceil \lg q \rceil$ bits each. Combine runs one partial verification per signer and grows accordingly, from 32.6 to 42.2 ms at $n = 60$ and $\kappa = 128$. Larger signing sets admit lighter and shorter reconstruction vectors: the average weight of $\boldsymbol{\lambda}_{\mathsf{act}}$ falls from 10.3 to 6 for $n = 40$ and from about 23 to 18 for $n = 60$, and $\lVert\boldsymbol{\lambda}_{\mathsf{act}}\rVert_2$ from 3.3 to 2.4 and from 5.6 to 4.2. The signature, whose width follows $\lVert\boldsymbol{\lambda}_{\mathsf{act}}\rVert_2$, becomes slightly shorter, by up to 0.8%. With all n parties there is a single signer set, so its vector is computed once; it has weight 6 for $n = 40$ and 18 for $n = 60$.

P.4. Encoding and Decoding

Table 13 gives the median cost, at $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$, of the encodings of Section 8.1 for the three objects that are transmitted in a session. At $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$, the median time to encode or decode a partial response is at most 0.2 ms, below 5% of the time of $\mathsf{ShareSign}_3$, which computes it, and that of an opening is at most 0.07 ms, below 2% of the time of $\mathsf{ShareSign}_1$, which computes it. A signature takes longer to encode than to decode because the encoder searches for the Golomb–Rice parameter that gives the shortest code, while the decoder only confirms it. The decoders reject an input before reading it if it is longer than any encoding of a response within the verification bound, so their work is bounded by the parameter set: a signature has at most 77,034 bytes for the largest set, 1.5% above the estimate of Table 5.

TABLE 12. MEDIAN RUNNING TIMES IN MS AND SIGNATURE SIZES FOR SIGNING SETS OF t_{sig} TO n PARTIES

n	κ	$\lvert\mathsf{act}\rvert$	wt_H	$\lVert\cdot\rVert_2$	$\boldsymbol{\lambda}_{\mathsf{act}}$	SS_3	PV	Comb	$\lvert\mathsf{sig}\rvert$
40	128	26	10.3	3.3	0.18	2.3	1.00	18.1	36.68
40	128	33	7.9	2.8	0.40	2.8	0.95	22.9	36.52
40	128	40	6.0	2.4	–	3.3	0.91	27.6	36.40
40	256	26	10.3	3.2	0.18	4.1	2.2	43.5	64.15
40	256	33	7.6	2.7	0.39	4.7	2.0	53.3	63.84
40	256	40	6.0	2.4	–	5.6	2.0	65.1	63.66
60	128	46	23.1	5.6	0.57	4.3	1.3	32.6	37.84
60	128	53	19.2	4.4	1.00	4.6	1.2	36.4	37.58
60	128	60	18.0	4.2	–	5.2	1.2	42.2	37.54
60	256	46	23.2	5.7	0.56	7.6	2.8	76.2	66.15
60	256	53	18.4	4.3	1.1	8.7	2.6	90.1	65.68
60	256	60	18.0	4.2	–	9.0	2.5	98.2	65.74

wt_H and $\lVert\cdot\rVert_2$ are the mean weight and Euclidean norm of $\boldsymbol{\lambda}_{\mathsf{act}}$; $\lvert\mathsf{sig}\rvert$ is the mean encoded signature size in KiB. 100 sessions for $\lvert\mathsf{act}\rvert = t_{\mathsf{sig}}$ and 30 otherwise.

TABLE 13. MEDIAN ENCODING AND DECODING TIMES IN MS

		$\boldsymbol{z}_i$		sig		$\boldsymbol{w}_i$	
n	κ	enc.	dec.	enc.	dec.	enc.	dec.
40	128	0.11	0.11	0.47	0.15	0.04	0.04
40	192	0.15	0.15	0.66	0.20	0.05	0.05
40	256	0.17	0.17	0.84	0.25	0.06	0.06
60	128	0.11	0.11	0.49	0.16	0.04	0.03
60	192	0.14	0.14	0.68	0.21	0.05	0.05
60	256	0.20	0.20	0.86	0.25	0.07	0.06