

“Preview Writeup”: In anticipation of a package submission to the NIST Threshold Call

Title: LaTSEC: Lattice-Based Threshold Signature from Expander Codes

Subtitle: Identifiable Abort from Everywhere-Short Secret Sharing over Integer Modules

Version: Preview Writeup 1.0 (2026-10-05)¹

Team name: LaTSEC: Lattice Threshold Signatures from Expander Codes Team

Team members: Muhammad Arzaki, Muhammed F. Esgin, Markku-Juhani O. Saarinen, Amin Sakzad, Ron Steinfeld

Abstract: The LaTSEC team plans to submit two related cryptosystems. The first is SEC, an everywhere-short secret sharing scheme over integer modules whose sparse distribution matrix is derived from a lossless bipartite expander. SEC is designed to give short shares, bounded integer reconstruction coefficients, and exactly one share per participant; its computational privacy is reduced to a matrix-hint variant of Module Learning With Errors, and hence to the standard Module Learning With Errors (MLWE) problem. The second is LaTSEC, a lattice-based threshold signature scheme with identifiable abort built on top of SEC, whose signing is a three-round protocol over an authenticated broadcast channel. Signing follows the Fiat–Shamir paradigm without rejection sampling used by Raccoon-style lattice signatures, and the reconstruction function of SEC acts as the aggregation function for commitments, responses, and public verification material. Each party publishes a share-verification key, so a combiner locally checks every partial response and attributes misbehavior to a specific party without zero-knowledge proofs. We prove identifiable abort, correctness, and threshold unforgeability under the standard Module Learning With Errors (MLWE) and Module Short Integer Solution (MSIS) assumptions.

Proposed crypto-systems:

- (I) LaTSEC: Threshold Signatures with Identifiable Abort from Expander-Code Secret Sharing (Categories S1)
- (II) SEC: Everywhere-Short Secret Sharing over Integer Modules (Categories S7)

Keywords: Threshold Cryptography; NIST Threshold Call; Threshold Signatures; Identifiable Abort; Secret Sharing; Expander Codes; Lattice-Based Cryptography; Post-Quantum Cryptography

Document License: [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

¹Revised version submitted to NIST-MPTC after review. Supersedes the preliminary version 0.1.

Preview writeup. This document is provided to NIST for online publication, to foster public awareness and support public discussion within the scope of the NIST First Call for Multi-Party Threshold Schemes [BP26]. This “preview writeup” represents a good-faith plan for a subsequent “package submission”. However, until the deadline for package submission, the team may still modify its own composition and the submission plan, including possible changes to the technical scope, and/or the used techniques or achieved results.

Team members: Muhammad Arzaki^{i1,a1}, Muhammed F. Esgin^{i2,a1}, Markku-Juhani O. Saarinen^{i3,a2}, Amin Sakzad^{i4,a1}, Ron Steinfeld^{i5,a1}

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0001-7626-6946); i2 (0000-0003-1650-3748); i3 (0000-0002-2555-235X); i4 (0000-0003-4569-3384); i5 (0000-0003-1745-4183)

Affiliations: ^{a1} Faculty of Information Technology, Monash University, Melbourne, Australia

^{a2} Faculty of Information Technology and Communication Sciences, Tampere University, Finland

Main contacts:

- **Team mailing list:** latsec-team@googlegroups.com
- **Primary technical contact person:** Muhammad Arzaki, muhammad.arzaki@monash.edu
- **Secondary contact person:** Markku-Juhani O. Saarinen, markku-juhani.saarinen@tuni.fi

Produced by humans. The team hereby confirms that the content in this preview writeup: (i) was produced by the team members, and (ii) was not produced by generative artificial intelligence (AI), with the possible exception of AI-proposed grammar improvements, minor integrated suggestions, or some well-identified and short localized portions of auxiliary content (e.g., some illustration); and (iii) was proofread by the team members.

1. Introduction

We plan to submit two related cryptosystems to the NIST Threshold Call [BP26]. The first, SEC, is an everywhere-short secret sharing scheme over integer modules, proposed in Category S7 (Gadgets). The second, LaTSEC, is a lattice-based threshold signature scheme with identifiable abort built on SEC, proposed in Category S1 (Signing). Both are specified and analyzed in our accompanying preprint [AESS26].

Motivation. In a Raccoon-style lattice signature [PKPR24], a signature is a pair (c, z) with response $z = cs + r$, and verification accepts only if z is short. In a threshold version, each party i holds a share s_i of s and contributes a partial response, and a combiner adds the partial responses using reconstruction coefficients λ_i . If s is shared with Shamir’s scheme, each share is uniform modulo q and each coefficient may be large modulo q ; a partial response is then neither short nor safe to reveal, so it cannot be checked against a norm bound on its own [PKMMP24; BCPEKNPT26]. Large coefficients cause a second, general problem: since $\|\sum_i \lambda_i z_i\| \leq \|\lambda\|_1 \max_i \|z_i\|$, even individually short responses can aggregate into a response that exceeds the verification bound. A sharing scheme whose shares and reconstruction coefficients are all short is called *everywhere-short* [PENP24; BCPEKNPT26]; it allows each partial response to be verified on its own, which gives native non-interactive identifiable abort. Prior everywhere-short secret-sharing constructions in the literature pay for this in the number of short sub-shares per participant (Table 2): with n parties and reconstruction threshold k_c , replicated sharing assigns $\binom{n-1}{k_c-1}$ of them and the random-assignment ramp sharing typically κ [PENP24], while Vandermonde sharing as used in Hermine assigns a number that grows with k_c and $\log n$ [BCPEKNPT26]. Black-box secret sharing over rings achieves small share complexity asymptotically, but does not by itself provide short lattice shares or a leakage profile compatible with MLWE-style reductions [CX20; ANP23].

Approach. SEC follows the coding-theoretic line of secret sharing [MS81; CX20] but works over a $\mathbb{Z}$ -module rather than a finite field, so that secrets, shares, and reconstruction coefficients are integer objects that compose with lattice arithmetic. Its generator matrix is $G = [g \mid \hat{G}] \in \mathbb{Z}^{n \times k}$, where the first column g carries the secret and $\hat{G}$ is a signed, permuted transpose of the bi-adjacency matrix of a lossless bipartite expander. A bipartite expander is a sparse bipartite graph in which every small set of left vertices has almost as many distinct neighbors as its number of edges allows; its bi-adjacency matrix is the parity-check matrix of the corresponding expander code. Sparsity keeps every share short, and expansion guarantees unimodular submatrices for every coalition of at most k_p parties, which rules out noise-cancellation attacks and is a precondition for computational privacy. A set T of parties reconstructs with an integer vector λ_T satisfying $\lambda_T^\top G_T = e_1^\top$. LaTSEC uses this reconstruction function of SEC, $z = \sum_{i \in T} \lambda_i z_i$ with small integer λ_i , as the aggregation function of a Raccoon-style Fiat–Shamir (FS) signing protocol [PKPR24; PKMMP24]. Each participant holds exactly one short share, and identifiable abort follows from local partial verification: each received partial response is checked independently and non-interactively against that signer’s public abort key, without zero-knowledge proofs [PKNRT25].

Fit with the Threshold Call. LaTSEC addresses featured example 1 of Section 11.1 of the call: threshold-friendly and quantum-resistant. It is a threshold scheme for a signing primitive that NIST has not standardized, so per Requirement 11.2 the Technical Specification will also specify the conventional (non-threshold) Raccoon-style scheme that serves as its algebraic core, together with at least one concrete parameter set. SEC is proposed under the third framing of Table 11 of the call, as a full-fledged cryptosystem for standalone analysis: it is a sophisticated variant of secret sharing, it has an independent security formulation and proof, and it is of independent interest to any team building lattice threshold schemes that need short shares and bounded reconstruction coefficients. Per Requirement 11.8, SEC will be thoroughly specified, implemented in the team’s core code with scripts for standalone testing, and reported separately in the experimental evaluation.

Real-World Pertinence. Threshold signing with identifiable abort is the operational requirement in custody, certificate issuance, and validator settings, where a stalled signing session must be attributed to a specific party rather than merely detected. Such deployments need not involve thousands of parties: Hermine [BCPEKNPT26], for example, targets up to 64 participants, with parameters selected for $n \leq 16$ and $t \leq 8$. LaTSEC is instantiated for $n = 40$ and $n = 60$, and the gap it targets is per-participant storage: achieving native non-interactive identifiable abort with post-quantum security while each participant holds one short signing share and is associated with one public abort key.

2. Specification

Organization. The Technical Specification will contain two chapters on cryptosystems. CS1 specifies SEC: syntax ($\text{Deal}_{\text{pp}}, \text{Rec}_{\text{pp}}$), the ramp threshold access structure with privacy threshold k_p and reconstruction threshold k_c , the construction of $\hat{G}$ from an expander, the construction of the first column g of the generator matrix G , share and reconstruction-coefficient bounds, and the leakage-resilient privacy theorem. CS2 specifies LaTSEC: the conventional Raccoon-style signature, key generation, the signing protocol, partial verification, aggregation, and the identifiable-abort and unforgeability theorems. Two components may be of independent interest to other teams: the expander-based sharing gadget itself, and the matrix-hint MLWE machinery used to prove privacy under multi-query leakage.

System Model. The setup fixes public parameters pp (the generator matrix G , ring and modulus, and Gaussian widths) and a public matrix $A \in \mathcal{R}_q^{p \times \ell}$ in systematic form; no party holds a trapdoor. The current construction assumes a trusted dealer: $\text{sk} := s$ is sampled short and shared as $(s_i)_{i \in [n]} \leftarrow \text{Deal}_{\text{pp}}(s)$; each party i is associated with the public abort key $\text{ak}_i := t_i := As_i$, and the global verification key is $\text{vk} := t := As$. The threshold profile is a *ramp* structure with two thresholds. The privacy threshold k_p is the corruption threshold: threshold unforgeability holds against at most $t_{\text{uf}} = k_p$ corrupted parties. The reconstruction threshold k_c is an empirical participation threshold: signing requires an admissible active signer set $\text{act} \subseteq [n]$, namely one

satisfying $e_1^\top \in \text{Row}_{\mathbb{Z}}(G_{\text{act}})$, and a random set of size at least k_c is admissible with estimated probability at least 0.99. Set admissibility is a public, efficiently checkable predicate. For our instantiations, $(n, k_p, k_c) = (40, 5, 26)$ and $(60, 6, 46)$: unforgeability holds against up to 5 and 6 corrupted parties, respectively, while signing uses admissible sets of at least 26 and 46 parties. This gap between k_p and k_c is the price of one short share per participant. Signing is a three-round protocol over an authenticated broadcast channel, modeled as synchronous rounds: hash commitments cmt_i , then openings of $w_i := Ar_i$, then responses $z_i := cs_i + r_i$ against the common challenge c . Aggregation applies Rec_{pp} coordinate-wise, and the combiner is untrusted: it performs only public operations, and any deviation is caught by the verifier. Identifiable abort attributes verifiably incorrect values only, namely an opening inconsistent with its commitment or a partial response failing the check against its abort key; it does not detect missing messages based on timing or network assumptions.

Security. Privacy of SEC is computational, not information theoretic. The adversary's view—raw shares of a corrupt coalition together with the linear leakage induced by signing queries—is expressed as noisy linear hints on the secret and reduced through a chain of assumptions from a matrix-hint variant of MLWE down to standard MLWE [KLSS23; ENP24], under smoothing, no-wrap, and noise-growth conditions. For LaTSEC we prove three properties:

- **Identifiable abort:** local verification of $Az_i - ct_i = w_i$ together with a partial-response norm bound attributes any invalid contribution to its sender, and honest parties are never blamed except with negligible Gaussian-tail probability.
- **Correctness:** it follows from the linearity of Rec_{pp} and its commutation with public linear maps.
- **Threshold unforgeability:** a hybrid argument replaces the real global verification key and the complete public abort-key tuple by simulated ones, while retaining the adversary's raw-share and multi-query response views, using the augmented privacy of SEC; any forgery then yields a solution to SelfTargetMSIS, and hence to MSIS by the standard reduction.

Regarding rushing adversaries, the first round fixes every opening w_j by a hash commitment before any opening is revealed, so a corrupted signer cannot choose its opening after seeing the honest ones; this commit-then-open round is the standard mechanism by which three-round protocols avoid the biasability of the aggregate commitment, and hence of the challenge, present in some two-round protocols. A corrupted signer can still withhold its messages, which our identifiable abort does not attribute. We state this as a design rationale; our preprint does not formalize non-biasability. Concrete security is estimated with the lattice estimator [APS15] against MLWE and MSIS at $\kappa \in \{128, 192, 256\}$: every retained parameter tuple is evaluated at its exact prime modulus using the full LWE attack set, including primal, BDD, dual, and hybrid attacks, under the reduction-cost model of [ADPS16].

3. Open-Source Implementation

Code Structure. The core code will consist of three modules. (i) A *parameter and construction toolkit* in SageMath, which builds expander generator matrices, computes Hermite normal forms with unimodular transformations, searches for reconstruction vectors, and evaluates the security-parameter constraints. This module exists today, is publicly available [Arz26], and produced the numbers in Section 4. (ii) A *Python reference implementation* of SEC and LaTSEC, written for clarity and intended as the byte-exact specification oracle, emitting deterministic test vectors from fixed seeds. (iii) A *performance implementation* of SEC and LaTSEC in Rust, required to reproduce the Python test vectors bit-for-bit. Modules (ii) and (iii) are planned and not yet written. The toolkit consists of Jupyter notebooks for SageMath 10.7 and depends on SageMath and the lattice estimator, together with numpy and pandas. The reference and performance implementations are planned to depend only on widely used libraries: a SHA-3/SHAKE implementation, a constant-time comparison primitive, and a secret-zeroizing primitive. Build and benchmarking will be driven by standard scripts for each language.

Code Progress and Availability. The SageMath toolkit is complete for the parameter regimes reported in Section 4 and is available in a public Git repository [Arz26]; the reference and performance implementations will be developed in the same repository.

Implementation of the Networking Model. The protocol assumes an authenticated broadcast channel (or public bulletin board) in every round, so that all honest signers derive the same aggregate commitment and hence the same challenge. On the baseline single-machine platform, we will model broadcast in process, with full serialization at every party boundary so that message sizes are measured rather than assumed. A loopback harness with configurable latency and packet loss will allow end-to-end evaluation under non-optimal network conditions.

Testing. Three testing challenges are specific to this submission. First, *reproducibility of randomized construction*: the expander generator matrix and the first column g are produced by randomized search, so the toolkit fixes seeds and ships the resulting matrices as data, and tests re-derive the published parameters from the seeds. Second, *admissibility*: because reconstruction is not available for every signer set of size at least k_c , the test suite must cover both admissible and inadmissible sets and confirm that the admissibility predicate agrees with the outcome of Rec_{pp} . Third, *malicious behavior*: negative-path tests will cover inconsistent openings, out-of-bound partial responses, replayed session identifiers, and cross-session transcript mixing, and will assert that the identified party is exactly the deviating one.

4. Experimental Performance Evaluation

Performance. Table 1 reports the selected parameters and the resulting sizes for $n \in \{40, 60\}$ at each security level, with ring degree $d = 512$, as derived in our preprint [AESS26].

Table 1: Selected LaTSEC parameters and sizes (ring degree $d = 512$; sizes in KiB). Unforgeability holds against at most k_p corrupted parties; signing uses admissible sets of size at least k_c . In every instance $\sigma_s = \sigma_R$. Secret shares and signatures are ideal entropy-coded sizes; vk and abort keys are encoded without bit dropping.

n	κ	k_p	k_c	$\lceil \lg q \rceil$	(ℓ, p)	ω_c	$\lg \sigma_s$	$\lg \sigma_r$	$ \text{sk}_i $	$ \text{ak}_i $	$ \text{vk} $	$ \text{sig} $
40	128	5	26	63	(12,6)	30	16.686	46.339	14.799	23.625	23.656	41.545
40	192	5	26	65	(16,8)	44	17.203	48.153	20.250	32.500	32.531	57.207
40	256	5	26	66	(20,10)	60	17.337	48.791	25.480	41.250	41.281	72.307
60	128	6	46	66	(12,6)	30	17.205	47.139	15.189	24.750	24.781	43.417
60	192	6	46	68	(16,8)	44	17.721	48.951	20.768	34.000	34.031	59.702
60	256	6	46	68	(20,10)	60	17.854	49.589	26.126	42.500	42.531	75.424

Our main efficiency claim is per-participant storage: SEC gives exactly one short share per participant, independent of n , k_p , and k_c . At $n = 60$ and $\kappa = 128$, each participant stores one 15.19-KiB signing share and is associated with one 24.75-KiB public abort key; Section 6 compares these sizes with other schemes. The cost of this is the ramp gap between k_p and k_c and the fact that admissibility is a property of the signer set rather than only of its size. We will additionally report the empirical fraction of admissible signer sets of each size, the observed distribution of reconstruction-coefficient norms, and wall-clock timings for dealing, signing, partial verification, aggregation, and verification.

Platform. The baseline platform of a single machine with 16 cores and 64 GB of RAM is adequate for the protocol itself: signing, partial verification, and aggregation are all linear-algebraic operations over short vectors, they parallelize across parties, and the working set is small. The demanding part is offline parameter selection: certifying expansion and computing the privacy-side and abort-key quantities require exhaustive enumeration of all coalitions of at most k_p parties, which is exponential in n , while reconstruction statistics are estimated by sampling. This computation is performed once, offline, and its output is a fixed public parameter set, so it does not affect protocol benchmarks; we will report it separately.

5. Licensing, Patent Claims, and Funding

Licensing. The SageMath toolkit is released under the GNU General Public License, version 3 (GPL-3.0), consistent with its dependency on SageMath (GPL); it uses the lattice estimator (LGPL-3.0-or-later) as an external dependency. The reference and performance implementations will be released under a permissive OSI-approved license (Apache-2.0 or MIT). The license of each bundled and external dependency will be documented in the package.

Patent Claims. To the best of the team’s current knowledge, no team member is an inventor, applicant, or assignee on a patent with claims that may cover the contents of this submission, and no entity employing or sponsoring a team member is asserting such rights. This statement will be reconfirmed at the package submission deadline.

Funding. This work is supported in part by the Indonesia Endowment Fund for Education Agency and also Australian Research Council (ARC). Concrete grant identifiers will be acknowledged in the packaged submission.

6. Comparison with Other Submissions

Several Round-1 preview writeups propose post-quantum lattice threshold signatures in Category S1 or in the ML-DSA-targeted subcategory N1.4. LaTSEC sits in the same MLWE/MSIS family, and we see it as complementary rather than competing along three axes.

Table 2: Secret sharing schemes used in lattice-based threshold signatures [AESS26]. “Shares per participant” counts the atomic sub-shares held by one participant; “Coeff.” are the reconstruction coefficients; “binary” ones lie in $\{0, 1\}$, “small” permits bounded signed integers.

Sharing scheme	Used in	Access	Shares per participant	Short	Coeff.	Privacy
Shamir [Sha79]	[PKMMPS24; ENP24; PKNRT25; GHKSS25]	threshold	1	×	large	perfect
Benaloh–Leichter + Boppana [BL90; Bop85]	[CTZ24; ZT25]	threshold	$O(\mu^{4.3} \log n)^a$	×	$\{0, \pm 1\}$	perfect ^b
Replicated [PENP24]	sDKG [PENP24]	threshold	$\binom{n-1}{k_c-1}$	✓	binary	comput.
Random-assignment ramp [PENP24]	sDKG [PENP24]	ramp ^c	typically κ	✓	binary	comput.
Vandermonde [DDB94; BCPEKNPT26]	Hermine [BCPEKNPT26]	threshold	grows with k_c and $\log n$	✓	binary	comput.
SEC (this work)	LaTSEC (this work)	ramp ^d	1	✓	small	comput.

^a $\mu := \min\{k_c, n - k_c\}$ [BCM21; CTZ24]. ^bAs instantiated in [CTZ24; ZT25], the sharing randomness is uniform over $\mathcal{R}_q$, so shares are not short and privacy is perfect; sampling short pieces instead would make shares short but privacy computational. ^cThe privacy and reconstruction thresholds hold with overwhelming probability over a public seed [PENP24]. ^dThe privacy threshold k_p holds for every coalition of at most k_p parties, given a lossless expander; the reconstruction threshold k_c is selected empirically.

Sharing Layer. Table 2 summarizes the substantive difference. Because Shamir shares are not short, Shamir-based schemes such as [PKMMPS24; PKNRT25] must mask partial responses, while prior everywhere-short layers assign several short sub-shares per participant. SEC assigns exactly one, at the price of a ramp access structure with a set-dependent admissibility predicate rather than an exact k -of- n threshold. Teams working on lattice threshold schemes may find SEC usable as a drop-in sharing gadget, which is why we propose it for standalone analysis in Category S7.

Identifiable Abort. Threshold Raccoon does not provide identifiable abort natively, and one route to adding it is an additional protocol after failed aggregation [PKNRT25]; Olingo relies on zero-knowledge proofs [GHKSS25]. Hermine and the sDKG schemes obtain non-interactive identifiable abort from everywhere-short sharing with several sub-shares per participant. LaTSEC

Table 3: Lattice-based threshold signatures with mechanisms for handling malicious signers, at $\kappa = 128$, in chronological order [AESSS26]. Sizes per participant in kB[§]; $|sk_i|$: long-term secret-share storage; $|ak_i|$: participant-specific public verification material; $Comm_i$: per-party signing communication (optimistic value in parentheses); n/r: not reported; n/a: not applicable.

Scheme	Max. n	$ sk_i $	$ ak_i $	$ sig $	$Comm_i$	Malicious	DKG
GKS [GKS24]	5	n/r	n/r	46.6 ^a	3000 ^a	proof IA ^a	✓
Pelican [ENP24]	48 ^b	n/r	n/r	12.3	26.8+56 ^t ^b	robust ^b	✓
sDKG-rep. [PENP24]	16	≥ 323.5 TiB ^c	≥ 0.63 TiB ^c	10.3	20.0	NI-IA	✓
sDKG-ramp [PENP24]	1024	≥ 963.8 ^c	≥ 1920 ^c	11.9	25.0	NI-IA	✓
TRaccoon-IA [PKNRT25]	1024	n/r	n/r	12.7	88.2+6.4 ^t (28.2)	int. IA	×
Olingo [GHKSS25]	1024	n/r	n/a ^d	11.3	852 (76)	proof IA	✓
Hermine [BCPEKNPT26]	64 ^e	≥ 699.2 ^f	≥ 2039.8 ^f	11.3 ^e	73.2 ^e	NI-IA	×
LaTSEC (ours)	60	15.19	24.75	43.42	61.68	NI-IA	×

NI-IA (non-interactive IA): each partial response is verified directly against participant-specific public material; int. IA: an additional protocol runs after failed aggregation; proof IA: non-interactive zero-knowledge proofs; robust: a valid signature is guaranteed when enough honest signers participate. sDKG-rep. and sDKG-ramp use the replicated and random-assignment ramp sharings of [PENP24]; the sharing layer of every scheme is in Table 2. [§]LaTSEC values are the $n = 60$ entries of Table 1 in KiB; prior-work values keep the decimal-kB units of their sources (a 2.4% difference). ^aAs reported in [GHKSS25, Table 1], which notes that misbehaving parties can be identified but gives no formal IA proof. ^bParameter set for $t \leq 16$, $n = 3t$ [ENP24, Table 2]; robustness tolerates $t - 1$ malicious parties among $3t$ active signers. ^cExtrapolated to $n = 60$ in [AESSS26]: replicated values are privacy-matched at $t = k_p + 1 = 7$, outside the supported range $n \leq 16$; ramp values are optimistic, with $\mu = \kappa = 128$ parallel sharings and $\nu = 7$ buckets. ^dOlingo obtains IA from zero-knowledge proofs, so no counterpart to $|ak_i|$ is defined. ^eHermine targets $n \leq 64$; its parameters are selected for $n \leq 16$, $t \leq 8$, with benchmarks up to $n = t = 64$ [BCPEKNPT26]. ^fSmallest per-participant value, privacy-matched at $t = k_p + 1 = 7$ and extrapolated to $n = 60$, outside Hermine’s parameter-selection range [AESSS26].

obtains it from one public abort key per participant and a norm check on each partial response: the check is a single linear equation, with no proof system in the trusted computing base.

Storage, Signature Size, and Round Complexity. Table 3 shows that LaTSEC is the only listed scheme combining non-interactive identifiable abort with one short share per participant. At $n = 60$, its per-participant secret-share storage is at least 46.0 and 63.5 times smaller than that of Hermine and the ramp sDKG, respectively, and its per-participant abort-key storage at least 82.4 and 77.6 times smaller, with both baselines extrapolated to $n = 60$ [AESSS26]. The aggregate signature of LaTSEC is approximately 3.5 to 4.3 times as large as those of Pelican, the sDKG schemes, TRaccoon-IA, Olingo, and Hermine, primarily because LaTSEC does not yet apply bit dropping; it is nevertheless smaller than the 46.6-kB signature of GKS and the 144.5-kB signature of the two-round line of [CTZ24; ZT25], as reported in [GHKSS25, Table 1]. LaTSEC is a three-round protocol and therefore does not compete with the two-round submissions [CTZ24; GKS24; EKT25] on latency. A useful joint question for Phase 3 would be whether the expander-code sharing layer composes with a two-round signing template, and whether the matrix-hint MLWE machinery used in our privacy proof transfers to other sharing layers.

References

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. “Post-quantum key exchange – A new hope”. In: *25th USENIX Security Symposium (USENIX Security 16)*. USENIX Association, 2016, pp. 327–343. URL: <https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/alkim>. Also at ia.cr/2015/1092.

[AESSS26] Muhammad Arzaki, Muhammed F. Esgin, Markku-Juhani O. Saarinen, Amin Sakzad, and Ron Steinfeld. *LaTSEC: Lattice-Based Threshold Signature Scheme from Expander Codes*. Preprint. October 2026. URL: <https://sites.google.com/monash.edu/arzaki-phdresearch/home>.

[ANP23] Benny Applebaum, Oded Nir, and Benny Pinkas. “How to recover a secret with $O(n)$ additions”. In: *Advances in Cryptology – CRYPTO 2023, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. Lecture Notes in Computer Science. Springer, 2023, pp. 236–262. DOI: [10.1007/978-3-031-38557-5_8](https://doi.org/10.1007/978-3-031-38557-5_8). Also at ia.cr/2023/838.

[APS15] Martin R. Albrecht, Rachel Player, and Sam Scott. “On the concrete hardness of Learning with Errors”. In: *Journal of Mathematical Cryptology* 9.3 (2015), pp. 169–203. DOI: [10.1515/jmc-2015-0016](https://doi.org/10.1515/jmc-2015-0016). Also at ia.cr/2015/046.

[Arz26] Muhammad Arzaki. *SEC-LaTSEC: SageMath Scripts for SEC and LaTSEC*. Public Git repository (Jupyter notebooks; SageMath 10.7). 2026. URL: <https://github.com/arzaki-cstheory/SEC-LaTSEC>.

[BÇM21] Marshall Ball, Alper Çakan, and Tal Malkin. “Linear threshold secret-sharing with binary reconstruction”. In: *2nd Conference on Information-Theoretic Cryptography (ITC 2021)*. Ed. by Stefano Tessaro. Vol. 199. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, 12:1–12:22. DOI: [10.4230/LIPIcs.ITC.2021.12](https://doi.org/10.4230/LIPIcs.ITC.2021.12).

[BCPEKNPT26] Giacomo Borin, Sofía Celi, Rafaël del Pino, Thomas Espitau, Shuichi Katsumata, Guilhem Niot, Thomas Prest, and Kaoru Takemure. “Hermine: An efficient lattice-based FROST-like threshold signature”. In: *Advances in Cryptology – ASIACRYPT 2026*. To appear. 2026. Also at ia.cr/2026/419.

[BL90] Josh Cohen Benaloh and Jerry Leichter. “Generalized secret sharing and monotone functions”. In: *Advances in Cryptology – CRYPTO ’88*. Ed. by Shafi Goldwasser. Vol. 403. Lecture Notes in Computer Science. Springer, 1990, pp. 27–35. DOI: [10.1007/0-387-34799-2_3](https://doi.org/10.1007/0-387-34799-2_3).

[Bop85] Ravi B. Boppana. “Amplification of probabilistic Boolean formulas”. In: *26th Annual Symposium on Foundations of Computer Science (FOCS 1985)*. IEEE Computer Society, 1985, pp. 20–29.

[BP26] Luís T. A. N. Brandão and René Peralta. *NIST first call for multi-party threshold schemes*. NIST Internal Report 8214C. National Institute of Standards and Technology, 2026. DOI: [10.6028/NIST.IR.8214C](https://doi.org/10.6028/NIST.IR.8214C).

[CTZ24] Rutchathon Chairattana-Apirom, Stefano Tessaro, and Chenzhi Zhu. “Partially non-interactive two-round lattice-based threshold signatures”. In: *Advances in Cryptology – ASIACRYPT 2024, Part IV*. Ed. by Kai-Min Chung and Yu Sasaki. Vol. 15487. Lecture Notes in Computer Science. Springer, 2024, pp. 268–302. DOI: [10.1007/978-981-96-0894-2_9](https://doi.org/10.1007/978-981-96-0894-2_9). Also at ia.cr/2024/467.

[CX20] Ronald Cramer and Chaoping Xing. “Blackbox secret sharing revisited: A coding-theoretic approach with application to expansionless near-threshold schemes”. In: *Advances in Cryptology – EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. Lecture Notes in Computer Science. Springer, 2020, pp. 499–528. DOI: [10.1007/978-3-030-45721-1_18](https://doi.org/10.1007/978-3-030-45721-1_18). Also at ia.cr/2019/1134.

[DDB94] Yvo Desmedt, Giovanni Di Crescenzo, and Mike Burmester. “Multiplicative non-abelian sharing schemes and their application to threshold cryptography”. In: *Advances in Cryptology – ASIACRYPT ’94*. Ed. by Josef Pieprzyk and Reihaneh Safavi-Naini. Vol. 917. Lecture Notes in Computer Science. Springer, 1994, pp. 19–32. DOI: [10.1007/BFb0000421](https://doi.org/10.1007/BFb0000421).

[EKT25] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. “Two-round threshold signature from algebraic one-more learning with errors”. In: *Journal of Cryptology* 38.4 (2025), p. 31. DOI: [10.1007/s00145-025-09549-2](https://doi.org/10.1007/s00145-025-09549-2). Also at ia.cr/2024/496.

[ENP24] Thomas Espitau, Guilhem Niot, and Thomas Prest. “Flood and submerge: Distributed key generation and robust threshold signature from lattices”. In: *Advances in Cryptology – CRYPTO 2024, Part VII*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14926. Lecture Notes in Computer Science. Springer, 2024, pp. 425–458. DOI: [10.1007/978-3-031-68394-7_14](https://doi.org/10.1007/978-3-031-68394-7_14). Also at ia.cr/2024/959.

[GHKSS25] Kamil Doruk Gur, Patrick Hough, Jonathan Katz, Caroline Sandsbråten, and Tjerand Silde. *Olingo: Threshold lattice signatures with DKG and identifiable abort*. Cryptology ePrint Archive, Paper 2025/1789. To appear at ACM CCS 2026. 2025. Also at ia.cr/2025/1789.

[GKS24] Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. “Two-round threshold lattice-based signatures from threshold homomorphic encryption”. In: *Post-Quantum Cryptography – PQCrypto 2024, Part II*. Ed. by Markku-Juhani O. Saarinen and Daniel Smith-Tone. Vol. 14772. Lecture Notes in Computer Science. Springer, 2024, pp. 266–300. DOI: [10.1007/978-3-031-62746-0_12](https://doi.org/10.1007/978-3-031-62746-0_12). Also at ia.cr/2023/1318.

[KLSS23] Duhyeong Kim, Dongwon Lee, Jinyeong Seo, and Yongsoo Song. “Toward practical lattice-based proof of knowledge from Hint-MLWE”. In: *Advances in Cryptology – CRYPTO 2023, Part V*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14085. Lecture Notes in Computer Science. Springer, 2023, pp. 549–580. DOI: [10.1007/978-3-031-38554-4_18](https://doi.org/10.1007/978-3-031-38554-4_18). Also at ia.cr/2023/623.

[MS81] Robert J. McEliece and Dilip V. Sarwate. “On sharing secrets and Reed–Solomon codes”. In: *Communications of the ACM* 24.9 (1981), pp. 583–584. DOI: [10.1145/358746.358762](https://doi.org/10.1145/358746.358762).

[PENP24] Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. *How to Shortly Share a Short Vector: DKG with Short Shares and Application to Lattice-Based Threshold Signatures with Identifiable Aborts*. Manuscript. 2024. URL: <https://gniots.fr/assets/papers/sdkg.pdf>.

[PKMMPS24] Rafaël del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. “Threshold Raccoon: Practical threshold signatures from standard lattice assumptions”. In: *Advances in Cryptology – EUROCRYPT 2024, Part II*. Ed. by Marc Joye and Gregor Leander. Vol. 14652. Lecture Notes in Computer Science. Springer, 2024, pp. 219–248. DOI: [10.1007/978-3-031-58723-8_8](https://doi.org/10.1007/978-3-031-58723-8_8). Also at ia.cr/2024/184.

[PKNRT25] Rafaël del Pino, Shuichi Katsumata, Guilhem Niot, Michael Reichle, and Kaoru Takemure. “Unmasking TRaccoon: A lattice-based threshold signature with an efficient identifiable abort protocol”. In: *Advances in Cryptology – CRYPTO 2025, Part VI*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16005. Lecture Notes in Computer Science. Springer, 2025, pp. 423–456. DOI: [10.1007/978-3-032-01887-8_14](https://doi.org/10.1007/978-3-032-01887-8_14). Also at ia.cr/2025/849.

[PKPR24] Rafaël del Pino, Shuichi Katsumata, Thomas Prest, and Mélissa Rossi. “Raccoon: A masking-friendly signature proven in the probing model”. In: *Advances in Cryptology – CRYPTO 2024, Part I*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14920. Lecture Notes in Computer Science. Springer, 2024, pp. 409–444. DOI: [10.1007/978-3-031-68376-3_13](https://doi.org/10.1007/978-3-031-68376-3_13). Also at ia.cr/2024/1291.

[Sha79] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613. DOI: [10.1145/359168.359176](https://doi.org/10.1145/359168.359176).

[ZT25] Chenzhi Zhu and Stefano Tessaro. “The algebraic one-more MISIS problem and applications to threshold signatures”. In: *Advances in Cryptology – CRYPTO 2025, Part I*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Vol. 16000. Lecture Notes in Computer Science. Springer, 2025, pp. 548–581. DOI: [10.1007/978-3-032-01855-7_18](https://doi.org/10.1007/978-3-032-01855-7_18). Also at ia.cr/2025/436.