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

Title: LoTRS: Practical Post-Quantum Structured Threshold Ring Signatures from Lattices

Version: Preview Writeup 0.1 (2026-05-18)[1]

Team name: LoTRS: Lattice Threshold Ring Signature Team

Team members: Muhammed F. Esgin, Nikai Jagganath, Dongxi Liu, Markku-Juhani O. Saarinen, Amin Sakzad, Ron Steinfeld

Abstract: The LoTRS team plans to submit a post-quantum threshold ring signature scheme based on standard structured-lattice assumptions (Module Learning With Errors and Module Short Integer Solution). LoTRS realizes a *structured threshold ring signature*: the public ring of size $T \cdot N$ is organized as a $T \times N$ table, and a quorum is exactly one hidden column of T keys (one per row); the T owners jointly produce a single signature. The verifier learns that some valid quorum signed but not which one. Two cryptographic layers compose: a two-round lattice-based multi-signature in the DualMS family provides threshold-style aggregation, while a lattice one-out-of-many selection-hiding proof anonymizes the chosen quorum within the public family of admissible quorums. Secret keys are not Shamir-shared; each signer holds an ordinary lattice signing key. At a 50-of-100 setting our concrete signature size is approximately 35 kilobytes. The package will include byte-exact Python and Rust reference implementations cross-validated through deterministic test vectors, a SageMath parameter estimator, and benchmarking scripts.

Proposed crypto-systems: (I) LoTRS: Structured Threshold Ring Signature from Lattices (Categories S1).

Keywords: Threshold Cryptography; Threshold Ring Signature; Post-Quantum Cryptography; Lattice-Based Cryptography; Multi-Signature; One-out-of-Many Proof; NIST Threshold Call

Document License: CC BY 4.0 International

[1]Preliminary version submitted to NIST-MPTC for review

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: Muhammed F. Esgin [i1,a1], Nikai Jagganath [i2,a1,a2], Dongxi Liu [i3,a2], Markku-Juhani O. Saarinen [i4,a3], Amin Sakzad [i5,a1], Ron Steinfeld [i6,a1]

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0003-1650-3748); i2 (0000-0002-2155-3179); i3 (0000-0002-0221-2571); i4 (0000-0002-2555-235X); i5 (0000-0003-4569-3384); i6 (0000-0003-1745-4183)

Affiliations: [a1] Faculty of Information Technology, Monash University, VIC, Australia. [a2] Data61, CSIRO, Australia. [a3] Faculty of Information Technology and Communication Sciences, Tampere University, Finland.

Main contacts:

- **Team mailing list:** lotrs-team@googlegroups.com
- **Primary technical contact person:** Nikai Jagganath, nikai.jagganath@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 will submit **LoTRS** [JESSSL26], a post-quantum lattice-based *threshold ring signature* (TRS) scheme. In a TRS [BSS02], any quorum of T signers drawn from a public ring [RST01] of N public keys can jointly produce a single signature on a message m. The verifier is convinced that some valid quorum signed, but not which one. A TRS combines the ad-hoc, group-manager-free anonymity of ring signatures with the quorum-endorsement guarantee of threshold signatures, making them a natural fit for privacy-preserving voting, whistleblowing, and endorsement workflows where a fixed family of admissible signer groups is publicly known. Still, the actual signer subset must be hidden.

In many practical deployments, the approval pattern is itself part of the public policy: federated boards sign joint statements with one representative per member institution; election officials certify results with one signer per precinct; multi-jurisdictional approvals follow a public "one approver per region" rule; and distributed-key-generation ceremonies are attested by one independent party per row of a public participant list. In all of these settings, the approval *template* — who is allowed to sign — is itself public; the only thing genuinely worth hiding is *which* named principal in each row contributed. Classical TRS anonymity over all $\binom{N}{T}$ subsets is then both the wrong notion and an overspend, and the existing post-quantum TRS literature pays for that hiding directly in proof size.

LoTRS targets the variant we call a *structured* threshold ring signature (sTRS): a TRS in which the public ring is augmented with a public structure descriptor that fixes a family of admissible signer subsets. LoTRS instantiates this by laying out the $M = T \cdot N$ public keys as a $T \times N$ table; the admissible subsets are exactly the N columns; and a valid quorum picks one key from each row of one hidden column. The relaxed anonymity is well defined, and the relaxation translates concretely into a substantial efficiency gain over the classical (T-out-of-N-anonymity) post-quantum TRS literature.

The construction comprises two lattice components: (i) a two-round lattice multi-signature in the DualMS [Che23] family that aggregates the T signers into a single signature relation, and (ii) a lattice one-out-of-many proof in the style of Esgin et al. [ESZ22] that hides which column of the table the quorum occupies. Secret keys are *not* Shamir-shared; each signer holds an ordinary lattice signing key and a public-key entry in the table. Quorum is enforced by aggregation, not by secret-sharing.

We propose LoTRS as a single crypto-system in **Category S1** of the NIST Threshold Call: a special, threshold-friendly, quantum-resistant signing primitive that is not currently standardized by NIST. LoTRS exhibits three of the S1 distinguishing factors at once: it is threshold-friendly and quantum-resistant, it is aggregatable (T signing contributions collapse into one signature), and it is ZKP-friendly (the anonymity layer is a lattice zero-knowledge proof). The output is a new primitive and is not interchangeable with any NIST-standardized signature, so LoTRS does not fit Class N.

2. Specification

Organization. The technical specification will be organized as a single Crypto-System chapter (CSX) covering the LoTRS sTRS scheme. Two sub-modules of independent interest will be specified separately and may be of use to other submitters: (i) the two-round lattice multi-signature derived from DualMS, and (ii) the lattice one-out-of-many selection-hiding proof. The composition layer that turns the two into an sTRS, the parameter set, and the encoding format are LoTRS-specific.

System Model. LoTRS operates with a public ring of $T \cdot N$ keys arranged as a $T \times N$ table $\mathsf{PK} = (\mathbf{t}_{u,i})$. A valid quorum selects one column $\ell \in [0, N-1]$ and contributes the T keys $\{\mathbf{t}_{u,\ell}\}_{u \in [0,T-1]}$. Setup is a one-time public procedure that fixes the lattice modulus, ring dimension, and challenge-distribution parameters; setup produces no trapdoors and no party holds long-term setup secrets. Key generation is fully local: each party samples its own short lattice secret and publishes its public key. The ring is formed by listing public keys in a public table; no distributed key generation, group manager, or enrollment authority is required. Signing is a two-round protocol: in round one, each of the T quorum members sends a commitment to the other $T-1$ quorum members over authenticated channels. In round two, after receiving the full set of round-one commitments, every quorum member locally computes the same aggregate commitment and derives the deterministic Fiat-Shamir challenge from this aggregate commitment, the message, and a digest of the public ring table. Each quorum member then returns a short multi-signature response together with the lattice one-out-of-many selection-hiding proof for the hidden column ℓ (which all T members of the column know); a designated combiner collects the round-two responses and assembles the final signature, and the per-signer copies of the selection-hiding proof must agree. A natural optimization in which only one party computes the selection proof once is straightforward. The combiner is a routing convenience and not a trust anchor: it cannot forge, modify, or recover signers' secrets, and dishonest behavior by the combiner is detected by an honest verifier. Verification is fully non-interactive.

Security. Existential unforgeability is reduced to standard structured-lattice assumptions: Module Learning With Errors (M-LWE) for key indistinguishability and the inner Module Short Integer Solution (M-SIS) problem for the binding component of both the multi-signature and the one-out-of-many proof. Anonymity is statistical, in the standard ring-signature anonymity game where the challenge quorum is a column of honestly-keyed signers; an adversary that has corrupted any subset of the remaining $M - T$ ring keys cannot tell which honest column produced the challenge signature. Both reductions are in the random-oracle model under the Fiat-Shamir transform. Security is proved using tree-finder extraction for the Fiat-Shamir-transformed lattice proofs; the DualMS trapdoor-free dual-signing simulation handles the two-round multi-signature layer. The core lattice-cost estimates are produced by the SageMath estimator in the artifact (see Section 3); they bundle the BKZ-based core-SVP heuristic for M-LWE and a refined approximate-SIS estimator for the binding terms.

3. Open-Source Implementation

Code Structure. Two complete reference implementations [LoT26] will ship with the package: a *Python* reference implementation written for clarity, and a *Rust* performance implementation. Both target the concrete $\kappa = 1$ instantiation of LoTRS specified in the technical specification. The Python reference is the byte-exact specification oracle: it emits a deterministic test-vector blob from fixed seeds, and the Rust implementation must reproduce setup parameters, secret keys, public keys, and signatures bit-for-bit. A SageMath parameter estimator reproduces parameter selection and the calculations of signature size and security level.

The Python reference depends only on `numpy`, `pycryptodome`, and `mpmath` as external dependencies. The Rust implementation depends on four small permissively-licensed crates: `sha3` (SHAKE-128), `subtle` (constant-time primitives), `zeroize` (secret-material clearing), and `rayon` (data parallelism for the per-signer loops). All are widely-used pure-Rust libraries with no native or system dependencies beyond a stable Rust toolchain. Both implementations include CDT-based and FACCT-style integer Gaussian samplers, a CRT-NTT polynomial backend for ring dimensions 128 and 256, Golomb-Rice signature encoding, and the full setup, key generation, aggregation, signing, and verification pipeline. Build is via `pip install -r requirements.txt` for the Python reference and `cargo build --release` for the Rust implementation. Benchmark and test-vector generation are exposed as `cargo run --release --example bench` and `python vectors.py --out vectors.json` respectively.

Code Progress and Availability. Both implementations and the estimator are complete and have been used to generate the size and timing numbers of [JESSSL26]. A preliminary implementation artifact [LoT26] is publicly available at https://github.com/lotrs-sig/lotrs. All sources are MIT-licensed; the SageMath estimator contains local copies of the lattice and ASIS/MSIS estimators, also under permissive licenses.

Networking Model. LoTRS signing is a two-round protocol with authenticated all-to-all communication among the T quorum members in round one and a designated combiner for collecting round-two responses and assembling the final signature. The combiner is not trusted; it only routes and assembles transcript-consistent contributions. For the baseline NIST evaluation platform (single 16-core machine, 64 GB RAM, Ubuntu 24.04 LTS) we will simulate the broadcast layer in-process; the current Rust benchmark reports the protocol cost on already-decoded inputs and emits a canonical Golomb-Rice signature at the combiner (verifier-side decoding is timed in the verification numbers). A loopback network harness with full per-message serialization at every party boundary, together with realistic latency and packet loss, will be included so the protocol can also be evaluated end-to-end on the wire.

Testing. The artifact [LoT26] today carries 134 Python tests and 46 Rust unit tests, plus 12 Python-Rust interop tests and a sampler known-answer test, covering ring arithmetic, the CRT-NTT fast path against a schoolbook reference, both Gaussian samplers, codec round-trip, parameter consistency, the end-to-end protocol, and tampering detection on every signature

component. The package will additionally include negative-path tests for malicious signers (incorrect commitments, premature responses, inconsistent decompositions) and a basic harness for arbitrary networking conditions.

4. Experimental Performance Evaluation

Performance. Table 1 summarizes current headline measurements from the package's experimental evaluation report. These are wall-clock timings at 128-bit security level on a laptop with 32 GB RAM and AMD Ryzen AI 9 HX 370 (Zen 5, 12 cores / 24 threads) CPU with frequency scaling (max 5.1 GHz). The implementation parallelizes per-signer loops via `rayon`. Signing measurements are end-to-end, including all rejection-sampling restarts. Verification includes the full one-out-of-many check. Sign / Verify entries are wall-clock timing means from 100 runs.

Parameter set	Ring N	Threshold T	Sig. size	Sign / Verify
`BENCH_4OF32`	32	4	23.9 KiB	120 ms / 26 ms
`BENCH_PARAMS`	32	16	25.0 KiB	149 ms / 43 ms
`PRODUCTION_PARAMS`	100	50	35.8 KiB	789 ms / 250 ms

Table 1: Headline parameter sets and measured timings (Rust release build, `rayon` multi-threaded, means over 100 signatures per cell).

At $N = 32$, $T = 16$ — i.e. a structured ring of $M = T \cdot N = 512$ keys — the signature is roughly 25 KiB. Compared to the previously best-performing lattice-based TRS in the literature [JHT25], LoTRS is approximately $3.5\times$ smaller at the headline $(T, N) = (50, 100)$ setting, while supporting larger thresholds and a wider range of (T, N) combinations.

Platform. The baseline NIST evaluation platform (single 16-core machine, 64 GB RAM) is more than adequate. The Rust implementation parallelizes the per-signer round-1 / round-2 loops and the verifier key-aggregation and selector sum with `rayon`, and uses well below 1 GiB of resident memory even for the largest published parameter set. At the headline $(N, T) = (100, 50)$ setting the dominant cost is the DualMS multi-signature layer (≈ 595 ms of ≈ 789 ms total signing; ≈ 192 ms of ≈ 250 ms total verification); the one-out-of-many proof is computed once per signing attempt and broadcast to all T signers, contributing ≈ 72 ms / ≈ 32 ms respectively. Scaling to the wider core count on the NIST platform is straightforward.

5. Licensing, Patent Claims, and Funding

Licensing. The LoTRS core code (Python reference, Rust implementation, SageMath estimator scripts, benchmark harness, and test vectors) is distributed under the MIT License (an OSI-approved permissive license). The Python reference depends on `numpy` (BSD-3-Clause), `pycryptodome` (BSD-2-Clause / Public Domain), and `mpmath` (BSD-3-Clause). The Rust implementation depends on four permissively-licensed crates: `sha3` (MIT/Apache-2.0), `subtle` (BSD-3-Clause),

`zeroize` (MIT/Apache-2.0), and `rayon` (MIT/Apache-2.0). The SageMath estimator vendors local copies of the lattice estimator and an approximate-SIS estimator, both under permissive licenses; provenance and license text are documented in the bundled estimator's README.

Patent Claims. To the best of our current knowledge, no team member is an inventor, applicant, or assignee on a patent that reads on the planned submission, and no team-affiliated entity is currently asserting patent rights over the techniques used in LoTRS. This statement will be reconfirmed at the package submission deadline.

Funding. This work is supported in part by CSIRO's Data61. Concrete grant identifiers will be acknowledged in the package submission.

6. Comparison with Other Submissions

Among Round-1 preview writeups, several PQ lattice-based threshold *signature* schemes were proposed in Category S1 and the ML-DSA-targeted N1.4 — notably Tanuki and Hermine (both threshold Raccoon variants), Mithril and Quorus (both threshold ML-DSA). LoTRS sits in the same M-LWE/M-SIS family but targets a different functional goal: signer *anonymity* within a public ring of $T \cdot N$ keys. We highlight three axes of comparison and complementarity.

Functional goal: anonymity vs. unforgeability only. The Round-1 PQ S1 submissions thresholdize an underlying *public-key* signature, with the threshold protecting against a corrupted minority of signers. LoTRS additionally hides *which* signers contributed within a public ring of $T \cdot N$ candidates; the verifier learns only that some valid quorum signed. Submissions that produce ML-DSA [DKLLSSS18]-compatible outputs (Class N1.4) and LoTRS occupy different points on the "interchangeability vs. functionality" axis: the former preserves the verification interface, the latter offers anonymity over a public ring. Combining the two — threshold ML-DSA with verifier-side anonymity — is not currently in scope for LoTRS but would be an interesting joint objective for Phase 3 analysis.

Anonymity layer. Other anonymous-signature submissions in Category S1 may instead use a T-out-of-N zero-knowledge membership proof (e.g. via succinct lattice arguments such as LaBRADOR [BS23], or via generic compilers). LoTRS instead uses a 1-out-of-many *selection* proof on a structured ring, which is asymptotically and concretely cheaper at the cost of the structured-anonymity assumption made explicit in the specification.

Multi-signature gadget. The two-round lattice multi-signature underlying LoTRS is itself reusable; teams submitting threshold-friendly M-LWE-based signatures (such as the Raccoon-derived schemes above) may find the same DualMS-style aggregation useful as a standalone gadget.

References

[BP26] Luís T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes*. NIST Internal Report NIST IR 8214C. National Institute of Standards and Technology, 2026. DOI: 10.6028/NIST.IR.8214C. doi.org/10.6028/NIST.IR.8214C.

[BS23] Ward Beullens and Gregor Seiler. "LaBRADOR: Compact Proofs for R1CS from Module-SIS". In: *CRYPTO 2023*. Vol. 14085. LNCS. Springer, 2023, pp. 518–548. Also at ia.cr/2022/1341.

[BSS02] Emmanuel Bresson, Jacques Stern, and Michael Szydlo. "Threshold Ring Signatures and Applications to Ad-Hoc Groups". In: *CRYPTO 2002*. Vol. 2442. LNCS. Springer, 2002, pp. 465–480. DOI: 10.5555/646767.704433.

[Che23] Yanbo Chen. "DualMS: Efficient Lattice-Based Two-Round Multi-Signature with Trapdoor-Free Simulation". In: *CRYPTO 2023*. Vol. 14085. LNCS. Springer, 2023, pp. 716–747. Also at ia.cr/2023/263.

[DKLLSSS18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme". In: vol. 2018. 1. 2018, pp. 238–268. Also at ia.cr/2017/633.

[ESZ22] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. "MatRiCT$^+$: More Efficient Post-Quantum Private Blockchain Payments". In: *IEEE Symposium on Security and Privacy (S&P) 2022*. IEEE, 2022, pp. 1281–1298. Also at ia.cr/2021/545.

[JESSSL26] Nikai Jagganath, Muhammed F. Esgin, Ron Steinfeld, Amin Sakzad, Markku-Juhani O. Saarinen, and Dongxi Liu. *LoTRS: Practical Post-Quantum Structured Threshold Ring Signatures from Lattices*. Cryptology ePrint Archive, Paper 2026/974. 2026. URL: https://eprint.iacr.org/2026/974.

[JHT25] Sohyun Jeon, Calvin Abou Haidar, and Mehdi Tibouchi. "LastRings: Lattice-Based Scalable Threshold Ring Signatures". In: *Information Security Conference (ISC) 2025*. LNCS. Springer, 2025, pp. 152–172. Also at ia.cr/2025/1633.

[LoT26] LoTRS Team. *LoTRS: Practical Post-Quantum Structured Threshold Ring Signatures from Lattices*. Public Reference Source Code. 2026. URL: https://github.com/lotrs-sig/lotrs.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. "How to Leak a Secret". In: *ASIACRYPT 2001*. Vol. 2248. LNCS. Springer, 2001, pp. 552–565.