

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

Title: Gargos: Threshold Schnorr Signature Scheme

Version: 0.1 (2026-01-19)¹

Team name: BDLR

Team members: Renas Bacho, Sourav Das, Julian Loss, Ling Ren

Abstract: This document specifies a three-round (t, n) -threshold variant of the Schnorr signature scheme that achieves full adaptive security under standard assumptions. Schnorr signatures are compact, efficient, and widely deployed, forming the basis of standardized schemes such as EdDSA and playing a central role in blockchain and distributed ledger systems. Their linear structure makes them particularly amenable to thresholdization, allowing multiple parties to jointly produce a single signature while preserving compatibility with existing Schnorr verification algorithms and encodings.

The proposed construction follows the three-round protocol of Bacho, Das, Loss, and Ren, consisting of commitment, opening, and response phases, followed by deterministic aggregation into a single Schnorr signature. Signing capability is distributed among n parties such that any subset of at least $t+1$ participants can generate a valid signature, while smaller subsets cannot do so. The scheme assumes a trusted-dealer setup for key generation and is proven adaptively secure against adversaries that may corrupt participants over time. Security is established in the random oracle model under the decisional Diffie–Hellman assumption.

The specification defines the cryptographic algorithms, system parameters, and operational considerations required for deployment. Owing to its efficiency, adaptive security, and reliance on standard assumptions, the scheme is well suited for high-assurance distributed applications targeted by NIST's threshold cryptography effort.

Proposed crypto-systems: Gargos: Threshold Schnorr Signature (Category S1)

Keywords: Threshold Cryptography; NIST Threshold Call; Threshold Schnorr Signatures; Adaptive Security

¹Preliminary 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 [NIST-IR8214C]. 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: Renas Bacho ^{i1,a1}, Sourav Das ^{i2,a2}, Julian Loss ^{i3,a3}, Ling Ren ^{i4,a4}

Open Researcher and Contributor Identifiers (ORCID):

i1 (0009-0007-7037-2458); i2 (0000-0001-5298-621X); i3 (0000-0002-7979-3810); i4 (0000-0003-3437-7570)

Affiliations:

^{a1} CISA Helmholtz Center for Information Security, Saarbrücken, Saarland, Germany

^{a2} Category Labs, New York, NY, USA

^{a3} Ruhr University Bochum, Bochum, North Rhine-Westphalia, Germany

^{a4} University of Illinois Urbana-Champaign, Urbana-Champaign, Illinois, USA

Main contacts:

- **Team mailing list:** gargos-bdlr@googlegroups.com
- **Primary technical contact person:** Sourav Das, souravdas1547@gmail.com
- **Secondary contact person 1:** Julian Loss, lossjulian@gmail.com

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

This package submission proposes a standardized Threshold Schnorr Signature Scheme in response to NIST’s Call for Multi-Party Threshold Schemes. The submission specifies a threshold signature construction that distributes signing capability among n participants such that any quorum of $t+1$ parties can jointly produce a valid Schnorr signature, while any smaller coalition cannot do so. The proposed cryptosystem preserves the verifier interface, encodings, and compatibility of standard single-party Schnorr signatures, enabling seamless use with existing verification code and APIs.

The proposed scheme falls within Category S1: [Special] Signing and specifies a complete threshold signing protocol based on a three-round interaction among signers. The construction follows the protocol of Bacho, Das, Loss, and Ren [BDLR25], consisting of a commitment phase, an opening phase, and a response phase, followed by deterministic aggregation into a single Schnorr signature. The submission assumes a trusted-dealer setup for key generation and focuses on the threshold signing and verification layers. Security is established under unforgeability against adaptive adversaries, with a reduction to the decisional Diffie–Hellman assumption in the random oracle model.

From a practical perspective, threshold Schnorr signatures address a critical need in distributed systems where single-key signing introduces unacceptable single points of failure. Owing to their pairing-free design, efficiency, and widespread deployment—most notably in EdDSA and blockchain systems—Schnorr signatures are particularly attractive for threshold use. By providing a complete specification of a three-round, adaptively secure threshold Schnorr signature scheme under standard assumptions, this submission contributes to the Threshold Call’s objective of building a public reference body for threshold cryptography. The proposed construction serves as a reference design for consistent implementation and comparative analysis of threshold Schnorr signatures in real-world distributed systems, in line with the goals of NIST’s Threshold Cryptography initiative.

2. Specification

The specification is organized to separate the cryptosystem definition, system model, and security formulation. The core threshold Schnorr signature scheme is specified in terms of its algorithms and parameters, with concrete scheme instantiations described within this framework.

2.1. System Model and Security Formulation

The system consists of n parties denoted P_1, \dots, P_n and a threshold parameter $t < n$. The parties communicate asynchronously over pairwise authenticated channels. All operations are defined over a cyclic group \mathbb{G} of prime order p with generator $g \in \mathbb{G}$. All scalar arithmetic is modulo p . Concatenation of byte strings is denoted by \parallel .

Security model. Security is defined via an unforgeability experiment against an *adaptive adversary* that may corrupt up to t parties over time. Upon corrupting a party P_i , the adversary obtains its

long-term secret key share sk_i and internal state and may thereafter control its behavior arbitrarily. The adversary may adaptively choose which parties to corrupt and may request partial signatures on messages of its choice.

Specifically, the scheme is *unforgeable under adaptive corruptions* if no probabilistic polynomial-time adversary can produce a valid signature σ^* on a message m^* under public key pk such that m^* was not previously queried for signing, except with negligible probability in the security parameter. For the formal security game, we refer to [BDLR25]. This definition models an interactive version of the threshold unforgeability notion TS-UF-0 in the hierarchy defined by Bellare et al. [BCKMTZ22]. However, we note that we work with the TS-UF-0 notion for simplicity and believe that a similar analysis extends to the stronger notions TS-UF- i for $i > 0$.

2.2. Proposed Cryptosystem

The threshold Schnorr signature scheme Gargos is defined as

$$\text{Gargos} = (\text{Setup}, \text{KeyGen}, \text{Sign}_1, \text{Sign}_2, \text{Sign}_3, \text{Combine}, \text{Verify}).$$

For each signing session, we assume that an external mechanism selects the message m and a signer set S with $|S| \geq t+1$. All signers are assumed to agree on the message m and the set S .

- **Setup** $\text{Setup}(1^\lambda)$. On input the cryptographic security parameter λ , select a prime-order p elliptic curve group \mathbb{G} at the desired security level (e.g., secp256k1 or ed25519 for $\lambda \approx 128$ -bit security), with independent generators $g, h, v \in \mathbb{G}$. Select independent hash functions, modeled as random oracles, parameterized by a domain-separation tag DST:

$$H, H_{\text{com}} : \{0, 1\}^* \rightarrow \mathbb{Z}_p, \quad G, F : \{0, 1\}^* \rightarrow \mathbb{G}.$$

Output the public parameters $\text{par} = (\mathbb{G}, p, g, h, v, H, H_{\text{com}}, G, F, \text{DST})$.

- **Key Generation** $\text{KeyGen}(\text{par}, n, t)$. A trusted dealer samples three independent uniformly random degree- t polynomials $s(X), r(X), u(X) \in \mathbb{Z}_p[X]$ with $r(0) = 0$ and $u(0) = 0$. The secret key share of party P_i is defined as $sk_i := (s(i), r(i), u(i))$, and its corresponding public key share as $pk_i := g^{s(i)} h^{r(i)} v^{u(i)}$. The group public key is $pk := g^{s(0)}$.
- **Round 1** $\text{Sign}_1(S, m, i, sk_i)$. In the first signing round, each signer P_i samples independent random values $a_i \leftarrow \mathbb{Z}_p$ and $\rho_i \leftarrow \{0, 1\}^\lambda$, and computes its ephemeral nonce

$$B_i := g^{a_i} \cdot F(0 \parallel \rho_i)^{r(i)} \cdot F(1 \parallel \rho_i)^{u(i)} \in \mathbb{G}.$$

It then sends the hash-commitment $\mu_i := H_{\text{com}}(i \parallel \rho_i \parallel B_i)$ to all other signers in S . The local state after this round is $st_i := (i, a_i, \rho_i, B_i)$.

- **Round 2** $\text{Sign}_2(S, m, i, (\mu_j)_{j \in S}, sk_i, st_i)$. In the second signing round, each signer P_i , upon receiving all commitments $\mu = (\mu_j)_{j \in S}$, computes its nonce

$$A_i := g^{a_i} \cdot G(0 \parallel m \parallel \mu)^{r(i)} \cdot G(1 \parallel m \parallel \mu)^{u(i)} \in \mathbb{G}.$$

It then generates a non-interactive zero-knowledge (NIZK) proof of correctness

$$\pi_i := \Sigma_P\left(\left(\text{pk}_i, A_i, B_i, g_0, g_1, \rho_i\right); (a_i, \text{sk}_i)\right)$$

for A_i with respect to (ρ_i, B_i) and pk_i , where $(g_0, g_1) := (G(0 \parallel m \parallel \mu), G(1 \parallel m \parallel \mu))$. Signer P_i then sends $(A_i, \rho_i, B_i, \pi_i)$ to all other signers in S . Its state after this round is $\text{st}_i := (i, a_i, A_i, \mu, g_0, g_1)$.

- **Round 3** $\text{Sign}_3(S, m, i, ((A_j, \rho_j, B_j, \pi_j))_{j \in S}, \text{sk}_i, \text{st}_i)$. In the third signing round, each signer P_i , upon receiving all second-round messages $\{(A_j, \rho_j, B_j, \pi_j)\}_{j \in S}$, performs for every $j \in S$ the following checks: First, the opening to the commitment is correct, i.e.,

$$\mu_j \stackrel{?}{=} H_{\text{com}}(j \parallel \rho_j \parallel B_j);$$

second, the NIZK proof π_j is valid, i.e., $\Sigma_V\left(\left(\text{pk}_j, A_j, B_j, g_0, g_1, \rho_j\right); \pi_j\right) = 1$. If any check fails, signer P_i outputs \perp and aborts.

Otherwise, it computes the aggregated nonce, challenge, and its partial signature as follows:

$$A := \prod_{j \in S} A_j^{L_{j,S}}, \quad c := H(\text{DST} \parallel A \parallel \text{pk} \parallel m), \quad z_i := L_{i,S} \cdot (a_i + c \cdot s(i)) \bmod p,$$

where $L_{j,S}$ denotes the Lagrange coefficient of index j for the set S . Finally, signer P_i sends its signature share z_i to all other signers in S .

- **Combining** $\text{Combine}(S, m, ((A_j, z_j))_{j \in S})$. Upon receiving all tuples $\{(A_j, z_j)\}_{j \in S}$, the combining algorithm Combine computes the final signature (A, z) as follows:

$$A := \prod_{j \in S} A_j^{L_{j,S}}, \quad z := \sum_{j \in S} z_j \bmod p.$$

The output $\sigma = (A, z)$ constitutes a standard Schnorr signature on the message m .

- **Verification** $\text{Verify}(\text{par}, \text{pk}, m, \sigma)$. The verification algorithm receives the public key pk , a message m , and a signature $\sigma = (A, z)$. It computes $c := H(\text{DST} \parallel A \parallel \text{pk} \parallel m)$, and accepts if and only if $g^z \stackrel{?}{=} A \cdot \text{pk}^c$. This verification equation is identical to that of a standard Schnorr signature.

2.2.1. Optional Extensions

The following optional extensions and optimizations may be supported for enhanced interoperability, robustness, or performance. These do not modify the core Gargos interface or its underlying security assumptions.

- **Identifiable aborts.** Each signer may attach a NIZK proof $\tilde{\pi}_i$ demonstrating the correctness of its partial signature z_i with respect to the verification key pk_i and nonce A_i . Specifically, π_i proves knowledge of $\text{sk}_i = (s(i), r(i), u(i))$ and a_i such that

$$A_i = g^{a_i} g_0^{r(i)} g_1^{u(i)}, \quad g^{z_i/L_{i,S}} = g^{a_i} g^{c s(i)}, \quad \text{pk}_i := g^{s(i)} h^{r(i)} v^{u(i)}.$$

This allows identification and penalization (e.g., removal or slashing) of disruptive signers.

- **Batch final verification.** Multiple final Schnorr signatures $\{\sigma_j = (R_j, s_j)\}_{j \in [k]}$ on distinct messages $\{m_j\}_{j \in [k]}$ under the same public key pk may be verified together by checking:

$$\prod_{j \in [k]} g^{r_j s_j} = \left(\prod_{j \in [k]} R_j^{r_j} \right) \cdot \left(\prod_{j \in [k]} \text{pk}^{r_j c_j} \right),$$

where $c_j = \text{H}(\text{DST} \parallel R_j \parallel \text{pk} \parallel m_j)$ and $\{r_j\}_{j \in [k]}$ are random non-zero scalars. Batch final signature verification reduces the number of verification equations from k to 1 per batch of k signatures, at the cost of three multi-exponentiations of degree k over group elements.

2.3. Security Guarantees

Correctness. Consider an execution of the signing protocol by a set $S \subseteq [n]$ of at least $t+1$ honest signers on a message m . If all parties follow the protocol and all verification checks succeed, then the protocol deterministically produces a pair (A, z) that constitutes a valid Schnorr signature on m under the public key pk . In particular, the aggregated nonce A and response z satisfy the standard Schnorr verification equation $g^z = A \cdot \text{pk}^c$, where $c := \text{H}(\text{DST} \parallel A \parallel \text{pk} \parallel m)$. Thus, any honest execution of the protocol yields a correctly generated Schnorr signature that is accepted by the standard Schnorr verification algorithm.

Underlying assumptions. Following [BDLR25], security is analyzed in the random oracle model (ROM) under the decisional Diffie–Hellman (DDH) assumption in the underlying group \mathbb{G} . The analysis also relies on the hardness of the discrete logarithm (DLOG) problem, which is implied by the hardness of the DDH problem. The scheme assumes a trusted-dealer setup for key generation and authenticated communication channels among participating parties during the signing protocol.

Security bounds. For any probabilistic polynomial-time adversary, issuing at most Q_h random oracle queries and Q_{sign} partial-signing queries, adaptively corrupting at most t parties, there is a reduction \mathcal{B}_1 to DLOG and a reduction \mathcal{B}_2 to DDH such that

$$\text{Adv}_{\mathcal{A}}^{\text{uf-cma}}(\lambda) \leq \sqrt{Q_{\text{sign}} \cdot \text{Adv}_{\mathcal{B}_1}^{\text{dlog}}(\lambda)} + \text{Adv}_{\mathcal{B}_2}^{\text{ddh}}(\lambda) + \text{negl}(\lambda),$$

where the running times of \mathcal{B}_1 and \mathcal{B}_2 are polynomially related to that of \mathcal{A} . For details, we refer to [BDLR25].

3. Open-Source Implementation (Planned)

Code structure. We plan to provide an open-source reference implementation with a small, modular *core code* that cleanly separates (i) group operations and hashing, (ii) protocol messages and serialization, (iii) signing logic, and (iv) verification and combination. The implementation is planned in *Go*, using the standard *Go* toolchain and module system. Cryptographic group operations will rely on the *Go* standard library where applicable or on a well-maintained external elliptic-curve library; large cryptographic libraries will not be vendored unless required for reproducibility.

Building and benchmarking will use Go’s native infrastructure (`go build`, `go test -bench`), with dedicated benchmarks for per-round signing, combination, and final verification.

Code progress and availability. At the time of submission, the implementation is *planned* and not yet publicly available. We intend to release a public Git-compatible repository containing the reference implementation, benchmarks, and documentation sufficient to reproduce reported performance results, together with a minimal example harness for local multi-party signing.

Implementation of the networking model. The planned implementation focuses on the cryptographic core and does not include a full networking stack. Communication is modeled as authenticated message passing among the signers in three explicit rounds. For benchmarking, all parties execute on a single host and exchange messages via in-memory or direct send/receive routines, isolating cryptographic costs from transport effects. In practice, authenticated point-to-point channels (and broadcast functionality, if required) are assumed to be provided by the surrounding system and are orthogonal to the threshold signature construction.

Testing. Testing will focus on correctness, performance, and reproducibility. This includes unit tests for each protocol round, end-to-end signing tests for multiple (t, n) configurations, and checks that malformed messages or invalid proofs are correctly detected and rejected. Benchmarking will be performed on a fixed hardware platform using the local message-passing harness. Adverse networking conditions are not explicitly simulated and are assumed to be handled at higher protocol layers.

4. Experimental Performance Evaluation (Expected)

Platform and instantiation. We consider the baseline evaluation platform specified in the Threshold Call: a single machine with 16 CPU cores and 64 GB of RAM. As a concrete instantiation, we target Schnorr signatures over the `secp256k1` elliptic curve, which is widely deployed and well supported by optimized libraries. All parties are assumed to execute on the same host for benchmarking purposes, with communication modeled via local message passing; reported costs therefore reflect cryptographic computation only.

Cost model. We use a representative cost model in which one scalar exponentiation in \mathbb{G} takes approximately 0.20 ms, a group multiplication 0.005 ms, and hash operations 0.01 ms (hash-to-group) or 0.005 ms (hash-to-scalar). Generation and verification of the NIZK proofs are modeled at $\Sigma_p \approx 1.2$ ms and $\Sigma_v \approx 0.9$ ms, respectively. These values are intended as order-of-magnitude estimates; actual performance depends on the chosen library and hardware.

Expected signing costs. Rounds 1 and 2 incur constant per-signer cost, while Round 3 scales linearly with the signing-set size $|S|$ due to verification of openings and zero-knowledge proofs from other participants. Combining is linear in $|S|$ and non-interactive. Final verification is identical to standard Schnorr verification and independent of t , n , and $|S|$.

Optional optimization: batch NIZK verification. Round 3 is dominated by verifying $|S|$ NIZK proofs. As an optional optimization, these proof verification equations can be *batch-verified* using

$ S $	Sign ₁	Sign ₂	Sign ₃	Signer total	Combiner	Sign ₃ (batch)	Signer total (batch)
3	0.64	1.83	3.35	5.82	0.61	1.83	4.30
5	0.64	1.83	5.55	8.02	1.02	2.45	4.92
9	0.64	1.83	10.01	12.48	1.84	3.69	6.16

Table 1: Expected per-session costs (ms) for secp256k1 on the baseline platform, using the illustrative cost model in the text. The “batch” columns assume batch verification of the $|S|$ NIZK proofs in Round 3 ($|S|\Sigma_V$ is replaced by one aggregated batch verification cost $\Sigma_V^{\text{batch}} \approx 0.9$ ms plus an additional multi-exponentiation overhead modeled as $0.10|S|$ ms). Final verification remains $\text{Verify} \approx 0.41$ ms and is not shown.

randomized aggregation, reducing the number of proof verifications from $|S|$ to (approximately) one aggregated check per session, at the cost of additional multi-exponentiations in \mathbb{G} .²

Evaluation discussion. These estimates indicate that the proposed three-round threshold Schnorr scheme achieves low per-signer latency for moderate signing-set sizes, with the dominant cost arising from proof verification in Round 3. All expensive operations are parallelizable, making the scheme well suited for multi-core platforms. The estimates further suggest that batch verification can significantly reduce Round 3 latency for larger signing sets by replacing the dominant $|S|\Sigma_V$ cost with a constant-sized aggregated check and a multi-exponentiation overhead. The performance gains increase with $|S|$, while the costs of combination and final signature verification remain unchanged. As with other batching techniques, batch failures require fallback to individual verification (or divide-and-conquer) to identify faulty signers.

5. Licensing, Patent Claims, and Funding

Open-source licenses. The planned reference implementation of the proposed threshold Schnorr signature scheme will be released under the MIT License. This permissive OSI-approved license allows free use, modification, and redistribution of the code, including for commercial purposes. Any external dependencies used in the implementation are expected to be distributed under permissive open-source licenses (e.g., MIT, BSD, or Apache 2.0), consistent with common cryptographic software ecosystems.

Patents. To the best of our knowledge, there are no known patents held by the authors, their employers, or sponsoring entities that do or could have claims covering the contents of this submission. The construction builds on publicly available and peer-reviewed research, and no patent claims are asserted in connection with this work at the time of submission.

External funding. Ling Ren is supported by the National Science Foundation award #2240976. Julian Loss is supported by the European Union, ERC-2023-StG-101116713. Views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

²Batch verification is sound except with negligible probability; if a batch fails, implementations should fall back to individual verification (or a divide-and-conquer strategy) to locate invalid proofs.

References

- [BCKMTZ22] Mihir Bellare, Elizabeth Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. “Better than Advertised Security for Non-interactive Threshold Signatures”. In: *Advances in Cryptology – CRYPTO 2022*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Cham: Springer Nature Switzerland, 2022, pp. 517–550. DOI: [10.1007/978-3-031-15985-5_18](https://doi.org/10.1007/978-3-031-15985-5_18). URL: https://crypto.iacr.org/2022/papers/538806_1_En_18_Chapter_OnlinePDF.pdf.
- [BDLR25] Renas Bacho, Sourav Das, Julian Loss, and Ling Ren. “Adaptively Secure Three-Round Threshold Schnorr Signatures from DDH”. In: *Advances in Cryptology – CRYPTO 2025*. Ed. by Yael Tauman Kalai and Seny F. Kamara. Cham: Springer Nature Switzerland, 2025, pp. 390–422. DOI: [10.1007/978-3-032-01887-8_13](https://doi.org/10.1007/978-3-032-01887-8_13). Also at ia.cr/2025/1009.
- [NIST-IR8214C] Luís T. A. N. Brandão and René Peralta. *NIST First Call for Multi-Party Threshold Schemes*. (National Institute of Standards and Technology) NIST Internal Report (NISTIR) 8214C. 2026. DOI: [10.6028/NIST.IR.8214C](https://doi.org/10.6028/NIST.IR.8214C).