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

Title: Haystack: Threshold and Distributed Stateful Hash-Based Signatures

Version: 0.1 (2026-01-21)[1]

Team name: Haystack

Team members: John Kelsey, Stefan Lucks, Nathalie Lang

Abstract: We propose techniques for making threshold and distributed versions of stateful hash-based signature schemes, such as those in NIST standard [SP800-208] and IETF standards XMSS [RFC8391] and LMS [RFC8554]. Our techniques require a trusted dealer and a large (several GiB) common reference string, use only point-to-point communications, and are very efficient. Our techniques permit both threshold signatures (any t of the n trustees may sign) and more flexible distributed signatures (during setup, we can choose arbitrary subsets of the trustees who may sign), so long as the total number of signing coalitions is reasonable (e.g., less than 100 or so). We have a reference implementation in Python of our techniques for LMS signatures. The scheme was published in CIC2025, and the paper includes a security proof reducing the security of the threshold/distributed version of the signature to that of the underlying stateful hash-based signature scheme. Along with providing a new and powerful functionality for stateful hash-based signatures, our techniques also help solve many of the practical problems of state management in these signature schemes, by distributing knowledge of the state in a way that guarantees that many devices must fail at the same time in order for a key to be reused.

Proposed crypto-systems: Haystack: Category N1.5

Keywords: Threshold Cryptography, NIST Threshold Call, Threshold Signatures, Hash-Based Signatures, Stateful Hash-Based Signatures, XMSS, LMS

Photo by Roman Tymochko: https://www.pexels.com/photo/brown-hays-on-a-grassy-field-8652819/

[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 [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: John Kelsey [i1,a1,a2], Stefan Lucks [i2,a3], Nathalie Lang [i3,a3]

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0002-3427-1744); i2 (0000-0003-4906-5131); i3 (0000-0002-2768-9878)

Affiliations:

[a1] National Institute of Standards and Technology @ Gaithersburg, MD, USA

[a2] Computer Security and Industrial Cryptography, KU Leuven @ Leuven, Belgium

[a3] Bauhaus-Universität Weimar @ Weimar, Germany

Main contacts:

- **Team mailing list:** haystack-threshold-hbs@googlegroups.com
- **Primary technical contact person:** John Kelsey, john.kelsey@nist.gov

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 submission provides techniques for turning stateful hash-based signatures (HBS), such as the schemes described in IETF standards LMS [RFC8554] and XMSS [RFC8391] and the NIST standard [SP800-208], into threshold and distributed signature schemes. These techniques were published in [KLL25].

Our techniques:

1. Require a trusted dealer for setup, and carry out signing as a protocol between an untrusted aggregator and a set of trustees.
2. Require a large common reference string, called the CRV, which is created during setup for each LMS or XMSS key. For typical LMS or XMSS parameters, the common reference string ranges in size from around 0.1 GiB to around 10 GiB.
3. Are very computationally efficient–during signing, the aggregator and trustees each do about the same amount of computation done during a normal LMS or XMSS signature.
4. Allow for low-end devices to serve as trustees. The aggregator must have access to the large common reference string, and so needs a local storage or internet access to a cloud storage service; each trustee needs only a small amount of state (to avoid key reuse) and a single PRF key.
5. Produce standard LMS or XMSS signatures; the verification algorithm is unchanged.
6. Use only symmetric cryptography and point-to-point communications.
7. Are postquantum, just like the underlying stateful hash-based signature schemes.
8. Support threshold signatures where the total number of valid signing coalitions[2] is reasonable—say, less than a hundred or so. A 3-of-5 scheme is very practical, a 5-of-9 scheme is still workable, but a 10-of-20 scheme is not.
9. Ensure that any group of trustees that does not make up a valid signing coalition cannot sign or forge a message. Thus, in the notation of [NIST-IR8214C], $f = t - 1$.
10. Support distributed signatures, allowing the dealer to establish arbitrary sets of signing coalitions, again with the constraint that the scheme becomes impractical for large numbers of signing coalitions.

Threshold signatures increase the security of stateful hash-based signatures. The most important practical security problem with stateful hash-based signatures is reusing a one-time key—something that can happen due to a variety of hardware, software, or user errors, and that allows attackers to sign arbitrary messages. By requiring multiple trusted devices to interact in order to sign with a given one-time key, it becomes almost impossible to accidentally reuse a key—multiple devices

[2] A signing coalition is a set of trustees who can collectively sign a message if every member of the coalition participates in the signature. For example, a 3-of-5 threshold signature scheme has $\binom{5}{3}$ distinct signing coalitions, since each different group of three trustees can work together to sign a message.

must fail at the same time in order for key reuse to occur. This makes these techniques very practically useful.

2. Specification

2.1. Organization

Our techniques apply to stateful hash-based signatures such as LMS [RFC8554] and XMSS [RFC8391]. While they could be applied to hypertree versions of these standards or to stateless hash-based signatures [FIPS205; FD24], doing this would generally require an impractically large setup time and common reference string size.

2.2. System Model

Our techniques require a trusted dealer for setup.

We assume only point-to-point communications between the aggregator and each trustee. Importantly, we never require a broadcast channel or even the use of public key cryptography in our protocols.

For each run of the signing protocol, we assume that the aggregator starts out knowing which coalition of trustees it will use to sign; any polling that might be needed to verify availability is assumed to be done before the signing protocol runs.

2.3. Security

We prove the security of our techniques by reducing them to the security of the underlying stateful HBS and the security of the PRF used to derive trustee shares and construct check values. Our proof does not require a random oracle assumption, and so is consistent with schemes like XMSS whose security is proven in the concrete model. We prove security against a malicious adversary who controls the aggregator and network and makes static corruptions of any subset of trustees that do not make up a valid signing coalition.

We prove one-more unforgability [NT24] instead of the more common (but weaker) CM-EUF unforgability. This guarantees that our threshold and distributed signatures preserve the strong unforgability offered by stateful hash-based signatures.

Stateful HBS schemes and symmetric PRFs remain secure even in a world with quantum computers, and our threshold and distributed signatures are likewise quantum-secure. As HBS schemes and PRFs are available supporting Category 1, 3, and 5 security, our techniques can likewise support these security levels.

3. Overview of Techniques

3.1. Background on Stateful Hash-Based Signatures

Hash-based signatures provide the functionality of digital signatures, based only on cryptographic hash functions—there is no underlying nice mathematical structure or hard problem, as with RSA or ML-DSA. A stateful hash-based signature key (such as used in LMS or XMSS) is actually composed of many (for example, 2^{15}) one-time signing keys. As the name suggests, each one-time signing key must never be used to sign more than one message. The one-time signing keys are collected in a Merkle tree, and the root of the Merkle tree (which commits to the entire set of one-time keys) becomes part of the public key of the scheme. The final signature from a stateful hash-based signature consists of two parts: A one-time signature on the message (using one of the one-time keys), and a Merkle tree path proving that this one-time key is one of the one-time keys associated with this public key. The signer must be very careful to keep track of which one-time keys have been used, as key reuse allows an attacker to forge signatures, and this is a major practical barrier to the widespread use of stateful hash-based signatures. For a more complete explanation of hash-based signatures, see [Gre18; Kel19; Men].

3.2. Why is it Hard?

There are a few fundamental problems that had to be solved in order to get threshold stateful HBS working:

1. No Algebraic Structure: Threshold RSA can make use of the algebraic structure of RSA; hash-based signatures have no such structure.
2. Randomized Hashes: All modern hash-based signatures sign a randomized hash of a message. This requires a distributed protocol for producing a randomizer for the hash, which adds overhead and complexity to the signing process.
3. One-Time Keys: Stateful HBS keys are constructed from a large number of one-time signing keys, and it is critical that these keys never be reused. Ensuring that we never reuse a one-time key in a distributed system creates another set of difficulties that must be overcome.

In all three cases, we considered a number of approaches, but converged on relatively straightforward (if perhaps inelegant) solutions that gave us a practical system based only on boring cryptography.

3.3. Overcoming Lack of Algebraic Structure

This is the first question most people ask us when we describe our scheme: *how is it even possible to do threshold signatures with something with no algebraic structure?*

While a full description of the scheme is beyond the scope of this preview, the key insight is that each one-time key can be represented as a two-dimensional array of values $X[0 \dots p-1, 0 \dots 2^w-1]$,

and signing can be done using only the randomized hash to be signed and this array. Each value in this array can be split into shares $X^{1\ldots n}[0 \ldots p-1, 0 \ldots 2^w - 1]$, and the shares distributed to the trustees. And then, the normal Winternitz signing algorithm can be carried out on these shares, and the resulting shares of signatures can be combined together by the aggregator to get a full valid Winternitz signature. In order to avoid requiring lots of storage for each trustee, we generate the trustees' shares pseudorandomly, and store additional information to reconstruct the correct values in the CRV. For full details, see [KLL25].

3.4. Randomization

In order to keep the security properties of randomized hashing, the randomizer must not be influenced by the aggregator or any group of trustees less that a signing coalition, and also neither the aggregator nor any group of trustees may learn the randomizer until the message has been seen by all the participating trustees. We accomplish this by having the randomizers for each one-time key (recall that each key is used only once) generated by the trusted dealer during the setup, along with check values that permit each trustee to verify that the randomizer for this one-time key is correct. Both randomizers and check values are then split up into shares in the same way as the one-time keys.

The full protocol involves two round trip messages:

1. Aggregator sends M to each participating trustee.
2. Each trustee t sends back a share of the randomizer and check value: R^t, CHK^t
3. The aggregator uses the CRV and the shares to construct the correct values R, CHK, and sends them to each participating trustee.
4. Each trustee t sends back its share of the one-time signature, Z^t.

At the end of the protocol, the aggregator combines the shares it has received, along with some information from the CRV, to construct the one-time signature on hash(R, M).

3.5. Key Reuse

The final problem is key reuse. Signing twice with the same key allows an attacker to forge arbitrary signatures. But in a distributed system of this kind, preventing key reuse without a broadcast channel is quite hard.

Our solution is to construct a t-of-n threshold signature scheme from a collection of $\binom{n}{t}$ t-of-t signature schemes. We start with a total of D one-time keys that are include in the stateful HBS scheme's key, and then assign a collection of these one-time keys (called a "shard") to each valid signing coalition. Thus, in a 2-of-3 threshold signature scheme, one third of the one-time keys will belong to the signing coalition consisting of trustees 1 and 2, another third will belong to the coalition of trustees 1 and 3, and the final third will belong to the coalition of trustees 2 and 3.

The one-time keys are collected together in the stateful hash-based signature key. From the perspective of a verifier, every signature is simply a normal signature from the stateful hash-based signature scheme. Our approach is conceptually similar to that of [BM18]—there, each signing coalition is a single trustee, and so they get a group signature scheme, or equivalently, a 1-of-n threshold signature scheme.

This somewhat inelegant scheme yields a surprising advantage–we are not restricted to threshold signatures, but instead can construct a *distributed signature scheme* [HPS03]. That is, we can select arbitrary coalitions of trustees to sign. For example, if Alice is the boss and she has three assistants, Bob, Carol, and Dave, she can set up a *distributed* signature scheme so that Alice can sign with any one assistant, but all three assistants must sign together if Alice is not present. Quite elaborate access structures can be constructed using sharding, since the dealer can select any set of signing coalitions it wants. The dealer can even allocate different numbers of one-time keys (and thus signatures) to each coalition. In the example above, Alice might consider the coalition of Bob, Carol, and Dave one that would be used only in rare emergencies, and so allocate most of the keys to the first three coalitions, and only a handful of keys to the last one.

4. Open-Source Implementation

We have a Python reference implementation of our techniques for LMS signatures which we will make publicly available before the final submission of the scheme is due. Our implementation is open-source, based on the open-source Python implementation of LMS by Russ Housley.

5. Experimental Performance Evaluation

5.1. Performance

The aggregator and each trustee each do approximately the same computational work for running the threshold signing protocol as is required to sign with the underlying stateful HBS. The work for setup is dominated by the time taken to generate the stateful HBS key. Our techniques use only symmetric cryptography and point-to-point communications between the aggregator and trustees.

5.2. Threshold Parameters and Number of Signing Coalitions

A *signing coalition* is a set of trustees that are able to sign a message if they all participate. For example, a 3-of-5 threshold signature scheme has $\binom{5}{3} = 10$ valid signing coalitions, one for each possible collection of three trustees.

A stateful hash-based signature key has some fixed number D of one-time signing keys, and our techniques assign each signing coalition exclusive ownership of a block of those keys. This means

that in order to have a reasonable number of signatures available for each signing coalition, we need to keep the total number of signing coalitions acceptably small. For example, consider an LMS key with 2^{20} one-time keys. No more than 2^{20} signatures may ever be done from this key. If we have a 5-of-10 threshold signature scheme, then each signing coalition is assigned about 4161 one-time keys, and so can sign no more than 4161 messages in its lifetime. By contrast, consider a 10-of-20 threshold scheme: each of the $\binom{20}{10} = 184756$ signing coalitions would get only five signing keys!

5.3. Platform

The aggregator needs access to the common reference string (CRV). For typical parameters, this will be between 0.1 GiB – 10 GiB.

For efficiency, the aggregator should keep track of which one-time keys have been used so far, but this is not critical for security, since no trustee will participate in a signature with a key it believes to have been used.

Trustees do not need access to the common reference string, but must keep track of state for each of their signing coalitions. For a t-of-n threshold signature scheme, each trustee must keep track of state for $\binom{n-1}{t-1}$ coalitions. As long as the number of coalitions is kept reasonable, this remains practical even for relatively low-end devices like smartcards.

5.4. Common Reference String Size

The common reference string grows linearly in the number of one-time keys D included in the hash-based signature key. The common reference string also grows linearly in the number of trustees, but due to other constraints on number of trustees discussed below, this has very little practical impact.

5.5. Practical Usefulness

The major benefit of our techniques is that they make state management much easier. A normal LMS signer will reuse a key (and thus allow forgeries) if the signer's device fails in certain ways; with a 3-of-5 threshold version of LMS, three separate trusted devices must fail in the same way for the same disaster to take place.

6. Licensing, Patent Claims, and Funding

We make no patent claims on these techniques, and it is our intent that they be available for universal use.

References

[BM18] Rachid El Bansarkhani and Rafael Misoczki. "G-Merkle: A Hash-Based Group Signature Scheme from Standard Assumptions". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. Ed. by Tanja Lange and Rainer Steinwandt. Vol. 10786. Lecture Notes in Computer Science. Springer, 2018, pp. 441–463. DOI: 10.1007/978-3-319-79063-3_21. Also at ia.cr/2018/274.

[FD24] Scott R. Fluhrer and Quynh Dang. "Smaller Sphincs$^+$". In: *IACR Cryptol. ePrint Arch.* (2024), p. 18. URL: https://ia.cr/2024/018.

[FIPS205] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard*. FIPS Federal Information Processing Standards Publications (FIPS) 205. Washington, D.C.: U.S. Department of Commerce, August 2024. DOI: 10.6028/NIST.FIPS.205.

[Gre18] Matt Green. *Hash-based Signatures: An illustrated Primer*. April 2018. URL: https://blog.cryptographyengineering.com/2018/04/07/hash-based-signatures-an-illustrated-primer/.

[HPS03] Javier Herranz, Carles Padró, and Germán Sáez. "Distributed RSA Signature Schemes for General Access Structures". In: *Information Security*. Ed. by Colin Boyd and Wenbo Mao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 122–136. DOI: 10.1007/10958513_10.

[Kel19] John Kelsey. *Introduction to Hash Based Signatures (COSIC Seminar)*. October 2019. URL: https://youtu.be/jiU0ICoiPI0.

[KLL25] John Kelsey, Nathalie Lang, and Stefan Lucks. "Turning Hash-Based Signatures into Distributed Signatures and Threshold Signatures: Delegate Your Signing Capability, and Distribute it Among Trustees". In: *IACR Commun. Cryptol.* 2.2 (2025), p. 24. DOI: 10.62056/A6KSUDY6B.

[Men] Alfred Menezes. *Lecture 1. Introduction (Hash-Based Signatures)*. URL: https://youtu.be/pt5WR8D4IXI.

[NT24] Sela Navot and Stefano Tessaro. "One-More Unforgeability for Multi - and Threshold Signatures". In: *Advances in Cryptology - ASIACRYPT 2024 - 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9-13, 2024, Proceedings, Part I*. Ed. by Kai-Min Chung and Yu Sasaki. Vol. 15484. Lecture Notes in Computer Science. Springer, 2024, pp. 429–462. DOI: 10.1007/978-981-96-0875-1_14. Also at ia.cr/2024/1947.

[RFC8391] Hülsing, Butin, Gazdag, Rijneveld, and Mohaisen. *XMSS: eXtended Merkle Signature Scheme*. RFC 8391. RFC Editor, May 2018. DOI: 10.17487/RFC8391.

[RFC8554] McGrew, Curcio, and Fluhrer. *Leighton-Micali Hash-Based Signatures*. RFC 8554. RFC Editor, April 2019. DOI: 10.17487/RFC8554.

[SP800-208] David Cooper, Daniel Apon, Quynh Dang, Michael Davidson, Morris Dworkin, and Carl Miller. *Recommendation for Stateful Hash-Based Signature Schemes*. Tech. rep. Gaithersburg, MD, United States: National Institute of Standards & Technology, 2020. DOI: 10.6028/NIST.SP.800-208.

[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.