

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

Title: TALH: Threshold Authentication via Leaf-sharing in Hash-based Signatures

Version: Preview Writeup 0.1 (2026-08-08)¹

Team name: TALH

Team members: Muhammed Said Gündoğan

Abstract: This document previews our TALH and ASPEN submissions to the NIST Multi-Party Threshold Cryptography (MPTC) Call. Both are hash-based threshold signature schemes, so their security rests on a hash function alone rather than on a structured algebraic problem. TALH (Threshold Authentication via Leaf-sharing in Hash-based Signatures) is stateless. It distributes among the trustees the secrets of a forest of random subsets (FORS), and signs in three rounds, two of which settle a randomizer by commit-then-reveal. Two ways of sharing are offered: a compact one, from replicated seeds, whose cost is combinatorial in the number of trustees; and a scalable one, from Vandermonde sharing, which avoids that growth and therefore also serves the medium and large party-count profiles of the Call. ASPEN is stateful. It uses no randomizer, or equivalently a constant one, so signing takes a single round. ASPEN thus turns the stateful standards LMS and XMSS into threshold schemes with a non-interactive signing phase. Both assume a trusted dealer; the aggregator holds only public data, so any trustee may play its role.

Proposed crypto-systems:

- (I) TALH (Categories N1.5 and S1)
- (II) ASPEN (Categories N1.5 and S1)

Keywords: Threshold Cryptography; NIST Threshold Call; Threshold Signatures; Hash-Based Signatures; Stateful Hash-Based Signatures; Stateless Hash-Based Signatures; XMSS; LMS; SPHINCS; SLH-DSA



Logo AI-Generated with Grok Imagine (xAI's Aurora model).

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: Muhammed Said Gündoğan^{i1,a1}

Open Researcher and Contributor Identifiers (ORCID):

i1 (0000-0001-9450-758X)

Affiliations:

^{a1} National Research Institute of Electronics and Cryptology (UEKAE), TÜBİTAK BİLGEM @ Gebze, Kocaeli, Türkiye

Main contacts:

- **Team mailing list:** muhammedsaidgundogan@gmail.com
- **Primary technical contact person:** Muhammed Said Gündoğan, muhammedsaidgundogan@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 submission outlines two hash-based signature (HBS) proposals: TALH, a stateless threshold signature scheme, and ASPEN, a stateful non-interactive one.

For category S1 we intend to instantiate TALH at NIST security categories 1, 3 and 5, and we target the dishonest-majority threshold profiles $n2$, $n3fD$, $nSfD$, $nMfD$ and $nLfD$. The same security categories and profiles are targeted for the N1.5 instantiation of TALH, whose signatures verify under the unmodified SLH-DSA verifier. ASPEN is likewise instantiated at security categories 1, 3 and 5 in category S1, but only the profiles $n2$, $n3fD$ and $nSfD$ are targeted under a dishonest majority; the medium and large profiles $nMfH$ and $nLfH$ require an honest majority. In category N1.5 it reaches only security category 1, since its deterministic message digest rests on collision resistance, so that a 256-bit digest yields only 128-bit security.

2. Specification

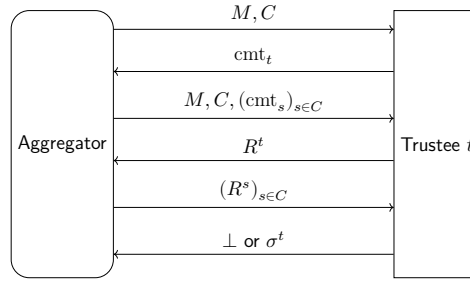


Figure 1: The three-round TALH protocol between the aggregator and each trustee t of the coalition C , common to the compact and the scalable version (Algorithms 1 and 2).

This section specifies the compact and the scalable version of TALH, and ASPEN. They are at their most efficient in category S1, where the parameters may be chosen for the construction at hand, but they can also be adapted to category N1.5.

TALH is stateless, and its key generation must materialize the entire tree, which is why it is proposed as a single-tree scheme. It can nevertheless be placed under SLH-DSA [FIPS205]: fixing the upper layers of the hypertree to constants and instantiating the construction on the lower layers yields signatures that verify under the unmodified SLH-DSA verifier. The price is a less efficient scheme with longer signatures than the S1 instantiation, and a signing capacity of 2^{10} to 2^{20} per key instead of the 2^{64} of SLH-DSA. ASPEN, in turn, derives the randomness of the message digest deterministically, so that 128-bit security is obtained only when it is adapted to XMSS at its 256-bit parameter set.

The Threshold Call explicitly accommodates such a trade-off: “in the interest of research, the computational security of a submitted threshold scheme does not need to be as high as that of the primitive being thresholdized” [NIST-IR8214C]. The reduced signing capacity of our schemes,

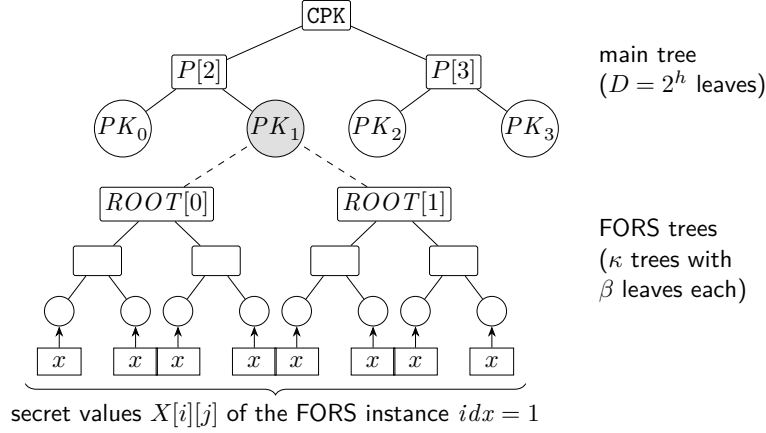


Figure 2: The single-tree stateless HBS for $D = 4$, $\kappa = 2$, $\beta = 4$. Only the values $X[i][j]$ are secret; everything derived from them by hashing is public. A signature reveals one secret per FORS tree, in the instance idx selected by the message digest.

relative to the primitives they thresholdize, is to be read in that light, and it is on this basis that we place both proposals in category N1.5.

2.1. TALH

TALH instantiates the single-tree stateless HBS built from FORS instances that is depicted in Figure 2, and distributes the FORS secrets among the trustees. As in other stateless constructions, a randomizer R is sampled and digested together with the message M ; the resulting digest determines both the FORS instance to be used and which secrets of that instance are revealed. The choice of R is therefore security-critical, and R is settled by commit–then–reveal in two rounds (see Figure 1). In the third round the trustees release their shares of the selected secrets, from which the aggregator reconstructs those secrets and assembles the signature (the aggregator holds only public values, so the role may equally be played by one of the trustees).

Two ways of sharing the FORS secrets are proposed.

- *Replicated secret sharing.* This construction is PRF-friendly: the dealer distributes n -bit seeds from which secrets are derived by PRF streams, so each trustee stores only a small amount of key material. Concretely, one seed is drawn for each of the $\binom{\ell}{k-1}$ maximal unqualified sets and replicated to the trustees outside that set; every FORS secret is the XOR of the corresponding PRF streams. Thus any k trustees jointly hold every seed, while any $k - 1$ miss the seed indexed by their own set (Algorithm 1). The number of seeds, and thus the cost, scales with $\binom{\ell}{k-1}$, so the construction suits small participant sets; we call it the *compact* version.
- *Vandermonde secret sharing.* This construction is not PRF-friendly: the dealer applies a Vandermonde (k, ℓ) -threshold sharing over $\text{GF}(2^\delta)$ to every FORS secret individually, so a trustee stores its shares in a table instead of deriving them, and the coalition reconstructs by interpolation instead of plain XOR (Algorithm 2). The amount of key material scales as

Algorithm 1: Key material and signing in the compact scheme

```

function TALH_Key_Gen_Comp( $D, k, \ell$ )
  Rep_Deal( $k, \ell$ )
  for all  $idx, i, j$  do
     $X_{idx}[i][j] \leftarrow \bigoplus_{B \in \mathfrak{B}} \text{PRF}_{K_B}^{\text{Sec}}(idx, i, j, n)$ 
  (CPK, CRV)  $\leftarrow$  BuildCRV( $X_{0 \dots D-1}$ )
  Publish (CPK, CRV); delete everything else.

function TALH_Sign_Comp( $M, \text{CRV}, C$ )
  for each  $t \in C$  do
     $\text{cmt}_t \leftarrow \text{Sign1}(t, M, C)$ 
  for each  $t \in C$  do
     $R^t \leftarrow \text{Sign2}(t, M, C, (\text{cmt}_s)_{s \in C})$ 
     $R \leftarrow \bigoplus_{s \in C} R^s$ 
     $\text{digest} \leftarrow \text{H}_{\text{msg}}(R \| M)$ 
     $(idx, c[0 \dots \kappa - 1]) \leftarrow \text{Parse}(\text{digest})$ 
    for each  $t \in C$  do
       $\sigma^t[0 \dots \kappa - 1] \leftarrow \text{Sign3\_Comp}(t, (R^s)_{s \in C})$ 
    for  $i \leftarrow 0 \dots \kappa - 1$  do
       $x[i] \leftarrow \bigoplus_{t \in C} \sigma^t[i]$ 
      if  $\text{H}_{\text{leaf}}(idx, i, c[i], x[i]) \neq \text{CRV}[idx].T[i][c[i] + \beta]$ 
      then
        return  $\perp$ 
       $\text{AUTH}[i] \leftarrow$  path of leaf  $c[i]$  read from  $\text{CRV}[idx].T[i]$ 
     $\text{PATH} \leftarrow$  path of leaf  $idx$  read from  $\text{CRV}.P$ 
    return  $(R, \text{PATH}, (x[0 \dots \kappa - 1], \text{AUTH}[0 \dots \kappa - 1]))$ 

function BuildCRV( $X_{0 \dots D-1}$ )
  for  $idx \leftarrow 0 \dots D - 1$  do
    for  $i \leftarrow 0 \dots \kappa - 1$  do
      for  $j \leftarrow 0 \dots \beta - 1$  do
         $Y_{idx}[i][j] \leftarrow \text{H}_{\text{leaf}}(idx, i, j, X_{idx}[i][j])$ 
         $\text{CRV}[idx].T[i] \leftarrow \text{MT\_Construct}(Y_{idx}[i][0 \dots \beta - 1])$ 
       $\text{ROOT}_{idx}[i] \leftarrow$  root of  $\text{CRV}[idx].T[i]$ 
     $\text{PK}_{idx} \leftarrow \text{H}_{\text{pk}}(idx, \text{ROOT}_{idx}[0] \parallel \dots \parallel \text{ROOT}_{idx}[\kappa - 1])$ 
   $\text{CRV}.P \leftarrow \text{MT\_Construct}(\text{PK}_{0 \dots D-1})$ 
   $\text{CPK} \leftarrow \text{CRV}.P[1]$ 
  return (CPK, CRV)

function Rep_Deal( $k, \ell$ )
  for each  $B \in \mathfrak{B}$  do
     $K_B \leftarrow \$ \{0, 1\}^n$ 
  for  $t \leftarrow 1 \dots \ell$  do
    Give  $\{K_B : t \notin B\}$  to trustee  $t$ 

function Rep_Share( $t, C, idx, c[0 \dots \kappa - 1]$ )
  for  $i \leftarrow 0 \dots \kappa - 1$  do
     $\sigma^t[i] \leftarrow \bigoplus_{B: \min(C \setminus B) = t} \text{PRF}_{K_B}^{\text{Sec}}(idx, i, c[i], n)$ 
  return  $\sigma^t[0 \dots \kappa - 1]$ 

function Sign1( $t, M, C, \text{st}_t$ )
   $R^t \leftarrow \$ \{0, 1\}^n$ 
   $\text{st}_t \leftarrow \text{st}_t \cup \{(M, C, R^t)\}$ 
  return  $(\text{H}_{\text{cmt}}(t, R^t), \text{st}_t)$ 

function Sign2( $t, M, C, (\text{cmt}_s)_{s \in C}, \text{st}_t$ )
  Pick  $(M, C, R^t)$  from  $\text{st}_t$ 
  assert  $\{\text{cmt}_t = \text{H}_{\text{cmt}}(t, R^t)\}$ 
   $\text{st}_t \leftarrow \text{st}_t \cup \{(M, C, (\text{cmt}_s)_{s \in C}, R^t)\}$ 
  return  $(R^t, \text{st}_t)$ 

function Sign3_Comp( $t, (R^s)_{s \in C}, \text{st}_t$ )
  Pick  $(M, C, (\text{cmt}_s)_{s \in C}, R^t)$  from  $\text{st}_t$ 
  assert  $\{\forall s \in C : \text{cmt}_s = \text{H}_{\text{cmt}}(s, R^s)\}$ 
   $R \leftarrow \bigoplus_{s \in C} R^s$ 
   $\text{digest} \leftarrow \text{H}_{\text{msg}}(R \| M)$ 
   $(idx, c[0 \dots \kappa - 1]) \leftarrow \text{Parse}(\text{digest})$ 
  return Rep_Share( $t, C, idx, c[0 \dots \kappa - 1]$ )

```

$O(\ell k)$ rather than $O\left(\binom{\ell}{k-1}\right)$, so the construction remains viable for large participant sets; we call it the *scalable* version.

2.2. ASPEN

ASPEN is designed so that signing completes in a single round. A fresh randomizer is unavoidable in the stateless setting, whereas a stateful scheme can do without one, or equivalently use a constant one. The cost is confined to the message digest, whose security argument then rests on

Algorithm 2: Key material and signing in the scalable scheme

```

function TALH_Key_Gen_Scal( $D, k, \ell$ )
  for all  $idx, i, j$  do
     $X_{idx}[i][j] \leftarrow \$ \{0, 1\}^n$ 
     $(w_1, \dots, w_\ell) \leftarrow \text{VDM\_Share}(X_{idx}[i][j], k, \ell)$ 
    for  $t \leftarrow 1 \dots \ell$  do
       $W[t][idx][i][j] \leftarrow w_t$ 
   $(\text{CPK}, \text{CRV}) \leftarrow \text{BuildCRV}(X_{0 \dots D-1})$ 
  Give  $W[t]$  to trustee  $t$ ; publish  $(\text{CPK}, \text{CRV})$ ;
  delete everything else.

function VDM_Share( $s, k, \ell$ )
   $a_0 \leftarrow s$ 
  for  $j \leftarrow 1 \dots k-1$  do
     $a_j \leftarrow \$ \{0, 1\}^{|s|}$ 
  for  $t \leftarrow 1 \dots \ell$  do
     $w_t \leftarrow \bigoplus_{j=0}^{k-1} x_t^j \cdot a_j$ 
  return  $(w_1, \dots, w_\ell)$ 

function VDM_Coef( $t_0, \dots, t_{k-1}$ )
  for  $i \leftarrow 0 \dots k-1$  do
     $\lambda_i \leftarrow \prod_{j \neq i} x_{t_j} \cdot (x_{t_j} \oplus x_{t_i})^{-1}$ 
  return  $(\lambda_0, \dots, \lambda_{k-1})$ 

function VDM_Rec( $(t_0, w_{t_0}), \dots, (t_{k-1}, w_{t_{k-1}})$ )
   $(\lambda_0, \dots, \lambda_{k-1}) \leftarrow \text{VDM\_Coef}(t_0, \dots, t_{k-1})$ 
  return  $\bigoplus_{i=0}^{k-1} \lambda_i \cdot w_{t_i}$ 

function TALH_Sign_Scal( $M, \text{CRV}, C$ )
  for each  $t \in C$  do
     $\text{cmt}_t \leftarrow \text{Sign1}(t, M, C)$ 
  for each  $t \in C$  do
     $R^t \leftarrow \text{Sign2}(t, M, C, (\text{cmt}_s)_{s \in C})$ 
   $R \leftarrow \bigoplus_{s \in C} R^s$ 
   $\text{digest} \leftarrow \text{H}_{\text{msg}}(R \| M)$ 
   $(idx, c[0 \dots \kappa-1]) \leftarrow \text{Parse}(\text{digest})$ 
  for each  $t \in C$  do
     $\sigma^t[0 \dots \kappa-1] \leftarrow \text{Sign3\_Scal}(t, (R^s)_{s \in C})$ 
  for  $i \leftarrow 0 \dots \kappa-1$  do
     $x[i] \leftarrow \text{VDM\_Rec}((t, \sigma^t[i])_{t \in C})$ 
    if  $\text{H}_{\text{leaf}}(idx, i, c[i], x[i]) \neq \text{CRV}[idx].T[i][c[i] + \beta]$  then
      return  $\perp$ 
   $\text{AUTH}[i] \leftarrow \text{path of leaf } c[i] \text{ read from } \text{CRV}[idx].T[i]$ 
   $\text{PATH} \leftarrow \text{path of leaf } idx \text{ read from } \text{CRV}.P$ 
  return  $(R, \text{PATH}, (x[0 \dots \kappa-1], \text{AUTH}[0 \dots \kappa-1]))$ 

function Sign3_Scal( $t, (R^s)_{s \in C}, \text{st}_t$ )
  Pick  $(M, C, (\text{cmt}_s)_{s \in C}, R^t)$  from  $\text{st}_t$ 
  assert  $\{ \forall s \in C : \text{cmt}_s = \text{H}_{\text{cmt}}(s, R^s) \}$ 
   $R \leftarrow \bigoplus_{s \in C} R^s$ 
   $\text{digest} \leftarrow \text{H}_{\text{msg}}(R \| M)$ 
   $(idx, c[0 \dots \kappa-1]) \leftarrow \text{Parse}(\text{digest})$ 
  return  $(W[t][idx][i][c[i]])_{i=0}^{\kappa-1}$ 

```

collision resistance rather than on preimage resistance. We therefore propose a one-round design in the stateful setting.

ASPEN produces stateful one-round signatures by either of two routes.

- Departing from XMSS [RFC8391] and LMS [RFC8554], it uses a $2n$ -bit message digest for n -bit security. The chain secrets remain n -bit hashes, since preimage resistance still governs them.
- Keeping XMSS and LMS unchanged, it fixes the randomizer r to zero and takes their 256-bit parameter sets for 128-bit security.

The secret sharing can be carried out exactly as in Haystack [Haystack-PW; KLL25]. To propose parameters for the medium and large profiles, Vandermonde secret sharing can be used for those profiles, at the cost of restricting the security guarantee to an honest majority.

2.3. System Model

All of our techniques assume a trusted setup: a dealer generates the secrets, distributes the shares or the seeds, publishes the public material, and deletes everything else. Beyond that, no party is trusted. In particular the aggregator holds only public data, so it is not an additional trusted party and its role may be played by one of the trustees. When the public material is small enough to be replicated, every trustee can hold a copy of it, and the aggregation can be carried out by whichever trustee responds last, which removes the distinguished role altogether.

The two sharing constructions differ in the access structures they support. Replicated secret sharing accommodates an arbitrary monotone access structure, since the dealer is free to choose which coalitions receive which seeds; the compact versions therefore extend beyond threshold profiles to the distributed setting. Vandermonde sharing admits no such freedom, so the scalable versions are confined to threshold profiles. Consequently TALH Compact and ASPEN support the distributed setting, whereas TALH Scalable targets threshold profiles only.

2.4. Security

For TALH we intend to propose parameters at security categories 1, 3 and 5, in both S1 and N1.5. The N1.5 instantiation deserves an explicit caveat: although its signatures are verified by the unmodified SLH-DSA verifier, its security model does not inherit the 2^{64} signatures per key that SLH-DSA tolerates, and assumes a far smaller number instead. For ASPEN we intend to propose parameters at categories 1, 3 and 5 in S1, but only at category 1 in N1.5, since the deterministic message digest requires a 256-bit parameter set to reach that level.

3. Open-Source Implementation

The core protocol will be implemented in C. A public Git repository will be made available.

4. Experimental Performance Evaluation

Our implementation will target the NIST baseline platform (single machine, 16 cores, 64 GB of RAM).

The number of signatures a key may produce, the size of the common reference value, and the work done by each trustee can be traded against one another. For both ASPEN and TALH we therefore aim to give two kinds of parameters: one supporting a small number of signatures and small enough to suit smart cards, and one aimed at more capable devices, which admits a correspondingly larger signing capacity.

Every proposed parameter set will be evaluated with respect to signature size, communication bandwidth, and cycle counts for key generation, signing and verification.

5. 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.

6. Comparison with Other Submissions

The closest submission we are aware of is Haystack [Haystack-PW], whose underlying techniques are published in [KLL25], and which turns the stateful hash-based signature standards LMS and XMSS into threshold and distributed schemes. Like this work, it uses only symmetric primitives, a trusted dealer, and a large common reference value.

ASPEN versus Haystack. Both are stateful; the difference is interaction. Haystack signs in two round trips, since the randomizer and its check values are reconstructed before the one-time signature shares are released. ASPEN uses no randomizer, or equivalently a constant one, which removes the first exchange and leaves a single round. ASPEN also admits the Vandermonde sharing of scalable TALH, which makes it viable for large numbers of parties but changes the security model: an adversary corrupting all but one trustee can attack the state [KLL25], so the scalable variant requires an honest majority. Haystack, for its part, retains the advantage of producing signatures that verify under the standard XMSS and LMS verifiers at the security level of the underlying parameter set, whereas the first of the two ASPEN routes departs from those standards and the second attains 128-bit security only from a 256-bit parameter set.

TALH versus Haystack. TALH is stateless, Haystack stateful, at the cost of one extra round: Haystack signs in two, TALH in three, since the randomizer must be settled by commit–then–reveal in every session. The two also scale differently. Haystack gives each coalition its own shard of one-time keys, so the $\binom{\ell}{k}$ coalitions divide a fixed budget of D keys and the signatures available per coalition collapse as ℓ grows, which is why it gives parameters only for small party counts. Compact TALH shares that limitation, its seed material scaling with $\binom{\ell}{k-1}$; scalable TALH does not, since Vandermonde sharing costs $O(\ell k)$ and the signature budget is independent of the coalition structure. We therefore expect to give parameters for the *medium* and *large* party-count profiles of the Threshold Call [NIST-IR8214C].

Comparison with Other Post-Quantum Proposals. Our main advantage over lattice- and multivariate-based proposals is that hash-based signatures rest on a hash function alone. The price is operational. A stateful scheme makes state management security-critical, while a stateless one must materialize the whole tree at key generation, and bounds a TALH key to some 2^{10} to 2^{20} signatures.

References

[FIPS205] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard*. NIST Federal Information Processing Standards Publication 205. August 2024. doi: [10.6028/NIST.FIPS.205](https://doi.org/10.6028/NIST.FIPS.205).

[Haystack-PW] John Kelsey, Stefan Lucks, and Nathalie Lang. *Haystack: Threshold and Distributed Stateful Hash-Based Signatures*. Preview Writeup (version 0.1) submitted to NIST-MPTC. January 2026.

[KLL25] John Kelsey, Nathalie Lang, and Stefan Lucks. “Turning hash-based signatures into distributed signatures and threshold signatures”. In: *IACR Communications in Cryptology* 2.2 (2025). doi: [10.62056/a6ksudy6b](https://doi.org/10.62056/a6ksudy6b).

[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).

[RFC8391] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. *XMSS: eXtended Merkle Signature Scheme*. IRTF RFC 8391. May 2018. doi: [10.17487/RFC8391](https://doi.org/10.17487/RFC8391).

[RFC8554] David McGrew, Michael Curcio, and Scott Fluhrer. *Leighton-Micali Hash-Based Signatures*. IRTF RFC 8554. April 2019. doi: [10.17487/RFC8554](https://doi.org/10.17487/RFC8554).