

TALH

Threshold **A**uthentication via **L**eaf-sharing in **H**ash-based Signatures

Muhammed Said Gündoğan

TÜBİTAK BİLGEM

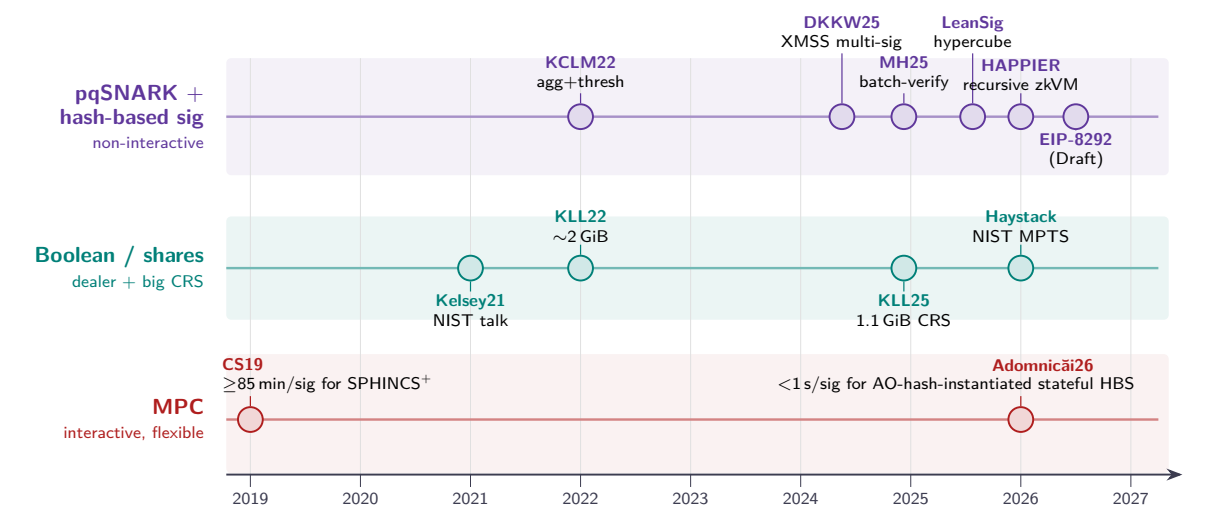
October 6, 2026

Table of Contents

1. Threshold HBS so far: Three Lines of Work
2. Stateful Hash-Based Signatures (HBS)
3. TALH: Stateless Threshold Signatures

Outline

1. Threshold HBS so far: Three Lines of Work
2. Stateful Hash-Based Signatures (HBS)
3. TALH: Stateless Threshold Signatures



pqSNARK + HBS

KCLM22 Khaburzaniya+ ASIACCS'22
MH25 de Meneses+ SBSeg'25
DKKW25 Drake+ IACR CiC 2(1)'25
LeanSig Drake+ ePrint 2025/1332
HAPPIER Saygan+ LightSec'25
EIP-8292 Ethereum Improvement Proposal (Draft)

Boolean / shares

Kelsey21 Kelsey NIST talk'21
KLL22 Kelsey+ ePrint 2022/241
KLL25 Kelsey+ IACR CiC 2(2)'25
Haystack Kelsey+ NIST MPTS'26

MPC

CS19 Cozzo+ IMACC'19 (ePrint 2019/1060)
Adomnicăi26 IACR CiC 2(4)'26 (ePrint 2025/2290)

Threshold Hash-Based Signatures: Comparison

	pqSNARK + HBS	Haystack (KLL25)	Adomnicăi (MPC)

Threshold Hash-Based Signatures: Comparison

	pqSNARK + HBS	Haystack (KLL25)	Adomnicăi (MPC)
Distributed keygen	✓✓ no DKG needed (independent keys)	✗ no (dealer-centric)	✓ yes (no dealer)
Trust / security	HBS + SNARK soundness (+ AO hash)	✓ HBS + PRF	HBS + MPC + AO hash

Threshold Hash-Based Signatures: Comparison

	pqSNARK + HBS	Haystack (KLL25)	Adomnicăi (MPC)
Distributed keygen	✓✓ no DKG needed (independent keys)	✗ no (dealer-centric)	✓ yes (no dealer)
Trust / security	HBS + SNARK soundness (+ AO hash)	✓ HBS + PRF	HBS + MPC + AO hash
Online interaction	✓ zero among signers	2 rounds per trustee	✗ many: $\delta(w-1)$ rounds*
Speed	✗ agg. 2.5–19.7 s (128–1024 sigs)	✓ fast	✓ $<$1 s/sig vs CS19 $\sim$85 min

* amortized to $\delta(w-1)/b$ by batching b OTS.

Threshold Hash-Based Signatures: Comparison

	pqSNARK + HBS	Haystack (KLL25)	Adomnicăi (MPC)
Distributed keygen	✓✓ no DKG needed (independent keys)	✗ no (dealer-centric)	✓ yes (no dealer)
Trust / security	HBS + SNARK soundness (+ AO hash)	✓ HBS + PRF	HBS + MPC + AO hash
Online interaction	✓ zero among signers	2 rounds per trustee	✗ many: $\delta(w-1)$ rounds*
Speed	✗ agg. 2.5–19.7 s (128–1024 sigs)	✓ fast	✓ <1 s/sig vs CS19 ∼85 min
Sig size & format	✗ proof 100 KB–3 MB, non-standard	✓✓ standard LMS/XMSS	standard sign/verify, but Poseidon2 (not an RFC/NIST std)
Aggregator storage	✓ none (pubkeys only)	✗ CRV ∼1.1 GiB (up to TiB)	✓ none (no aggregator)

* amortized to $\delta(w-1)/b$ by batching b OTS.

Threshold Hash-Based Signatures: Comparison

	pqSNARK + HBS	Haystack (KLL25)	Adomnicăi (MPC)
Distributed keygen	✓✓ no DKG needed (independent keys)	✗ no (dealer-centric)	✓ yes (no dealer)
Trust / security	HBS + SNARK soundness (+ AO hash)	✓ HBS + PRF	HBS + MPC + AO hash
Online interaction	✓ zero among signers	2 rounds per trustee	✗ many: $\delta(w-1)$ rounds*
Speed	✗ agg. 2.5–19.7 s (128–1024 sigs)	✓ fast	✓ $<$1 s/sig vs CS19 $\sim$85 min
Sig size & format	✗ proof 100 KB–3 MB, non-standard	✓✓ standard LMS/XMSS	standard sign/verify, but Poseidon2 (not an RFC/NIST std)
Aggregator storage	✓ none (pubkeys only)	✗ CRV $\sim$1.1 GiB (up to TiB)	✓ none (no aggregator)
Threshold scaling	✓ scales to many signers	✗ poor: $\binom{n}{k}$ coalitions, $n \lesssim 10$	✓ scales to many signers

* amortized to $\delta(w-1)/b$ by batching b OTS.

Threshold Hash-Based Signatures: Comparison

	pqSNARK + HBS	Haystack (KLL25)	Adomnicăi (MPC)
Distributed keygen	✓✓ no DKG needed (independent keys)	✗ no (dealer-centric)	✓ yes (no dealer)
Trust / security	HBS + SNARK soundness (+ AO hash)	✓ HBS + PRF	HBS + MPC + AO hash
Online interaction	✓ zero among signers	2 rounds per trustee	✗ many: $\delta(w-1)$ rounds*
Speed	✗ agg. 2.5–19.7 s (128–1024 sigs)	✓ fast	✓ <1 s/sig vs CS19 ∼85 min
Sig size & format	✗ proof 100 KB–3 MB, non-standard	✓✓ standard LMS/XMSS	standard sign/verify, but Poseidon2 (not an RFC/NIST std)
Aggregator storage	✓ none (pubkeys only)	✗ CRV ∼1.1 GiB (up to TiB)	✓ none (no aggregator)
Threshold scaling	✓ scales to many signers	✗ poor: $\binom{n}{k}$ coalitions, $n \lesssim 10$	✓ scales to many signers

stateful

* amortized to $\delta(w-1)/b$ by batching b OTS.

Outline

1 Threshold HBS so far: Three Lines of Work

2 Stateful Hash-Based Signatures (HBS)

3 TALH: Stateless Threshold Signatures

Stateful Hash-Based Signatures (HBS)

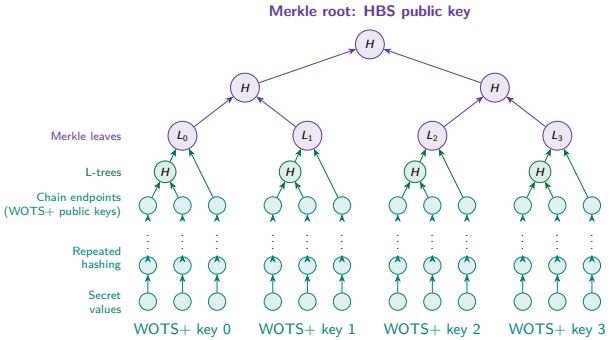
- Post-quantum and very efficient
  - One assumption only: a secure hash function

Stateful Hash-Based Signatures (HBS)

- Post-quantum and very efficient
  - One assumption only: a secure hash function
- Built from a one-time signature (OTS) scheme and a Merkle tree
  - Generate D OTS key pairs
  - Use their public keys to construct the Merkle leaves
  - The HBS public key includes the Merkle root
  - Sign at most once with each OTS key

Key reuse is a disaster that allows forgeries!

Signing simply means to **reveal** selected chain values and Merkle authentication nodes.

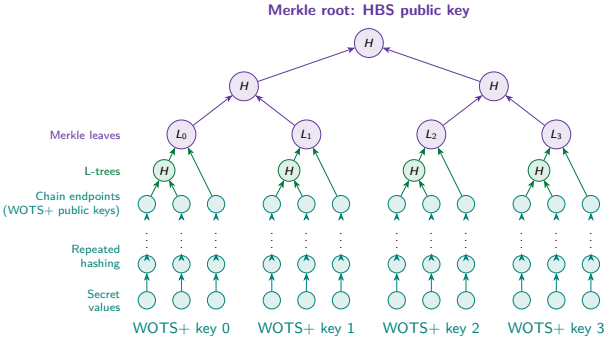


Haystack: XOR Sharing

1. **XOR sharing:** XOR-share all values in this tree among n parties. Together, they XOR their shares to reconstruct the values needed for a signature.

Do trustees need so much memory? Can their secret keys be smaller?

Signing simply means to **reveal** selected chain values and Merkle authentication nodes.



Haystack: Reducing Trustee Storage

2. **Aggregator idea:** Use $n + 1$ XOR shares instead of n.

The trick

For every value X to be shared, trustee T_t's share is *deterministically* derived from its PRF key:

$$X^{(t)} = \mathrm{PRF}^{\mathrm{LABEL}}_{K[t]}(\mathrm{tag}_X)$$

Then the dealer picks X^{CRV} to be whatever XOR-fixes things:

$$X^{\mathrm{CRV}} = X \oplus X^{(1)} \oplus \cdots \oplus X^{(n)}.$$

Trustee stores the PRF key only. The aggregator stores the **public CRV**, which absorbs the "fix-up."

Haystack: The Randomiser

Randomized message hashing relies on **second-preimage resistance** rather than collision resistance. For λ-bit *classical* security, the OTS can sign a λ-bit digest instead of a 2λ-bit digest (generic hash-attack bounds).

Haystack: The Randomiser

Randomized message hashing relies on **second-preimage resistance** rather than collision resistance. For λ-bit *classical* security, the OTS can sign a λ-bit digest instead of a 2λ-bit digest (generic hash-attack bounds).

The challenge

R must remain unpredictable to any $n-1$ trustees until **all** n **trustees have seen the message**. The aggregator must not be able to substitute a different R.

Haystack: The Randomiser

Randomized message hashing relies on **second-preimage resistance** rather than collision resistance. For λ-bit *classical* security, the OTS can sign a λ-bit digest instead of a 2λ-bit digest (generic hash-attack bounds).

The challenge

R must remain unpredictable to any $n-1$ trustees until **all** n **trustees have seen the message**. The aggregator must not be able to substitute a different R.

A commit–reveal phase would take two rounds, followed by one round to reveal the XOR signature shares: **three rounds in total**.

The trick: Fix and share R at setup

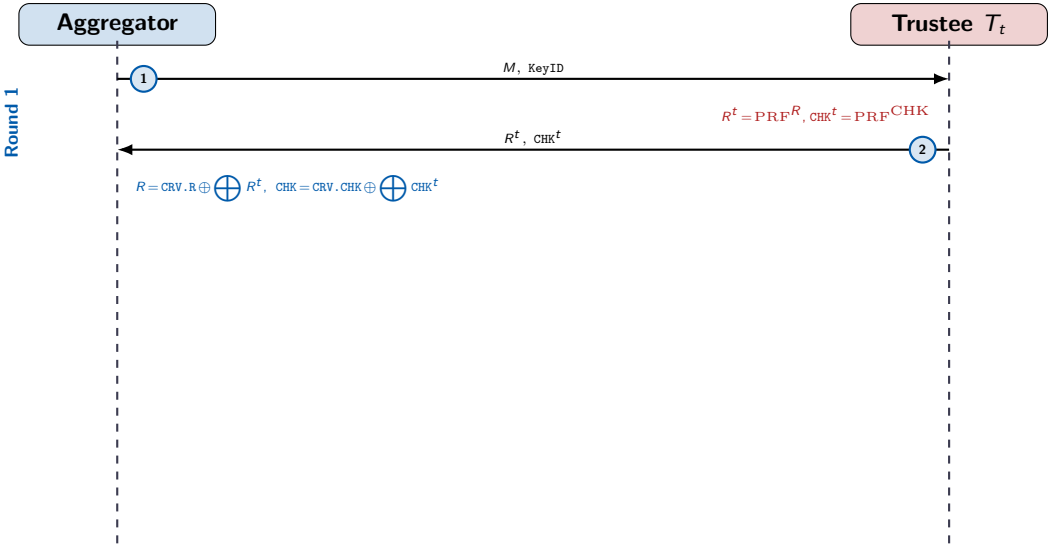
- The dealer chooses R for each `KeyID` and stores its remaining XOR share in the public CRV.
- After receiving the message, each trustee derives its share from its PRF key. The aggregator reconstructs R:

$$R^{(t)} = \mathrm{PRF}^{R}_{K[t]}(\texttt{KeyID}), \qquad R = R^{(1)} \oplus \cdots \oplus R^{(n)} \oplus \texttt{CRV.R}.$$

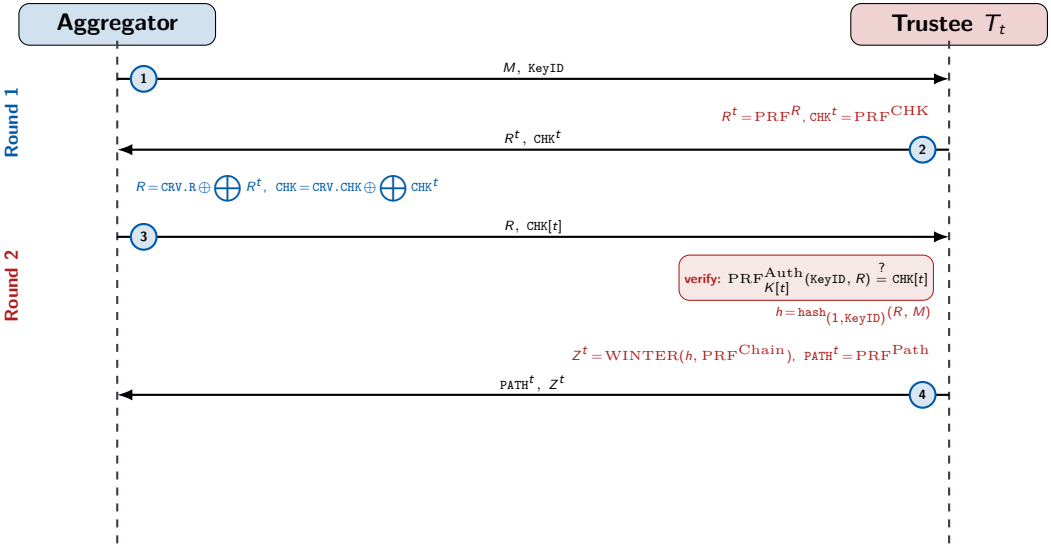
- Trustees also provide shares of a check value. Each trustee checks the returned R before releasing signature shares.

Two round trips: First establish R and its check value; then check R and produce the signature shares.

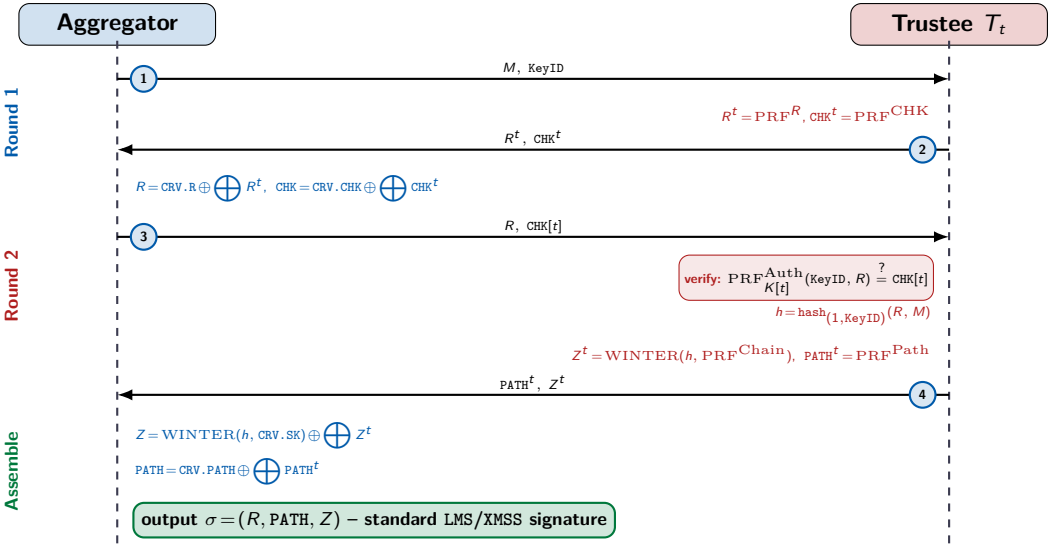
Signing protocol: two round-trips



Signing protocol: two round-trips



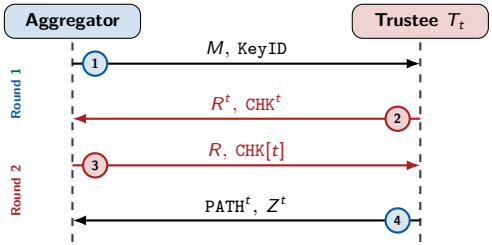
Signing protocol: two round-trips



Can we sign in one round?

Why two round-trips?

R buys **second-preimage** instead of collision resistance, halving the digest: λ bits instead of 2λ.



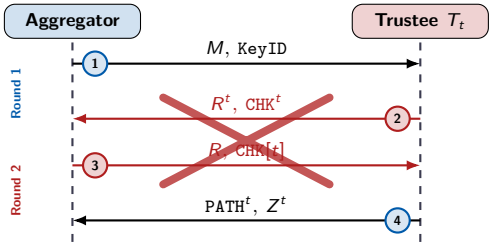
Can we sign in one round?

Why two round-trips?

R buys **second-preimage** instead of collision resistance, halving the digest: λ bits instead of 2λ.

ASPEN: drop the randomiser

Choose R as a constant (e.g. $R = 0$).



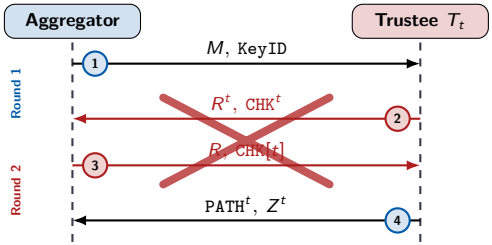
Can we sign in one round?

Why two round-trips?

R buys **second-preimage** instead of collision resistance, halving the digest: λ bits instead of 2λ.

ASPEN: drop the randomiser

Choose R as a constant (e.g. $R = 0$).



The main price: signature size

	Digest	# chains	Chain values	Signature	Std. verifier	Categories
Haystack (fresh R)	λ	1×	λ	1×	✓	1, 3, 5
ASPEN, S1	2λ	2×	λ	**2**×	✗	1, 3, 5
ASPEN, N1.5 ($r = 0$)	2λ	2×	2λ	**4**×	✓	1 only*

* 128-bit security needs the nominally 256-bit parameter set.
Category 5 is reachable with the RFC 8391 512-bit sets (e.g. `XMSS-SHA2_10_512`), which SP 800-208 does not approve.

From n-out-of-n to k-out-of-n

We have built an n-**out-of-**n scheme.

How do we get k-out-of-n?

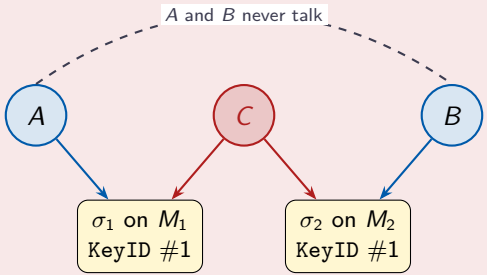
From n-out-of-n to k-out-of-n

We have built an n-**out-of-**n scheme.

How do we get k-out-of-n?

The risk: key reuse

2-of-3 scheme, C is corrupt



C holds two signatures under one OTS key $\Rightarrow$ forgery

Haystack: Sharding

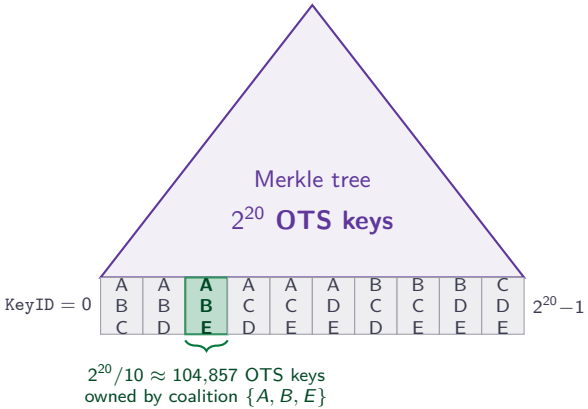
3. **Sharding:** each `KeyID` is owned by **exactly one** signing coalition, so no key can be reused across coalitions.
   - k-of-n threshold scheme
   - practical for $n \lesssim 10$

Haystack: Sharding

3. **Sharding:** each `KeyID` is owned by **exactly one** signing coalition, so no key can be reused across coalitions.
   - k-of-n threshold scheme
   - practical for $n \lesssim 10$

Trustees A, B, C, D, E in a 3-of-5 scheme

Allocate a range of `KeyIDs` to each coalition: $\binom{5}{3} = 10$ coalitions.



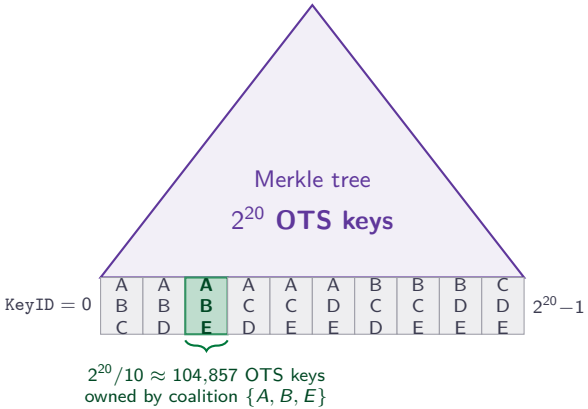
Haystack: Sharding

3. **Sharding:** each `KeyID` is owned by **exactly one** signing coalition, so no key can be reused across coalitions.
   - k-of-n threshold scheme
   - practical for $n \lesssim 10$

Every coalition needs its own shard: $\binom{n}{k}$ of them. Is there a scalable way?

Trustees A, B, C, D, E in a 3-of-5 scheme

Allocate a range of `KeyID`s to each coalition: $\binom{5}{3} = 10$ coalitions.



Scaling up: Shamir-share everything

an idea we are still evaluating

1. **Shamir sharing:** share *every* secret of the tree among all n trustees with a k-of-n Shamir sharing over $\mathrm{GF}(2^{\delta})$.
   - one key pool: all D `KeyID`s serve every coalition
   - storage independent of n

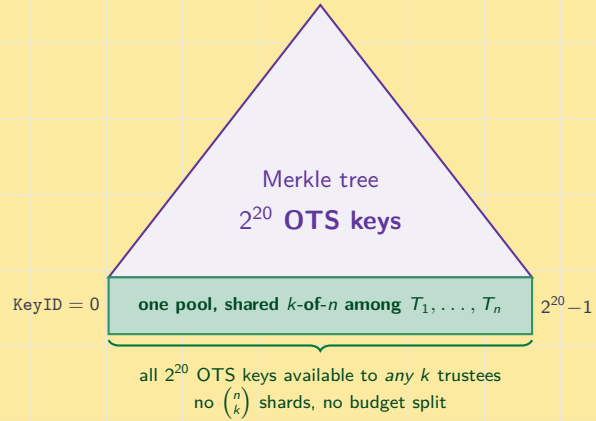
Scaling up: Shamir-share everything

an idea we are still evaluating

- **Shamir sharing:** share *every* secret of the tree among all n trustees with a k-of-n Shamir sharing over $\mathrm{GF}(2^\delta)$.
  - one key pool: all D `KeyID`s serve every coalition
  - storage independent of n

Trustees $T_1, \ldots, T_n$ in a k-of-n scheme

No shards: every `KeyID` is shared among all n trustees, and any k of them reconstruct it.



Scaling up: Shamir-share everything

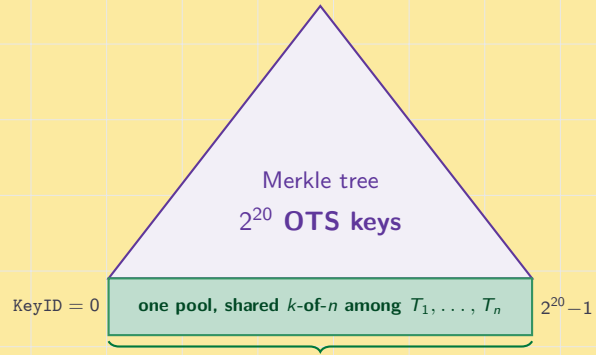
an idea we are still evaluating

1. **Shamir sharing:** share *every* secret of the tree among all n trustees with a k-of-n Shamir sharing over $\mathrm{GF}(2^\delta)$.
   - one key pool: all D `KeyID`s serve every coalition
   - storage independent of n

Any k trustees can now use any `KeyID`. Is the key-reuse risk back?

Trustees $T_1, \ldots, T_n$ in a k-of-n scheme

No shards: every `KeyID` is shared among all n trustees, and any k of them reconstruct it.

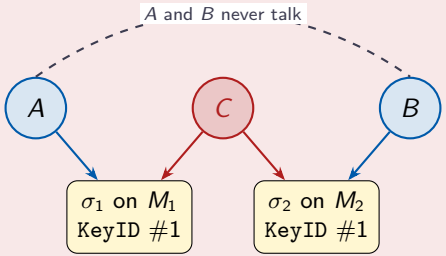


The risk is back

Every `KeyID` now serves every coalition, and two coalitions need not share an honest trustee: $2k - n$ **corrupt trustees can force a key reuse** (KLL25, §4.1).

The risk: key reuse

2-of-3 scheme, C is corrupt



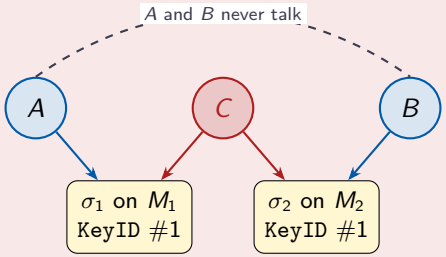
C holds two signatures under one OTS key $\Rightarrow$ forgery

The risk is back

Every `KeyID` now serves every coalition, and two coalitions need not share an honest trustee: $2k - n$ **corrupt trustees can force a key reuse** (KLL25, §4.1).

The risk: key reuse

2-of-3 scheme, C is corrupt



C holds two signatures under one OTS key $\Rightarrow$ forgery

The way out: make sure that *every two* signing coalitions share at least one honest trustee. That trustee sees both requests and refuses the second one.

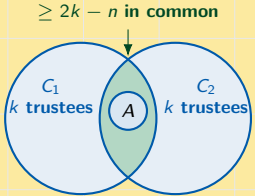
A high threshold buys safety

Safety condition

Two coalitions of size k overlap in at least $2k - n$ trustees. With at most

$$f \leq 2k - n - 1$$

corrupt trustees, every overlap contains an honest one, so no `KeyID` is ever used twice.



honest A sits in both:
refuses the second use of the `KeyID`

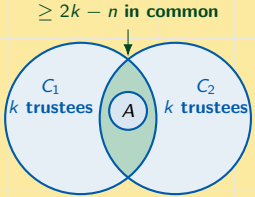
A high threshold buys safety

Safety condition

Two coalitions of size k overlap in at least $2k - n$ trustees. With at most

$$f \leq 2k - n - 1$$

corrupt trustees, every overlap contains an honest one, so no `KeyID` is ever used twice.



Use case: an approval mechanism with threshold 80%

if 40% of the participants are honest → **then** $f \leq 0.6n - 1$ corrupt trustees → **so** $k = 0.8n$ is safe: an 80% approval quorum is feasible

- Applications??

The price of scalability: trust

	Sharding (Haystack)	**Shamir sharing**
Unforgeable if	$f \leq k - 1$	$f \leq 2k - n - 1$
Available if	$f \leq n - k$	$f \leq n - k$

The price of scalability: trust

	Sharding (Haystack)	Shamir sharing
Unforgeable if	$f \leq k-1$	$f \leq 2k-n-1$
Available if	$f \leq n-k$	$f \leq n-k$

Corruption proportion		Number of parties (n)					
f/n	Majority type	Two (2) $n=2$	Three (3) $n=3$	Small (S) $4 \leq n \leq 8$	Medium (M) $9 \leq n \leq 64$	Large (L) $65 \leq n \leq 1024$	Enormous (E) $n \geq 1025$
$\geq 1/2$	Dishonest (D)	$n2$	$n3fD$	$nSfD$	$nMfD$	$nLfD$	$nEfD$
$> 1/3$	Honest (h)	—	$n3fh$	$nSfh$	$nMfh$	$nLfh$	$nEfh$
$< 1/3$	2/3 Honest (H)	—	—	$nSfH$	$nMfH$	$nLfH$	$nEfH$

NIST IR 8214C, Table 17: template threshold profiles (n parties, f corrupt; k = participation threshold).

The price of scalability: trust

	Sharding (Haystack)	8214C label	Shamir sharing	8214C label
Unforgeable if	$f \le k-1$		$f \le 2k-n-1$	
Available if	$f \le n-k$		$f \le n-k$	
$k = 0.8n$	$f \le 0.8n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.6n-1$ (40% honest)	D: $n\mathrm{M}f\mathrm{D}$, $n\mathrm{L}f\mathrm{D}$
$k = 0.7n$	$f \le 0.7n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.4n-1$ (60% honest)	h: $n\mathrm{M}f\mathrm{h}$, $n\mathrm{L}f\mathrm{h}$
$k = 0.6n$	$f \le 0.6n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.2n-1$ (80% honest)	H: $n\mathrm{M}f\mathrm{H}$, $n\mathrm{L}f\mathrm{H}$
$k = 0.5n$	$f \le 0.5n-1$	h: $n\mathrm{S}f\mathrm{h}$	$f \le -1$ (unsafe even if all honest)	–

Corruption proportion		Number of parties (n)					
f/n	Majority type	Two (2) $n=2$	Three (3) $n=3$	Small (S) $4 \le n \le 8$	Medium (M) $9 \le n \le 64$	Large (L) $65 \le n \le 1024$	Enormous (E) $n \ge 1025$
$\ge 1/2$	Dishonest (D)	$n2$	$n3f\mathrm{D}$	$n\mathrm{S}f\mathrm{D}$	$n\mathrm{M}f\mathrm{D}$	$n\mathrm{L}f\mathrm{D}$	$n\mathrm{E}f\mathrm{D}$
$> 1/3$	Honest (h)	—	$n3f\mathrm{h}$	$n\mathrm{S}f\mathrm{h}$	$n\mathrm{M}f\mathrm{h}$	$n\mathrm{L}f\mathrm{h}$	$n\mathrm{E}f\mathrm{h}$
$< 1/3$	2/3 Honest (H)	—	—	$n\mathrm{S}f\mathrm{H}$	$n\mathrm{M}f\mathrm{H}$	$n\mathrm{L}f\mathrm{H}$	$n\mathrm{E}f\mathrm{H}$

NIST IR 8214C, Table 17: template threshold profiles (n parties, f corrupt; k = participation threshold).

The price of scalability: trust

	Sharding (Haystack)	8214C label	Shamir sharing	8214C label
Unforgeable if	$f \le k-1$		$f \le 2k-n-1$	
Available if	$f \le n-k$		$f \le n-k$	
$k = 0.8n$	$f \le 0.8n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.6n-1$ (40% honest)	D: $n\mathrm{M}f\mathrm{D}$, $n\mathrm{L}f\mathrm{D}$
$k = 0.7n$	$f \le 0.7n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.4n-1$ (60% honest)	h: $n\mathrm{M}f\mathrm{h}$, $n\mathrm{L}f\mathrm{h}$
$k = 0.6n$	$f \le 0.6n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.2n-1$ (80% honest)	H: $n\mathrm{M}f\mathrm{H}$, $n\mathrm{L}f\mathrm{H}$
$k = 0.5n$	$f \le 0.5n-1$	h: $n\mathrm{S}f\mathrm{h}$	$f \le -1$ (unsafe even if all honest)	–

Corruption proportion		Number of parties (n)					
f/n	Majority type	Two (2) $n=2$	Three (3) $n=3$	Small (S) $4 \le n \le 8$	Medium (M) $9 \le n \le 64$	Large (L) $65 \le n \le 1024$	Enormous (E) $n \ge 1025$
$\ge 1/2$	Dishonest (D)	$n2$	$n3f\mathrm{D}$	$n\mathrm{S}f\mathrm{D}$	$n\mathrm{M}f\mathrm{D}$	$n\mathrm{L}f\mathrm{D}$	$n\mathrm{E}f\mathrm{D}$
$> 1/3$	Honest (h)	—	$n3f\mathrm{h}$	$n\mathrm{S}f\mathrm{h}$	$n\mathrm{M}f\mathrm{h}$	$n\mathrm{L}f\mathrm{h}$	$n\mathrm{E}f\mathrm{h}$
$< 1/3$	2/3 Honest (H)	—	—	$n\mathrm{S}f\mathrm{H}$	$n\mathrm{M}f\mathrm{H}$	$n\mathrm{L}f\mathrm{H}$	$n\mathrm{E}f\mathrm{H}$

NIST IR 8214C, Table 17: template threshold profiles (n parties, f corrupt; k = participation threshold).

Trade-off: party count vs. trust

Sharding keeps the ideal $f \le k-1$ for $n \lesssim 10$; Shamir reaches M and L by trusting that enough participants are honest.

The price of scalability: trust

	Sharding (Haystack)	8214C label	Shamir sharing	8214C label
Unforgeable if	$f \le k-1$		$f \le 2k-n-1$	
Available if	$f \le n-k$		$f \le n-k$	
$k = 0.8n$	$f \le 0.8n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.6n-1$ (40% honest)	D: $n\mathrm{M}f\mathrm{D}$, $n\mathrm{L}f\mathrm{D}$
$k = 0.7n$	$f \le 0.7n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.4n-1$ (60% honest)	h: $n\mathrm{M}f\mathrm{h}$, $n\mathrm{L}f\mathrm{h}$
$k = 0.6n$	$f \le 0.6n-1$	D: $n\mathrm{S}f\mathrm{D}$	$f \le 0.2n-1$ (80% honest)	H: $n\mathrm{M}f\mathrm{H}$, $n\mathrm{L}f\mathrm{H}$
$k = 0.5n$	$f \le 0.5n-1$	h: $n\mathrm{S}f\mathrm{h}$	$f \le -1$ (unsafe even if all honest)	–

Corruption proportion		Number of parties (n)					
f/n	Majority type	Two (2) $n=2$	Three (3) $n=3$	Small (S) $4 \le n \le 8$	Medium (M) $9 \le n \le 64$	Large (L) $65 \le n \le 1024$	Enormous (E) $n \ge 1025$
$\ge 1/2$	Dishonest (D)	$n2$	$n3f\mathrm{D}$	$n\mathrm{S}f\mathrm{D}$	$n\mathrm{M}f\mathrm{D}$	$n\mathrm{L}f\mathrm{D}$	$n\mathrm{E}f\mathrm{D}$
$> 1/3$	Honest (h)	—	$n3f\mathrm{h}$	$n\mathrm{S}f\mathrm{h}$	$n\mathrm{M}f\mathrm{h}$	$n\mathrm{L}f\mathrm{h}$	$n\mathrm{E}f\mathrm{h}$
$< 1/3$	2/3 Honest (H)	—	—	$n\mathrm{S}f\mathrm{H}$	$n\mathrm{M}f\mathrm{H}$	$n\mathrm{L}f\mathrm{H}$	$n\mathrm{E}f\mathrm{H}$

NIST IR 8214C, Table 17: template threshold profiles (n parties, f corrupt; k = participation threshold).

Trade-off: party count vs. trust

Sharding keeps the ideal $f \le k-1$ for $n \lesssim 10$; Shamir reaches M and L by trusting that enough participants are honest.

Question: the majority type of 8214C changes with the participation threshold k. Which label should such a scheme carry?

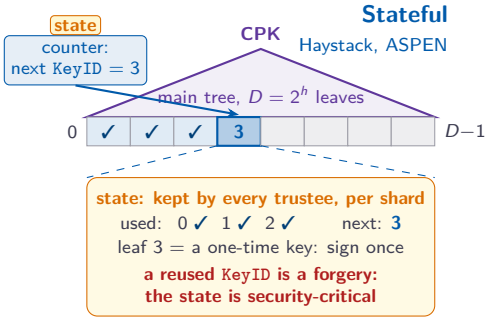
Outline

1 Threshold HBS so far: Three Lines of Work

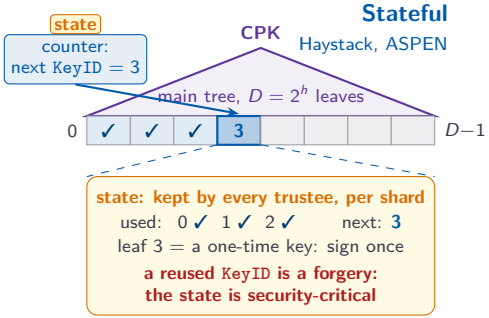
2 Stateful Hash-Based Signatures (HBS)

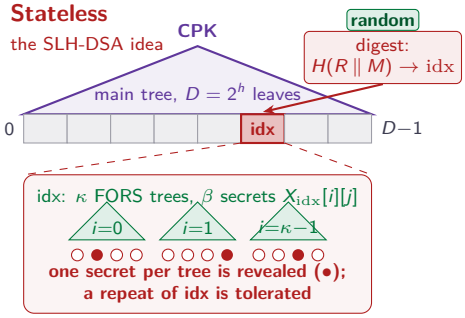
3 TALH: Stateless Threshold Signatures

Can we go stateless?

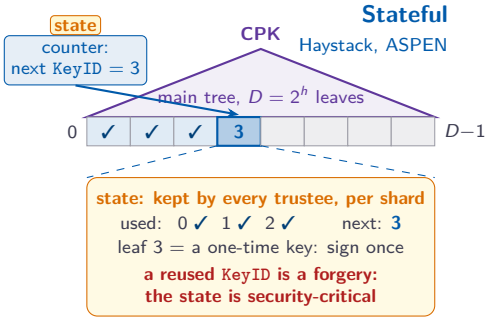


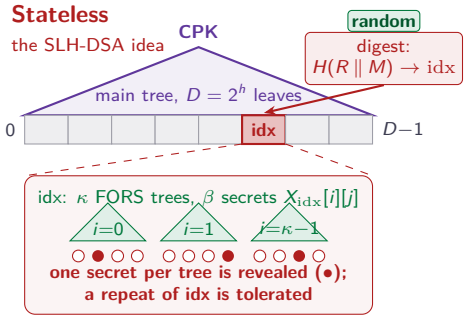
Can we go stateless?





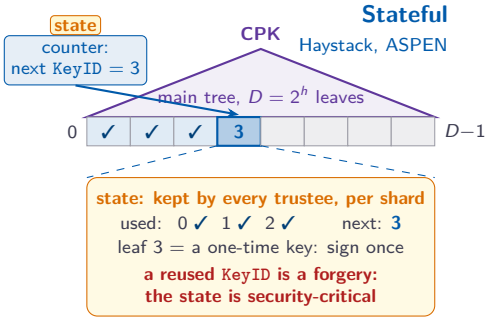
Can we go stateless?

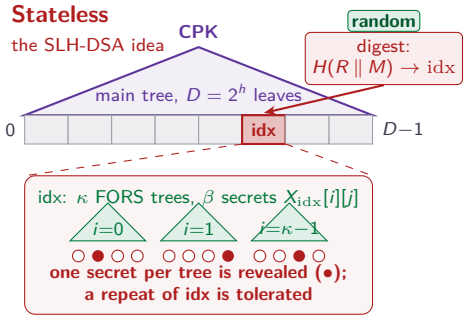




	Stateful	Stateless
Leaf picked by	a counter: the next KeyID	the digest $H(R \| M)$: a random idx
Leaf is	a one-time key: sign once	a few-time FORS instance: a repeat is tolerated
Remembered	the used KeyIDs, by every trustee	nothing

Can we go stateless?





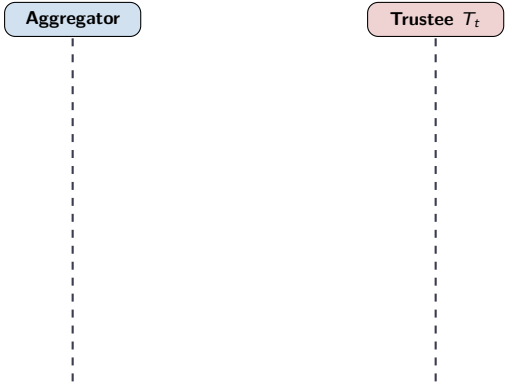
	Stateful	Stateless
Leaf picked by	a counter: the next KeyID	the digest $H(R \| M)$: a random idx
Leaf is	a one-time key: sign once	a few-time FORS instance: a repeat is tolerated
Remembered	the used KeyIDs, by every trustee	nothing

the price: no state means nothing to pin R to, so R must be fresh and unpredictable for every signature (next slide)

The price: a fresh randomiser every time

No slot to pin R to

Haystack stores one R per `KeyID`; a stateless instance has no slot, $H(R \,\|\, M)$ picks it. So R must be unpredictable to everyone, controlled by no one: **coalition** C tosses it together.

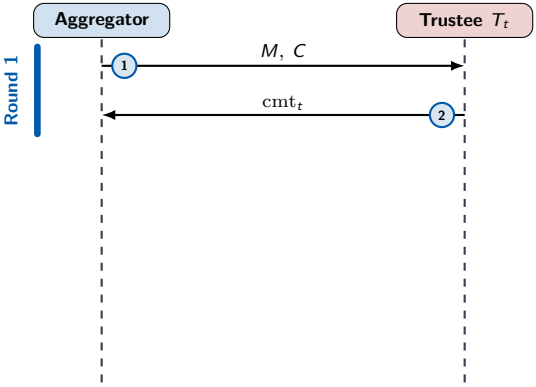


The price: a fresh randomiser every time

No slot to pin R to

Haystack stores one R per `KeyID`; a stateless instance has no slot, $H(R \,\|\, M)$ picks it. So R must be unpredictable to everyone, controlled by no one: **coalition** C tosses it together.

Round 1, commit: toss $R_t \leftarrow_{\$} \{0,1\}^{\lambda}$; send only $\mathrm{cmt}_t = H(t, R_t)$.



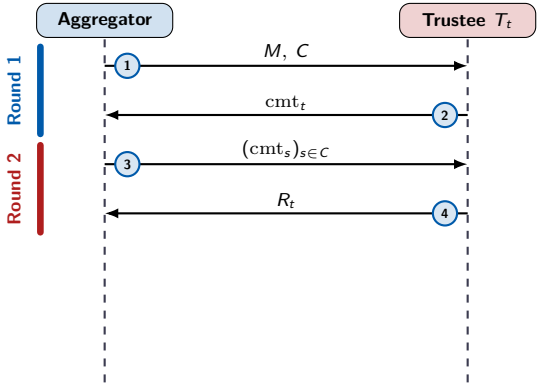
The price: a fresh randomiser every time

No slot to pin R to

Haystack stores one R per `KeyID`; a stateless instance has no slot, $H(R \,\|\, M)$ picks it. So R must be unpredictable to everyone, controlled by no one: **coalition** C tosses it together.

Round 1, commit: toss $R_t \leftarrow_{\$} \{0,1\}^{\lambda}$; send only $\mathrm{cmt}_t = H(t, R_t)$.

Round 2, reveal: all commitments are in; send R_t. The toss can no longer change.



The price: a fresh randomiser every time

No slot to pin R to

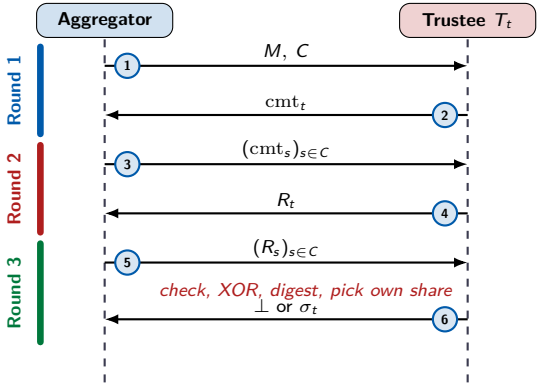
Haystack stores one R per `KeyID`; a stateless instance has no slot, $H(R \,\|\, M)$ picks it. So R must be unpredictable to everyone, controlled by no one: **coalition** C tosses it together.

Round 1, commit: toss $R_t \leftarrow_{\$} \{0,1\}^{\lambda}$; send only $\mathrm{cmt}_t = H(t, R_t)$.

Round 2, reveal: all commitments are in; send R_t. The toss can no longer change.

Round 3, shares:

```
if cmt_s = H(s, R_s) ∀s ∈ C:
    R ← ⊕_{s∈C} R_s
    (idx, c) ← H(R ‖ M)
    release σ_t
else: output ⊥
```



The price: a fresh randomiser every time

No slot to pin R to

Haystack stores one R per `KeyID`; a stateless instance has no slot, $H(R \,\|\, M)$ picks it. So R must be unpredictable to everyone, controlled by no one: **coalition** C tosses it together.

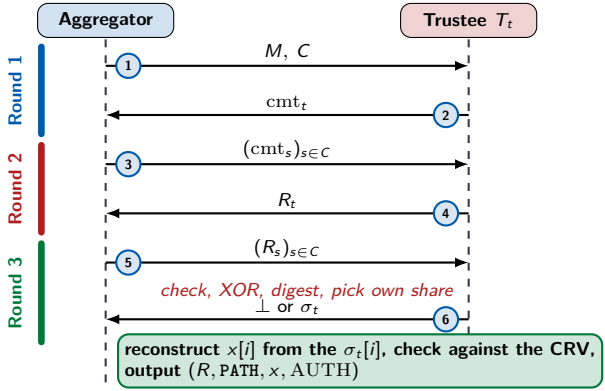
Round 1, commit: toss $R_t \leftarrow_{\$} \{0,1\}^{\lambda}$; send only $\mathrm{cmt}_t = H(t, R_t)$.

Round 2, reveal: all commitments are in; send R_t. The toss can no longer change.

Round 3, shares:

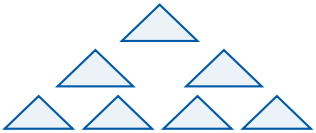
```
if cmt_s = H(s, R_s) ∀s ∈ C:
    R ← ⊕_{s∈C} R_s
    (idx, c) ← H(R ‖ M)
    release σ_t
else: output ⊥
```

Three rounds, two of them for R.



One tree, built in full: SLH-DSA vs. TALH

SLH-DSA: hypertree

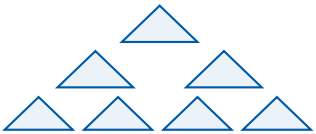


2^{64} **leaves, derived from** `SK.seed`

	SLH-DSA
Key generation	the top root only
At signing	rebuild nodes from `SK.seed`
Signatures per key	2^{64}
Verifier	SLH-DSA

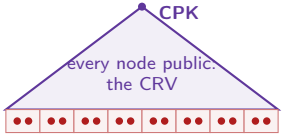
One tree, built in full: SLH-DSA vs. TALH

SLH-DSA: hypertree



2^{64} **leaves, derived from** `SK.seed`

TALH, S1: one tree



$D = 2^h$ **leaves,** $h = 10 \ldots 20$**: every secret shared**

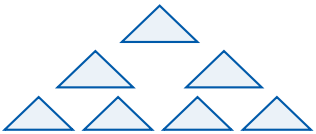
	SLH-DSA	TALH, S1
Key generation	the top root only	every secret and node: the CRV
At signing	rebuild nodes from `SK.seed`	no rebuild: read the CRV
Signatures per key	2^{64}	$2^{10} \ldots 2^{20}$
Verifier	SLH-DSA	TALH

Smaller SPHINCS$^+$ regime

Fluhrer–Dang (ePrint 2024/018); draft SP 800-230 allows 2^{24} signatures per key.
Few signatures per key buy smaller, faster signatures.

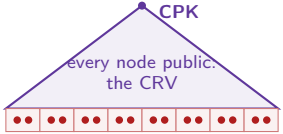
One tree, built in full: SLH-DSA vs. TALH

SLH-DSA: hypertree



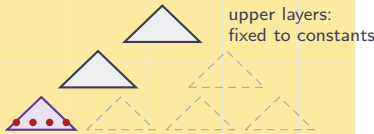
2^{64} leaves, derived from `SK.seed`

TALH, S1: one tree



$D = 2^h$ leaves, $h = 10 \ldots 20$: every secret share

TALH, N1.5: under SLH-DSA



lower layers: TALH, 2^h live leaves

	SLH-DSA	TALH, S1	TALH, N1.5
Key generation	the top root only	every secret and node: the CRV	S1 under constant upper layers
At signing	rebuild nodes from `SK.seed`	no rebuild: read the CRV	the same
Signatures per key	2^{64}	$2^{10} \ldots 2^{20}$	$2^{10} \ldots 2^{20}$
Verifier	SLH-DSA	TALH	unmodified SLH-DSA

Smaller SPHINCS$^+$ regime

Fluhrer–Dang (ePrint 2024/018); draft SP 800-230 allows 2^{24} signatures per key.
Few signatures per key buy smaller, faster signatures.

SP 800-230 parameters

NIST SP 800-230 (Initial Public Draft)

Additional SLH-DSA Parameter Sets for Limited Signature Use Cases

Documentation | Topics

Date Published: 04/13/2026
Comments Due: June 12, 2026 (public comment period is CLOSED)
Email Questions to: SP800-230-comments@nist.gov

Planning Note (07/01/2026):

The public comments received are now available.

SP 800-230 ipd: SLH-DSA at the 2^{24} signature limit

	n	h	d	h'	a	k	lg_w	m	security category	pk bytes	sig bytes
SLH-DSA-SHA2-128-24 SLH-DSA-SHAKE-128-24	16	22	1	22	24	6	2	21	1	32	3,856
SLH-DSA-SHA2-192-24 SLH-DSA-SHAKE-192-24	24	21	1	21	25	9	3	32	3	48	7,752
SLH-DSA-SHA2-256-24 SLH-DSA-SHAKE-256-24	32	21	1	21	25	12	2	41	5	64	14,944

Table 1: SLH-DSA parameter sets at 2^{24} signature limit per signing key. The parameters $n, h, d, h', a, k, lg_w, m$ are defined in Section 11 of FIPS 205.

image from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-230.ipd.pdf

SP 800-230 parameters

NIST SP 800-230 (Initial Public Draft)

Additional SLH-DSA Parameter Sets for Limited Signature Use Cases

Documentation | Topics

Date Published: 04/13/2026
Comments Due: June 12, 2026 (public comment period is CLOSED)
Email Questions to: SP800-230-comments@nist.gov

Planning Note (07/01/2026):

The public comments received are now available.

SP 800-230 ipd: SLH-DSA at the 2^{24} signature limit

	n	h	d	h'	a	k	lg_w	m	security category	pk bytes	sig bytes
SLH-DSA-SHA2-128-24 SLH-DSA-SHAKE-128-24	16	22	1	22	24	6	2	21	1	32	3,856
SLH-DSA-SHA2-192-24 SLH-DSA-SHAKE-192-24	24	21	1	21	25	9	3	32	3	48	7,752
SLH-DSA-SHA2-256-24 SLH-DSA-SHAKE-256-24	32	21	1	21	25	12	2	41	5	64	14,944

Table 1: SLH-DSA parameter sets at 2^{24} signature limit per signing key. The parameters $n, h, d, h', a, k, lg_w, m$ are defined in Section 11 of FIPS 205.

image from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-230.ipd.pdf

Plan

Instantiate TALH with these sets.

SP 800-230 parameters

NIST SP 800-230 (Initial Public Draft)

Additional SLH-DSA Parameter Sets for Limited Signature Use Cases

Documentation | Topics

Date Published: 04/13/2026
Comments Due: June 12, 2026 (public comment period is CLOSED)
Email Questions to: SP800-230-comments@nist.gov

Planning Note (07/01/2026):

The public comments received are now available.

SP 800-230 ipd: SLH-DSA at the 2^{24} signature limit

	n	h	d	h'	a	k	lg_w	m	security category	pk bytes	sig bytes
SLH-DSA-SHA2-128-24 SLH-DSA-SHAKE-128-24	16	22	1	22	24	6	2	21	1	32	3,856
SLH-DSA-SHA2-192-24 SLH-DSA-SHAKE-192-24	24	21	1	21	25	9	3	32	3	48	7,752
SLH-DSA-SHA2-256-24 SLH-DSA-SHAKE-256-24	32	21	1	21	25	12	2	41	5	64	14,944

Table 1: SLH-DSA parameter sets at 2^{24} signature limit per signing key. The parameters $n, h, d, h', a, k, lg_w, m$ are defined in Section 11 of FIPS 205.

image from https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-230.ipd.pdf

Plan

Instantiate TALH with these sets.

The question

Is TALH then an N1.5 scheme?

Sharing the FORS secrets, 1: replicated seeds

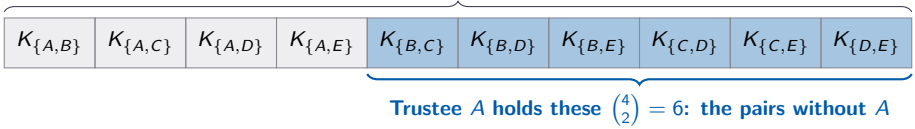
3-of-5: $k-1=2$, so $\mathfrak{B}$ runs over the $\binom{5}{2}=10$ pairs: one seed $K_{\mathfrak{B}}$ each

$K_{\{A,B\}}$	$K_{\{A,C\}}$	$K_{\{A,D\}}$	$K_{\{A,E\}}$	$K_{\{B,C\}}$	$K_{\{B,D\}}$	$K_{\{B,E\}}$	$K_{\{C,D\}}$	$K_{\{C,E\}}$	$K_{\{D,E\}}$

Who gets $K_{\mathfrak{B}}$? $K_{\mathfrak{B}}$ goes to all trustees except the members of $\mathfrak{B}$.

Sharing the FORS secrets, 1: replicated seeds

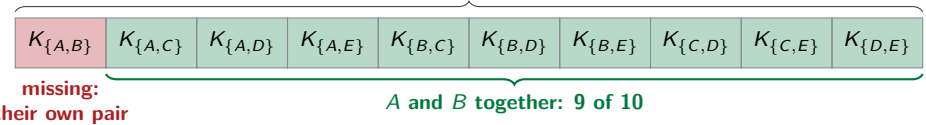
3-of-5: $k-1=2$, so $\mathfrak{B}$ runs over the $\binom{5}{2}=10$ pairs: one seed $K_{\mathfrak{B}}$ each



Who gets $K_{\mathfrak{B}}$? $K_{\mathfrak{B}}$ goes to all trustees except the members of $\mathfrak{B}$.

Sharing the FORS secrets, 1: replicated seeds

3-of-5: $k-1=2$, so $\mathfrak{B}$ runs over the $\binom{5}{2}=10$ pairs: one seed $K_{\mathfrak{B}}$ each



Who gets $K_{\mathfrak{B}}$? $K_{\mathfrak{B}}$ goes to all trustees except the members of $\mathfrak{B}$.

Sharing the FORS secrets, 1: replicated seeds

3-of-5: $k-1=2$, so $\mathfrak{B}$ runs over the $\binom{5}{2}=10$ pairs: one seed $K_{\mathfrak{B}}$ each

$K_{\{A,B\}}$	$K_{\{A,C\}}$	$K_{\{A,D\}}$	$K_{\{A,E\}}$	$K_{\{B,C\}}$	$K_{\{B,D\}}$	$K_{\{B,E\}}$	$K_{\{C,D\}}$	$K_{\{C,E\}}$	$K_{\{D,E\}}$

A, *B*, *C* **together: every pair misses one of them, so they hold all 10 seeds**

Who gets $K_{\mathfrak{B}}$? $K_{\mathfrak{B}}$ goes to all trustees except the members of $\mathfrak{B}$.

Sharing the FORS secrets, 1: replicated seeds

3-of-5: $k-1=2$, so $\mathfrak{B}$ runs over the $\binom{5}{2}=10$ pairs: one seed $K_{\mathfrak{B}}$ each

$K_{\{A,B\}}$	$K_{\{A,C\}}$	$K_{\{A,D\}}$	$K_{\{A,E\}}$	$K_{\{B,C\}}$	$K_{\{B,D\}}$	$K_{\{B,E\}}$	$K_{\{C,D\}}$	$K_{\{C,E\}}$	$K_{\{D,E\}}$

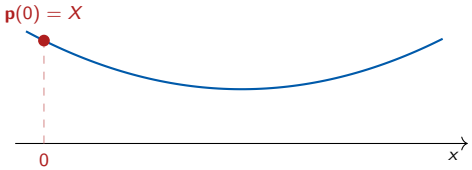
A, *B*, *C* **together: every pair misses one of them, so they hold all 10 seeds**

Who gets $K_{\mathfrak{B}}$? $K_{\mathfrak{B}}$ goes to all trustees except the members of $\mathfrak{B}$.

What is stored? Only $\binom{n-1}{k-1}$ seeds; every secret is derived, $X_{\mathrm{idx}}[i][j]=\bigoplus_{\mathfrak{B}} \mathrm{PRF}_{K_{\mathfrak{B}}}(\mathrm{idx},i,j)$: PRF-friendly.

k-of-*n*	3-of-5	10-of-20	33-of-64
seeds per trustee $\binom{n-1}{k-1}$	6	92 378 (≈3 MB)	$\sim 10^{18}$: infeasible

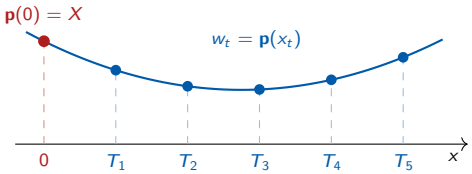
Sharing the FORS secrets, 2: Shamir shares



$k = 3$ of $n = 5$, over the reals; TALH: $\mathrm{GF}(2^\delta)$.

Deal. For every secret $X = X_{\mathrm{idx}}[i][j]$ the dealer picks a random polynomial $\mathbf{p}$ of degree $k-1$ with $\mathbf{p}(0) = X$.

Sharing the FORS secrets, 2: Shamir shares

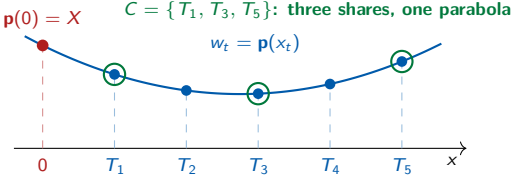


$k = 3$ of $n = 5$, over the reals; TALH: $\mathrm{GF}(2^\delta)$.

Deal. For every secret $X = X_{\mathrm{idx}}[i][j]$ the dealer picks a random polynomial $\mathbf{p}$ of degree $k-1$ with $\mathbf{p}(0) = X$.

Share. Trustee t gets $w_t = \mathbf{p}(x_t)$ and stores it.

Sharing the FORS secrets, 2: Shamir shares



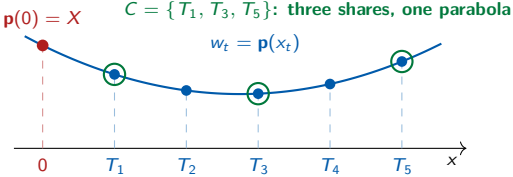
$k = 3$ of $n = 5$, over the reals; TALH: $\mathrm{GF}(2^\delta)$.

Deal. For every secret $X = X_{\mathrm{idx}}[i][j]$ the dealer picks a random polynomial $\mathbf{p}$ of degree $k-1$ with $\mathbf{p}(0) = X$.

Share. Trustee t gets $w_t = \mathbf{p}(x_t)$ and stores it.

Any k shares fix $\mathbf{p}$, hence X.

Sharing the FORS secrets, 2: Shamir shares



$k = 3$ of $n = 5$, over the reals; TALH: $\mathrm{GF}(2^\delta)$.

Deal. For every secret $X = X_{\mathrm{idx}}[i][j]$ the dealer picks a random polynomial $\mathbf{p}$ of degree $k-1$ with $\mathbf{p}(0) = X$.

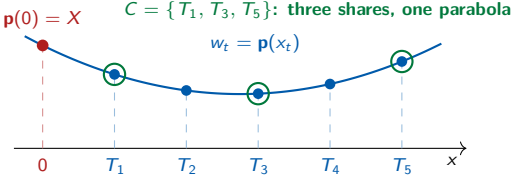
Share. Trustee t gets $w_t = \mathbf{p}(x_t)$ and stores it.

Any k shares fix $\mathbf{p}$, hence X.

Key material per trustee

k-of-n	**replicated seeds (compact)**	**Shamir table (scalable)**
3-of-5	6 seeds	$D \cdot \kappa \cdot \beta$ entries
10-of-20	92 378 seeds	$D \cdot \kappa \cdot \beta$ entries
33-of-64	$\sim 10^{18}$ seeds	$D \cdot \kappa \cdot \beta$ entries
grows with	$\binom{n-1}{k-1}$	number of secrets ($D \cdot \kappa \cdot \beta$)

Sharing the FORS secrets, 2: Shamir shares



$k = 3$ of $n = 5$, over the reals; TALH: $\mathrm{GF}(2^8)$.

Deal. For every secret $X = X_{\mathrm{idx}}[i][j]$ the dealer picks a random polynomial $\mathbf{p}$ of degree $k-1$ with $\mathbf{p}(0) = X$.

Share. Trustee t gets $w_t = \mathbf{p}(x_t)$ and stores it.

Any k shares fix $\mathbf{p}$, hence X.

Key material per trustee

k-of-n	**replicated seeds (compact)**	**Shamir table (scalable)**
3-of-5	6 seeds	$D \cdot \kappa \cdot \beta$ entries
10-of-20	92 378 seeds	$D \cdot \kappa \cdot \beta$ entries
33-of-64	$\sim 10^{18}$ seeds	$D \cdot \kappa \cdot \beta$ entries
grows with	$\binom{n-1}{k-1}$	number of secrets ($D \cdot \kappa \cdot \beta$)
PRF-friendly	✓ store the seeds, derive every share	✗ shares are random: one field element stored for each secret

Where TALH sits on the 8214C map

	Two (2) $n = 2$	**Three (3)** $n = 3$	**Small (S)** $4 \le n \le 8$	**Medium (M)** $9 \le n \le 64$	**Large (L)** $65 \le n \le 1024$
Dishonest (D) $f \ge n/2$	*n*2	*n*3*f*D	*n*S*f*D	*n*M*f*D	*n*L*f*D
Honest (h) $f < n/2$	—	*n*3*f*h	*n*S*f*h	*n*M*f*h	*n*L*f*h
2/3 honest (H) $f < n/3$	—	—	*n*S*f*H	*n*M*f*H	*n*L*f*H

Where TALH sits on the 8214C map

compact: replicated seeds

	Two (2) $n = 2$	Three (3) $n = 3$	Small (S) $4 \le n \le 8$	Medium (M) $9 \le n \le 64$	Large (L) $65 \le n \le 1024$
Dishonest (D) $f \ge n/2$	$n2$	$n3f\mathrm{D}$	$n\mathrm{S}f\mathrm{D}$	$n\mathrm{M}f\mathrm{D}$	$n\mathrm{L}f\mathrm{D}$
Honest (h) $f < n/2$	—	$n3f\mathrm{h}$	$n\mathrm{S}f\mathrm{h}$	$n\mathrm{M}f\mathrm{h}$	$n\mathrm{L}f\mathrm{h}$
2/3 honest (H) $f < n/3$	—	—	$n\mathrm{S}f\mathrm{H}$	$n\mathrm{M}f\mathrm{H}$	$n\mathrm{L}f\mathrm{H}$

Where TALH sits on the 8214C map

	compact: replicated seeds			scalable: Shamir shares	
	Two (2) $n = 2$	**Three (3)** $n = 3$	**Small (S)** $4 \le n \le 8$	**Medium (M)** $9 \le n \le 64$	**Large (L)** $65 \le n \le 1024$
Dishonest (D) $f \ge n/2$	*n*2	*n*3*f*D	*n*S*f*D	*n*M*f*D	*n*L*f*D
Honest (h) $f < n/2$	—	*n*3*f*h	*n*S*f*h	*n*M*f*h	*n*L*f*h
2/3 honest (H) $f < n/3$	—	—	*n*S*f*H	*n*M*f*H	*n*L*f*H

Where TALH sits on the 8214C map

	compact: replicated seeds			scalable: Shamir shares	
	Two (2) $n = 2$	**Three (3)** $n = 3$	**Small (S)** $4 \le n \le 8$	**Medium (M)** $9 \le n \le 64$	**Large (L)** $65 \le n \le 1024$
Dishonest (D) $f \ge n/2$	$n2$	$n3f\mathrm{D}$	$n\mathrm{S}f\mathrm{D}$	$n\mathrm{M}f\mathrm{D}$	$n\mathrm{L}f\mathrm{D}$
Honest (h) $f < n/2$	—	$n3f\mathrm{h}$	$n\mathrm{S}f\mathrm{h}$	$n\mathrm{M}f\mathrm{h}$	$n\mathrm{L}f\mathrm{h}$
2/3 honest (H) $f < n/3$	—	—	$n\mathrm{S}f\mathrm{H}$	$n\mathrm{M}f\mathrm{H}$	$n\mathrm{L}f\mathrm{H}$

scalable ASPEN
1 round

Where TALH sits on the 8214C map

	compact: replicated seeds			scalable: Shamir shares	
	Two (2) $n = 2$	**Three (3)** $n = 3$	**Small (S)** $4 \le n \le 8$	**Medium (M)** $9 \le n \le 64$	**Large (L)** $65 \le n \le 1024$
Dishonest (D) $f \ge n/2$	$n2$	$n3f\mathrm{D}$	$n\mathrm{S}f\mathrm{D}$	$n\mathrm{M}f\mathrm{D}$	$n\mathrm{L}f\mathrm{D}$
Honest (h) $f < n/2$	—	$n3f\mathrm{h}$	$n\mathrm{S}f\mathrm{h}$	$n\mathrm{M}f\mathrm{h}$	$n\mathrm{L}f\mathrm{h}$
2/3 honest (H) $f < n/3$	—	—	$n\mathrm{S}f\mathrm{H}$	$n\mathrm{M}f\mathrm{H}$	$n\mathrm{L}f\mathrm{H}$

TALH (both)
3 rounds
$f \le k - 1$ **corrupt**

scalable ASPEN
1 round

Thank you

questions · answers · comments · contributions

Muhammed Said Gündoğan · muhammedsaidgundogan@gmail.com