

# Haystack: Threshold and Distributed Hash- Based Signatures

John Kelsey, NIST + COSIC/KU Leuven

Stefan Lucks, Bauhaus-Universität, Weimar

Nathalie Lang, Bauhaus-Universität, Weimar\*

**MPTS 2026: NIST Workshop on Multi-Party Threshold Schemes 2026**  
**28 Jan 2026**

Photo by Roman Tymochko:

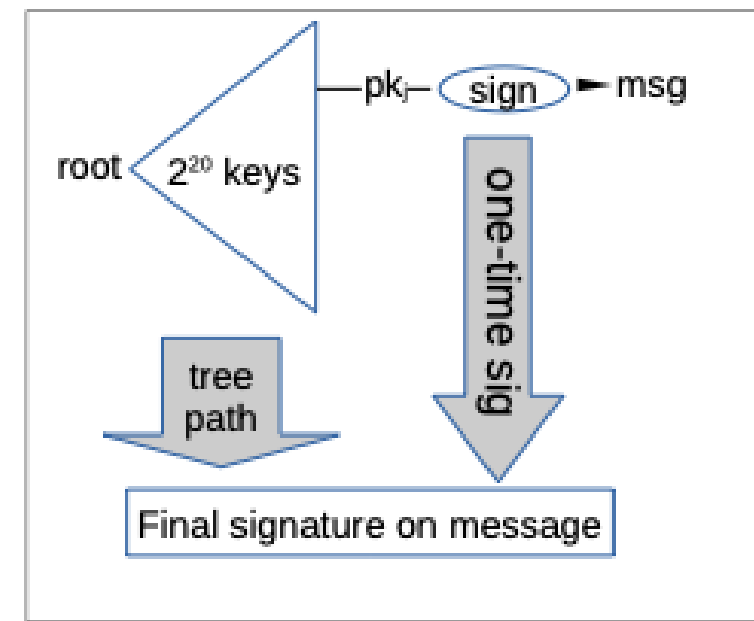
<https://www.pexels.com/photo/brown-hays-on-a-grassy-field-8652819/>



# What We Do

- Start with any **stateful hash-based signature** scheme
  - For example, IETF standards LMS and XMSS
- Turn it into a **threshold signature**
  - n-of-n
  - k-of-n
- Or a **distributed signature**
  - Arbitrary signing coalitions





# Stateful Hash-Based Signatures in a Nutshell

# Stateful Hash-Based Signatures (HBS)

- Postquantum and very efficient
- Based only on cryptographic hash function like SHA256
- Start with a one-time signature (OTS)
  - Can build this with only hash functions
  - Secure as long as we only sign once with each OTS key

*Key reuse is a disaster that allows forgeries!*
- Generate D OTS keys
  - Put them into a Merkle tree
  - Our *composite* HBS public key includes root of Merkle tree

# Stateful HBS (2)

Signer is “stateful”

- Keeps track of which OTS keys have been used

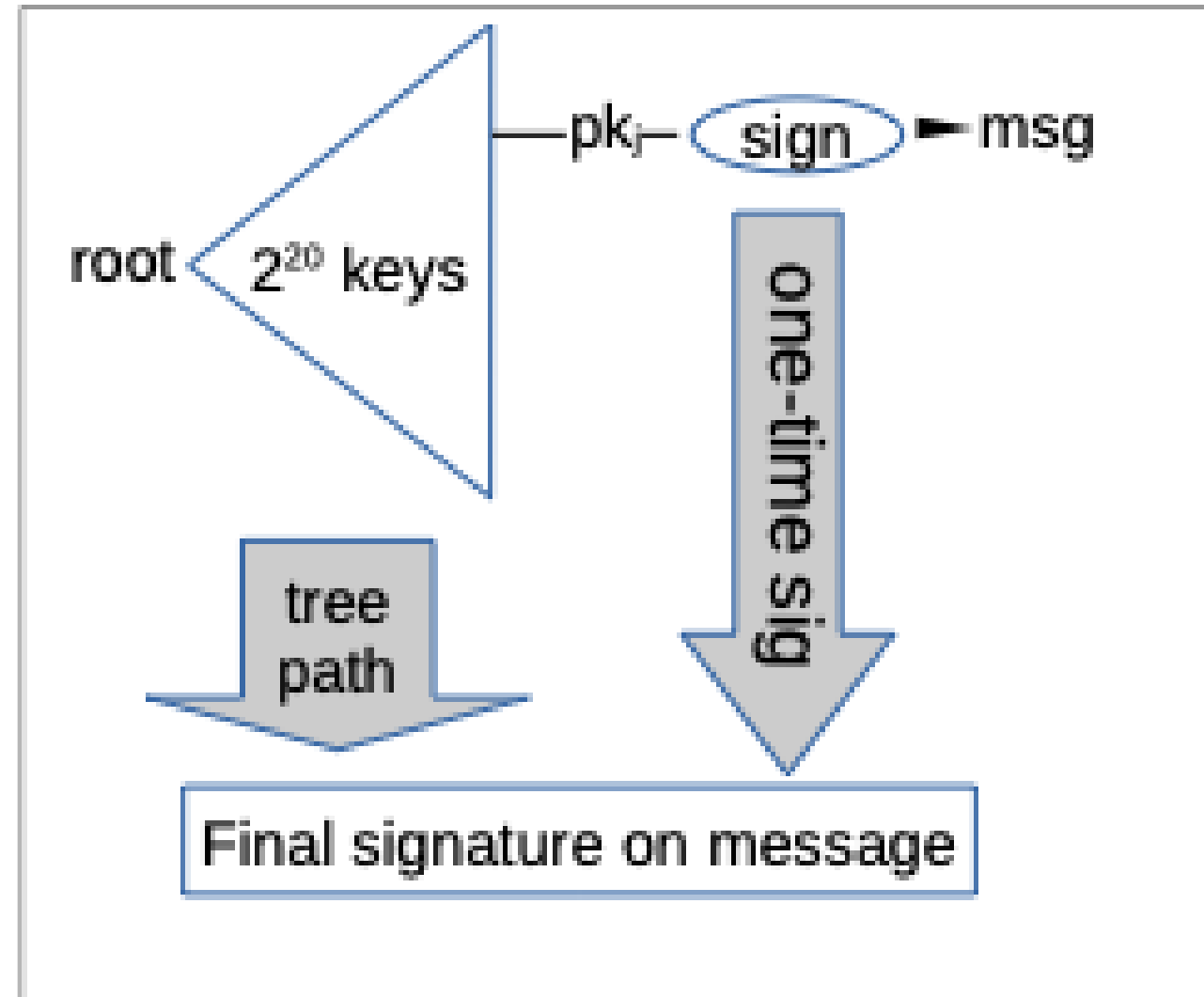
To sign:

- Choose an unused OTS key\*
- Sign message with OTS key
- Construct Merkle tree path

Final sig:

- Merkle tree path of OTS key
- OTS on message

*\*But don't reuse a key!*



# Stateful HBS: Standards

## Two IETF standards

- LMS (RFC 8554)
- XMSS (RFC 8391)

## NIST standard

- SP 800-208

## Real-world uses

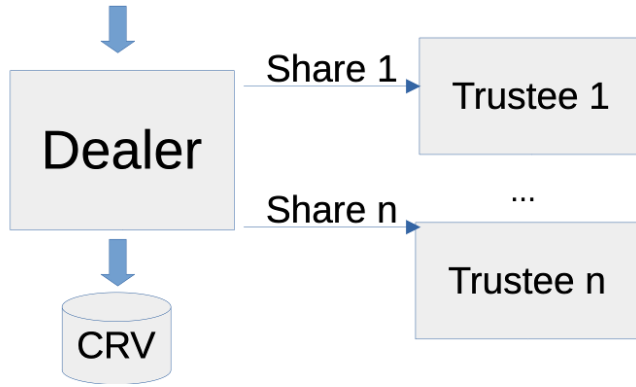
- Firmware signing
- Certificate signing

Matt Green blog post on HBS: <https://blog.cryptographyengineering.com/2018/04/07/hash-based-signatures-an-illustrated-primer/>

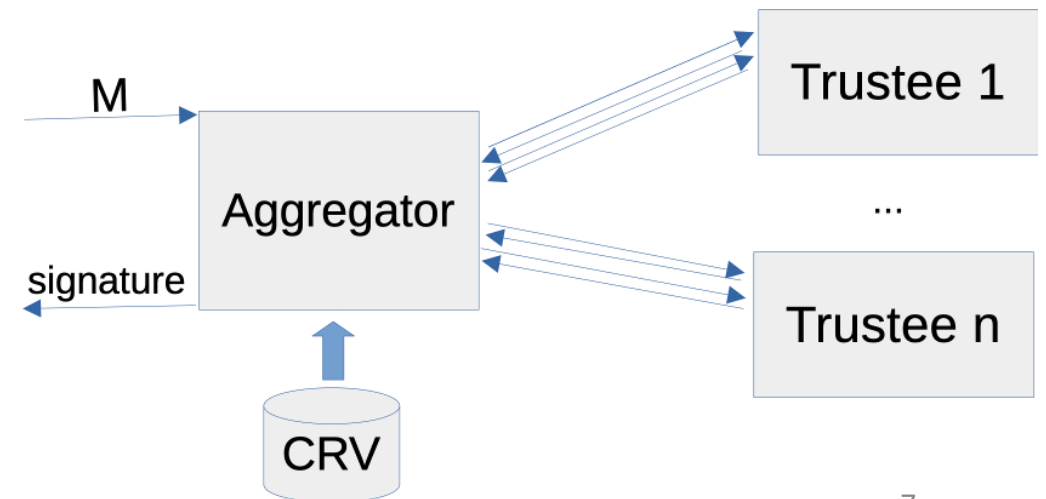
Alfred Menezes Youtube tutorial on HBS: URL: <https://youtu.be/pt5WR8D4IXI>

My Youtube tutorial on HBS: <https://www.youtube.com/watch?v=jiU0ICoiPI0>

Stateful HBS  
Private Key



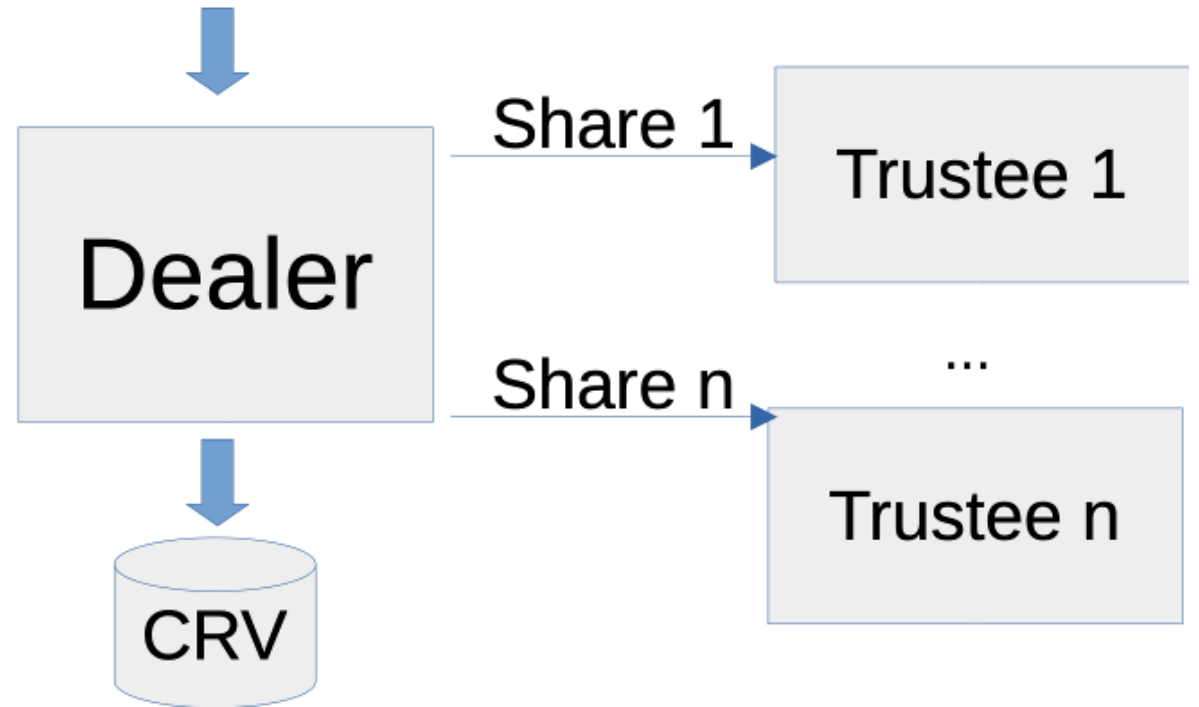
# Overview of Our Scheme



# SETUP

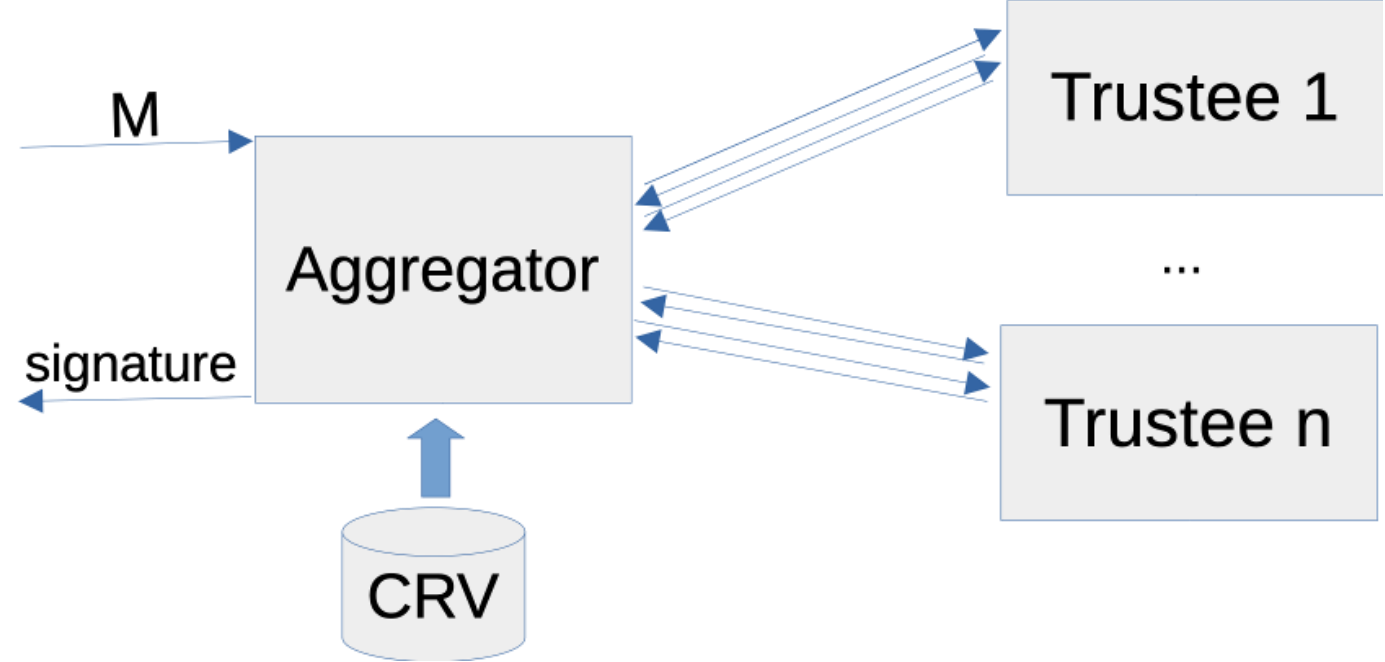
- Setup: Trusted dealer
  - Generates stateful HBS key
  - Splits into shares
  - Constructs CRV (common reference string)
  - Distributes shares to trustees

Stateful HBS  
Private Key



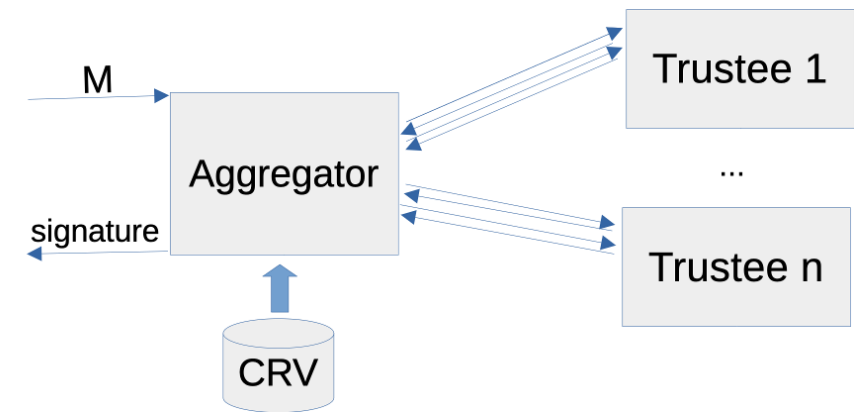
- Dealer does about 2x total work of generating the stateful HBS key
- Common reference string (CRV) is big—typically 0.1 GiB – 10 GiB
- Shares are a few hundred bytes

# SIGNING



- Aggregator = untrusted party that coordinates signing protocol
  - Needs access to CRV
  - Talks to trustees over point-to-point network
  - Only a few hundred bytes + message
  - Four message protocol with each trustee
- Trustee = trusted party with one share of the key
  - Responds to messages from aggregator
  - About same work as a normal stateful HBS signature

# The Fine Print: Limitations



Threshold parameters (t-of-n)

- Practical: 2-of-3, 3-of-5, 5-of-9
- Not practical:  $n > 10$

Aggregator must know who's participating at start of protocol

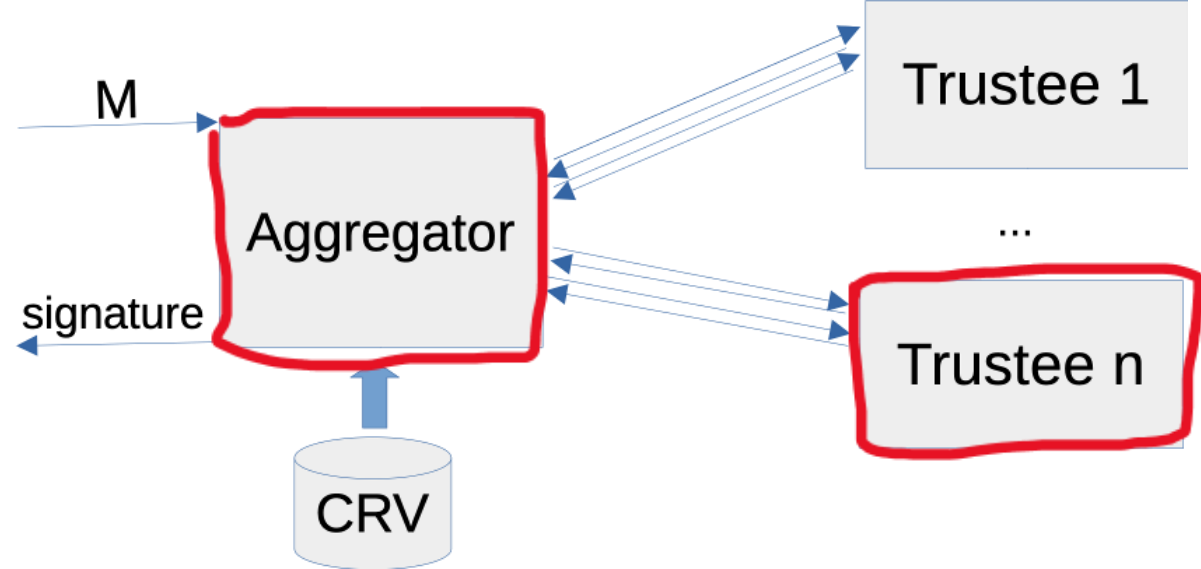
- If it needs to poll trustees, that happens before our protocol

One-level trees only

- Hypertree variants (HSS / XMSS<sup>MT</sup>) not really practical
- Stateless HBS (Sphincs+ aka SLH-DSA) not really practical

# Security

- Malicious adversaries
- Static corruption of trustees
- One-more unforgeability (stronger than EU-CMA)



Reduction proof: Forging signature with incomplete signing coalition →

- Forging signatures in underlying stateful HBS scheme  
or
- Distinguishing a PRF from random function

# Why It Matters: The Foot Cannon

*Biggest problem with stateful HBS is **key reuse***

- Signing two messages with same one-time key = disaster

This can happen accidentally in many ways!

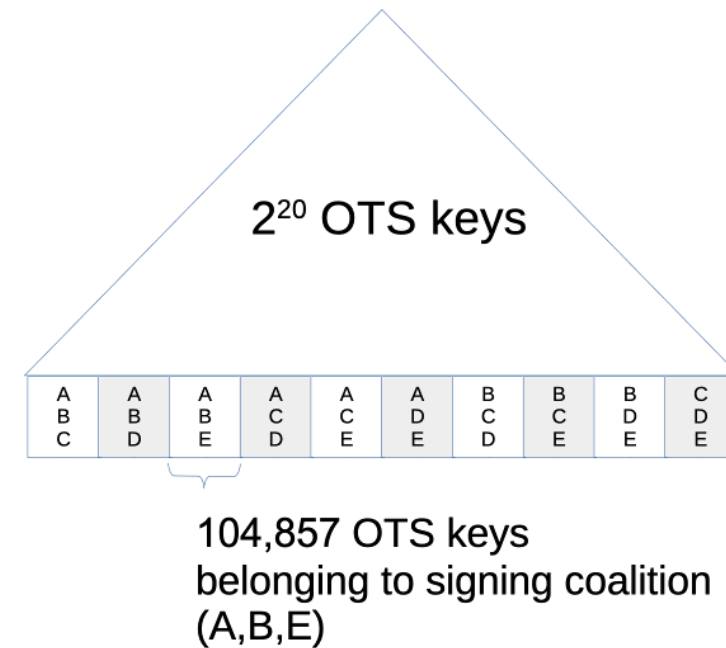
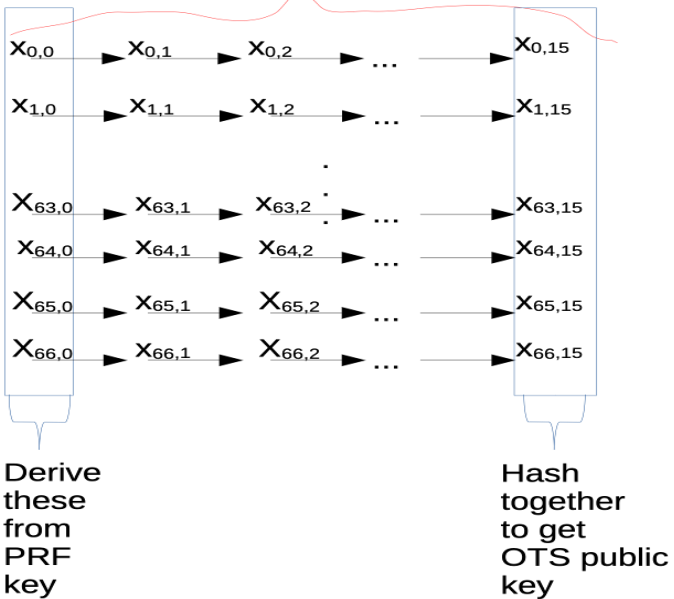
- Write failure, VM cloning, Backup/restore

Splitting control across many trustees *basically solves this problem*

- No honest trustee will reuse a key
- k-of-n threshold sig: Only reuse a key *if k trustees fail at same time!*

**Threshold version of stateful HBS more secure than non-threshold version!**

16 hash values in each line



# How The Heck Does That Even Work???

\* The full scheme is too complicated to explain in detail; this is a high-level summary

# Why is this hard?

1. No algebraic structure
2. Randomized hashing
3. Key reuse

Stateful HBS are a very different animal than other cryptosystems!

# Winternitz One Time Signatures (OTS)

OTS private key = PRF key

- $\text{PRF}(K, (\text{keyid}, 0, 0)) \Rightarrow X_{0,0}$

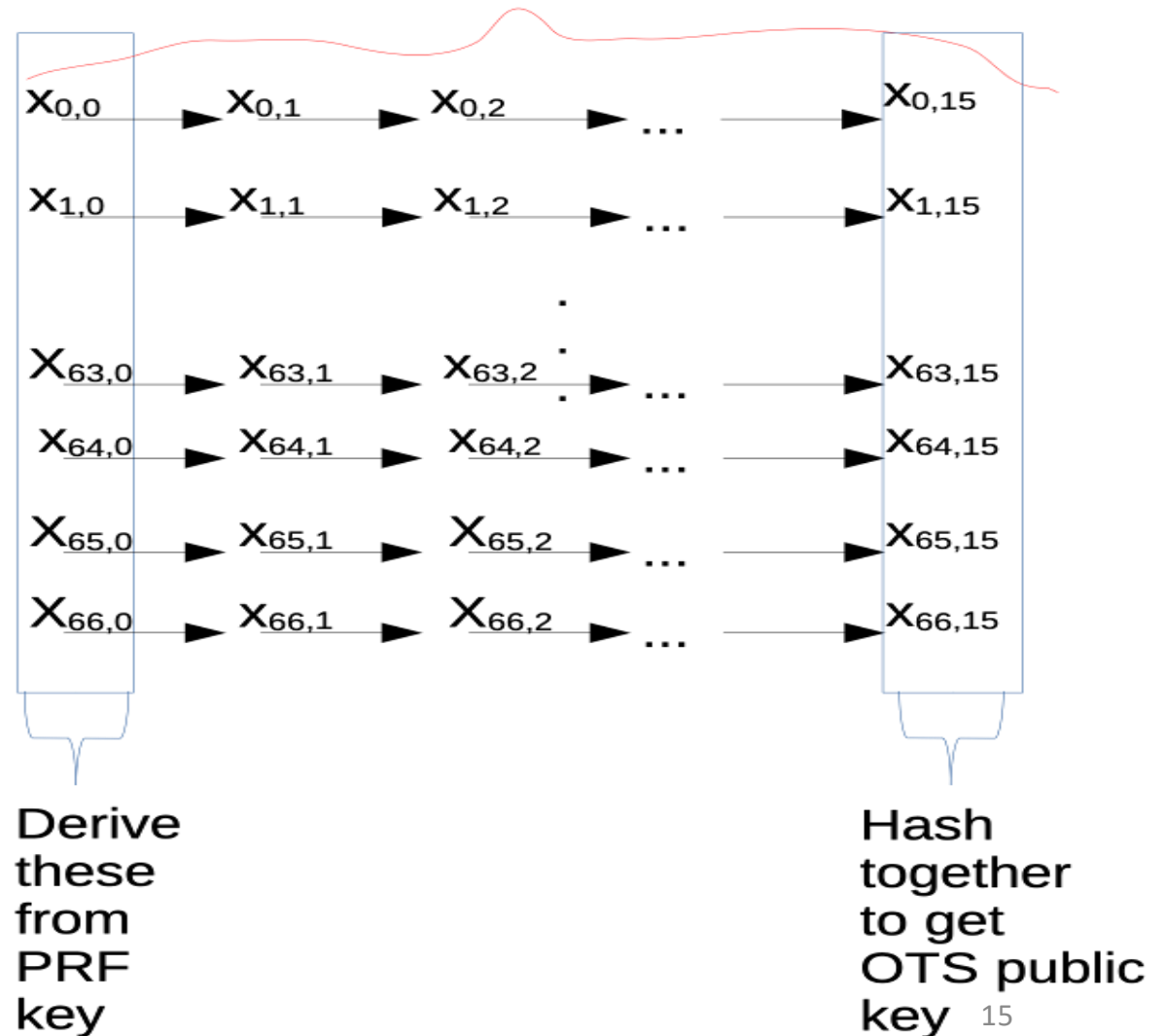
Construct rows of hash chains

- $X_{i,j} = \text{Hash}(X_{i,j-1})$

OTS public key:

- Hash of last entry in each row

16 hash values in each line



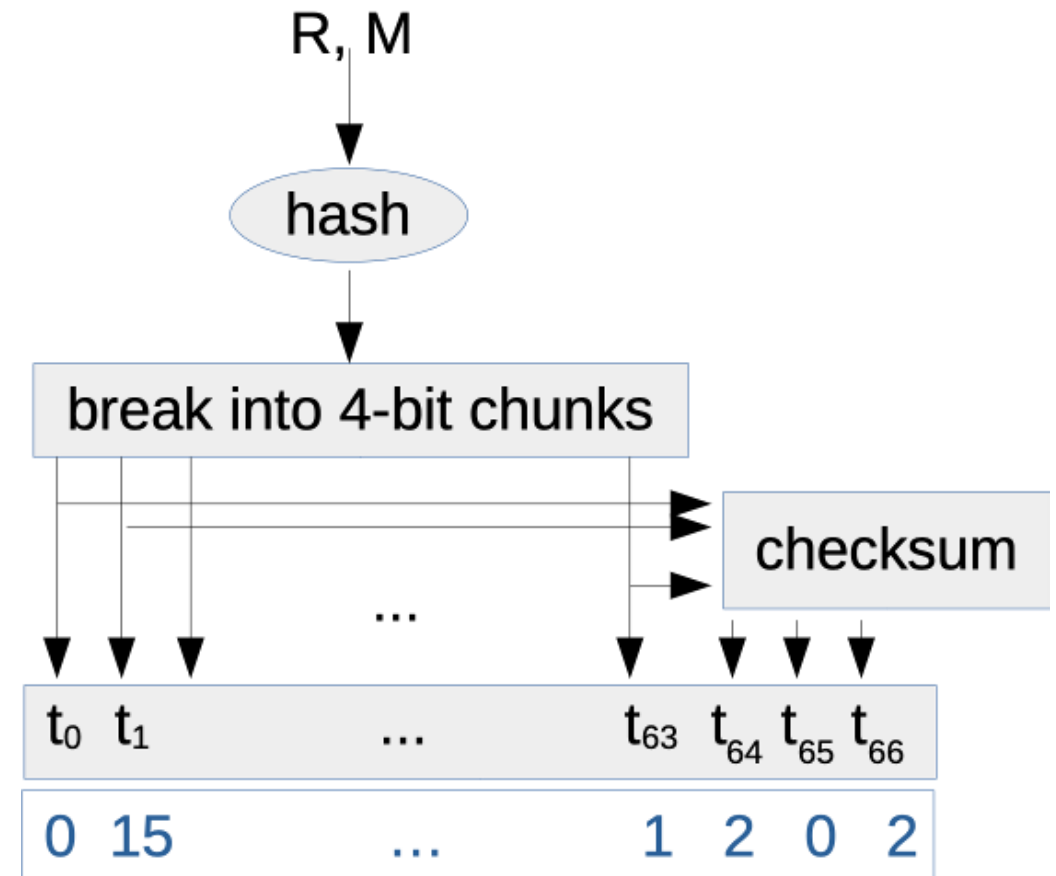
# Hashing message and adding checksum

- Randomized hash  
 $H = \text{hash}(R, M)$   
Break into 4-bit digits
- Compute checksum on digits  
Needed to block an attack
- Append checksum digits

Result:

*hash\_digits || checksum\_digits*

*\* Note: We can do all this without knowing private key*

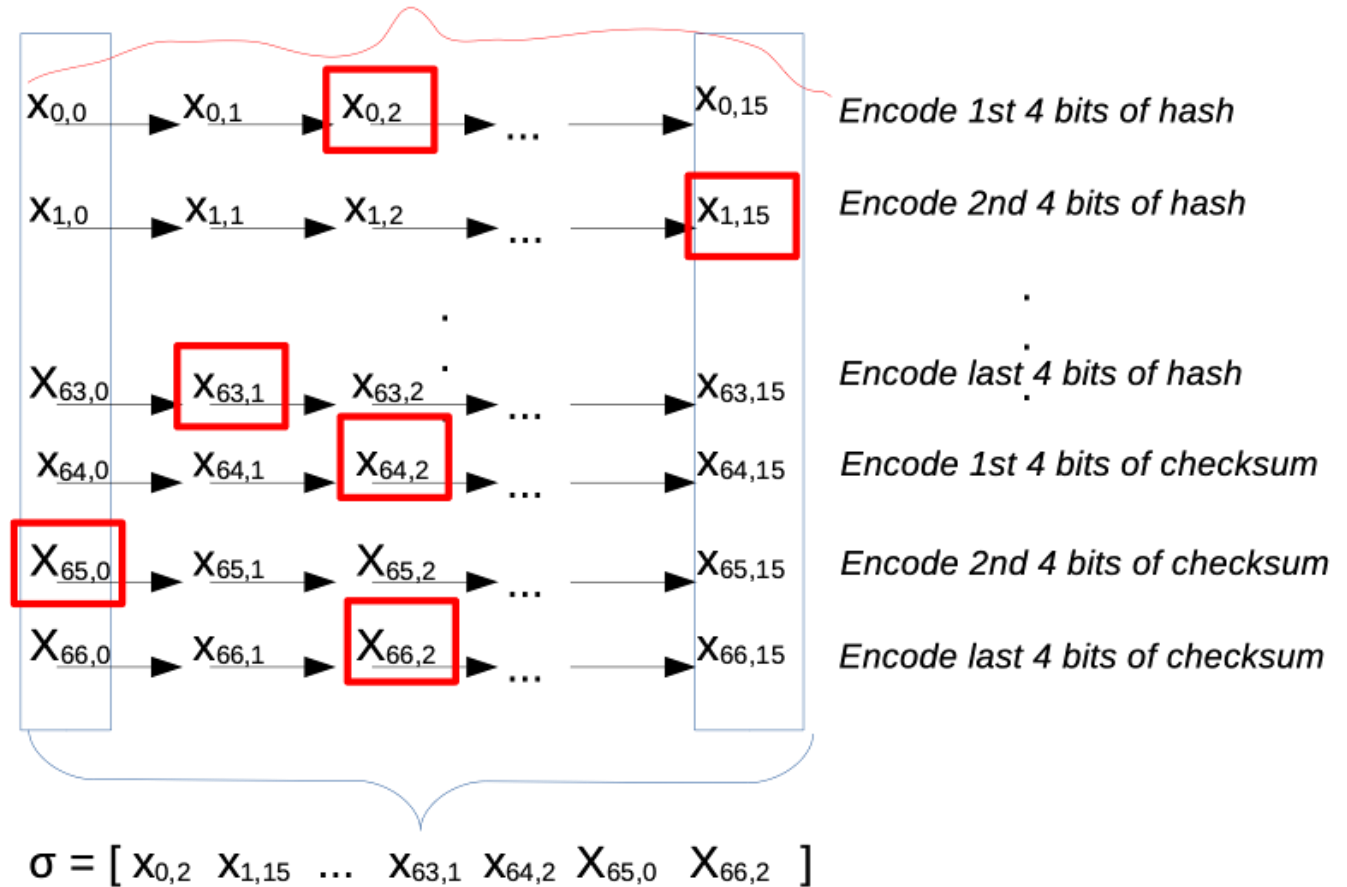


# Signing:

*Signing involves a lot of hashing*

- Take hash || checksum digits  
 $h = [2, 15, \dots, 1, 2, 0, 2]$
- Each digit says how many times to step forward in hash chain
- Lots of hashing with secret inputs.

16 hash values in each line-- 64 chains for hash, 3 for checksum



Reveal values selected by digits of hash+checksum to form signature

# Table Representation

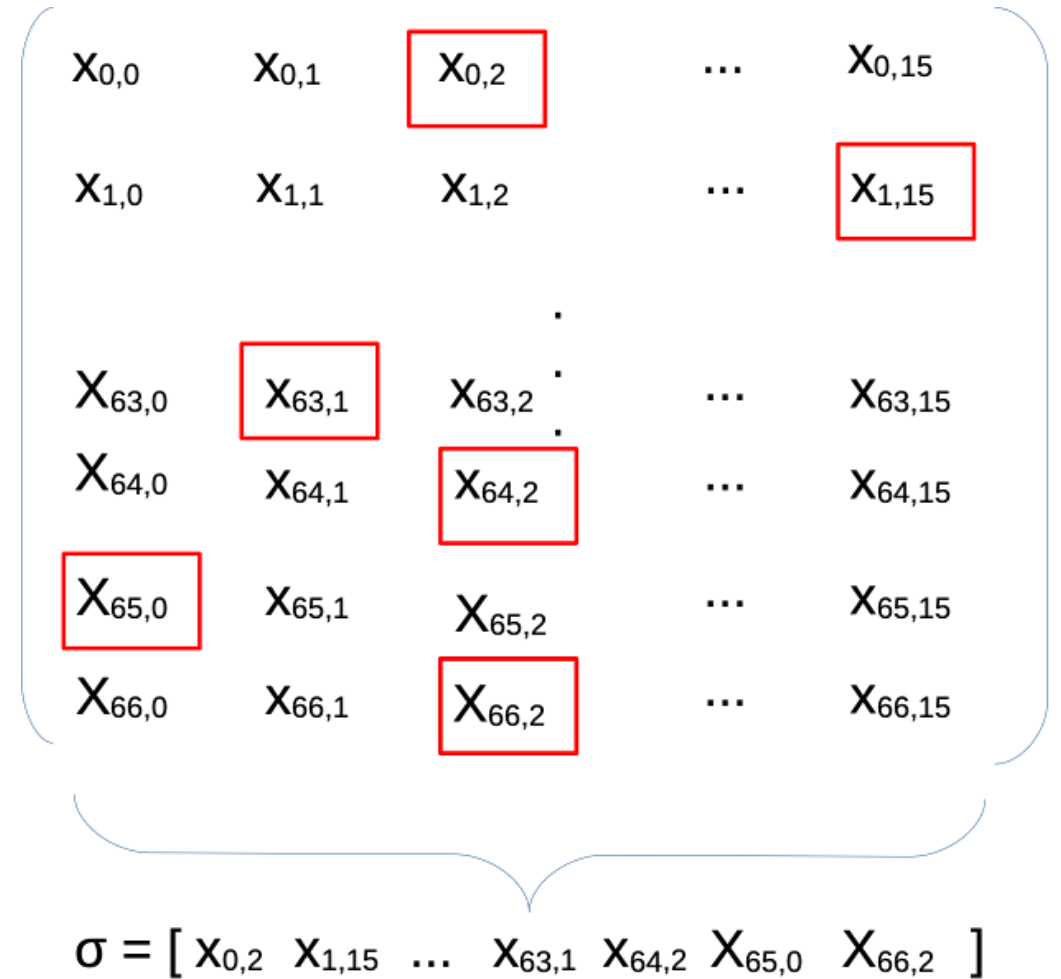
- Precompute all hash chains
- Store as a 2D array

$$X[0..p-1, 0..15]$$

When we want to sign

- hash + checksum:  $h = [2, 15, \dots, 1, 2, 0, 2]$
- Signing done with table lookups only!  
For  $i = 0..p-1$ :  
 $\sigma[i] = X[i, h[i]]$

*This is trick that makes threshold HBS possible.*

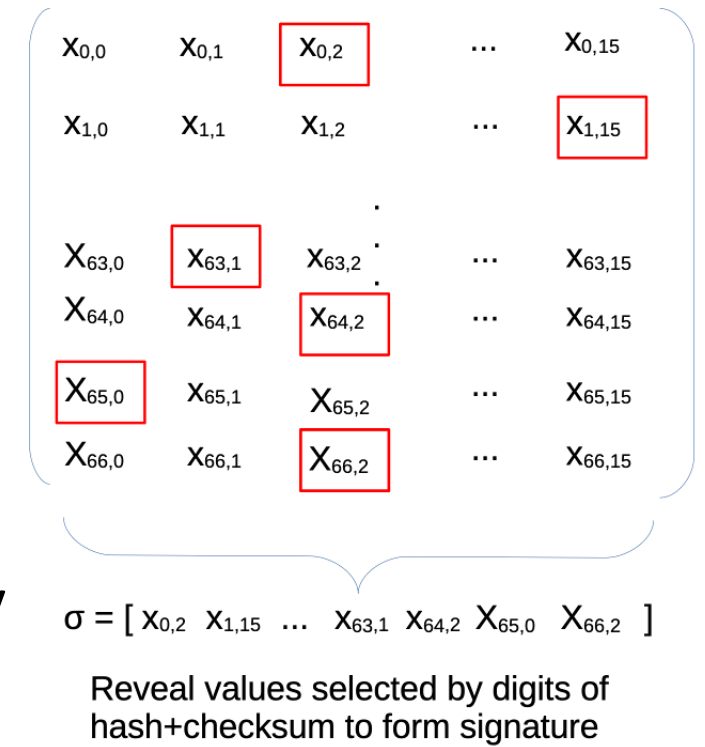


Reveal values selected by digits of hash+checksum to form signature

# How can we thresholdize this?

Setup:

- Trusted dealer generates 2D array for each OTS key
- Splits each element in array into  $n$  shares  
$$X^1[i,j], X^2[i,j], \dots, X^n[i,j] = \text{SplitIntoShares}(X[i,j], n)$$
- Each trustee  $t$  gets  $X^t[0..p-1, 0..15]$



# Threshold signing (n-of-n)

Given  $h[0..p-1]$  and  $X^t[0..p-1, 0..15]$

- Sign  $h$  with my share of table:

For  $i = 0..p-1$ :

$$\sigma^t[i] = X^t[i, h[i]]$$

When we combine all shares of signature, we get full signature

For  $i = 0..p-1$ :

$$\sigma[i] = \text{CombineShares}(\sigma^1[i], \sigma^2[i], \dots, \sigma^n[i])$$

# But those are big shares!

As described, each OTS key requires several KiB of storage

- Full Stateful HBS key might have  $2^{15}$  OTS keys, so LOTS of storage

Solution:

- Derive trustees' shares using a PRF

$$X^t[\text{keyid}, i, j] = \text{PRF}(K[t], (\text{keyid}, i, j))$$

- Add one high-memory share and store it in common reference string  
 $\text{CRV}[\text{keyid}].X[i,j] = \text{XOR of all trustee shares XOR } X[i,j]$

# From n-of-n to t-of-n

Obvious way to go to t-of-n:

- Use t-of-n Shamir shares of table entries

Problem: Key reuse

- 2-of-3 scheme
- Trustees: A, B, C ← C is corrupt
- First signature: A, C sign with keyid #1
- Second signature: B, C sign with keyid #1

*Result: C sees two sigs with same OTS key, can forge anything*

# Sharding

- Consider trustees A,B,C,D,E in a 3-of-5 scheme
- Allocate a range of the OTS keys for each signing coalition

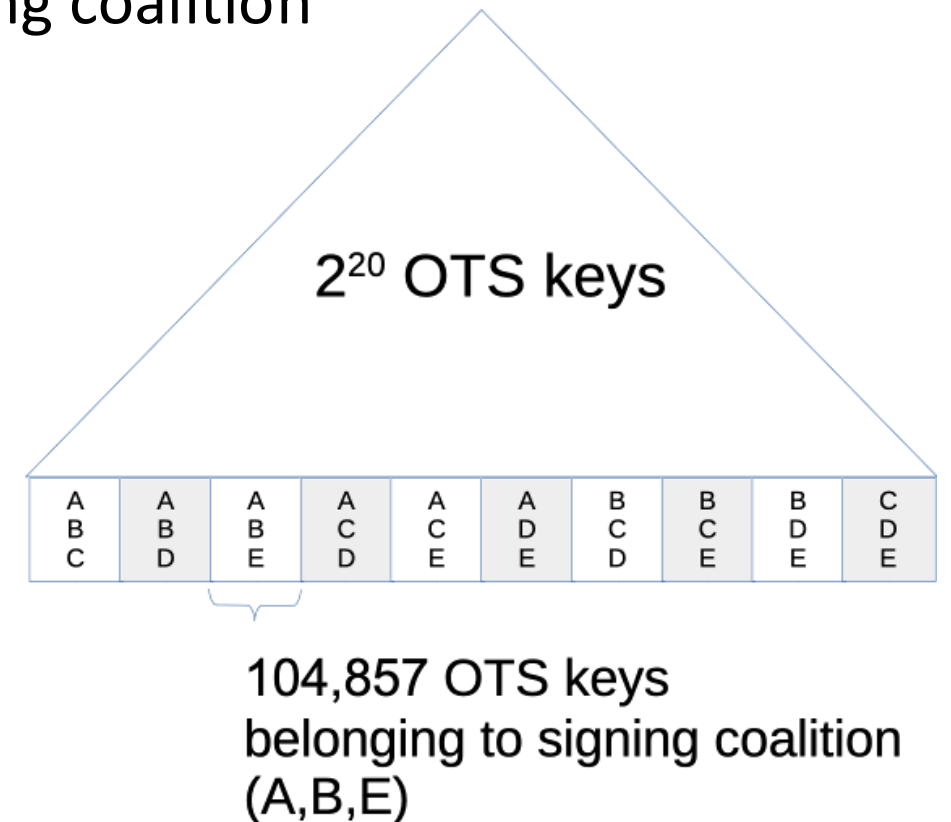
$C(5,3) = 10$  coalitions

ABC ABD ABE ACD ACE

ADE BCD BCE BDE CDE

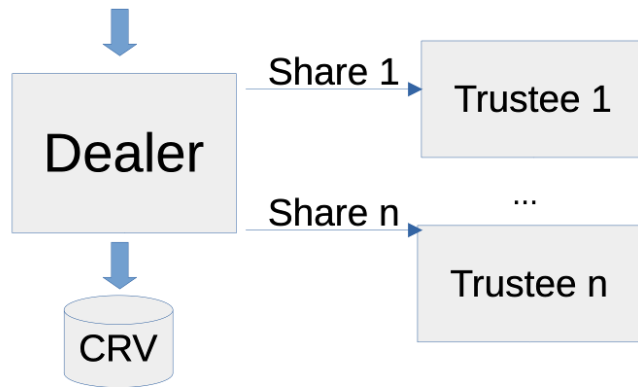
- Trustee must keep state for all its coalitions  
Six coalitions in this case  
State = two 32-bit integers

- No change to stateful HBS signature  
Merkle tree path + OTS

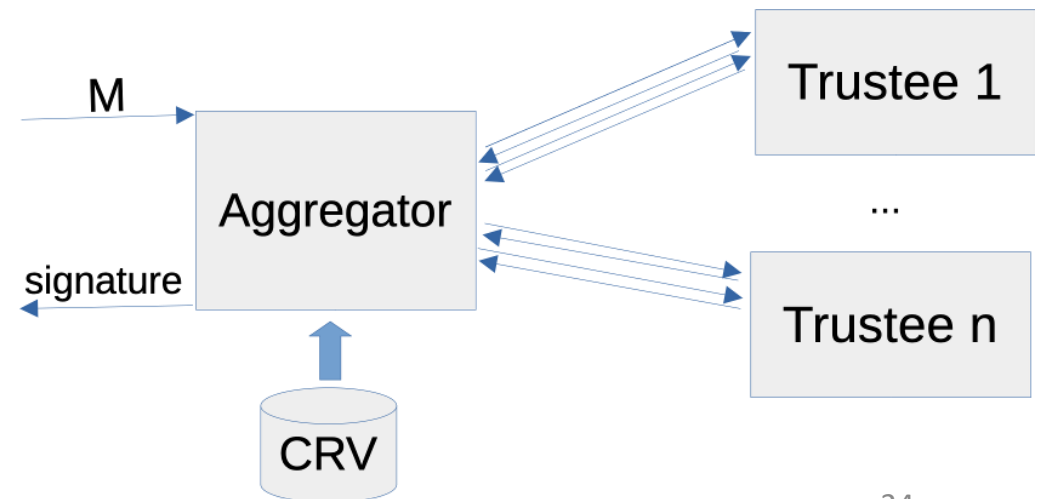


*Result: t-of-n threshold scheme practical when  $n < 10$  or so.*

Stateful HBS  
Private Key



# Wrapup



# Can I Use It?

- We have no patents on this scheme and do not know of any patents
  - Our hope is that our scheme gets widely used
- Open-source Python implementation
  - LMS only so far
  - Working on getting it through NIST approval process to publish
- Published in IACR Communications in Cryptography last year
  - 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

# Summary

- We can make threshold signatures for LMS and XMSS
  - Existing stateful HBS standards in current use!
- Our schemes are practical and efficient
  - Big CRS needed only by aggregator
  - Trustees can be implemented on a smartcard
  - Point to point communications, four-message protocol
  - No broadcast channel, no MPC protocols
- Big practical impact
  - Prevents key reuse—biggest practical security issue with stateful HBS
  - Actually makes signature schemes stronger!
- Reference code for LMS available soon in Python

