Stateful HBS Backups Revising 208

John Kelsey, NIST, Nov 2024

Stateful HBS Review

Stateful HBS

- Big Picture
  - Digital signatures based on hash functions
  - No other hardness assumptions
- Advantages
  - Small, fast PQ signatures
  - High confidence vs cryptanalysis
- Disadvantages
  - Tricky to implement and use securely
  - Backups are a big problem

Building Block: One-Time Signatures

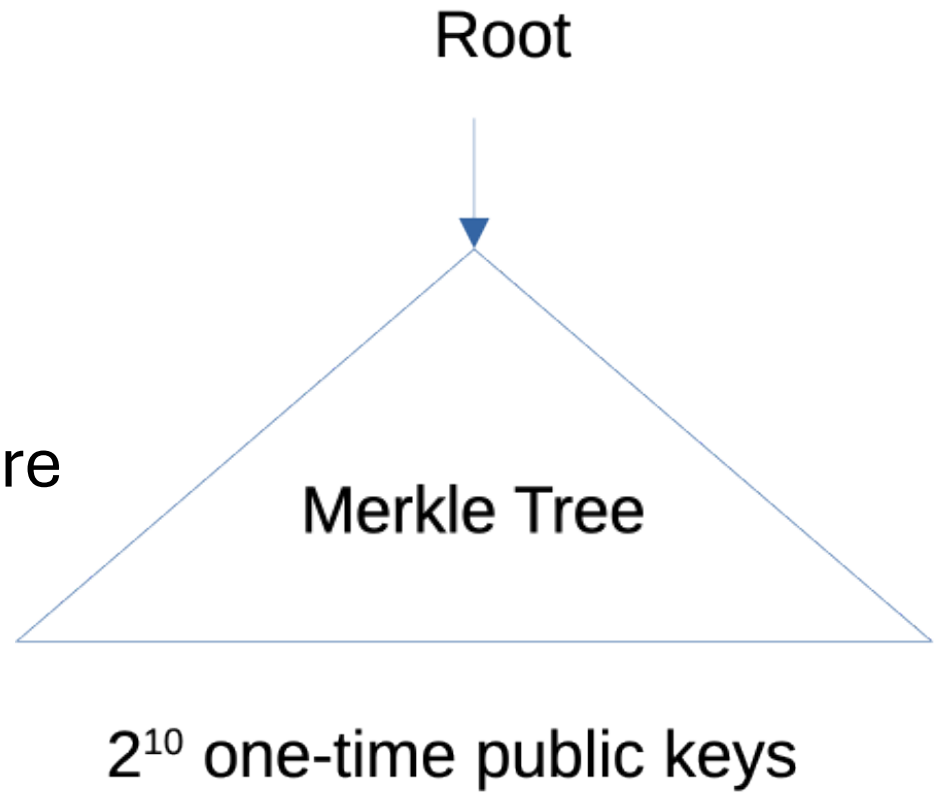
- Public key signature scheme
  - Keygen() → PK, SK
  - Sign(SK, M) → sig
  - Verify(PK, M, sig) → True/False

- One-time*
  - Sign two messages with same key → leak signing key
  - Anyone who sees both sigs can forge messages!

*This creates all the operational headaches with stateful HBS

Merkle Tree

- Generate 2^d one-time keys
- Hash them all together in a data structure called a Merkle tree
- Publish the root of the tree
  - This is the hash of all the leaf values and commits to all of them
- Can produce a path from any leaf value to root
- Path is d hashes long



Stateful HBS: One-Time Sigs + Merkle Tree

Public key = Merkle tree root + some metadata

To sign:

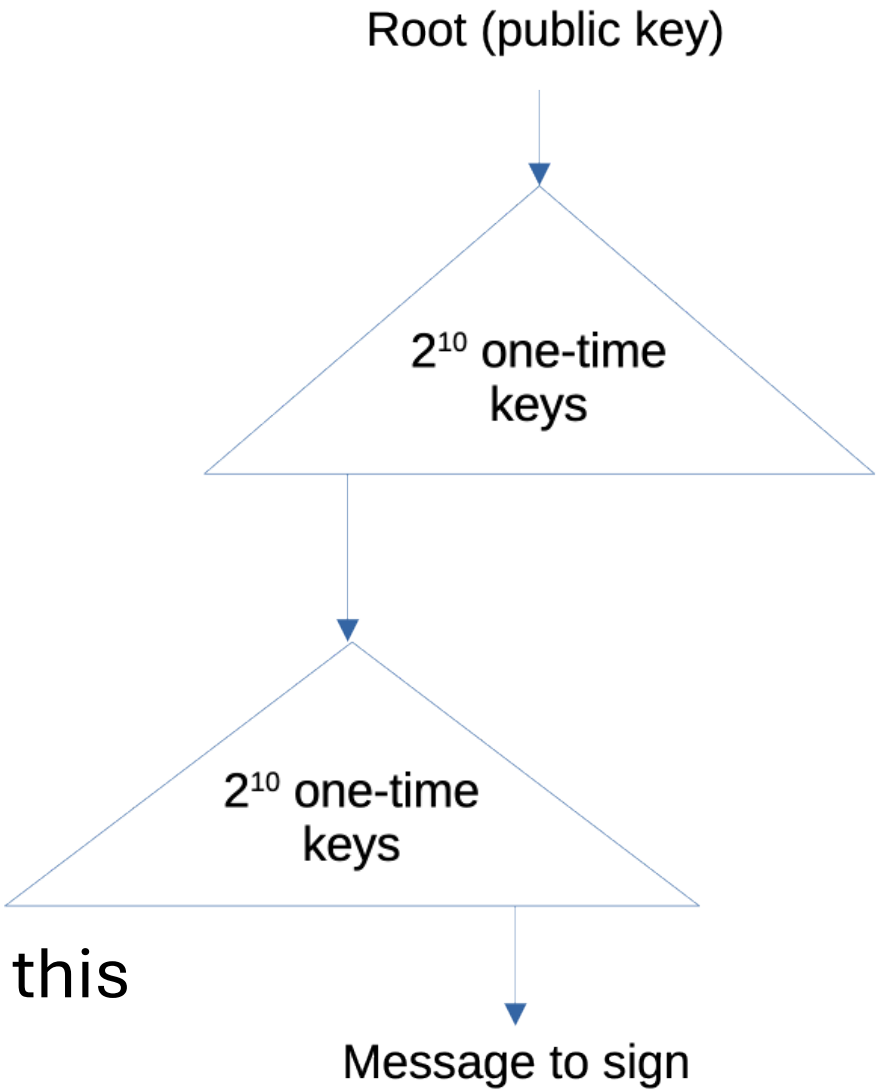
- Choose an unused one-time key
- Create one-time sig from that key
- Create Merkle tree path from that key to root
- Signature = Merkle tree path + one-time sig

To verify:

- Use one-time sig + message → one-time public key
- Verify Merkle tree path back to root

Two level trees

- Setup (keygen):
  - Generate 2^{10} one time keys
  - Put into Merkle tree
  - Root plus metadata = public key of scheme
- Each key can sign a new Merkle tree root
  - 2^{10} one-time keys
  - This allows a total of 2^{20} one-time signatures
- LMS and XMSS both have options to allow this
  - CNSA guidance does NOT allow it



Stateful HBS is Stateful!

The key thing to remember about these schemes is:

…they are secure *as long as a key never signs two messages!*

- Implementation must keep track of one-time keys used so far
  - That's the "state"

- If this state is repeated, we reuse a key
  - And attackers can forge signatures from us

SP 800-208 and Backups

The Backup Problem

- High-value system with long lifetime
- Example: Firmware signing
- Want to keep keys in HSM
- Problem: HSM could die
  - Disaster destroys HSM
  - HSM stops working
  - Someone steals HSM
  - PIN for HSM lost/forgotten
  - HSM has known security flaw and is out of service

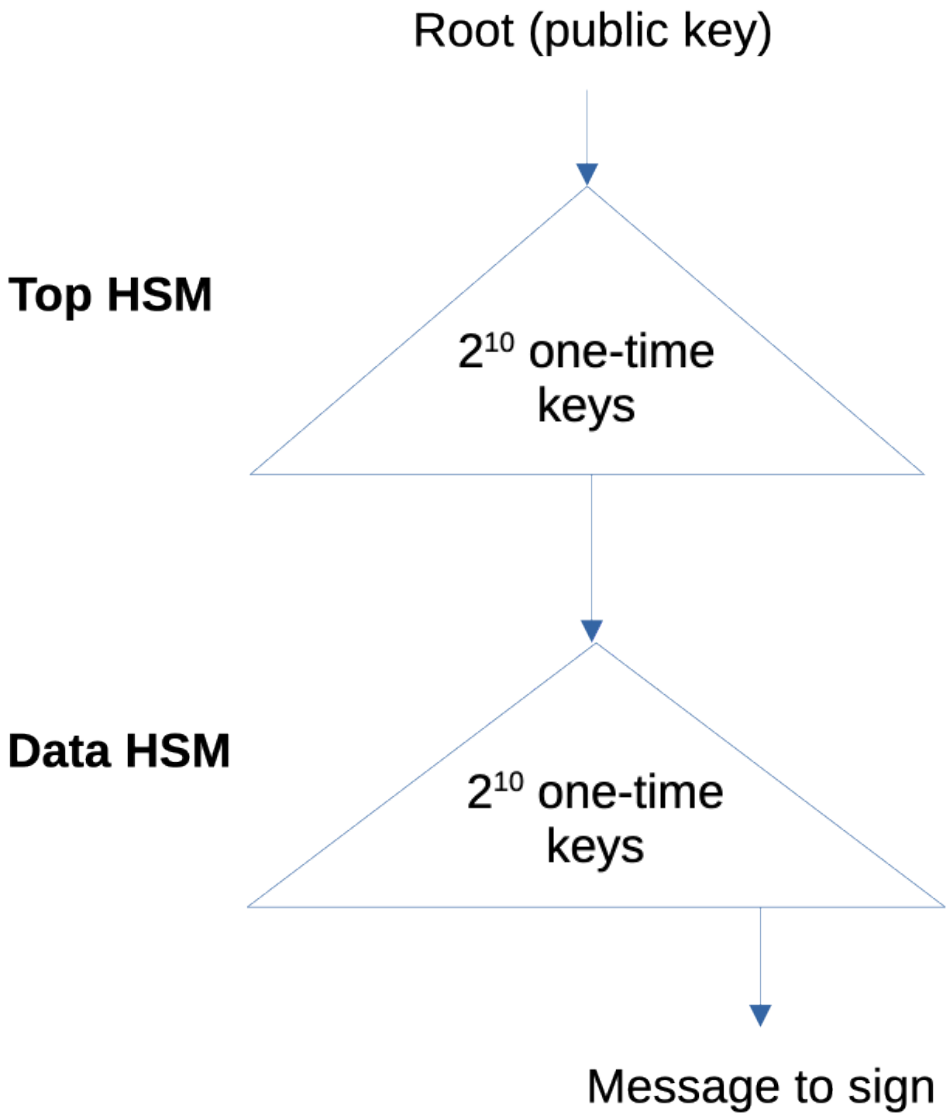
Need to be able to sign firmware updates for whole system lifetime.

SP 800-208 Solution

- Use a two-level tree
  - Top HSM generates first tree
  - Each data HSM generates second-level tree
  - Top HSM key signs root of data HSM tree

- Use the data HSMs to sign actual data
- Discard top HSM or keep a few keys to create new data HSMs

- MANY data HSMs
- As long as one works, system keeps running



Why doesn’t this work?

Problem: HSM could die

- Disaster destroys HSM
- HSM stops working
  - If we run out of HSMs while system is still in use, major problem!
- Someone steals HSM
- PIN for HSM lost/forgotten
- HSM has known security flaw and is out of service
  - 30 years is a long time, vendors can go out of business

Requirements and constraints

- Must recover from device failure
- No way to know how many devices will be needed
- Move between vendors over time

- Minimal changes to 208
- Ideally avoid forcing anyone to use some patented thing
- Data always signed using HSM

Our Approach: Overview

Three basic operations

- Setup:
  - Generate keys and set up system
  - Produce public and private key material
- Provision:
  - Set up a data HSM to be able to sign data
  - Must ensure we never provision two HSMs with same key material
- Sign:
  - Data HSM signs data on demand

Our Approach: Overview (2)

- Setup:
  - Could happen in software or hardware
  - Store private key data for provisioning
  - Need to ensure provisioning can happen even if devices fail
- Provision:
  - Use private key data from setup
  - Procedural defenses required here
  - Always end up with data HSM
- Sign:
  - Signing data only happens on HSM

Scheme #1: Extending the 208 Technique

- **Setup:**
  - Generate top-level key
    2^d one-time keys
  - PK = root + metadata
  - Store each one-time key to allow provisioning

- **Provision:**
  - Have data HSM generate second-level key
    2^d one-time keys
  - Choose next top-level one-time key to sign root
  - Procedural defenses must prevent reuse of top-level one-time keys

Scheme #1 Pros and Cons

Advantages of two-level scheme

- Minimal change to 208
  - No change to requirements for data HSM
- Interoperable-–just have to comply with LMS/XMSS standard

Disadvantages:

- Two level keys ➔ much bigger signatures
- Doesn’t fit with CNSA guidance

Scheme #2: One-Level Signatures

- Setup:
  - Generate 2^d random seeds, S[0..2^d-1]
  - Use each to generate 2^d one-time keys
  - Construct Merkle tree

    PK = root + metadata
  - Store each seed to allow provisioning

- Provision:
  - Choose which seed to load into data HSM
  - Procedural defenses must prevent reuse of seed
  - HSM uses seed to generate bottom d layers of Merkle tree
  - Transfer rest of Merkle tree path into HSM to allow signatures

Scheme #2 Pros and Cons

Advantages:

- Works with CNSA guidance
- Much shorter signatures

Disadvantages:

- Bigger changes to 208
  - Data HSMs must be changed!
- Interoperability harder
  - Need to import a seed and do the right thing with it
  - If HSM follows LMS/XMSS guidance on keygen this will work

Procedural defenses

For both schemes:

- Setup produces black of secret data for provisioning
- Provisioning step must ensure no reuse of keys
- How can we do this?

Note: Procedural defenses can’t be verified by a lab

Procedural defenses: ideas

- Each provisioning key/seed assigned to one date
  - Never provision two HSMs on same day
  - Could also provision by date + location

- Store provisioning keys on HSM
  - Must have way to back up or re-provision HSM
  - Procedural defenses, again

- Use distributed database
  - Keep track of used provisioning keys/seeds

Questions

- Is there some reason this won’t work?
- How hard will these be to allow in 208?
- Will these meet industry need?