Stateful Hash-Based Signature Schemes

Volker Krummel
Chapter Lead PQC at Utimaco
Stateful Hash-Based Signature Schemes

Dr. Volker Krummel, Chapter Lead PQC
07.11.2023
Agenda

1. Stateful Hash Based Signatures (s-HBS)
2. Limitations of s-HBS
3. Proper State Handling Approach
Cryptographic Hash Function H

**Hash function means...**
- $H : \{0,1\}^* \rightarrow \{0,1\}^{256}$
- a method of compressing strings
- input is called “message”, output is called “digest”

**Cryptographic means... (in this context)**
- **One-way:** Given D, hard to find M such that $H(M)=D$
- **Collision resistance:** Hard to find $M \neq M'$ for which $H(M)=H(M')$
- **Unpredictability:** $M \rightarrow H(R,M)$ unpredictable when $R$ is secret
- **Extraction:** if $M$ has high entropy then $H(M)$ is ~ uniform
Stateful Hash based Signatures

One Time Signature – Basic Idea

Stateful Hash based Signatures

Example: signing and verifying message “1”

<table>
<thead>
<tr>
<th>Signer: publish ‘1’ and its signature</th>
<th>Verifier: knows ( x_3 ) and ( x_1 )</th>
<th>Checks if</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_0 \rightarrow H \rightarrow x_1 )</td>
<td>( x_1 \rightarrow H \rightarrow x_2 )</td>
<td>( x_1 \rightarrow H \rightarrow x_2 \rightarrow H \rightarrow x_3 )</td>
</tr>
</tbody>
</table>

Combining many OTS Schemes

- OTS would require too many public keys
- Idea: build up a tree structure \( \rightarrow \) single public key

Challenges remain

- Keep track about which OTS private key was already used \( \rightarrow \) State handling
- Limited number of signatures
Pros and Cons of s-HBS

Pros of s-HBS

1. best ratio of pub key size and signature
2. well understood security guarantees
3. simple & mature design
4. already standardized
5. recommended as 1:1 substitution
   -> may skip hybrid-approach

Cons of s-HBS

1. Limited number of OTS-keys
   -> limited number of signing operations
2. Stateful

Holy Grail of PQC-Signatures!?
The Great Seal!?

"You chose wisely. But the Grail cannot pass beyond the great seal. That is the boundary, and the price of immortality."
Case Study Chip Manufacturer

Challenges of Distributed Sites

- Headquarter
- Development Site
- Manufacturing Site
- Supplier Site

Manufacturing Sites:
- HSM

Distribution and Connectivity:
- HSM

Network Connections:
- Dotted lines indicate connectivity among sites.

Geographical Locations:
- World map background.
Selection of PQC Algorithms by parameter (example)

PQC signature algorithms compared to ECC / RSA

- SPHINCS+256
- SPHINCS+192
- SPHINCS+128
- HSS/XMSS-MT
- LMS/XMSS
- RSA 2k
- RSA 8k
- Falcon-512
- Falcon-1024
- Dilithium2
- Dilithium3
- Dilithium5
- ECC 256
- ECC 384
- ECC 512
- ECC 2048
- RSA 2k
- RSA 8k
- Dilithium2
- Dilithium3
- Dilithium5
- Falcon-512
- Falcon-1024

Signature Size (Bytes)
Public Key Size (Bytes)
Thoughts on Limitation #1 – limited number of keys

- Restricts application to use cases with reliable estimation of number of signatures
  -> adds a further risk of running out of keys
- be as close as possible to real number of signatures
  -> keeps size of signatures low
- Works well for long term „static“ security use cases
  - Root-CA
  - Firmware Signing
- **Option1**: enable multi-tree variant
- **Option2**: establish procedure for key substitution (good practice!)
Thoughts on Limitation #2 – state handling

Backup & Restore

- Classical Backup & Restore procedures restore an old state -> violate the security requirement!
Thoughts on Limitation #2 – state handling

- simple Backup & Restore procedures restore an old state -> violate the security requirement!

- **Option 1**: adapt backup & restore procedure to support disaster recovery
  1. „know what you signed“
  2. Add-on: if double usage is detected -> **revoke the key**

- **Option 2**: establish a proper state handling mechanism
State Handling

OTS Keys must be used maximal once!

- Usage of multiple HSM instances
- Usage of Backup & Restore

Simple bookkeeping becomes complex

State handling must be
- Secure (must have)
- Flexible (disaster recovery / performance)

Design Properties of a Secure State Handling Architecture

1. Authentic and confidential end-to-end export and import of key / state information
   - Do not use asymmetric PQC algorithms – not an adequate level of maturity
   - Use symmetric cryptography (maximum entropy)
2. Establish a reliable trust relationship between the HSM instances (in secure environment)
   - Allows a highly flexible transfer even during operating in the field
3. Prevent replays – protect the freshness
4. Prepare for offline data – allow for external key storage
5. Separate keys and state information (least to know principle)
6. Asynchronous → no need for direct (real time) communication between HSMs
Proper State Handling Approach – Security is Paramount

1. Setup phase (set up trust relationship)
2. Generate key in HQ
3. Distribute subsets to destinations
4. Operate …

   1. If risk of key exhaustion at one site - Securely transfer further keys from any other site(s)
   2. If site will be shut down - Securely transfer remaining keys to other site(s)
   3. Attacks – e.g., if A replays key transfer -> blocked
   4. Risk of faulty app exhausting all keys - only import small portions of the key; keep rest offline
   5. If HSM is destroyed -> loss is limited to a well defined subset of the key

Logical connection (network, portable storage, …)

External key storage (optional)
State handling like this is not an option for your use case?

Then ...
Thoughts on Limitation #2 – state handling

Thoughts state handling - Overview

- simple Backup & Restore procedures / key distribution mechanisms restore an old state - > violate the security requirement!

Options

1. adapt backup & restore procedure to support disaster recovery (know what you signed)
2. establish a proper state handling mechanism
3. go for a stateless signature algorithm
Any more questions?
Thank you for your attention!