Hash-based Signatures

Cryptology (1)

Martin Stanek

2025

KI FMFI UK Bratislava

Introduction

- signature schemes resistant to quantum computers
  - security is not based on hardness of factorization, discrete log etc.
  - security based on properties of hash functions, such as preimage resistance, 2nd preimage resistance, collision resistance
- we discuss some schemes and their limitations
  - number of signing operations
  - state
  - key/signature length

One-time signatures

Lamport one-time signature scheme

- Lamport (1979)
- $f : X \to Y$ (for example a hash function)
- message/hash $m = m_1 \ldots m_n \in \{0,1\}^n$
- private key: $x_{i,j} \in_R X$
  for $i = 1, \ldots, n$, and $j \in \{0,1\}$
- public key: $y_{i,j} = f(x_{i,j})$
- signature $\sigma = (x_{1,m_1}, x_{2,m_2}, \ldots, x_{n,m_n})$
- verification of $\sigma = (\sigma_1, \ldots, \sigma_n)$ given m:

$$f(\sigma_i) \stackrel{?}{=} y_{i,m_i} \quad \forall i = 1, \ldots, n$$

Sizes (example)

- 256-bit hash function f and $n = 256$
- private key: $2 \cdot 256 \cdot 256 = 16\,384$ B
  - assuming $|x_{i,j}| = 256$ bits
- public key: 16 384 B
- signature length: 8 192 B
- the scheme is fast

Lamport scheme – properties and remarks

One-time signature scheme

- signing two messages (if they differ in more than one bit) ⇒ combine signatures to forge signature for a new message
- signing a hash does not help – signatures of $O(\lg n)$ messages are enough to cover 0 and 1 on almost all hash positions

Shorter keys

- private key (seed)
  - generate with PRNG or PRF
- public key (hash all values together)
  - $y = H(y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$
  - add values $y_{i,1-m_i}$ to the signature
  - σ is 2 times longer now
  - verification:
    compute $y_{i,m_i} = f(x_{i,m_i})$
    verify $y = H(y_{1,0}, y_{1,1}, \ldots, y_{n,0}, y_{n,1})$

Lamport scheme – improvement with a counter

- reducing the length of keys and signatures by half (approximately)
- add $\lfloor \lg n \rfloor + 1$ bits to the input, counting the number of 0 bits: $m' = m \parallel (\#_0 m)_2$
- let $n' = |m'|$
- private key: $x_i \in_R X$ for $i = 1, \ldots, n'$
- public key: $y_i = f(x_i)$
- signature is a sequence: $\sigma = \langle x_i \rangle_{i \in I}$, for indices $I = \{1 \leq i \leq n' \mid m'_i = 1\}$
- verification is obvious

Observation (why is this one-time signature scheme secure):

- m' contains at least one bit 1
- change from 0 to 1 in m – corresponding x_i is not in the signature
- change from 1 to 0 in m – more zeroes, hence at least one 0 changes to 1 in the counter and corresponding x_i is not in the signature

Winternitz one-time signature scheme (WOTS)

- reducing signature length and increasing time complexity (TMTO)
- multiple variants – what function is used and how it is iterated
  - example: $k \to H(k) \to H^2(k) \to \ldots \to H^{w-1}(k)$
  - we use slightly more complex iteration (but the main idea is the same)
  - impact on security assumptions (collision resistance vs. PRF)
  - standard model (no ROM)
- key-dependent function $f : X \times X \to X$ (like MAC), notation: $f(k, x) = f_k(x)$
- iterating f for $k \in X$ and $x \in X$:
  - $f_k^0(x) = k \quad \to \quad f_k^1(x) = f_k(x) \quad \to \quad f_{k(x)}^2 = f_{f_{k(x)}}(x) \quad \to \quad \ldots$
  - for $r \geq 1$: $\quad f_k^r(x) = f_{f_k^{r-1}(x)}(x)$

WOTS – generalized checksum, keys and signature

- Winternitz parameter $w > 1$
  for example $w = 16$
- w-ary representation of m:
  $m = (m_1, \ldots, m_{l_1})$, where $0 \leq m_i < w$
- checksum: $C = \sum_{i=1}^{l_1} (w - 1 - m_i)$
- let $l = l_1 + l_2$, where l_2 is a max length of C (w-ary representation)

Keys

- private key: $k_1, \ldots, k_l \in_R X$
- public key: $(x, y_1, \ldots, y_l)$, where
  - $x \in_R X$
  - $y_i = f_{k_i}^{w-1}(x)$, for $i = 1, \ldots, l$

$\mathbf{Sig_{sk}}(\boldsymbol{m})$

1. split into blocks: $m \parallel C \mapsto m_1, \ldots, m_l$
2. signature
   $\sigma = (\sigma_1, \ldots, \sigma_l) = \left(f_{k_1}^{m_1}(x), \ldots, f_{k_l}^{m_l}(x)\right)$

$\mathbf{Vrf_{pk}}(\boldsymbol{m}, \boldsymbol{\sigma})$

1. split into blocks: $m \parallel C \mapsto m_1, \ldots, m_l$
2. verify

$$f_{\sigma_i}^{w-1-m_i}(x) \stackrel{?}{=} y_i, \quad \text{for } i = 1, \ldots, l$$

WOTS – remarks

- correctness is trivial
- the scheme is insecure without the checksum:
  - take σ_i for m_i and for any $m'_i > m_i$ compute

$$\sigma'_i = f_{k_i}^{m'_i}(x) = f_{\sigma_i}^{m'_i - m_i}(x)$$

  - σ'_i is a correct signature for i-th block m'_i
  - the checksum prevents these shifts
- still one-time scheme

WOTS – a sample parameters instantiation

- let $X = \{0,1\}^{256}$, $|m| = 256$, and $w = 16$
  - $l_1 = 256/4 = 64$, $l_2 = \lfloor \lg(64 \cdot 15)/4 \rfloor + 1 = 3 \quad \Rightarrow \quad l = 67$
- keys:
  - private key: $l \cdot 256 = 2\,144$ B
  - public key: $(l + 1) \cdot 256 = 2\,176$ B
- signature: $l \cdot 256 = 2\,144$ B
  - approximately $\lg w = 4$ times shorter than the original Lamport scheme
- avg. speed – comparison with the original Lamport scheme
  - signing: $\approx l \cdot w/2$ calls of f vs. 0
  - verification: $\approx l \cdot w/2$ calls of f vs. $|m|$ calls (≈ 536 vs. 256)
    - WOTS is $\approx w/(2 \cdot \lg w)$ times slower

- Hülsing (2013)
- similar to WOTS, different iteration of f ⇒ tighter security proof
  - weaker or more standard security assumptions (f properties)
  - WOTS$^+$ as a replacement for WOTS in other schemes
- iteration of $f : K \times X \rightarrow X$
  - input: key $k \in K$, $x \in X$, counter $i \in \mathbb{N}$, randomization values $\boldsymbol{r} = (r_1, ..., r_j)$ for $j \geq i$
  - computation:

$$
\begin{aligned}
c_k^0(x, \boldsymbol{r}) &= x \\
c_k^1(x, \boldsymbol{r}) &= f_k\left(c_k^0(x, \boldsymbol{r}) \oplus r_1\right) \\
&\dots \\
c_k^i(x, \boldsymbol{r}) &= f_k\left(c_k^{i-1}(x, \boldsymbol{r}) \oplus r_i\right)
\end{aligned}
$$

WOTS$^+$ – keys, signature, and verification

- parameters w and $l = l_1 + l_2$ just like in WOTS
- private key: $x_1, \ldots, x_l \in_R X$
- public key: $((\boldsymbol{r}, k), y_1, \ldots, y_l)$
  - $k \in_R K$
  - $\boldsymbol{r} = (r_1, \ldots, r_{w-1})$, where $r_i \in_R X$
  - $y_i = c_k^{w-1}(x_i, \boldsymbol{r})$, for $i = 1, \ldots, l$
- checksum C (just like in WOTS)
  - the checksum ensures that for given $m_1, \ldots, m_l$ any other message contains at least one $m_j' < m_j$.

$\mathbf{Sig_{sk}}(\boldsymbol{m})$

1. split into blocks: $m \parallel C \mapsto m_1, \ldots, m_l$
2. signature $\sigma = (\sigma_1, \ldots, \sigma_l) = \left(c_k^{m_1}(x_1, \boldsymbol{r}), \ldots, c_k^{m_l}(x_l, \boldsymbol{r})\right)$

$\mathbf{Vrf_{pk}}(\boldsymbol{m}, \boldsymbol{\sigma})$

1. split into blocks: $m \parallel C \mapsto m_1, \ldots, m_l$
2. verify

$$c_k^{w-1-m_i}\left(\sigma_i, \boldsymbol{r}_{m_i+1,w-1}\right) \stackrel{?}{=} y_i \quad \forall i = 1, \ldots, l$$

where $\boldsymbol{r}_{m_i+1,w-1} = \left(r_{m_i+1}, \ldots, r_{w-1}\right)$

WOTS$^+$ – remarks

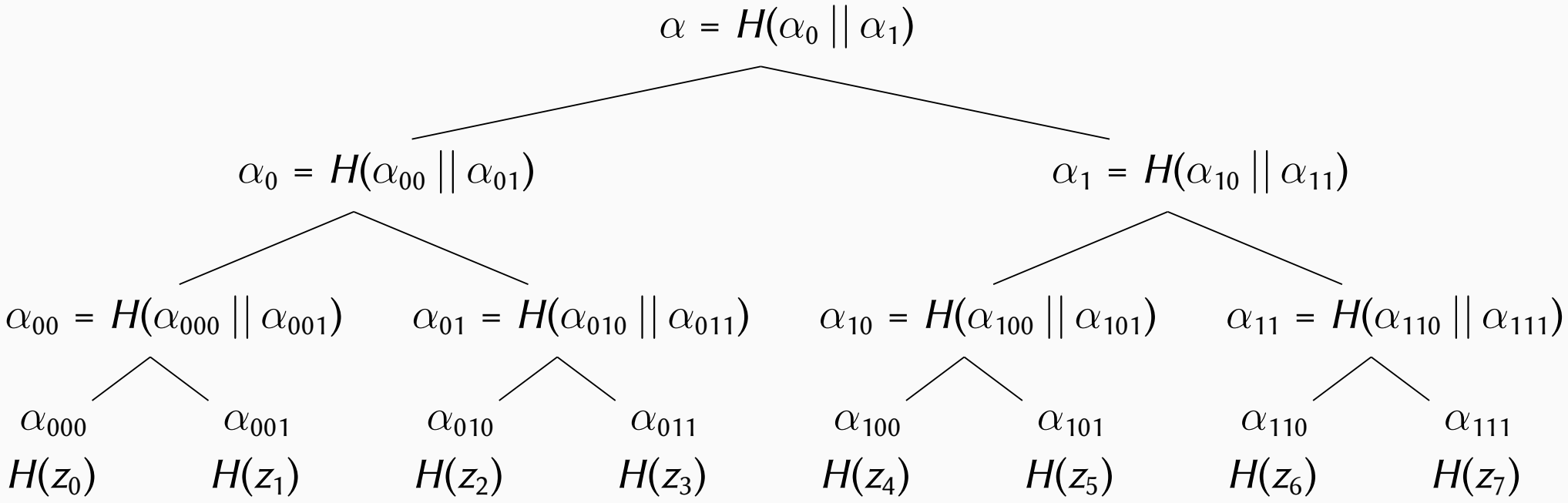
- reducing the key length:
  - all values y_i in WOTS and WOTS$^+$ are computed in verification, the public key can be replaced by their hash
  - $\boldsymbol{r}$ can be generated from a seed (PRNG or PRF)
  - similarly for values $x_1, \ldots, x_l$ in the private key

Multiple-time signatures (with state)

Merkle hash tree

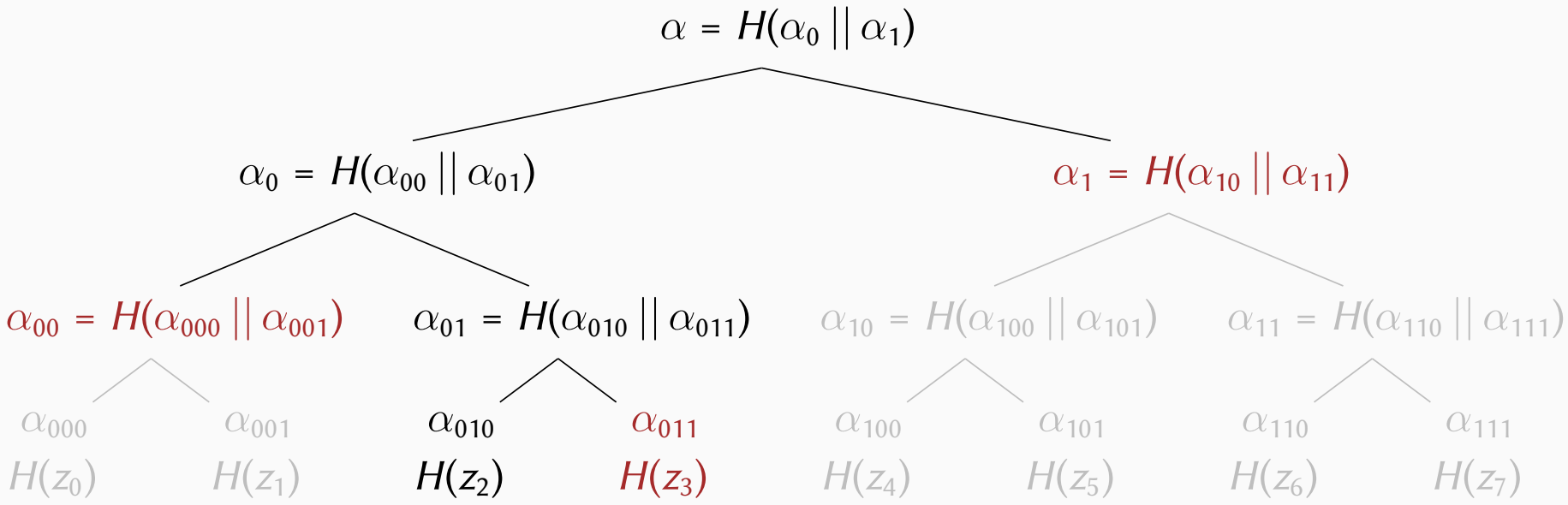
- Merkle (1979)
- signature scheme for a single message is impractical
- multiple one-time signature schemes
  - combined together into one tree-like structure
- Merkle hash tree – many applications, e.g., file systems (ZFS), BitTorrent, Bitcoin, Git, …
- binary tree of hash values:
  - input: data $z_0, \ldots, z_{2^h-1}$, for $h \geq 1$ (we assume 2^h input data for simplicity)
  - H – hash function
  - values in the leaf nodes: $H(z_i)$, for $i = 0, \ldots, 2^h - 1$
  - value in an inner node v: $H(a \parallel b)$,
    where a (or b) is the value of the left (or right) child of v

Example: Merkle hash tree for $h = 3$



Merkle hash tree – authentication path

- authentication path for a given leaf:
  - a sequence of sibling nodes values on the path to the root
  - it allows to compute the root value from the leaf and additional h values



authentication path for α_{010}: $\text{auth}(\alpha_{010}) = (\alpha_{011}, \alpha_{00}, \alpha_1)$

Merkle Signature Scheme (MSS)

- leaf data – public keys of OTS schemes
- public key: root value of the Merkle hash tree
- private key:
  - key for some (suitable) algorithm to generate OTS schemes
  - alternatively a sequence of OTS schemes private keys
- we are able to sign up to 2^h messages:
  - use OTS schemes sequentially one by one
  - saved state – counter of already used OTS schemes
- a signature contains:
  - signature using the next OTS scheme
  - public key of this OTS scheme
  - authentication path

MSS – verification

- input:
  - MSS public key (root value)
  - message
  - signature in OTS scheme and its public key
  - authentication path
- verification steps:
  1. verify signature in OTS scheme
  2. compute a root value (from public key and authentication path)
  3. compare the root value with the public key of MSS
- security depends on properties of these elements:
  - H used in Merkle hash tree
  - used OTS scheme
  - cryptographic constructions used in generation of OTS schemes

MSS – remarks

- all OTS schemes and entire tree must be generated for MSS instantiation
  - limit on h (big trees are impractical), and therefore on the number of signatures
- authentication path computation:
  - Merkle tree traversal – sequential computation of authentication paths
  - possible in linear time and memory $O(h)$ (per authentication path)
- What if 2^h OTS schemes were used?
  - generate a new instance (tree) and distribute public key
  - use the last OTS scheme to sign a new tree
  - create a sufficiently large instance such so this does not happen
- signatures are longer (authentication path) – h hash values
  - size of the tree impacts the length of the signature

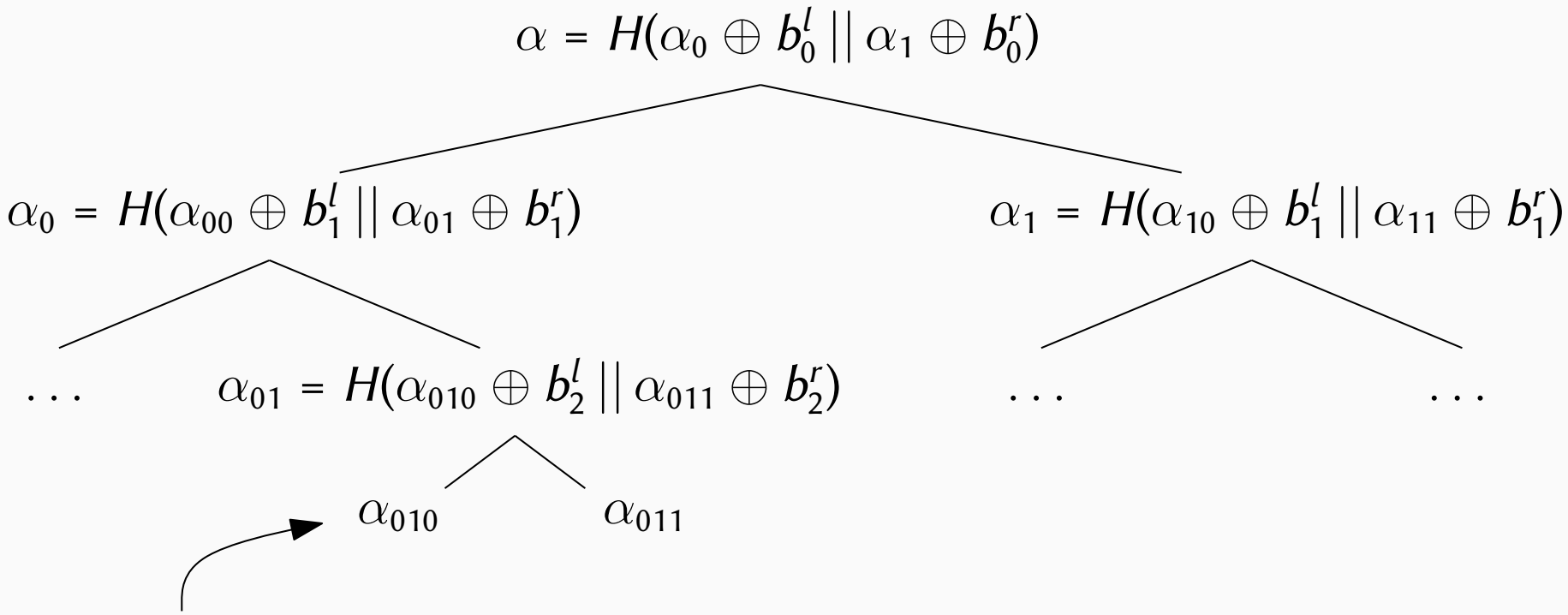
MSS – remarks (2)

- if WOTS (WOTS$^+$) is used in MSS
  - public key of OTS scheme is computed from a signature (beside x for WOTS or $(\boldsymbol{r}, k)$ for WOTS$^+$)
  - signature is shorter, this idea used, for example, in XMSS
- problem in MSS: no longer one-time scheme but now it is *stateful*
  - important for security: avoid using one OTS scheme multiple times
  - important for efficiency: authentication path computation
- state might be acceptable for some use cases
  - for example CA signs certificates in HSM
- in general having state is undesirable and potential problem
  - load-balancing, system recovery from backups (old state), etc.

XMSS (eXtended Merkle Signature Scheme)

- Buchmann, Dahmen, Hülsing (2011)
- RFC 8391, NIST SP 800-208
- modification of Merkle hash tree
  - xor *masks* when aggregating
  - L-trees hanging in leaves – storing public keys for WOTS$^+$ schemes (hashing public key – useful in security proofs)
  - assumption of second preimage resistance is sufficient, instead of collision resistance
- masks (random string with suitable length)
  - left and right masks chosen for each non-leaf height

XMSS tree



root of L-tree (for public key in WOTS$^+$ scheme)
(new set of masks, shared among all L-trees)

XMSS – remarks

- L-tree
  - construction like XMSS tree (new independent masks, but same for each L-tree)
  - leaves – public keys for WOTS scheme $(x, y_1, \ldots, y_l)$
  - not necessary a power of 2, but still binary tree
- still stateful
- example parameters (XMSS-SHA2_20_256, one of “required” in RFC 8391):
  - $h = 20$, capacity for 2^{20} signatures
  - $w = 16$, SHA2-256
  - signature size: 2 820 bytes

Few-time signatures (without state)

HORS (Hash to Obtain Random Subset)

- reveal only a subset of private key values for each signature
- hash function $H : \{0,1\}^* \to \{0,1\}^n$
- preimage resistant function $f : X \to X$, similar to Lamport scheme
- parameters:
  - k – $H(m)$ is divided into k chunks $h_0, \ldots, h_{k-1}$, i.e., $k \mid n$
  - $a = n/k$ (chunk length)
  - $t = 2^a$ – the number of private key components

Gen

- private key: $x_0, \ldots, x_{t-1} \in_R X$
- public key: $y_0, \ldots, y_{t-1}$, where $y_i = f(x_i)$

$\mathbf{Sig_{sk}}(\boldsymbol{m})$

1. split $H(m) \mapsto h_0, h_1, \ldots, h_{k-1}$
2. $\sigma = \left(x_{h_0}, x_{h_1}, \ldots, x_{h_{k-1}}\right)$

$\mathbf{Vrf_{pk}}(\boldsymbol{m}, \boldsymbol{\sigma})$

1. let $\sigma = (\sigma_0, \ldots, \sigma_{k-1})$
2. split $H(m) \mapsto h_0, h_1, \ldots, h_{k-1}$
3. verify $f(\sigma_i) \stackrel{?}{=} y_{h_i} \quad \forall i = 0, \ldots, k-1$

HORS remarks

- sizes (let's assume $|X| = 2^{256}$, $n = 256$, $k = 16$, $t = 2^{256/16} = 2^{16}$):
  - private key, public key: tn bits (2 097 152 bytes)
  - signature: kn bits (512 bytes)
- What is the probability that after signing $c = 8$ messages, all hash chunks for a message m' are presented in previous signatures?

$$\Pr \leq \left(\frac{ck}{t}\right)^k \qquad = \left(\frac{8 \cdot 16}{2^{16}}\right)^{16} = 2^{-144}$$

  - the probability increases with larger c, and multiple attempts to find m'
- HORST: fix unacceptable public key size
  - combining HORS with a Merkle tree: leaves are the private key values $x_0, \ldots, x_{t-1}$
  - public key is the root of the Merkle tree
  - signature: k signature leaves, together with their authentication paths

FORS (Forest of Random Subsets)

FORS is used in SPHINCS+ scheme, improvement of HORST scheme

- better security, signature speed and size (smaller parameters)

Setup:

- k trees are used instead of 1
- k sets of t private key values, each set is used to form a tree
- public key: a single hash value of root nodes
- private key: pseudorandom seed

Signature:

- k private key values with authentication paths
- h_i selects private key value in the i-th tree

Verification:

- compute root nodes $^{(*)}$ of all trees and verify the public key

  $^{(*)}$ using *candidate* public keys computed from the signature, message and authentication paths

Stateless schemes

Goldreich's idea

- goals: no state in the scheme & sign large number of messages
- huge binary tree, for example $h = 256$ (2^{256} leaves)
  - every node represents an OTS scheme
  - every scheme can be generated using private key and position of the node (PRF/ PRNG)
- private key: random bit string
- public key: public key of the OTS scheme in the root of the tree (Y)

Signing message m

1. $H(m)$ is an index to some leaf β in the tree
2. compute keys for OTS schemes for all nodes (and their siblings) on path from β to the root (public keys $Y_h^l, Y_h^r, \ldots, Y_1^l, Y_1^r$)
3. sign $H(m)$ using OTS scheme for $\beta \mapsto \sigma_m$
4. every sibling pair of public keys is signed by OTS scheme for their parent node, obtaining signatures $(\sigma_0, \sigma_1, \ldots, \sigma_{h-1})$, where σ_0 is the signature in the root
5. signature: $\sigma = \left(\sigma_m, \sigma_0, \ldots, \sigma_{h-1}, Y_1^l, Y_1^r, \ldots, Y_h^l, Y_h^r\right)$

Verification

- input:
  - message m (or $H(m)$)
  - public key for root OTS scheme Y
  - signature $\sigma = \left(\sigma_m, \sigma_0, \ldots, \sigma_{h-1}, Y_1^l, Y_1^r, \ldots, Y_h^l, Y_h^r\right)$
- verification steps:
  1. determine a leaf β from $H(m)$
  2. verify signature σ_m using the correct Y_h^l or Y_h^r
  3. verify *certification path*, i.e., signatures $(\sigma_0, \ldots, \sigma_{h-1})$, using public keys from σ and using public key Y for σ_0 verification
- state not needed, probability of collision is negligible
- signature in any node is always the same
  - public keys of children are certified
  - OTS scheme is sufficient

Stateless scheme optimizations (1)

- problem with the previous approach: very long signatures, not practical
- usually a combination of multiple techniques
- deterministic/pseudorandom leaf selection in the tree
- using few-time signature schemes instead of WOTS/WOTS$^+$ in leaves
  - HORS, HORST, FORS, etc.
  - just for message signing
  - secure signing of few messages
  - smaller tree (less leaves)
- change the structure from a single huge tree to multiple levels of smaller Merkle trees
  - the size of authentication path is smaller than the size of equally long (element-wise) chain of one-time signatures

Hypertree

- multiple layers of small Merkle trees
- OTS: signing the roots of lower trees
- few-time signatures at the bottom
- randomized hashing of the message (random selection of FTS scheme)
- all schemes (their private keys) are derived from a seed
- limit the number of messages that can be signed, for security estimates

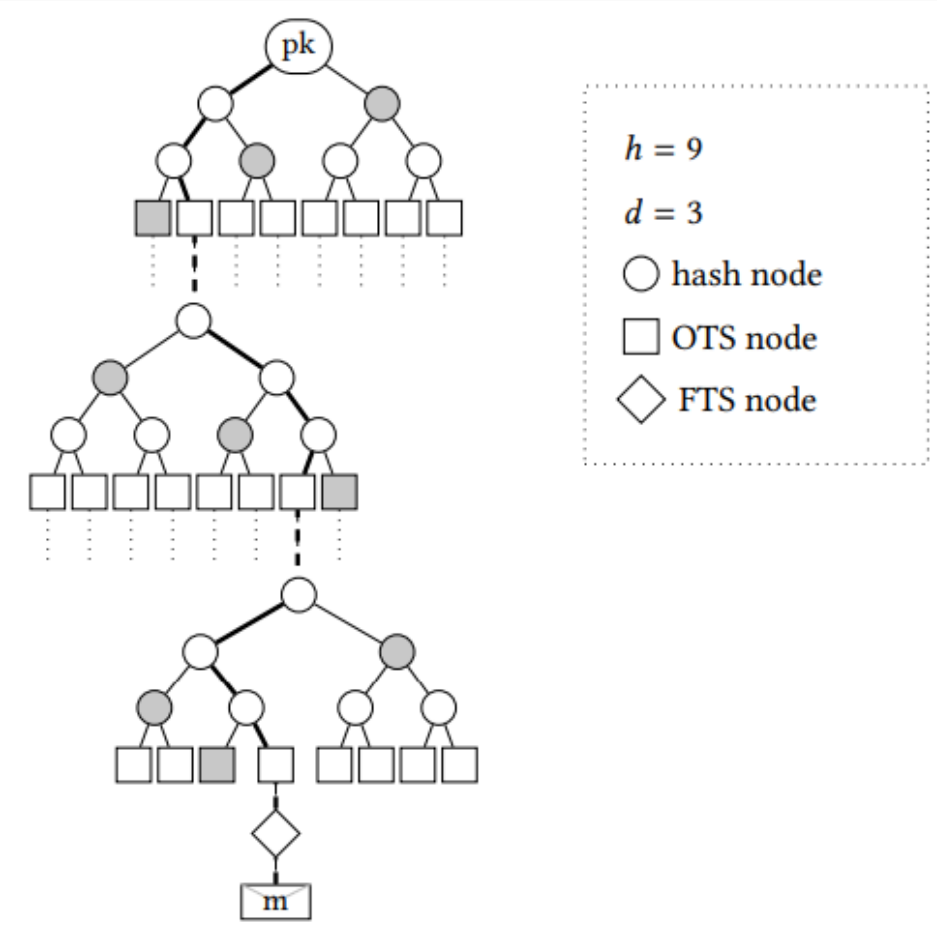


Figure 1: An illustration of a (small) SPHINCS structure.

Figure source:
D. Bernstein et al.: The SPHINCS+ Signature Framework (2019)

PQC standardization

- SPHINCS+ if one of the three signature schemes selected for standardization
  - FIPS 205 Stateless Hash-Based Digital Signature Standard (2024)
  - SLH-DSA based on SHAKE or SHA2 hash functions
- NIST Third Round Status Report:
  SPHINCS+ was selected for standardization because it provides a workable (albeit rather large and slow) signature scheme whose security seems quite solid and is based on an entirely different set of assumptions than those of our other signature schemes to be standardized.
- SPHINCS+: hypertree, OTS: WOTS+, FTS: FORS

SLH-DSA parameters

Table 2. SLH-DSA parameter sets

	n	h	d	h'	a	k	lg_w	m	security category	pk bytes	sig bytes
SLH-DSA-SHA2-128s SLH-DSA-SHAKE-128s	16	63	7	9	12	14	4	30	1	32	7 856
SLH-DSA-SHA2-128f SLH-DSA-SHAKE-128f	16	66	22	3	6	33	4	34	1	32	17 088

Table source: NIST, FIPS 205 (2024)

- "s" variant optimized for size, "f" variant optimized for speed
- security category 1 means 128-bit security, with limit 2^{64} signatures
- hypertree: d layers, h' height of a single layer, $h = d \cdot h'$
- FORS parameters: a, k; WOTS+ parameter: w
- m – the length of the message digest (in bytes)

Conclusion

- hash-based signature schemes
  - limited number of signatures
  - simple principles, usually simple requirements/assumptions
  - optimization aiming at practicality and weaker assumptions complicate construction

Exercises

1. WOTS$^?$ *simplifies calculation of* C *in WOTS scheme in the following way:* $C = \sum_{i=1}^{l_1} m_i$. *The rest of the scheme remains intact. Is this variant secure (explain why)?*
2. *Consider HORS scheme with:* $|X| = 2^{256}$, $n = 256$, $k = 16$, $t = 2^{256/16} = 2^{16}$. *Implement an experiment that answers the following question: How many signatures we need to collect on average so we can produce a valid signature of some fixed message* m'*?*
3. *Calculate the size of signatures in Goldreich's construction for* $h = 256$. *Assume that WOTS/*WOTS$^+$ *is used for OTS with* $w = 16$ *and SHA2-256.*