SPHINCS:
practical stateless hash-based signatures

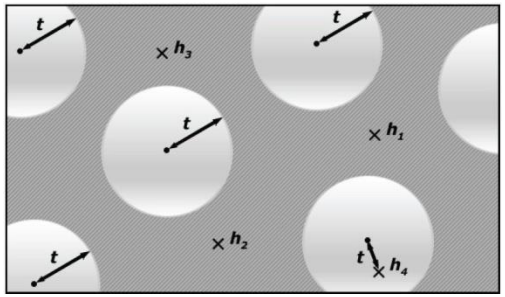
**Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing,
Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou,
Michael Schneider, Peter Schwabe, Zooko Wilcox-O'Hearn**

Where innovation starts

Post-Quantum Signatures

Lattice, MQ, Coding

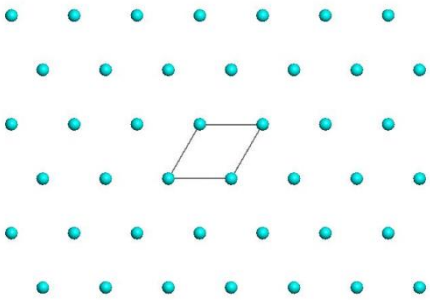
- Signature and/or key sizes
- Runtimes
- Secure parameters



$$y_1 = x_1^2 + x_1x_2 + x_1x_4 + x_3$$

$$y_2 = x_3^2 + x_2x_3 + x_2x_4 + x_1 + 1$$

$$y_3 = \ldots$$



Hash-based Signature Schemes [Mer89]

- Post quantum
- Only secure hash function
- Security well understood
- Fast
- Stateful

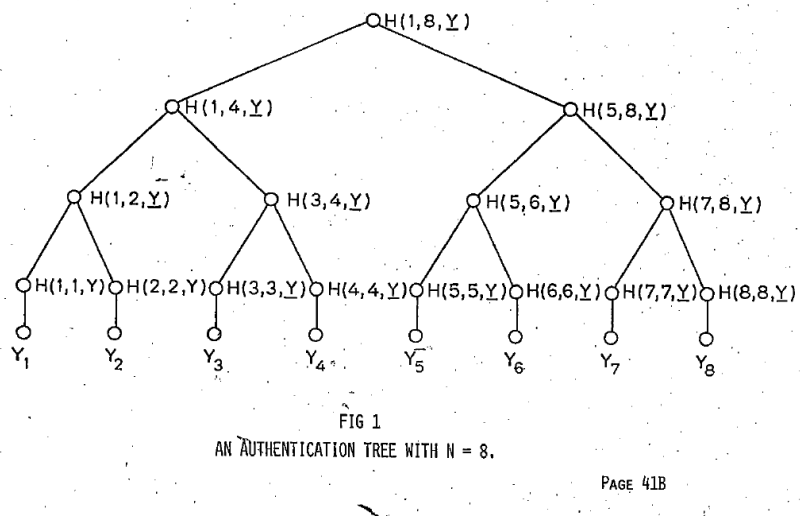
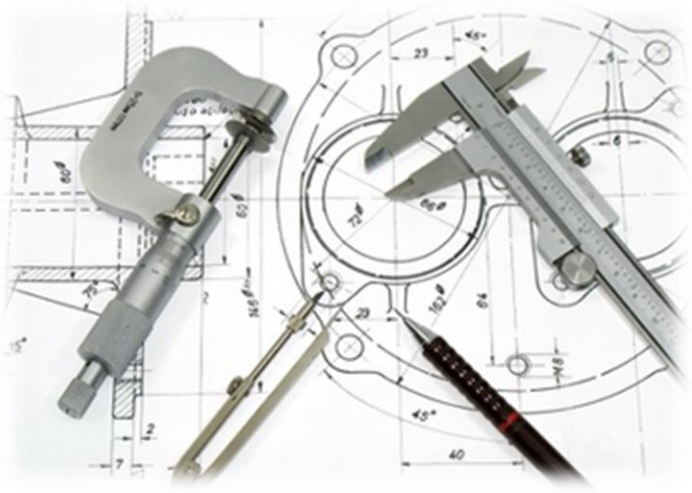


FIG 1
AN AUTHENTICATION TREE WITH N = 8.

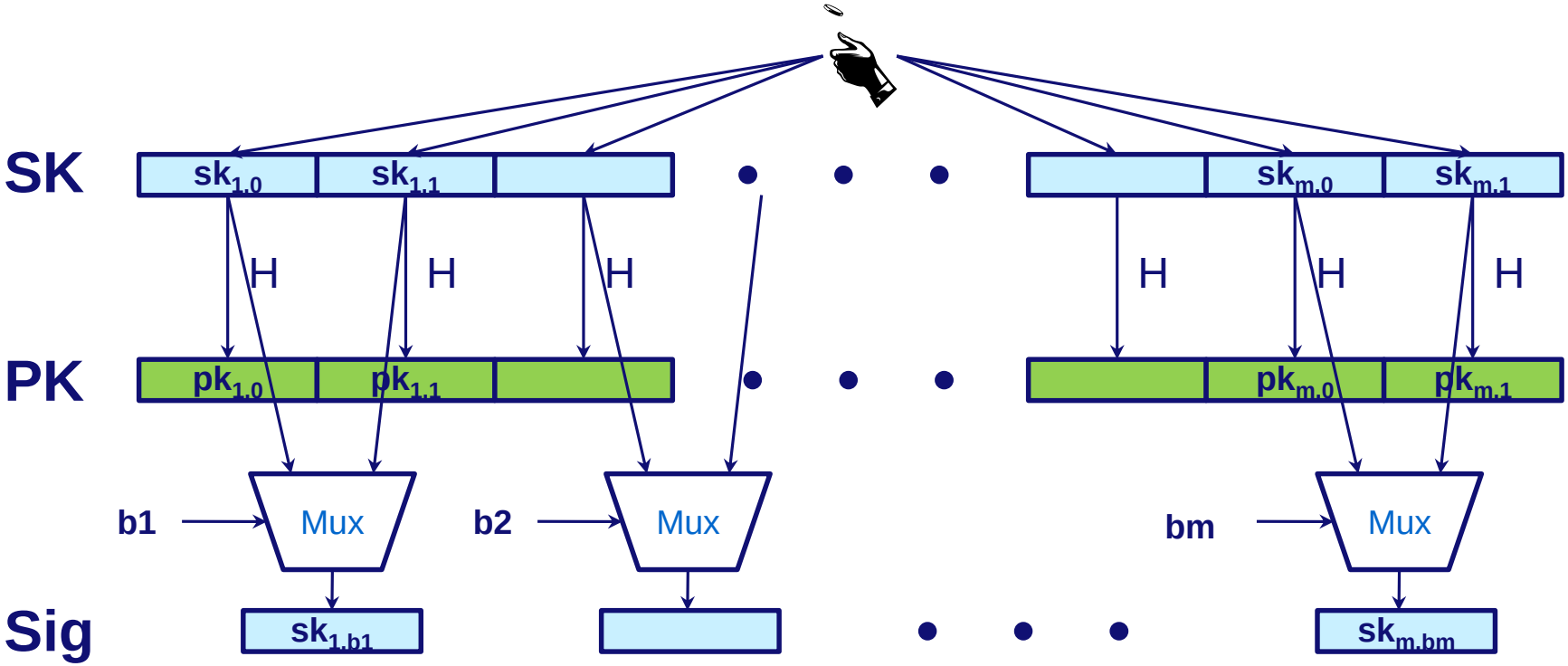
PAGE 41B

Basic Construction

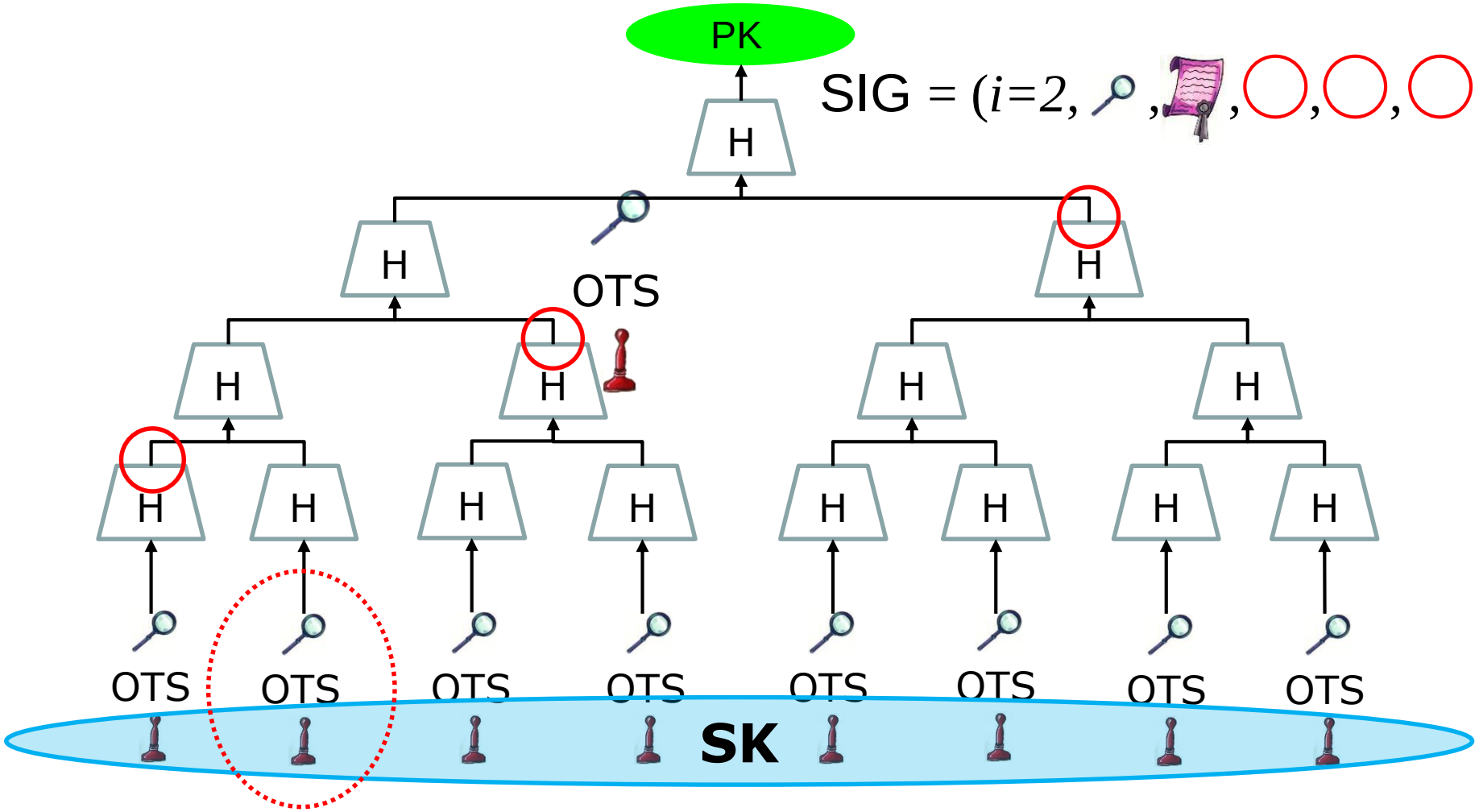


Lamport-Diffie OTS [Lam79]

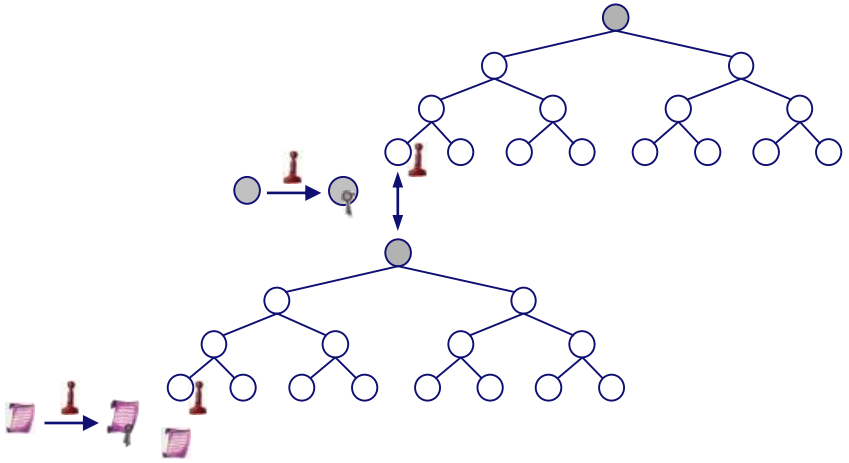
Message M = b1,…,bm, OWF H * **= n bit**



Merkle's Hash-based Signatures



Known Improvements



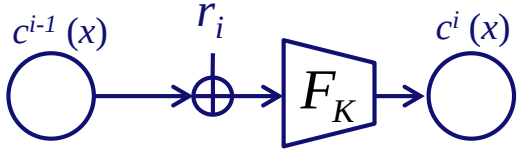
WOTS
Function Chain

Function family: $\mathcal{F}_n = \{F_K : \{0,1\}^n \rightarrow \{0,1\}^n \mid K \in \{0,1\}^{n'}\}$

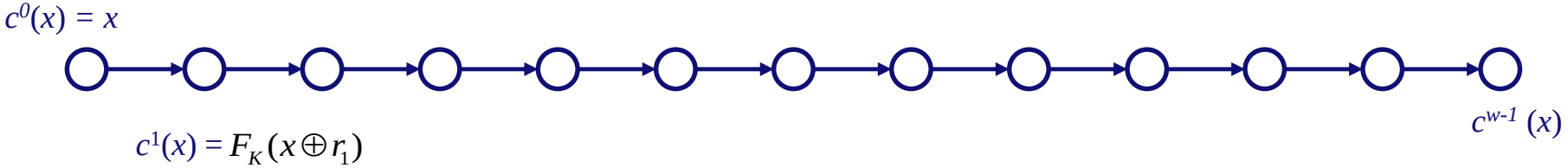
Formerly: $c^i(x) = F_K(c^{i-1}(x)) = \underbrace{F_K \circ F_K \circ \ldots \circ F_K(x)}_{i-times}, K \in \{0,1\}^{n'}$

WOTS$^+$

For $w \geq 2$ select $\mathcal{R} = (r_1, \ldots, r_{w-1}) \in \{0,1\}^{n \times w-1}, K \in \{0,1\}^{n'}$



$$c^i(x) = F_K(c^{i-1}(x) \oplus r_i)$$

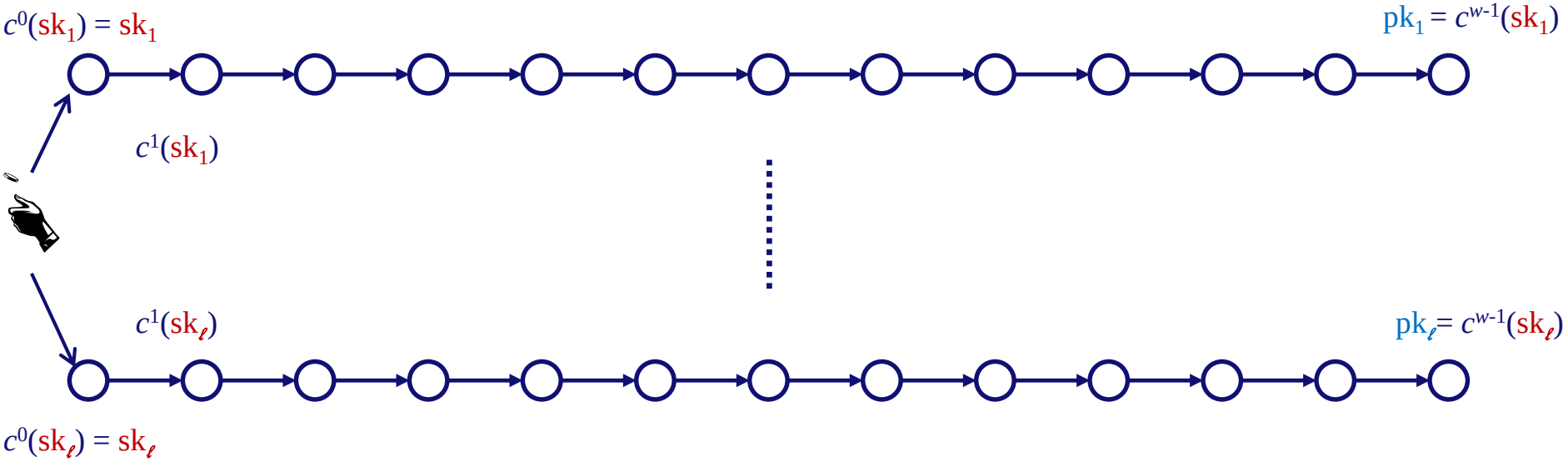


WOTS$^+$

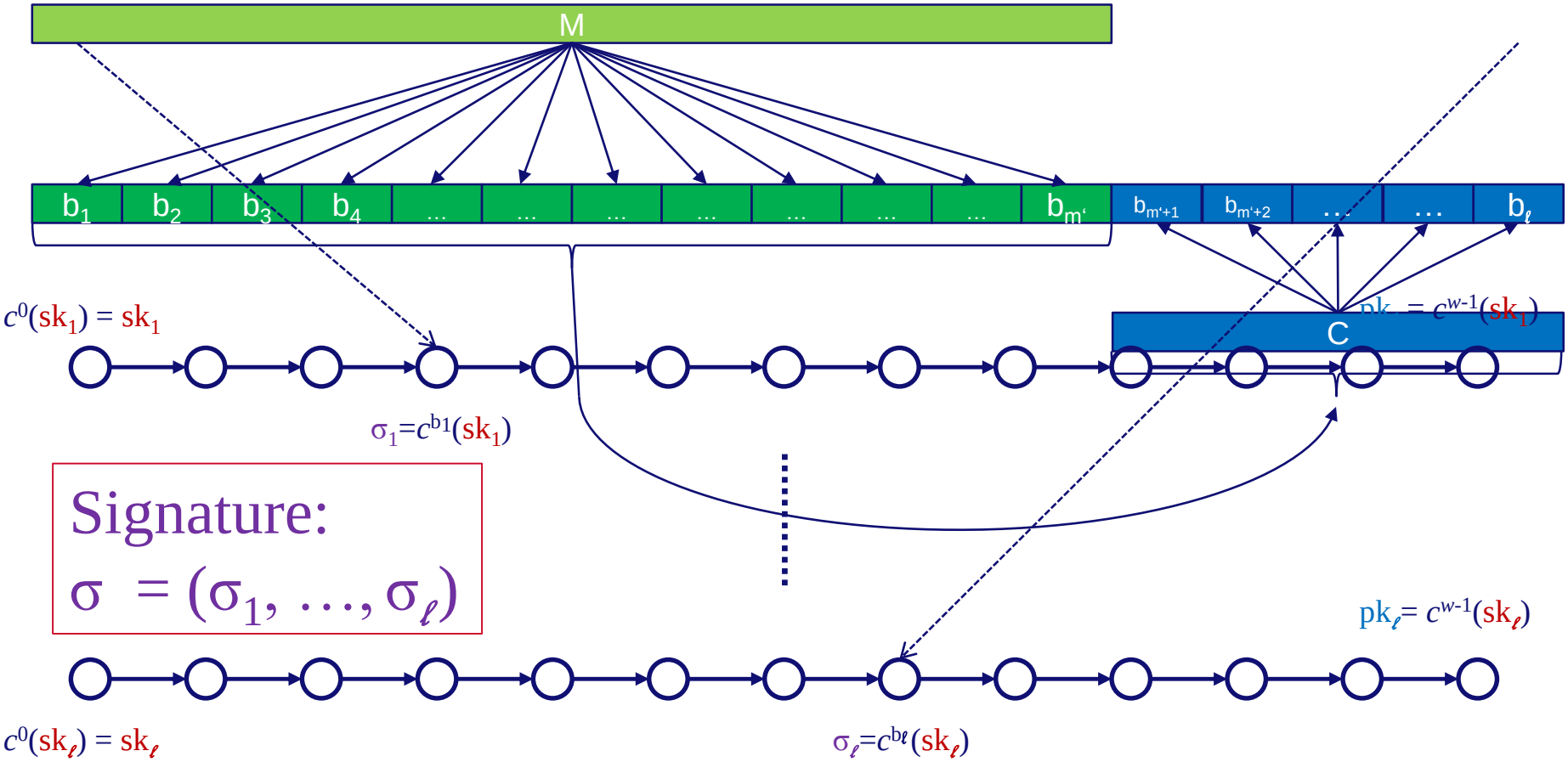
Winternitz parameter w, security parameter n, message length m, function family

$$\mathcal{F}_n = \{F_K : \{0,1\}^n \rightarrow \{0,1\}^n \mid K \in \{0,1\}^{n'}\}$$

Key Generation: Compute ℓ, sample K, sample $\mathcal{R}$

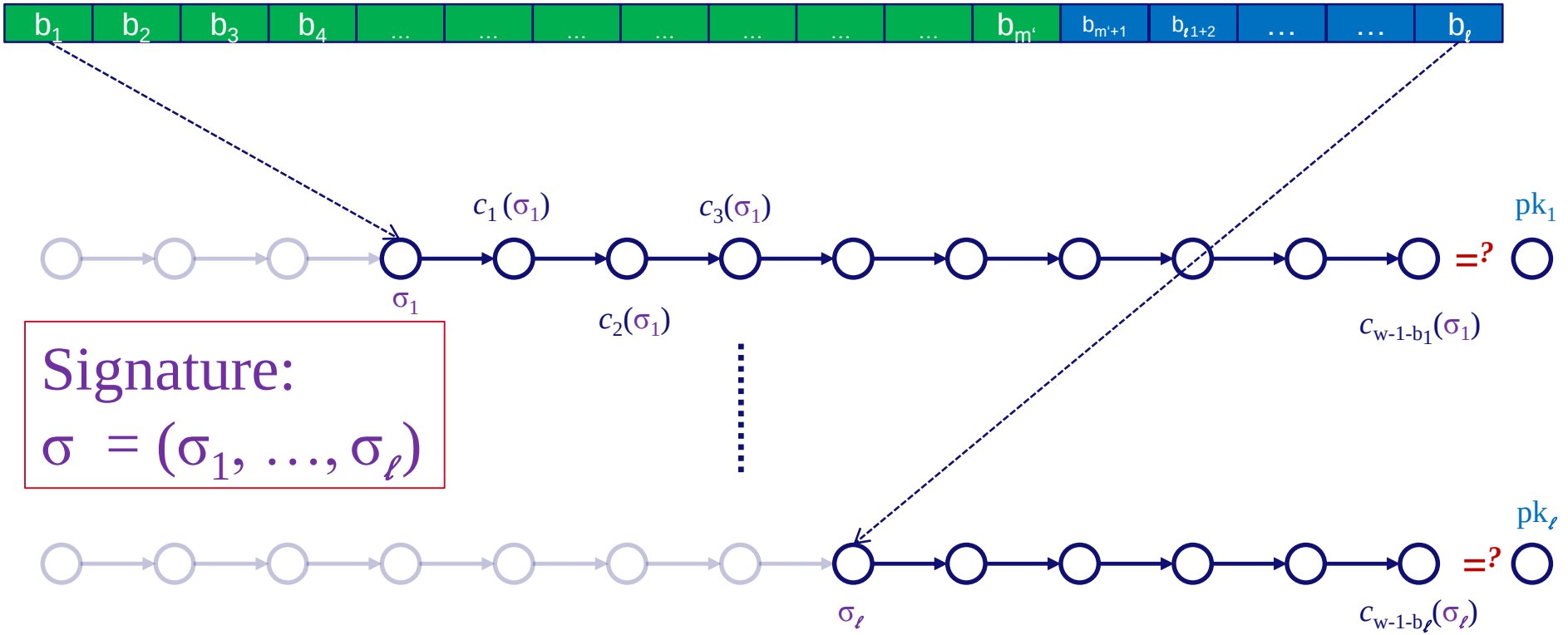


WOTS+ Signature generation

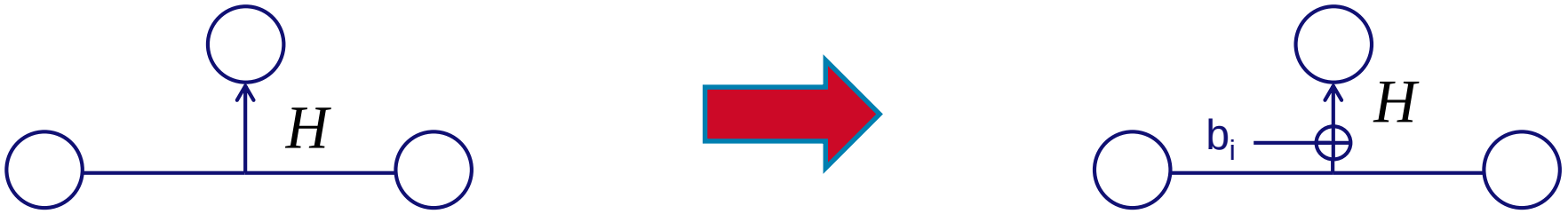


WOTS+ Signature Verification

Verifier knows: M, w

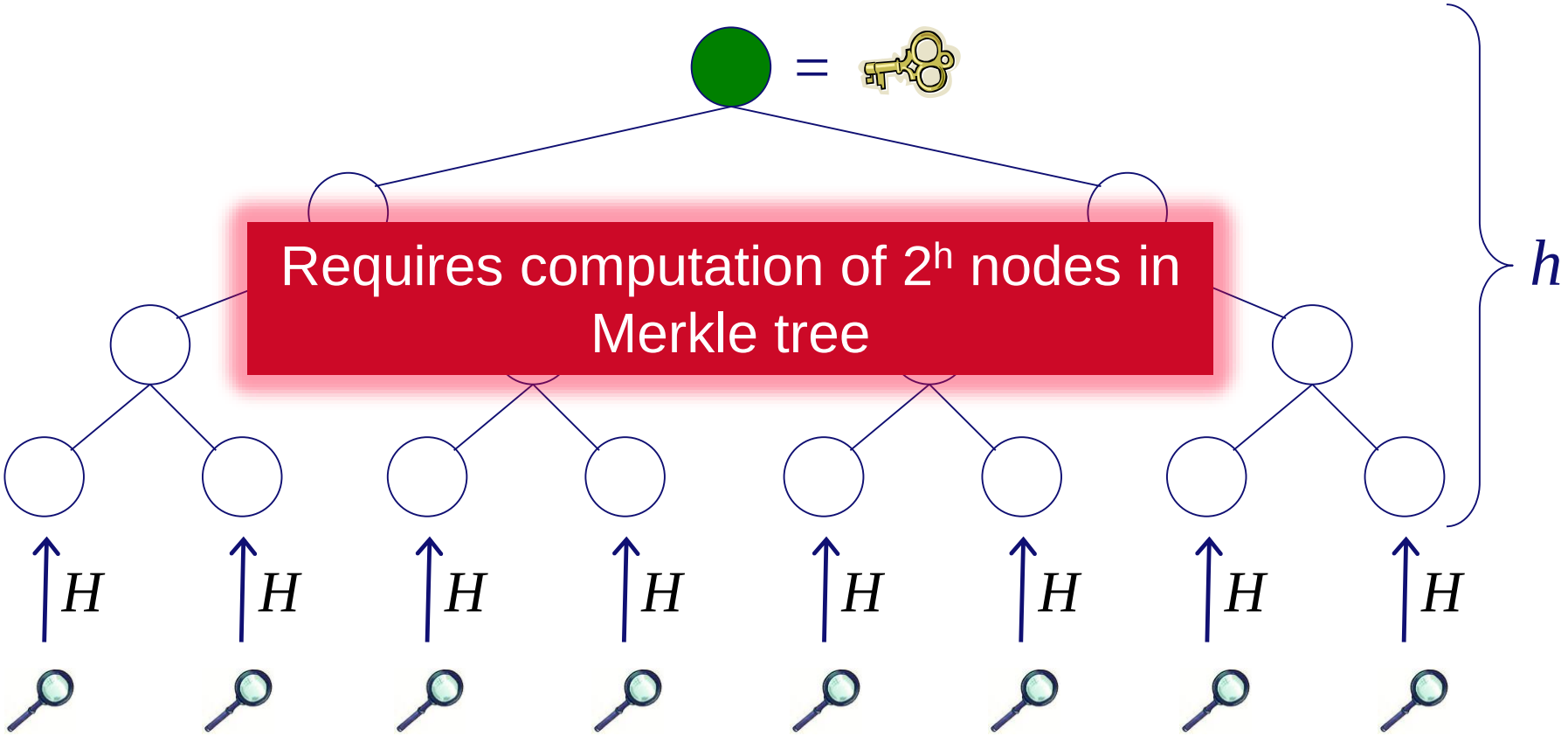


MSS-SPR [DOTV08]



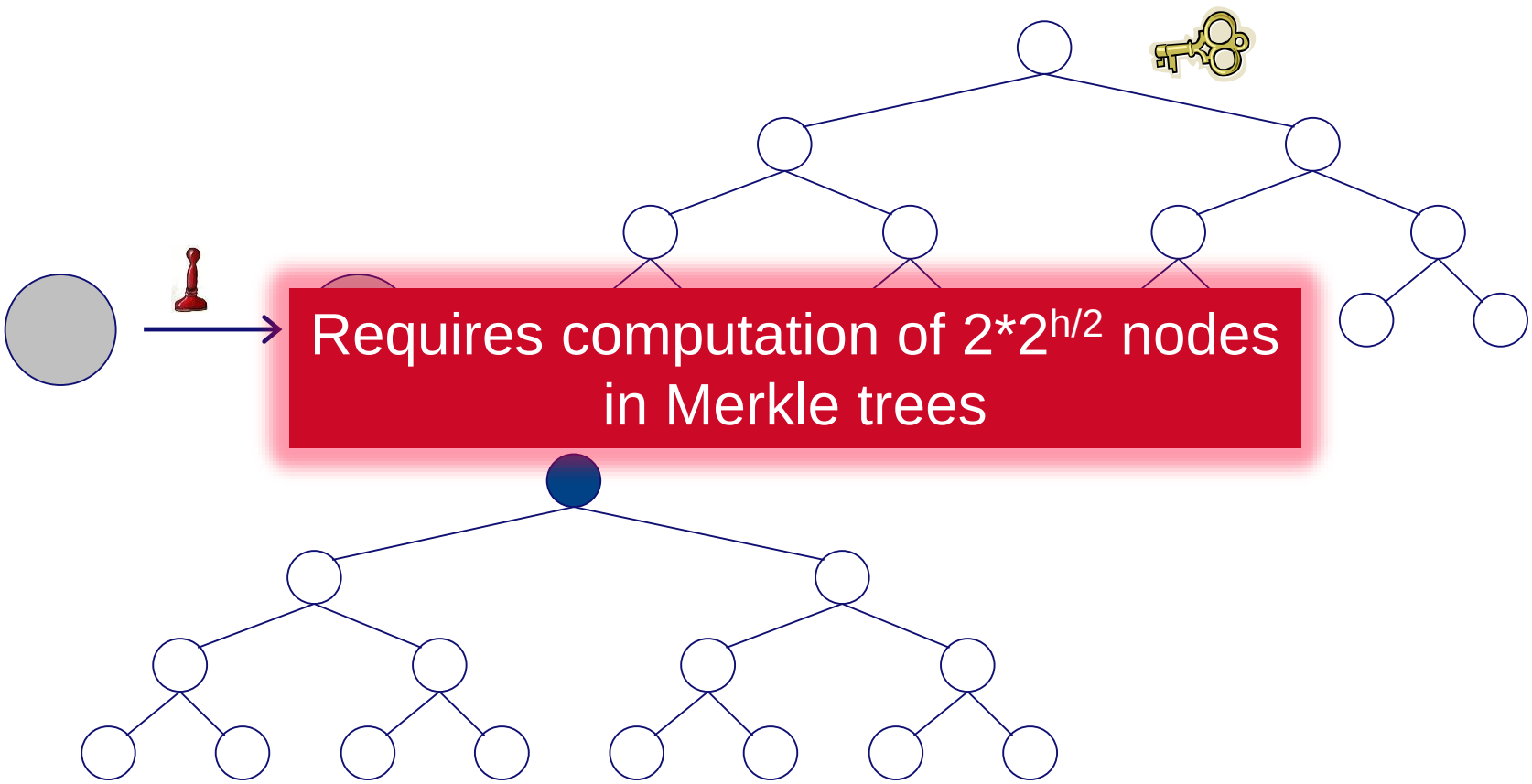
- Hashing one-time PK‘s using tree
- Requirements: CRHF -> SPRHF
- PK includes ~h additional values

XMSS Public Key Generation



Requires computation of 2^h nodes in Merkle tree

Two Layer
Key generation



About the state

- **Used for security:**
  - Stores index i ⇒ Prevents using one-time keys twice.
- **Used for efficiency:**
  - Stores intermediate results for fast Auth computation.
- **Problems:**
  - Load-balancing
  - Multi-threading
  - Backups
  - Virtual-machine images
  - ...
- **“Huge foot-cannon” (Adam Langley, Google)**
- **Not only a hash-based issue!**


ELIMINATE
THE STATE

Protest?

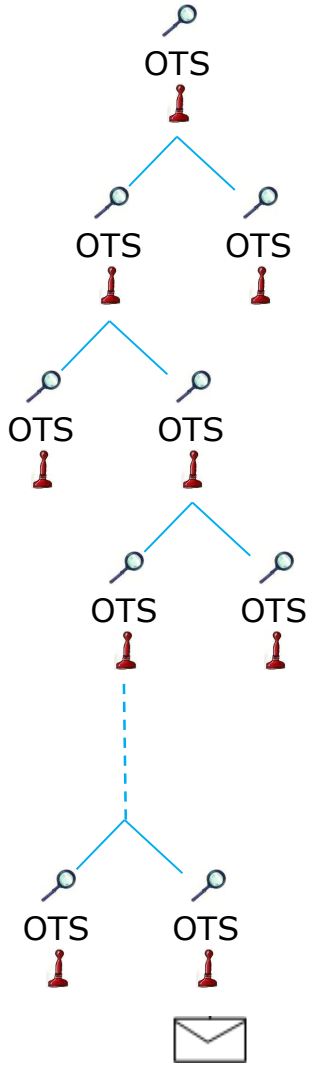
Stateless hash-based signatures [NY89,Gol87,Gol04]

Goldreich's approach [Gol04]:

Security parameter $\lambda = 128$

Use binary tree as in Merkle, but...

- **…for security**
  - pick index i at random;
  - requires huge tree to avoid index collisions (e.g., height $h = 2\lambda = 256$).
- **…for efficiency:**
  - use binary certification tree of OTS;
  - all OTS secret keys are generated pseudorandomly.



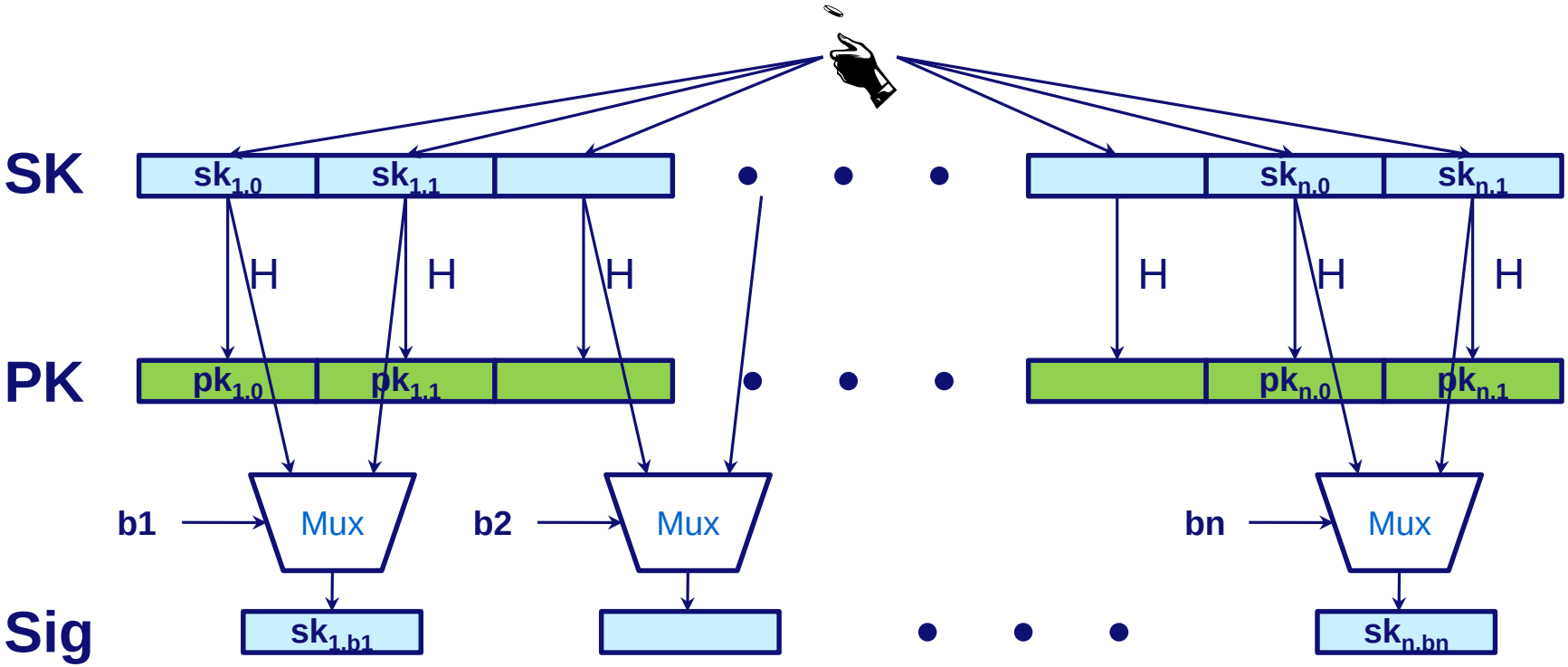
It works, but signature are p a i n f u l l y l o n g

- 0.6 MB for Goldreich signature using short-public-key Winternitz-16 one-time signatures.
- Would dominate traffic in typical applications, and add user-visible latency on typical network connections.
- Example:
  - Debian operating system is designed for frequent upgrades.
  - At least one new signature for each upgrade.
  - Typical upgrade: one package or just a few packages.
  - 1.2 MB average package size.
  - 0.08 MB median package size.
- Example:
  - HTTPS typically sends multiple signatures per page.
  - 1.8 MB average web page in Alexa Top 1000000.

Few-Time Signature Schemes

Recap LD-OTS

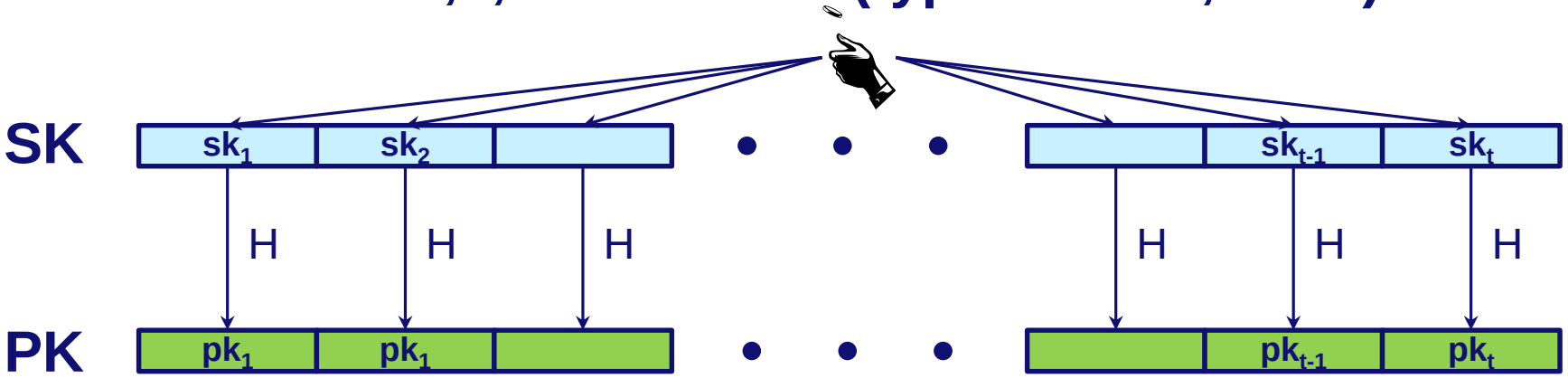
Message M = b1,…,bn, OWF H * = n bit



HORS [RR02]

Message M, OWF H, CRHF H' [*] **= n bit**

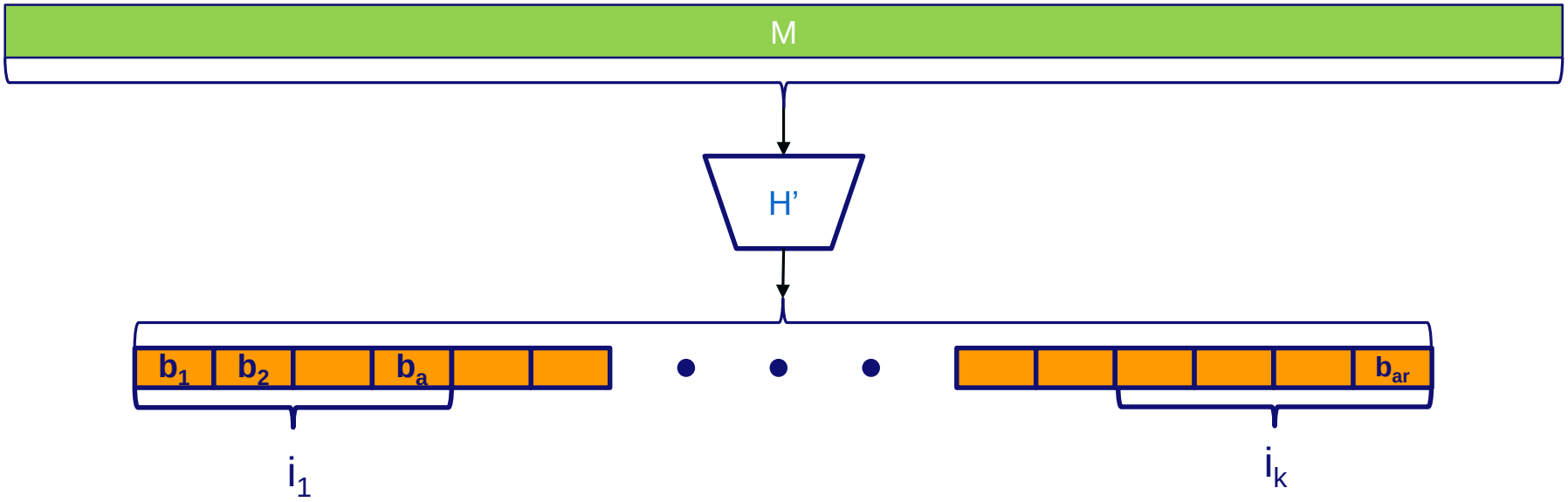
Parameters $t=2^a$,k, with m = ka (typical a=16, k=32)



HORS mapping function

Message M, OWF H, CRHF H’ $\boxed{*} = n$ bit

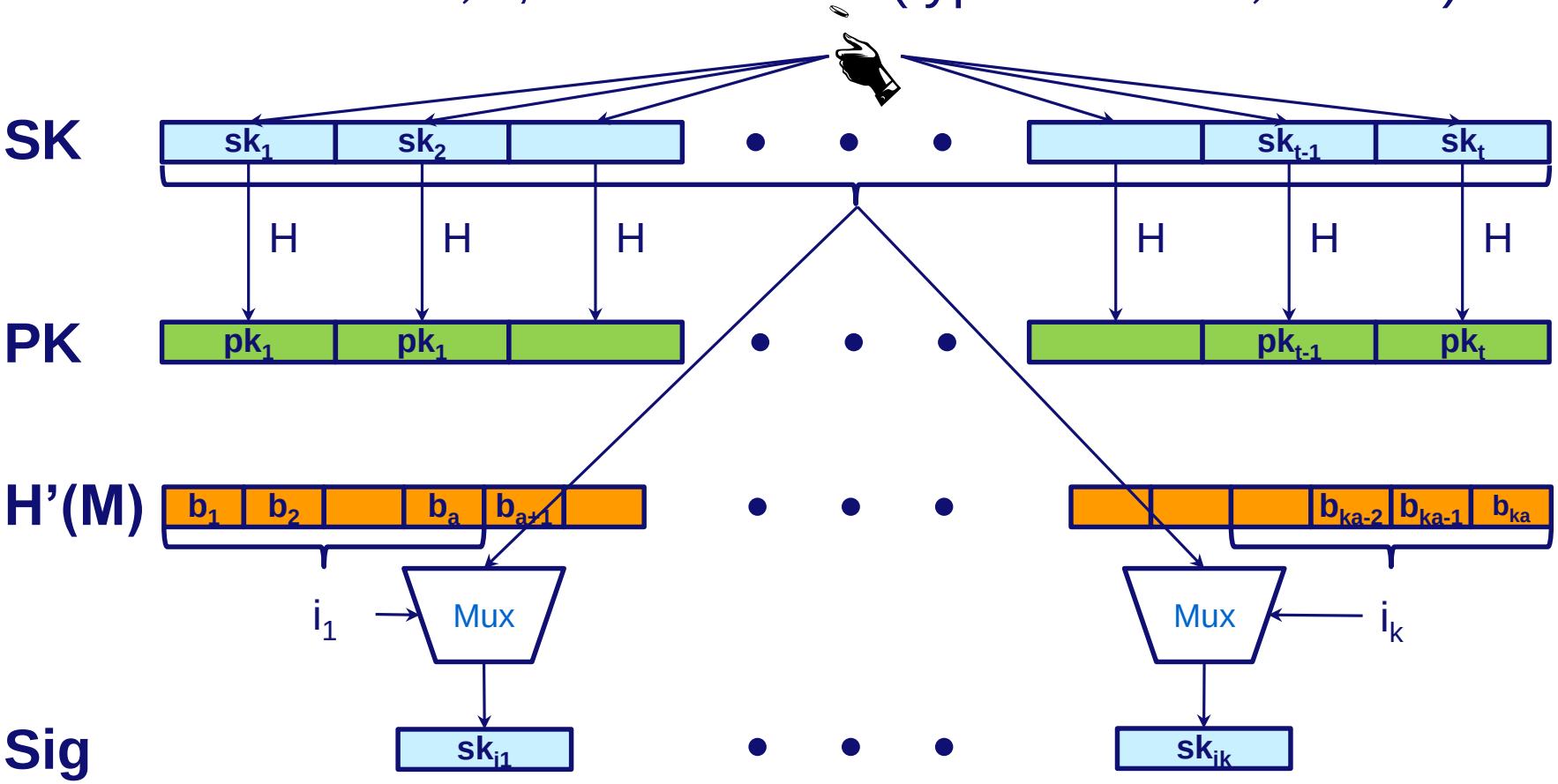
Parameters $t = 2^a, k$, with $m = ka$ (typical $a = 16, k = 32$)



HORS

Message M, OWF H, CRHF H’ [*] = n bit

Parameters $t = 2^a, k$, with $m = ka$ (typical $a = 16, k = 32$)



HORS Security

- M mapped to k element index set $M^i \in \{1, \ldots, t\}^k$
- Each signature publishes k out of t secrets
- Either break one-wayness or…

- **r-Subset-Resilience:** After seeing index sets M_j^i for r messages $msg_j, 1 \leq j \leq r$, hard to find $msg_{r+1} \neq msg_j$ such that $M_{r+1}^i \in \cup_{1 \leq j \leq r} M_j^i$.

- **Best generic attack:** $\text{Succ}_{\text{r-SSR}}(A,q) = q(rk / t)^k$

→ Security shrinks with each signature!

HORST

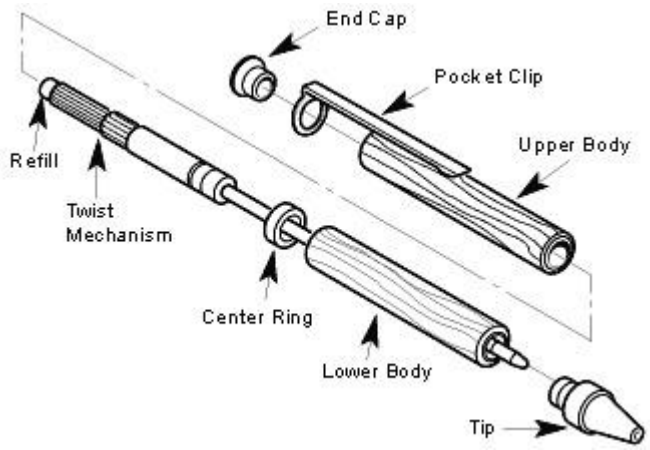
Using HORS with MSS requires adding PK (tn bits) to MSS signature. (SPHINCS-256: $n = 256, t = 2^{16}$, $k = 32$)

HORST: Merkle Tree on top of HORS-PK

- New PK = Root
- Publish Auth-Paths for HORS signature values
- PK can be computed from Sig

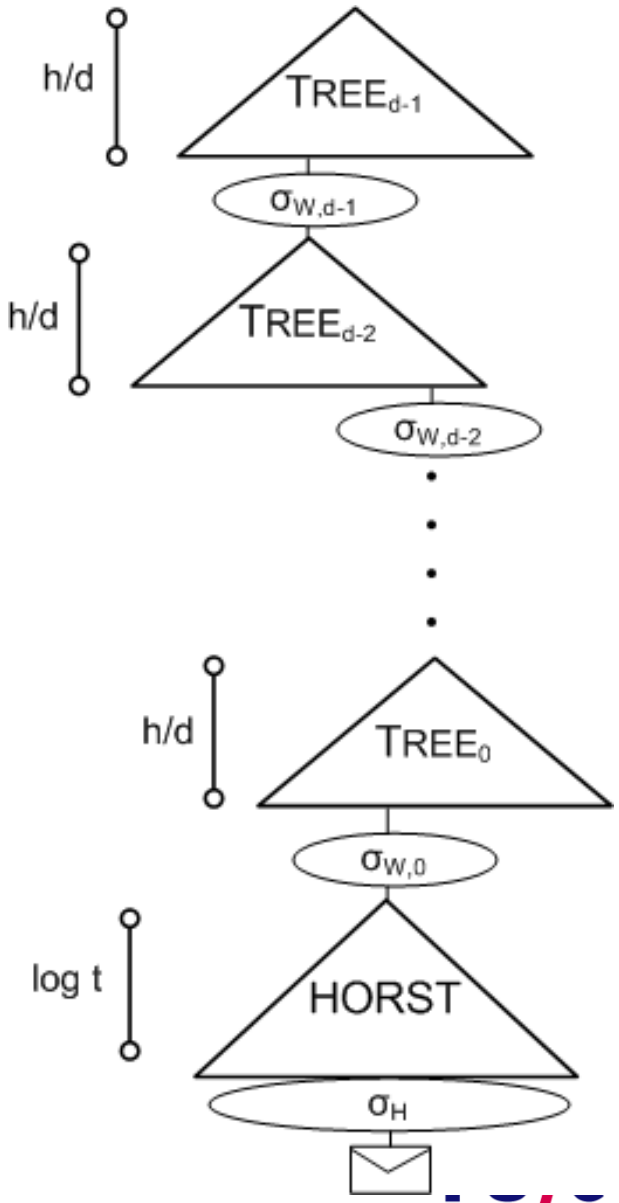
- With optimizations: $tn \rightarrow (k(log\, t - x + 1) + 2^x)n$
  - E.g. SPHINCS-256: 2 MB → 16 KB
- Use randomized message hash

Assembling SPHINCS



The SPHINCS Approach

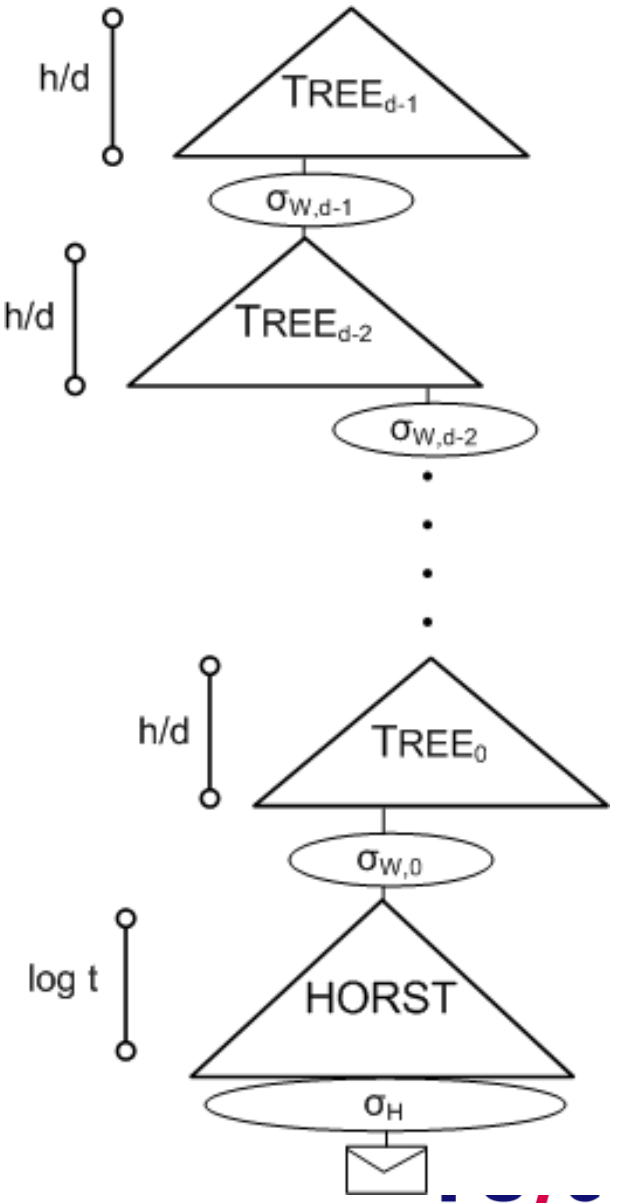
- Use a “hyper-tree” of total height h
- Parameter $d \geq 1$, such that $d \mid h$
- Each (Merkle) tree has height h/d
- (h/d)-ary certification tree



The SPHINCS Approach

- Pick index (pseudo-)randomly
- Messages signed with few-time signature scheme
- Significantly reduce total tree height
- Require

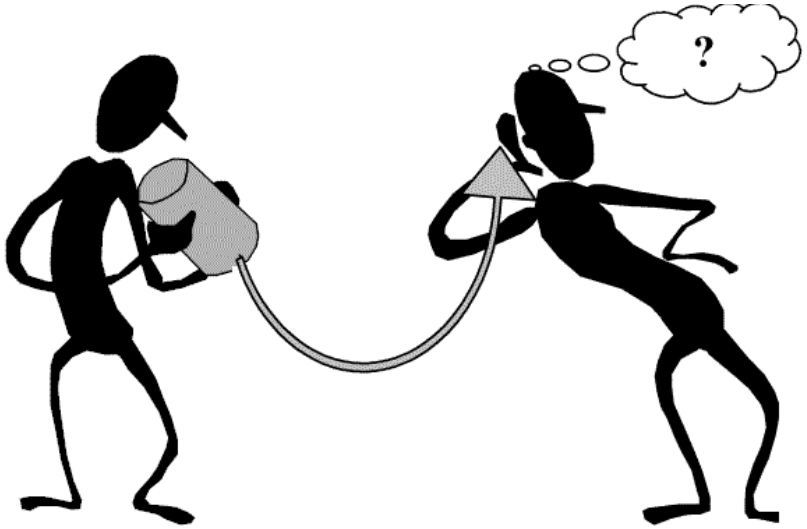
$$\sum_{r\in[0,\infty]}(\Pr[r-times\ index\ collision]\ *\ Succ_{r-SSR}(A))\ =\ negl(n)$$



And now more from Peter...

Thank you!

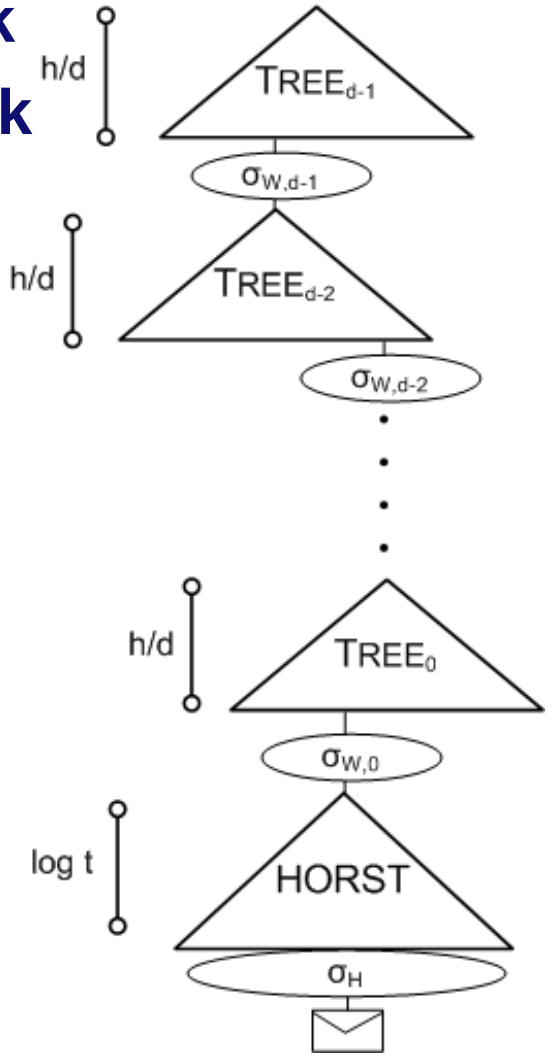
Questions?



For references & further literature see
https://huelsing.wordpress.com/hash-based-signature-schemes/literature/

SPHINCS Sign

1. Select (pseudo-)random HORST sk
2. Sign message using this HORST sk
3. Build parent tree
4. Use tree to sign HORST pk
5. If tree != top, goto 3.
6. Output Sig:
   1. Index
   2. HORST signature
   3. XMSS signature chain



Long-Standing Problem: Statefulness

- No problem in many cases.
  - Qualified signatures,
  - Keys on smartcard, ...
- Necessary for forward-security!

But:

- Key back-ups undermine security
- Parallel use of key problematic
  - Multi-threading,
  - Load balancing...
- Do not fit standard API