Hash-based Signatures and SPHINCS

Andreas Hülsing

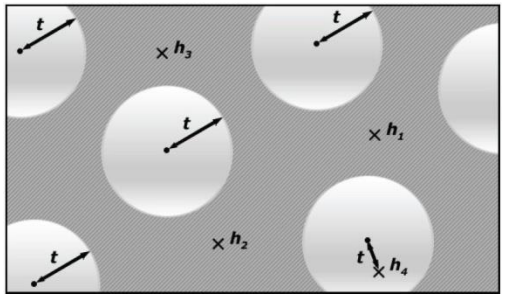
Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Post-Quantum Signatures

Lattice, MQ, Coding

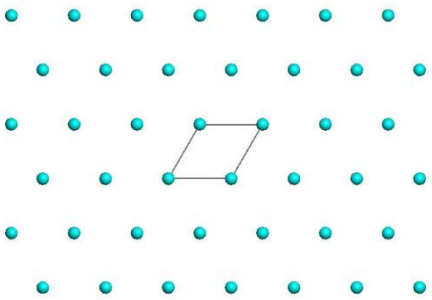
- **Signature and/or key sizes**
- **Runtimes**
- **Secure parameters**



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

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

$$y_3 = \ldots$$



TU/e Technische Universiteit Eindhoven University of Technology

Hash-based Signature Schemes [Mer89]

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

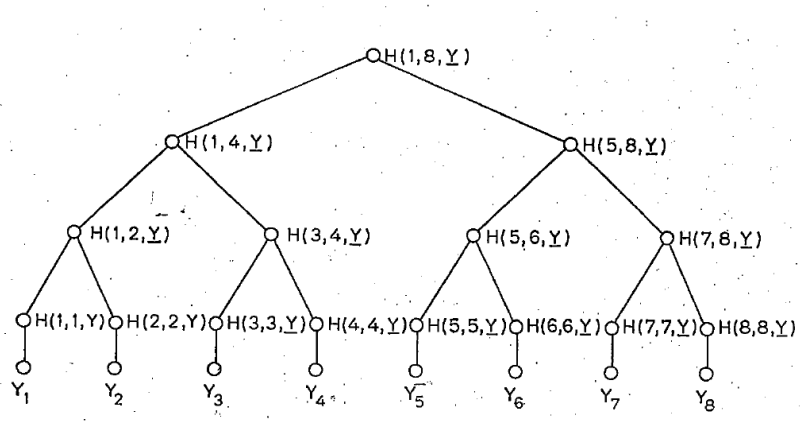
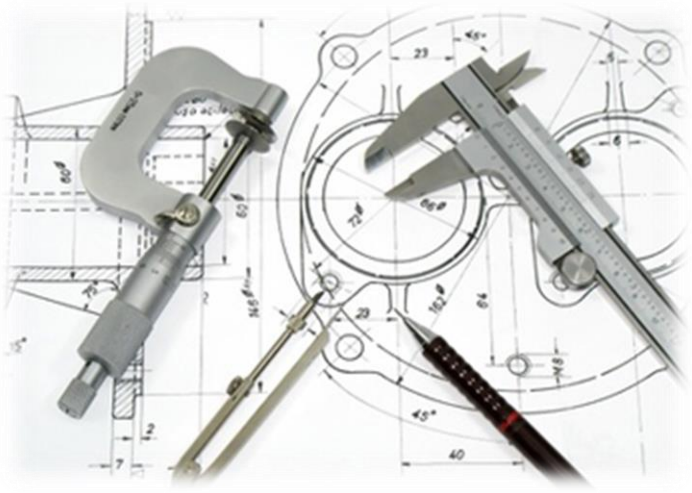


FIG 1
AN AUTHENTICATION TREE WITH N = 8.

PAGE 41B

TU/e Technische Universiteit Eindhoven University of Technology

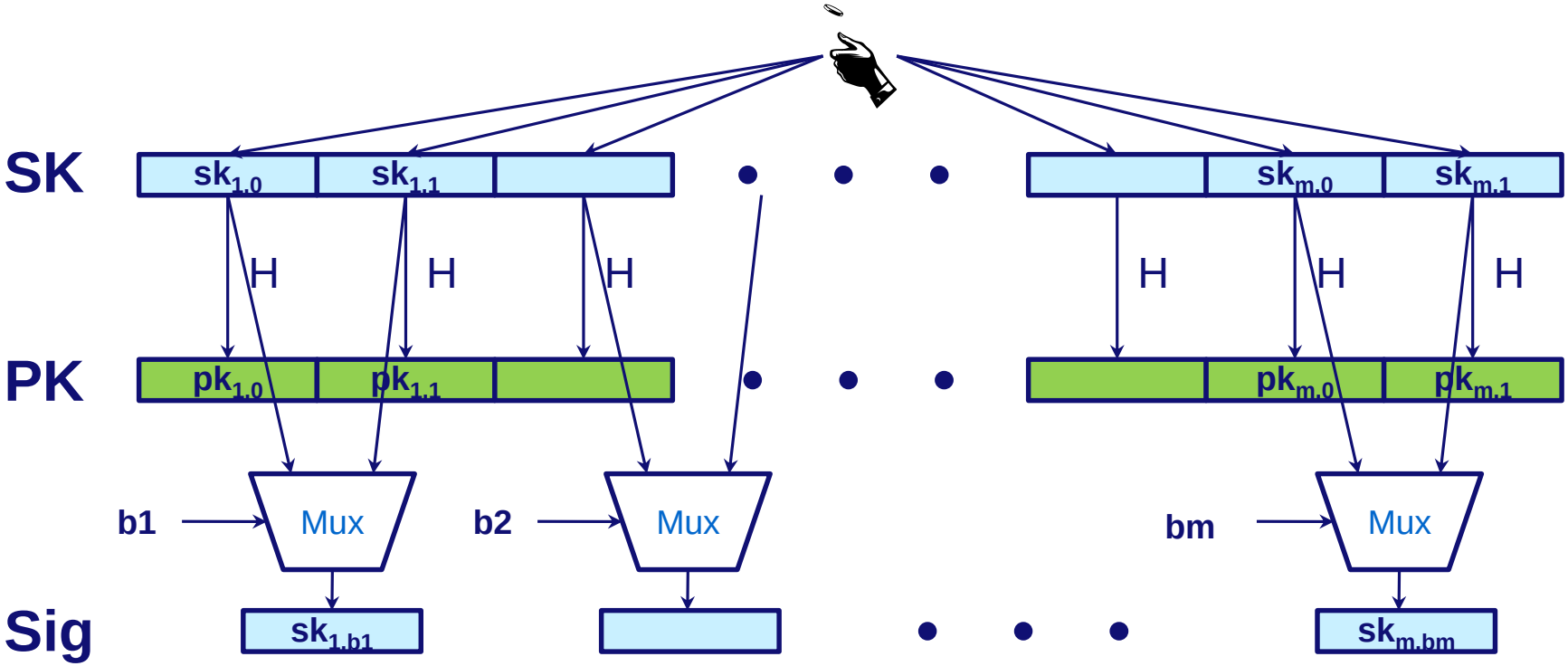
Basic Construction



TU/e Technische Universiteit Eindhoven University of Technology

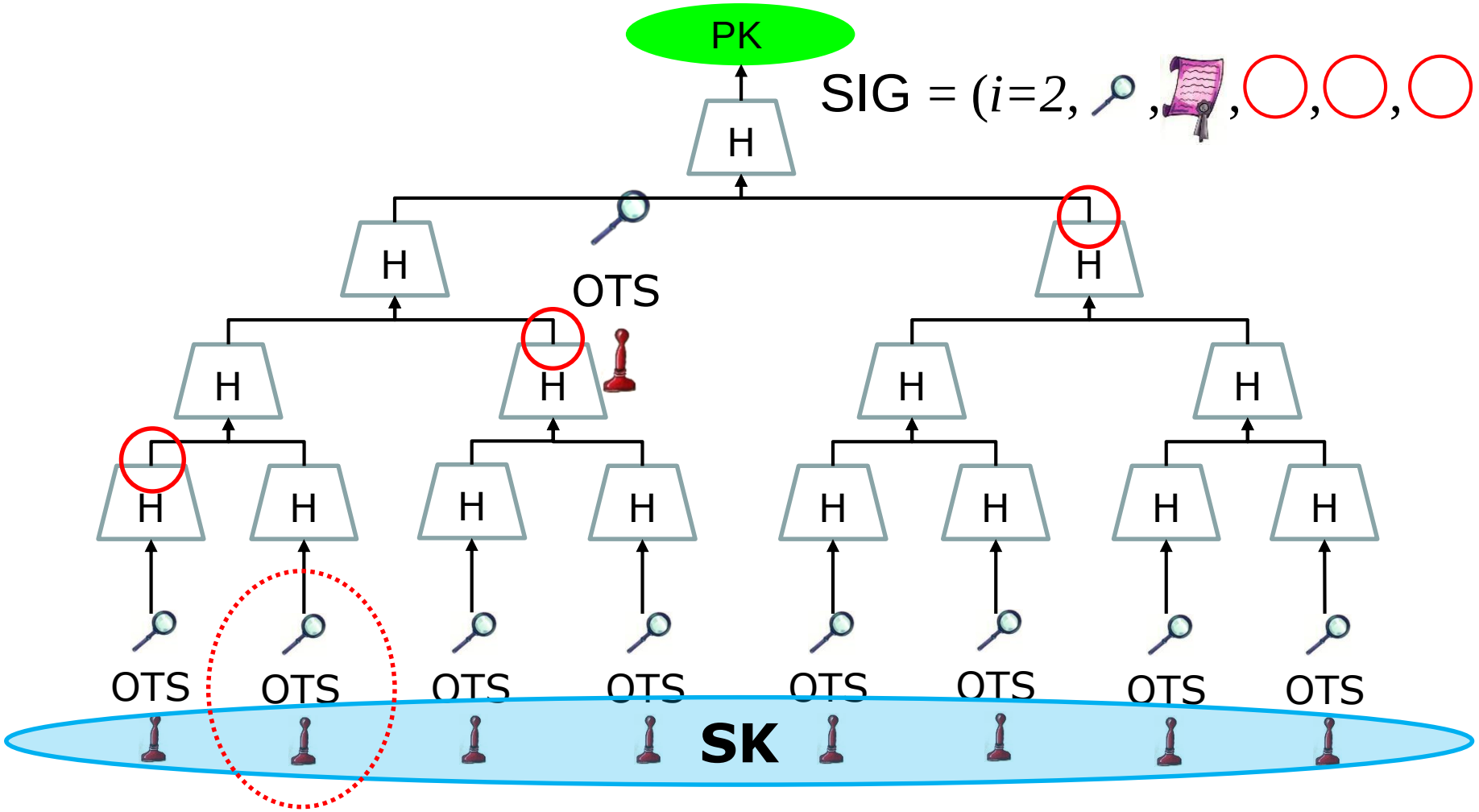
Lamport-Diffie OTS [Lam79]

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



TU/e Technische Universiteit Eindhoven University of Technology

Merkle's Hash-based Signatures



TU/e Technische Universiteit Eindhoven University of Technology

XMSS:

A practical signature scheme with minimal security assumptions

Johannes Buchmann, Carlos Coronado, Erik Dahmen, Andreas Hülsing

TU/e Technische Universiteit Eindhoven University of Technology

XMSS Security

Security parameter n

Requires family of functions $\mathcal{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$

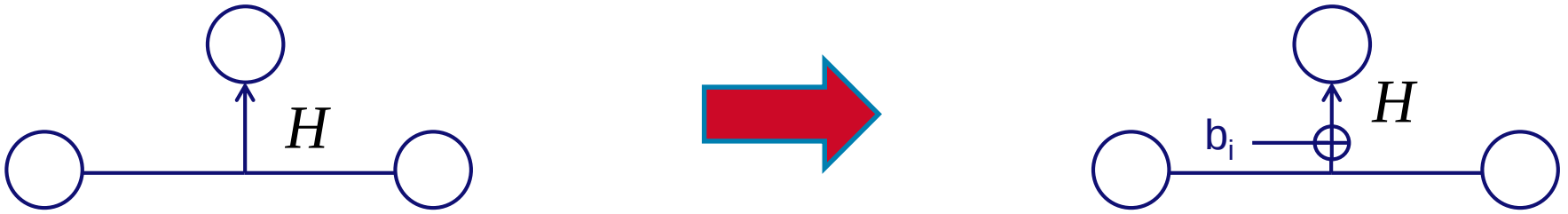
Requires family of functions $\mathcal{G} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$

Requires family of functions $\mathcal{H} : \{0,1\}^{2n} \times \{0,1\}^n \to \{0,1\}^n$

Theorem:

XMSS is existentially unforgeable under adaptive chosen message attacks if $\mathcal{F}$ is a 2^{nd}-preimage-resistant family of undetectable one-way functions, $\mathcal{G}$ is a pseudorandom function family, and $\mathcal{H}$ is a 2^{nd}-preimage-resistant function family.

XMSS Tree

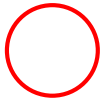


- Hashing one-time PK‘s using tree
- Requirements: CRHF -> SPRHF
- PK includes ~h additional values

XMSS uses Winternitz OTS

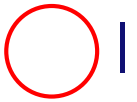
Security level b

SIG = (i, )

|  | = |  | = m * |  | = m*b

1.  = f ()

2. Trade-off between runtime and signature size

|  | ~ m/log w * |  |

TU/e Technische Universiteit Eindhoven University of Technology

Winternitz OTS (WOTS)

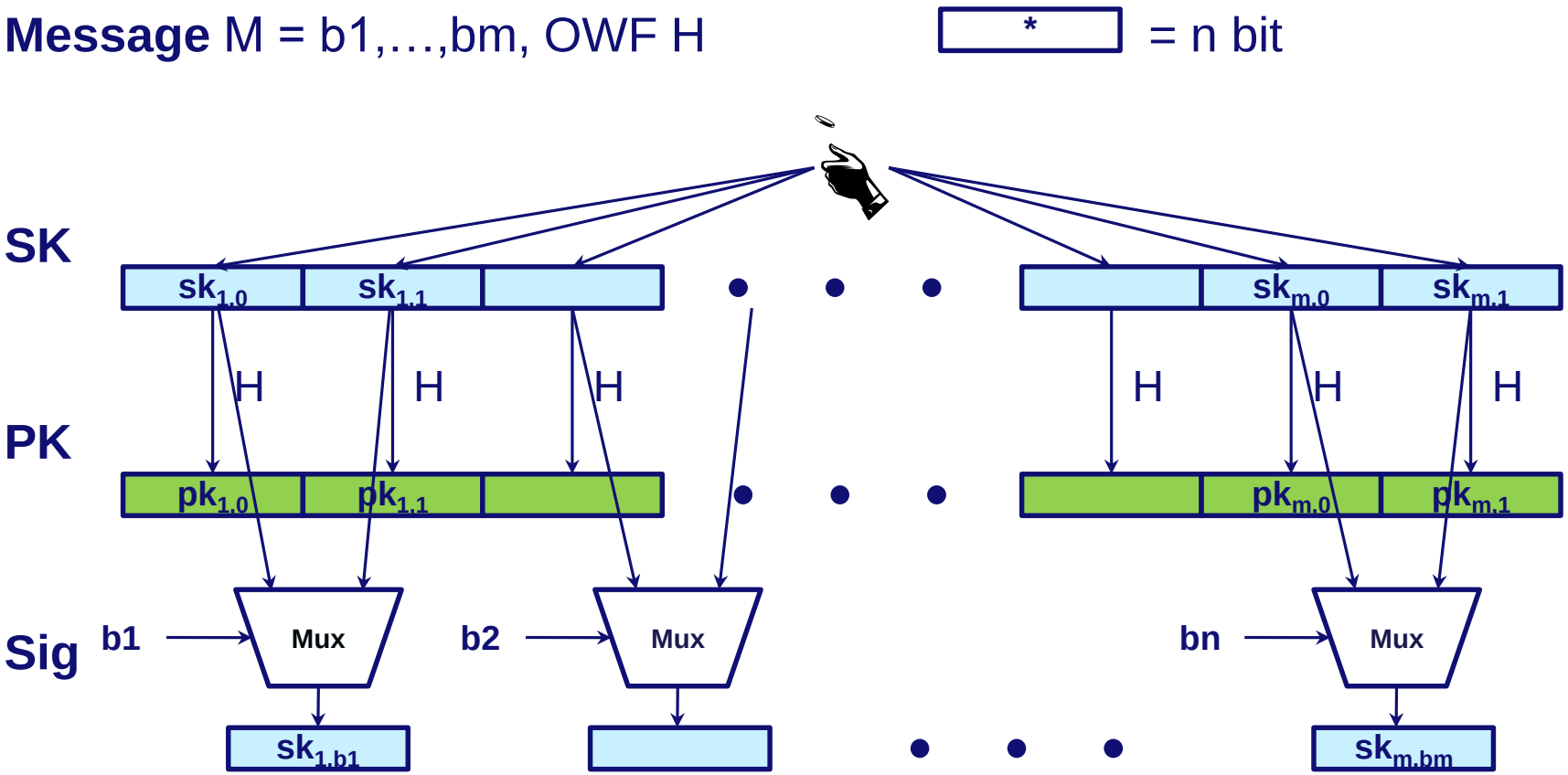
First idea: Winternitz (Mer89)
Full scheme: Even et al. (EGM96)
Security Proofs: Hevia & Micciancio (HM02)
Dods et al. (DSS05)
Requires collision-resistant undetectable one-way function family.

WOTS$^{\$}$: Buchmann et al. (BDEH^{+}11)
Requires pseudorandom function family.

WOTS^{+}: Hülsing (Hül13)
Requires second preimage resistant undetectable one-way function family.

Recap LD-OTS [Lam79]

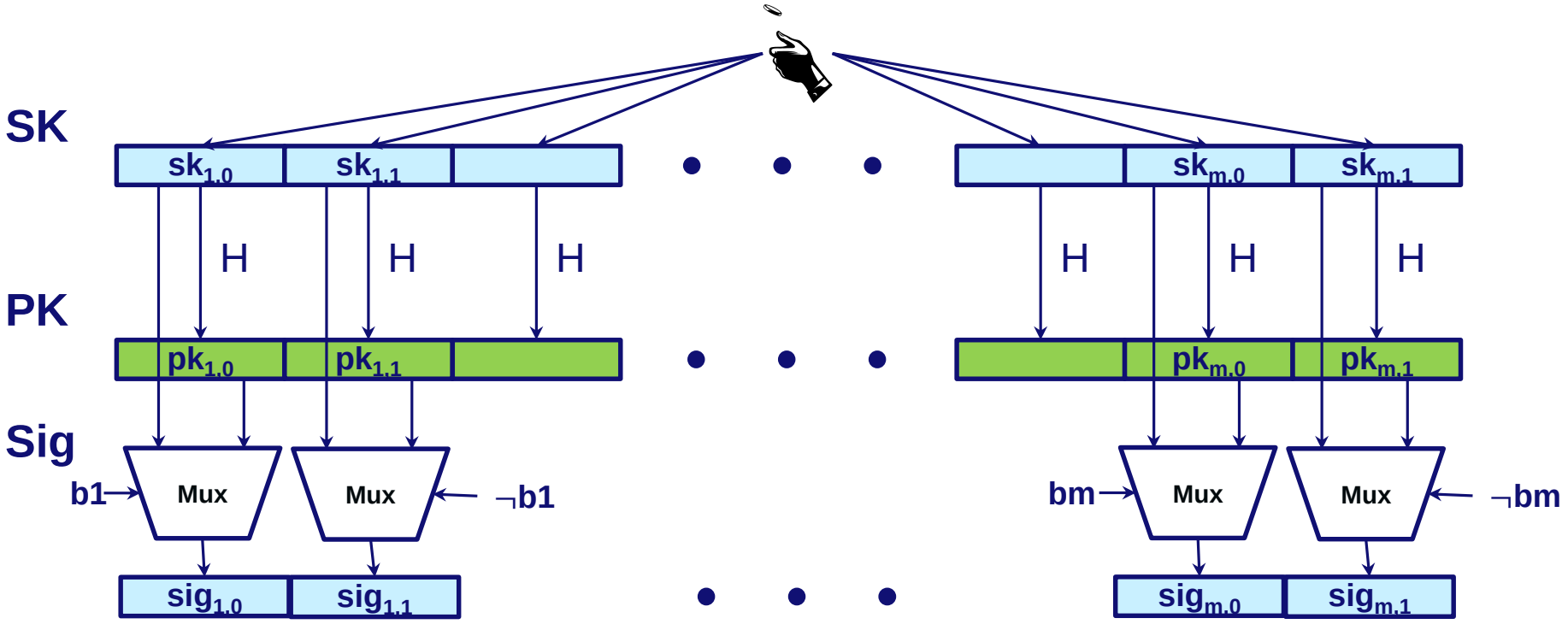
Message M = b1,…,bm, OWF H



Trivial Optimization

Message M = b1,…,bm, OWF H

* = n bit



TU/e Technische Universiteit Eindhoven University of Technology

Non-trivial Optimization

Message $M = b_1,\ldots,b_m$, OWF H

SK: $sk_1,\ldots,sk_m,sk_{m+1},\ldots,sk_{2m}$

PK: $H(sk_1),\ldots,H(sk_m),H(sk_{m+1}),\ldots,H(sk_{2m})$

Encode M: $M' = b_1,\ldots,b_m,\neg b_1,\ldots,\neg b_m$

Sig: $sig_i =$ $\begin{cases} sk_i & \text{, if } b_i = 1 \\ H(sk_i) & \text{, otherwise} \end{cases}$

Checksum with bad performance!

TU/e Technische Universiteit Eindhoven University of Technology

Non-trivial Optimization, cont‘d

Message M = $b_1,\ldots,b_m$, OWF H

SK: $sk_1,\ldots,sk_m,sk_{m+1},\ldots,sk_{m+\log m}$

PK: $H(sk_1),\ldots,H(sk_m),H(sk_{m+1}),\ldots,H(sk_{m+\log m})$

Encode M: M‘ = $b_1,\ldots,b_m,\neg \sum_1^m b_i$

Sig: sig_i **=**

$$\begin{cases} sk_i & \text{, if } b_i = 1 \\ H(sk_i) & \text{, otherwise} \end{cases}$$

IF one b_i is flipped from 1 to 0, another b_j will flip from 0 to 1

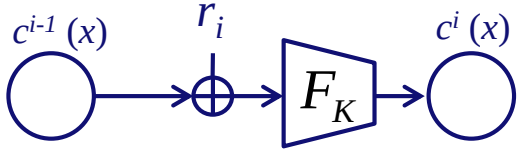
WOTS
Function Chain

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

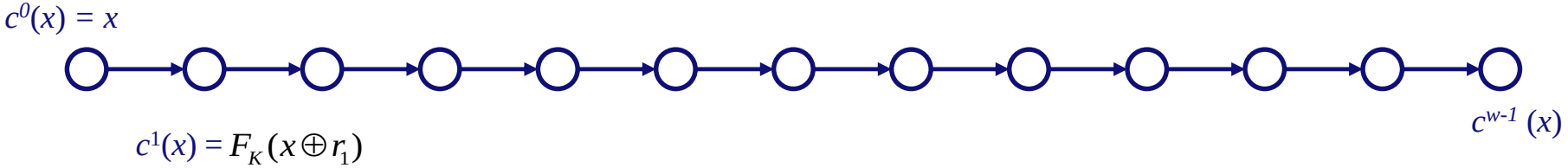
Formerly: $c^i(x) = F_K(c^{i-1}(x)) = \underbrace{F_K \circ F_K \circ \ldots \circ F_K}_{i-times}(x), K \in \{0,1\}^{n'}$

WOTS$^+$

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



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



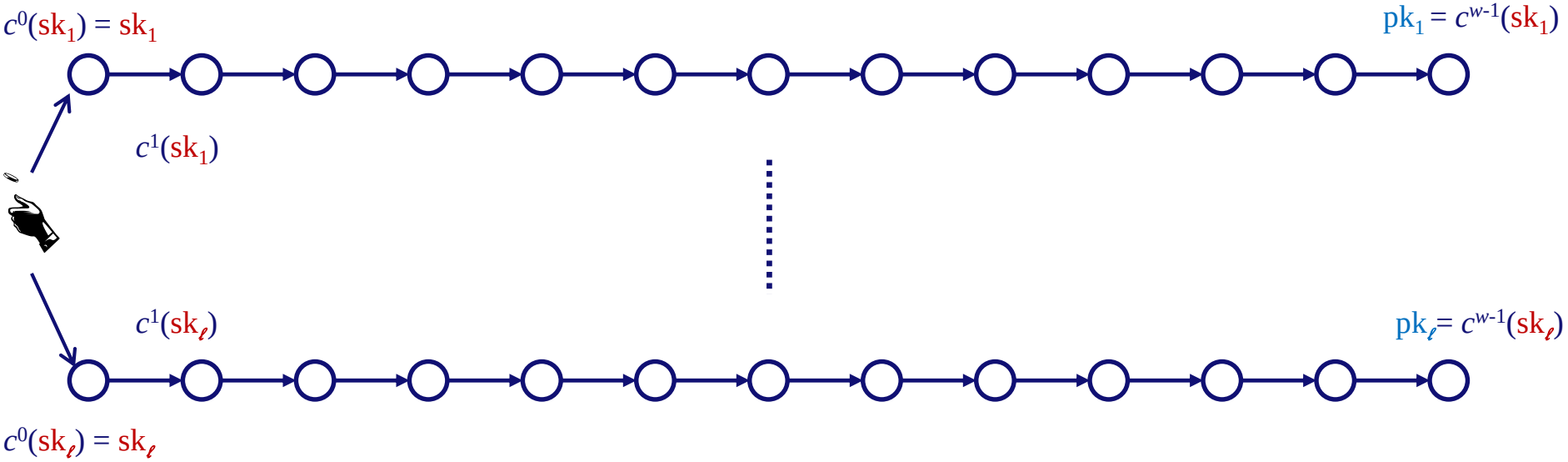
TU/e Technische Universiteit Eindhoven University of Technology

WOTS+

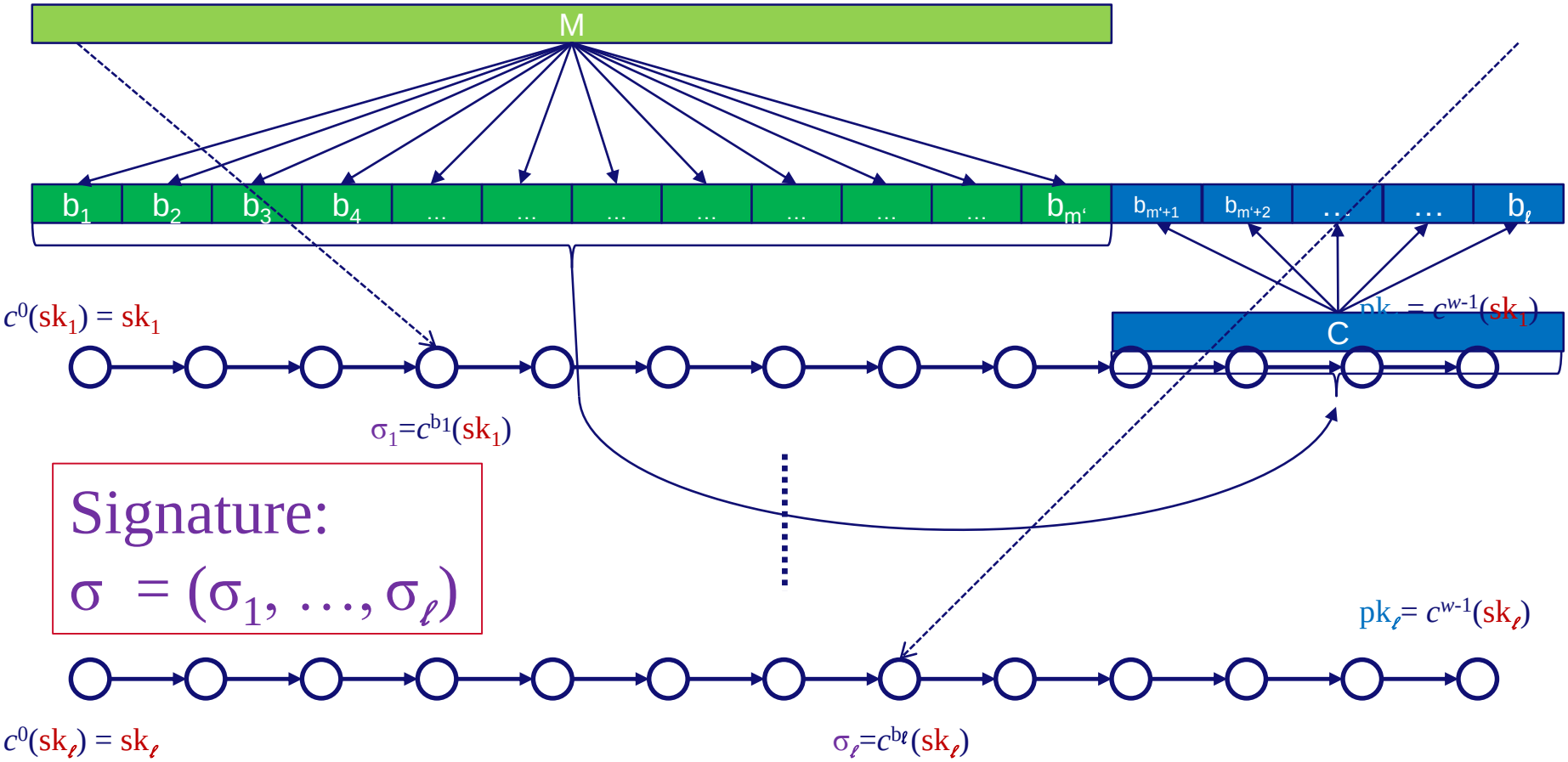
Winternitz parameter w, security parameter n, message length m, function family

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

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



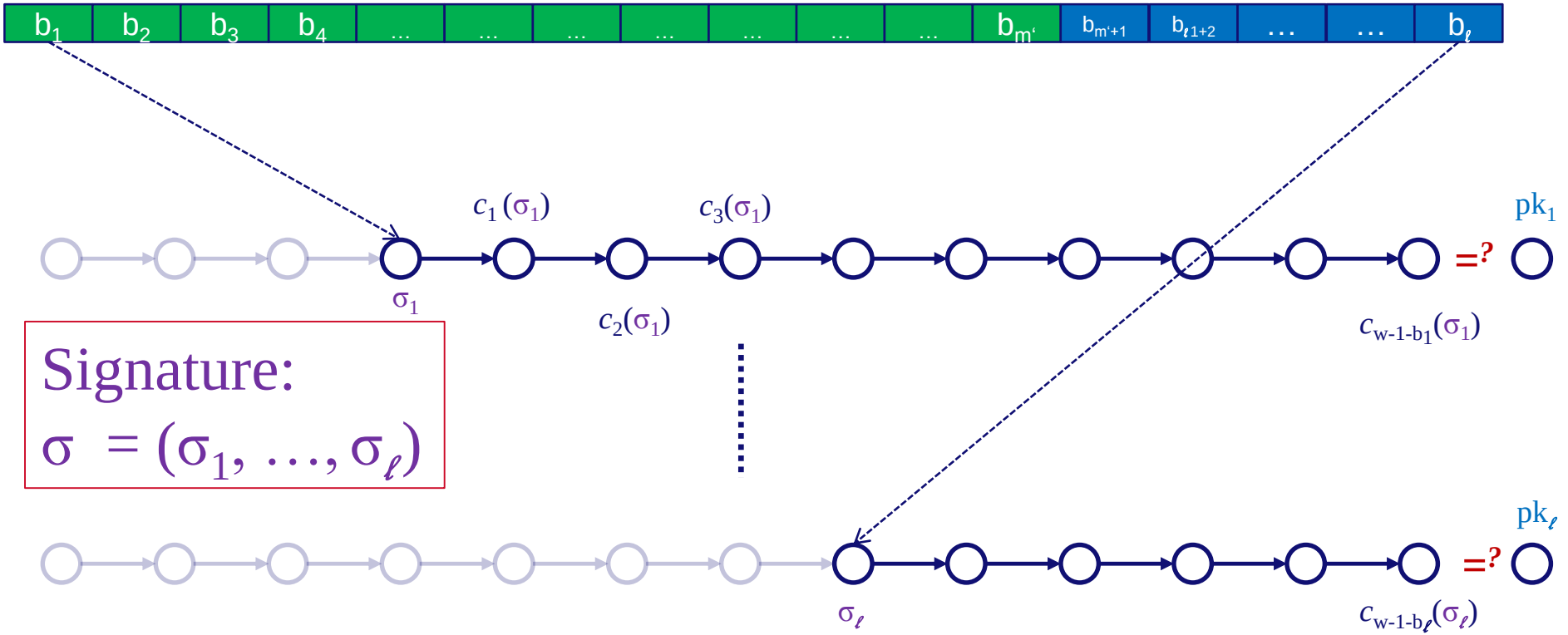
WOTS+ Signature generation



TU/e Technische Universiteit Eindhoven University of Technology

WOTS+ Signature Verification

Verifier knows: M, w



TU/e Technische Universiteit Eindhoven University of Technology

WOTS Function Chains

For $x \in \{0,1\}^n$ define $c_0(x) = x$ and

- **WOTS:** $c_i(x) = F_K(c_{i-1}(x))$
- **WOTS$^{\$}$:** $c_i(x) = F_{c_{i-1}(x)}(r)$
- **WOTS^{+}:** $c_i(x) = F_K(c_{i-1}(x) \oplus r_i)$

WOTS Security

Theorem (informally):

W-OTS is strongly unforgeable under chosen message attacks if $\mathcal{F}$ is a collision resistant family of undetectable one-way functions.

W-OTS$^{\$}$ is existentially unforgeable under chosen message attacks if $\mathcal{F}$ is a pseudorandom function family.

W-OTS^{+} is strongly unforgeable under chosen message attacks if $\mathcal{F}$ is a 2nd-preimage resistant family of undetectable one-way functions.

WOTS Sizes and Runtimes

	Lamport-Diffie	WOTS	WOTS$^{\$}$	WOTS^{+}
Public Key Size	2bm	ℓ 2b ~ 2bm/log w	ℓ b (+b) ~ bm/log w	ℓ b (+(w-1)b) ~ bm/log w
Secret Key Size	2bm	ℓ 2b ~ 2bm/log w	ℓ b ~ bm/log w	ℓ b ~ bm/log w
Signature Size	bm	ℓ 2b ~ 2bm/log w	ℓ b ~ bm/log w	ℓ b ~ bm/log w
Key Generation Time	~ 2m	ℓ w ~ wm/log		ℓ w ~ wm/log w

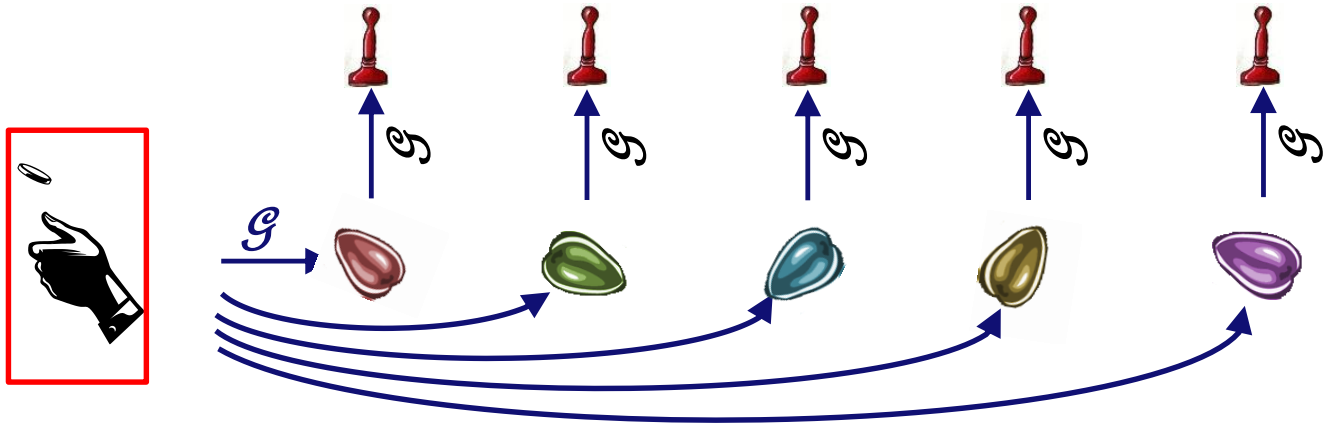
WOTS$^{\$}$:
Securitly loss linear in w
-> Only small w

Security level b, Winternitz parameter w, Message Length m,
$\ell = \ell(w,m) \sim m / \log w$

Secret Key Generation

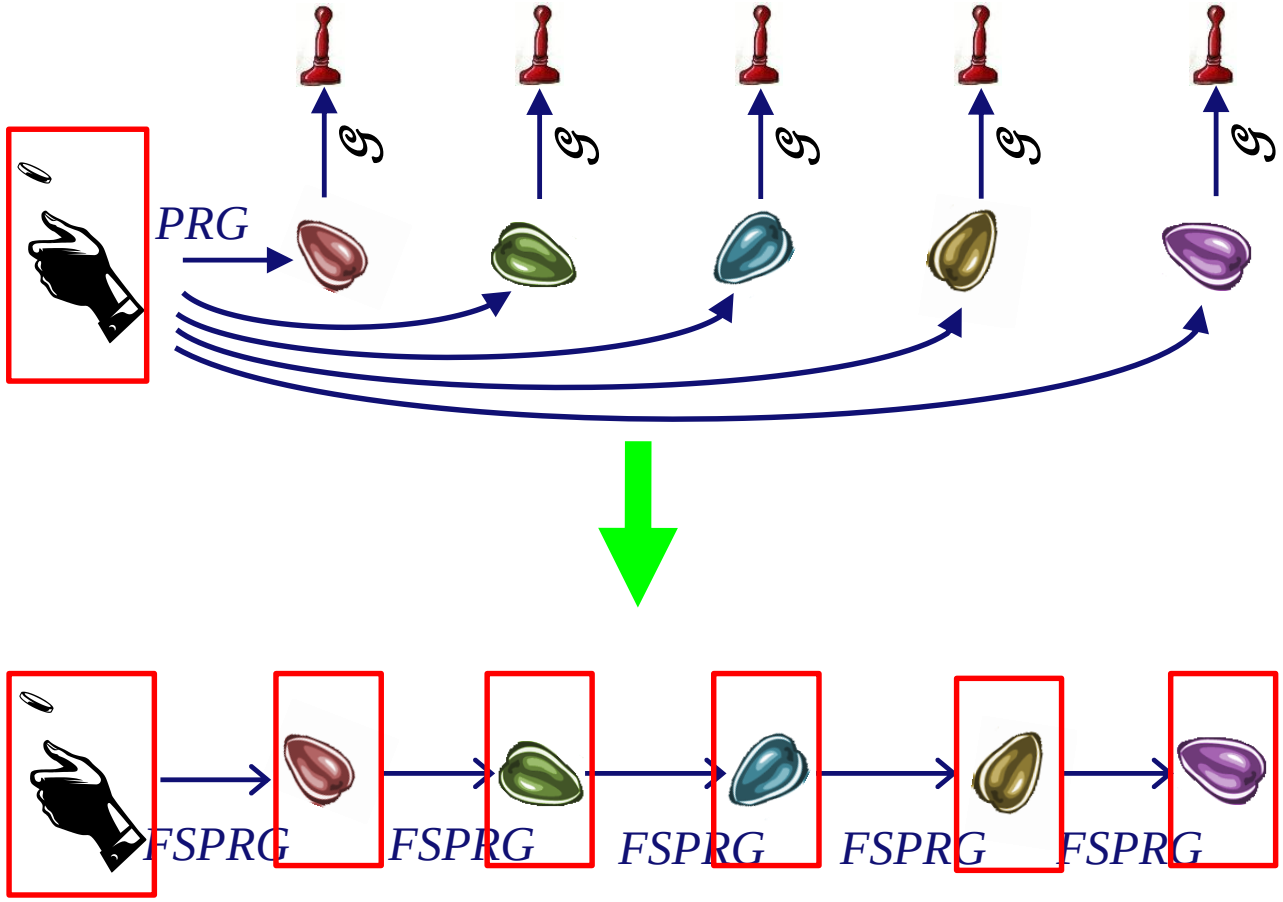
TU/e Technische Universiteit Eindhoven University of Technology

XMSS – Secret key

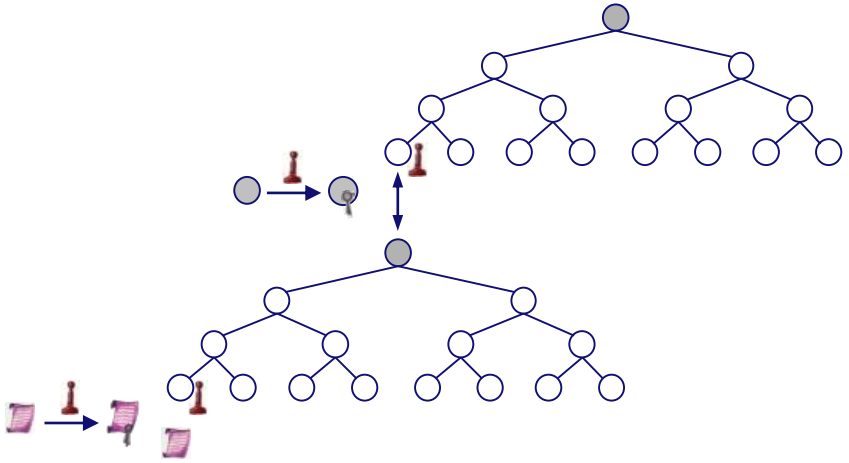


Secret Key Size: $2^{h}\ell\, b \rightarrow b$

XMSS forward secure

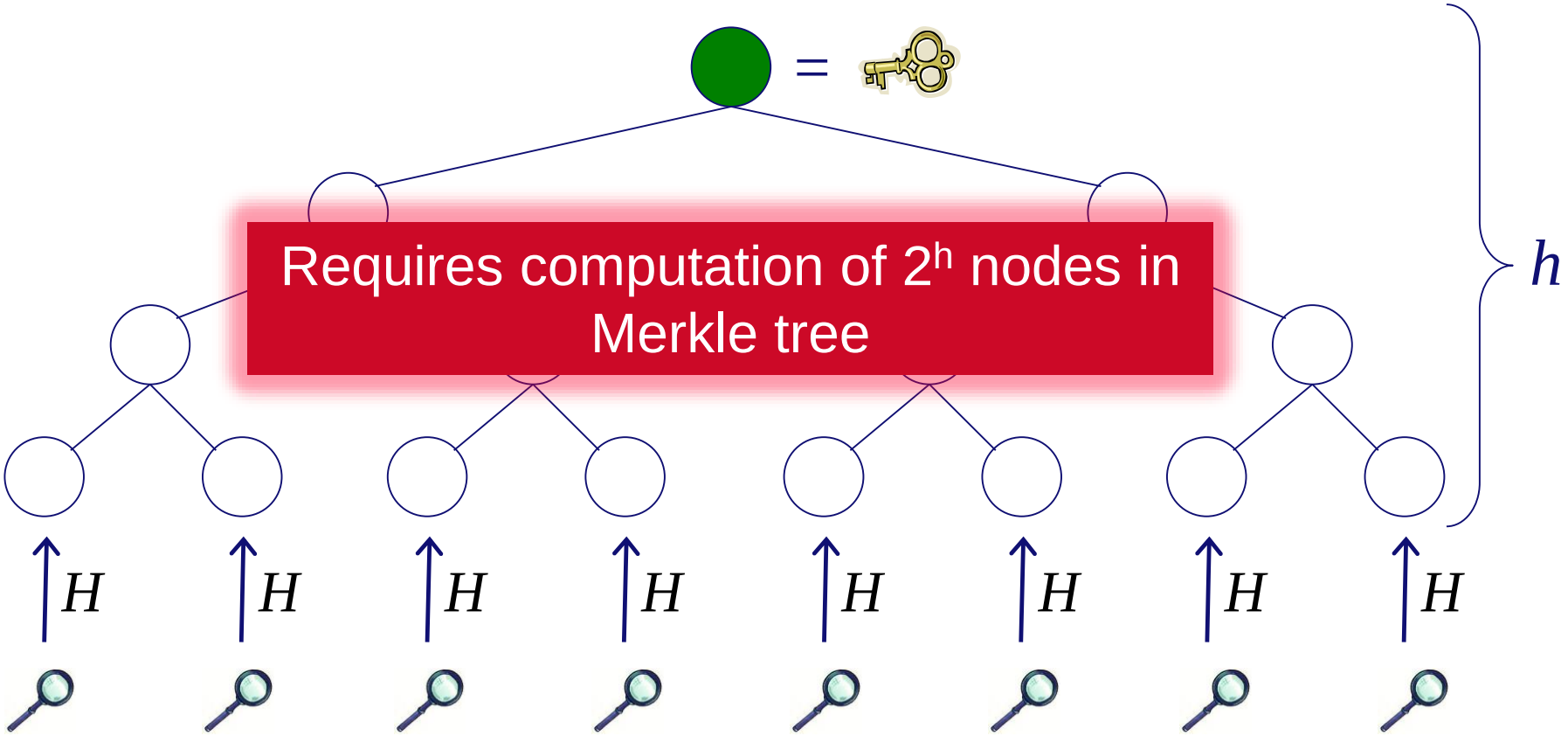


Tree Chaining



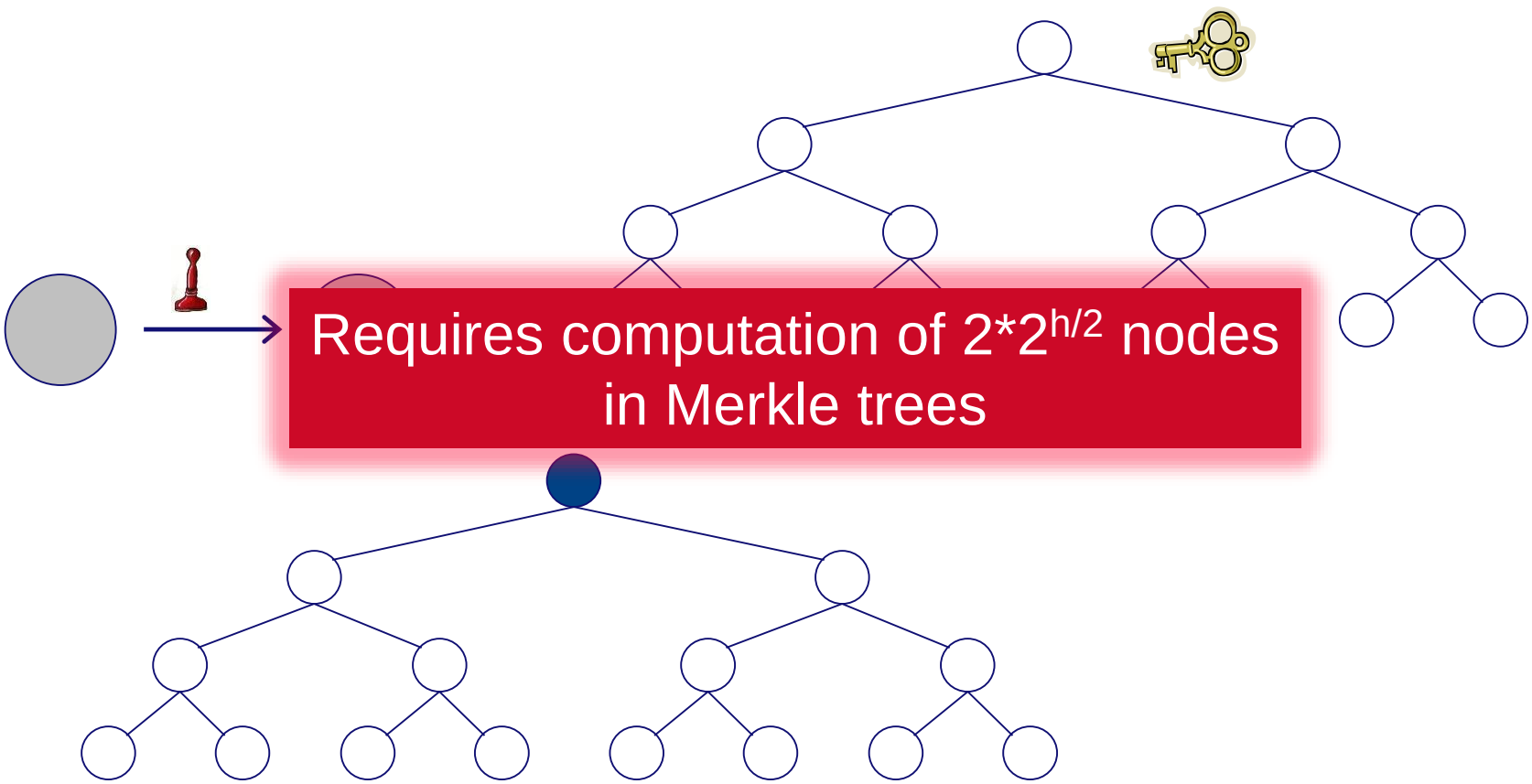
TU/e Technische Universiteit Eindhoven University of Technology

XMSS Public Key Generation

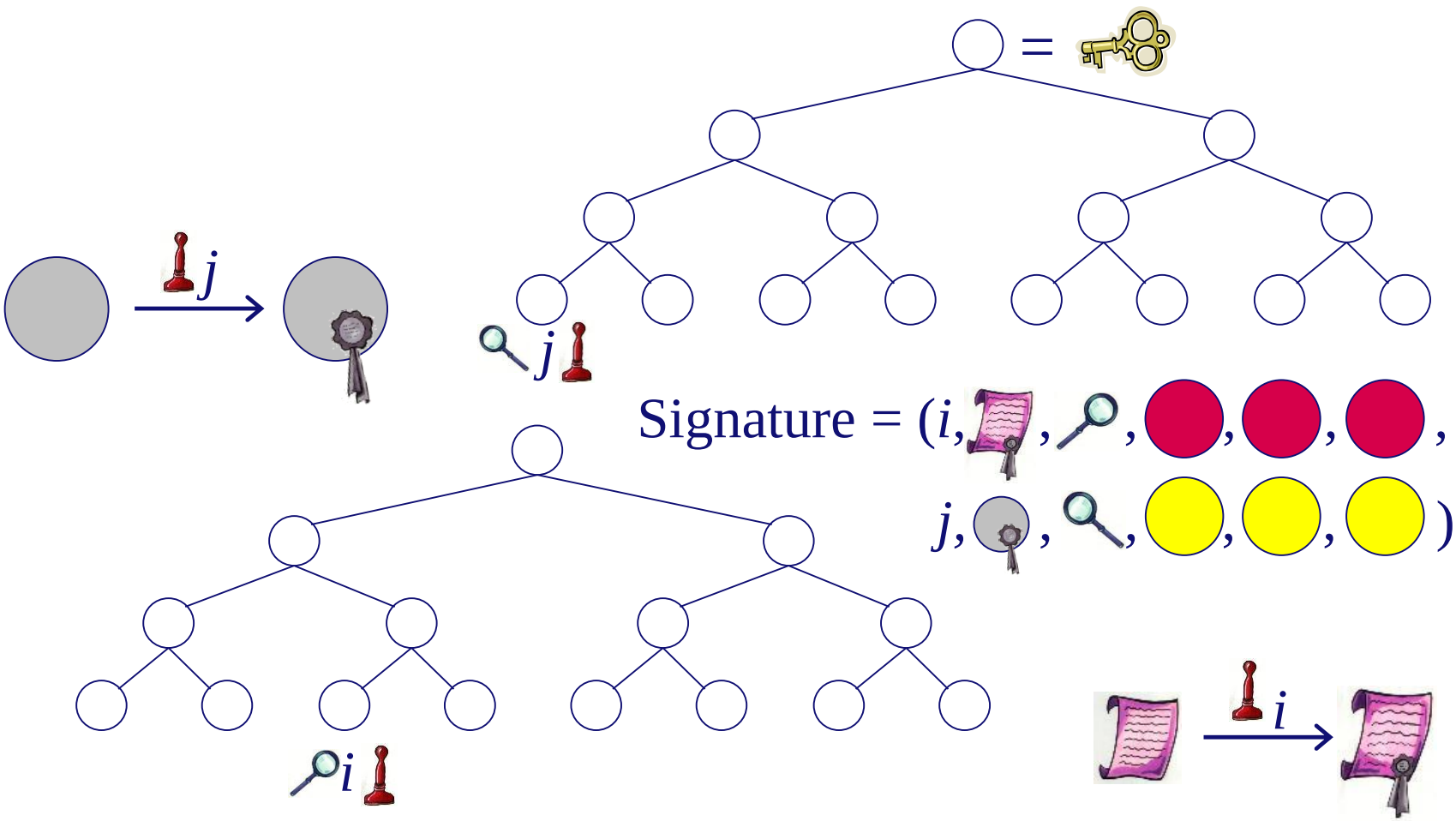


Requires computation of 2^h nodes in Merkle tree

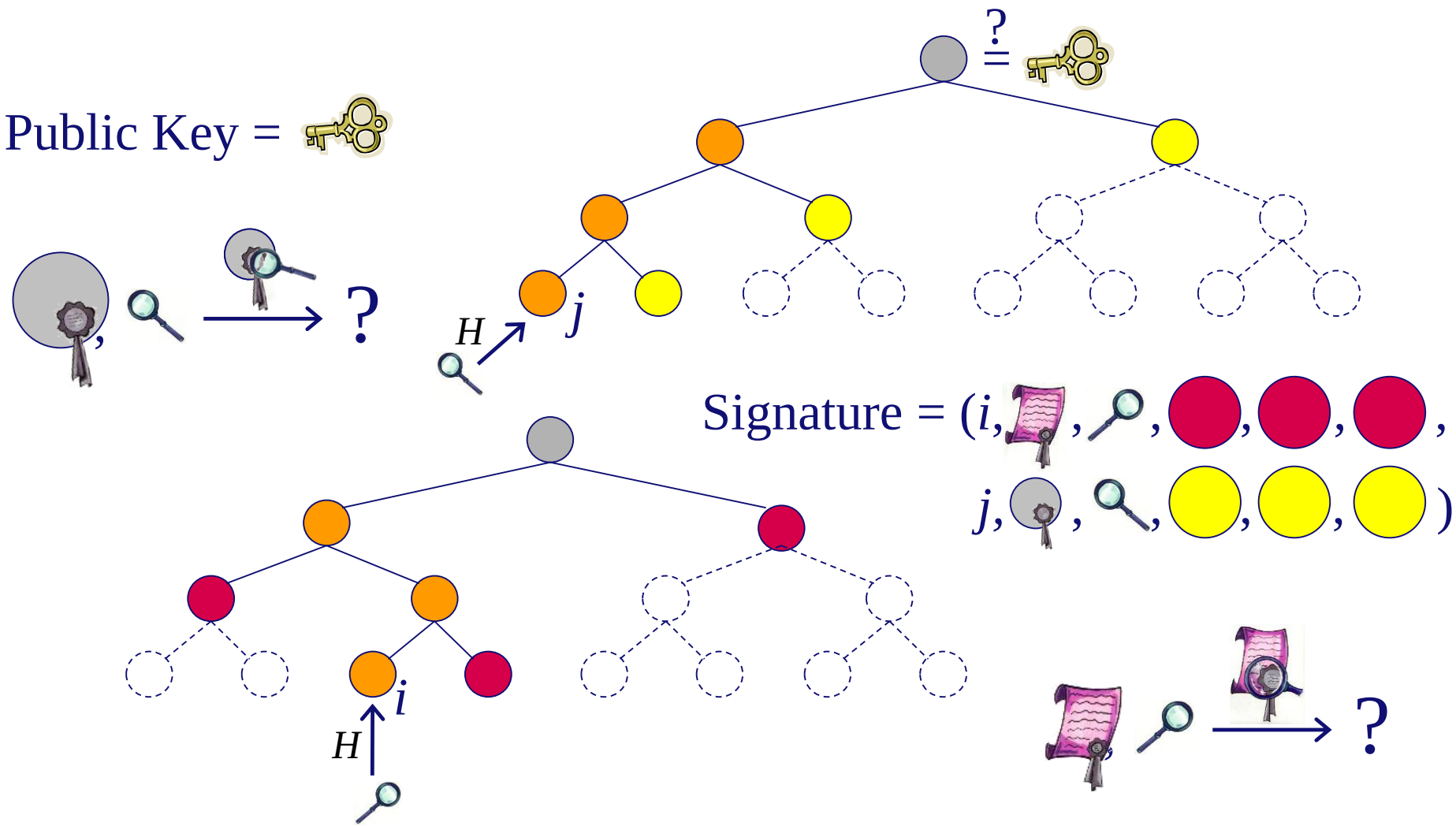
Two Layer
Key generation



Two Layer
Signing



Two Layer
Verifying



XMSS Public Key Generation

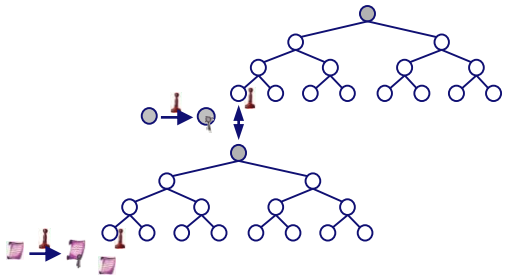
Security level b, tree height h

MSS

- **Generate tree of size 2^h**
- **Cost ~ 2^h**

XMSS

- **Tree chaining**
- **Use d layers of trees of height h/t**
- **Generate d trees of height $2^{h/d}$**
- **Cost ~ $d*2^{h/d}$**
- **Example: h = 40, d = 2, costs ~ $2*2^{20} = 2^{21}$**
- **Slightly increased signature size (+d-1 one-time sigs)**



XMSS Authentication Path Generation

Straight forward: 2^h-1 leaf + 2^h-h node computations

BDS Algorithm:

Runtime

$(h-k)/2+1$ leaf and

$3(h-k-1)/2+1$ node computations.

$+(h-k)$ calls to FSPRG for forward secure XMSS in the worst case.

Storage

$$3h+\left\lfloor \frac{h}{2} \right\rfloor -3k-2+2^k \; n \text{ bit nodes}$$

$+2h-2k$ n bit seeds for forward secure XMSS.

TU/e Technische Universiteit Eindhoven University of Technology

XMSS Implementations
C Implementation

C Implementation, using OpenSSL [BDH2011]

	Sign (ms)	Verify (ms)	Signature (bit)	Public Key (bit)	Secret Key (byte)	Bit Security	Comment
XMSS-SHA-2	35.60	1.98	16,672	13,600	3,364	157	h = 20, w = 64,
XMSS-AES-NI	0.52	0.07	19,616	7,328	1,684	84	h = 20, w = 4
XMSS-AES	1.06	0.11	19,616	7,328	1,684	84	h = 20, w = 4
RSA 2048	3.08	0.09	≤ 2,048	≤ 4,096	≤ 512	87	

Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz with Intel AES-NI

XMSS Implementations
Smartcard Implementation

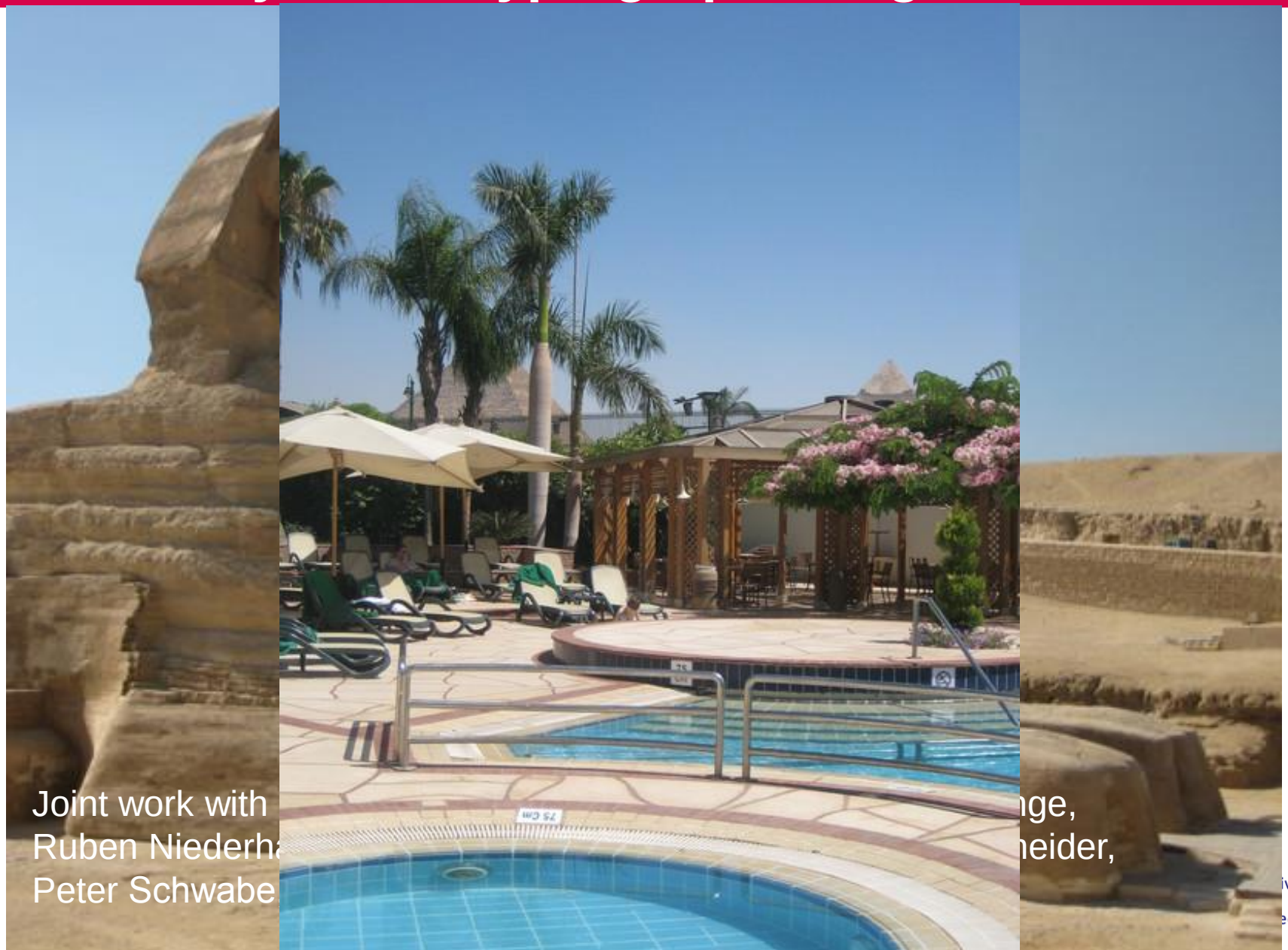
	Sign (ms)	Verify (ms)	Keygen (ms)	Signature (byte)	Public Key (byte)	Secret Key (byte)	Bit Sec.	Comment
XMSS	134	23	925,400	2,388	800	2,448	92	H = 16, w = 4
XMSS$^+$	106	25	5,600	3,476	544	3,760	94	H = 16, w = 4
RSA 2048	190	7	11,000	≤ 256	≤ 512	≤ 512	87	

Infineon SLE78 16Bit-CPU@33MHz, 8KB RAM, TRNG, sym. & asym. co-processor

NVM: Card 16.5 million write cycles/ sector,
XMSS$^+$ < 5 million write cycles (h=20)

[HBB12]

SPHINCS: Stateless Practical Hash-based Incredibly Nice Cryptographic Signatures



Joint work with
Ruben Niederha
Peter Schwabe

nge,
heider,

iversiteit
echnology

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

SPHINCS Properties

Stateless

128bit Quantum Security

Practical Speed

Practical Signature Size

TU/e Technische Universiteit Eindhoven University of Technology

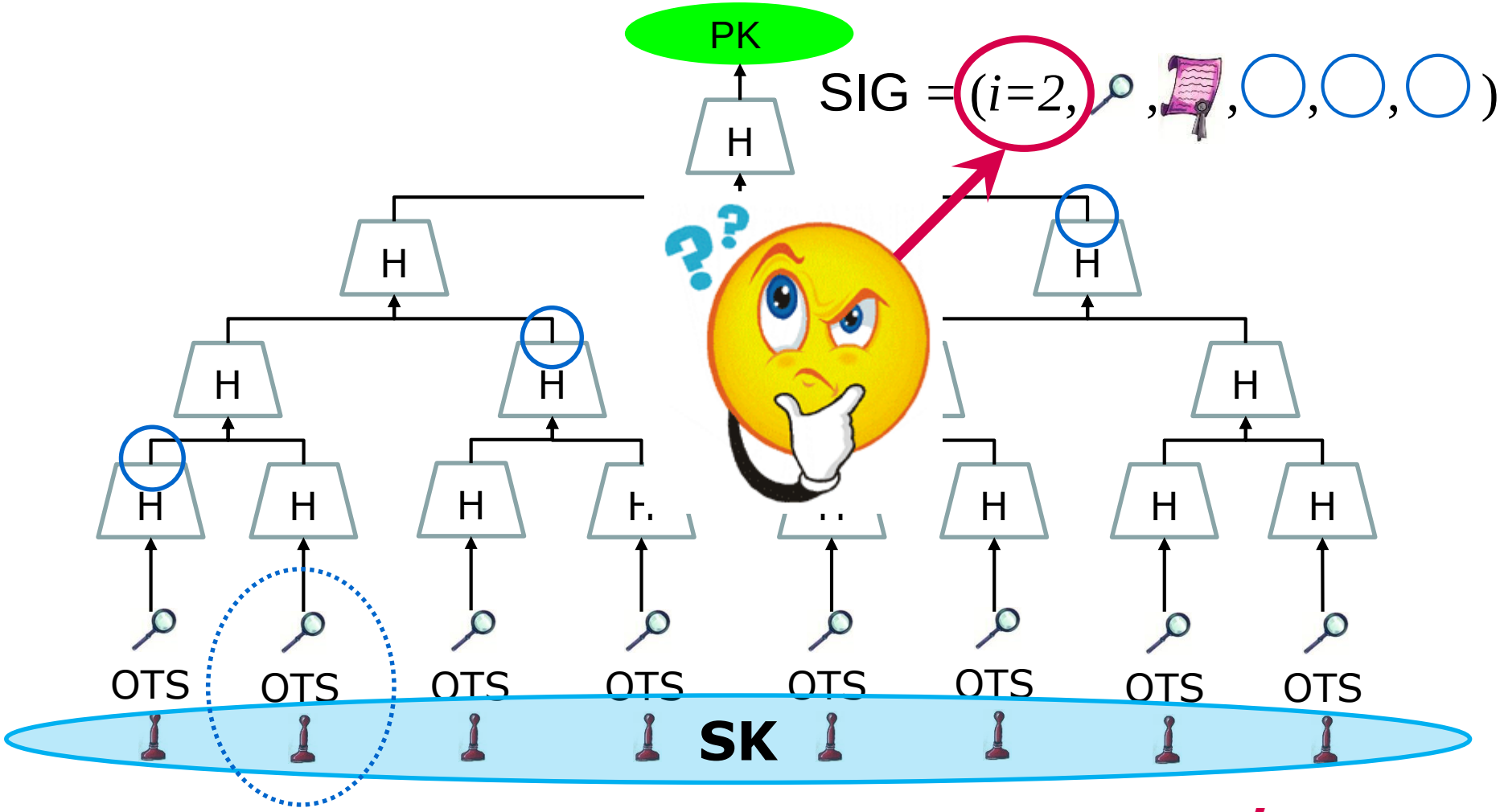
How to Eliminate the State

TU/e Technische Universiteit Eindhoven University of Technology

Protest?

Straight Forward

- **Run MSS without State**



TU/e Technische Universiteit Eindhoven University of Technology

Approach 1: Message Hash

i = Hash(Message);

- 128bit Quantum Sec.

  → $n = 256$ bit Hash [Ber09]

  → #Indices = 2^{256}

  → $h = n = 256$

- h depends on n!
- Best we can do:

  $t_{Sign} \approx n^3 / \log n \; t_{Hash} = 2M \; t_{Hash} \approx 15$ min*

  $|Sig| \approx n^3 / \log n > 256$ kb

* (OpenSSL SHA2)

TU/e Technische Universiteit Eindhoven University of Technology

Approach 2: Random Index

$$I \leftarrow_{\$} U_{\#Indices}$$

- 128bit Quantum Sec.
  → Sampled by Signer
    → #Indices ← collision prob.
    → #Indices = 2^{256}
    → $h = 256$

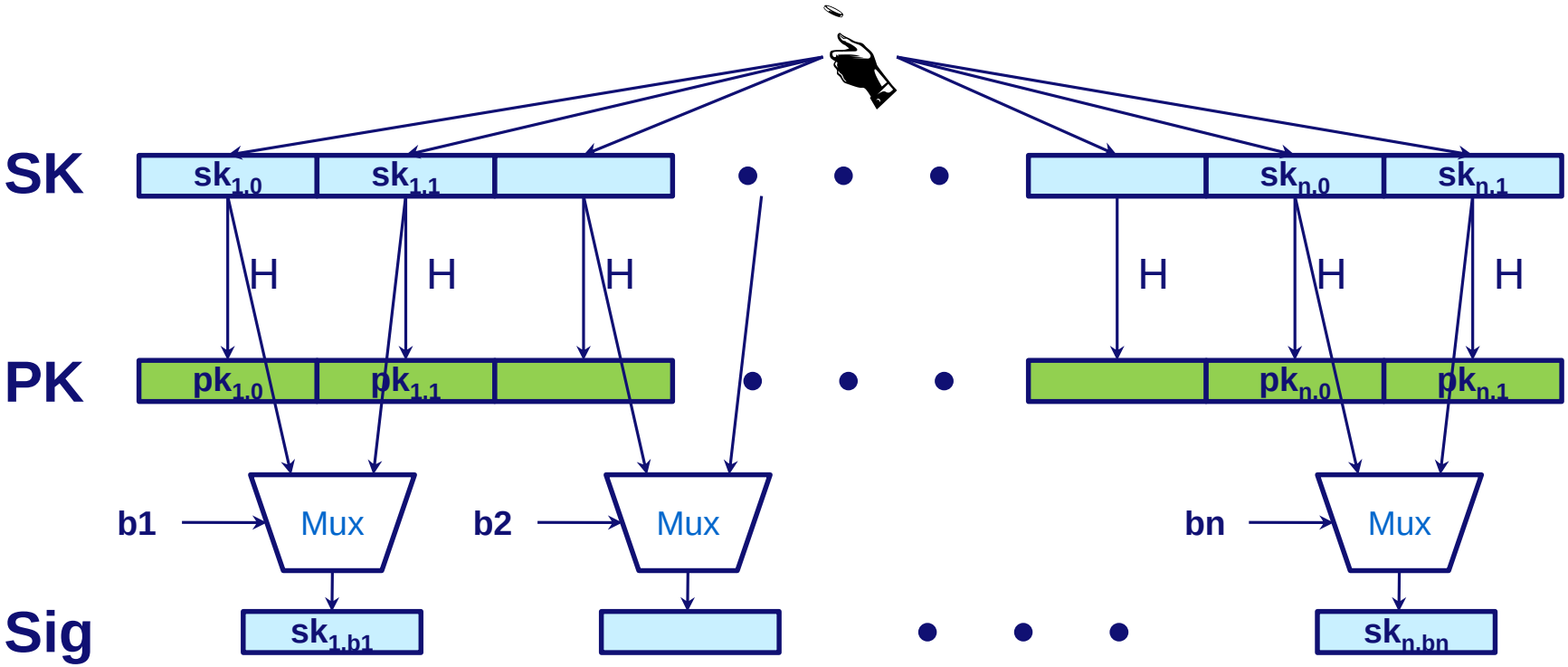
- Impossible to make this efficient, again…
- BUT:
  - h independent of n
  - Statistical collision probability NOT collision resistance

TU/e Technische Universiteit Eindhoven University of Technology

Few-Time Signature Schemes

Recap LD-OTS

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

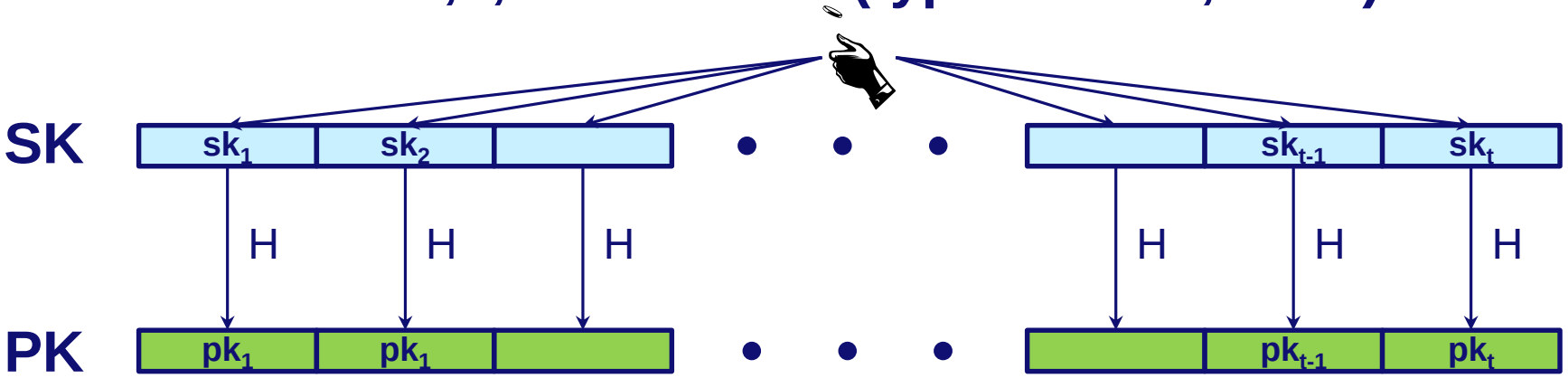


TU/e Technische Universiteit Eindhoven University of Technology

HORS [RR02]

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

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

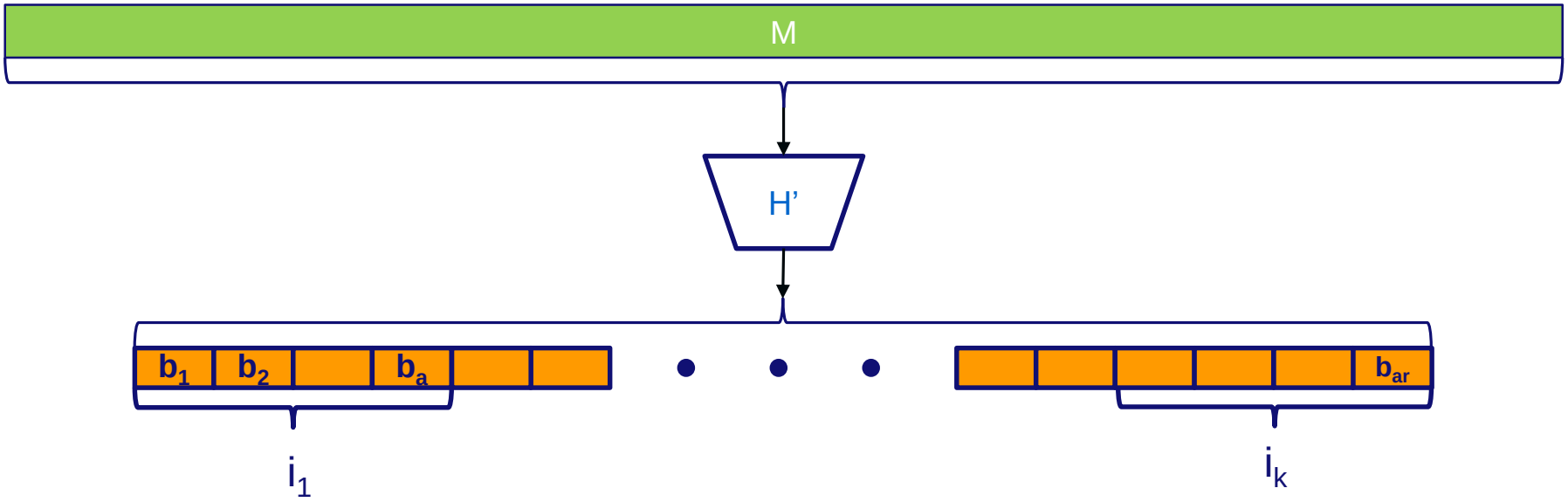


TU/e Technische Universiteit Eindhoven University of Technology

HORS mapping function

Message M, OWF H, CRHF H’ ***** **= n bit**

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

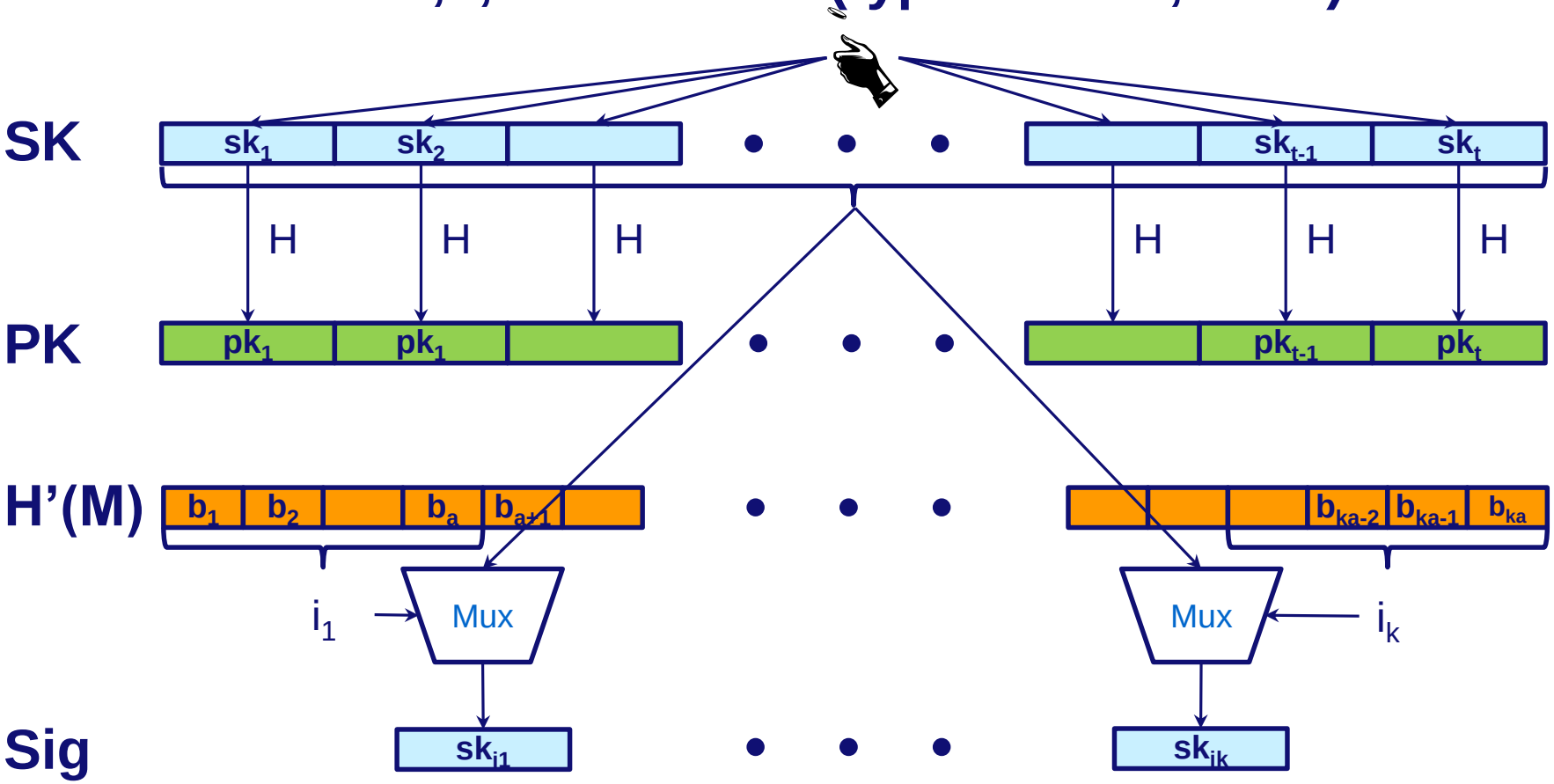


TU/e Technische Universiteit Eindhoven University of Technology

HORS

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

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



TU/e Technische Universiteit Eindhoven University of Technology

HORS Security

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

- r-Subset-Resilience: After seeing index sets M^i_j for r messages msg_j, $1 <= j <= r$, hard to find $msg_{r+1} \neq msg_j$ such that $M^i_{r+1} \in \cup_{1<=j<=r} M^i_j$.

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

→ Security shrinks with each signature!

TU/e Technische Universiteit Eindhoven University of Technology

HORST

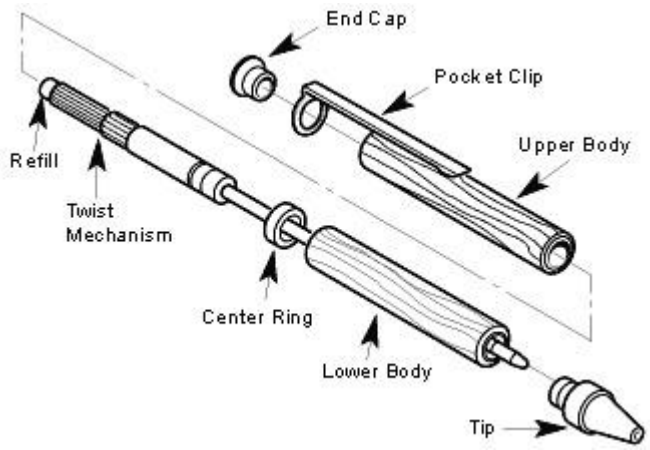
Using HORS with MSS requires adding PK (tn) to MSS signature.

HORST: Merkle Tree on top of HORS-PK

- **New PK = Root**
- **Publish Authentication 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**

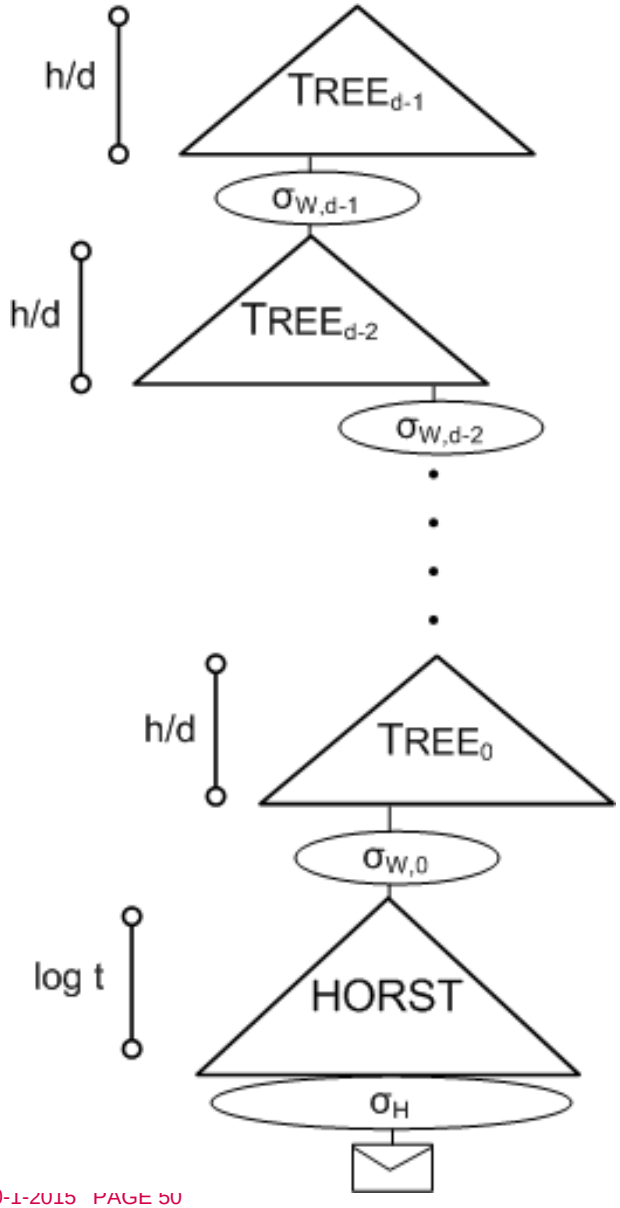
TU/e Technische Universiteit Eindhoven University of Technology

Assembling SPHINCS



TU/e Technische Universiteit Eindhoven University of Technology

SPHINCS Signature



TU/e Technische Universiteit Eindhoven University of Technology

SPHINCS Key Ideas

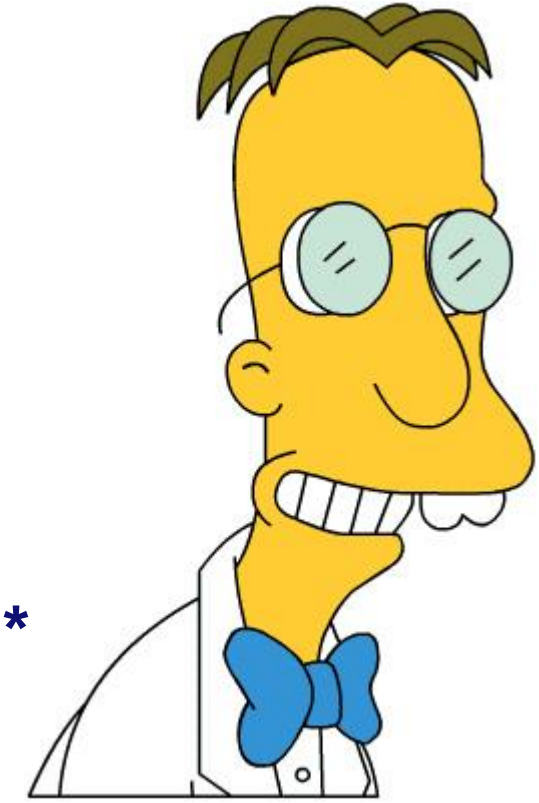
Use HORST key pairs to sign messages

Authenticate HORST key pairs using hypertree (of XMSS trees)

Use random index

Select Parameters such that

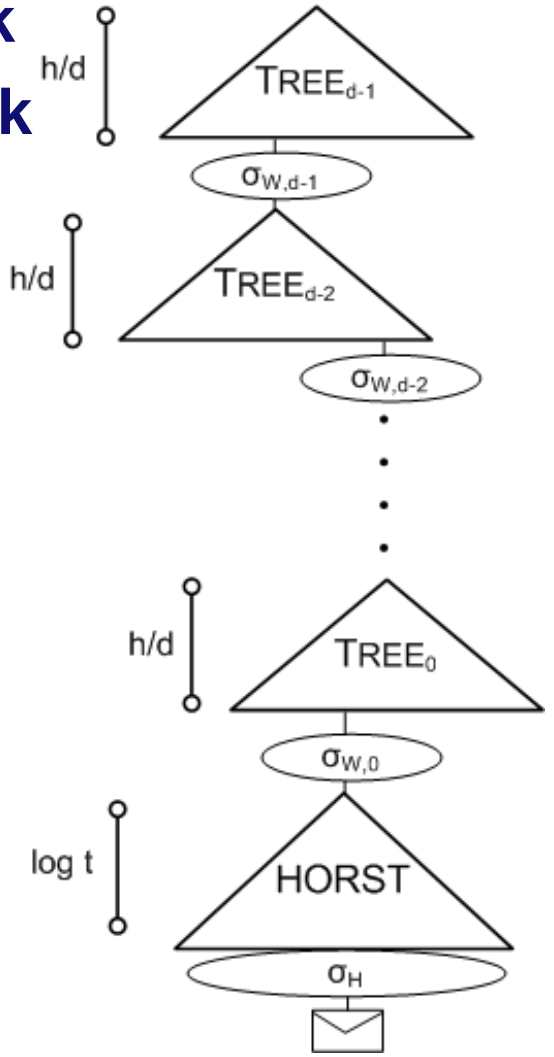
$$\text{sum}_{r \in [0,\infty)}(\Pr[\text{r-times index collision}] * \text{Succ}_{\text{r-SSR}}(A)) = \text{negl}(n)$$



TU/e Technische Universiteit Eindhoven University of Technology

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



TU/e Technische Universiteit Eindhoven University of Technology

SPHINCS-256

Table 1: SPHINCS-256 parameters and functions for the 128-bit post-quantum security level and resulting signature and key sizes.

Parameter	Value	Meaning
n	256	bitlength of hashes in HORST and WOTS
m	512	bitlength of the message hash
h	60	height of the hyper tree
d	12	layers of the hyper tree
w	16	Winternitz parameter used for WOTS signatures
t	2^{16}	number of secret-key elements of HORST
k	32	number of revealed secret-key elements per HORST signature
Functions		
Hash $\mathcal{H}$:	$\mathcal{H}(R, M) = \text{BLAKE-512}(R\|\|M)$	
PRF $\mathcal{F}_a$:	$\mathcal{F}_a(A, K) = \text{BLAKE-256}(K\|\|A)$	
PRF $\mathcal{F}$:	$\mathcal{F}(M, K) = \text{BLAKE-512}(K\|\|M)$	
PRG G_λ:	$\text{G}_\lambda(\text{SEED}) = \text{ChaCha12}_{\text{SEED}}(0)_{0,\ldots,\lambda-1}$	
Hash F:	$\text{F}(M_1) = \text{CHOP}(\pi_{ChaCha}(M_1\|\|C), 256)$	
Hash H:	$\text{H}(M_1\|\|M_2) = \text{CHOP}\left(\pi_{ChaCha}\left(\pi_{ChaCha}\left(M_1\|\|C\right) \oplus \left(M_2\|\|0^{256}\right)\right), 256\right)$	
Sizes		
Signature size:	41000 bytes	
Public-key size:	1056 bytes	
Private-key size:	1088 bytes	

TU/e Technische Universiteit Eindhoven University of Technology

SPHINCS-256 Speed

- **Key generation:** 3,051,562 cycles
- **Verification:** 1,369,060 cycles
- **Signature:** 47,466,005 cycles

- **Still hundreds of messages per second on a modern 4-core 3.5GHz Intel CPU (13.56 ms / Sig on 1 Core)**

- **Remember: Optimized Folklore**

$$t_{Sign} \approx 15 \text{ min}^*$$
$$|Sig| > 256 \text{ kb}$$

TU/e Technische Universiteit Eindhoven University of Technology

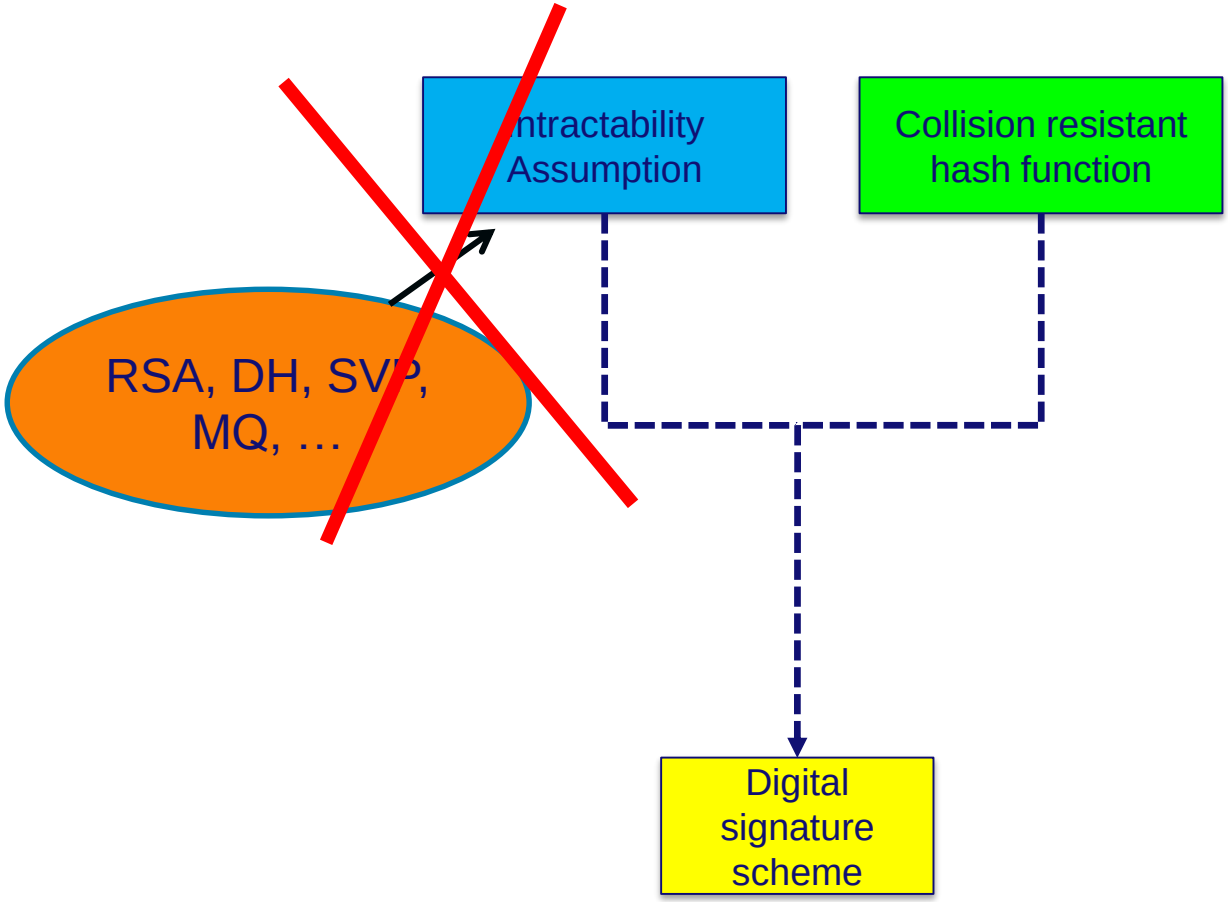
In Paper (http://eprint.iacr.org/2014/795)

- Standard model security reduction without collision resistance
- Complexity of generic quantum attacks
- Efficient fixed-input length hashing
- Optimized implementation

Advantages of Hash-based Signatures

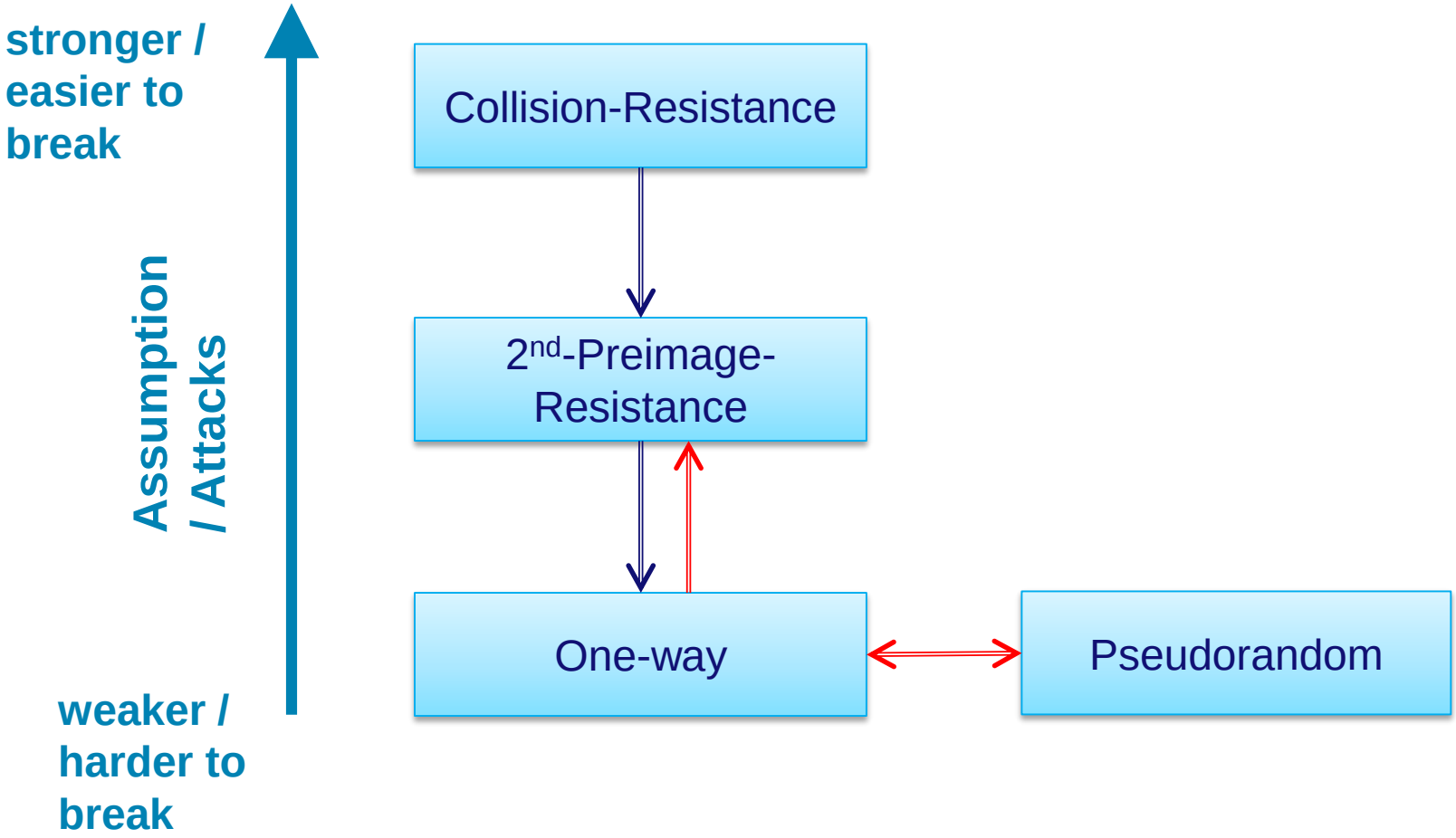
TU/e Technische Universiteit Eindhoven University of Technology

RSA – DSA – EC-DSA...



TU/e Technische Universiteit Eindhoven University of Technology

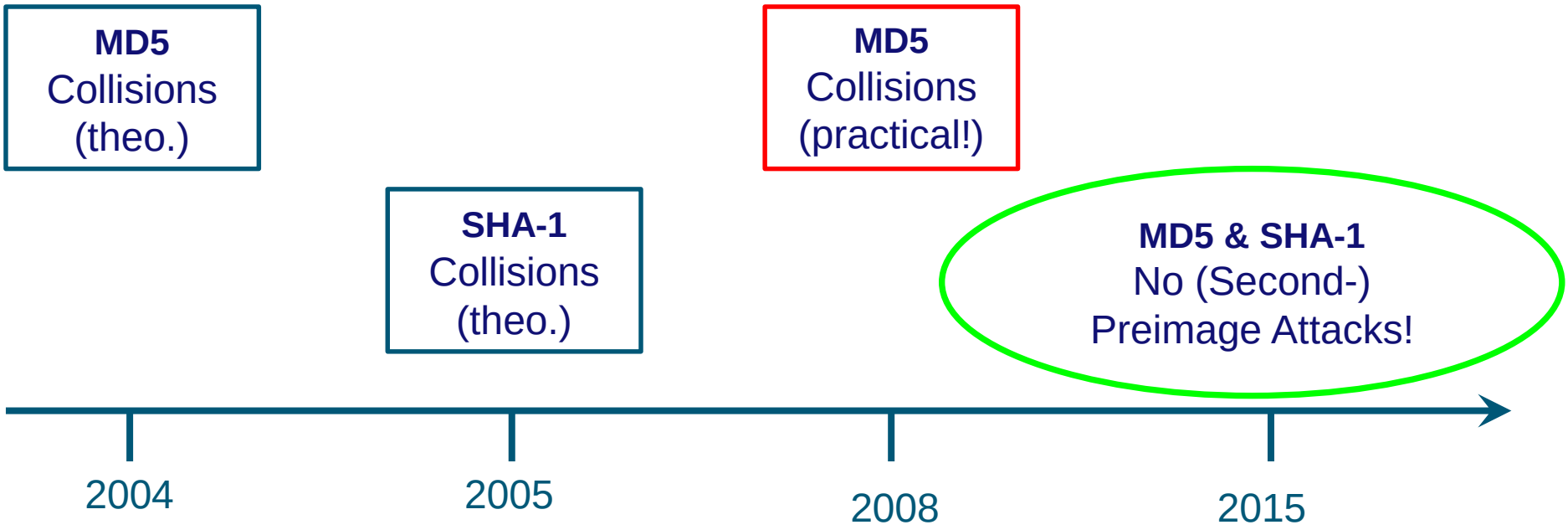
Early Warning System (only XMSS & SPHINCS)
Hash-function properties



TU/e Technische Universiteit Eindhoven University of Technology

Early Warning System (only XMSS & SPHINCS)

Attacks on Hash Functions



TU/e Technische Universiteit Eindhoven University of Technology

Easy Redundancy

Hash-Combiner

- Collision-Resistance / 2nd-Preimage-Resistance:

$$h_k(x) = g_k(x) \| f_k(x)$$

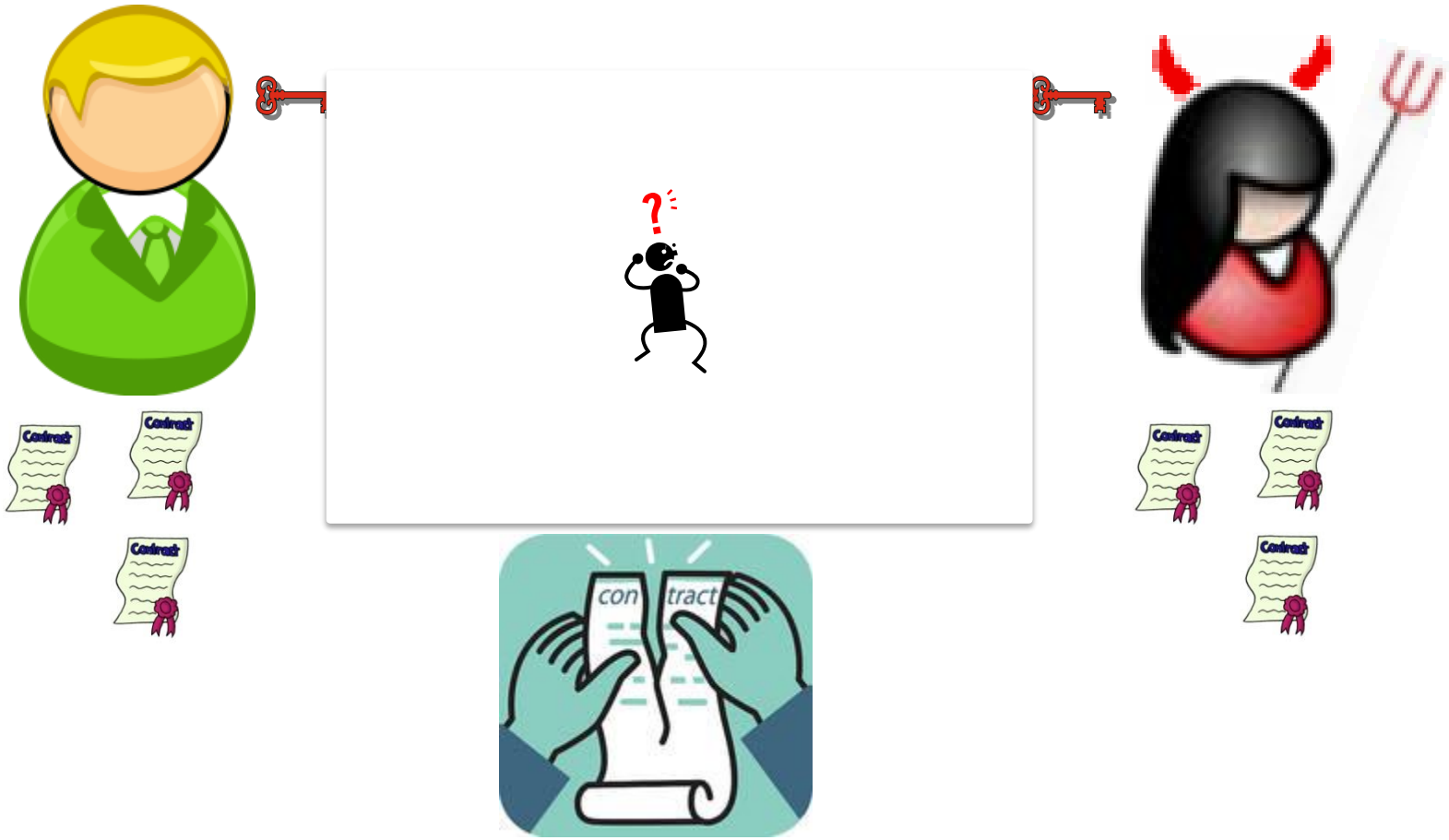
- PRF:

$$h_k(x) = g_k(x) \oplus f_k(x)$$

- **No sudden break**
- **Replaces double signature**
- **Signature size only grows by h*n**
- **Runtime ~ doubled**

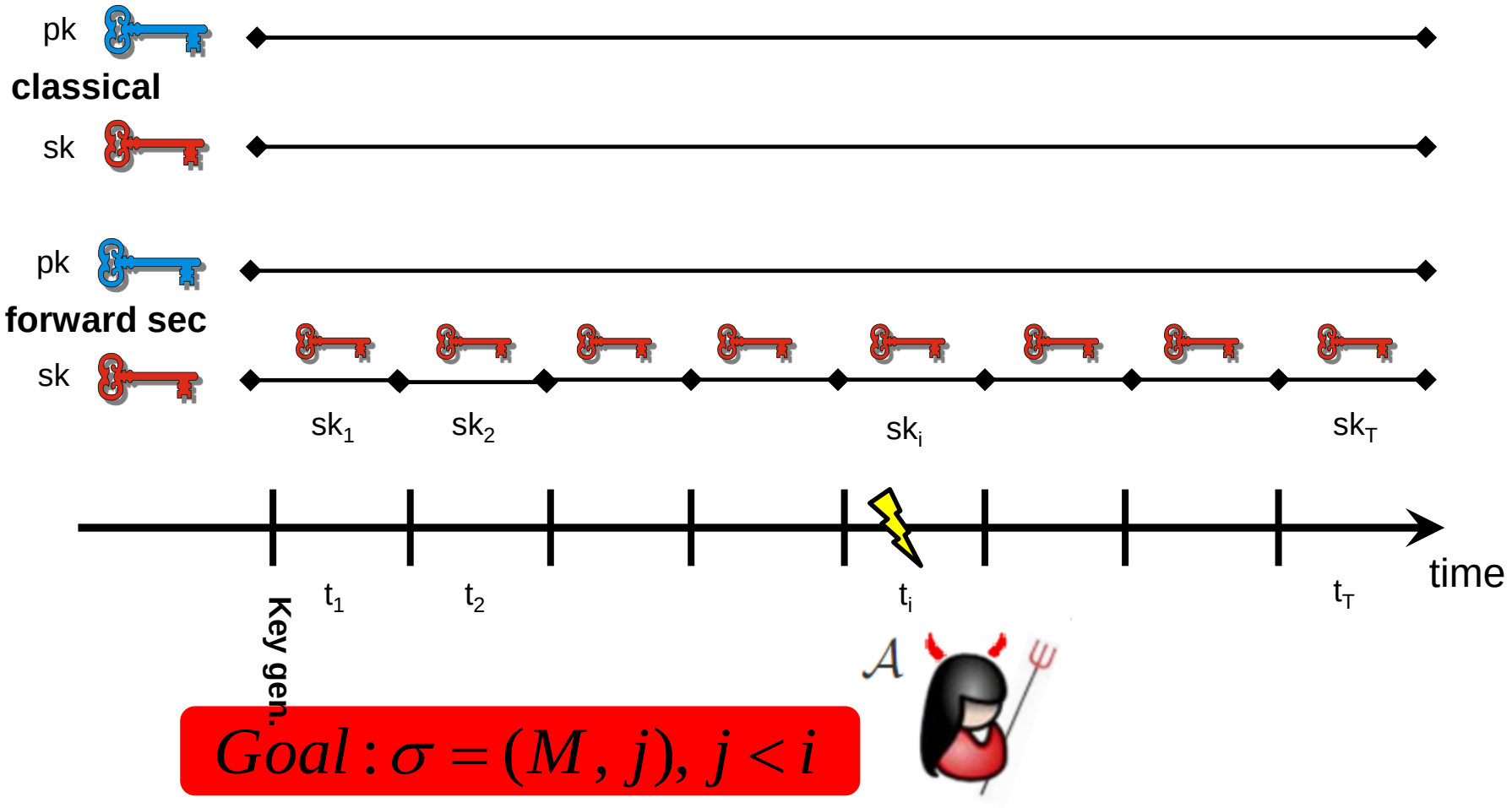
TU/e Technische Universiteit Eindhoven University of Technology

Forward Security (only XMSS)



TU/e Technische Universiteit Eindhoven University of Technology

Forward Security - cont'd



TU/e Technische Universiteit Eindhoven University of Technology

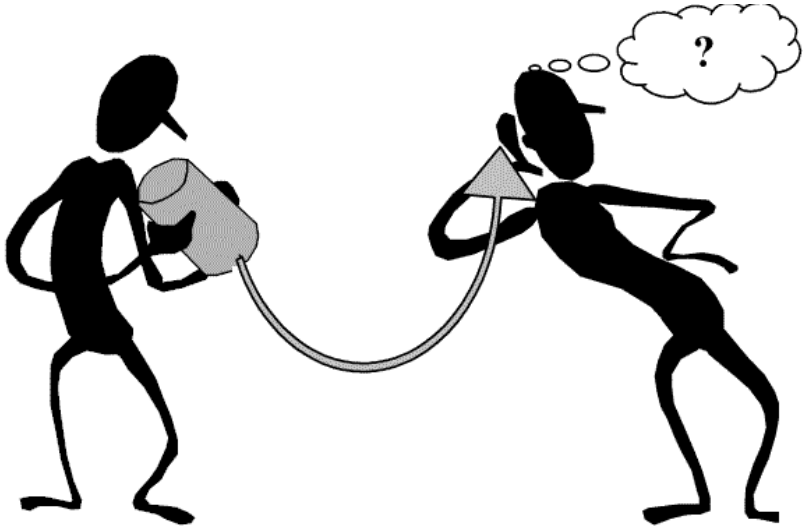
Conclusion

- **Hash-based signatures currently most confidence inspiring pq-signature scheme**
- **If you can live with a state: Go for XMSS.**
- **Otherwise:**
  - **Go for Sphincs-256!**
  - **First stateless signature scheme with post-quantum secure parameters**
  - **Practical speed and sizes**

TU/e Technische Universiteit Eindhoven University of Technology

Thank you!

Questions?



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