Long-term secure authenticity using hash-based signatures

Andreas Hülsing

PLLS 2018
17/09/2018

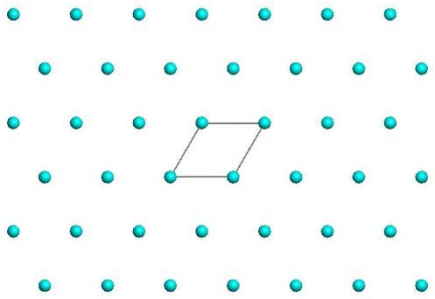
Requirements for long-term authenticity

- Example: Land registry
- Lifetime? 100+ years
- Known solution:
  - Digital Archiving with signature renewal
- Requirements:
  - Security of signature scheme must„fade out“ rather than „vanish suddenly“
  - Can be achieved using double signature
- What about quantum computers?
  - How many different sigs do we need?

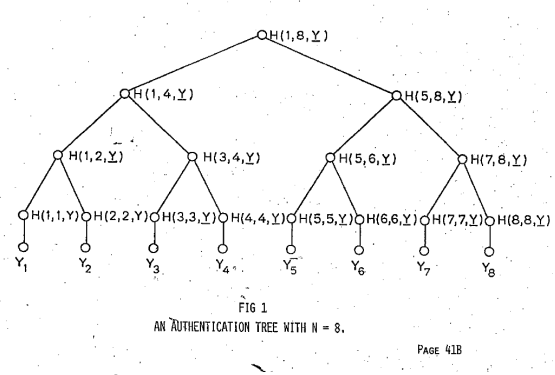
Post-quantum signature schemes

Proposals from all areas of post-quantum cryptography:

Lattice-based: SVP / CVP



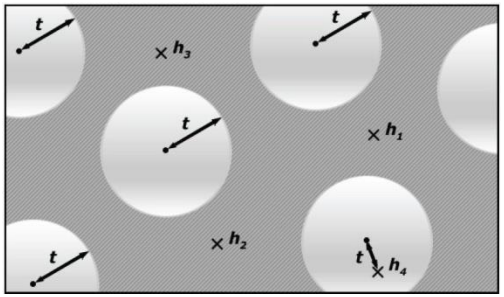
Hash-based: CR / SPR / ...



New

Isogenies

Code-based: SD



Multivariate: MQ

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

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

$$y_3 = \ldots$$

Hash-based Signature Schemes

[Mer89]

The conservative approach:

Instead of introducing new hardness assumptions...

...reduce the amount of assumptions

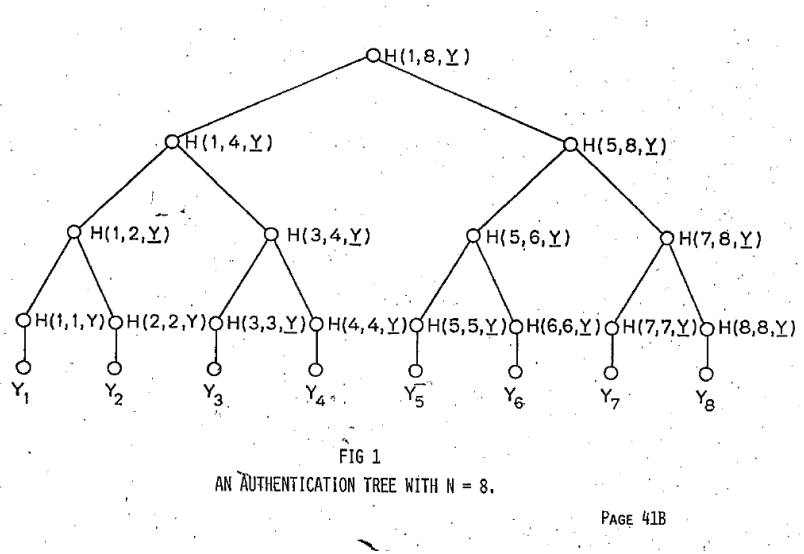
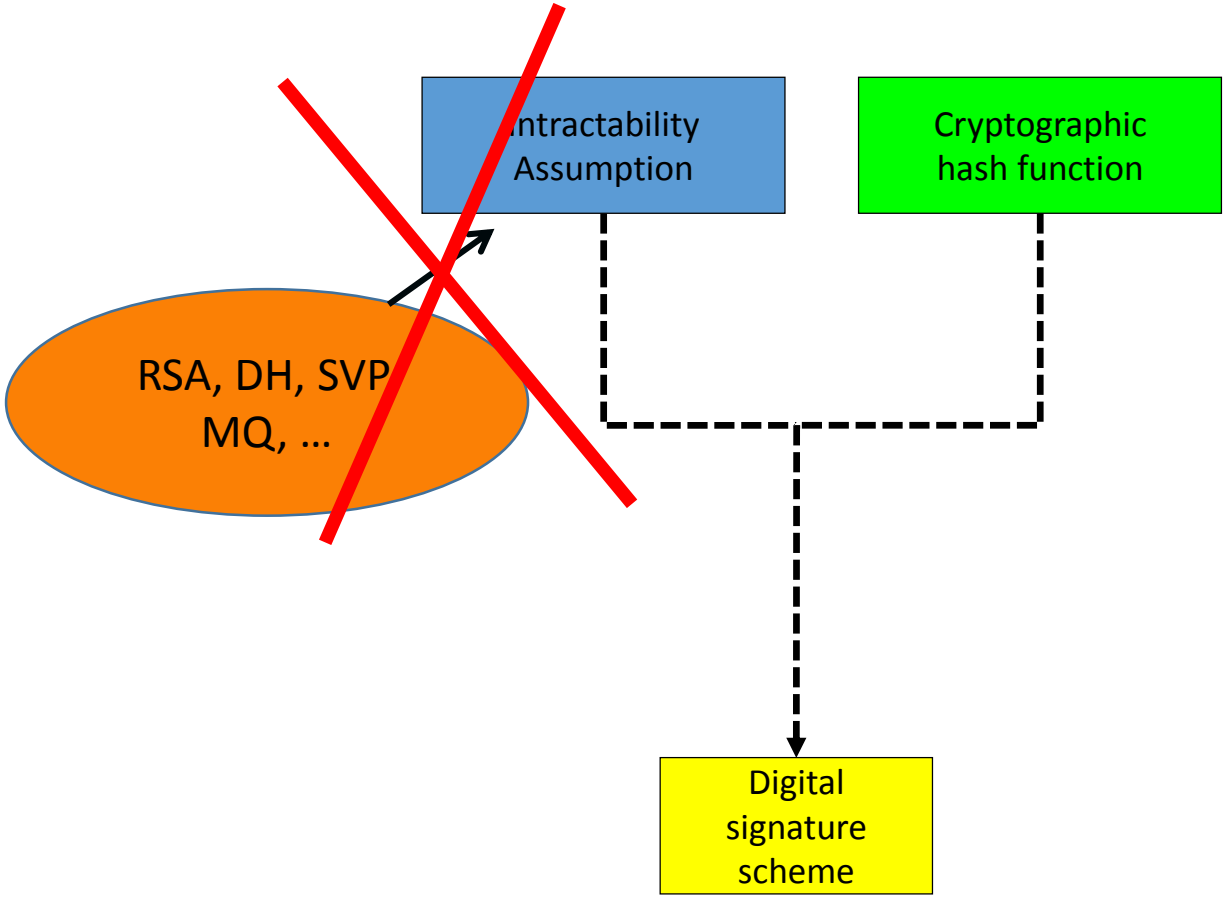


FIG 1

AN AUTHENTICATION TREE WITH N = 8.

PAGE 41B

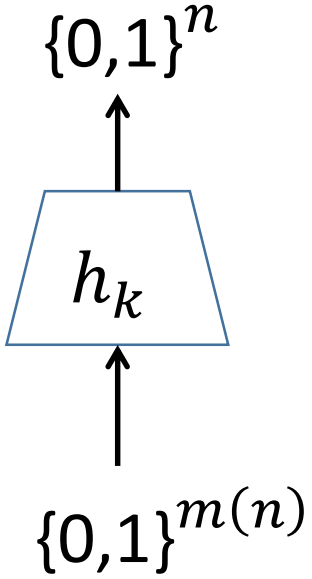
RSA – DSA – EC-DSA…



Hash function families

(Hash) function families

- $H_n := \{h_k : \{0,1\}^{m(n)} \rightarrow \{0,1\}^n | k \in \{0,1\}^n\}$
- $m(n) \geq n$
- „efficient“



One-wayness

$H_n := \{h_k : \{0,1\}^{m(n)} \rightarrow \{0,1\}^n\}$

$h_k \xleftarrow{\$} H_n$

$x \xleftarrow{\$} \{0,1\}^{m(n)}$

$y_c \leftarrow h_k(x)$

Success if $h_k(x^*) = y_c$

Collision resistance

$H_n := \{h_k : \{0,1\}^{m(n)} \rightarrow \{0,1\}^n\}$

$h_k \xleftarrow{\$} H_n$

Success if

$h_k(x_1^*) = h_k(x_2^*)$ and
$x_1^* \neq x_2^*$

Second-preimage resistance

$H_n := \{h_k : \{0,1\}^{m(n)} \to \{0,1\}^n\}$

$h_k \xleftarrow{\$} H_n$

$x_c \xleftarrow{\$} \{0,1\}^{m(n)}$

Success if

$h_k(x_c) = h_k(x^*)$ and

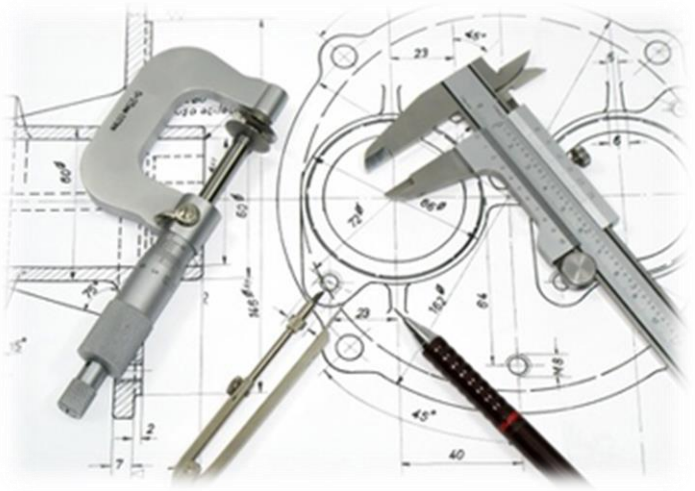
$x_c \neq x^*$

Generic Security

	OW	SPR	CR	UD*	PRF*
Classical	$\Theta(2^n)$	$\Theta(2^n)$	$\Theta(2^{n/2})$	$\Theta(2^n)$	$\Theta(2^n)$
Quantum	$\Theta(2^{n/2})$	$\Theta(2^{n/2})$	$\Theta(2^{n/3})$	$\Theta(2^{n/2})$	$\Theta(2^{n/2})$

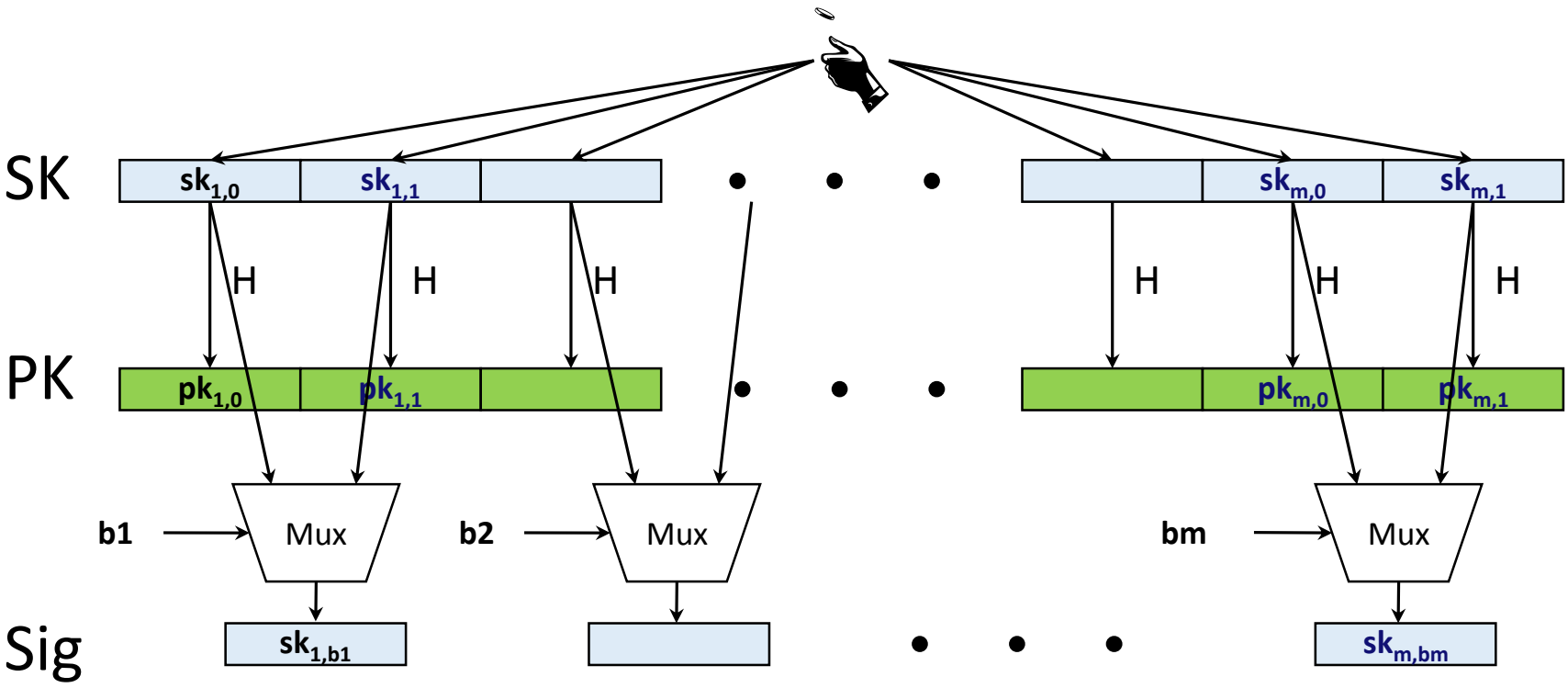
* conjectured, no proof

Basic Construction



Lamport-Diffie OTS [Lam79]

Message M = b1,…,bm, OWF H * = n bit

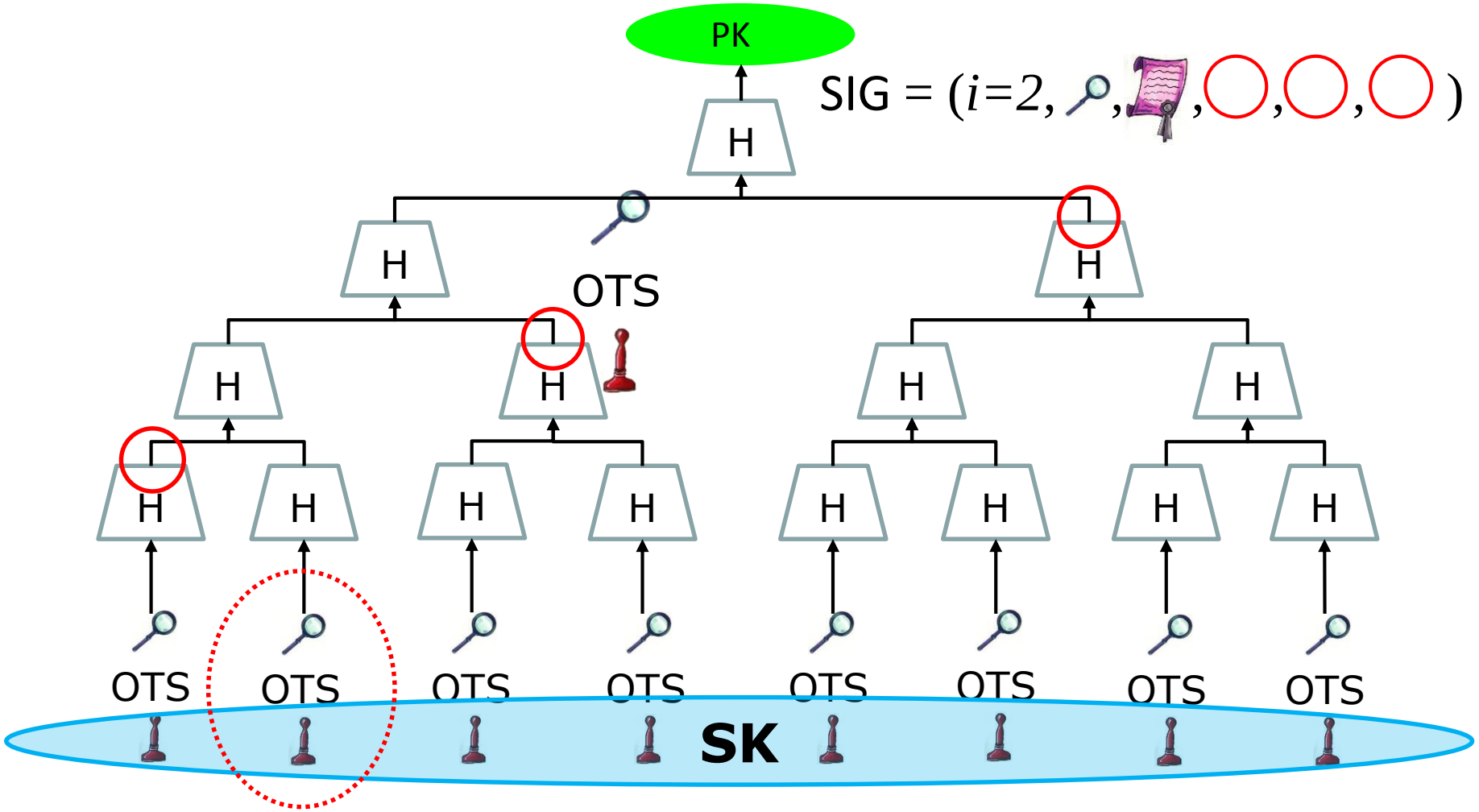


Security

Theorem:

If H is one-way then LD-OTS is one-time eu-cma-secure.

Merkle's Hash-based Signatures



Security

Theorem:

MSS is eu-cma-secure if OTS is a one-time eu-cma secure signature scheme and H is a random element from a family of collision resistant hash functions.

Winternitz-OTS

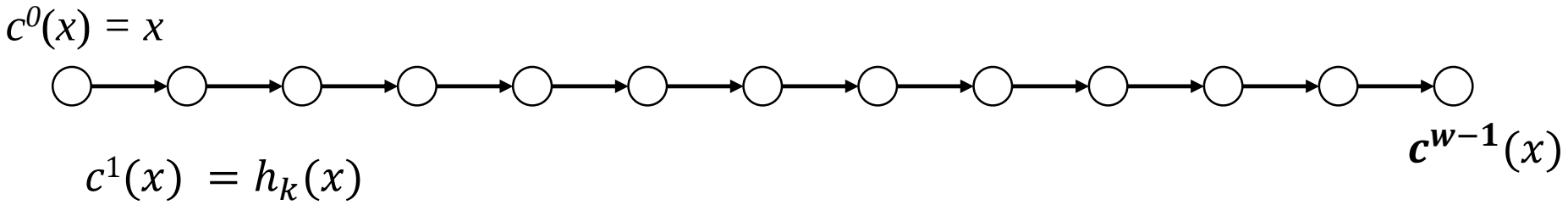
Function chains

Function family: $H_n := \{h_k : \{0,1\}^n \rightarrow \{0,1\}^n\}$

$h_k \xleftarrow{\$} H_n$

Parameter w

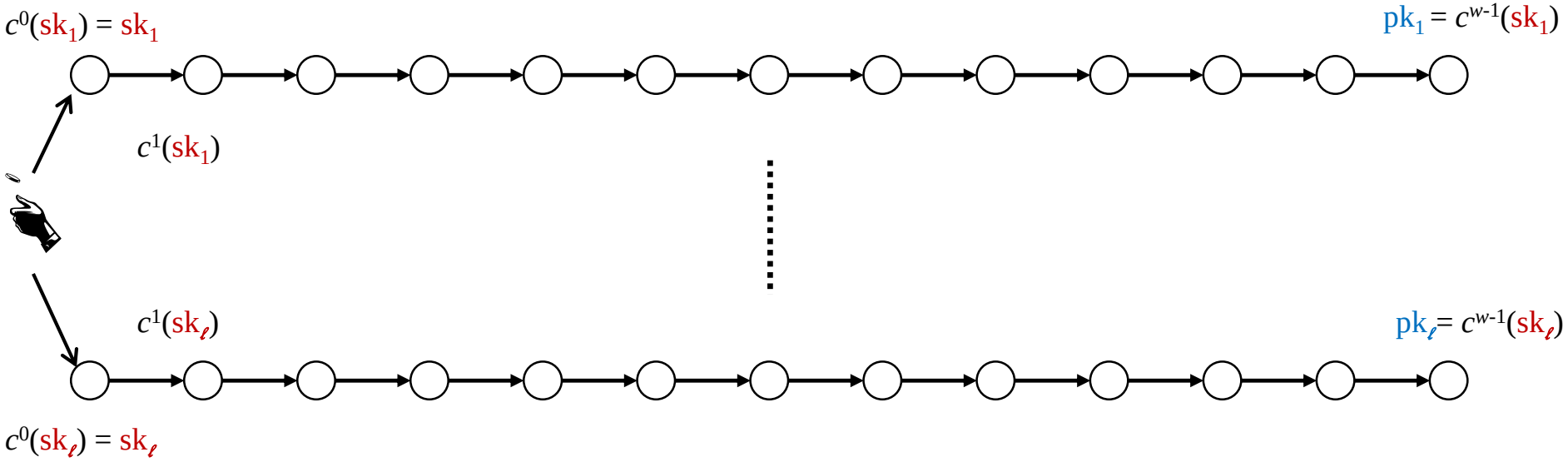
Chain: $c^i(x) = h_k\left(c^{i-1}(x)\right) = \underbrace{h_k \circ h_k \circ \cdots \circ h_k}_{\textit{i-times}}$



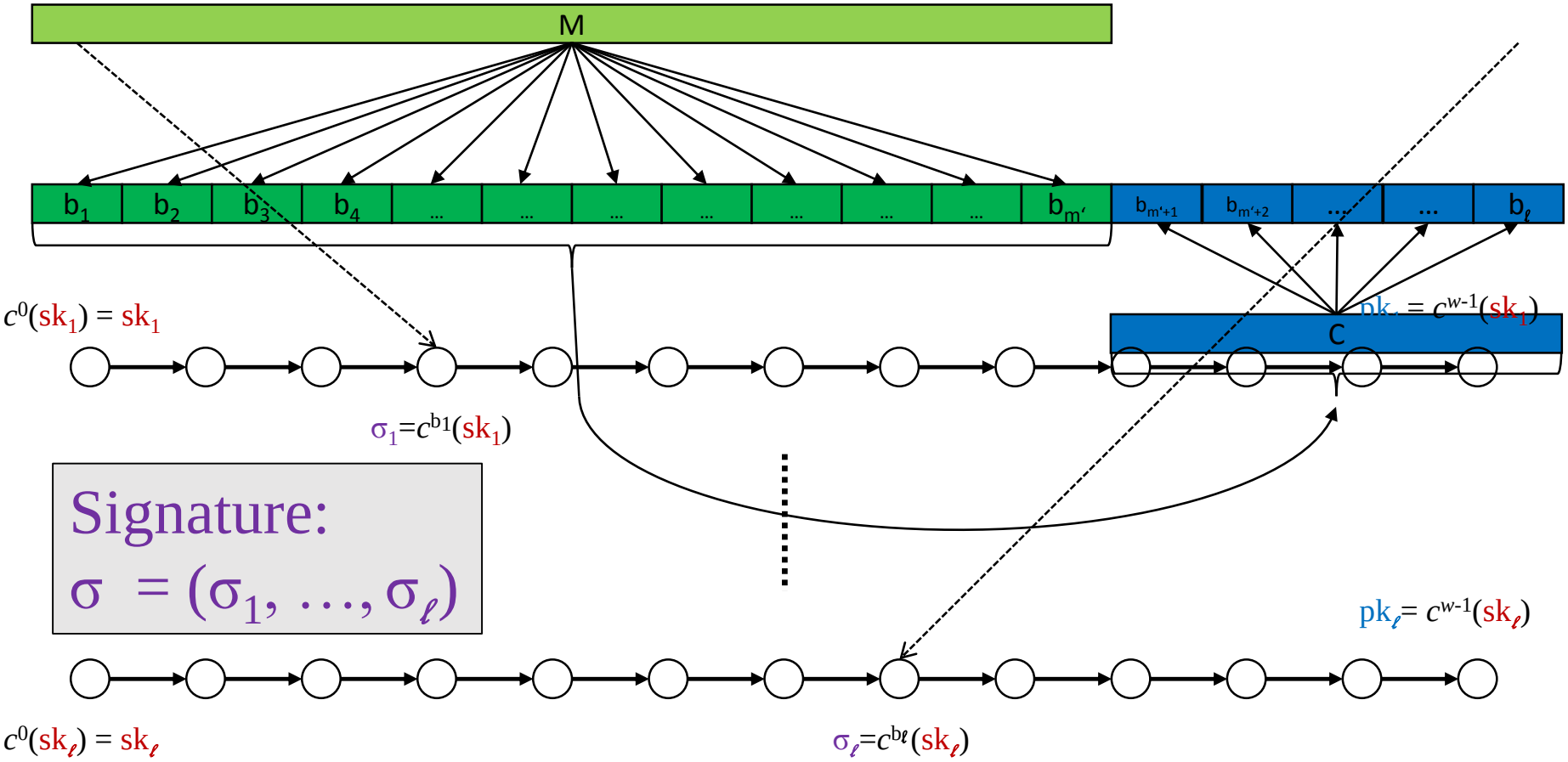
WOTS

Winternitz parameter $\boldsymbol{w}$, security parameter $\boldsymbol{n}$, message length $\boldsymbol{m}$, function family H_n

Key Generation: Compute l, sample h_k

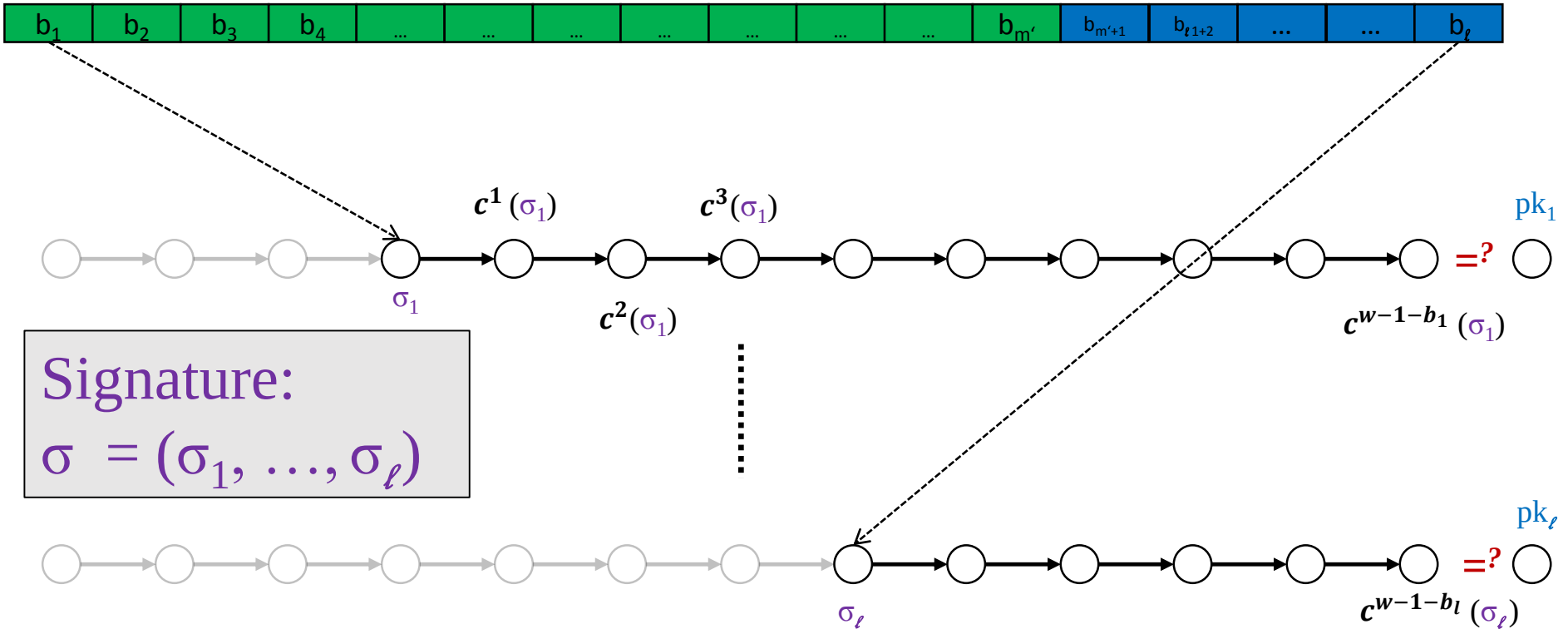


WOTS Signature generation



WOTS Signature Verification

Verifier knows: M, w



WOTS Function Chains

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

- WOTS: $c^i(x) = h_k(c^{i-1}(x))$
- WOTS⁺: $c^i(x) = h_k(c^{i-1}(x) \oplus r_i)$

WOTS Security

Theorem (informally):

W-OTS *is strongly unforgeable under chosen message attacks if* H_n *is a collision resistant family of undetectable one-way functions.*

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

WOTS in MSS

SIG = (i=2,  , ◯, ◯, ◯)

Verification:

1. Compute  from 
2. Verify authenticity of 

Steps 1 + 2 together verify 

Size decrease of factor $\approx 4 \log_2 w$

XMSS

XMSS

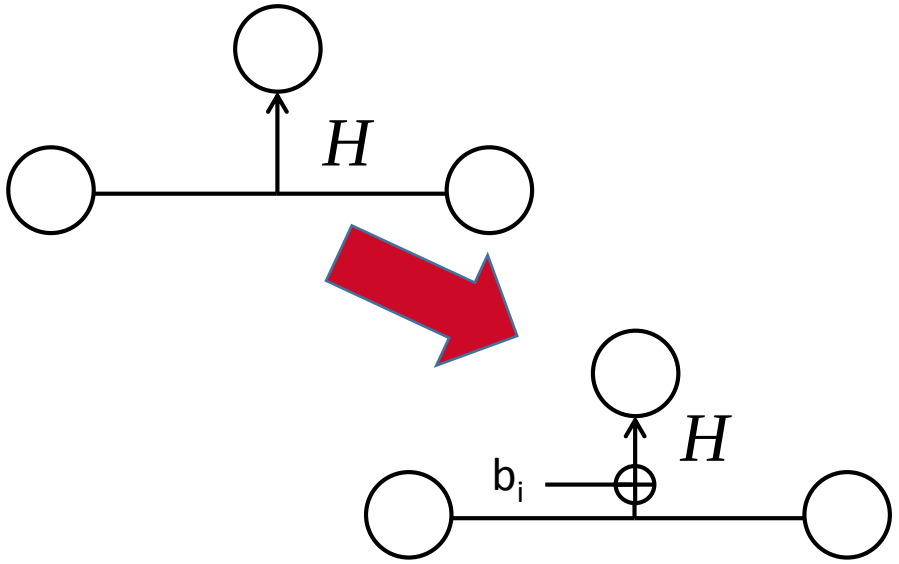
Applies several tricks to achieve **collision-resilience**
-> signature size halved

Tree: Uses bitmasks

Leafs: Use binary tree with bitmasks

OTS: WOTS$^+$

Mesage digest: Randomized hashing

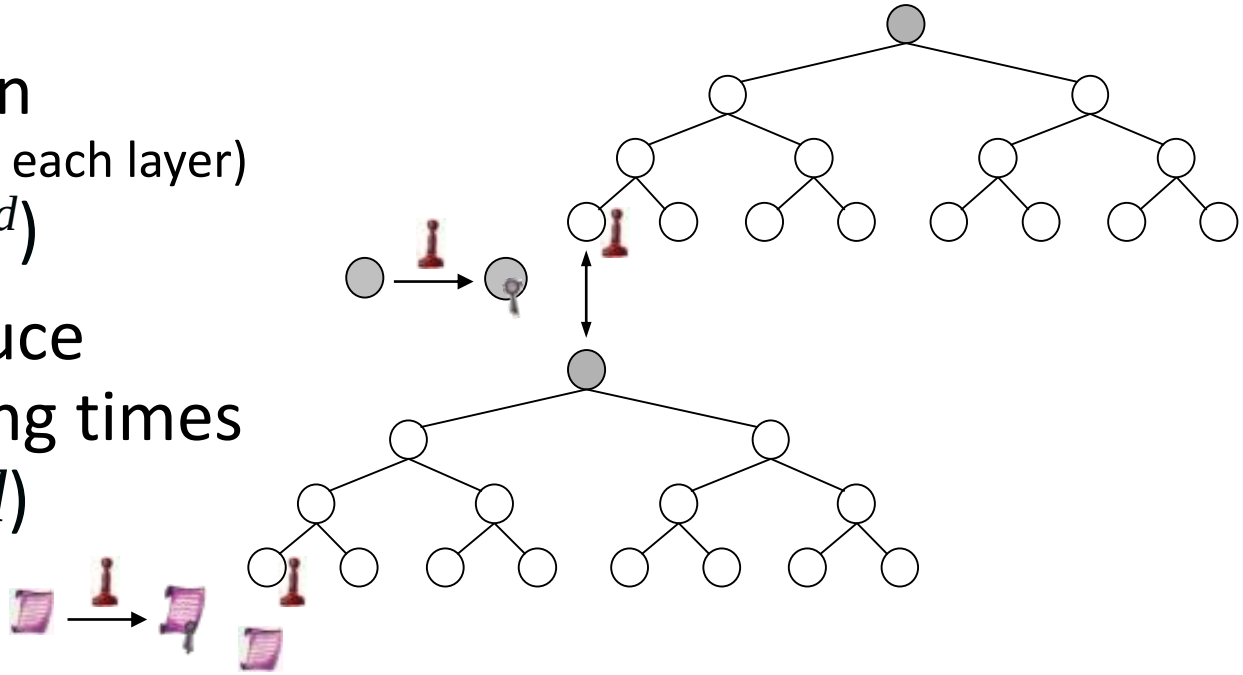


Multi-Tree XMSS

Uses multiple layers of trees to reduce key generation time

-> Key generation
(= Building first tree on each layer)
$\Theta(2^h) \rightarrow \Theta(d*2^{h/d})$

-> Allows to reduce worst-case signing times
$\Theta(h/2) \rightarrow \Theta(h/2d)$

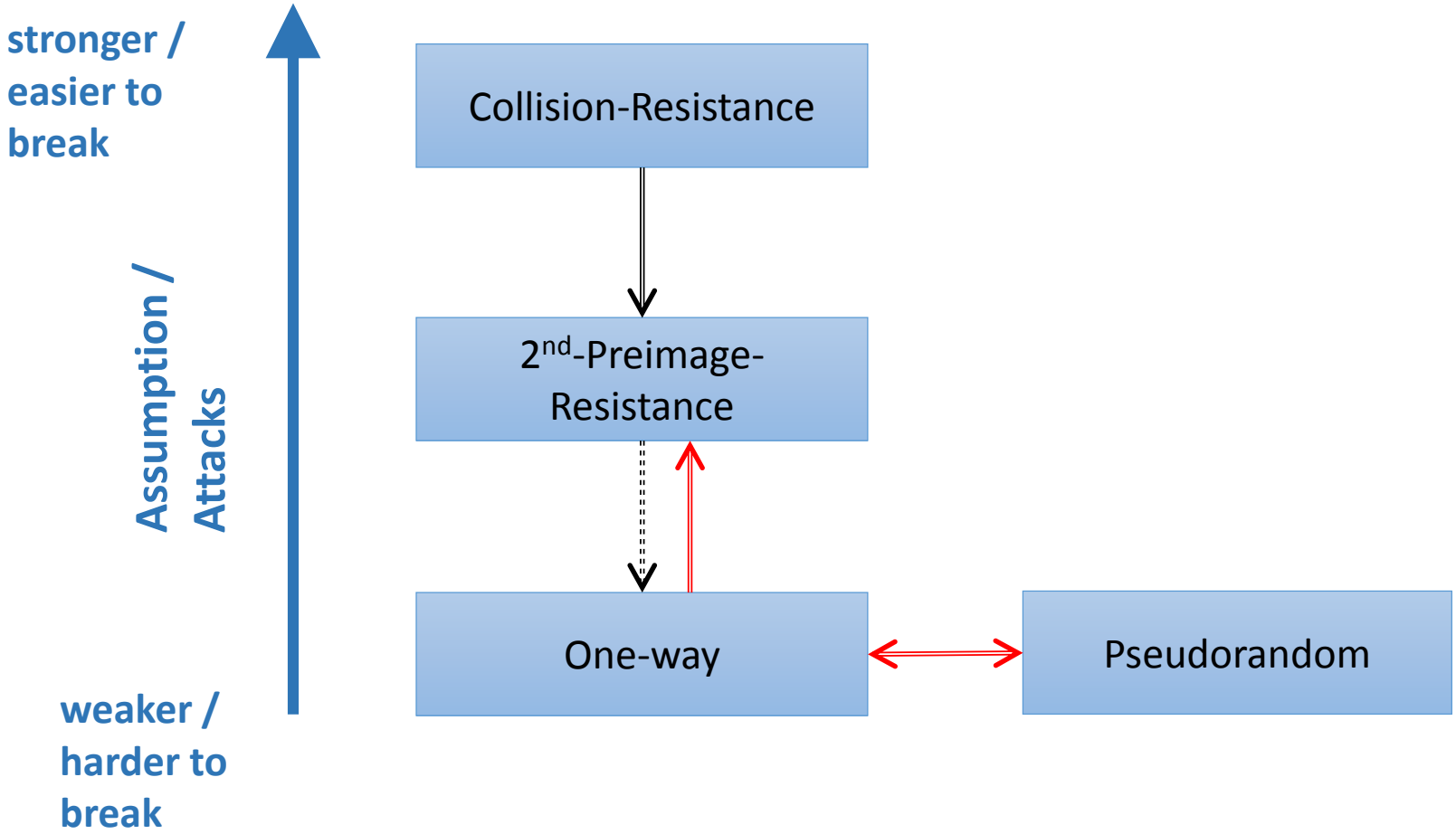


Multi-target attack mitigation

- Problem: An attack that succeeds when it solves one out of many instances (targets)
  - Typical case: Security level drops by log t for t instances
- XMSS-T / LMS / SPHINCS+ apply mitigation techniques:
  - Attack complexity for t targets becomes same as for 1 target
  - Solution: Tweakable hash function
  - Idea: Make hash calls independent
  - XMSS-T / SPHINCS+ in standard model with an additional assumption (that holds in QROM)
  - LMS in (Q)ROM

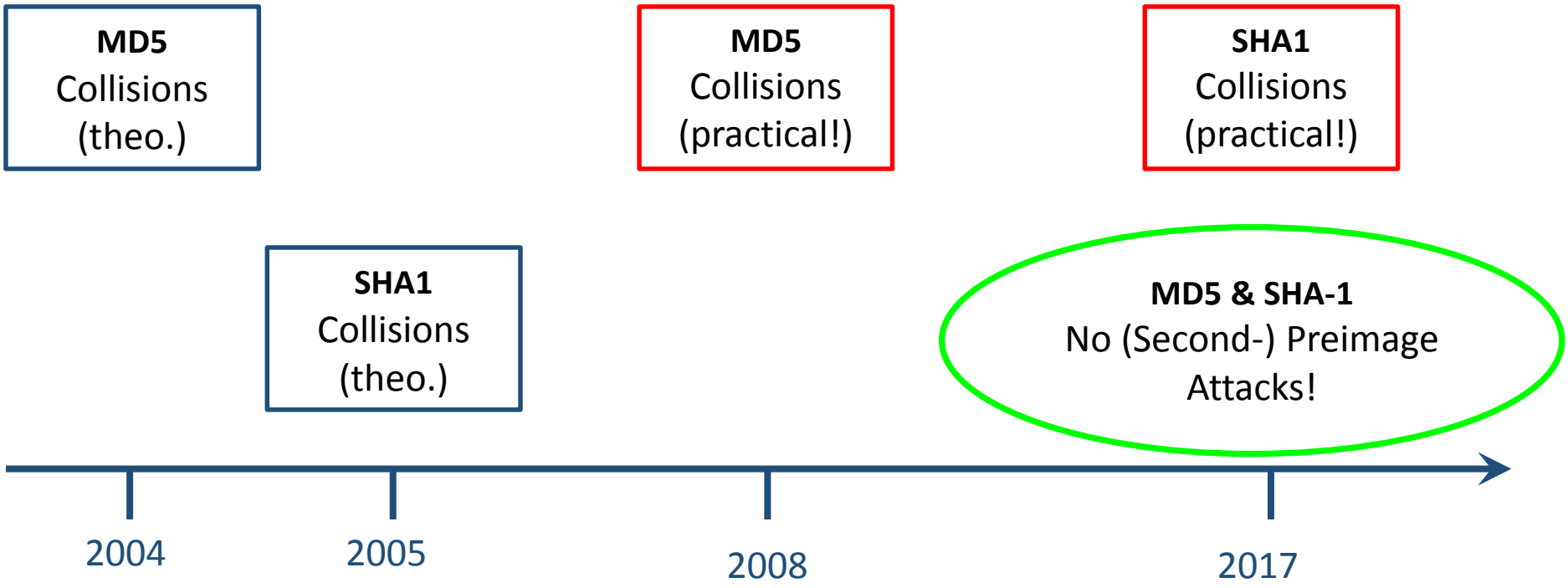
What if long-term security is needed?

Hash-function properties



This hardness gap can be used as early warning system!

Attacks on Hash Functions



Cheap Redundancy

Hash-Combiner

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

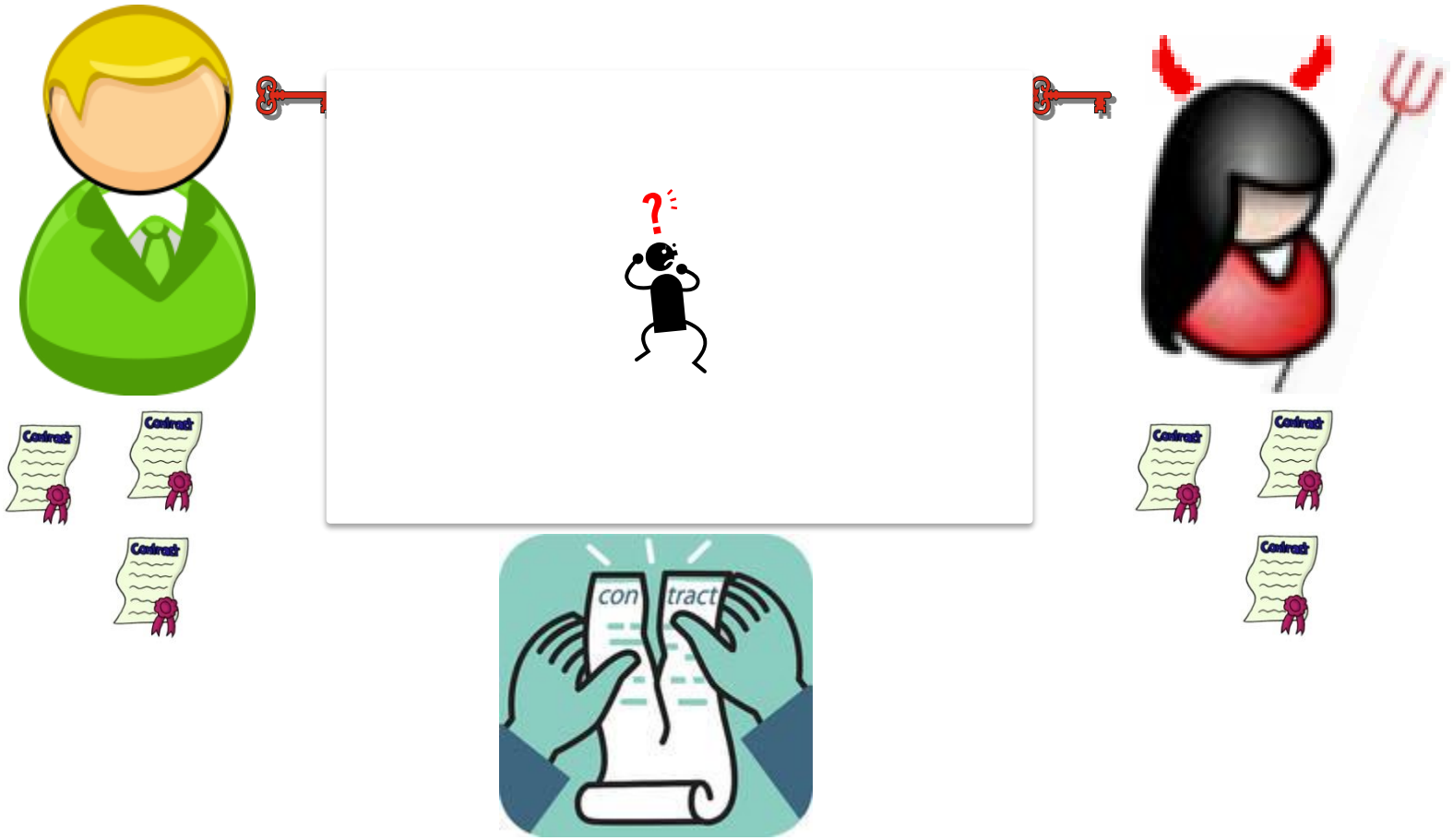
$$h_k(x) = g_k(x) \| f_k(x)$$

- PRF:

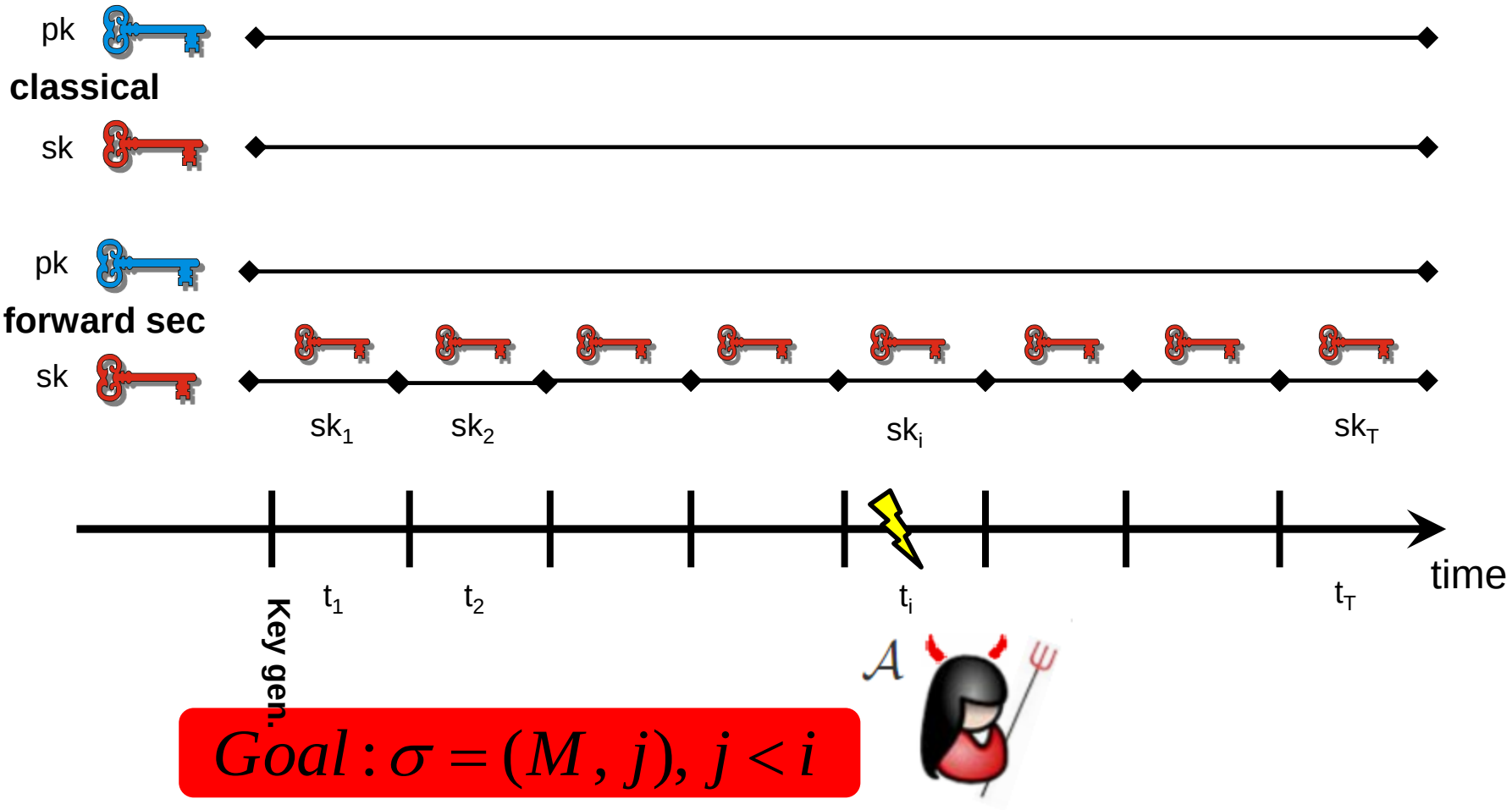
$$h_k(x) = g_k(x) \oplus f_k(x)$$

- **No sudden break**
- **Changes only in hash function**
- **Replaces double signature**
- **Signature size and runtime doubled**

Forward Security



Forward Security - cont‘d



XMSS in practice

RFC 8391: XMSS

- RFC since May 2018
- NIST promised to adopt
- Equal to XMSS-T [HRS16] up-to message digest
- Function families based on SHA2 or SHAKE
- Mandatory: Support for verification with all SHA2-256 parameter sets
- Suggested parameters for different szenarios

XMSS / XMSS-T Implementation

C Implementation, using OpenSSL [HRS16]

	Sign (ms)	Signature (kB)	Public Key (kB)	Secret Key (kB)	Bit Security classical/ quantum	Comment
XMSS	3.24	2.8	1.3	2.2	236 / 118	h = 20, d = 1,
XMSS-T	9.48	2.8	**0.064**	2.2	**256 / 128**	h = 20, d = 1
XMSS	3.59	8.3	1.3	14.6	196 / 98	h = 60, d = 3
XMSS-T	10.54	8.3	**0.064**	14.6	**256 / 128**	h = 60, d = 3

Intel(R) Core(TM) i7 CPU @ 3.50GHz
All using SHA2-256, w = 16 and k = 2

SPHINCS

About the statefulness

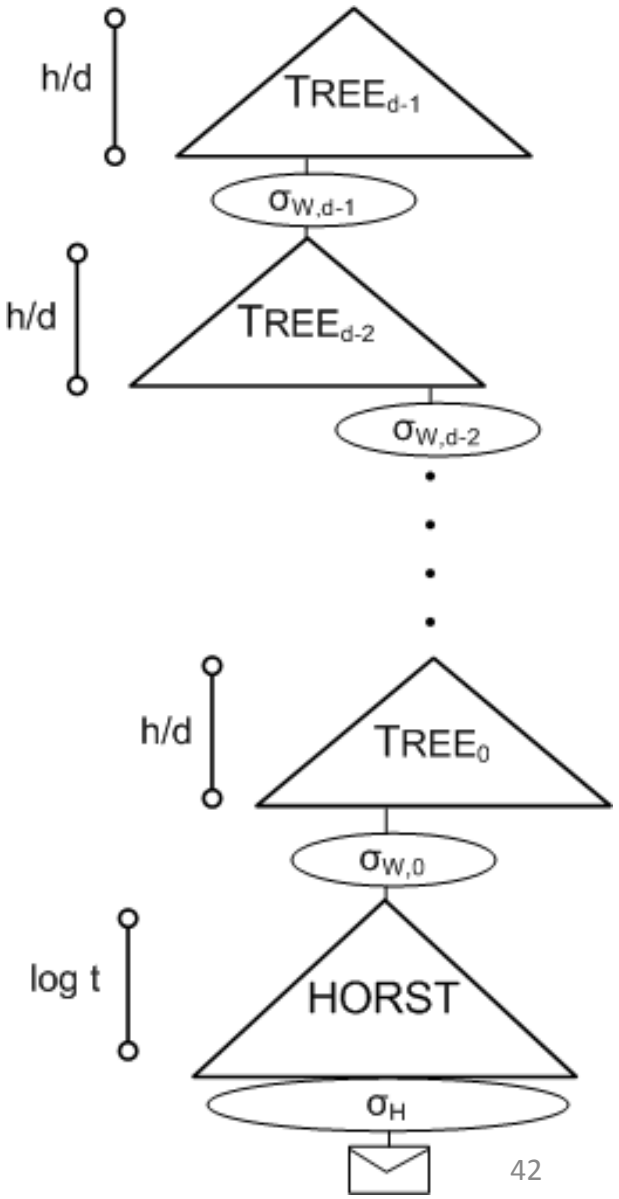
- Works great for some settings

- However....
  - ... back-up
  - ... multi-threading
  - ... load-balancing


ELIMINATE
THE STATE

SPHINCS

- Stateless Scheme
- XMSS^{MT} + HORST
  + (pseudo-)random index
- Collision-resilient
- Deterministic signing
- SPHINCS-256:
  - 128-bit post-quantum secure
  - (at least we thought so)
  - Hundreds of signatures / sec
  - 41 kb signature
  - 1 kb keys



SPHINCS$^+$ (our NIST submission)

- Strengthened security gives smaller signatures
- Collision- and multi-target attack resilient
- Small keys, medium size signatures (lv 3: 17kB)
- THE conservative choice
- No citable speeds yet

Instantiations

- SPHINCS$^+$-SHAKE256
- SPHINCS$^+$-SHA-256
- SPHINCS$^+$-Haraka

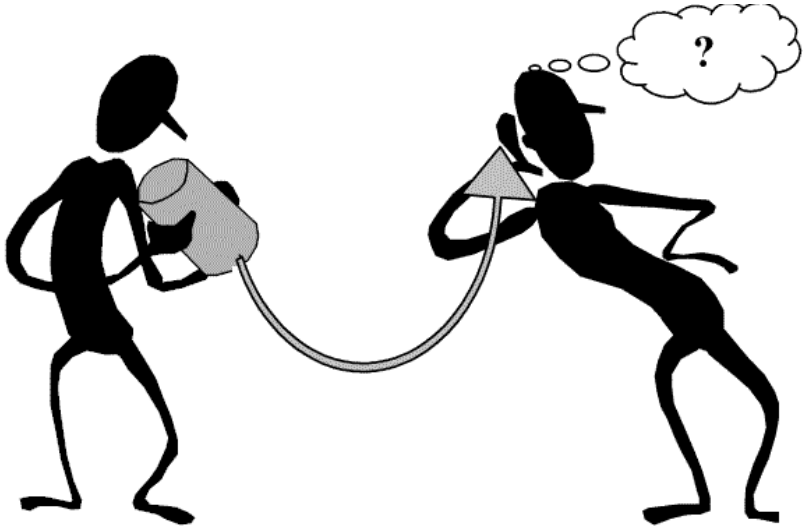
Instantiations (small vs fast)

	n	h	d	$\log(t)$	k	w	bitsec	sec level	sig bytes
SPHINCS$^+$-128s	16	64	8	15	10	16	133	**1**	8 080
SPHINCS$^+$-128f	16	60	20	9	30	16	128	**1**	16 976
SPHINCS$^+$-192s	24	64	8	16	14	16	196	**3**	17 064
SPHINCS$^+$-192f	24	66	22	8	33	16	194	**3**	35 664
SPHINCS$^+$-256s	32	64	8	14	22	16	255	**5**	29 792
SPHINCS$^+$-256f	32	68	17	10	30	16	254	**5**	49 216

Conclusion

- Practical stateful and stateless solutions
- Forward-security only possible for stateful schemes!
- Stateful only if you are 100% sure you can handle state
- My suggestion: Stateful on dedicated HW (Smartcard, HSM,...)
- Everywhere else: In case of doubt use stateless

Thank you!
Questions?



For references, literature & longer lectures see https://huelsing.net