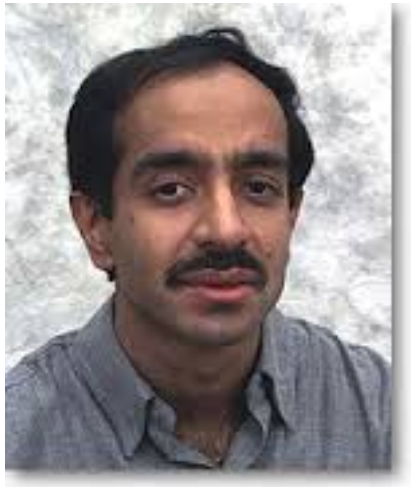
An attempt to simplify security arguments for hash-based signatures

Andreas Hülsing
Eindhoven University of Technology

Oxford Post-Quantum Cryptography Workshop 2019

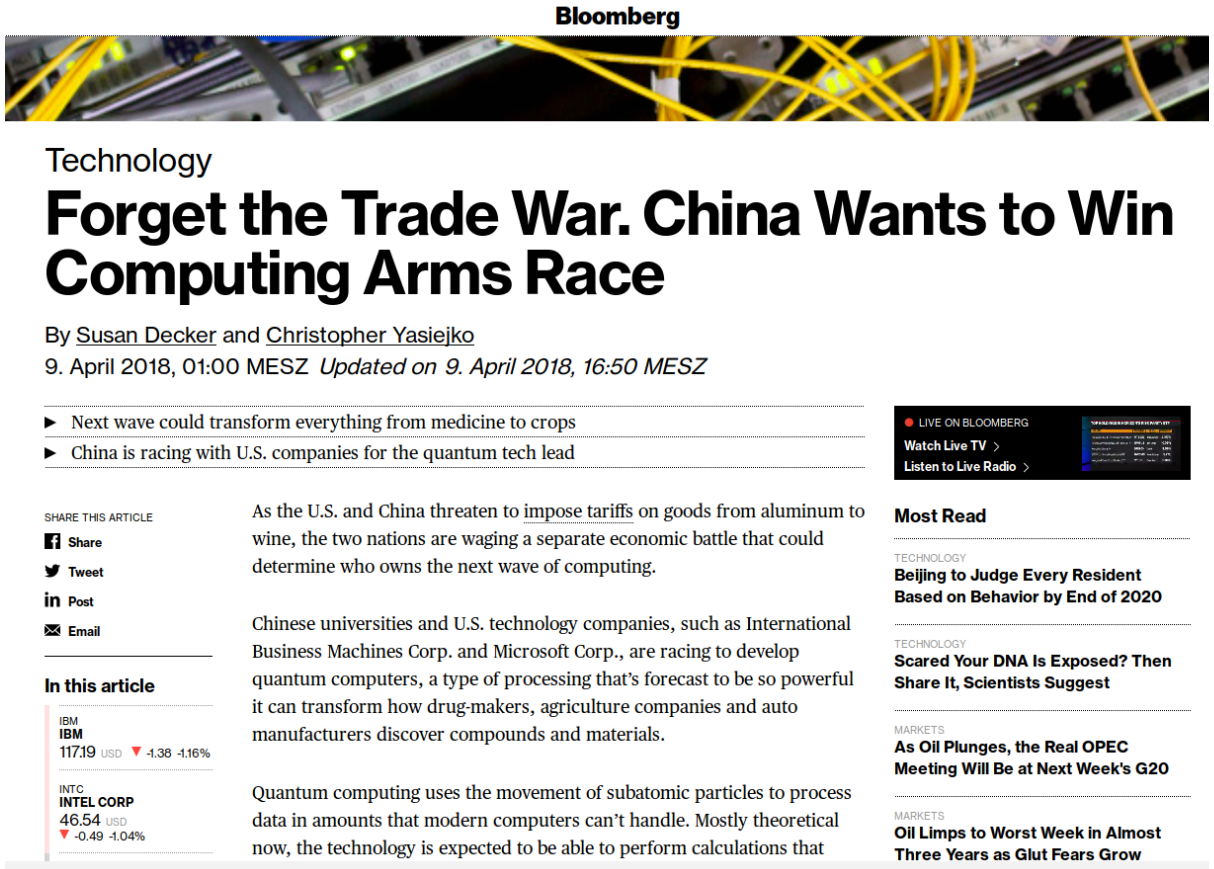
The quantum threat

- **Shor's algorithm** breaks RSA, (EC)DSA, (EC)DH,…
- **Grover's algorithm** asymptotically reduces complexity of brute-force search attacks by a square-root factor.



Why care today

- **EU** launched a one billion Euro project on quantum technologies
- Similar range is spent in **China**
- **US** administration passed a bill on spending $1.275 billion US dollar on quantum computing research
- **Google**, **IBM**, **Microsoft**, **Alibaba**, and others run their own research programs.



Bloomberg

Technology

Forget the Trade War. China Wants to Win Computing Arms Race

By Susan Decker and Christopher Yasiejko

9. April 2018, 01:00 MESZ *Updated on 9. April 2018, 16:50 MESZ*

- Next wave could transform everything from medicine to crops
- China is racing with U.S. companies for the quantum tech lead

SHARE THIS ARTICLE

Share

Tweet

Post

Email

In this article

IBM
IBM
117.19 USD -1.38 -1.16%

INTC
INTEL CORP
46.54 USD
-0.49 -1.04%

As the U.S. and China threaten to impose tariffs on goods from aluminum to wine, the two nations are waging a separate economic battle that could determine who owns the next wave of computing.

Chinese universities and U.S. technology companies, such as International Business Machines Corp. and Microsoft Corp., are racing to develop quantum computers, a type of processing that's forecast to be so powerful it can transform how drug-makers, agriculture companies and auto manufacturers discover compounds and materials.

Quantum computing uses the movement of subatomic particles to process data in amounts that modern computers can't handle. Mostly theoretical now, the technology is expected to be able to perform calculations that

LIVE ON BLOOMBERG
Watch Live TV
Listen to Live Radio

Most Read

TECHNOLOGY
Beijing to Judge Every Resident Based on Behavior by End of 2020

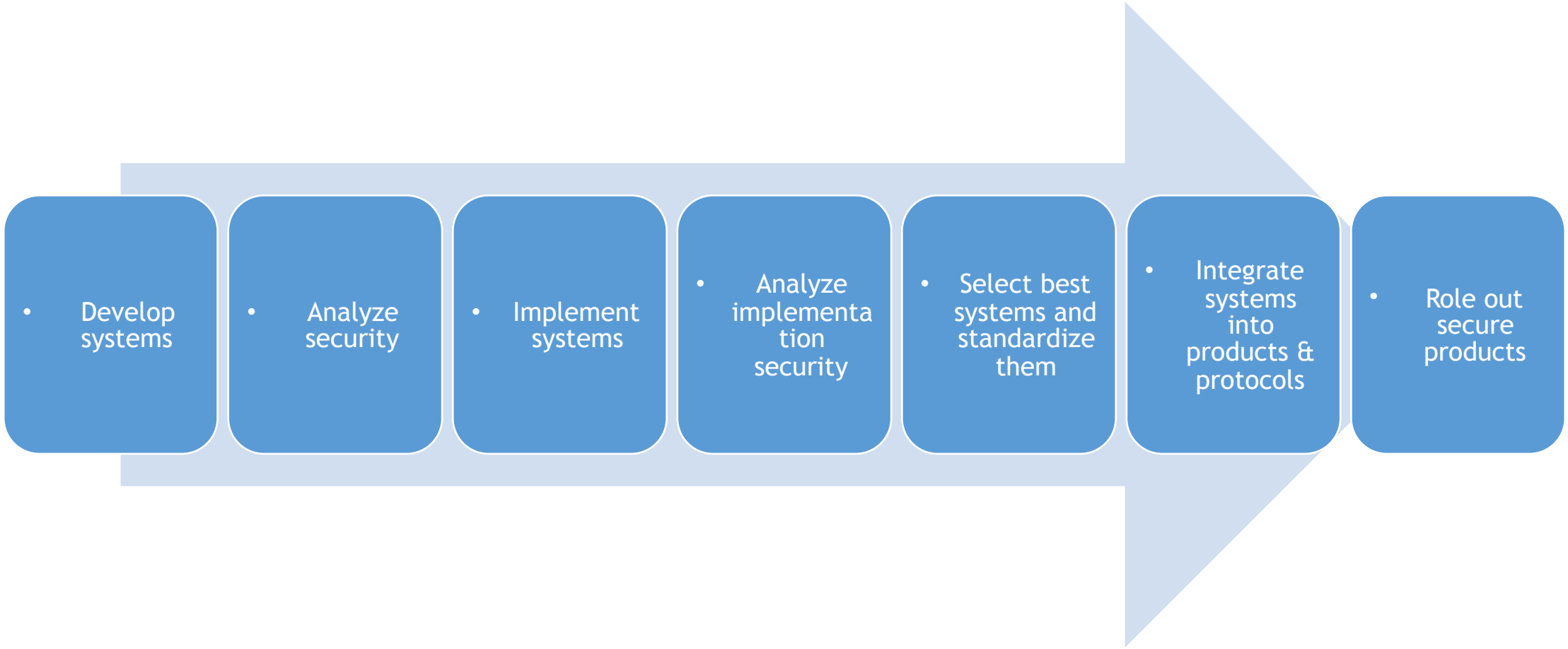
TECHNOLOGY
Scared Your DNA Is Exposed? Then Share It, Scientists Suggest

MARKETS
As Oil Plunges, the Real OPEC Meeting Will Be at Next Week's G20

MARKETS
Oil Limps to Worst Week in Almost Three Years as Glut Fears Grow

It‘s a question of risk assessment

Real world cryptography development



Who would store all encrypted data traffic? That must be expensive!

TU/e

Who would store all encrypted data traffic? That must be expensive!

Who would store all encrypted data traffic? That must be expensive!

Defending Our Nation. Securing The Citizens.

Long-lived systems

- Development time easily 10+ years
- Lifetime easily 10+ years

- At least make sure you got a secure update channel!

Hash-based signatures
[Lam79,Mer89]

- No new hardness assumptions
- Provably (post-quantum) secure if (post-quantum) secure hash function is used
- Basic concept extremely easy
- Stateful

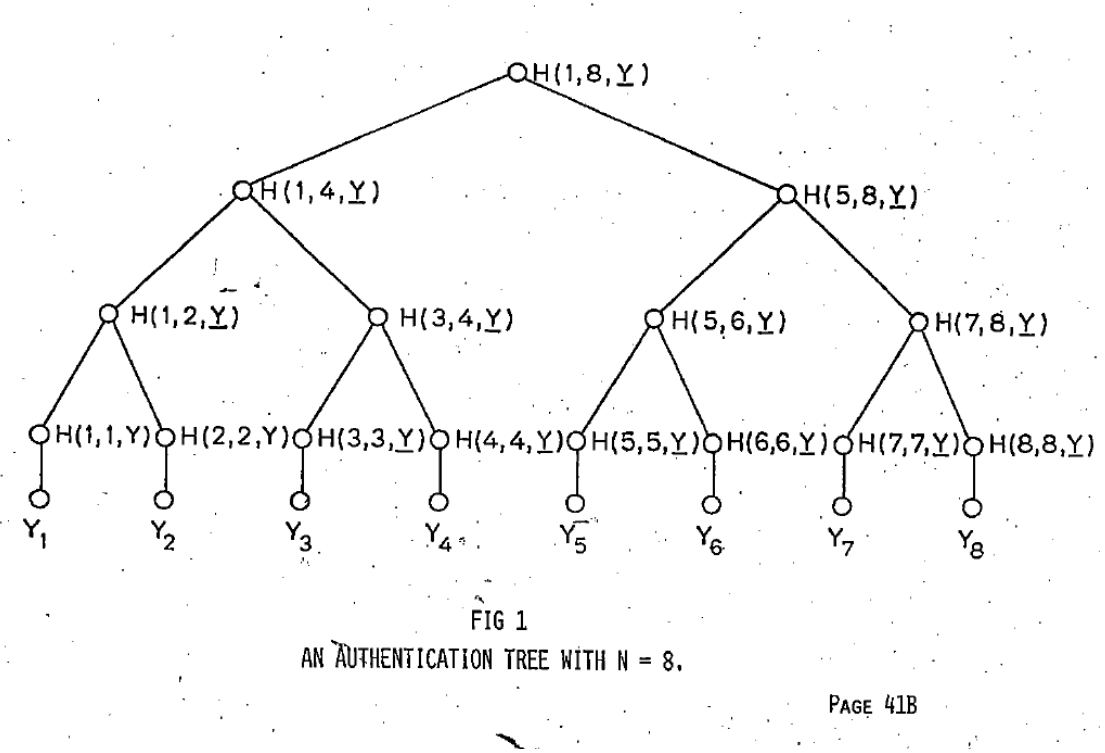
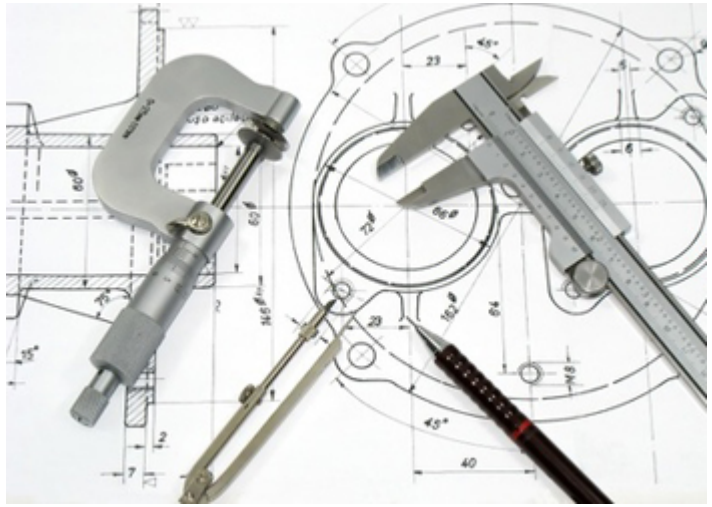


FIG 1
AN AUTHENTICATION TREE WITH N = 8.

PAGE 41B

Basic construction



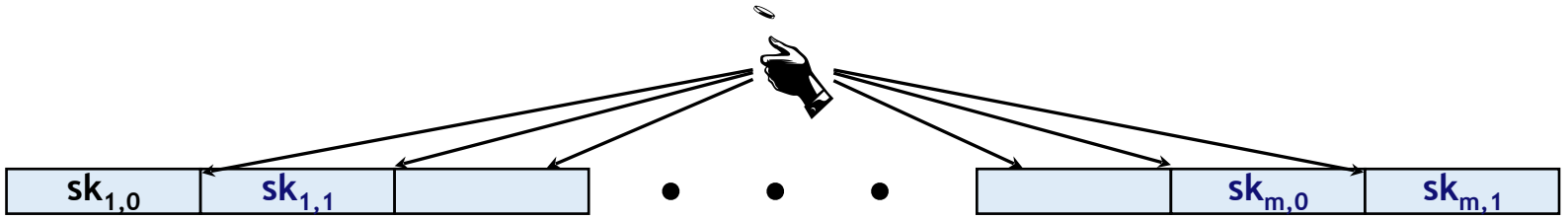
Lamport OTS [Lam79]

Message M = b1,…,bm, OWF H  = n bit

Lamport OTS [Lam79]

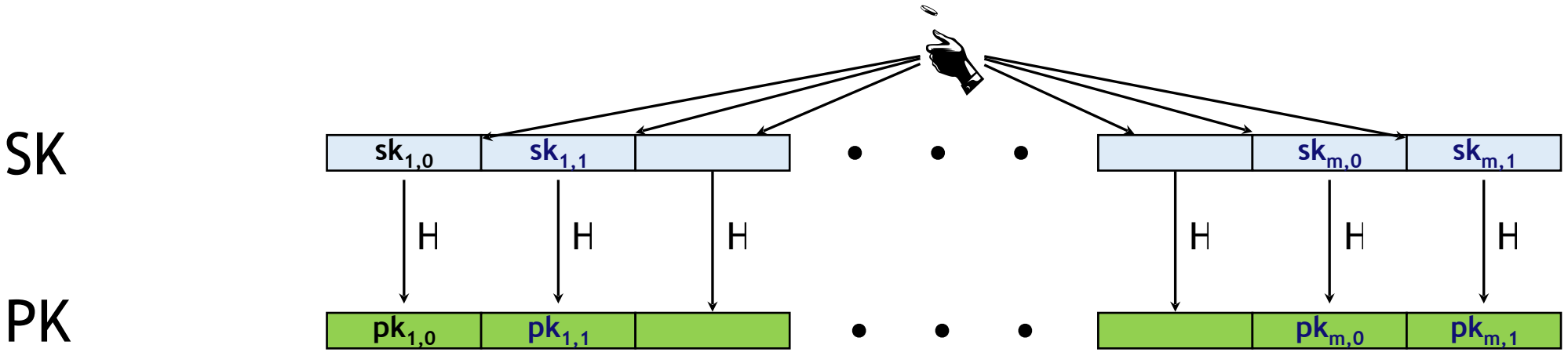
Message M = b1,…,bm, OWF H * = n bit

SK



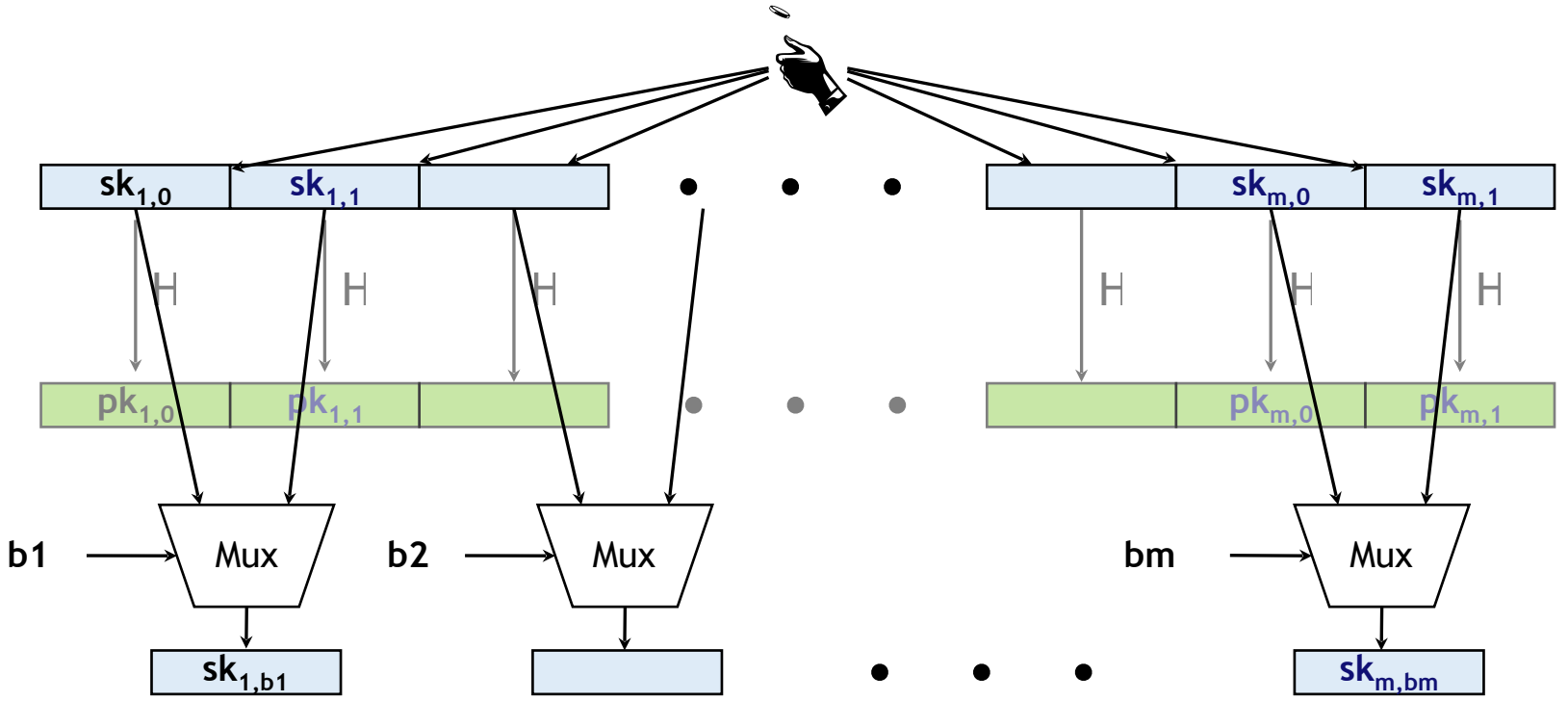
Lamport OTS [Lam79]

Message M = b1,…,bm, OWF H * = n bit



Lamport OTS [Lam79]

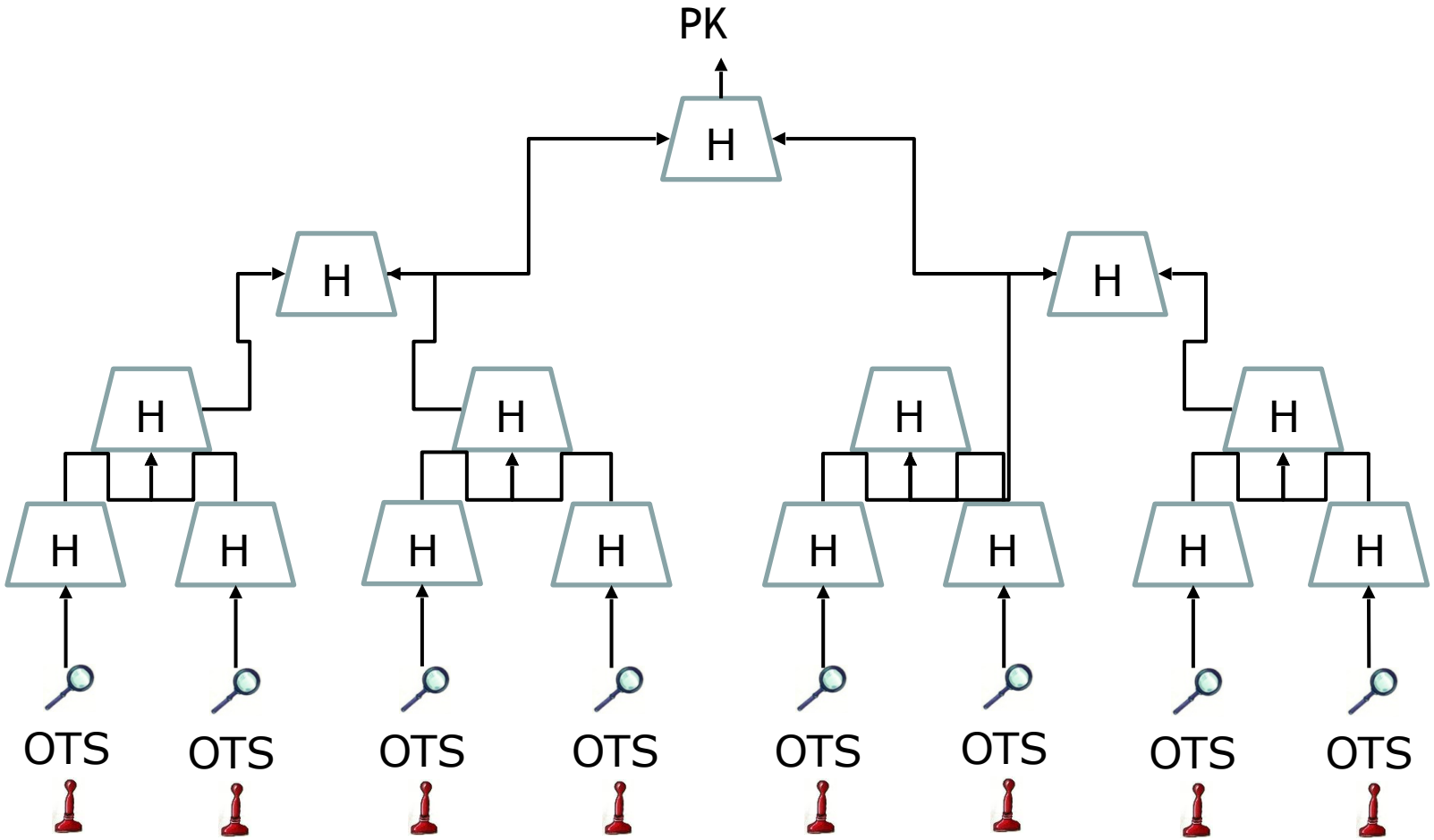
Message M = b1,…,bm, OWF H * = n bit



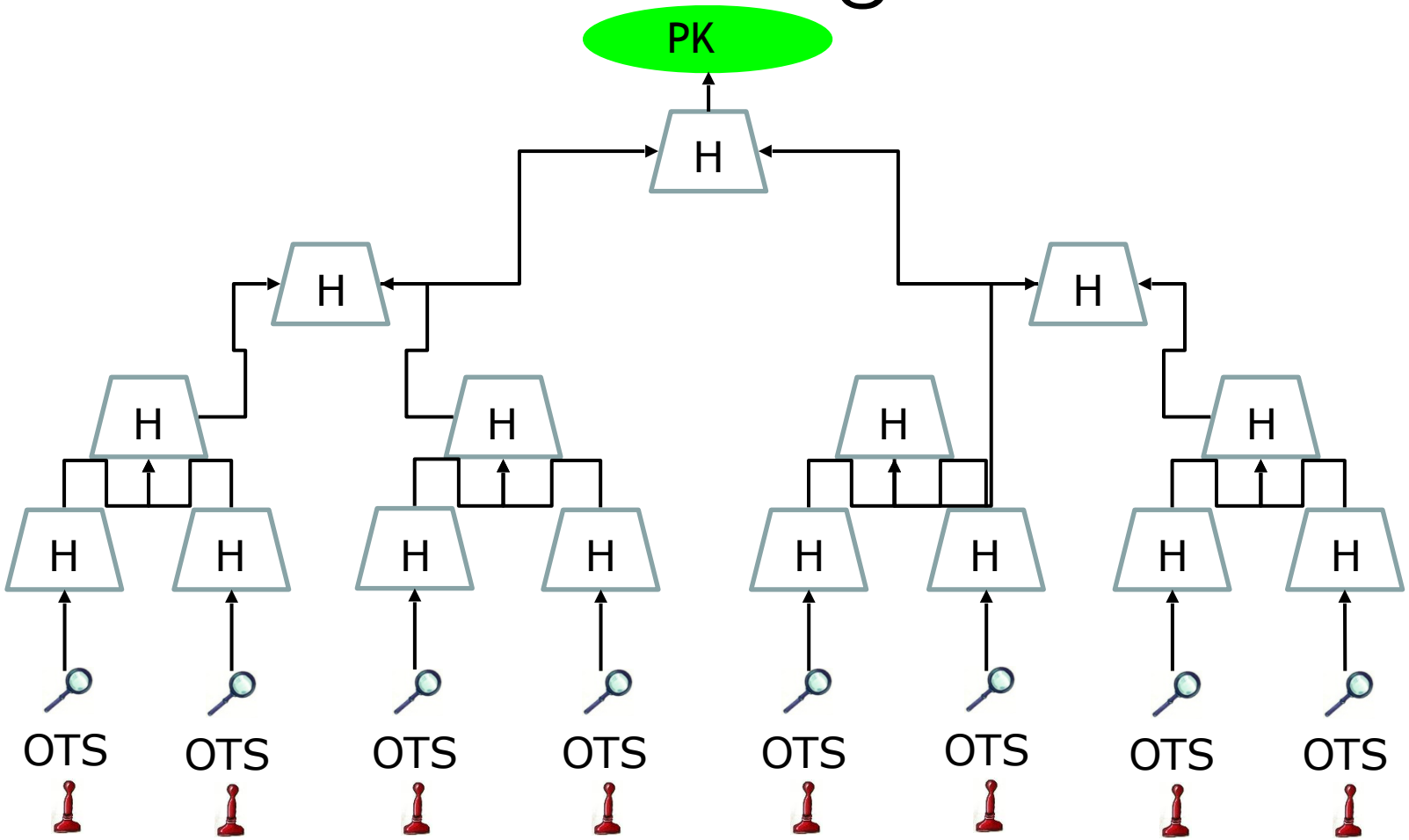
Merkle's Hash-based Signatures

OTS

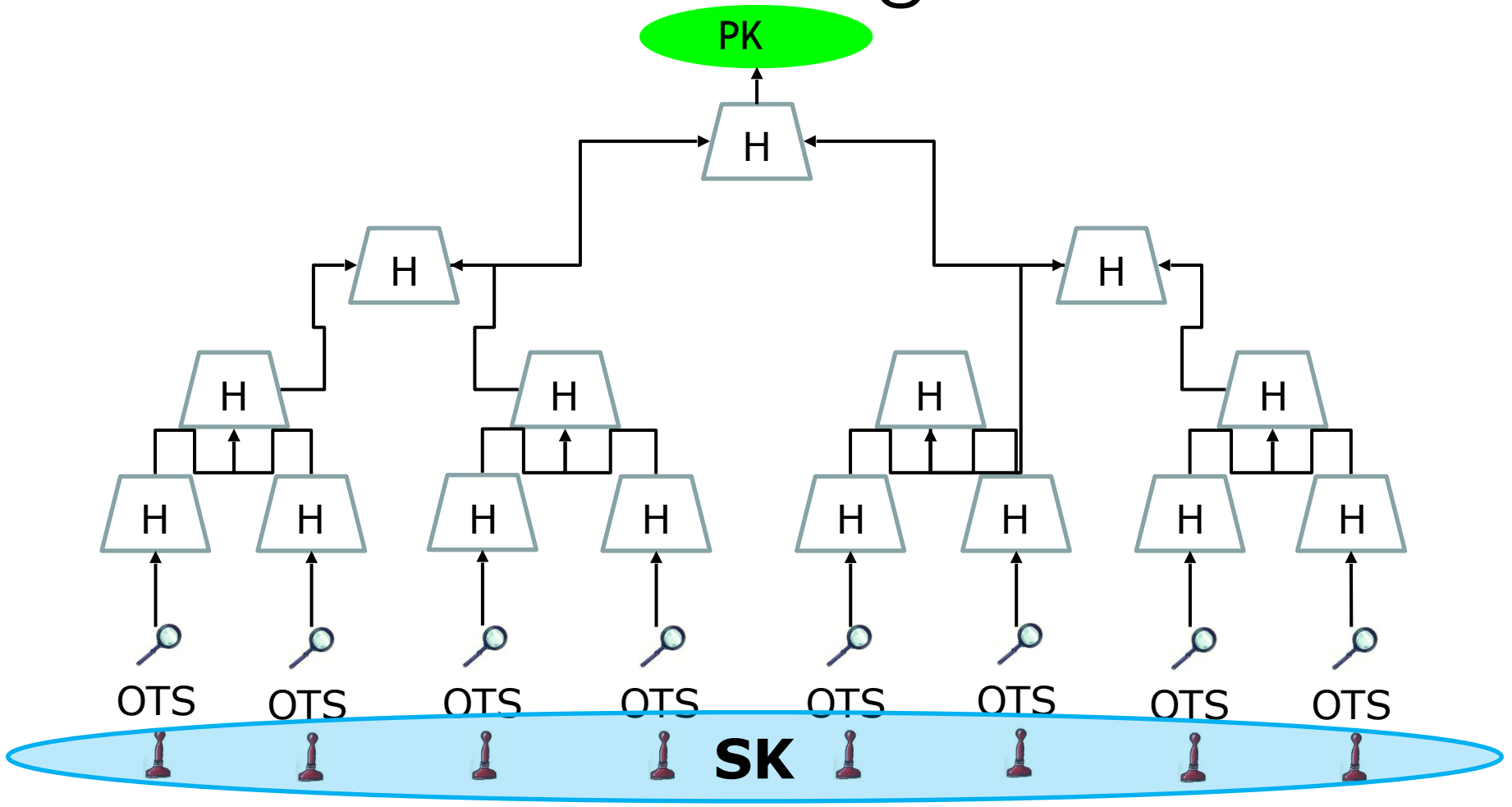
Merkle's Hash-based Signatures



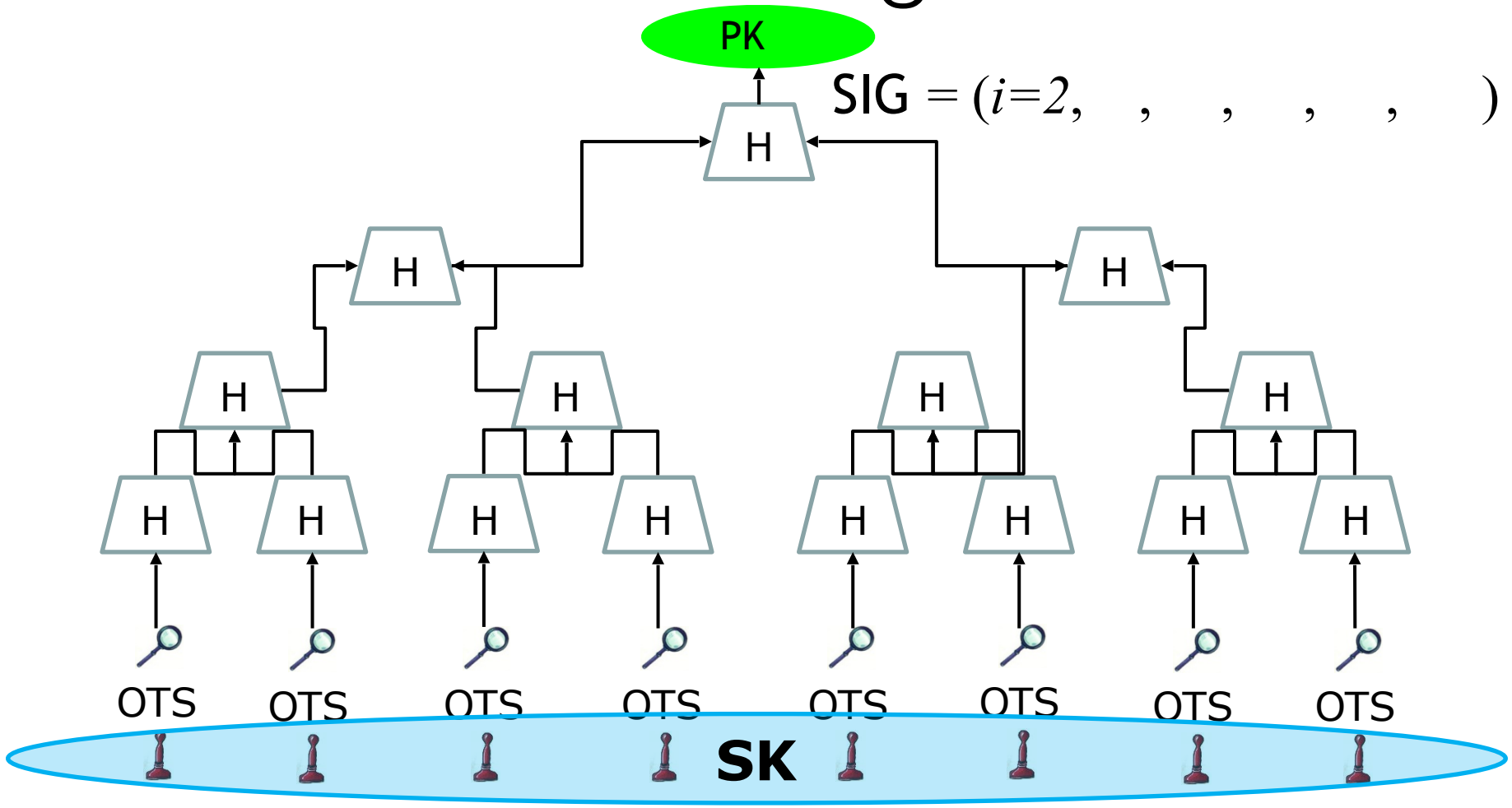
Merkle's Hash-based Signatures



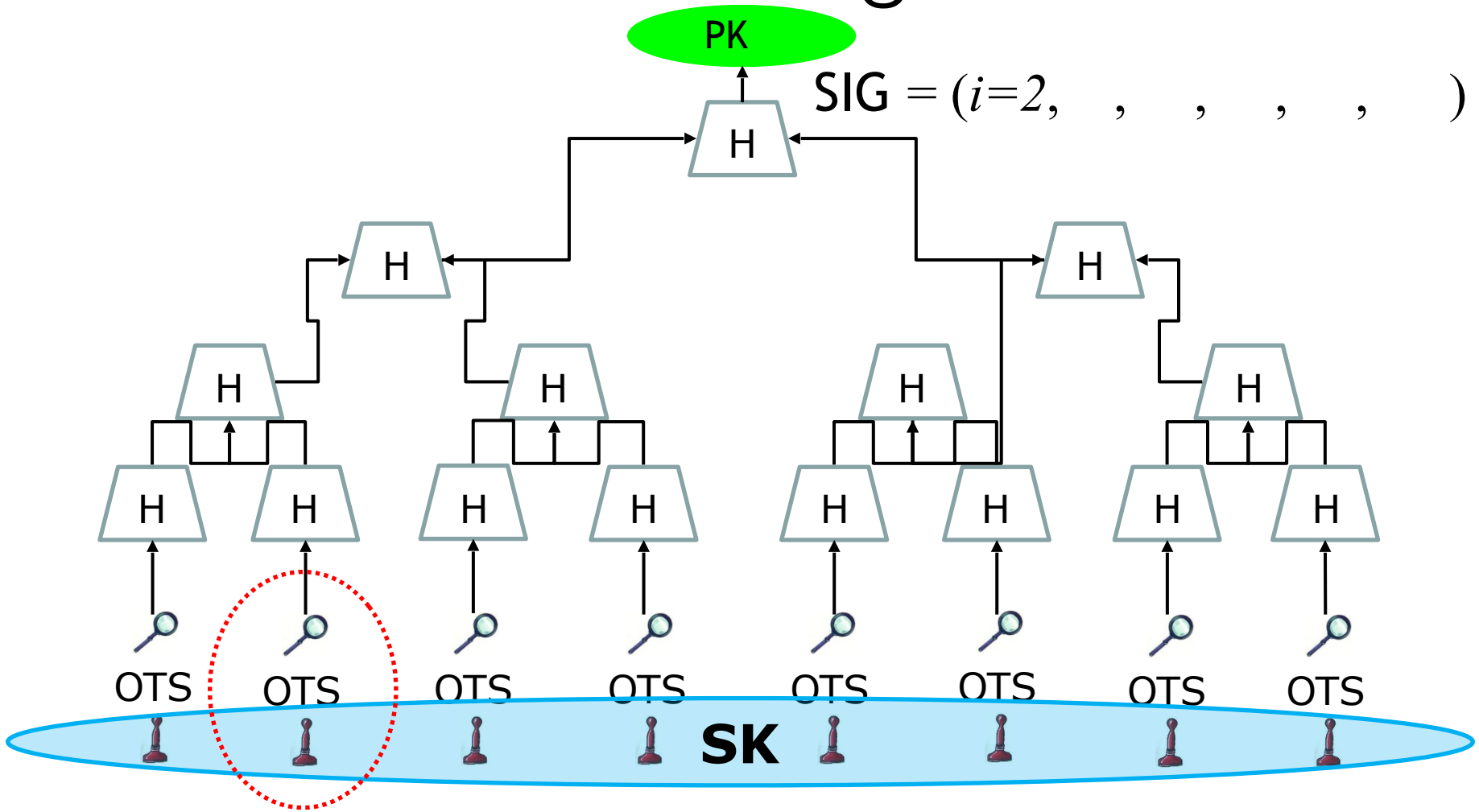
Merkle's Hash-based Signatures



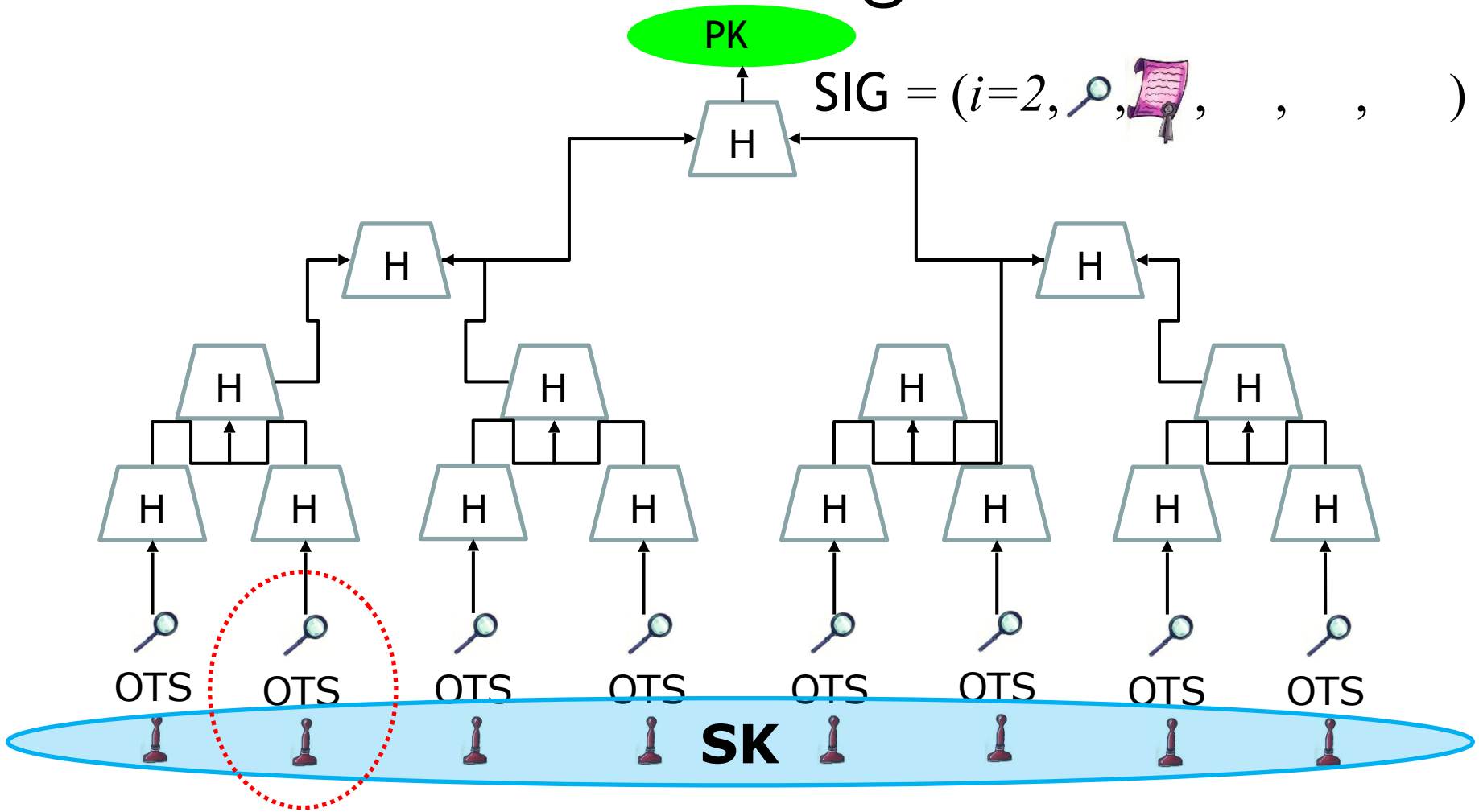
Merkle's Hash-based Signatures



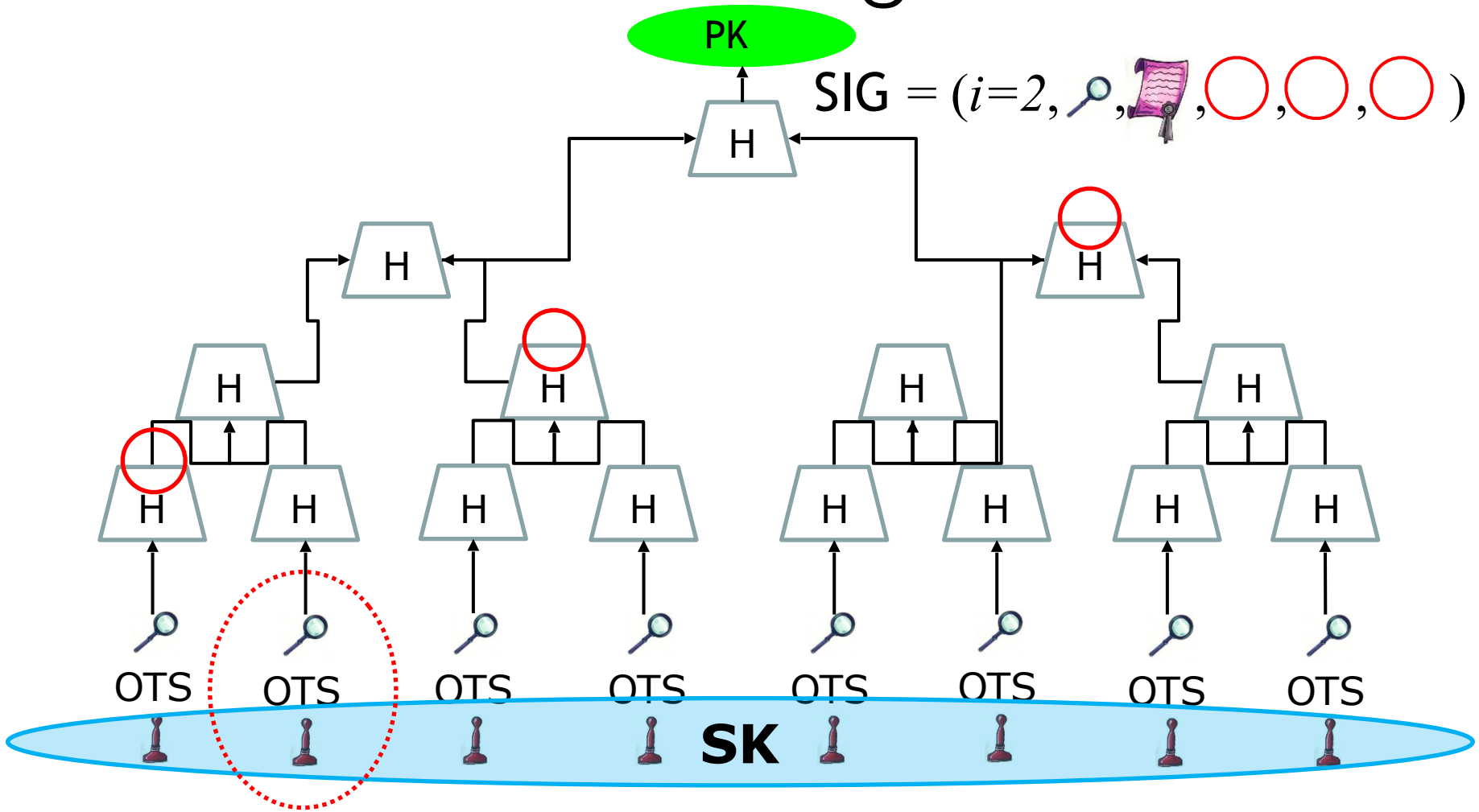
Merkle's Hash-based Signatures



Merkle's Hash-based Signatures



Merkle's Hash-based Signatures



Winternitz-OTS

Lamport-OTS in MSS

SIG = (i=2,  , , ◯, ◯, ◯)

Lamport-OTS in MSS

SIG = (i=2,  , , ◯, ◯, ◯)

Verification:

1. Verify 
2. Verify authenticity c 

We can do better!

WOTS in MSS

SIG = (i=2, )

WOTS in MSS

SIG = (i=2,  , ○, ○, ○)

WOTS in MSS

SIG = $(i=2, \ldots, \bigcirc, \bigcirc, \bigcirc)$

Verification:

1. Compute
2. Verify authenticity

Steps 1 + 2 together verif

Function chains

Hash function

Parameter

Chain:

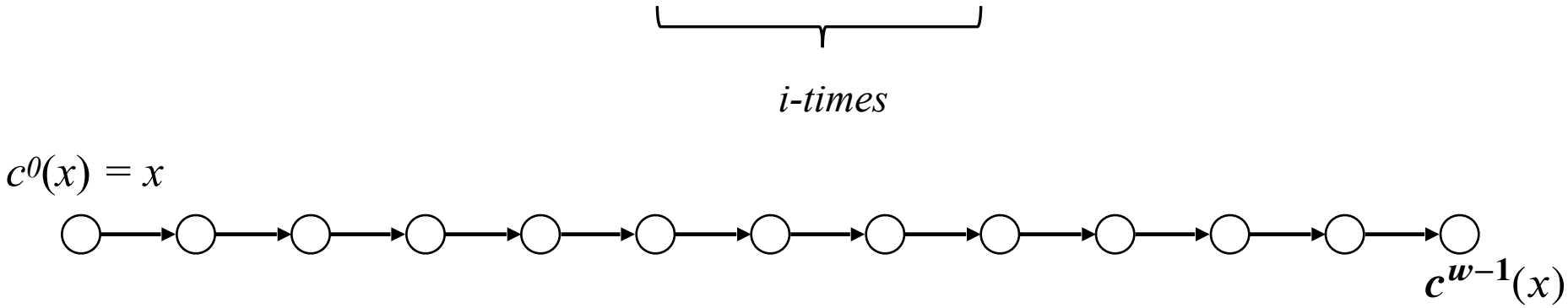
i-times

Function chains

Hash function

Parameter

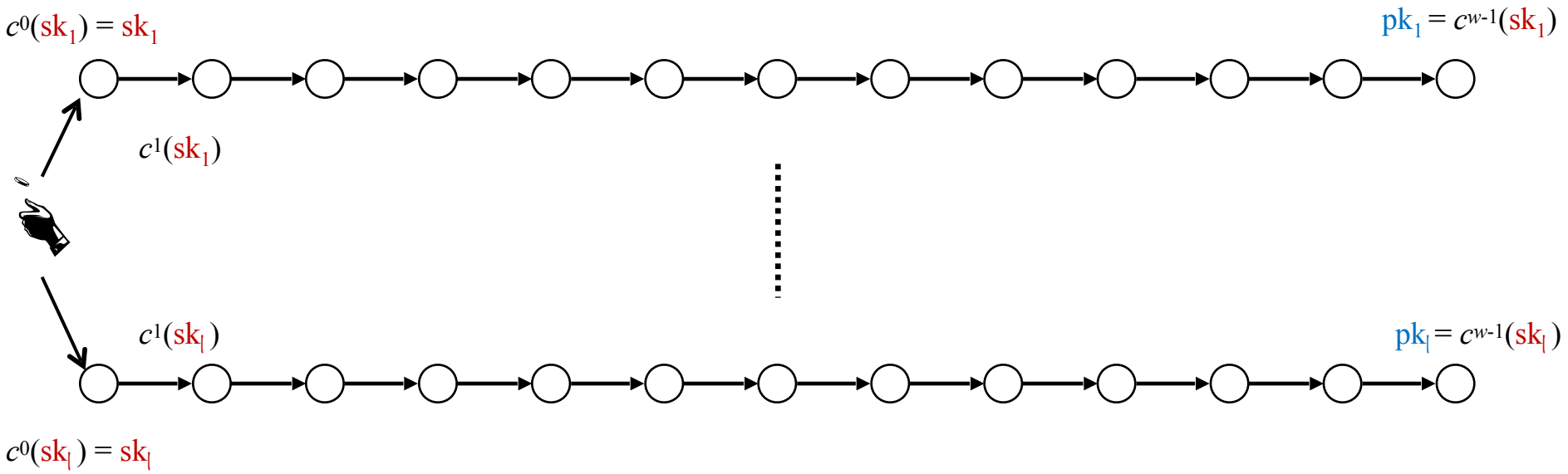
Chain:



WOTS

Winternitz parameter w (usually a power of 2), security parameter n, message length m, hash function

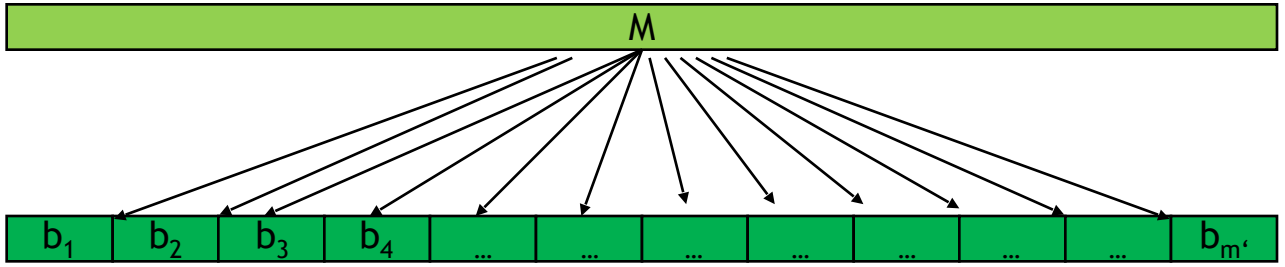
Key Generation: Compute , sample



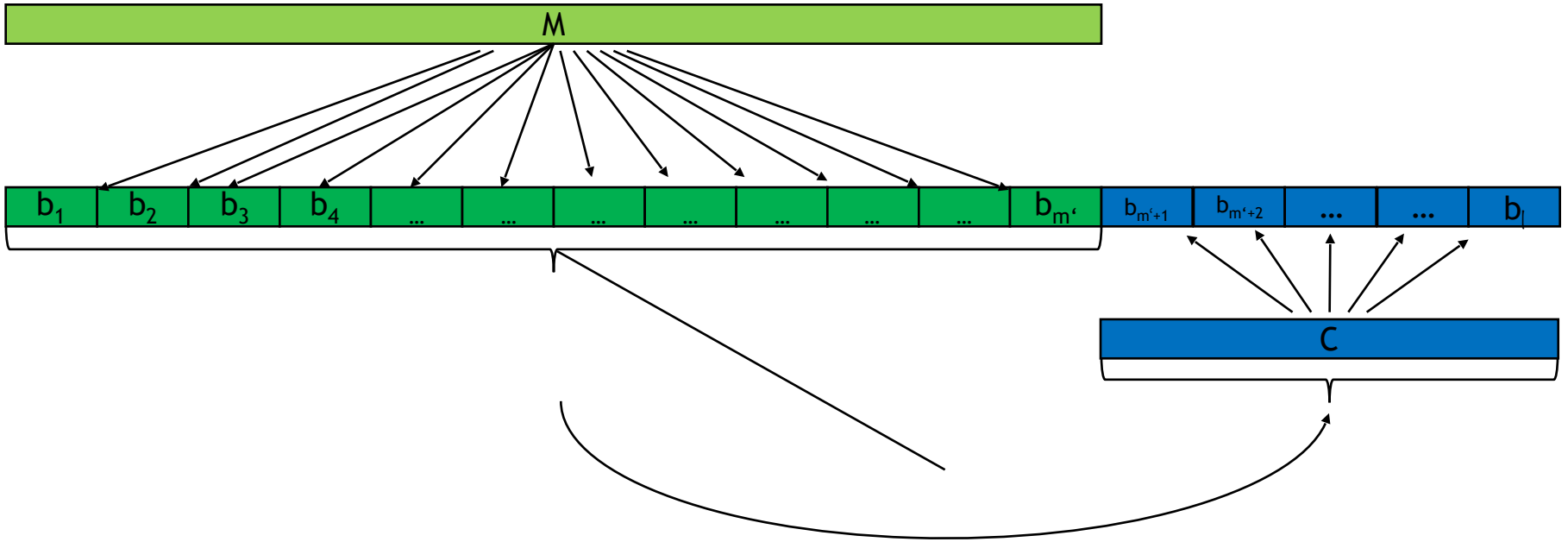
WOTS Signature generation

WOTS Signature generation

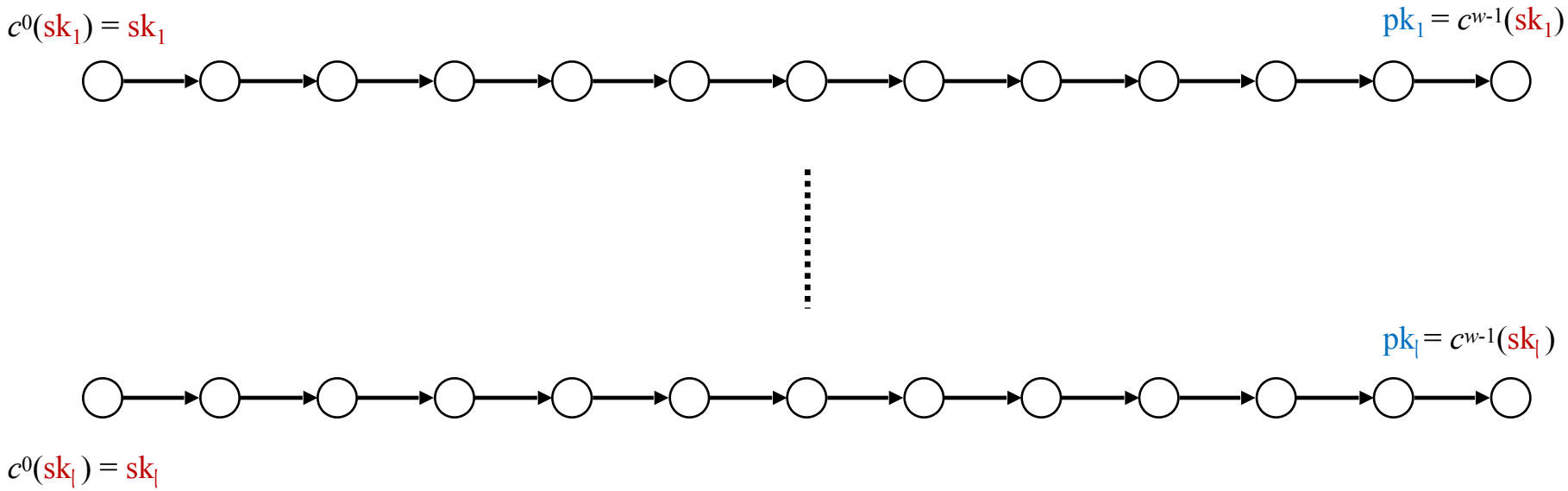
WOTS Signature generation



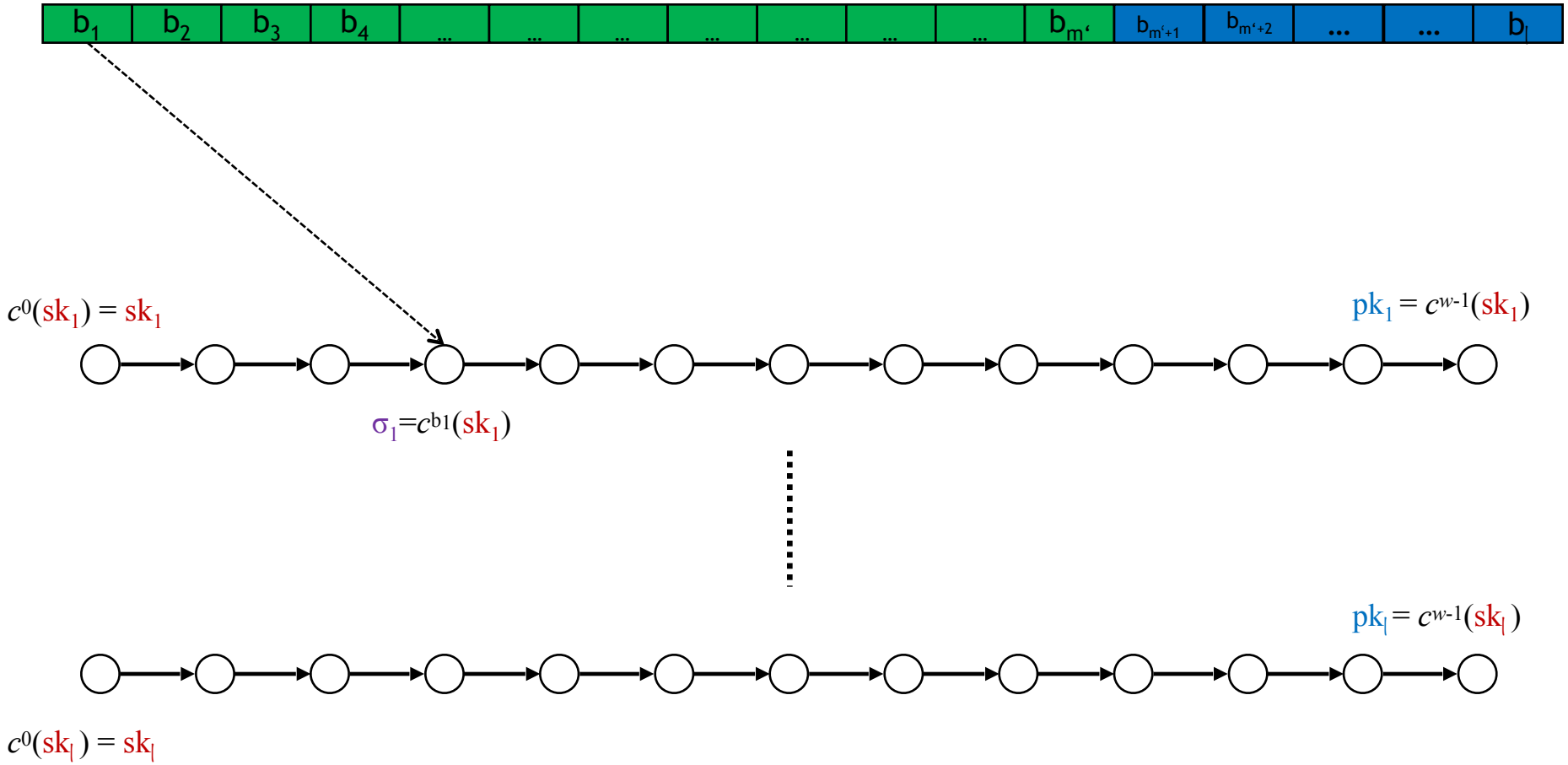
WOTS Signature generation



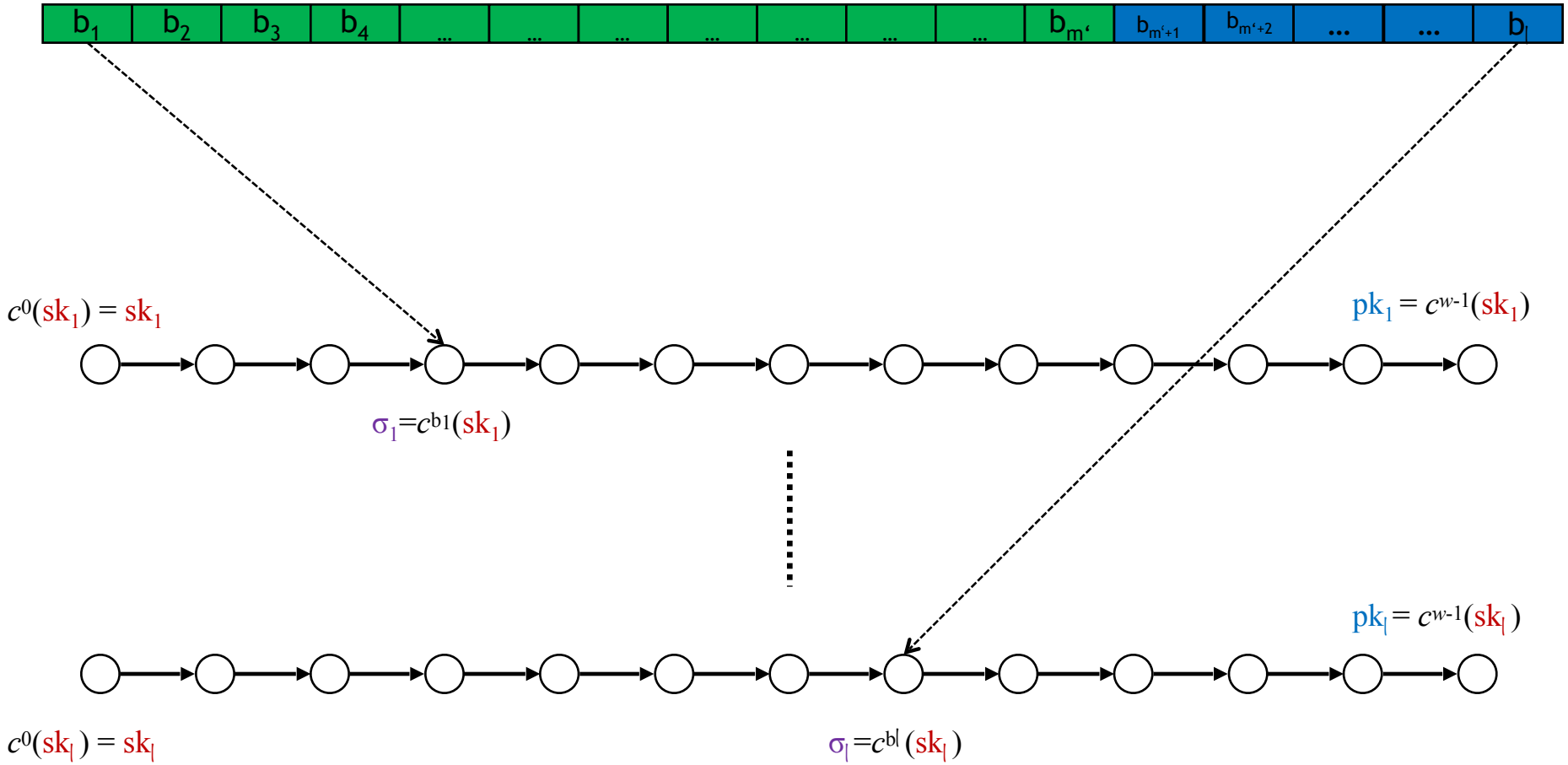
WOTS Signature generation



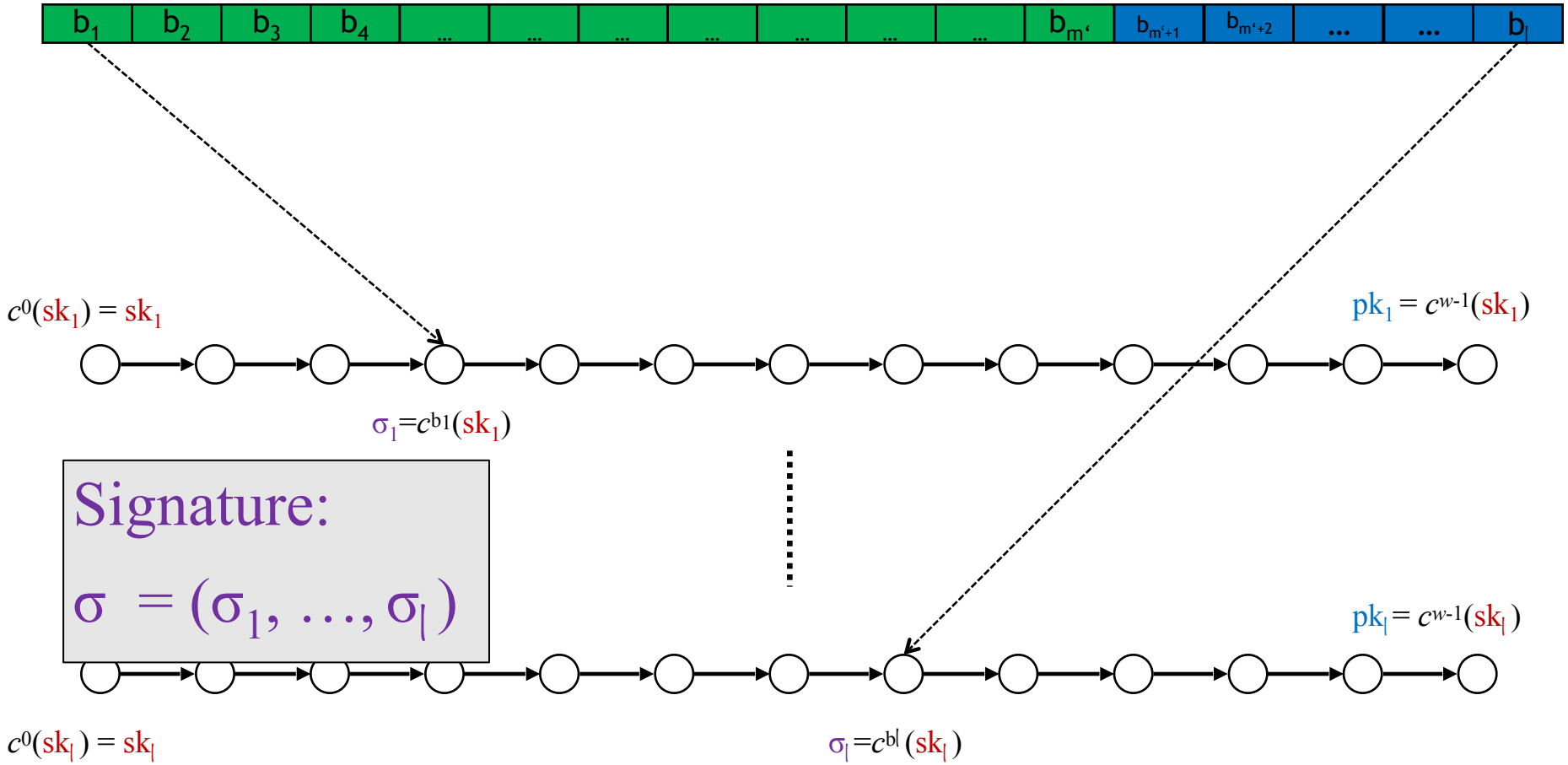
WOTS Signature generation



WOTS Signature generation

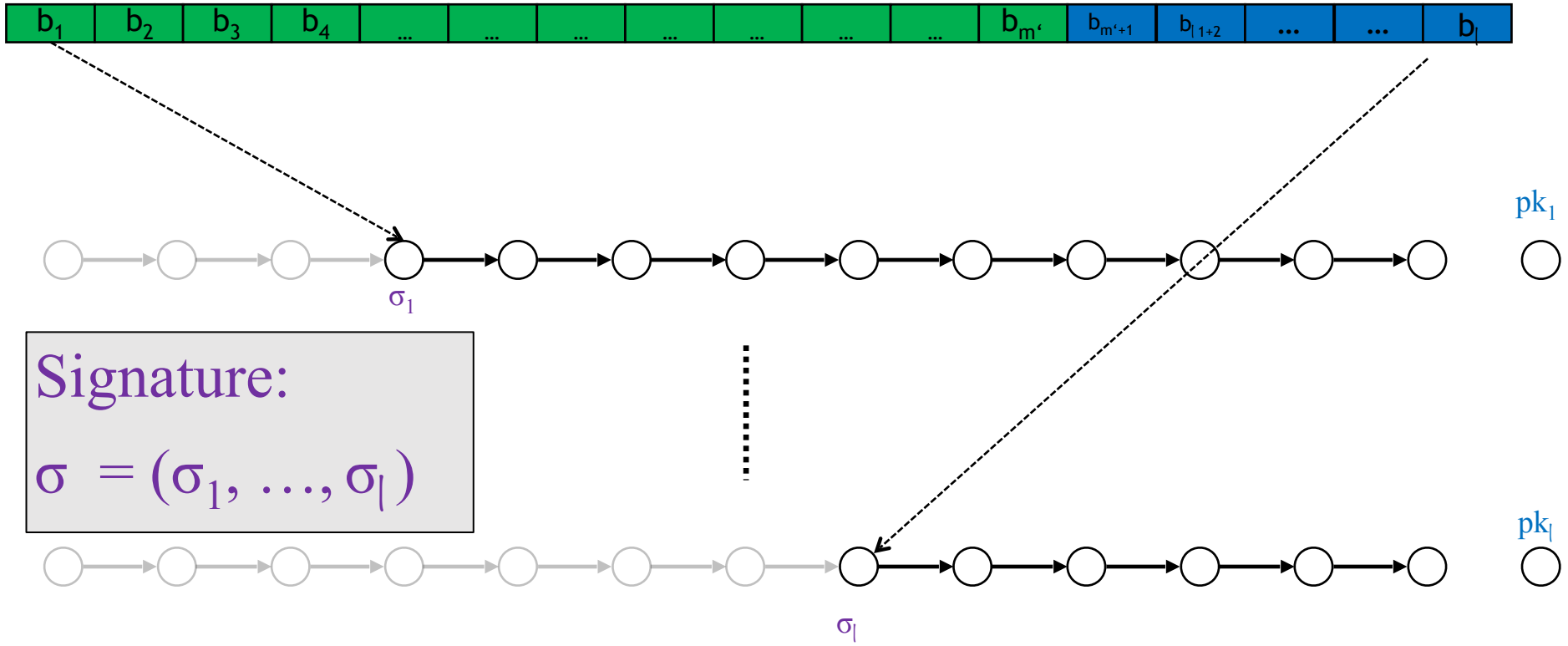


WOTS Signature generation



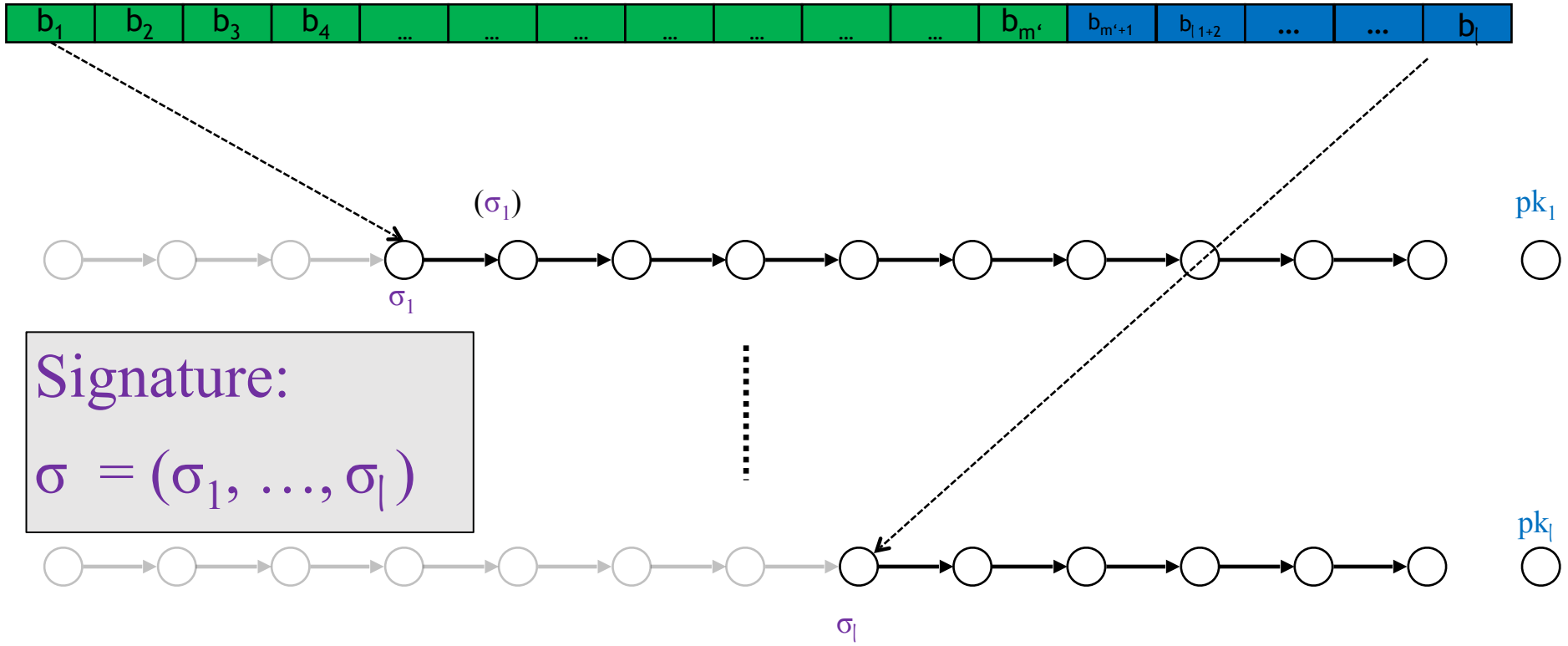
WOTS Signature Verification

Verifier knows: M, w



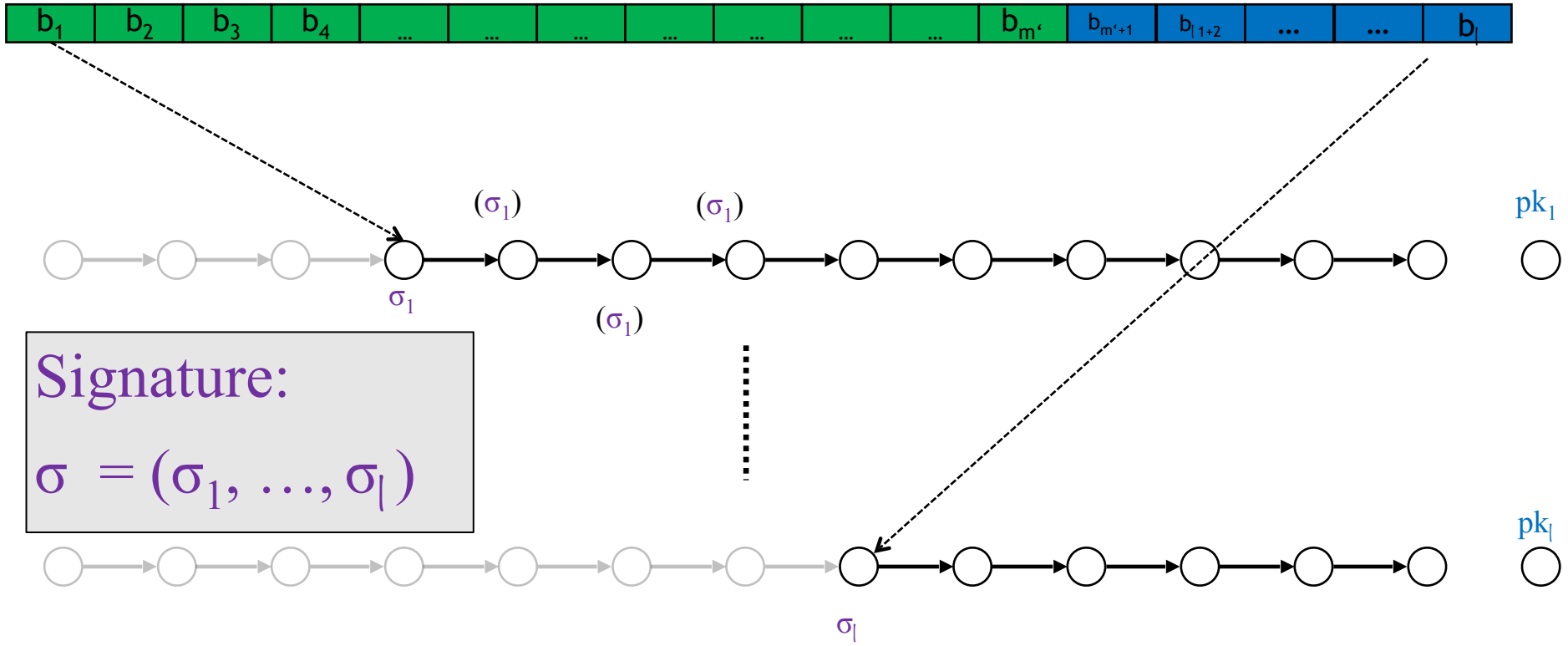
WOTS Signature Verification

Verifier knows: M, w



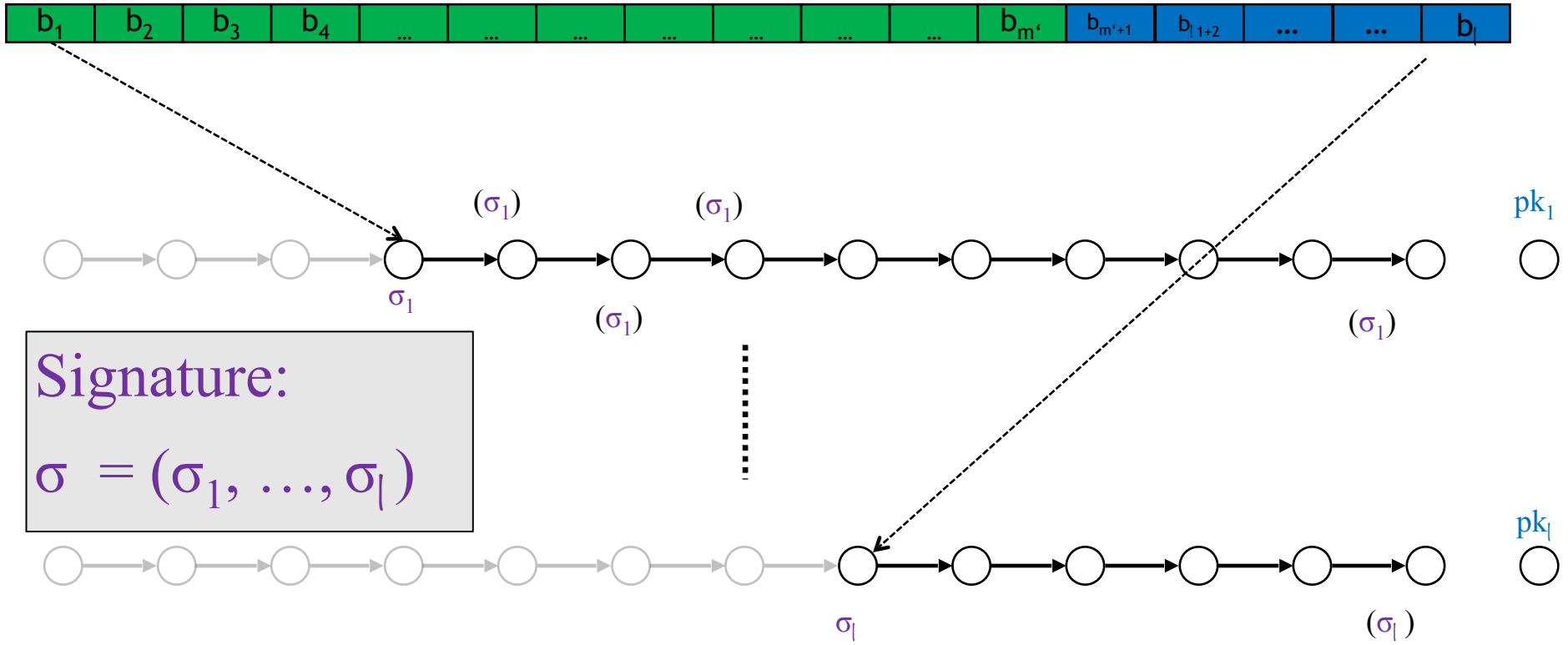
WOTS Signature Verification

Verifier knows: M, w



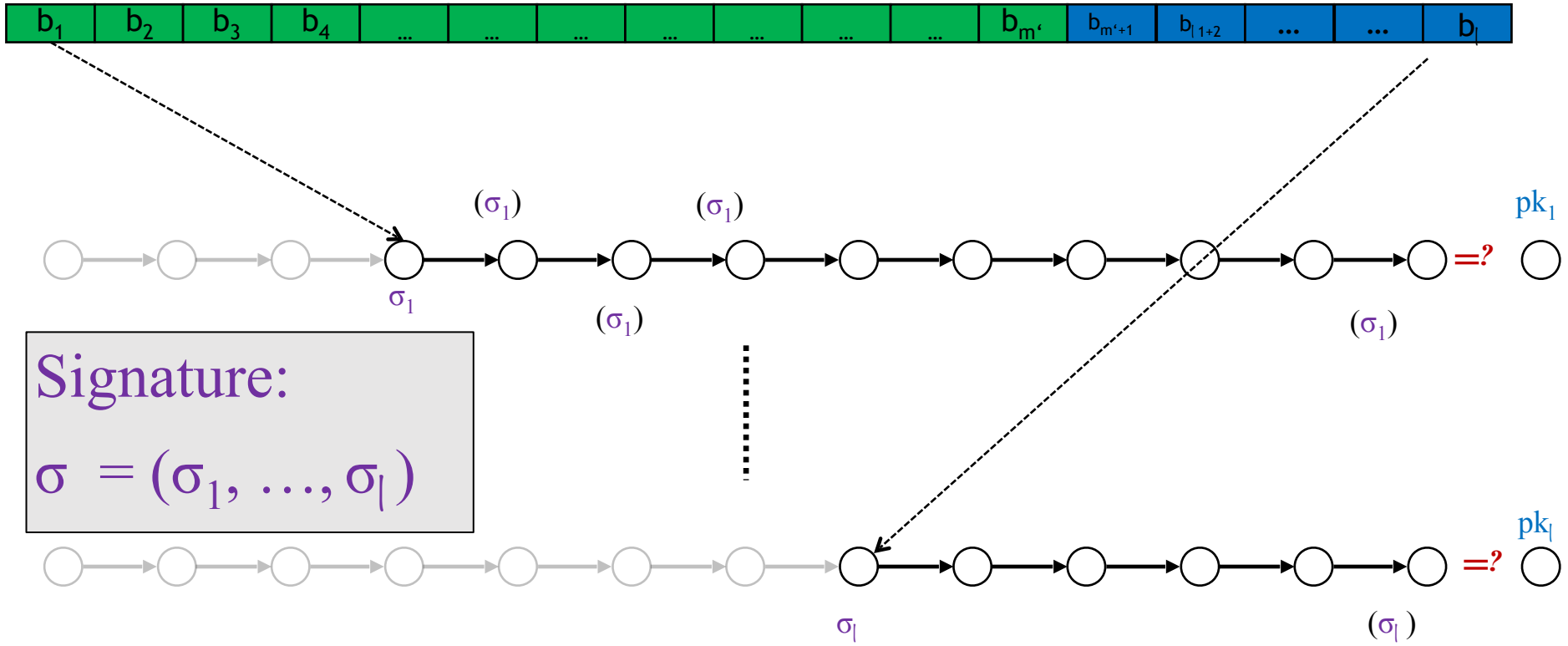
WOTS Signature Verification

Verifier knows: M, w



WOTS Signature Verification

Verifier knows: M, w



Multi-Tree MSS

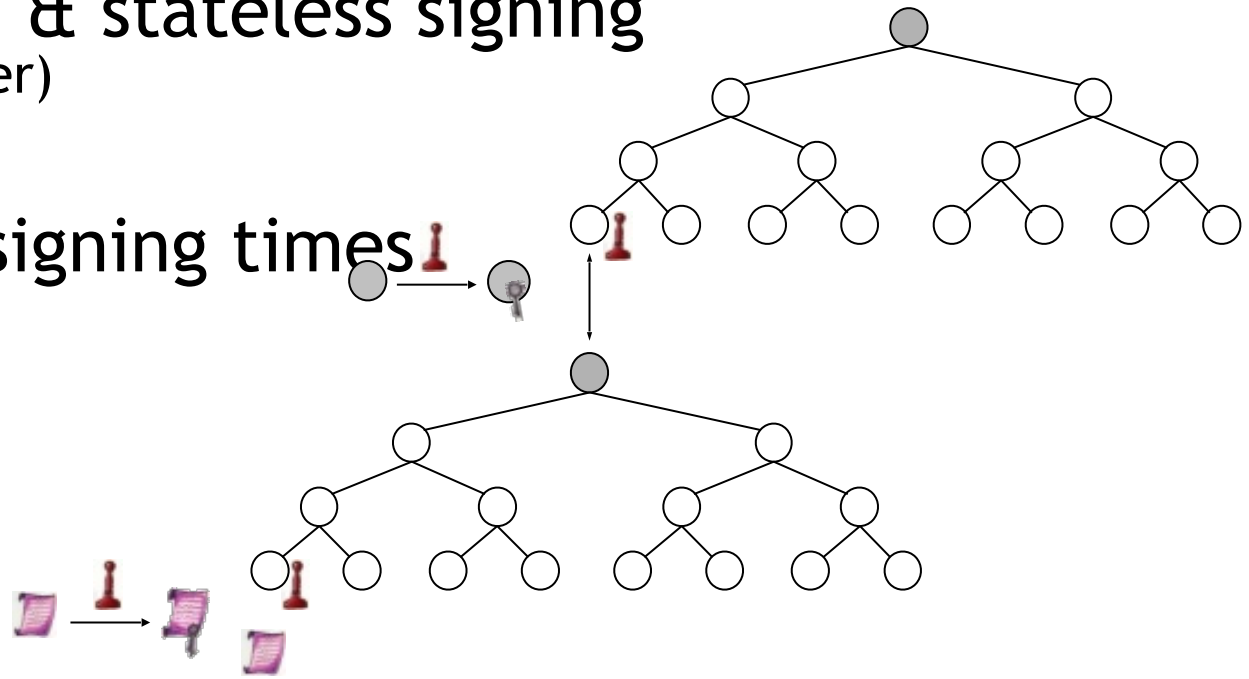
Multi-Tree MSS / Hypertree

Uses multiple layers of trees to reduce key generation time

-> Key state generation & stateless signing
(= Building one tree on each layer)

-> Worst-case stateful signing times

-> Increases signature size by d-1 one-time signatures



TU/e

SPHINCS

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

Goldreich's approach [Gol04]:

Security parameter

Use binary tree as in Merkle, but...

- ...for security
  - pick index i at random;
  - requires huge tree to avoid index collisions (e.g., height).
- ...for efficiency:
  - use binary certification tree of OTS key pairs (= Hypertree with
  - all OTS secret keys are generated pseudorandomly.

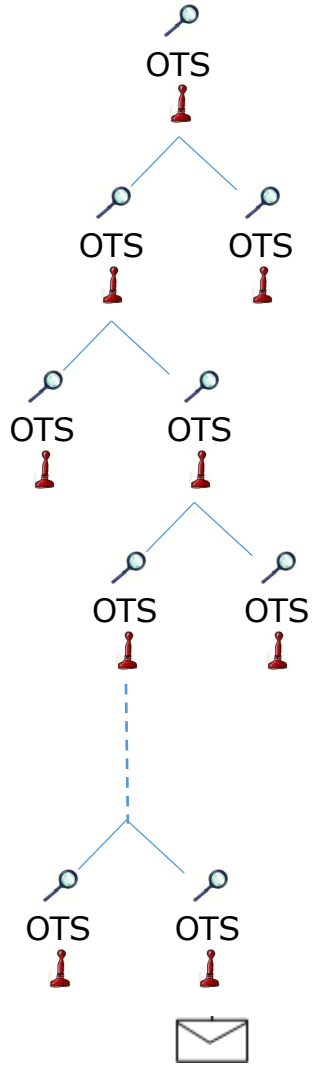
Stateless hash-based signatures
[NY89,Gol87,Gol04]

Goldreich's approach [Gol04]:

Security parameter

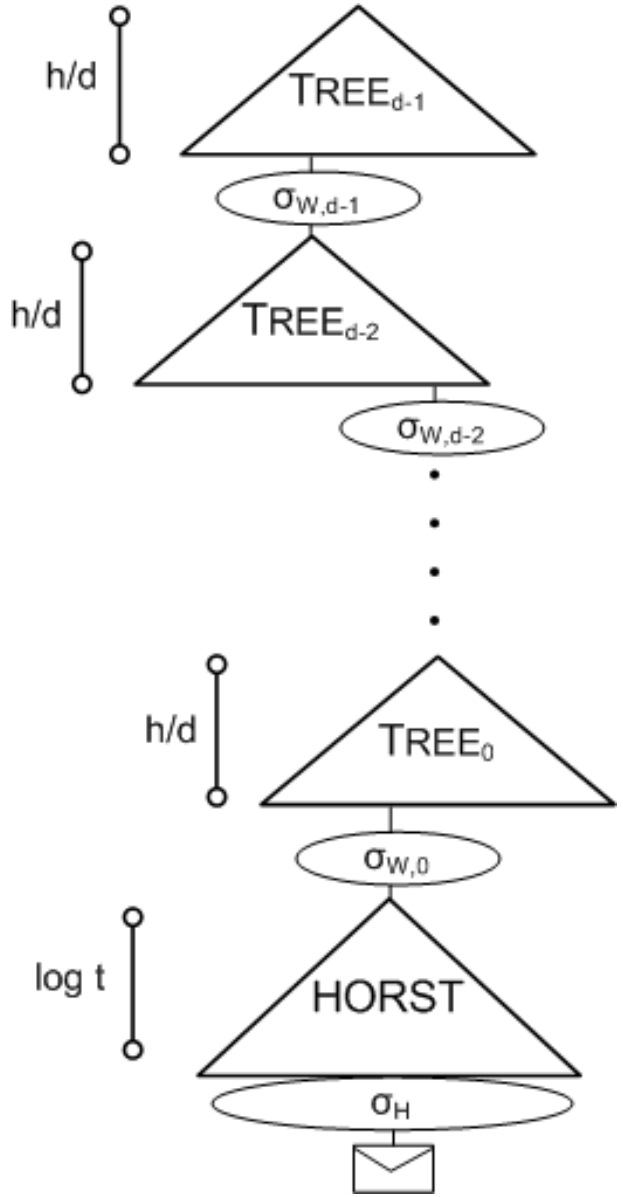
Use binary tree as in Merkle, but...

- …for security
  - pick index i at random;
  - requires huge tree to avoid index collisions (e.g., height).
- …for efficiency:
  - use binary certification tree of OTS key pairs (= Hypertree with
  - all OTS secret keys are generated pseudorandomly.



SPHINCS [BHH+15]

- Select index pseudo-randomly
- Use a few-time signature key-pair on leaves to sign messages
  - Few index collisions allowed
  - Allows to reduce tree height
- Use hypertree: Use d << h.



Security arguments

Requirements

Reductions should lead to

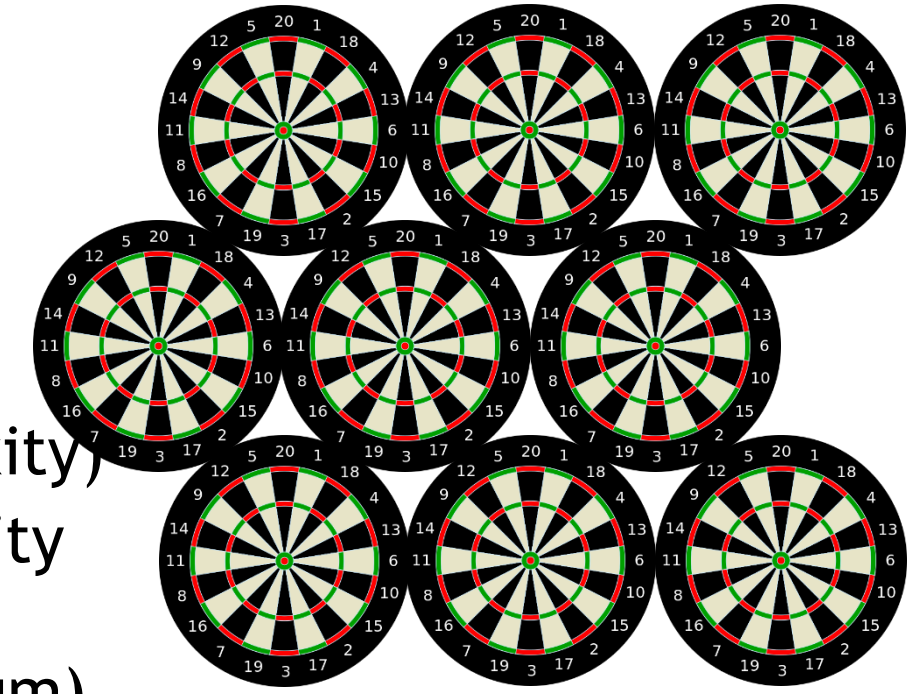
- collision-resilience,
- multi-target attack protection,
- tight security reductions,

and allow for

- easy verification, and
- maintainability.

Multi-target attacks

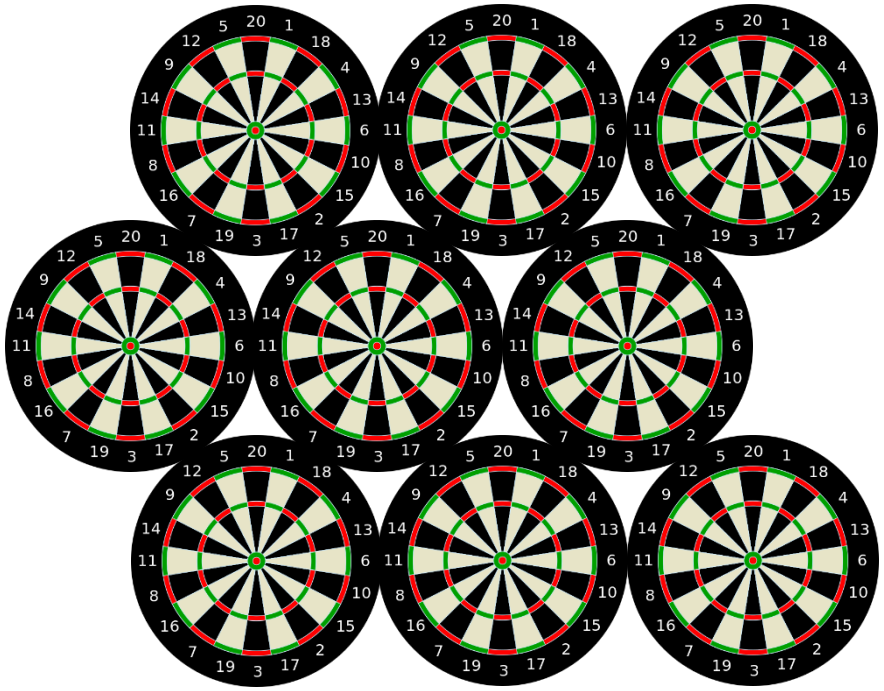
- WOTS & Lamport need hash function to be one-way
- Hypertree of total height 60 with WOTS (w=16) leads images.
- Inverting one of them allows existential forgery (at least massively reduces complexity)
- q-query brute-force succeeds with probability conventional and quantum
- We loose 66 bits of security! (33 bits quantum)

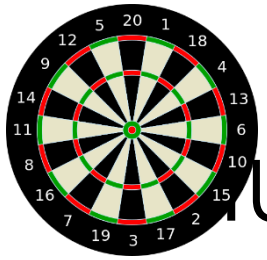
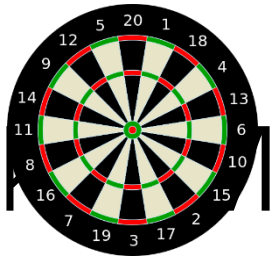
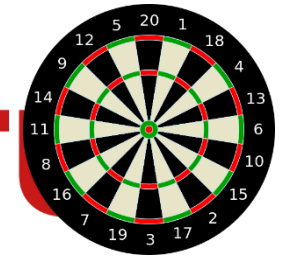


Multi-target attacks: Mitigation

- Mitigation: Separate targets [HRS16]
- Common approach:
  - In addition to hash function description and „input“ take
    - Hash „Address“ (uniqueness in key pair)
    - Hash „key“ used for all hashes of one key pair (uniqueness among key pairs)
- Tweakable Hash Function:

P: Public parameters (one per key pair)
T: Tweak (one per hash call)
M: Message
MD: Message Digest

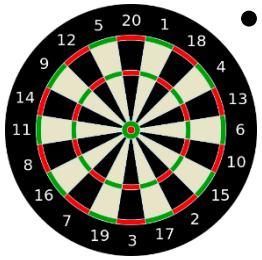
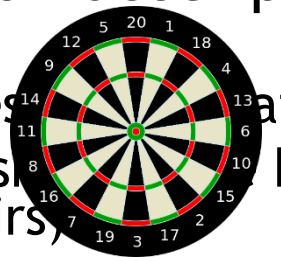
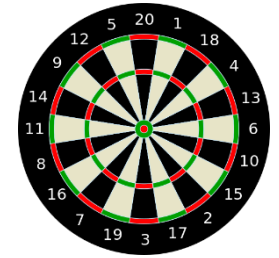
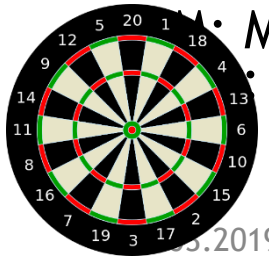


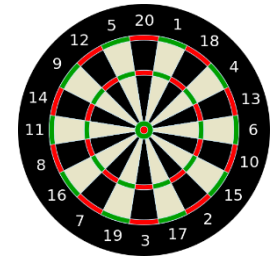




Multi-target attack: Mitigation

- Mitigation: Separate targets [HRS16]
- Common approach:
  - In addition to hash function description and „input“ take
    - Hash „Address“ (uniqueness among key pair)
    - Hash „key“ used for all hash calls for one key pair (uniqueness among key pairs)
- Tweakable Hash Function:

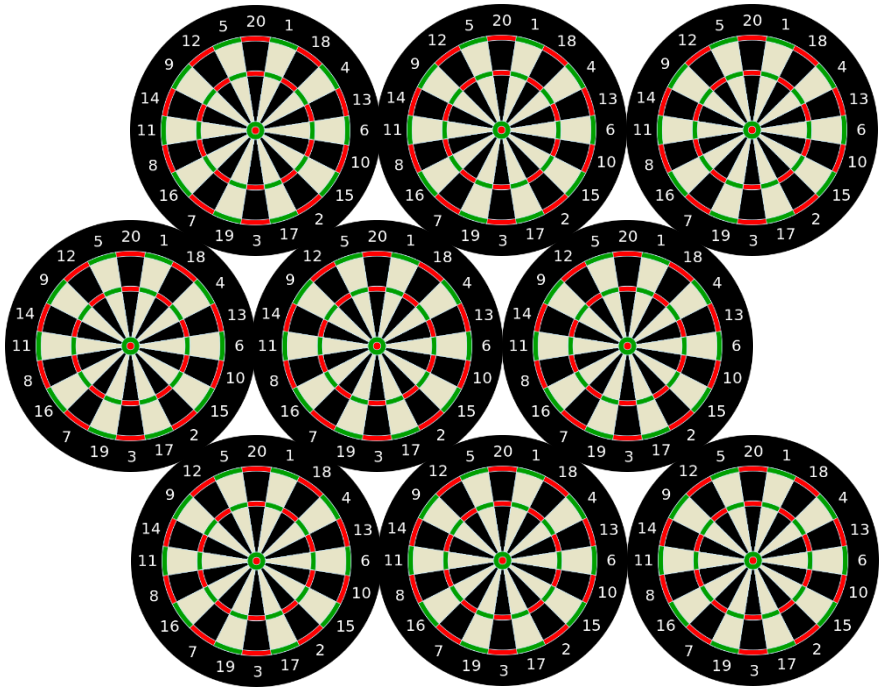
P: Public parameters (one per key pair)
T: Tweak (one per hash call)
M: Message
: Message Digest



Multi-target attacks: Mitigation

- Mitigation: Separate targets [HRS16]
- Common approach:
  - In addition to hash function description and „input“ take
    - Hash „Address“ (uniqueness in key pair)
    - Hash „key“ used for all hashes of one key pair (uniqueness among key pairs)



New intermediate abstraction: Tweakable Hash Function

TU/e

- Tweakable Hash Function:

P: Public parameters (one per key pair)
T: Tweak (one per hash call)
M: Message
MD: Message Digest

- Security in two steps:

1. Prove security of SPHINCS(+), XMSS, LMS,..... using tweakable hash functions
2. Prove tweakable hash function security

So what properties do we need?

Single-function multi-target collision resistance for distinct tweaks

TU/e

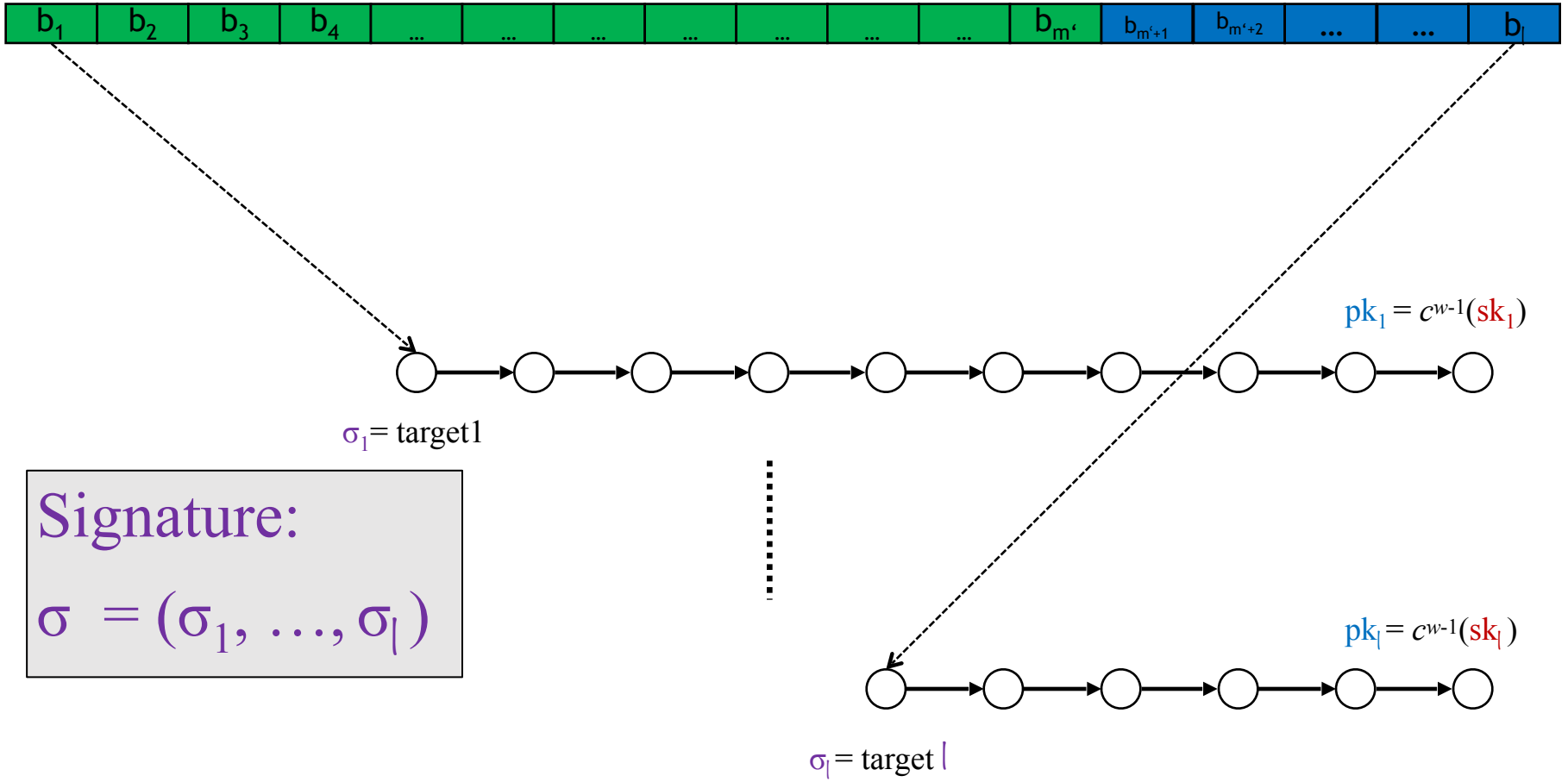
- Intuition:
  - Adversary gets black box access to for random P.
  - Adversary can adapatively query with restriction to use each tweak only once.
  - Adversary receives P and has to find second-preimage for one of its previous queries (such that P and T are the same).
- This is what the hashing in [HRS16] already tightly achieves!
  - Generating pseudorandom bitmasks & function keys from P and T.

Decisional second-preimage resistance

- (actually: Single-function multi-target decisional second preimage resistance for distinct tweaks)
- [HRS16] required statistical property: Every message input has to have a sibling (colliding value) under for the length-preserving case (|M| = |MD|).
- Reason: Want reduction using SPR instead of OW.

WOTS reduction from PRE

(assume adversary that always inverts one of the signature query elements)



Decisional second-preimage resistance

- (actually: Single-function multi-target decisional second preimage resistance for distinct tweaks)
- HRS16 required statistical property: Every message input has to have a sibling (colliding value) under for the length-preserving case (|M| = |MD|).
- Reason: Want reduction using SPR instead of OW.
  - WOTS reduction fails if guess was incorrect (Recall, in SPHINCS we have to make guesses)
  - When reducing SPR, we know full chain -> no guesses
- WOTS reduction gives us Inverter with non-negligible success probability

SPR PRE for length-preserving functions

- Reduction idea: Return
  - If **has sibling**, reduction loss
- Cannonical counter example: Identity function
- What about random functions?
  - About of inputs has no second preimage
  - Unbounded adversary might only return preimage if it is a singleton
- Reduction works if cannot tell singletons from values with siblings...
  - ... better than guessing.
- This is formalized in DSPR.

DSPR

Definition 3 (SPexists). *The second-preimage-exists predicate* $\mathrm{SPexists}(\mathrm{H})$ *for a hash function* H *is the function* $P : \mathcal{X} \to \{0,1\}$ *defined as follows:*

$$P(x) \stackrel{def}{=} \begin{cases} 1 & \text{if } |\mathrm{H}^{-1}(\mathrm{H}(x))| \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4 (SPprob). *The second-preimage-exists probability* $\mathrm{SPprob}(\mathrm{H})$ *for a hash function* H *is* $\mathsf{Pr}\left[x \leftarrow_R \mathcal{X} : P(x) = 1\right]$, *where* $P = \mathrm{SPexists}(\mathrm{H})$.

Definition 5 (DSPR). *The advantage of an algorithm* $\mathcal{A}$ *against the decisional second-preimage resistance of a hash function* H *is*

$$\mathrm{Adv}^{\mathrm{DSPR}}_{\mathrm{H}}(\mathcal{A}) \stackrel{def}{=} \max\left\{0, \mathsf{Pr}\left[x \leftarrow_R \mathcal{X}; b \leftarrow \mathcal{A}(x) : P(x) = b\right] - p\right\}$$

where $P = \mathrm{SPexists}(\mathrm{H})$ *and* $p = \mathrm{SPprob}(\mathrm{H})$.

DSPR

- Result: **DSPR + SPR PRE**

More:

- Best generic attack we found needs a high probability second-preimage finder (probability SPprob).
- Quantum query complexity is the same as for SPR
- Strongly compressing random functions have SPprob negligibly close to 1
  DSPR advantage can only be negligible.

Instantiating the tweakable hash (for SHA2)

SPHINCS$^+$-robust (XMSS)

- K = SHA2(pad(P) T),
  BM = SHA2(pad(P) T+1),
  MD= SHA2(pad(K) M BM)
- Standard model proof if K & BM were random,
- (Q)ROM proof when generating K & BM as above (modeling those SHA2 invocations as RO)
- Tight proof

SPHINCS$^+$- simple (LMS)

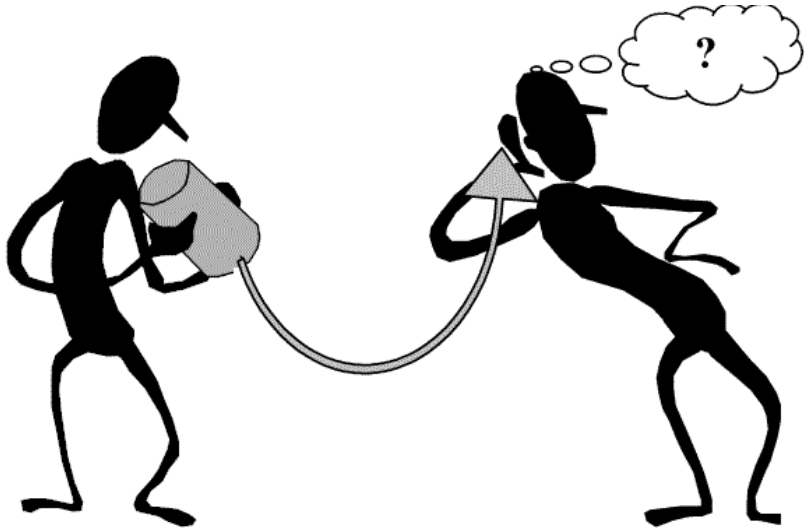
- MD = SHA2(P T M)
- QROM proof assuming SHA2 is QRO
- Tight proofs conjectured (LMS has tight proof)

Instantiating the tweakable hash

TU/e

- SPHINCS$^+$-simple is factor 3 faster
- SPHINCS$^+$-simple makes somewhat stronger assumptions about the security properties of the used hash function
- More research on direct constructions needed

Thank you!
Questions?



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

SPHINCS+

Joint work with Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe

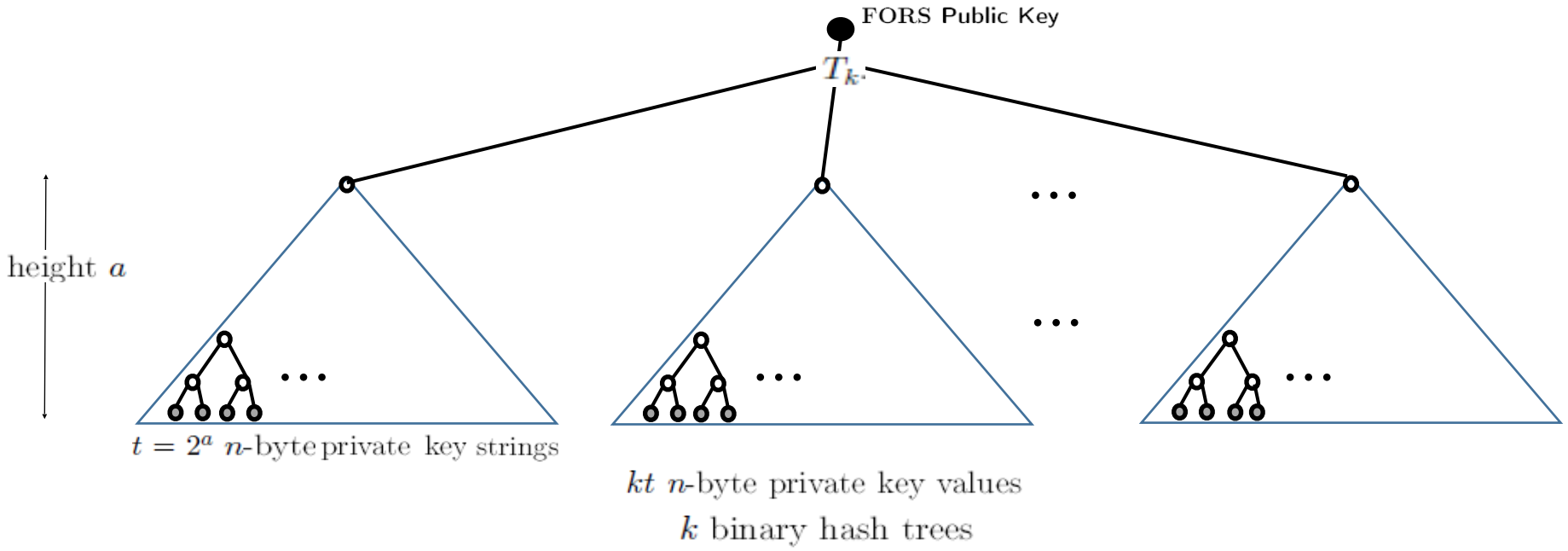
SPHINCS+ (our NIST submission)

TU/e

- Strengthened security gives smaller signatures
- Collision- and multi-target attack resilient (XMSS tweakable hash)
- Fixed length signatures
- Small keys, medium size signatures (lv 3: 17kB)
- Sizes can be much smaller if q_sign gets reduced
- The **conservative choice**

FORS (Forest of random subsets)

- Parameters t, a = log t, k such that ka = m



Verifiable index selection
(and optionally non-deterministic randomness)

TU/e

- SPHINCS:

- SPHINCS$^+$:

Verifiable index selection

TU/e

Improves FORS security

- SPHINCS: Attacks can target „weakest“ HORST key pair
- SPHINCS$^+$: Every hash query also selects FORS key pair
  - Leads to notion of interleaved target subset resilience

Instantiations (after second round tweaks)

- SPHINCS$^+$-SHAKE256-robust
- SPHINCS$^+$-SHAKE256-simple NEW!
- SPHINCS$^+$-SHA-256-robust
- SPHINCS$^+$-SHA-256-simple NEW!
- SPHINCS$^+$-Haraka-robust
- SPHINCS$^+$-Haraka-simple NEW!

Instantiations (small vs fast)

TU/e

	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

Comparison to SPHINCS-128 at same security level

	Signing median cycles	Verifying median cycles	Signature bytes
SPHINCS+ (n = 24, h = 55, d = 11, b = 8, k = 30, w = 16)	67 017 940	1 911 684	21 288
SPHINCS+ (n = 24, h = 51, d = 17, b = 9, k = 30, w = 16)	40 117 282	2 724 094	29 256
SPHINCS-128 (n=32, h=60, d=12, t=2^16, k=32, w = 16)	51 636 372	1 451 004	41 000

Hash-based Signatures in NIST „Competition“

TU/e

- SPHINCS+
  - FORS as few-time signature
  - XMSS-T tweakable hash
- Gravity-SPHINCS (R.I.P.)
  - PORS as few-time signature
  - Requires collision-resistance -> no tweakable hash
- (PICNIC)