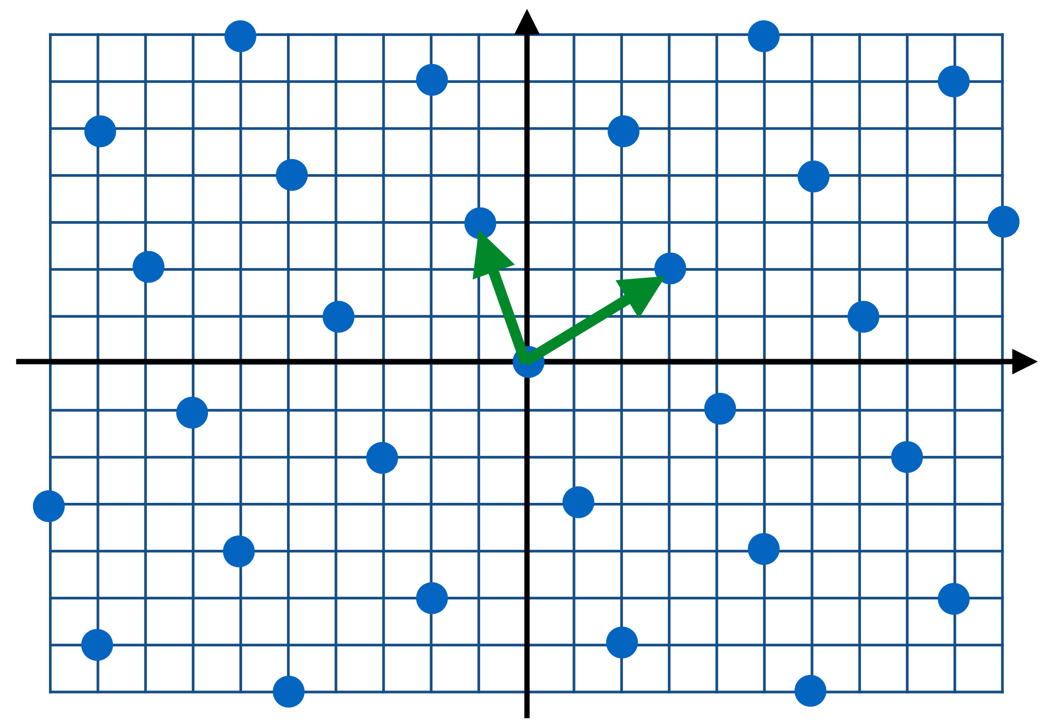
Lattice-based Cryptography and Cryptanalysis

Brice Minaud

email: brice.minaud@ens.fr

MPRI, 2025-2026

Post-Quantum Signatures



Hard problems in post-quantum world

Post-quantum candidate hard problems:

- Lattices.
- Code-based crypto.
- Isogenies.
- Symmetric crypto (→ signatures).
- Multivariate crypto.

Number Theory

Lattices are the mainstream candidate. Other PQ approaches for Public-Key crypto “only” motivated by PQ. Lattice-based crypto stands on its own:

- Simplicity (of some schemes, not their analysis).
- Security from worst-case hardness (in theory).
- Very expressive/versatile, e.g. FHE etc.

Lattices, codes,…
(conjectured)

Timeline for post-quantum transition

Aug. 2016: NSA surprise call for post-quantum security. (Updated) FAQ:
https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/1/CSI_CNSA_2.0_FAQ_.PDF

Dec. 2016: NIST call for post-quantum signature & key exchange schemes.
https://csrc.nist.gov/projects/post-quantum-cryptography

2022: Second NIST call for post-quantum signatures.
https://csrc.nist.gov/Projects/pqc-dig-sig/standardization/call-for-proposals

2025: ICCS call for post-quantum cryptographic algorithms.
https://niccs.org.cn/en/notice/202502/t20250205_378200.html

???

By 2035: most public-key cryptography should have transitioned (NIST).

What about France and Europe? 🇪🇺

Lots of French & European expertise on crypto primitives.

- Especially symmetric, code-based, lattice-based, isogenies, multivariate.

Many (most?) NIST winners are created in Europe.

ANSSI recommends hybrid approaches (w/o specific timeline).

Large national projects on cybersecurity, including post-quantum.

Current outlook for NIST PQ signatures (as of 2025)

First call, signature finalists (2022):

- Dilithium. [Lattices*]
- Falcon. [Lattices*]
- ~~Rainbow.~~ [Multivariate, broken]

Alternate signatures:

- SPHINCS+. [hash-based]
- Picnic. [MPC-in-the-head]
- ~~GeMSS.~~ [Multivariate, broken]

Second call for new post-quantum signatures (2023), especially:

- Not based on lattices.
- With short signatures.

Main Approaches for post-quantum signatures:

Lattices, codes, isogenies, multivariate, hash-based, MPC-in-the-head.

*Structured lattices.

How to build a signature scheme?

Two main paradigms.

Hash-and-Sign signatures

- Similar to RSA signatures.
- Based on a *trapdoor permutation*.

Lattices,
Multivariate,
Codes

Fiat-Shamir signatures

- Similar to Schnorr signatures.
- Based on a *zero-knowedge proof* with Fiat-Shamir transform.

Lattices,
Isogenies,
MPC-in-the-head

(**Hash-based signatures**

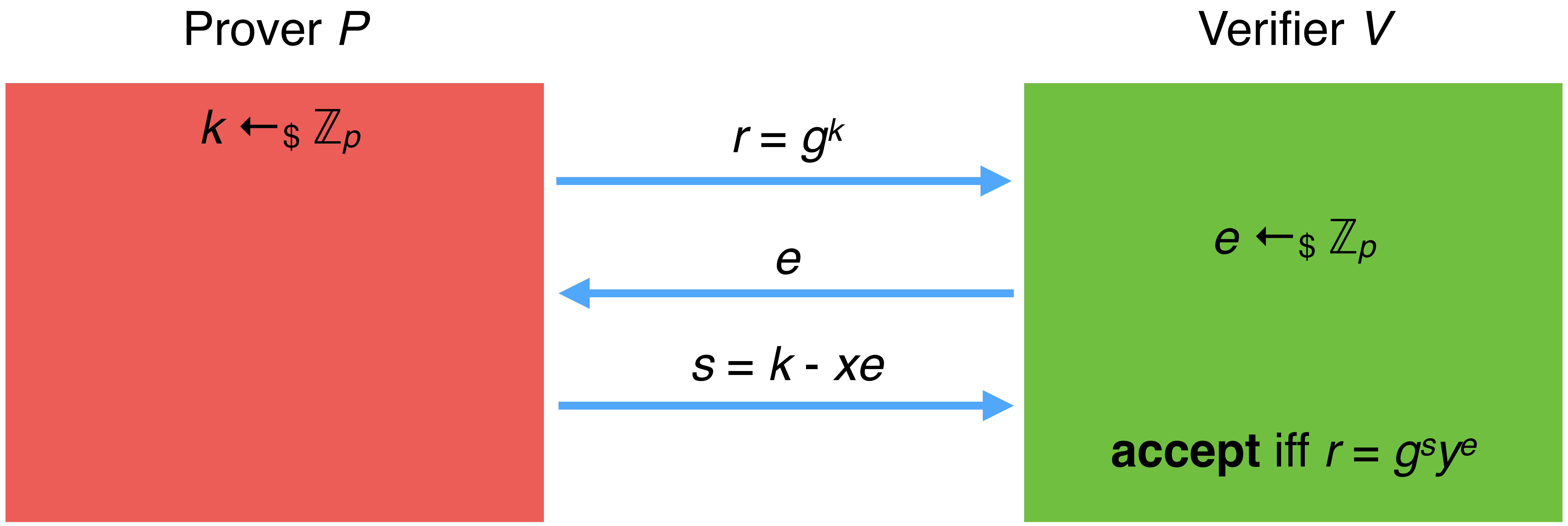
- Similar to SPHINCS/XMSS signatures.
- Based on a *one-time/few-time signature* with Goldreich/Merkle transform.)

A Fiat-Shamir lattice signature

DILITHIUM

Reminder: Schnorr protocol

- Let $\mathbb{G} = \langle g \rangle \sim \mathbb{Z}_p$ and $y \in \mathbb{G}$. I know $x \in \mathbb{Z}_p$ such that $y = g^x$.
- Corresponding language is trivial! $\forall y\ \exists x,\ y = g^x$. But proof of knowledge still makes sense.


Prover P
Verifier V
$k \leftarrow_{\$} \mathbb{Z}_p$
$r = g^k$
e
$e \leftarrow_{\$} \mathbb{Z}_p$
$s = k - xe$
accept iff $r = g^s y^e$

This is a proof of knowledge for knowing the discrete log x of y.

Fiat-Shamir: sigma protocol → signature

NIZK knowledge proof: "I know a witness w for R(x,w)" and can prove it non-interactively without revealing anything about w.

This is an identification scheme.

Sigma protocol → can integrate message into challenge randomness.

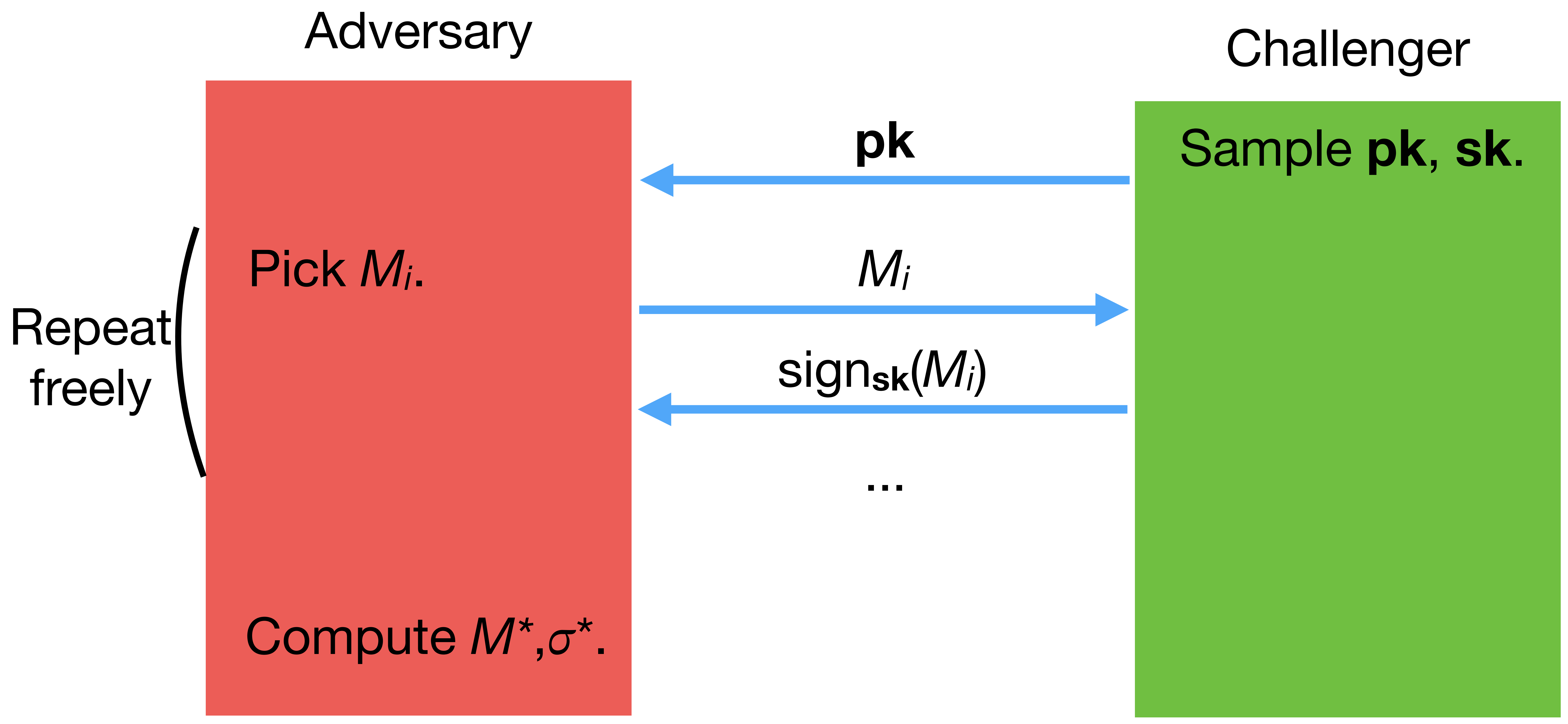
This yields a signature scheme!

- **Public key**: x
- **Secret key**: w
- **Sign**(m): signature = NIZK proof with challenge = hash(commit,m)
- **Verify** signature = verify proof.

That is the Fiat-Shamir transform.

If we want a lattice signature, all we need is a lattice-based sigma protocol.

EUF-CMA: existential unforgeability under chosen message attacks



The adversary wins iff $\text{verify}_{\mathbf{pk}}(M^*, \sigma^*) = \text{True}$, and $M^* \notin \{M_i\}$.

The signature scheme is **secure** if no PPT adversary wins, except with negligible probability.

Why Fiat-Shamir works

- Completeness of ZK proof ⇒ completeness of resulting signature

  Honest signatures are accepted = ZK proof is accepted.

- Soundess of ZK proof ⇒ hard to forge a signature

  Forging *knowing only the public key* = creating a ZK proof *knowing only the instance x.* (And based on a fresh challenge, due to the dependence on m.)

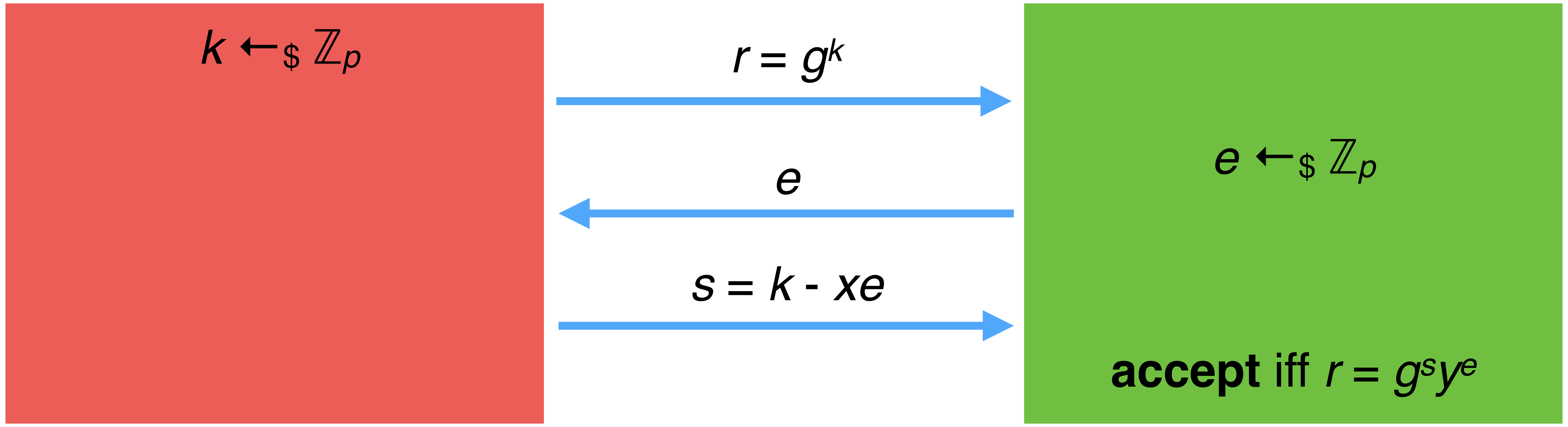
- Zero-Knowledge ⇒ the signature oracle can be simulated (i.e. it reveals nothing).

In a nutshell: simulatability of the signature oracle + hard to forge from only public-key ⇒ EUF-CMA security of the signature scheme.

Security reduction requires to model the hash function as a *Random Oracle*.

Schnorr signature

Schnorr protocol:


$k \leftarrow_\$ \mathbb{Z}_p$
$r = g^k$
$e \leftarrow_\$ \mathbb{Z}_p$
e
$s = k - xe$
accept iff $r = g^s y^e$

Schnorr signature:

- **Public key**: $y = g^x$
- **Secret key**: x
- **Sign**(m): signature $\sigma = (r,s)$ with $r = g^k$ for $k \leftarrow_\$ \mathbb{Z}_p$, $s = k - xH(r,m)$.
- **Verify**(σ,m): accept iff $r = g^s y^{H(r,m)}$.

Security reduces to Discrete Log, in the Random Oracle Model.

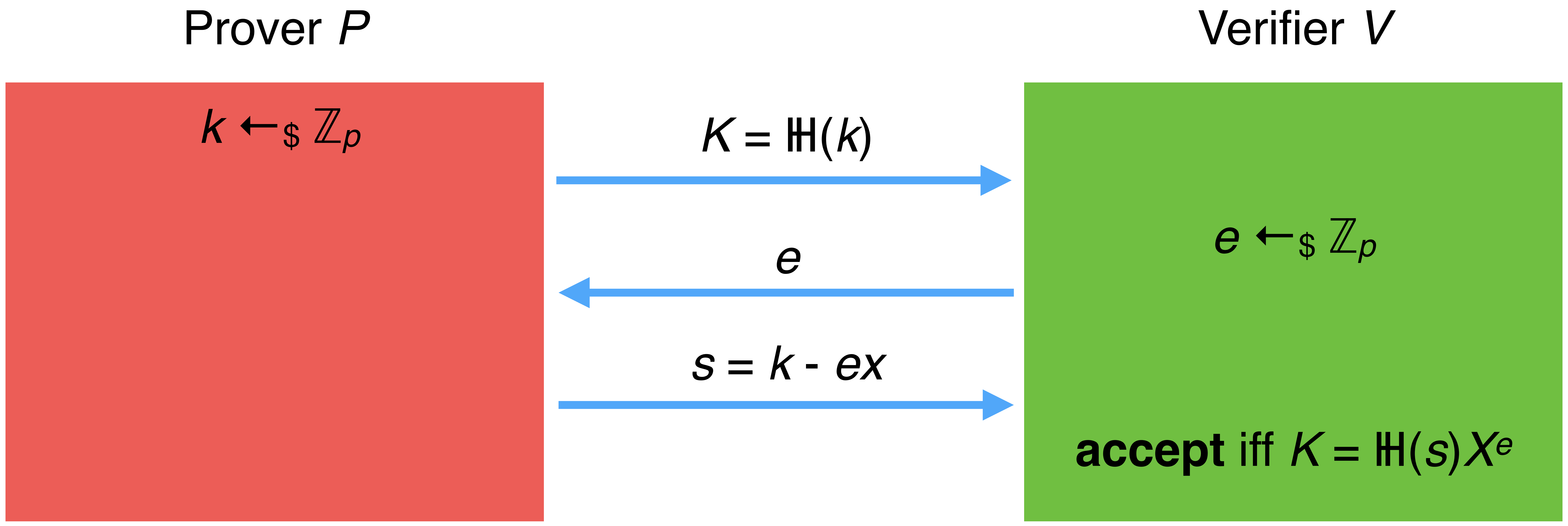
Homomorphic hash function

Let's look at $\mathbb{H}$: $x \mapsto g^x$ as a (strange) **hash function**.

- It is preimage-resistant (from hardness of discrete log problem).
- It is homomorphic: $\mathbb{H}(x+y) = \mathbb{H}(x)\mathbb{H}(y)$.

Rewritten Schnorr protocol

- Let $\mathbb{G} = \langle g \rangle \sim \mathbb{Z}_p$ and $X \in \mathbb{G}$. I know $x \in \mathbb{Z}_p$ such that $X = \mathbb{H}(x)$.



Prover P
Verifier V
$k \leftarrow_{\$} \mathbb{Z}_p$
$K = \mathbb{H}(k)$
e
$e \leftarrow_{\$} \mathbb{Z}_p$
$s = k - ex$
accept iff $K = \mathbb{H}(s)X^e$

This is correct as long as $\mathbb{H}$ is homomorphic.

Don't need $\mathbb{H} = x \mapsto g^x$ specifically.

...do you know a lattice-based "hash function" that is homomorphic?

Ajtai's hash function!

Short Integer Solution (SIS)

Ajtai '96 (the foundational article of Lattice-based crypto).

Say I have $m > n$ vectors a_i in $\mathbb{Z}_q^n$.

Problem: find **short** $x = (x_1,\ldots,x_m)$ in $\mathbb{Z}_q^m$ such that $\sum x_i a_i = 0$.
Here, **short** means of small norm: $\|x\| \leq \beta$.

- The crucial point is the norm constraint β. Otherwise this is just a linear system.
- Typically, Euclidian norm, with representatives in $[-q/2,q/2]$.
- Solution must exist as long as there are at least q^n vectors of norm $\leq \beta/\sqrt{2}$, due to collisions. E.g. $\beta > \sqrt{n \log q}$ and $m \geq n \log q$.

SIS and lattices

Equivalent formulation:

SIS problem. Given a uniform matrix $A \in \mathbb{Z}_q^{n \times m}$, find $x \in \mathbb{Z}_q^m$ with and $||x|| \leq \beta$ such that $Ax = 0$.

For *A* as above, define $\mathcal{L}^\perp(A) = \{x \in \mathbb{Z}_q^m : Ax = 0\}$(in $\mathbb{Z}_q$).

This is a (*q*-ary) lattice!

SIS = finding a short vector in $\mathcal{L}^\perp(A)$.

Better! Ajtai '96: Solving SIS (for uniformly random *A*) implies solving GapSVP$_{\beta\sqrt{n}}$ in dimension *n* for **any** lattice!

→ "Worst-case to average-case" reduction. Note *m* irrelevant.

(Cryptographic) hash function

Hash function H: $\{0,1\}^* \rightarrow \{0,1\}^n$.

Preimage resistance: for uniform $y \in \{0,1\}^n$, hard to find x such that $H(x) = y$.

Collision resistance: hard to find $x \neq y \in \{0,1\}^*$ such that $H(x) = H(y)$.

Note: collision is ill-defined for a single hash function. (why?)

→ To formally define hash functions, usually assume they are a *family* of functions. Parametrized by a “key”.

(See also Random Oracle Model.)

(Cryptographic) hash function

In theory, collision-resistance ⇒ preimage resistance.

Argument: if the hash function is “compressing” enough, whp the preimage computed by a preimage algorithm, on input H(x), will be distinct from x. (Because most points will have many preimages.)

In practice, preimage resistance should cost 2^n, while collision resistance should cost $2^{n/2}$. → Previous reduction is not so relevant.

Right now we are more in the world of theory, so we’ll only care about collision resistance.

Ajtai's hash function

Pick random $A \in \mathbb{Z}_q^{n\times m}$. Define:

$$\begin{aligned} H_A : \{0,1\}^m &\to \mathbb{Z}_q^n \\ x &\mapsto Ax \end{aligned}$$

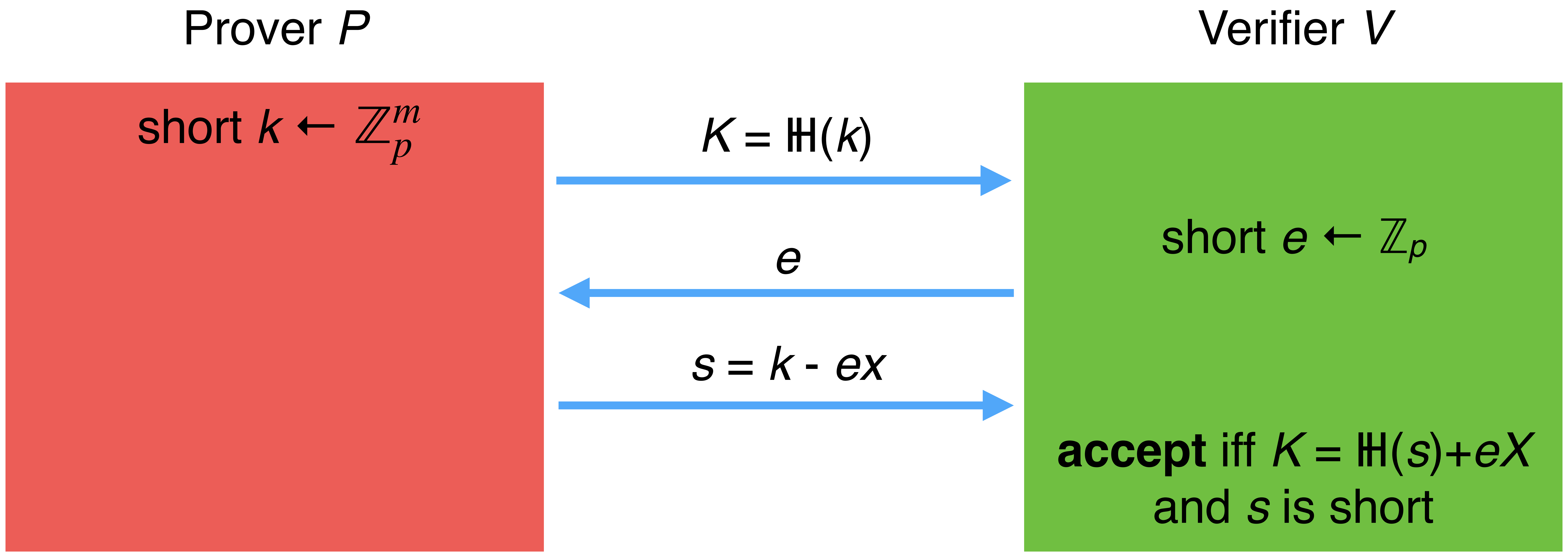
Finding a collision for random *A* yields a SIS solution with $\beta = \sqrt{m}$.

Indeed, $H_A(x) = H_A(x)$ yields $A(y-x) = 0$ with $y-x \in \{-1,0,1\}^m$.

Example: $q = n^2$, $m = 2n \log q$ (compression factor 2), need roughly $n \sim 100$, $mn \sim 100000$…

Lattice-based "Schnorr" protocol

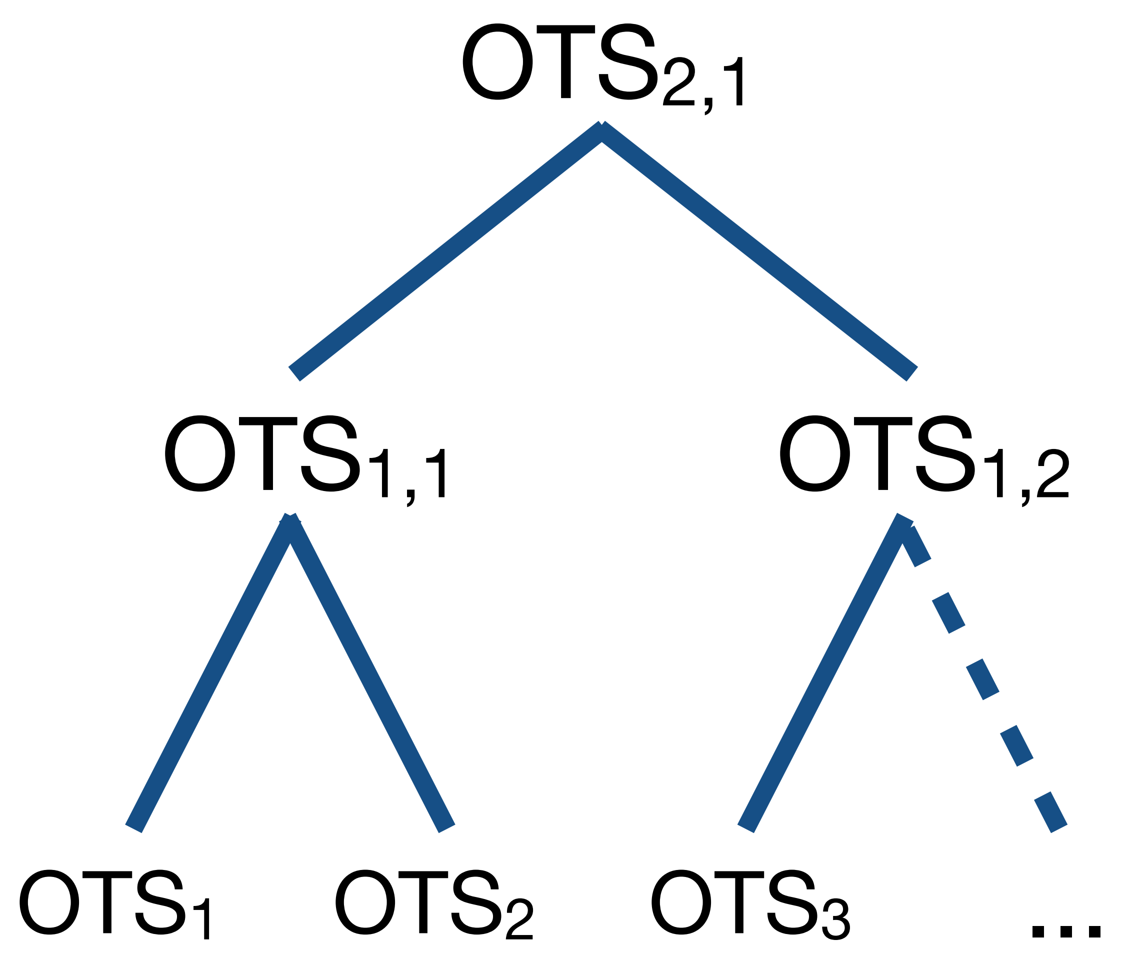
Let $\mathbb{H}: \mathbb{Z}_p^m \to \mathbb{Z}_p^n$ such that $\mathbb{H}(x) = Ax$. I know short x such that $X = \mathbb{H}(x)$.


Prover P
Verifier V
short $k \leftarrow \mathbb{Z}_p^m$
$K = \mathbb{H}(k)$
short $e \leftarrow \mathbb{Z}_p$
e
$s = k - ex$
accept iff $K = \mathbb{H}(s) + eX$ and s is short

This is exactly Schnorr, using Ajtai's hash function in place of $x \mapsto g^x$.

Remark. Because domain($\mathbb{H}$) = short vectors, need to add some shortness conditions.

The third family: hash-based



Hash-based signatures

For signing, a hash function is needed.

$$\mathsf{hash} : \{0,1\}^* \to \{0,1\}^n$$

We need to assume the hash function is hard to invert: it is *preimage-resistant*.

In fact, this is enough to build a signature scheme!

+	Minimalist assumption. High level of confidence in security.
-	Huge signatures, slow. *No lattices :(*

How?

Challenge: given a one-way function, build a signature scheme.

We start with a *one-time signature* (OTS).

A one-time-signature is secure as long as you use it to sign a single message.

Note: the message is chosen *after* the signature key is published.

Lamport signature

One-time-signature for a single bit from a hash function h.

Pick two random values x_0 and x_1.

- The secret key is sk = (x_0,x_1).
- The public key is pk = (y_0,y_1) with y_0 = h(x_0), y_1 = h(x_1).

Signature: to sign the bit b, reveal x_b:

$$s = x_b$$

Verification: simply check h(x_b) = y_b.

Could instantiate with Ajtai's hash function. Can do better...

Lyubashevsky-Micciancio One-Time Signature

One-time-signature for multiple bits from Ajtai's hash function.

Take $H \in (\mathbb{Z}_p)^{n \times m}$ a uniformly random matrix.

- Secret key: uniformly random $K \in (\mathbb{Z}_p)^{m \times k}$ with $\|K\|_\infty \leq 1$.
- Public key: uniformly random $H \in (\mathbb{Z}_p)^{n \times m}$, and $H' = HK$.

Signature: the signature of a message $m \in \{0,1\}^k$ with $\mathrm{Ham}(m) = w$ is:

$$s = Km$$

Verification: check $\|s\|_\infty \leq w$, and $H'm = Hs$.

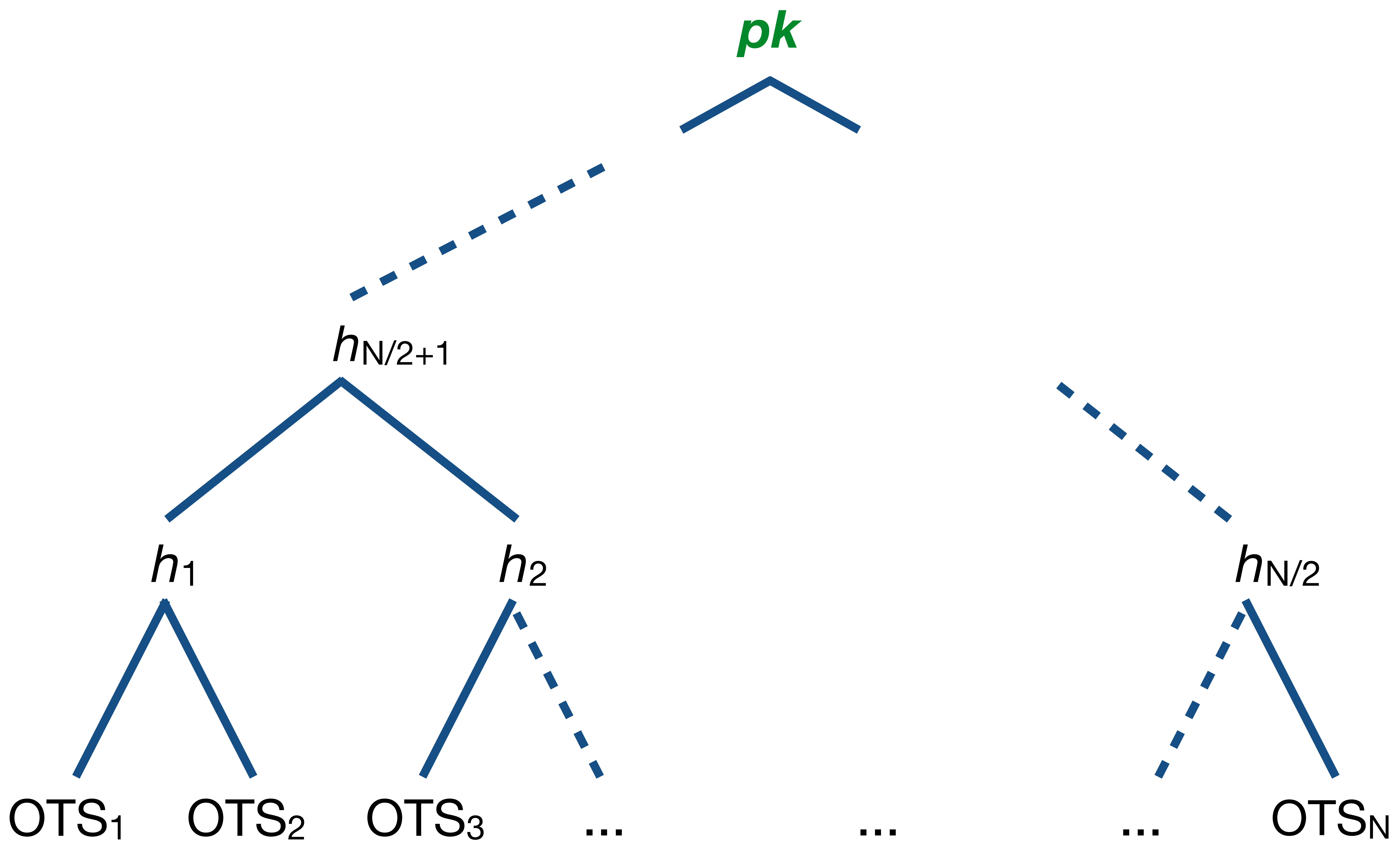
(Parameters: Can set $m = (\lambda + n \log_2 p)/\log_2 3$, k & w are security/hardness tradeoff.)

Lamport signature

There are also hash-based schemes for multiple bits (Winternitz signatures). More efficient than multiple instances of Lamport.

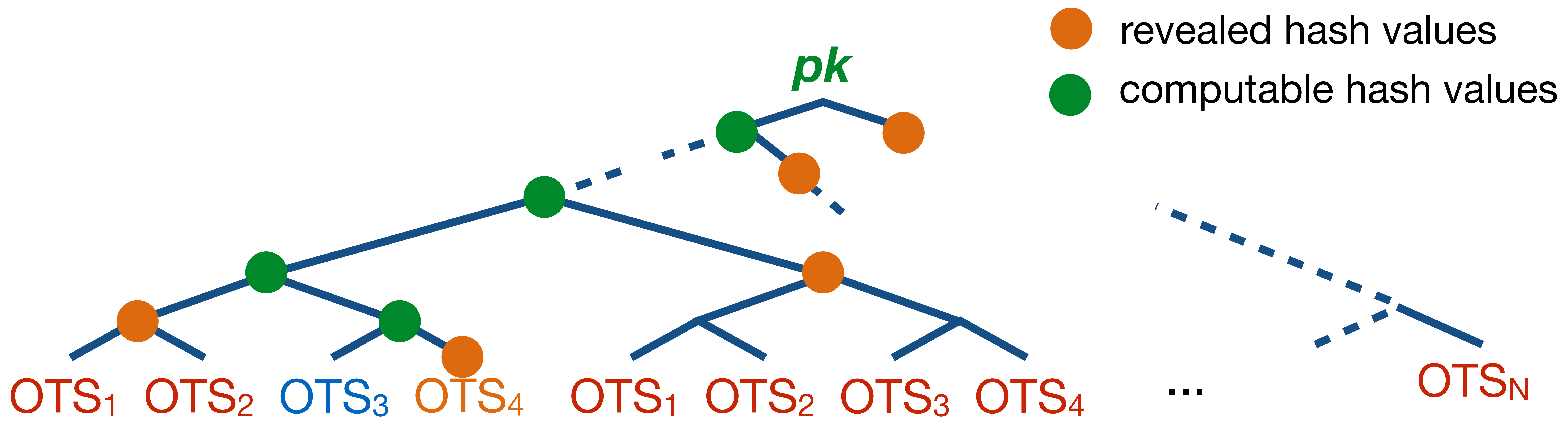
Next challenge: how to go from one-time signature to many-time signature?

Solution 1: Merkle trees



Each node in the Merkle tree is a hash of its children.

Solution 1: Merkle trees



pk
revealed hash values
computable hash values
OTS_1 OTS_2 OTS_3 OTS_4 OTS_1 OTS_2 OTS_3 OTS_4 ... OTS_N

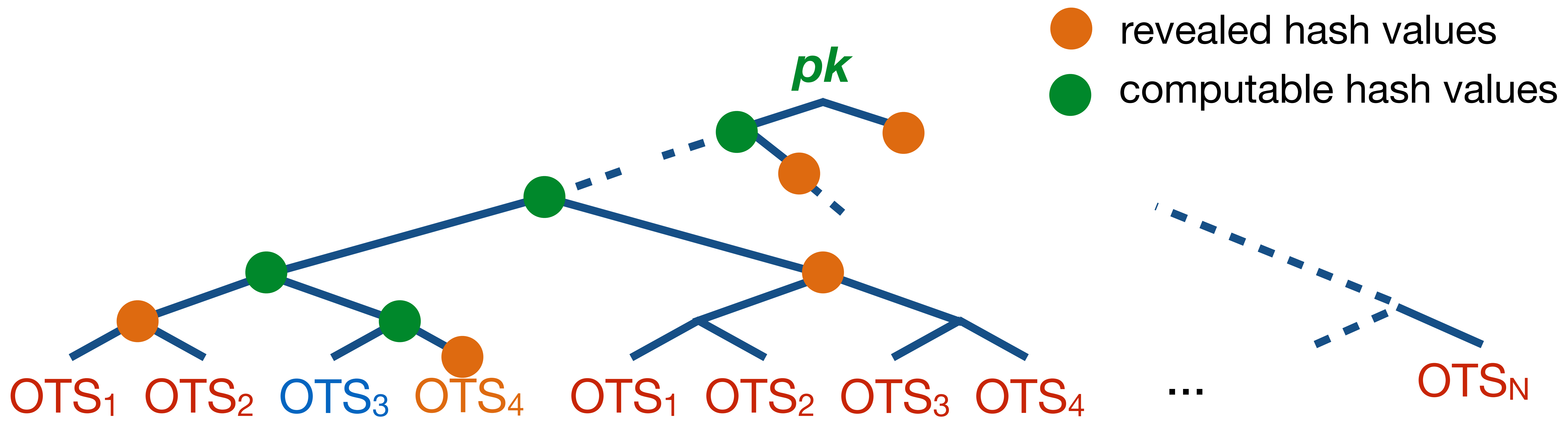
- The secret key is sk = (OTS_1,OTS_2,...,OTS_N).
- The public key is the root of the tree *pk*.

Signature: to sign the *i*-th message, reveal hash values in the tree forming a path from OTS_i to the root *pk*, and use OTS_i to sign:

$$s = h_{i1}, \ldots, h_{ik}, OTS_i, OTS_i(m)$$

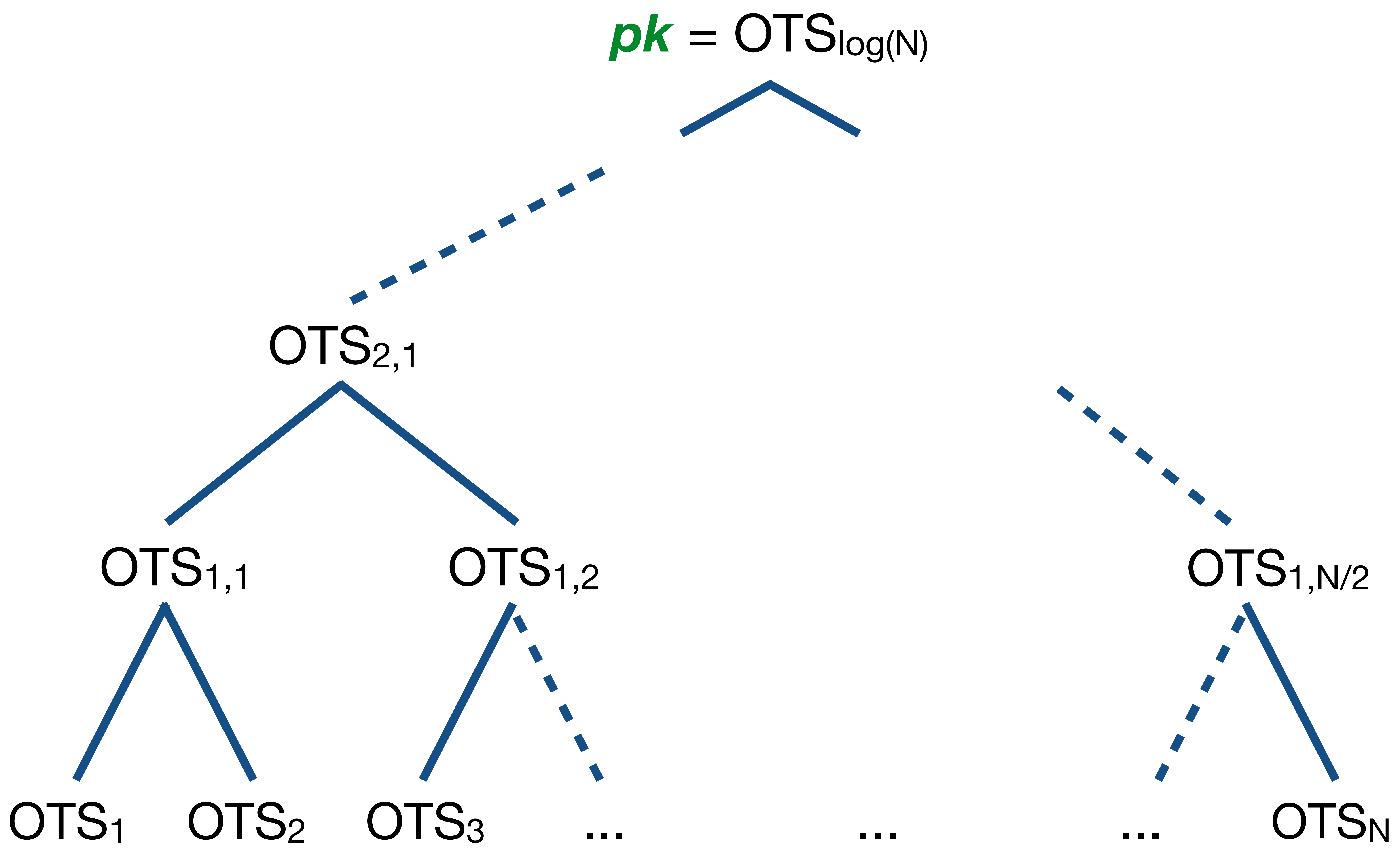
Verification: check the OTS_i signature, and all hashes.

Solution 1: Merkle trees



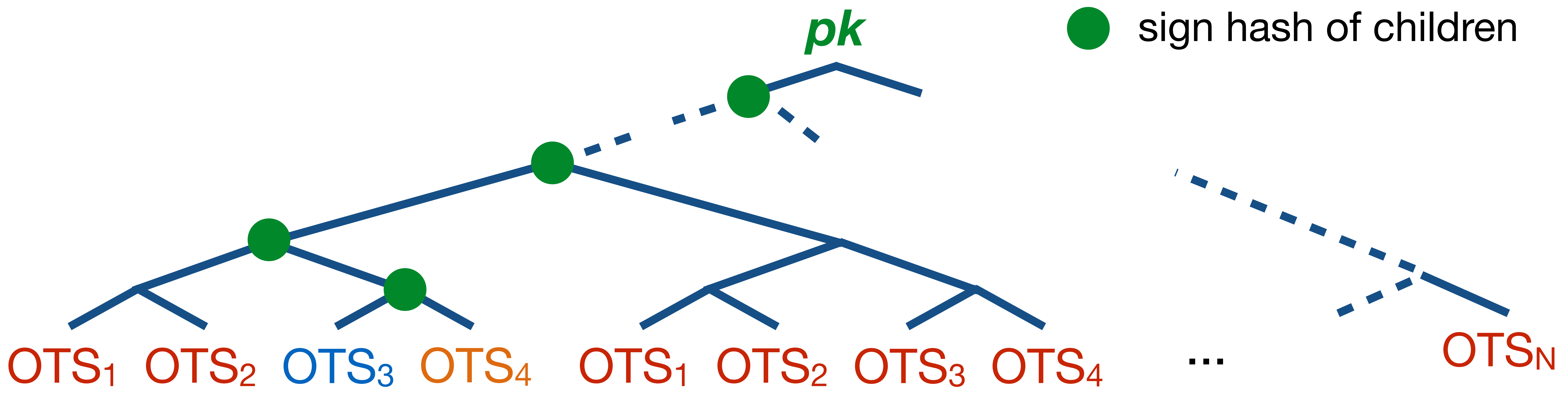
- Can sign up to N messages.
- Signatures are length $O(\log(N))$.
- Needs a state to store which OTS_i is next to be used.
- **Problem**: need $O(N)$ precomputation to get *pk*!

Solution 2: Goldreich scheme



Each node in the Goldreich tree is a separate OTS scheme.

Solution 2: Goldreich scheme

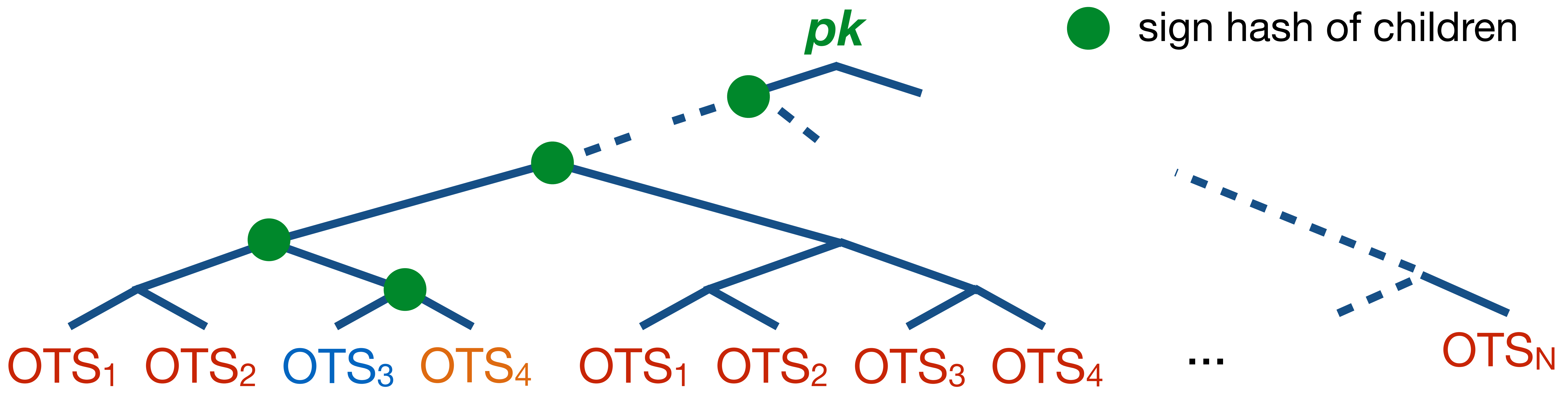


pk
● sign hash of children
OTS_1 OTS_2 OTS_3 OTS_4 OTS_1 OTS_2 OTS_3 OTS_4
…
OTS_N

- The secret key is a random seed used to generate all OTS_i's.
- The public key is the (hash of the) OTS at the root of the tree.

Signature: to sign the *i*-th message, use the *i*-th OTS_i scheme at a leaf, then use each OTS along the path from OTS_i to the root to sign the hash of both children.
Verification: check the final and all intermediate OTS signatures, and that the hash of the root matches *pk*.

Solution 2: Goldreich scheme



- Can sign up to N messages.
- Signatures are length $O(\log(N))$.
- Needs a state to store which OTS_i is next to be used.
- $O(1)$ precomputation to get *pk*!
- Longer signatures.

A Hash-and-Sign lattice signature

FALCON

Fast-Fourier Lattice-based
Compact Signatures over NTRU

Trapdoor permutation

Trapdoor permutation

Sample: outputs key **k** and *trapdoor* **t**.

Forward: given key **k** and input x, can compute y = F(**k**,x) in PPT.

Inverse: given trapdoor **t** and target y, can compute x such that y = F(**k**,x) in PPT.

Security: given target y, **cannot** compute x such that y = F(**k**,x) in PPT (except with negligible probability of success).

Example: RSA is a trapdoor permutation, with key **N** = pq, trapdoor **d** = e^{-1} mod ϕ(N), F(**N**,x) = x^e mod N, F^{-1}(**N**,**d**,y) = y^d mod N.

Hash-and-Sign Signatures

Hash-and-Sign signature   Given: 'hash' = hash function.

Public key: pk = key for trapdoor permutation F (allows to compute F).

Secret key: trapdoor **t** for F (allows to compute F^{-1}).

- **Sign(m):** $\sigma = F^{-1}(\text{hash}(m))$, computed using **t**.
- **Verify(m,σ):** check $F(\sigma) = \text{hash}(m)$, computed using **pk**.

This blueprint* transforms a *trapdoor permutation* into a *signature scheme*.

$\Rightarrow$ all we need is a lattice-based trapdoor permutation.

Remark: we need the hash function to map into the range of F.

*only a “blueprint” because it does not necessarily yield a secure signature scheme (cf. later).

Short Integer Solution (SIS)

Ajtai ’96 (the foundational article of Lattice-based crypto).

Say I have $m > n$ vectors a_i in $\mathbb{Z}_q^n$.

Problem: find **short** $x = (x_1,\ldots,x_m)$ in $\mathbb{Z}_q^m$ such that $\sum x_i a_i = 0$.
Here, **short** means of small norm: $\|x\| \leq \beta$.

- The crucial point is the norm constraint β. Otherwise this is just a linear system.
- Typically, Euclidian norm, with representatives in $[-q/2,q/2]$.
- Solution must exist as long as there are at least q^n vectors of norm $\leq \beta/\sqrt{2}$, due to collisions. E.g. $\beta > \sqrt{n \log q}$ and $m \geq n \log q$.

Inhomogeneous SIS problem

ISIS problem. Given a uniform matrix $A \in \mathbb{Z}_q^{n\times m}$, and $t \in \mathbb{Z}_q^n$, find $x \in \mathbb{Z}_q^m$ with and $||x|| \leq \beta$ such that $Ax = t$.

ISIS ⇔ finding a preimage for the hash function x ↦ Ax.

⚠️ x is required to be *short*.

⇒ ISIS implies preimage resistance for x ↦ Ax, while SIS implies collision resistance.

Adding a trapdoor

ISIS problem. Given a uniform matrix $A \in \mathbb{Z}_q^{n \times m}$, and $t \in \mathbb{Z}_q^n$, find $x \in \mathbb{Z}_q^m$ with and $||x|| \leq \beta$ such that $Ax = t$.

SIS = finding a *short* vector in $\mathscr{L}^{\perp}(A)$.

ISIS = finding a *close* vector in $\mathscr{L}^{\perp}(A)$.

⇒ a good basis of $\mathscr{L}^{\perp}(A)$ yields a “trapdoor” for SIS and ISIS.

⇒ it gives a trapdoor for $x \mapsto Ax$, turning it into a trapdoor permutation.

GPV/Falcon-style Hash-and-Sign lattice-based signature

<u>Falcon-style signature</u>

Public key: A = matrix for which $F_{\mathbf{A}}$: $x \mapsto \mathbf{A}x$ is preimage-resistant.*

Secret key: trapdoor **t** for $F_{\mathbf{A}}$ = good basis of $\mathcal{L}^{\perp}(A)$.*

- **Sign(m):** short $\sigma = F_{\mathbf{A}}^{-1}(\text{hash}(m))$, computed using **t**.
- **Verify(m,σ):** check $F_{\mathbf{A}}(\sigma) = \text{hash}(m)$, *and* check σ is short.

Given only **A**, forging a signature

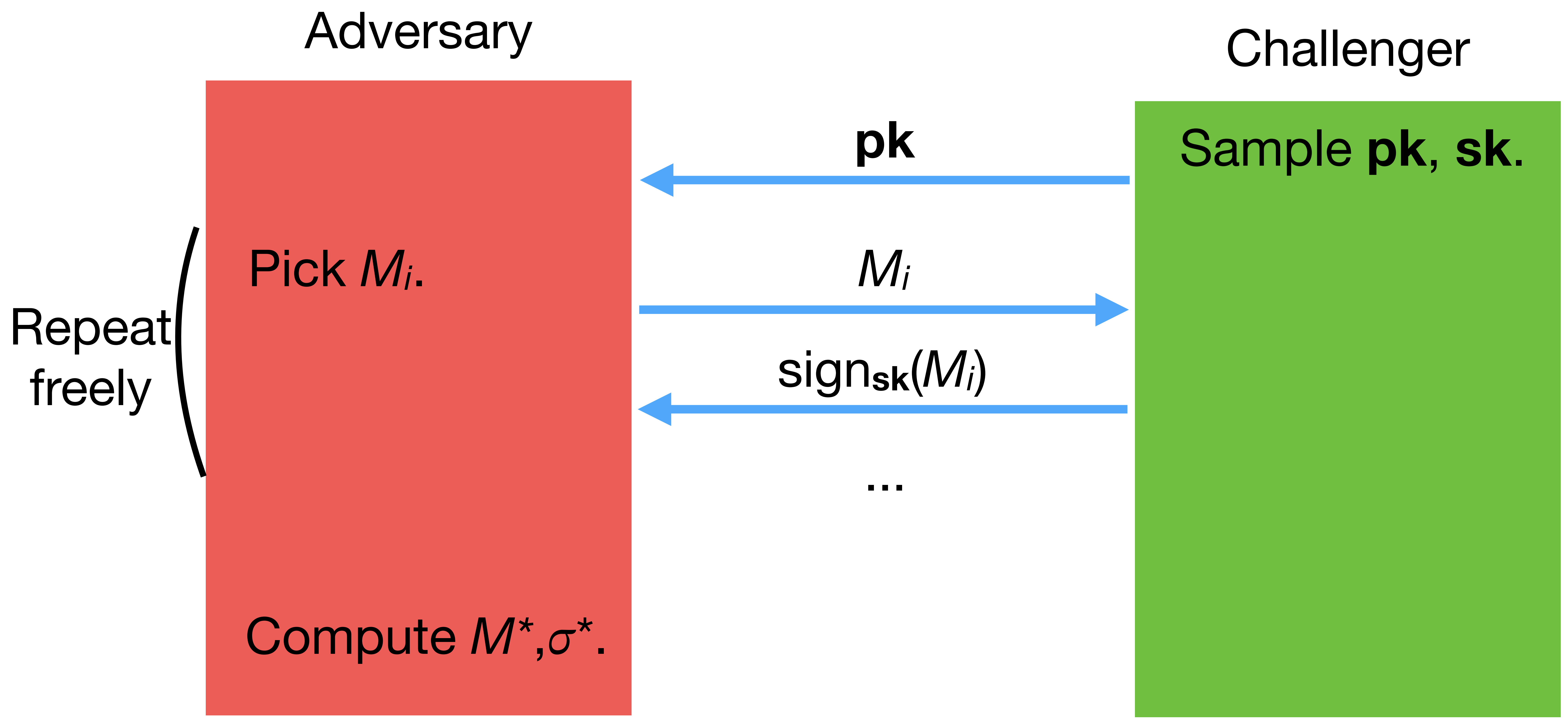
= finding a short preimage of hash(m) for the function $x \mapsto \mathbf{A}x$

= solving ISIS for random target.

But this is not enough for security. *(Why? More to come...)*

*Falcon uses NTRU lattices.

EUF-CMA: existential unforgeability under chosen message attacks



The adversary wins iff $\text{verify}_{\mathbf{pk}}(M^*, \sigma^*) = \text{True}$, and $M^* \notin \{M_i\}$.

The signature scheme is **secure** if no PPT adversary wins, except with negligible probability.

Arguing security for Hash-and-Sign signatures

General idea:

- Without signature oracle, trapdoor permutation security ⇒ unforgeability.

  So all we need is to argue the signature oracle does not leak information.

- To do that, argue (message,signature) pairs can be simulated *without* **sk**.

  Common strategy: show that (x,F(x)) is indistinguishable from (F^{-1}(y),y).

Roughly, in our lattice-based scheme, this means showing:*

“(x,Ax) where x ← short”

is indistinguishable from:

“(F^{-1}_A(y),y) where y ← uniform”

*Falcon uses a form of rejection sampling to enforce this.

More about security

Analysis of modified Schnorr protocol

- (Perfect) Completeness.

  Follows directly from homorphism of $\mathbb{H}$.

- (Special) Knowledge soundness.

  Extractor **for original Schnorr:** gets $K = g^k$, asks two challenges $e \neq e'$, gets back s, s' with $K = \mathbb{H}(s) + eX = \mathbb{H}(s') + e'X$. Implies $\mathbb{H}(s\text{-}s')$ = $(e'\text{-}e)X$. Yields $X = \mathbb{H}((s-s')(e'-e)^{-1}) \Rightarrow$ preimage of X.

  This argument no longer works. *(Why?)*

- Honest-verifier zero knowledge.

  Simulator **for original Schnorr:** draw $e \leftarrow_{\$} \mathbb{Z}_p$, $s \leftarrow_{\$} \mathbb{Z}_p$, **then** $K = \mathbb{H}(s) + eX$. Return transcript (K,e,s). Note K, e still uniform and independent $\rightarrow$ distribution is identical to real transcript.

  This argument no longer works. *(Why?)*

Lyu-09: fixing the modified Schnorr