Code-based, Multivariate and Hash-based Cryptography

Brice Minaud

Information Security Group

ROYAL HOLLOWAY UNIVERSITY OF LONDON

Quantum Communications Hub Training Workshop, 27th May 2018

Plan

Introduction.

1. Code-based Cryptography.
2. Multivariate Cryptography.
3. Hash-based Signatures.

Post-quantum schemes (that are not lattices).

Conclusion.

RSA

- Select a pair of random primes p, q. Set $N = pq$.
- Select integers d, e such that $de = 1 \bmod (p-1)(q-1)$.
  - The public key is $pk = (e,N)$.
  - The secret key is $sk = d$.

Encryption: for a message $m \in [1,N-1]$, the ciphertext is:

$$c = m^e \bmod N.$$

Decryption: for a ciphertext c, the message is:

$$m = c^d \bmod N.$$

You can think of $e = 3$.

Hard problem: computing third root modulo N.

Trapdoor: knowledge of prime decomposition $N = p \cdot q$.

Post-quantum hard problems

There is nothing wrong with the general outline of building encryption or signatures from a **hard problem** + **trapdoor**.

- Ultimately, post-quantum cryptography is "just" about changing the underlying hard problems.

...and ensuring post-quantum resistance.
...and selecting concrete parameters.
...and ensuring side-channel resistance.
...and changing proof models (quantum random oracles, post-post quantum cryptography...).
...and optimising classical efficiency.
.....and actually deploying the result.

Code-based Cryptography

(Linear) error-correcting codes

We operate on $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is a finite field with q elements. Think $q = 2$.

The *Hamming distance* between two elements of $\mathbb{F}_q^n$ is the number of bit positions where they differ.

dist(00101,00011) = 2

The (Hamming) *weight* of $x \in \mathbb{F}_q^n$ is the number of non-zero coordinates.

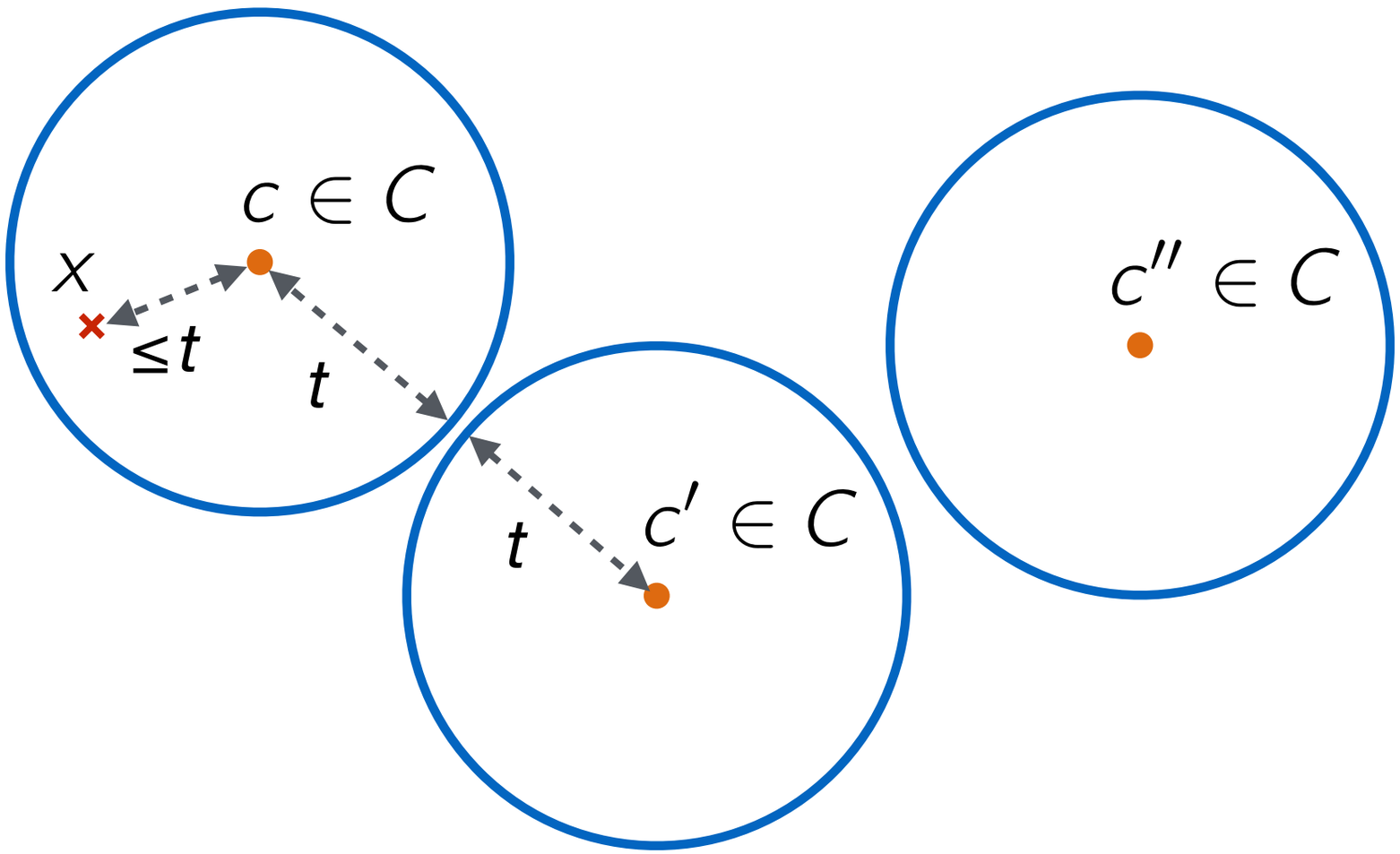
hw(0100**11**) = 3

A (linear) *code* of length n, rank k, and distance d is a linear subspace of $\mathbb{F}_q^n$ of dimension k, such that the minimum distance between distinct elements is d.

(Linear) error-correcting codes

If the distance of a code C is $d = 2t + 1$, then C can correct up to t errors.

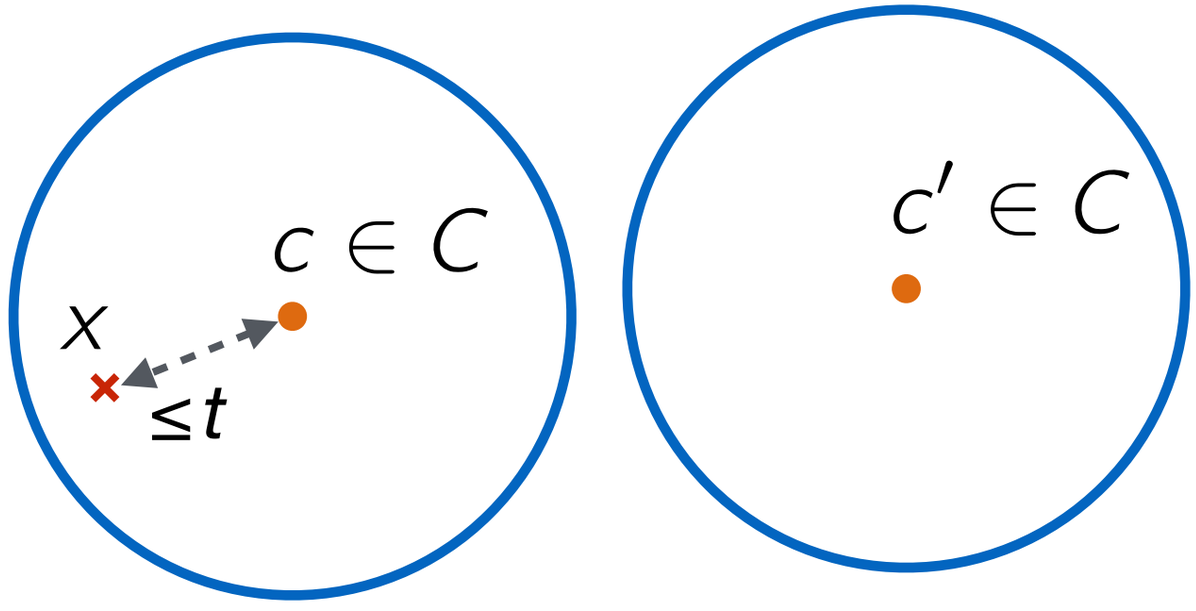
- That is, for $x \in \mathbb{F}_q^n$, if dist(x-c) ≤ t for some $c \in C$, then c is unique.



Decoding

Recall a code C is a linear subspace. Concretely, C may be represented by some basis. A matrix G whose row vectors span the code is called a *generator matrix*.

Problem: given a generator matrix G (i.e. a basis of C) and some x such that x-$c \leq t$ for some c in C, find c.



- For a random linear code, this is a **hard problem**!

Trapdoor

In practice, codes are generally not random, but structured.

E.g. Goppa codes.

The structure ensures that decoding is efficient.

E.g. Patterson's algorithm.

- Knowledge of the structure enables efficient decoding. Otherwise it is a **hard problem**...

McEliece

Robert McEliece, 1978.

Pick a binary t-correcting Goppa code with generator matrix G.

Public key: $G' = S \cdot G \cdot P$, where S is a random invertible matrix, and P is a random permutation matrix.

Secret key: S, G, P.

Encrypt: encode a message m into the code C' (generated by G'), pick a random error vector e of weight t. The ciphertext c is:

$$c = m + e$$

Decrypt: given a ciphertext c, decode c using knowledge of the equivalence between C and C' (via S, P).

McEliece

Underlying hard problem(s):

- It is hard to distinguish C' from a random linear code.
- It is hard to decode a random linear code.

Sometimes described as "reducing" to random linear decoding...

Warning: whether the first problem is actually hard is highly dependent on the type of linear code used.

History

The original McEliece, using binary Goppa codes, is essentially unbroken since 1978.

- Best attack is generic linear decoding using Information Set Decoding. Very stable complexity.

Also enables signatures via Niederreiter variant.

Various later efficiency enhancements using other types of codes were broken.

Code-based crypto: conclusion

\+ High security confidence for original McEliece.
Well-studied.
Cheap encryption.

\- Public key = G' → large key sizes (several Mb).
Heuristic security reduction.

Somewhat ignored in practice until post-quantum cryptography came along.

Current schemes mainly try to reduce public key size, using e.g. cyclic codes.

Multivariate Cryptography

Multivariate Cryptography

Hard problem: solving a system of random, say, quadratic, equations over some finite field (MQ).

The MQ problem is NP-hard.

This is good for post-quantum security!

With some caveats (asymptotic notion, average-case hardness...).

Multivariate Cryptography

Hard problem: solving a system of random quadratic equations over some finite field (MQ).

→ How to get an encryption scheme $\mathbb{F}_q^n \to \mathbb{F}_q^n$:

Public key: encryption function **F** given as sequence of *n* quadratic polynomials in *n* variables.

Private key: hidden structure (decomposition) of **F** that makes it easy to invert.

+: small message space, fast with private key.
-: slow public-key operations, large public key.

Quadratic polynomials

Homogeneous degree-two polynomials (over a field of odd characteristic) may be represented as a symmetric matrix:

$$x^2 + 4xy + 3z^2 = \begin{pmatrix} x & y & z \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$= X^{\mathsf{T}} \cdot M \cdot X \quad \text{for } X = \begin{pmatrix} x & y & z \end{pmatrix}$$

Trapdoor

Say the bottom right quadrant of the matrix is zeros...

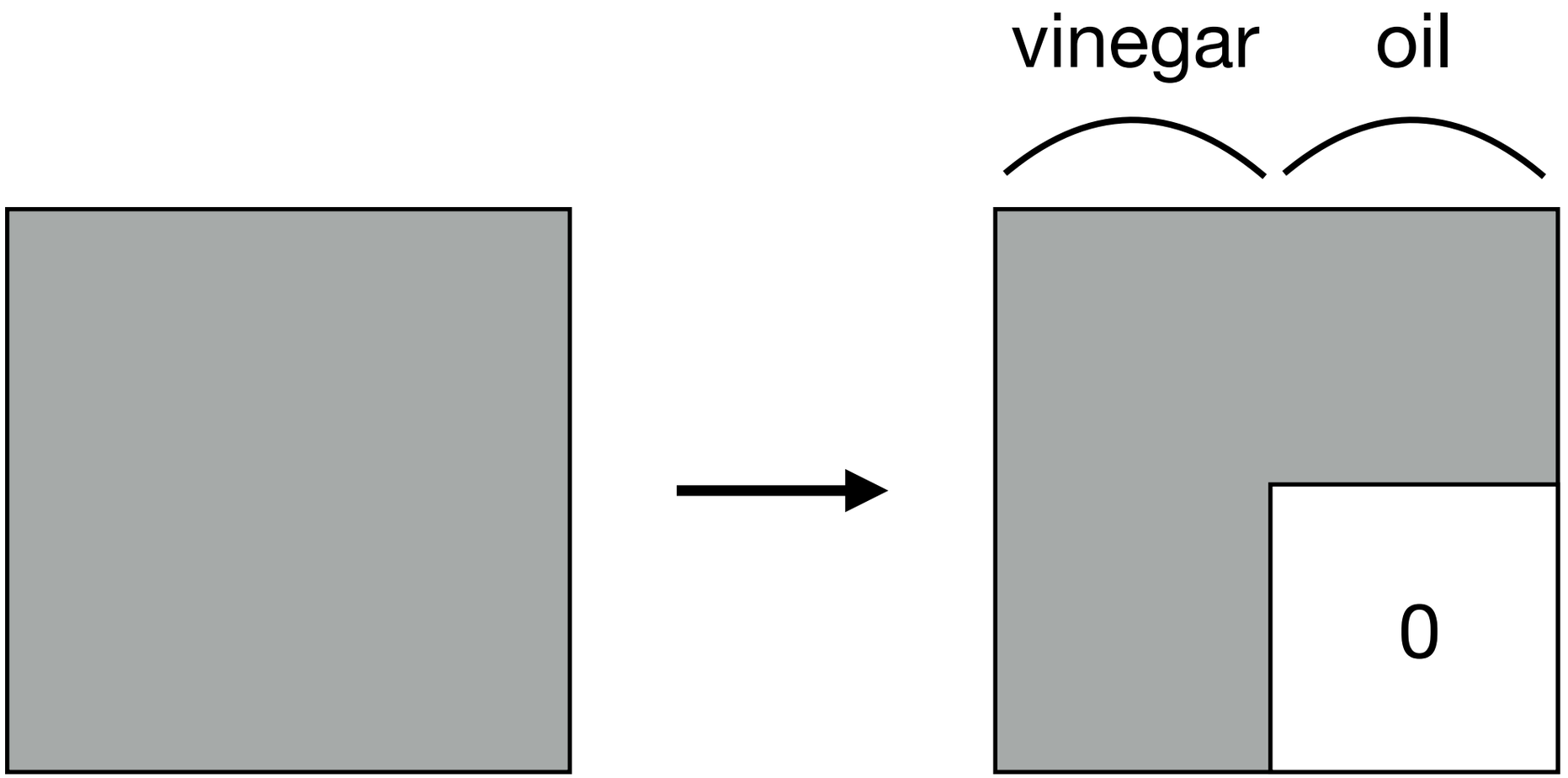
$$w^2 + 4wx + 3x^2 + 2wy - 4wz + 2wz + 6xz$$

$$= \begin{pmatrix} w & x & y & z \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 & 1 \\ 2 & 3 & 3 & -2 \\ 1 & 3 & 0 & 0 \\ 1 & -2 & 0 & 0 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix}$$

And say we magically know the value of variables in the top left quadrant, e.g. $w = 1$, $x = 1$, then the equation becomes linear:

$$1 + 4 + 3 + 2y - 4z + 2z + 6z$$

In pictures

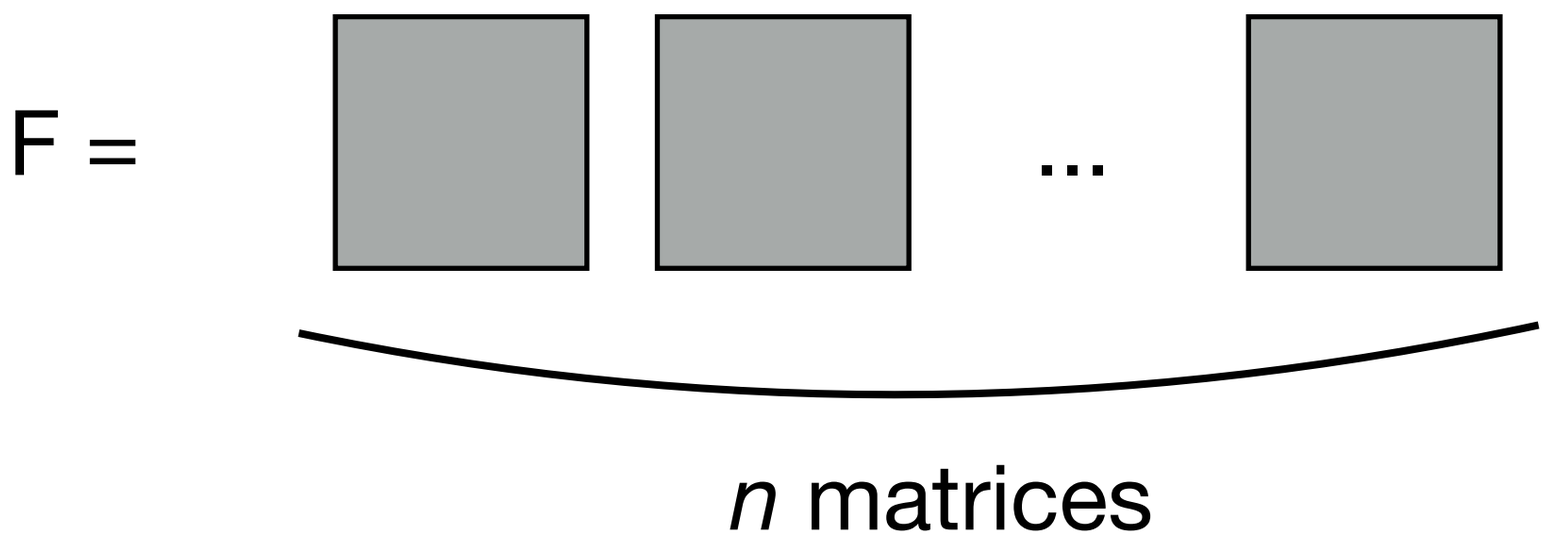


General matrix

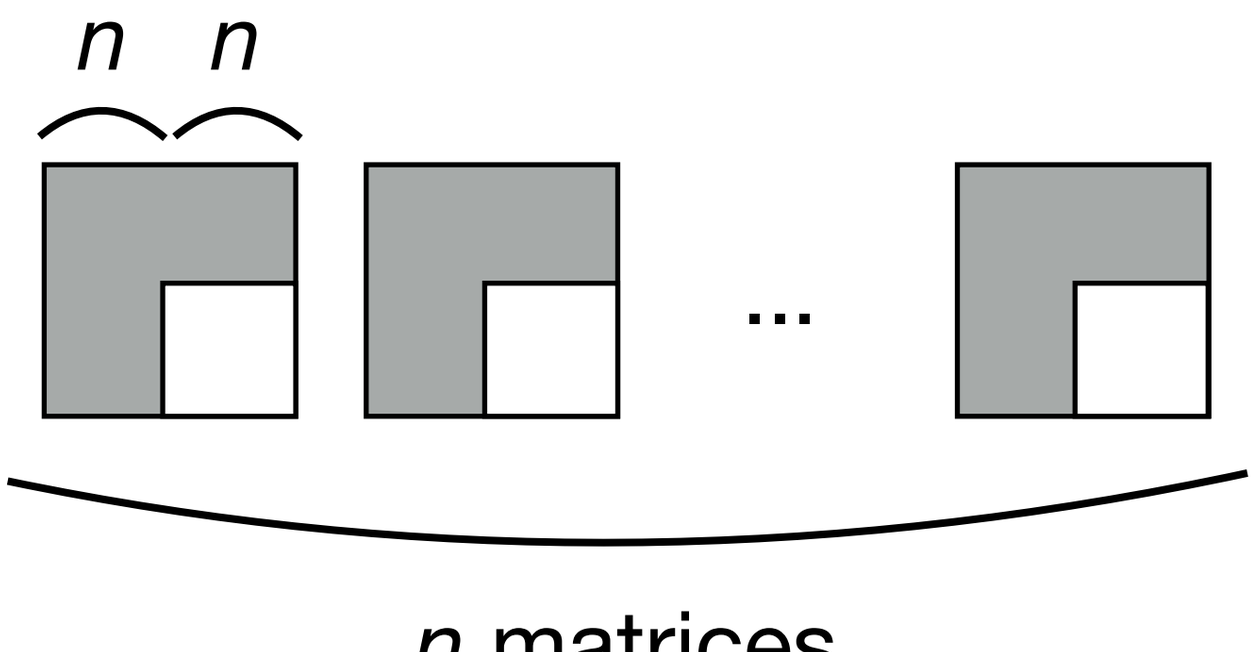
Trapdoored matrix

Quadratic polynomials

So a quadratic function $\mathbb{F}_q^m \to \mathbb{F}_q^n$ may be represented by a sequence of *n* square *m*×*m* matrices:



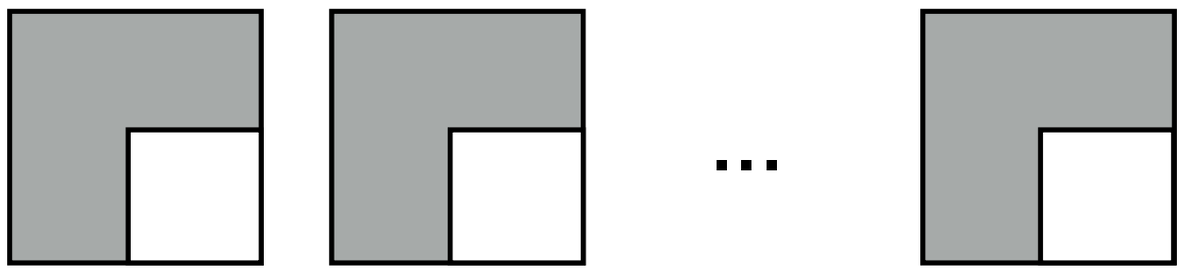
Trapdoor

F = 

Hard problem: given F(*x*) for uniform *x*, find *x*.

Additional trick: 2*n* input variables, *n* output variables. → vinegar variables can be picked freely.

Hiding the trapdoor

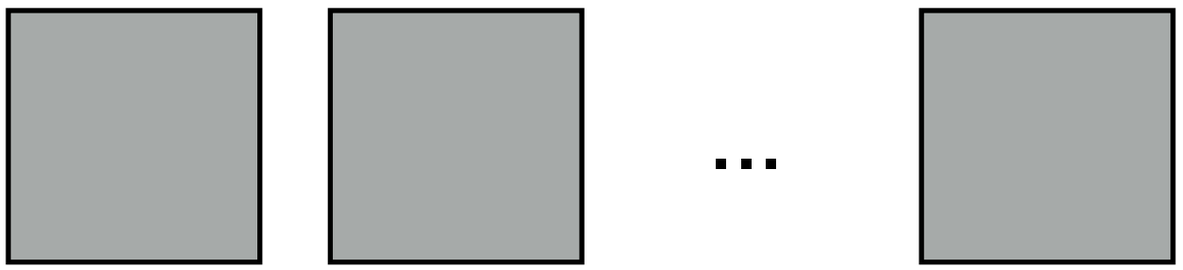
F = 

M_1 M_2 ... M_n

Just do a change of basis!

$$M'_i \leftarrow S^{\mathsf{T}} M_i S$$

for a random invertible matrix S.

F' = $S^{\mathsf{T}}FS$ = 

M'_1 M'_2 ... M'_n

A multivariate signature scheme

- The secret key is F = (M_1,M_2,...,M_n).
- The public key is F' = (M'_1,M'_2,...,M'_n) for $M'_i = S^T \cdot M'_i \cdot S$.

Signature: hash the message m into h = hash(m):

$$s = \text{sign}(m) = F'^{-1}(h)$$

Verification: for signature s for message m with h = hash(m), check:

$$F'(s) = h$$

So signing = inverting F'.

What is the underlying hard problem(s)?

Underlying hard problem

Underlying hard problem(s):

- It is hard to distinguish F' from a random system of quadratic equations.
- It is hard to invert a system of random quadratic equations (MQ).

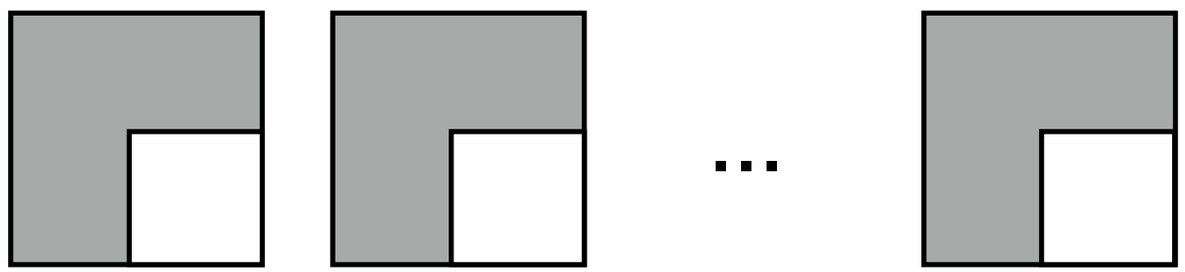
Sometimes described as "reducing" to MQ...

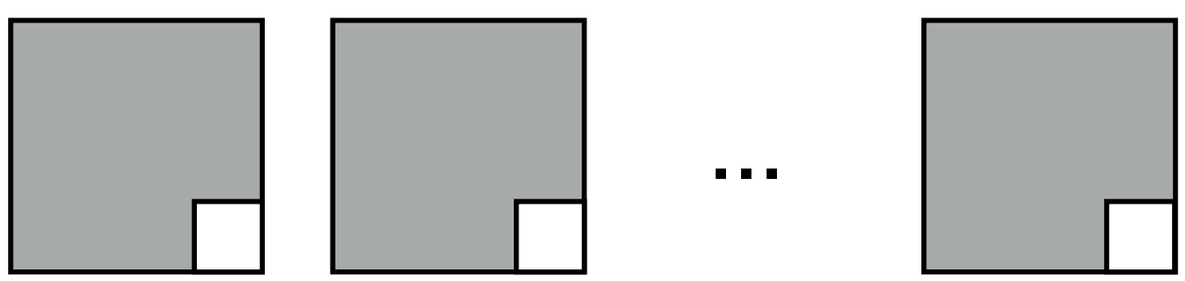
Warning: whether the first problem is hard is highly dependent on how F' is generated.

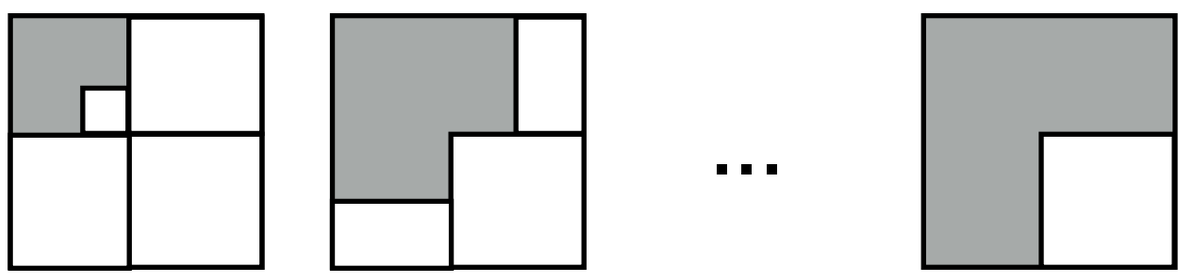
Some history

Oil-and-Vinegar as described was broken by Kipnis and Shamir.

Several fixes :

F =  Oil and Vinegar

F =  Unbalanced Oil and Vinegar

F =  Rainbow

Multivariate crypto: conclusion

\+ Fast secret key operations.
Small signatures/ciphertexts.
Cheap encryption.

\- Public key = F' → large public key sizes (up to 1 Mb).
Heuristic security reduction.
Not a high level of confidence in security.

Was considered mostly dead until post-quantum cryptography came along.

Now trying to gain credibility in terms of security.

Hash-based Signatures

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.

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 $h(x_b) = y_b$.

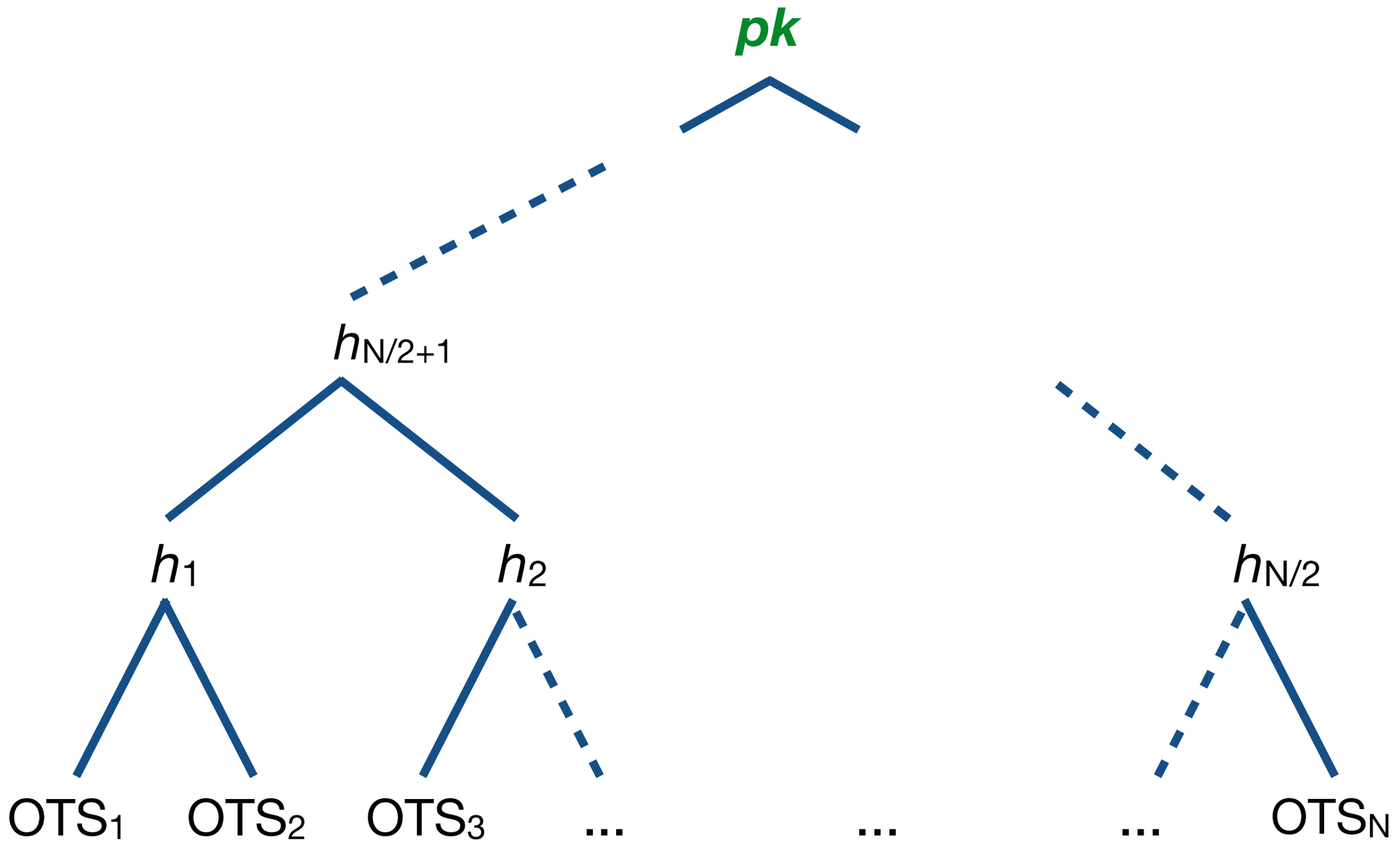
Lamport signature

This can be extended to multiple bits by using multiple copies of the scheme.

There are more efficient schemes for multiple bits (Winternitz signatures), but we shall skip them here.

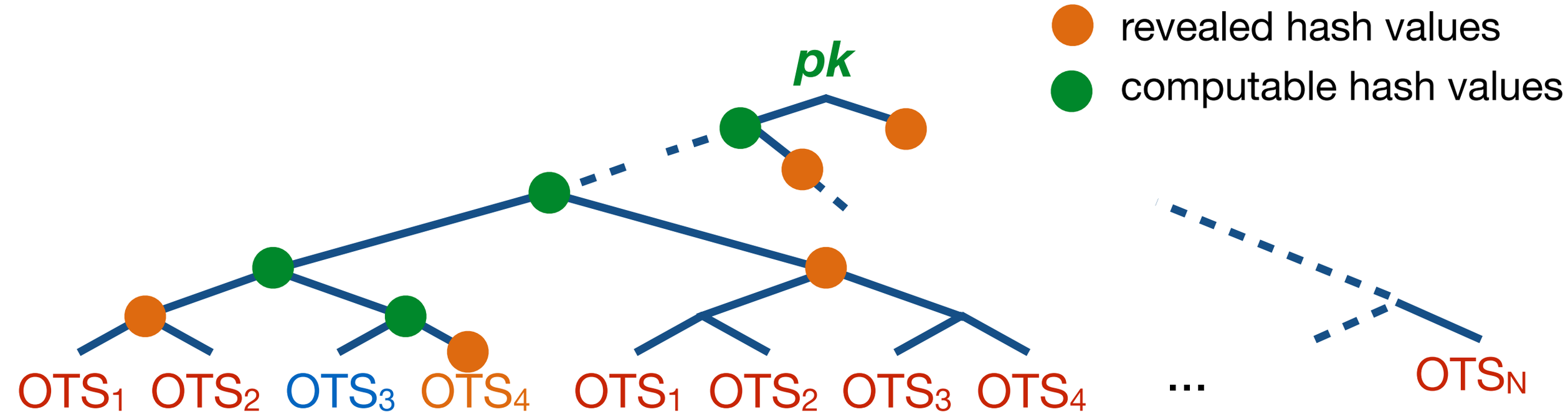
Next challenge: how to go from one-time signature to many-time signature?

Solution 1: Merkle trees



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

Solution 1: Merkle trees



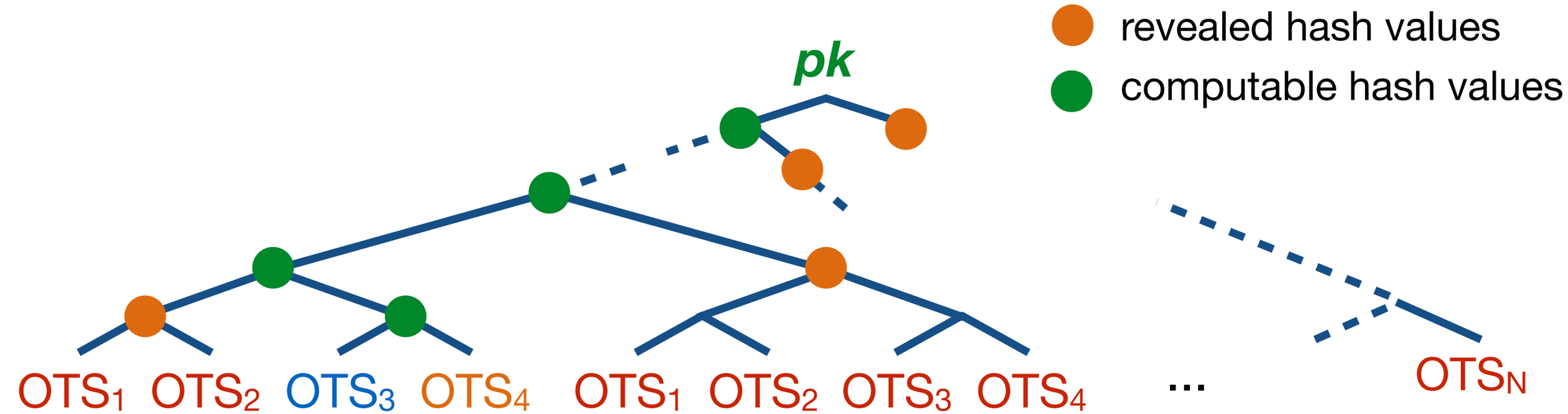
- The secret key is sk = (OTS_1,OTS_2,...,OTS_N).
- The public key is the root of the tree *pk*.

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:

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

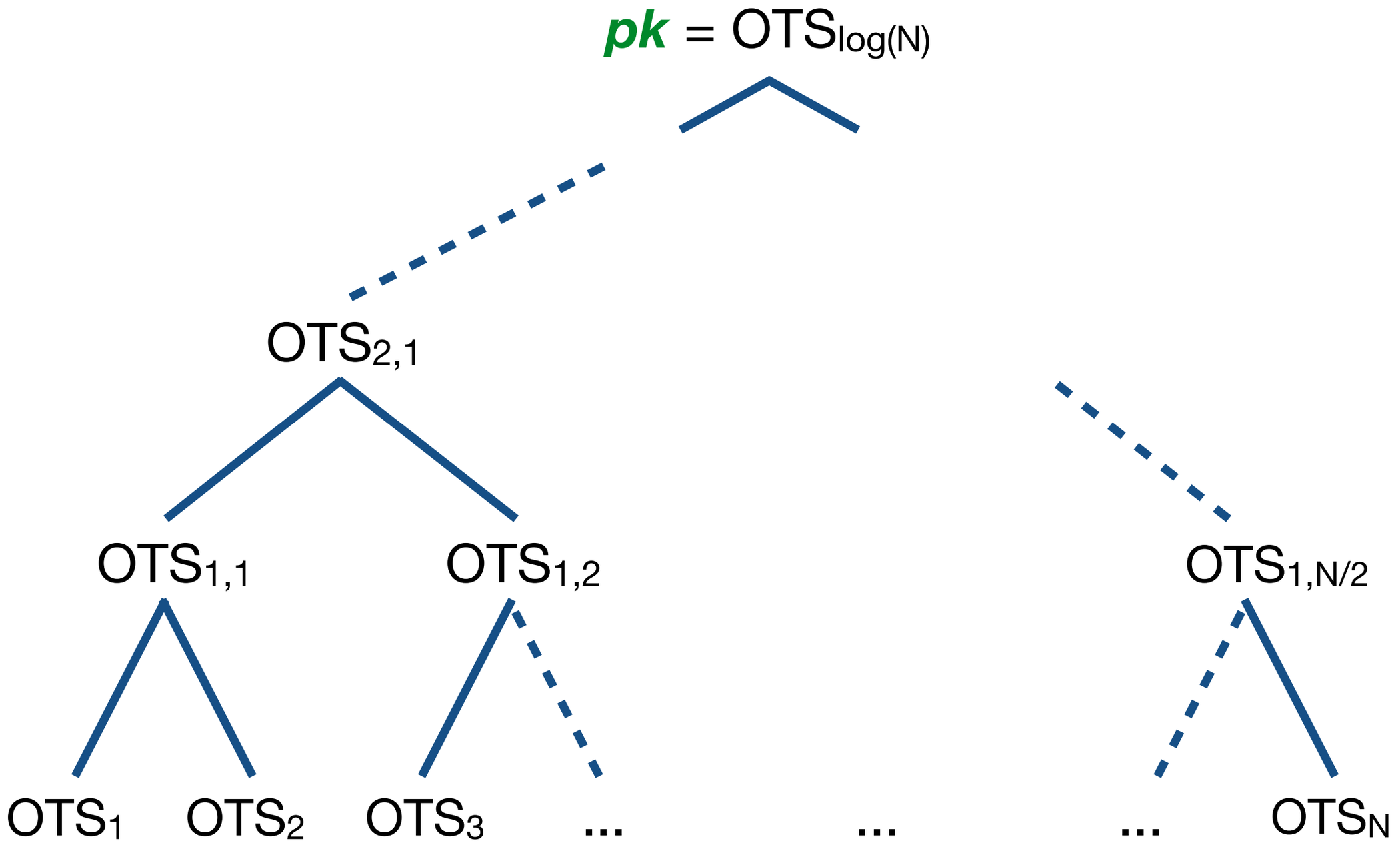
Verification: check the OTS_i signature, and all hashes.

Solution 1: Merkle trees



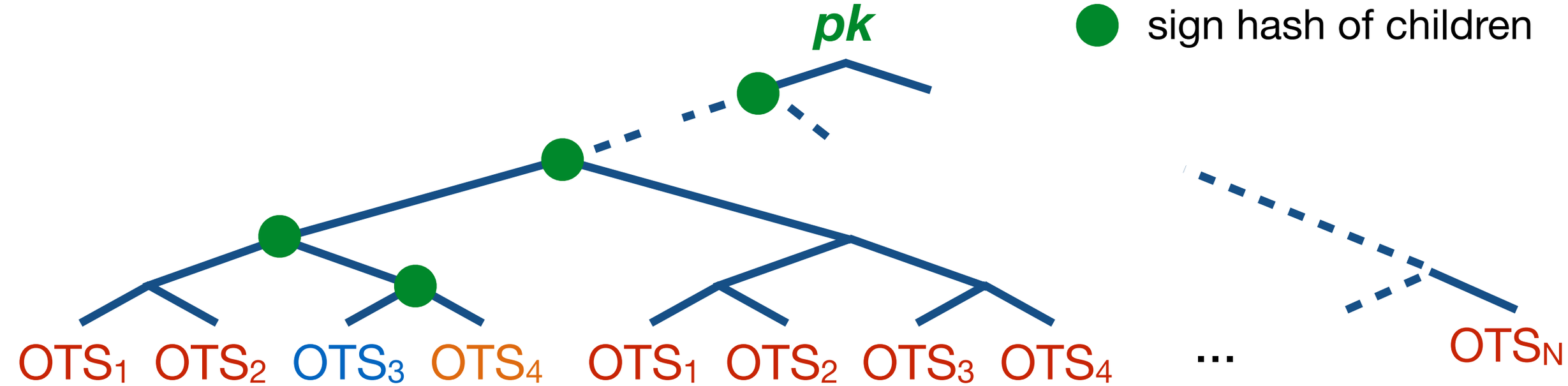
- Can sign up to N messages.
- Signatures are length $O(\log(N))$.
- Needs a state to store which OTS_i is next to be used.
- **Problem**: need $O(N)$ precomputation to get *pk*!

Solution 2: Goldreich scheme



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

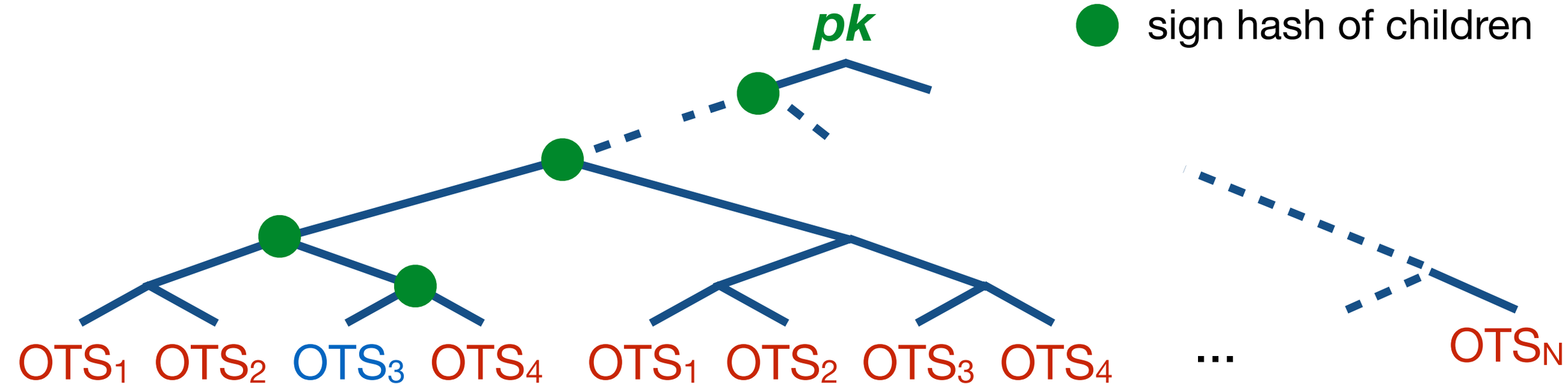
Solution 2: Goldreich scheme



- 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.

SPHINCS

XMSS: Merkle trees are used as nodes within a Goldreich scheme.

SPHINCS: add some other tricks to get rid of the state.

Other hash-based signature schemes: from Zero Knowledge and Multi-Party Computation.

Hash-based crypto: conclusion

\+ Minimalist assumption.
High level of confidence in security.

\- Large signatures (10s of Kb!).
Slow signatures (10000s hashes!).
Only signatures.

Interest from industry due to high security level.

Conclusion

Different trade-offs.

Lattices are the mainstream solution. Well-studied and good parameters in general. Main candidate.

Other solutions offer different trade-offs. **Code-based** and **hash-based** schemes are the most credible alternatives.

Interesting side-effect: has given new purpose to old schemes that were otherwise outpaced in a classical setting.