Post-Quantum Cryptography

Brice Minaud

email: brice.minaud@ens.fr
website: www.di.ens.fr/brice.minaud/init-crypto.html

Initiation à la Cryptologie, ENS/MPRI, 2019-2020

Meta information

Exam: Monday, May 25, 2pm to Wednesday 27, 5pm.

Register here:

https://www.di.ens.fr/david.pointcheval/cours.html

All other info for this course, including past lectures/TAs:

https://www.di.ens.fr/brice.minaud/init-crypto.html

(The diff with last week's information is in red.)

Quantum computing

New model of computation. Computes on superposition of *n*-bit strings.

Grover algorithm: given arbitrary (efficient) $F: \{0,1\}^n \rightarrow \{0,1\}$, find x such that $F(x) = 1$ in time $O(2^{n/2})$.

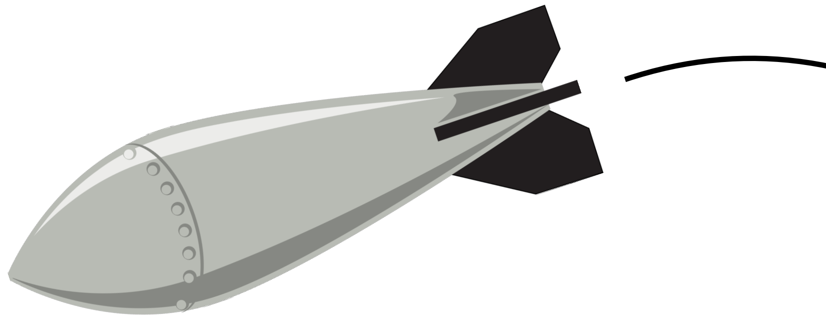
Shor algorithm: factors integer and computes discrete logarithm in polynomial time.

Fine print: existence of efficient quantum computers in the forseeable future is still up for debate. “Weak” forms of quantum computing already exist.

Post-Quantum crypto

Today, most of public-key crypto is based on hard problems arising from number theory:

- Integer Factorization *RSA*
- Discrete Log *Diffie-Hellman, ElGamal…*
- Elliptic Curves *ECDSA, pairing-based crypto, including most SNARKs…*



Broken in **quantum polynomial time** by Shor's algorithm.

In short: efficient quantum computers ⇒ global crypto catastrophe.

(Caveats apply.)

→ Need to anticipate. To have solutions ready, + forward security.

→ Ongoing NIST-organized "selection process" (a.k.a. competition) to define new post-quantum standards.

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 crypto

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

Hard problems in post-quantum world

Post-quantum candidate hard problems:

- Lattices.
- Code-based crypto.
- Isogenies.
- Symmetric crytpo ($\rightarrow$ signatures).
- Multivariate crypto.

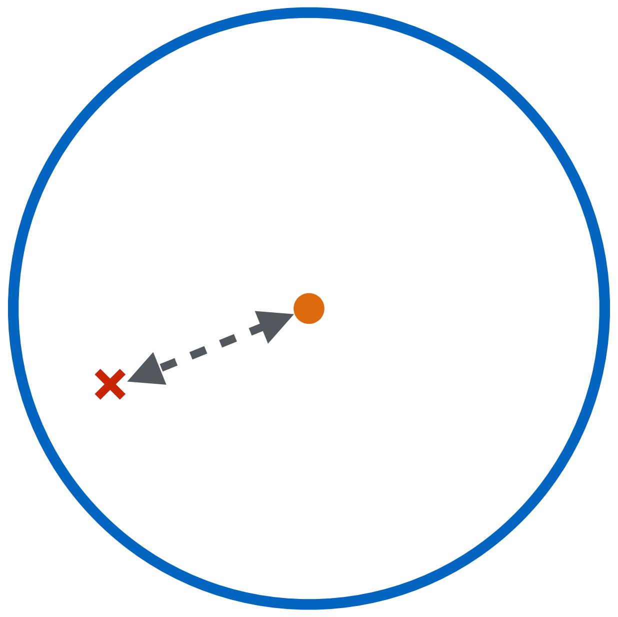
Lattices are the mainstream candidate. Other PQ approaches for Public-Key crypto "only" motivated by PQ. Lattice-based crypto stands on its own:

- Simplicity (of schemes, not analysis).
- Security from worst-case hardness.
- Very expressive/verstatile, much beyond PKE/sig.

Number Theory

Lattices, codes,…
(conjectured)

Error-correcting codes



(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.

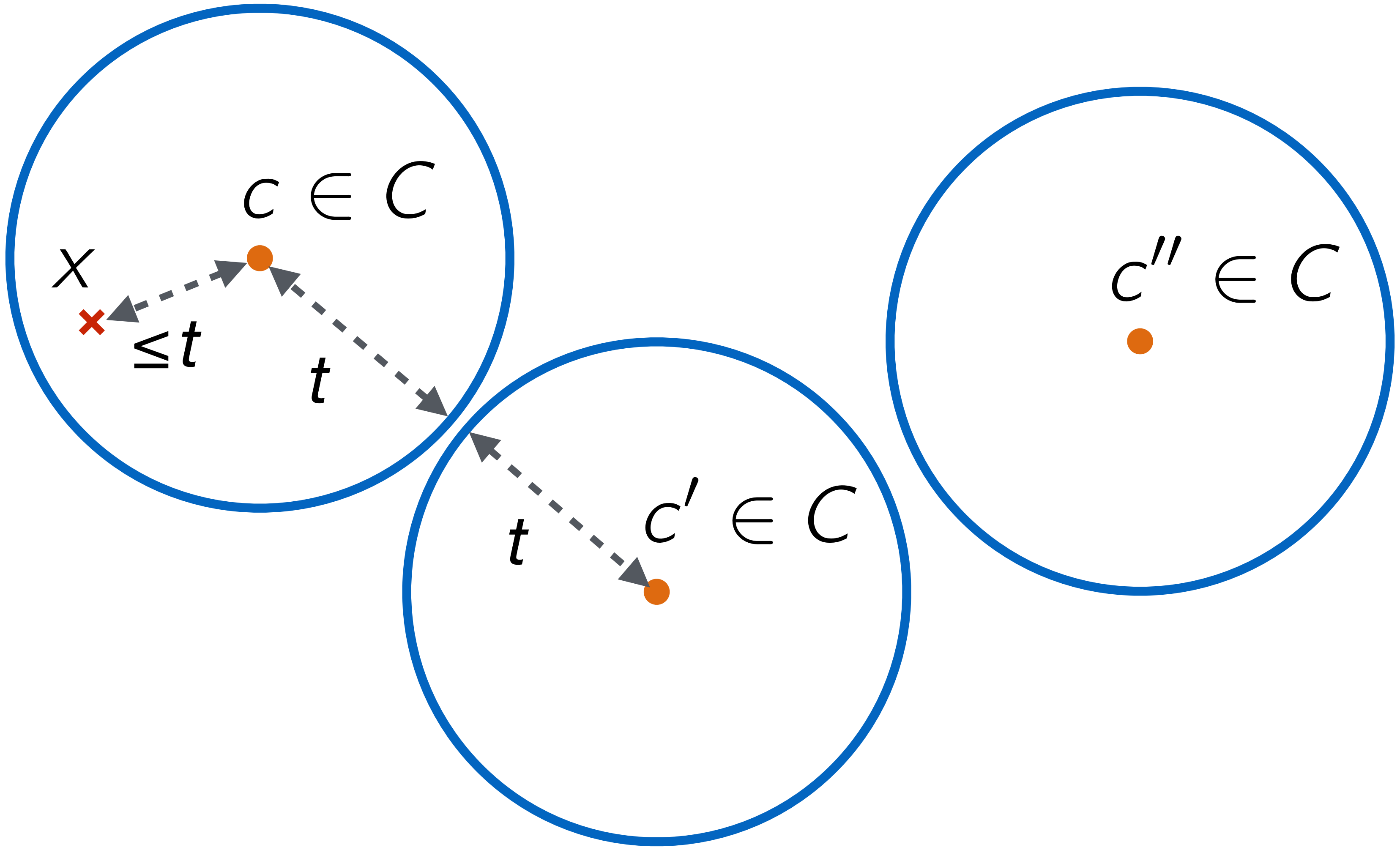
hw(010011) = 3

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* = 2*t* + 1, then *C* can correct up to *t* errors.

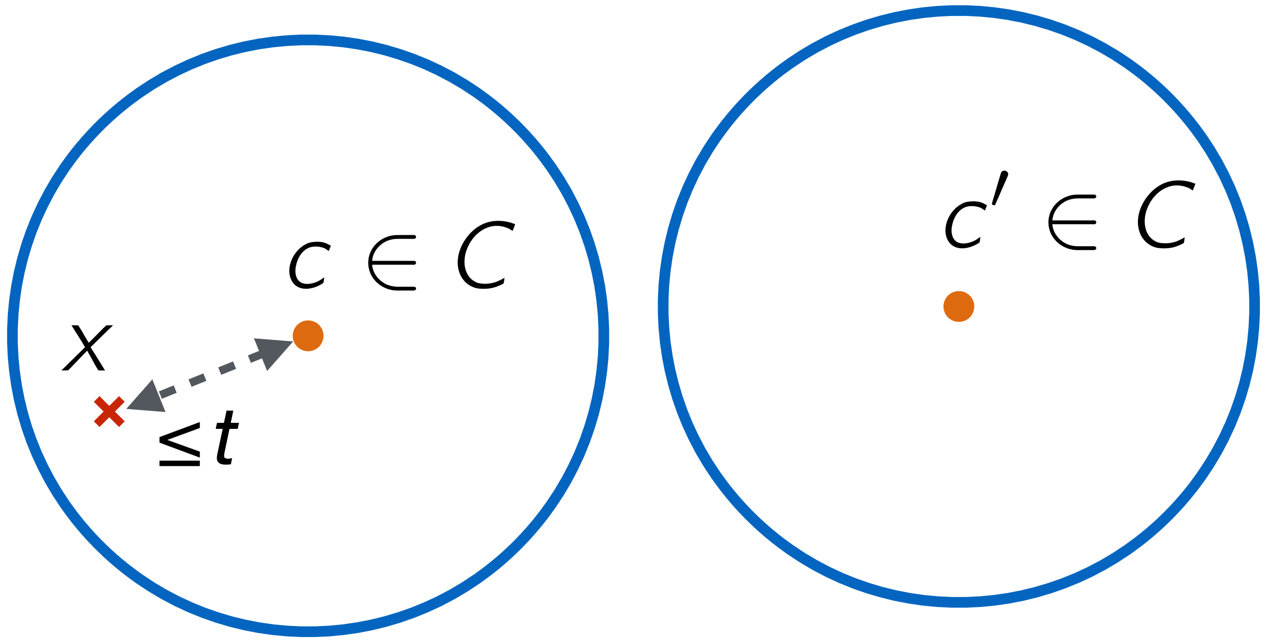
- That is, for $x \in \mathbb{F}_q^n$, if dist(*x*-*c*) ≤ *t* for some $c \in C$, then *c* is unique.



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 dist(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

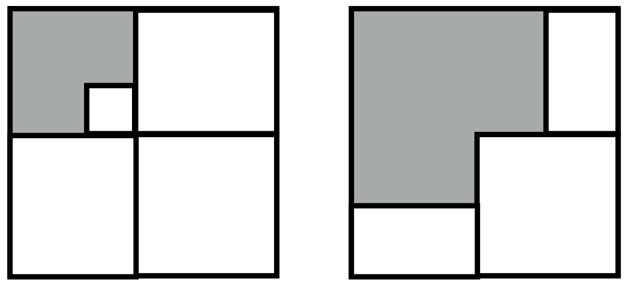
The original McEliece, using binary Goppa codes, is essentially unbroken since 1978.

> 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.

Multivariate Cryptography

F =  ...

Multivariate cryptography

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

→ How to get an encryption scheme:

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

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

Signature of message *Y* is *X* such that *Y* = **F**(*X*).

+: small signature, 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

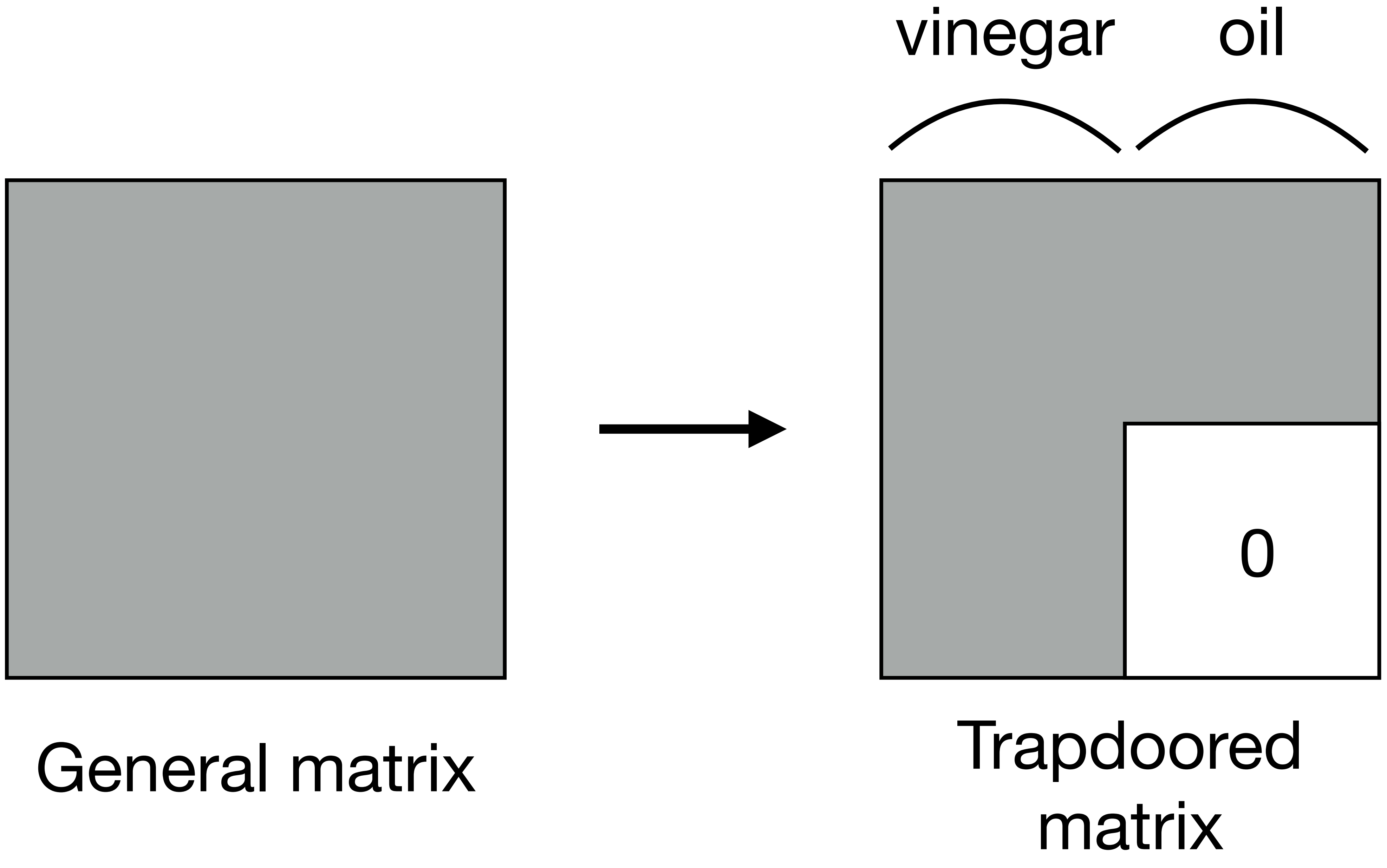
Say the bottom right quadrant of the matrix is zeros...

$$w^2 + 4wx + 3x^2 + 2wy - 4wz + 2wz + 6xz$$
$$= \begin{pmatrix} w & x & y & z \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 & 1 \\ 2 & 3 & 3 & -2 \\ 1 & 3 & 0 & 0 \\ 1 & -2 & 0 & 0 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix}$$

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:

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

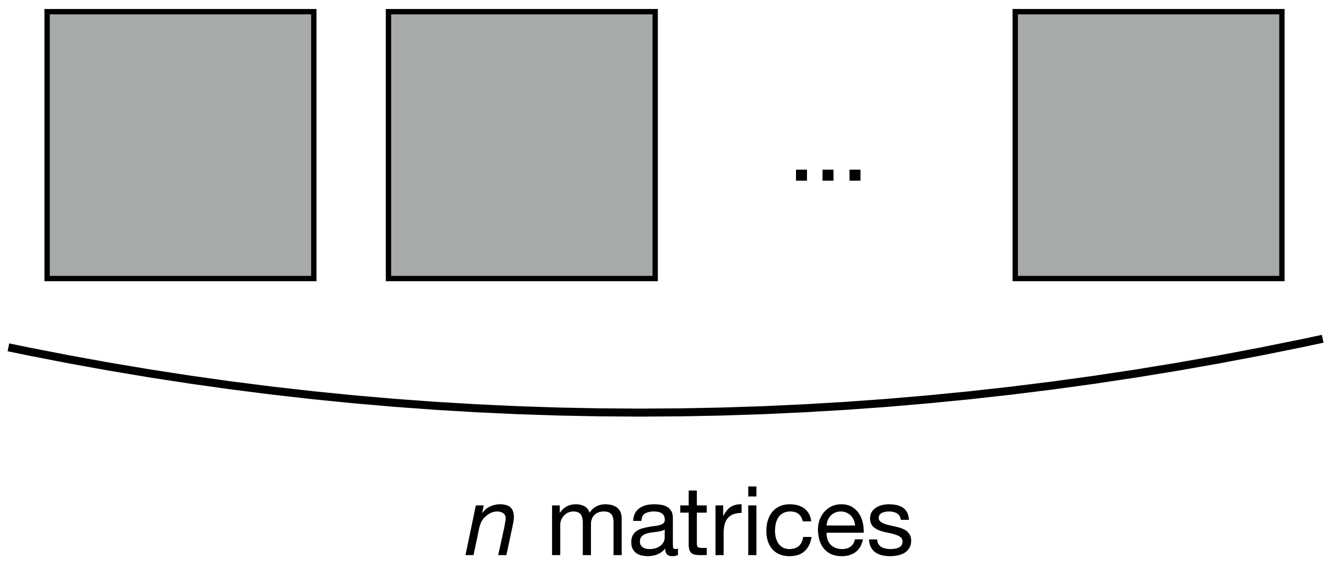
In pictures



Quadratic polynomials

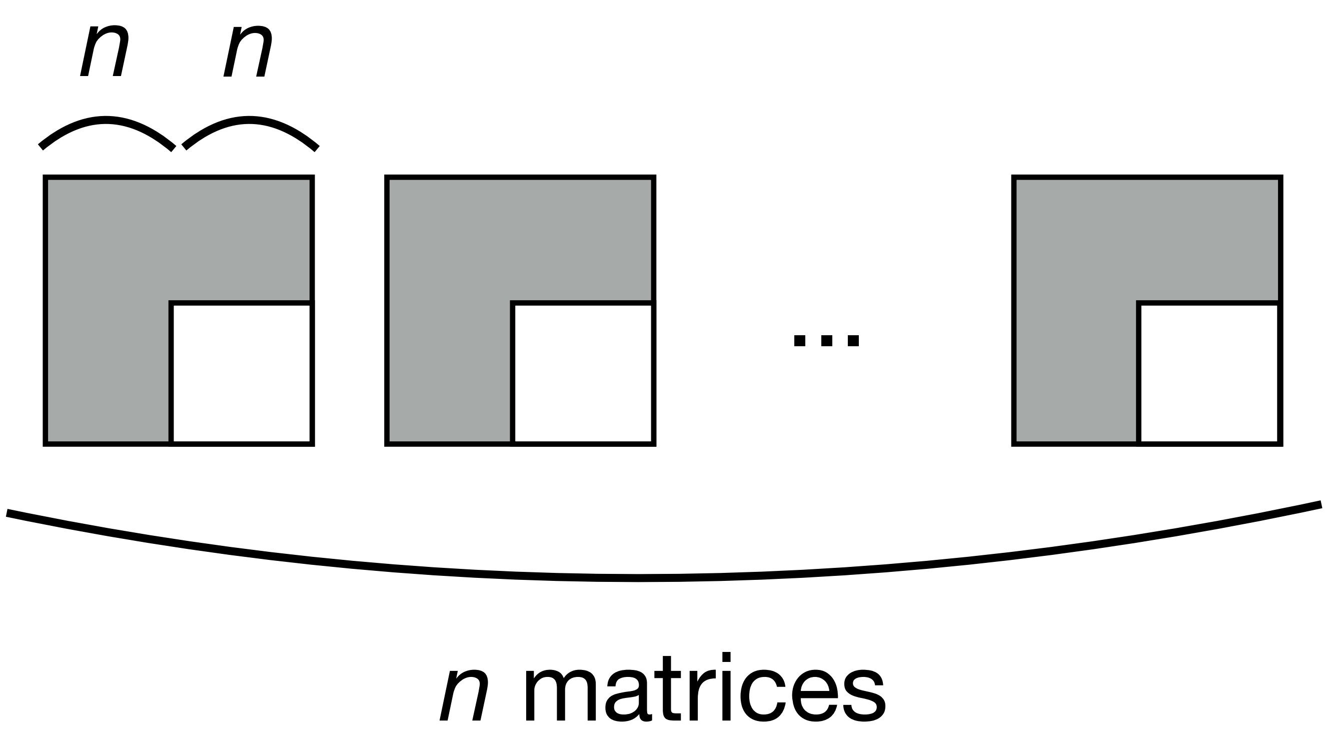
So a quadratic function $\mathbb{F}_q^m \to \mathbb{F}_q^n$ may be represented by a sequence of *n* square *m*×*m* matrices:

F =



Trapdoor

F =

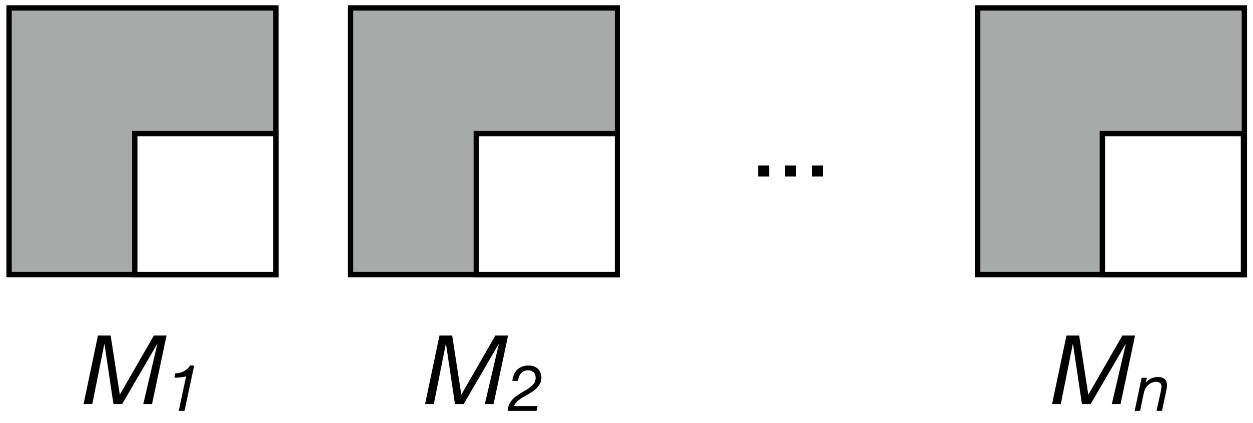


Hard problem: given F(*x*) for uniform *x*, find *x*.

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

Hiding the trapdoor

F =

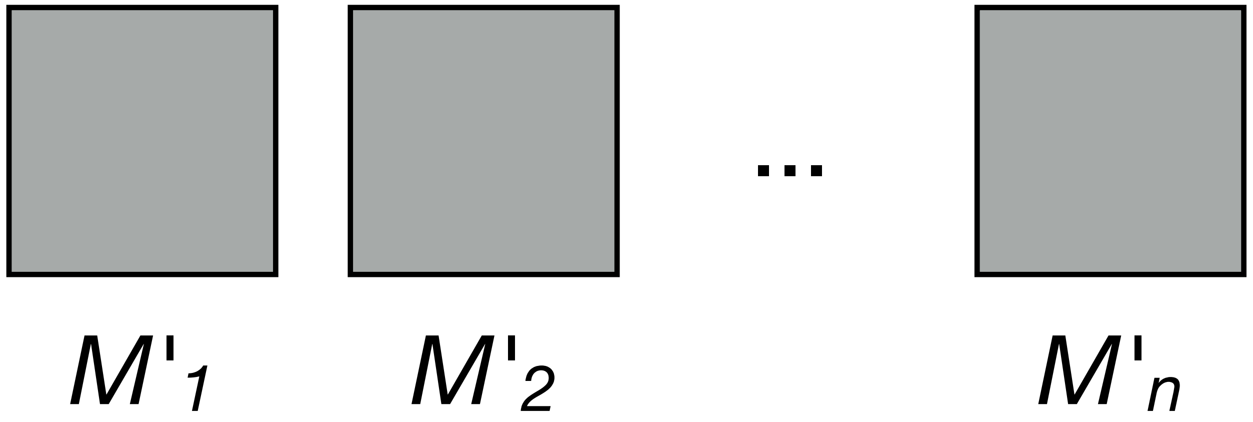


Just do a change of basis!

$$M_i' \leftarrow S^\top M_i S$$

for a random invertible matrix *S*.

F' = S^TFS =



A multivariate signature scheme

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

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

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

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...

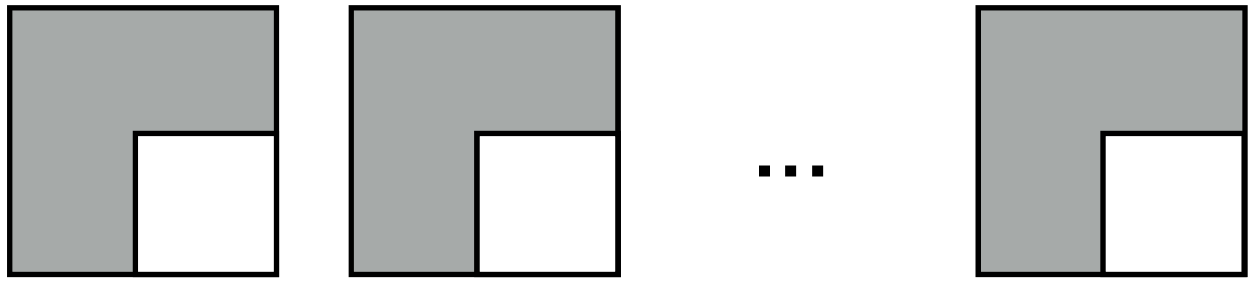
Warning: whether the first problem is hard is highly dependent on how *F*' is generated.

Some history

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

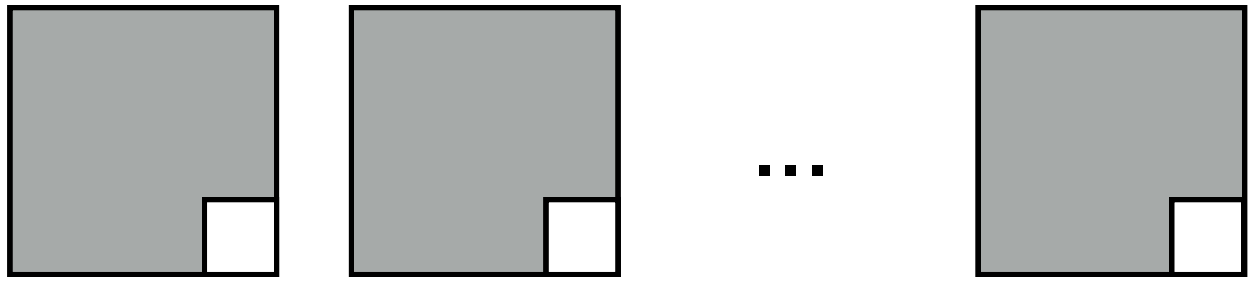
Several fixes :

F =



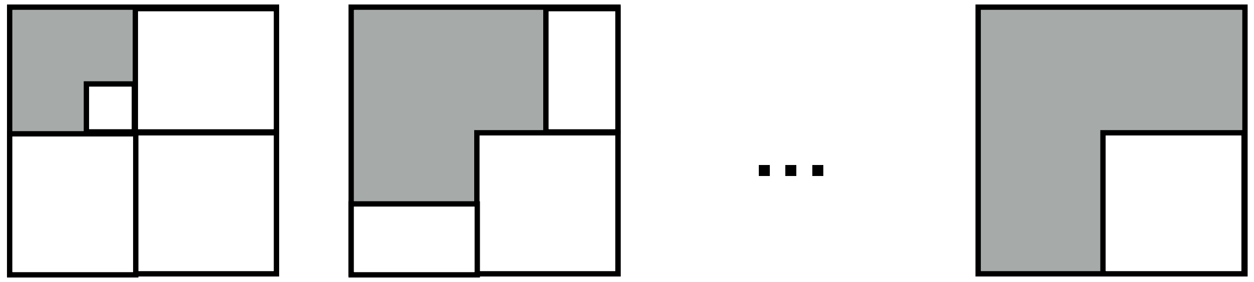
Oil and Vinegar

F =



Unbalanced Oil and Vinegar

F =



Rainbow

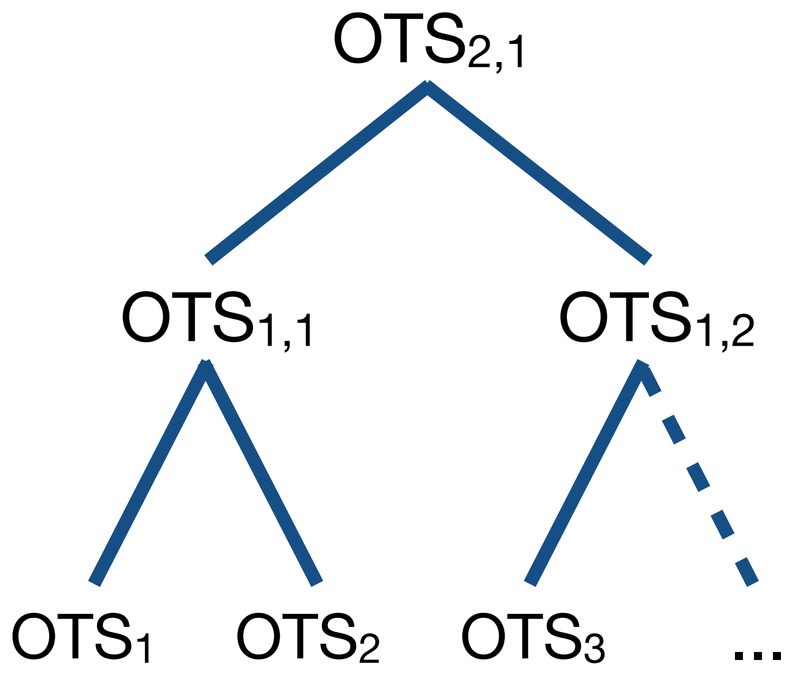
Multivariate crypto: conclusion

+	Fast secret key operations. Small signatures/ciphertexts. Cheap encryption.
-	Public key = F' $\rightarrow$ 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 $h(x_b) = y_b$.

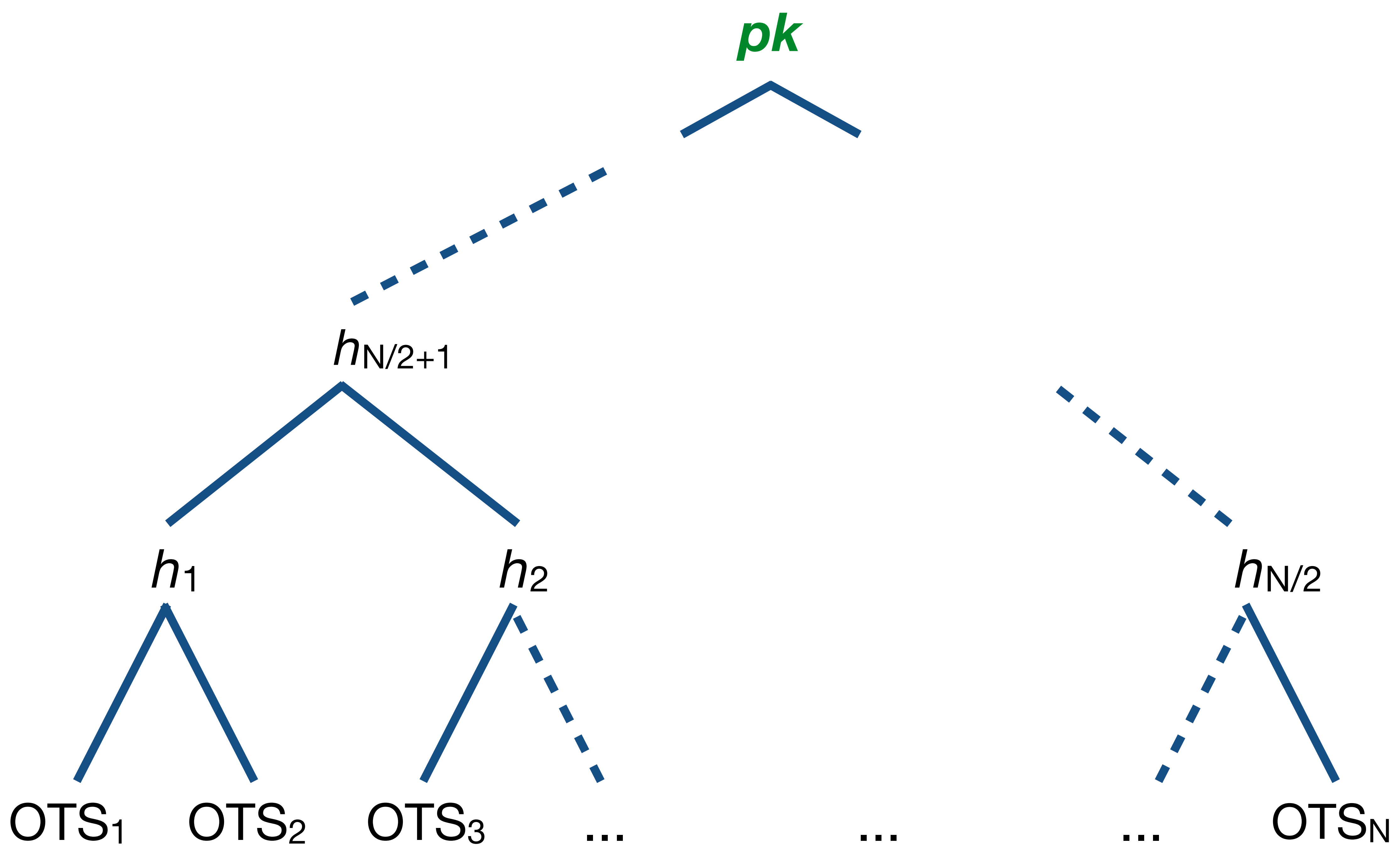
Lamport signature

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.

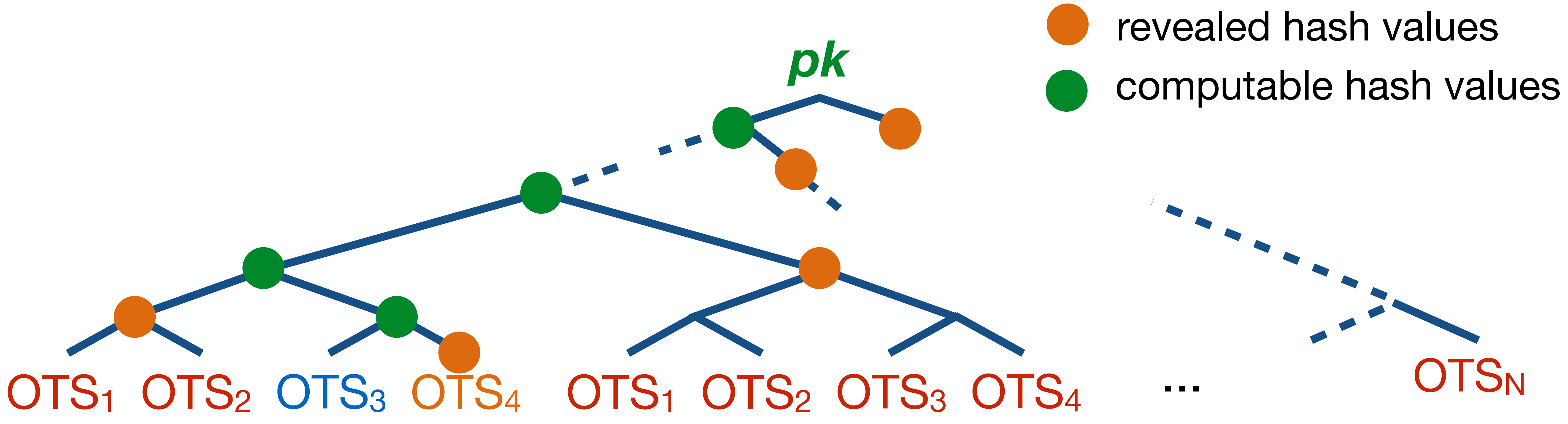
Next challenge: how to go from one-time signature to many-time signature?

Solution 1: Merkle trees



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

Solution 1: Merkle trees



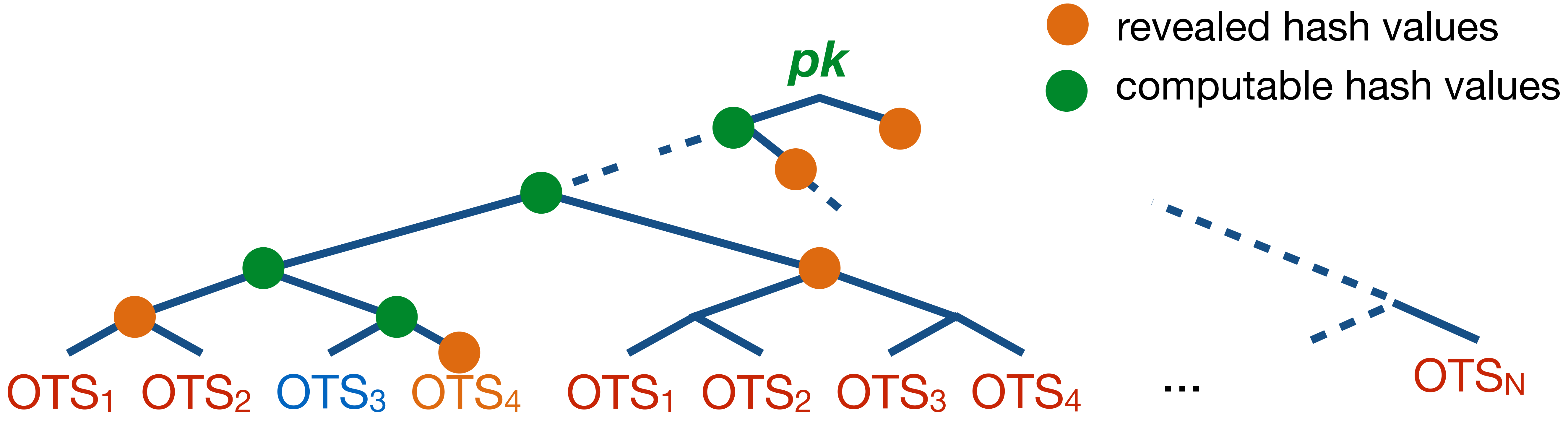
- The secret key is sk = (OTS_1,OTS_2,...,OTS_N).
- The public key is the root of the tree *pk*.

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:

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

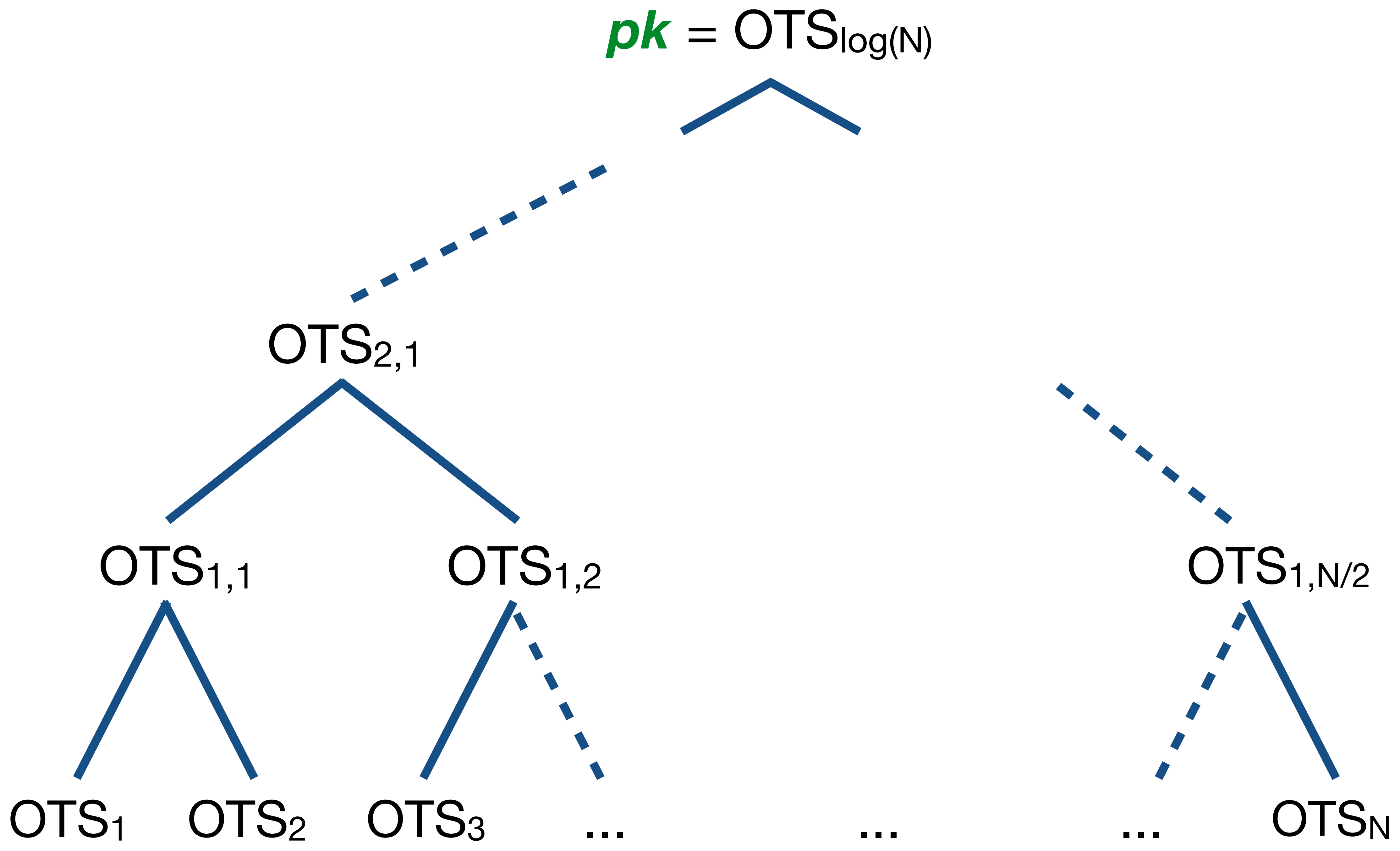
Verification: check the OTS_i signature, and all hashes.

Solution 1: Merkle trees



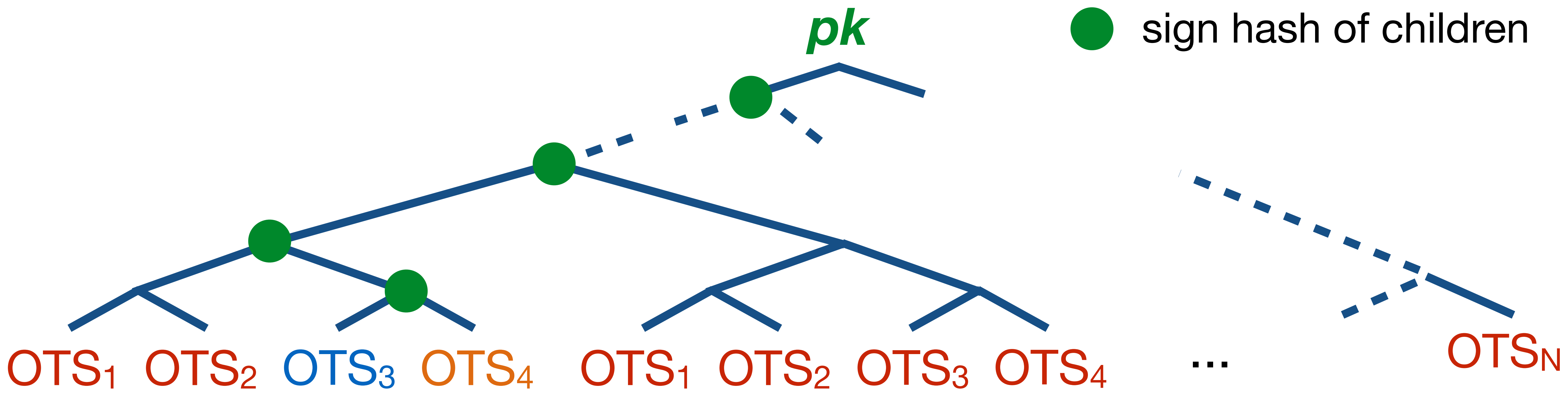
- Can sign up to N messages.
- Signatures are length $O(\log(N))$.
- Needs a state to store which OTS_i is next to be used.
- **Problem**: need $O(N)$ precomputation to get *pk*!

Solution 2: Goldreich scheme



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

Solution 2: Goldreich scheme

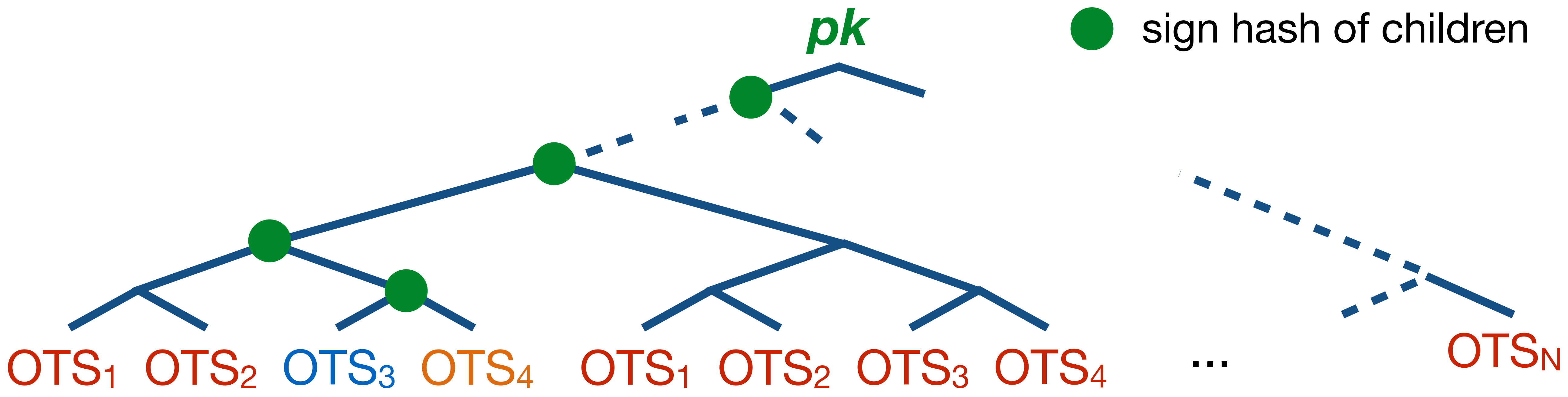


pk
● sign hash of children
OTS_1 OTS_2 OTS_3 OTS_4 OTS_1 OTS_2 OTS_3 OTS_4 … OTS_N

- 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



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

- 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.

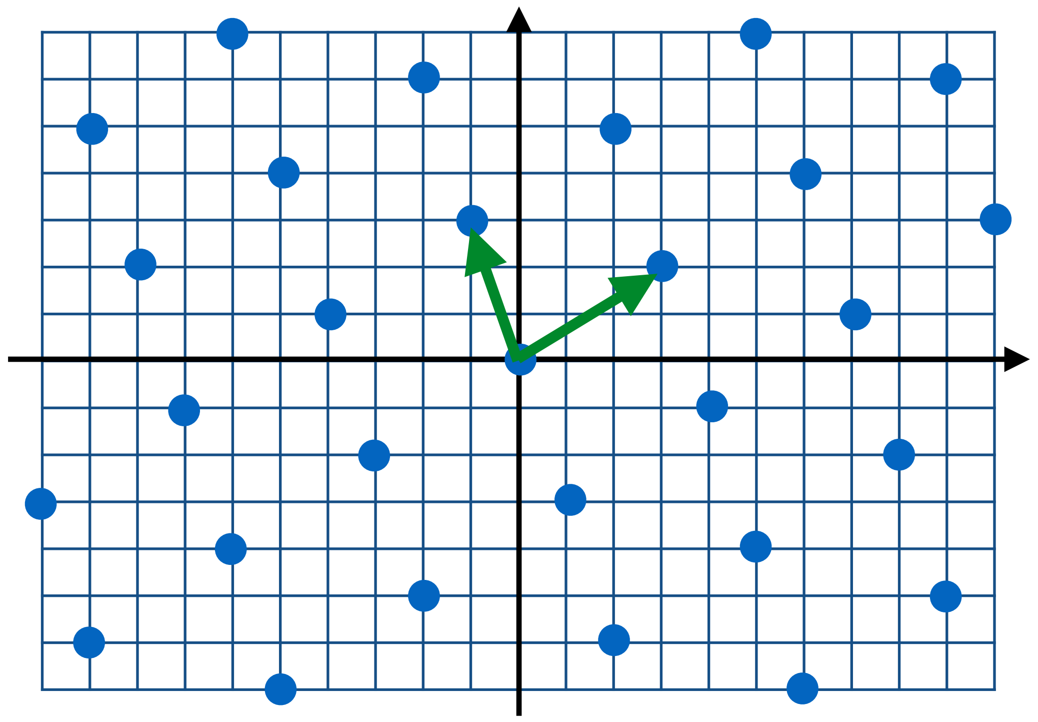
SPHINCS

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

SPHINCS: add some other tricks to get rid of the state.
v2 currently in NIST competition.

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

Lattices



Lattices

Lattice. A lattice $\mathscr{L}$ is:

- An additive subgroup of $\mathbb{R}^n$.
- Discrete (not dense).

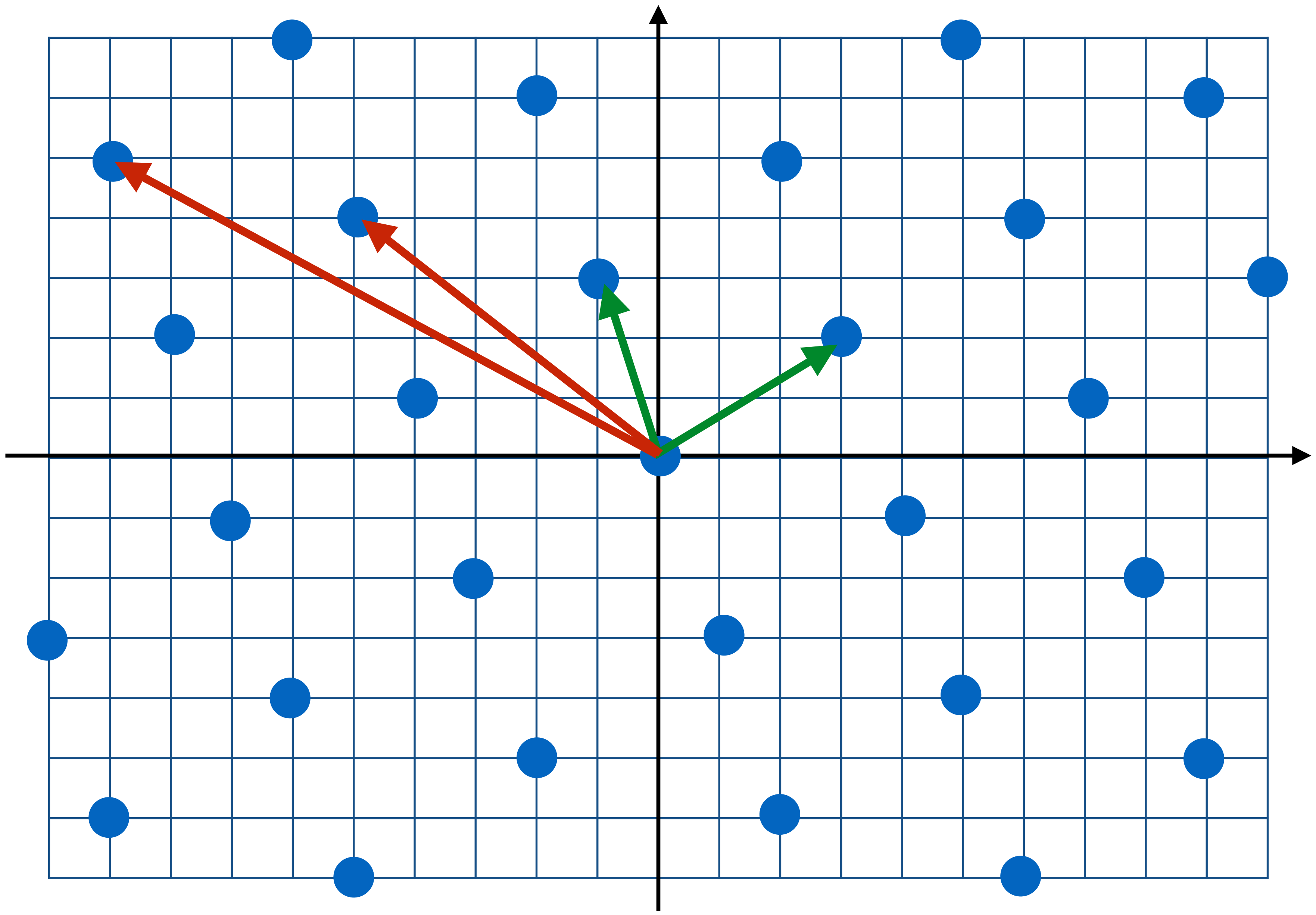
In practice, in crypto, $\mathscr{L}$ often:

- Spans $\mathbb{R}^n$, a.k.a. “full-rank”.
- Typically $\subseteq \mathbb{Z}^n$.
- Often “q-ary”: all $qe_i = (0,\ldots,0,q,0,\ldots,0)$’s are in $\mathscr{L}$. That is, the lattice wraps around mod q. Can be regarded as in $\mathbb{Z}_q^n$.

Concretely, $\mathscr{L}$ can be defined by a basis $B \in \mathbb{Z}^{n \times n}$:

$$\mathscr{L} = B\mathbb{Z}^n$$

In pictures



Basis B.

Basis B'.

Dual lattice

Dual lattice. The **dual** $\mathscr{L}^*$ of a lattice $\mathscr{L} \subseteq \mathbb{R}^n$ is:

$$\mathscr{L}^* = \{x \in \mathbb{R}^n : \forall\, y \in \mathscr{L},\ {}^t x y \in \mathbb{Z}\}$$

Properties of the dual:

- It is a lattice.
- It characterizes the lattice $\mathscr{L}$: $\mathscr{L}^{**} = \mathscr{L}$.
- If B is a basis of $\mathscr{L}$, $({}^t B)^{-1}$ is a basis of $\mathscr{L}^*$.

Hermite Normal Form

A lattice can be charaterized by a basis in **Hermite Normal Form**.

HNF basis is unique and easy to compute from any basis → "neutral" description of the lattice.

Hermite Normal Form. A basis $B \in \mathbb{Z}^{nxn}$ of a (full-rank) lattice is HNF iff:

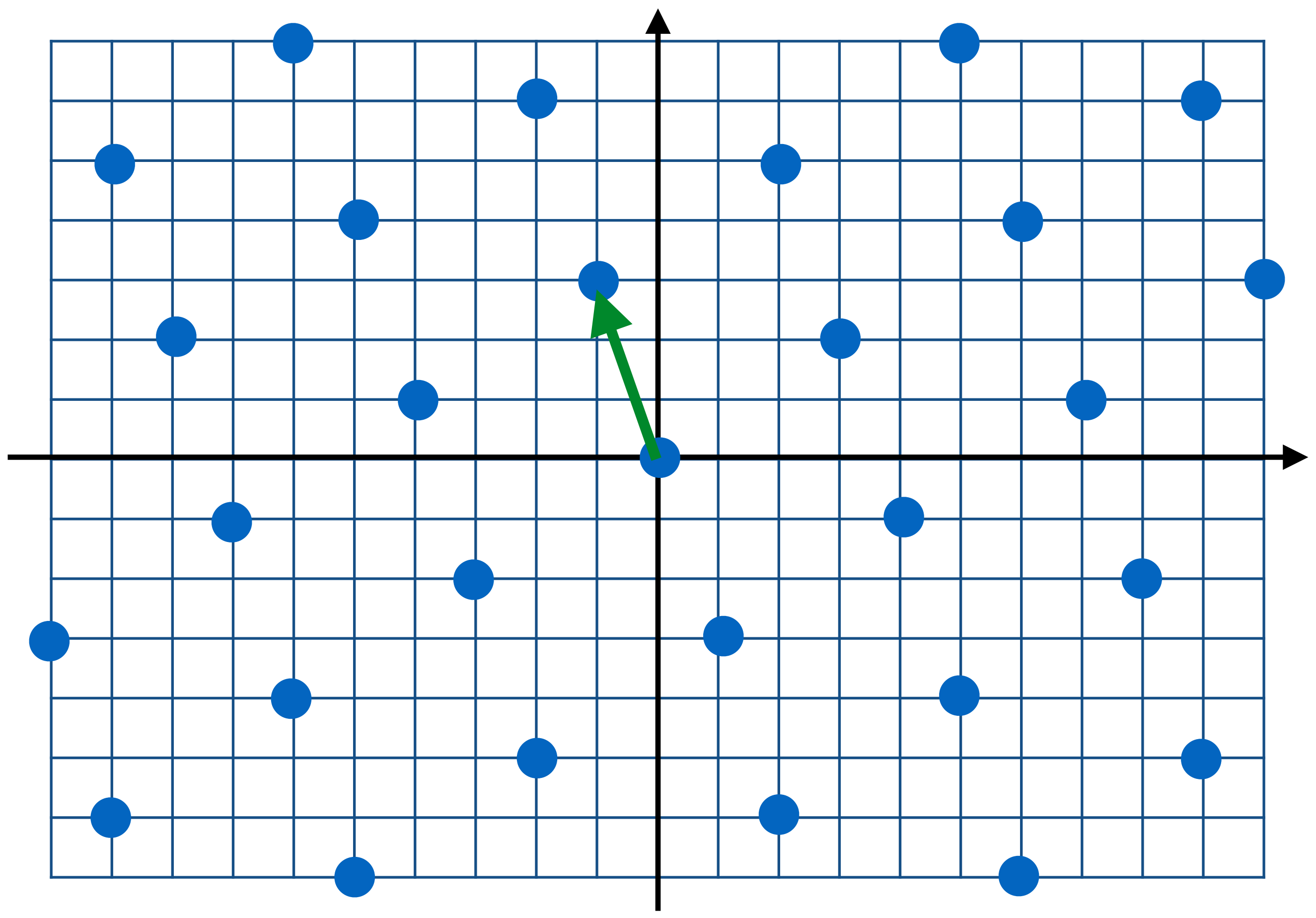
- It is upper triangular, with > 0 diagonal elements.
- Elements to the right of a diagonal element $m_{i,i}$ are ≥ 0 and $< m_{i,i}$.

Hard problems in lattices

Define the usual ℓ^2 norm on $\mathbb{R}^n$.

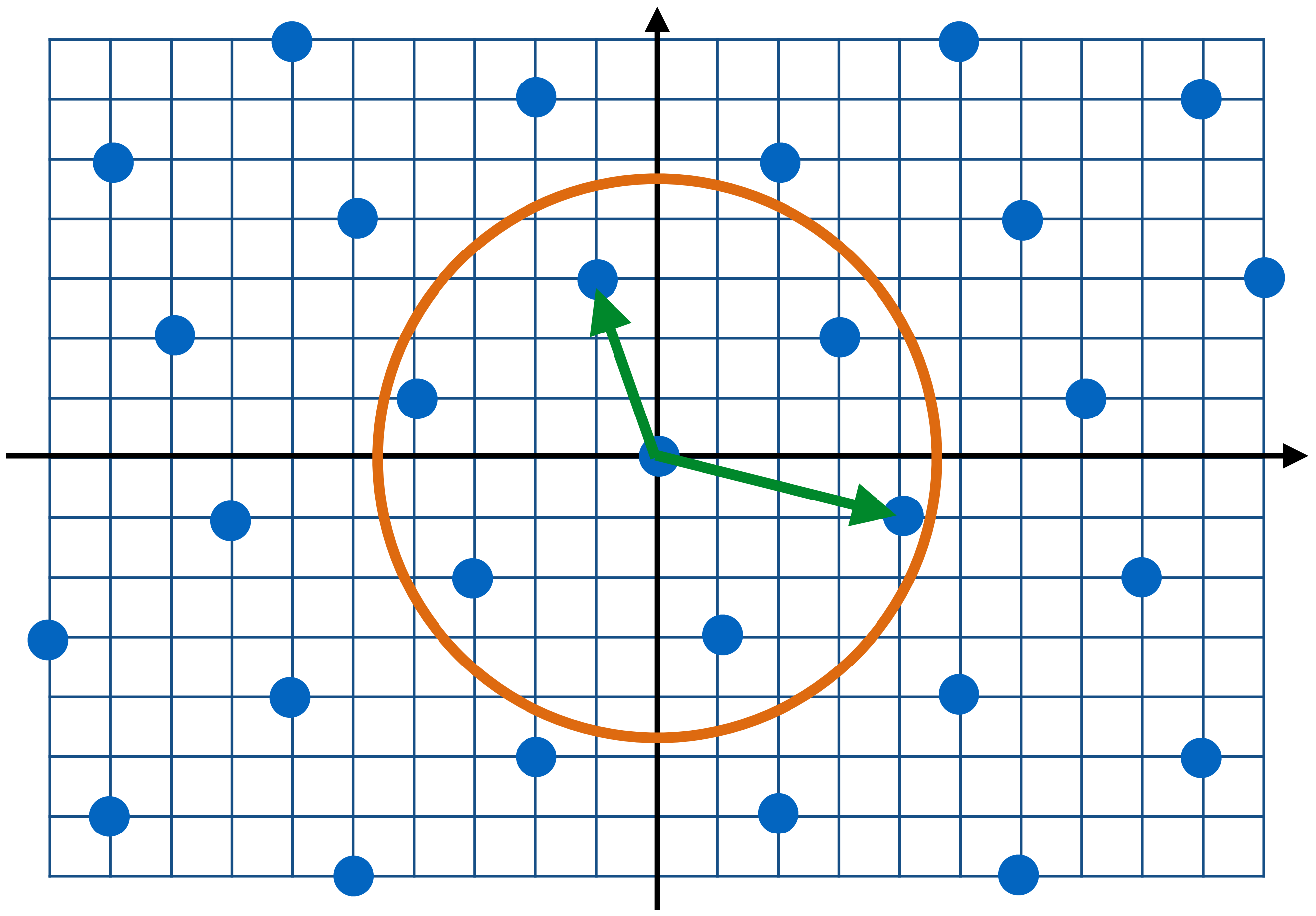
Define $\lambda_i(\mathscr{L})$ to be the smallest vector independent from $\lambda_1(\mathscr{L}), \ldots, \lambda_{i-1}(\mathscr{L})$.

Shortest Vector Problem (SVP). Given a basis B of a lattice $\mathscr{L}$, find the smallest non-zero lattice vector. I.e., find x $\in \mathscr{L}$ s.t. $||x|| = \lambda_1(\mathscr{L})$.



Hard problems in lattices

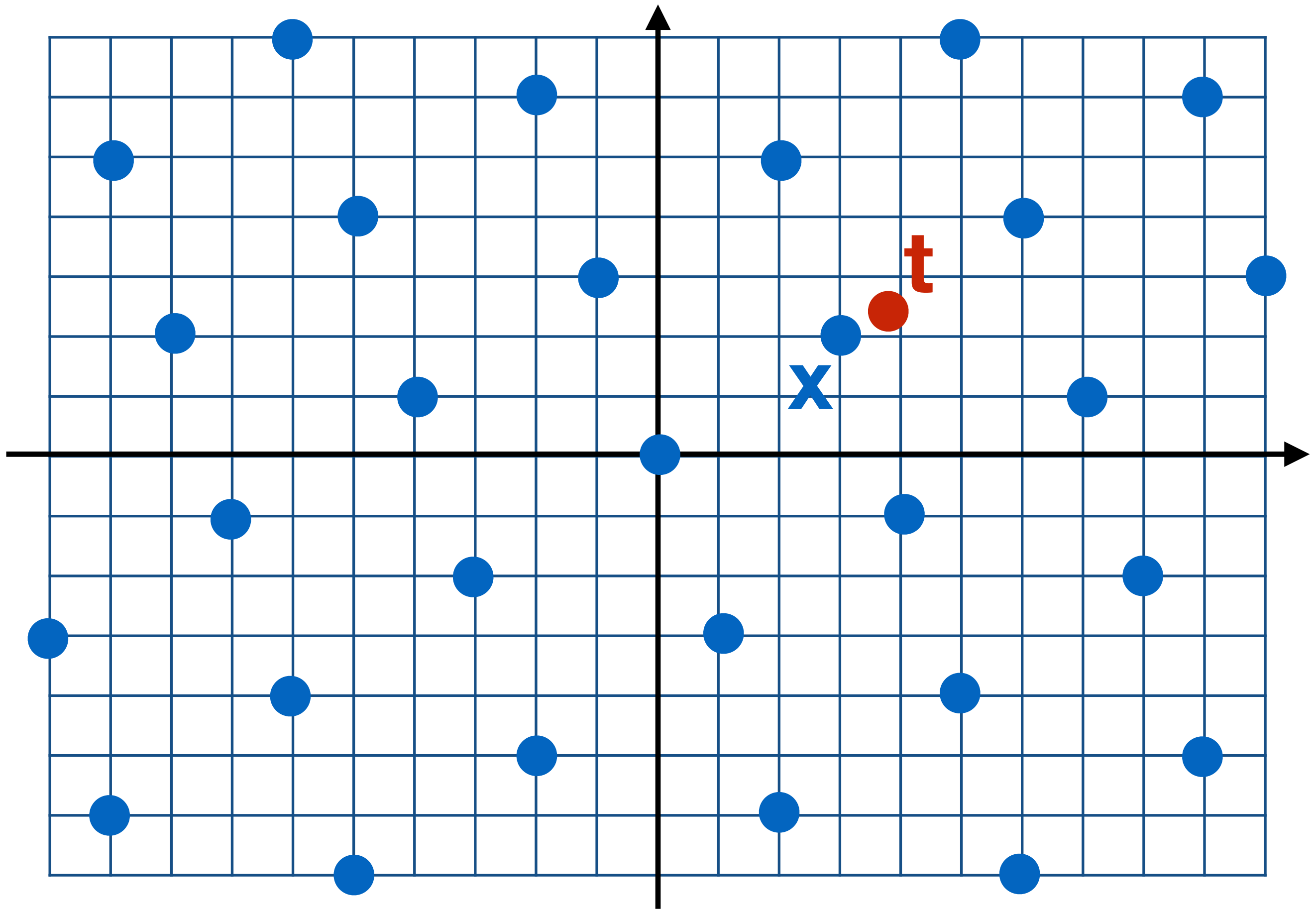
Shortest Vector Problem (SVP$_\gamma$). Given a basis B of a lattice $\mathscr{L} \subseteq \mathbb{R}^n$, find a vector x of norm $\leq \gamma(n) \cdot \lambda_1(\mathscr{L})$.



Decisional Shortest Vector Problem (GapSVP$_\gamma$). Given a basis B of a lattice $\mathscr{L} \subseteq \mathbb{R}^n$, decide if $\lambda_1(\mathscr{L}) \leq 1$ or $\lambda_1(\mathscr{L}) \geq \gamma(n)$.

Hard problems in lattices

Bounded Distance Decoding (BDD_γ). Given a basis B of a lattice $\mathscr{L} \subseteq \mathbb{R}^n$ and t$\in\mathbb{R}^n$, with the promise: ∃ x$\in \mathscr{L}$, $||t - x|| < \lambda_1(\mathscr{L})/(2\gamma(n))$, find x (necessarily unique for $\gamma \geq 1$).

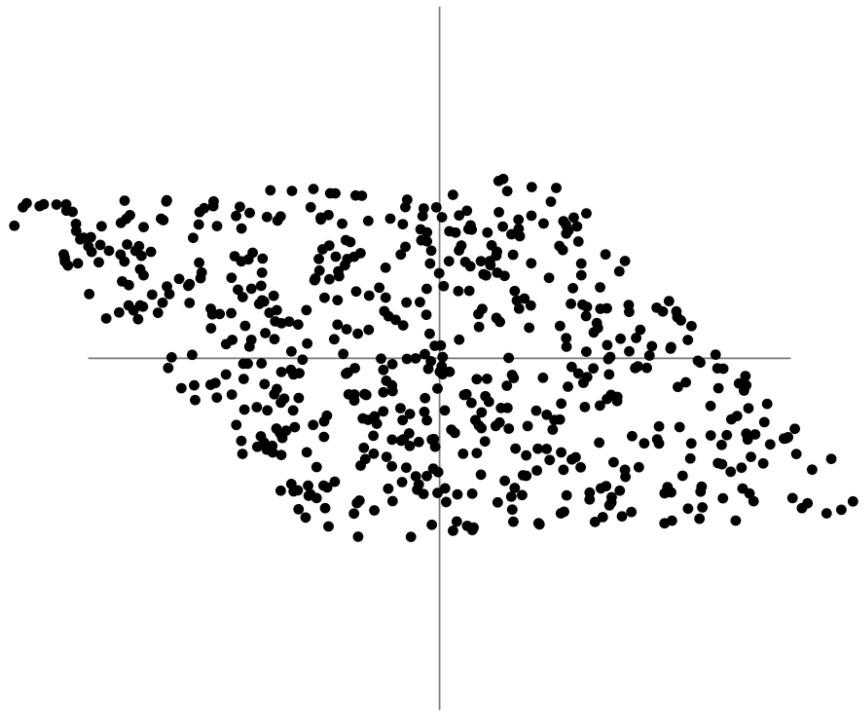


How hard are these problems?

- Deep and well-studied area → confidence in hardness.
- No known significant quantum speedup.
- However, not (believed to be) NP-hard.

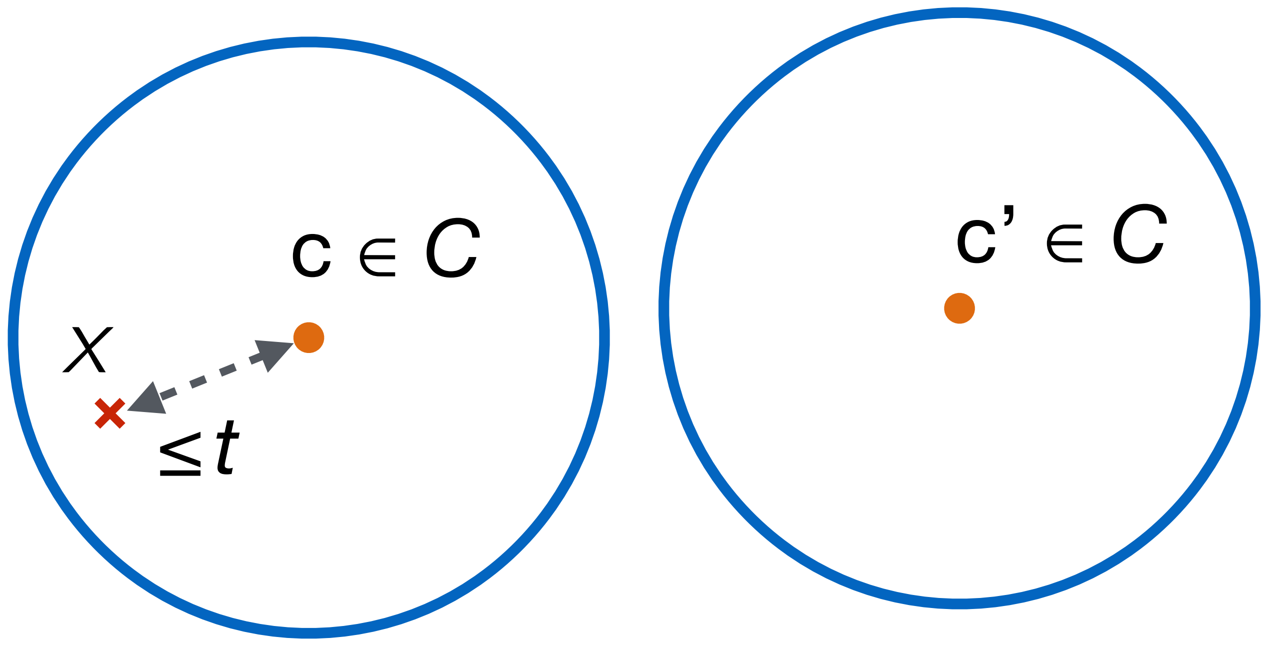
  For typical choice in crypto of $\gamma \geq \in \mathrm{Poly}(n)$ with $\gamma \geq \sqrt{n}$, GapSVP is in NP∩coNP.

Crypto from lattices



Recall code-based crypto…

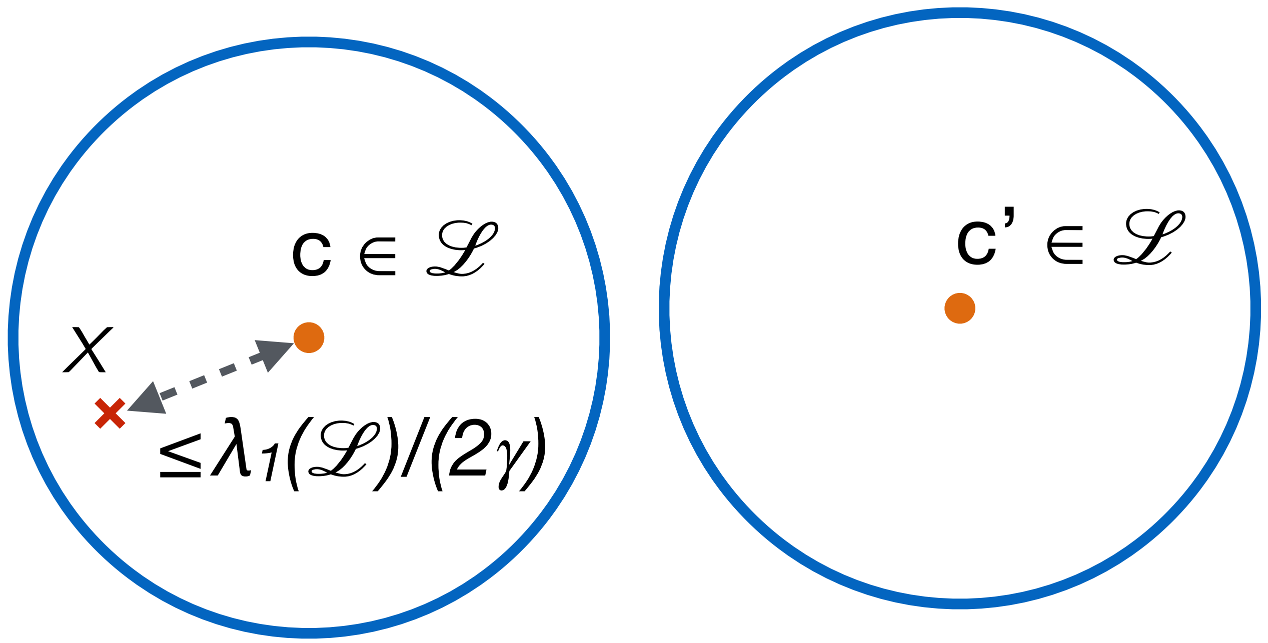
Problem: given a generator matrix G (i.e. a basis of C) and some x such that dist(x-c) $\leq t$ for some c in C, find c.



- For a random linear code, this is a **hard problem**!
- Except if you have a trapdoor (the code is secretly a "permutation" of an efficiently decodable code).

Now with lattices…

Problem: given a random lattice in $\mathbb{Z}_q$ (given as HNF of a uniform matrix) and some *x* such that dist(*x*-$\mathscr{L}$) ≤ $\lambda_1(\mathscr{L})/2\gamma$, find *c*.



- This is **BDD$_\gamma$**! It is a **hard problem**.
- Except if you have a trapdoor: namely, a good base of the lattice. You can then apply Babai’s rounding algorithm.

The McEliece cryptosystem

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).

The GGH cryptosystem

Golreich, Goldwasser, Halevi 1997.

Pick a good basis G of some lattice L in $\mathbb{Z}_q$.

Public key: Hermite Normal Form B of G.

Secret key: G.

Encrypt: encode a message m into the lattice L (generated by B), pick a small enough random error vector e. The ciphertext c is:

$$c = m + e$$

Decrypt: given a ciphertext c, retrieve closest lattice point m using knowledge of the good basis G (using Babai's rounding algorithm).

The GGH cryptosystem

- Warning: Like RSA or basic McEliece, this is actually a **trapdoor permutation**. It is not a PKE: not IND-CCA secure (why?).
- Some care is needed regarding how the message is encoded into the lattice.
- In theory: **No reduction** → "heuristic" security.
- In practice: impossibly large parameters.

GGH signatures

Golreich, Goldwasser, Halevi 1997.

Pick a good basis G of some lattice L in $\mathbb{Z}_q$.

Public key: Hermite Normal Form B of G.

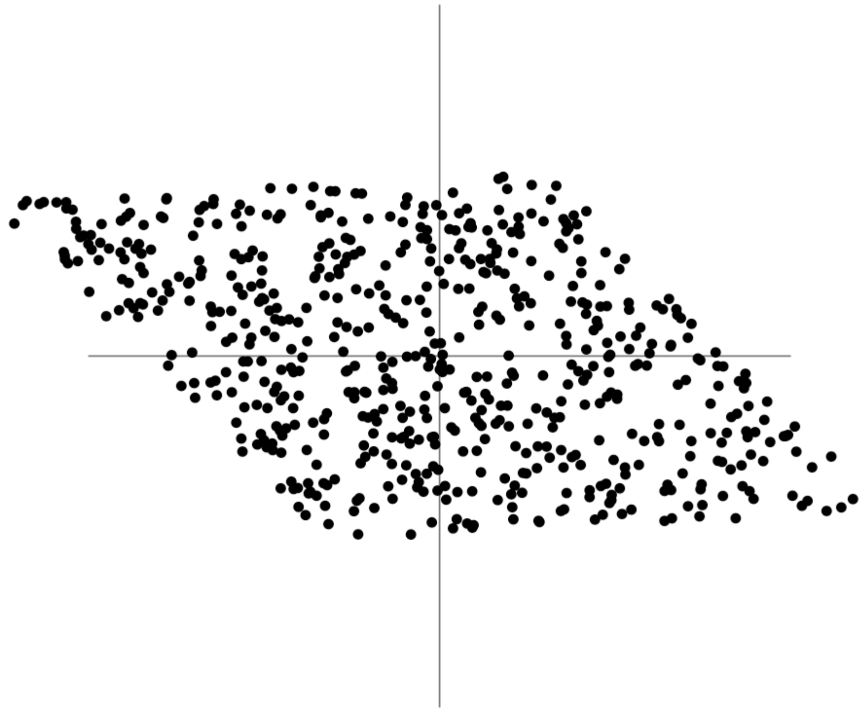
Secret key: G.

Sign: encode a message m as a point in $\mathbb{Z}_q$. The signature of m is the closest lattice point x (computed using G).

Verify: check that the signature x is close enough to m.

GGH signatures

- This time, similarities to Niederreiter signatures in codes.
- Again, **no reduction** → “heuristic” security.
- In fact, broken asymptotically and in practice! Nguyen-Regev ’06.



- Idea: the value *x-m* is uniformly distributed in the fundamental parallelipiped $G \cdot [-1/2,1/2]^n$. Yields a learning problem: the Hidden Parallelipiped Problem.