Post-Quantum Crypto and Lattices

Brice Minaud

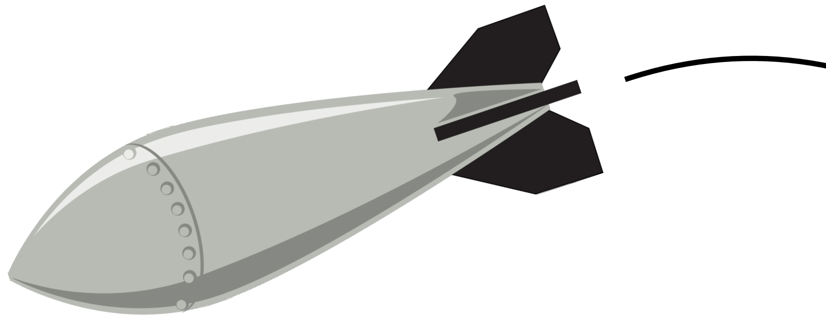
email: brice.minaud@inria.fr
website: www.di.ens.fr/brice.minaud/

MPRI, 2019

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.
- Multivariate crypto.

Also: symmetric crypto (incl. signatures!)

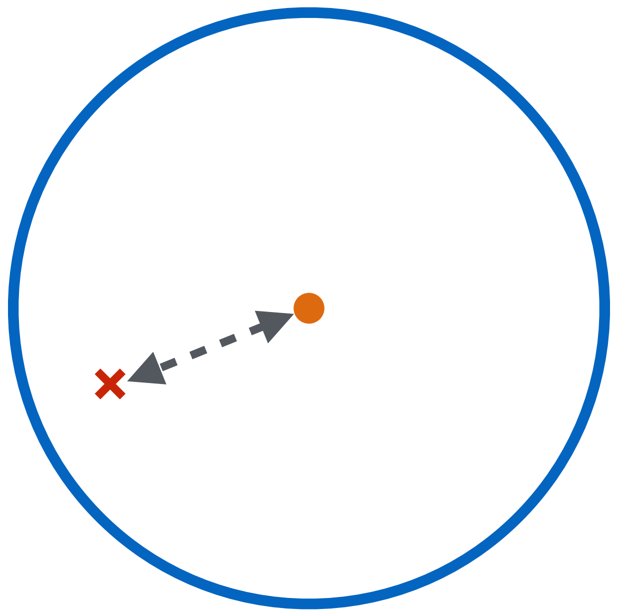
Number Theory

Lattices, codes,…
(conjectured)

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.

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.

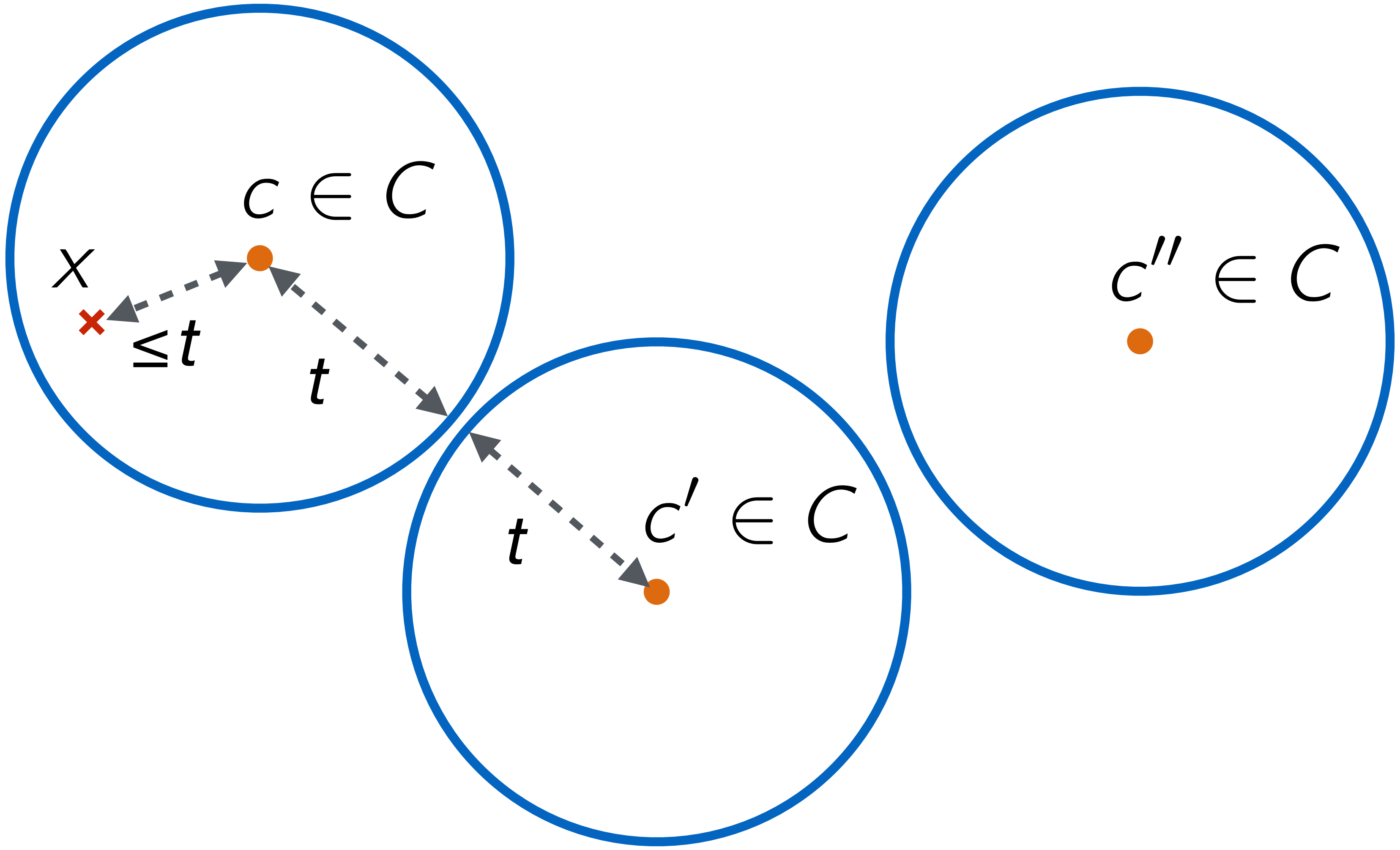
hw(010011) = 3

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

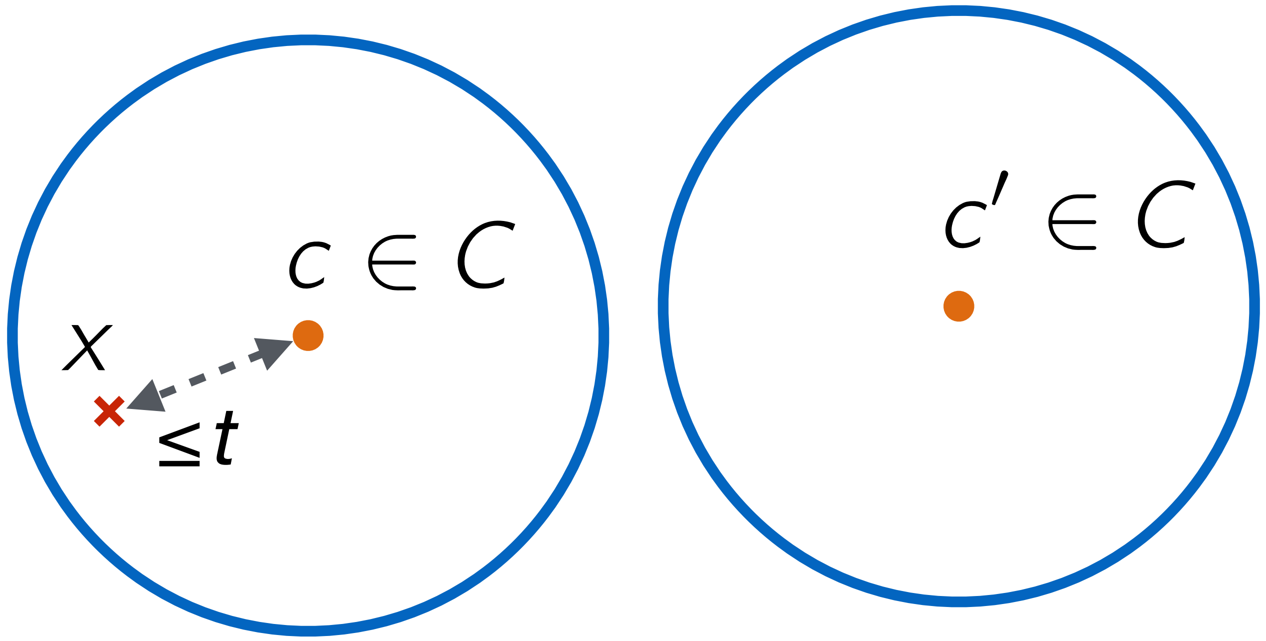
- That is, for $x \in \mathbb{F}_q^n$, if dist(x-c) ≤ t for some $c \in C$, then c is unique.



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 dist(*x*-*c*) ≤ *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

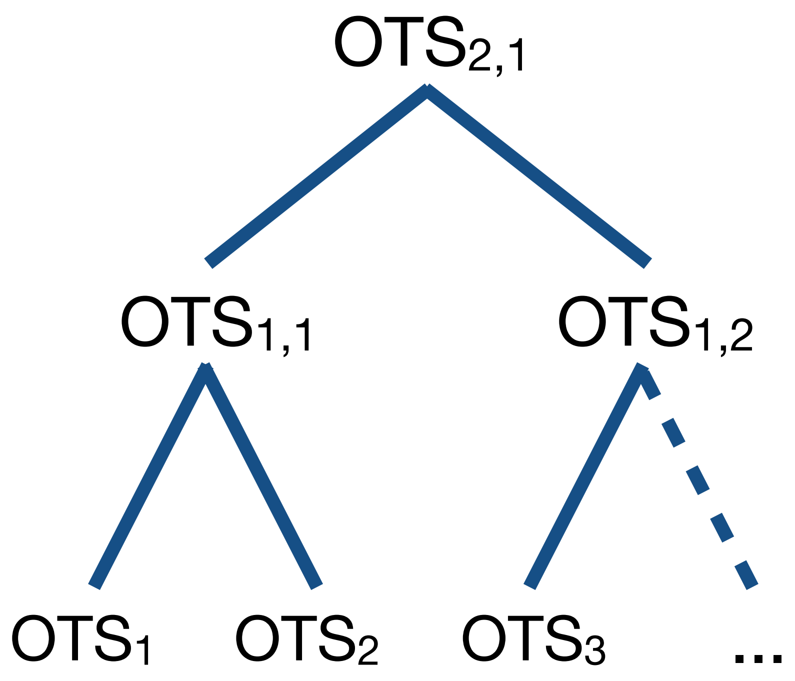
The original McEliece, using binary Goppa codes, is essentially unbroken since 1978.

> 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.

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 $h(x_b) = y_b$.

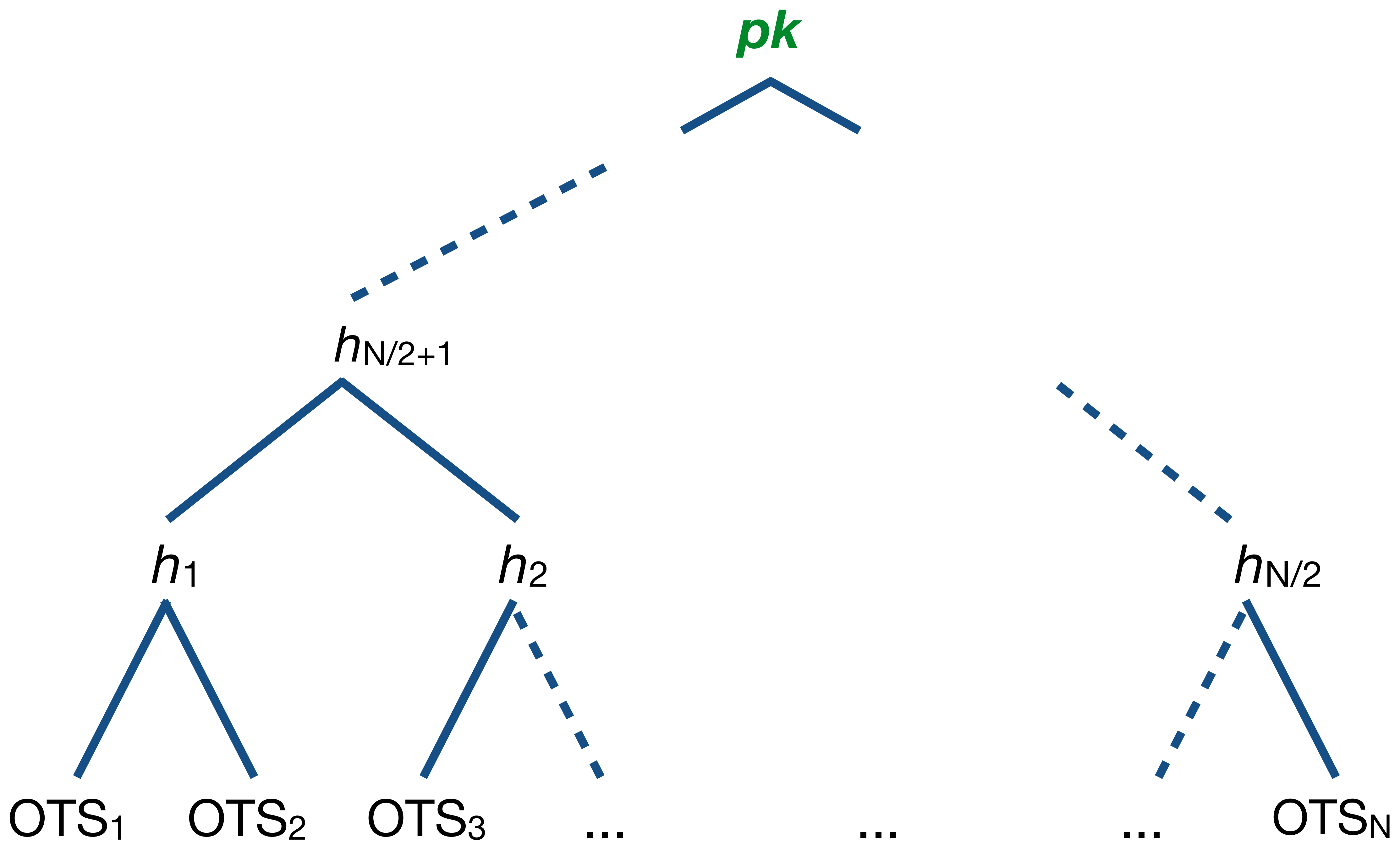
Lamport signature

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.

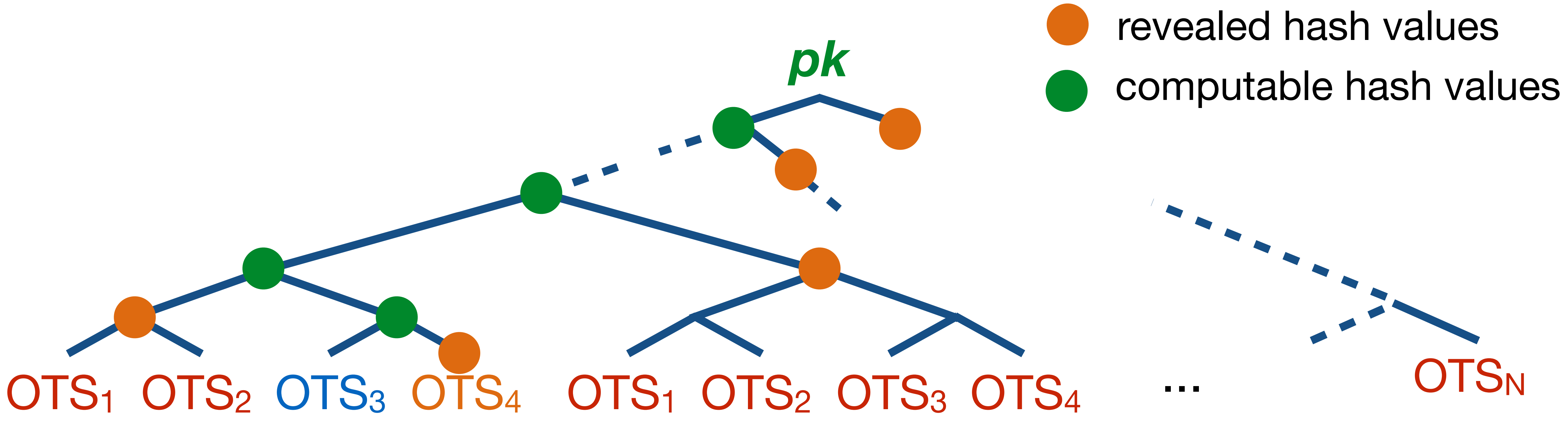
Next challenge: how to go from one-time signature to many-time signature?

Solution 1: Merkle trees



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

Solution 1: Merkle trees



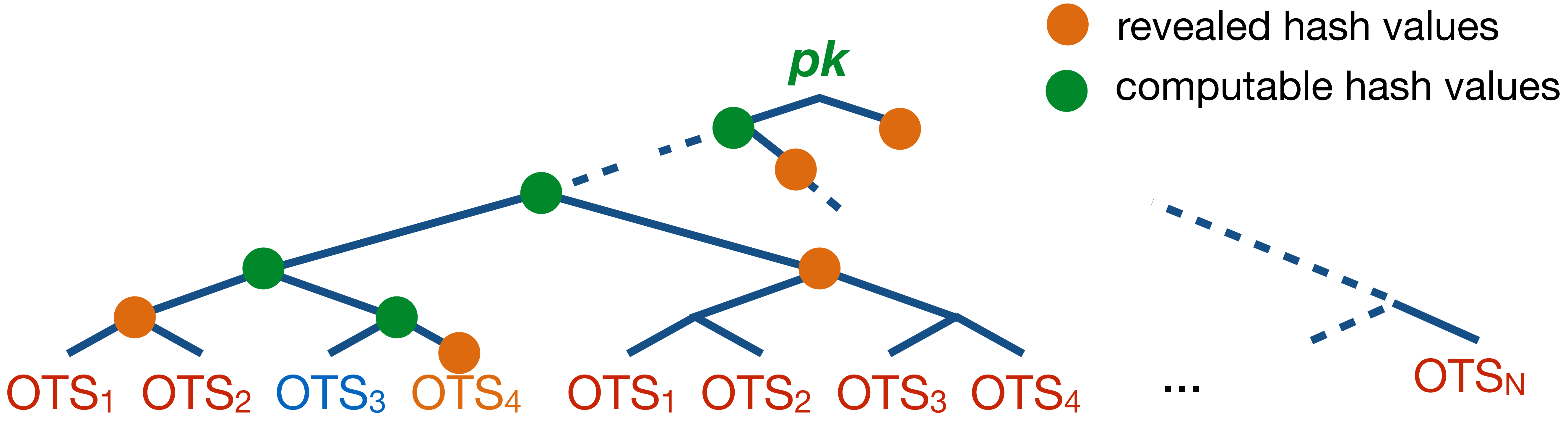
- The secret key is sk = (OTS_1,OTS_2,...,OTS_N).
- The public key is the root of the tree *pk*.

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:

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

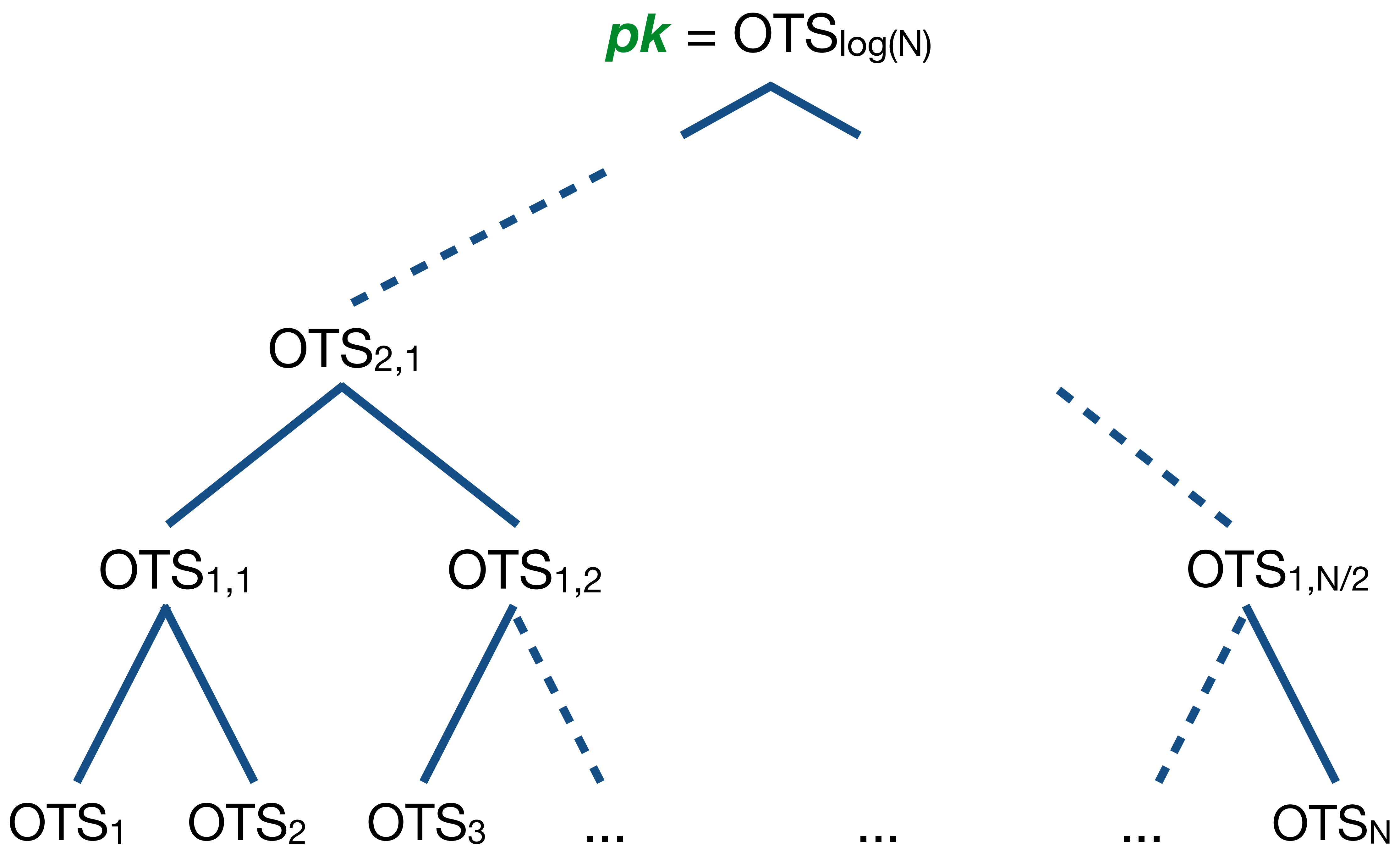
Verification: check the OTS_i signature, and all hashes.

Solution 1: Merkle trees



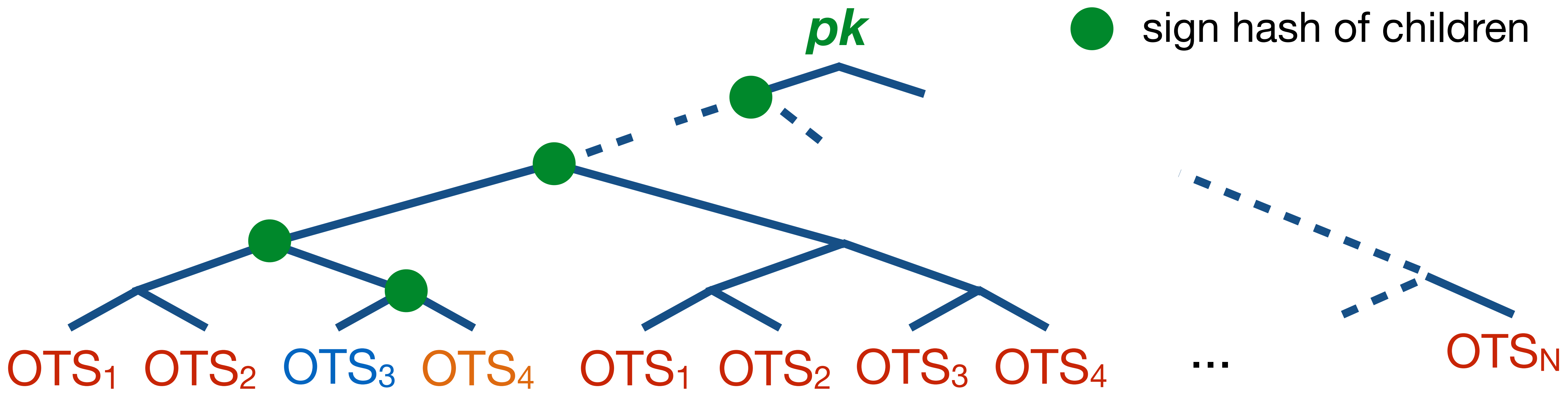
- Can sign up to N messages.
- Signatures are length *O*(log(N)).
- Needs a state to store which OTS_i is next to be used.
- **Problem**: need *O*(N) precomputation to get *pk*!

Solution 2: Goldreich scheme



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

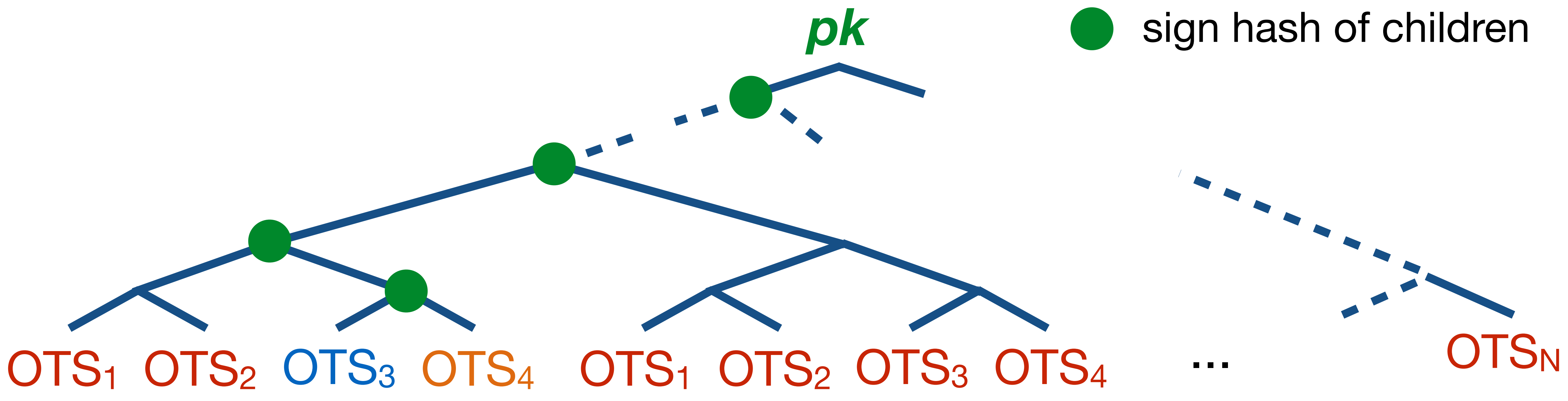
Solution 2: Goldreich scheme



- 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.

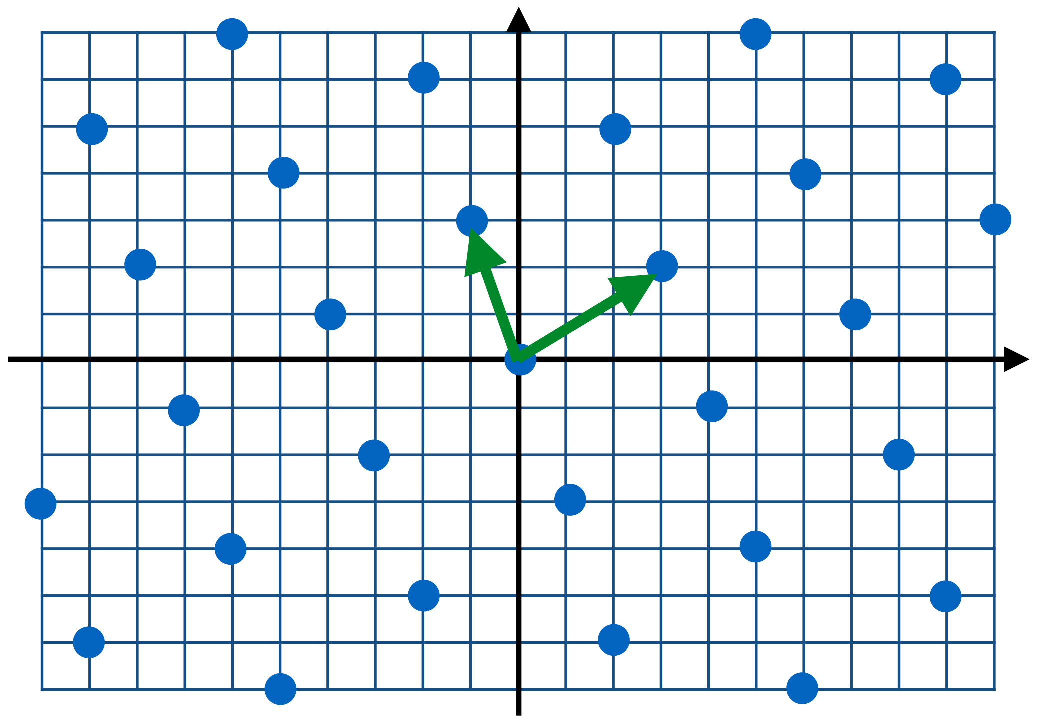
SPHINCS

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

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

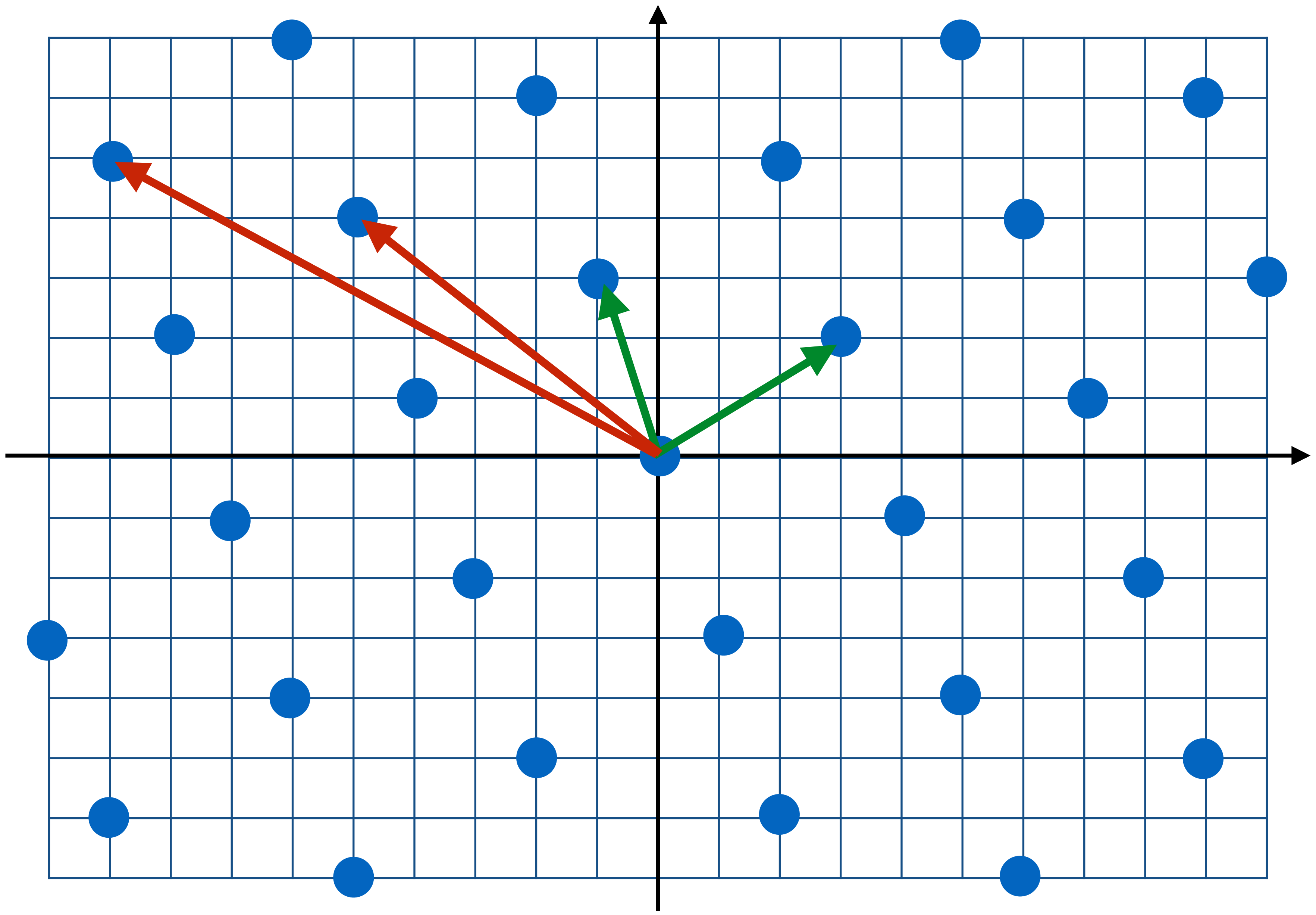
In practice, in crypto, $\mathscr{L}$ often:

- 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 $\mathscr{L}$, $({}^tB)^{-1}$ 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 $B \in \mathbb{Z}^{n\text{x}n}$ of a (full-rank) lattice is HNF iff:

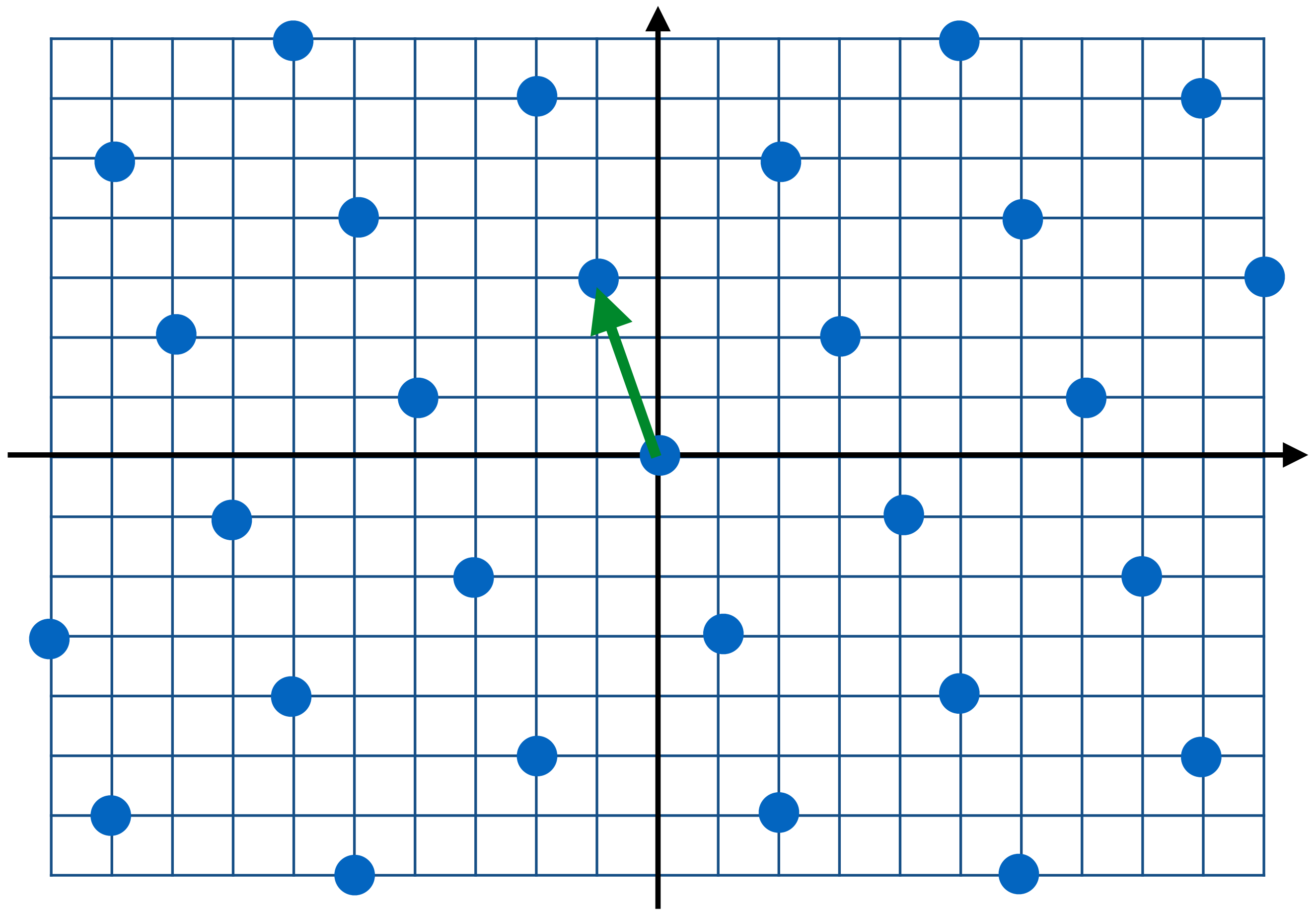
- It is upper triangular, with > 0 diagonal elements.
- Elements to the right of a diagonal element $m_{i,i}$ are ≥0 and < $m_{i,i}$.

Hard problems in lattices

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

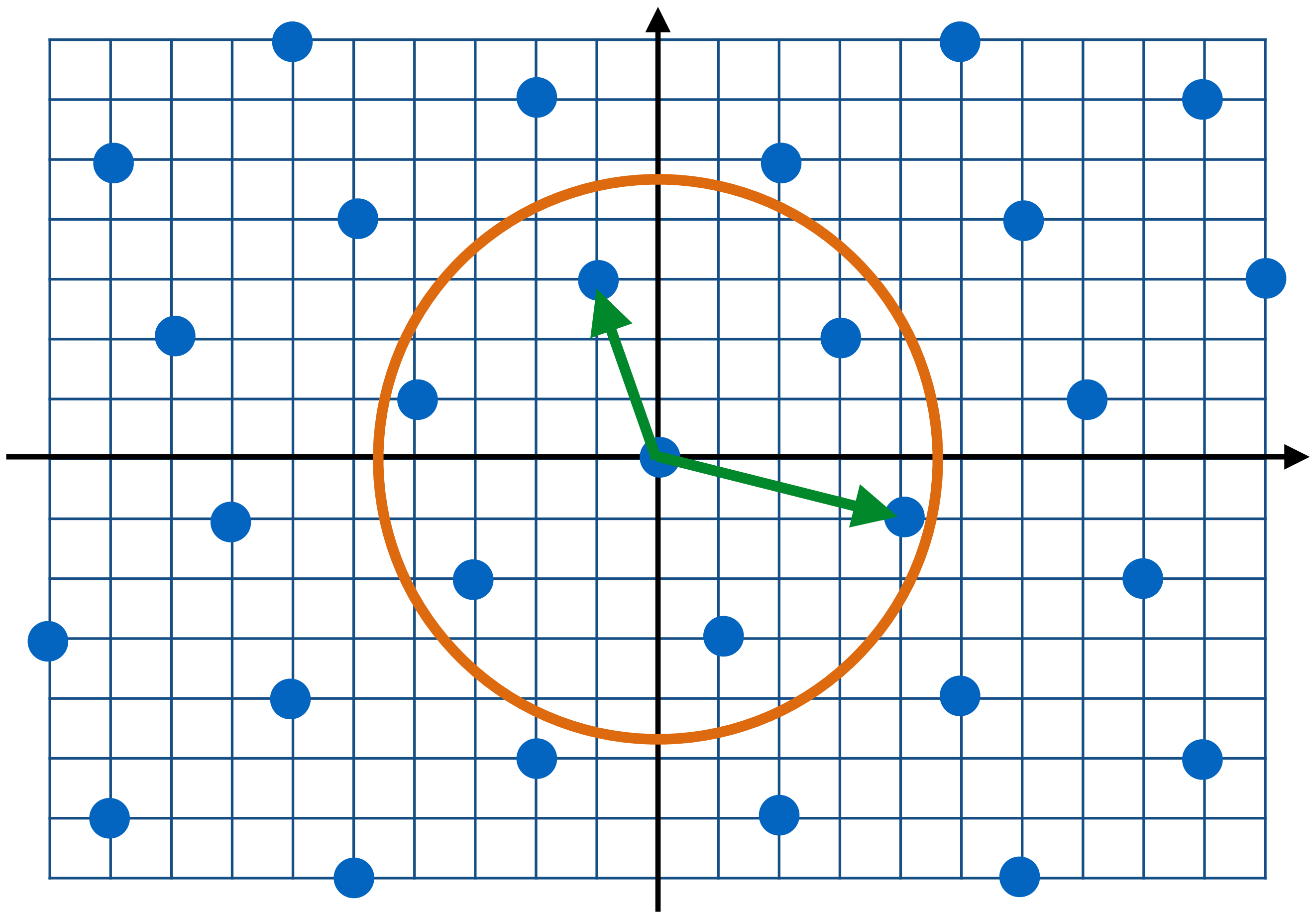
Define $\lambda_i(\mathscr{L})$ to be the smallest vector independent from $\lambda_1(\mathscr{L}), \ldots, \lambda_{i-1}(\mathscr{L})$.

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

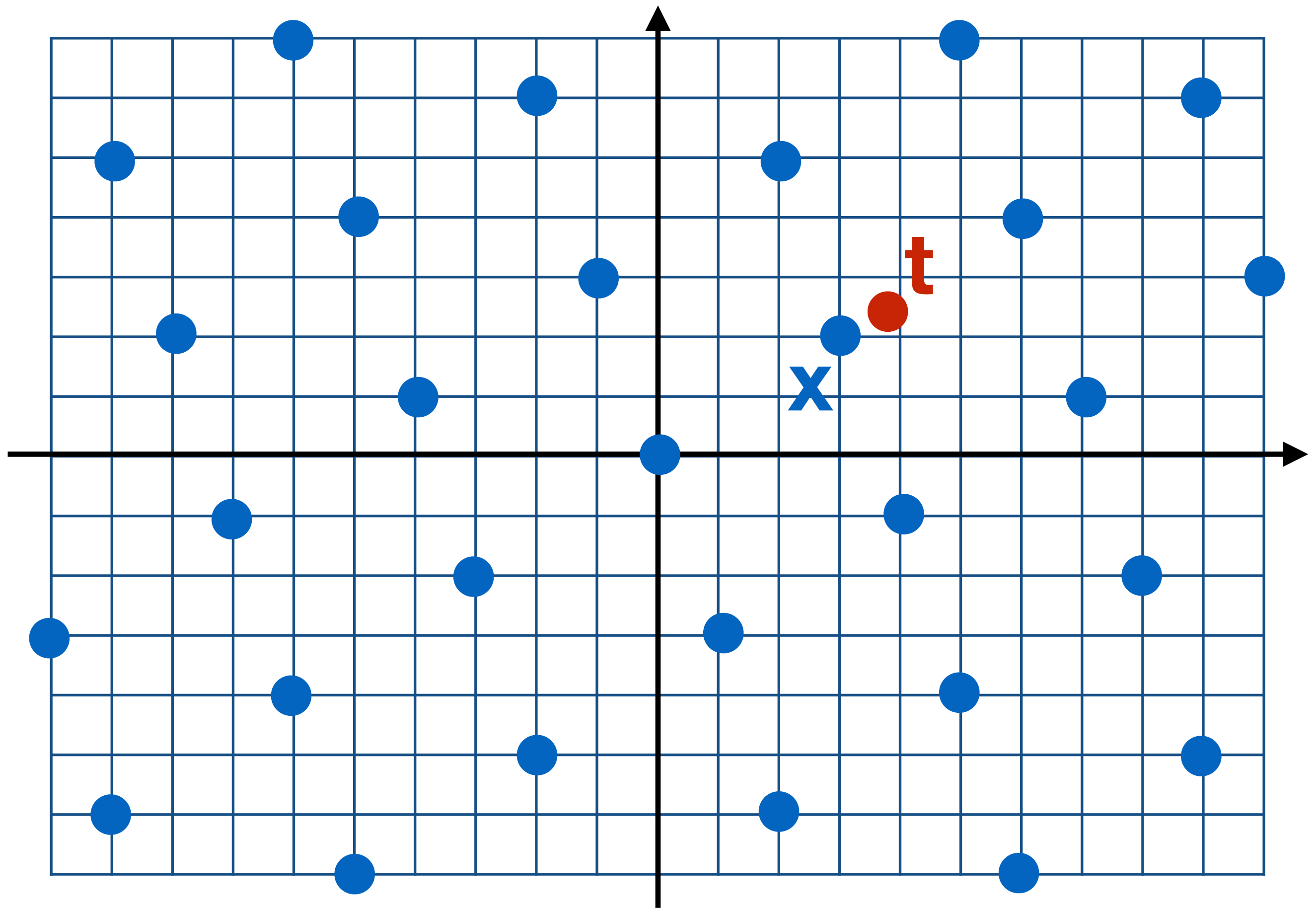
Shortest Vector Problem (SVP$_\gamma$). Given a basis B of a lattice $\mathscr{L} \subseteq \mathbb{R}^n$, find a vector x of norm $\leq \gamma(n) \cdot \lambda_1(\mathscr{L})$.



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$_\gamma$). 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?

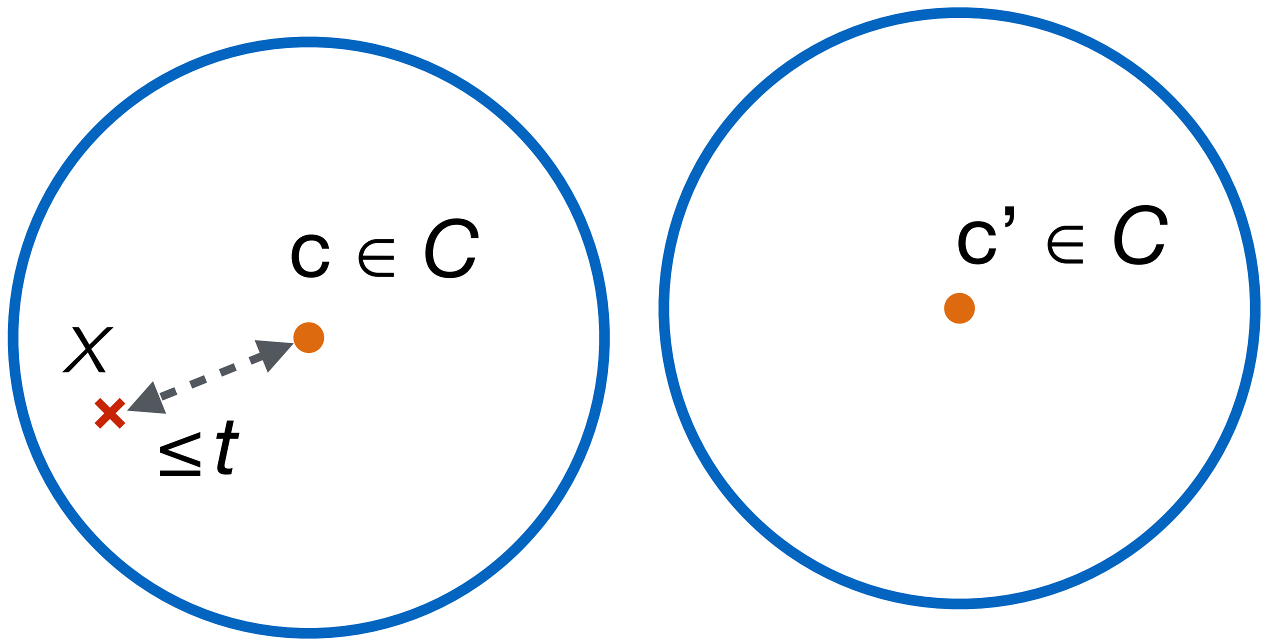
- Deep and well-studied area → confidence in hardness.
- No known significant quantum speedup.
- However, not (believed to be) NP-hard.

  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…

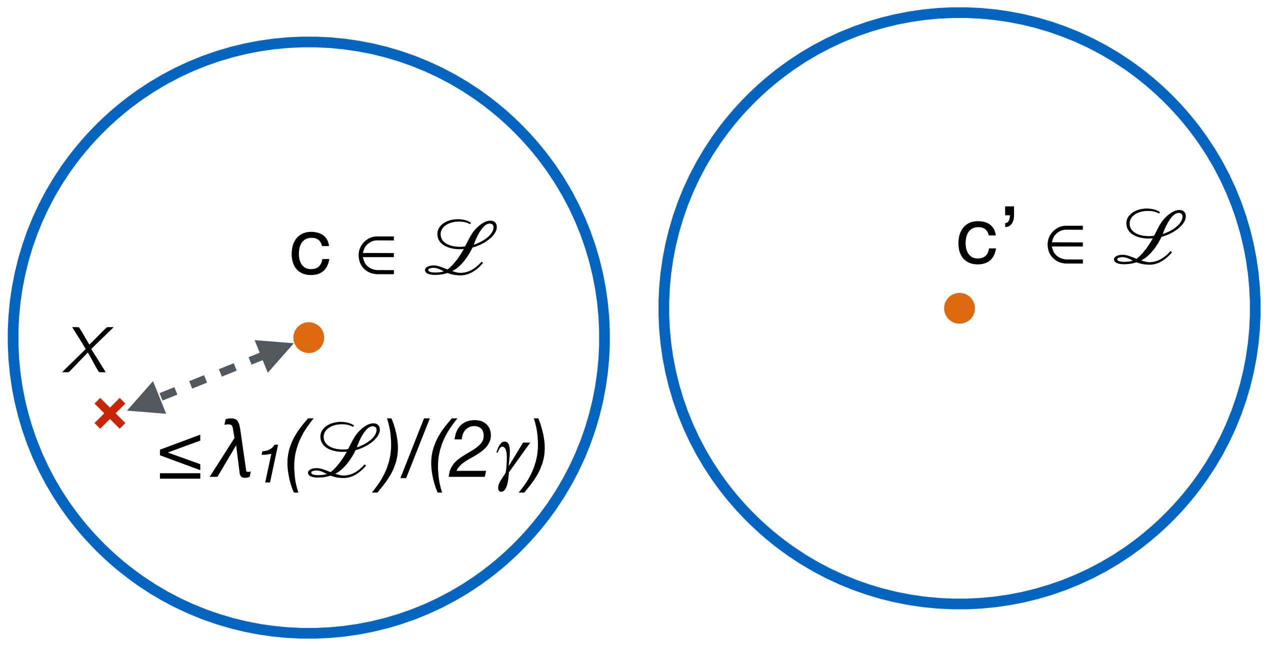
Problem: given a generator matrix G (i.e. a basis of C) and some x such that dist(x-c) $\leq t$ for some c in C, find c.



- 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.

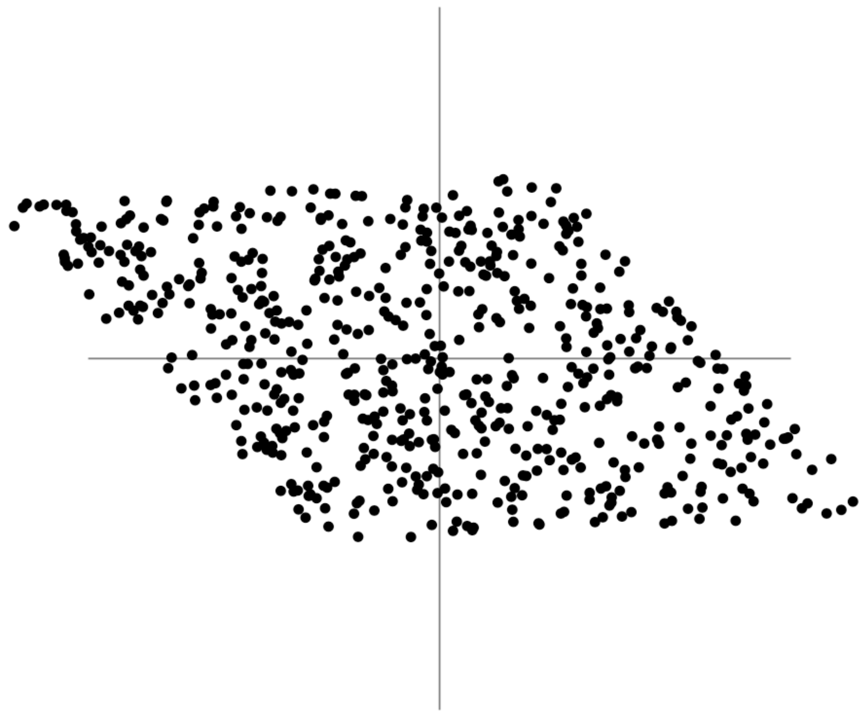
Secret key: G.

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.

Modern approach, part I

SIS: short integer solution

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 $\mathscr{L}^\perp(A) = \{x \in \mathbb{Z}_q^m : Ax = 0\}$(in $\mathbb{Z}_q$).

This is a (q-ary) lattice!

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

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 $n \sim 100$, $mn \sim 100000$…

Modern approach, part II

LWE: learning with errors

Learning Parity with Noise (LPN)

Say I have $m > n$ vectors a_i in $\mathbb{Z}_2^n$.
I am given $a_i \cdot s + e_i$ (scalar product) for some secret s, $e_i \in \mathbb{Z}_2$ drawn from Bernoulli distribution B(η) (i.e. $\Pr(e_i = 1) = \eta$).

Problem: find s.

Oracle $O_\$$: returns (a,b) for a uniform in $\mathbb{Z}_2^n$, b uniform in $\mathbb{Z}_2$.
Oracle O_s: returns $(a, a \cdot s + e)$ for a uniform in $\mathbb{Z}_2^n$, e drawn from B(η).

LPN problem. Let $s \in \mathbb{Z}_2^n$ be drawn uniformly at random. Given access to either $O_\$$ or O_s, distinguish between the two.

LPN problem (bounded samples). Let $A \in \mathbb{Z}_2^{m \times n}$ and $b, s \in \mathbb{Z}_2^n$ be drawn uniformly at random, and $e \in \mathbb{Z}_2^m$ drawn according to B(η).
Distinguish between $(A, As + e)$, and (A, b).

Learning Parity with Noise (LPN)

- Famous problem in learning theory.
- Trivial without the noise.
- Believed to be very hard, even given unbounded samples. Best algorithm slightly sub-exponential: Blum-Kalai-Wasserman 2003. Complexity roughly $2^{n/\log n}$ in time and #queries.
- For bounded samples, same as decoding a random linear code.

Secret-key encryption using LPN

Attempt #1.

Pick a secret *s* uniformly in $\mathbb{Z}_2^n$.

Secret key: *s*.

Encrypt: to encrypt one bit *b*: give *m* samples from $O_\$$ if b=0, *m* samples from O_s if b=1.

Decrypt: use *s* to distinguish the two oracles.

Secret-key encryption using LPN

Attempt #2.

Pick a secret s uniformly in $\mathbb{Z}_2^n$.

Secret key: s.

Encrypt: to encrypt one bit b: give m samples from $(a, a\cdot s+b+e)$.

Decrypt: compute $a\cdot s$ to retrieve $b+e$, determine e by majority vote.

Secret-key encryption using LPN

Attempt #3.

Pick a secret *S* uniformly in $\mathbb{Z}_2^{m\times n}$.

Secret key: *S*.

Encrypt: to encrypt message *m*: (*a*,*Sa*+C(*m*)+*e*) where C($\cdot$) encodes the message into $\mathbb{Z}_2^m$ with error correction.

Decrypt: use *S* to retrieve C(*m*)+*e*, use error correction to remove *e*.

Additional tweaks: LPN-C cryptosystem (Gilbert et al. ’08).

Learning with Errors (LWE)

Regev '05. Milestone result.

Pick *s* uniformly in $\mathbb{Z}_q^n$.

Oracle O$_\$$: returns (*a*,*b*) for *a* uniform in $\mathbb{Z}_q^n$, *b* uniform in $\mathbb{Z}_q$.

Oracle O$_s$: returns (*a*,*a*$\cdot$*s*+*e*) for *a* uniform in $\mathbb{Z}_q^n$, *e* drawn from χ.

Typically, χ is a discrete Gaussian distribution with std deviation αq.

LWE. Let $s \in \mathbb{Z}_q^n$ be drawn uniformly at random. Given access to either O$_\$$ or O$_s$, distinguish between the two.

LWE (bounded samples). Let $A \in \mathbb{Z}_q^{m \times n}$ and $b, s \in \mathbb{Z}_q^n$ be drawn uniformly at random, and $e \in \mathbb{Z}_q^m$ drawn according to χ.
Distinguish between (*A*, *As* + *e*), and (*A*, *b*).

Search and Decision variants

LWE (decisional). Let $s \in \mathbb{Z}_q^n$ be drawn uniformly at random. Given access to either O$_\$$ or O$_s$, distinguish between the two.

LWE (search). Let $s \in \mathbb{Z}_q^n$ be drawn uniformly at random. Given access to either O$_s$, find s.

Proposition 1: the two problems are equivalent up to polynomial reductions (“hybrid” technique).

Proposition 2: given an efficient algorithm that solves SIS with parameters n, m, q, β, there is an efficient algorithm that solves LWE with the same parameters, assuming (roughly) $\alpha\beta \ll 1$.

LWE and BDD

LWE (bounded samples). Let $A \in \mathbb{Z}_q^{m\times n}$ and $b, s \in \mathbb{Z}_q^n$ be drawn uniformly at random, and $e \in \mathbb{Z}_q^m$ drawn according to χ.

Distinguish between (A, As + e), and (A, b).

Proposition 3: LWE reduces to BDD with $\gamma = q^{n/m}/\alpha$.

Consider the lattice $\mathscr{L} = A\mathbb{Z}_q^n$ generated by A.
The shortest vector is expected to have norm $\lambda_1(A) \sim \sqrt{(m)}q^{(m-n)/m}$.
The standard deviation of e is $\sqrt{m\alpha q}$.
(In particular we can expect the closest lattice point to As+e *is* As.)

Better! Regev '05: Solving LWE (for uniformly random A) implies **quantumly** solving GapSVP in dimension n for **any** lattice!

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

Classical reduction in dim $\sqrt{n}$, Peikert '09.

Flexibility of LWE I

Random self-reducibility. Consider a variant of LWE where the secret s is drawn according to some distribution σ, instead of uniformly. This variant reduces to standard LWE.

→ "average-case" LWE (for the secret) is hardest possible.

Flexibility of LWE II

Secret-error switching. Consider a variant of LWE where the secret s is drawn according to the error distribution χ, instead of uniformly. This reduces to standard LWE, and conversely.

→ drawing the secret from χ is also hardest possible. (This reduction costs sacrificing n samples.)

Flexibility of LWE III

Consider a variant of LWE where instead of publishing samples (a, $a \cdot s + e$), with n-dimensional secret vector s, samples are (a, $a \cdot S + e$), with $k \times n$ secret matrix S: LWE with "k secrets".

LWE with multiple secrets. LWE with k secrets reduces to standard LWE (with k calls to the LWE oracle).

Flexibility of LWE IV

Many other variants of LWE still reduce to LWE:

- Binary-LWE: s is in $\{0,1\}^n$ (with limited samples).
- Learning with Rounding (LWR): the error is uniform in a small range instead of Gaussian. Amounts to deterministic rounding!
- …

Can be used for a host of applications:

- Secret-key encryption, PRF.
- PKE, key exchange.
- Identity-based encryption (see Michel's course), FHE.
- …

Secret-key encryption using LWE

Like LPN:

Pick a secret s uniformly in $\mathbb{Z}_q^n$.

Secret key: s.

Encrypt: to encrypt one bit b: give $(a, a \cdot s + b\lfloor q/2 \rfloor + e)$.

Decrypt: compute $a \cdot s$ to retrieve $b\lfloor q/2 \rfloor + e$, output b=1 iff closer to $\lfloor q/2 \rfloor$ than to 0.

IND-CPA security sketch: $(a, a \cdot s + e)$ is indistinguishable from uniform, hence so is $(a, a \cdot s + b\lfloor q/2 \rfloor + e)$.

A public sampler for LWE

To make previous scheme public-key, we'd like a public "sampler" for LWE. Should not require knowing the secret s.

Setup:

- Pick a secret s uniformly in $\mathbb{Z}_q^n$.
- Publish m LWE(q,n,χ) samples for large enough m (value TBD).

That is, publish $(A, As+e)$ for $m \times n$ matrix A.

Now to get a fresh LWE sample:

- Pick x uniformly in $\{0,1\}^n$.
- Publish $({}^t xA, {}^t x(As+e))$.

With the right parameters, this yields a distribution statistically close to LWE(q,n,χ'), where if χ is Gaussian with variance σ^2, χ' is Gaussian with variance $m\sigma^2$.

Argument: Leftover Hash Lemma. Example: $m = 2n \log q$ suffices.
Remark: recognize the Ajtai hash function from earlier/subset sum.

Public-key encryption* using LWE

Regev '05: **Regev encryption.**
Idea: same as secret-key scheme, but with public sampler.

Pick a secret s uniformly in $\mathbb{Z}_q^n$, A uniformly in $\mathbb{Z}_q^{m\times n}$.

Public key: $(A, b = As + e)$.

Secret key: s.

Encrypt: to encrypt one bit k: draw x in $\{0,1\}^m$, output:

$$({}^t xA, {}^t xb + k\lfloor q/2 \rfloor).$$

Decrypt: upon receipt of ciphertext (c,d), output 0 if $d - c \cdot s$ is closer to 0 than to $\lfloor q/2 \rfloor$, 1 otherwise.

Proof argument. Step 1: public key is indistinguishable from uniform. Step 2: assuming uniform public key, ciphertexts are statistically close to uniform.

*malleability → not IND-CCA.

Practical (in)efficiency

Example parameters: q prime $\cong n^2$, $m = 2\, n \log q$, $\alpha = 1/(\sqrt{n} \log^2 n)$.
In practice, e.g. $n \cong 200$.

Terrible efficiency:
- $O(n^2)$ operations for encryption.
- $O(n \log n)$ ciphertext for 1 bit of plaintext!

Multi-bit Regev encryption

Idea: use multiple secrets.

Pick a secret **matrix** S uniformly in $\mathbb{Z}_q^{\ell\times n}$, A uniformly in $\mathbb{Z}_q^{m\times n}$.

Public key: $(A, B = AS + E)$.

Secret key: S.

Encrypt: to encrypt ℓ bits $k \in \{0,1\}^\ell$: draw x in $\{0,1\}^m$, output:

$$({}^t xA, {}^t xB + \lfloor q/2 \rfloor k).$$

Decrypt: upon receipt of ciphertext (C,D), output $k \in \{0,1\}^\ell$ such that $D\text{-}C\cdot S$ is closest to $\lfloor q/2 \rfloor k$.

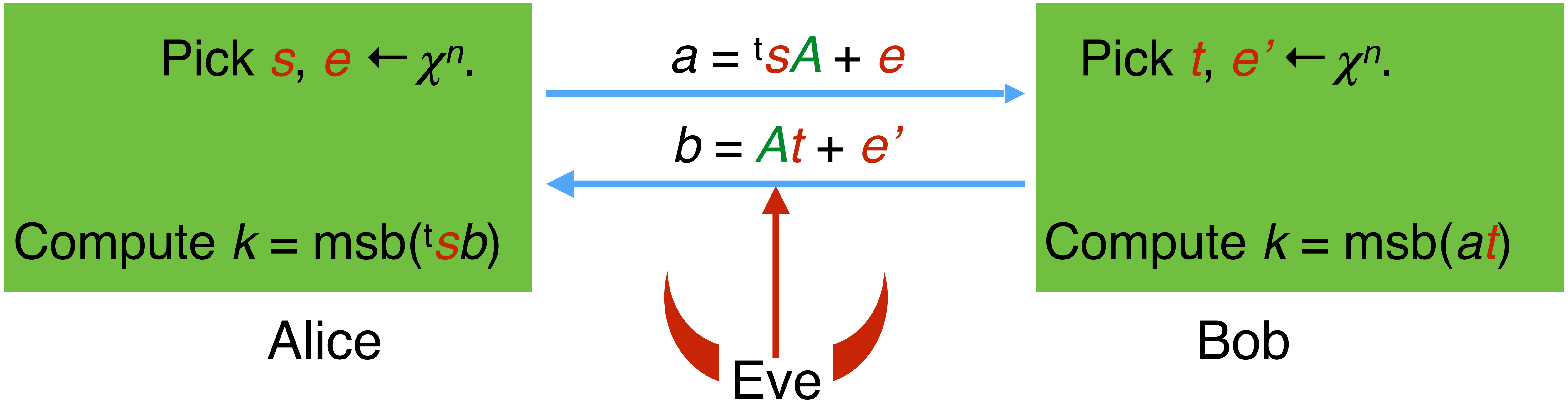
Proof argument: use multiple-secret LWE.

Ciphertext expansion $(n/\ell + 1) \log q$.

Other idea: encode multiple bits per element in $\mathbb{Z}_q$. (use high-order bits.)

Key exchange

Setup: pick public A uniformly in $\mathbb{Z}_q^{n\times n}$.


Pick s, $e \leftarrow \chi^n$.
Compute $k = \mathrm{msb}({}^t s b)$
Alice
$a = {}^t s A + e$
$b = At + e'$
Eve
Pick t, $e' \leftarrow \chi^n$.
Compute $k = \mathrm{msb}(at)$
Bob

Here, msb = most significant bit.

Both parties get ${}^t sAt$ up to error terms. msb gets rid of error.

Equivalent of DDH: Eve wants to distinguish (A,a,b,k) from $(A,\$,\$,\$)$.

Proof argument: 1st hybrid $(A,\$,b,k)$. 2nd hybrid $(A,\$,\$,\$)$. Use LWE with secret-error switching on A, then $(A|a)$.

Practical aspects

Improving efficiency: compressing *A*

LWE (decisional). Let $s \in \mathbb{Z}_q^n$ be drawn uniformly at random.
Distinguish $(a, a \cdot s + e)$ from (a, b) for uniform a, b, and $e \leftarrow \chi$.

To get one "usable" *b* you need to publish the corresponding *a*, which is *n* times larger.

It'd be nice if the matrix *A* of *a*'s was structured → compressible.

Simple idea: cyclic *A*. (See cyclic codes…)

Amounts to operating in ring $\mathbb{Z}_q[X]/(X^n - 1)$ → **Ring-LWE**.

Ring-LWE

Let R = $\mathbb{Z}_q[X]/P$ for some polynomial *P* (think irreducible).

Ring-LWE (decisional). Let $s \in R$ be drawn uniformly at random. Distinguish (a,$a \cdot s$+e) from (a,b) for uniform a, $b \leftarrow R$, and $e \leftarrow \chi$.

The “usable” part *b* is now the same size as the uniform part *a*.

Example: **Regev encryption**
- ciphertext expansion O(1) instead of O(n).
- with proper choice of ring (e.g. arising from cyclotomic polynomials), $a \cdot s$ can be computed in $n \log n$, not n^2, using FFT.

Theoretical concern: reduces to hard *ideal* lattice problems. Believed to be as hard as general case, beside a few “trivial” properties (e.g. SVP = SIVP, collision on Ajtai hash function).

Concrete security

For factorization or Discrete Log, essentially one *family* of attacks.

For LWE and other lattice-based schemes, much more difficult:
- lattice reduction algorithms: LLL, BKZ.
- BKW-type algorithms (connection with LPN).
- ISD algorithms (connection with decoding random code).
- For low errors, such as Arora-Ge and Gröbner bases (connection with multivariate system solving).

→ ongoing NIST standardization process to fix concrete parameters.