Introduction to lattice-based cryptography

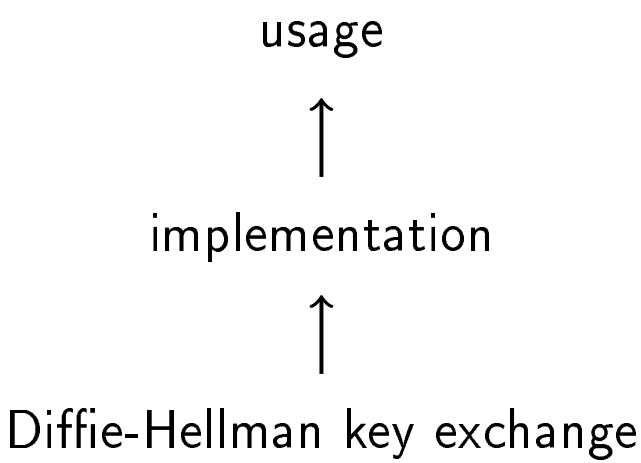
Alice Pellet-Mary

Amusec 2024, Marseille

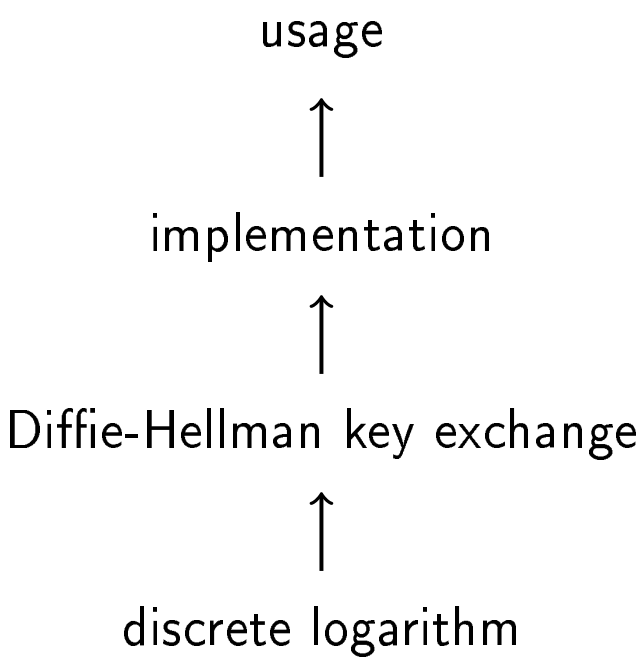
Context: example

Diffie-Hellman key exchange

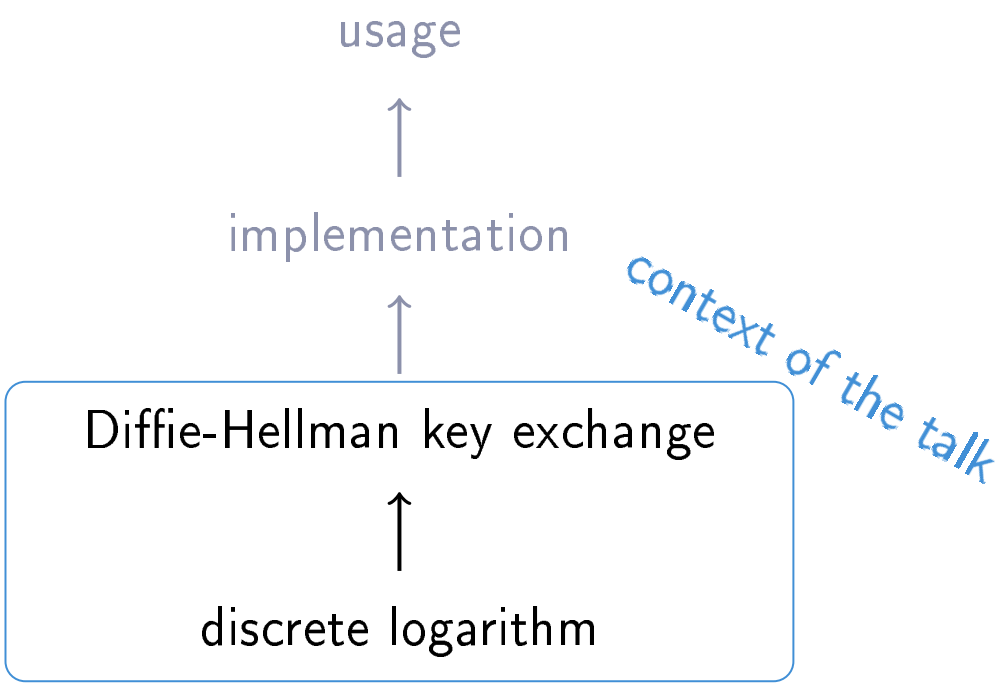
Context: example



Context: example



Context: example



Zooming in: public key cryptography

Cryptographic primitives

public key encryption | signature | homomorphic encryption | ...

Zooming in: public key cryptography

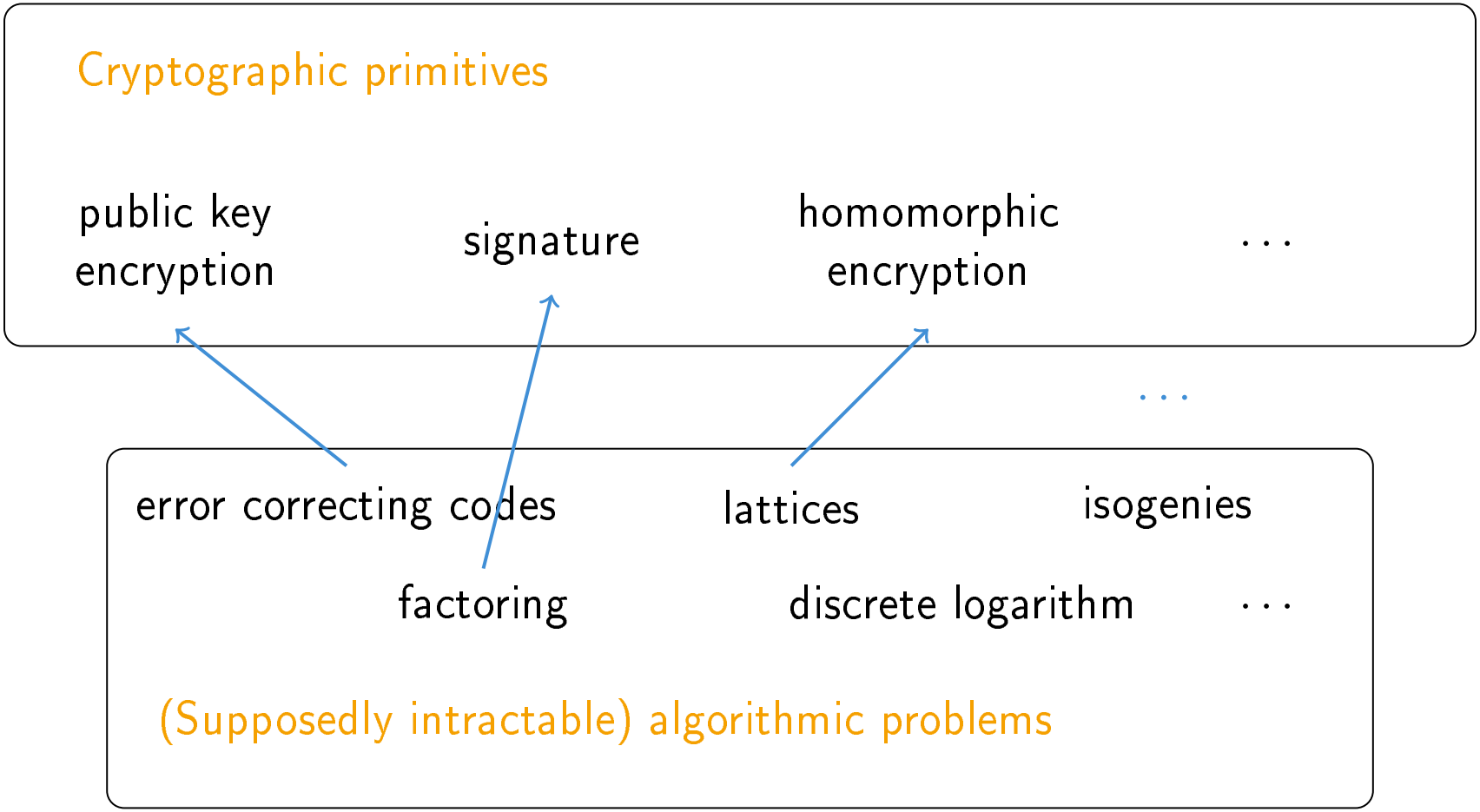
Cryptographic primitives

public key encryption | signature | homomorphic encryption | ...

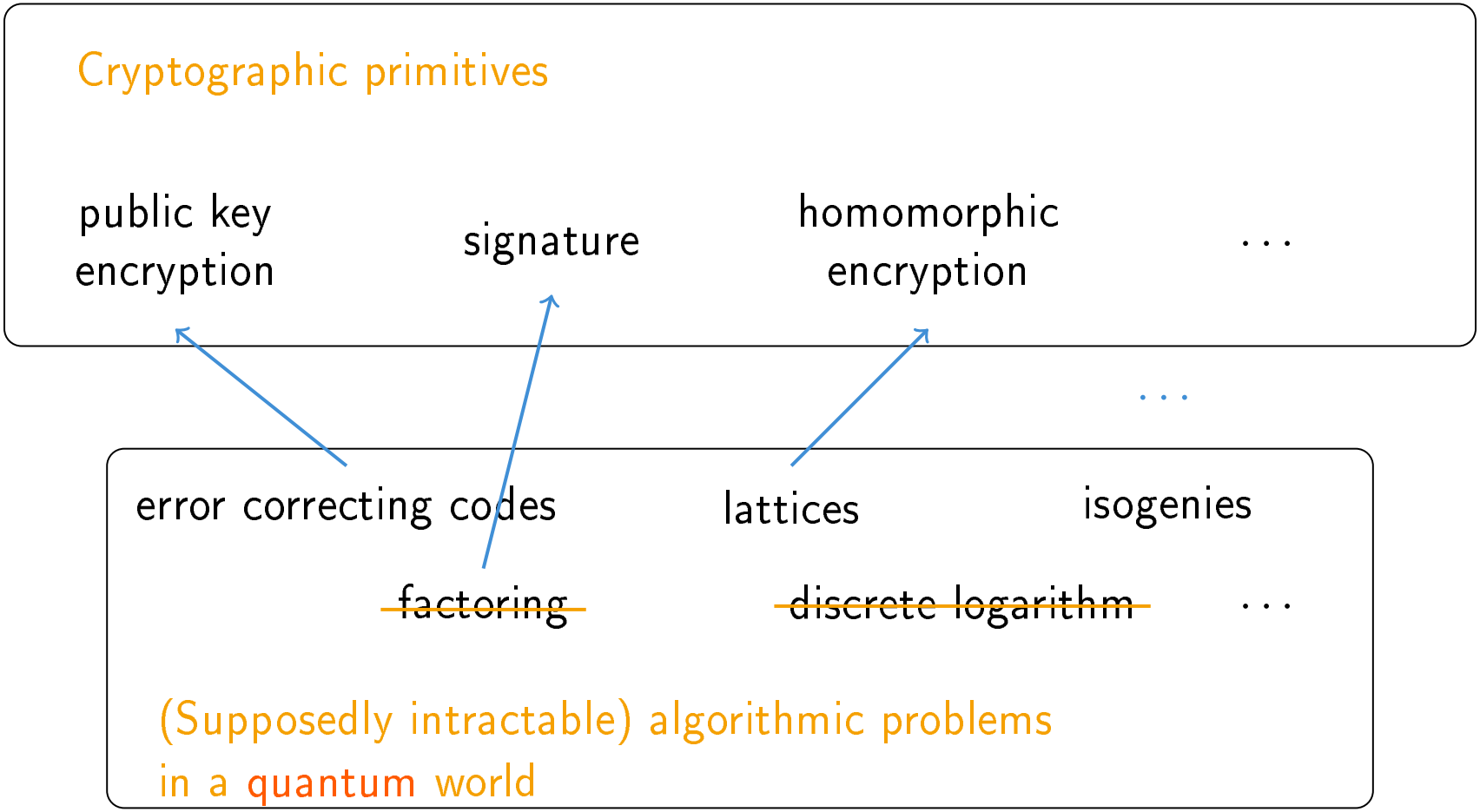
error correcting codes | lattices | isogenies | factoring | discrete logarithm | ...

(Supposedly intractable) algorithmic problems

Zooming in: public key cryptography



Zooming in: public key cryptography



"quantum world" = assuming attackers have access to a quantum computers

Post-quantum cryptography vs quantum cryptography

Post-quantum cryptography:

- uses classical infrastructures
- resists to attackers with quantum computers

Quantum cryptography:

- uses quantum mechanics and dedicated infrastructures
- resists to attackers with quantum computers

Post-quantum cryptography vs quantum cryptography

Post-quantum cryptography:

- uses classical infrastructures
- resists to attackers with quantum computers

Quantum cryptography:

- uses quantum mechanics and dedicated infrastructures
- resists to attackers with quantum computers

Which one should I use? (if I want protection against quantum attackers)

Post-quantum cryptography vs quantum cryptography

Post-quantum cryptography:

- uses classical infrastructures
- resists to attackers with quantum computers

Quantum cryptography:

- uses quantum mechanics and dedicated infrastructures
- resists to attackers with quantum computers

Which one should I use? (if I want protection against quantum attackers)

$\Rightarrow$ ANSSI recommends post-quantum cryptography[1]

ANSSI: Agence nationale de la sécurité des systèmes d'information (France)

[1] https://cyber.gouv.fr/en/publications/should-quantum-key-distribution-be-used-secure-communications

Should we be afraid of quantum attackers?

Should we be afraid of quantum attackers?

I don't know

Should we be afraid of quantum attackers?

I don't know . . . but I am not the one in making decisions

Should we be afraid of quantum attackers?

I don't know . . . but I am not the one in making decisions

NIST and ANSSI decided that the threat cannot be neglected

- even if the probability is very small, do we want to risk it?
- changing all cryptographic protocols takes time

NIST: the National Institute of Standards and Technology (USA)

ANSSI: Agence nationale de la sécurité des systèmes d'information (France)

Should we be afraid of quantum attackers?

I don't know . . . but I am not the one in making decisions

NIST and ANSSI decided that the threat cannot be neglected

- even if the probability is very small, do we want to risk it?
- changing all cryptographic protocols takes time

$\Rightarrow$ they encourage a progressive transition to post-quantum cryptography

Post-quantum cryptography is going to be deployed
in a (relatively) near future

NIST: the National Institute of Standards and Technology (USA)

ANSSI: Agence nationale de la sécurité des systèmes d'information (France)

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

https://csrc.nist.gov/projects/post-quantum-cryptography

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

2016 NIST first call for proposals

https://csrc.nist.gov/projects/post-quantum-cryptography

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

- 2016 NIST first call for proposals
- 2017 $\approx$ 70 submissions

https://csrc.nist.gov/projects/post-quantum-cryptography

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

- 2016 NIST first call for proposals
- 2017 $\approx$ 70 submissions
- 2022 NIST announced the first 4 selected algorithms

  (some others are still being analyzed)
  - 1 key exchange: Kyber (lattices)
  - 3 signatures: Dilithium (lattices), Falcon (lattices), SPHINCS+ (hash)

https://csrc.nist.gov/projects/post-quantum-cryptography

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

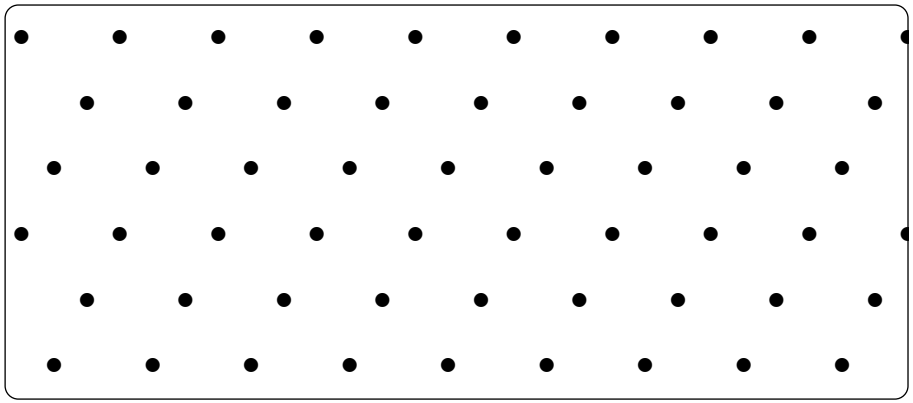
- 2016 NIST first call for proposals
- 2017 $\approx$ 70 submissions
- 2022 NIST announced the first 4 selected algorithms
  (some others are still being analyzed)
  - 1 key exchange: Kyber (lattices)
  - 3 signatures: Dilithium (lattices), Falcon (lattices), SPHINCS+ (hash)
- 2023 call for additional digital signatures ("not lattice-based")
  - 40 submissions, still under consideration

https://csrc.nist.gov/projects/post-quantum-cryptography

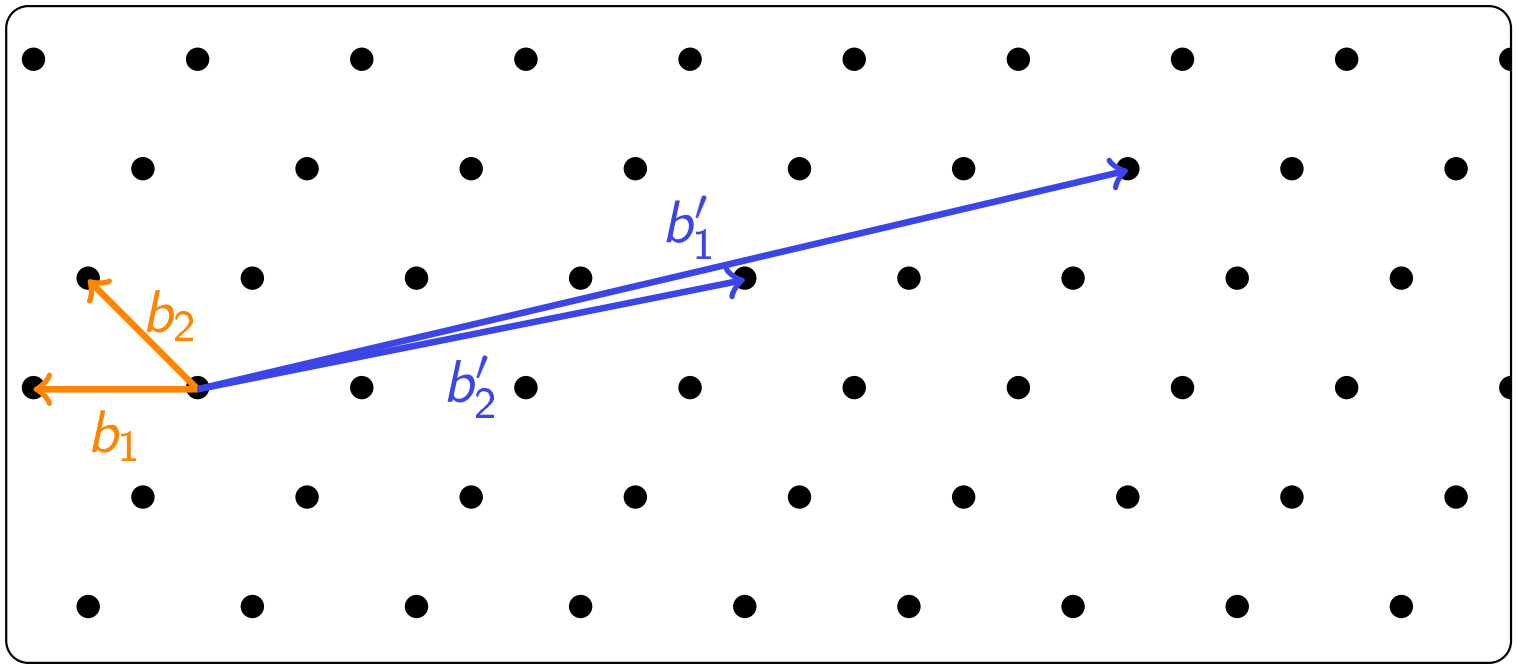
Plan of the talk

1. Lattices and lattice problems
2. Are lattice problems (quantumly) hard?
3. Constructing signatures from lattices
4. Falcon's signature scheme

Lattices and lattice problems



Lattices



- $\mathcal{L} = \{\sum_{i=1}^{n} x_i b_i \mid \forall i,\ x_i \in \mathbb{Z}\}$ is a lattice
- $(b_1, \ldots, b_n) =: B \in \mathrm{GL}_n(\mathbb{R})$ is a basis (not unique)
- n is the dimension (or rank)

Algorithmic problems on lattices

Example of problems:

(1) Testing equality of lattices

(2) Intersecting two lattices

(3) Computing a short basis of a lattice

Algorithmic problems on lattices

Example of problems:

(1) Testing equality of lattices

(2) Intersecting two lattices

(3) Computing a short basis of a lattice

Quiz: which ones are easy or hard?

easy: polynomial time hard: no polynomial time algorithm known

Algorithmic problems on lattices

Example of problems:

(1) Testing equality of lattices $\Rightarrow$ easy

(2) Intersecting two lattices $\Rightarrow$ easy

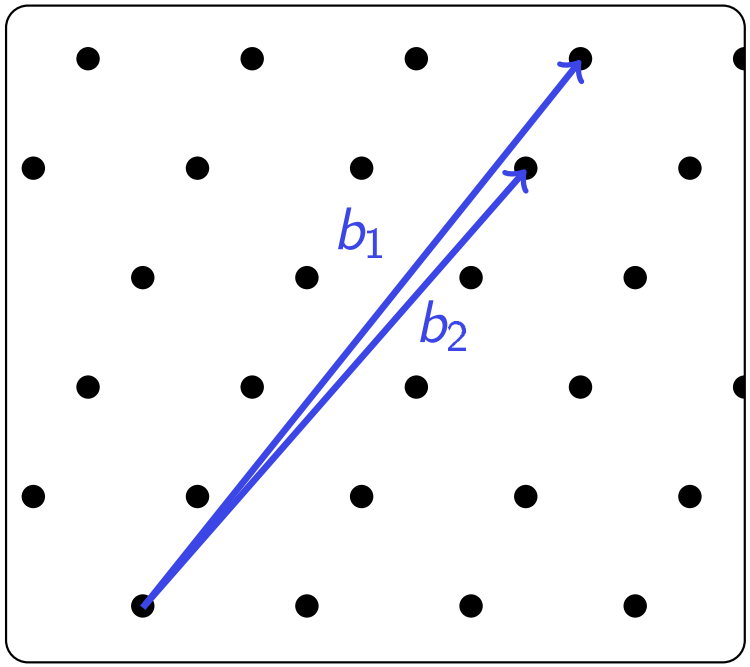
(3) Computing a short basis of a lattice $\Rightarrow$ hard

Quiz: which ones are easy or hard?

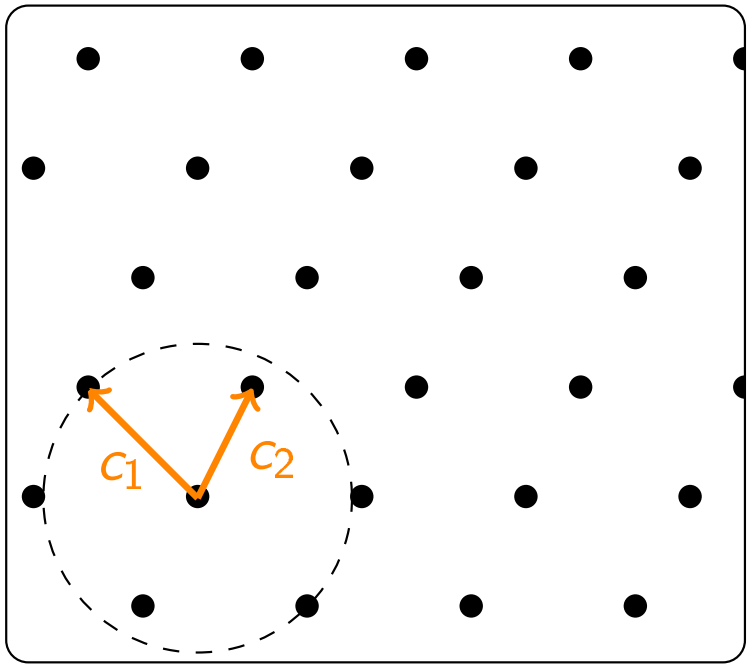
easy: polynomial time hard: no polynomial time algorithm known

Short basis problem

Input:



Output:

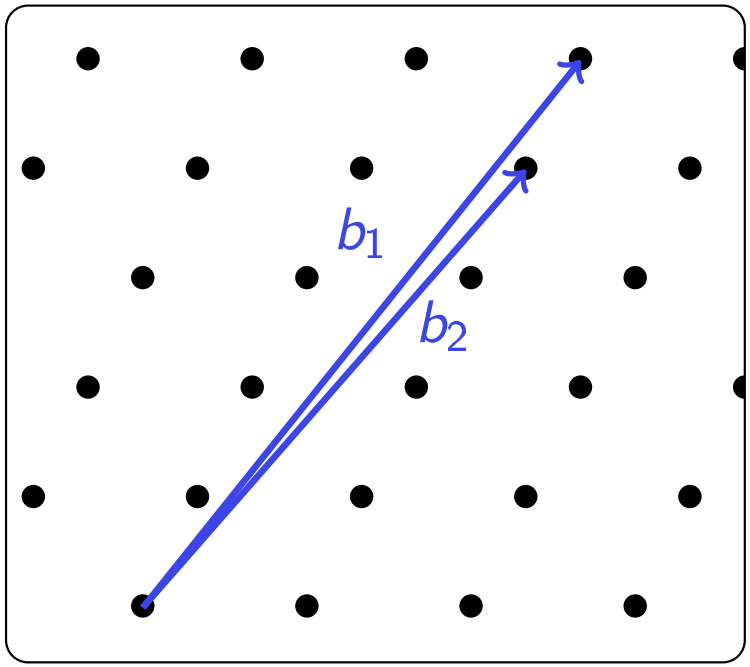


Shortest basis problem

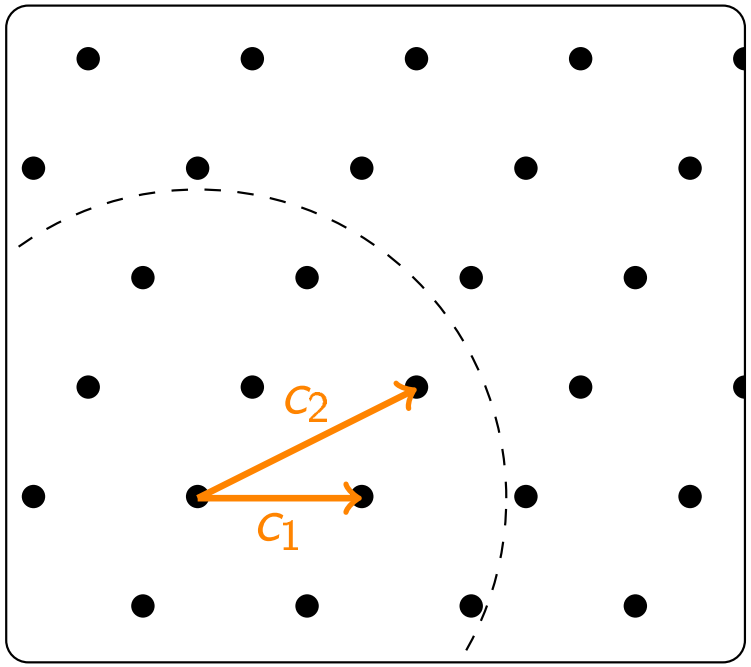
$$\max_i \|c_i\| \leq \min_{B' \text{ basis of } \mathcal{L}} \left(\max_i \|b'_i\| \right)$$

Short basis problem

Input:



Output:



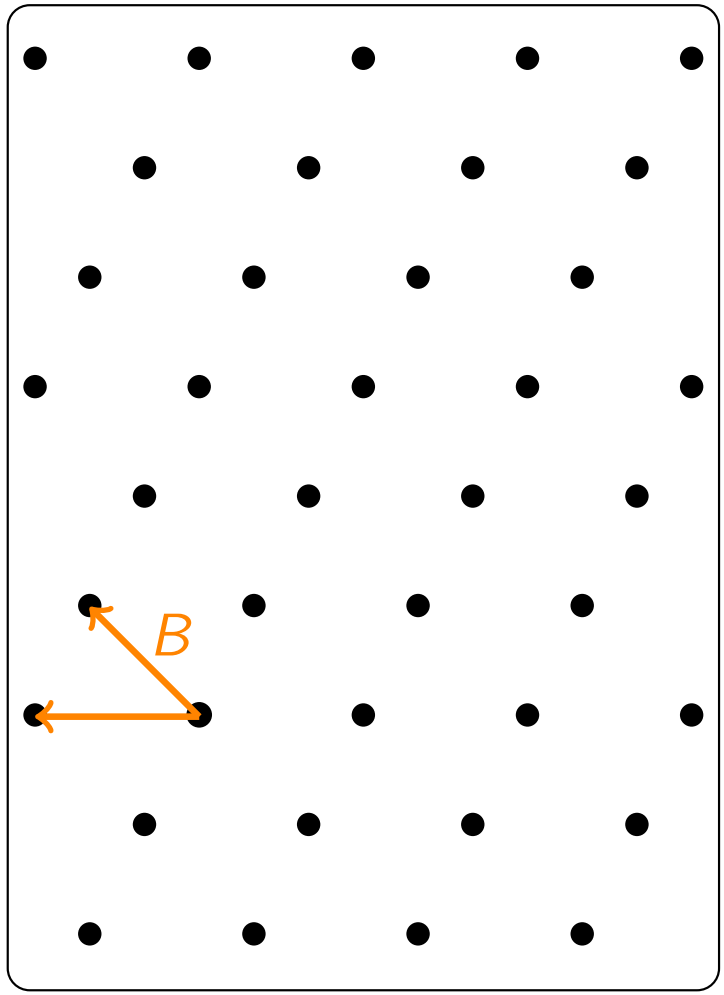
Approximate short basis problem

$$\max_i \|c_i\| \leq \gamma \cdot \min_{B' \text{ basis of } \mathcal{L}} \left(\max_i \|b'_i\| \right)$$

Digression: canonical representation

Representation of a lattice $\mathcal{L}$:

a basis $B \in \mathbb{Z}^{n \times n}$ of $\mathcal{L}$



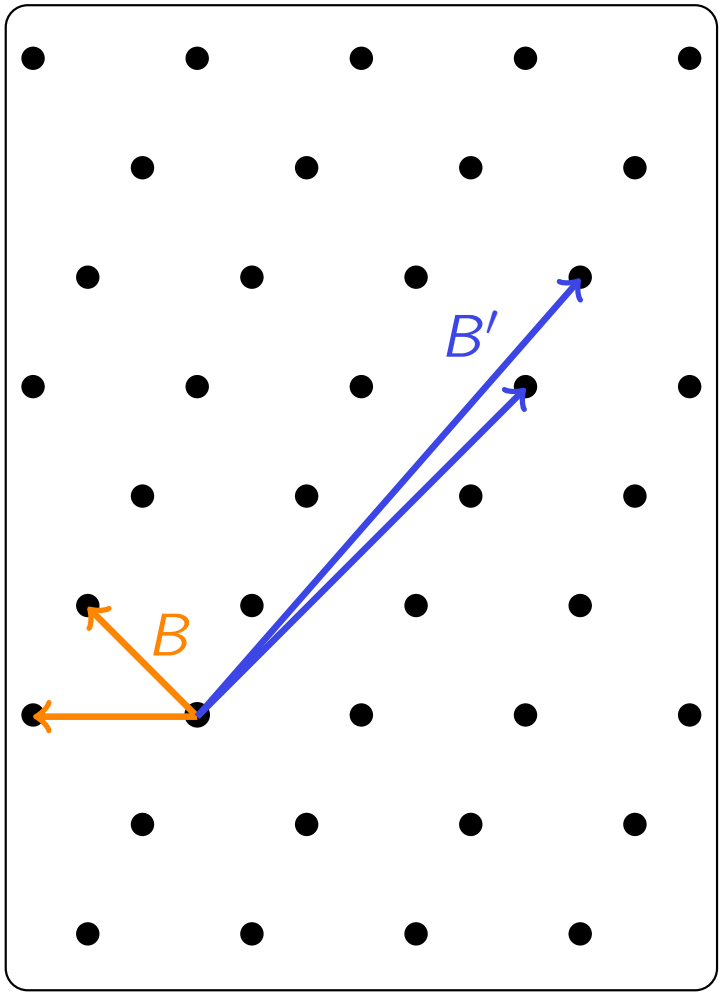
Digression: canonical representation

Representation of a lattice $\mathcal{L}$:

a basis $B \in \mathbb{Z}^{n \times n}$ of $\mathcal{L}$

Difficulty:

- the basis B is not unique
- some choices of B can make some algorithmic problems easier



Digression: canonical representation

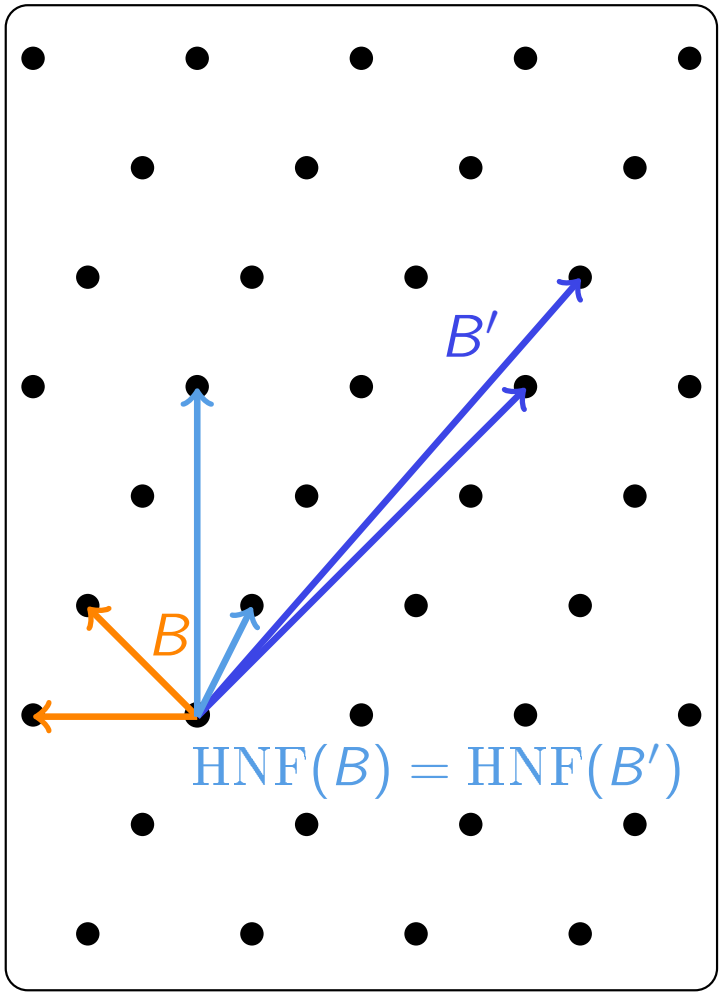
Representation of a lattice $\mathcal{L}$:
a basis $B \in \mathbb{Z}^{n \times n}$ of $\mathcal{L}$

Difficulty:

- the basis B is not unique
- some choices of B can make some algorithmic problems easier

Solution: take the Hermite Normal Form (HNF) of any B

- it is unique $(\mathrm{HNF}(B) = \mathrm{HNF}(B'))$
- it is efficiently computable



Digression: canonical representation

Representation of a lattice $\mathcal{L}$:
a basis $B \in \mathbb{Z}^{n \times n}$ of $\mathcal{L}$

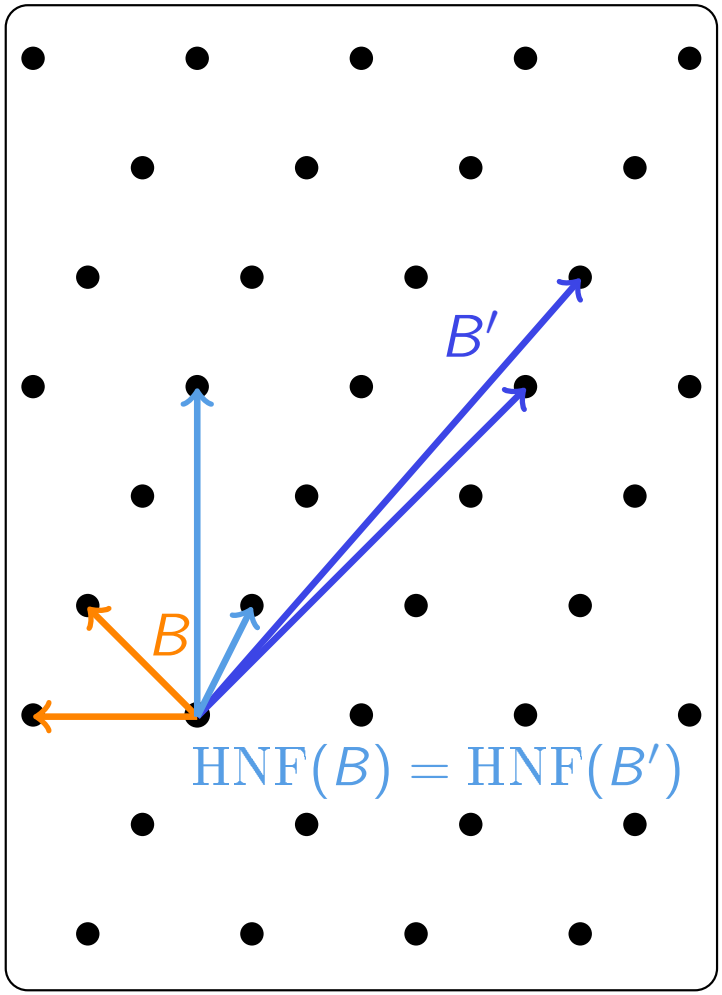
Difficulty:

- the basis B is not unique
- some choices of B can make some algorithmic problems easier

Solution: take the Hermite Normal Form (HNF) of any B

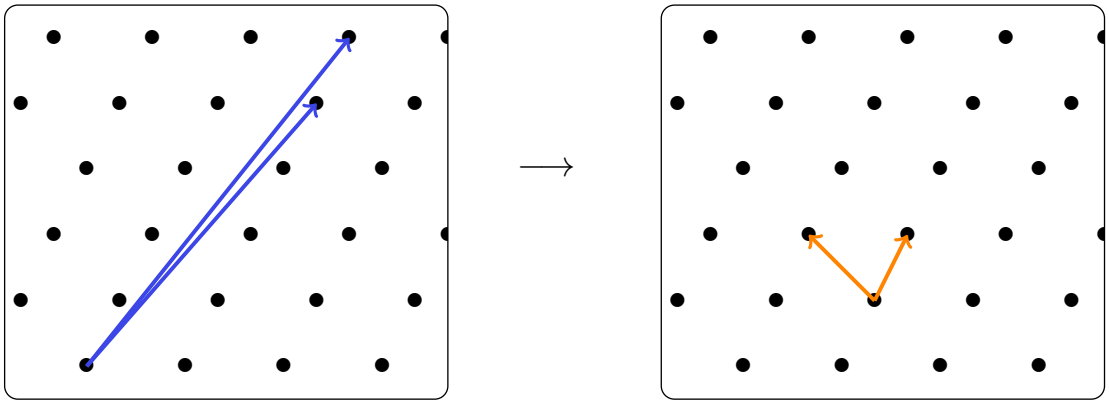
- it is unique $(\mathrm{HNF}(B) = \mathrm{HNF}(B'))$
- it is efficiently computable

$\Rightarrow$ canonical representation of $\mathcal{L}$
(i.e., worse basis ever)



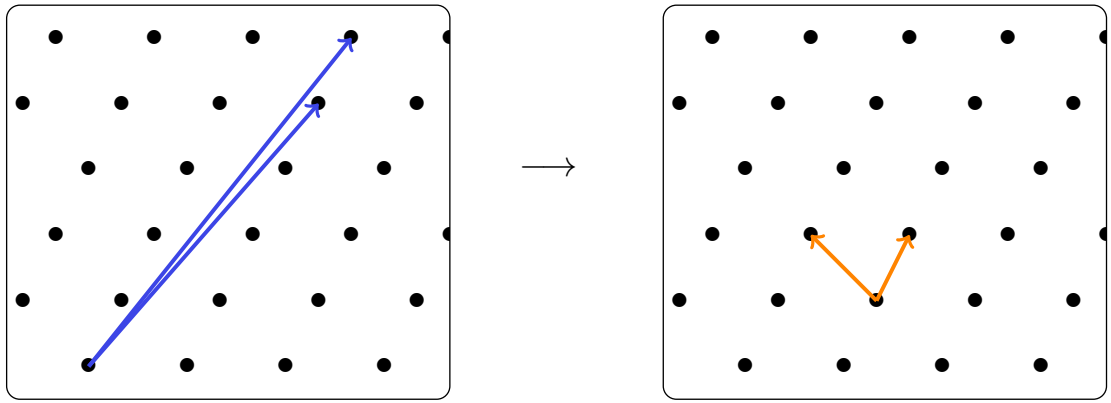
Section's conclusion

We have seen one lattice problem: the (approximate-)short basis problem



Section's conclusion

We have seen one lattice problem: the (approximate-)short basis problem



There exists many other lattice problems:

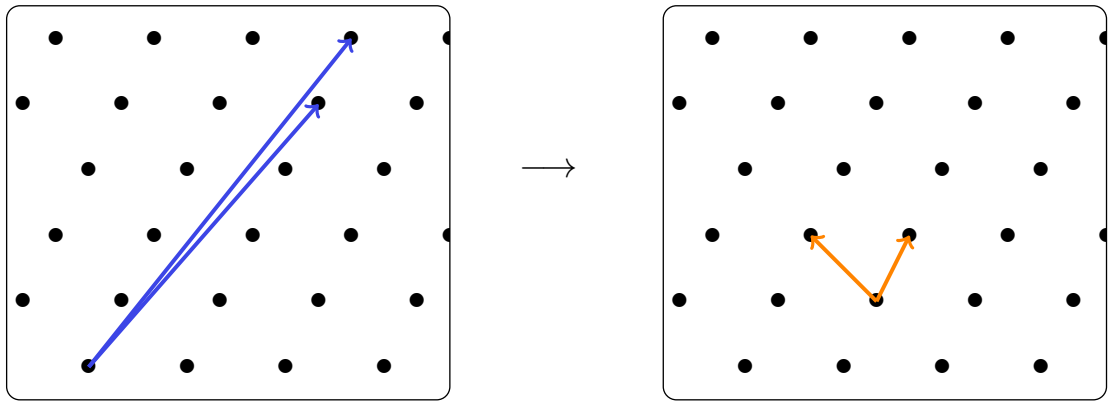
- the shortest vector problem
- the decoding problem (see later in the talk)
- ...

$\Rightarrow$ all of them are somehow equivalent in hardness

Nice survey of many lattice problems by Stephens-Davidowitz: http://www.noahsd.com/latticeproblems.pdf

Lattice reduction algorithms

or how to compute a short basis from a long one



Dimension 2: Lagrange-Gauss algorithm

video

Dimension 2: Lagrange-Gauss algorithm

video

Theorem: The algorithm

- finds a shortest basis
- runs in polynomial time

The LLL algorithm [LLL82]

Input: basis $B = (b_1, \ldots, b_n)$

[LLL82] Lenstra, Lenstra, and Lovász. Factoring polynomials with rational coefficients. Mathematische annalen.

The LLL algorithm [LLL82]

Input: basis $B = (b_1, \ldots, b_n)$

Main idea: improve the basis locally on blocks of dimension 2

(using Lagrange-Gauss algorithm)

[LLL82] Lenstra, Lenstra, and Lovász. Factoring polynomials with rational coefficients. Mathematische annalen.

The LLL algorithm [LLL82]

Input: basis $B = (b_1, \ldots, b_n)$

Main idea: improve the basis locally on blocks of dimension 2
(using Lagrange-Gauss algorithm)

Algorithm:

- while there exists i such that (b_i, b_{i+1}) is not a shortest basis of L_i
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

[LLL82] Lenstra, Lenstra, and Lovász. Factoring polynomials with rational coefficients. Mathematische annalen.

The LLL algorithm [LLL82]

Input: basis $B = (b_1, \ldots, b_n)$

Main idea: improve the basis locally on blocks of dimension 2
(using Lagrange-Gauss algorithm)

Algorithm:

- while there exists i such that (b_i, b_{i+1}) is not a shortest basis of L_i
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- finds an approximate short basis with $\gamma = 2^n$

[LLL82] Lenstra, Lenstra, and Lovász. Factoring polynomials with rational coefficients. Mathematische annalen.

The LLL algorithm [LLL82]

Input: basis $B = (b_1, \ldots, b_n)$

Main idea: improve the basis locally on blocks of dimension 2
(using Lagrange-Gauss algorithm)

Algorithm:

- while there exists i such that (b_i, b_{i+1}) is not a shortest basis of L_i
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- finds an approximate short basis with $\gamma = 2^n$
- does not run in polynomial time

[LLL82] Lenstra, Lenstra, and Lovász. Factoring polynomials with rational coefficients. Mathematische annalen.

The LLL algorithm [LLL82]

Input: basis $B = (b_1, \ldots, b_n)$

Main idea: improve the basis locally on blocks of dimension 2
(using Lagrange-Gauss algorithm)

Algorithm:

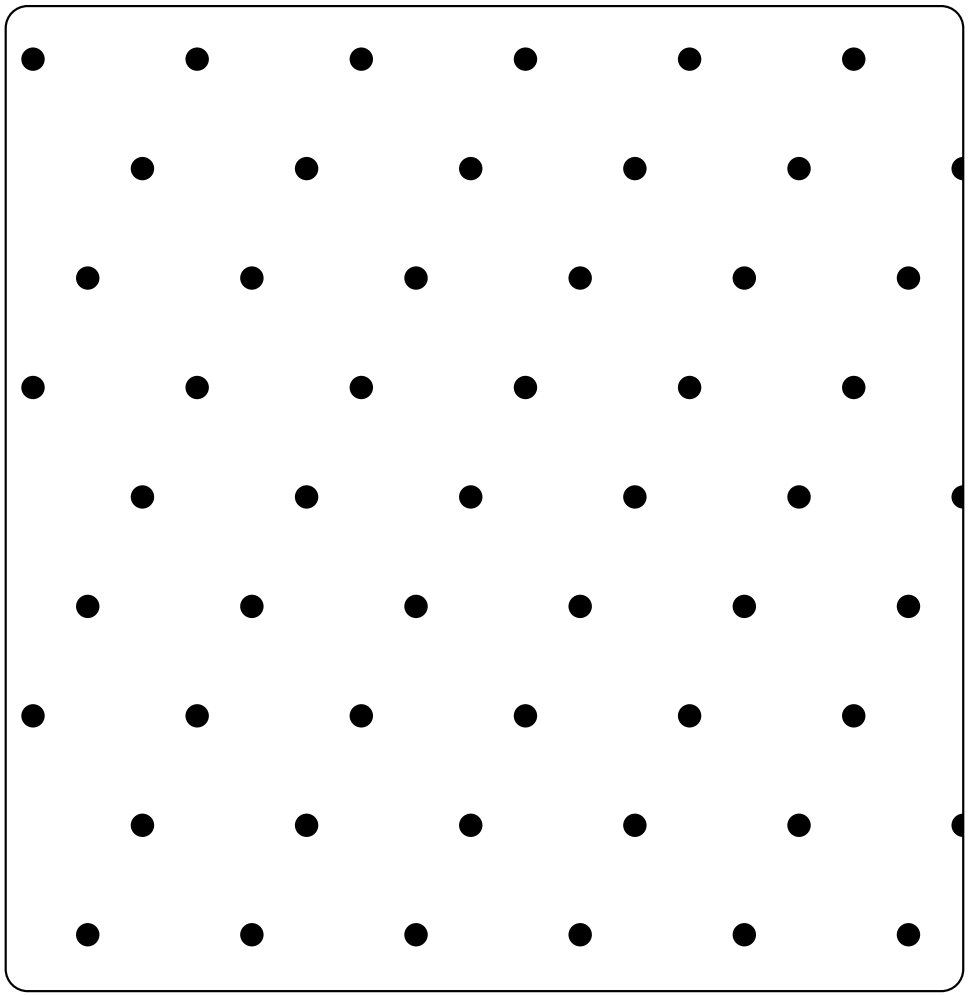
- while there exists i such that (b_i, b_{i+1}) is not a γ'-short basis of L_i with $\gamma' = 4/3$
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- finds an approximate short basis with $\gamma = 2^n$
- runs in polynomial time

[LLL82] Lenstra, Lenstra, and Lovász. Factoring polynomials with rational coefficients. Mathematische annalen.

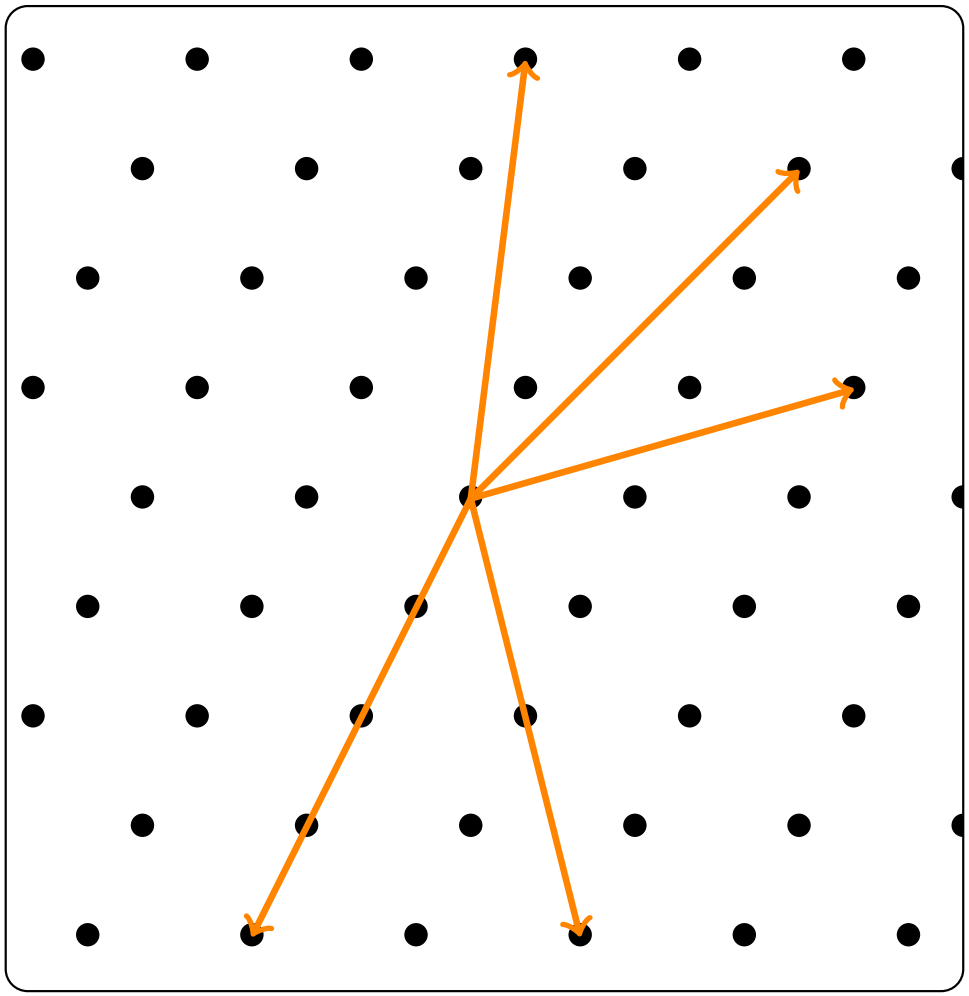
Sieving algorithm [AKS01]



Sieving:

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]

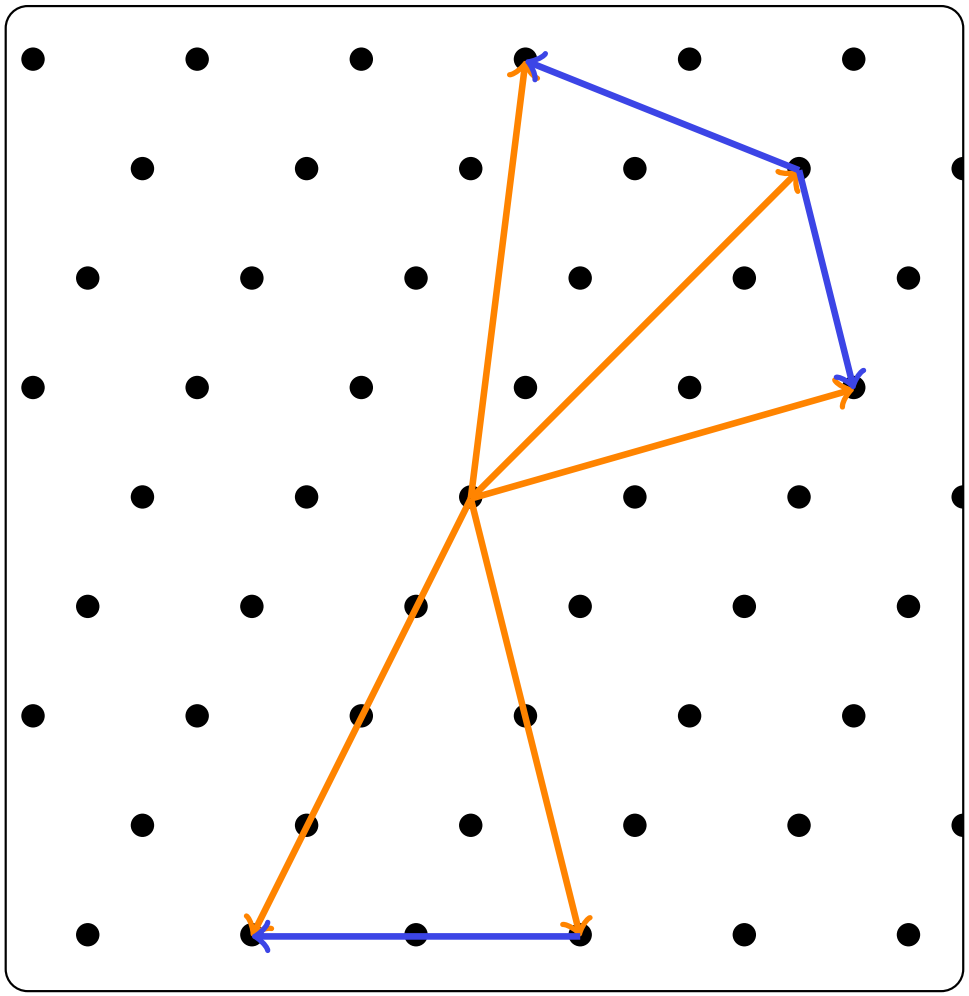


Sieving:

- Create many large vectors

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]

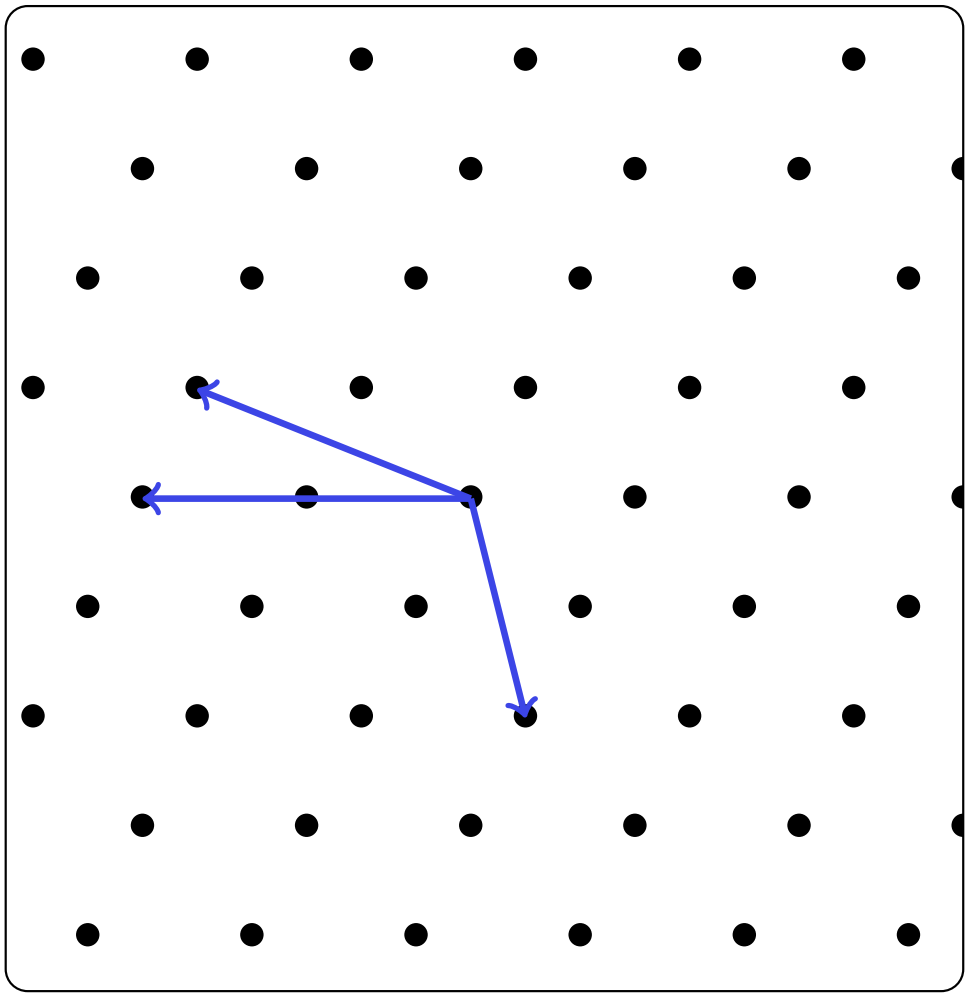


Sieving:

- Create many large vectors
- Subtract close ones to create shorter vectors

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]

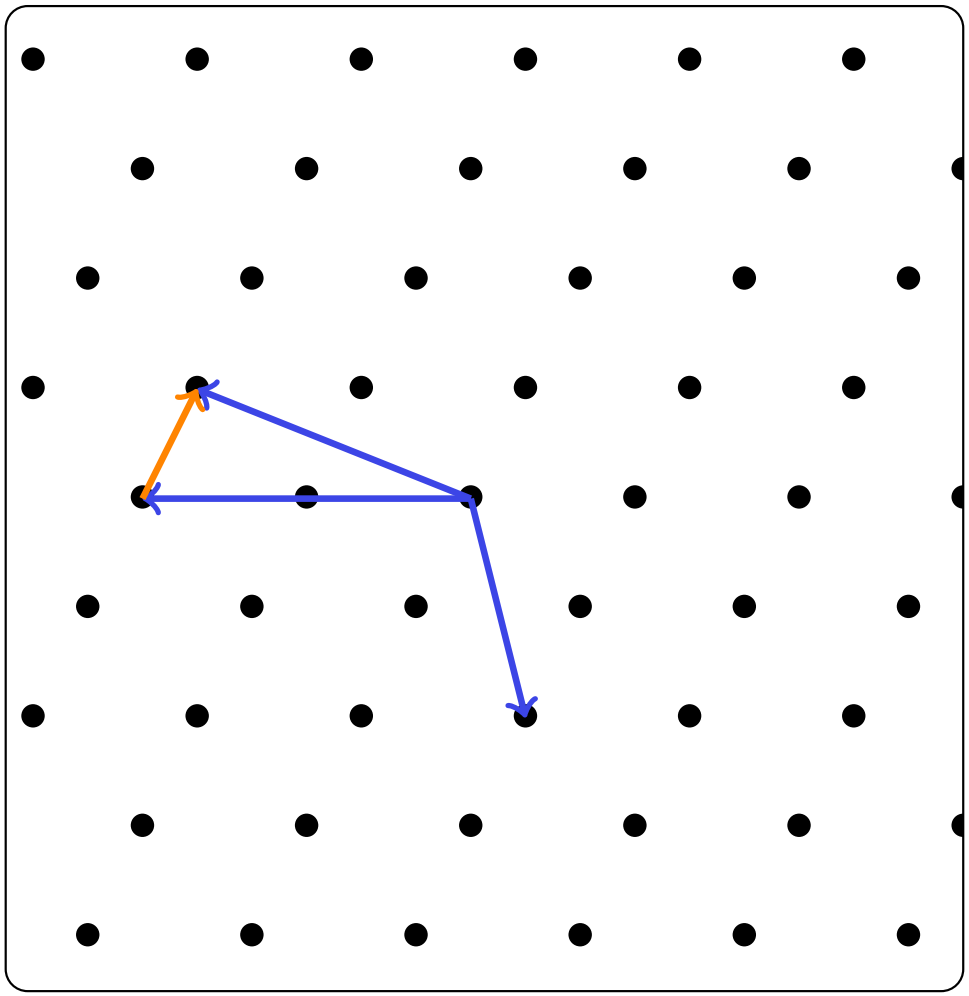


Sieving:

- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]

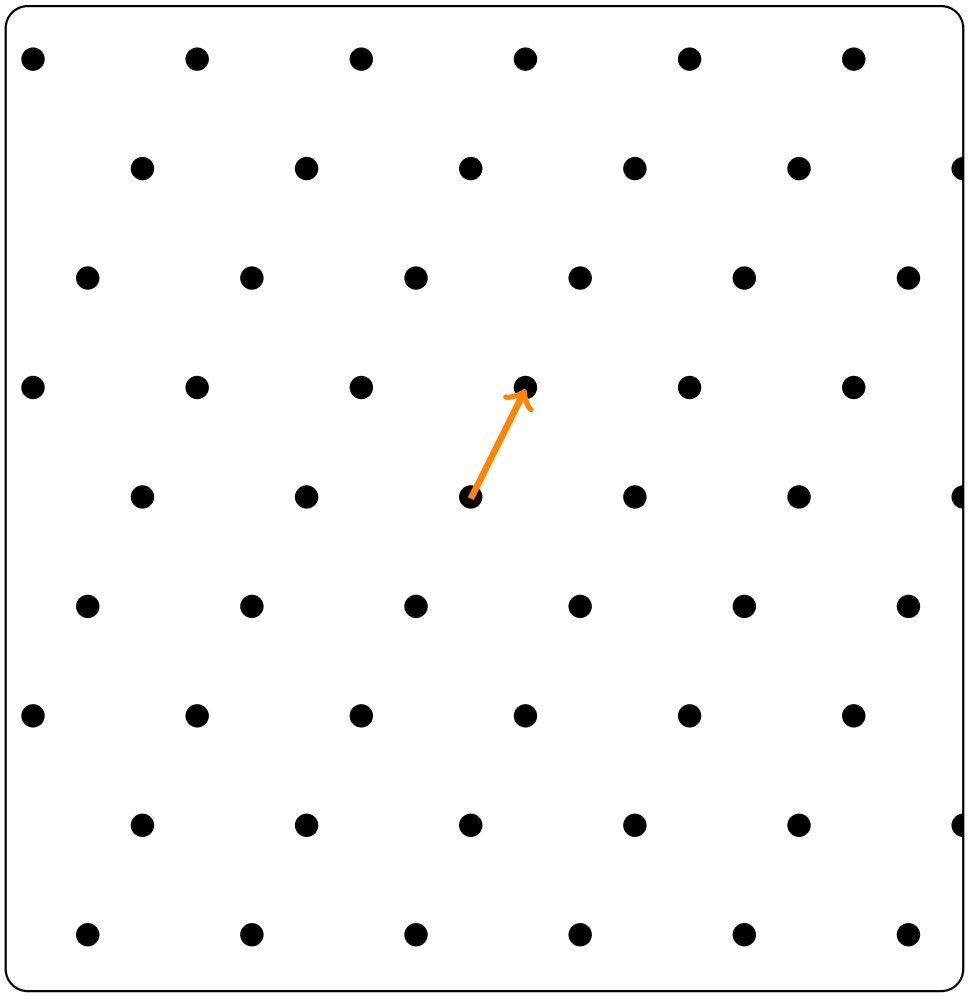


Sieving:

- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]

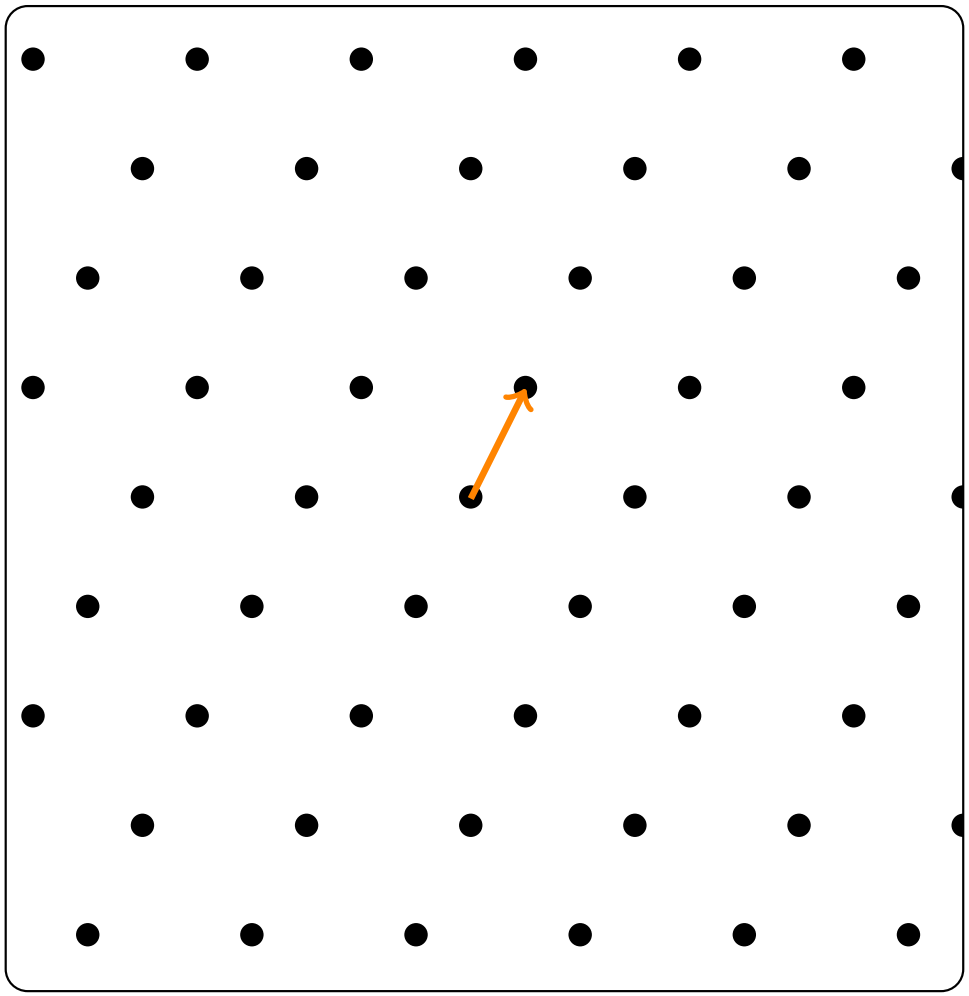


Sieving:

- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]



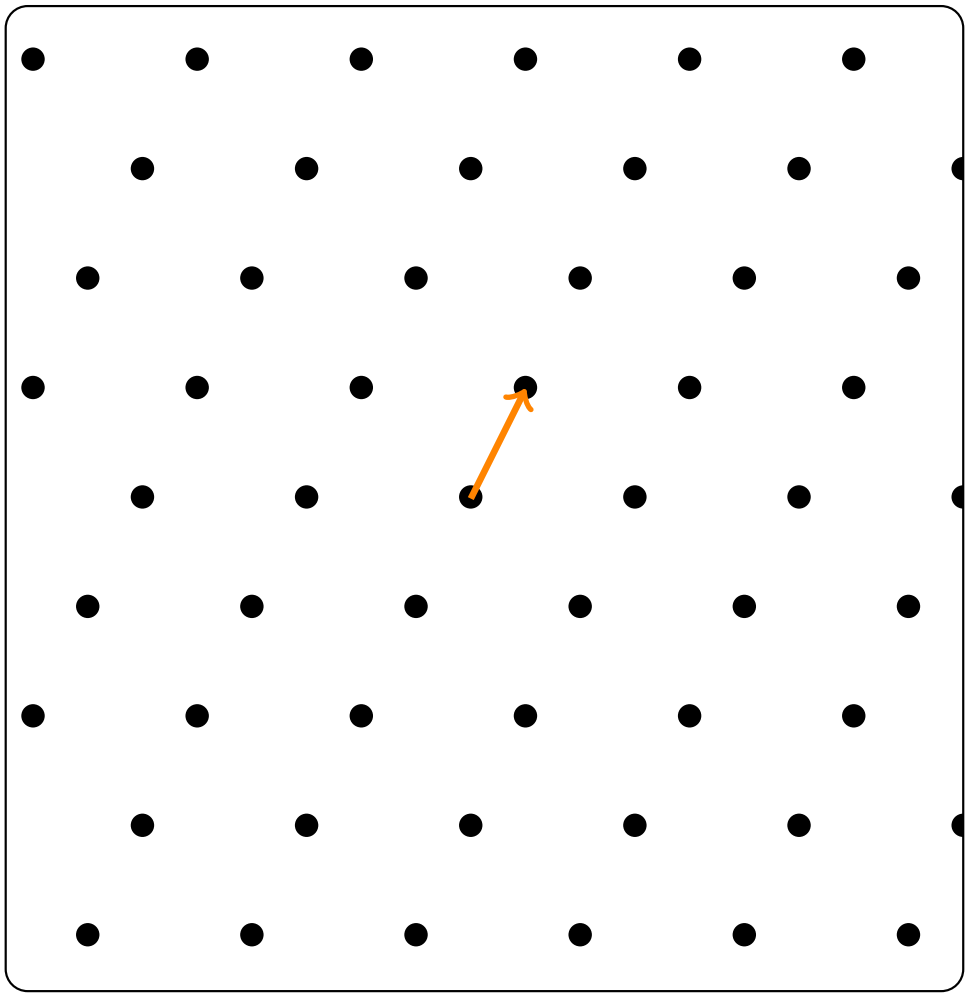
Sieving:

- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

Size of the initial list: $2^{O(n)}$

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]



Sieving:

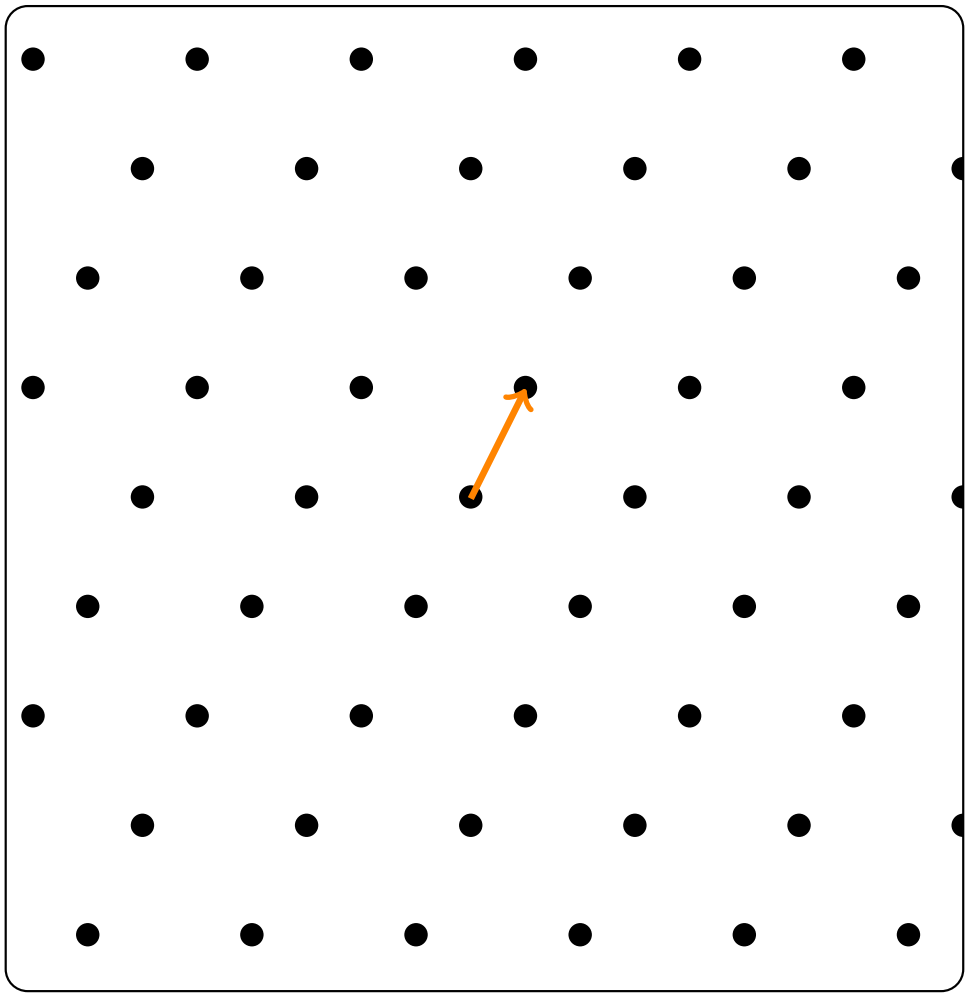
- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

Size of the initial list: $2^{O(n)}$

- finds a shortest basis

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Sieving algorithm [AKS01]



Sieving:

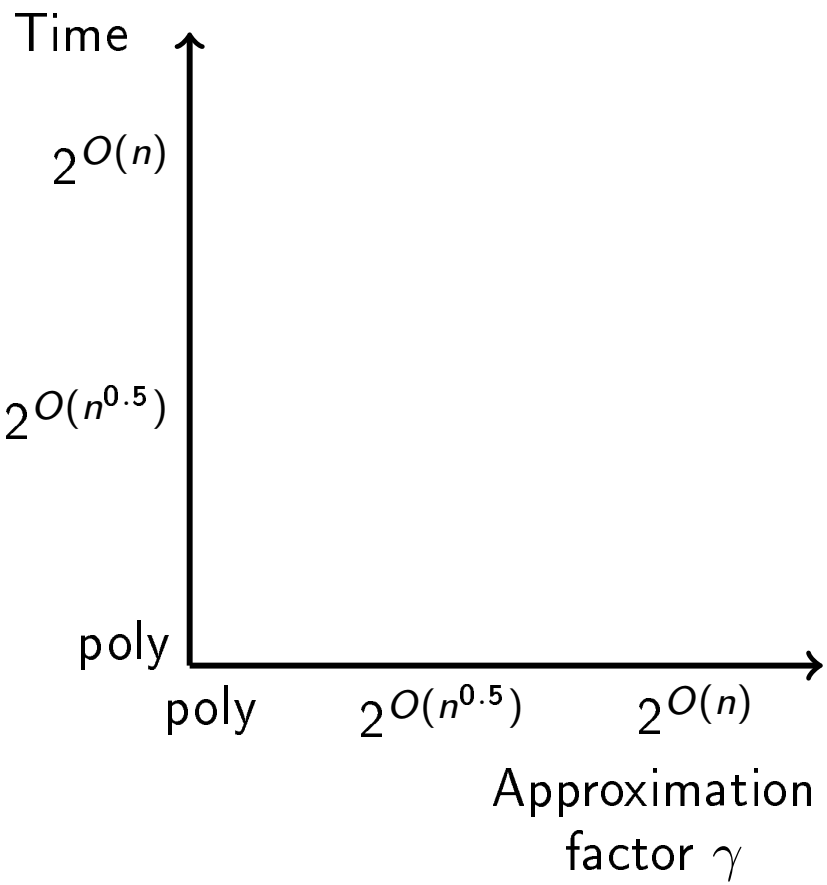
- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

Size of the initial list: $2^{O(n)}$

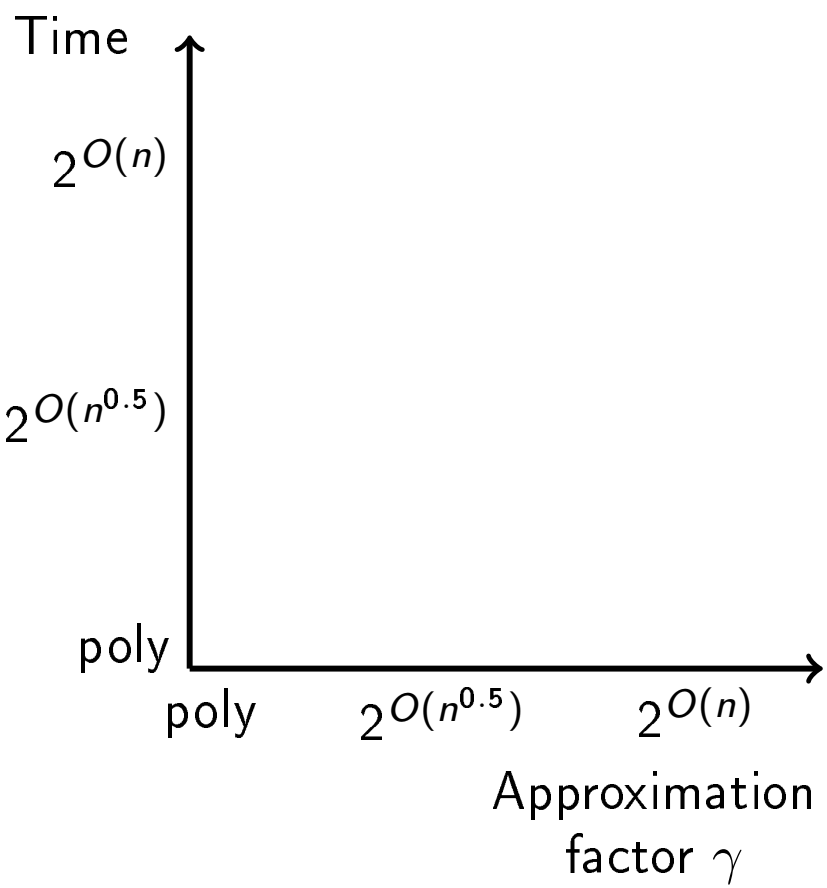
- finds a shortest basis
- runs in time $2^{O(n)}$

[AKS01] Ajtai, Kumar, and Sivakumar. A sieve algorithm for the shortest lattice vector problem. STOC

Summary and BKZ



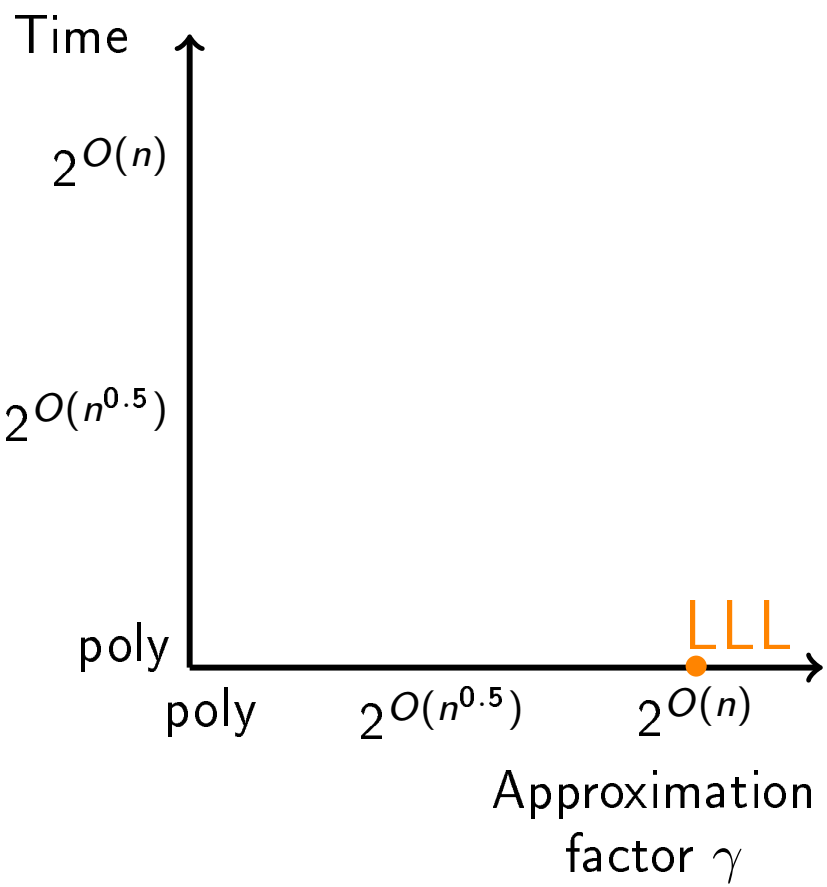
Summary and BKZ



Lagrange-Gauss algorithm: dim 2

- shortest basis
- polynomial time

Summary and BKZ



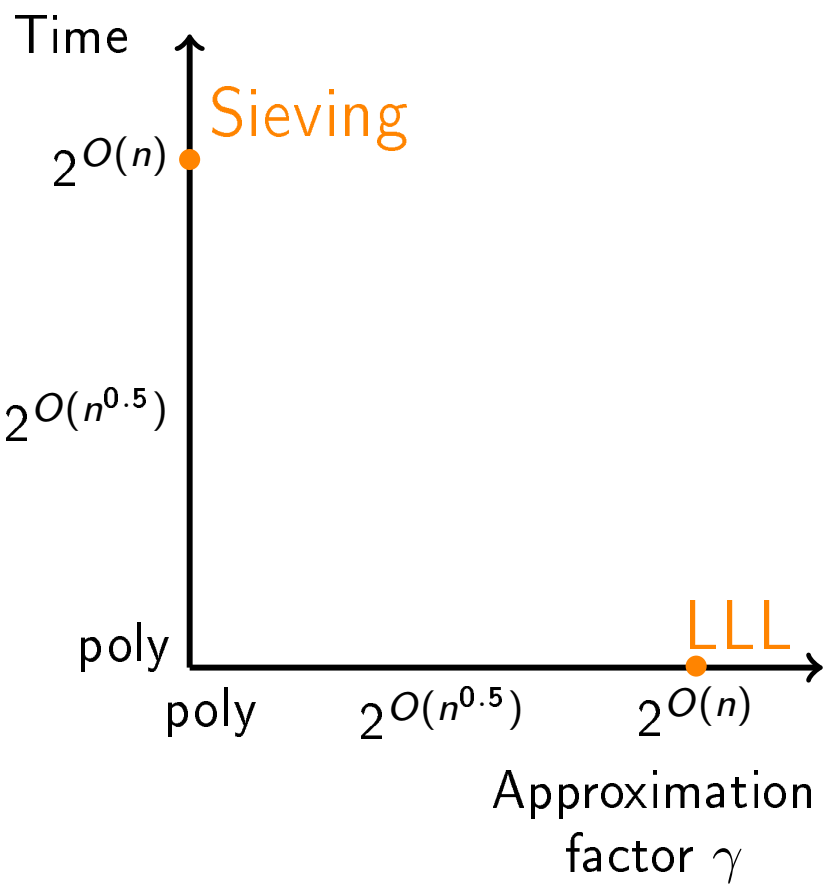
Lagrange-Gauss algorithm: dim 2

- shortest basis
- polynomial time

LLL algorithm: dim n

- γ-short basis with $\gamma = 2^n$
- polynomial time

Summary and BKZ



Lagrange-Gauss algorithm: dim 2

- shortest basis
- polynomial time

LLL algorithm: dim n

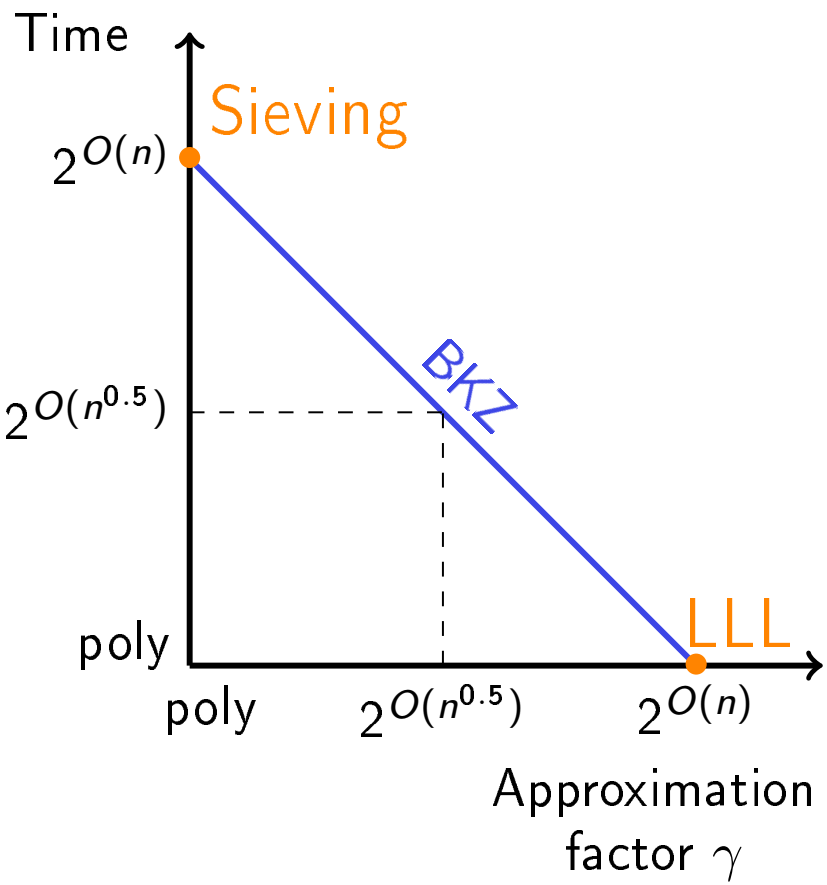
- γ-short basis with $\gamma = 2^n$
- polynomial time

Sieving algorithm: dim n

- shortest basis
- time $2^{O(n)}$

Summary and BKZ

BKZ trade-offs



Lagrange-Gauss algorithm: dim 2

- shortest basis
- polynomial time

LLL algorithm: dim n

- γ-short basis with $\gamma = 2^n$
- polynomial time

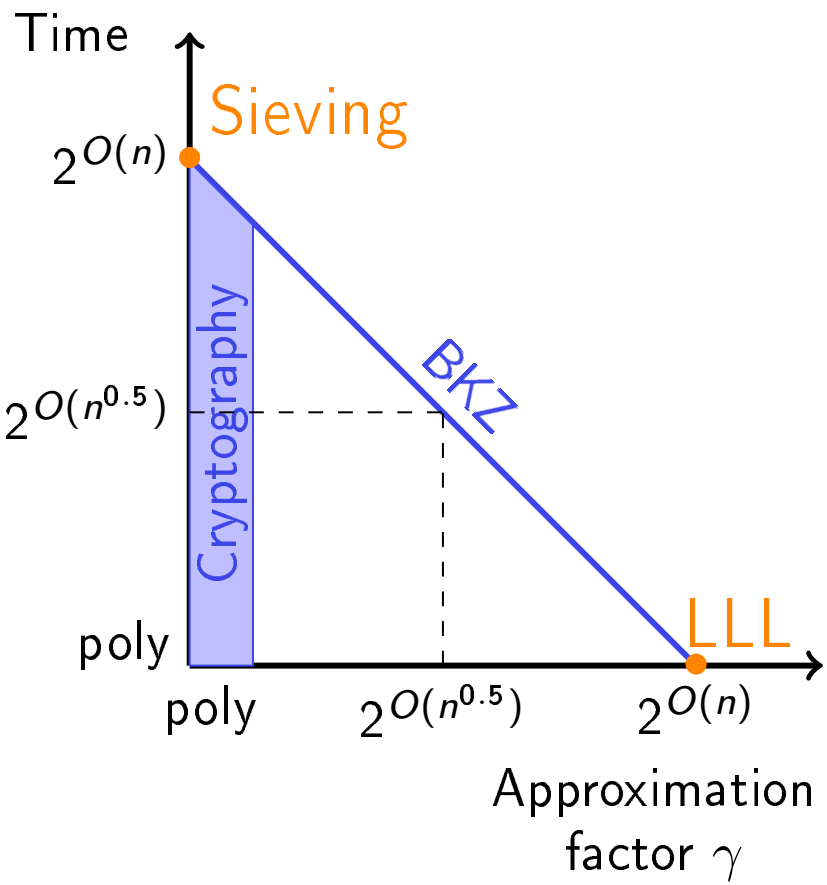
Sieving algorithm: dim n

- shortest basis
- time $2^{O(n)}$

BKZ algorithm: combine LLL + Sieving $\Rightarrow$ various trade-offs

Summary and BKZ

BKZ trade-offs



Lagrange-Gauss algorithm: dim 2

- shortest basis
- polynomial time

LLL algorithm: dim n

- γ-short basis with $\gamma = 2^n$
- polynomial time

Sieving algorithm: dim n

- shortest basis
- time $2^{O(n)}$

BKZ algorithm: combine LLL + Sieving $\Rightarrow$ various trade-offs

Some concrete numbers

Finding a shortest basis in practice:

- $n = 2 \rightsquigarrow$ easy, very efficient in practice

Some concrete numbers

Finding a shortest basis in practice:

- $n = 2 \rightsquigarrow$ easy, very efficient in practice
- up to $n = 60$ or $n = 80 \rightsquigarrow$ a few minutes on a personal laptop

Some concrete numbers

Finding a shortest basis in practice:

- $n = 2 \rightsquigarrow$ easy, very efficient in practice
- up to $n = 60$ or $n = 80 \rightsquigarrow$ a few minutes on a personal laptop
- up to $n = 180 \rightsquigarrow$ few days on big computers with good code [DSW21]

[DSW21] Ducas, Stevens, van Woerden. Advanced Lattice Sieving on GPUs, with Tensor Cores.

Some concrete numbers

Finding a shortest basis in practice:

- $n = 2 \rightsquigarrow$ easy, very efficient in practice
- up to $n = 60$ or $n = 80 \rightsquigarrow$ a few minutes on a personal laptop
- up to $n = 180 \rightsquigarrow$ few days on big computers with good code [DSW21]
- from $n = 400$ to $n = 1000 \rightsquigarrow$ cryptography

[DSW21] Ducas, Stevens, van Woerden. Advanced Lattice Sieving on GPUs, with Tensor Cores.

Section's conclusion

Takeaway: If the dimension n is large enough (e.g., $n \geq 1000$) computing short bases of lattices with approximation factor $\gamma \lesssim n$ is very hard.

Section's conclusion

Takeaway: If the dimension n is large enough (e.g., $n \geq 1000$) computing short bases of lattices with approximation factor $\gamma \lesssim n$ is very hard.

- even with a quantum computer

Section's conclusion

Takeaway: If the dimension n is large enough (e.g., $n \geq 1000$) computing short bases of lattices with approximation factor $\gamma \lesssim n$ is very hard.

- even with a quantum computer

But...

Section's conclusion

Takeaway: If the dimension n is large enough (e.g., $n \geq 1000$) computing short bases of lattices with approximation factor $\gamma \lesssim n$ is very hard.

- even with a quantum computer

But...

- It is hard if you want an algorithm that works on all lattices

Section's conclusion

Takeaway: If the dimension n is large enough (e.g., $n \geq 1000$) computing short bases of lattices with approximation factor $\gamma \lesssim n$ is very hard.

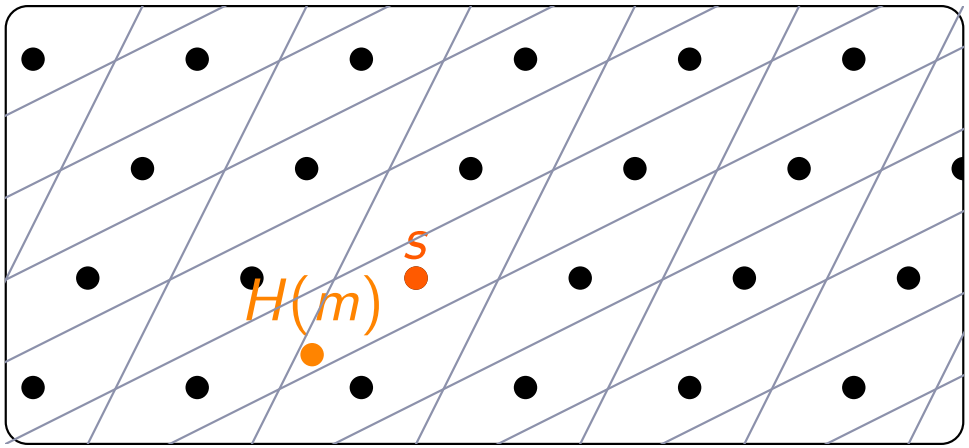
- even with a quantum computer

But...

- It is hard if you want an algorithm that works on all lattices
- It may be easy for some lattices!
  (We will come back to this...)

Constructing signatures from lattices

(using hash-and-sign technique)



Digital Signature

Three algorithms:

- $\mathsf{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathsf{Sign}(m, sk) \rightsquigarrow s$
- $\mathsf{Verify}(m, s, pk) \rightsquigarrow$ yes/no

Digital Signature

Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Sign}(m, sk) \rightsquigarrow s$
- $\mathrm{Verify}(m, s, pk) \rightsquigarrow$ yes/no

Alice		Bob
$(pk, sk) \leftarrow \mathrm{KeyGen}()$	$\xrightarrow{pk}$	store pk

Digital Signature

Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Sign}(m, sk) \rightsquigarrow s$
- $\mathrm{Verify}(m, s, pk) \rightsquigarrow$ yes/no

Alice		Bob
$(pk, sk) \leftarrow \mathrm{KeyGen}()$	$\xrightarrow{pk}$	store pk
message m $s \leftarrow \mathrm{Sign}(m, sk)$	$\xrightarrow{m,s}$	$\mathrm{Verify}(m, s, pk)$? (message really from Alice?)

Digital Signature

Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Sign}(m, sk) \rightsquigarrow s$
- $\mathrm{Verify}(m, s, pk) \rightsquigarrow$ yes/no

Alice		Bob
$(pk, sk) \leftarrow \mathrm{KeyGen}()$	$\xrightarrow{pk}$	store pk
message m $s \leftarrow \mathrm{Sign}(m, sk)$	$\xrightarrow{m,s}$	$\mathrm{Verify}(m, s, pk)$? (message really from Alice?)

Correctness: $\mathrm{Verify}(m, s, pk) = \mathrm{yes}$ (if $s \leftarrow \mathrm{Sign}(m, sk)$ and $(pk, sk) \leftarrow \mathrm{KeyGen}$)

Digital Signature

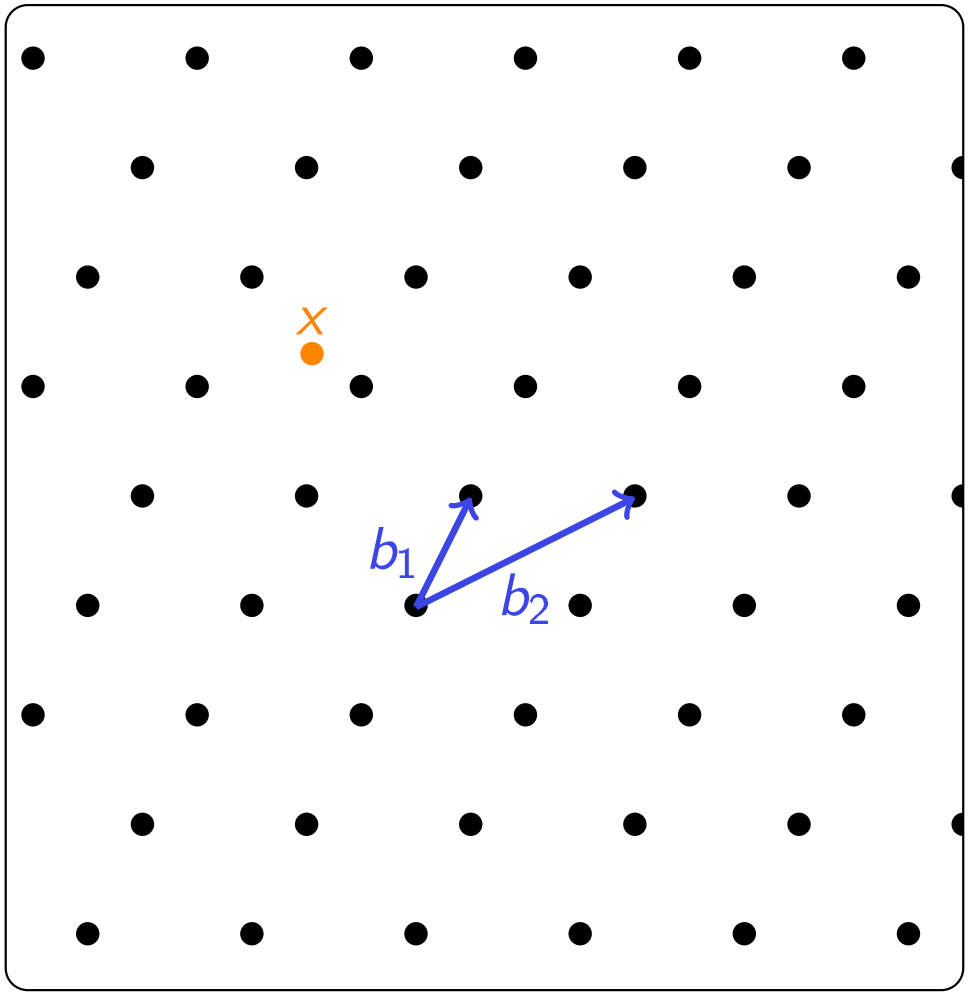
Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Sign}(m, sk) \rightsquigarrow s$
- $\mathrm{Verify}(m, s, pk) \rightsquigarrow$ yes/no

Alice		Bob
$(pk, sk) \leftarrow \mathrm{KeyGen}()$	$\xrightarrow{pk}$	store pk
message m $s \leftarrow \mathrm{Sign}(m, sk)$	$\xrightarrow{m,s}$	$\mathrm{Verify}(m, s, pk)$? (message really from Alice?)

Correctness: $\mathrm{Verify}(m, s, pk) = \mathrm{yes}$ (if $s \leftarrow \mathrm{Sign}(m, sk)$ and $(pk, sk) \leftarrow \mathrm{KeyGen}$)

Security: an attacker not knowing sk cannot forge valid pairs (m, s)
(i.e., such that $\mathrm{Verify}(m, s, pk) = \mathrm{yes}$)

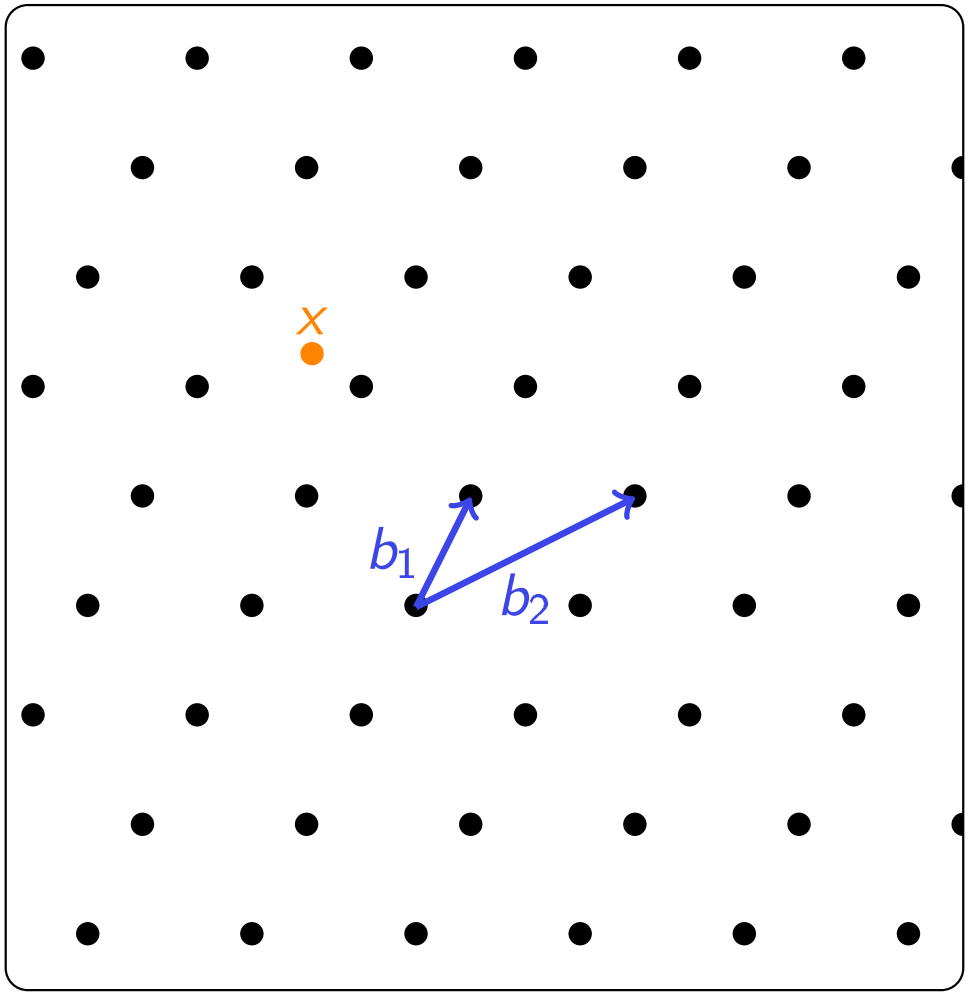
Decoding in a lattice using a short basis



Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

Objective: find $s \in \mathcal{L}$ close to x

Decoding in a lattice using a short basis

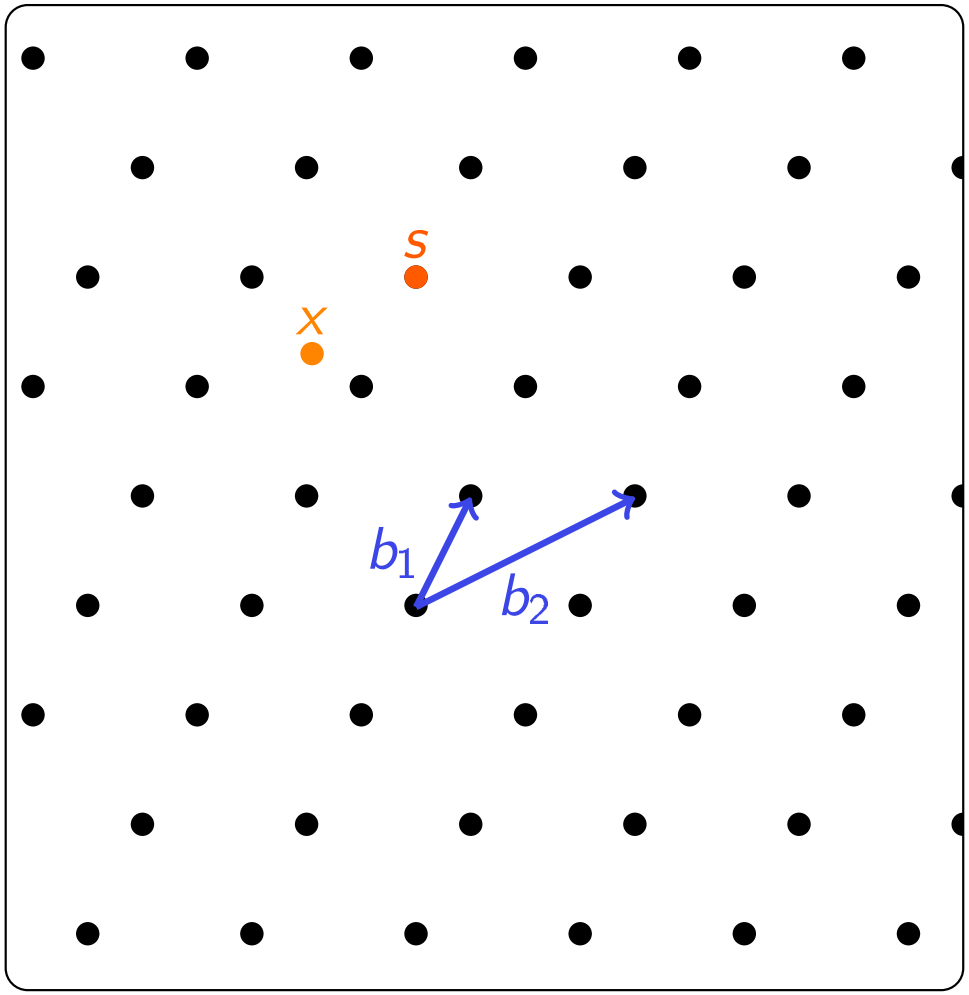


Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

Objective: find $s \in \mathcal{L}$ close to x

Algo: round each coordinate

Decoding in a lattice using a short basis



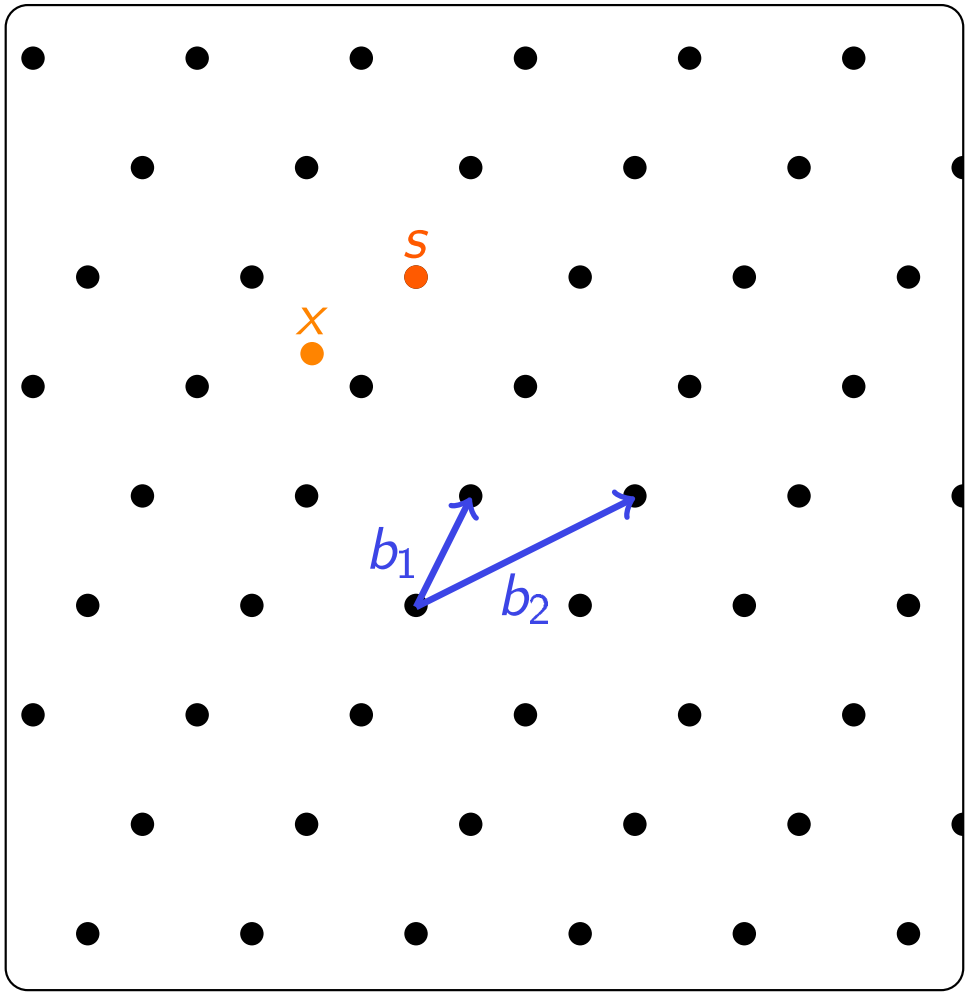
Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

Objective: find $s \in \mathcal{L}$ close to x

Algo: round each coordinate

Output: $s = 4 \cdot b_1 - 1 \cdot b_2$

Decoding in a lattice using a short basis



Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

Objective: find $s \in \mathcal{L}$ close to x

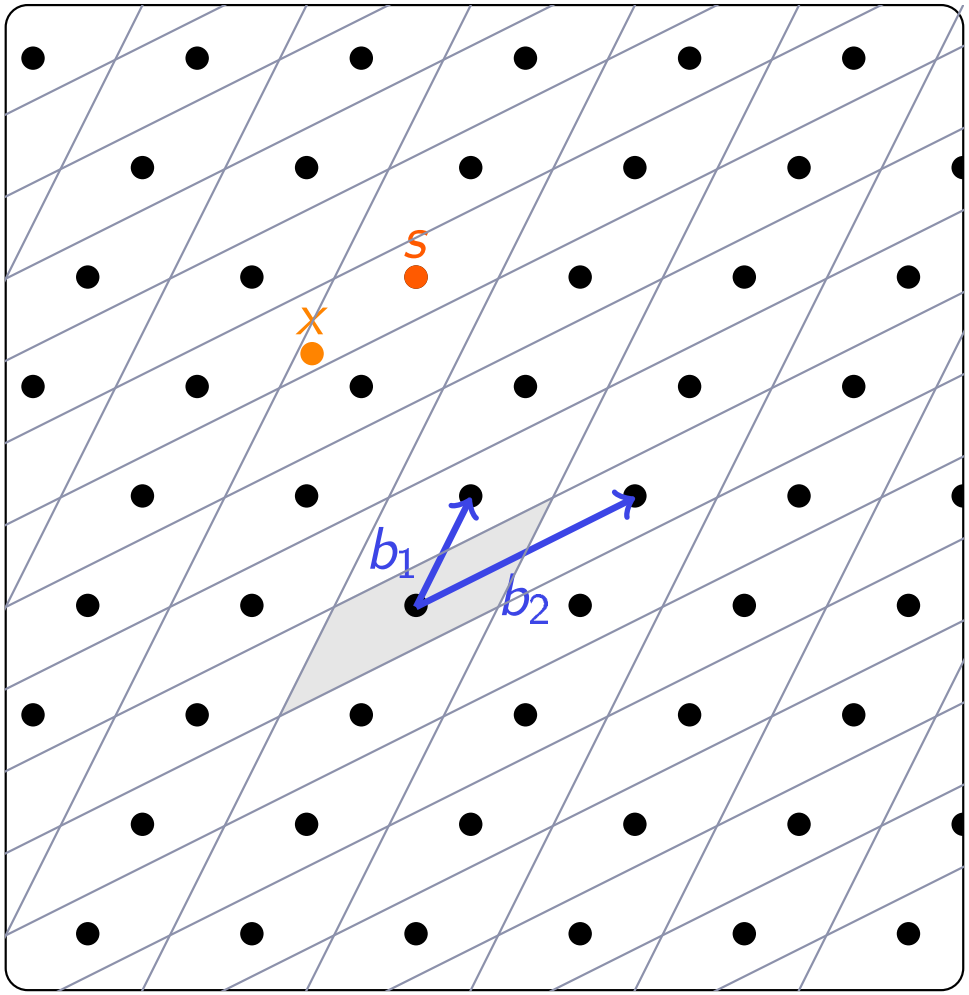
Algo: round each coordinate

Output: $s = 4 \cdot b_1 - 1 \cdot b_2$

The smaller the basis, the closer the solution

(called Babai's round-off algorithm)

Decoding in a lattice using a short basis



Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

Objective: find $s \in \mathcal{L}$ close to x

Algo: round each coordinate

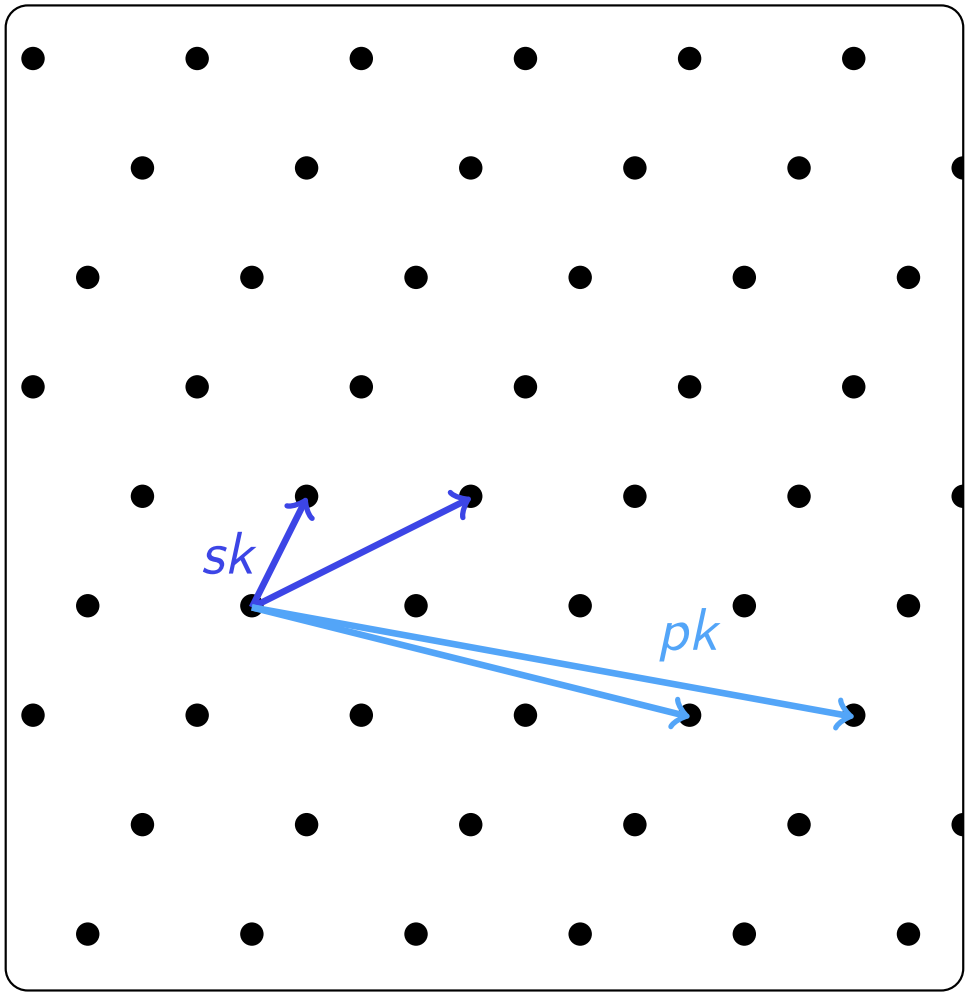
Output: $s = 4 \cdot b_1 - 1 \cdot b_2$

The smaller the basis, the closer the solution

(called Babai's round-off algorithm)

$$= \left\{ x_1 b_1 + x_2 b_2 \,\middle|\, |x_i| \leq \tfrac{1}{2} \right\}$$

Hash-and-sign: first idea [GGH97]

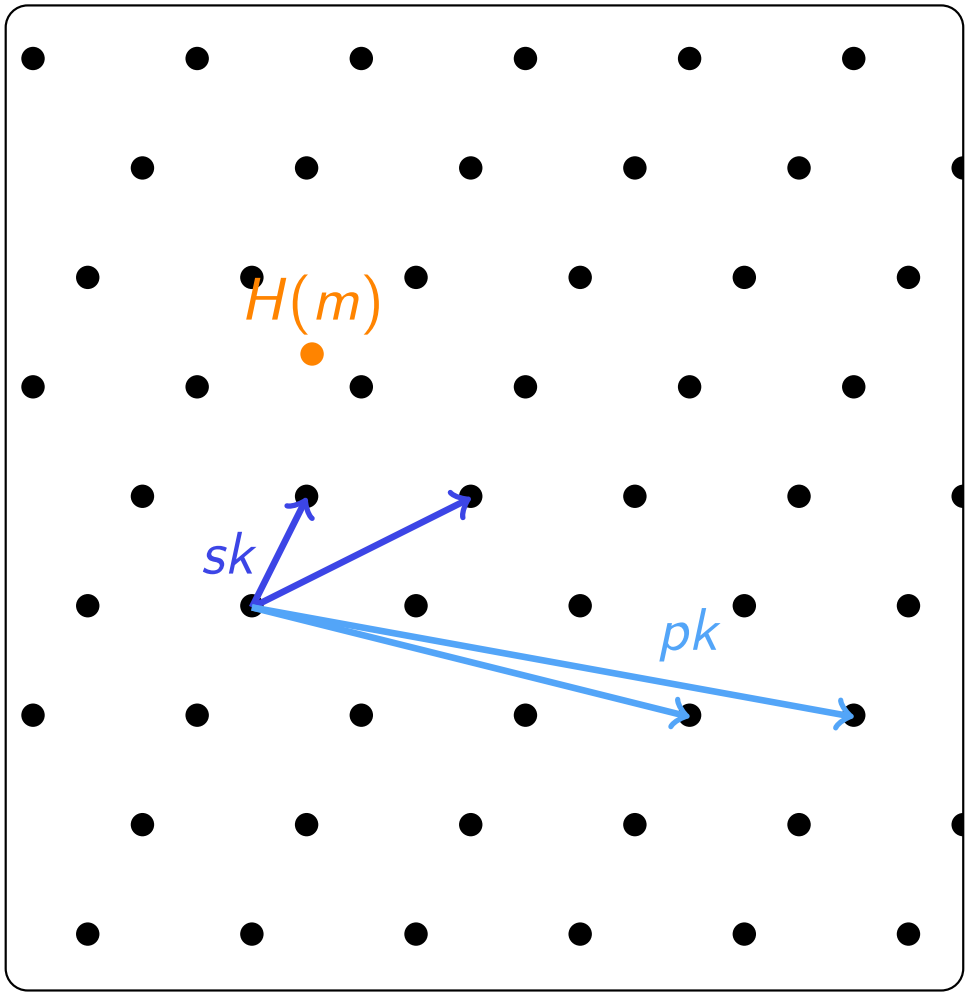


KeyGen:

- pk = bad basis of $\mathcal{L}$
  (e.g., HNF basis)
- sk = short basis of $\mathcal{L}$

[GGH97] Goldreich, Goldwasser, and Halevi. Public-key cryptosystems from lattice reduction problems. CRYPTO

Hash-and-sign: first idea [GGH97]



KeyGen:

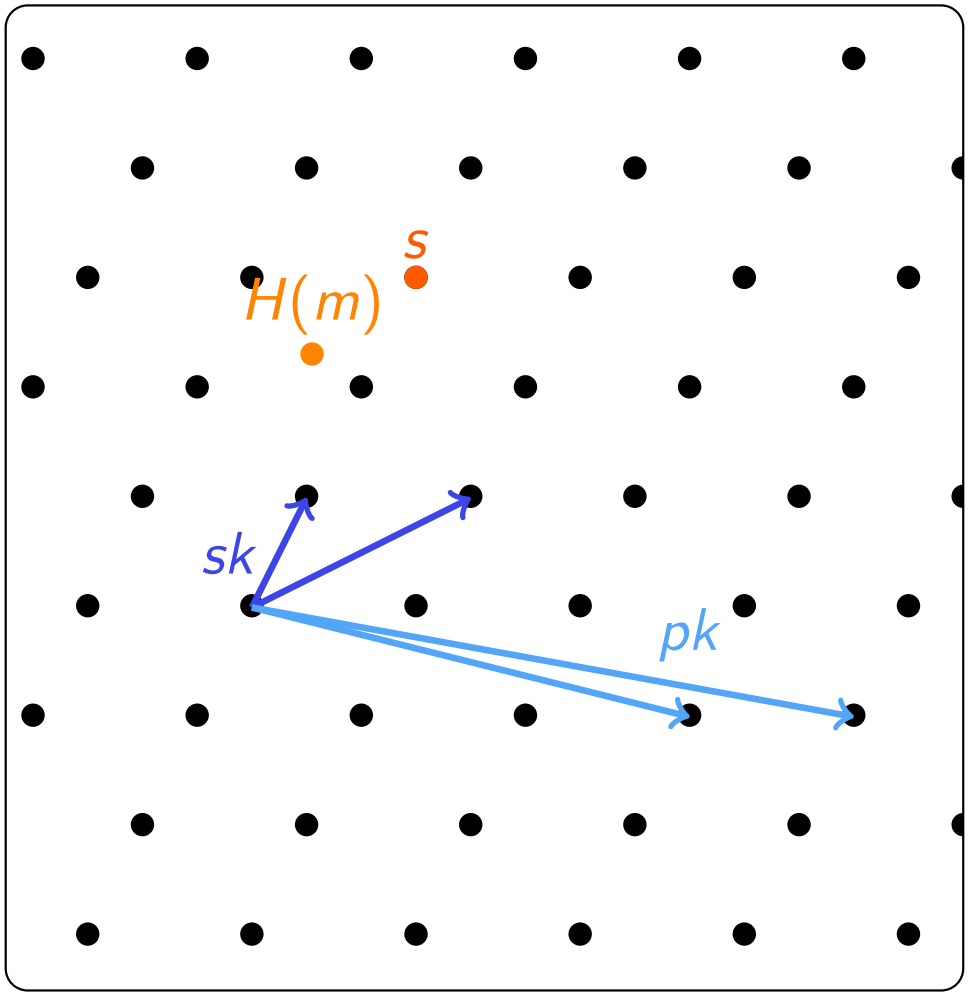
- pk = bad basis of $\mathcal{L}$
  (e.g., HNF basis)
- sk = short basis of $\mathcal{L}$

Sign(m, sk):

- $x = H(m)$ (hash the message)

[GGH97] Goldreich, Goldwasser, and Halevi. Public-key cryptosystems from lattice reduction problems. CRYPTO

Hash-and-sign: first idea [GGH97]



KeyGen:

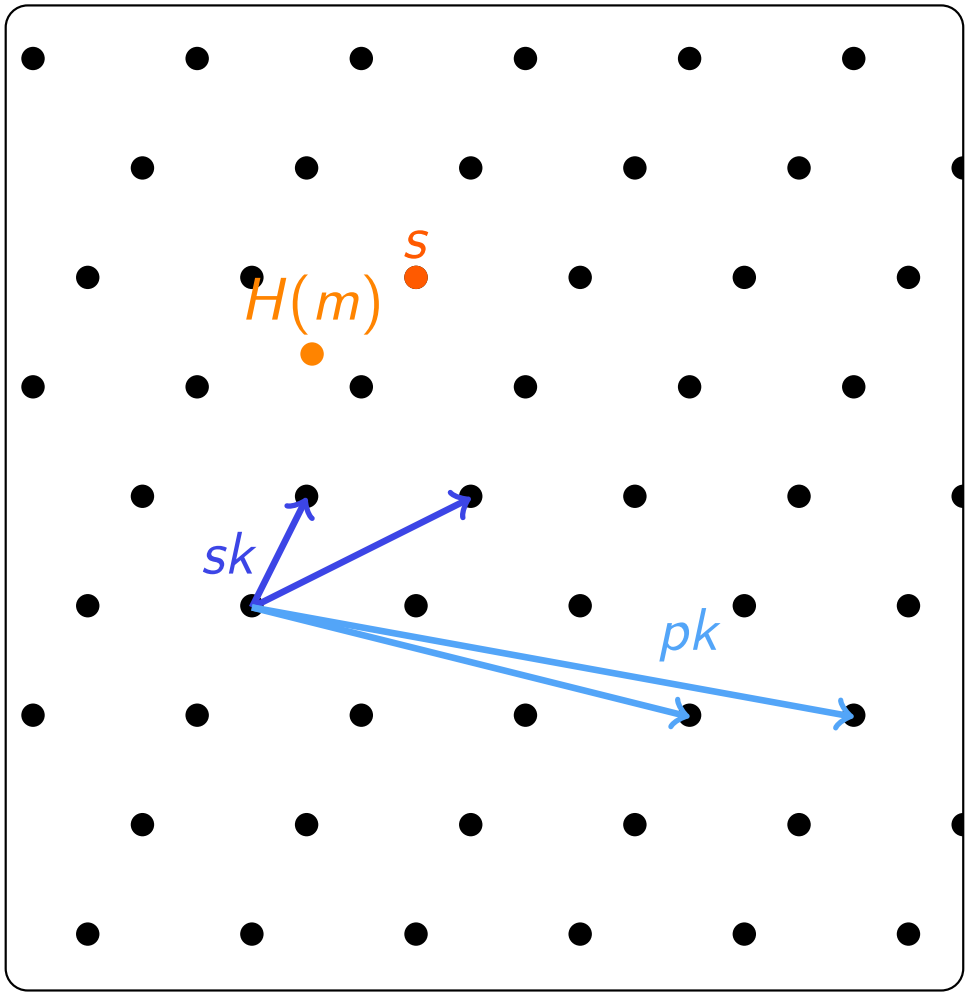
- pk = bad basis of $\mathcal{L}$
  (e.g., HNF basis)
- sk = short basis of $\mathcal{L}$

Sign(m, sk):

- $x = H(m)$ (hash the message)
- output $s \in \mathcal{L}$ close to $H(m)$

[GGH97] Goldreich, Goldwasser, and Halevi. Public-key cryptosystems from lattice reduction problems. CRYPTO

Hash-and-sign: first idea [GGH97]



KeyGen:

- pk = bad basis of $\mathcal{L}$
  (e.g., HNF basis)
- sk = short basis of $\mathcal{L}$

Sign(m, sk):

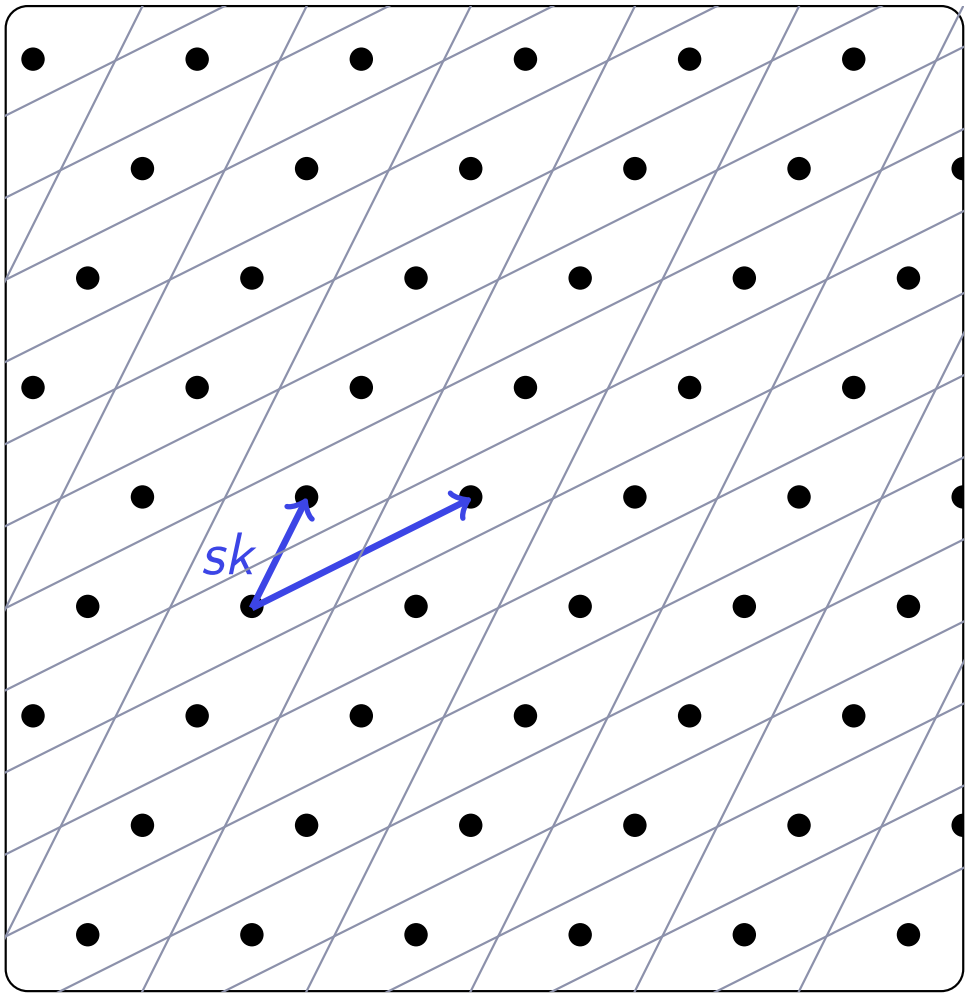
- $x = H(m)$ (hash the message)
- output $s \in \mathcal{L}$ close to $H(m)$

Verify(m, s, pk):

- check that $s \in \mathcal{L}$
- check that $H(m) - s$ is small

[GGH97] Goldreich, Goldwasser, and Halevi. Public-key cryptosystems from lattice reduction problems. CRYPTO

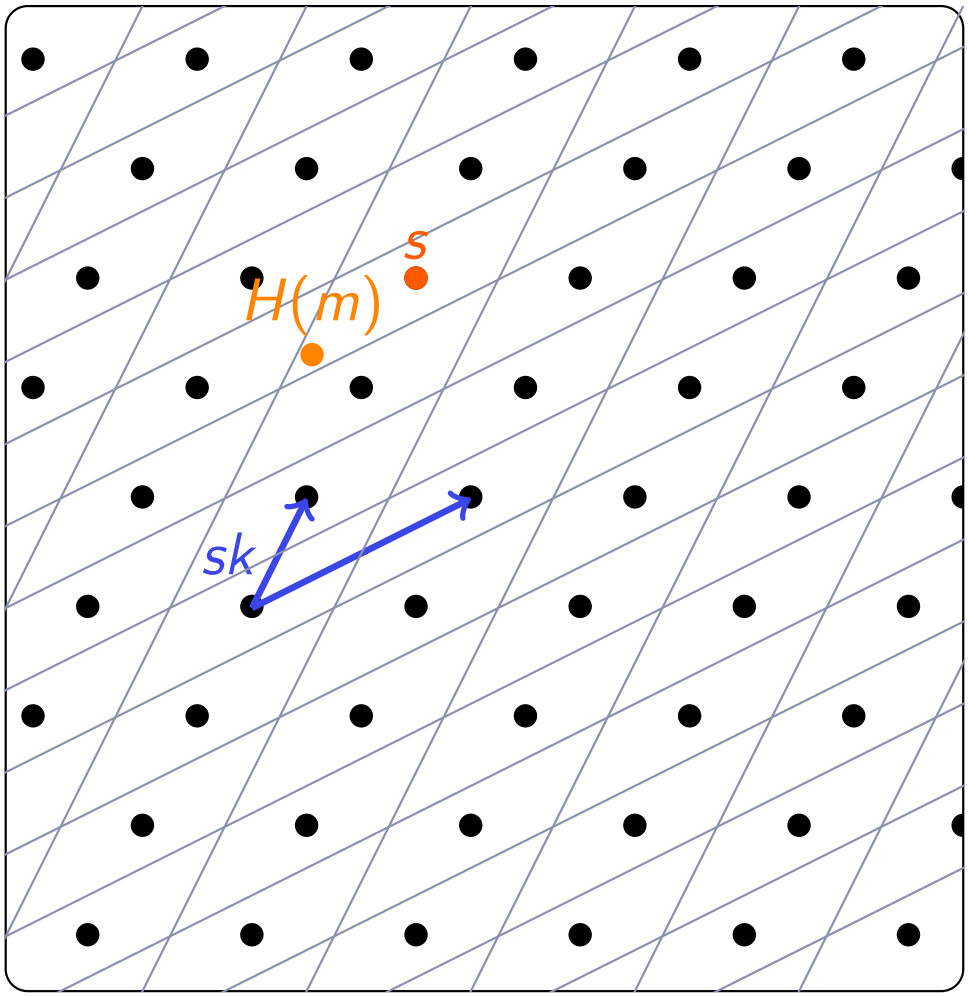
Attack on this first idea [NR06]



Parallelepiped attack:

[NR06] Nguyen and Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology

Attack on this first idea [NR06]

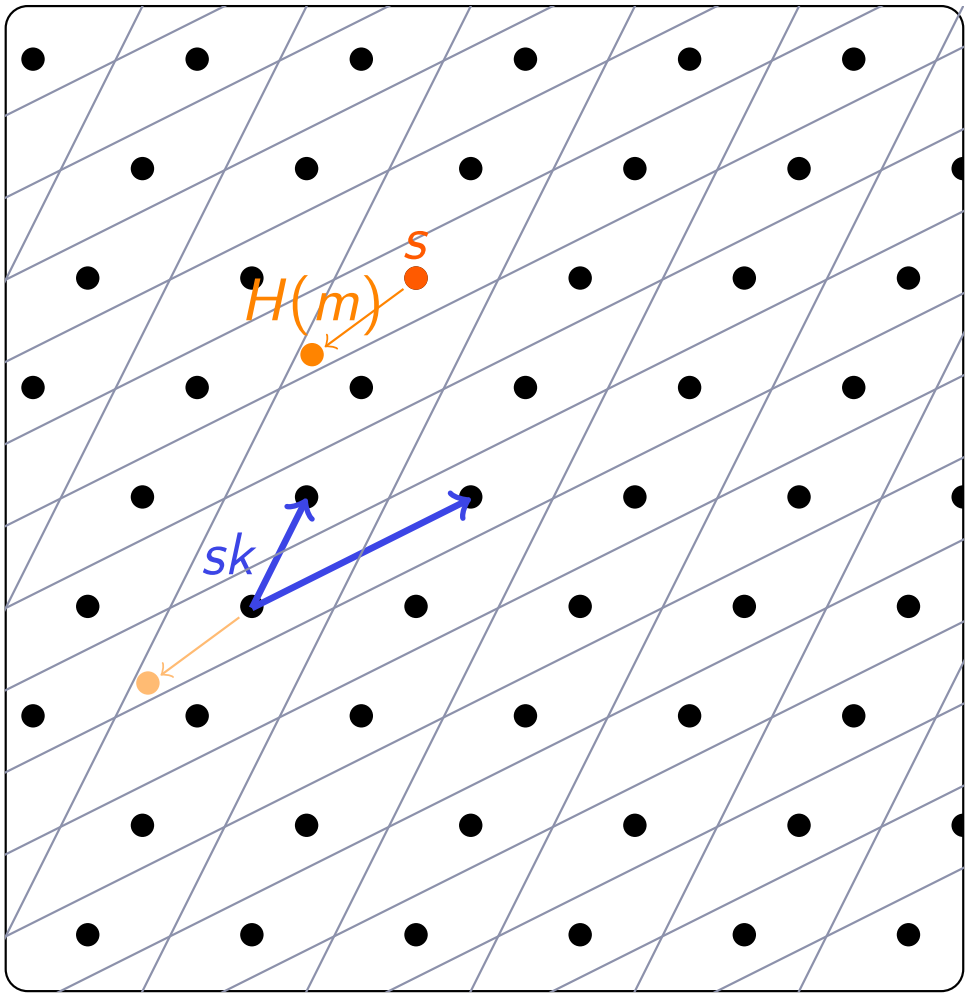


Parallelepiped attack:

- ask for a signature s on m

[NR06] Nguyen and Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology

Attack on this first idea [NR06]

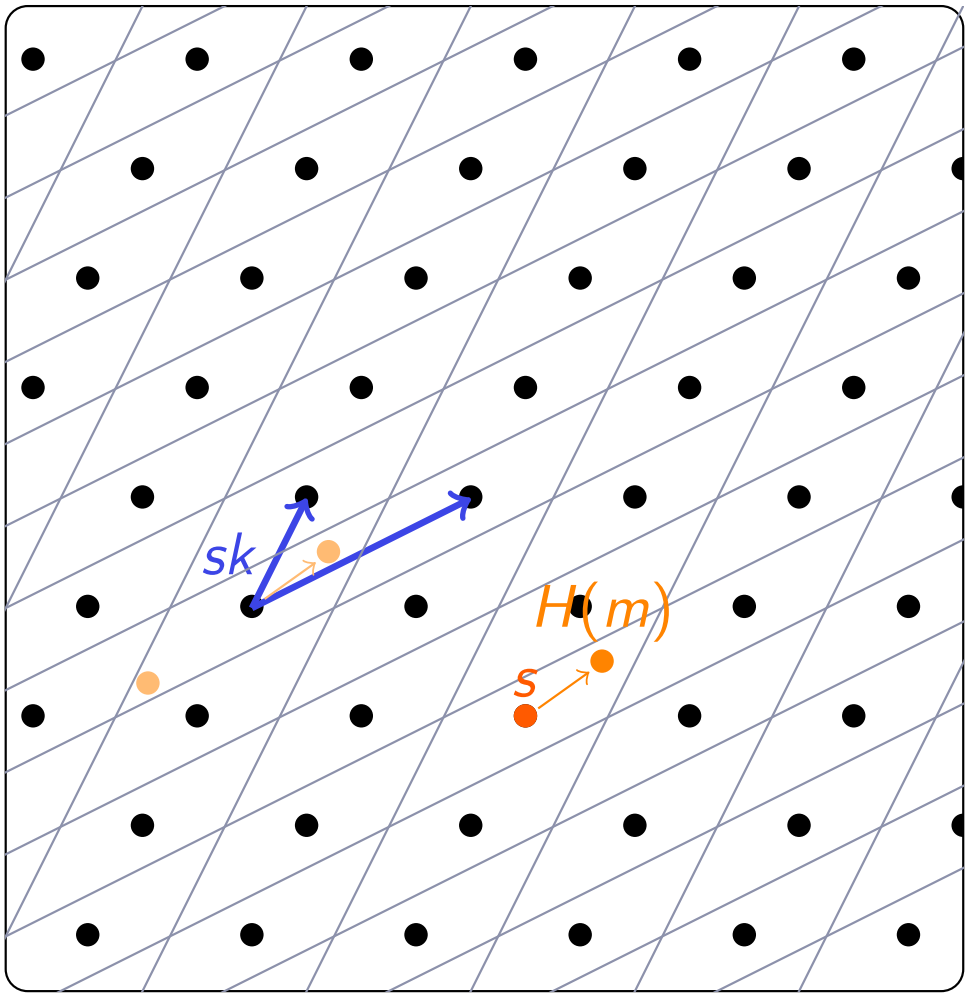


Parallelepiped attack:

- ask for a signature s on m
- plot $H(m) - s$

[NR06] Nguyen and Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology

Attack on this first idea [NR06]

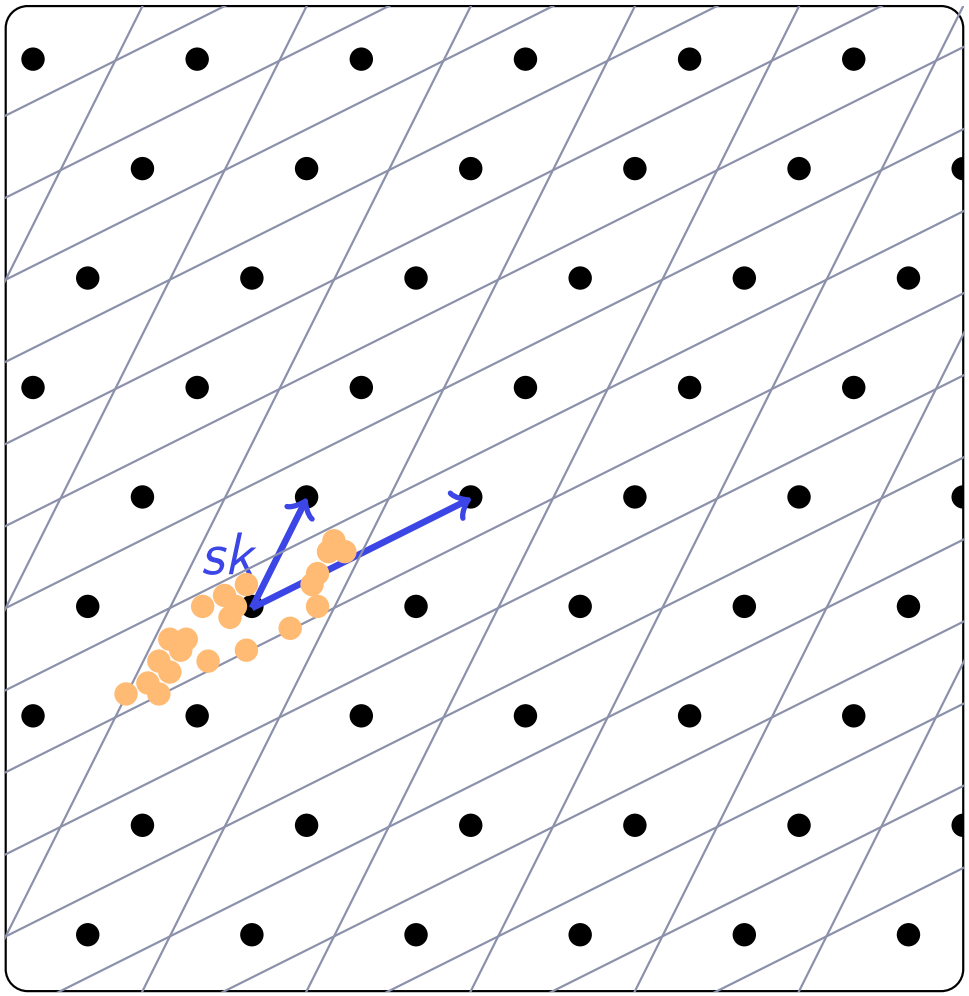


Parallelepiped attack:

- ask for a signature s on m
- plot $H(m) - s$
- repeat

[NR06] Nguyen and Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology

Attack on this first idea [NR06]

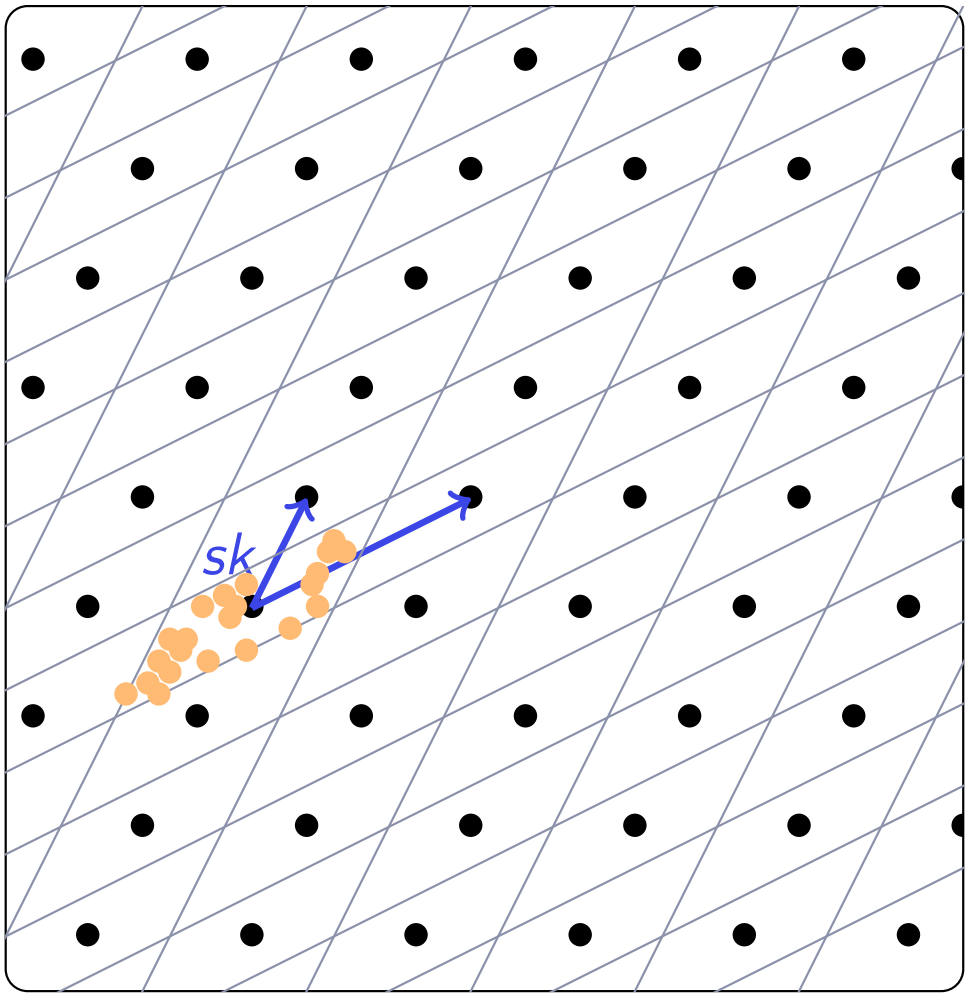


Parallelepiped attack:

- ask for a signature s on m
- plot $H(m) - s$
- repeat

[NR06] Nguyen and Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology

Attack on this first idea [NR06]



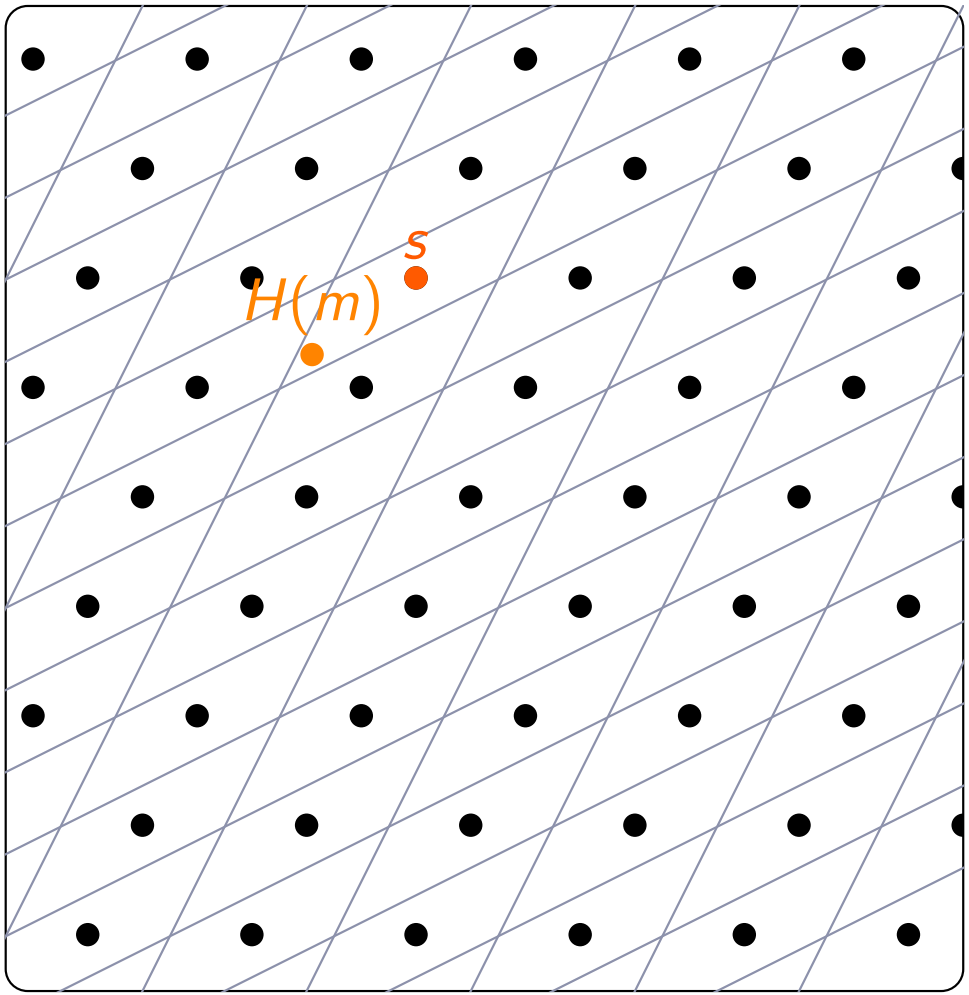
Parallelepiped attack:

- ask for a signature s on m
- plot $H(m) - s$
- repeat

From the shape of the parallelepiped, one can recover the short basis

[NR06] Nguyen and Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptology

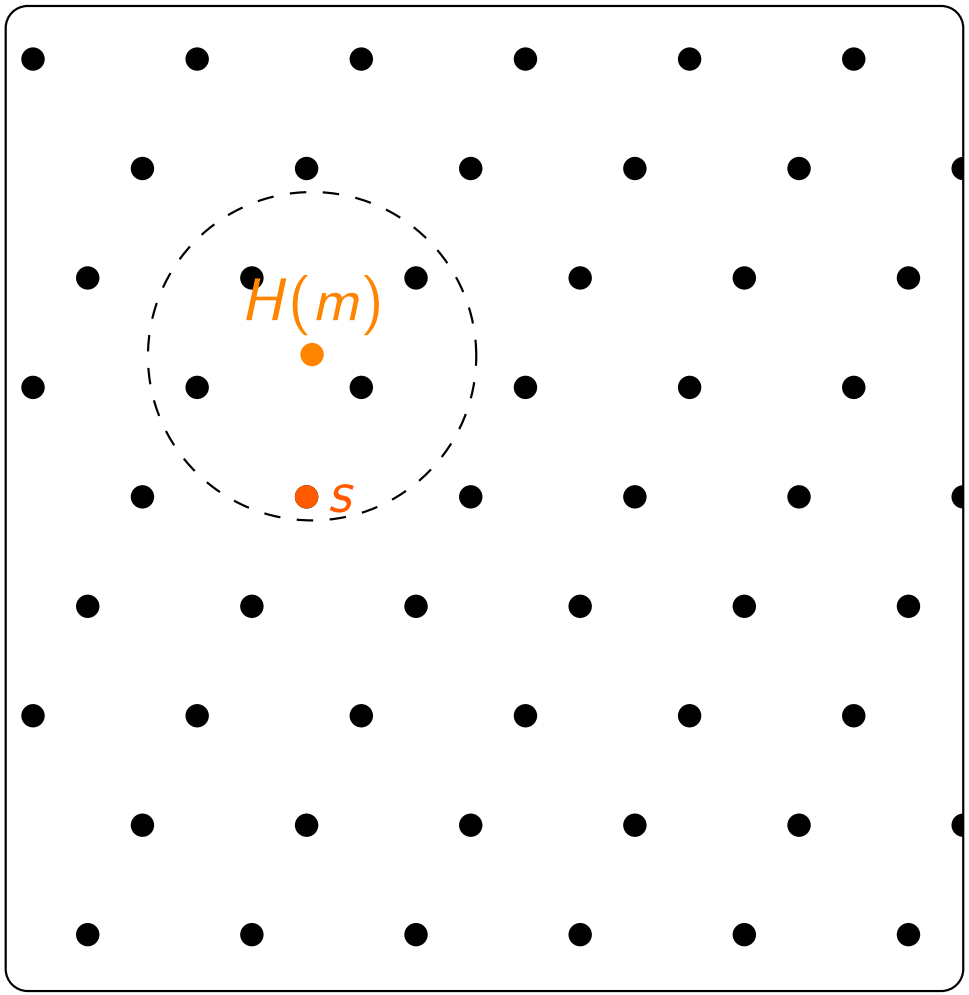
Preventing the attack [GPV08]



Idea: do not decode deterministically but randomly

[GPV08] Gentry, Peikert, and Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. STOC.

Preventing the attack [GPV08]



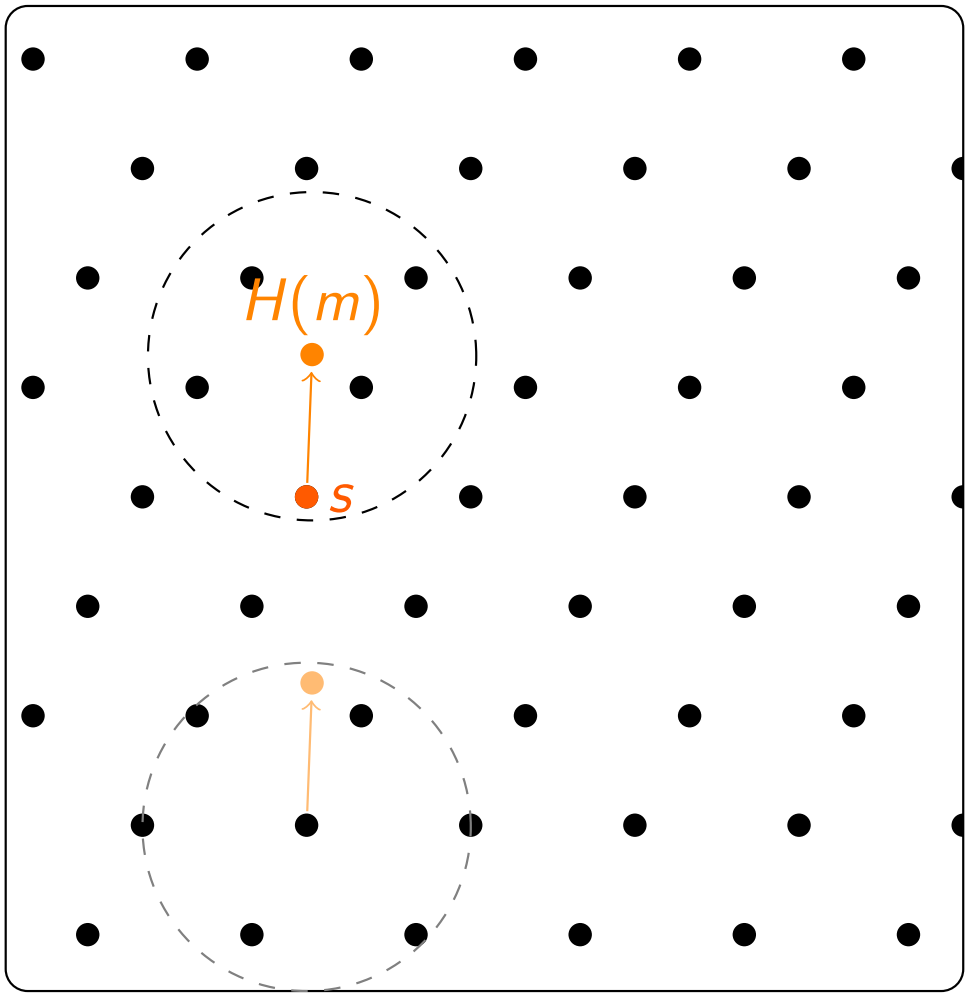
Idea: do not decode deterministically but randomly

Sign(m, sk):

- $x = H(m)$ (hash the message)
- sample $s \in \mathcal{L} \cap \mathcal{B}_r(x)$ (small radius r)

[GPV08] Gentry, Peikert, and Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. STOC.

Preventing the attack [GPV08]



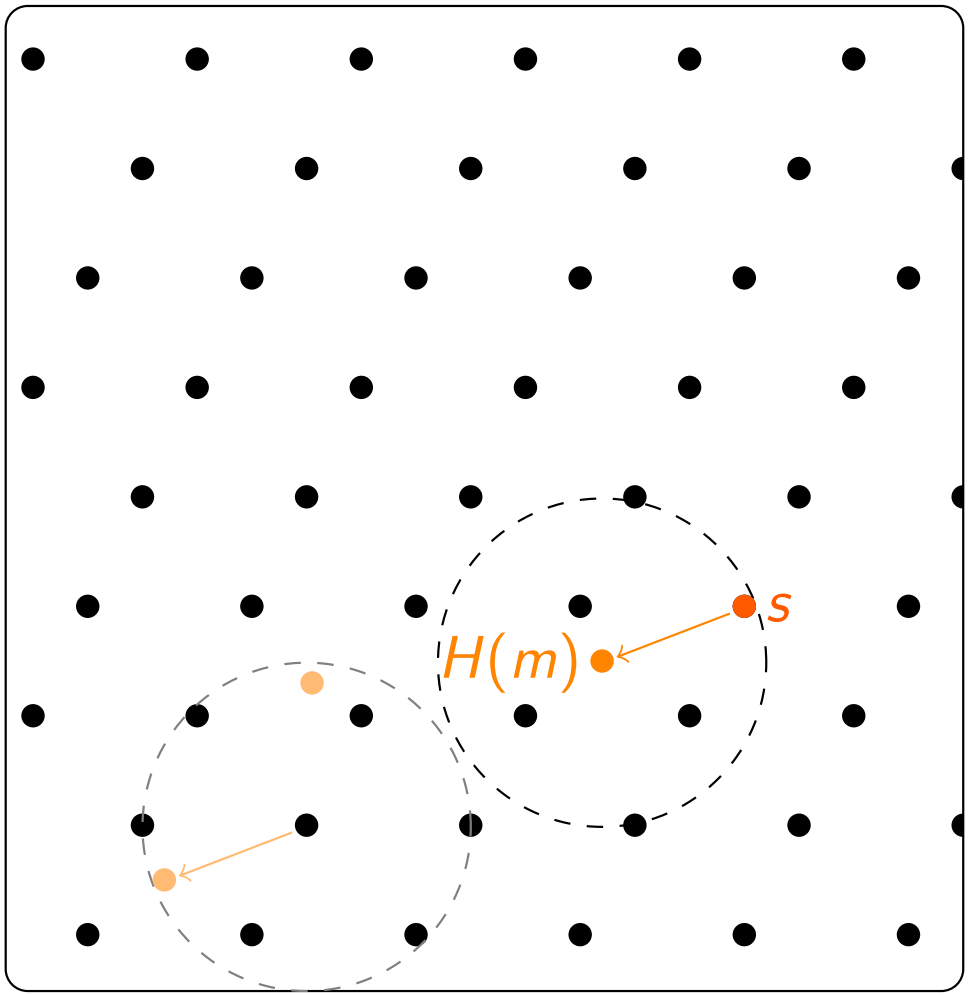
Idea: do not decode deterministically but randomly

Sign(m, sk):

- $x = H(m)$ (hash the message)
- sample $s \in \mathcal{L} \cap \mathcal{B}_r(x)$ (small radius r)

[GPV08] Gentry, Peikert, and Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. STOC.

Preventing the attack [GPV08]



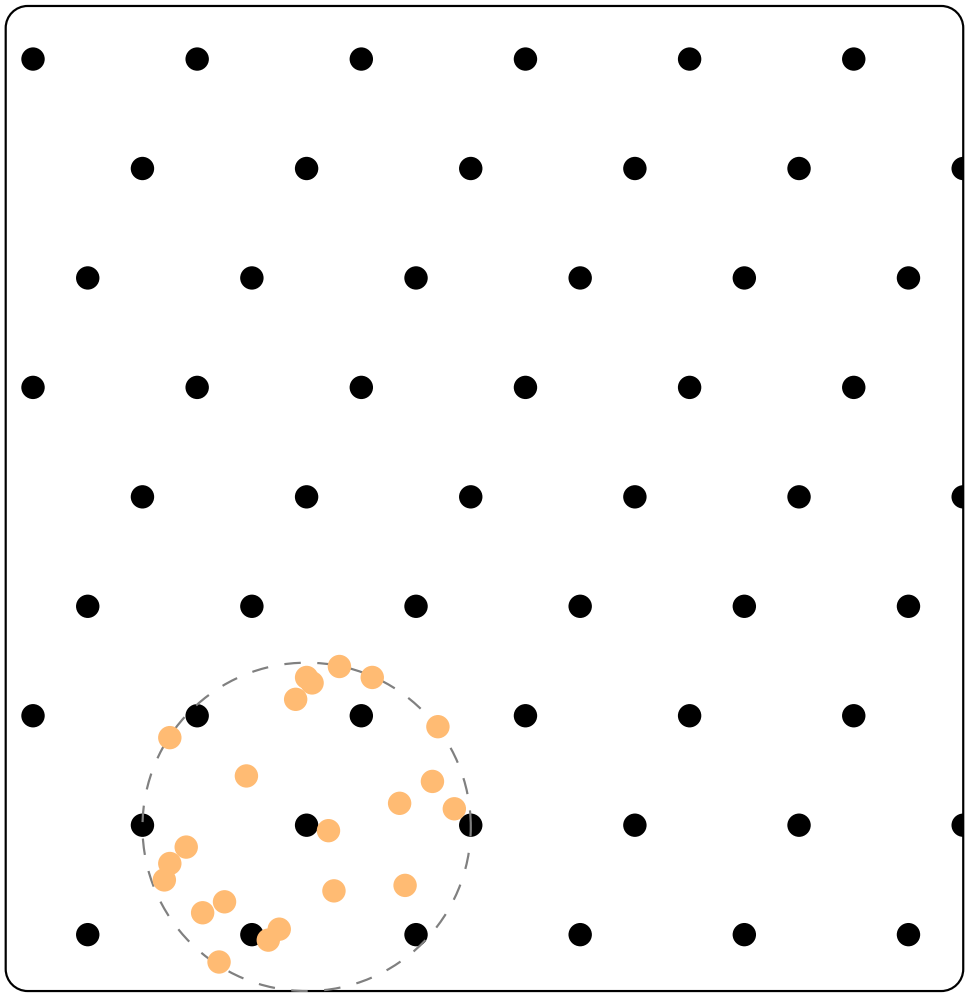
Idea: do not decode deterministically but randomly

Sign(m, sk):

- $x = H(m)$ (hash the message)
- sample $s \in \mathcal{L} \cap \mathcal{B}_r(x)$ (small radius r)

[GPV08] Gentry, Peikert, and Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. STOC.

Preventing the attack [GPV08]



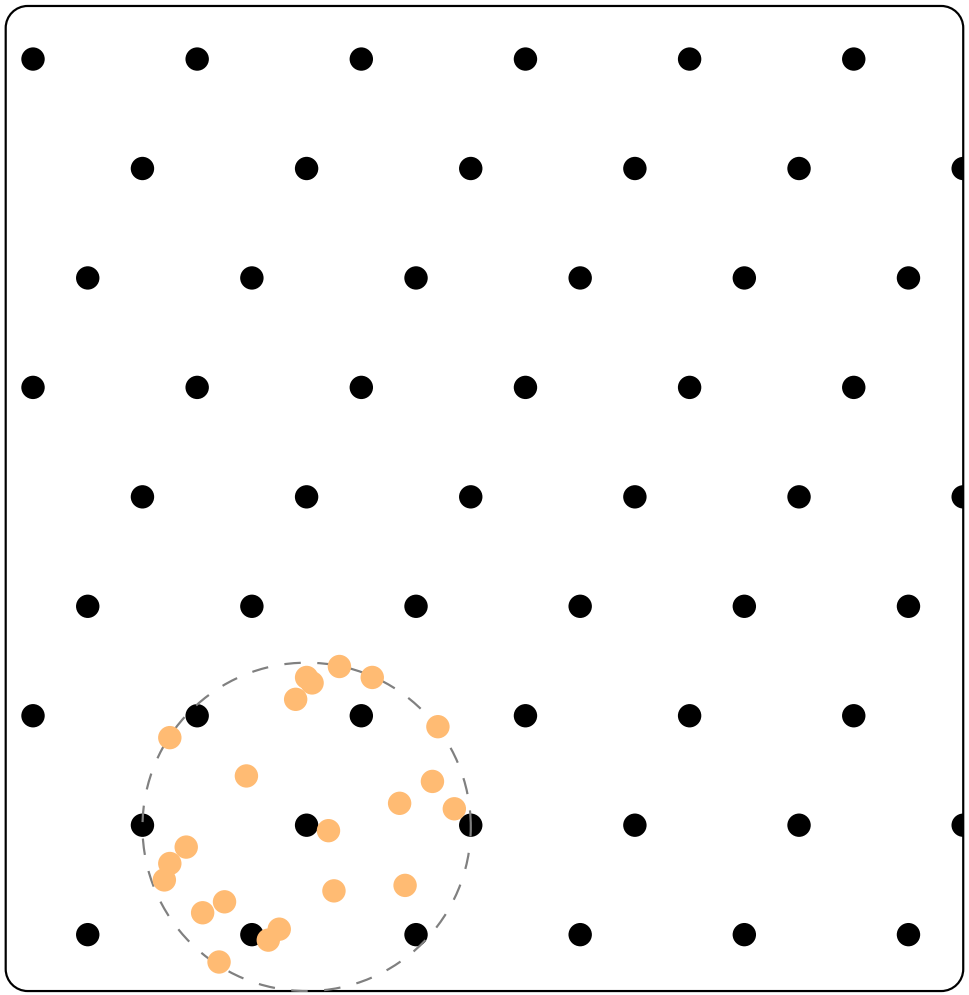
Idea: do not decode deterministically but randomly

Sign(m, sk):

- $x = H(m)$ (hash the message)
- sample $s \in \mathcal{L} \cap \mathcal{B}_r(x)$ (small radius r)

[GPV08] Gentry, Peikert, and Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. STOC.

Preventing the attack [GPV08]



Idea: do not decode deterministically but randomly

Sign(m, sk):

- $x = H(m)$ (hash the message)
- sample $s \in \mathcal{L} \cap \mathcal{B}_r(x)$ (small radius r)

Lemma: if an adversary can forge signatures, then she can recover a short basis of $\mathcal{L}$ using only pk (in the ROM)

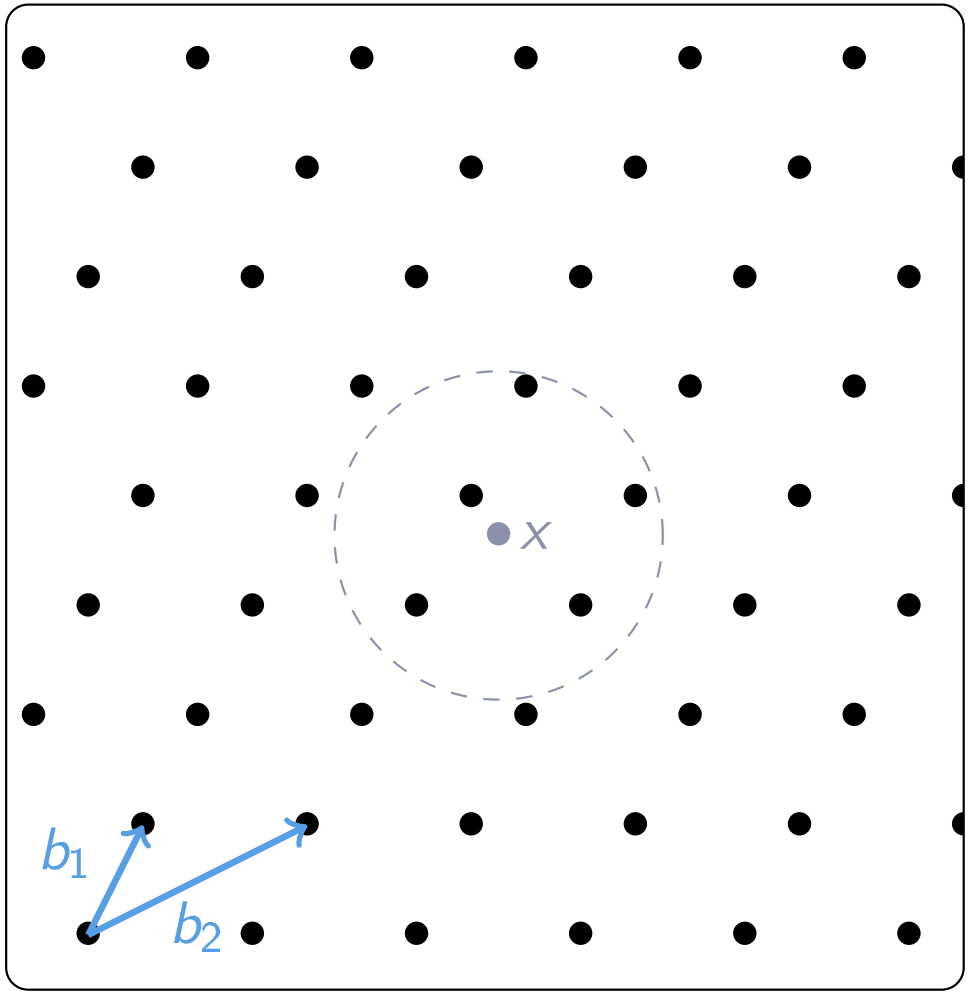
[GPV08] Gentry, Peikert, and Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. STOC.

Digression: Sampling uniformly in a ball [PP21]

Input: center x, radius r

(and a short basis $(b_1, \ldots, b_n)$)

Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

Digression: Sampling uniformly in a ball [PP21]

Input: center x, radius r

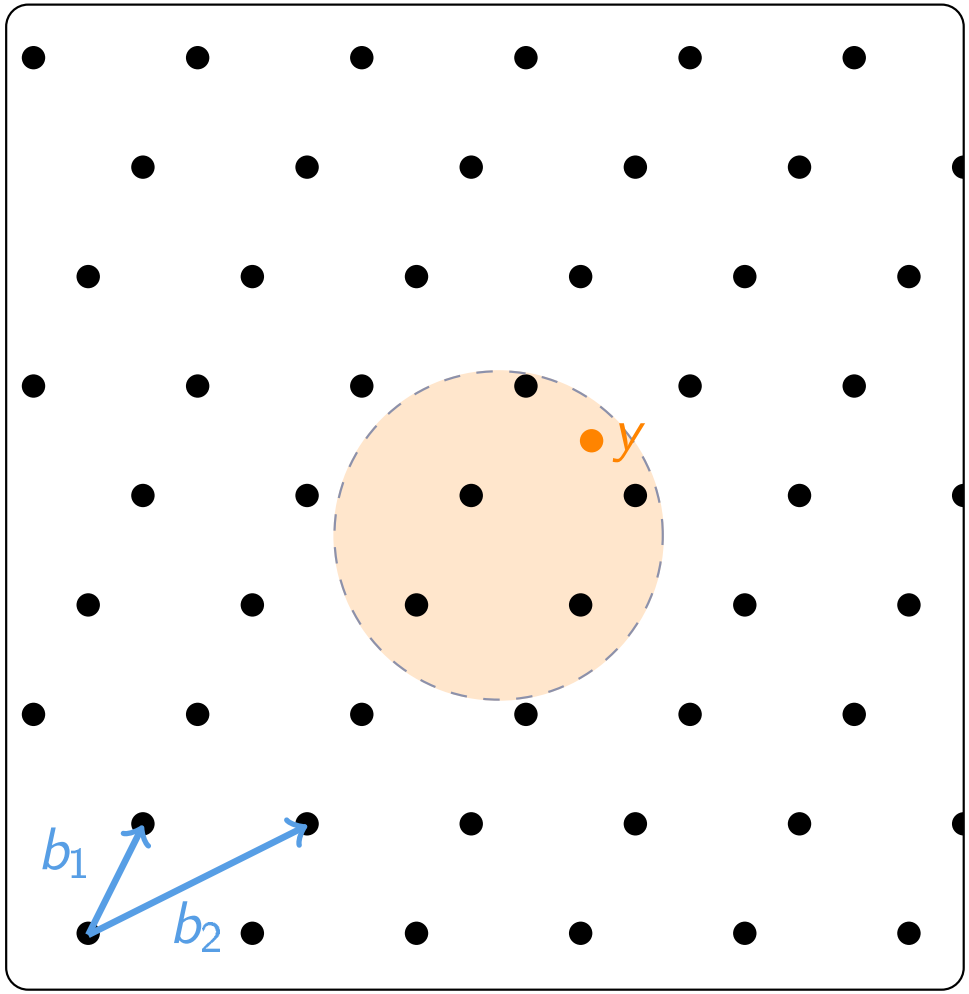
(and a short basis $(b_1, \ldots, b_n)$)

Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

Algo:

- Sample $y \leftarrow \mathcal{U}(\mathcal{B}_r(x))$

  (continuous distribution)



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

Digression: Sampling uniformly in a ball [PP21]

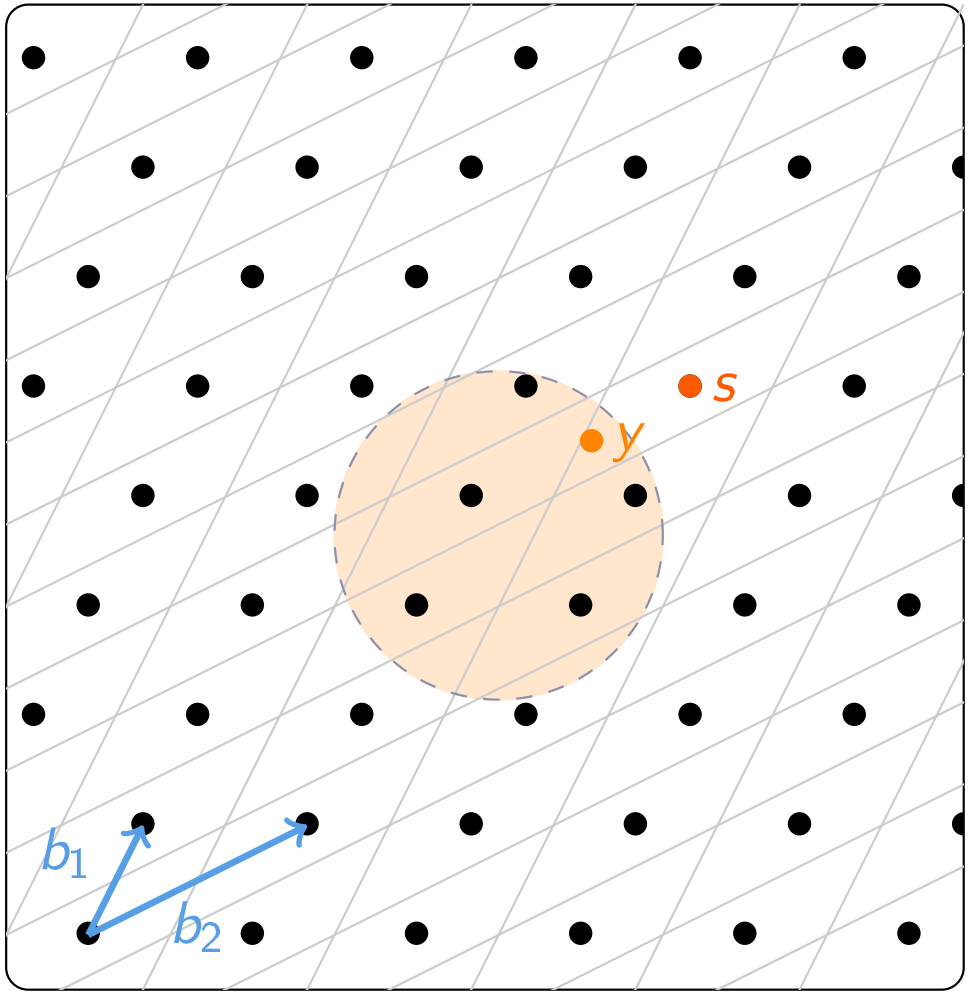
Input: center x, radius r

(and a short basis $(b_1, \ldots, b_n)$)

Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

Algo:

- Sample $y \leftarrow \mathcal{U}(\mathcal{B}_r(x))$
  (continuous distribution)
- $s \leftarrow \texttt{Babai\_decoding}(y)$



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

Digression: Sampling uniformly in a ball [PP21]

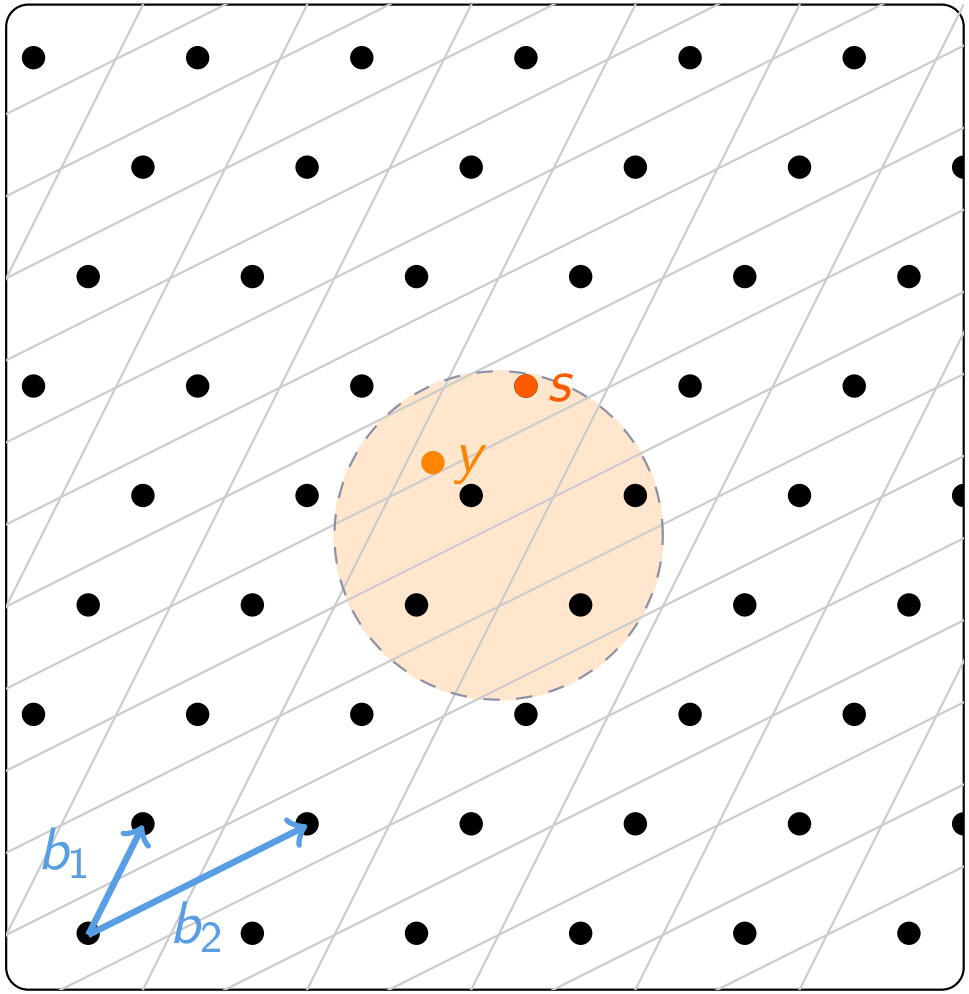
Input: center x, radius r

(and a short basis $(b_1, \ldots, b_n)$)

Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

Algo:

- Sample $y \leftarrow \mathcal{U}(\mathcal{B}_r(x))$
  (continuous distribution)
- $s \leftarrow \texttt{Babai\_decoding}(y)$
- repeat until $s \in \mathcal{B}_r(x)$



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

Digression: Sampling uniformly in a ball [PP21]

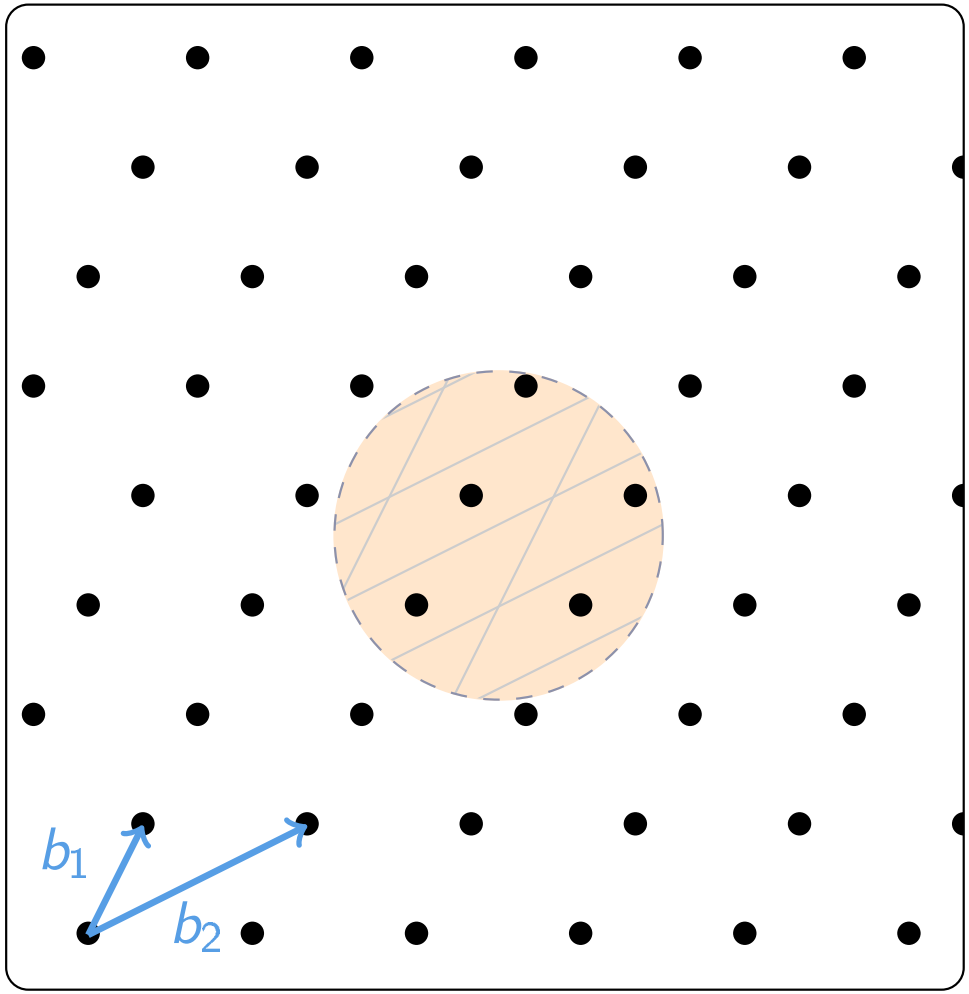
Input: center x, radius r

(and a short basis $(b_1, \ldots, b_n)$)

Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

Algo:

- Sample $y \leftarrow \mathcal{U}(\mathcal{B}_r(x))$
  (continuous distribution)
- $s \leftarrow \texttt{Babai\_decoding}(y)$
- repeat until $s \in \mathcal{B}_r(x)$



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

Digression: Sampling uniformly in a ball [PP21]

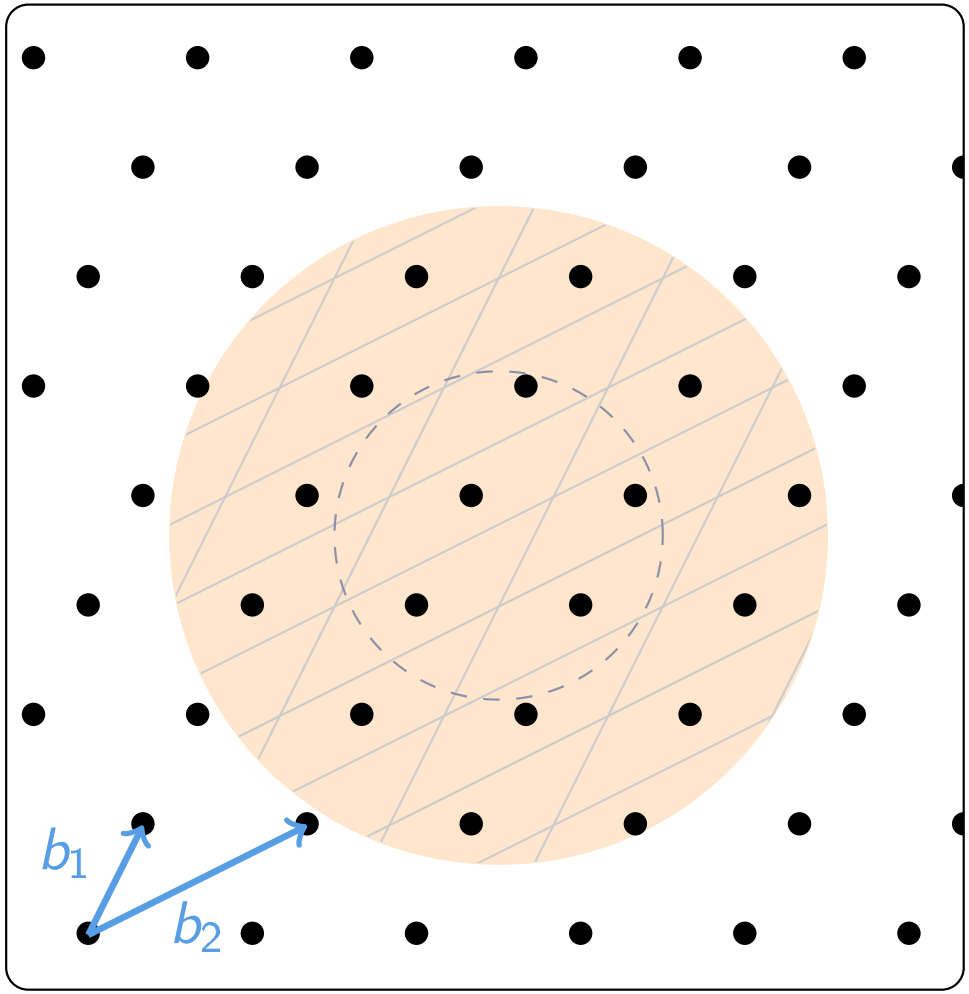
Input: center x, radius r

(and a short basis $(b_1, \ldots, b_n)$)

Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

Algo:

- Sample $y \leftarrow \mathcal{U}(\mathcal{B}_{r'}(x))$
  (continuous distribution)
- $s \leftarrow \texttt{Babai\_decoding}(y)$
- repeat until $s \in \mathcal{B}_r(x)$



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

Digression: Sampling uniformly in a ball [PP21]

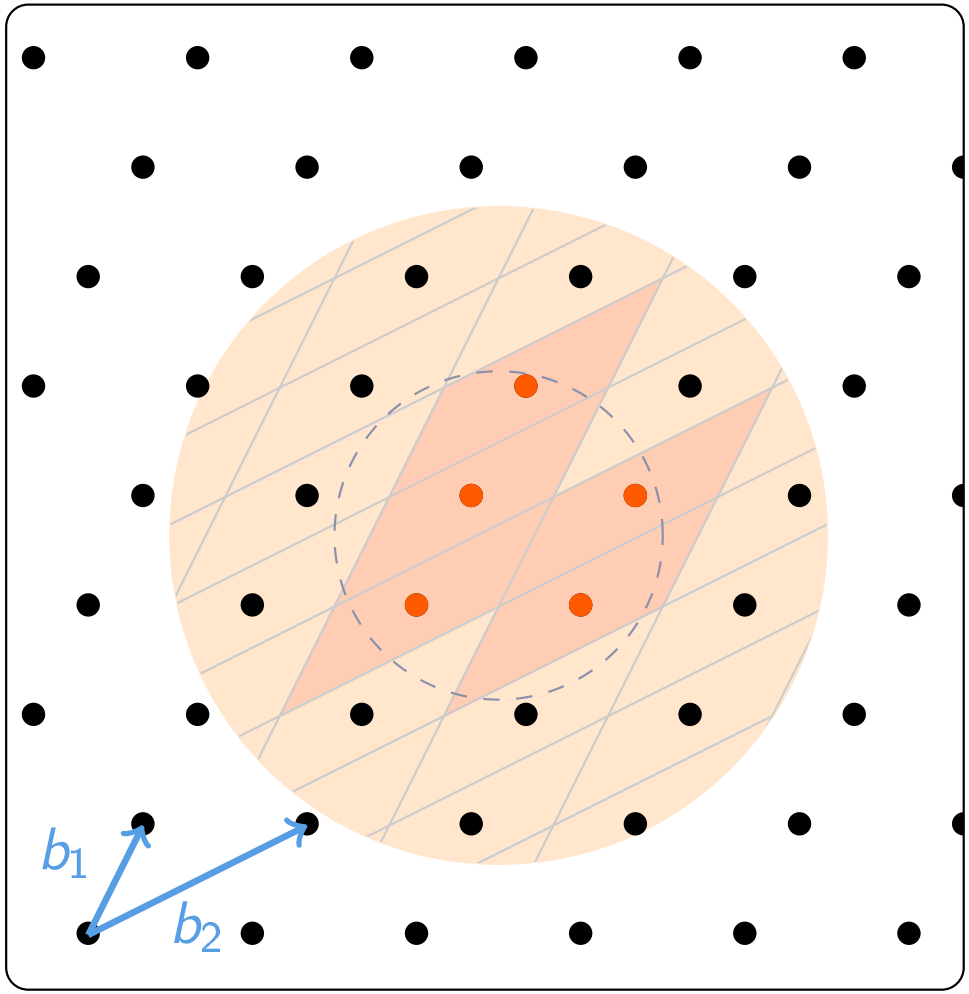
Input: center x, radius r

(and a short basis $(b_1, \ldots, b_n)$)

Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

Algo:

- Sample $y \leftarrow \mathcal{U}(\mathcal{B}_{r'}(x))$
  (continuous distribution)
- $s \leftarrow \texttt{Babai\_decoding}(y)$
- repeat until $s \in \mathcal{B}_r(x)$



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

Digression: Sampling uniformly in a ball [PP21]

Input: center x, radius r

(and a short basis $(b_1, \ldots, b_n)$)

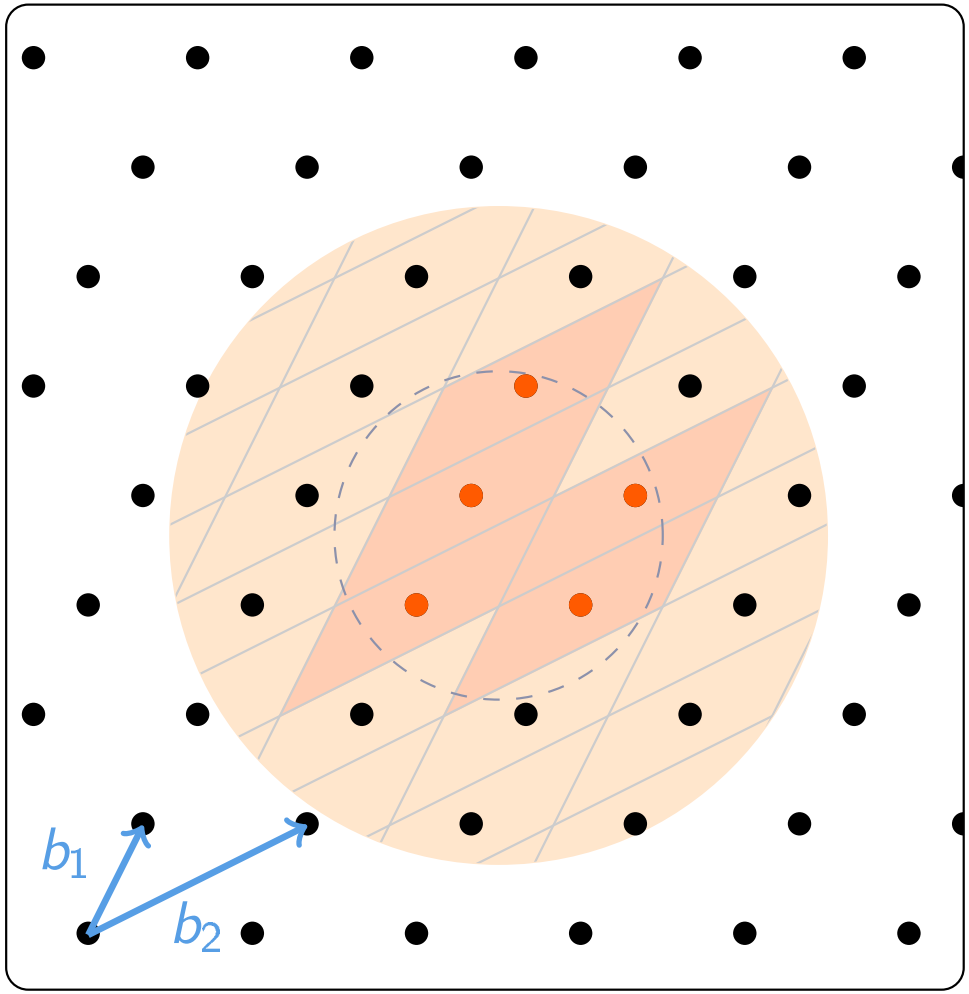
Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

Algo:

- Sample $y \leftarrow \mathcal{U}(\mathcal{B}_{r'}(x))$
  (continuous distribution)
- $s \leftarrow \texttt{Babai\_decoding}(y)$
- repeat until $s \in \mathcal{B}_r(x)$

polynomial time if

$r \geq 2n^2 \cdot \max_i \|b_i\|$



[PP21] Plançon and Prest. Exact Lattice Sampling from Non-Gaussian Distributions. PKC.

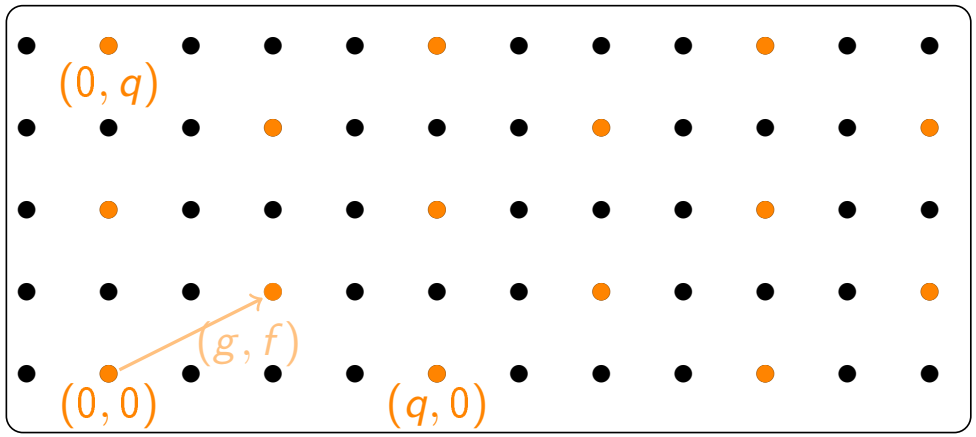
Section's conclusion

Hash-and-sign signature scheme:

- requires a lattice $\mathcal{L}$ + a short basis B_s + a bad basis B_p
- provably secure if recovering a short basis from B_p is hard

Falcon's signature scheme

(one of the three 2022 NIST standard)



Summary of previous episodes

Episode 1: no efficient algorithm that computes short bases for all lattices

Summary of previous episodes

Episode 1: no efficient algorithm that computes short bases for all lattices

- at least some lattices are hard
- but this does not mean that all lattices are hard

Summary of previous episodes

Episode 1: no efficient algorithm that computes short bases for all lattices

- at least some lattices are hard
- but this does not mean that all lattices are hard

Analogy with factoring: factoring is hard

- for some integers $\rightsquigarrow N = pq$ with p, q prime is hard
- but not for all integers $\rightsquigarrow N = p$ with p prime is easy

Summary of previous episodes

Episode 1: no efficient algorithm that computes short bases for all lattices

- at least some lattices are hard
- but this does not mean that all lattices are hard

Analogy with factoring: factoring is hard

- for some integers $\rightsquigarrow N = pq$ with p, q prime is hard
- but not for all integers $\rightsquigarrow N = p$ with p prime is easy

Sage demo

Summary of previous episodes

Episode 1: no efficient algorithm that computes short bases for all lattices

- at least some lattices are hard
- but this does not mean that all lattices are hard

Analogy with factoring: factoring is hard

- for some integers $\rightsquigarrow N = pq$ with p, q prime is hard
- but not for all integers $\rightsquigarrow N = p$ with p prime is easy

Sage demo

Episode 2: a random hard lattice $\mathcal{L}$ (+ a short basis) $\Rightarrow$ signature

Summary of previous episodes

Episode 1: no efficient algorithm that computes short bases for all lattices

- at least some lattices are hard
- but this does not mean that all lattices are hard

Analogy with factoring: factoring is hard

- for some integers $\rightsquigarrow N = pq$ with p, q prime is hard
- but not for all integers $\rightsquigarrow N = p$ with p prime is easy

Sage demo

Episode 2: a random hard lattice $\mathcal{L}$ (+ a short basis) $\Rightarrow$ signature

How do we generate a hard lattice (+ a short basis)?

NTRU [HPS98]

Parameters:
- q prime and large (e.g., $q = 16411$)
- $B \ll \sqrt{q}$ integer (e.g., $B = 5$)

[HPS98] Hoffstein, Pipher, and Silverman. NTRU: a ring based public key cryptosystem. ANTS.

NTRU [HPS98]

Parameters:
- q prime and large (e.g., $q = 16411$)
- $B \ll \sqrt{q}$ integer (e.g., $B = 5$)

NTRU instance:
- sample f, g random integers in $\{-B, \dots, B\}$
  (e.g., $f = -2$, $g = 3$)
- return $h = f \cdot g^{-1} \bmod q$
  ($h = (-2) \times 3^{-1} \bmod 16411 = -5471$)

[HPS98] Hoffstein, Pipher, and Silverman. NTRU: a ring based public key cryptosystem. ANTS.

NTRU [HPS98]

Parameters:
- q prime and large (e.g., $q = 16411$)
- $B \ll \sqrt{q}$ integer (e.g., $B = 5$)

NTRU instance:
- sample f, g random integers in $\{-B, \dots, B\}$
  (e.g., $f = -2$, $g = 3$)
- return $h = f \cdot g^{-1} \bmod q$
  ($h = (-2) \times 3^{-1} \bmod 16411 = -5471$)

NTRU assumption: there is no efficient algorithm that can distinguish h from uniform mod q

[HPS98] Hoffstein, Pipher, and Silverman. NTRU: a ring based public key cryptosystem. ANTS.

NTRU [HPS98]

Parameters:
- q prime and large (e.g., $q = 16411$)
- $B \ll \sqrt{q}$ integer (e.g., $B = 5$)

NTRU instance:
- sample f, g random integers in $\{-B, \dots, B\}$
  (e.g., $f = -2$, $g = 3$)
- return $h = f \cdot g^{-1} \bmod q$
  ($h = (-2) \times 3^{-1} \bmod 16411 = -5471$)

NTRU assumption: there is no efficient algorithm that can distinguish h from uniform mod q

Wait... NTRU assumption obviously does not hold
(we can test all the small (f, g))

[HPS98] Hoffstein, Pipher, and Silverman. NTRU: a ring based public key cryptosystem. ANTS.

NTRU [HPS98]

Parameters:
- q prime and large (e.g., $q = 16411$)
- $B \ll \sqrt{q}$ integer (e.g., $B = 5$)

NTRU instance:
- sample f, g random integers in $\{-B, \dots, B\}$ (e.g., $f = -2$, $g = 3$)
- return $h = f \cdot g^{-1} \bmod q$ ($h = (-2) \times 3^{-1} \bmod 16411 = -5471$)

NTRU assumption: there is no efficient algorithm that can distinguish h from uniform mod q

Wait... NTRU assumption obviously does not hold (we can test all the small (f, g))

- we should replace integers by polynomials (degree 512 or 1024)
- for this talk, let's keep integers (and pretend it is hard)

[HPS98] Hoffstein, Pipher, and Silverman. NTRU: a ring based public key cryptosystem. ANTS.

Hard lattice from NTRU assumption

NTRU assumption: an adversary cannot distinguish $h = f \cdot g^{-1} \bmod q$ (with $|f|, |g| \leq B$) from a uniform h in $\mathbb{Z}/q\mathbb{Z}$.

How do we get a hard lattice from this?

Hard lattice from NTRU assumption

NTRU assumption: an adversary cannot distinguish $h = f \cdot g^{-1} \bmod q$ (with $|f|, |g| \leq B$) from a uniform h in $\mathbb{Z}/q\mathbb{Z}$.

How do we get a hard lattice from this?

Let $h = f \cdot g^{-1}$ an NTRU instance.

$$\mathrm{B}_h = \begin{pmatrix} 1 & 0 \\ h & q \end{pmatrix} \quad \text{and} \quad \mathcal{L}_h = \mathcal{L}(\mathrm{B}_h) \text{ (spanned by the columns of } \mathrm{B}_h)$$

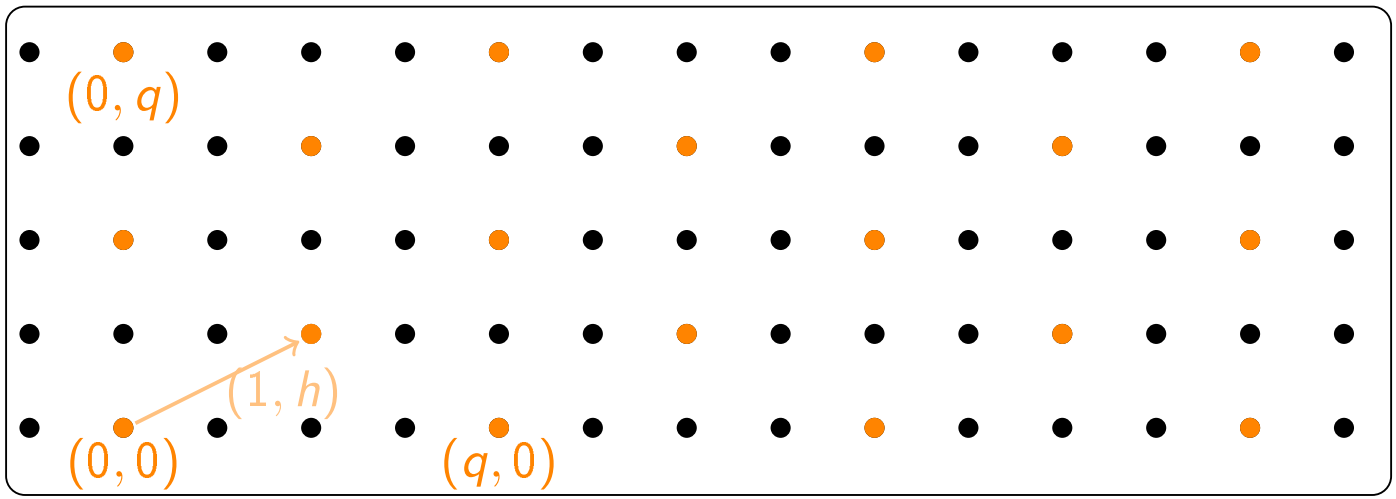
Hard lattice from NTRU assumption

NTRU assumption: an adversary cannot distinguish $h = f \cdot g^{-1} \bmod q$ (with $|f|, |g| \leq B$) from a uniform h in $\mathbb{Z}/q\mathbb{Z}$.

How do we get a hard lattice from this?

Let $h = f \cdot g^{-1}$ an NTRU instance.

$$\mathrm{B}_h = \begin{pmatrix} 1 & 0 \\ h & q \end{pmatrix} \quad \text{and} \quad \mathcal{L}_h = \mathcal{L}(\mathrm{B}_h) \text{ (spanned by the columns of } \mathrm{B}_h)$$



Hard lattice from NTRU assumption

NTRU assumption: an adversary cannot distinguish $h = f \cdot g^{-1} \bmod q$ (with $|f|, |g| \leq B$) from a uniform h in $\mathbb{Z}/q\mathbb{Z}$.

How do we get a hard lattice from this?

Let $h = f \cdot g^{-1}$ an NTRU instance.

$$\mathrm{B}_h = \begin{pmatrix} 1 & 0 \\ h & q \end{pmatrix} \quad \text{and} \quad \mathcal{L}_h = \mathcal{L}(\mathrm{B}_h) \quad \text{(spanned by the columns of } \mathrm{B}_h\text{)}$$

Property: $\begin{pmatrix} u \\ v \end{pmatrix} \in \mathcal{L}_h \iff h = v \cdot u^{-1} \bmod q$ (or $u = v = 0 \bmod q$)

Hard lattice from NTRU assumption

NTRU assumption: an adversary cannot distinguish $h = f \cdot g^{-1} \bmod q$ (with $|f|, |g| \leq B$) from a uniform h in $\mathbb{Z}/q\mathbb{Z}$.

How do we get a hard lattice from this?

Let $h = f \cdot g^{-1}$ an NTRU instance.

$$\mathrm{B}_h = \begin{pmatrix} 1 & 0 \\ h & q \end{pmatrix} \quad \text{and} \quad \mathcal{L}_h = \mathcal{L}(\mathrm{B}_h) \quad \text{(spanned by the columns of } \mathrm{B}_h\text{)}$$

Property: $\begin{pmatrix} u \\ v \end{pmatrix} \in \mathcal{L}_h \iff h = v \cdot u^{-1} \bmod q$ (or $u = v = 0 \bmod q$)

Finding a short vector in $\mathcal{L}_h$ $\implies$ distinguishing h from uniform

Hard lattice from NTRU assumption

NTRU assumption: an adversary cannot distinguish $h = f \cdot g^{-1} \bmod q$ (with $|f|, |g| \leq B$) from a uniform h in $\mathbb{Z}/q\mathbb{Z}$.

How do we get a hard lattice from this?

Let $h = f \cdot g^{-1}$ an NTRU instance.

$$\mathrm{B}_h = \begin{pmatrix} 1 & 0 \\ h & q \end{pmatrix} \quad \text{and} \quad \mathcal{L}_h = \mathcal{L}(\mathrm{B}_h) \quad \text{(spanned by the columns of } \mathrm{B}_h\text{)}$$

Property: $\begin{pmatrix} u \\ v \end{pmatrix} \in \mathcal{L}_h \iff h = v \cdot u^{-1} \bmod q$ (or $u = v = 0 \bmod q$)

Finding a short vector in $\mathcal{L}_h$ $\implies$ distinguishing h from uniform

no adversary can compute a short basis of $\mathcal{L}_h$ $\impliedby$ NTRU assumption holds

NTRU: wrapping up

We have seen: under the NTRU assumption, the following algorithm produces random hard lattices.

- sample random f, g in $\{-B, \dots, B\}$
- compute $h = f \cdot g^{-1} \bmod q$
- return $\mathcal{L}_h$, spanned by $\begin{pmatrix} 1 \\ h \end{pmatrix}$ and $\begin{pmatrix} 0 \\ q \end{pmatrix}$

NTRU: wrapping up

We have seen: under the NTRU assumption, the following algorithm produces random hard lattices.

- sample random f, g in $\{-B, \dots, B\}$
- compute $h = f \cdot g^{-1} \bmod q$
- return $\mathcal{L}_h$, spanned by $\begin{pmatrix} 1 \\ h \end{pmatrix}$ and $\begin{pmatrix} 0 \\ q \end{pmatrix}$

Wait... $\mathcal{L}_h$ has dimension 2, how can it be hard?

NTRU: wrapping up

We have seen: under the NTRU assumption, the following algorithm produces random hard lattices.

- sample random f, g in $\{-B, \dots, B\}$
- compute $h = f \cdot g^{-1} \bmod q$
- return $\mathcal{L}_h$, spanned by $\begin{pmatrix} 1 \\ h \end{pmatrix}$ and $\begin{pmatrix} 0 \\ q \end{pmatrix}$

Wait... $\mathcal{L}_h$ has dimension 2, how can it be hard?

This is because we work with integers instead of polynomials (for simplicity)

$\Rightarrow$ if f, g and h are polynomials of degree d, $\mathcal{L}_h$ has dimension $2d$ instead of 2

Falcon signature scheme

Falcon signature:

Hash-and-sign signature

+

hard lattice from NTRU assumption

with polynomials of degree $d = 512$ or 1024 instead of integers

Falcon is one of the three post-quantum signature scheme standardized by the NIST in 2022.

[Falcon] Fouque, Hoffstein, Kirchner, Lyubashevsky, Pornin, Prest, Ricosset, Seiler, Whyte, Zhang. NIST standard

Falcon's performances

			Post-quantum standards (NIST 2022)		
	RSA	EdDSA	**Falcon**	Dilithium	SPHINCS+
Public key size (bytes)	256	32	**897**	1312	32
Signature size (bytes)	256	64	**666**	2420	17088
Signature time (µs)	665	51	**241**	208	35584
Verification time (µs)	19	142	**52**	74	2091

- Comparison for $\approx$ 128-bits security
- Timings for a processor of 1.6 GHz
  - RSA and EdDSA are obtained experimentally with openssl on my laptop
  - Falcon, Dilithium and SPHINCS+ are obtained from the number of cycles provided on their websites (assuming a 1.6 GHz processor)
- for SPHINCS+, this is the variant using SHAKE hash function

Summing up and open questions

Falcon signature:

- post-quantum security
- fast (comparable to RSA/EdDSA)
- relatively compact ($\approx$ 3 times RSA, $\approx$ 15 times EdDSA)

Summing up and open questions

Falcon signature:

- post-quantum security
- fast (comparable to RSA/EdDSA)
- relatively compact ($\approx$ 3 times RSA, $\approx$ 15 times EdDSA)

But there are still important open questions for actual deployment

Summing up and open questions

Falcon signature:

- post-quantum security
- fast (comparable to RSA/EdDSA)
- relatively compact ($\approx$ 3 times RSA, $\approx$ 15 times EdDSA)

But there are still important open questions for actual deployment

- requires floating point
  - not all devices have floating point units

Summing up and open questions

Falcon signature:

- post-quantum security
- fast (comparable to RSA/EdDSA)
- relatively compact ($\approx$ 3 times RSA, $\approx$ 15 times EdDSA)

But there are still important open questions for actual deployment

- requires floating point
  - not all devices have floating point units
- difficult to mask
  - can be subject to certain side-channel attacks
  - see Raccoon if you want a masking-friendly lattice-based signature

---

[Raccoon] del Pino, Espitau, Katsumata, Maller, Mouhartem, Prest, Rossi, Saarinen. Technical report, NIST.

Summing up and open questions

Falcon signature:

- post-quantum security
- fast (comparable to RSA/EdDSA)
- relatively compact ($\approx$ 3 times RSA, $\approx$ 15 times EdDSA)

But there are still important open questions for actual deployment

- requires floating point
  - not all devices have floating point units
- difficult to mask
  - can be subject to certain side-channel attacks
  - see Raccoon if you want a masking-friendly lattice-based signature
- KeyGen and Sign are complex to implement
  - can lead to bugs in implementation

[Raccoon] del Pino, Espitau, Katsumata, Maller, Mouhartem, Prest, Rossi, Saarinen. Technical report, NIST.

Thank you