Introduction to lattice-based cryptography

Part I

Alice Pellet-Mary

PEPR PQ-TLS summer school, Anglet

Plan

Part I

Lattice problems

- Lattices
- Hard problems

Algorithms for lattice problems

- LLL, sieving, BKZ

Constructions

- Public key encryption
- Signatures

Part II

NTRU and signature

- NTRU, Falcon

LWE and public key encryption

- LWE
- Regev's encryption, Kyber

Algebraic Lattices

- Ideal and module lattices
- NTRU, RLWE, mod-LWE

Reductions between problems

https://wesselvanwoerden.com/

Plan

Part I

Lattice problems

- Lattices
- Hard problems

Algorithms for lattice problems

- LLL, sieving, BKZ

Constructions

- Public key encryption
- Signatures

Part II

NTRU and signature

- NTRU, Falcon

LWE and public key encryption

- LWE
- Regev's encryption, Kyber

Algebraic Lattices

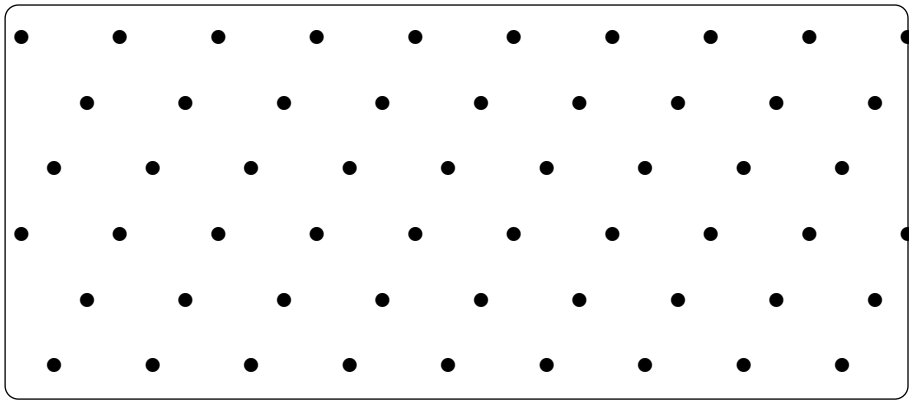
- Ideal and module lattices
- NTRU, RLWE, mod-LWE

Reductions between problems

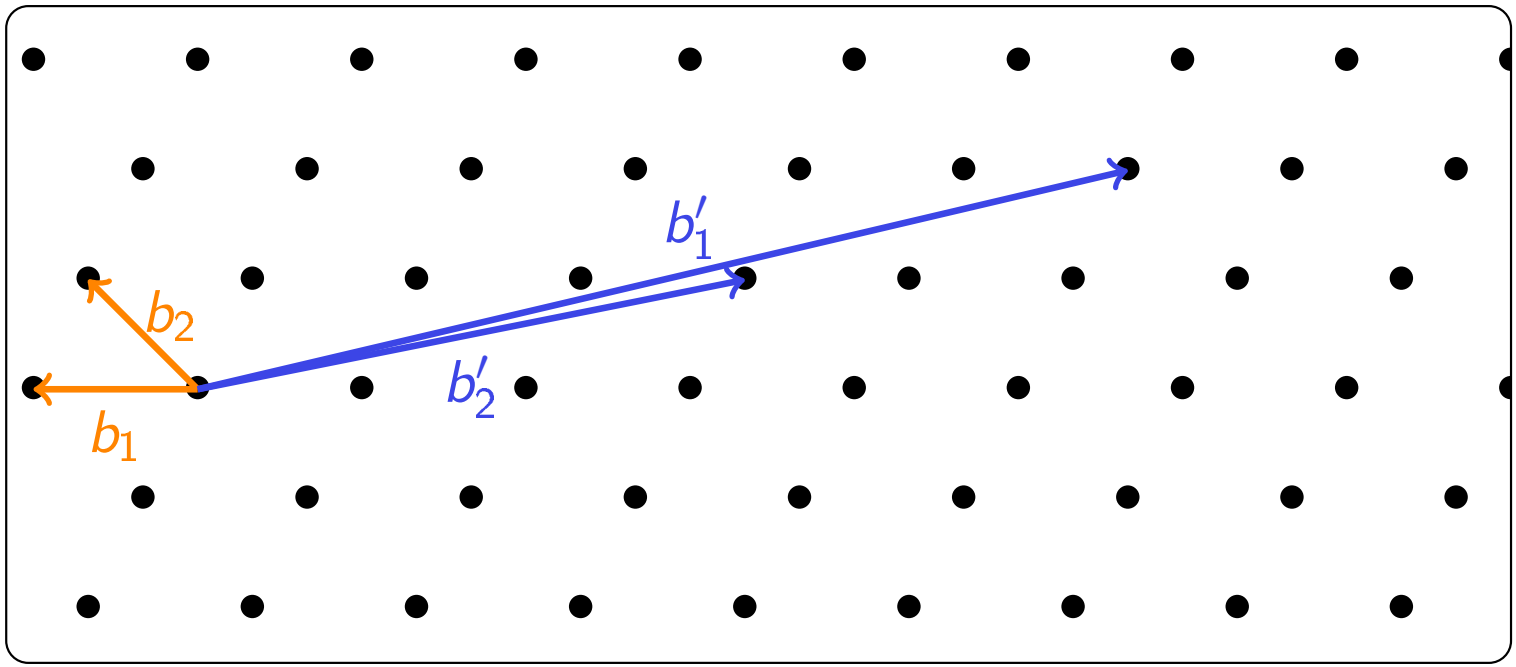
acknowledgements: some slides are from Wessel van Woerden!

https://wesselvanwoerden.com/

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 non-zero vector of a lattice

Algorithmic problems on lattices

Example of problems:

(1) Testing equality of lattices

(2) Intersecting two lattices

(3) Computing a short non-zero vector 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

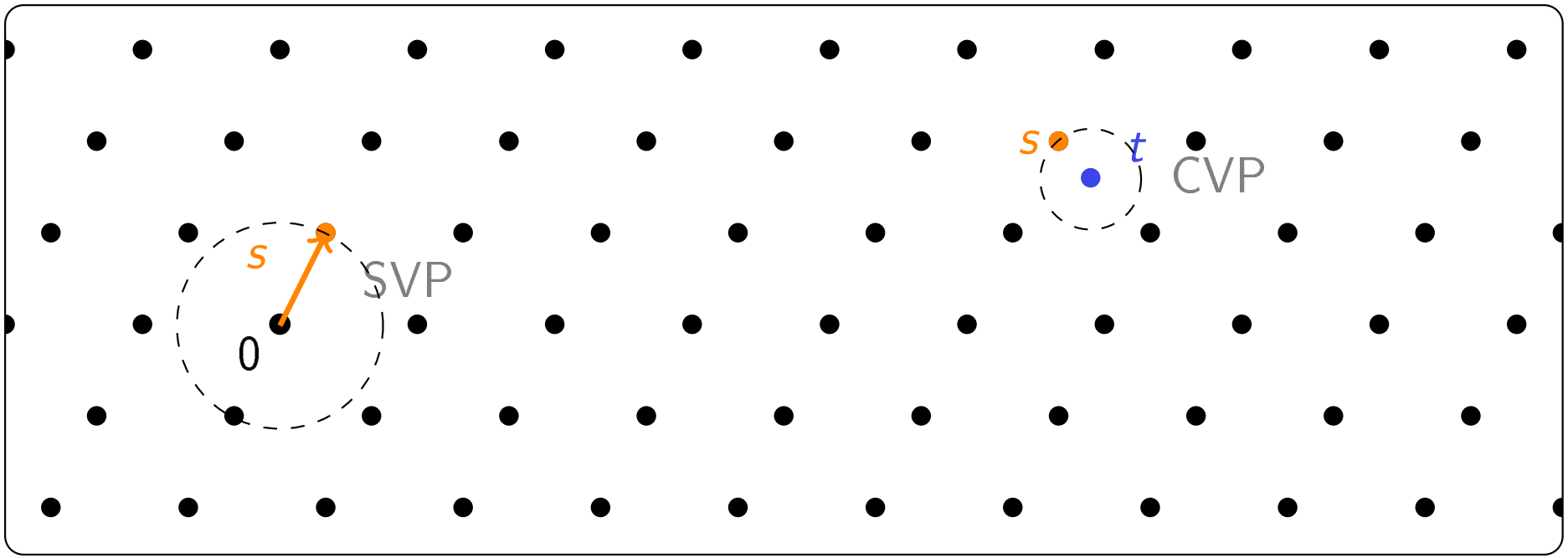
(2) Intersecting two lattices $\Rightarrow$ easy

(3) Computing a short non-zero vector of a lattice $\Rightarrow$ hard

Quiz: which ones are easy or hard?

easy: polynomial time hard: no polynomial time algorithm known

Shortest and closest vector problems



SVP: Shortest Vector Problem

$$\|s\| = \lambda_1(\mathcal{L})$$

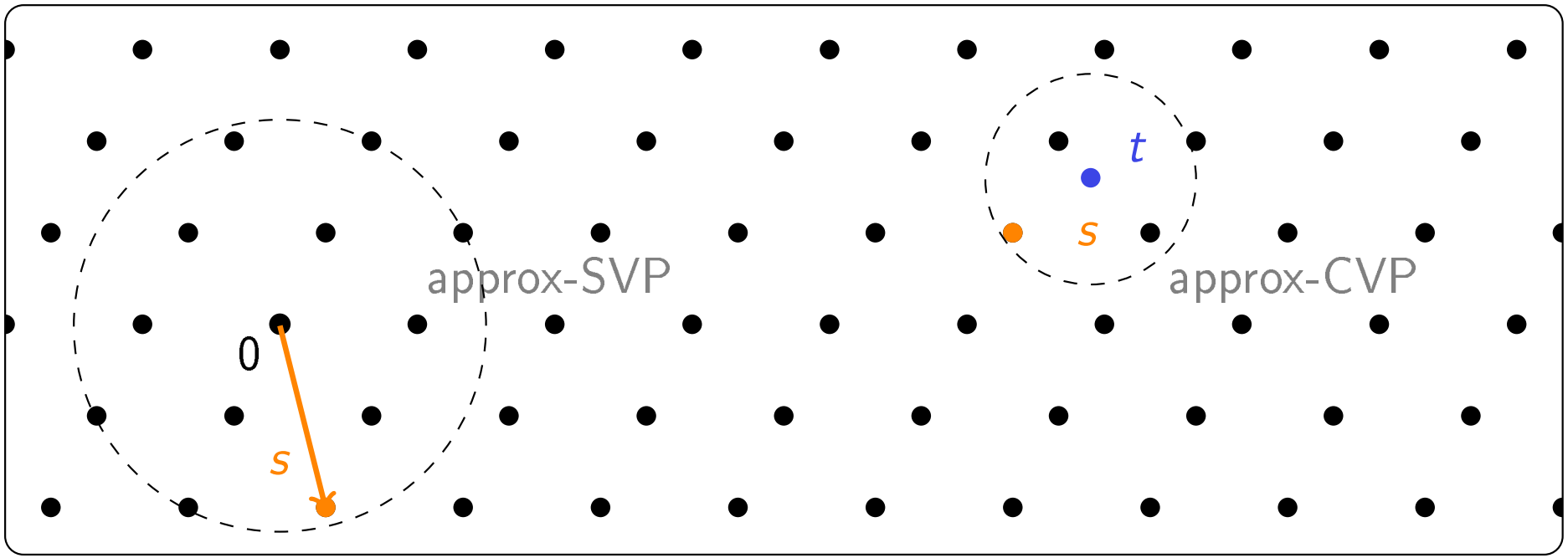
$$\lambda_1(\mathcal{L}) = \min_{\substack{v \in \mathcal{L} \\ v \neq 0}} \|v\|$$

CVP: Closest Vector Problem

$$\|t - s\| = \mathrm{dist}(t, \mathcal{L})$$

$$\mathrm{dist}(t, \mathcal{L}) = \min_{v \in \mathcal{L}} \|t - v\|$$

Shortest and closest vector problems



approx-SVP: Shortest Vector Problem

$$\|s\| \leq \gamma \cdot \lambda_1(\mathcal{L})$$

$$\lambda_1(\mathcal{L}) = \min_{\substack{v \in \mathcal{L} \\ v \neq 0}} \|v\|$$

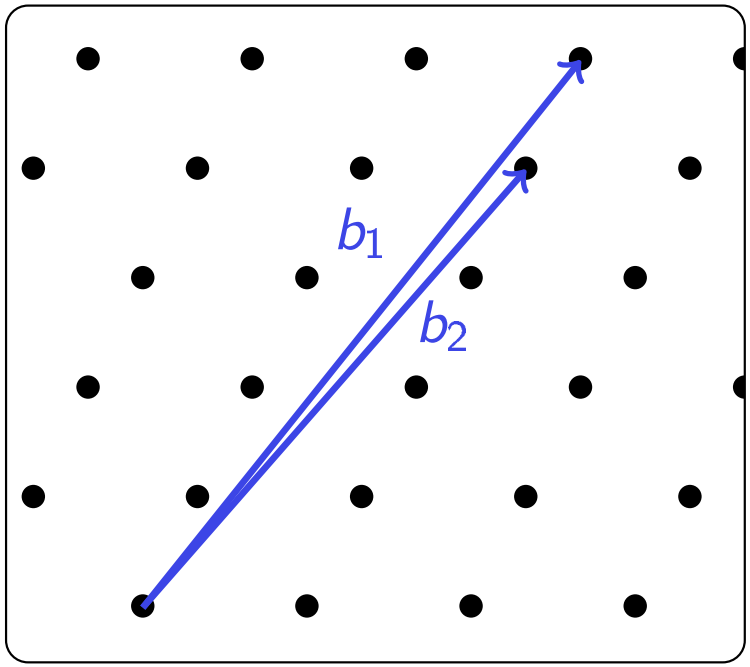
approx-CVP: Closest Vector Problem

$$\|t - s\| \leq \gamma \cdot \mathrm{dist}(t, \mathcal{L})$$

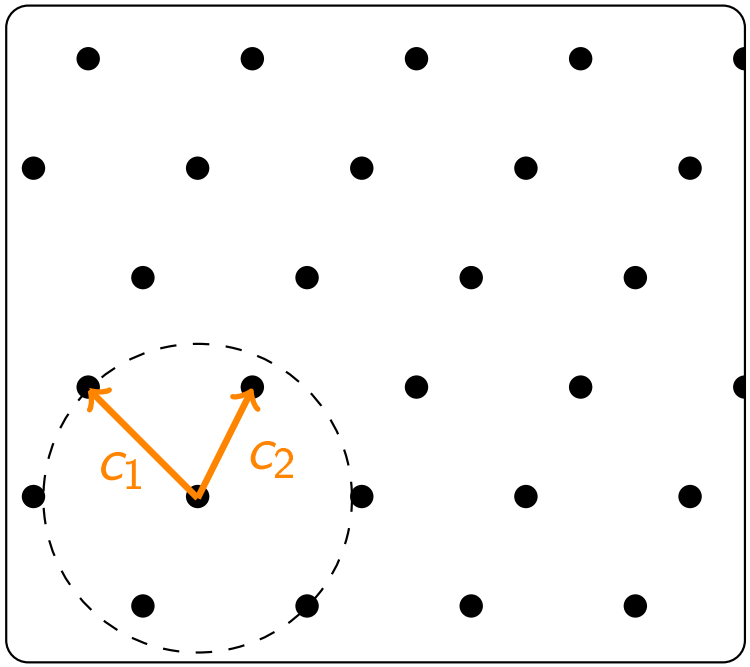
$$\mathrm{dist}(t, \mathcal{L}) = \min_{v \in \mathcal{L}} \|t - v\|$$

Short basis problem

Input:



Output:

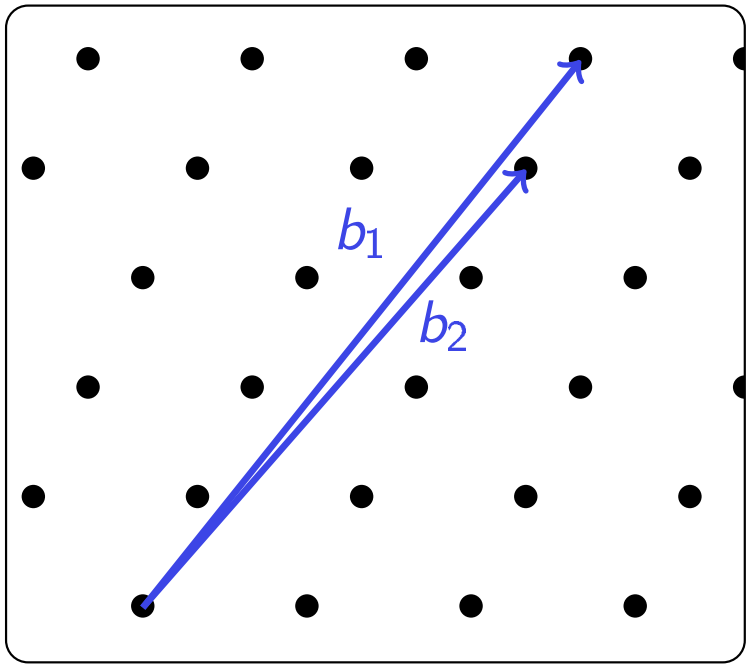


Shortest basis problem

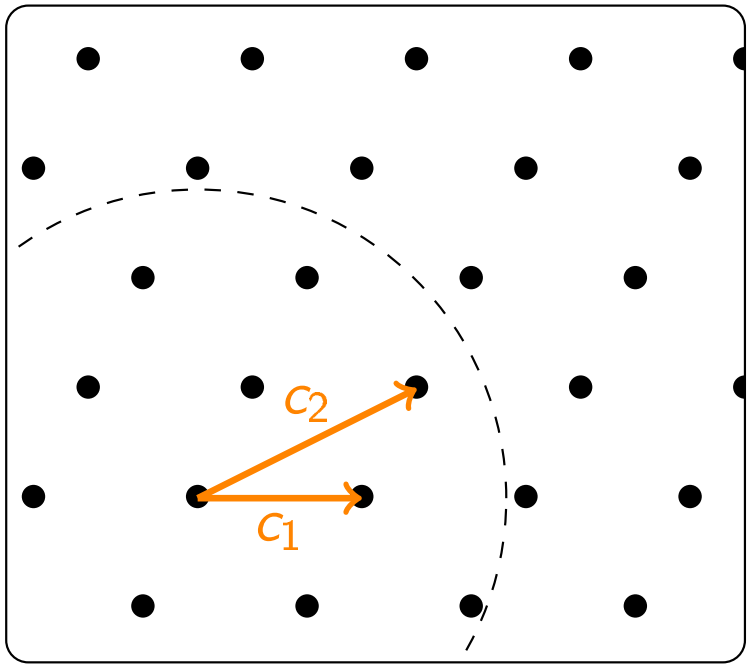
$$\max_i \|c_i\| \leq \min_{B' \text{ basis of } \mathcal{L}} \left(\max_i \|b'_i\| \right)$$

Short basis problem

Input:



Output:



Approximate short basis problem

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

Three different problems?

In terms of hardness:

$$\gamma\text{-SVP} \quad \approx \quad \gamma\text{-CVP} \approx \quad \gamma\text{-approximate short basis}$$

Three different problems?

In terms of hardness:

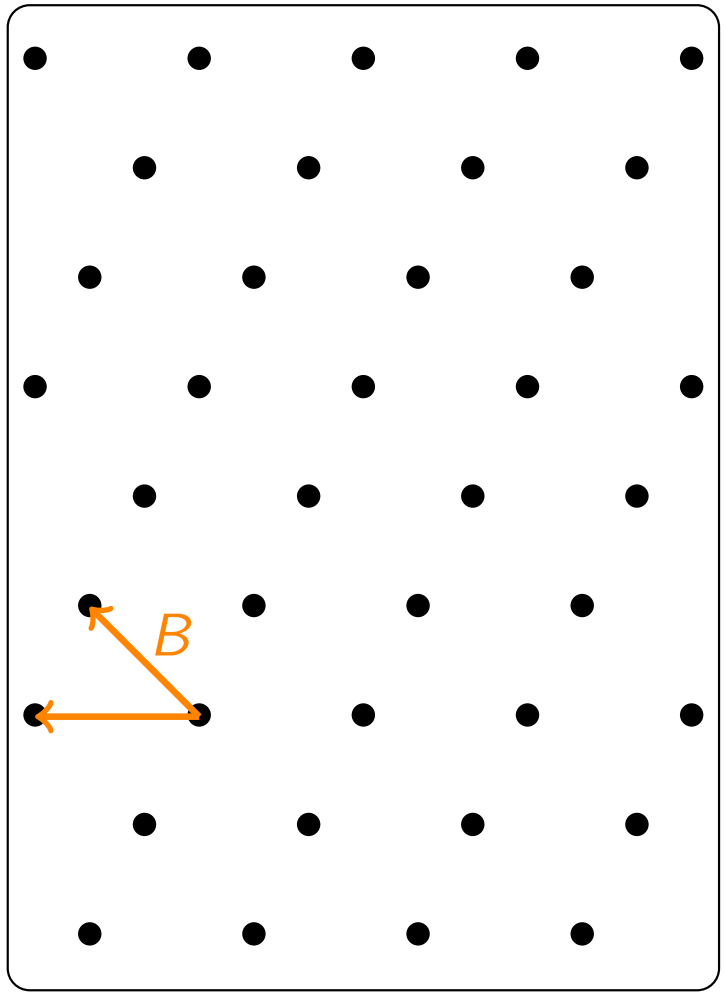
$$\gamma\text{-SVP} \quad \approx \quad \gamma\text{-CVP} \approx \quad \gamma\text{-approximate short basis}$$

- in theory: reductions between (some variants of) the problems
- in practice: same algorithm for all three problems

Digression: canonical representation

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

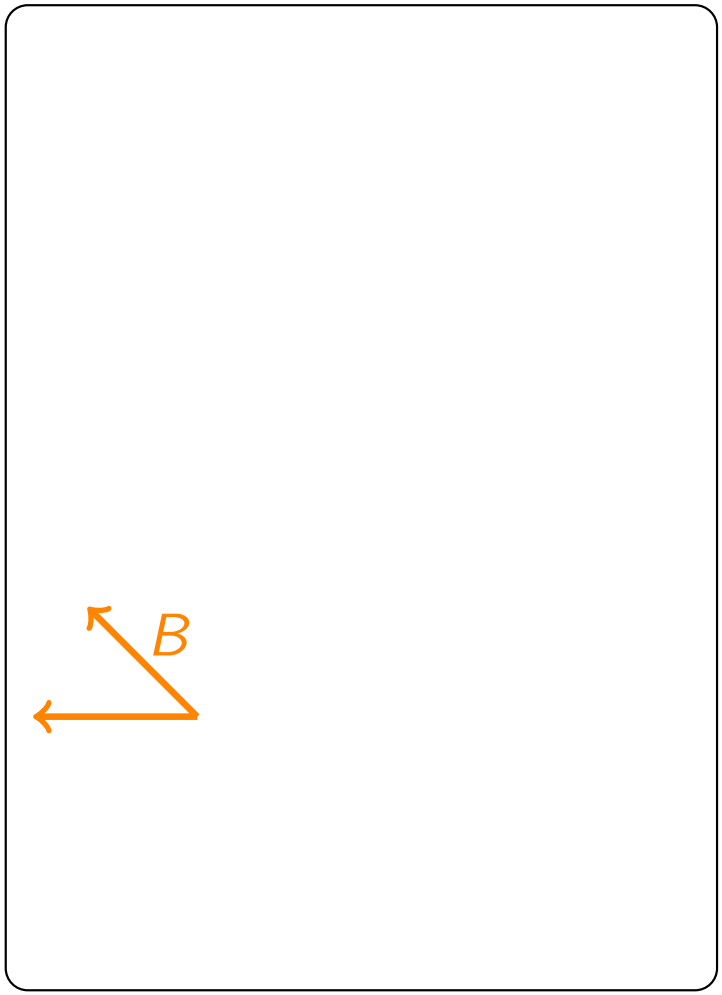
a basis $B \in \mathbb{Z}^{n \times n}$ of $\mathcal{L}$



Digression: canonical representation

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

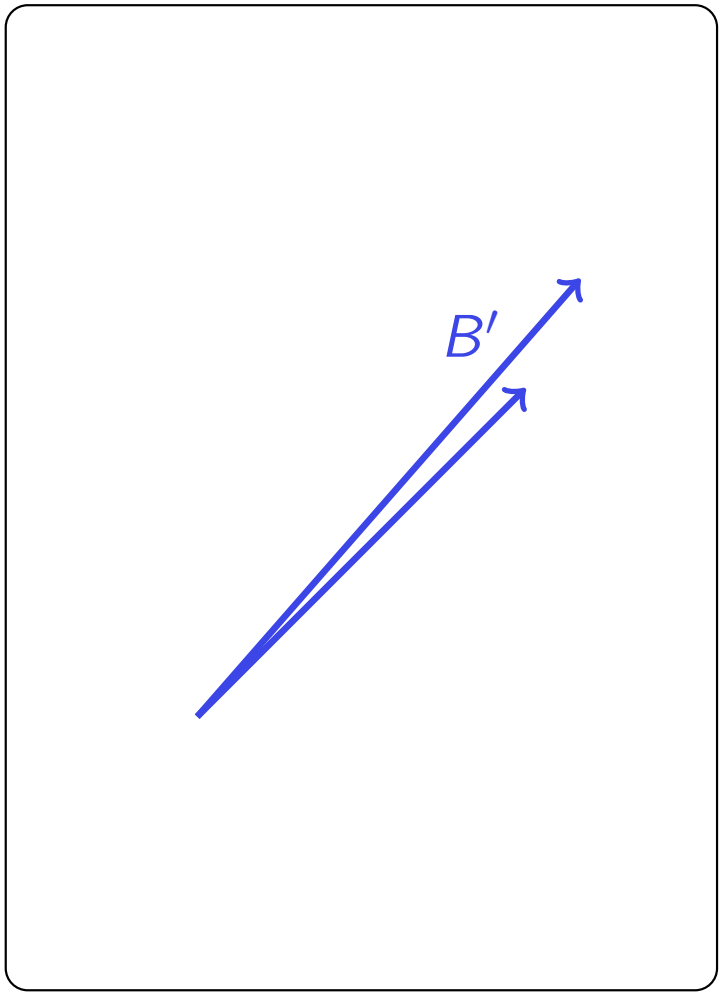
a basis $B \in \mathbb{Z}^{n \times n}$ of $\mathcal{L}$



Digression: canonical representation

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

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



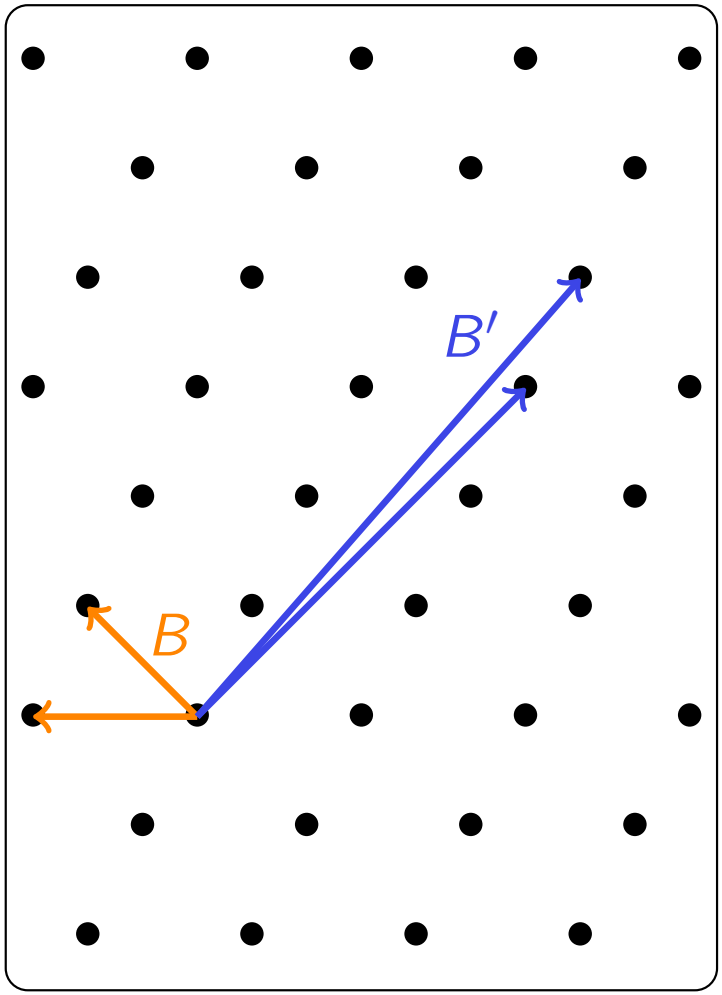
Digression: canonical representation

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

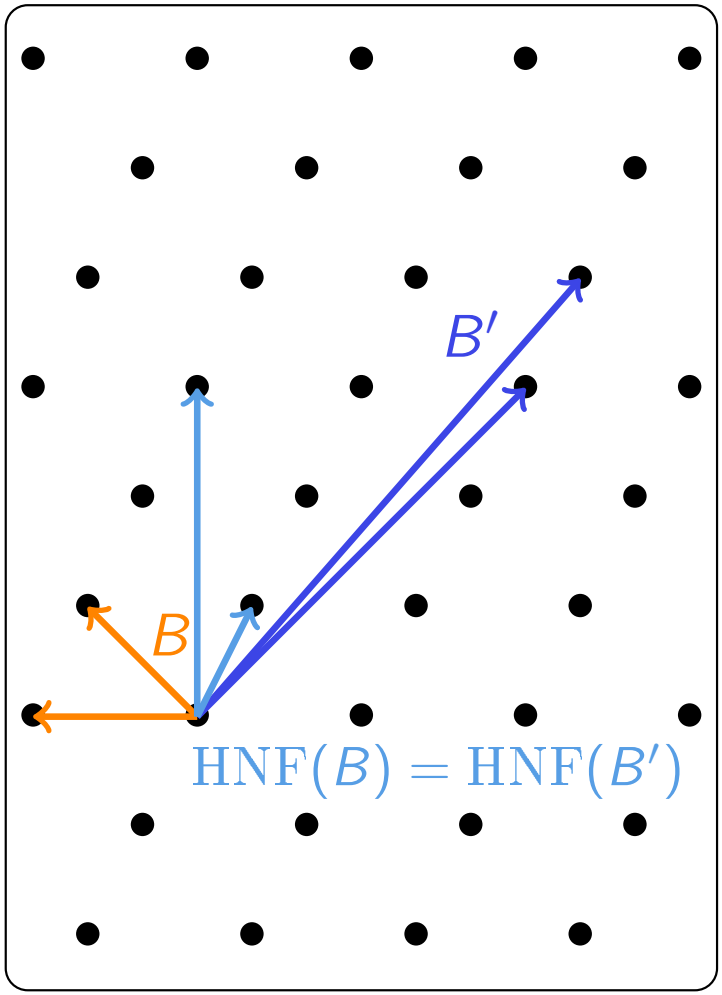
Representation of a lattice $\mathcal{L}$:
a basis $B \in \mathbb{Z}^{n \times n}$ of $\mathcal{L}$

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 $\mathcal{L}$

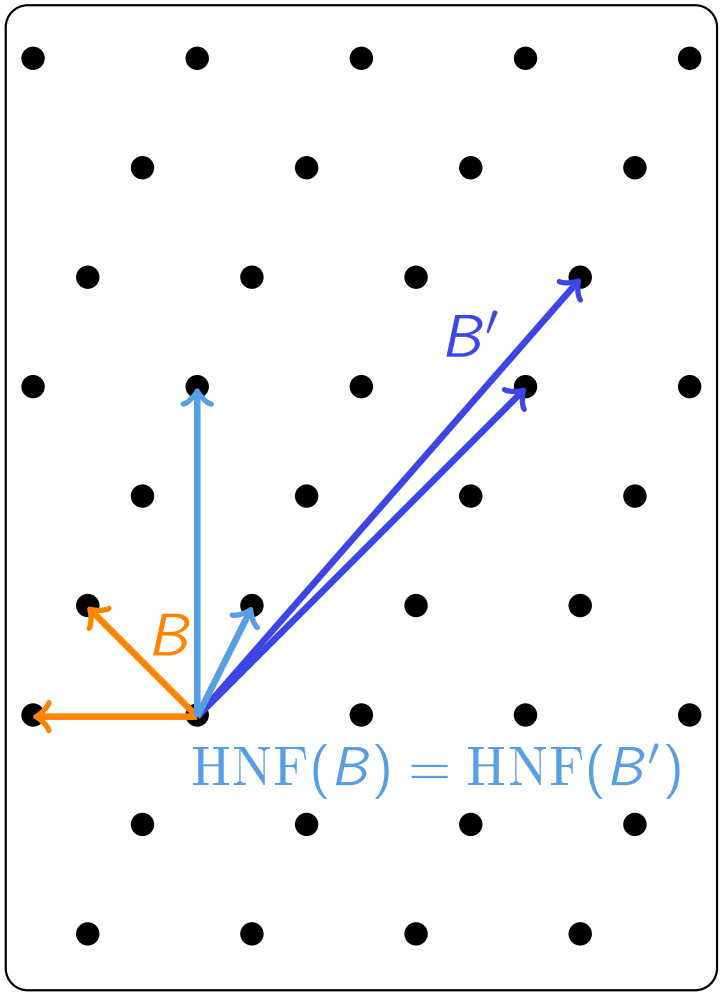
Difficulty:

- 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

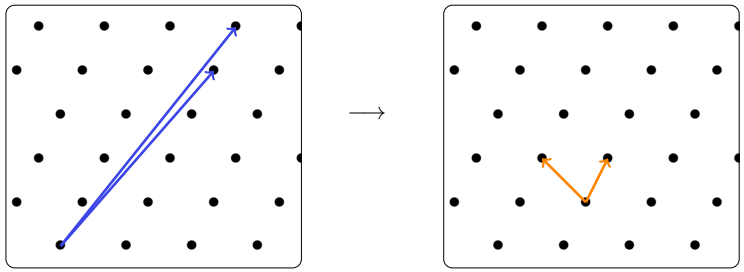
$\Rightarrow$ canonical representation of $\mathcal{L}$
(i.e., worse basis ever)



Section's conclusion

We have seen three lattice problems:

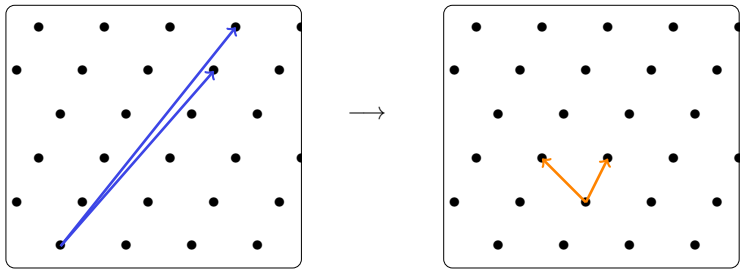
- shortest vector problem (SVP)
- closest vector problem (CVP)
- shortest basis problem



Section's conclusion

We have seen three lattice problems:

- shortest vector problem (SVP)
- closest vector problem (CVP)
- shortest basis problem



Many other variants:

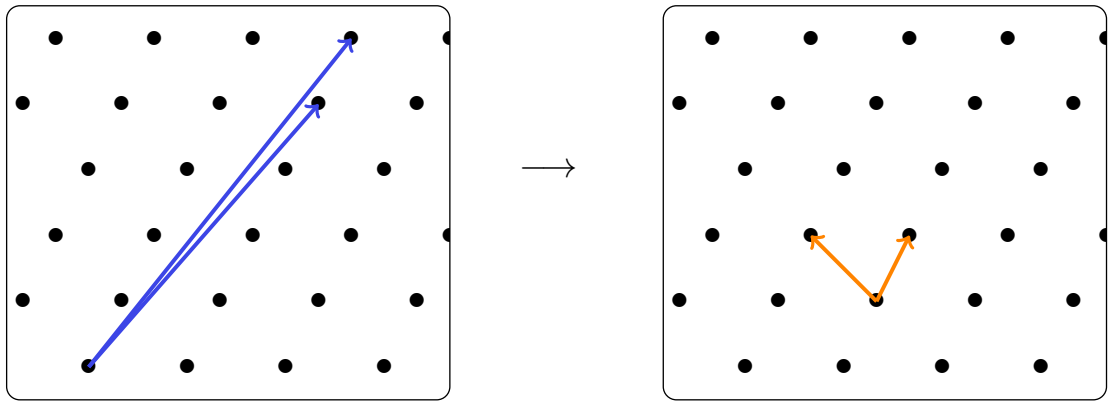
- exact vs approx
- search vs decision
- promise variant

$\Rightarrow$ 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 $\mathcal{L}_i$
  ($\mathcal{L}_i$ is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on $\mathcal{L}_i$

[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 $\mathcal{L}_i$
  ($\mathcal{L}_i$ is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on $\mathcal{L}_i$

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 $\mathcal{L}_i$
  ($\mathcal{L}_i$ is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on $\mathcal{L}_i$

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:

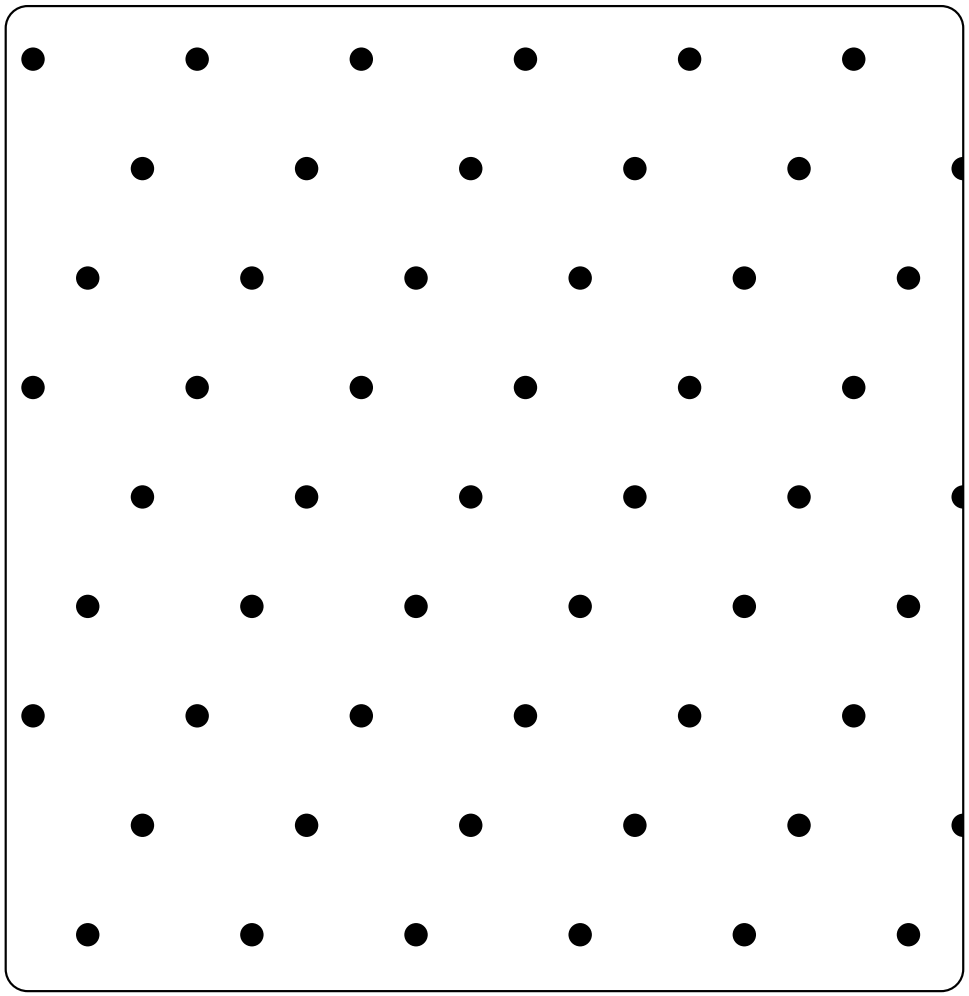
- while there exists i such that (b_i, b_{i+1}) is not a γ'-short basis of $\mathcal{L}_i$ with $\gamma' = 4/3$
  ($\mathcal{L}_i$ is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on $\mathcal{L}_i$

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.

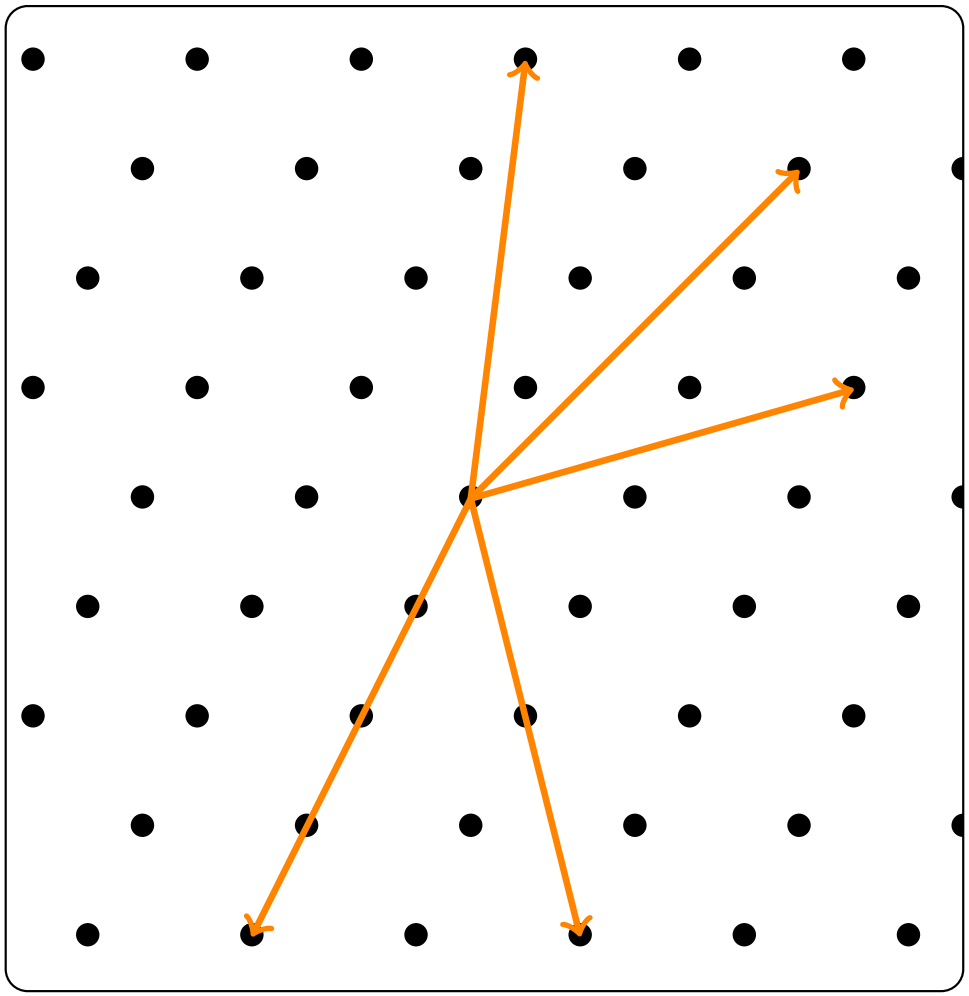
Sieving algorithm [AKS01]



Sieving:

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

Sieving algorithm [AKS01]

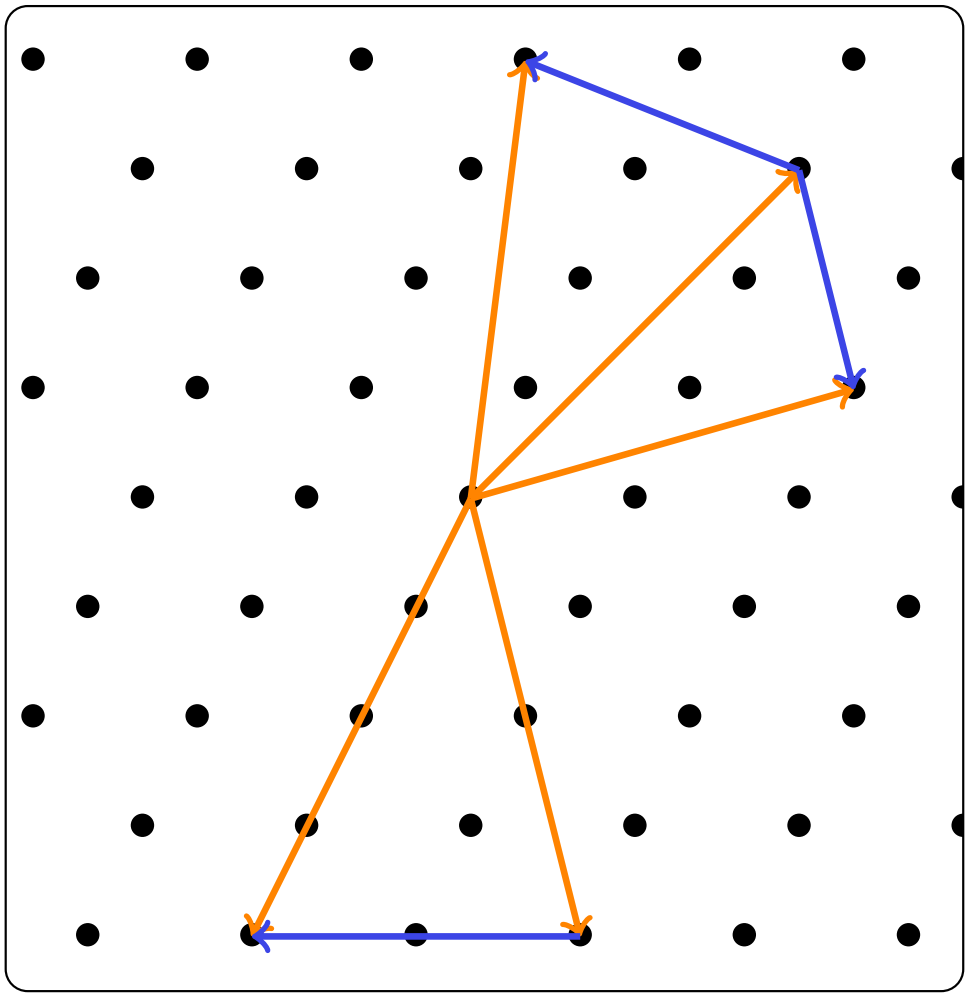


Sieving:

- Create many large vectors

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

Sieving algorithm [AKS01]

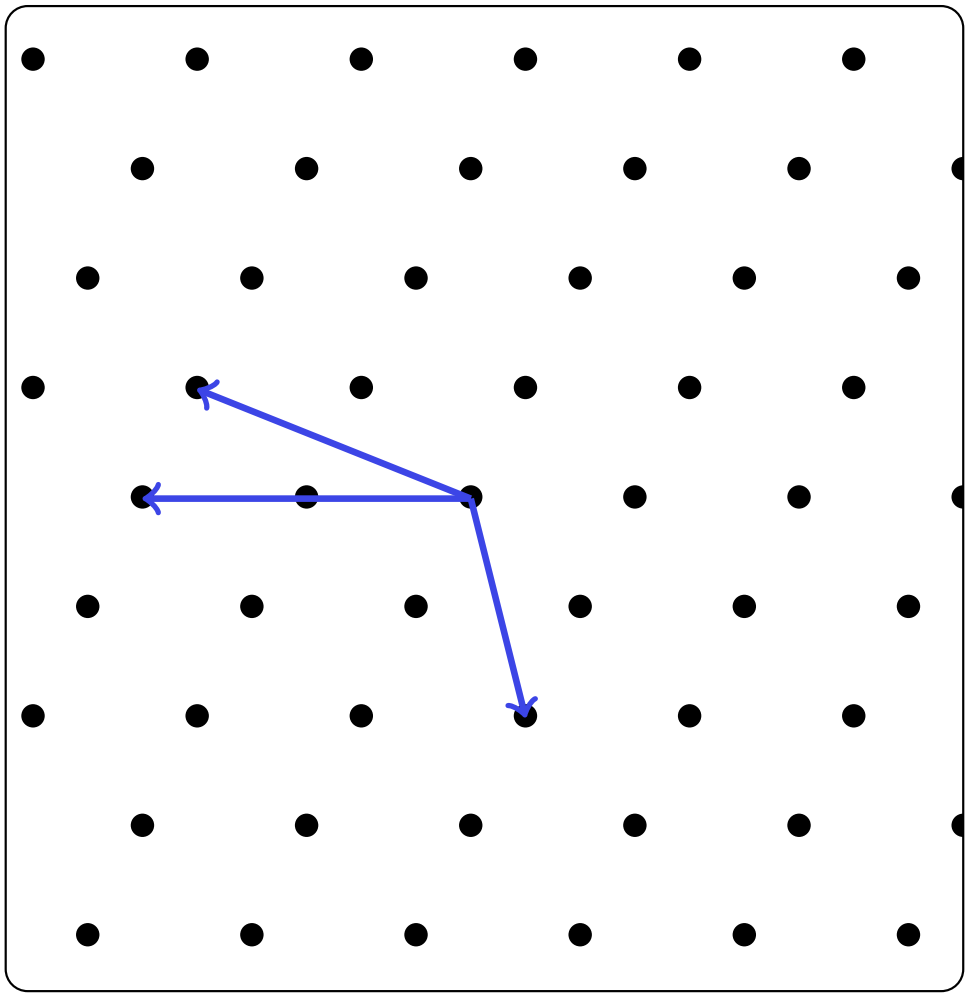


Sieving:

- 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]

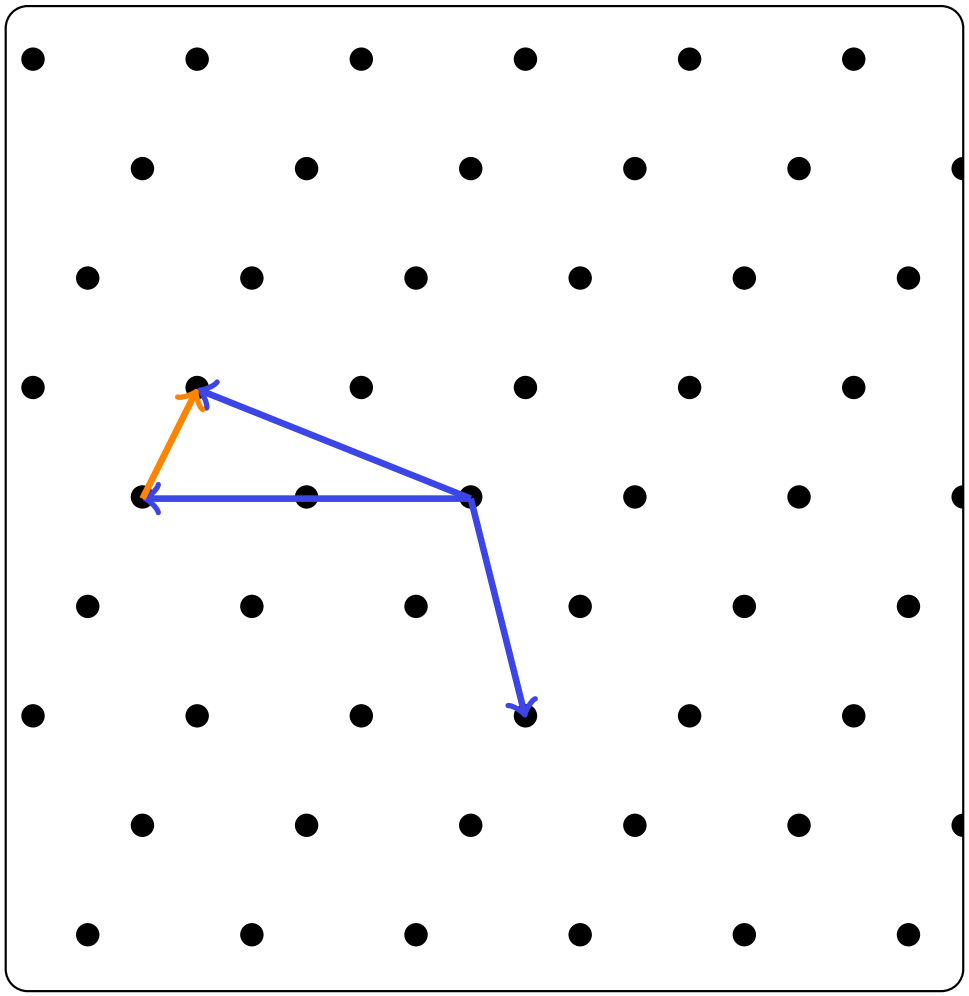


Sieving:

- 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]

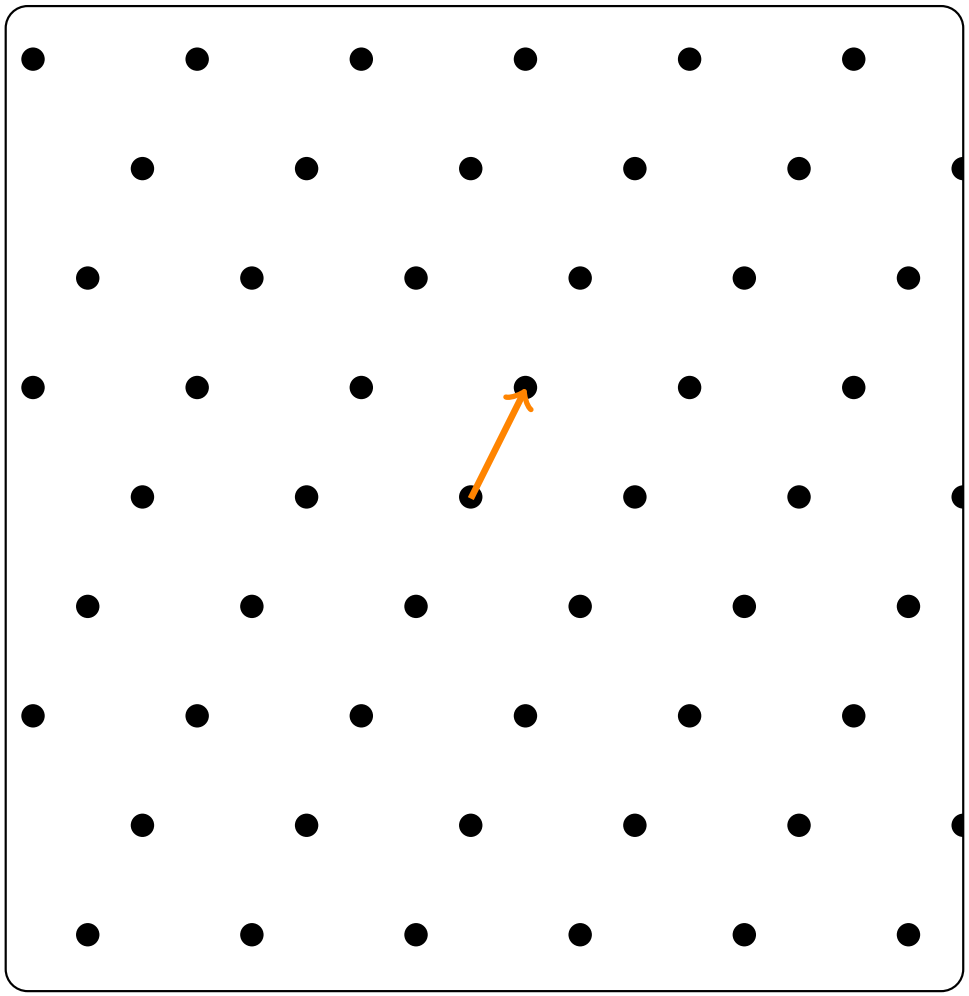


Sieving:

- 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]

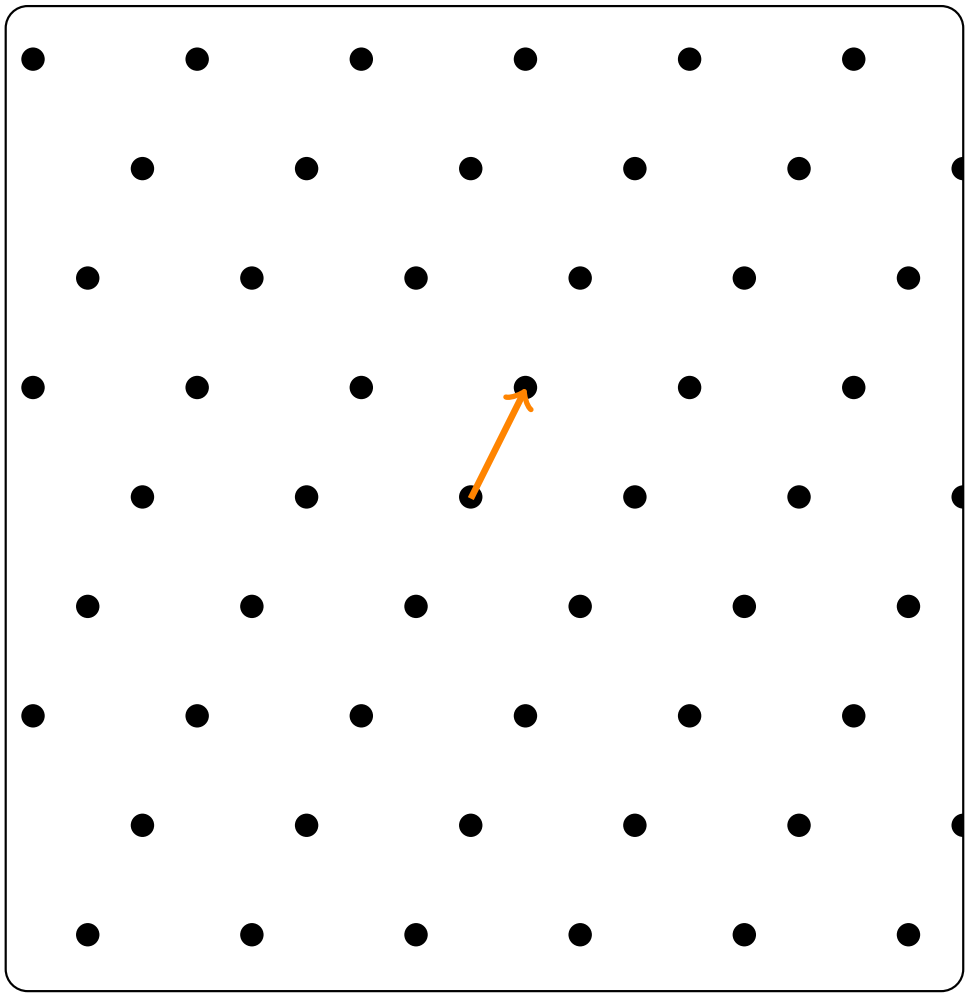


Sieving:

- 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]



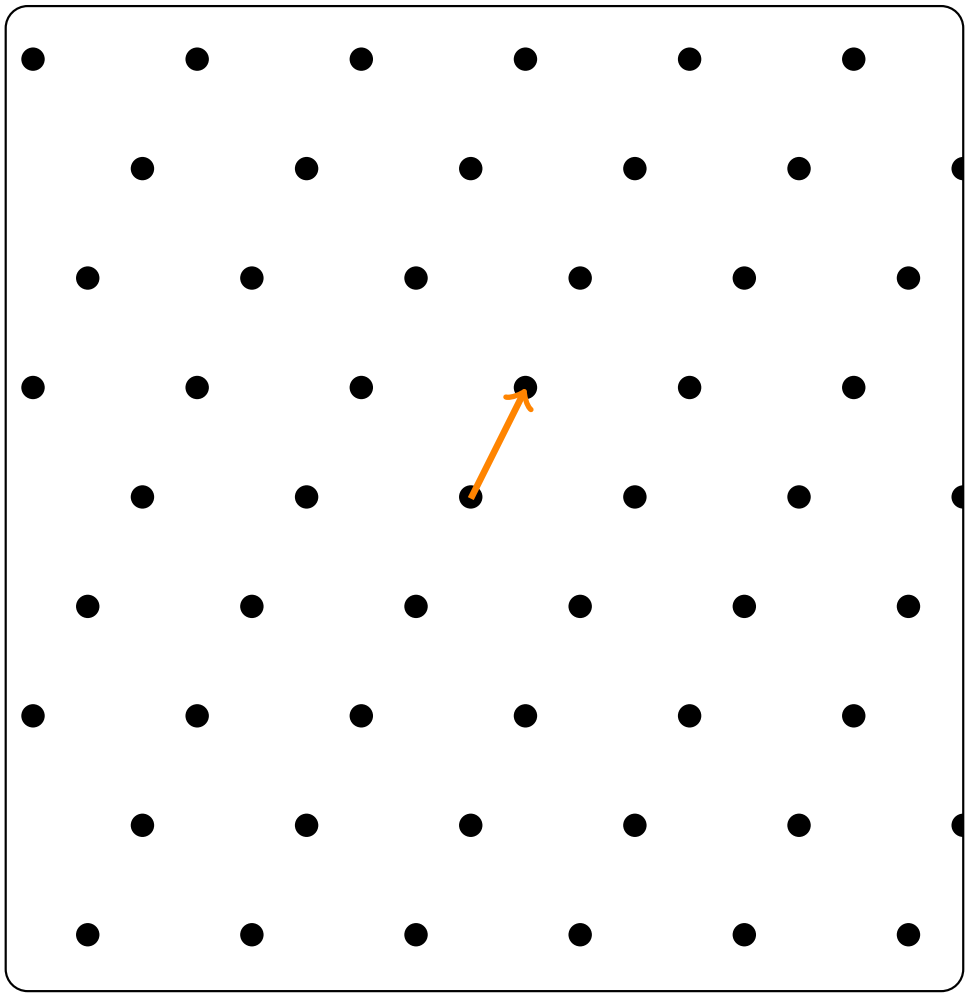
Sieving:

- 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:

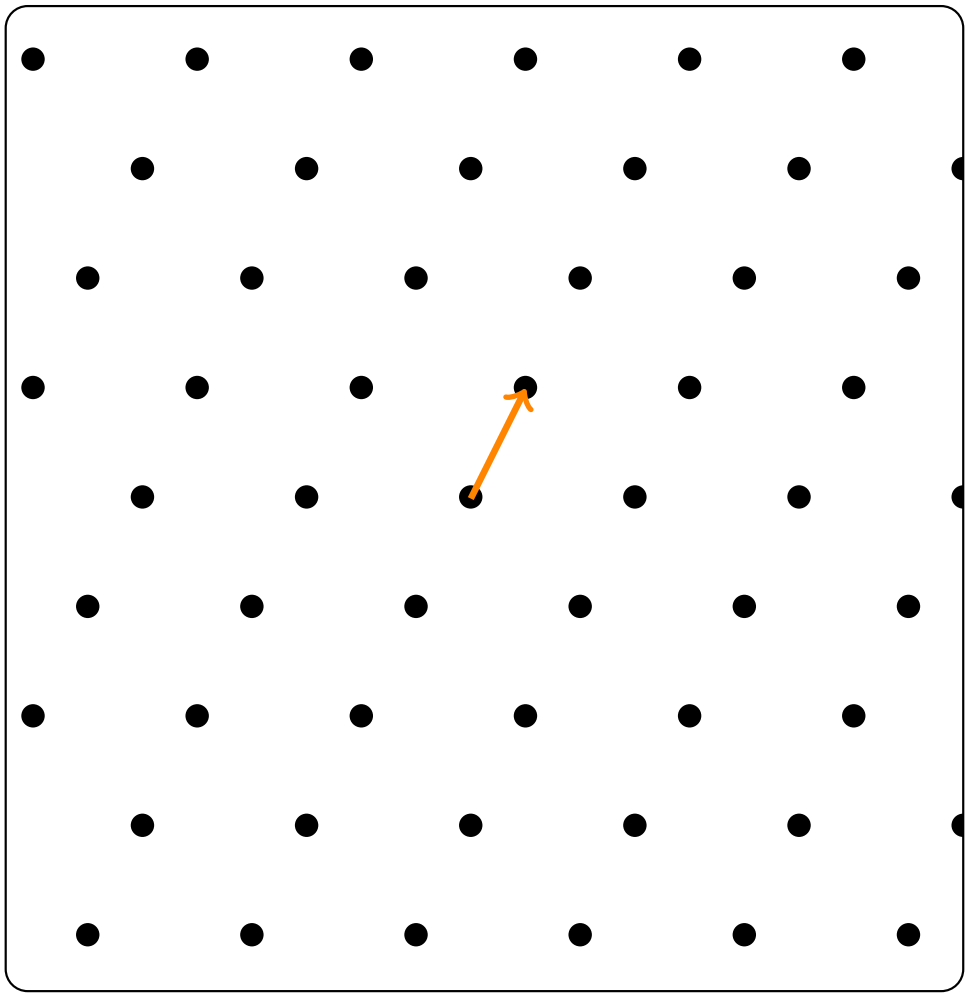
- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

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:

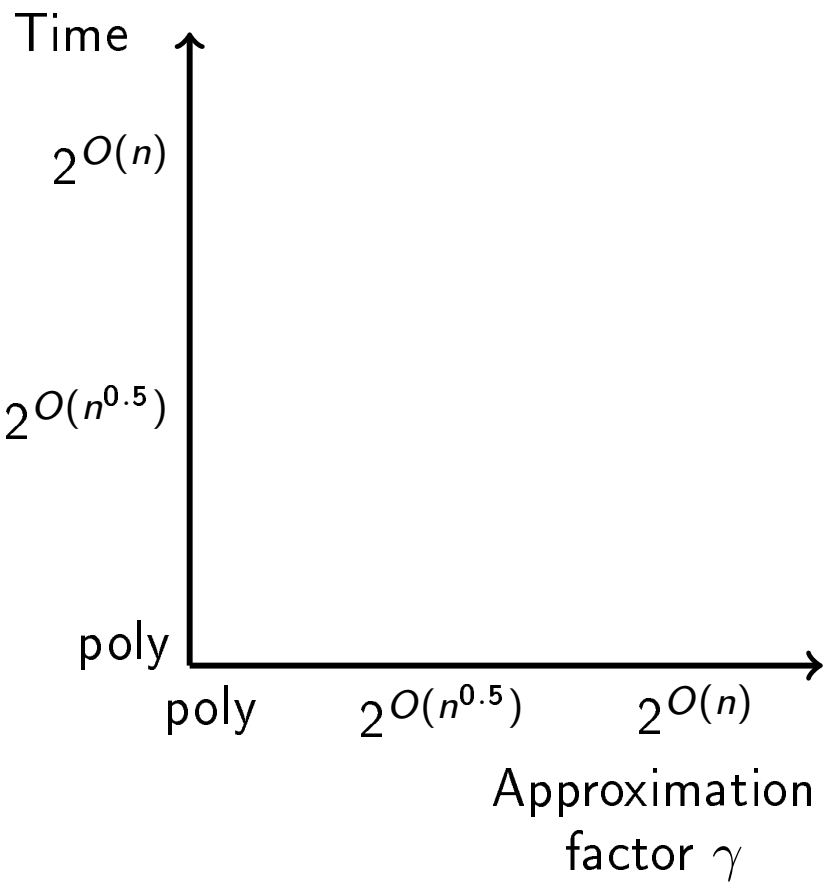
- Create many large vectors
- Subtract close ones to create shorter vectors
- Repeat with the shorter vectors

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

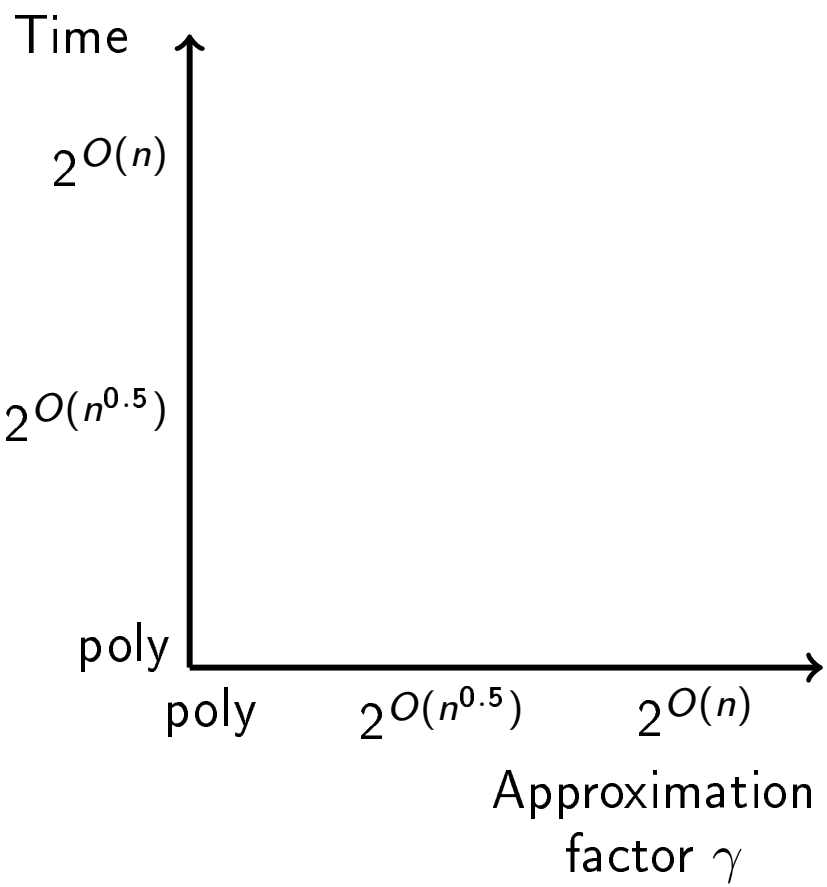
- finds a shortest basis
- runs in time $2^{O(n)}$

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

Summary and BKZ



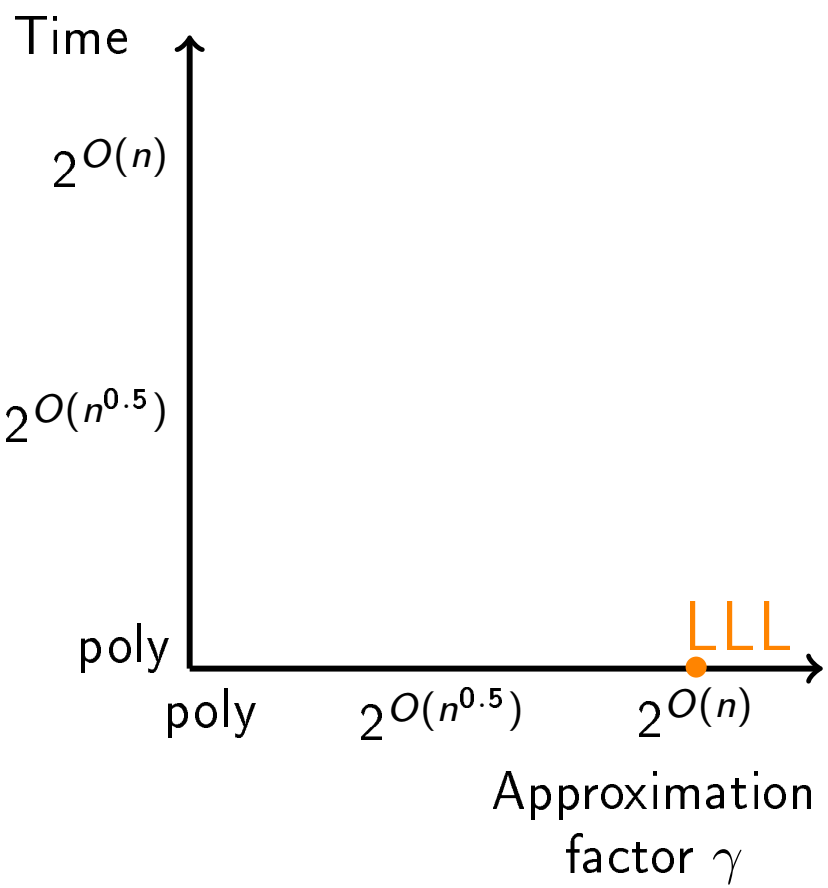
Summary and BKZ



Lagrange-Gauss algorithm: dim 2

- shortest basis
- polynomial time

Summary and BKZ



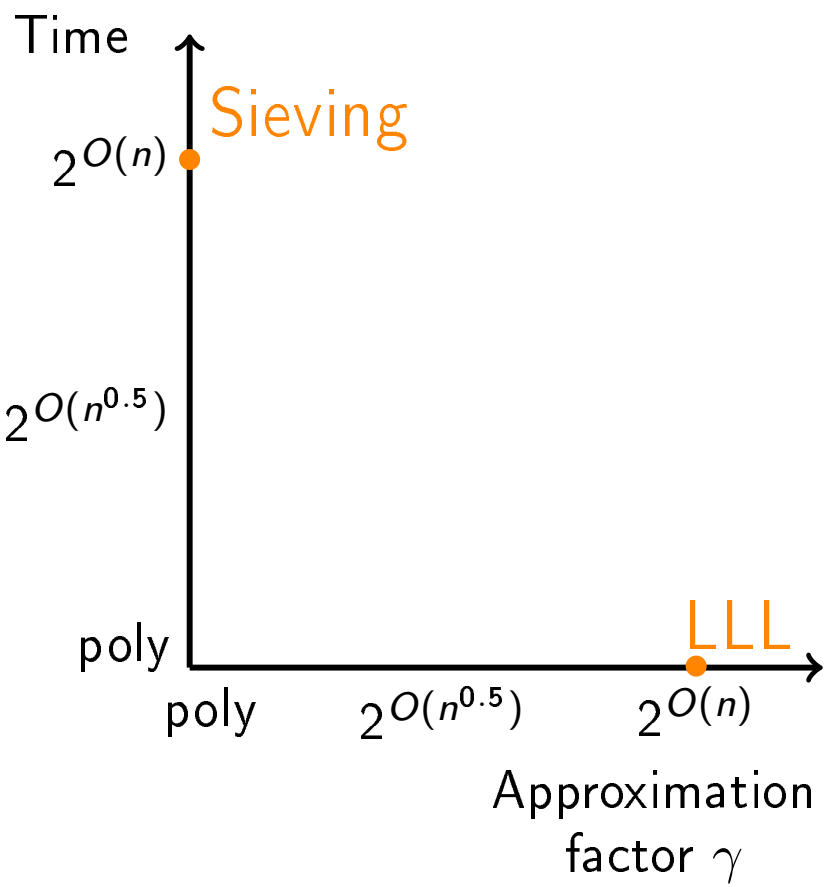
Lagrange-Gauss algorithm: dim 2

- 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

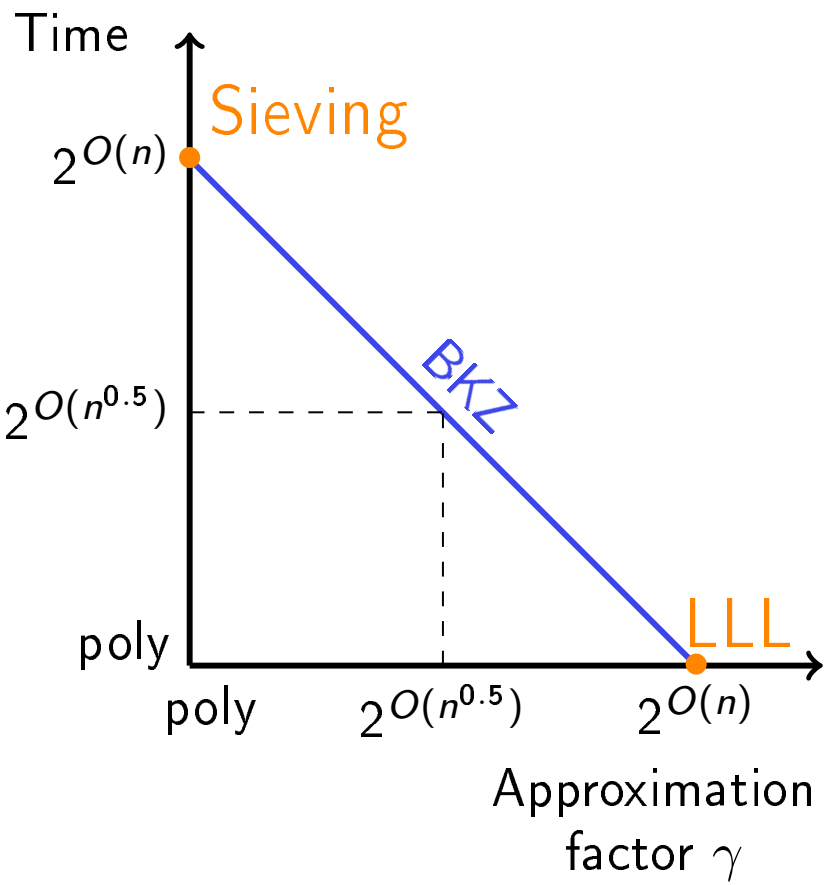
- γ-short basis with $\gamma = 2^n$
- polynomial time

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

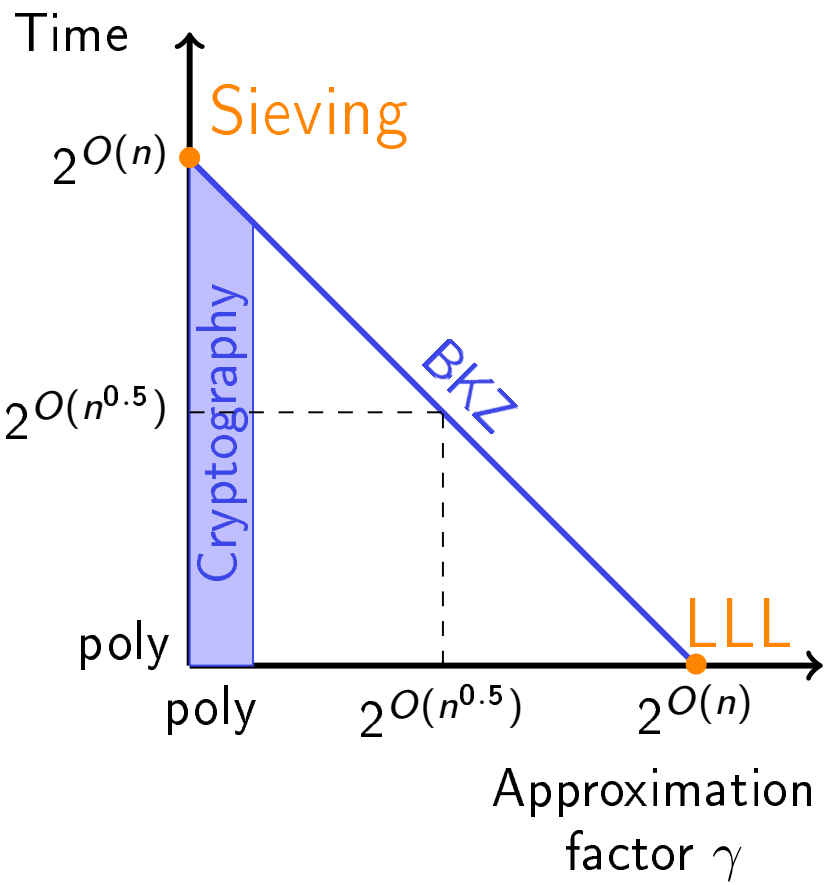
Sieving algorithm: dim n

- 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: computing short bases of lattices is very hard (even quantumly)

- if n (the dimension) is large enough (e.g., $n \geq 1000$)
- if γ (the approx factor) is small enough (e.g., $\gamma \lesssim n$)

Section's conclusion

Takeaway: computing short bases of lattices is very hard (even quantumly)

- if n (the dimension) is large enough (e.g., $n \geq 1000$)
- if γ (the approx factor) is small enough (e.g., $\gamma \lesssim n$)

But...

Section's conclusion

Takeaway: computing short bases of lattices is very hard (even quantumly)

- if n (the dimension) is large enough (e.g., $n \geq 1000$)
- if γ (the approx factor) is small enough (e.g., $\gamma \lesssim n$)

But...

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

Section's conclusion

Takeaway: computing short bases of lattices is very hard (even quantumly)

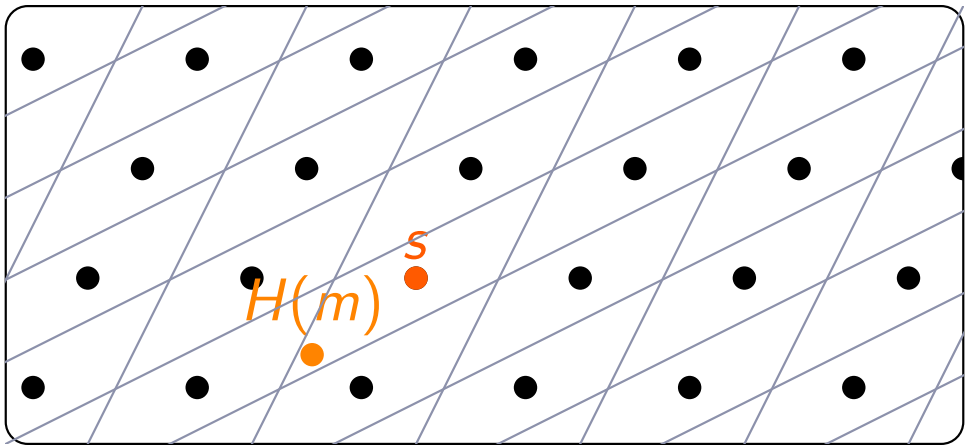
- if n (the dimension) is large enough (e.g., $n \geq 1000$)
- if γ (the approx factor) is small enough (e.g., $\gamma \lesssim n$)

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

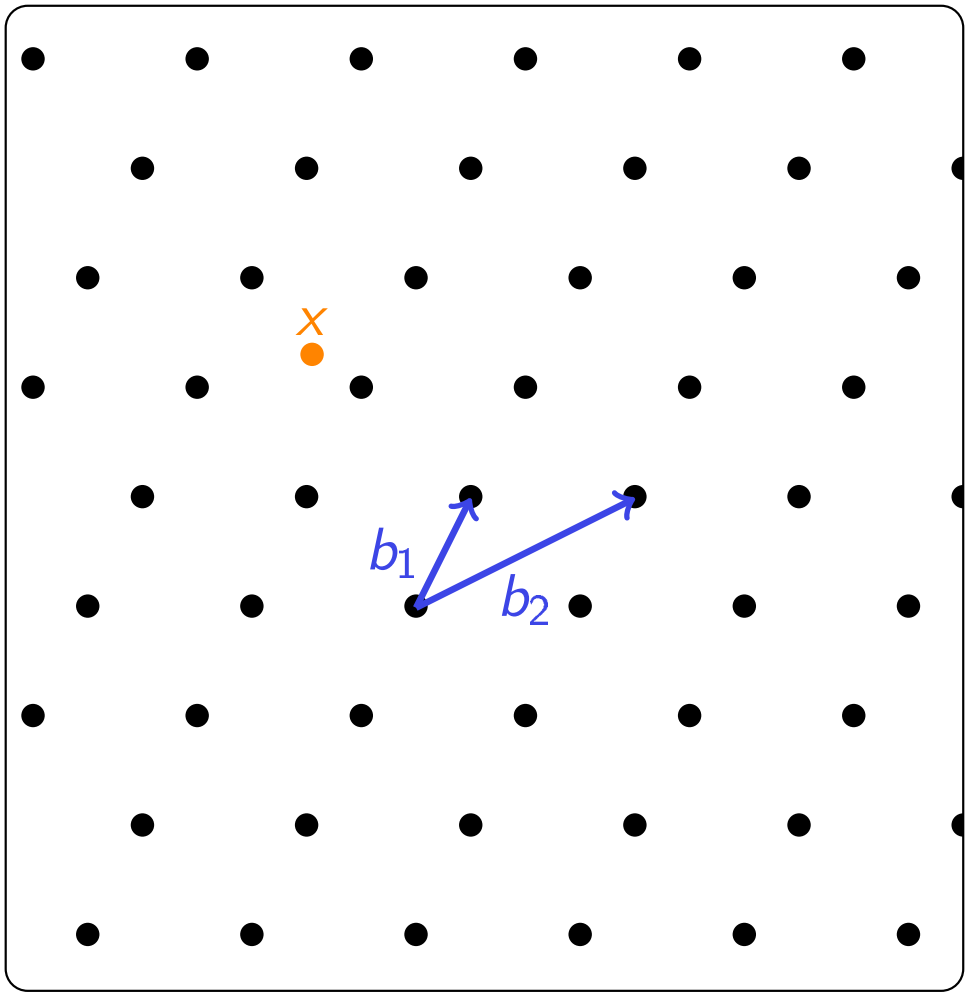
Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Sign}(m, sk) \rightsquigarrow s$
- $\mathrm{Verify}(m, s, pk) \rightsquigarrow$ yes/no

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 $\mathrm{Verify}(m, s, pk) = \mathrm{yes}$)

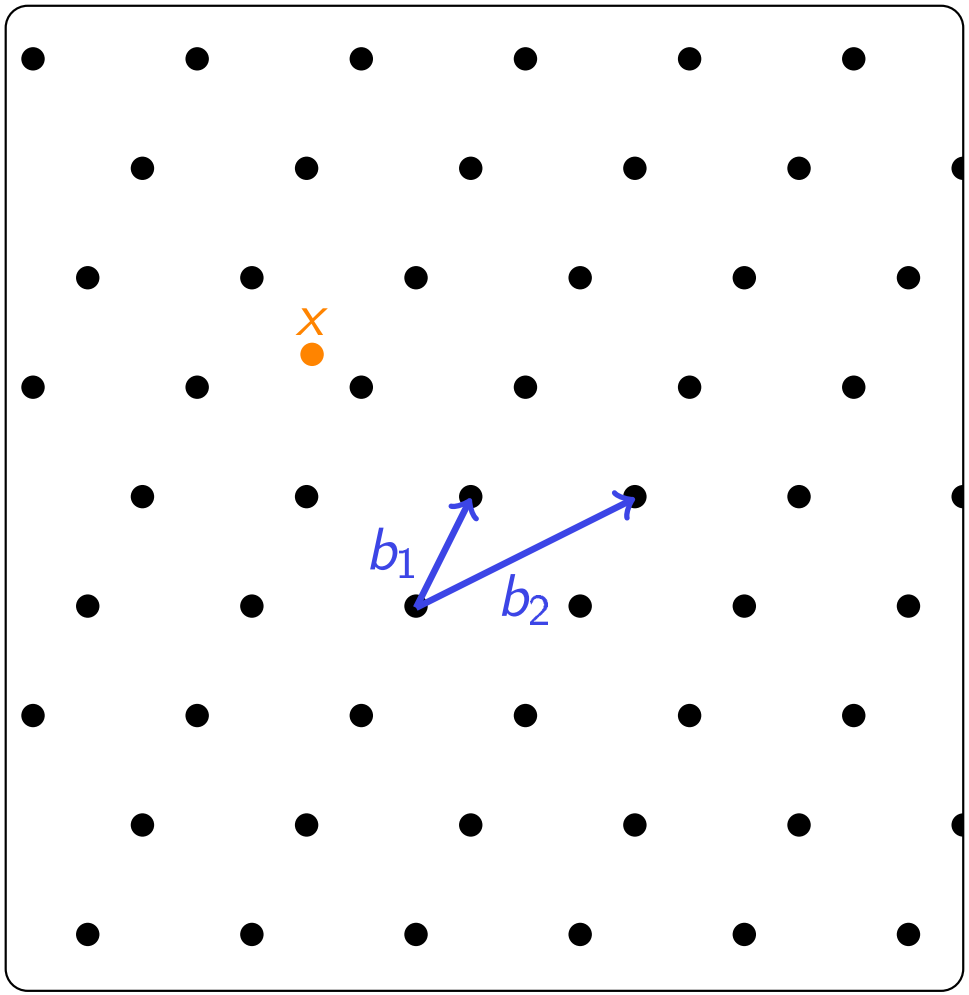
Finding a close vector using a short basis



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

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

Finding a close vector using a short basis

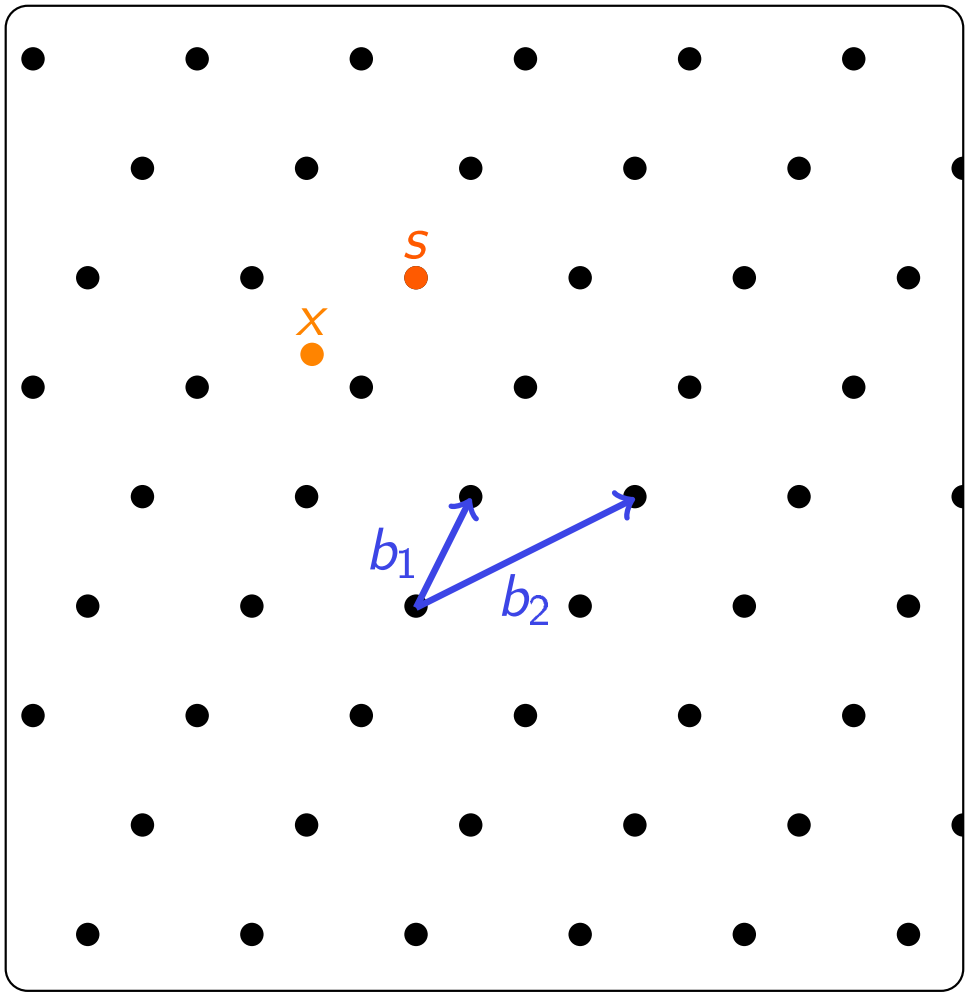


Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

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

Algo: round each coordinate

Finding a close vector using a short basis



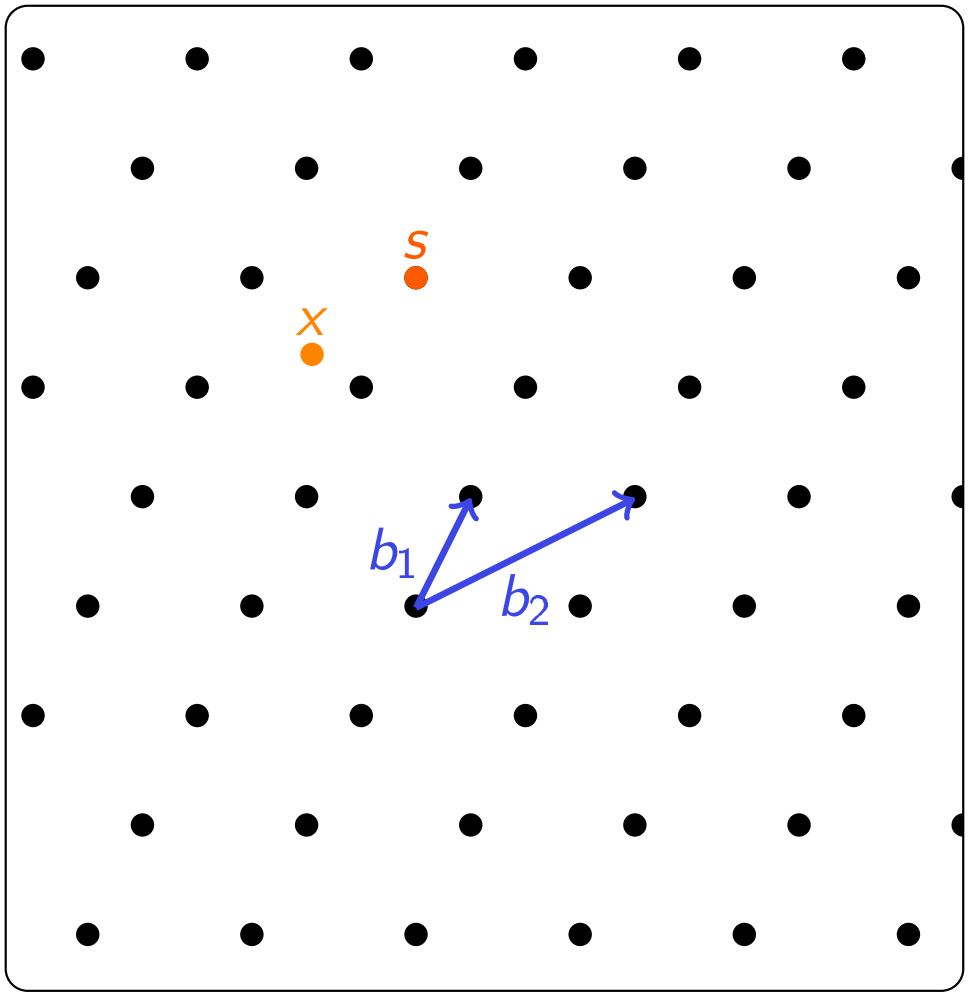
Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

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

Algo: round each coordinate

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

Finding a close vector 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

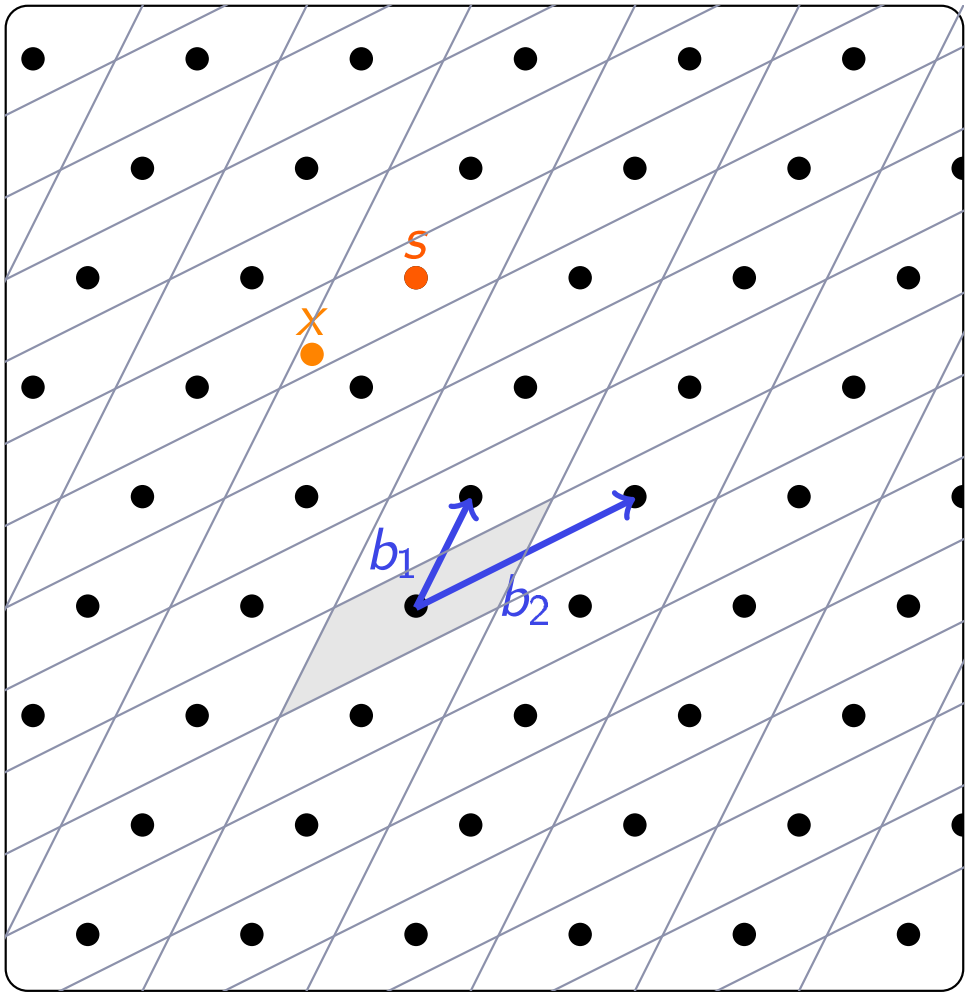
Algo: round each coordinate

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

The smaller the basis, the closer the solution

(called Babai's round-off algorithm)

Finding a close vector 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

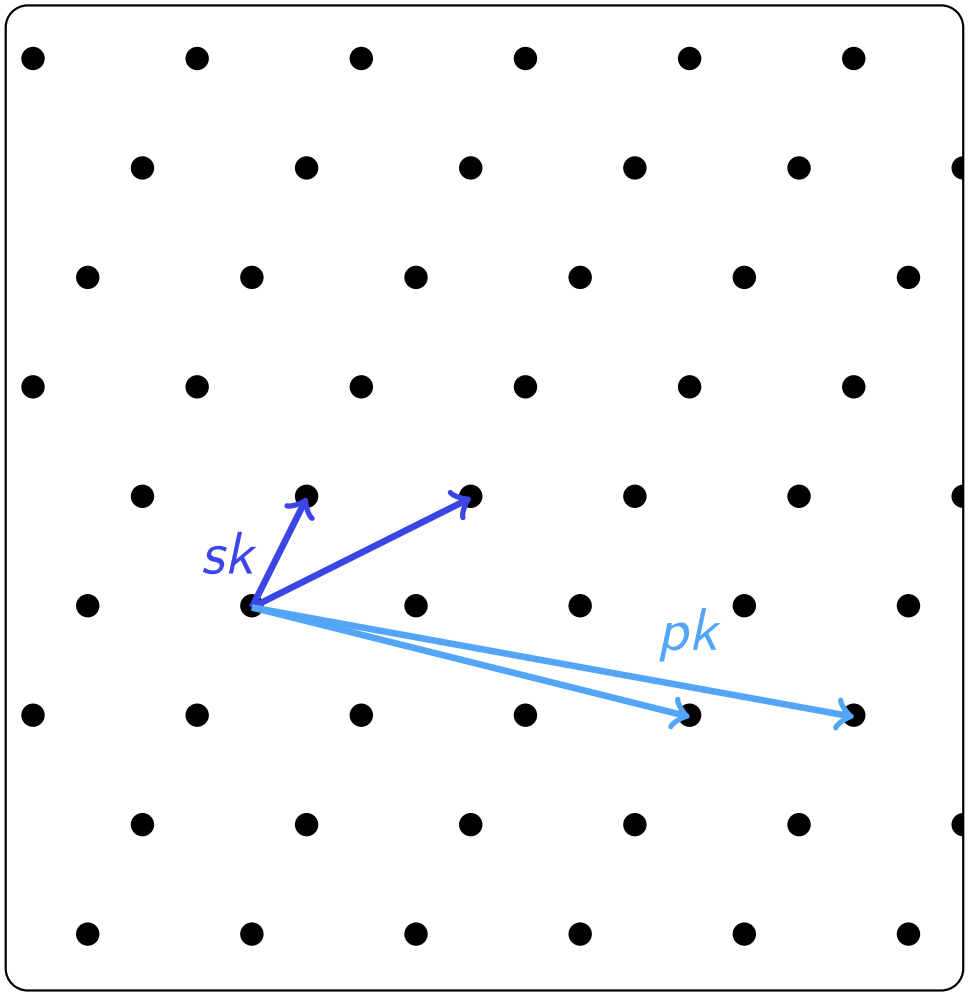
Output: $s = 4 \cdot b_1 - 1 \cdot b_2$

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]

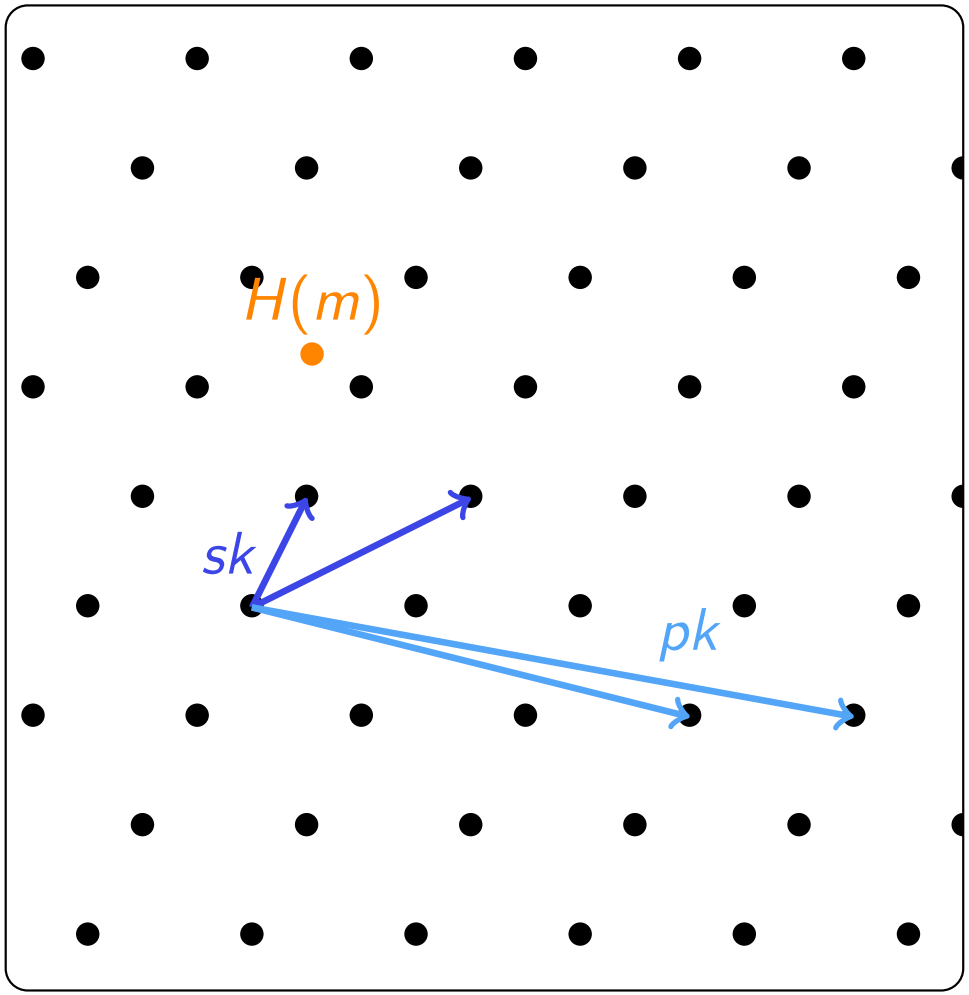


KeyGen:

- 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:

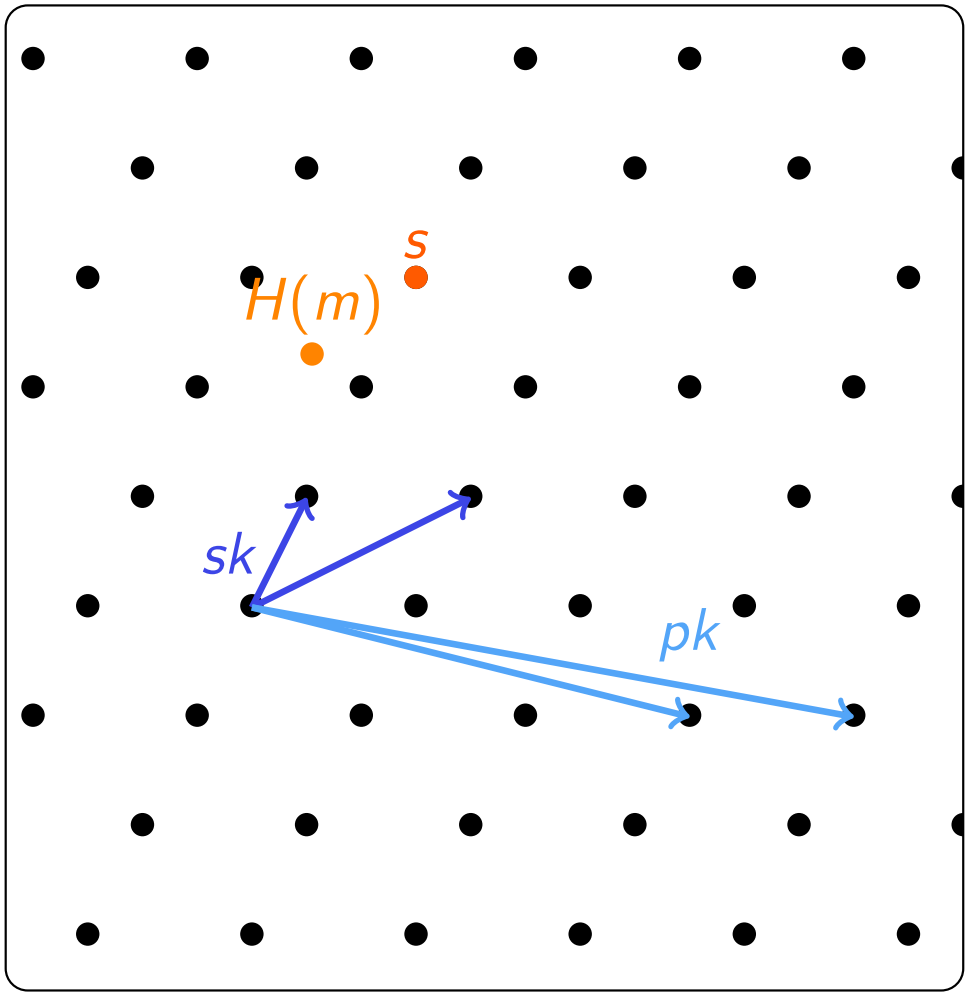
- pk = bad basis of $\mathcal{L}$
  (e.g., HNF basis)
- sk = short basis of $\mathcal{L}$

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:

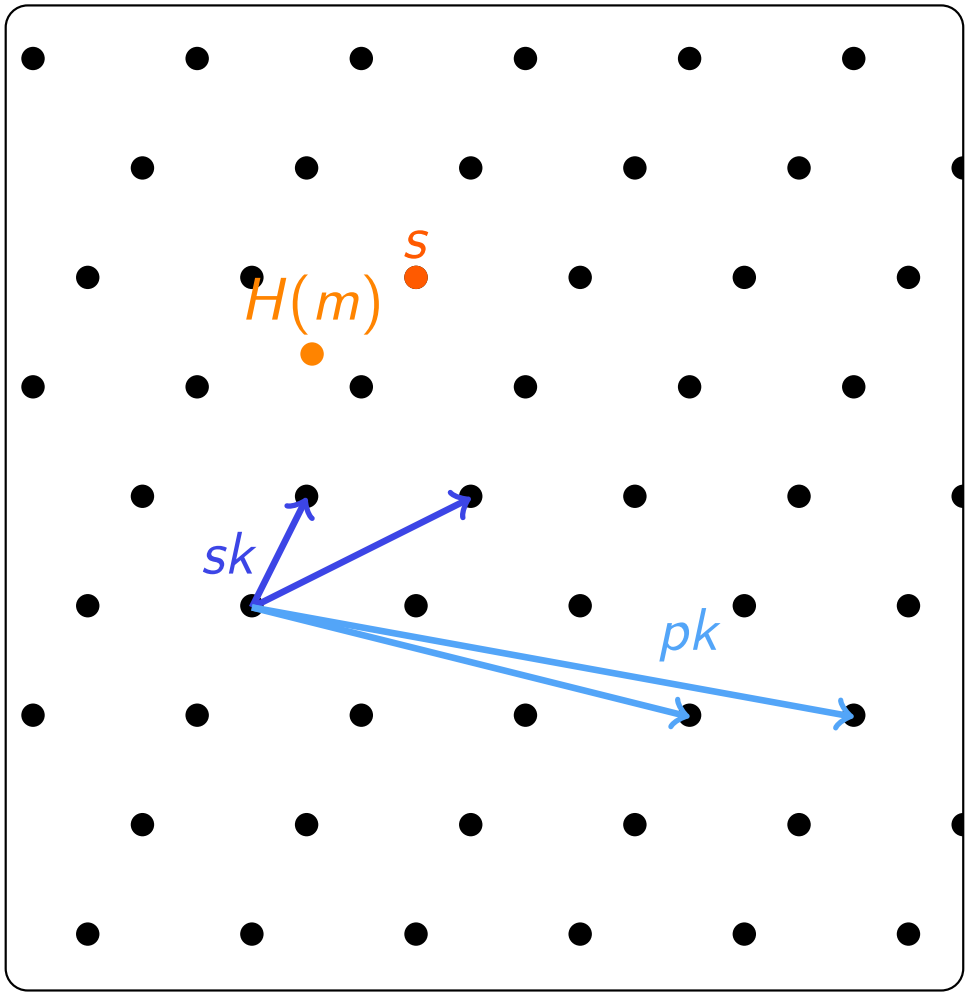
- pk = bad basis of $\mathcal{L}$
  (e.g., HNF basis)
- sk = short basis of $\mathcal{L}$

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):

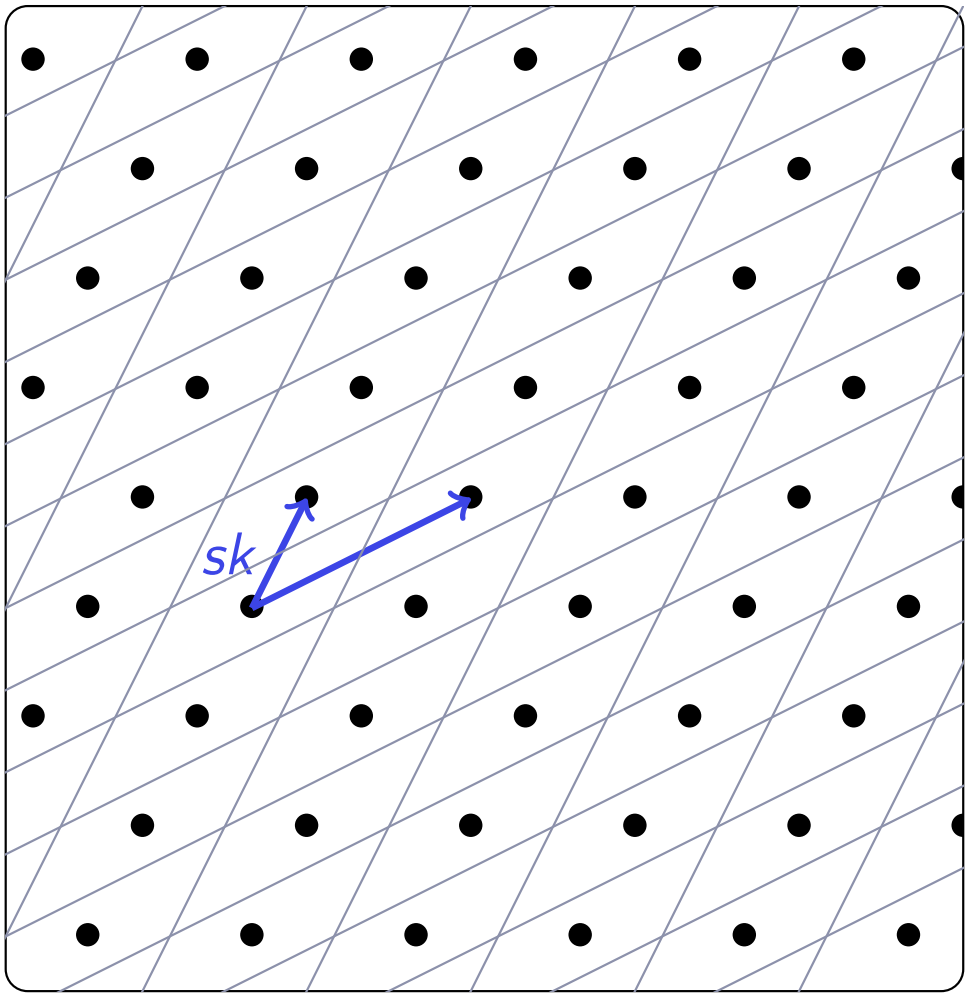
- $x = H(m)$ (hash the message)
- output $s \in \mathcal{L}$ close to $H(m)$

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

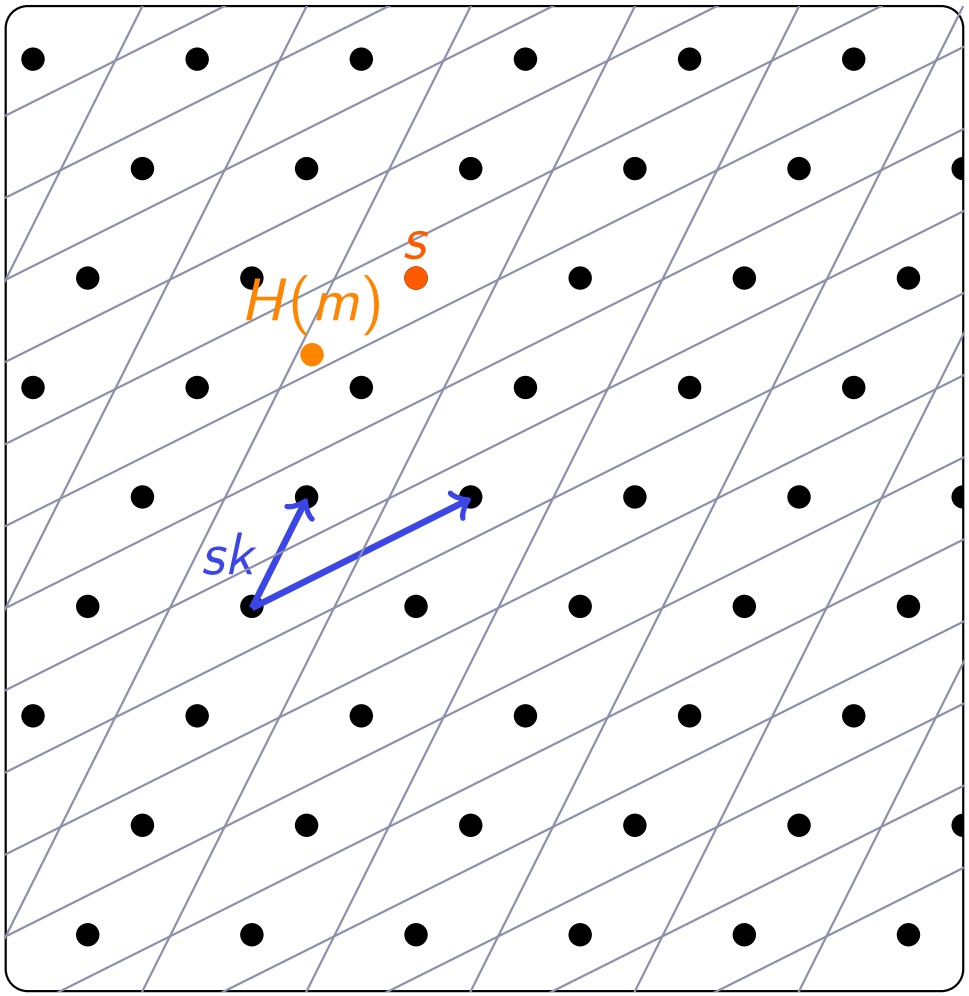
Attack on this first idea [NR06]



Parallelepiped attack:

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

Attack on this first idea [NR06]

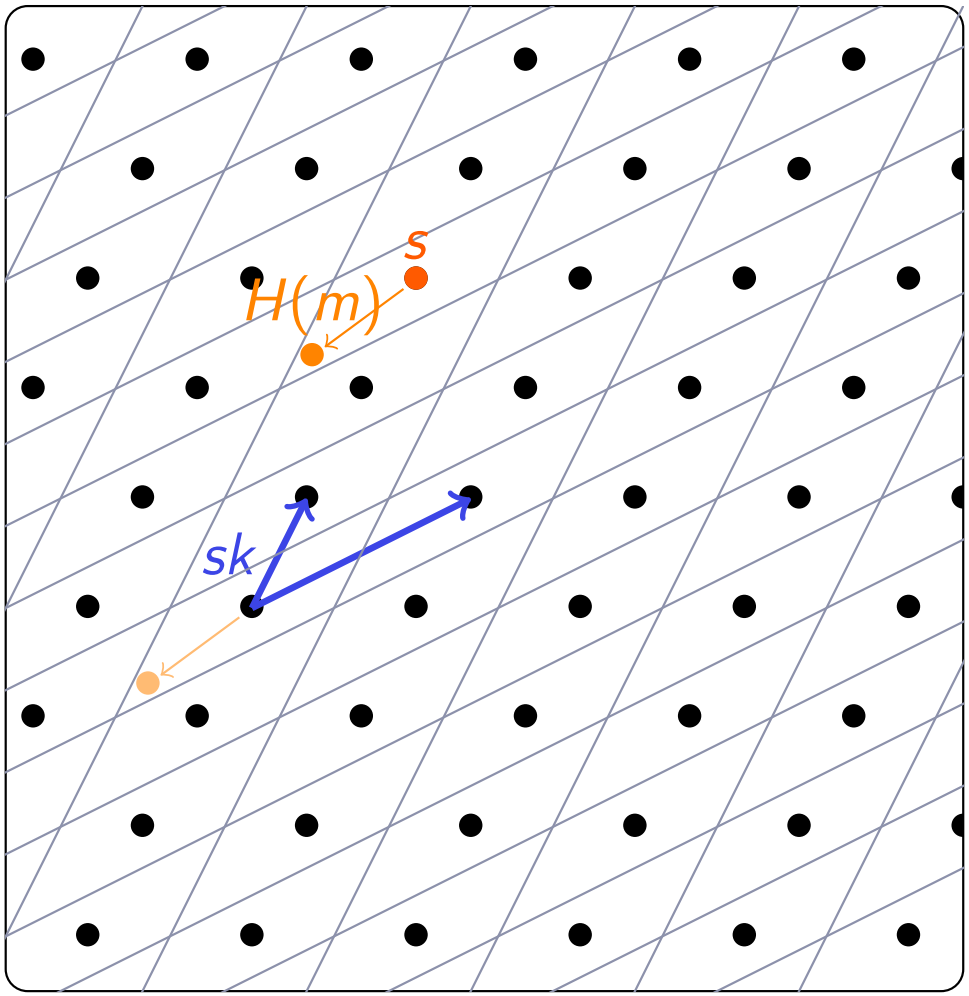


Parallelepiped attack:

- 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]

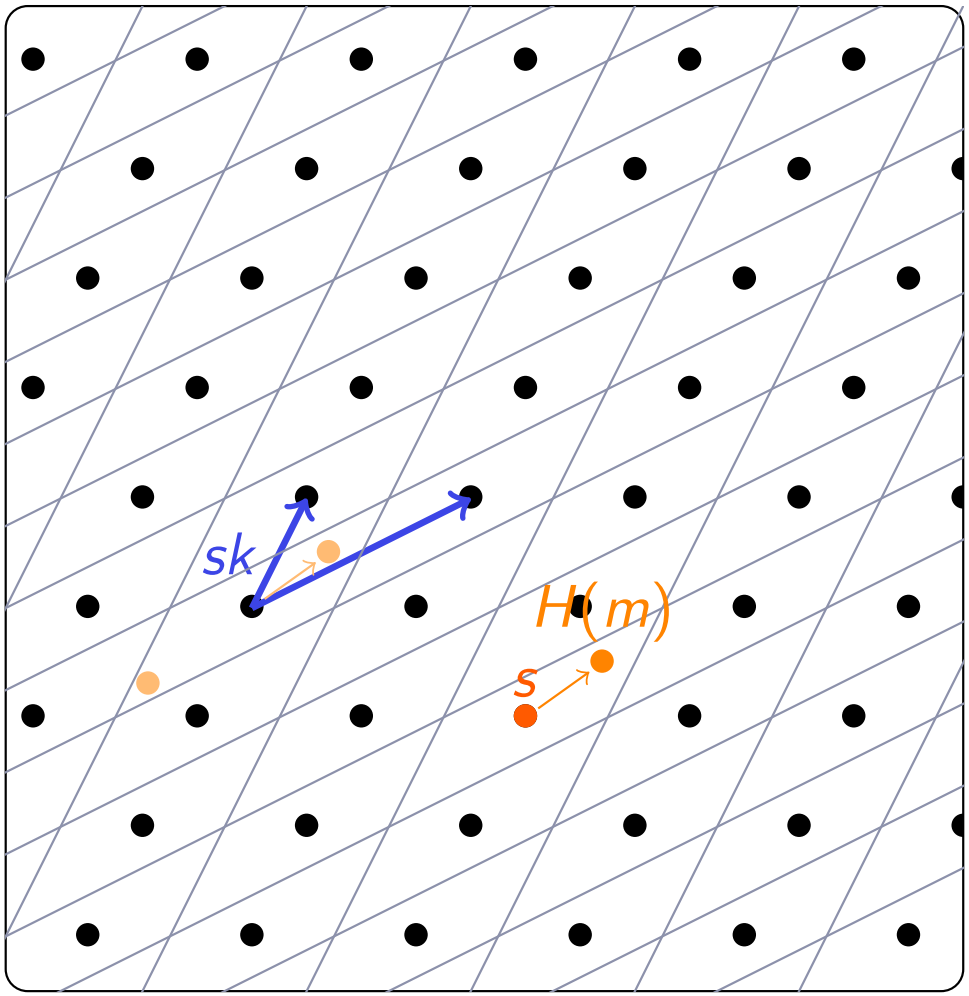


Parallelepiped attack:

- 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]

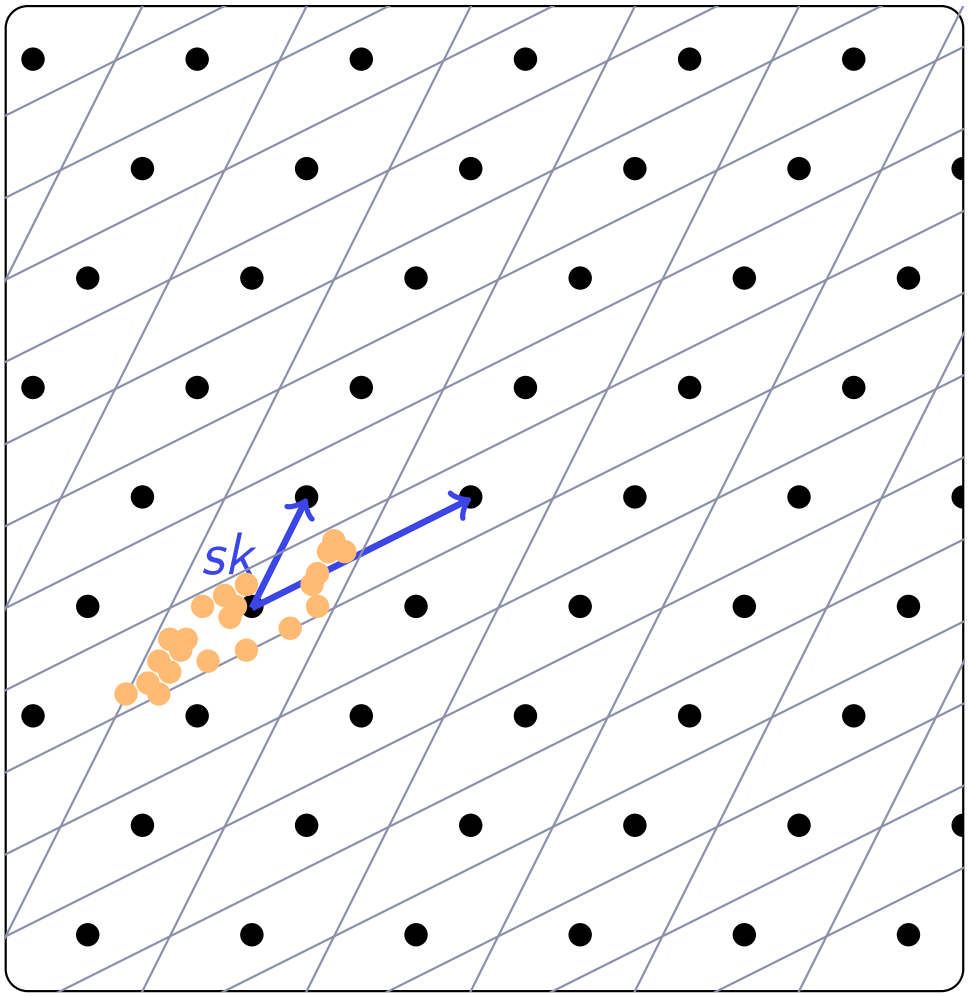


Parallelepiped attack:

- 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]

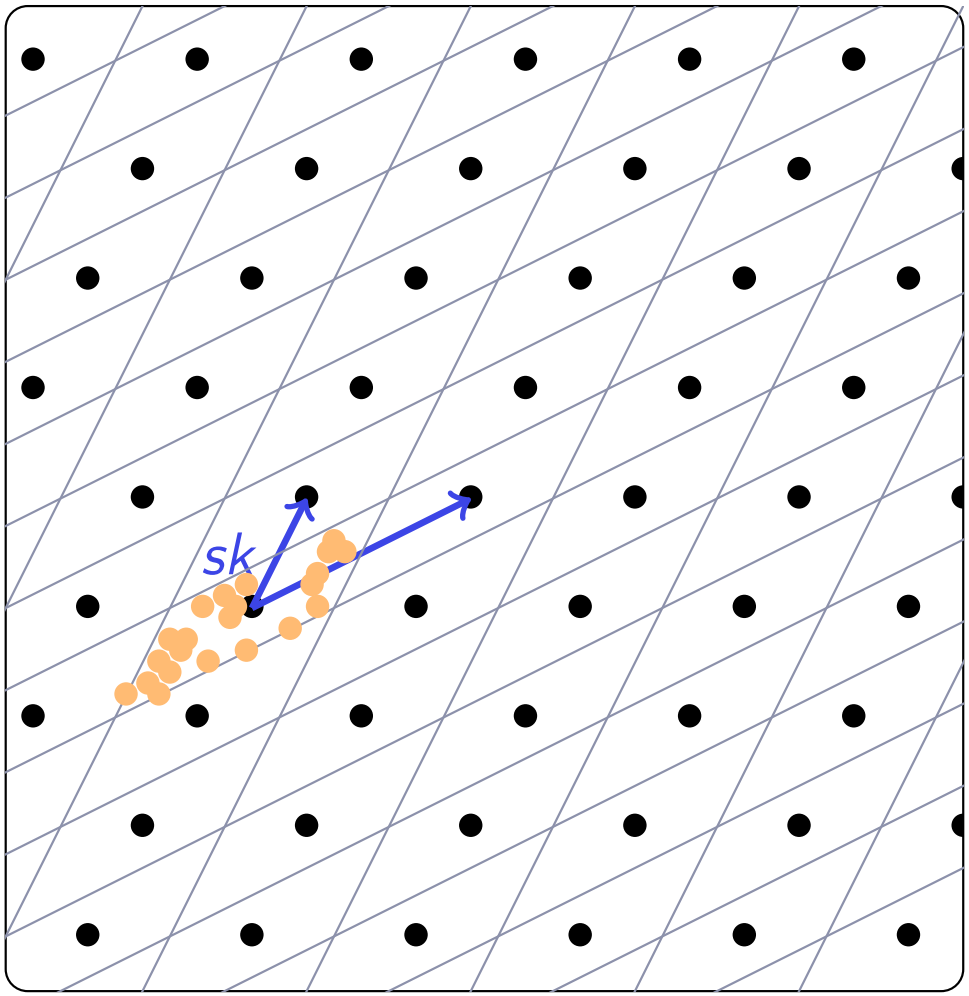


Parallelepiped attack:

- 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]



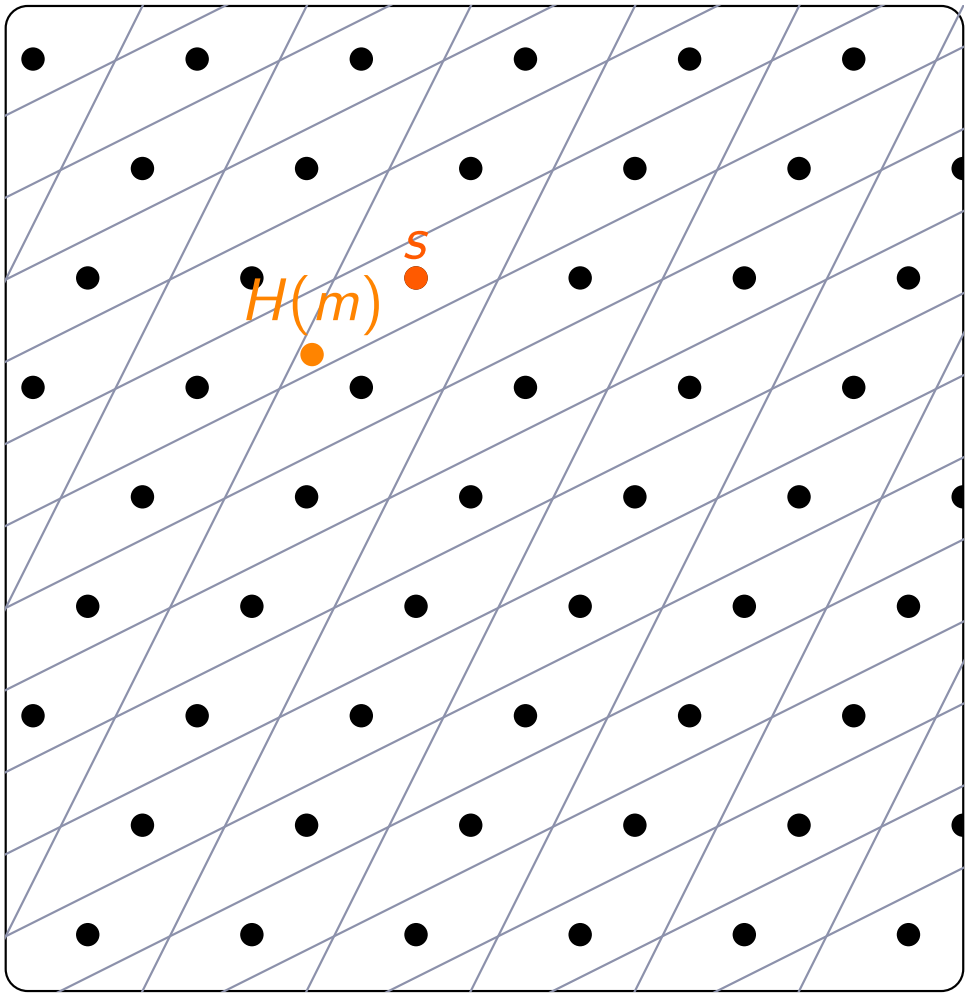
Parallelepiped attack:

- 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

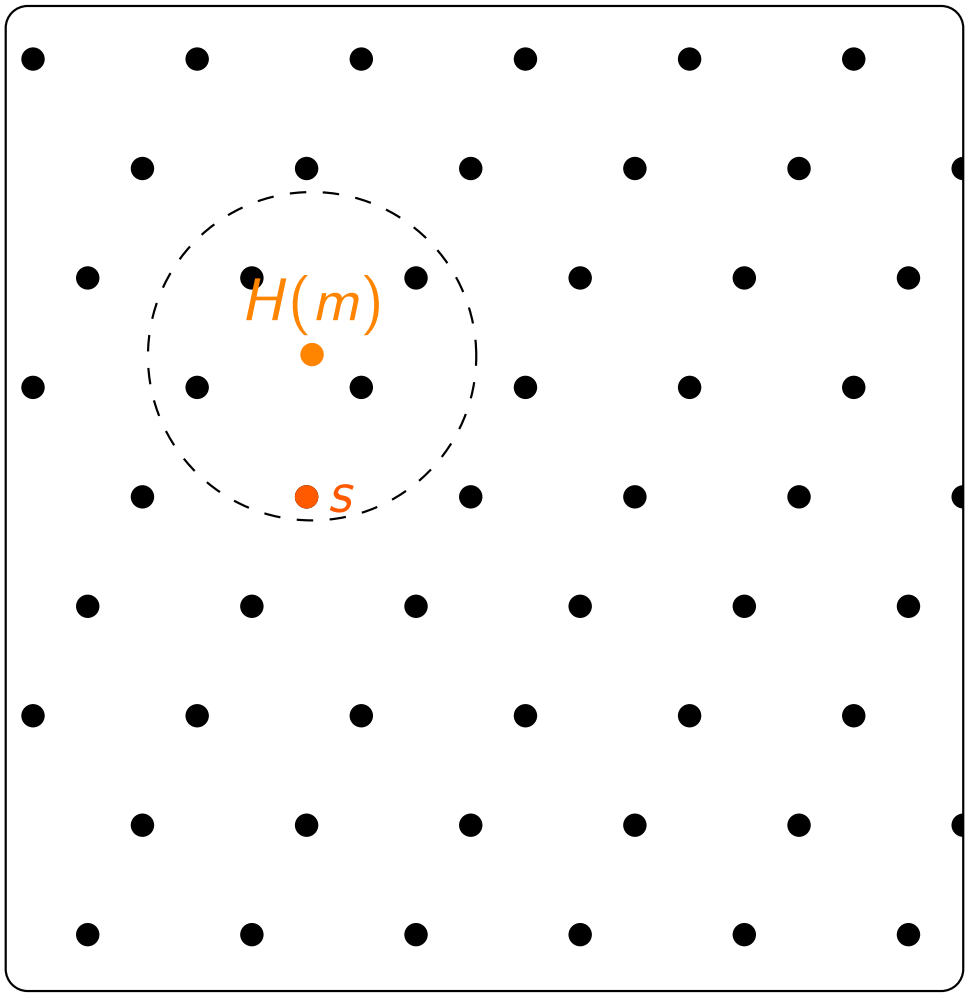
Preventing the attack [GPV08]



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]



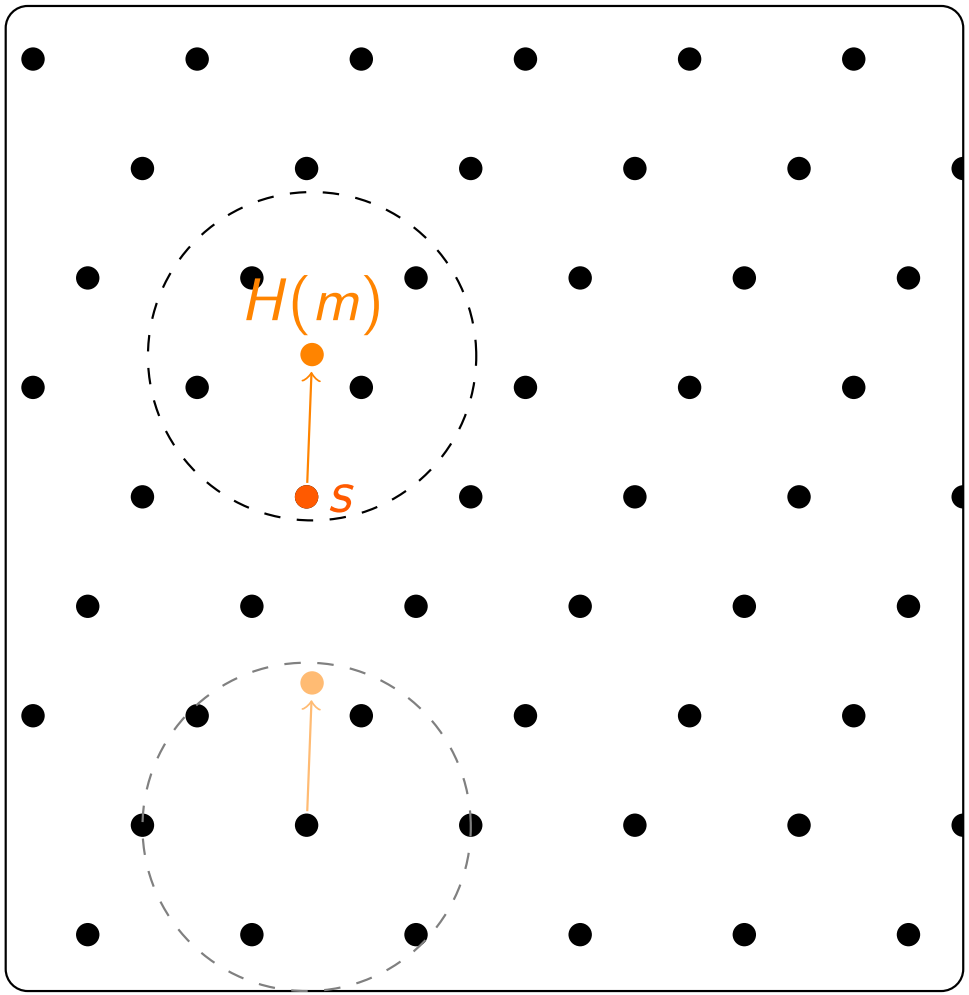
Idea: do not decode deterministically but randomly

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]



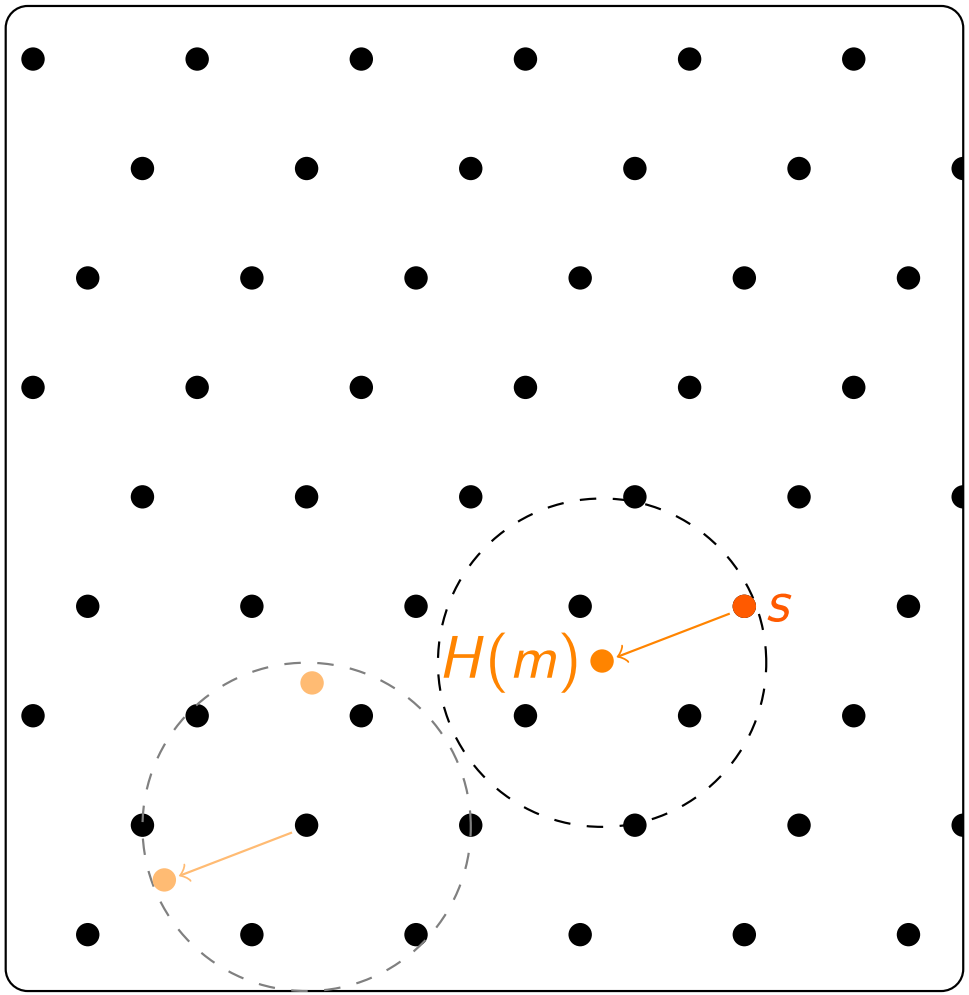
Idea: do not decode deterministically but randomly

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]



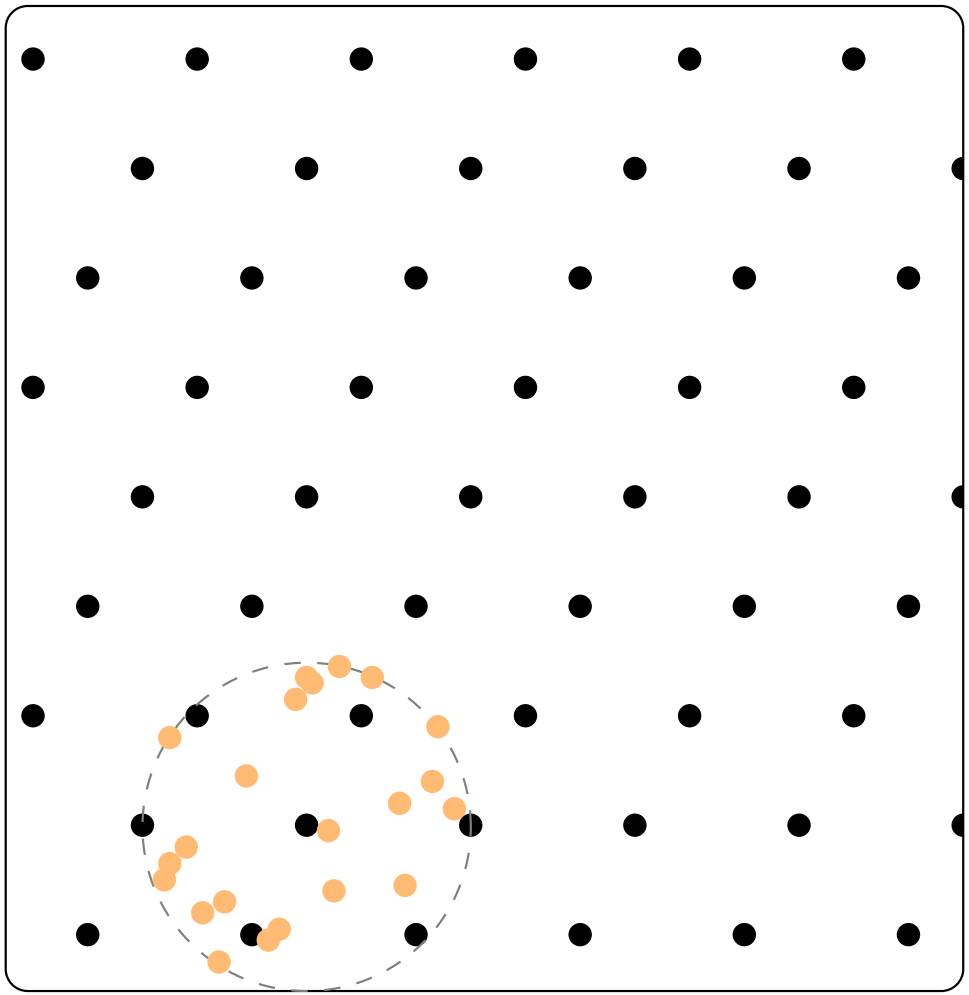
Idea: do not decode deterministically but randomly

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]



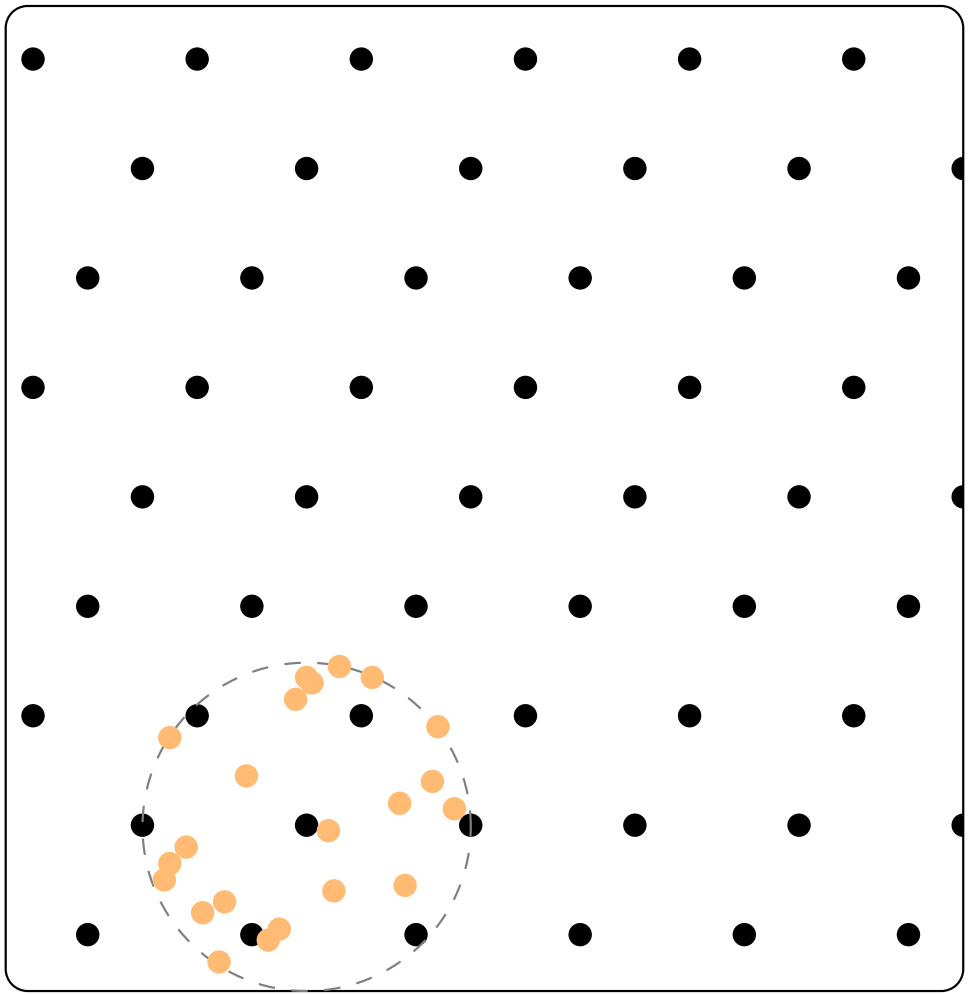
Idea: do not decode deterministically but randomly

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)

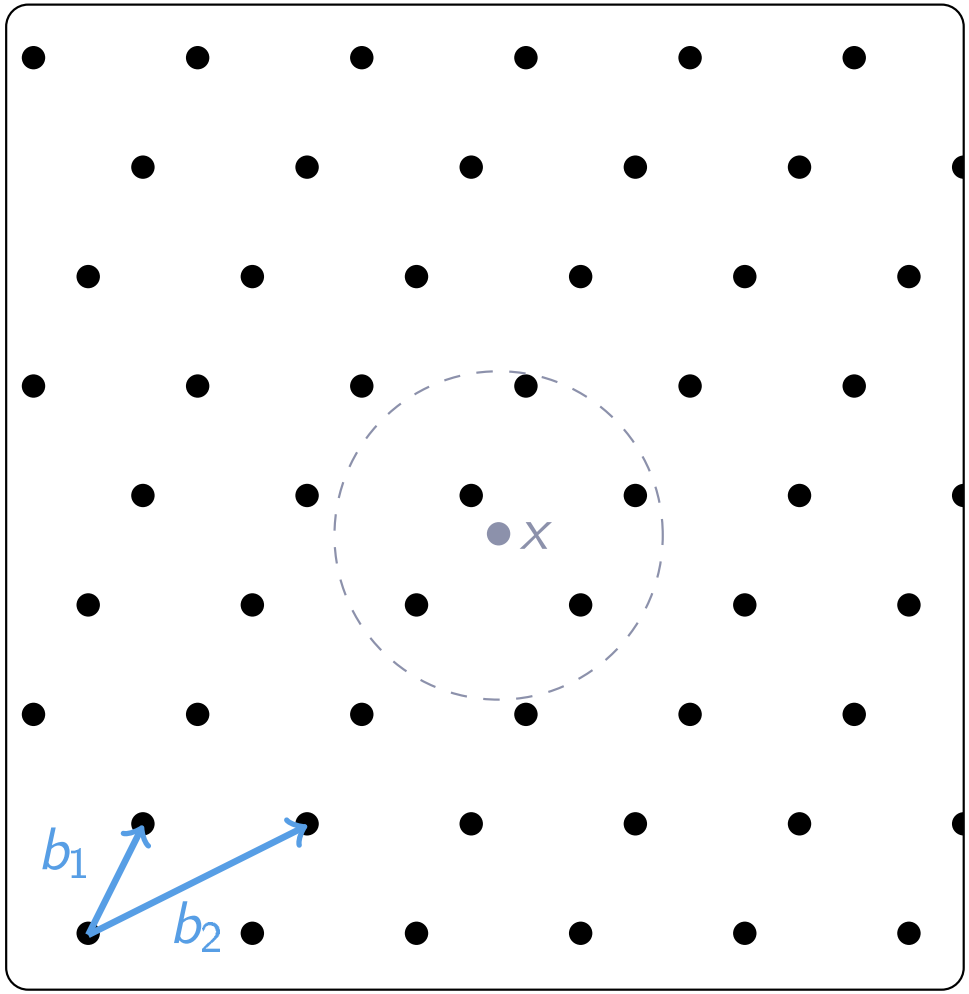
[GPV08] Gentry, Peikert, and Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. STOC.

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

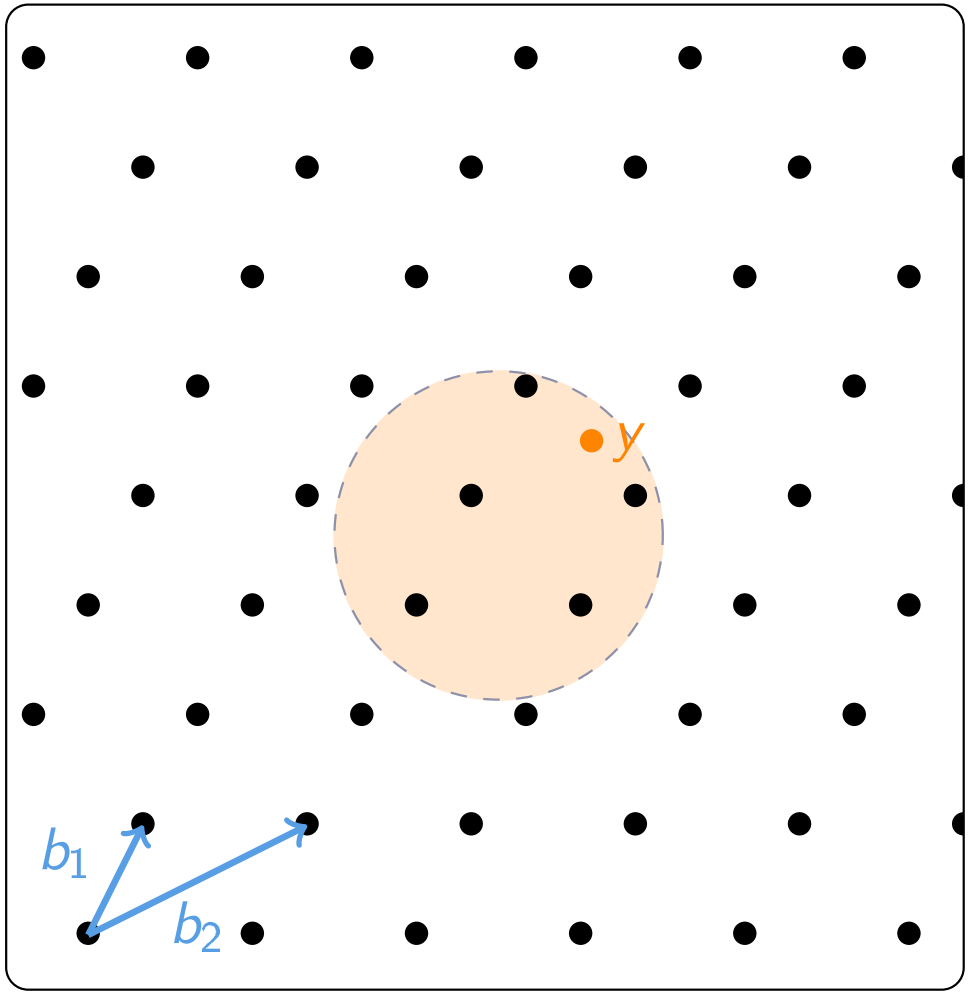
(and a short basis $(b_1, \ldots, b_n)$)

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]

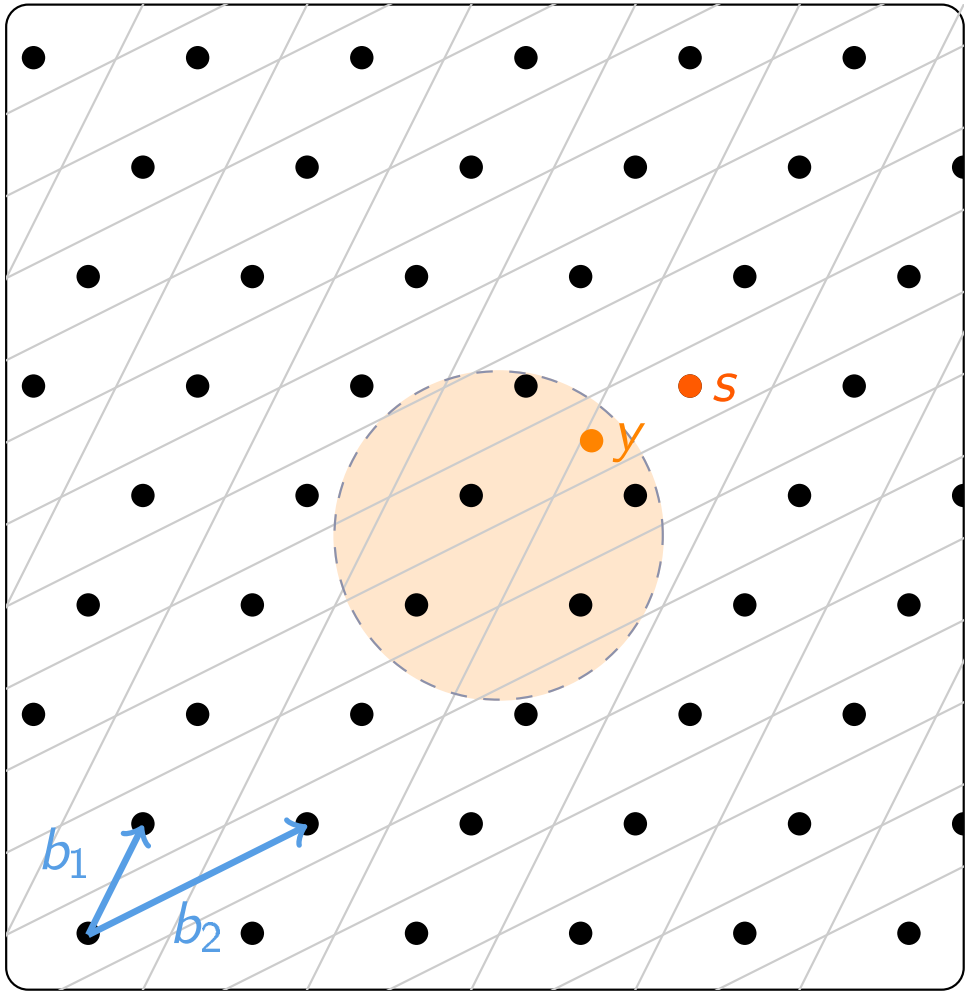
Input: center x, radius r

(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]

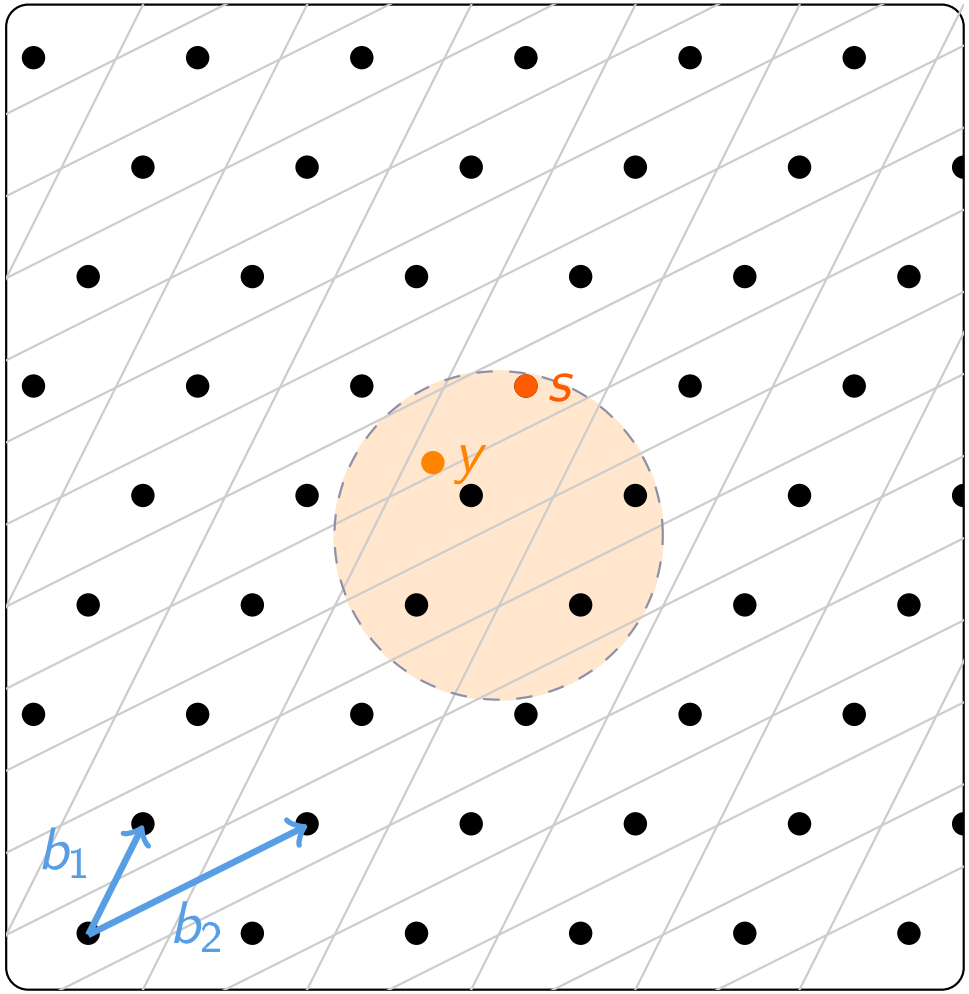
Input: center x, radius r

(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]

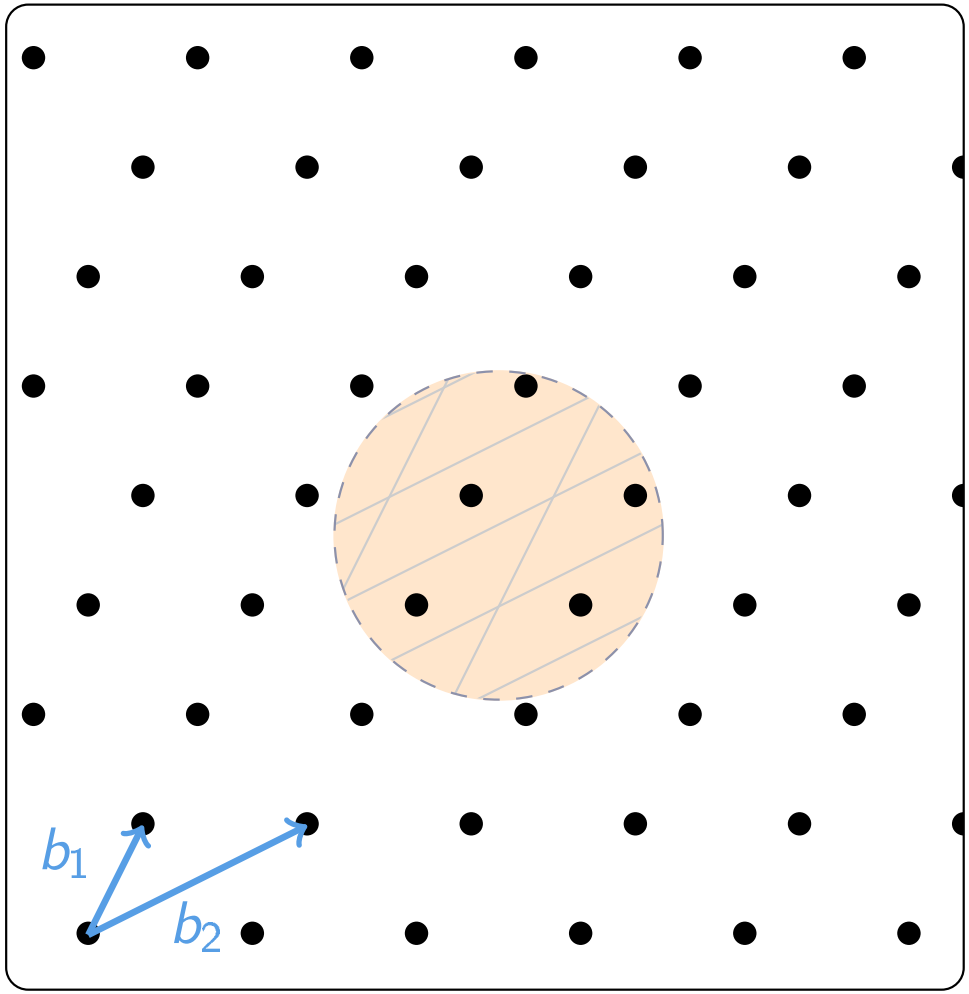
Input: center x, radius r

(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]

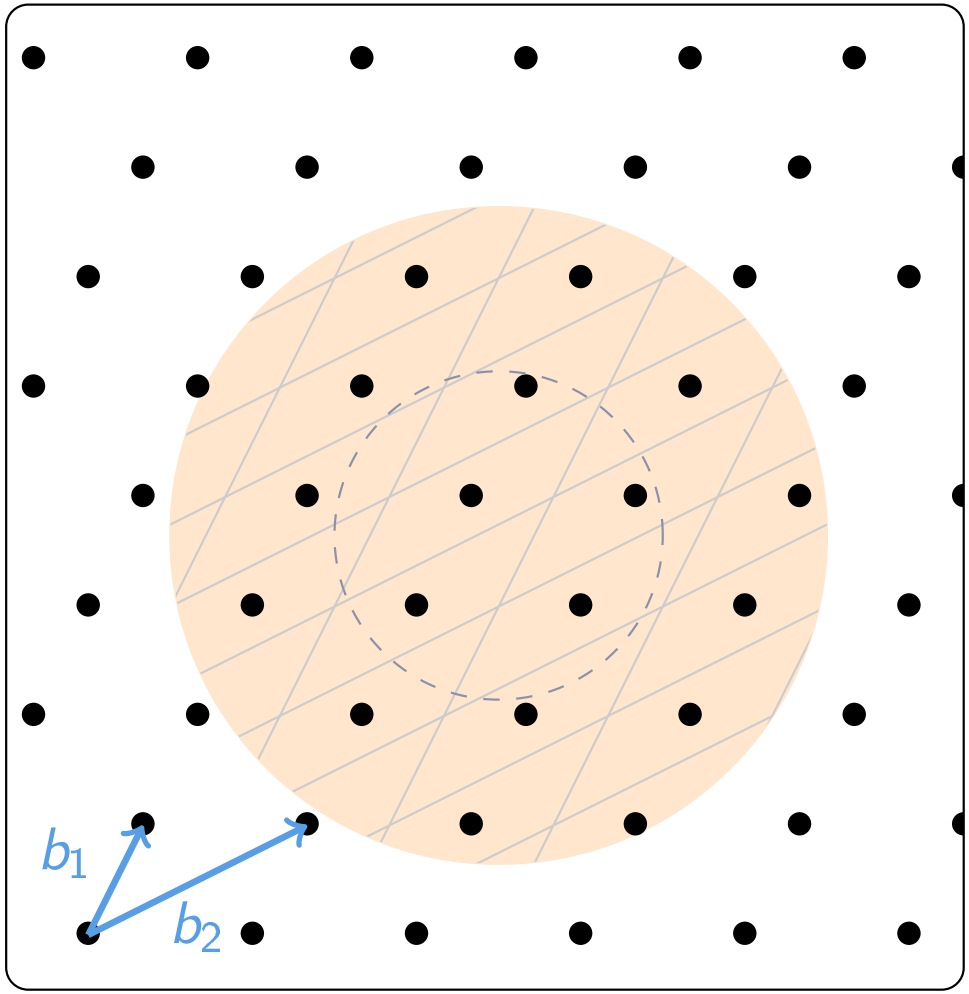
Input: center x, radius r

(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]

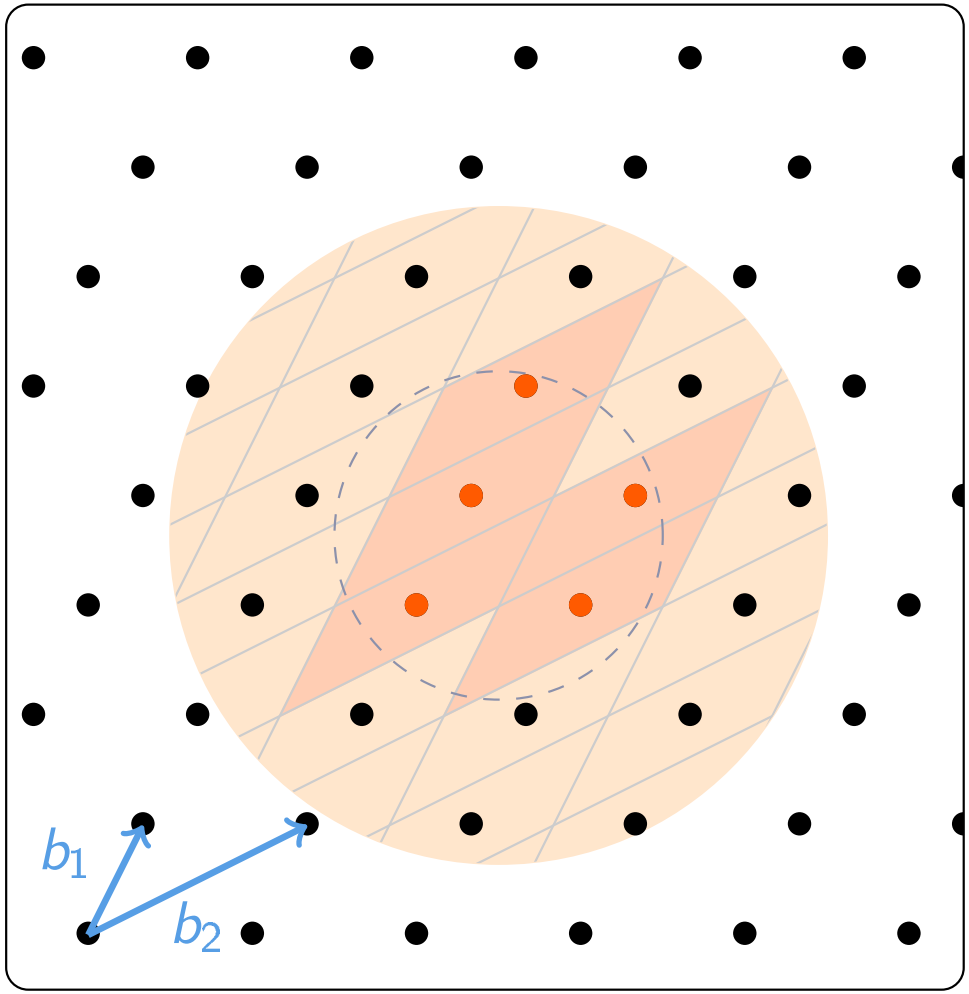
Input: center x, radius r

(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 $(b_1, \ldots, b_n)$)

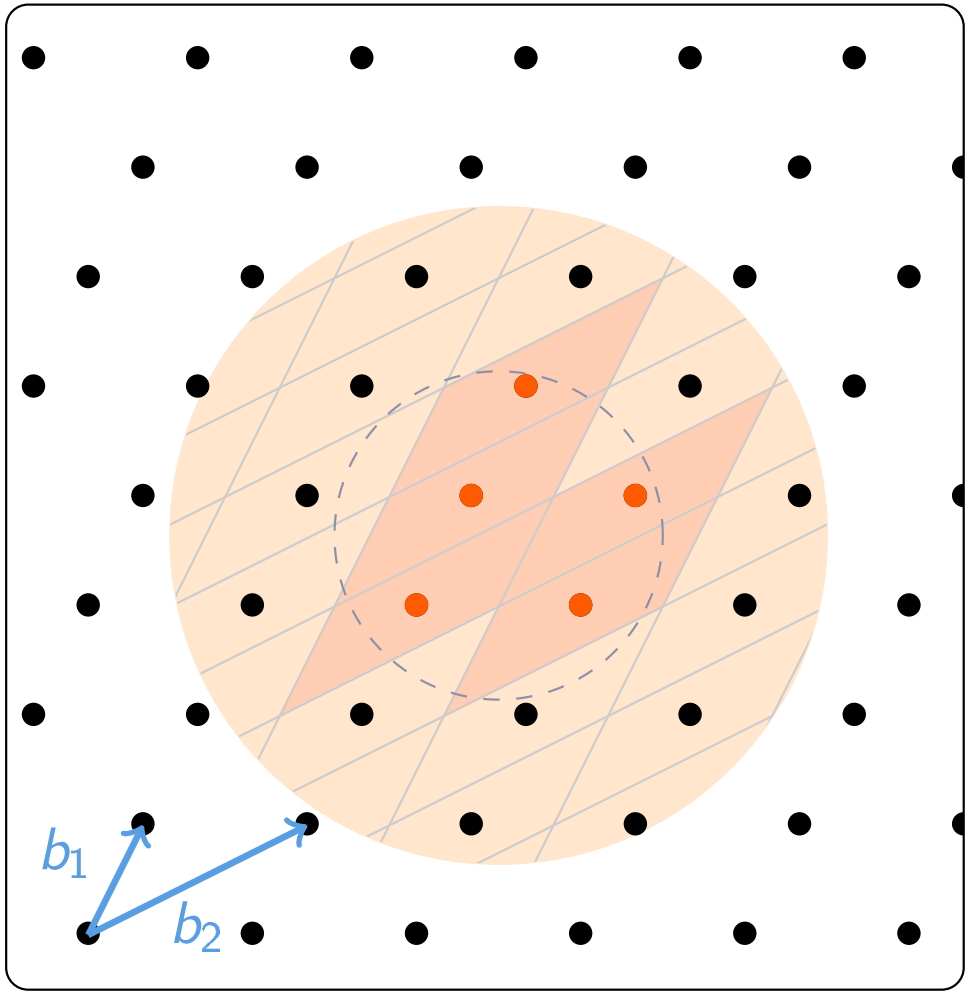
Output: $s \leftarrow \mathcal{U}(\mathcal{L} \cap \mathcal{B}_r(x))$

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.

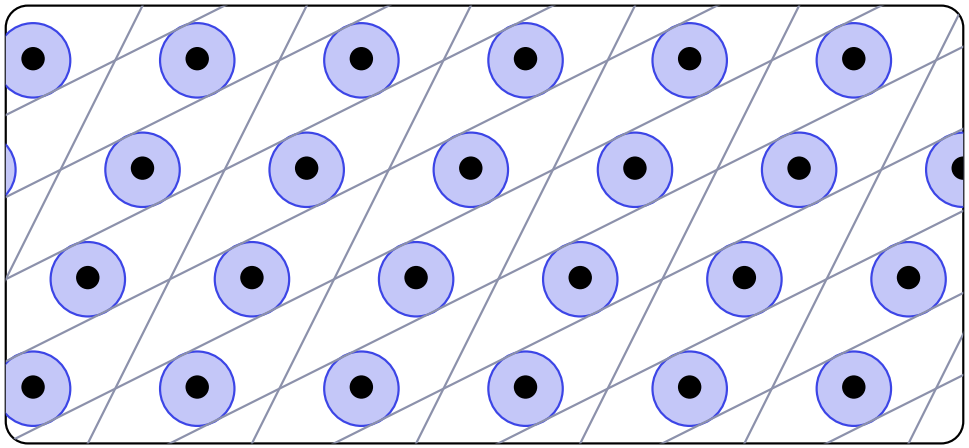
Section's conclusion

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

Constructing public key encryption from lattices

(Regev encryption and variants)



Public key encryption (PKE)

Three algorithms:

- $\mathsf{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathsf{Enc}(m, pk) \rightsquigarrow c$
- $\mathsf{Dec}(c, sk) \rightsquigarrow m'$

Public key encryption (PKE)

Three algorithms:
- $\mathsf{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathsf{Enc}(m, pk) \rightsquigarrow c$
- $\mathsf{Dec}(c, sk) \rightsquigarrow m'$

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

Public key encryption (PKE)

Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Enc}(m, pk) \rightsquigarrow c$
- $\mathrm{Dec}(c, sk) \rightsquigarrow m'$

Alice		Bob
$(pk, sk) \leftarrow \mathrm{KeyGen}()$	$\xrightarrow{pk}$	
		message $m \in \{0, 1\}$
$m' \leftarrow \mathrm{Dec}(c, sk)$	$\xleftarrow{c}$	$c \leftarrow \mathrm{Enc}(m, pk)$
(hopefully $m' = m$)		

Public key encryption (PKE)

Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Enc}(m, pk) \rightsquigarrow c$
- $\mathrm{Dec}(c, sk) \rightsquigarrow m'$

Alice		Bob
$(pk, sk) \leftarrow \mathrm{KeyGen}()$	$\xrightarrow{pk}$	
		message $m \in \{0, 1\}$
$m' \leftarrow \mathrm{Dec}(c, sk)$	$\xleftarrow{c}$	$c \leftarrow \mathrm{Enc}(m, pk)$
(hopefully $m' = m$)		

Correctness: $\mathrm{Dec}(c, sk) = m$ (when $c \leftarrow \mathrm{Enc}(m, pk)$ and $(pk, sk) \leftarrow \mathrm{KeyGen}$)

Public key encryption (PKE)

Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Enc}(m, pk) \rightsquigarrow c$
- $\mathrm{Dec}(c, sk) \rightsquigarrow m'$

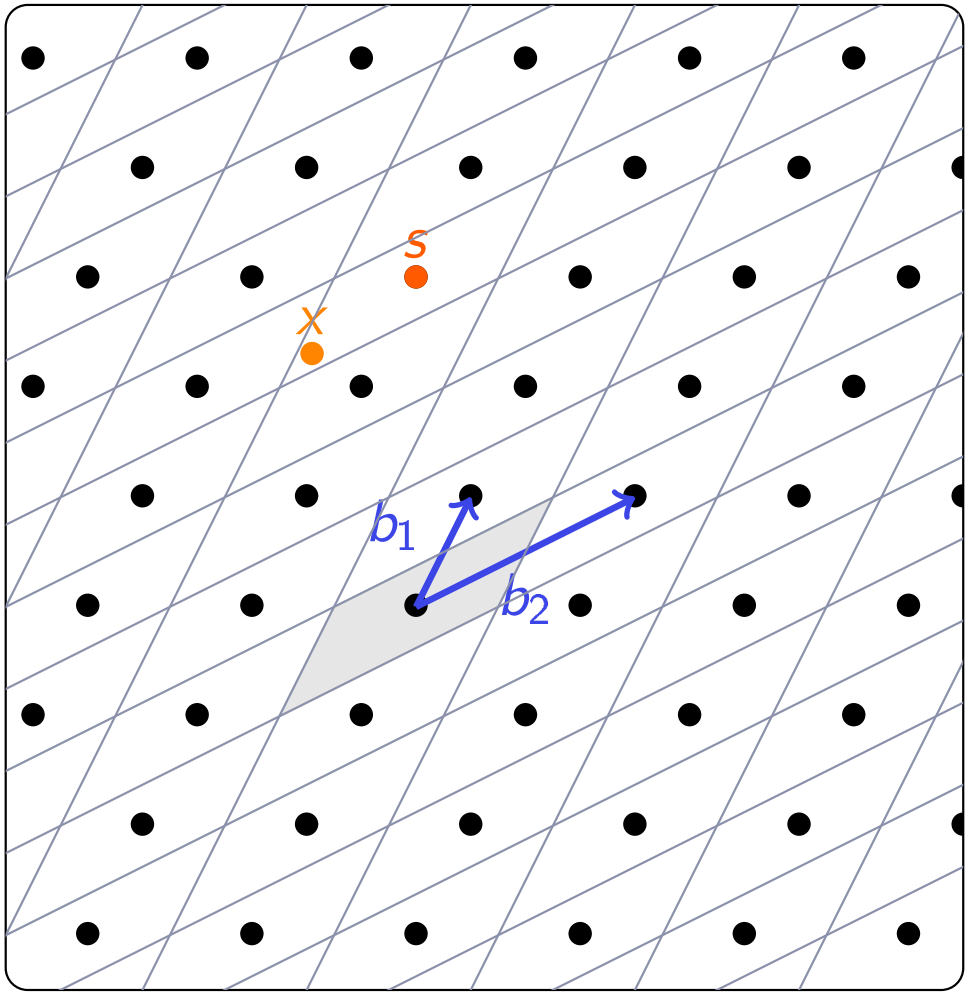
Alice		Bob
$(pk, sk) \leftarrow \mathrm{KeyGen}()$	$\xrightarrow{pk}$	
		message $m \in \{0,1\}$
$m' \leftarrow \mathrm{Dec}(c, sk)$	$\xleftarrow{c}$	$c \leftarrow \mathrm{Enc}(m, pk)$
(hopefully $m' = m$)		

Correctness: $\mathrm{Dec}(c, sk) = m$ (when $c \leftarrow \mathrm{Enc}(m, pk)$ and $(pk, sk) \leftarrow \mathrm{KeyGen}$)

Security: an attacker cannot distinguish $\mathrm{Enc}(0, pk)$ from $\mathrm{Enc}(1, pk)$

$$\left(\text{i.e., } \Pr_{\substack{m \xleftarrow{\$} \{0,1\} \\ c \leftarrow \mathrm{Enc}(m, pk)}} \left(\mathcal{A}(c, pk) = m\right) \approx 1/2\right)$$

Remember Babai

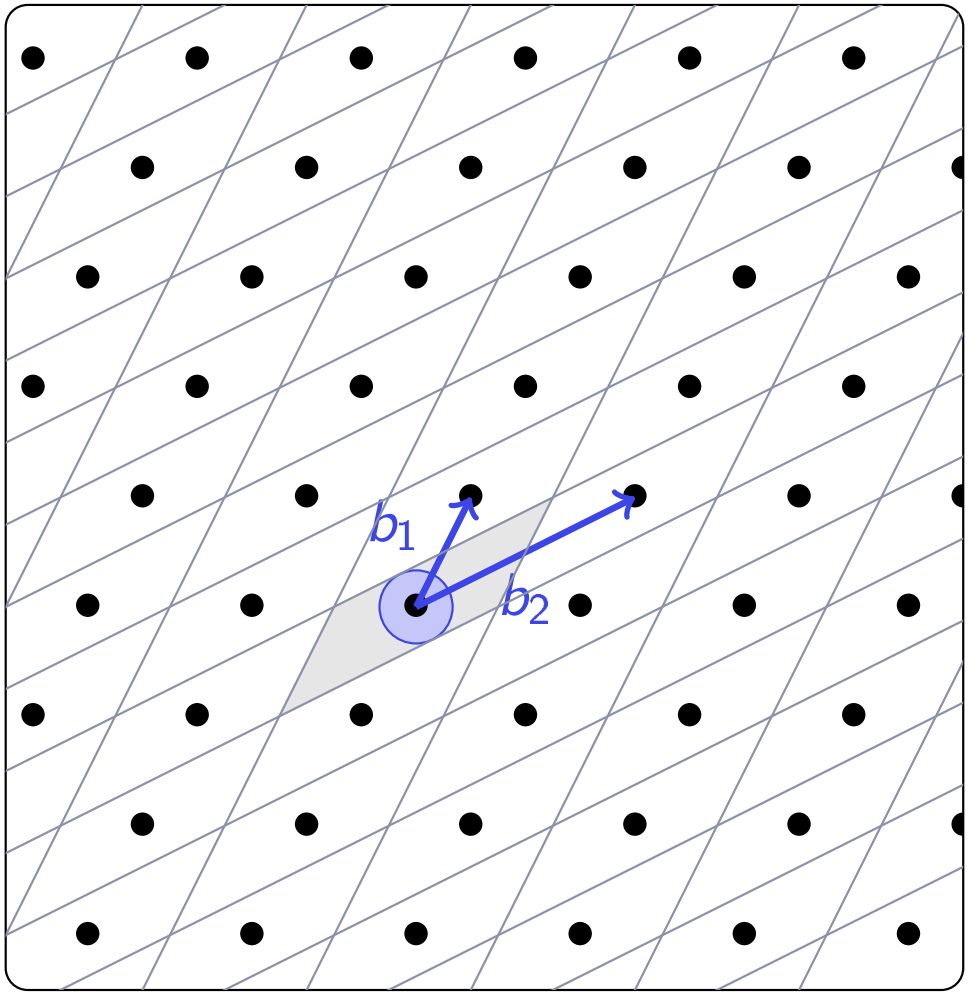


$\texttt{Babai}_B(x) = s \quad (\in \mathcal{L}, \text{ close to } x)$

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

(fundamental parallelepiped)

Remember Babai



$\texttt{Babai}_B(x) = s$ ($\in \mathcal{L}$, close to x)

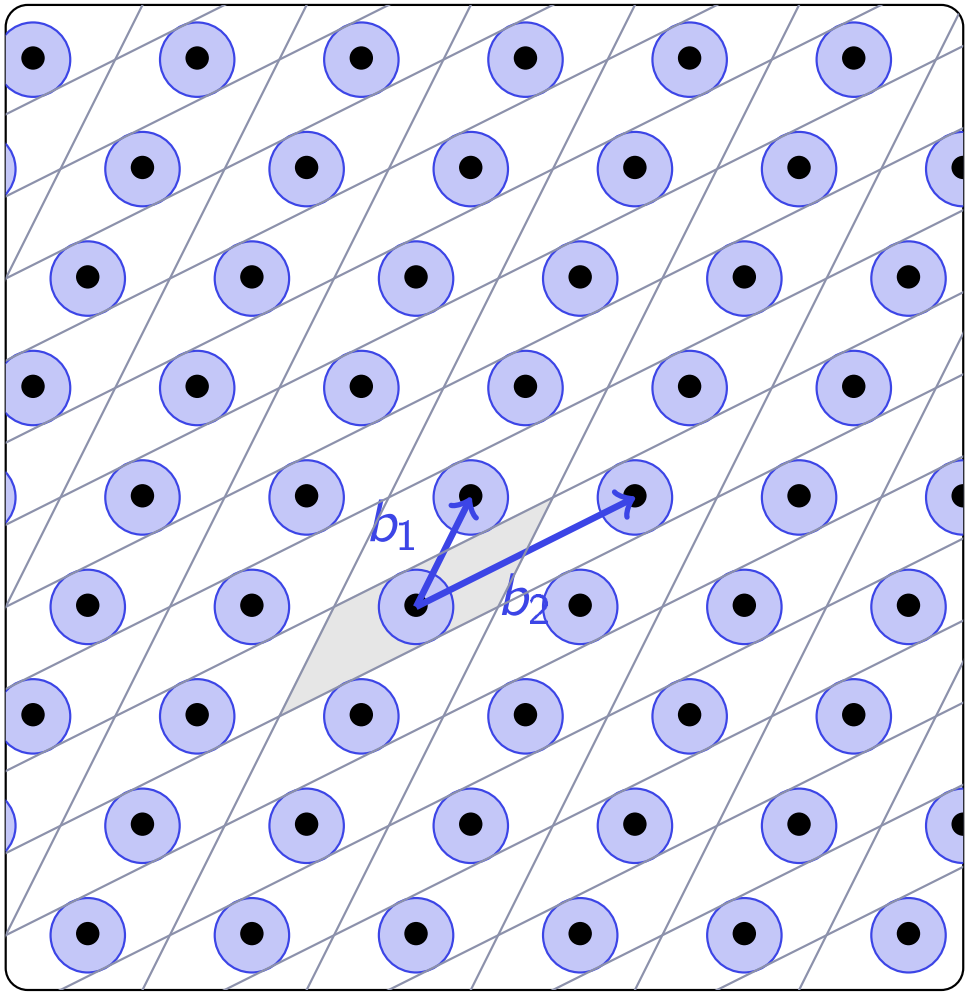
$$= \left\{ x_1 b_1 + x_2 b_2 \;\middle|\; |x_i| \leq \tfrac{1}{2} \right\}$$

(fundamental parallelepiped)

○ = largest circle ⊆

(radius of ○: decoding radius)

Remember Babai



$\texttt{Babai}_B(x) = s$ ($\in \mathcal{L}$, close to x)

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

(fundamental parallelepiped)

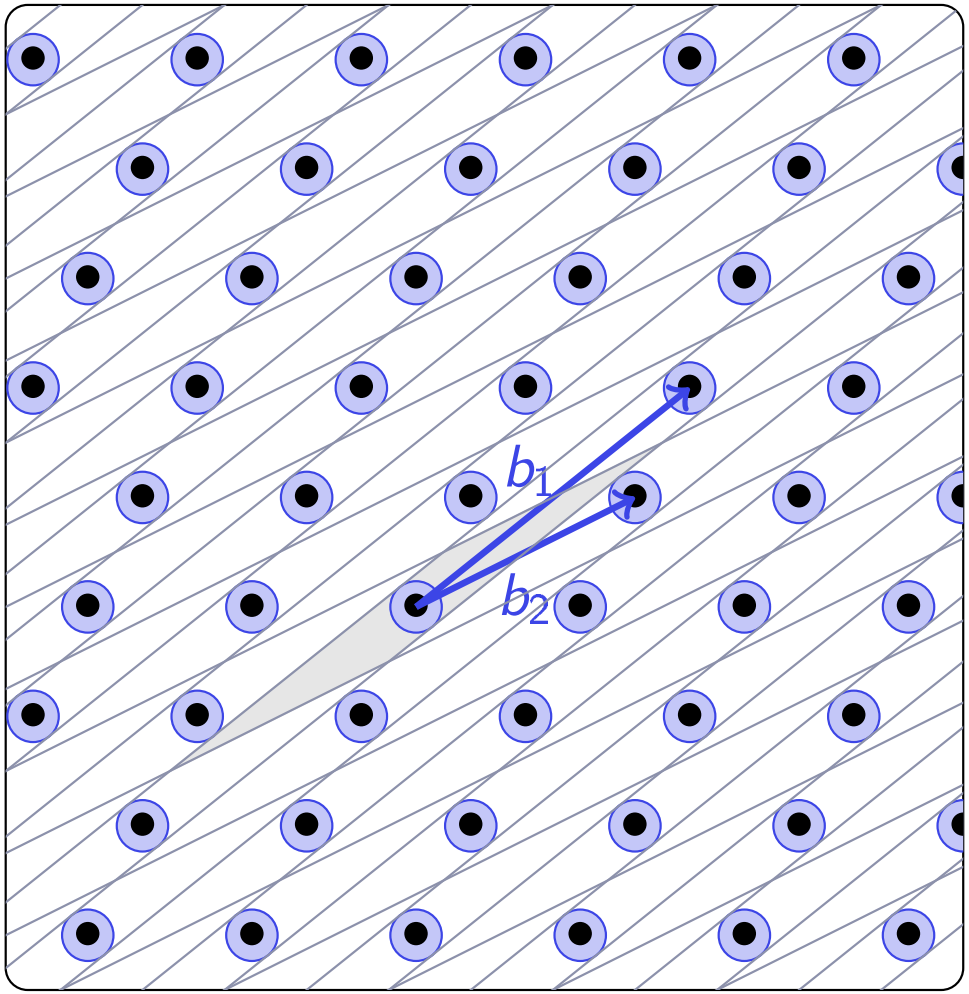
● = largest circle $\subseteq$

(radius of ●: decoding radius)

Lemma: $\forall s \in \mathcal{L}$ and $\forall e \in$ ●

$$\texttt{Babai}_B(s + e) = s$$

Remember Babai



$\texttt{Babai}_B(x) = s$ ($\in \mathcal{L}$, close to x)

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

(fundamental parallelepiped)

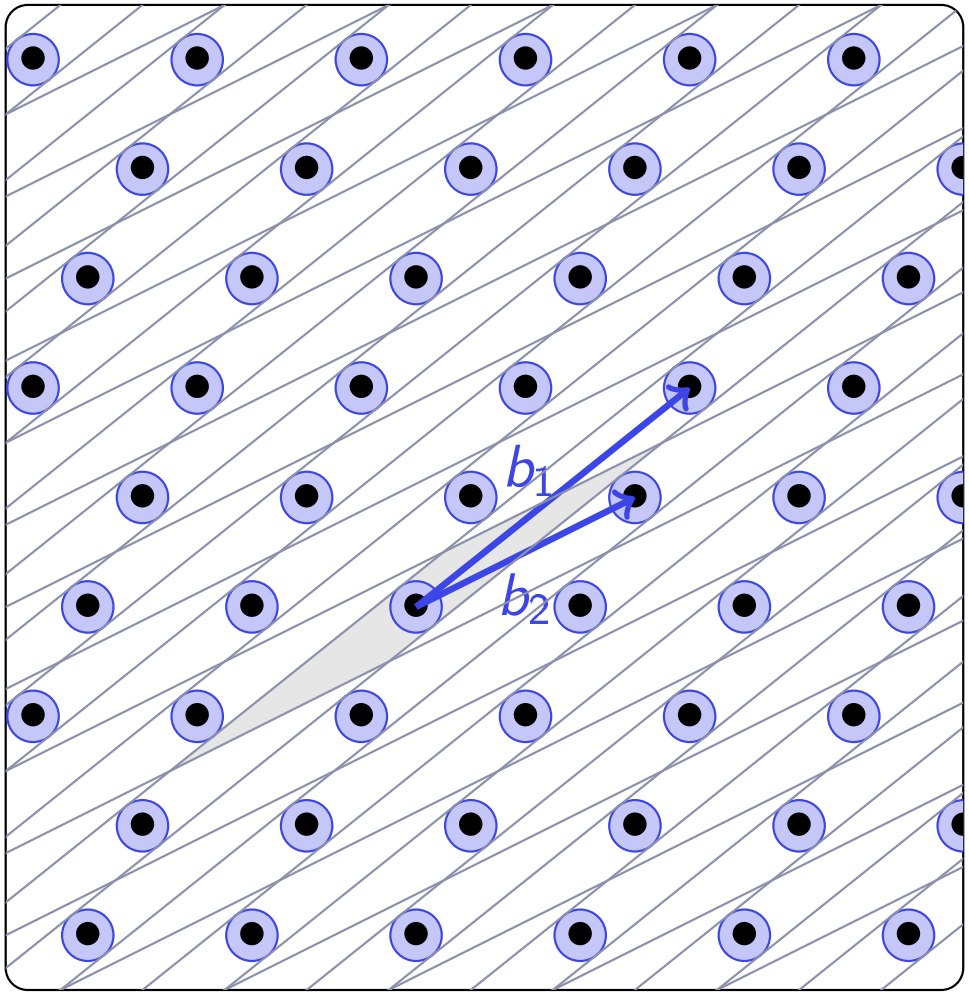
● = largest circle $\subseteq$

(radius of ●: decoding radius)

Lemma: $\forall s \in \mathcal{L}$ and $\forall e \in$ ●

$$\texttt{Babai}_B(s + e) = s$$

Remember Babai



$\texttt{Babai}_B(x) = s \quad (\in \mathcal{L}$, close to $x)$

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

(fundamental parallelepiped)

● = largest circle $\subseteq$

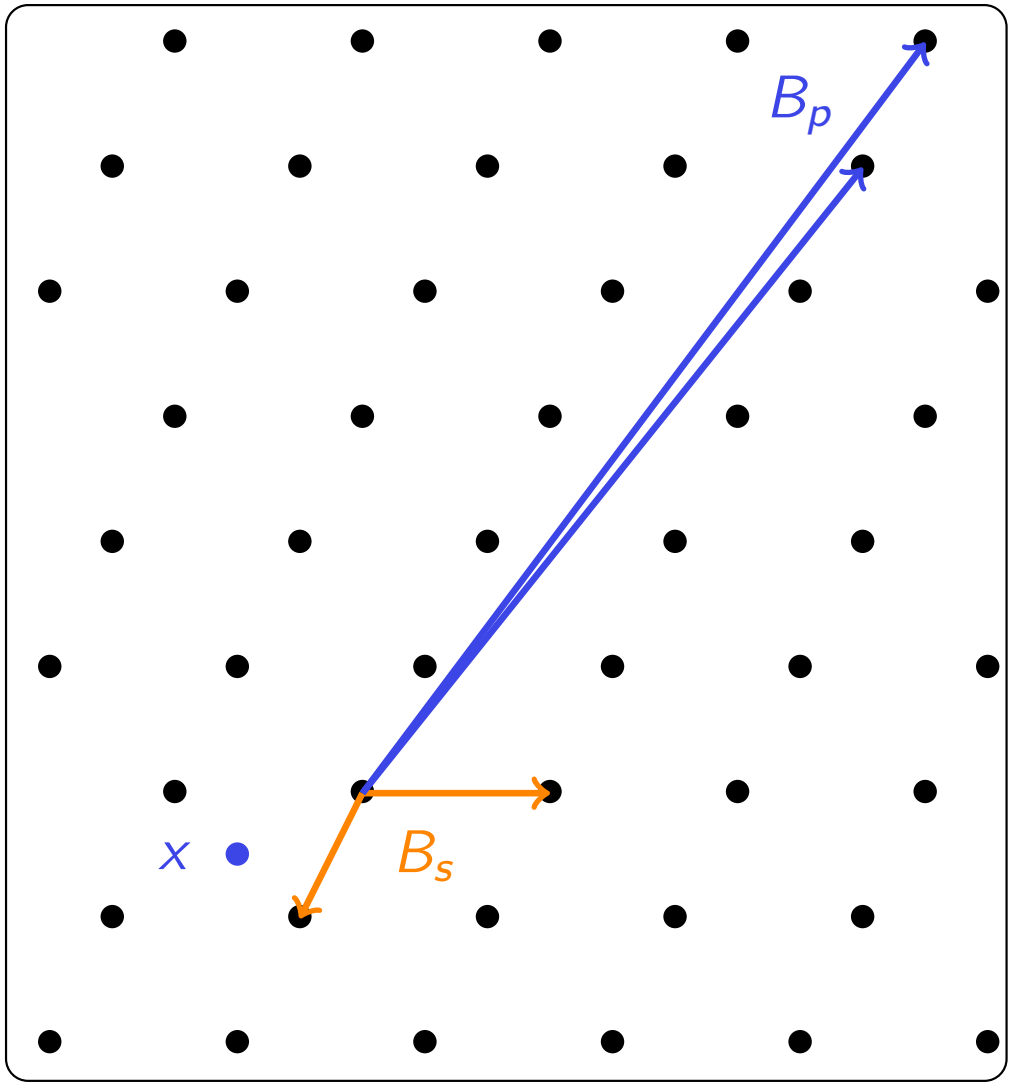
(radius of ●: decoding radius)

Lemma: $\forall s \in \mathcal{L}$ and $\forall e \in$ ●

$$\texttt{Babai}_B(s + e) = s$$

Smaller basis $\Leftrightarrow$ larger ●

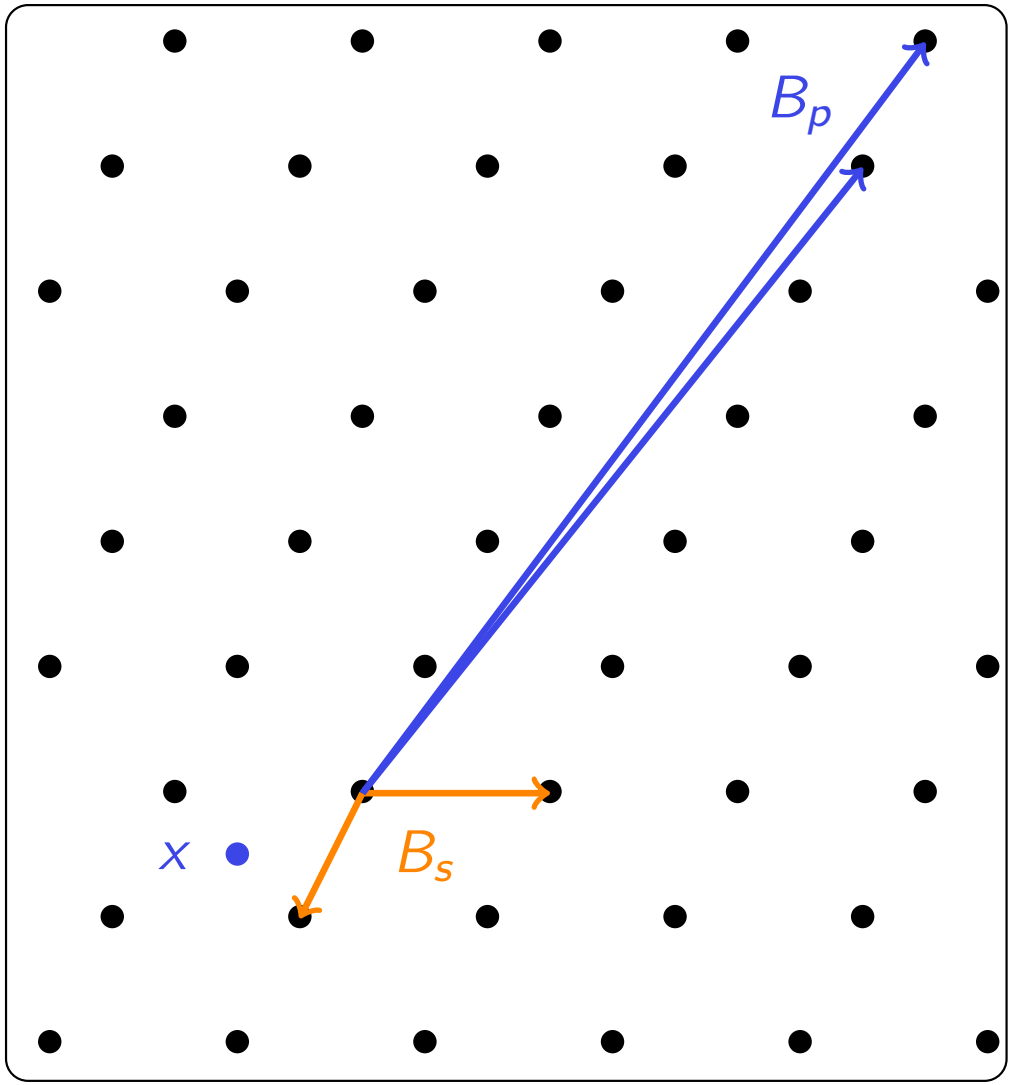
Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



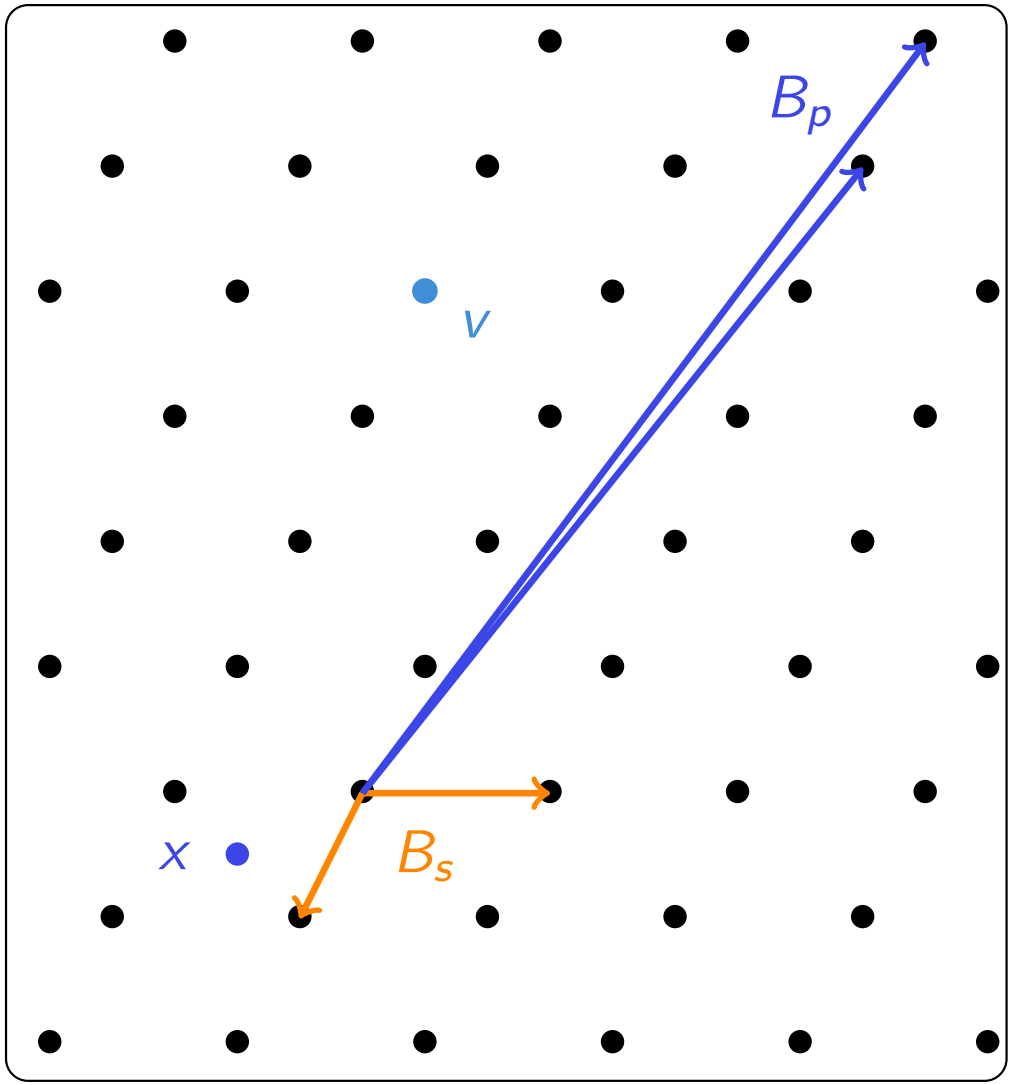
$\mathrm{pk} = (B_p, x)$

$\mathrm{sk} = B_s$

message: $m \in \{0, 1\}$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

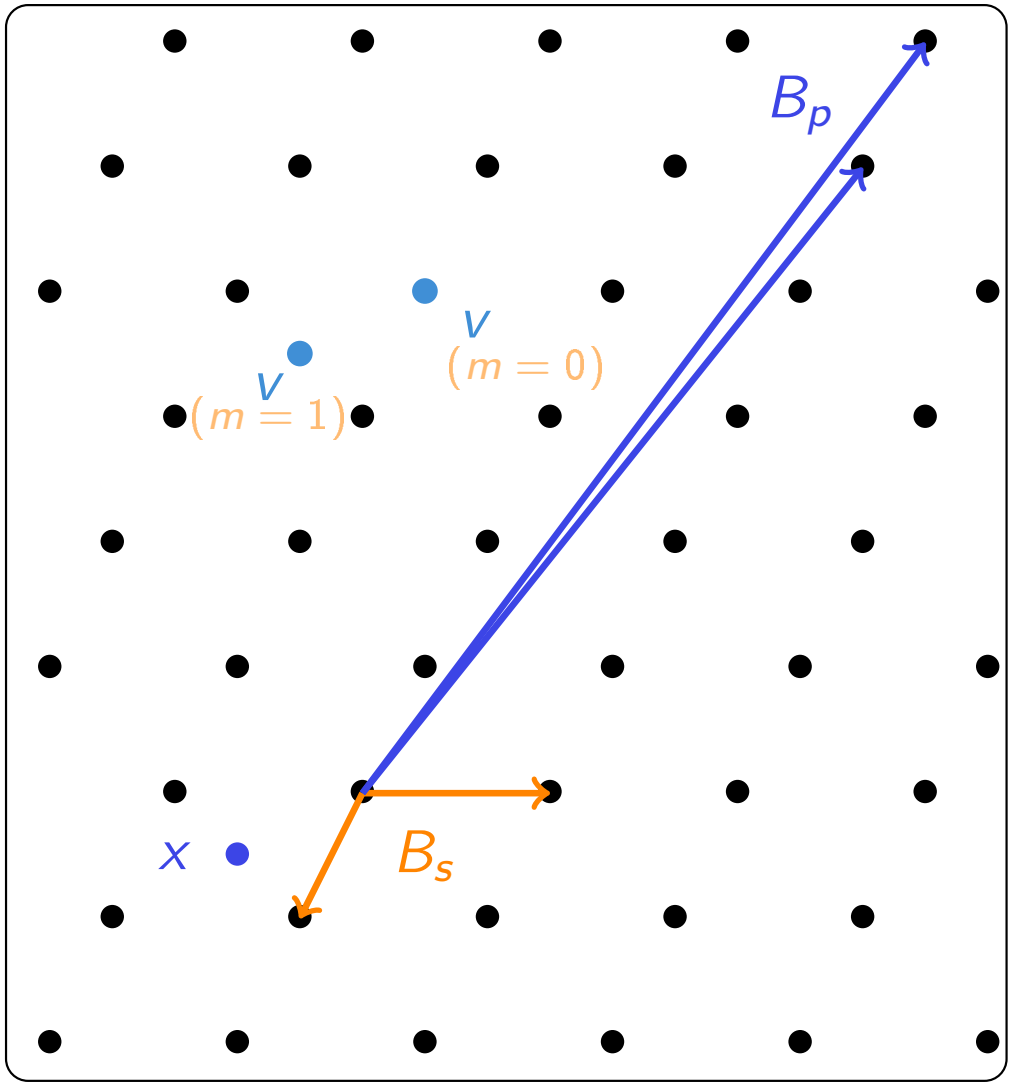
message: $m \in \{0, 1\}$

$\mathrm{Enc}(m, \mathrm{pk})$:

- Sample random $v \in L$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

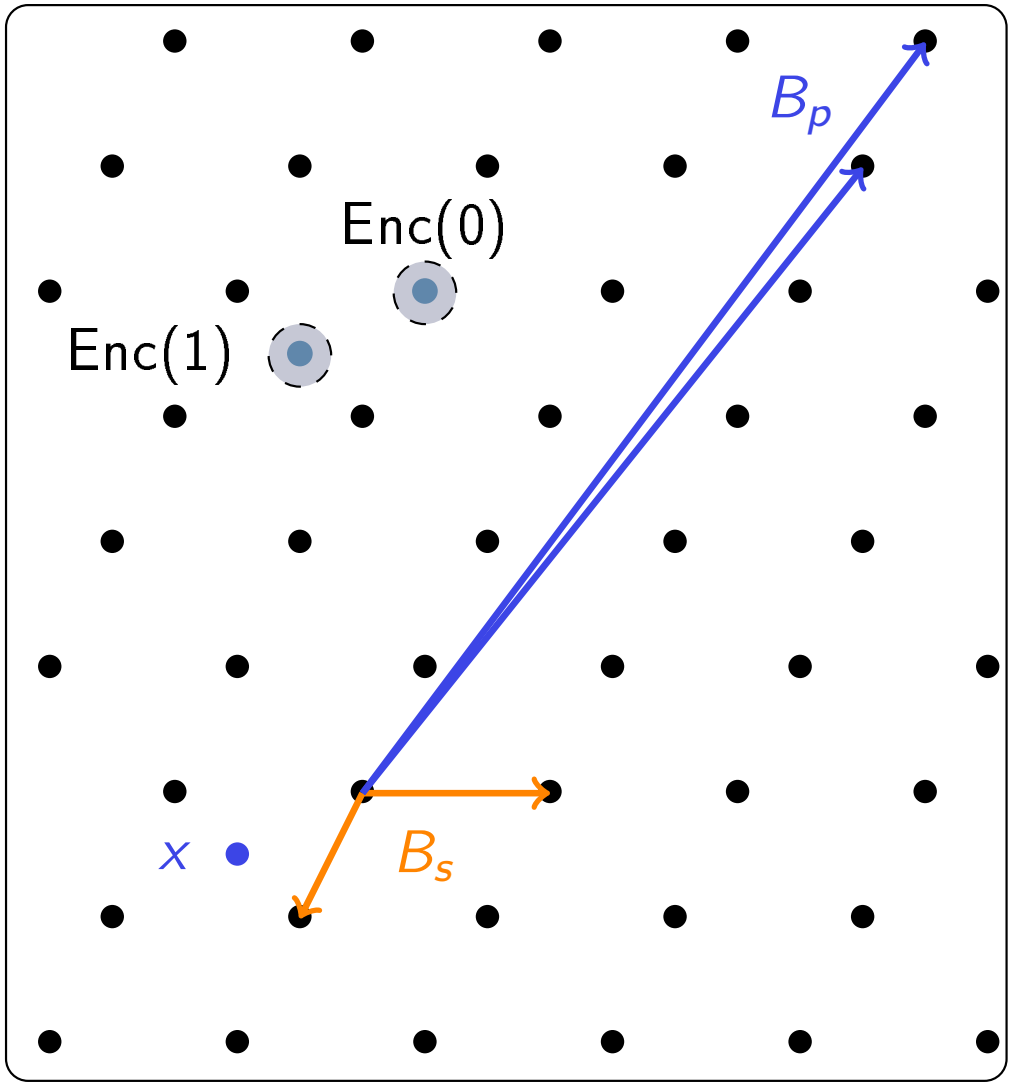
message: $m \in \{0, 1\}$

$\mathrm{Enc}(m, \mathrm{pk})$:

- Sample random $v \in L$
- if $m = 1$: $v \leftarrow v + x$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

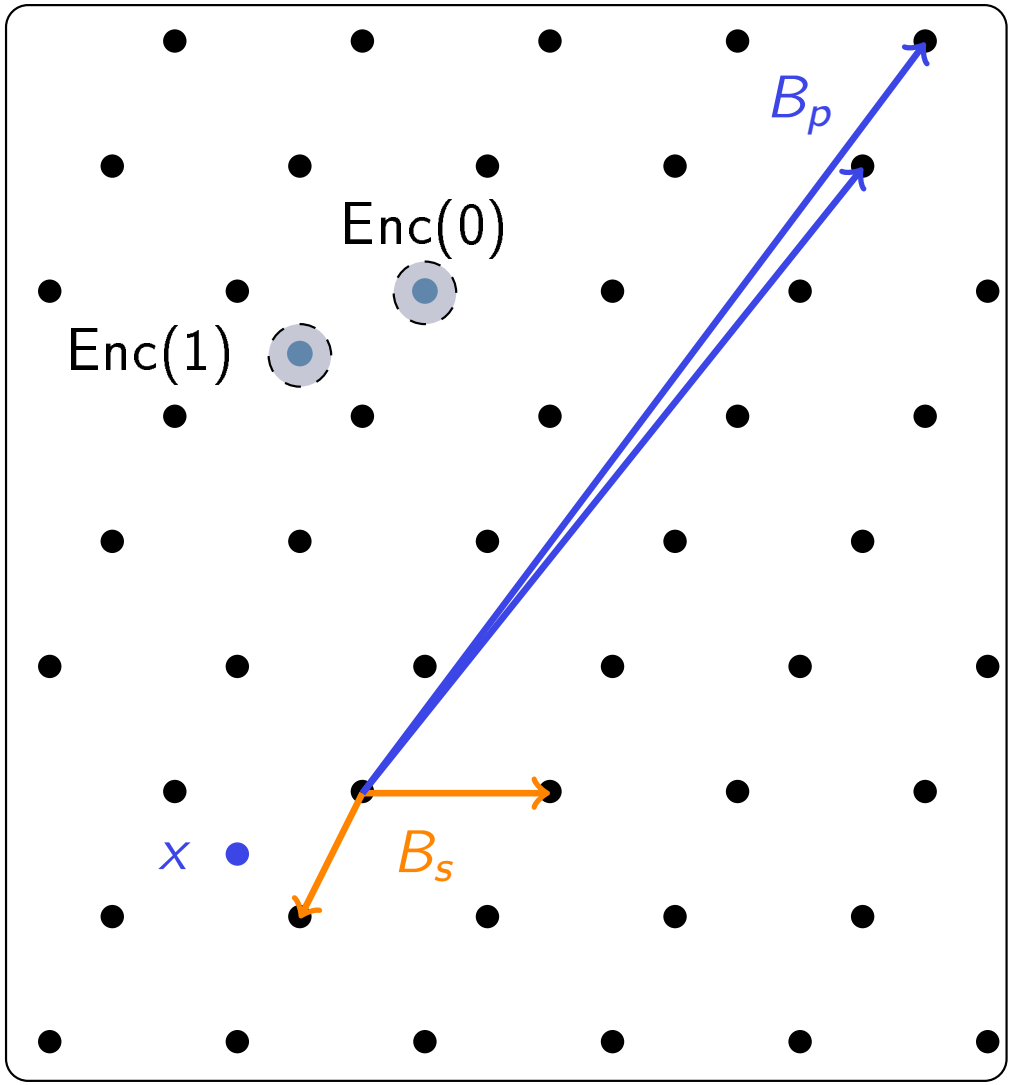
message: $m \in \{0, 1\}$

$\mathrm{Enc}(m, \mathrm{pk})$:

- Sample random $v \in L$
- if $m = 1$: $v \leftarrow v + x$
- Sample small $e \in \mathbb{R}^n$
- return $c = v + e$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

message: $m \in \{0, 1\}$

$\mathrm{Enc}(m, \mathrm{pk})$:

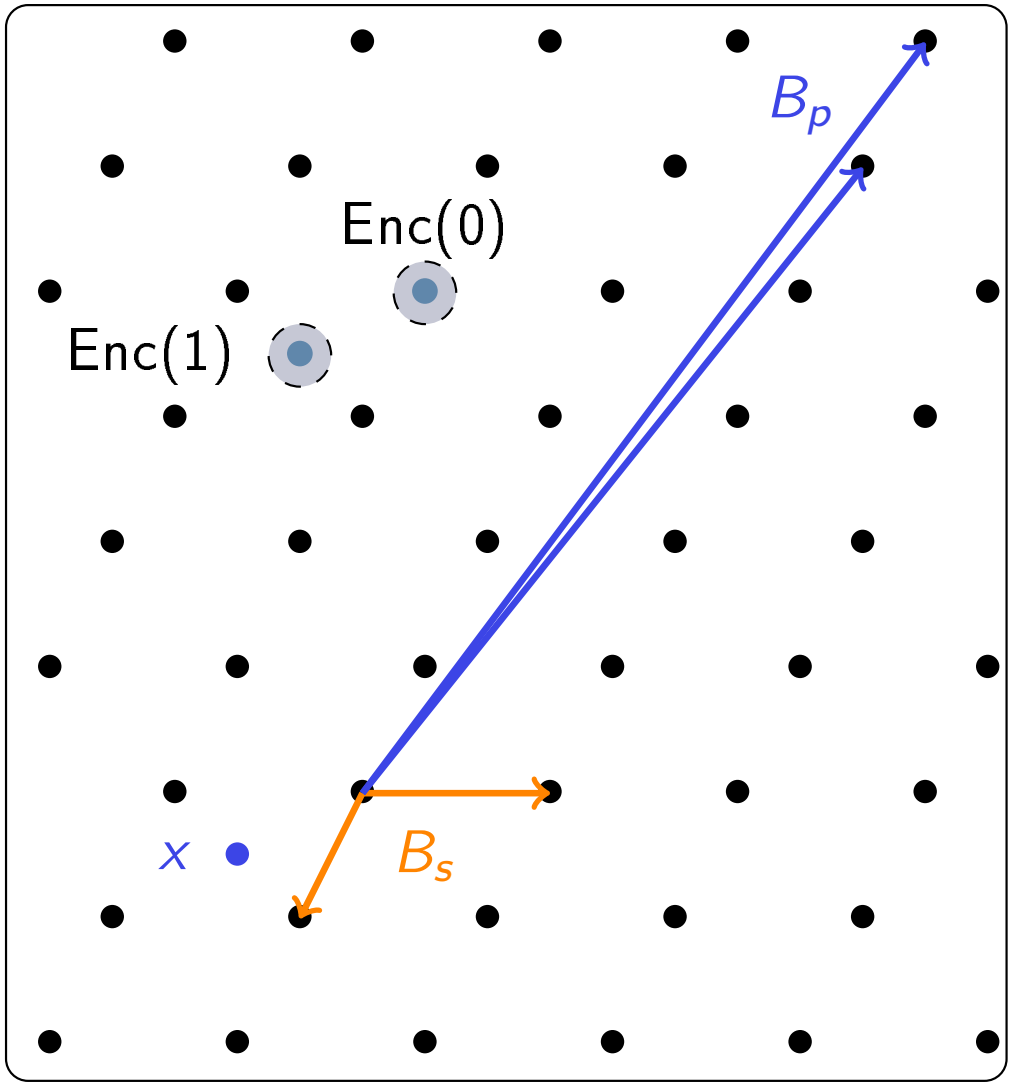
- Sample random $v \in L$
- if $m = 1$: $v \leftarrow v + x$
- Sample small $e \in \mathbb{R}^n$
- return $c = v + e$

$\mathrm{Dec}(c, \mathrm{sk})$:

- $s \leftarrow \mathrm{Babai}(c)$
- if $\|s - c\|$ small $\rightsquigarrow m = 0$
- else $\rightsquigarrow m = 1$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

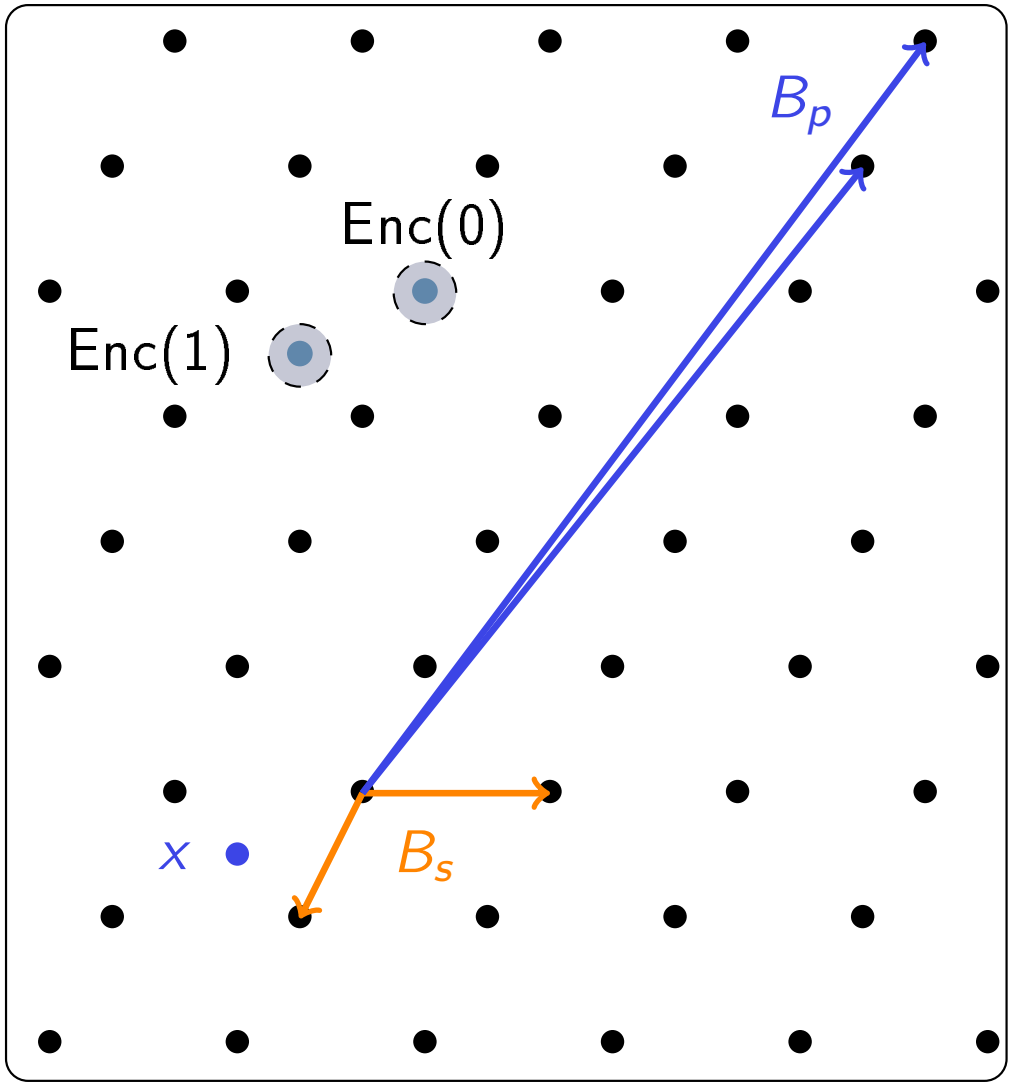
message: $m \in \{0, 1\}$

Correctness:

- if $m = 1$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

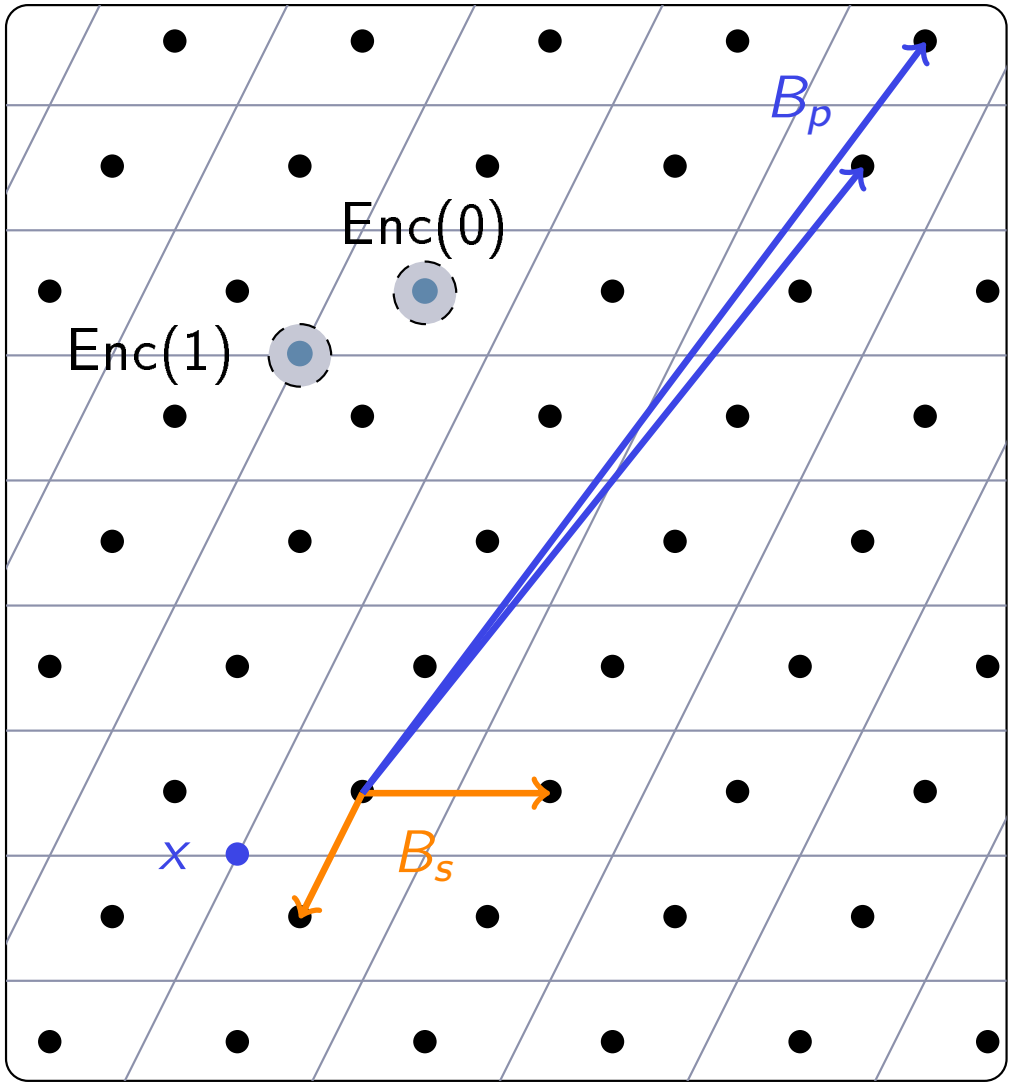
message: $m \in \{0, 1\}$

Correctness:

- if $m = 1$ ✓

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

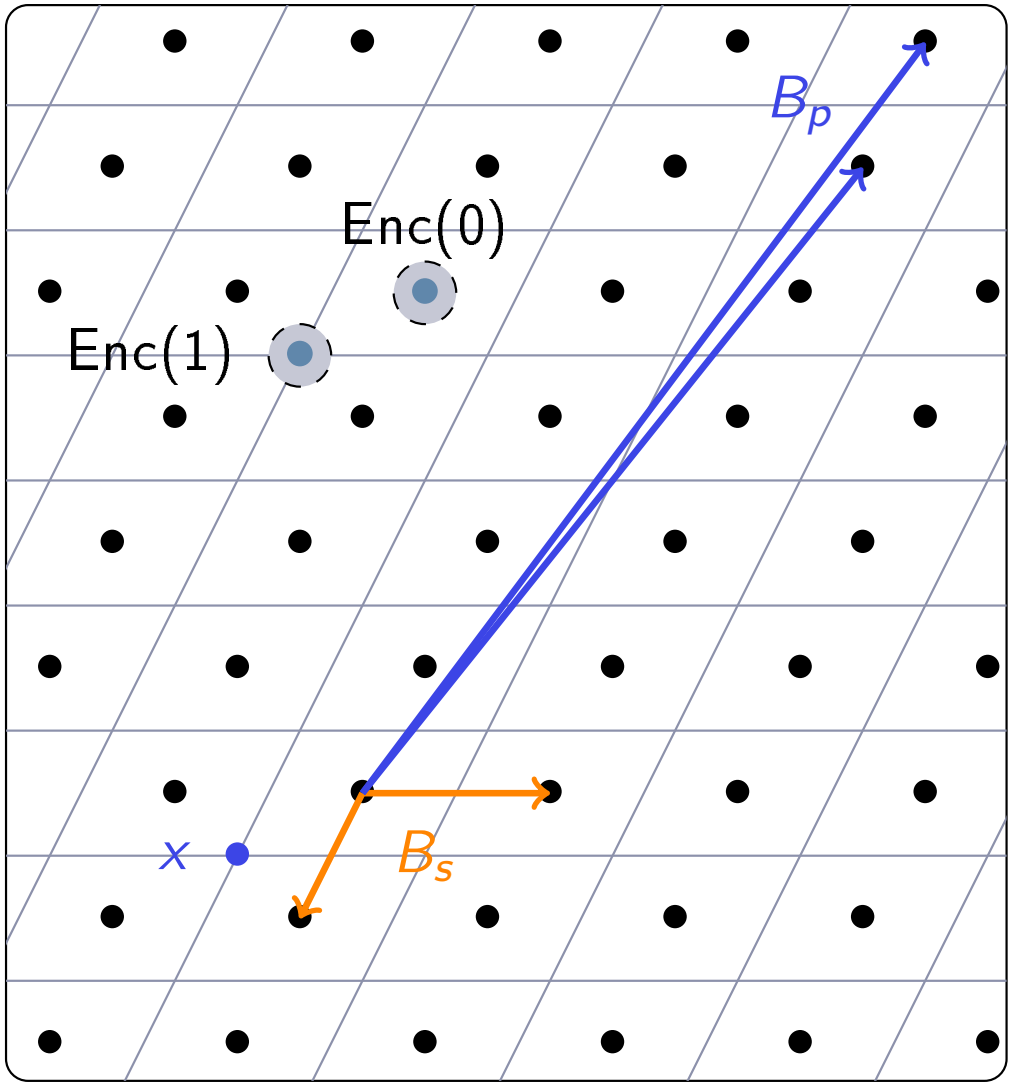
message: $m \in \{0, 1\}$

Correctness:

- if $m = 1$ ✓
- if $m = 0$

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

message: $m \in \{0, 1\}$

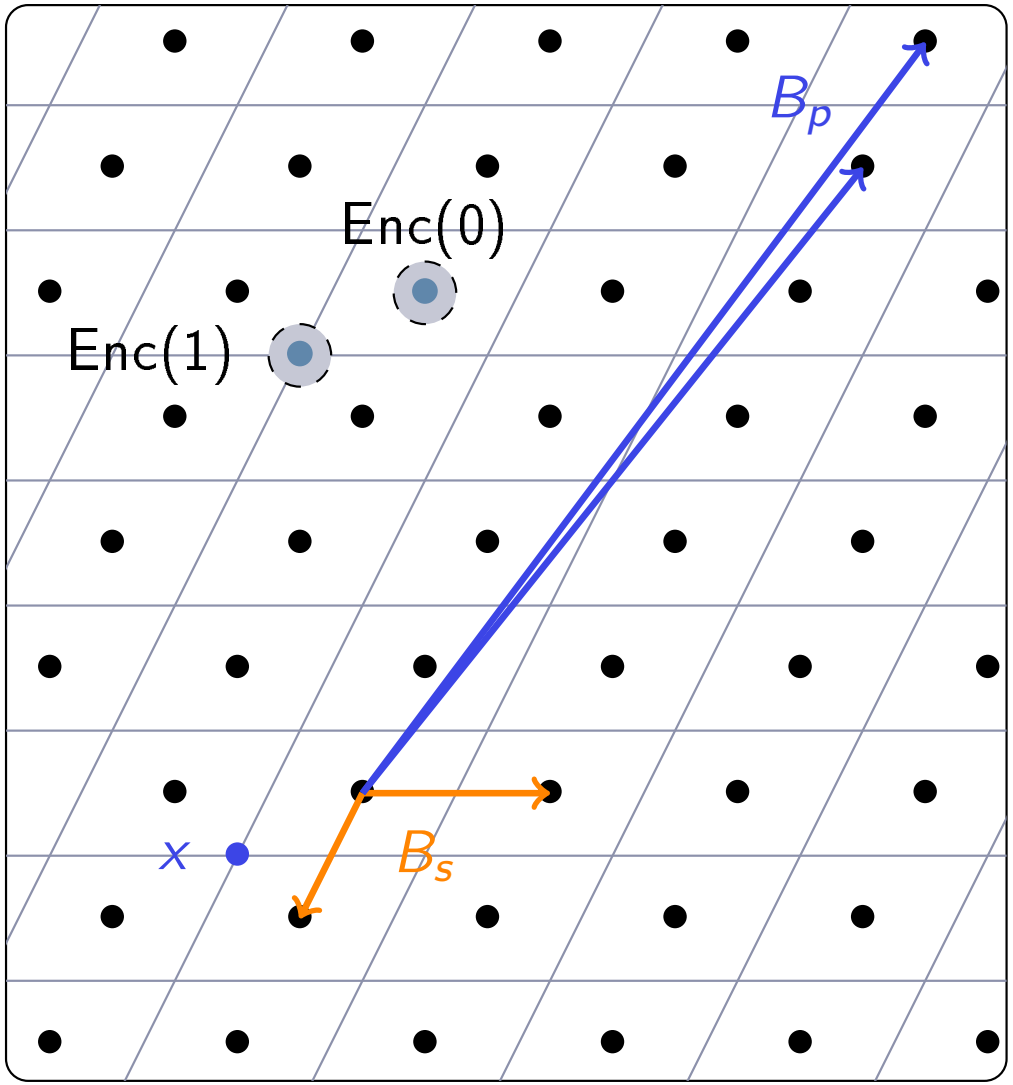
Correctness:

- if $m = 1$ ✓
- if $m = 0$ ✓

  if $\|e\| \leq$ decoding radius

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

message: $m \in \{0, 1\}$

Correctness:

- if $m = 1$ ✓
- if $m = 0$ ✓
  if $\|e\| \leq$ decoding radius

Security : ✓ if $\mathcal{L}$ is well chosen
(see later)

[Reg05] Regev. On lattices, learning with errors, random linear codes, and cryptography. STOC.

Section's conclusion

Regev-like encryption:

- requires a lattice $\mathcal{L}$ + a short basis B_s + a bad basis B_p + a point x far from $\mathcal{L}$
- seems secure if finding close vectors given only B_p is hard

Preview of next episodes

q-ary lattices

Notations: q, n, m integers, $1 \leq n \ll m$, $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$

Definition A lattice $\mathcal{L}$ of dimension m is called q-ary if

$$q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m.$$

q-ary lattices

Notations: q, n, m integers, $1 \leq n \ll m$, $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$

Definition A lattice $\mathcal{L}$ of dimension m is called q-ary if

$$q\mathbb{Z}^m \subset \mathcal{L} \subset \mathbb{Z}^m.$$

How to construct q-ary lattices:

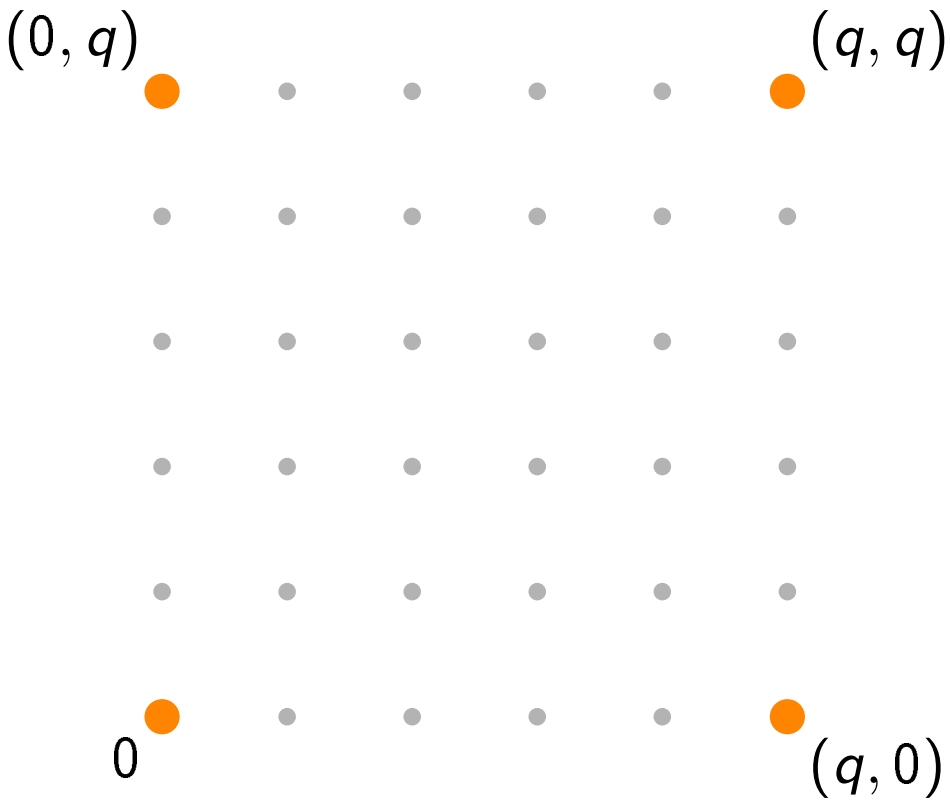
- pick $A \in \mathbb{Z}_q^{m \times n}$
- the image lattice of A:

$$\Lambda_q(A) := \{y \in \mathbb{Z}^m \,|\, y \equiv Ax \bmod q \text{ for some } x \in \mathbb{Z}_q^n\} = A\mathbb{Z}^n + q\mathbb{Z}^m$$

- the kernel lattice of A:

$$\Lambda_q^{\perp}(A) := \{x \in \mathbb{Z}^m \,|\, x^{\top} A \equiv 0 \bmod q\}$$

Example



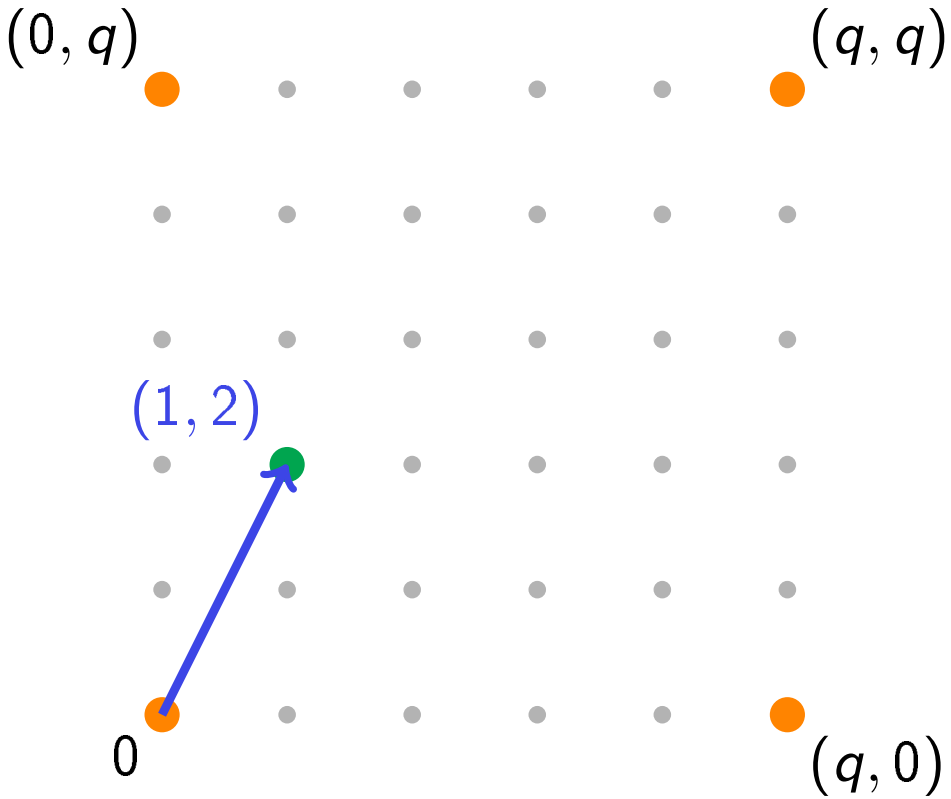
Suppose $q = 5, n = 1, m = 2$,

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\Lambda_q(A) = A\mathbb{Z}^n + q\mathbb{Z}^m$$

$$= \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \mathbb{Z} + 5\mathbb{Z}^2$$

Example



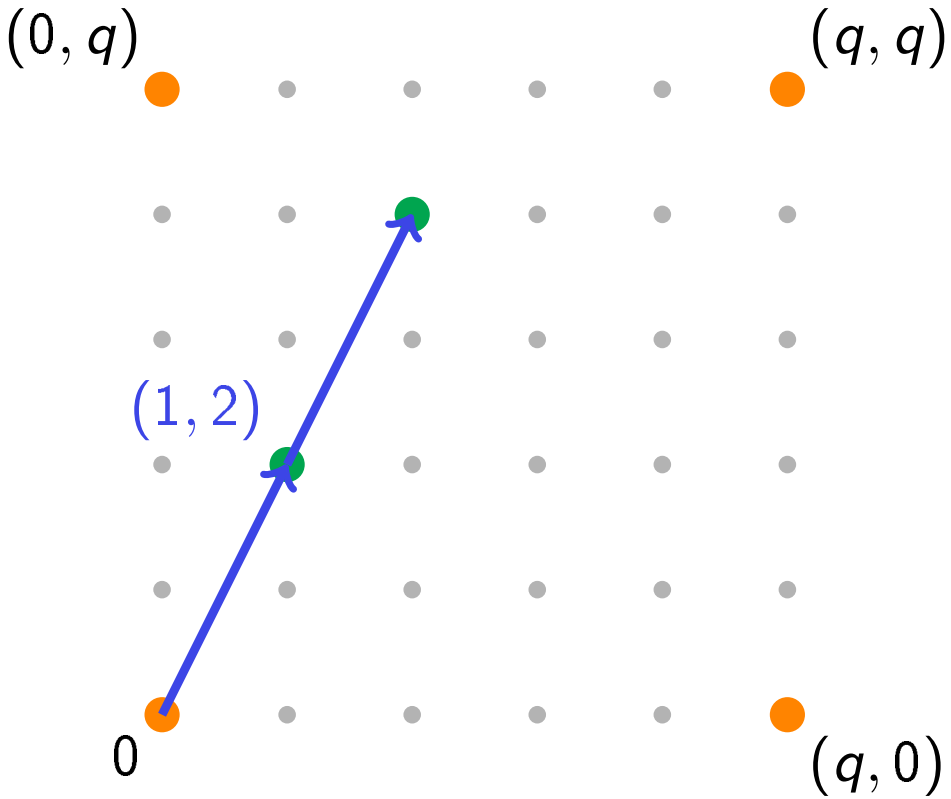
Suppose $q = 5, n = 1, m = 2$,

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\Lambda_q(A) = A\mathbb{Z}^n + q\mathbb{Z}^m$$

$$= \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \mathbb{Z} + 5\mathbb{Z}^2$$

Example



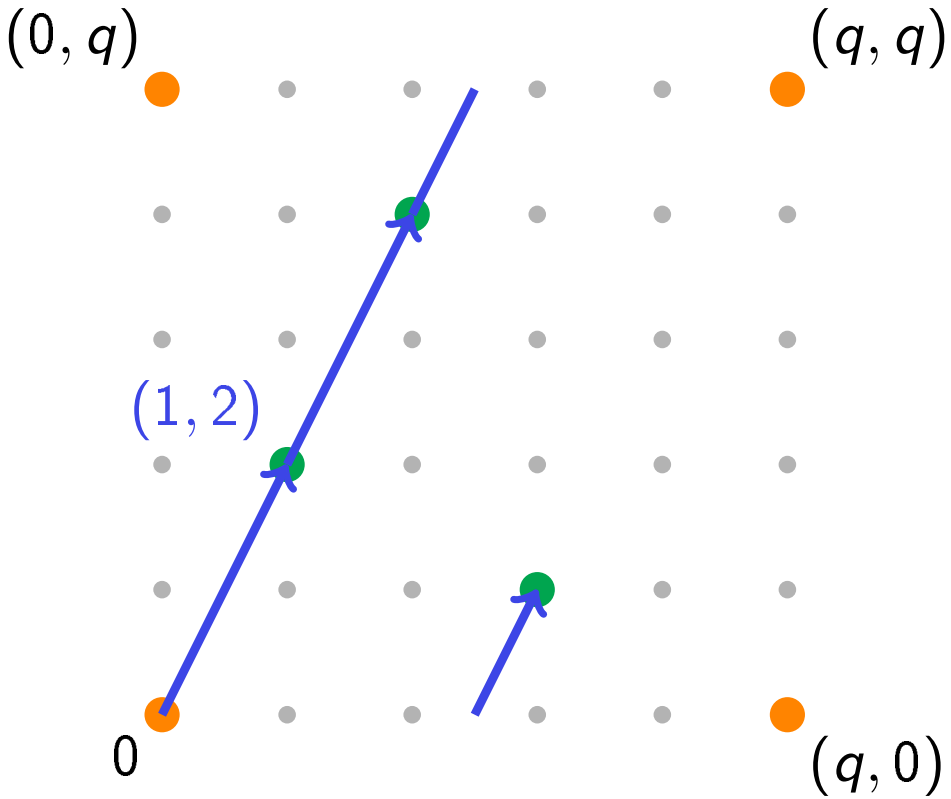
Suppose $q = 5, n = 1, m = 2$,

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\Lambda_q(A) = A\mathbb{Z}^n + q\mathbb{Z}^m$$

$$= \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \mathbb{Z} + 5\mathbb{Z}^2$$

Example



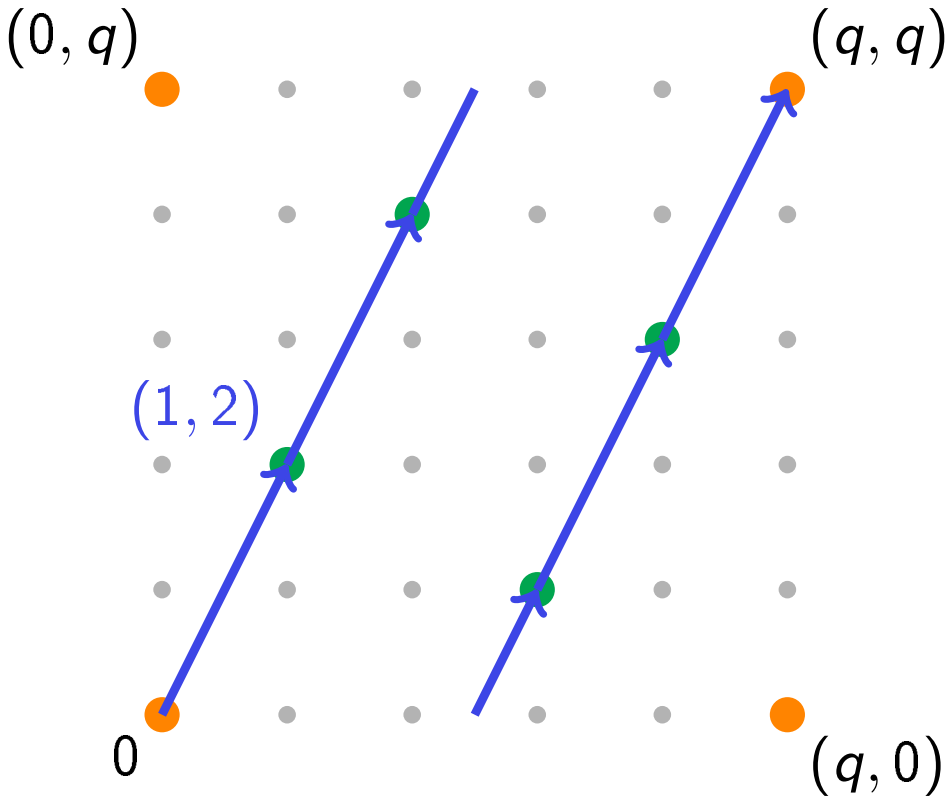
Suppose $q = 5, n = 1, m = 2$,

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\Lambda_q(A) = A\mathbb{Z}^n + q\mathbb{Z}^m$$

$$= \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \mathbb{Z} + 5\mathbb{Z}^2$$

Example

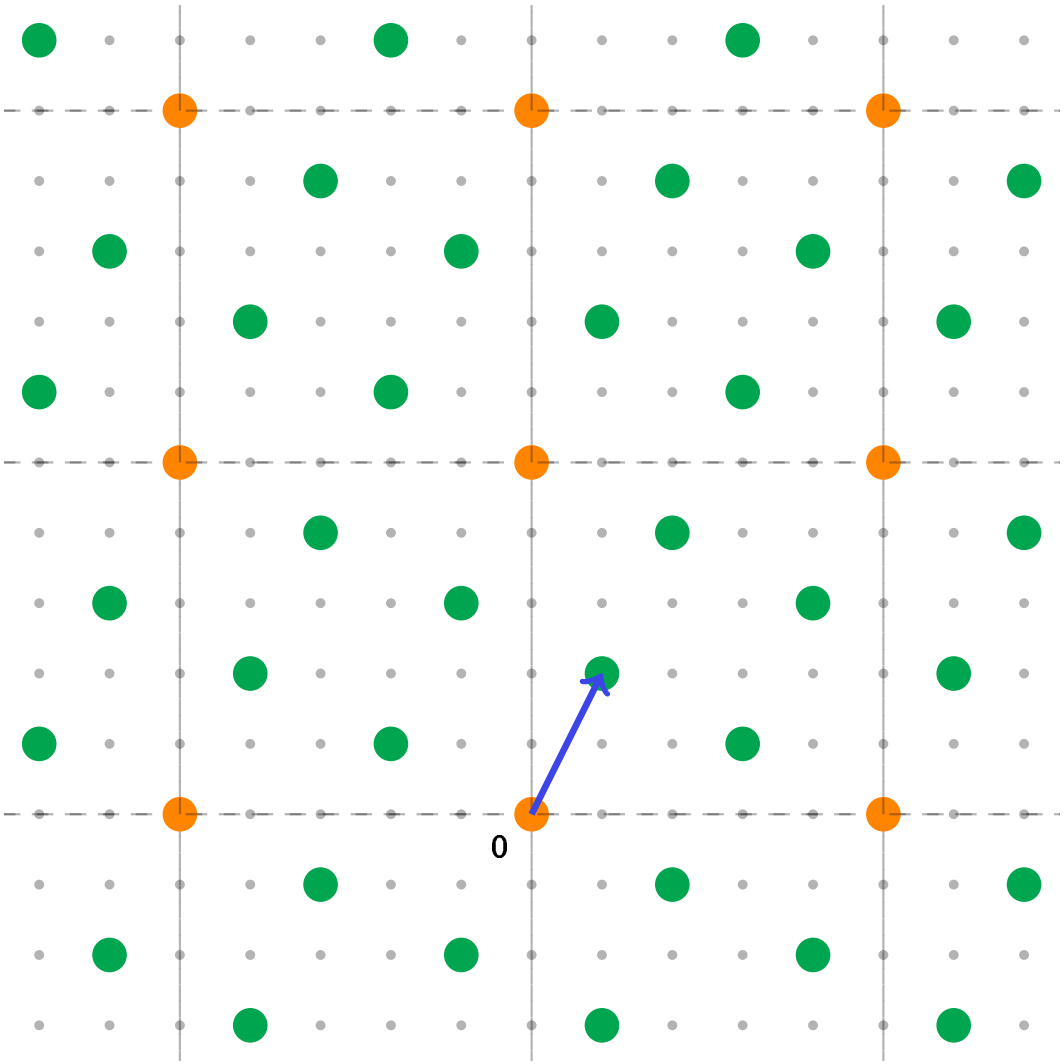


Suppose $q = 5, n = 1, m = 2$,

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\Lambda_q(A) = A\mathbb{Z}^n + q\mathbb{Z}^m$$
$$= \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \mathbb{Z} + 5\mathbb{Z}^2$$

Example



Suppose $q = 5, n = 1, m = 2$,

$$A = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

$$\Lambda_q(A) = A\mathbb{Z}^n + q\mathbb{Z}^m$$

$$= \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \mathbb{Z} + 5\mathbb{Z}^2$$