

Cryptography and isomorphism of lattices and graphs

Alice Pellet-Mary

(based on multiple works by Wessel van Woerden and co-authors)

Journées combinatoires de Bordeaux

February 2025

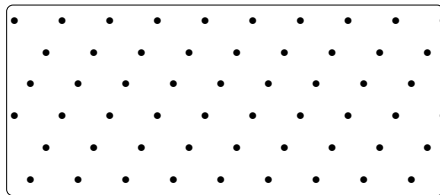


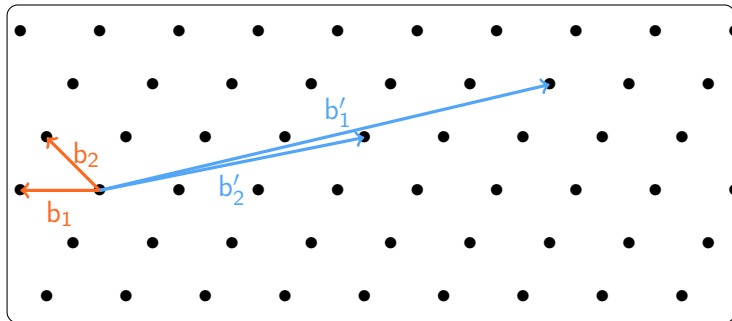
université
de **BORDEAUX**

Introduction: Lattices and lattice problems

1. Building public key encryption from lattices
 2. Algorithms for computing isomorphism of lattices
- } independent

Lattices and lattice problems

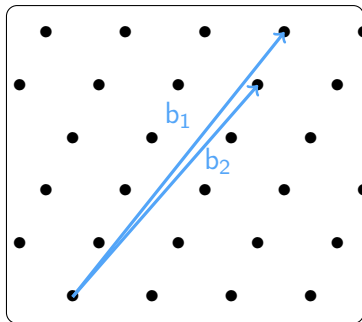




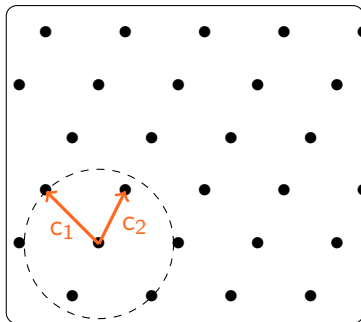
- ▶ $\mathcal{L} = \{\sum_{i=1}^n x_i b_i \mid \forall i, x_i \in \mathbb{Z}\}$ is a **lattice**
- ▶ $(b_1, \dots, b_n) =: B \in GL_n(\mathbb{R})$ is a **basis** (not unique)
- ▶ n is the **dimension** (or rank)

Short basis problem

Input:



Output:

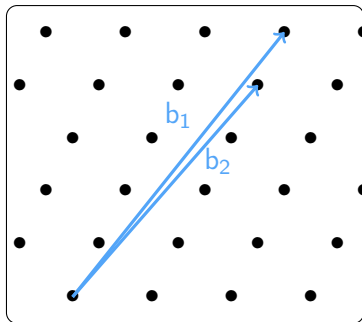


Shortest basis problem

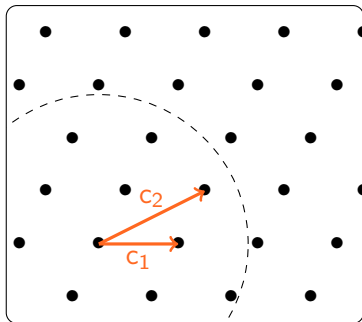
$$\max_i \|c_i\| \leq \min_{B' \text{ basis of } \mathcal{L}} \left(\max_i \|b'_i\| \right)$$

Short basis problem

Input:



Output:



Approximate short basis problem

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

The short basis problem is hard

Hardness of the short basis problem

The best known algorithms solving the short basis problem have complexity $\approx \exp(n)$ (for $\text{poly}(n)$ approximation factors)

The best known algorithms solving the short basis problem have complexity $\approx \exp(n)$ (for $\text{poly}(n)$ approximation factors)

Consequences

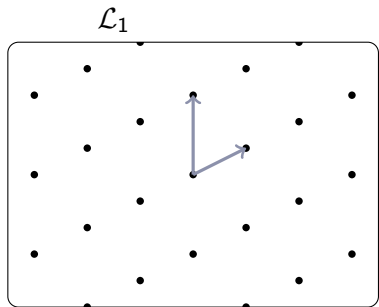
- ▶ can be used to build crypto

The best known algorithms solving the short basis problem have complexity $\approx \exp(n)$ (for $\text{poly}(n)$ approximation factors)

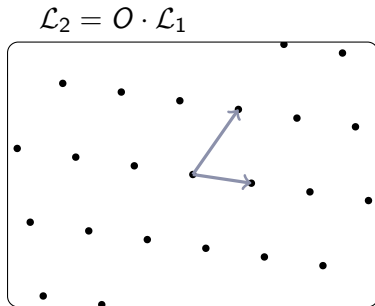
Consequences

- ▶ can be used to build crypto
- ▶ n has to be somewhat large (say 700 for crypto)

Isomorphic lattices



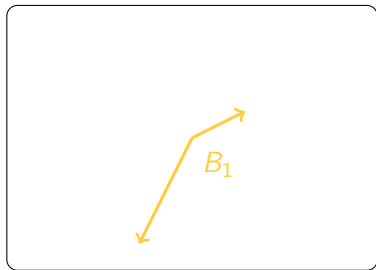
\cong



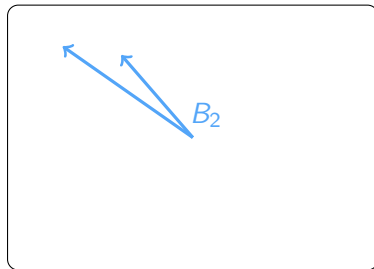
Definition: Two lattices \mathcal{L}_1 and \mathcal{L}_2 are **isomorphic** if there exists $O \in \mathcal{O}_n(\mathbb{R})$ such that $\mathcal{L}_2 = O\mathcal{L}_1$.
($\mathcal{O}_n(\mathbb{R}) =$ orthonormal transformations $\Rightarrow O^T O = I_n$)

The lattice isomorphism problem

The lattice isomorphism problem (LIP):
given B_1 and B_2 , bases of two isomorphic lattices,
find $O \in \mathcal{O}_n(\mathbb{R})$ such that $\mathcal{L}(B_1) = O\mathcal{L}(B_2)$.

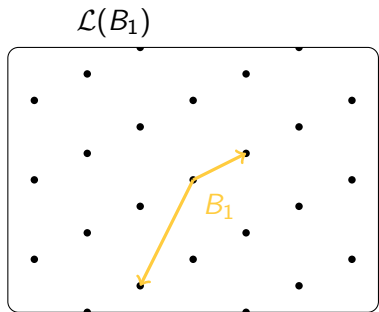


\simeq

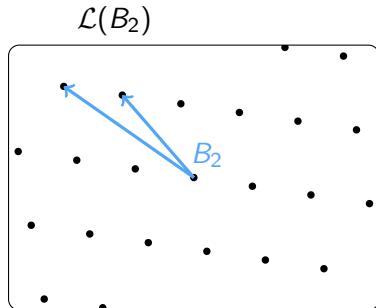


The lattice isomorphism problem

The lattice isomorphism problem (LIP):
given B_1 and B_2 , bases of two isomorphic lattices,
find $O \in \mathcal{O}_n(\mathbb{R})$ such that $\mathcal{L}(B_1) = O\mathcal{L}(B_2)$.

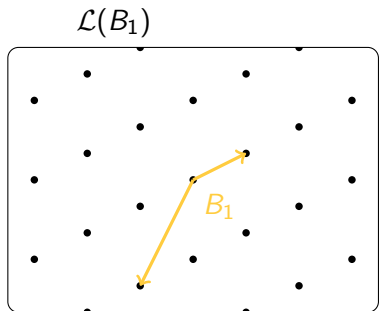


\simeq

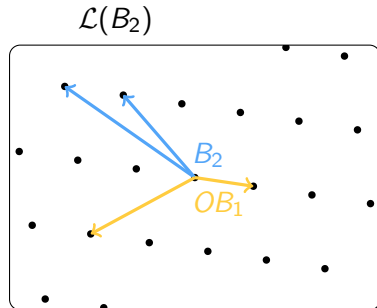


The lattice isomorphism problem

The lattice isomorphism problem (LIP):
given B_1 and B_2 , bases of two isomorphic lattices,
find $O \in \mathcal{O}_n(\mathbb{R})$ such that $\mathcal{L}(B_1) = O\mathcal{L}(B_2)$.

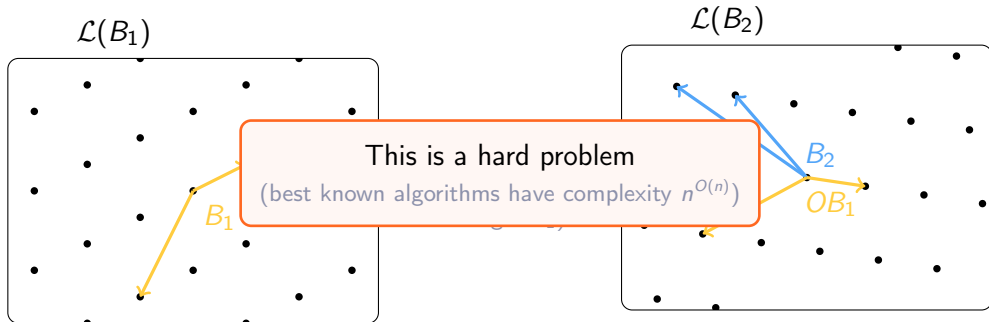


\approx
(finding $O \Leftrightarrow$
finding OB_1)



The lattice isomorphism problem

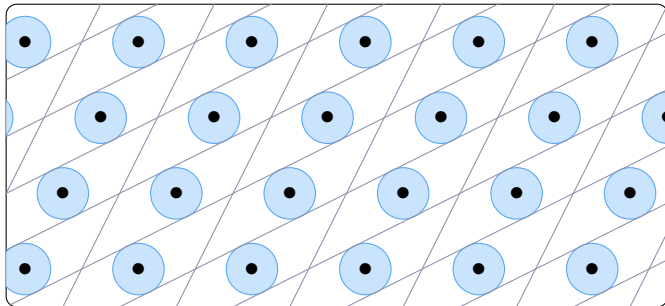
The lattice isomorphism problem (LIP):
given B_1 and B_2 , bases of two isomorphic lattices,
find $O \in \mathcal{O}_n(\mathbb{R})$ such that $\mathcal{L}(B_1) = O\mathcal{L}(B_2)$.



We have seen:

- ▶ lattices
 - ▶ the short basis problem
 - ▶ the lattice isomorphism problem
- } hard algorithmic problems

Constructing public key encryption



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

Public key encryption

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

Alice

Bob

$(pk, sk) \leftarrow \text{KeyGen}() \xrightarrow{pk}$

Public key encryption

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

Alice

Bob

$$(pk, sk) \leftarrow \text{KeyGen}() \xrightarrow{pk}$$

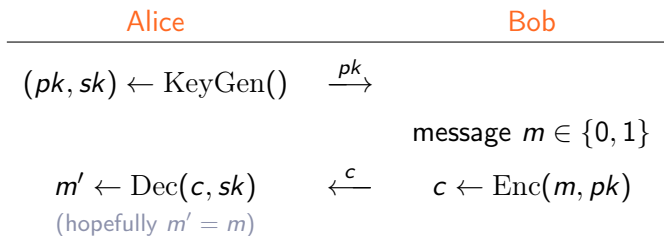
message $m \in \{0, 1\}$

$$m' \leftarrow \text{Dec}(c, sk) \xleftarrow{c} c \leftarrow \text{Enc}(m, pk)$$

(hopefully $m' = m$)

Public key encryption

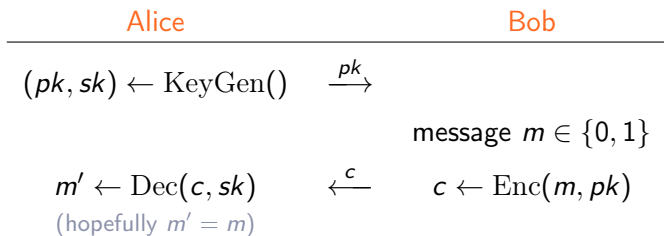
- Three algorithms:
- ▶ $\text{KeyGen}() \rightsquigarrow (pk, sk)$
 - ▶ $\text{Enc}(m, pk) \rightsquigarrow c$
 - ▶ $\text{Dec}(c, sk) \rightsquigarrow m'$



Correctness: $\text{Dec}(\text{Enc}(m, pk), sk) = m$

Public key encryption

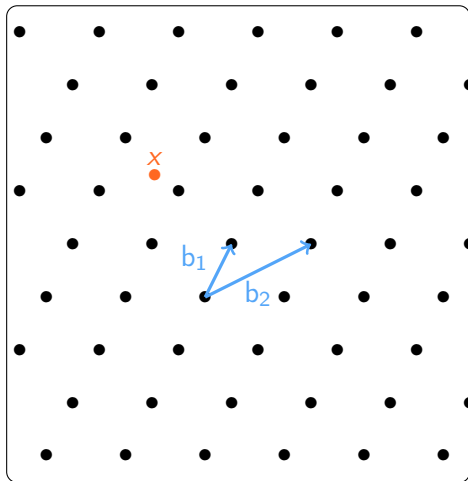
- Three algorithms:
- ▶ $\text{KeyGen}() \rightsquigarrow (pk, sk)$
 - ▶ $\text{Enc}(m, pk) \rightsquigarrow c$
 - ▶ $\text{Dec}(c, sk) \rightsquigarrow m'$



Correctness: $\text{Dec}(\text{Enc}(m, pk), sk) = m$

Security: an attacker cannot guess m from pk and $\text{Enc}(m, pk)$

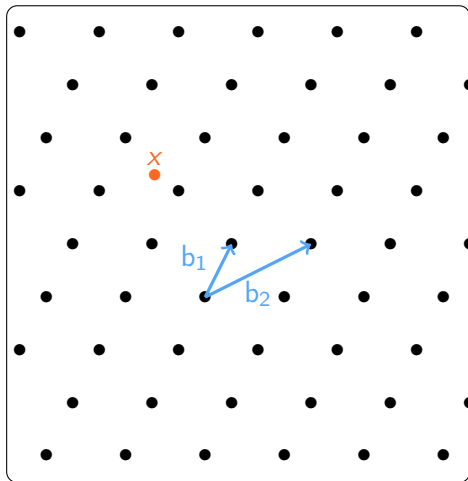
Decoding in a lattice



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

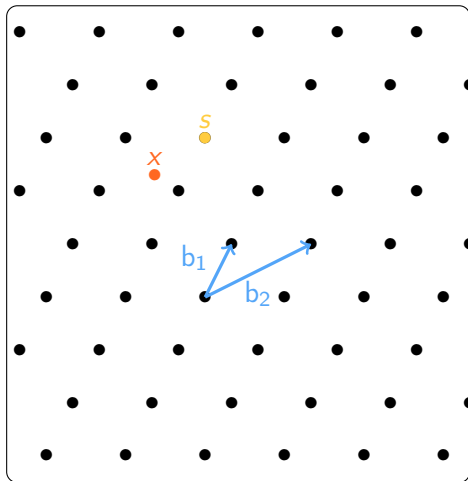


Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

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

Algo: round each coordinate
(called Babai's round-off algorithm)

Decoding in a lattice



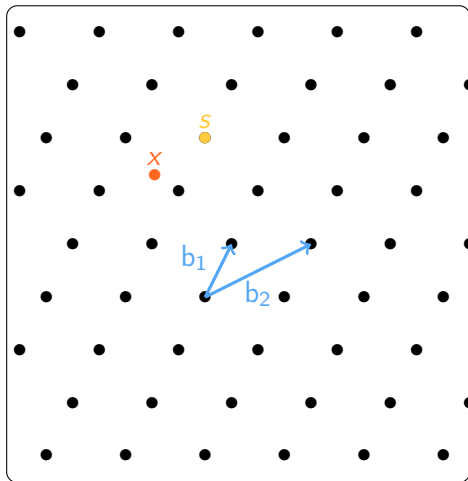
Input: $x = 3.7 \cdot b_1 - 1.4 \cdot b_2$

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

Algo: round each coordinate
(called Babai's round-off algorithm)

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

Decoding in a lattice



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

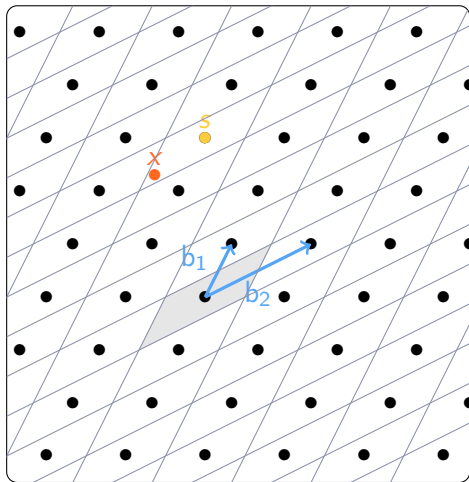
Objective: find $s \in \mathcal{L}$ close to x

Algo: round each coordinate
(called Babai's round-off algorithm)

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

Solves approximate decoding: the smaller the basis, the closer the solution

Decoding in a lattice




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

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

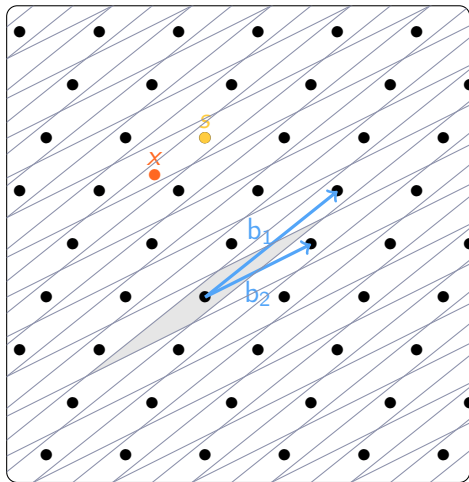
Algo: round each coordinate
(called Babai's round-off algorithm)

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

Solves approximate decoding: the smaller the basis, the closer the solution

 $= \left\{ x_1 b_1 + x_2 b_2 \mid |x_i| \leq \frac{1}{2} \right\}$
(fundamental parallelepiped)

Decoding in a lattice




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

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

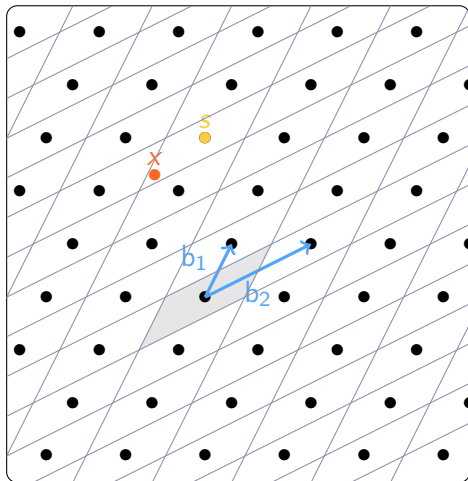
Algo: round each coordinate
(called Babai's round-off algorithm)

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

Solves approximate decoding: the smaller the basis, the closer the solution

 $= \left\{ x_1 b_1 + x_2 b_2 \mid |x_i| \leq \frac{1}{2} \right\}$
(fundamental parallelepiped)

What about exact decoding?

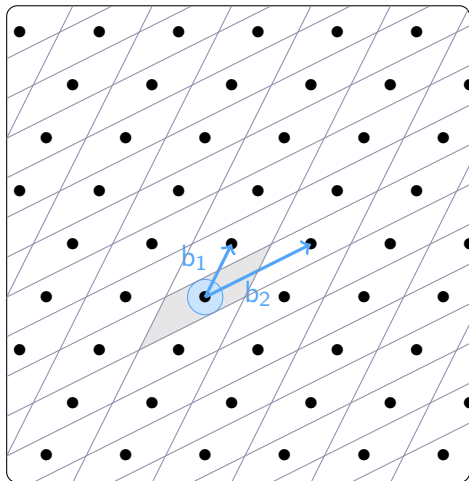


$$\text{Babai}_B(x) = s \quad (\in \mathcal{L}, \text{ close to } x)$$

$$\text{parallelepiped} = \left\{ x_1 b_1 + x_2 b_2 \mid |x_i| \leq \frac{1}{2} \right\}$$

(fundamental parallelepiped)

What about exact decoding?

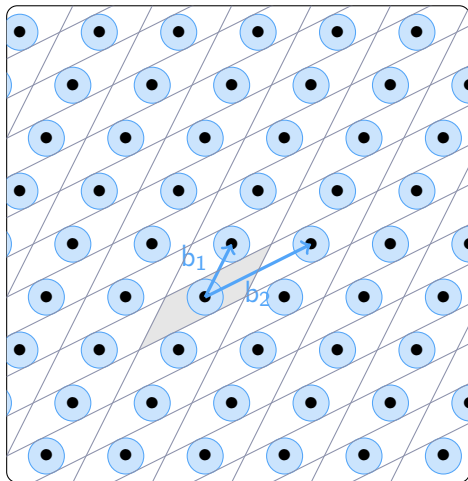


$\text{Babai}_B(x) = s$ ($s \in \mathcal{L}$, close to x)


$\text{▱} = \left\{ x_1 b_1 + x_2 b_2 \mid |x_i| \leq \frac{1}{2} \right\}$
(fundamental parallelepiped)



$\text{⊙} = \text{largest circle} \subseteq \text{▱}$
(radius of ⊙ : decoding radius)

What about exact decoding?



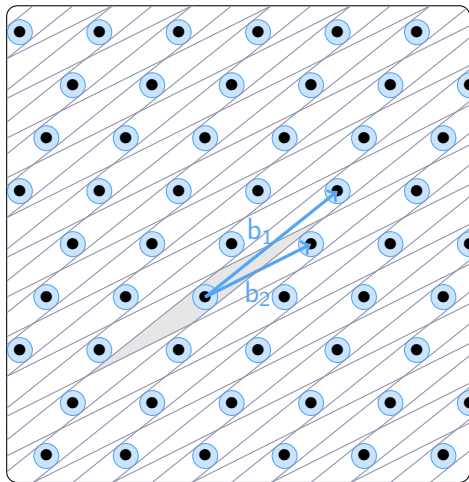
$\text{Babai}_B(x) = s$ ($s \in \mathcal{L}$, close to x)

 $= \left\{ x_1 b_1 + x_2 b_2 \mid |x_i| \leq \frac{1}{2} \right\}$
(fundamental parallelepiped)

 $=$ largest circle \subseteq 
(radius of : decoding radius)

Lemma: $\forall s \in \mathcal{L}$ and $\forall e \in \text{circle}$
 $\text{Babai}_B(s + e) = s$

What about exact decoding?



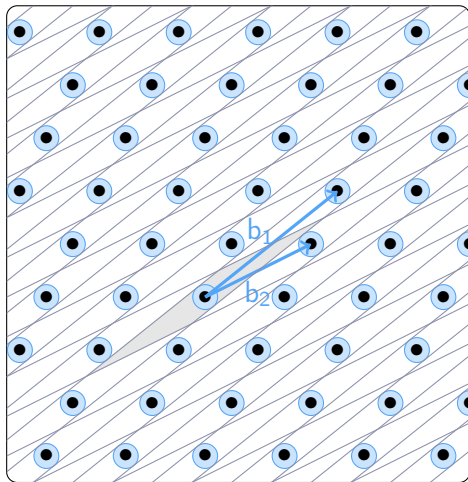
$\text{Babai}_B(x) = s$ ($s \in \mathcal{L}$, close to x)

$\text{▱} = \left\{ x_1 b_1 + x_2 b_2 \mid |x_i| \leq \frac{1}{2} \right\}$
(fundamental parallelepiped)

\bullet = largest circle $\subseteq \text{▱}$
(radius of \bullet : decoding radius)

Lemma: $\forall s \in \mathcal{L}$ and $\forall e \in \bullet$
 $\text{Babai}_B(s + e) = s$

What about exact decoding?



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

$$\text{parallelepiped} = \left\{ x_1 b_1 + x_2 b_2 \mid |x_i| \leq \frac{1}{2} \right\}$$

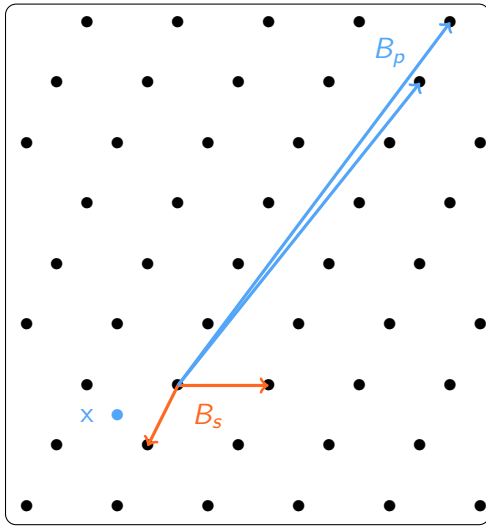
(fundamental parallelepiped)

● = largest circle \subseteq parallelepiped
(radius of ●: decoding radius)

Lemma: $\forall s \in \mathcal{L}$ and $\forall e \in \text{●}$
 $\text{Babai}_B(s + e) = s$

Smaller basis \iff larger ●

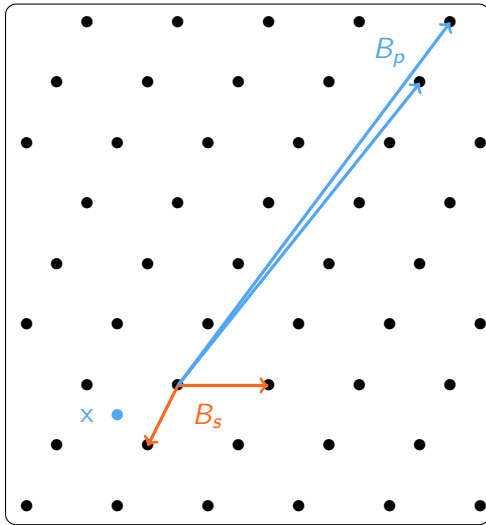
Public key encryption from lattices [Reg05]



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

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

Public key encryption from lattices [Reg05]

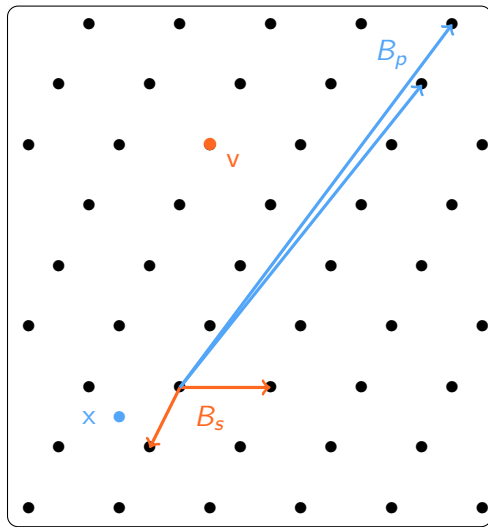


$$\text{pk} = (B_p, x)$$
$$\text{sk} = B_s$$

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

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

Public key encryption from lattices [Reg05]



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

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

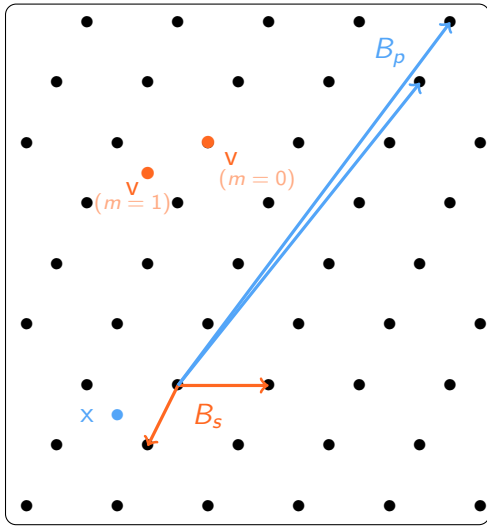
$$\text{sk} = B_s$$

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

- ▶ Sample random $v \in \mathcal{L}$

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

Public key encryption from lattices [Reg05]



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

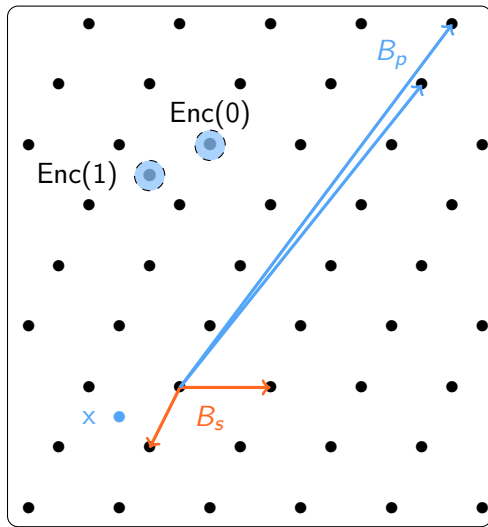
message: $m \in \{0, 1\}$

$$\text{sk} = B_s$$

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

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

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



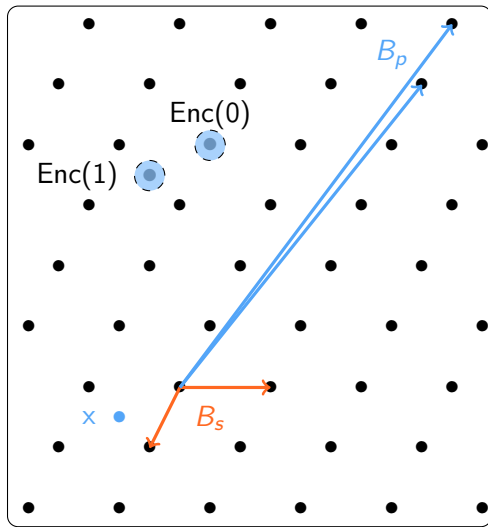
$$pk = (B_p, x)$$

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

$$sk = B_s$$

$Enc(m, pk)$:

- ▶ Sample random $v \in \mathcal{L}$
- ▶ if $m = 1$: $v \leftarrow v + x$
- ▶ Sample small $e \in \bullet$
- ▶ return $c = v + e$



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

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

$$\text{sk} = B_s$$

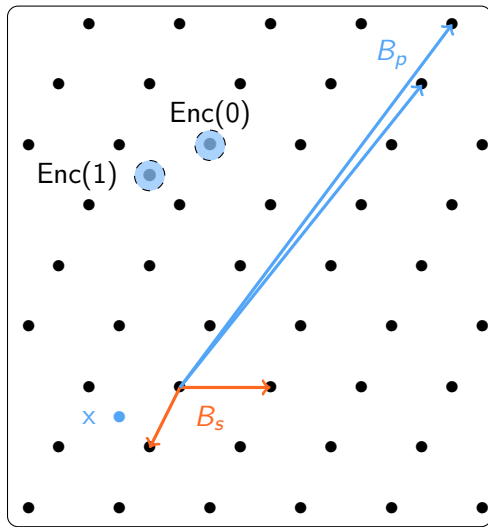
$\text{Enc}(m, \text{pk})$:

- ▶ Sample random $v \in \mathcal{L}$
- ▶ if $m = 1$: $v \leftarrow v + x$
- ▶ Sample small $e \in \bullet$
- ▶ return $c = v + e$

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

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

Public key encryption from lattices [Reg05]



$$pk = (B_p, x)$$

$$sk = B_s$$

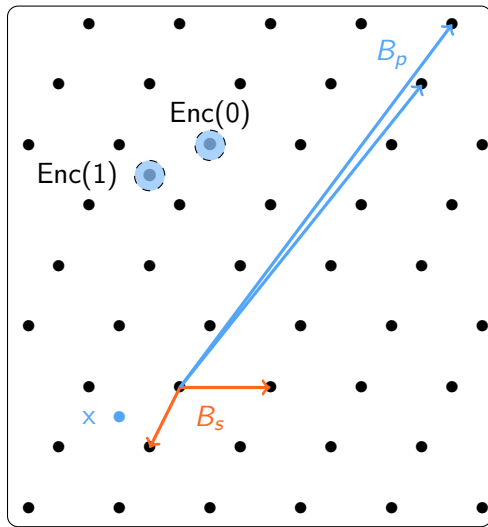
message: $m \in \{0, 1\}$

Correctness:

▶ if $m = 1$

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

Public key encryption from lattices [Reg05]



$$pk = (B_p, x)$$

$$sk = B_s$$

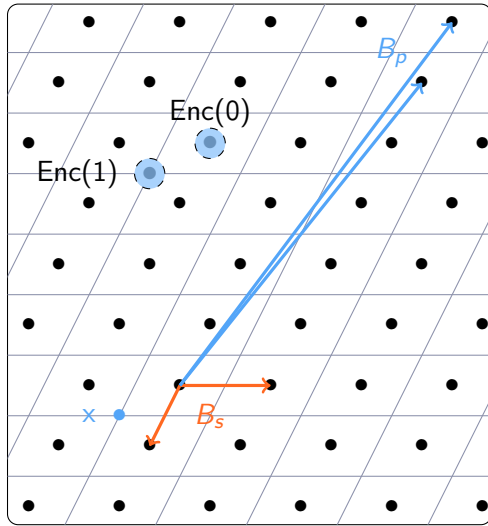
message: $m \in \{0, 1\}$

Correctness:

▶ if $m = 1$ ✓

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

Public key encryption from lattices [Reg05]



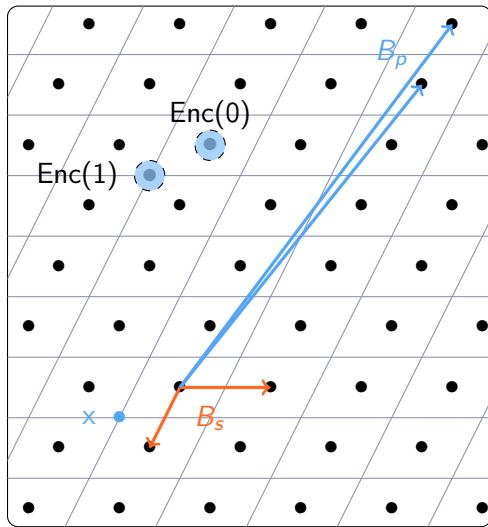
$$\text{pk} = (B_p, x)$$
$$\text{sk} = B_s$$

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.



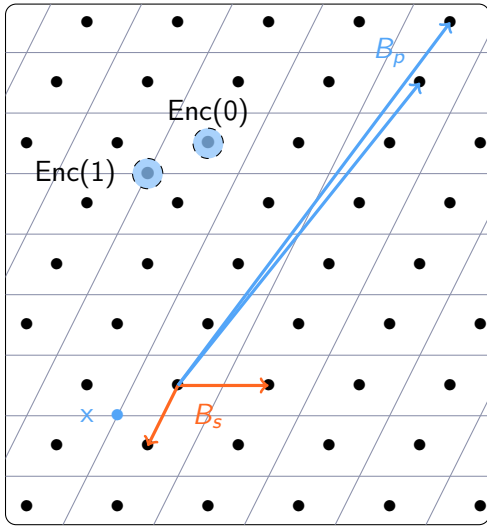
$$pk = (B_p, x)$$

$$sk = B_s$$

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

Correctness:

- ▶ if $m = 1$ ✓
- ▶ if $m = 0$ ✓ (if $\|e\| \leq \text{decoding radius}$)



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

$$\text{sk} = B_s$$

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

Correctness:

- ▶ if $m = 1$ ✓
- ▶ if $m = 0$ ✓ (if $\|e\| \leq \text{decoding radius}$)

Security : ✓ if \mathcal{L} is well chosen

(under some assumption, see next slide)

What we want: An algorithm `KeyGen` such that

- ▶ `KeyGen` computes
 - ▶ a random lattice \mathcal{L}
 - ▶ a short basis B_s of \mathcal{L} (sk)
 - ▶ a long basis B_p of \mathcal{L} (pk)
 - ▶ a point x far from \mathcal{L}

What we want: An algorithm `KeyGen` such that

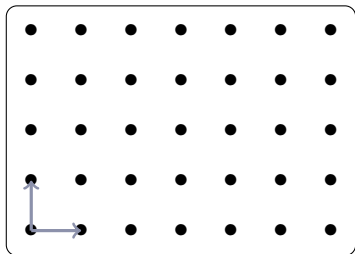
- ▶ `KeyGen` computes
 - ▶ a random lattice \mathcal{L}
 - ▶ a short basis B_s of \mathcal{L} (sk)
 - ▶ a long basis B_p of \mathcal{L} (pk)
 - ▶ a point x far from \mathcal{L}
- ▶ decoding errors in \odot is **easy** given B_s (correctness)

What we want: An algorithm `KeyGen` such that

- ▶ `KeyGen` computes
 - ▶ a random lattice \mathcal{L}
 - ▶ a short basis B_s of \mathcal{L} (sk)
 - ▶ a long basis B_p of \mathcal{L} (pk)
 - ▶ a point x far from \mathcal{L}
- ▶ decoding errors in \odot is **easy** given B_s (correctness)
- ▶ decoding errors in \odot is **hard** given B_p (security)

Constructing good lattices

$$\mathcal{L}_0 = \mathbb{Z}^n$$



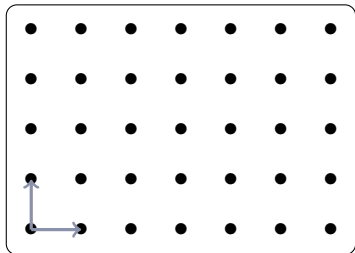
Keygen:

1. Start with $\mathcal{L}_0 = \mathbb{Z}^n$



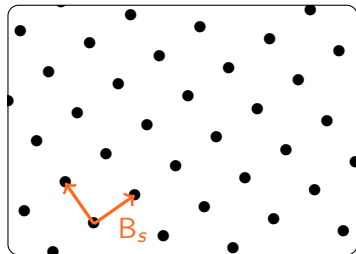
Constructing good lattices

$$\mathcal{L}_0 = \mathbb{Z}^n$$



rotate
→
(choose O
orthogonal matrix)

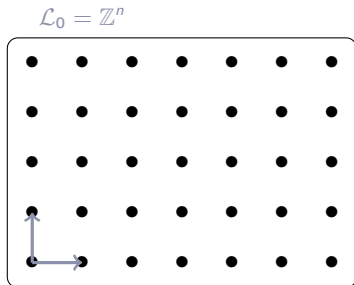
$$\mathcal{L} = O\mathbb{Z}^n$$



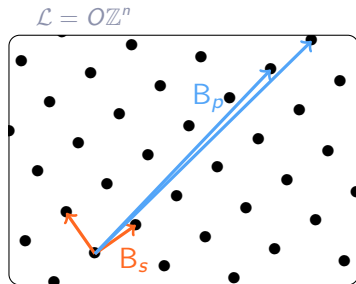
Keygen:

1. Start with $\mathcal{L}_0 = \mathbb{Z}^n$
2. Compute a random $O \in \mathcal{O}_n(\mathbb{R})$
 - ▶ $\mathcal{L} = O\mathcal{L}_0$
 - ▶ $B_s = O \cdot I_n = O$

Constructing good lattices



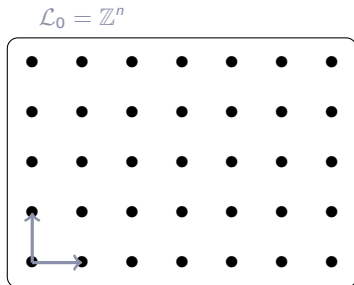
rotate
→
(choose O
orthogonal matrix)



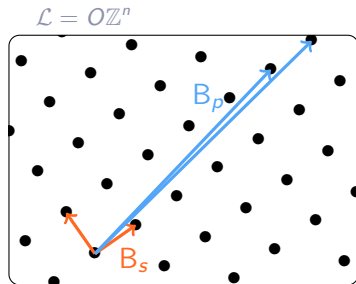
Keygen:

1. Start with $\mathcal{L}_0 = \mathbb{Z}^n$
2. Compute a random $O \in \mathcal{O}_n(\mathbb{R})$
 - ▶ $\mathcal{L} = O\mathcal{L}_0$
 - ▶ $B_s = O \cdot I_n = O$
3. Compute a long basis of \mathcal{L} ▶ B_p

Constructing good lattices



rotate
→
(choose O
orthogonal matrix)

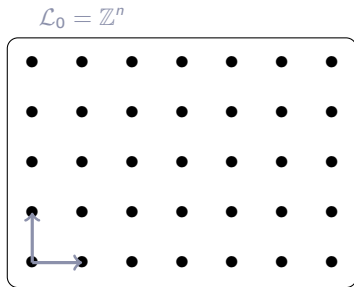


Keygen:

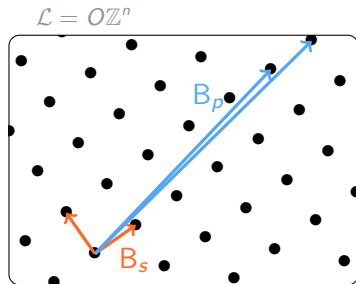
1. Start with $\mathcal{L}_0 = \mathbb{Z}^n$
2. Compute a random $O \in \mathcal{O}_n(\mathbb{R})$
 - ▶ $\mathcal{L} = O\mathcal{L}_0$
 - ▶ $B_s = O \cdot I_n = O$
3. Compute a long basis of \mathcal{L} ▶ B_p

Correctness: B_s corrects errors $\|e\| \leq 1/2$

Constructing good lattices



rotate
→
(choose O
orthogonal matrix)



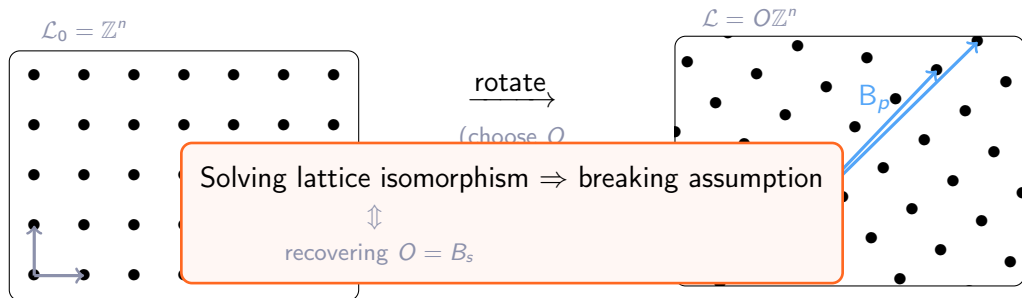
Keygen:

1. Start with $\mathcal{L}_0 = \mathbb{Z}^n$
2. Compute a random $O \in \mathcal{O}_n(\mathbb{R})$
 - ▶ $\mathcal{L} = O\mathcal{L}_0$
 - ▶ $B_s = O \cdot I_n = O$
3. Compute a long basis of \mathcal{L} ▶ B_p

Correctness: B_s corrects errors $\|e\| \leq 1/2$

Assumption: decoding errors $\|e\| \leq 1/2$
in \mathcal{L} given B_p is hard

Constructing good lattices



Keygen:

1. Start with $\mathcal{L}_0 = \mathbb{Z}^n$
2. Compute a random $O \in \mathcal{O}_n(\mathbb{R})$
 - ▶ $\mathcal{L} = O\mathcal{L}_0$
 - ▶ $B_s = O \cdot I_n = O$
3. Compute a long basis of \mathcal{L} ▶ B_p

Correctness: B_s corrects errors $\|e\| \leq 1/2$

Assumption: decoding errors $\|e\| \leq 1/2$
in \mathcal{L} given B_p is hard

We have seen:

- ▶ construction of public key encryption
 - ▶ correct thanks to Babai decoding algorithm
 - ▶ secure if decoding with a long basis is hard

We have seen:

- ▶ construction of public key encryption
 - ▶ correct thanks to Babai decoding algorithm
 - ▶ secure if decoding with a long basis is hard
- ▶ instantiated with rotations of \mathbb{Z}^n [DW22,BGPS23]
 - ▶ recovering isomorphism breaks the encryption scheme

[DW22] Ducas and van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices and cryptography. Eurocrypt
[BGPS23] Bennett, Ganju, Peetathawatchai, Stephens-Davidowitz. Just how hard are rotations of \mathbb{Z}^n ? [...] Eurocrypt

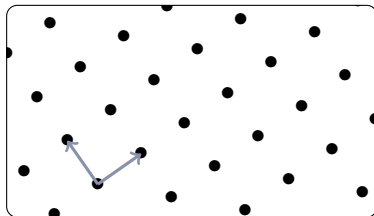
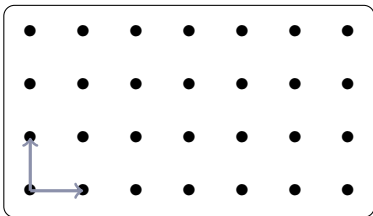
We have seen:

- ▶ construction of public key encryption
 - ▶ correct thanks to Babai decoding algorithm
 - ▶ secure if decoding with a long basis is hard
- ▶ instantiated with rotations of \mathbb{Z}^n [DW22,BGPS23]
 - ▶ recovering isomorphism breaks the encryption scheme

How hard is the lattice isomorphism problem?

[DW22] Ducas and van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices and cryptography. Eurocrypt
[BGPS23] Bennett, Ganju, Peetathawatchai, Stephens-Davidowitz. Just how hard are rotations of \mathbb{Z}^n ? [...] Eurocrypt

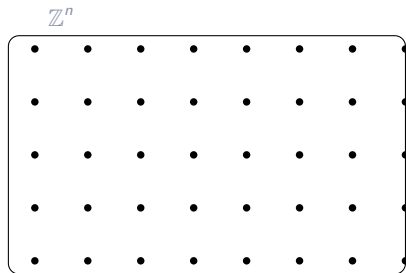
Solving the lattice isomorphism problem



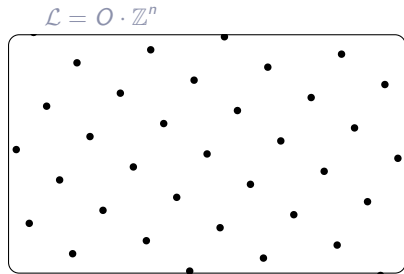
Example: the \mathbb{Z}^n case

Objective: given $\mathcal{L} = O \cdot \mathbb{Z}^n$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathbb{Z}^n)



\mathbb{R}^n

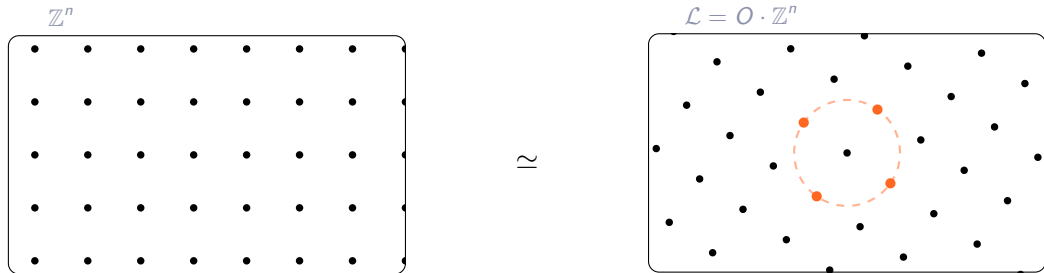


Example: the \mathbb{Z}^n case

Objective: given $\mathcal{L} = O \cdot \mathbb{Z}^n$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathbb{Z}^n)

Algorithm: 1. Compute the shortest non-zero vectors of \mathcal{L}

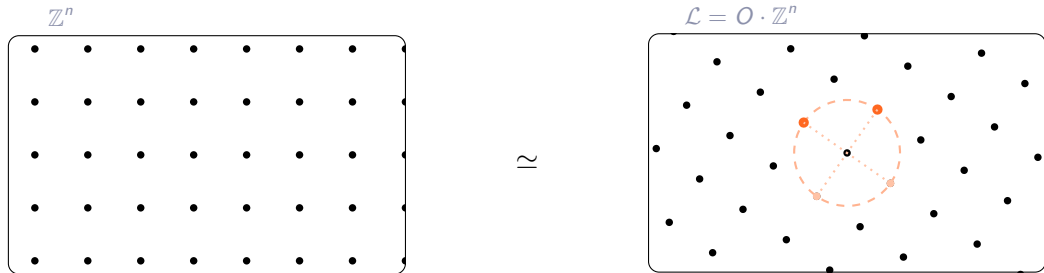


Example: the \mathbb{Z}^n case

Objective: given $\mathcal{L} = O \cdot \mathbb{Z}^n$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathbb{Z}^n)

- Algorithm:**
1. Compute the shortest non-zero vectors of \mathcal{L}
 2. Keep one vector per pair $(v, -v)$ (arbitrary)

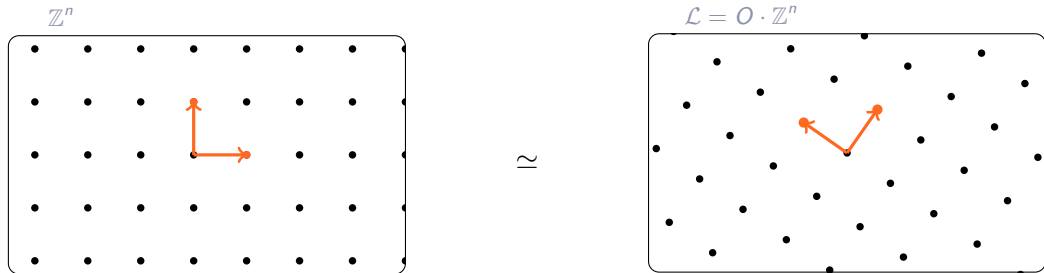


Example: the \mathbb{Z}^n case

Objective: given $\mathcal{L} = O \cdot \mathbb{Z}^n$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathbb{Z}^n)

- Algorithm:**
1. Compute the shortest non-zero vectors of \mathcal{L}
 2. Keep one vector per pair $(v, -v)$ (arbitrary)
 3. Match these with the e_i basis of \mathbb{Z}^n (arbitrary)



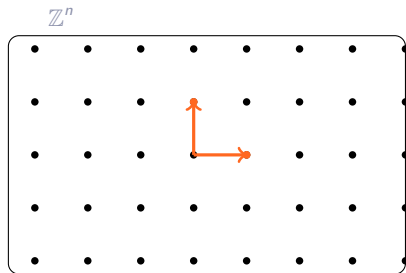
Example: the \mathbb{Z}^n case

Objective: given $\mathcal{L} = O \cdot \mathbb{Z}^n$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

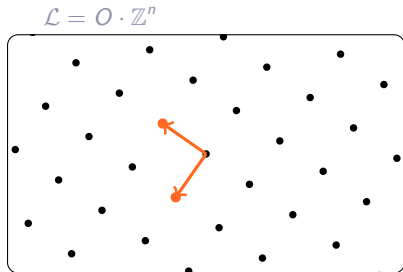
Remark: O is not unique (automorphisms of \mathbb{Z}^n)

- Algorithm:**
1. Compute the shortest non-zero vectors of \mathcal{L}
 2. Keep one vector per pair $(v, -v)$ (arbitrary)
 3. Match these with the e_i basis of \mathbb{Z}^n (arbitrary)

Every arbitrary choice
gives a (different)
solution



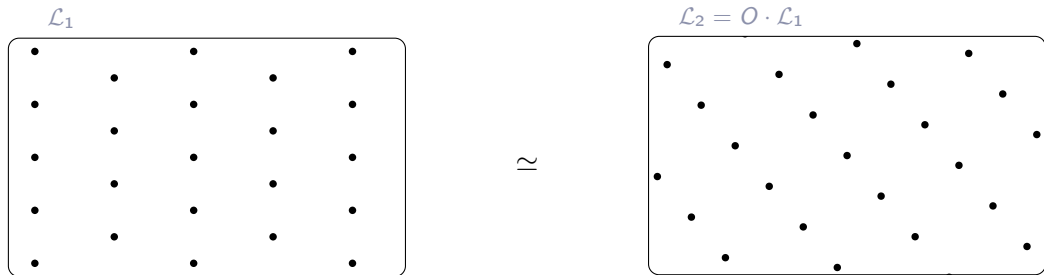
\cong



General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

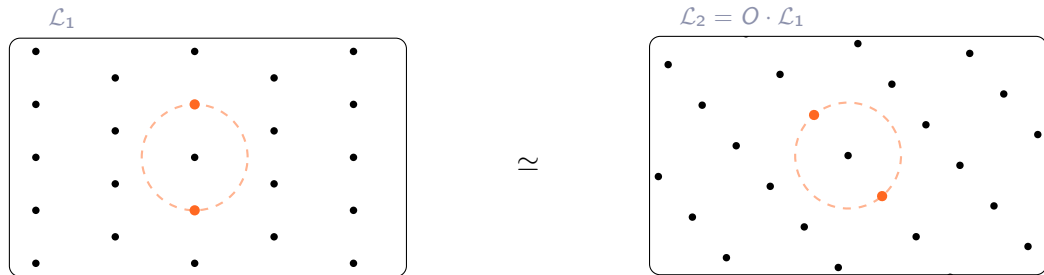


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

Algorithm: 1. Compute the shortest non-zero vectors of \mathcal{L}_1 and \mathcal{L}_2

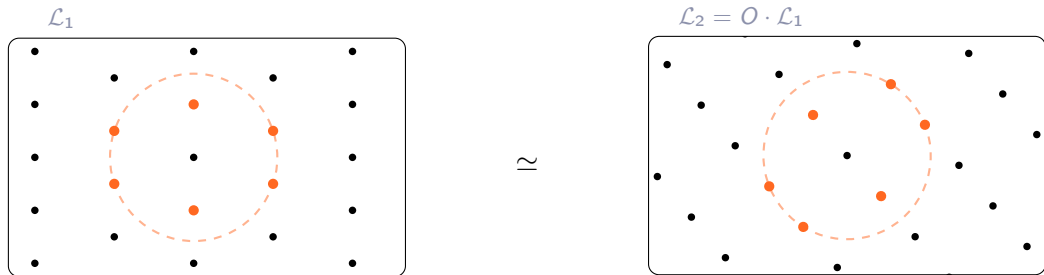


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

Algorithm: 1. Compute the **all shortish** vectors of \mathcal{L}_1 and \mathcal{L}_2

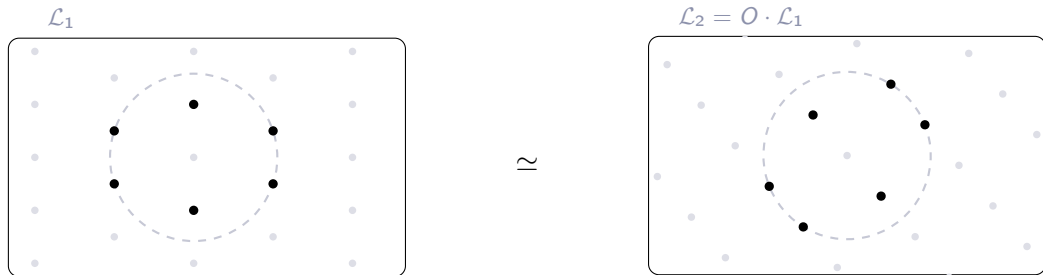


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

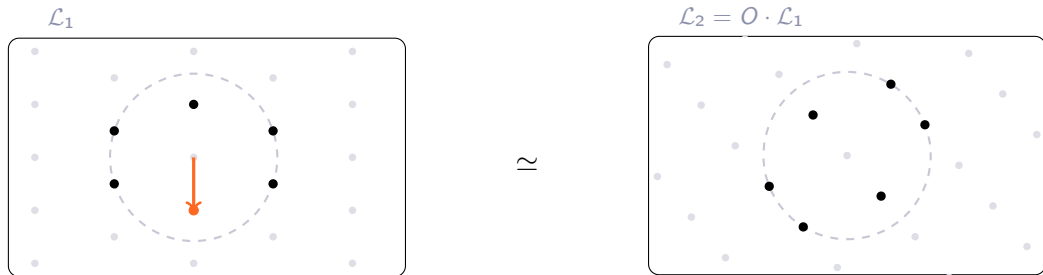


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

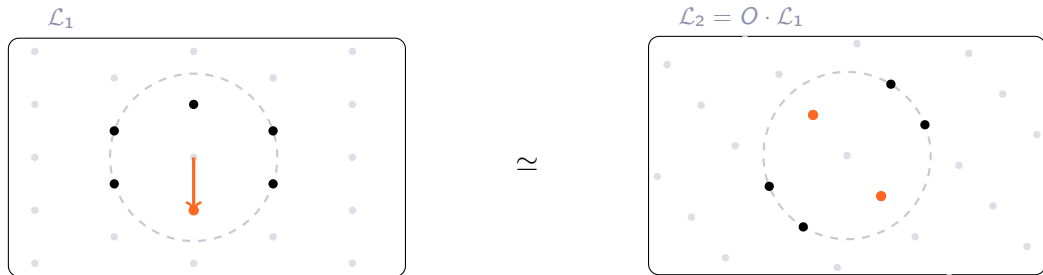


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

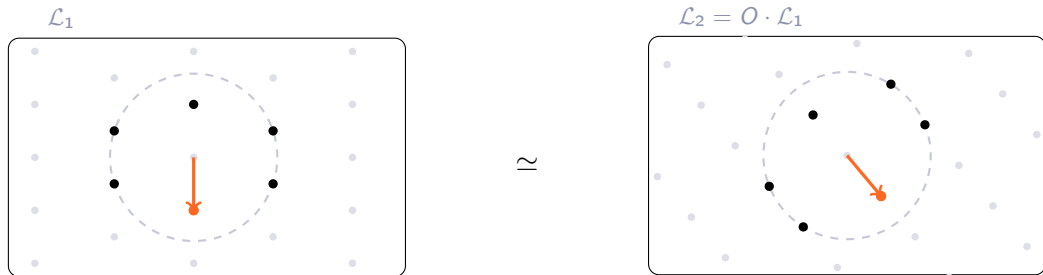


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

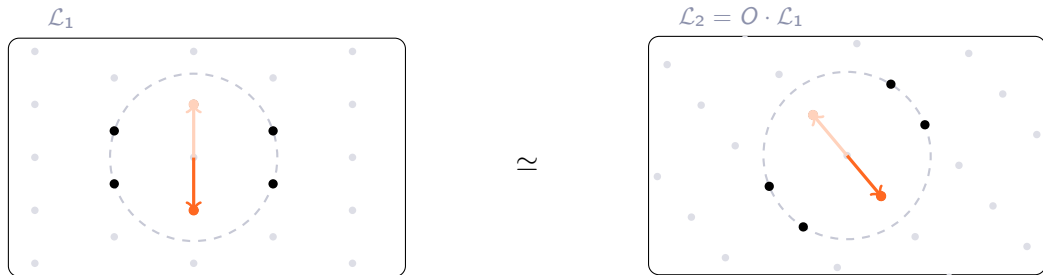


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

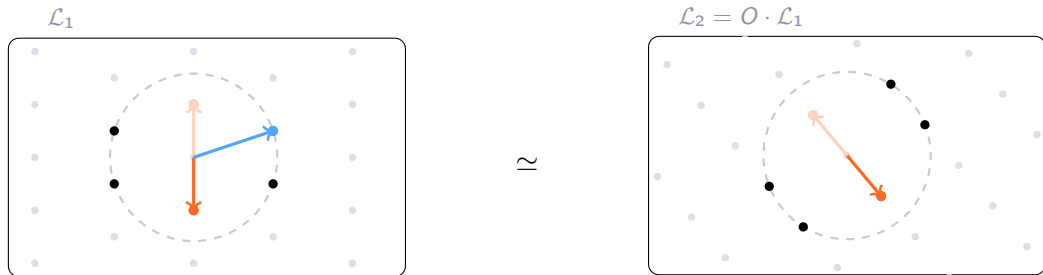


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

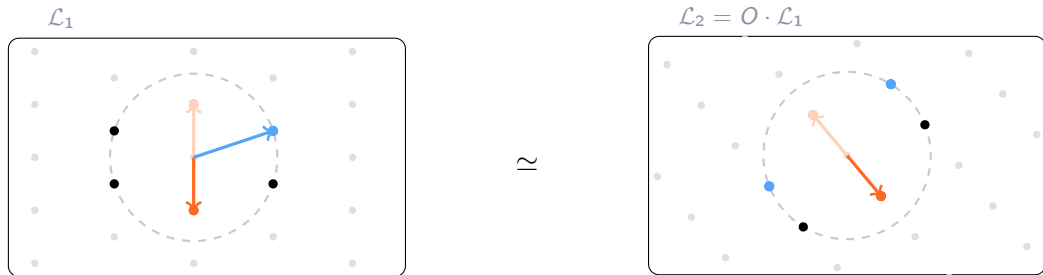


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

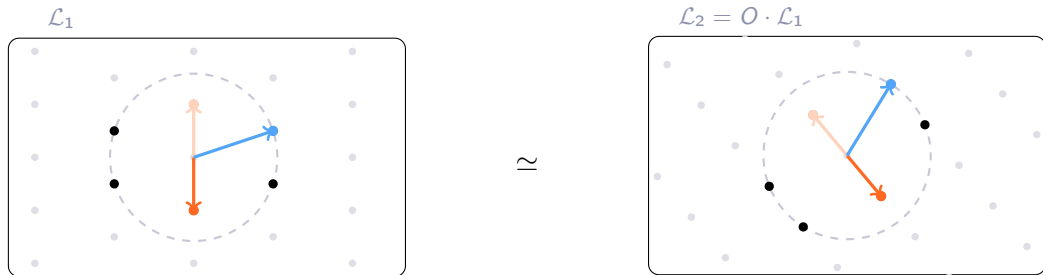


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

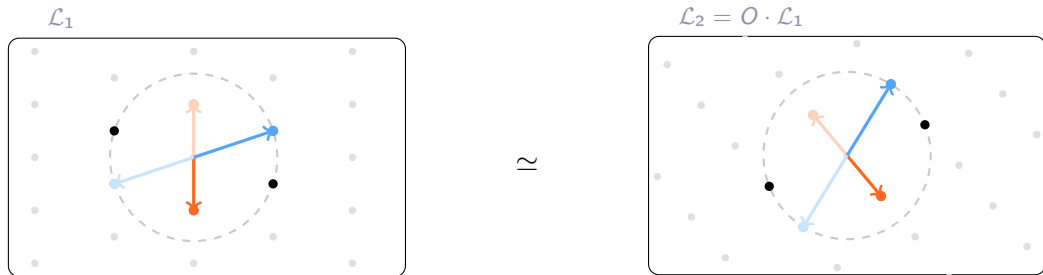


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

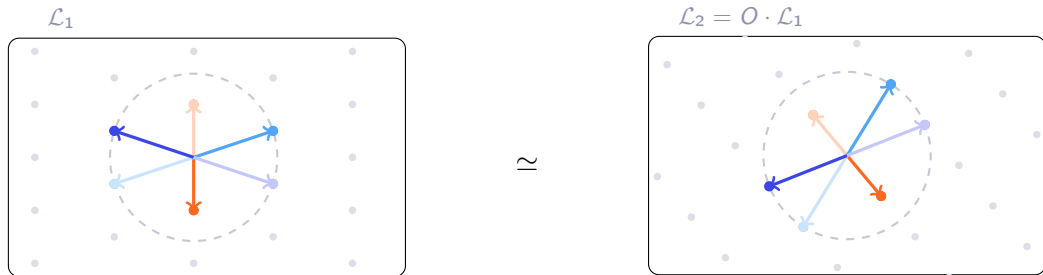


General case

Objective: given \mathcal{L}_1 and $\mathcal{L}_2 = O \cdot \mathcal{L}_1$, recover O ($O \in \mathcal{O}_n(\mathbb{R})$)

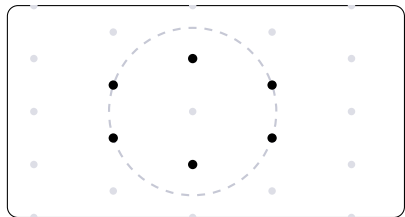
Remark: O is not unique (automorphisms of \mathcal{L}_1)

- Algorithm:**
1. Compute the all shortish vectors of \mathcal{L}_1 and \mathcal{L}_2
 2. Match the vectors in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)

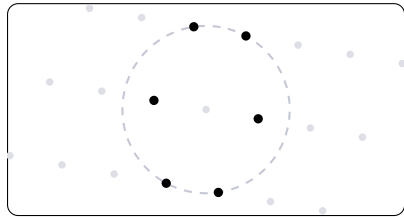


Matching vectors

Objective: given N vectors in \mathcal{L}_1 and \mathcal{L}_2 , match them in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)



\cong



Matching vectors

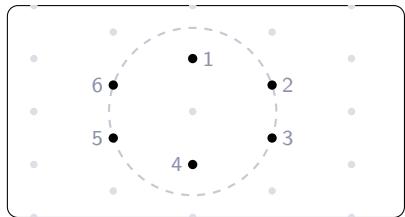
Objective: given N vectors in \mathcal{L}_1 and \mathcal{L}_2 , match them in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)



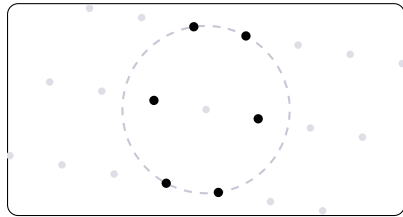
Idea: create weighted graph s.t. lattice isomorphism \Leftrightarrow graph isomorphism

Matching vectors

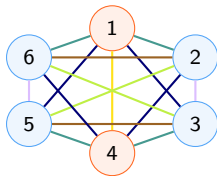
Objective: given N vectors in \mathcal{L}_1 and \mathcal{L}_2 , match them in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)



\approx



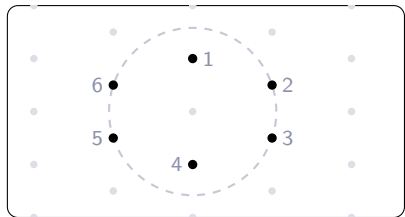
Idea: create weighted graph s.t. lattice isomorphism \iff graph isomorphism



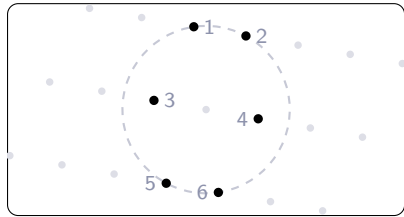
- ▶ N vertices (one per lattice vector)
- ▶ all edges (complete graph)
- ▶ vertex weight: $\|x\|_2$
- ▶ edge weight: $\langle x, y \rangle$

Matching vectors

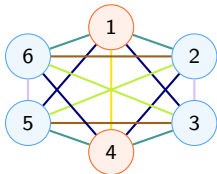
Objective: given N vectors in \mathcal{L}_1 and \mathcal{L}_2 , match them in a consistent way
(respecting $\|x\|_2$ and $\langle x, y \rangle$)



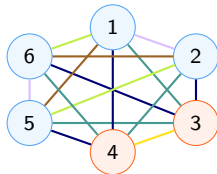
\cong



Idea: create weighted graph s.t. lattice isomorphism \iff graph isomorphism



\cong



Summing up and complexity

- Algorithm:
1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2 (until they generate the lattices)
 2. Create the complete graphs G_1 and G_2 with N vertices
 - ▶ weight $\|x\|_2$ on vertices
 - ▶ weight $\langle x, y \rangle$ on edges
 3. Solve graph isomorphism with (G_1, G_2)
 4. Recover lattice isomorphism from graph isomorphism

Summing up and complexity

- Algorithm:**
1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2 (until they generate the lattices)
 2. Create the complete graphs G_1 and G_2 with N vertices
 - ▶ weight $\|x\|_2$ on vertices
 - ▶ weight $\langle x, y \rangle$ on edges
 3. Solve graph isomorphism with (G_1, G_2)
 4. Recover lattice isomorphism from graph isomorphism

Analysis: Step 1. time complexity $2^{O(n)} \cdot N$ (n dimension of \mathcal{L}_1 and \mathcal{L}_2)

Summing up and complexity

- Algorithm:**
1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2 (until they generate the lattices)
 2. Create the complete graphs G_1 and G_2 with N vertices
 - ▶ weight $\|x\|_2$ on vertices
 - ▶ weight $\langle x, y \rangle$ on edges
 3. Solve graph isomorphism with (G_1, G_2)
 4. Recover lattice isomorphism from graph isomorphism

- Analysis:**
- Step 1.** time complexity $2^{O(n)} \cdot N$ (n dimension of \mathcal{L}_1 and \mathcal{L}_2)
- ▶ for “most” lattices: $N = \text{poly}(n)$
 - ▶ in bad cases: $N \approx 2^n$

Summing up and complexity

- Algorithm:**
1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2 (until they generate the lattices)
 2. Create the complete graphs G_1 and G_2 with N vertices
 - ▶ weight $\|x\|_2$ on vertices
 - ▶ weight $\langle x, y \rangle$ on edges
 3. Solve graph isomorphism with (G_1, G_2)
 4. Recover lattice isomorphism from graph isomorphism

- Analysis:**
- Step 1.** time complexity $2^{O(n)} \cdot N$ (n dimension of \mathcal{L}_1 and \mathcal{L}_2)
- ▶ for “most” lattices: $N = \text{poly}(n)$
 - ▶ in bad cases: $N \approx 2^n$
- Step 3.** time complexity $2^{(\log N)^{O(1)}}$ (quasi-polynomial) [Bab16]

Summing up and complexity

- Algorithm:**
1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2 (until they generate the lattices)
 2. Create the complete graphs G_1 and G_2 with N vertices
 - ▶ weight $\|x\|_2$ on vertices
 - ▶ weight $\langle x, y \rangle$ on edges
 3. Solve graph isomorphism with (G_1, G_2)
 4. Recover lattice isomorphism from graph isomorphism

- Analysis:**
- Step 1.** time complexity $2^{O(n)} \cdot N$ (n dimension of \mathcal{L}_1 and \mathcal{L}_2)
- ▶ for “most” lattices: $N = \text{poly}(n)$
 - ▶ in bad cases: $N \approx 2^n$
- Step 3.** time complexity $2^{(\log N)^{O(1)}}$ (quasi-polynomial) [Bab16]

Overall complexity: $2^{n^{O(1)}}$ (and $2^{O(n)}$ in “most” cases)

Main strategy:

1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2
2. Try to match the short vectors in a consistent way (that respects inner products)

Main strategy:

1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2
2. Try to match the short vectors in a consistent way (that respects inner products)

How to match the vectors:

[PS97] backtracking algorithm

- ▶ good in practice, but bad provable complexity

Main strategy:

1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2
2. Try to match the short vectors in a consistent way (that respects inner products)

How to match the vectors:

[PS97] backtracking algorithm

- ▶ good in practice, but bad provable complexity

[HR14] use dual lattice vectors to canonically order the N short vectors

- ▶ provable $n^{O(n)}$ complexity (best known so far)

[HR14] Haviv, Regev. On the lattice isomorphism problem. SODA.

Main strategy:

1. Compute N short vectors of \mathcal{L}_1 and \mathcal{L}_2
2. Try to match the short vectors in a consistent way (that respects inner products)

How to match the vectors:

[PS97] backtracking algorithm

- ▶ good in practice, but bad provable complexity

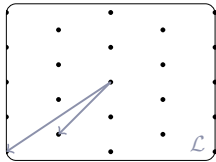
[HR14] use dual lattice vectors to canonically order the N short vectors

- ▶ provable $n^{O(n)}$ complexity (best known so far)

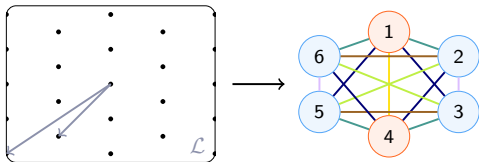
[SHVW20] graph isomorphism (this talk)

[SHVW20] Sikirić, Haensch, Voight, van Woerden. A canonical form for positive definite matrices. Open book series.

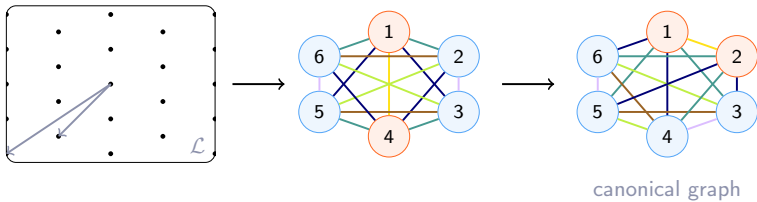
Bonus: canonical representation



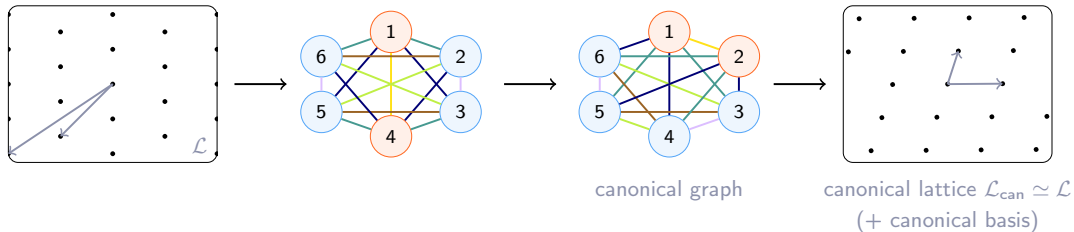
Bonus: canonical representation



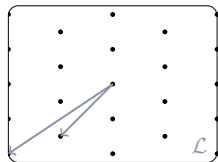
Bonus: canonical representation



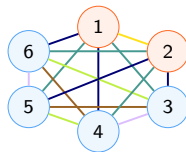
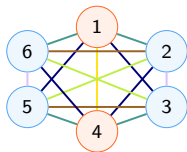
Bonus: canonical representation



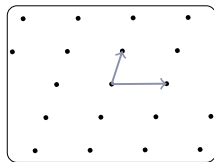
Bonus: canonical representation



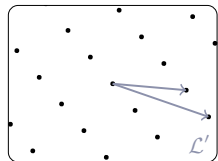
\mathcal{L}



canonical graph

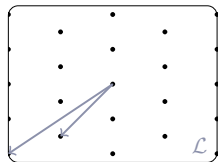


canonical lattice $\mathcal{L}_{\text{can}} \simeq \mathcal{L}$
(+ canonical basis)

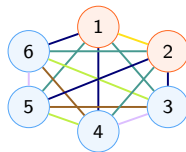
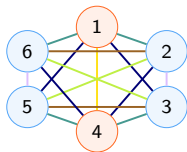


\mathcal{L}'

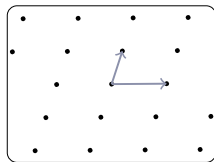
Bonus: canonical representation



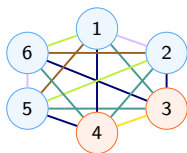
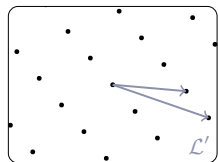
\mathcal{L}



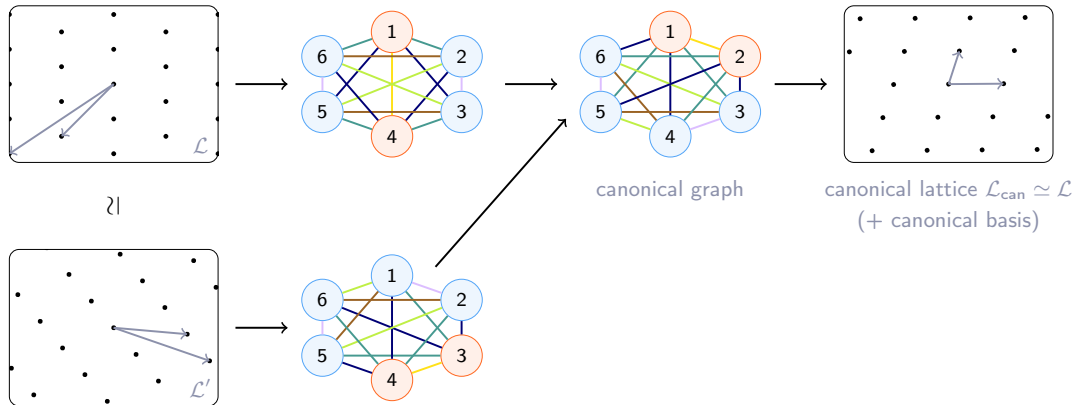
canonical graph



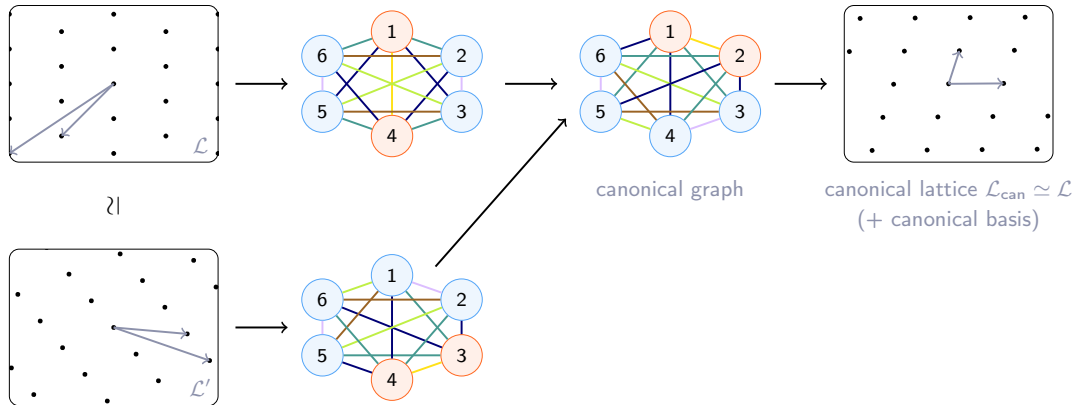
canonical lattice $\mathcal{L}_{\text{can}} \simeq \mathcal{L}$
(+ canonical basis)



Bonus: canonical representation



Bonus: canonical representation



Useful for: enumerating lattices up to isomorphism

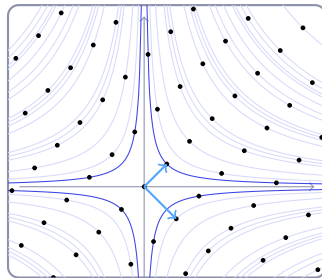
(e.g., enumerating all perfect lattices of dimension 9 [Woe25])

[Woe25] Wessel van Woerden, on going.

Algorithms for solving the lattice isomorphism problem

- ▶ try to match short vectors in a consistent way
- ▶ some variants rely on graph isomorphism
 - ▶ allows to construct a canonical lattice per isomorphism class
- ▶ complexity between $2^{O(n)}$ (average case) and $n^{O(n)}$ (worst case)
 - ▶ no efficient algorithm when n is large (e.g., $n = 700$)

Conclusion



What happens if: we replace integers by polynomials?

$$\mathbb{Z} \longleftrightarrow \mathbb{Z}[X]/P(X) \quad (P \text{ irreducible})$$

Example: lattice basis of $\dim 2 \times 2$ over $\mathbb{Z}[X]/(X^{512} + 1)$
(dimension 1024 over \mathbb{Z})

I used beamer theme begles (<https://framagit.org/squirrrr/beamerthemebegles>)

Thanks to Wessel van Woerden for sharing his slides and answering my questions.

What happens if: we replace integers by polynomials?

$$\mathbb{Z} \longleftrightarrow \mathbb{Z}[X]/P(X) \quad (P \text{ irreducible})$$

Example: lattice basis of $\dim 2 \times 2$ over $\mathbb{Z}[X]/(X^{512} + 1)$
(dimension 1024 over \mathbb{Z})

Crypto constructions:

- ▶ more efficient in time and space

I used beamer theme begles (<https://framagit.org/squirrrr/beamerthemebegles>)

Thanks to Wessel van Woerden for sharing his slides and answering my questions.

What happens if: we replace integers by polynomials?

$$\mathbb{Z} \longleftrightarrow \mathbb{Z}[X]/P(X) \quad (P \text{ irreducible})$$

Example: lattice basis of $\dim 2 \times 2$ over $\mathbb{Z}[X]/(X^{512} + 1)$
(dimension 1024 over \mathbb{Z})

Crypto constructions:

- ▶ more efficient in time and space
- ▶ is security still ok?

I used beamer theme begles (<https://framagit.org/squirrrr/beamerthemebegles>)

Thanks to Wessel van Woerden for sharing his slides and answering my questions.

What happens if: we replace integers by polynomials?

$$\mathbb{Z} \longleftrightarrow \mathbb{Z}[X]/P(X) \quad (P \text{ irreducible})$$

Example: lattice basis of $\dim 2 \times 2$ over $\mathbb{Z}[X]/(X^{512} + 1)$
(dimension 1024 over \mathbb{Z})

Crypto constructions:

- ▶ more efficient in time and space
- ▶ is security still ok?
 - ▶ in some cases it is not! [MPPW24,APW25]

[MPPW24] Mureau, Pellet-Mary, Pliatsok, Wallet. Cryptanalysis of rank-2 module-LIP in Totally Real Number Fields. Eurocrypt.

[APW25] Allombert, Pellet-Mary, van Woerden. Cryptanalysis of rank-2 module-LIP: a single real embedding is all it takes. Eurocrypt.

What happens if: we replace integers by polynomials?

$$\mathbb{Z} \longleftrightarrow \mathbb{Z}[X]/P(X) \quad (P \text{ irreducible})$$

Example: lattice basis of $\dim 2 \times 2$ over $\mathbb{Z}[X]/(X^{512} + 1)$
(dimension 1024 over \mathbb{Z})

Crypto constructions:

- ▶ more efficient in time and space
- ▶ is security still ok?
 - ▶ in some cases it is not! [MPPW24,APW25]

Thank you

I used beamer theme begles (<https://framagit.org/squirrrr/beamerthemebegles>)

Thanks to Wessel van Woerden for sharing his slides and answering my questions.