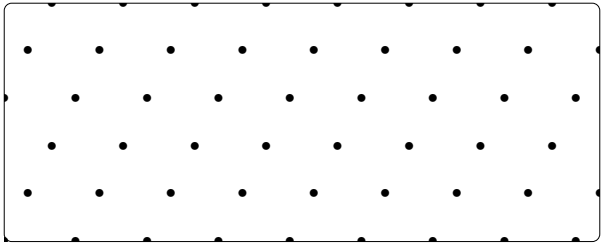
How to build cryptography from lattices

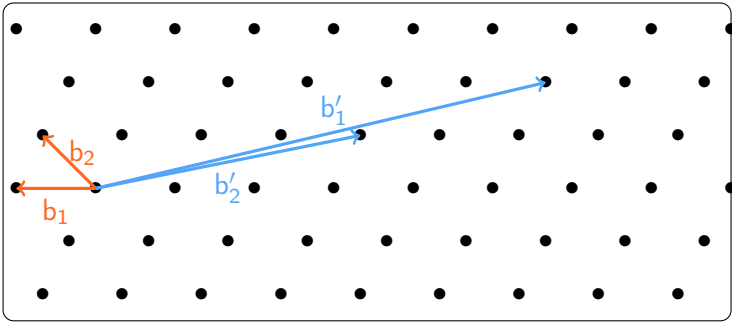
Alice Pellet-Mary

CAIPI Symposium, Limoges

Lattices and lattice problems



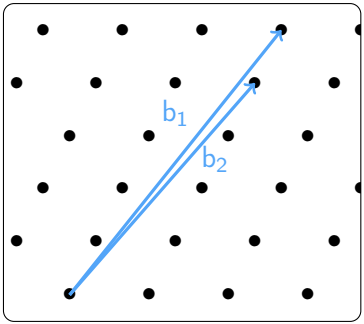
Lattices



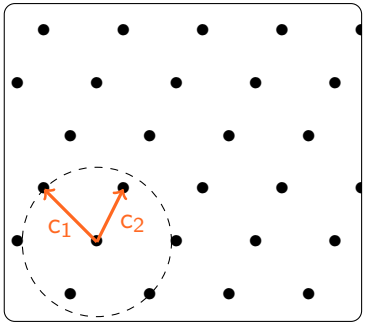
- $\mathcal{L} = \{\sum_{i=1}^{n} x_i \mathbf{b}_i \mid \forall i,\ x_i \in \mathbb{Z}\}$ is a lattice
- $(\mathbf{b}_1, \ldots, \mathbf{b}_n) =: B \in \mathrm{GL}_n(\mathbb{R})$ is a basis (not unique)
- n is the dimension (or rank)

Short basis problem

Input:



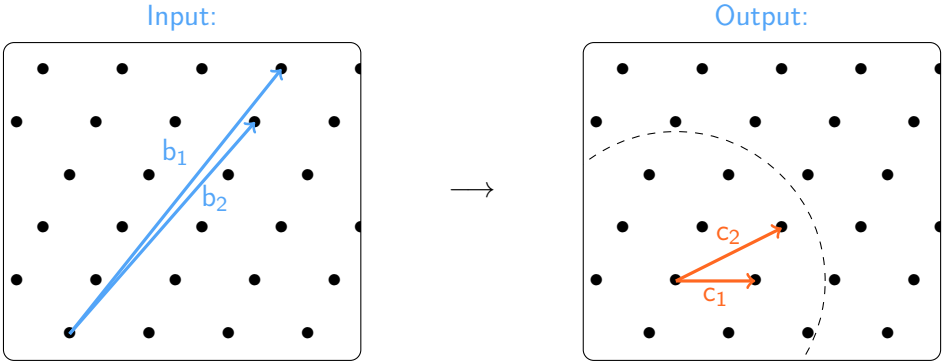
Output:



Shortest basis problem

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

Short basis problem

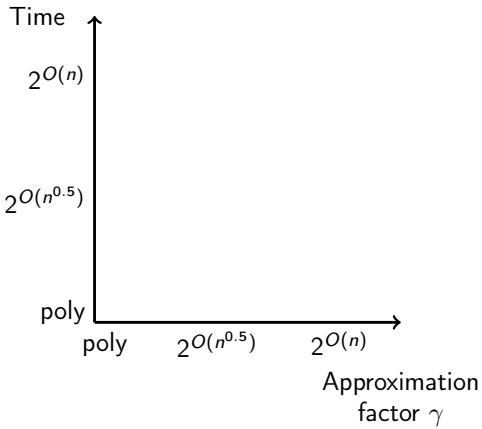


Approximate short basis problem

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

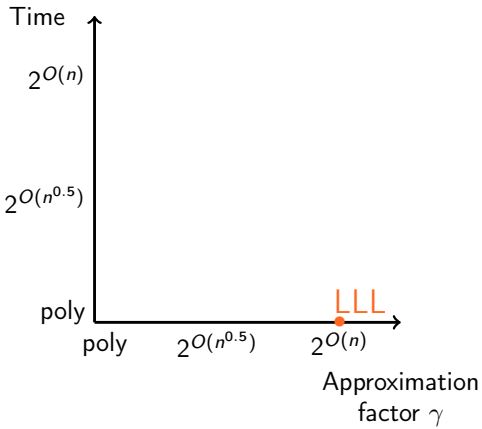
Asymptotic hardness

Best known asymptotic algorithms for the γ-short basis problem (classical and quantum)



Asymptotic hardness

Best known asymptotic algorithms for the γ-short basis problem (classical and quantum)

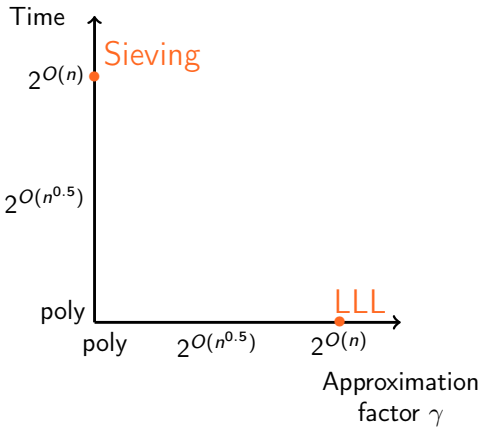


LLL algorithm: dim n

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

Asymptotic hardness

Best known asymptotic algorithms for the γ-short basis problem (classical and quantum)



LLL algorithm: dim n

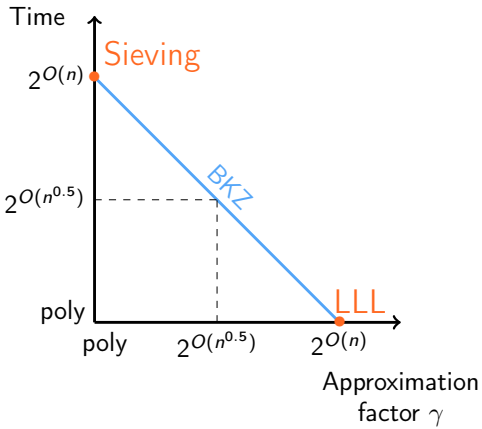
- γ-short basis with $\gamma = 2^n$
- polynomial time

Sieving/enumeration algorithm: dim n

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

Asymptotic hardness

Best known asymptotic algorithms for the γ-short basis problem (classical and quantum)



LLL algorithm: dim n

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

Sieving/enumeration algorithm: dim n

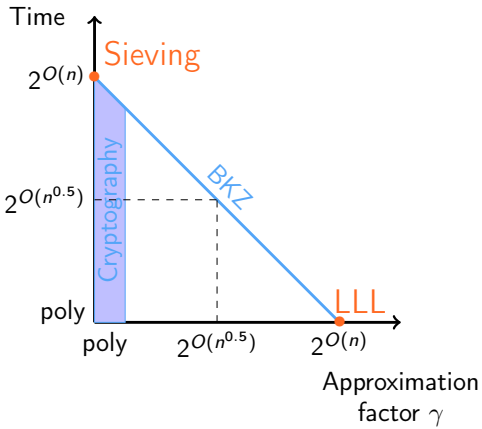
- shortest basis
- time $2^{O(n)}$

BKZ algorithm: combine LLL + Sieving

$\Rightarrow$ various trade-offs

Asymptotic hardness

Best known asymptotic algorithms for the γ-short basis problem (classical and quantum)



LLL algorithm: dim n

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

Sieving/enumeration algorithm: dim n

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

BKZ algorithm: combine LLL + Sieving

$\Rightarrow$ various trade-offs

Finding a shortest basis in practice ($\gamma = 1$):

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

Finding a shortest basis in practice ($\gamma = 1$):

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

Finding a shortest basis in practice ($\gamma = 1$):

- $n = 2$ $\rightsquigarrow$ 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.

Finding a shortest basis in practice ($\gamma = 1$):

- $n = 2$ $\rightsquigarrow$ 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 seems 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 seems 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 seems 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 seems 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!

Section's conclusion

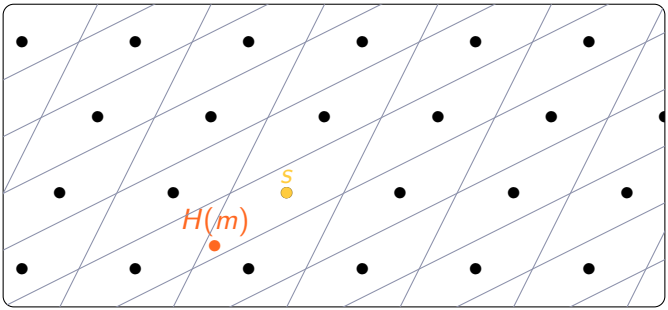
Takeaway: computing short bases of lattices seems very hard (even quantumly)

- 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!
- For crypto we want to generate random hard lattices
  (We will come back to this...)

Constructing signatures from lattices



Digital Signature

Three algorithms:

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

Digital Signature

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

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

Digital Signature

Three algorithms:
- KeyGen() $\rightsquigarrow (pk, sk)$
- Sign$(m, sk) \rightsquigarrow s$
- 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:
- KeyGen() $\rightsquigarrow (pk, sk)$
- Sign$(m, sk) \rightsquigarrow s$
- 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

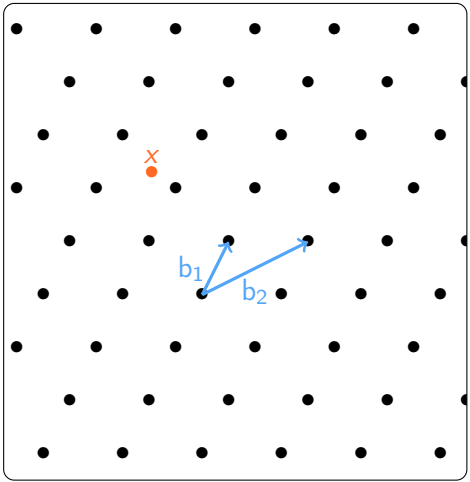
Three algorithms:
- KeyGen() $\rightsquigarrow (pk, sk)$
- Sign$(m, sk) \rightsquigarrow s$
- Verify$(m, s, pk) \rightsquigarrow$ yes/no

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

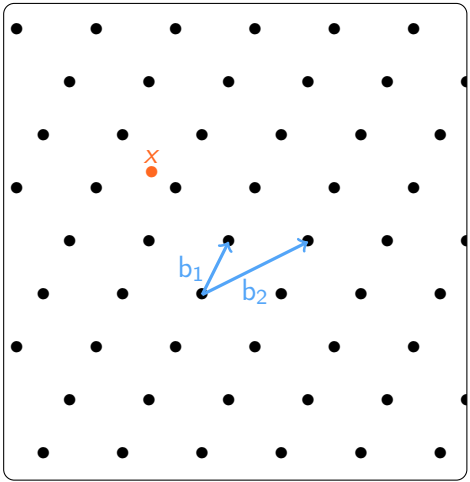
Finding a close vector using a short basis



Input: $x = 3.7 \cdot \mathbf{b}_1 - 1.4 \cdot \mathbf{b}_2$

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

Finding a close vector using a short basis

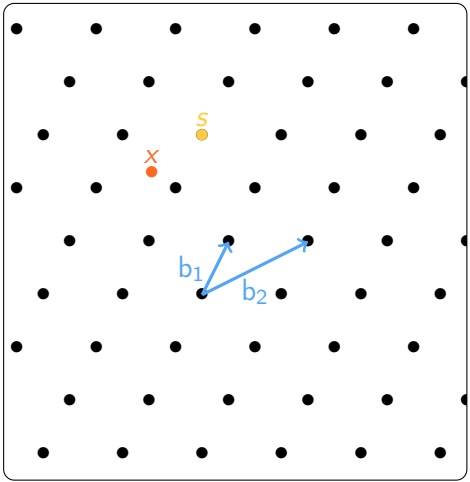


Input: $x = 3.7 \cdot \mathbf{b}_1 - 1.4 \cdot \mathbf{b}_2$

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

Algo: round each coordinate

Finding a close vector using a short basis



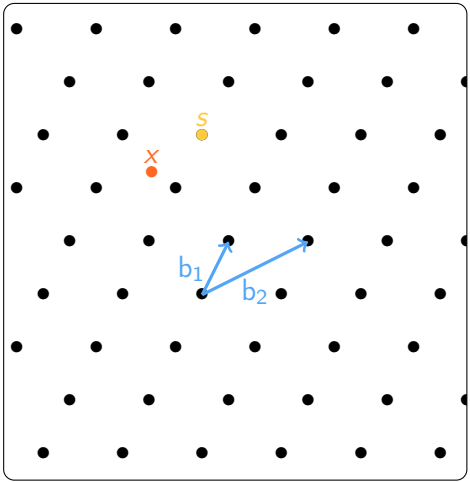
Input: $x = 3.7 \cdot \mathbf{b}_1 - 1.4 \cdot \mathbf{b}_2$

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

Algo: round each coordinate

Output: $s = 4 \cdot \mathbf{b}_1 - 1 \cdot \mathbf{b}_2$

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

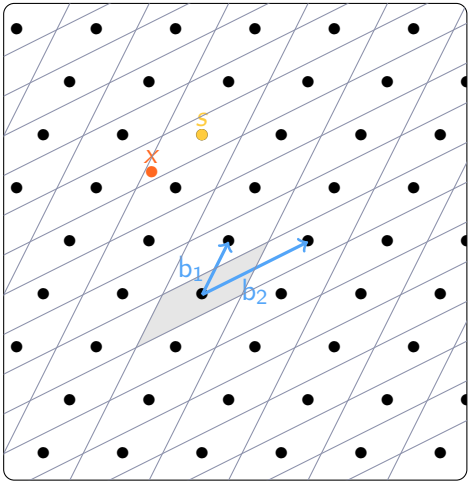
Algo: round each coordinate

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

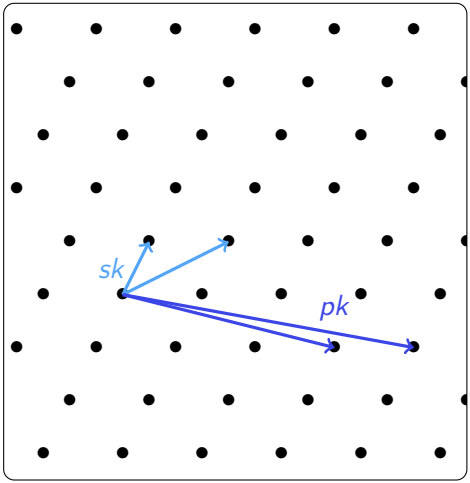
Output: $s = 4 \cdot b_1 - 1 \cdot b_2$

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 lattice signature [GGH97]

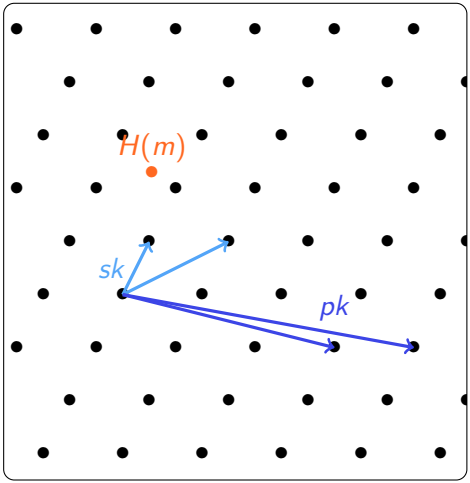


KeyGen:

- pk = bad basis of $\mathcal{L}$
- sk = short basis of $\mathcal{L}$

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

Hash-and-sign lattice signature [GGH97]



KeyGen:

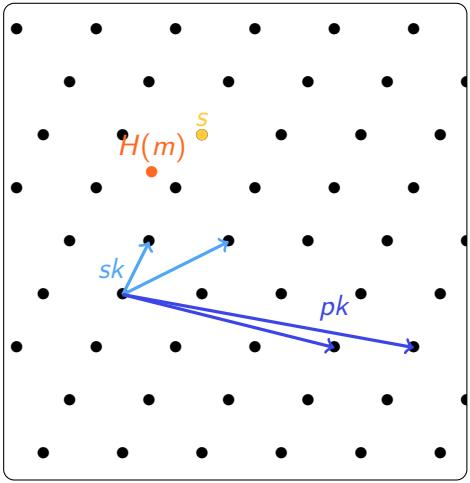
- pk = bad basis of $\mathcal{L}$
- sk = short basis of $\mathcal{L}$

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 lattice signature [GGH97]



KeyGen:

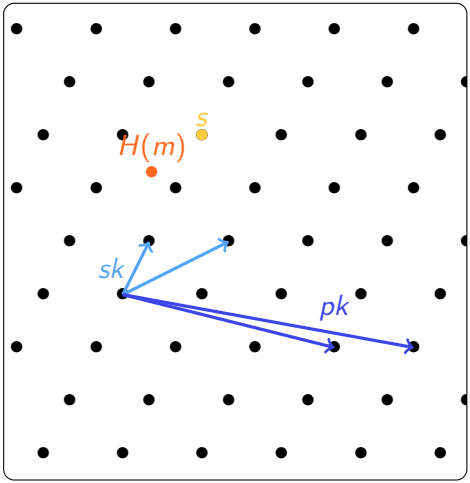
- pk = bad basis of $\mathcal{L}$
- sk = short basis of $\mathcal{L}$

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 lattice signature [GGH97]



KeyGen:

- pk = bad basis of $\mathcal{L}$
- 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

Warning: The scheme from previous slide is insecure [NR06] but it can be fixed with a small change [GPV08]

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

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

Warning: The scheme from previous slide is insecure [NR06] but it can be fixed with a small change [GPV08]

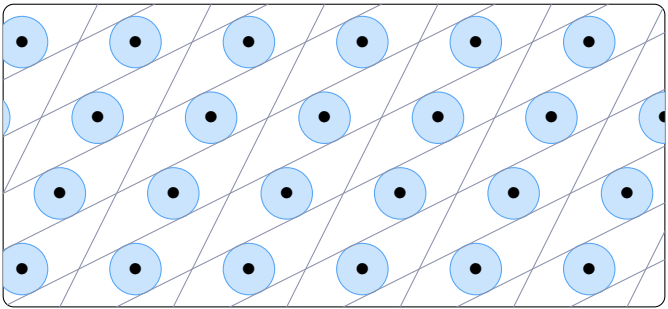
Hash-and-sign signature scheme:

- requires a lattice $\mathcal{L}$ + a short basis $\mathbf{B}_s$ + a bad basis $\mathbf{B}_p$
- provably secure if recovering a short basis from $\mathbf{B}_p$ is hard

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

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

Constructing public key encryption from lattices



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

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

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

Public key encryption (PKE)

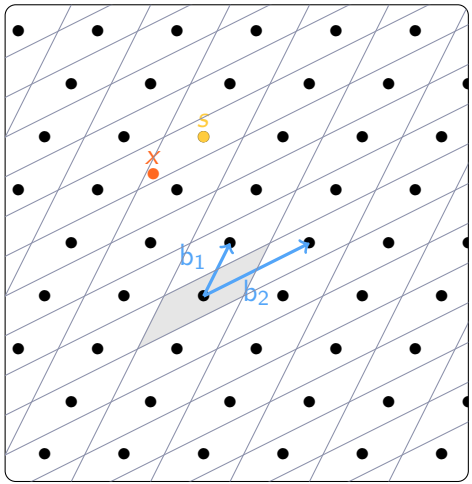
Three algorithms:
- $\mathrm{KeyGen}() \rightsquigarrow (pk, sk)$
- $\mathrm{Enc}(m, pk) \rightsquigarrow c$
- $\mathrm{Dec}(c, sk) \rightsquigarrow m'$

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: $\mathrm{Dec}\big(\mathrm{Enc}(m, pk), sk\big) = m$

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

Remember Babai decoding

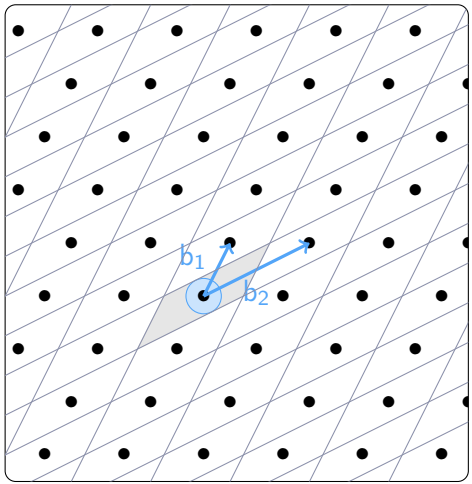


$\texttt{Babai}_B(x) = s \quad (\in \mathcal{L}, \text{ close to } x)$

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

(fundamental parallelepiped)

Remember Babai decoding



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

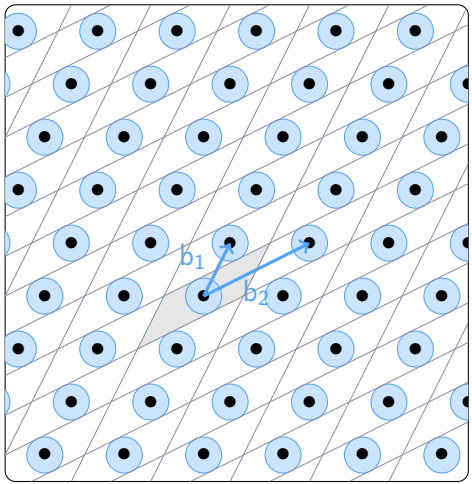
$= \left\{ x_1 b_1 + x_2 b_2 \,\middle|\, |x_i| \leq \frac{1}{2} \right\}$

(fundamental parallelepiped)

● = largest circle $\subseteq$

(radius of ●: decoding radius)

Remember Babai decoding



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

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

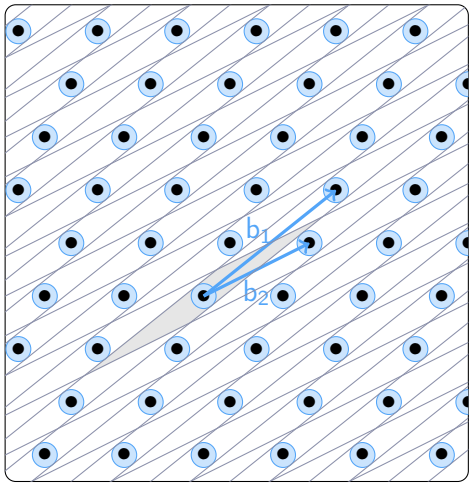
(fundamental parallelepiped)

● = largest circle $\subseteq$

(radius of ●: decoding radius)

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

Remember Babai decoding



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

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

(fundamental parallelepiped)

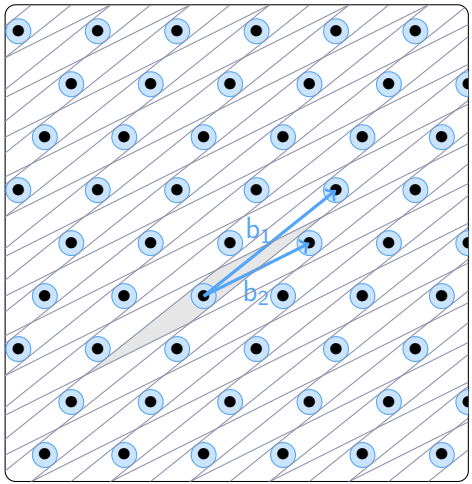
● = largest circle $\subseteq$ ▱

(radius of ●: decoding radius)

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

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

Remember Babai decoding



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

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

(fundamental parallelepiped)

● = largest circle $\subseteq$ ▱

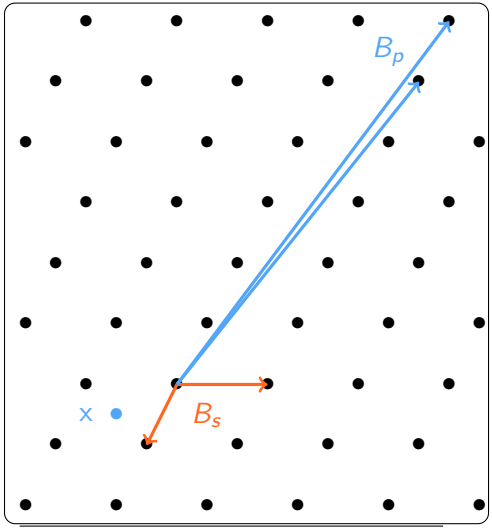
(radius of ●: decoding radius)

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

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

Smaller basis $\iff$ larger ●

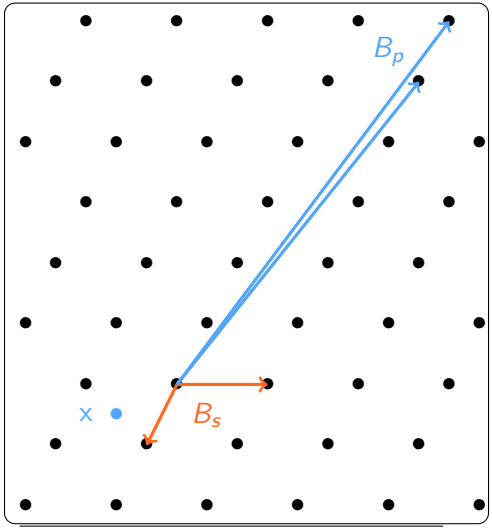
Public key encryption from lattices [Reg05]



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

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

Public key encryption from lattices [Reg05]

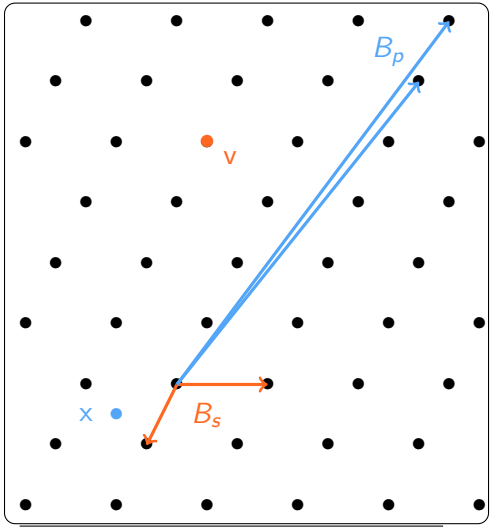


$\mathrm{pk} = (B_p, \mathrm{x})$
$\mathrm{sk} = B_s$

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

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

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, \mathrm{x})$ message: $m \in \{0, 1\}$

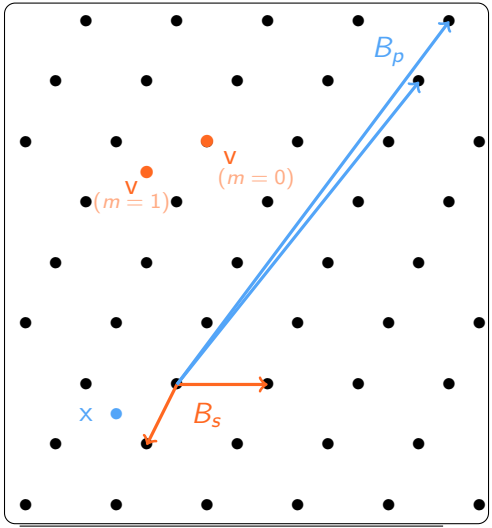
$\mathrm{sk} = B_s$

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

- Sample random $\mathrm{v} \in L$

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

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, \mathsf{x})$ message: $m \in \{0, 1\}$

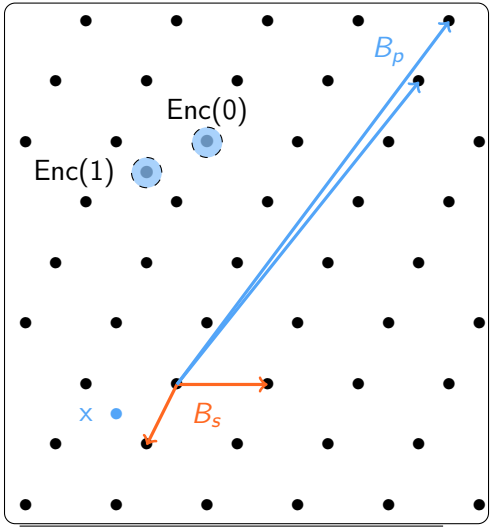
$\mathrm{sk} = B_s$

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

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

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

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, \mathrm{x})$ message: $m \in \{0, 1\}$

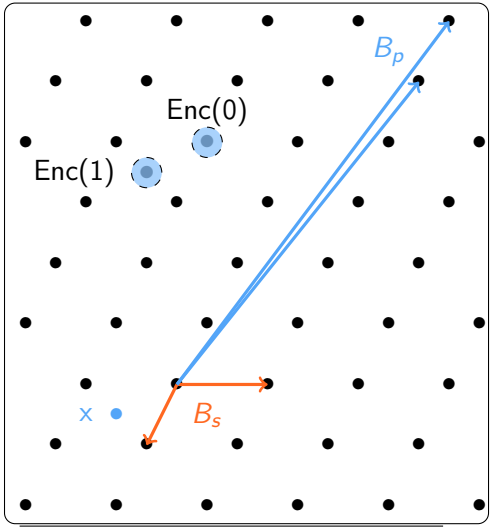
$\mathrm{sk} = B_s$

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

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

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

Public key encryption from lattices [Reg05]



$\mathrm{pk} = (B_p, \mathrm{x})$ message: $m \in \{0, 1\}$

$\mathrm{sk} = B_s$

Enc(m, pk):

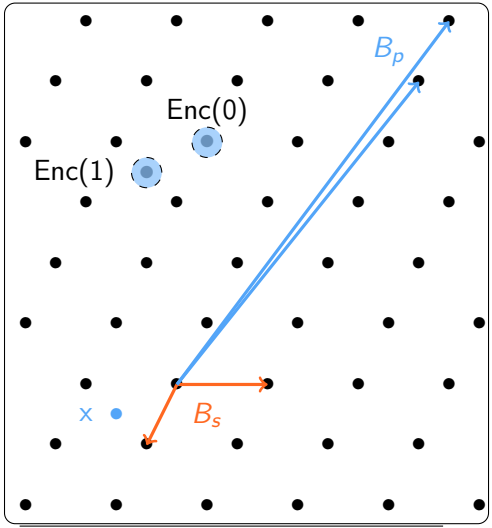
- Sample random $\mathrm{v} \in L$
- if $m = 1$: $\mathrm{v} \leftarrow \mathrm{v} + \mathrm{x}$
- Sample small $\mathrm{e} \in$ ◯
- return $\mathrm{c} = \mathrm{v} + \mathrm{e}$

Dec(c, sk):

- $\mathrm{s} \leftarrow \mathrm{Babai}_{sk}(\mathrm{c})$
- if $\|\mathrm{s} - \mathrm{c}\|$ 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]



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

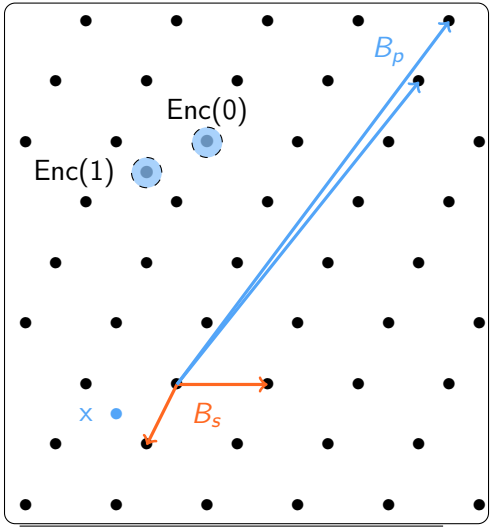
message: $m \in \{0, 1\}$

Correctness:

- if $m = 1$

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

Public key encryption from lattices [Reg05]



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

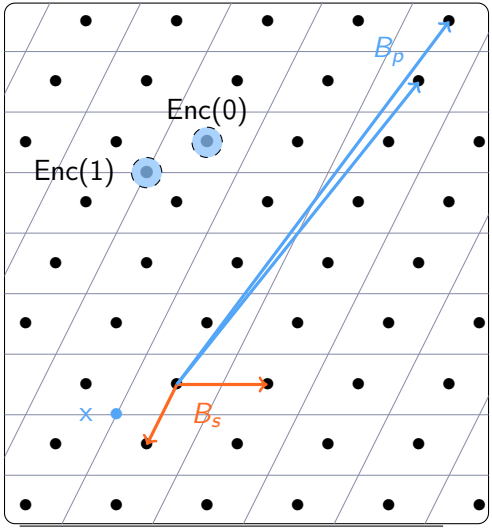
message: $m \in \{0, 1\}$

Correctness:

- if $m = 1$ ✓

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

Public key encryption from lattices [Reg05]



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

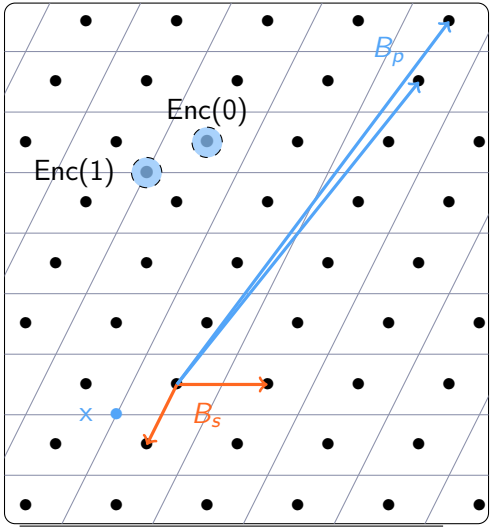
message: $m \in \{0, 1\}$

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]



$\mathrm{pk} = (B_p, \mathrm{x})$ message: $m \in \{0, 1\}$

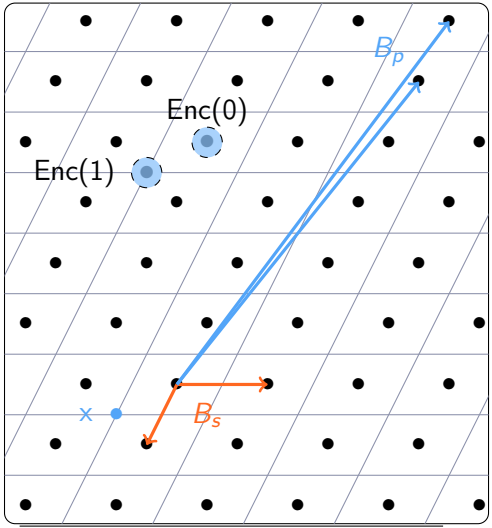
$\mathrm{sk} = B_s$

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]



$\mathrm{pk} = (B_p, \mathrm{x})$ message: $m \in \{0, 1\}$

$\mathrm{sk} = B_s$

Correctness:

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

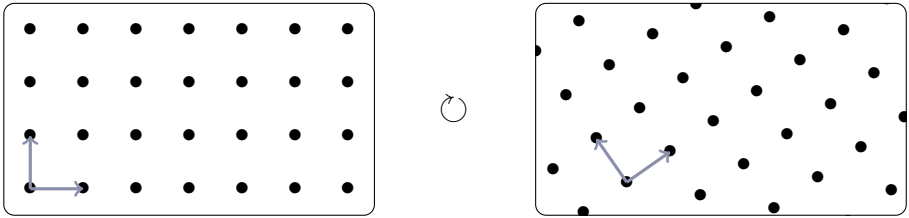
Security : ✓ if $\mathcal{L}$ is well chosen

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

Regev 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

The lattice isomorphism problem



Objective

What we want: An algorithm `KeyGen` such that

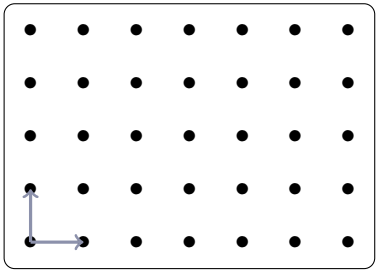
- `KeyGen` computes
  - a random lattice $\mathcal{L}$
  - a short basis B_s of $\mathcal{L}$ (sk)
  - a bad basis B_p of $\mathcal{L}$ (pk)
  - (optional) a point x far from $\mathcal{L}$

Objective

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 bad basis B_p of $\mathcal{L}$ (pk)
  - (optional) a point x far from $\mathcal{L}$
- computing a short basis of $\mathcal{L}$ from B_p is hard with overwhelming probability

The lattice isomorphism problem [DW22,BGPS23]

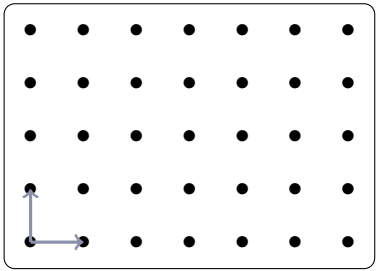


$$\mathcal{L}_0 = \mathbb{Z}^n$$

[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

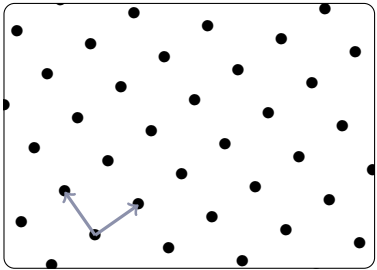
The lattice isomorphism problem [DW22,BGPS23]



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

rotate

(choose O orthogonal matrix)

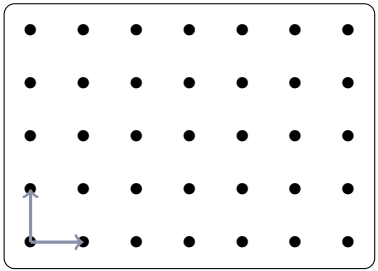


$\mathcal{L} = O \cdot \mathbb{Z}^n$

[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

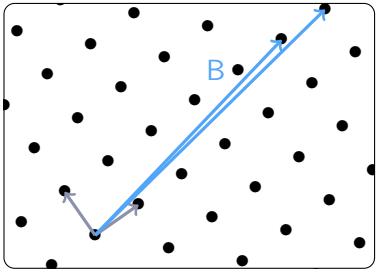
The lattice isomorphism problem [DW22,BGPS23]



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

rotate

(choose O orthogonal matrix)



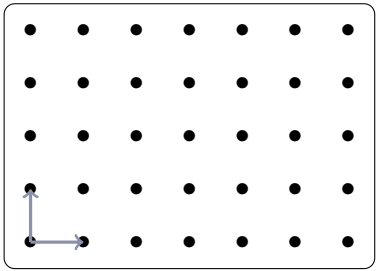
$\mathcal{L} = O \cdot \mathbb{Z}^n$

B bad basis of $\mathcal{L}$

[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

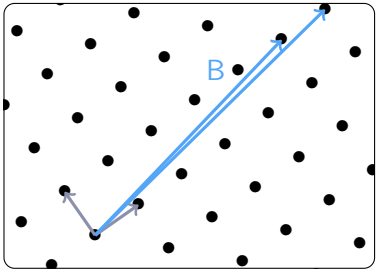
The lattice isomorphism problem [DW22,BGPS23]



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

rotate

(choose O orthogonal matrix)



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

B bad basis of $\mathcal{L}$

Lattice Isomorphism Problem (LIP) assumption

recovering O from B is hard

$\Leftrightarrow$ computing a shortest basis of $\mathcal{L}$ is hard

Equivalent formulation with Gram matrices

Lattice isomorphism problem: Given $B = O \cdot C$ with

- $O \in O_n(\mathbb{R})$ orthogonal
- C a basis of $\mathbb{Z}^n$

Find O (equivalently: find C)

Equivalent formulation with Gram matrices

Lattice isomorphism problem: Given $B = O \cdot C$ with
- $O \in O_n(\mathbb{R})$ orthogonal
- C a basis of $\mathbb{Z}^n$

Find O (equivalently: find C)

Gram matrix associated to B: $G = B^T B = C^T (O^T O) C = C^T C$

$\Rightarrow$ O has disappeared

Equivalent formulation with Gram matrices

Lattice isomorphism problem: Given $B = O \cdot C$ with
- $O \in O_n(\mathbb{R})$ orthogonal
- C a basis of $\mathbb{Z}^n$

Find O (equivalently: find C)

Gram matrix associated to B: $G = B^T B = C^T (O^T O) C = C^T C$

$\Rightarrow$ O has disappeared

Lattice isomorphism problem (Gram matrix formulation):
Given $G = C^T C$ with C a (secret) basis of $\mathbb{Z}^n$, find C.

Section's conclusion

The lattice isomorphism problem (LIP):

- allows to generate random lattices
- where the short basis problem is believed to be hard to solve
- together with a short secret basis (and a point x far from the lattice)

We can use LIP to build public key encryption and signatures [DW22,BGPS23,DPPW23]

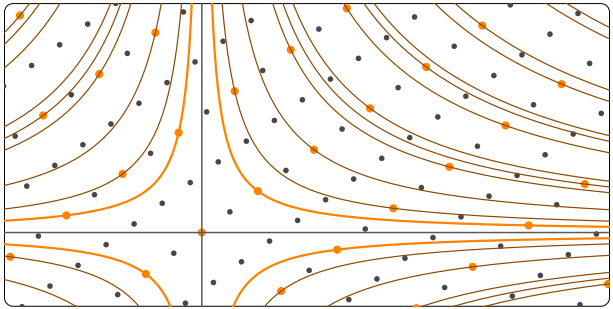
(There exists many other problems that provide (other distributions of) random hard lattices: LWE, SIS, module-LWE, module-SIS, NTRU, ...)

[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

[DPPW23] Ducas, Postlethwaite, Pulles, van Woerden. Hawk: Module LIP makes lattice signatures Fast, Compact and Simple. Asiacrypt

Module lattices



Number fields

Number field: $K = \mathbb{Q}[X]/P(X)$ (P irreducible, $\deg(P) = d$)

Number fields

Number field: $K = \mathbb{Q}[X]/P(X)$ (P irreducible, $\deg(P) = d$)

- $K = \mathbb{Q}$
- $K = \mathbb{Q}[X]/(X^d + 1)$ with $d = 2^\ell$ $\rightsquigarrow$ power-of-two cyclotomic field
- $K = \mathbb{Q}[X]/(X^d - X - 1)$ with d prime $\rightsquigarrow$ NTRUPrime field

Number fields

Number field: $K = \mathbb{Q}[X]/P(X)$ (P irreducible, $\deg(P) = d$)

- $K = \mathbb{Q}$
- $K = \mathbb{Q}[X]/(X^d + 1)$ with $d = 2^\ell \rightsquigarrow$ power-of-two cyclotomic field
- $K = \mathbb{Q}[X]/(X^d - X - 1)$ with d prime $\rightsquigarrow$ NTRUPrime field

Ring of integers: $\mathcal{O}_K \subset K$, for this talk $\mathcal{O}_K = \mathbb{Z}[X]/P(X)$

(more generally $\mathbb{Z}[X]/P(X) \subseteq \mathcal{O}_K$ but $\mathcal{O}_K$ can be larger)

Number fields

Number field: $K = \mathbb{Q}[X]/P(X)$ (P irreducible, $\deg(P) = d$)

- $K = \mathbb{Q}$
- $K = \mathbb{Q}[X]/(X^d + 1)$ with $d = 2^\ell \rightsquigarrow$ power-of-two cyclotomic field
- $K = \mathbb{Q}[X]/(X^d - X - 1)$ with d prime $\rightsquigarrow$ NTRUPrime field

Ring of integers: $\mathcal{O}_K \subset K$, for this talk $\mathcal{O}_K = \mathbb{Z}[X]/P(X)$

(more generally $\mathbb{Z}[X]/P(X) \subseteq \mathcal{O}_K$ but $\mathcal{O}_K$ can be larger)

- $\mathcal{O}_K = \mathbb{Z}$
- $\mathcal{O}_K = \mathbb{Z}[X]/(X^d + 1)$ with $d = 2^\ell \rightsquigarrow$ power-of-two cyclotomic ring
- $\mathcal{O}_K = \mathbb{Z}[X]/(X^d - X - 1)$ with d prime $\rightsquigarrow$ NTRUPrime ring of integers

The canonical embedding

$(K = \mathbb{Q}[X]/P(X), \quad \alpha_1, \cdots, \alpha_d$ complex roots of $P(X))$

Canonical embedding: $\sigma : \begin{array}{rcl} K & \to & \mathbb{C}^d \\ a(X) & \mapsto & (a(\alpha_1), \cdots, a(\alpha_d)) \end{array}$

The canonical embedding

($K = \mathbb{Q}[X]/P(X)$, $\alpha_1, \cdots, \alpha_d$ complex roots of $P(X)$)

Canonical embedding: $\sigma : \begin{array}{rcl} K & \to & \mathbb{C}^d \\ a(X) & \mapsto & (a(\alpha_1), \cdots, a(\alpha_d)) \end{array}$

Via σ we can see K as a subset of $\mathbb{C}^d$

- $K \simeq \sigma(K) \subset \mathbb{C}^d$ (σ is injective)

The canonical embedding

($K = \mathbb{Q}[X]/P(X)$, $\alpha_1, \cdots, \alpha_d$ complex roots of $P(X)$)

Canonical embedding: $\sigma : \begin{array}{rcl} K & \to & \mathbb{C}^d \\ a(X) & \mapsto & (a(\alpha_1), \cdots, a(\alpha_d)) \end{array}$

Via σ we can see K as a subset of $\mathbb{C}^d$

- $K \simeq \sigma(K) \subset \mathbb{C}^d$ (σ is injective)
- addition and multiplication in $\mathbb{C}^d$ are performed coefficient-wise

The canonical embedding

($K = \mathbb{Q}[X]/P(X)$, $\alpha_1, \cdots, \alpha_d$ complex roots of $P(X)$)

Canonical embedding: $\sigma : \begin{array}{ccc} K & \to & \mathbb{C}^d \\ a(X) & \mapsto & (a(\alpha_1), \cdots, a(\alpha_d)) \end{array}$

Via σ we can see K as a subset of $\mathbb{C}^d$

- $K \simeq \sigma(K) \subset \mathbb{C}^d$ (σ is injective)
- addition and multiplication in $\mathbb{C}^d$ are performed coefficient-wise
- this induces a geometry on K (using the hermitian norm in $\mathbb{C}^d$): for $a \in K$

$$\|a\| := \|\sigma(a)\|_2 = \sqrt{\sum_i |a(\alpha_i)|^2}.$$

The canonical embedding

($K = \mathbb{Q}[X]/P(X)$, $\alpha_1, \cdots, \alpha_d$ complex roots of $P(X)$)

Canonical embedding: $\sigma : \begin{array}{rcl} K & \to & \mathbb{C}^d \\ a(X) & \mapsto & (a(\alpha_1), \cdots, a(\alpha_d)) \end{array}$

Via σ we can see K as a subset of $\mathbb{C}^d$

- $K \simeq \sigma(K) \subset \mathbb{C}^d$ (σ is injective)
- addition and multiplication in $\mathbb{C}^d$ are performed coefficient-wise
- this induces a geometry on K (using the hermitian norm in $\mathbb{C}^d$): for $a \in K$

$$\|a\| := \|\sigma(a)\|_2 = \sqrt{\sum_i |a(\alpha_i)|^2}.$$

We extend $\sigma : K^k \to \mathbb{C}^{dk}$ coordinate-wise, and similarly $\|v\| := \|\sigma(v)\|_2$ for $v \in K^k$

(Free) module:

$$M = \{B \cdot x \,|\, x \in {\mathcal{O}_K}^k\} \text{ for some matrix } B \in {\mathcal{O}_K}^{k \times k} \text{ with } \det_K(B) \neq 0$$

(Free) module:

$$M = \{B \cdot x \,|\, x \in {\mathcal{O}_K}^k\} \text{ for some matrix } B \in {\mathcal{O}_K}^{k \times k} \text{ with } \det_K(B) \neq 0$$

- k is the module rank
- B is a module basis of M

  (if the module is not free, it has a "pseudo-basis" instead)

(Free) module:

$$M = \{B \cdot x \,|\, x \in {\mathcal{O}_K}^k\} \text{ for some matrix } B \in {\mathcal{O}_K}^{k \times k} \text{ with } \det_K(B) \neq 0$$

- k is the module rank
- B is a module basis of M
  (if the module is not free, it has a "pseudo-basis" instead)

$\sigma(M)$ is a lattice:

- with basis $(\sigma(b_i X^j))_{\substack{1 \leq i \leq k \\ 0 \leq j < d}}$ (b_i columns of B)
- of $\mathbb{Z}$-rank $n := d \cdot k$, included in $\mathbb{C}^n$

An example of module lattice

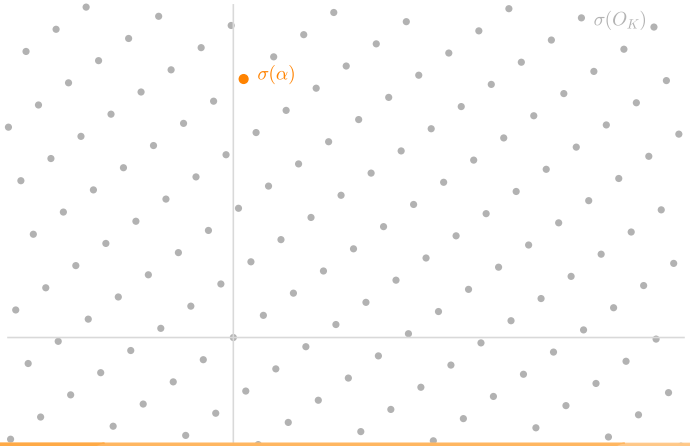
$K = \mathbb{Q}[X]/(X^2+2)$ $\qquad \sigma : K \to \mathbb{R}^2$

$\mathcal{O}_K = \mathbb{Z}[X]/(X^2+2)$ $\qquad a_0 + a_1 X \mapsto (a_0 + a_1\sqrt{2}, a_0 - a_1\sqrt{2})$

An example of module lattice

$K = \mathbb{Q}[X]/(X^2+2)$ $\quad \sigma : K \to \mathbb{R}^2$ $\quad k = 1,\ B = (\alpha) \in \mathcal{O}_K{}^{1\times 1}$

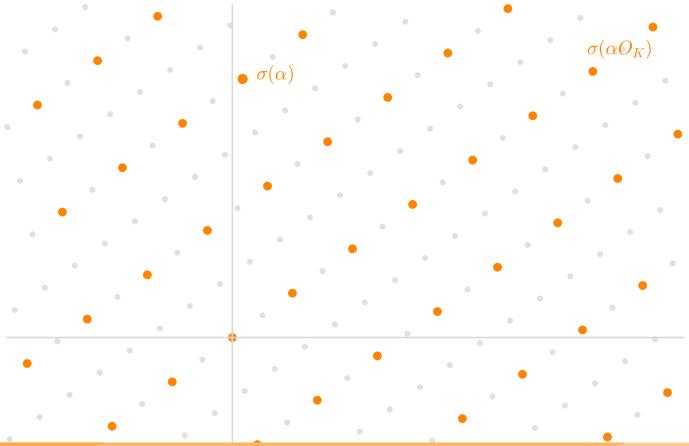
$\mathcal{O}_K = \mathbb{Z}[X]/(X^2+2)$ $\quad a_0 + a_1 X \mapsto (a_0 + a_1\sqrt{2}, a_0 - a_1\sqrt{2})$



An example of module lattice

$K = \mathbb{Q}[X]/(X^2+2)$ $\quad \sigma : K \to \mathbb{R}^2$ $\quad k = 1,\ B = (\alpha) \in {\mathcal{O}_K}^{1\times 1}$

$\mathcal{O}_K = \mathbb{Z}[X]/(X^2+2)$ $\quad a_0 + a_1 X \mapsto (a_0 + a_1\sqrt{2}, a_0 - a_1\sqrt{2})$ $\quad M = \{\alpha x \,|\, x \in \mathcal{O}_K\}$



Section's conclusion

Module lattices: module (algebraic object) + lattice (geometric object)

Section's conclusion

Module lattices: module (algebraic object) + lattice (geometric object)

Advantages:

- smaller storage size required, faster operations
  - more efficient cryptographic protocols
  - used in all standardized lattice-based schemes (Kyber, Dilithium, Falcon) (usually with $d = 256$ or 512 and $k = 2, 3$ or 4)

Section's conclusion

Module lattices: module (algebraic object) + lattice (geometric object)

Advantages:

- smaller storage size required, faster operations
  - more efficient cryptographic protocols
  - used in all standardized lattice-based schemes (Kyber, Dilithium, Falcon) (usually with $d = 256$ or 512 and $k = 2, 3$ or 4)

Drawbacks:

- maybe more efficient attackers against some algorithmic problems?
  - for the short basis problem: better attacker if module rank $k = 1$, no better algorithm known when $k \geq 2$
  - for the lattice isomorphism problem: better attackers if module rank $k = 1$ and for some number fields when $k = 2$.

Wrapping up everything

To build crypto: we need random hard lattices
(lattices for which computing a short basis is hard)

The lattice isomorphism problem: allows to generate random hard lattices

Module lattices: add algebraic structure to improve efficiency of crypto constructions

What is the impact of the added algebraic structure on security?

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

Wrapping up everything

To build crypto: we need random hard lattices
(lattices for which computing a short basis is hard)

The lattice isomorphism problem: allows to generate random hard lattices

Module lattices: add algebraic structure to improve efficiency of crypto constructions

What is the impact of the added algebraic structure on security?

Thank you

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