Lattice-based cryptography

Alice Pellet-Mary

CIMPA school in Douala

Introduction

Public key cryptography

Cryptographic primitives

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

Public key cryptography

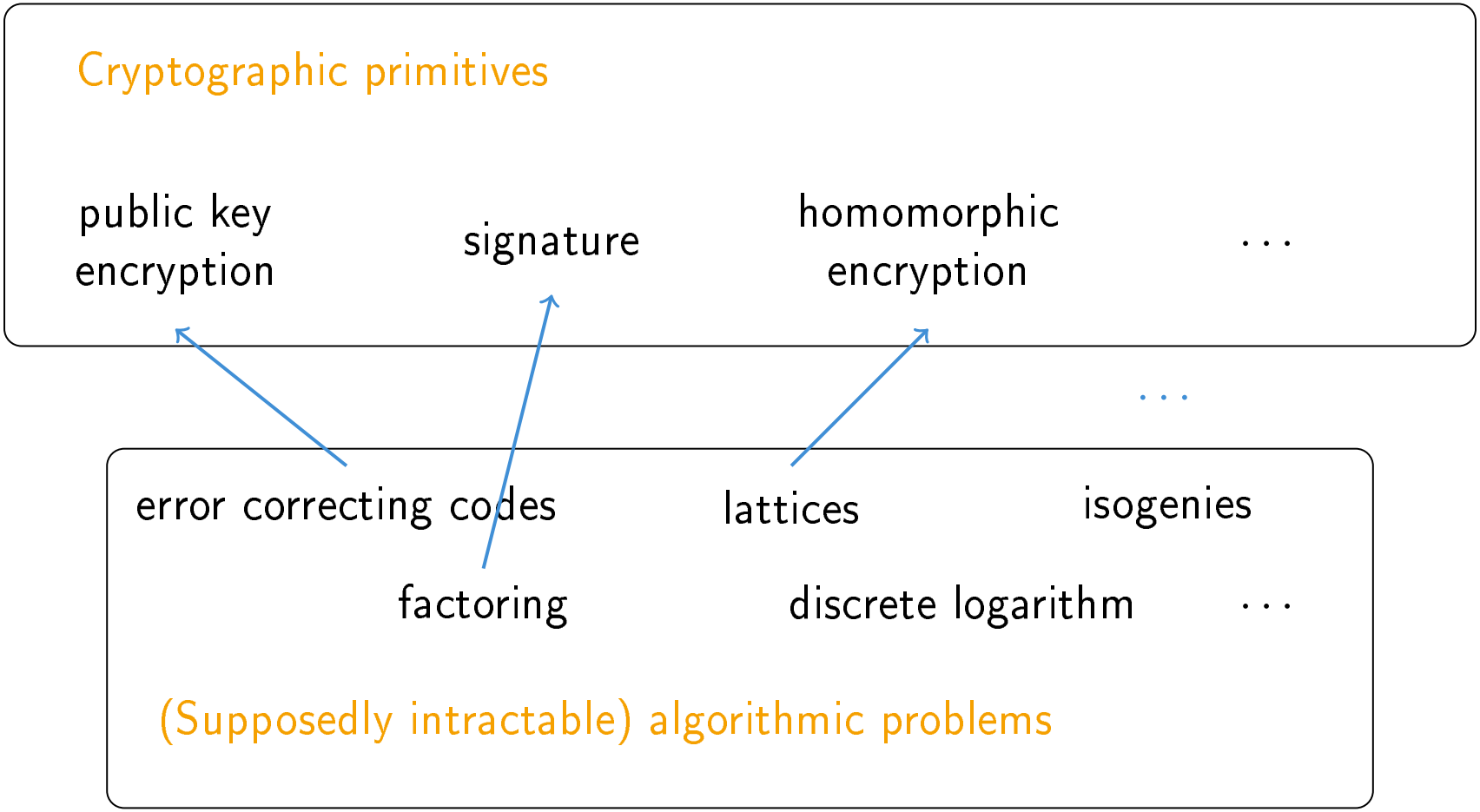
Cryptographic primitives

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

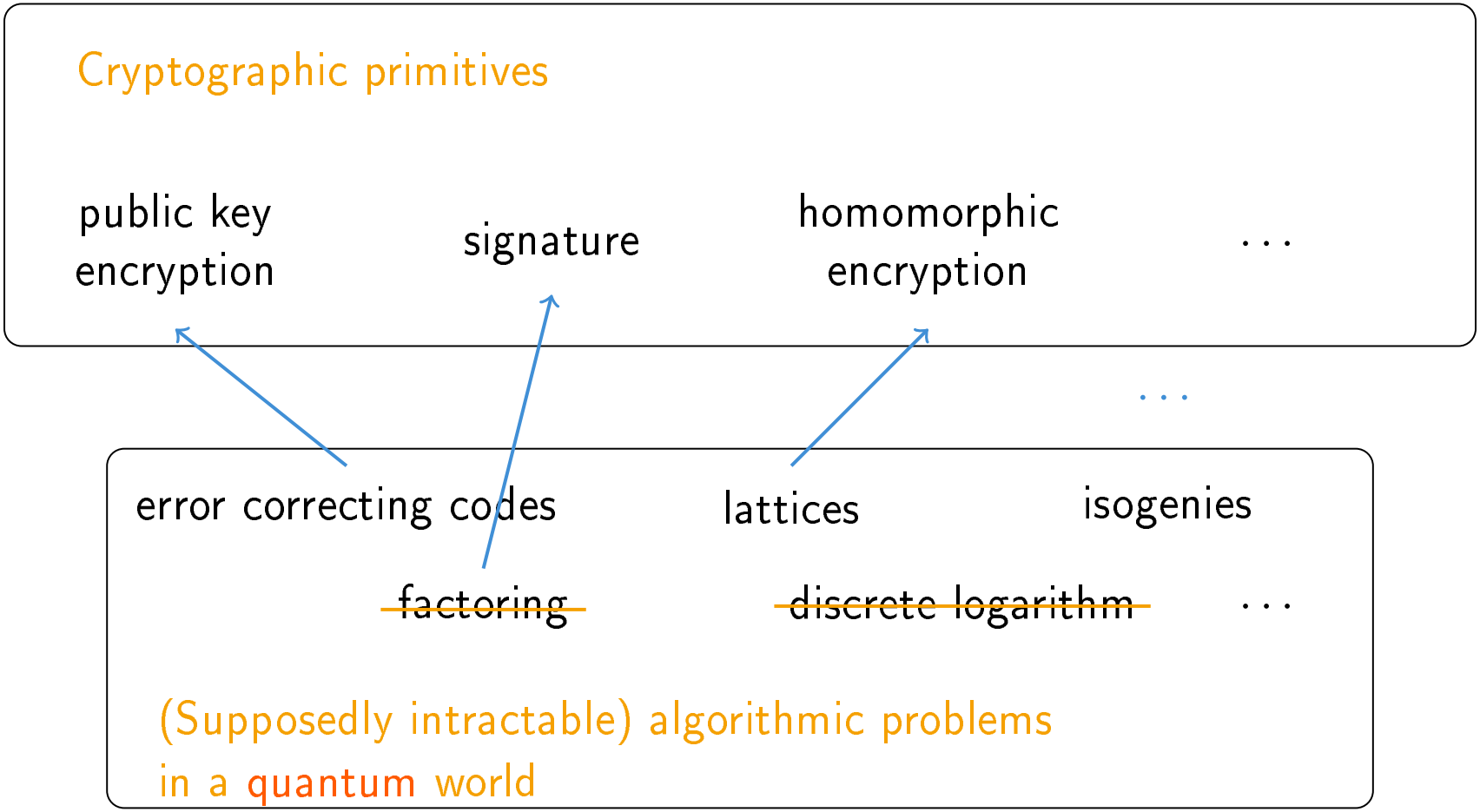
error correcting codes | lattices | isogenies | factoring | discrete logarithm | ...

(Supposedly intractable) algorithmic problems

Public key cryptography



Public key cryptography



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

Post-quantum cryptography vs quantum cryptography

Post-quantum cryptography vs quantum cryptography

Post-quantum cryptography:

- uses classical infrastructures
- resists to attackers with quantum computers

Quantum cryptography:

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

Post-quantum cryptography vs quantum cryptography

Post-quantum cryptography:

- uses classical infrastructures
- resists to attackers with quantum computers

Quantum cryptography:

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

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

Post-quantum cryptography vs quantum cryptography

Post-quantum cryptography:

- uses classical infrastructures
- resists to attackers with quantum computers

Quantum cryptography:

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

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

$\Rightarrow$ Four European agencies recommend post-quantum cryptography[1]

Germany (BSI), France (ANSSI), the Netherlands (NLNCSA) and Sweden (NCSA)

[1] https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Crypto/Quantum_Positionspapier.html?nn=916626

Should we be afraid of quantum attackers?

Should we be afraid of quantum attackers?

I don't know

Should we be afraid of quantum attackers?

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

Should we be afraid of quantum attackers?

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

NIST decided that the threat cannot be neglected

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

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

Should we be afraid of quantum attackers?

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

NIST decided that the threat cannot be neglected

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

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

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

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

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

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

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

2016 NIST first call for proposals

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

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

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

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

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

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

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

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

NIST standardization process

NIST competition for post-quantum key exchange and signatures:

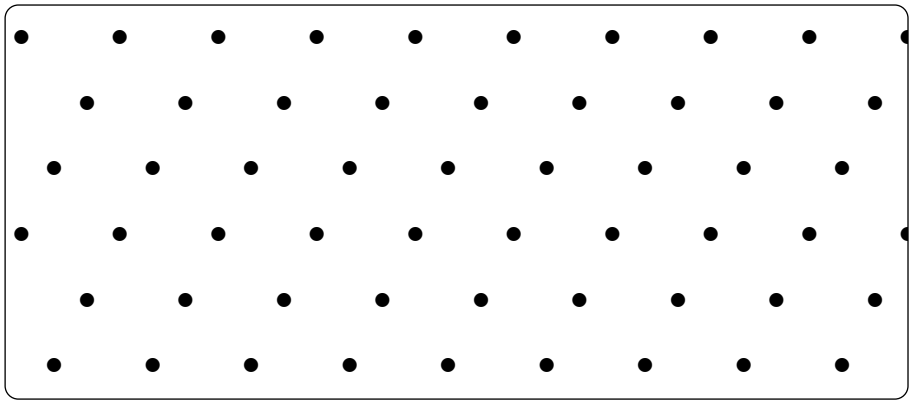
- 2016 NIST first call for proposals
- 2017 $\approx$ 70 submissions
- 2022 NIST announced the first 4 selected algorithms
  (some others are still being analyzed)
  - 1 key exchange: Kyber (lattices)
  - 3 signatures: Dilithium (lattices), Falcon (lattices), SPHINCS+ (hash)
- 2023 call for additional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

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

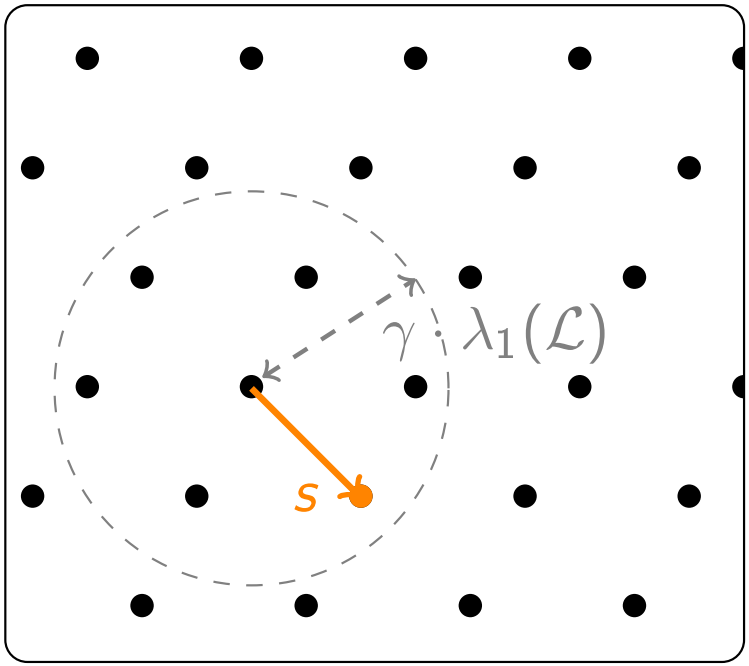
Plan of the lectures

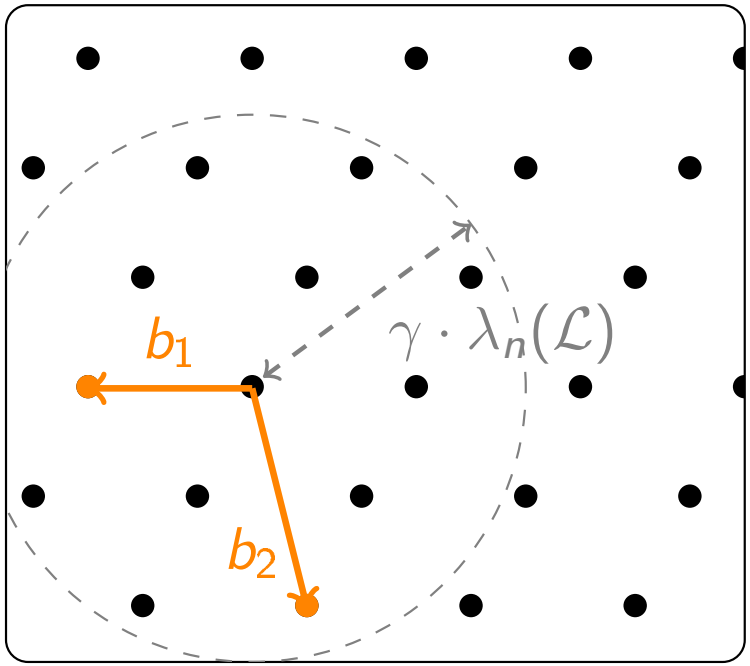
1. Lattices and lattice problems
2. Are lattice problems (quantumly) hard?
3. Cryptographic problems
4. Constructing cryptographic primitives
5. Structured lattices

Lattice problems



Shortest (independent) vector problem





SVP_γ: Short Vector Problem

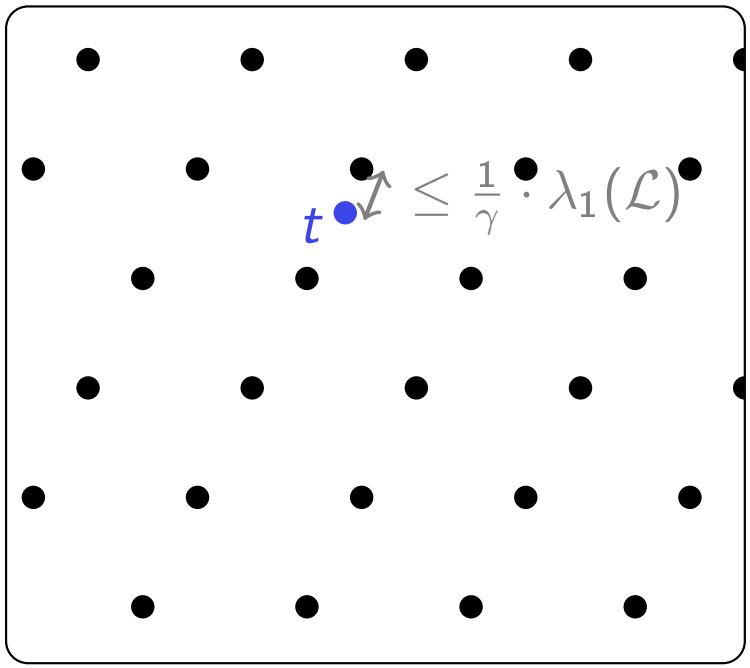
$$\|s\| \leq \gamma \cdot \lambda_1(\mathcal{L})$$

SIVP_γ: Short independent vectors problem

$$\|b_1\|, \ldots, \|b_n\| \leq \gamma \cdot \lambda_n(\mathcal{L})$$

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

Bounded distance decoding problem



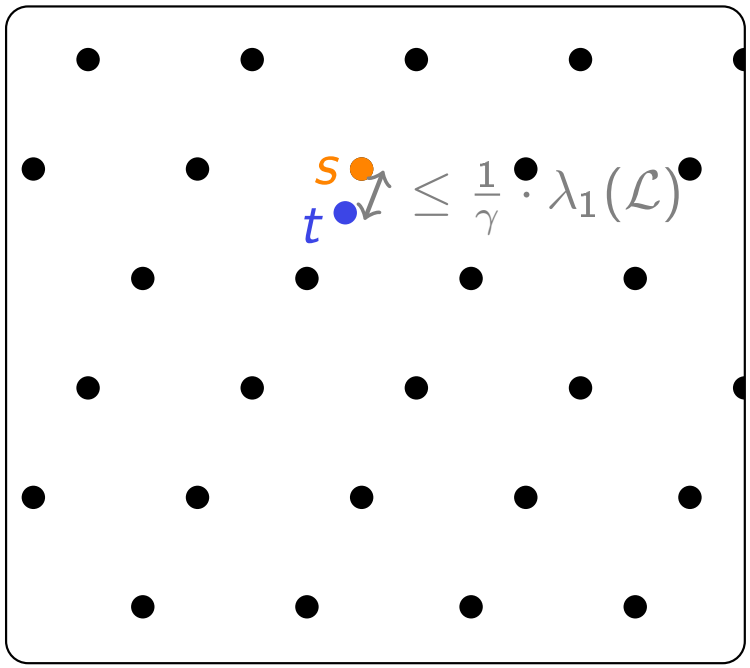
BDD_γ

bounded distance decoding problem

Promise: $\mathrm{dist}(t, \mathcal{L}) \leq 1/\gamma \cdot \lambda_1(\mathcal{L})$

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

Bounded distance decoding problem



BDD_γ

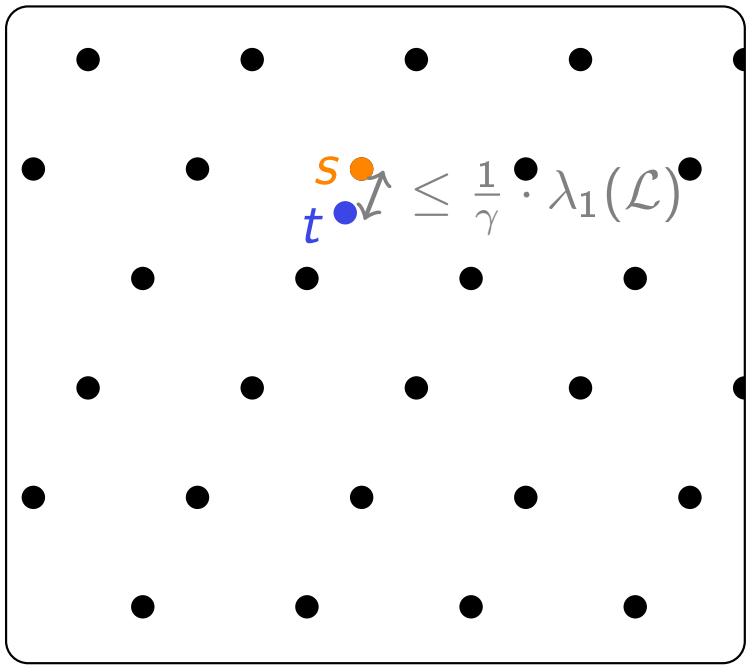
bounded distance decoding problem

Promise: $\mathrm{dist}(t, \mathcal{L}) \leq 1/\gamma \cdot \lambda_1(\mathcal{L})$

Objective: $s \in \mathcal{L}$ s.t. $\|s - t\| \leq 1/\gamma \cdot \lambda_1(\mathcal{L})$

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

Bounded distance decoding problem



BDD_γ

bounded distance decoding problem

Promise: $\mathrm{dist}(t, \mathcal{L}) \leq 1/\gamma \cdot \lambda_1(\mathcal{L})$

Objective: $s \in \mathcal{L}$ s.t. $\|s - t\| \leq 1/\gamma \cdot \lambda_1(\mathcal{L})$

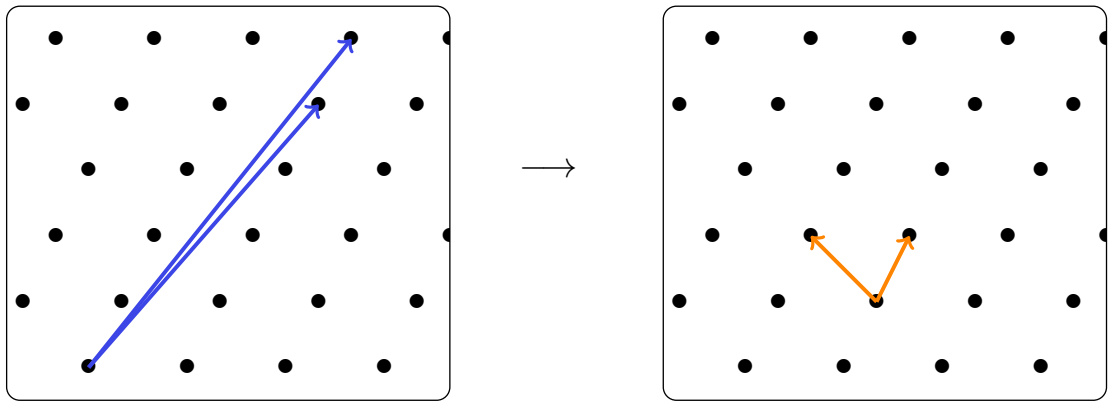
Def: $\mathrm{dist}(t, \mathcal{L}) := \min_{v \in \mathcal{L}}(\|v - t\|)$

Lemma

If $\gamma > 2$, then s is unique

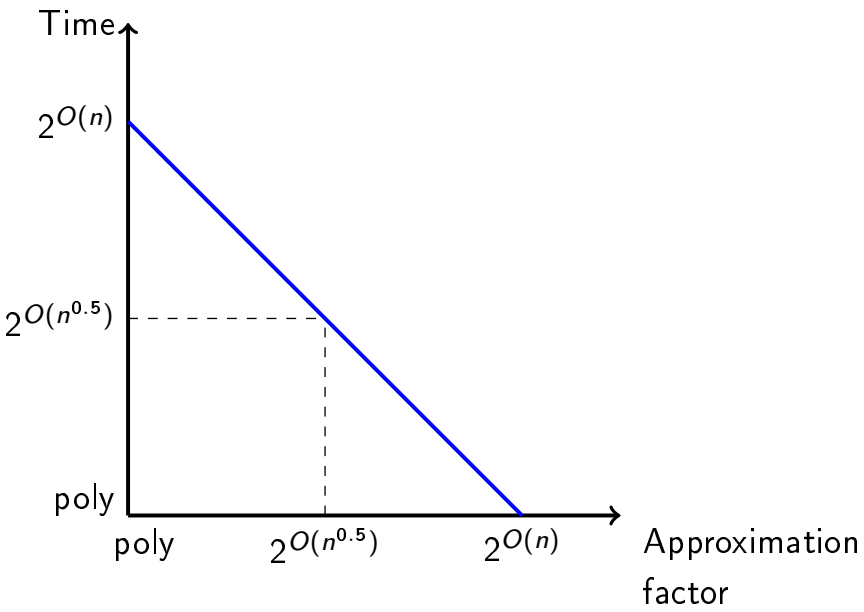
Lattice reduction algorithms

or how to compute a short basis from a long one



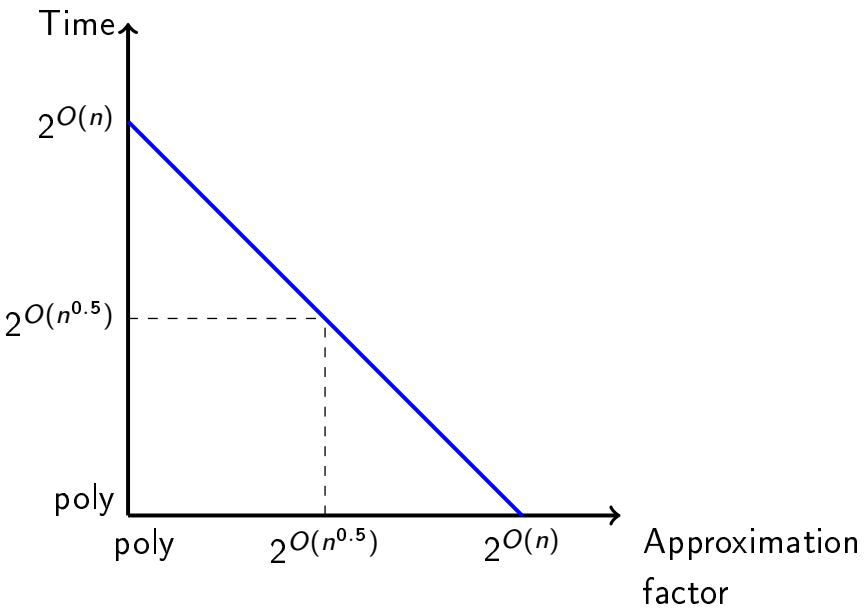
Various SVP algorithms

BKZ trade-offs



Various SVP algorithms

BKZ trade-offs

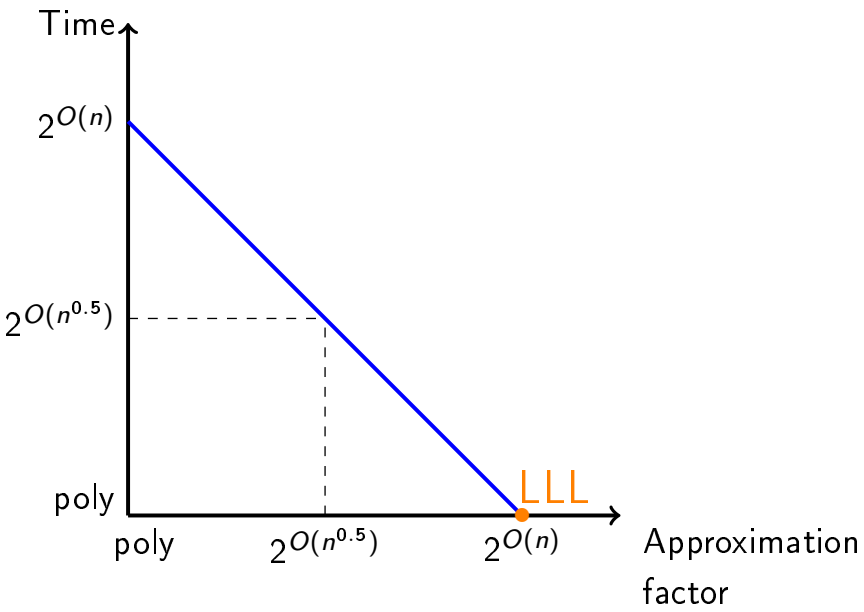


Lagrange-Gauss algorithm: dim 2

- exact SVP/SIVP
- polynomial time

Various SVP algorithms

BKZ trade-offs



Lagrange-Gauss algorithm: dim 2

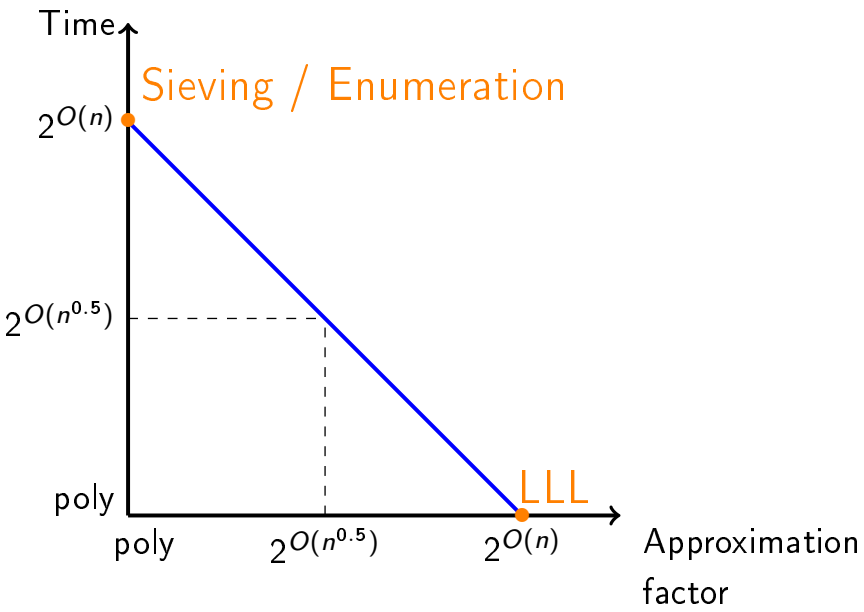
- exact SVP/SIVP
- polynomial time

LLL algorithm: dim n

- SVP_γ / SIVP_γ with $\gamma = 2^n$
- polynomial time

Various SVP algorithms

BKZ trade-offs



Lagrange-Gauss algorithm: dim 2

- exact SVP/SIVP
- polynomial time

LLL algorithm: dim n

- SVP_γ / SIVP_γ with $\gamma = 2^n$
- polynomial time

Sieving algorithm: dim n

- exact SVP
- time $2^{O(n)}$

Lagrange-Gauss algorithm

video

Lagrange-Gauss algorithm

video

Theorem: the algorithm

- solves SVP_γ / SIVP_γ in L for $\gamma = 1$
- runs in polynomial time

The LLL algorithm [LLL82]

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

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

The LLL algorithm [LLL82]

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

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

(using Lagrange-Gauss algorithm)

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

The LLL algorithm [LLL82]

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

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

Algorithm:

- while there exist i such that $\|b_i\|_2 > \lambda_1(L_i)$
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

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

The LLL algorithm [LLL82]

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

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

Algorithm:

- while there exist i such that $\|b_i\|_2 > \lambda_1(L_i)$
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- solves SVP_γ / SIVP_γ in L for $\gamma = 2^n$

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

The LLL algorithm [LLL82]

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

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

Algorithm:

- while there exist i such that $\|b_i\|_2 > \lambda_1(L_i)$
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- solves SVP_γ / SIVP_γ in L for $\gamma = 2^n$
- does not run in polynomial time

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

The LLL algorithm [LLL82]

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

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

Algorithm:

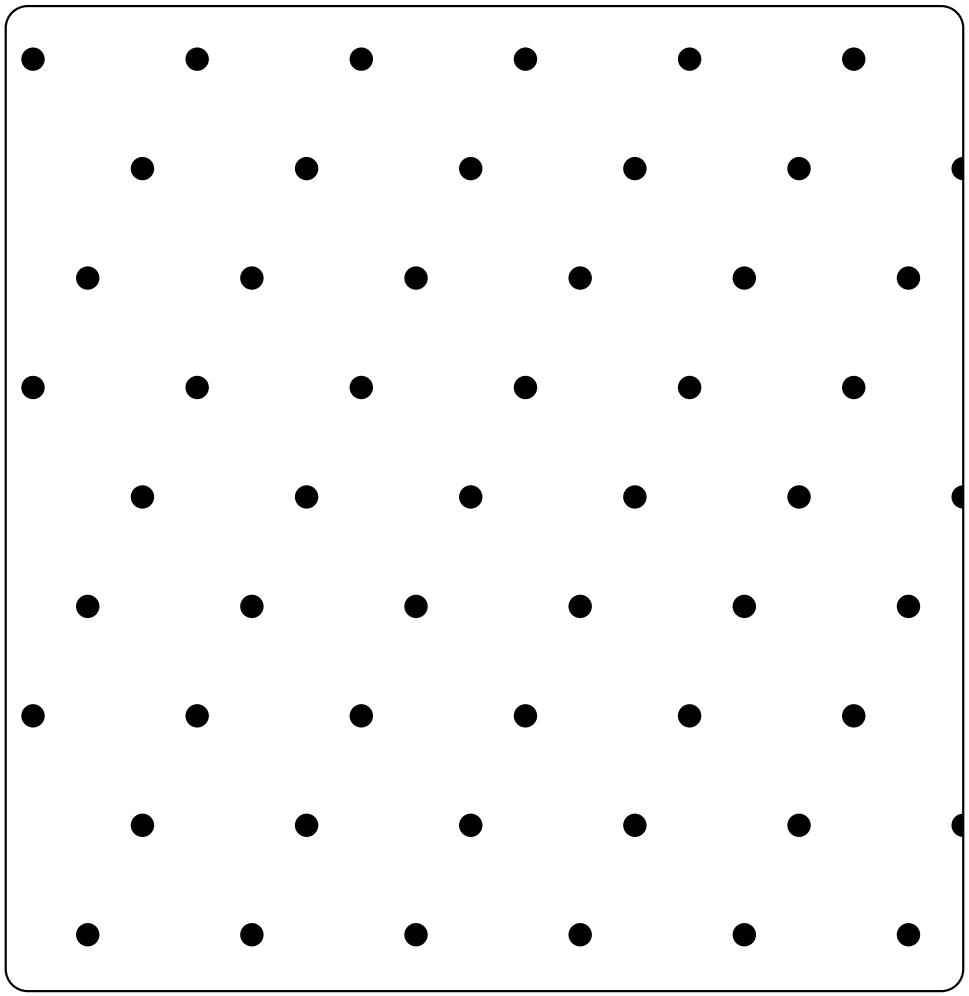
- while there exist i such that $\|b_i\|_2 > 4/3 \cdot \lambda_1(L_i)$
  (L_i is roughly the lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- solves SVP_γ / SIVP_γ in L for $\gamma = 2^n$
- runs in polynomial time

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

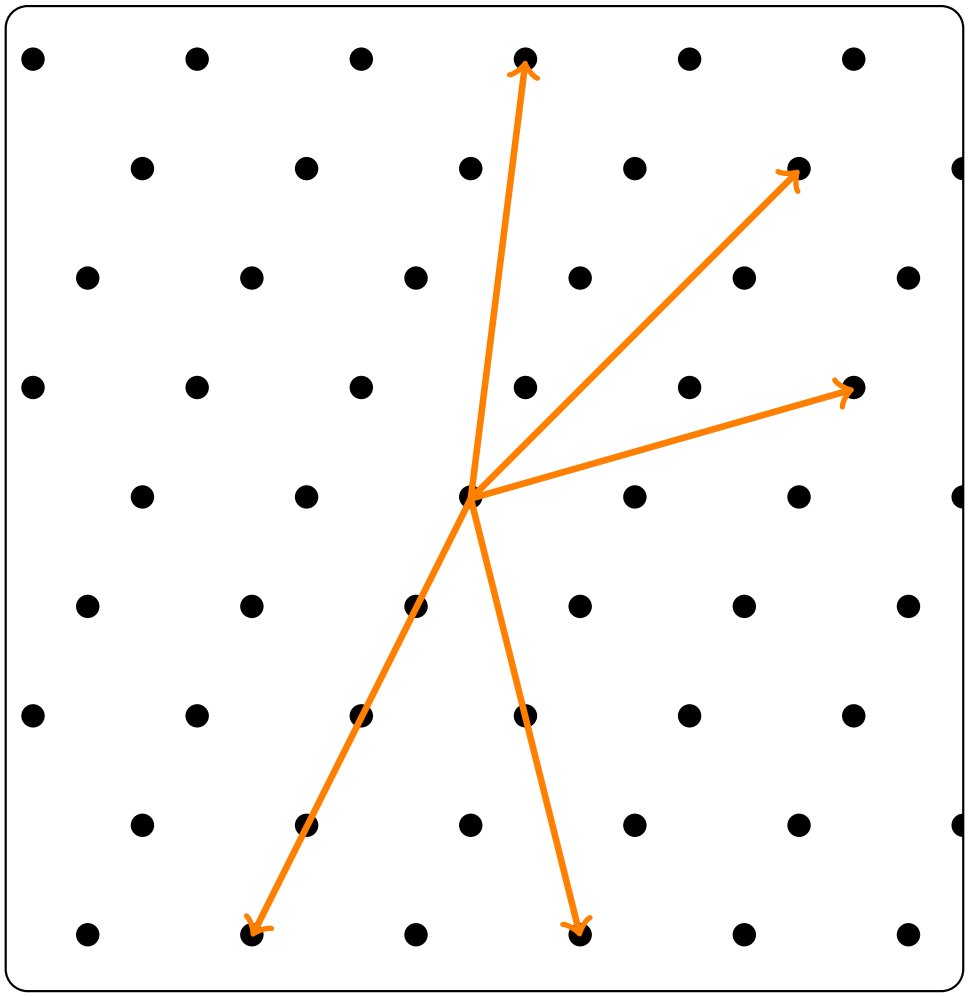
Sieving algorithm [AKS01]



Sieving:

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

Sieving algorithm [AKS01]

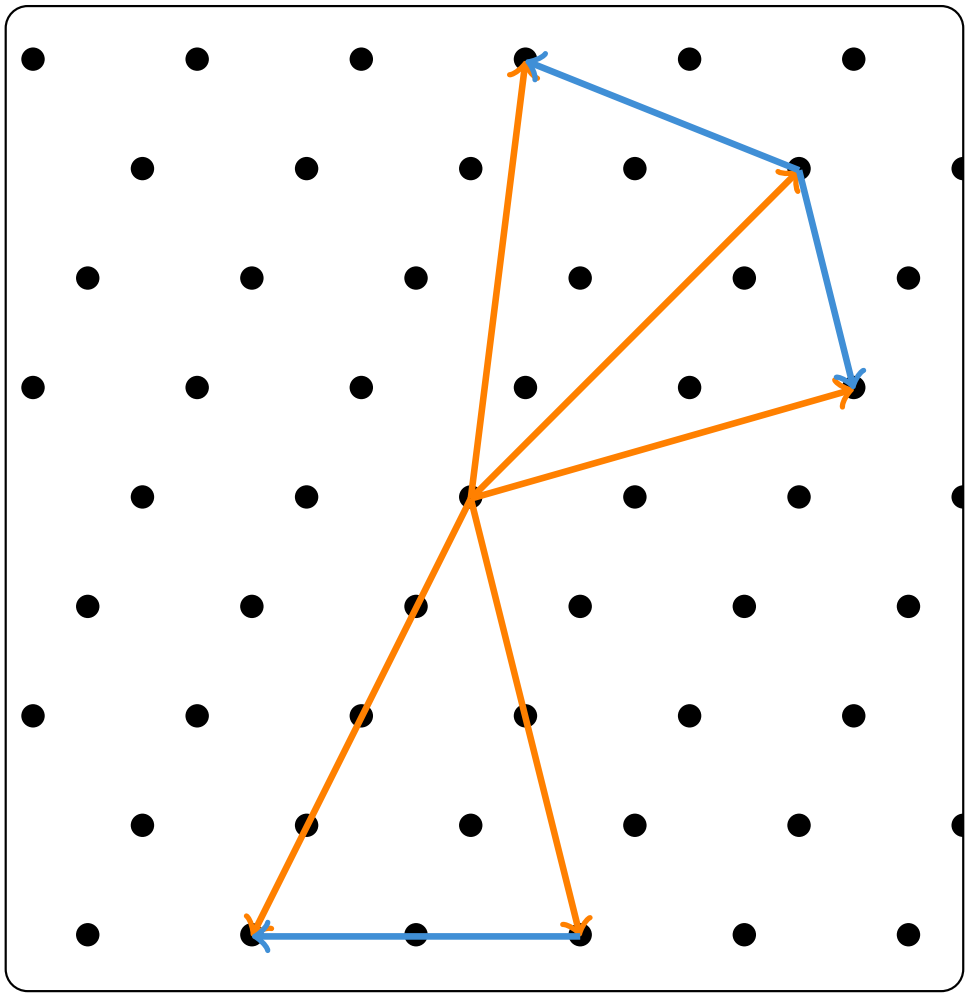


Sieving:

- Create many large vectors

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

Sieving algorithm [AKS01]

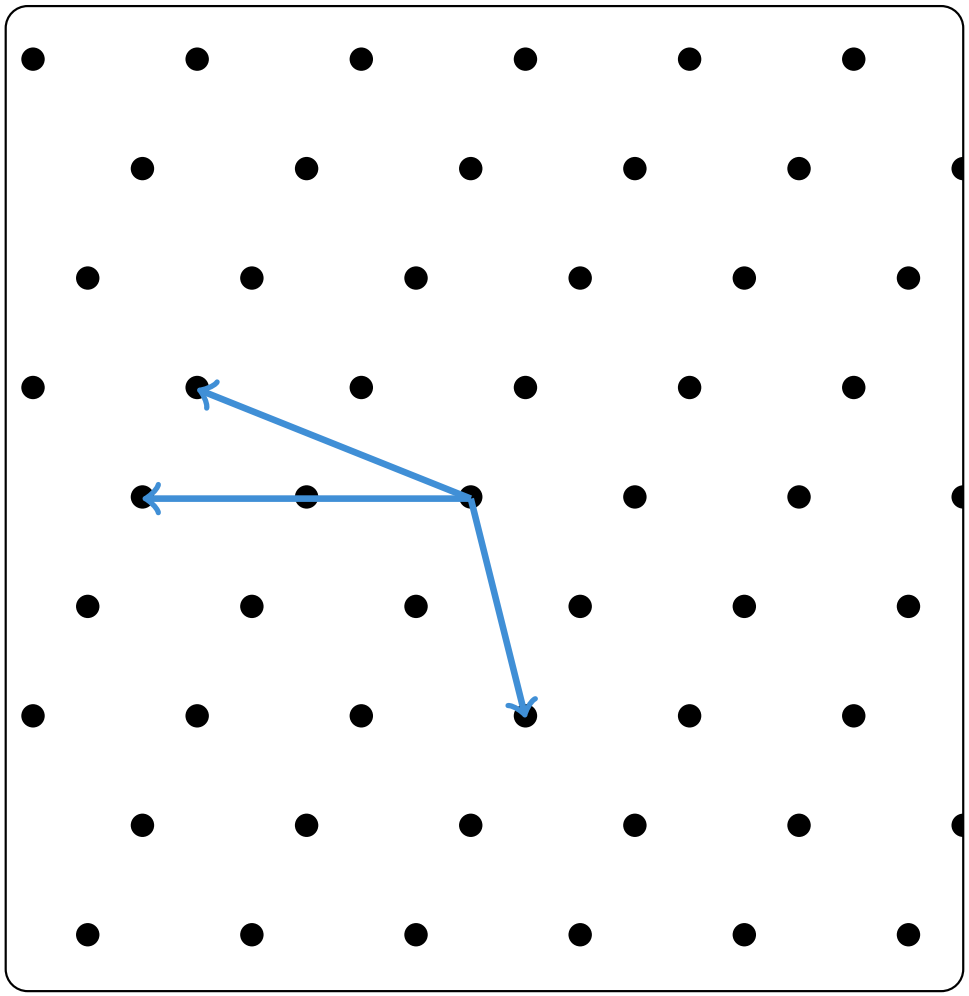


Sieving:

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

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

Sieving algorithm [AKS01]

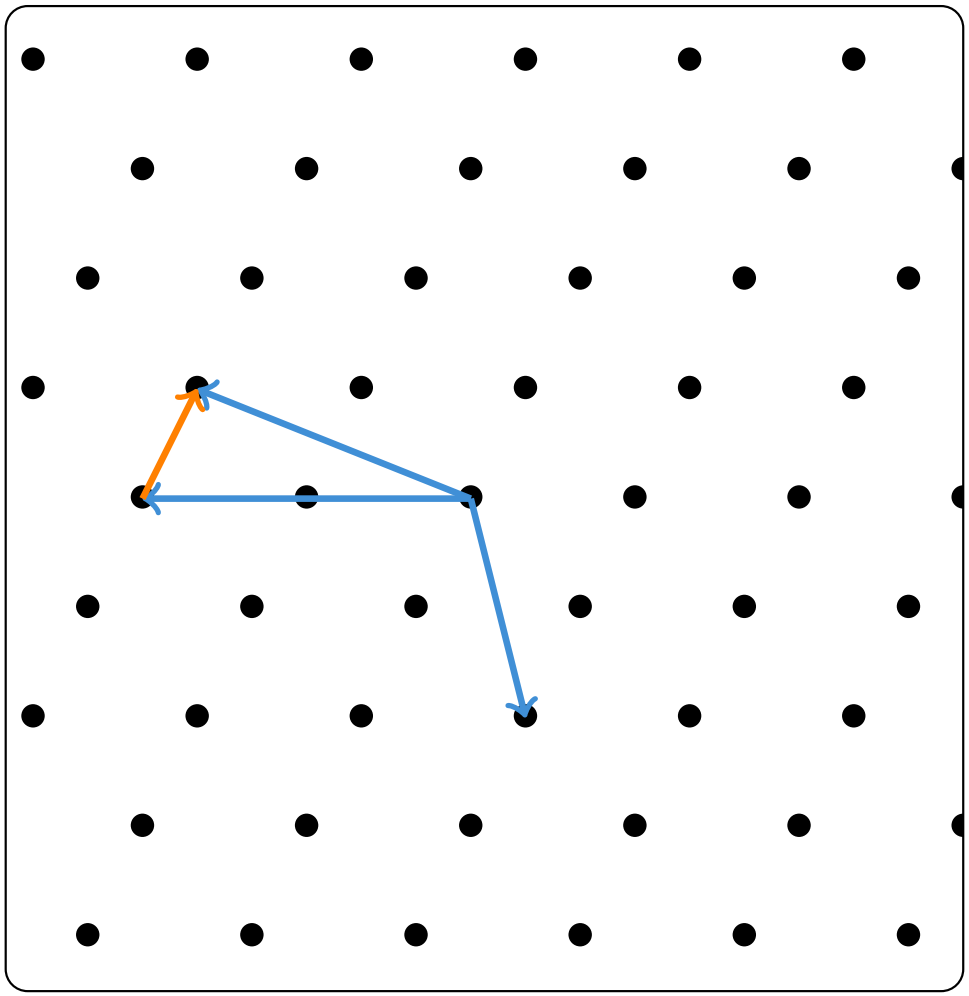


Sieving:

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

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

Sieving algorithm [AKS01]

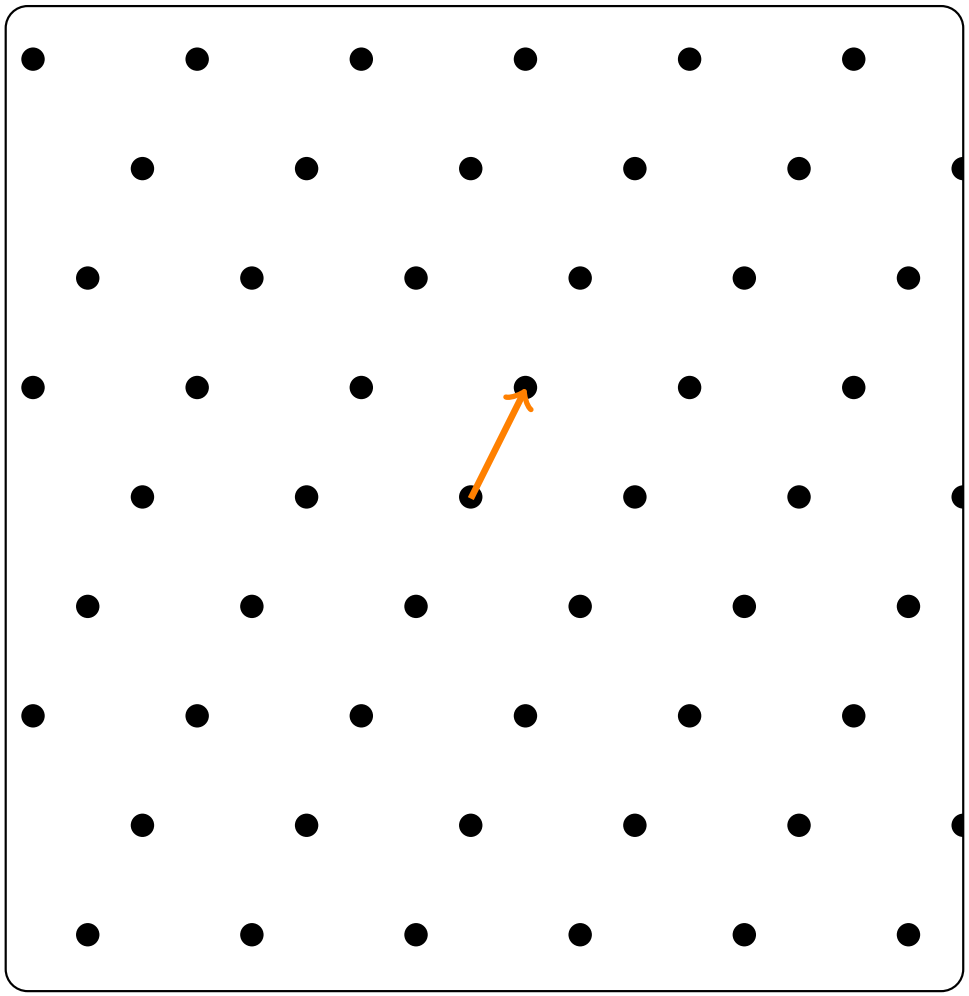


Sieving:

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

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

Sieving algorithm [AKS01]

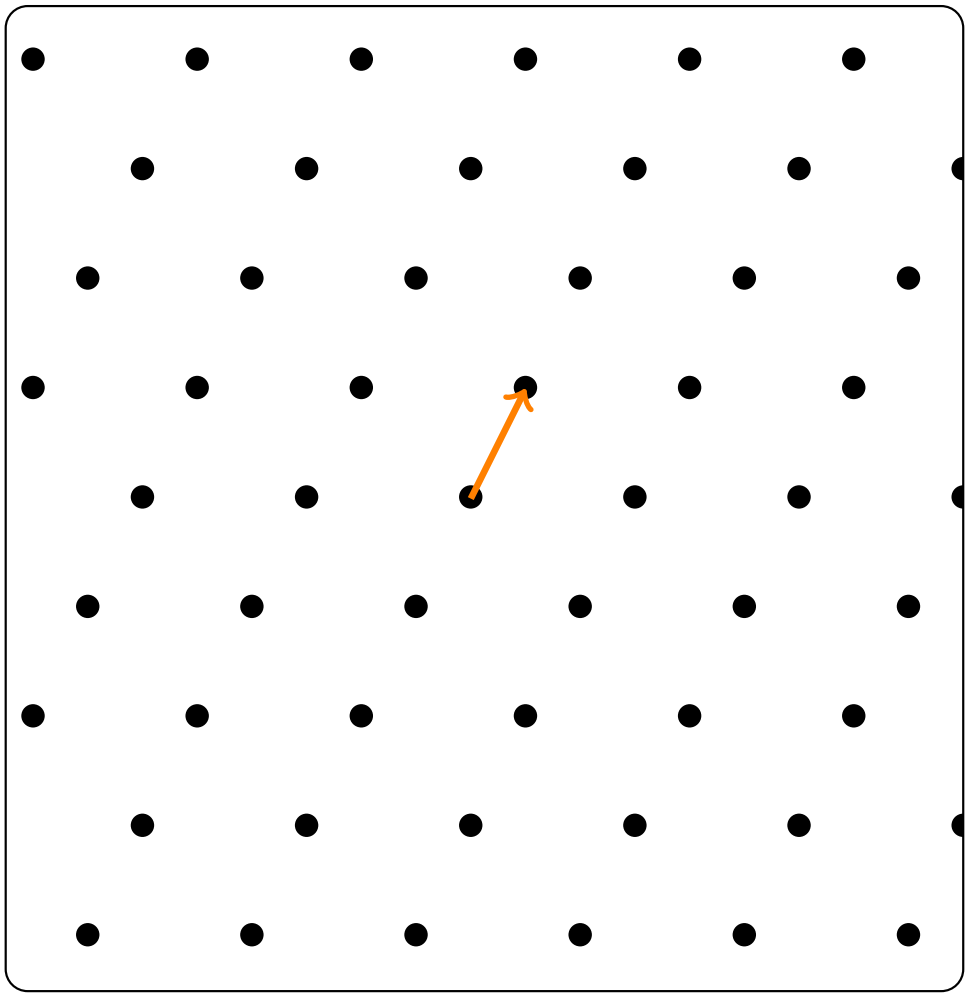


Sieving:

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

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

Sieving algorithm [AKS01]



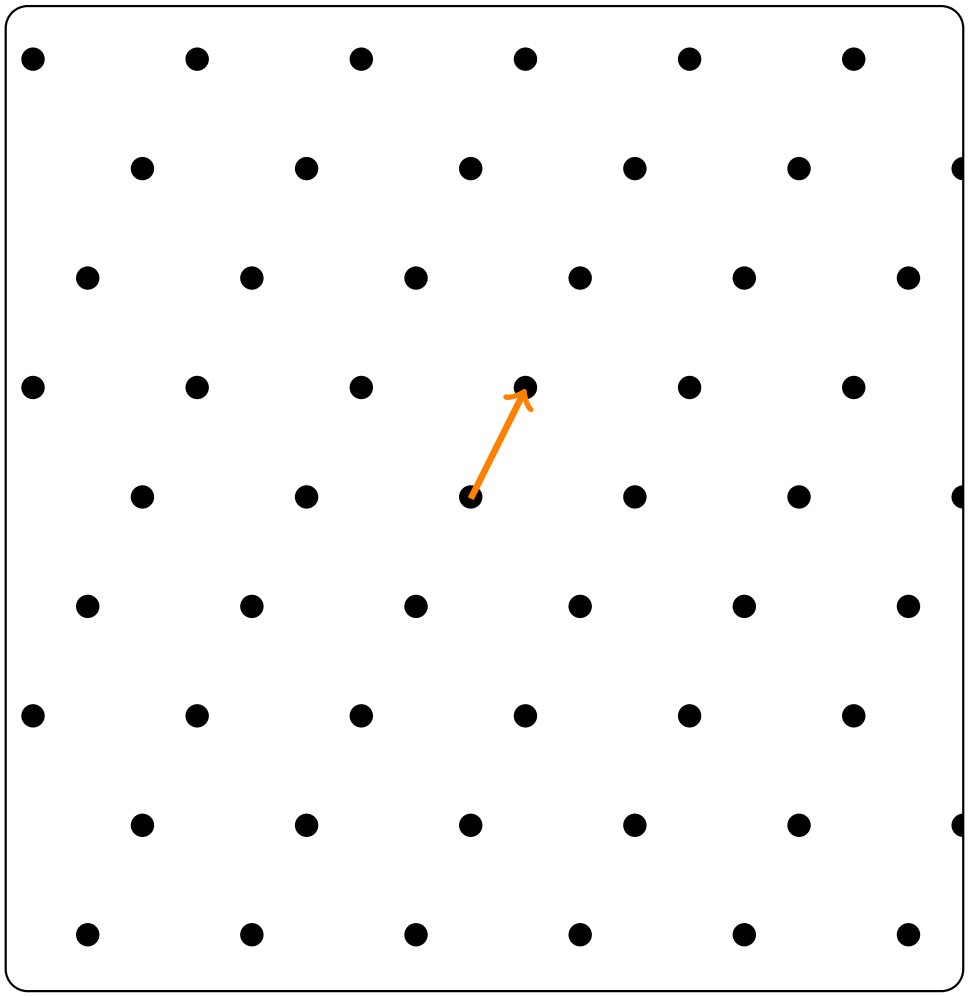
Sieving:

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

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

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

Sieving algorithm [AKS01]



Sieving:

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

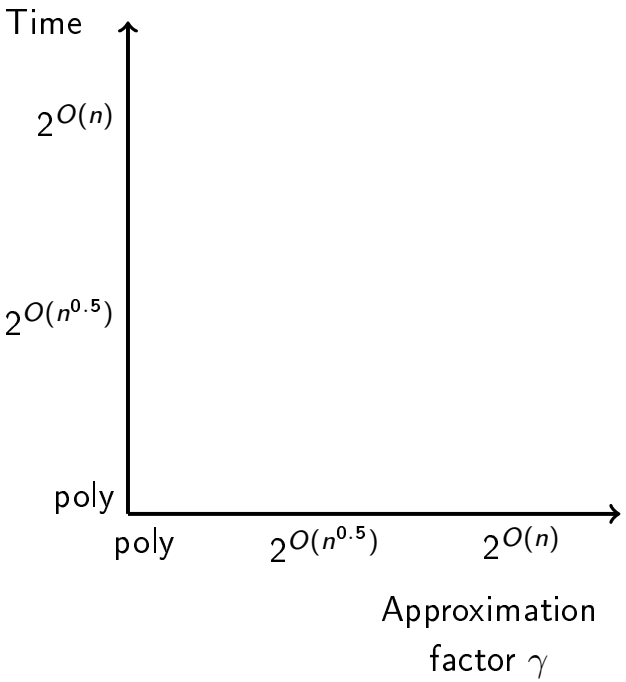
Size of the initial list: $2^{O(n)}$

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

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

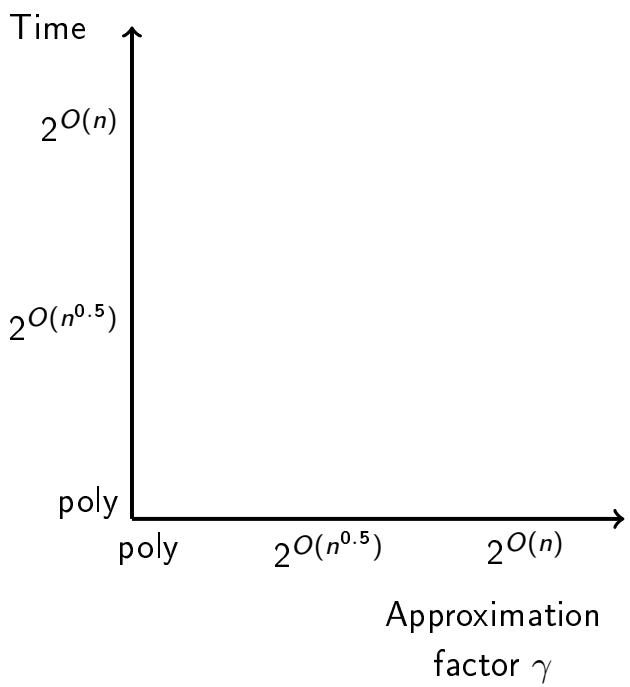
Summing up and BKZ

Algorithms for for SVP_γ



Summing up and BKZ

Algorithms for for SVP_γ

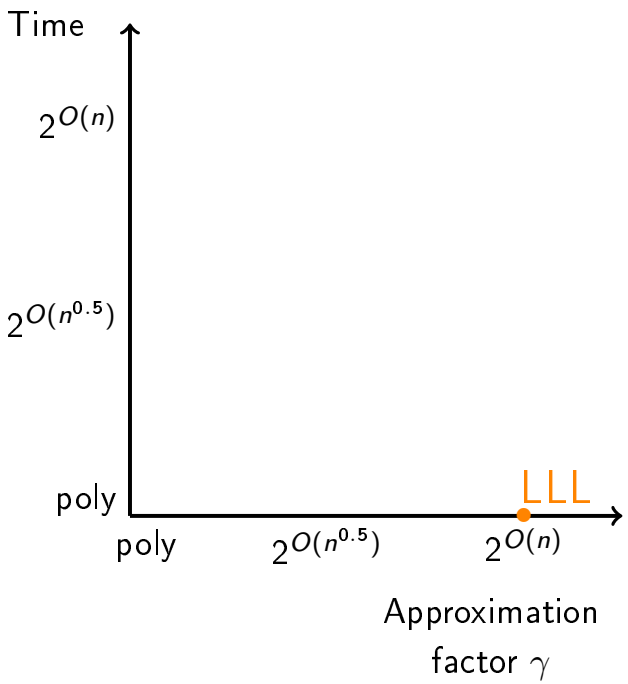


Lagrange-Gauss algorithm: dim 2

- $\gamma = 1$
- polynomial time

Summing up and BKZ

Algorithms for for SVP_γ



Lagrange-Gauss algorithm: dim 2

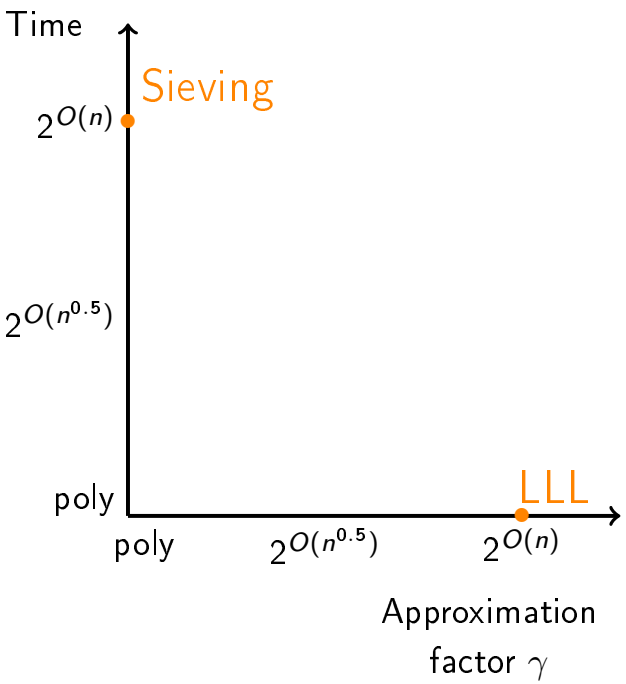
- $\gamma = 1$
- polynomial time

LLL algorithm: dim n

- $\gamma = 2^n$
- polynomial time

Summing up and BKZ

Algorithms for for SVP_γ



Lagrange-Gauss algorithm: dim 2

- $\gamma = 1$
- polynomial time

LLL algorithm: dim n

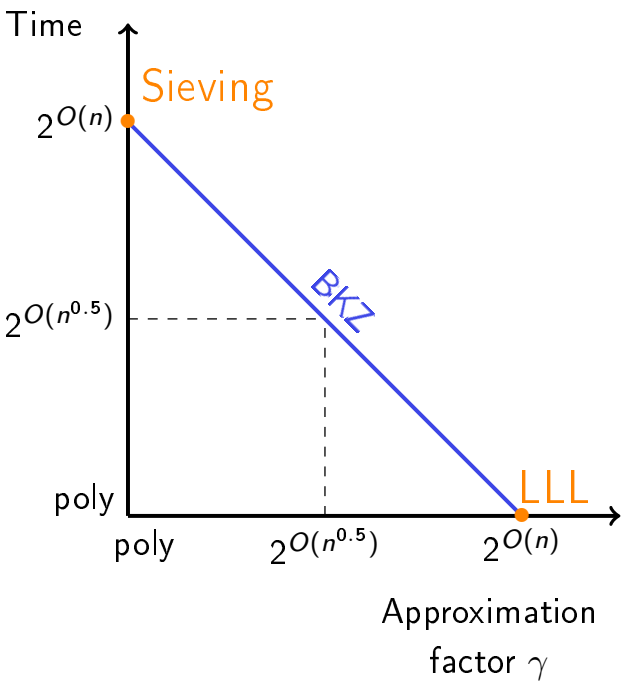
- $\gamma = 2^n$
- polynomial time

Sieving algorithm: dim n

- $\gamma = 1$
- time $2^{O(n)}$

Summing up and BKZ

Algorithms for for SVP_γ



Lagrange-Gauss algorithm: dim 2

- $\gamma = 1$
- polynomial time

LLL algorithm: dim n

- $\gamma = 2^n$
- polynomial time

Sieving algorithm: dim n

- $\gamma = 1$
- time $2^{O(n)}$

BKZ algorithm: combine LLL + Sieving

$\Rightarrow$ various trade-offs

Some concrete numbers

Solving SVP_γ in practice for $\gamma = 1$:

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

Some concrete numbers

Solving SVP_γ in practice for $\gamma = 1$:

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

Some concrete numbers

Solving SVP_γ in practice for $\gamma = 1$:

- $n = 2 \rightsquigarrow$ 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.

Some concrete numbers

Solving SVP_γ in practice for $\gamma = 1$:

- $n = 2 \rightsquigarrow$ 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 $n = 500$ to $n = 1000 \rightsquigarrow$ cryptography

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

Summary and disclaimer

We have seen

there is no efficient algorithm that computes short bases for all lattices

Summary and disclaimer

We have seen

there is no efficient algorithm that computes short bases for all lattices

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

Summary and disclaimer

We have seen

there is no efficient algorithm that computes short bases for all lattices

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

Analogy with factoring: factoring is hard

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

Summary and disclaimer

We have seen

there is no efficient algorithm that computes short bases for all lattices

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

Analogy with factoring: factoring is hard

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

Sage demo

Summary and disclaimer

We have seen

there is no efficient algorithm that computes short bases for all lattices

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

Analogy with factoring: factoring is hard

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

Sage demo

How do we generate a hard lattice?

Cryptographic constructions

Reminder

What we have seen: LWE problem

- average case problem
- expressed using simple linear algebra
- best known algorithm takes time $2^{\Omega(n)}$ (if well chosen parameters)
  - even quantumly
- practical hardness quite well understood

Disclaimer

So far we have seen:

- that LWE is as hard as worst-case lattice problems

  (i.e., if we can solve LWE with good proba, we can solve some lattice problem over all lattices)

Disclaimer

So far we have seen:

- that LWE is as hard as worst-case lattice problems

  (i.e., if we can solve LWE with good proba, we can solve some lattice problem over all lattices)

Now we will see:

- public key encryption based on LWE

Disclaimer

So far we have seen:

- that LWE is as hard as worst-case lattice problems

  (i.e., if we can solve LWE with good proba, we can solve some lattice problem over all lattices)

Now we will see:

- public key encryption based on LWE

But...

- for practical constructions, we choose parameters for which the reductions to worst-case problem do not hold
  - e.g., binary noise, small modulus q, ...

Disclaimer

So far we have seen:

- that LWE is as hard as worst-case lattice problems

  (i.e., if we can solve LWE with good proba, we can solve some lattice problem over all lattices)

Now we will see:

- public key encryption based on LWE

But...

- for practical constructions, we choose parameters for which the reductions to worst-case problem do not hold
  - e.g., binary noise, small modulus q, ...
- reductions are used to show that there is no fundamental flaw in the design
  - taking larger parameters, we can prove that the schemes are as secure as worst case lattice problems

Regev's encryption scheme

(and a little bit of Kyber)

$$\left(A \cdot b = A \, s + e \right)$$

Public key encryption (PKE)

Three algorithms:

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

Public key encryption (PKE)

Three algorithms:

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

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

Public key encryption (PKE)

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

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

Public key encryption (PKE)

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

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

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

Public key encryption (PKE)

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

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

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

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

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

Regev-like encryption scheme [LPS10,LP11]

Parameters: $n, q, B \in \mathbb{Z}_{>0}$ (notation: $\mathcal{U}_B := \mathcal{U}(\{-B, \ldots, B\})$)

[LPS10] Lyubashevsky, Palacio, Segev. Public-key cryptographic primitives provably as secure as subset sum. TCC.

[LP11] Lindner and Peikert. Better key sizes (and attacks) for LWE-based encryption. CT-RSA.

Regev-like encryption scheme [LPS10,LP11]

Parameters: $n, q, B \in \mathbb{Z}_{>0}$ (notation: $\mathcal{U}_B := \mathcal{U}(\{-B, \dots, B\})$)

KeyGen: Sample $A \leftarrow \mathrm{Uniform}(\mathbb{Z}_q^{n \times n})$ and $s, e \leftarrow \mathcal{U}_B^n$

Return $pk = (A, b = As + e \bmod q)$ and $sk = s$

[LPS10] Lyubashevsky, Palacio, Segev. Public-key cryptographic primitives provably as secure as subset sum. TCC.

[LP11] Lindner and Peikert. Better key sizes (and attacks) for LWE-based encryption. CT-RSA.

Regev-like encryption scheme [LPS10,LP11]

Parameters: $n, q, B \in \mathbb{Z}_{>0}$ (notation: $\mathcal{U}_B := \mathcal{U}(\{-B, \dots, B\})$)

KeyGen: Sample $A \leftarrow \mathrm{Uniform}(\mathbb{Z}_q^{n \times n})$ and $s, e \leftarrow \mathcal{U}_B^n$

Return $pk = (A, b = As + e \bmod q)$ and $sk = s$

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

sample $\tilde{s}^T, \tilde{e}^T \leftarrow \mathcal{U}_B^n$ and $e' \leftarrow \mathcal{U}_B$

return $c = (\tilde{s}^T A + \tilde{e}^T, \tilde{s}^T b + e' + m \cdot \lfloor q/2 \rceil)$

(all $\bmod q$)

[LPS10] Lyubashevsky, Palacio, Segev. Public-key cryptographic primitives provably as secure as subset sum. TCC.

[LP11] Lindner and Peikert. Better key sizes (and attacks) for LWE-based encryption. CT-RSA.

Regev-like encryption scheme [LPS10,LP11]

Parameters: $n, q, B \in \mathbb{Z}_{>0}$ (notation: $\mathcal{U}_B := \mathcal{U}(\{-B, \dots, B\})$)

KeyGen: Sample $A \leftarrow \text{Uniform}(\mathbb{Z}_q^{n \times n})$ and $s, e \leftarrow \mathcal{U}_B^n$

Return $pk = (A, b = A\,s + e \bmod q)$ and $sk = s$

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

sample $\tilde{s}^T, \tilde{e}^T \leftarrow \mathcal{U}_B^n$ and $e' \leftarrow \mathcal{U}_B$

return $c = (\tilde{s}^T A + \tilde{e}^T, \tilde{s}^T b + e' + m \cdot \lfloor q/2 \rceil)$

(all $\bmod q$)

Decrypt: $c = (c_1{}^T, c_2)$

compute $r = c_1{}^T s - c_2 \bmod q$ ($r \in [0, q]$)

return 1 if r is in $[q/4, 3q/4]$ and 0 otherwise

[LPS10] Lyubashevsky, Palacio, Segev. Public-key cryptographic primitives provably as secure as subset sum. TCC.

[LP11] Lindner and Peikert. Better key sizes (and attacks) for LWE-based encryption. CT-RSA.

Correctness

Theorem

If $q \geq 8 \cdot n \cdot B^2 + 4 \cdot B$, then the scheme is correct.

Correctness: for any message m and any $(pk, sk) \leftarrow \mathsf{KeyGen}$, it holds that

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m.$$

Correctness

Theorem

If $q \geq 8 \cdot n \cdot B^2 + 4 \cdot B$, then the scheme is correct.

Correctness: for any message m and any $(pk, sk) \leftarrow \mathrm{KeyGen}$, it holds that

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

Proof:

$$r = \tilde{s}^T A s + \tilde{e}^T s - \tilde{s}^T A s - \tilde{s}^T e + e' + m \cdot \lfloor q/2 \rceil$$

Correctness

Theorem

If $q \geq 8 \cdot n \cdot B^2 + 4 \cdot B$, then the scheme is correct.

Correctness: for any message m and any $(pk, sk) \leftarrow \mathsf{KeyGen}$, it holds that

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m.$$

Proof:

$$\begin{aligned} r &= \cancel{\tilde{s}^T A s} + \tilde{e}^T s - \cancel{\tilde{s}^T A s} - \tilde{s}^T e + e' + m \cdot \lfloor q/2 \rceil \\ &= m \cdot \lfloor q/2 \rceil + \tilde{e}^T s - \tilde{s}^T e + e' \end{aligned}$$

Correctness

Theorem

If $q \geq 8 \cdot n \cdot B^2 + 4 \cdot B$, then the scheme is correct.

Correctness: for any message m and any $(pk, sk) \leftarrow \mathsf{KeyGen}$, it holds that

$$\mathsf{Dec}(sk, \mathsf{Enc}(pk, m)) = m.$$

Proof:

$$\begin{aligned} r &= \cancel{\tilde{s}^T A s} + \tilde{e}^T s - \cancel{\tilde{s}^T A s} - \tilde{s}^T e + e' + m \cdot \lfloor q/2 \rceil \\ &= m \cdot \lfloor q/2 \rceil + \underbrace{\tilde{e}^T s - \tilde{s}^T e + e'}_{\leq nB^2 + nB^2 + B \leq q/4} \end{aligned}$$

Security: high level

Theorem (informal)

If dec-LWE is hard to solve, then the scheme is IND-CPA secure.

IND-CPA security: $\mathrm{Enc}(pk, 0) \approx_c \mathrm{Enc}(pk, 1)$

Public information:

A

$b = A\ s + e$

$c_1{}^T = \tilde{s}^T\ A + \tilde{e}^T$

$c_2 = \tilde{s}^T\ b + e' + m \cdot \lfloor q/2 \rceil$

Security: high level

Theorem (informal)

If dec-LWE is hard to solve, then the scheme is IND-CPA secure.

IND-CPA security: $\mathrm{Enc}(pk, 0) \approx_c \mathrm{Enc}(pk, 1)$

Public information:

$$A$$

$$b = A \; s + e$$

$$c_1{}^T = \tilde{s}^T \; A + \tilde{e}^T$$

$$c_2 = \tilde{s}^T \; b + e' + m \cdot \lfloor q/2 \rceil$$

Decision-LWE: $b \approx b$

Security: high level

Theorem (informal)

If dec-LWE is hard to solve, then the scheme is IND-CPA secure.

IND-CPA security: $\mathrm{Enc}(pk, 0) \approx_c \mathrm{Enc}(pk, 1)$

Public information:

A

$b = A\ s + e$

$c_1^T = \tilde{s}^T\ A + \tilde{e}^T$

$c_2 = \tilde{s}^T\ b + e' + m \cdot \lfloor q/2 \rceil$

Decision-LWE: $b \approx b$

Security: high level

Theorem (informal)

If dec-LWE is hard to solve, then the scheme is IND-CPA secure.

IND-CPA security: $\mathsf{Enc}(pk, 0) \approx_c \mathsf{Enc}(pk, 1)$

Public information:

$$A$$

$$b = A \; s + e$$

$$c_1{}^T = \tilde{s}^T \; A + \tilde{e}^T$$

$$c_2 = \tilde{s}^T \; b + e' + m \cdot \lfloor q/2 \rceil$$

Decision-LWE:

$$b \approx b$$

Decision-LWE:

$$\tilde{s}^T \; A \; b + \tilde{e}^T \; e' \approx c_1{}^T \; c$$

Security: high level

Theorem (informal)

If dec-LWE is hard to solve, then the scheme is IND-CPA secure.

IND-CPA security: $\mathsf{Enc}(pk, 0) \approx_c \mathsf{Enc}(pk, 1)$

Public information:

A

$b = A\ s + e$

$c_1^T = \tilde{s}^T\ A + \tilde{e}^T$

$c_2 = c + m \cdot \lfloor q/2 \rceil$

Decision-LWE:

$b \approx b$

Decision-LWE:

$\tilde{s}^T\ A\ b + \tilde{e}^T\ e' \approx c_1^T\ c$

Security: high level

Theorem (informal)

If dec-LWE is hard to solve, then the scheme is IND-CPA secure.

IND-CPA security: $\mathrm{Enc}(pk, 0) \approx_c \mathrm{Enc}(pk, 1)$

Public information:

A

$b = A\ s + e$

$c_1^T = \tilde{s}^T\ A + \tilde{e}^T$

$c_2 = c + m \cdot \lfloor q/2 \rceil$

Decision-LWE:

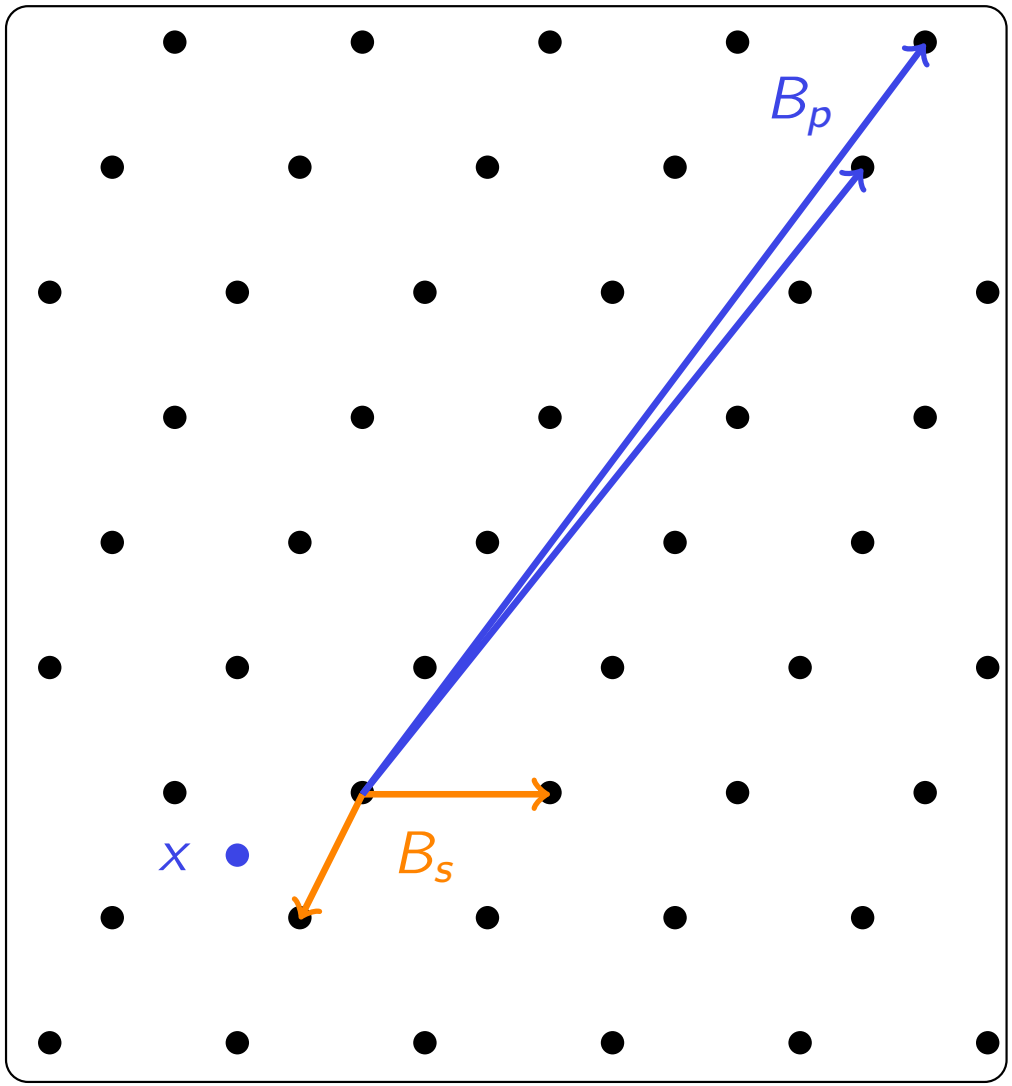
$b \approx b$

Decision-LWE:

$\tilde{s}^T\ A\ b + \tilde{e}^T\ e' \approx c_1^T\ c$

What is happening with pictures?

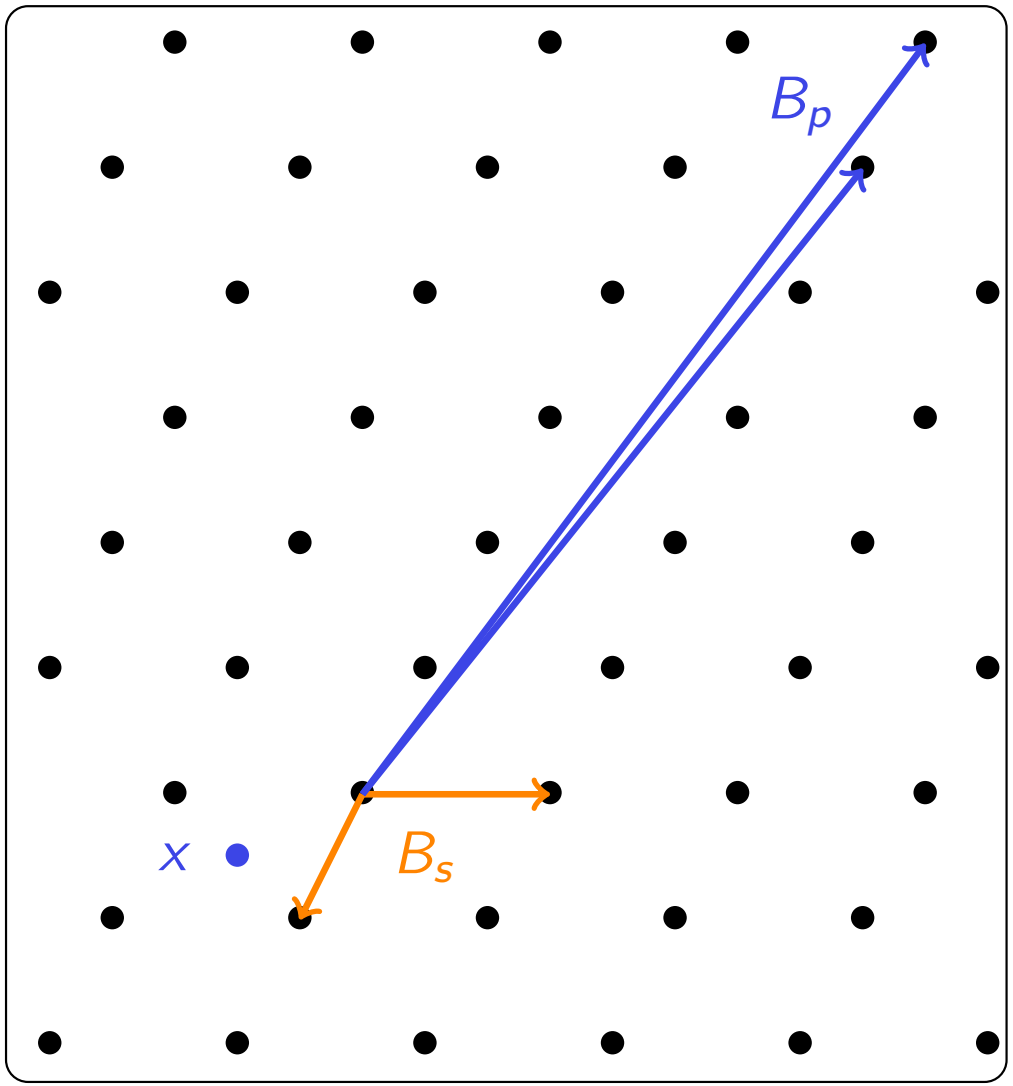
What is happening with pictures?



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

$\mathrm{sk} = B_s$

What is happening with pictures?

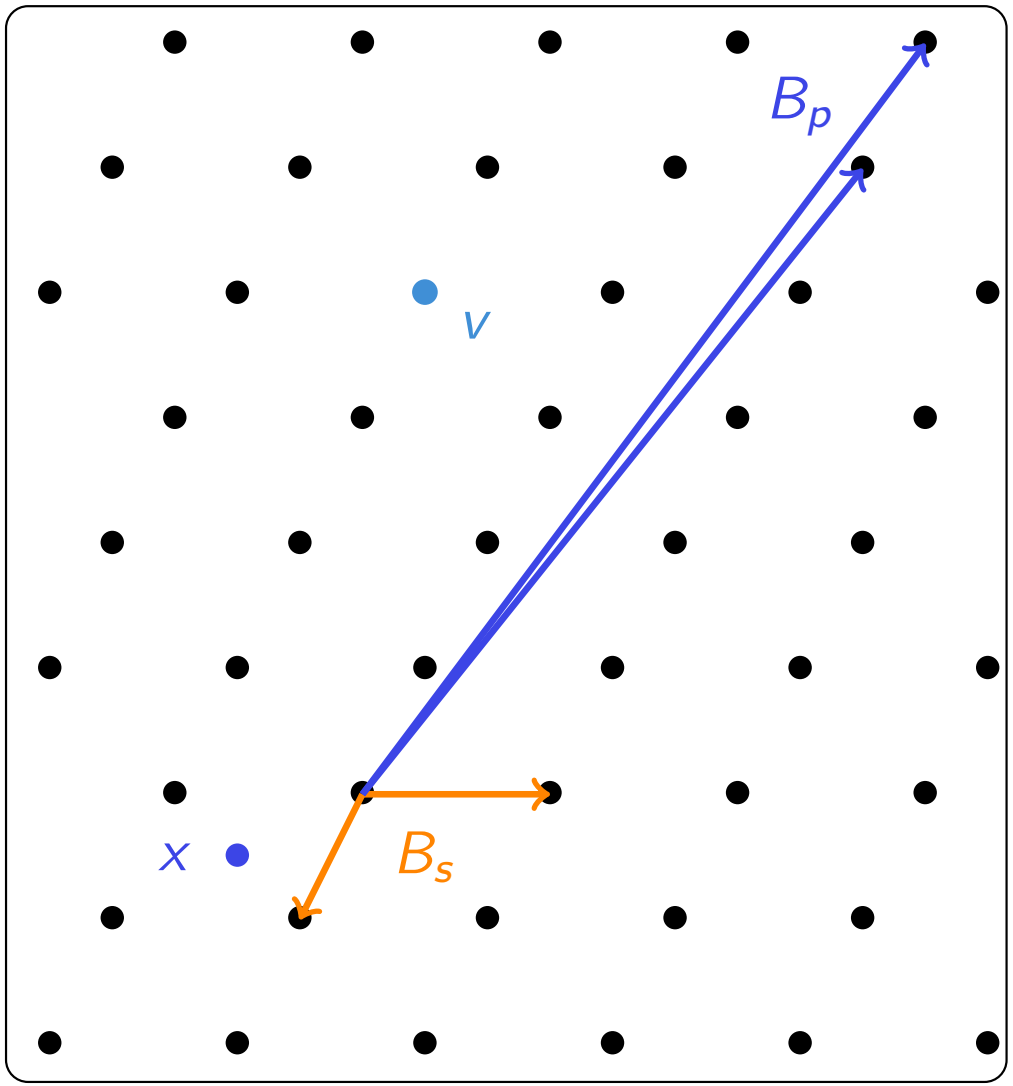


$\mathrm{pk} = (B_p, x)$

$\mathrm{sk} = B_s$

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

What is happening with pictures?



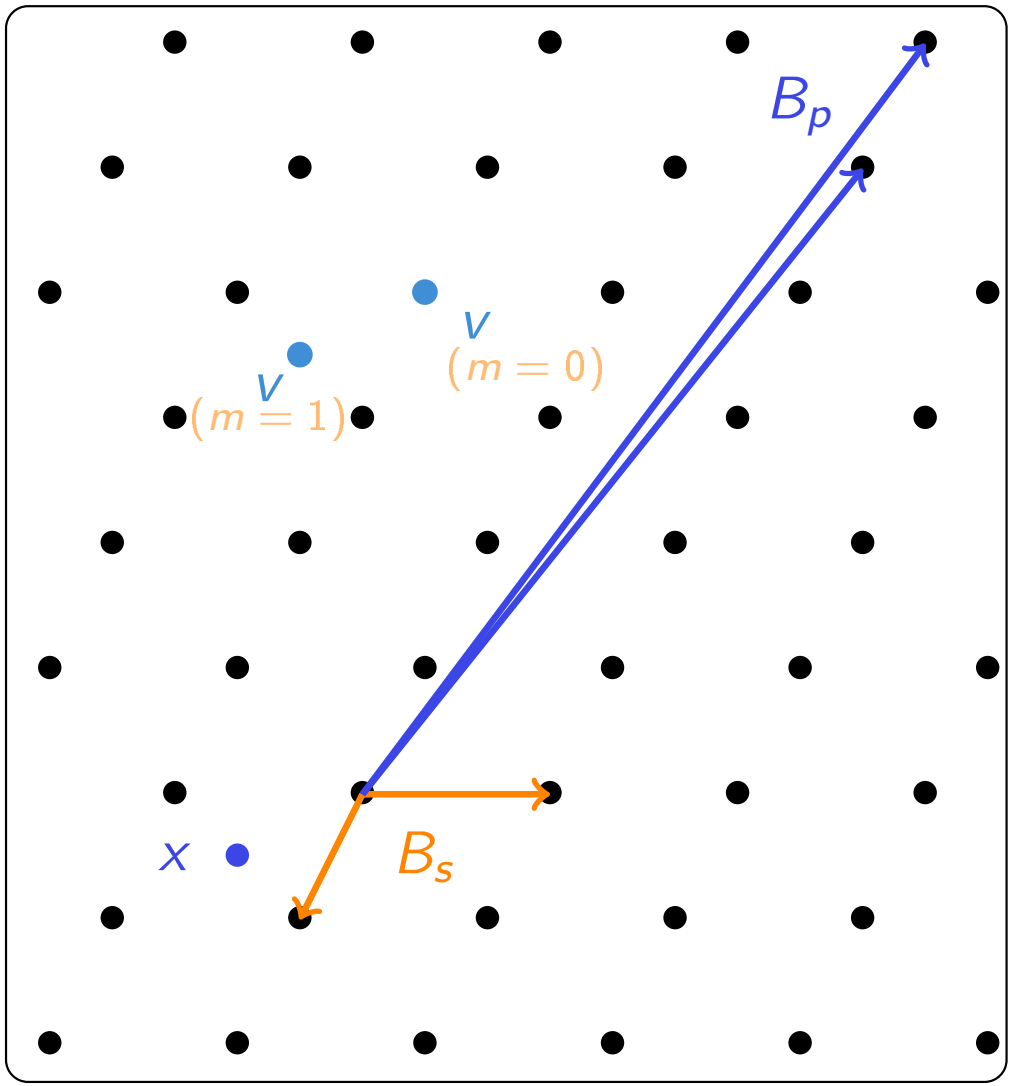
$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

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

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

- Sample $v \xleftarrow{\$} \mathcal{L}$ (using B_p)

What is happening with pictures?



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

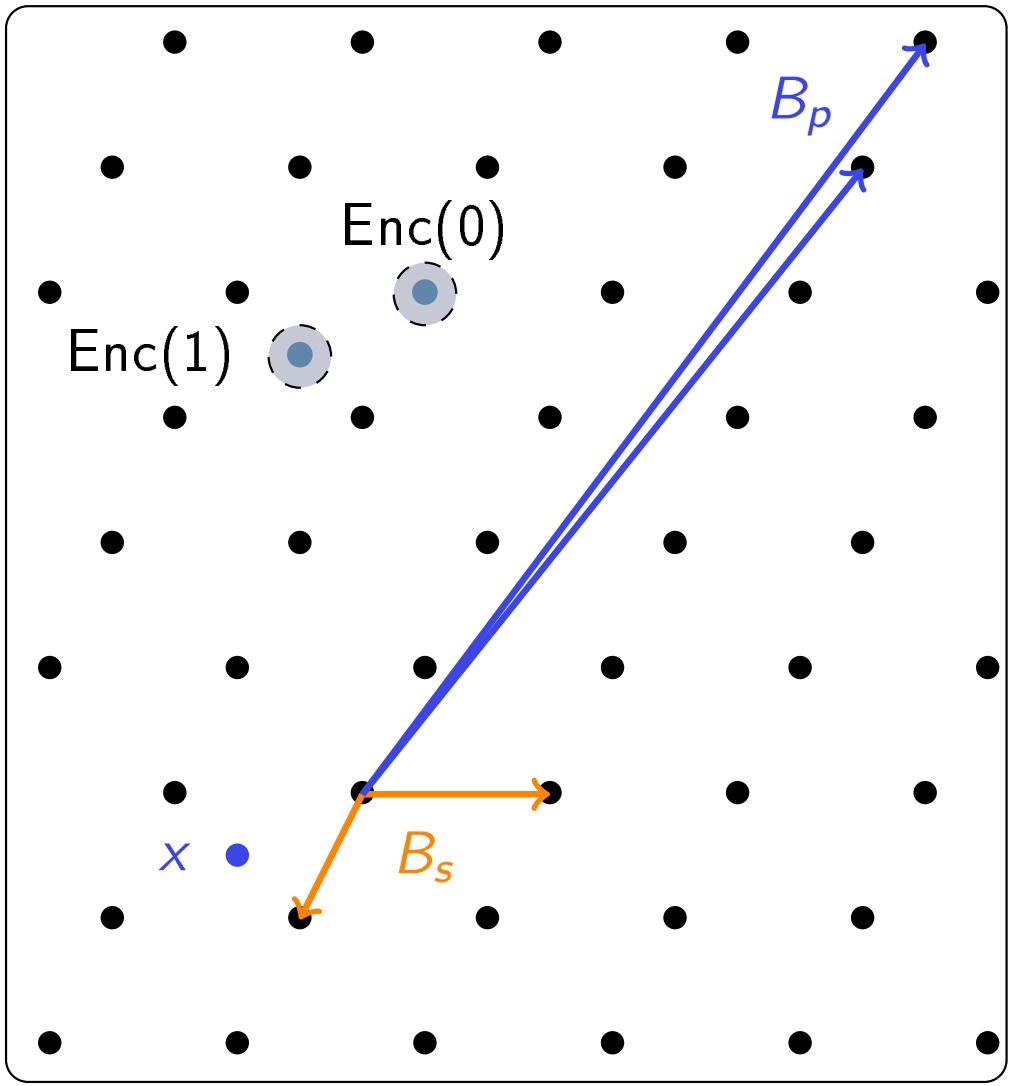
$\mathrm{sk} = B_s$

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

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

- Sample $v \xleftarrow{\$} \mathcal{L}$ (using B_p)
- if $m = 1$: $v \leftarrow v + x$

What is happening with pictures?



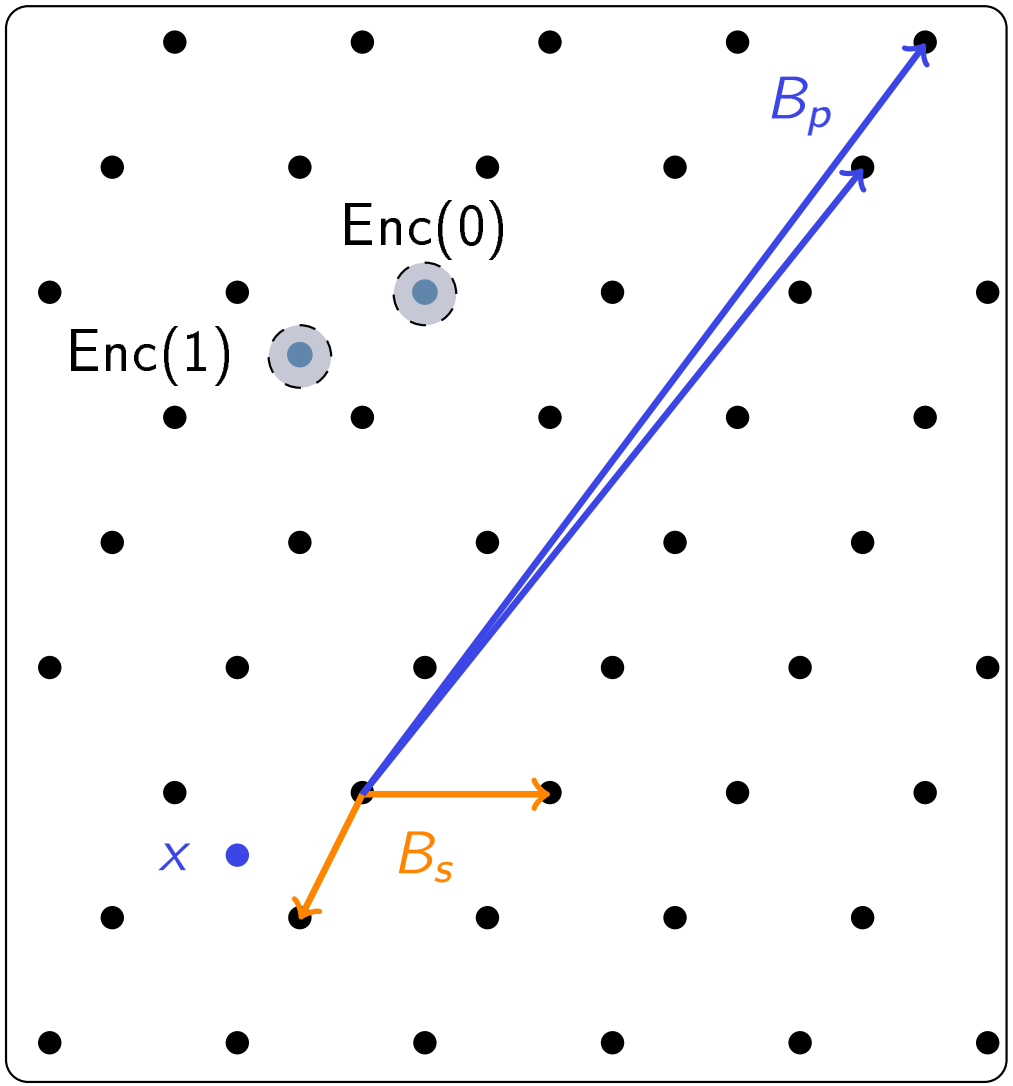
$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

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

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

- Sample $v \xleftarrow{\$} \mathcal{L}$ (using B_p)
- if $m = 1$: $v \leftarrow v + x$
- Sample small $e \in \mathbb{R}^n$
- return $c = v + e$

What is happening with pictures?



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

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

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

- Sample $v \xleftarrow{\$} \mathcal{L}$ (using B_p)
- if $m = 1$: $v \leftarrow v + x$
- Sample small $e \in \mathbb{R}^n$
- return $c = v + e$

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

- $s \leftarrow$ decode c using sk
- if $\|s - c\|$ small $\rightsquigarrow$ $m = 0$
- else $\rightsquigarrow$ $m = 1$

Exercise

Question: Relate the Regev-like encryption scheme from Slide 24 to the pictures from Slide 27

- what is $\mathcal{L}$?
- what are B_p and x?
- what is B_s? $\rightsquigarrow$ actually, there is no B_s, but something plays its role

Exercise

Question: Relate the Regev-like encryption scheme from Slide 24 to the pictures from Slide 27

- what is $\mathcal{L}$?
- what are B_p and x?
- what is B_s? $\rightsquigarrow$ actually, there is no B_s, but something plays its role

Solution:

- let $\bar{A} = \left[\begin{array}{c|c} A & b \end{array}\right] \in \mathbb{Z}^{n\times(n+1)}$, then

$$\mathcal{L} = \{v \in \mathbb{Z}^{n+1} \mid \exists x \in \mathbb{Z}^n,\ v = x\bar{A} \bmod q\}$$

- $B_p = (A, q)$ (fully describes $\mathcal{L}$) and $x = (0, \ldots, 0, \lfloor q/2 \rceil)$
- $B_s = \bar{s} := (s, -1) \in \mathbb{Z}^{n+1}$ $\rightsquigarrow$ this is not a basis of $\mathcal{L}$ but allows to distinguish points that are close to $\mathcal{L}$ from points that are far from $\mathcal{L}$

Kyber

Kyber key exchange:

Regev-like public key encryption

(in its KEM version)

+

module variant of LWE

(see next part)

Kyber is the only post-quantum key encapsulation mechanism standardized by the NIST in 2022.

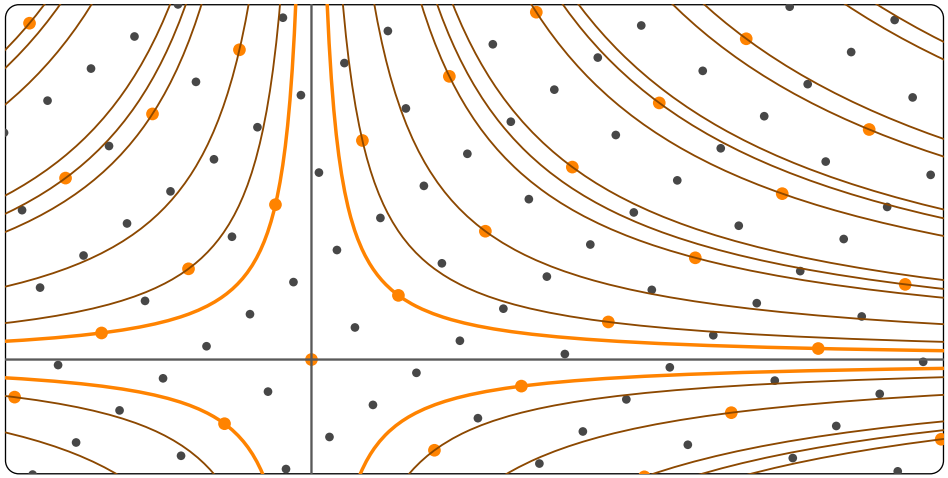
[Kyber] Avanzi, Bos, Ducas, Kiltz, Lepoint, Lyubashevsky, Schanck, Schwabe, Seiler, Stehlé. NIST standard

Advanced constructions

One can construct many advanced primitives from lattices:

- (fully) homomorphic encryption
- identity based encryption
- functional encryption for linear functions
- ...

Algebraic lattices



NTRU [HPS98]

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

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

NTRU [HPS98]

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

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

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

NTRU [HPS98]

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

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

NTRU assumption: given h, there is no efficient algorithm that can find u and v such that $|u|, |v| \leq B$ and $h = u \cdot v^{-1} \bmod q$

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

NTRU [HPS98]

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

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

NTRU assumption: given h, there is no efficient algorithm that can find u and v such that $|u|, |v| \leq B$ and $h = u \cdot v^{-1} \bmod q$

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

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

NTRU [HPS98]

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

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

NTRU assumption: given h, there is no efficient algorithm that can find u and v such that $|u|, |v| \leq B$ and $h = u \cdot v^{-1} \bmod q$

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

- we should replace integers by polynomials in $\mathbb{Z}[X]/(X^n + 1)$
  ($n = 512$ or 1024)

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

Some useful links

Learning about lattices

- Many references on the lattice club webpage
- "Complexity of lattice problems", by Micciancio and Goldwasser

  Lattices and lattice problems (not so much of crypto)
- "A decade of lattice-based cryptography", by Chris Peikert

  Survey of lattice-based crypto from its beginning to ≈ 2015. A bit old but very nice if you need some basic result. Be careful about historical claims made in this survey, the author tends to be quite biased towards his own articles (always double check with other sources to be sure who did what first).
- Lecture notes by Katharina Boudgoust

  Shorter than the previous 2 references, very nice introduction to many aspects of lattice-based crypto. Also one of the only reference mentioning ideal and module lattices.

Other links related to cryptography

International Association for Cryptographic Research (IACR)

- IACR website
- ePrint (finding research articles)
- job announcements (PhD, post-doc, ...)

Youtube channels

- IACR youtube channel (recorded conference talks)
- De Cifris channel: Trends in modern cryptography (2022, 2023, 2024)
  (collection of small courses on various topics of crypto)

For women

- Women in cryptography (discord channel, coffee breaks, seminars)