Euclidean lattices in cryptography

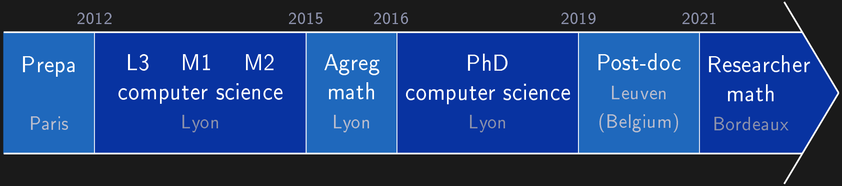
Alice Pellet--Mary

CNRS and Université de Bordeaux

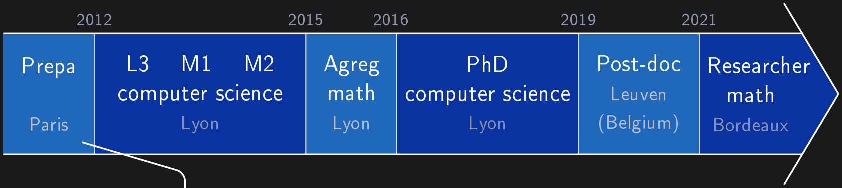
MARGAUx PhD days

Bordeaux

What I did until now



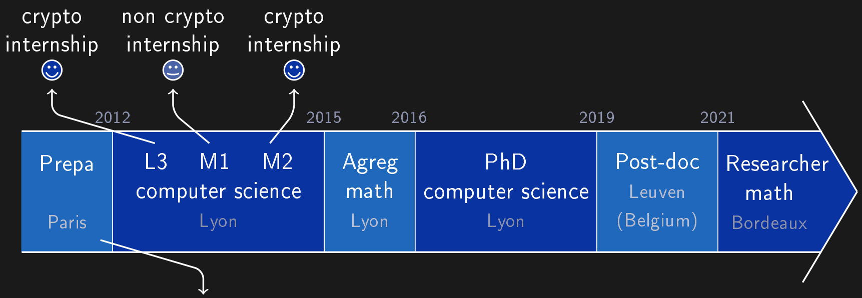
What I did until now



"If you want to do research, go for computer science, they have more money"

math teacher

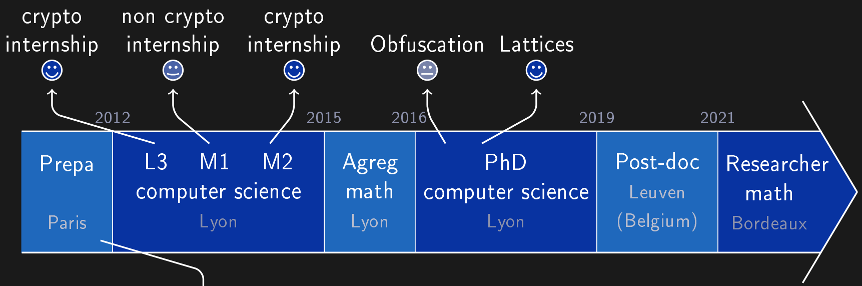
What I did until now



"If you want to do research, go for computer science, they have more money"

math teacher

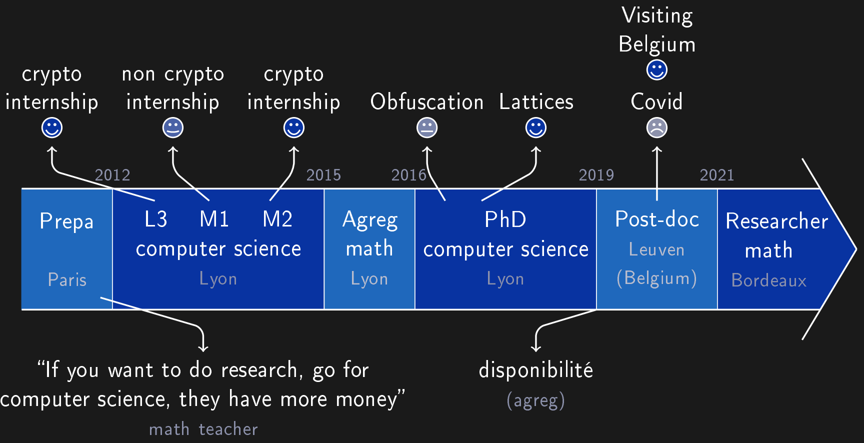
What I did until now



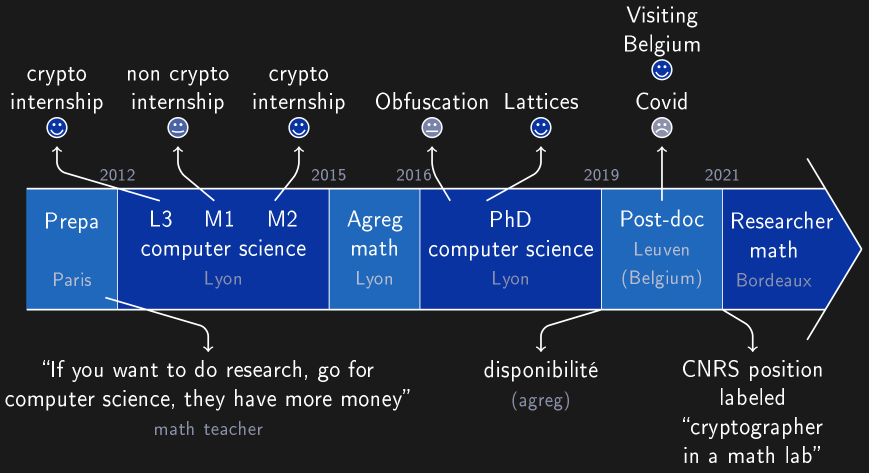
"If you want to do research, go for computer science, they have more money"

math teacher

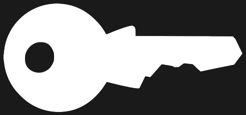
What I did until now



What I did until now



Cryptography



Cryptography

(Cryptology =) Cryptography = science of secrets

Examples: encryption, signatures, homomorphic encryption, e-voting . . .

Cryptography

(Cryptology =) Cryptography = science of secrets

Examples: encryption, signatures, homomorphic encryption, e-voting . . .

A cryptographic primitive is mathematically defined by

- some correctness properties
- some security properties
  - need to model the attacker

Encryption: symmetric

Alice	Bob
s $(\in \mathbb{Z})$	s $(\in \mathbb{Z})$

Encryption: symmetric

Alice

s $(\in \mathbb{Z})$

$c = \mathrm{Enc}(m, s)$

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

Bob

s $(\in \mathbb{Z})$

Encryption: symmetric

Alice		Bob
s $(\in \mathbb{Z})$		s $(\in \mathbb{Z})$
$c = \text{Enc}(m, s)$		
(message $m \in \{0, 1\}$)	$\xrightarrow{c}$	

Encryption: symmetric

Alice		Bob
s $(\in \mathbb{Z})$		s $(\in \mathbb{Z})$
$c = \mathrm{Enc}(m, s)$		
(message $m \in \{0,1\}$)	$\xrightarrow{\quad c \quad}$	
		$m' = \mathrm{Dec}(c, s)$

Encryption: symmetric

Alice		Bob
s $(\in \mathbb{Z})$		s $(\in \mathbb{Z})$
$c = \mathrm{Enc}(m, s)$		
(message $m \in \{0,1\}$)	$\xrightarrow{c}$	
		$m' = \mathrm{Dec}(c, s)$

Correctness:

$$\forall m \in \{0,1\}, \;\; \mathrm{Dec}\Big(\mathrm{Enc}(m,s), s\Big) = m$$

Encryption: symmetric

Alice		Bob
s $(\in \mathbb{Z})$		s $(\in \mathbb{Z})$
$c = \mathrm{Enc}(m, s)$		
(message $m \in \{0,1\}$)	$\xrightarrow{c}$	
		$m' = \mathrm{Dec}(c, s)$

Correctness:

$$\forall m \in \{0,1\}, \;\; \mathrm{Dec}\Big(\mathrm{Enc}(m,s), s\Big) = m$$

Security: (against chosen plaintext attacks)
for any polynomial time algorithm $\mathcal{A}$,

$$\Pr_{m \leftarrow \mathcal{U}(\{0,1\})}\Big(\mathcal{A}(\mathrm{Enc}(m,s)) = m\Big) = 1/2 + \varepsilon$$

(usually require $|\varepsilon| \leq 2^{-128}$)

Encryption: asymmetric [DH76, RSA78]

Alice		Bob
	$\xleftarrow{\;pk\;}$	sk, pk $(\in \mathbb{Z})$

[DH76] Diffie, Hellman. New Directions in Cryptography.

[RSA78] Rivest, Shamir, Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

Alice		Bob
	$\xleftarrow{pk}$	sk, pk $(\in \mathbb{Z})$
$c = \mathrm{Enc}(m, pk)$ (message $m \in \{0,1\}$)	

[DH76] Diffie, Hellman. New Directions in Cryptography.

[RSA78] Rivest, Shamir, Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

Encryption: asymmetric [DH76, RSA78]

Alice		Bob
	$\xleftarrow{\quad pk \quad}$	sk, pk $(\in \mathbb{Z})$
$c = \mathrm{Enc}(m, pk)$ (message $m \in \{0,1\}$)	$\xrightarrow{\quad c \quad}$

[DH76] Diffie, Hellman. New Directions in Cryptography.

[RSA78] Rivest, Shamir, Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

Encryption: asymmetric [DH76, RSA78]

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

[DH76] Diffie, Hellman. New Directions in Cryptography.

[RSA78] Rivest, Shamir, Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

Encryption: asymmetric [DH76, RSA78]

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

Correctness:

$$\forall m \in \{0,1\}, \;\; \mathrm{Dec}\Big(\mathrm{Enc}(m, pk), sk\Big) = m$$

[DH76] Diffie, Hellman. New Directions in Cryptography.

[RSA78] Rivest, Shamir, Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

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

Correctness:

$$\forall m \in \{0,1\}, \;\; \mathrm{Dec}\left(\mathrm{Enc}(m, pk), sk\right) = m$$

Security: (against chosen plaintext attacks)

for any polynomial time algorithm $\mathcal{A}$,

$$\Pr_{m \leftarrow \mathcal{U}(\{0,1\})}\Big(\mathcal{A}\big(pk, \mathrm{Enc}(m, pk)\big) = m\Big) = 1/2 + \varepsilon$$

[DH76] Diffie, Hellman. New Directions in Cryptography.

[RSA78] Rivest, Shamir, Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.

Security guarantees

For almost any interesting crypto primitives,
we don't know how to prove that a construction is secure

Security guarantees

For almost any interesting crypto primitives,
we don't know how to prove that a construction is secure

Solution: we rely on the supposed hardness of some algorithmic problems.
Ideally, we want

- few underlying problems
- that are simple to describe

Security guarantees

For almost any interesting crypto primitives,
we don't know how to prove that a construction is secure

Solution: we rely on the supposed hardness of some algorithmic problems.
Ideally, we want

- few underlying problems
- that are simple to describe

Examples: factoring, discrete logarithm, ...

Security guarantees

For almost any interesting crypto primitives,
we don't know how to prove that a construction is secure

Solution: we rely on the supposed hardness of some algorithmic problems.
Ideally, we want

- few underlying problems
- that are simple to describe

Examples: factoring, discrete logarithm, ...

Definition: an algorithmic problem is hard if there is no algorithm solving it in polynomial time

Foundation of asymmetric cryptography

Cryptographic primitives

asymmetric encryption | signature | homomorphic encryption | ...

error correcting codes | lattices | isogenies

factoring | discrete logarithm | ...

(Supposedly hard) algorithmic problems

Foundation of asymmetric cryptography

Cryptographic primitives

asymmetric encryption | signature | homomorphic encryption | ...

error correcting codes | lattices | isogenies

~~factoring~~ | ~~discrete logarithm~~ | ...

(Supposedly hard) algorithmic problems in a quantum world

Foundation of asymmetric cryptography

Cryptographic primitives

asymmetric encryption | signature | homomorphic encryption | ...

error correcting codes | lattices | isogenies

~~factoring~~ | ~~discrete logarithm~~ | ...

(Supposedly hard) algorithmic problems in a quantum world

My research: classify lattice problems

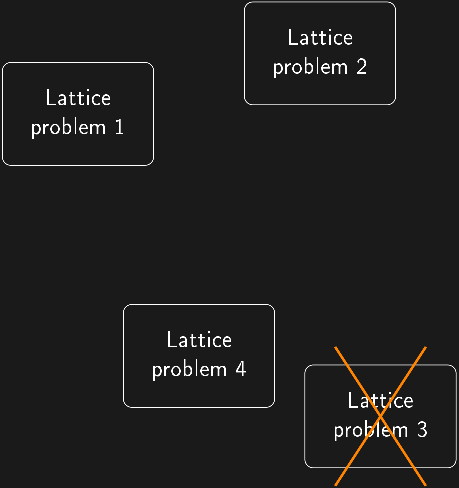
Lattice problem 2

Lattice problem 1

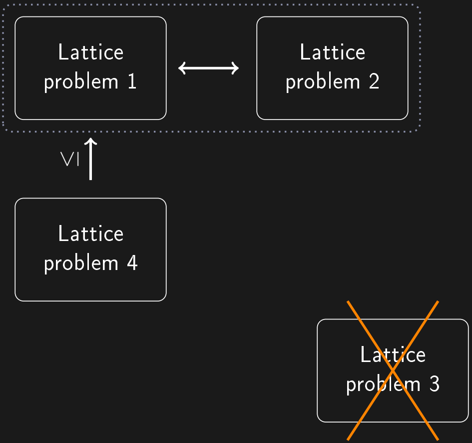
Lattice problem 3

Lattice problem 4

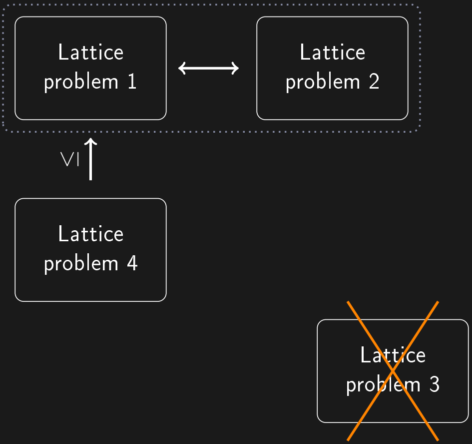
My research: classify lattice problems



My research: classify lattice problems



My research: classify lattice problems

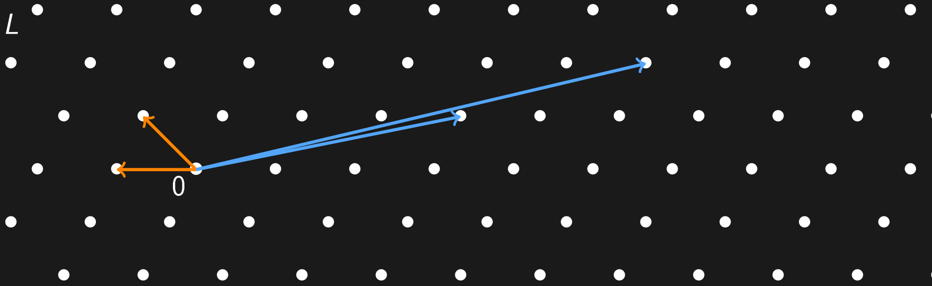


Tools:

- algorithms
- number theory
- a little programming
  (just toy examples)

Lattices and algorithmic problems

Lattices

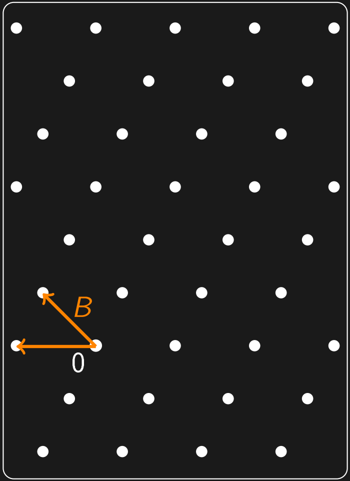


- $L = \{Bx \mid x \in \mathbb{Z}^n\}$ is a lattice
- $B \in \mathrm{GL}_n(\mathbb{R})$ is a basis
- n is the dimension of L

Representing a lattice

Representation of a lattice L:

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

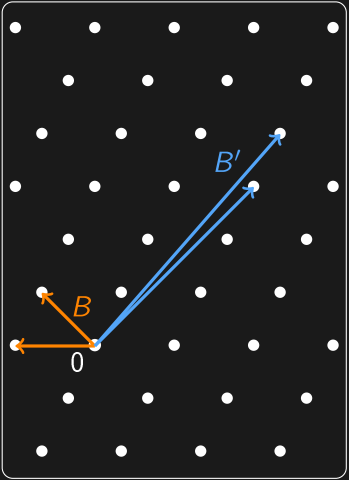


Representing a lattice

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

Difficulty:

- the basis B is not unique
- some choices of B may render some algorithmic problems easier



Representing a lattice

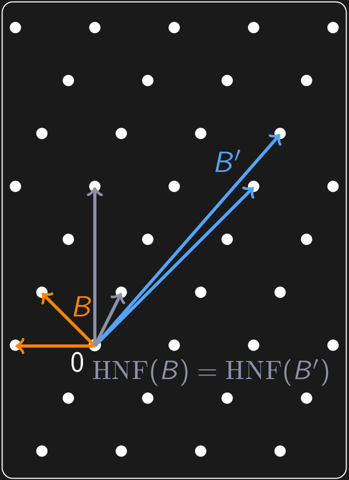
Representation of a lattice L:
a basis $B \in \mathbb{Z}^{n \times n}$ of L

Difficulty:

- the basis B is not unique
- some choices of B may render some algorithmic problems easier

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

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



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

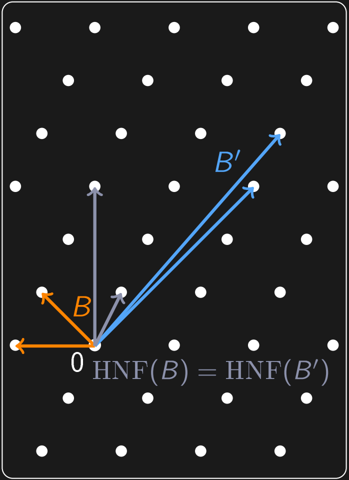
Difficulty:

- the basis B is not unique
- some choices of B may render some algorithmic problems easier

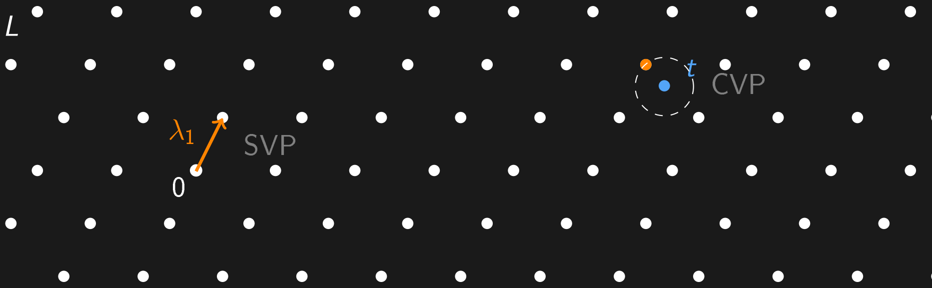
Solution: take the Hermite Normal Form (HNF) of any B

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

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



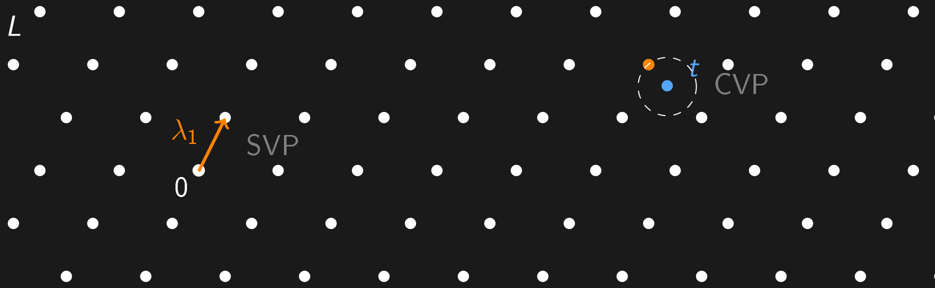
Algorithmic problems



SVP : Shortest Vector Problem
(input: HNF basis of L)

CVP : Closest Vector Problem
(input: HNF basis of L and target t)

Algorithmic problems



SVP : Shortest Vector Problem
(input: HNF basis of L)

CVP : Closest Vector Problem
(input: HNF basis of L and target t)

Supposedly hard to solve when n is large
(even with a quantum computer)

How hard is SVP/CVP?

In theory: best algorithm has asymptotic complexity $2^{c \cdot n + o(n)}$
(for some $c \approx 0.292$, or $c \approx 0.265$ for quantum computers [Laa15])

$\Rightarrow$ not polynomial

[Laa15] Laarhoven. Search problems in cryptography.

How hard is SVP/CVP?

In theory: best algorithm has asymptotic complexity $2^{c \cdot n + o(n)}$
(for some $c \approx 0.292$, or $c \approx 0.265$ for quantum computers [Laa15])

$$\Rightarrow \text{not polynomial}$$

In practice:

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

[Laa15] Laarhoven. Search problems in cryptography.

How hard is SVP/CVP?

In theory: best algorithm has asymptotic complexity $2^{c \cdot n + o(n)}$
(for some $c \approx 0.292$, or $c \approx 0.265$ for quantum computers [Laa15])

$\Rightarrow$ not polynomial

In practice:

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

[Laa15] Laarhoven. Search problems in cryptography.

How hard is SVP/CVP?

In theory: best algorithm has asymptotic complexity $2^{c \cdot n + o(n)}$

(for some $c \approx 0.292$, or $c \approx 0.265$ for quantum computers [Laa15])

$\Rightarrow$ not polynomial

In practice:

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

[Laa15] Laarhoven. Search problems in cryptography.

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

How hard is SVP/CVP?

In theory: best algorithm has asymptotic complexity $2^{c \cdot n + o(n)}$
(for some $c \approx 0.292$, or $c \approx 0.265$ for quantum computers [Laa15])

$\Rightarrow$ not polynomial

In practice:

- $n = 2 \rightsquigarrow$ easy, very efficient in practice
- up to $n = 60$ or $n = 80 \rightsquigarrow$ a few minutes on a personal laptop
- up to $n = 180 \rightsquigarrow$ few days on big computers with good code [DSW21]
- from $n = 500$ to $n = 1000 \rightsquigarrow$ cryptography

[Laa15] Laarhoven. Search problems in cryptography.

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

The zoo of lattice problems

Exact solution vs Approximation

- find a shortest vector
- find a vector $\leq \gamma$

The zoo of lattice problems

Exact solution	vs	Approximation
▶ find a shortest vector		▶ find a vector $\leq \gamma$
Search	vs	Decision
▶ find a short vector		▶ decide whether there is a vector of length $\leq t$

The zoo of lattice problems

Exact solution vs Approximation

- find a shortest vector
- find a vector $\leq \gamma$

Search vs Decision

- find a short vector
- decide whether there is a vector of length $\leq t$

Worst-case vs Average-case

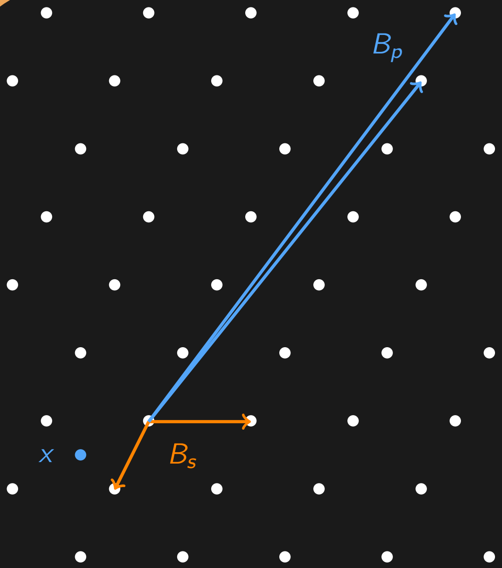
- find a short vector in any possible input lattice L
- find a short vector with good probability (when L is random)

The zoo of lattice problems

Exact solution	vs	**Approximation**
▶ find a shortest vector		▶ find a vector $\leq \gamma$
Search	vs	**Decision**
▶ find a short vector		▶ decide whether there is a vector of length $\leq t$
Worst-case	vs	**Average-case**
▶ find a short vector in any possible input lattice L		▶ find a short vector with good probability (when L is random)
Plain lattices	vs	**Algebraic lattice**
▶ find a short vector in a lattice over $\mathbb{Z}$		▶ find a short vector in a lattice over $\mathcal{O}_K$ (ring of integers of number field K)

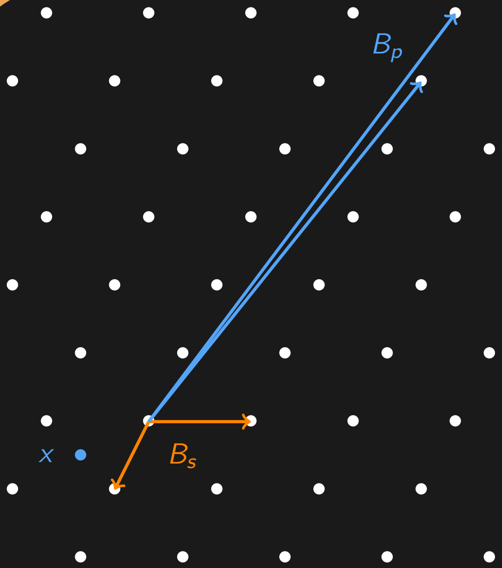
Digression: building cryptography from lattices

Asymmetric encryption from lattices



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

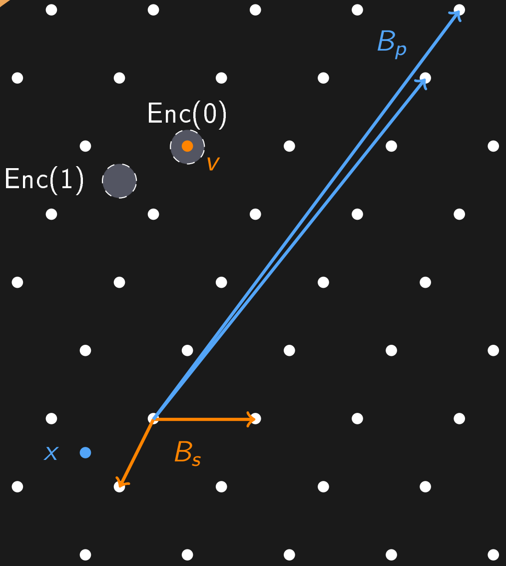
Asymmetric encryption from lattices



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

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

Asymmetric encryption from lattices



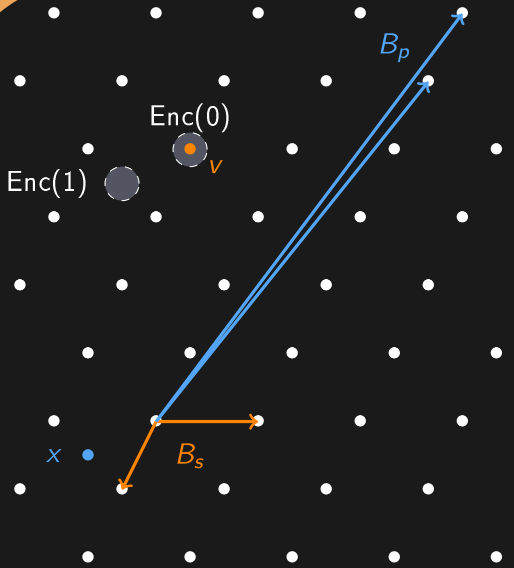
$\mathrm{pk} = (B_p, x)$
$\mathrm{sk} = B_s$

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

Encryption(m, pk):

- sample random $v \in L$
- sample small $e \in \mathbb{R}^n$
- return $c = v + e + m \cdot x$

Asymmetric encryption from lattices



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

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

Encryption(m, pk):

- sample random $v \in L$
- sample small $e \in \mathbb{R}^n$
- return $c = v + e + m \cdot x$

Decryption(c, sk):

- find $w \in L$ closest to c
- if c is very close to w, return $m = 0$
- otherwise return $m = 1$

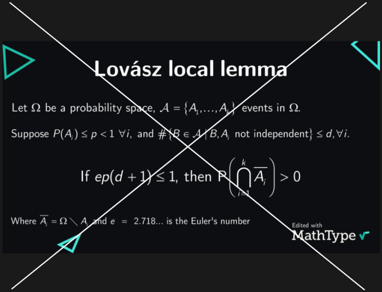
Correctness and security

Theorem

The encryption construction is correct and secure assuming that the problem decision-CVP is hard.

decision-CVP: given the HNF basis of L and a target t, decide whether t is close to a point of L or not.

The LLL algorithm



Objective

The LLL algorithm

- runs in polynomial time
- finds a vector $v \in L$ with $\|v\|_2 \leq 2^n \cdot \lambda_1(L)$ $\quad (\lambda_1(L) = \min_{\substack{w \in L \\ w \neq 0}} \|w\|_2)$

[LLL82] A. K. Lenstra, H. W. Lenstra, L. Lovász. Factoring polynomials with rational coefficients.

Dimension 2: Lagrange-Gauss algorithm

video

Dimension 2: Lagrange-Gauss algorithm

video

Theorem: the algorithm

- finds a shortest vector of L
- runs in polynomial time

Larger dimension: LLL algorithm

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

Larger dimension: LLL algorithm

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

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

(using Lagrange-Gauss algorithm)

Larger dimension: LLL algorithm

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 lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

Larger dimension: LLL algorithm

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 lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- finds $v \in L$ with $\|v\|_2 \leq 2^n \cdot \lambda_1(L)$

Larger dimension: LLL algorithm

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 lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- finds $v \in L$ with $\|v\|_2 \leq 2^n \cdot \lambda_1(L)$
- does not run in polynomial time

Larger dimension: LLL algorithm

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

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 lattice spanned by (b_i, b_{i+1}))
  - run Lagrange-Gauss on L_i

This algorithm

- finds $v \in L$ with $\|v\|_2 \leq 2^n \cdot \lambda_1(L)$
- runs in polynomial time

LLL over a number field?

Can we adapt LLL to lattices over $\mathcal{O}_K$?

[LPSW19] Lee, Pellet-Mary, Stehlé, Wallet. An LLL algorithm for module lattices.

Can we adapt LLL to lattices over $\mathcal{O}_K$?

For LLL we need:

- QR factorization $\Rightarrow$ ok

[LPSW19] Lee, Pellet-Mary, Stehlé, Wallet. An LLL algorithm for module lattices.

LLL over a number field?

Can we adapt LLL to lattices over $\mathcal{O}_K$?

For LLL we need:

- QR factorization $\Rightarrow$ ok
- Euclidean division $\Rightarrow$ not ok ($\mathcal{O}_K$ is usually not Euclidean)

[LPSW19] Lee, Pellet-Mary, Stehlé, Wallet. An LLL algorithm for module lattices.

LLL over a number field?

Can we adapt LLL to lattices over $\mathcal{O}_K$?

For LLL we need:

- QR factorization $\Rightarrow$ ok
- Euclidean division $\Rightarrow$ not ok   ($\mathcal{O}_K$ is usually not Euclidean)

(Partial) solution: use pseudo-euclidean division

($|au + bv| < 1/2 \cdot |a|$ and $|v|$ not too big   instead of $|au + v| < 1/2 \cdot |a|$)

[LPSW19] Lee, Pellet-Mary, Stehlé, Wallet. An LLL algorithm for module lattices.

LLL over a number field?

Can we adapt LLL to lattices over $\mathcal{O}_K$?

For LLL we need:

- QR factorization $\Rightarrow$ ok
- Euclidean division $\Rightarrow$ not ok ($\mathcal{O}_K$ is usually not Euclidean)

(Partial) solution: use pseudo-euclidean division

($|au + bv| < 1/2 \cdot |a|$ and $|v|$ not too big instead of $|au + v| < 1/2 \cdot |a|$)

- the LLL algorithm still works with pseudo-division ☺

[LPSW19] Lee, Pellet-Mary, Stehlé, Wallet. An LLL algorithm for module lattices.

LLL over a number field?

Can we adapt LLL to lattices over $\mathcal{O}_K$?

For LLL we need:

- QR factorization $\Rightarrow$ ok
- Euclidean division $\Rightarrow$ not ok   ($\mathcal{O}_K$ is usually not Euclidean)

(Partial) solution: use pseudo-euclidean division

($|au + bv| < 1/2 \cdot |a|$ and $|v|$ not too big   instead of $|au + v| < 1/2 \cdot |a|$)

- the LLL algorithm still works with pseudo-division ☺
- computing the pseudo-division is super costly ☹

[LPSW19] Lee, Pellet-Mary, Stehlé, Wallet. An LLL algorithm for module lattices.

LLL over a number field?

Can we adapt LLL to lattices over $\mathcal{O}_K$?

For LLL we need:

- QR factorization $\Rightarrow$ ok
- Euclidean division $\Rightarrow$ not ok ($\mathcal{O}_K$ is usually not Euclidean)

(Partial) solution: use pseudo-euclidean division
($|au + bv| < 1/2 \cdot |a|$ and $|v|$ not too big instead of $|au + v| < 1/2 \cdot |a|$)

- the LLL algorithm still works with pseudo-division ☺
- computing the pseudo-division is super costly ☹

$\Rightarrow$ we obtain LLL over $\mathcal{O}_K$ but not polynomial time anymore

[LPSW19] Lee, Pellet-Mary, Stehlé, Wallet. An LLL algorithm for module lattices.

Conclusion

Take-away: crypto is fun!

(It is a good way to do nice math and have founding)

Conclusion

Take-away: crypto is fun!

(It is a good way to do nice math and have founding)

Advertisement:

- we are hiring a 2 years post-doc working on algebraic lattices
  (geometry of numbers, ideals in numbers fields, automorphic forms)
- concours Alkindi
  (Tip: very useful to keep a "stagiaire de 3eme" busy for a few hours)

Conclusion

Take-away: crypto is fun!

(It is a good way to do nice math and have founding)

Advertisement:

- we are hiring a 2 years post-doc working on algebraic lattices
  (geometry of numbers, ideals in numbers fields, automorphic forms)
- concours Alkindi
  (Tip: very useful to keep a “stagiaire de 3eme” busy for a few hours)

Thank you