Advanced Crypto 2024
Lattice-based cryptography

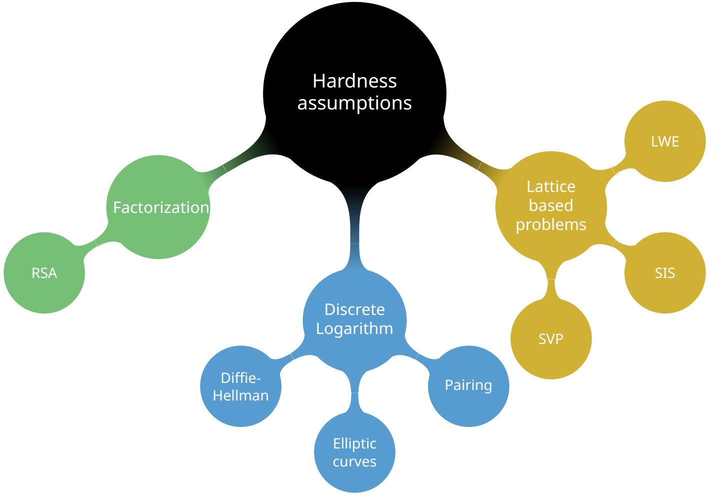
Léo Colisson Palais

leo.colisson-palais@univ-grenoble-alpes.fr

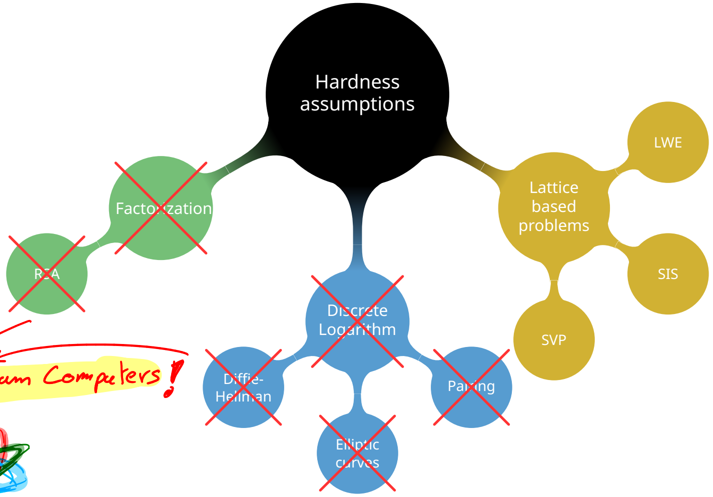
`https://leo.colisson.me/teaching.html`

Motivations

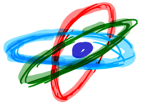
Motivations



Motivations



Broken by Quantum Computers !



Motivations

Motivations

- Should we change technology **now or** can we **wait** until quantum computers arrive?

Motivations

- Should we change technology **now or** can we **wait** until quantum computers arrive?

  $\Rightarrow$ **Cannot wait!** “Harvest now, decrypt later”

Motivations

- Should we change technology **now or** can we **wait** until quantum computers arrive?

  $\Rightarrow$ **Cannot wait!** “Harvest now, decrypt later”

- **Warning**: can’t change too quickly; **need enough time** to analyse the new candidate

Motivations

- Should we change technology **now or** can we **wait** until quantum computers arrive?

  $\Rightarrow$ **Cannot wait!** "Harvest now, decrypt later"

- **Warning**: can't change too quickly; **need enough time** to analyse the new candidate

  (RSA/ECDSA/...are much more studied than most post-quantum alternatives)

Motivations

- Should we change technology **now or** can we **wait** until quantum computers arrive?

  $\Rightarrow$ **Cannot wait!** "Harvest now, decrypt later"

- **Warning**: can't change too quickly; **need enough time** to analyse the new candidate

  (RSA/ECDSA/. . . are much more studied than most post-quantum alternatives)

  $\Rightarrow$ Creation of a **standardization competition** by NIST!

Motivations

- Should we change technology **now or** can we **wait** until quantum computers arrive?

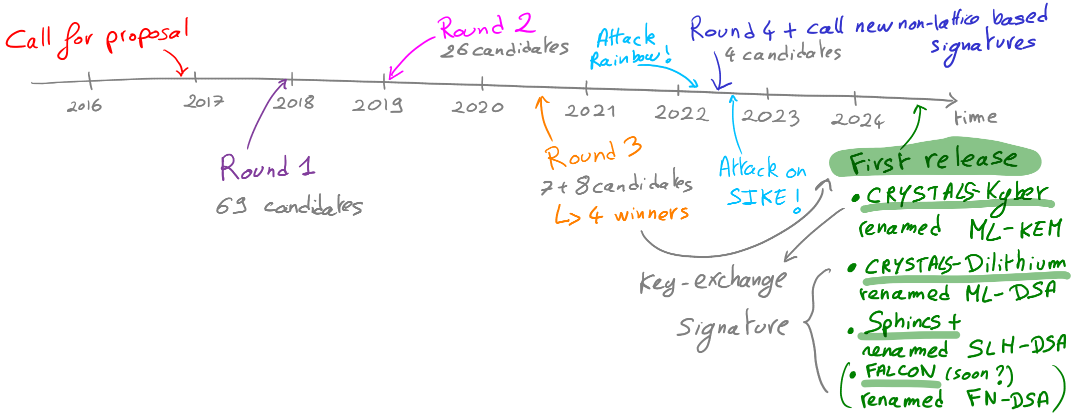
  $\Rightarrow$ **Cannot wait!** "Harvest now, decrypt later"

- **Warning**: can't change too quickly; **need enough time** to analyse the new candidate

  (RSA/ECDSA/…are much more studied than most post-quantum alternatives)

  $\Rightarrow$ Creation of a **standardization competition** by NIST!
  $\Rightarrow$ For now, safer to use it **on top** of non-post-quantum solutions!

NIST post-quantum cryptography standardization



NIST post-quantum cryptography standardization

First release (2024)

Key-exchange (for encryption)

Signature

	• CRYSTALS-Kyber renamed ML-KEM	• CRYSTALS-Dilithium renamed ML-DSA	• Sphincs+ renamed SLH-DSA
Hardness Assumption	Lattice-based	Lattice-based	Hash-based
In bytes, Level 3: $\lvert pk \rvert$:	1184	1952	48
$\lvert sk \rvert$:	2400	4032	96
$\lvert cipher \rvert$:	1088		
$\lvert shared\ key \rvert$:	32		
$\lvert Signature \rvert$:		3309	16224

Less efficient than ML-DSA
⇒ in case ML-DSA is broken

NIST post-quantum cryptography standardization

First release (2024)

Key-exchange (for encryption)

- CRYSTALS-Kyber

renamed ML-KEM

Compare with ECDH with Curve 25519 (Not post-quantum!)

In bytes, level 3

Hardness Assumption	Lattice-based ✓	Elliptic-curves ✗
$\lvert pk \rvert$:	1184 ✗	32 ✓
$\lvert sk \rvert$:	2400 ✗	32 ✓
$\lvert \text{cipher} \rvert$:	1088 ✗	64 (2×32) ✓
$\lvert \text{shared key} \rvert$:	32 ✗	32 ✓

NIST post-quantum cryptography standardization

First release (2024)

Key-exchange (for encryption)

- CRYSTALS-Kyber

renamed ML-KEM

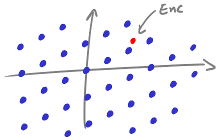
Compare with ECDH with Curve 25519 (Not post-quantum!)

Hardness Assumption	Lattice-based ✓	Elliptic-curves ✗
In bytes, level 3		
$\lvert pk \rvert$:	1184 ✗	32 ✓
$\lvert sk \rvert$:	2400 ✗	32 ✓
$\lvert cipher \rvert$:	1088 ✗	64 (2x 32) ✓
$\lvert shared\ key \rvert$:	32 ✗	32 ✓

Post-quantum is less efficient (but hopefully more secure)

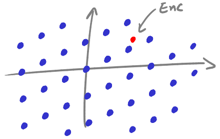
Famous post-quantum candidates

① Lattice-based Crypto

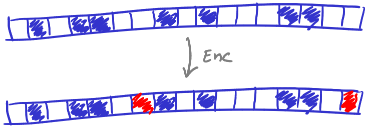


Famous post-quantum candidates

1. Lattice-based Crypto

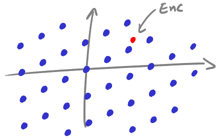


2. Code-based Crypto

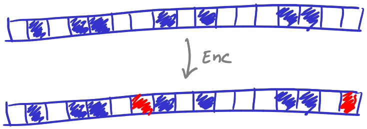


Famous post-quantum candidates

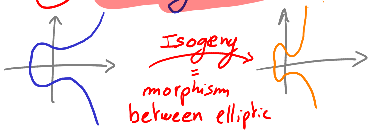
1 Lattice-based Crypto



2 Code-based Crypto



3 Isogenies

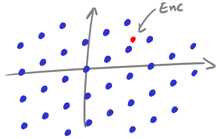


=

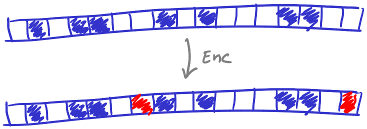
morphism between elliptic curves

Famous post-quantum candidates

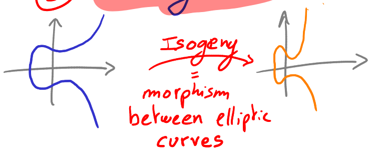
1 Lattice-based Crypto



2 Code-based Crypto



3 Isogenies

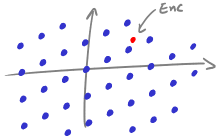


4 Multivariate Crypto

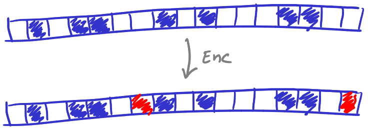
$$P_k := \begin{cases} \cdot\ 1 + x_1 + 2x_0x_3 \\ \cdot\ 4 + x_4 + 3x_1^2x_8 + x_9 \\ \cdot\ x_6 + x_2^3x_5 + x_7x_5 \end{cases} \xrightarrow{Enc} P_k(m)$$

Famous post-quantum candidates

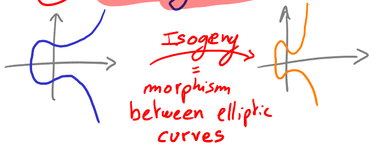
① Lattice-based Crypto



② Code-based Crypto



③ Isogenies



④ Multivariate Crypto

$$P_k := \begin{cases} \cdot\ 1 + x_1 + 2x_0x_3 \\ \cdot\ 4 + x_4 + 3x_1^2x_8 + x_9 \\ \cdot\ x_6 + x_2^3x_5 + x_7x_5 \end{cases} \xrightarrow{\text{Enc}} P_k(m)$$

⑤ + Symmetric crypto (incl. signatures)

Famous post-quantum candidates

① Lattice-based Crypto

- ✓ • studied extensively
- ✓ • efficient
- ✓ • simple
- ✓ • versatile (FHE ..)
- ✓ • hard also on average !

② Code-based Crypto

- ✓ • simple
- ✗ • no worst case ⟶ average case reduction
- ✗ • FHE impossible

③ Isogenies

- ✗ • SIDH broken ⟹ lost confidence
- ✗ • complicated

④ Multivariate Crypto

- ✗ • many candidates were broken ⟹ lost confidence

⑤ + Symmetric crypto (incl. signatures)

Famous post-quantum candidates

1. Lattice-based Crypto
   - ✓ studied extensively
   - ✓ efficient
   - ✓ simple
   - ✓ versatile (FHE ..)
   - ✓ hard also on average !

2. Code-based Crypto
   - ✓ simple
   - ✗ no worst case → average case reduction
   - ✗ FHE impossible

3. Isogenies
   - ✗ SIDH broken ⇒ lost confidence
   - ✗ complicated

4. Multivariate Crypto
   - ✗ many candidates were broken ⇒ lost confidence

5. + Symmetric crypto (incl. signatures)

Introduction to lattices

References

- Great survey: *A Decade of Lattice Cryptography*, Chris Peikert
- Course `https://people.csail.mit.edu/vinodv/COURSES/CSC2414-F11/`
- Course `https://www.di.ens.fr/brice.minaud/cours/2019/MPRI-3.pdf`
- Course `https://www.di.ens.fr/~pnguyen/SLIDES/SlidesLuminy2010.pdf`
- Course `https://www.youtube.com/watch?v=XEMEiBcwSKc`

Lattices: applications beyond cryptography

Lattices

Algorithms

LLL ⇒ many applications
↳ Integer Linear Programming
↳ Polynomial factorisation over rationals

Cryptography

↳ Attacks: LLL = break knapsack-based crypto, RSA (for some parameters), ECDSA (partially known nounces)...
↳ New cryptosystems
Encryption, signatures, FHE...

Complexity theory

Rare example of worst-case to average-case reduction

Number theory

↳ Disprove Mertens conjecture
...
↳ Many links: Minkowski's theorem, Functional analysis, Convex geometry

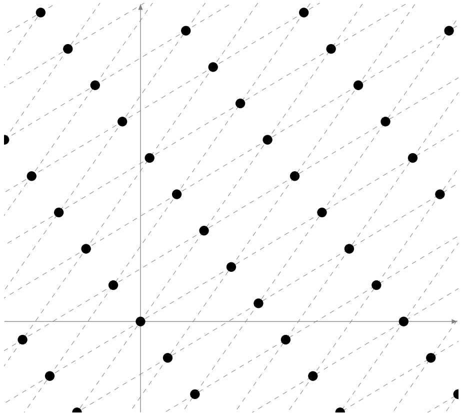
Lattices

Definition (Lattice)

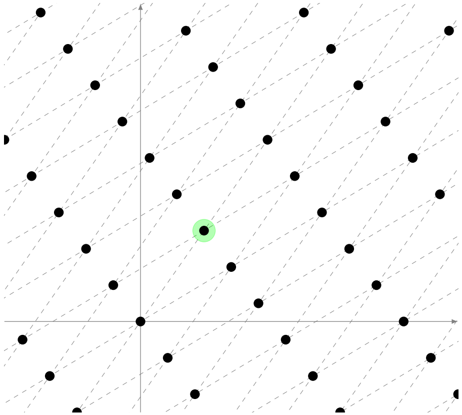
An n-dimensional *lattice* $\mathcal{L}$ is any subset of $\mathbb{R}^n$ that is both:

- an **additive subgroup**:
  $0 \in \mathcal{L}, \forall x, y \in \mathcal{L}, -x \in \mathcal{L}$ and $x + y \in \mathcal{L}$
- **discrete**:
  every $x \in \mathcal{L}$ has a neighbourhood in $\mathbb{R}^n$ in which x is the only lattice point

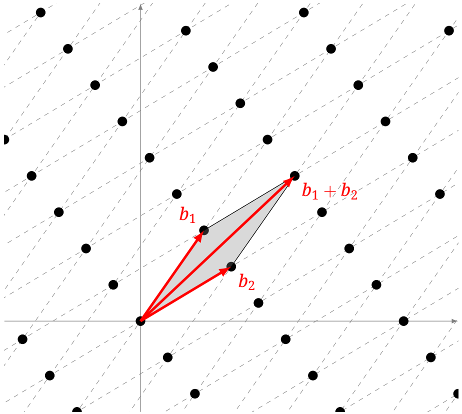
Lattice



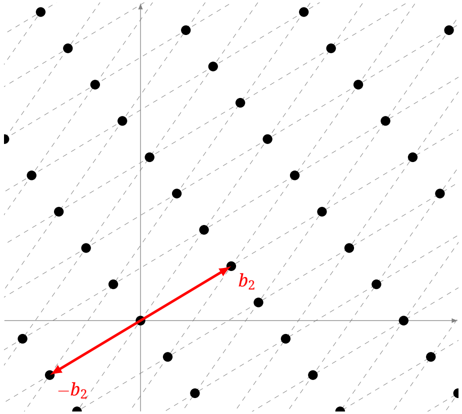
Lattice



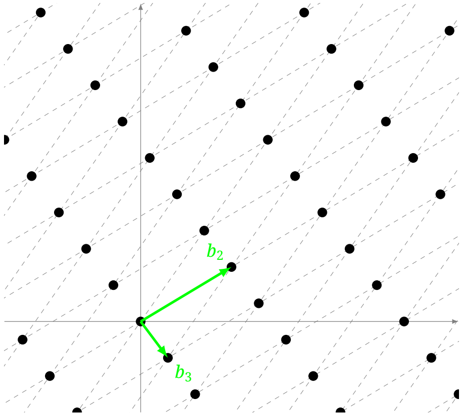
Lattice



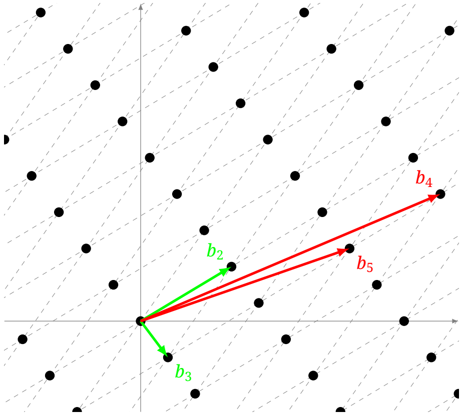
Lattice



Lattice

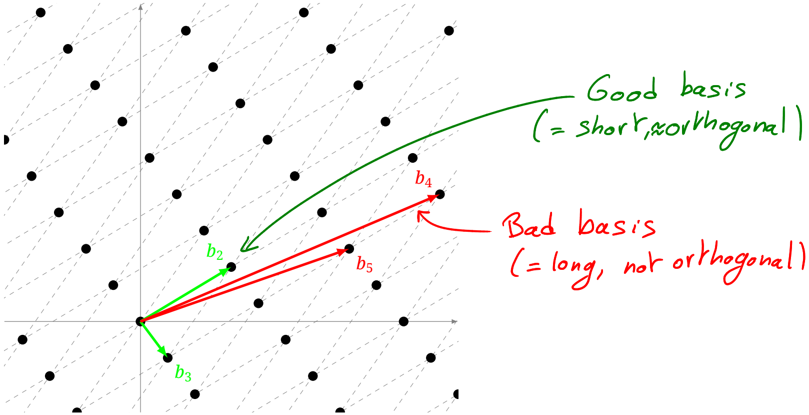


Lattice



Basis = not unique !

Lattice



b_2
b_3
b_4
b_5
Good basis
(= short, ≈orthogonal)
Bad basis
(= long, not orthogonal)

Lattice: basis

Definition (Basis)

If $\mathcal{L}$ is a lattice, then it admits a basis $\mathbf{B} = \begin{bmatrix}\mathbf{b}_1 & \dots & \mathbf{b}_k\end{bmatrix} \in \mathbb{R}^{n\times k}$ such that

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^k = \left\{ \sum_{i=1}^{k} z_i \mathbf{b}_i \right\}$$

k is the **rank** of the lattice. If $k = n$, the lattice has **full-rank** (often the case).

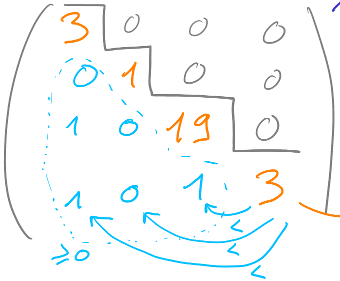
The basis is **not unique**: for any invertible matrix $\mathbf{U} \in \mathbb{Z}^{k\times k}$ s.t. $\mathbf{U}^{-1} \in \mathbb{Z}^{k\times k}$, $\mathbf{B} \cdot \mathbf{U}$ is also a basis of $\mathcal{L}(\mathbf{B})$.

Lattice: basis

So **which basis** to choose?

$\Rightarrow$ **Hermite normal form** can always be efficiently be computed and is unique: Good reference basis.

Form:



$$\begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 19 & 0 \\ 1 & 0 & 1 & 3 \end{pmatrix}$$

• lower triangular
+ columns of 0 to the right:
$\begin{pmatrix} \cdots & \begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix} \end{pmatrix}$

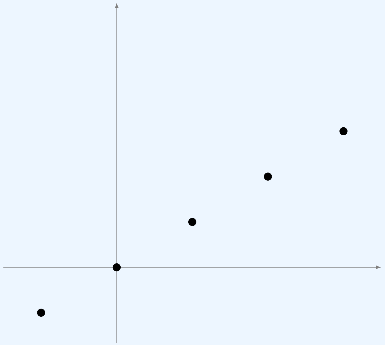
= right-most $\neq 0$ coef

pivots follow a "staircase" pattern
(never aligned, possibly not on the diagonal)

Lattice: basis

?

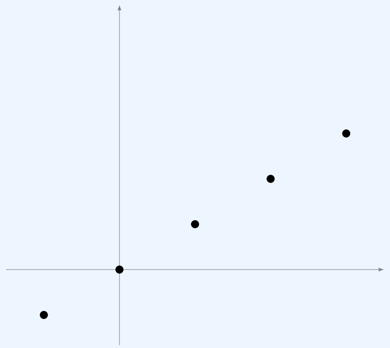
What is the dimension and rank of this lattice?



Lattice: basis

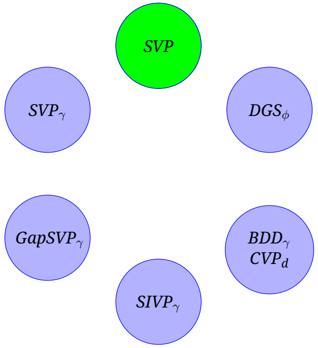
?

What is the dimension and rank of this lattice?

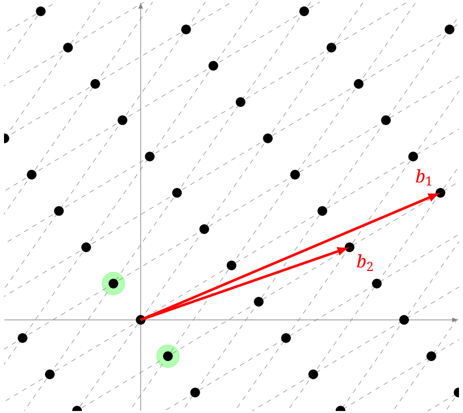


$\Rightarrow$ Dimension is 2, rank is 1

Lattice : what is hard to do?



Shortest Vector Problem

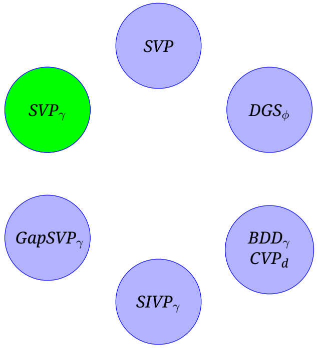


Compute basis B with the smallest length for each vector

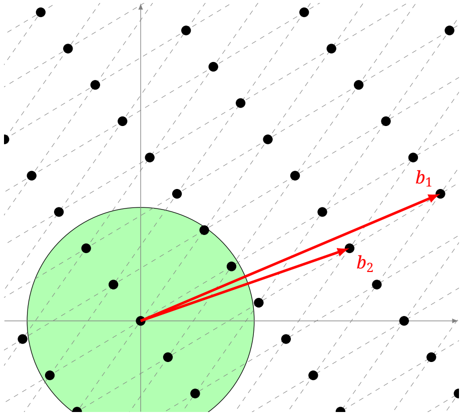
λ_i = length of i-th largest vector in B

Goal: Given a basis B of a lattice $\mathcal{L}$, find a vector $x \in \mathcal{L} \setminus \{0\}$ with the smallest norm $\lambda_1(\mathcal{L})$.

Lattice : what is hard to do?



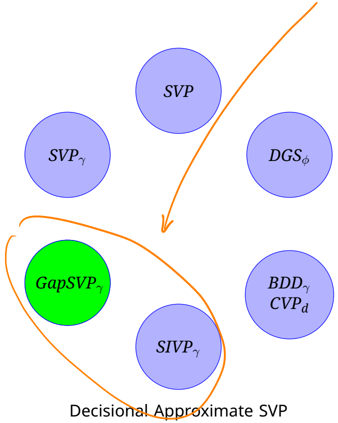
Approximate Shortest Vector Problem



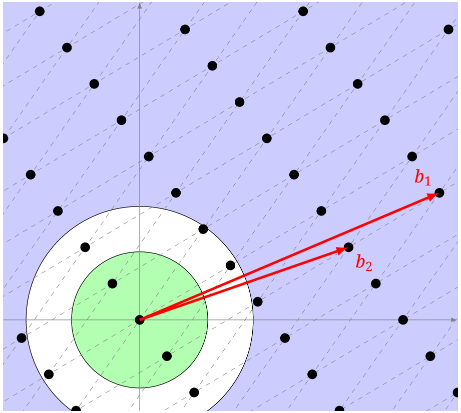
Goal: Given a basis B of a lattice $\mathcal{L}$, find a vector $x \in \mathcal{L} \setminus \{0\}$ s.t. $\|x\| \leq \gamma(n)\lambda_1(\mathcal{L})$.

Lattice : what is hard to do?

Hard to reduce to SVP/SVP$_\gamma$: most reductions reduce to GapSVP or SIVP

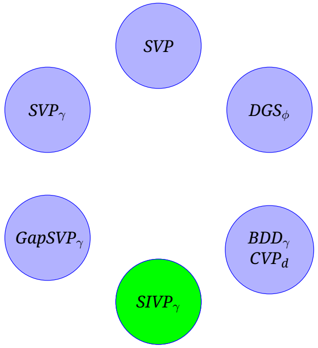


Decisional Approximate SVP

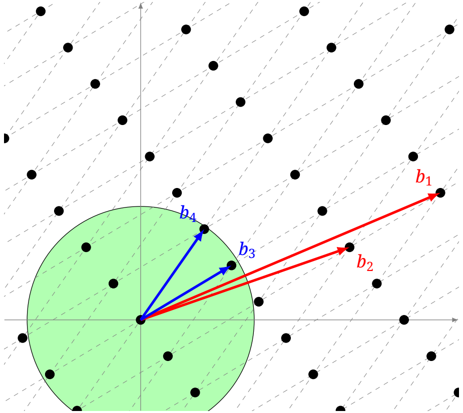


Goal: Given a basis B of a lattice $\mathcal{L}$, with the promise that $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma(n)$, determine which is the case.

Lattice : what is hard to do?

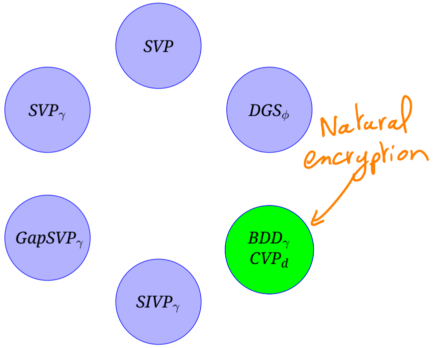


Approximate Shortests Independent Vectors Problem

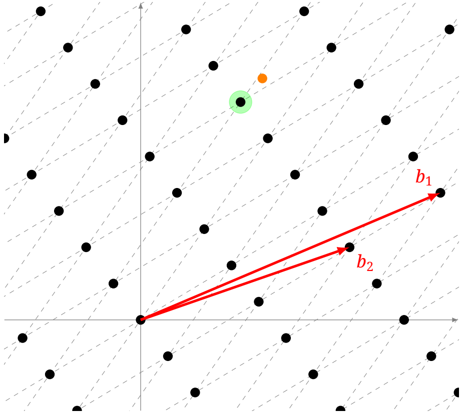


Goal: Given a basis B of a full-rank lattice $\mathcal{L}$, output a set $\{s_i\} \subset \mathcal{L}$ of n linearly independent lattice vectors where $\forall i, \|s_i\| \leq \gamma(n).\lambda_n(\mathcal{L})$.

Lattice : what is hard to do?



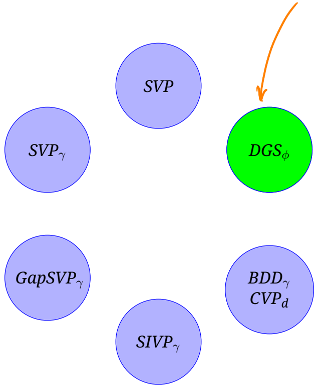
Bounded Distance Decoding Problem
and Closest Vector Problem



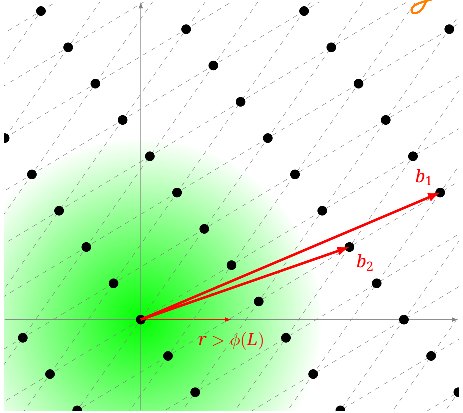
Goal: Given a basis B of a lattice $\mathcal{L}$ and a target $t \in \mathbb{R}^n$ s.t. $\text{dist}(t, \mathcal{L}) < d := \lambda_1(\mathcal{L})/(2\gamma(n))$, find the unique v s.t. $\|t - v\| < d$.

Lattice : what is hard to do?

Often used in proofs (e.g. LWE as hard as GapSVP$_\gamma$)
use DGS internally

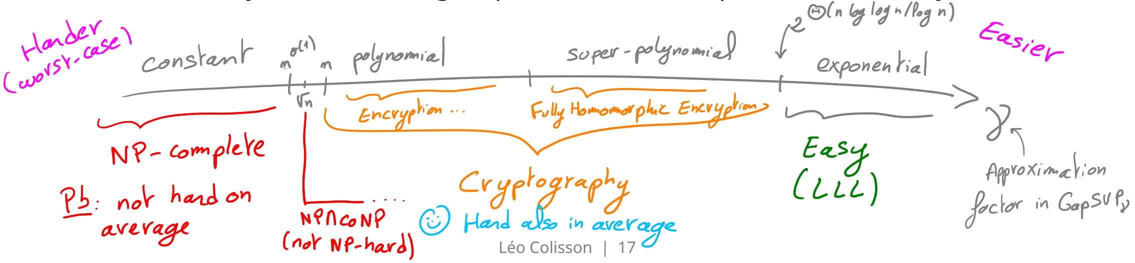


Discrete Gaussian
Sampling problem



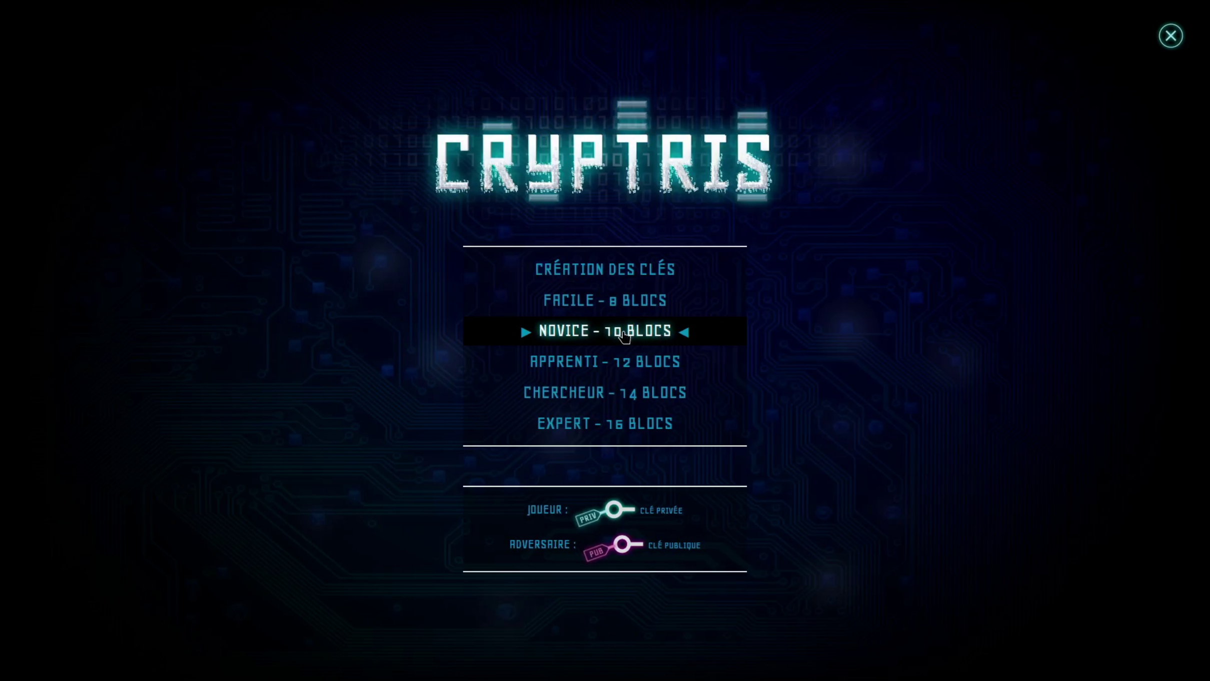
Lattice: Why is it hard

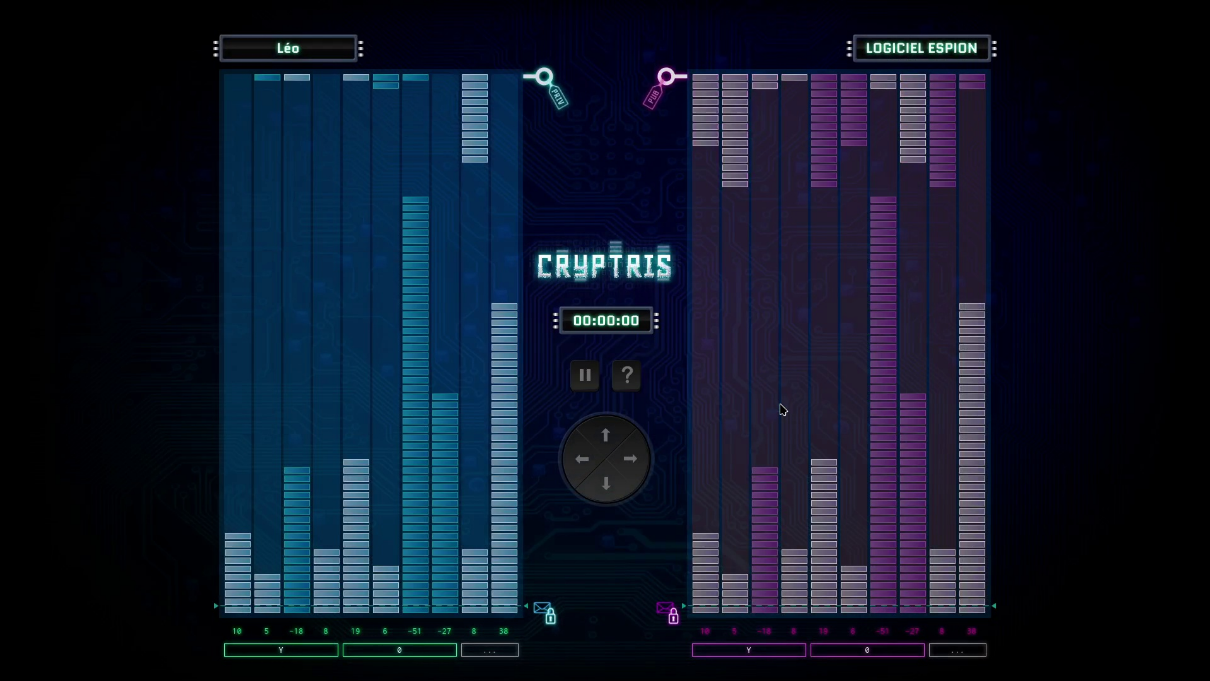
- Simple in dimension 2, **hard bigger dimensions**
- **Best known algorithm** (quantum and classical):
  - Typically Lenstra–Lenstra–Lovász (LLL): poly-time, but bad approximation factor (nearly exponential).
  - For smaller factors, Block Korkine-Zolotarev (BKZ) is often used, but runs in exponential time.
  - For exact versions (SVP): lattice enumeration (super-exponential time, poly memory), lattice sieving (exponential time, exponential memory)...

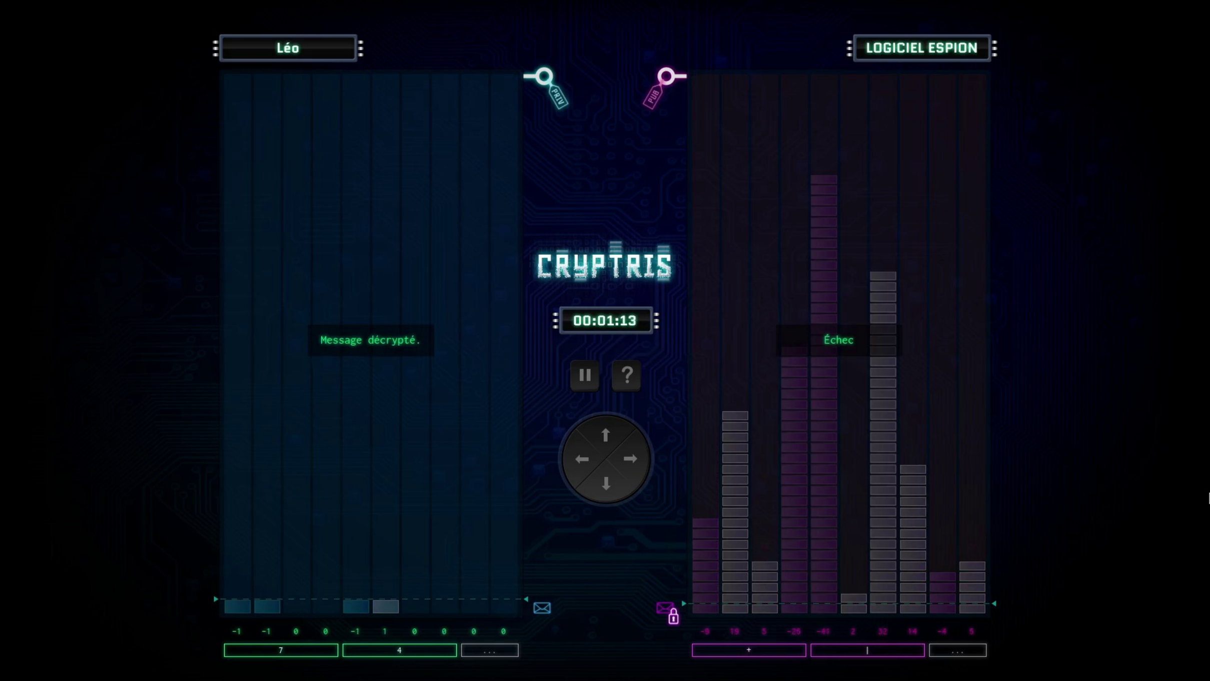


Lattice: Why is it hard

Want to try yourself? Play `https://inriamecsci.github.io/cryptris/`!


CRYPTRIS
CRÉATION DES CLÉS
FACILE - 8 BLOCS
NOVICE - 10 BLOCS
APPRENTI - 12 BLOCS
CHERCHEUR - 14 BLOCS
EXPERT - 16 BLOCS
JOUEUR :
PRIV
CLÉ PRIVÉE
ADVERSAIRE :
PUB
CLÉ PUBLIQUE


Léo
LOGICIEL ESPION
PRIV
PUB
CRYPTRIS
00:00:00
?
10 5 -18 8 19 6 -51 -27 8 38
Y
0
...
10 5 -18 8 19 6 -51 -27 8 38
Y
0
...



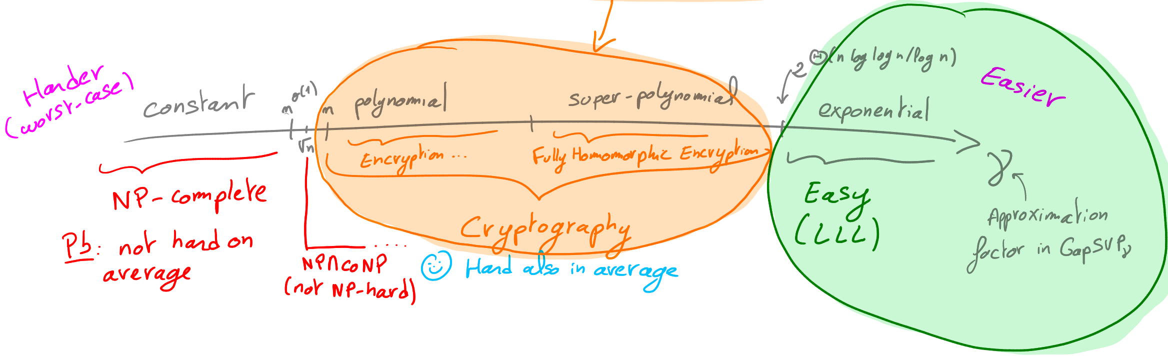
Message décrypté.

Échec

This course

Two goals:

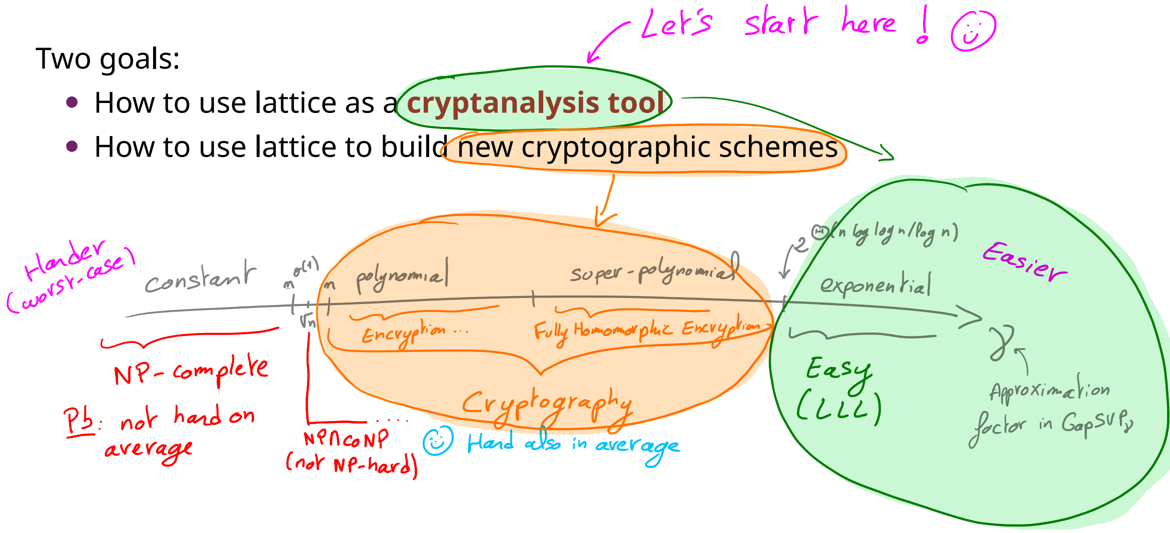
- How to use lattice as a **cryptanalysis tool**
- How to use lattice to build new cryptographic schemes



This course

Two goals:

- How to use lattice as a **cryptanalysis tool**
- How to use lattice to build new cryptographic schemes

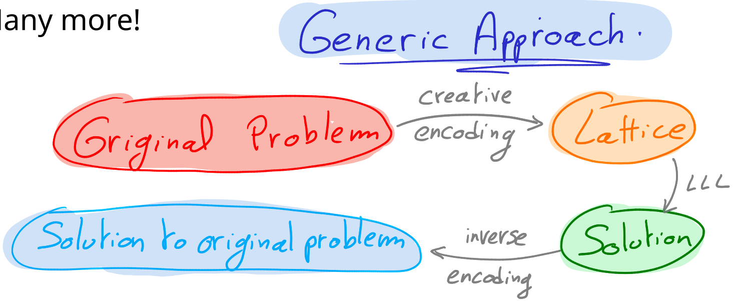


Cryptanalysis based on lattice

Lattice-based cryptanalysis: targets

Many possible targets:

- Knapsack-based crypto-systems
- RSA (e.g. for some parameters or if high bits are known, see for instance *Survey: Lattice Reduction Attacks on RSA*, Wong)
- Elliptic curves (if nonces has leading zeros)
- Many more!

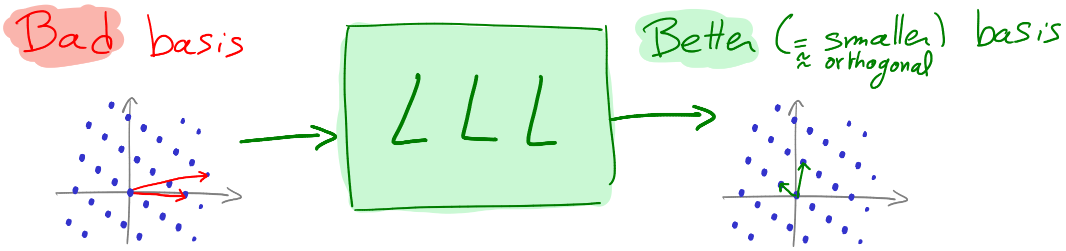


LLL

Super famous : 6 256 citations, implemented in Sage, Maple............

LLL

Super famous : 6 256 citations, implemented in Sage, Maple ....



LLL

$$\longrightarrow \boxed{LLL} \longrightarrow$$

Generic idea :

analogue of Euclid's algorithm to compute GCD

- integers $\longrightarrow$ vectors of integers
- similar operations, $\approx$ as efficient

LLL

Reminder Euclid's algo (gcd, high-school level)

$$\gcd(\underline{100}, \underline{42}) = \gcd(42, 100) = \gcd\Big(\underline{42}, \underbrace{100 - \left\lfloor \tfrac{100}{42} \right\rfloor \times 42}_{=100-2\times 42=16}\Big)$$

shorten

"⌊2.38⌋" $\,^2$

SWAP: we want ordered list: $42 < 100$.

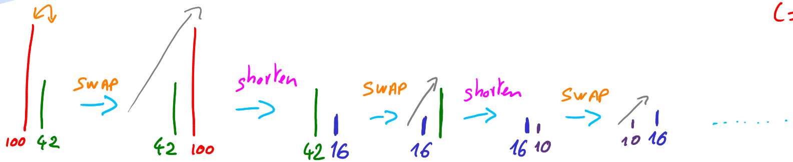
simplify

$$= \gcd(\underline{42}, \underline{16}) = \gcd(\underline{16}, \underline{42}) = \gcd\Big(16, \underbrace{42 - \left\lfloor \tfrac{42}{16} \right\rfloor \times 16}_{10}\Big) = \dots$$

$=2$

repeat until one is "small enough" (=0)

In picture:



⇒ only 2 operations: SWAP and shorten until small enough

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g. $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

$\cancel{b - \left\lfloor \frac{b}{a} \right\rceil a} \rightsquigarrow$ Gram Schmidt

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}_j^*$$

$$\mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

"Sized-reduce" : $\forall i,j$: $|\mu_{ij}| \leq \frac{1}{2}$

$\cancel{b - \left\lfloor \frac{b}{a} \right\rceil a} \rightsquigarrow$ Gram Schmidt

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}_j^*$$

$$\mu_{ij} = \frac{\langle \vec{b}_i , \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g. $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

if ~~$a<b$~~ ⇝ Lovasz condition:

$$\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2) \langle b_{k-1}^*, b_{k-1}^* \rangle$$

~~$b - \lfloor \frac{b}{a} \rceil a$~~ ⇝ Gram Schmidt

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}_j^*$$

$$\mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

"Sized-reduce": $\forall i,j$: $|\mu_{ij}| \leq \frac{1}{2}$

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g. $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

if a ~~<~~ b ⇝ Lovasz condition:

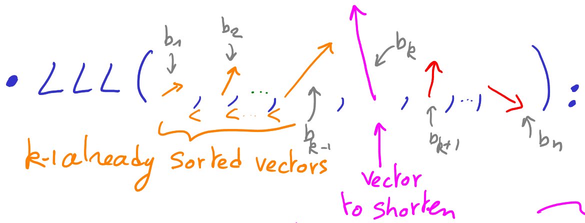
$$\langle b^*_k, b^*_k \rangle > (\delta - \mu^2_{k,k-1}) \langle b^*_{k-1}, b^*_{k-1} \rangle$$

$\cancel{b - \lfloor \frac{b}{a} \rceil a}$ ⇝ Gram Schmidt

$$\vec{b}^*_i = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}^*_j$$

$$\mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}^*_j \rangle}{\| \vec{b}^*_j \|}$$

"Sized-reduce": $\forall i,j$: $|\mu_{ij}| \leq \frac{1}{2}$



Step 1: shorten 1st non sorted vector

```
For j = k-1 to 1:
    | $b_k \leftarrow b_k - \lfloor \mu_{kj} \rceil b_j$
```

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

if a ~~$<$~~ b $\rightsquigarrow$ Lovasz condition:

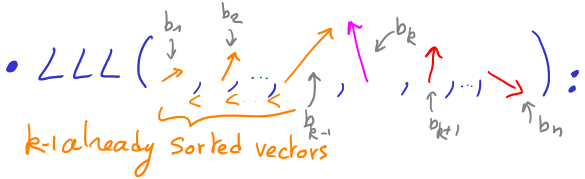
$$\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2) \langle b_{k-1}^*, b_{k-1}^* \rangle$$

~~$b - \lfloor \frac{b}{a} \rfloor a$~~ $\rightsquigarrow$ Gram Schmidt

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}_j^*$$

$$\mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

"Sized-reduce": $\forall i,j$: $|\mu_{ij}| \leq \frac{1}{2}$



Step 1: shorten 1st non sorted vector

```
③ ② ① For $j = k-1$ to $1$:
    $b_k \leftarrow b_k - \lfloor \mu_{kj} \rceil b_j$
```

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g. $3/4$: time/quality trade-off)

SWAP and shorten until small enough

if ~~a < b~~ ⇝ Lovasz condition:

$$\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2) \langle b_{k-1}^*, b_{k-1}^* \rangle$$

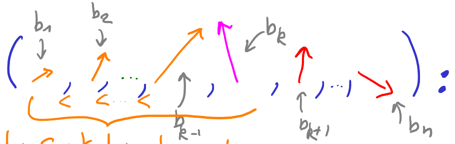
~~$b - \lfloor \frac{b}{a} \rceil a$~~ ⇝ Gram Schmidt

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}_j^*$$

$$\mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

"Sized-reduce" : $\forall i,j : |\mu_{ij}| \le \frac{1}{2}$

• LLL(...):



k already sorted vectors !

Step 1: shorten 1st non sorted vector

```
For j = k-1 to 1:
    $b_k \leftarrow b_k - \lfloor \mu_{kj} \rceil b_j$
```

Step 2: If well sorted (Lovasz condition), go to next vector k+1, else swap

LLL

LLL algo (param $\frac{1}{4} < \delta < 1$, e.g $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

if a ~~<~~ b ⇝ Lovasz condition:

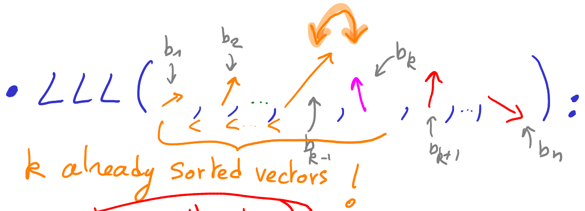
$$\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2) \langle b_{k-1}^*, b_{k-1}^* \rangle$$

~~$b - \left\lfloor \frac{b}{a} \right\rceil a$~~ ⇝ Gram Schmidt

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}_j^*$$

$$\mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

"Sized-reduce": $\forall i,j$: $|\mu_{ij}| \leq \frac{1}{2}$



Step 1: shorten 1st non sorted vector

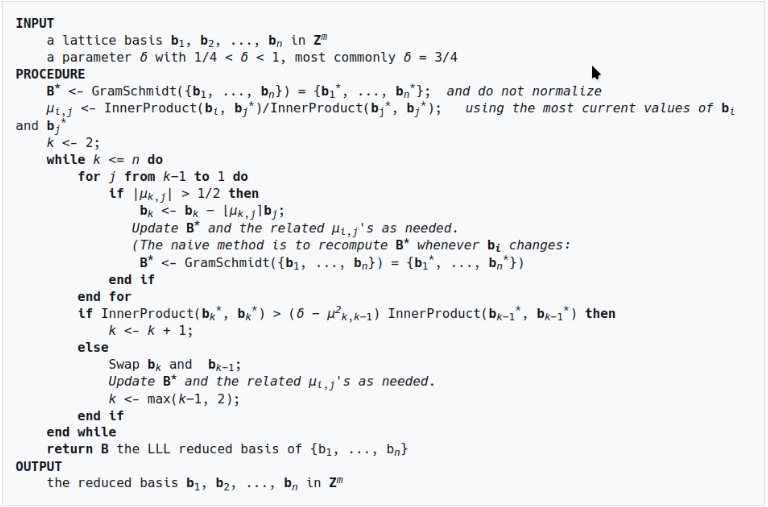
For $j = k-1$ to 1:
  $b_k \leftarrow b_k - \lfloor \mu_{kj} \rceil b_j$

Step 2: If well sorted (Lovasz condition), go to next vector $k+1$, else swap + restart

STOP when sorted + small enough (sized-reduce)

LLL

Summary



```
INPUT
    a lattice basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ in $\mathbf{Z}^m$
    a parameter $\delta$ with $1/4 < \delta < 1$, most commonly $\delta = 3/4$
PROCEDURE
    $\mathbf{B}^*$ <- GramSchmidt({$\mathbf{b}_1, \ldots, \mathbf{b}_n$}) = {$\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$};  and do not normalize
    $\mu_{i,j}$ <- InnerProduct($\mathbf{b}_i$, $\mathbf{b}_j^*$)/InnerProduct($\mathbf{b}_j^*$, $\mathbf{b}_j^*$);   using the most current values of $\mathbf{b}_i$
and $\mathbf{b}_j^*$
    $k$ <- 2;
    while $k$ <= $n$ do
        for $j$ from $k$−1 to 1 do
            if $|\mu_{k,j}| > 1/2$ then
                $\mathbf{b}_k$ <- $\mathbf{b}_k - \lfloor\mu_{k,j}\rceil\mathbf{b}_j$;
               Update $\mathbf{B}^*$ and the related $\mu_{i,j}$'s as needed.
               (The naive method is to recompute $\mathbf{B}^*$ whenever $\mathbf{b}_i$ changes:
                $\mathbf{B}^*$ <- GramSchmidt({$\mathbf{b}_1, \ldots, \mathbf{b}_n$}) = {$\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$})
            end if
        end for
        if InnerProduct($\mathbf{b}_k^*$, $\mathbf{b}_k^*$) > ($\delta - \mu^2_{k,k-1}$) InnerProduct($\mathbf{b}_{k-1}^*$, $\mathbf{b}_{k-1}^*$) then
            $k$ <- $k$ + 1;
        else
            Swap $\mathbf{b}_k$ and  $\mathbf{b}_{k-1}$;
            Update $\mathbf{B}^*$ and the related $\mu_{i,j}$'s as needed.
            $k$ <- max($k$−1, 2);
        end if
    end while
    return B the LLL reduced basis of {$b_1, \ldots, b_n$}
OUTPUT
    the reduced basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ in $\mathbf{Z}^m$
```

LLL properties

Theorem (LLL)

After running δ-LLL on a lattice $\mathcal{L}$ with basis $\mathbf{b}_1, \ldots, \mathbf{b}_n$:

1. The first vector in the basis cannot be much larger than the shortest non-zero vector: $\|\mathbf{b}_1\| \leq (2/(\sqrt{4\delta - 1}))^{n-1} \cdot \lambda_1(\mathcal{L})$
2. The first vector in the basis is also bounded by the determinant of the lattice: $\|\mathbf{b}_1\| \leq (2/(\sqrt{4\delta - 1}))^{(n-1)/2} \cdot (\det(\mathcal{L}))^{1/n}$
3. The product of the norms of the vectors in the basis cannot be much larger than the determinant of the lattice: let $\delta = 3/4$, then $\prod_{i=1}^{n} \|\mathbf{b}_i\| \leq 2^{n(n-1)/4} \cdot \det(\mathcal{L})$

In practice, it works often **even better**!

Application: breaking the Merkle-Hellman cryptosystem

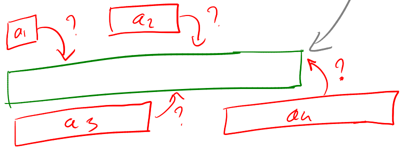
Contexte

Merkle-Hellman:

- cryptosystem published in 1978
- (simpler) competitor of RSA
- broken by Shamir in 1982:
  $\Rightarrow$ starting point of many LLL-based attacks

Merkle-Hellman

Based on knapsack problem + trapdoor

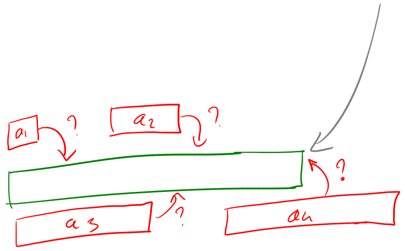


Goal: find subset of a_i's filling the bag

↳ NP-Hard (worst-case)

Merkle-Hellman

Based on knapsack problem + trapdoor



Goal: find subset of a_i's filling the bag
↳ NP-Hard (worst case)

Key generation:

- super-increasing sequence $\{a_1, \dots, a_n\}$ (i.e. $\forall i, a_i > \quad {}_{j<i} a_j$)
- Let $N > \quad {}_i a_i$ and $A < N$, $\gcd(A, N) = 1$
- Public key: pk $:= \{b_i := Aa_i \pmod N\}$, private key: sk $:= (N, A, \{a_i\}_i)$

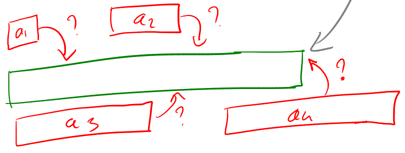
Encryption: ← message = subset of elements to take

$\text{Enc}_{\text{pk}}(m := (m_1, \dots, m_n)) = \quad {}_i m_i b_i$

Decryption: (not relevant, but based on $A^{-1}(\quad {}_i m_i b_i) \bmod N = \quad {}_i x_i a_i)$ + use fact that sequence is super-increasing

Merkle-Hellman

Based on knapsack problem + trapdoor



Goal: find subset of a_i's filling the bag
↳ NP-Hard (worst-case)

Key generation:

- super-increasing sequence $\{a_1, \dots, a_n\}$ (i.e. $\forall i, a_i > \ {}_{j<i} a_j$)
- Let $N > \ {}_i a_i$ and $A < N$, $\gcd(A, N) = 1$
- Public key: pk $:= \{b_i := A a_i \pmod N\}$, private key: sk $:= (N, A, \{a_i\}_i)$

Hide sequence (trivial otherwise)

Encryption: message = subset of elements to take

$\mathsf{Enc}_{\mathsf{pk}}(m := (m_1, \dots, m_n)) = \ {}_i m_i b_i$

Decryption: (not relevant, but based on $A^{-1}(\ {}_i m_i b_i) \bmod N = \ {}_i x_i a_i)$ + use fact that sequence is super-increasing

Merkle-Hellman

?

If pk $:= [10, 3, 16, 15]$, what is $\mathsf{Enc}_{\mathsf{pk}}(1101)$?

- Ⓐ 16
- Ⓑ 28
- Ⓒ 44

Merkle-Hellman

? If pk $:= [10, 3, 16, 15]$, what is $\mathsf{Enc}_{\mathsf{pk}}(1101)$?

- (A) 16
- (B) 28 ✓ $= 1 \times 10 + 1 \times 3 + 0 \times 16 + 1 \times 15$
- (C) 44

Merkle-Hellman attack

To decrypt a ciphertext $c = \sum_i m_i b_i$, we want to find a lattice $\mathcal{L}$ such that:

- The solution can be encoded into a vector $v \in \mathcal{L}$
- v has small (non-null) norm
- From v we can recover m

Merkle-Hellman attack

To decrypt a ciphertext $c = \sum_i m_i b_i$, we want to find a lattice $\mathcal{L}$ such that:

- The solution can be encoded into a vector $v \in \mathcal{L}$
- v has small (non-null) norm
- From v we can recover m

First attempt: show that if we choose the “basis” B that contains for all i the vector (b_i) and $(-c)$, then there exists a non-null linear combination of vectors in B that produces the vector 0.

Merkle-Hellman attack

To decrypt a ciphertext $c = \sum_i m_i b_i$, we want to find a lattice $\mathcal{L}$ such that:

- The solution can be encoded into a vector $v \in \mathcal{L}$
- v has small (non-null) norm
- From v we can recover m

?

First attempt: show that if we choose the "basis" B that contains for all i the vector (b_i) and $(-c)$, then there exists a non-null linear combination of vectors in B that produces the vector 0.
$\Rightarrow v := (\sum_i m_i (b_i)) + 1 \times (-c) = (c - c) = (0)$

Merkle-Hellman attack

To decrypt a ciphertext $c = \sum_i m_i b_i$, we want to find a lattice $\mathcal{L}$ such that:

- The solution can be encoded into a vector $v \in \mathcal{L}$
- v has small (non-null) norm
- From v we can recover m

?

First attempt: show that if we choose the "basis" B that contains for all i the vector (b_i) and $(-c)$, then there exists a non-null linear combination of vectors in B that produces the vector 0.
$\Rightarrow v := (\sum_i m_i (b_i)) + 1 \times (-c) = (c - c) = (0)$

Problems:

- not a basis (vectors are not independent)
- since v is null, this gives no information about m_i's

How to fix that?

Merkle-Hellman attack

Let $B := \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ b_1 & b_2 & \cdots & b_n & -c \end{pmatrix}$ (all unspecified entries are 0).

? Show that $\mathcal{L}(B)$ admits a non-null vector v of norm $\leq \sqrt{n}$, and show how to recover m from v.

Merkle-Hellman attack

Let $B := \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ b_1 & b_2 & \cdots & b_n & -c \end{pmatrix}$ (all unspecified entries are 0).

Show that $\mathcal{L}(B)$ admits a non-null vector v of norm $\leq \sqrt{n}$, and show how to recover m from v.

Solution: $v := B \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ 1 \end{pmatrix} = \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ {}_i\, b_i m_i - c = 0 \end{pmatrix}$, and has norm $\|v\| := \overline{|\{i \mid m_i = 1\}|} \leq \sqrt{n}$.

Merkle-Hellman attack

Let $B := \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ b_1 & b_2 & \cdots & b_n & -c \end{pmatrix}$ (all unspecified entries are 0).

? Show that $\mathcal{L}(B)$ admits a non-null vector v of norm $\leq \sqrt{n}$, and show how to recover m from v.

Solution: $v := B \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ 1 \end{pmatrix} = \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ {}_i\, b_i m_i - c = 0 \end{pmatrix}$, and has norm

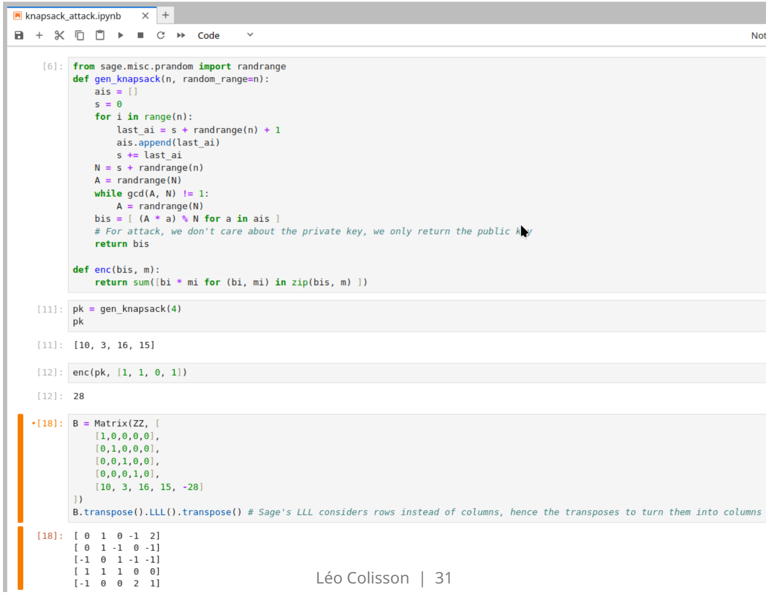
$\|v\| := \quad \overline{|\{i \mid m_i = 1\}|} \leq \sqrt{n}.$

Merkle-Hellman attack

Attack against Merkle-Hellman:

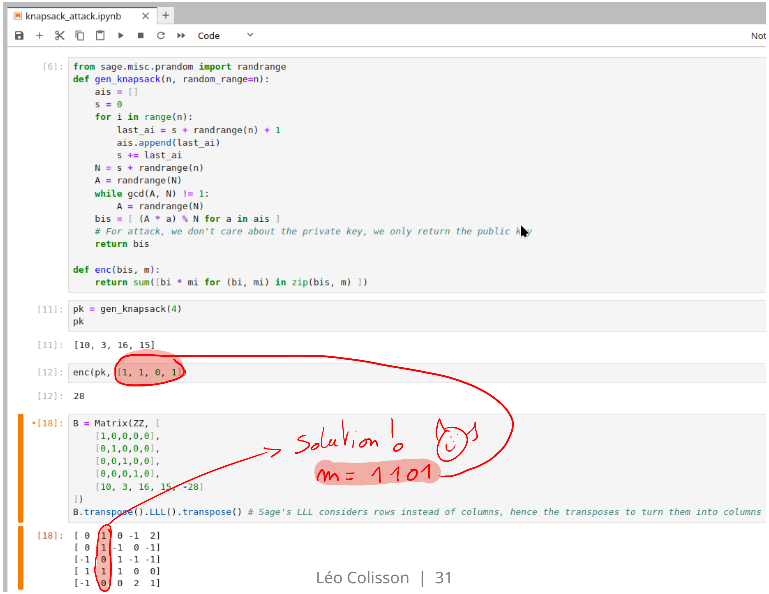
1. **run LLL on B** (from previous slide)
2. We get a list of small vectors v: if one has only binary entries and ends with a 0, extract m and check if solution! (demo next slide)

Merkle-Hellman attack: demo in sagemath



```
[6]:  from sage.misc.prandom import randrange
      def gen_knapsack(n, random_range=n):
          ais = []
          s = 0
          for i in range(n):
              last_ai = s + randrange(n) + 1
              ais.append(last_ai)
              s += last_ai
          N = s + randrange(n)
          A = randrange(N)
          while gcd(A, N) != 1:
              A = randrange(N)
          bis = [ (A * a) % N for a in ais ]
          # For attack, we don't care about the private key, we only return the public key
          return bis

      def enc(bis, m):
          return sum([bi * mi for (bi, mi) in zip(bis, m) ])

[11]: pk = gen_knapsack(4)
      pk

[11]: [10, 3, 16, 15]

[12]: enc(pk, [1, 1, 0, 1])

[12]: 28

[18]: B = Matrix(ZZ, [
          [1,0,0,0,0],
          [0,1,0,0,0],
          [0,0,1,0,0],
          [0,0,0,1,0],
          [10, 3, 16, 15, -28]
      ])
      B.transpose().LLL().transpose() # Sage's LLL considers rows instead of columns, hence the transposes to turn them into columns

[18]: [ 0  1  0 -1  2]
      [ 0  1 -1  0 -1]
      [-1  0  1 -1 -1]
      [ 1  1  1  0  0]
      [-1  0  0  2  1]
```

Merkle-Hellman attack: demo in sagemath



```
[6]:  from sage.misc.prandom import randrange
      def gen_knapsack(n, random_range=n):
          ais = []
          s = 0
          for i in range(n):
              last_ai = s + randrange(n) + 1
              ais.append(last_ai)
              s += last_ai
          N = s + randrange(n)
          A = randrange(N)
          while gcd(A, N) != 1:
              A = randrange(N)
          bis = [ (A * a) % N for a in ais ]
          # For attack, we don't care about the private key, we only return the public key
          return bis

      def enc(bis, m):
          return sum([bi * mi for (bi, mi) in zip(bis, m) ])

[11]: pk = gen_knapsack(4)
      pk

[11]: [10, 3, 16, 15]

[12]: enc(pk, [1, 1, 0, 1])

[12]: 28

•[18]: B = Matrix(ZZ, [
          [1,0,0,0,0],
          [0,1,0,0,0],
          [0,0,1,0,0],
          [0,0,0,1,0],
          [10, 3, 16, 15, -28]
      ])
      B.transpose().LLL().transpose() # Sage's LLL considers rows instead of columns, hence the transposes to turn them into columns

[18]: [ 0  1  0 -1  2]
      [ 0  1 -1  0 -1]
      [-1  0  1 -1 -1]
      [ 1  1  1  0  0]
      [-1  0  0  2  1]
```

Solution! o

m = 1101

Cryptanalysis via LLL: conclusion

Take home message:

LLL reductions = very powerful tool to attack cryptosystems (and more!)