Advanced Crypto 2024
Lattice-based cryptography

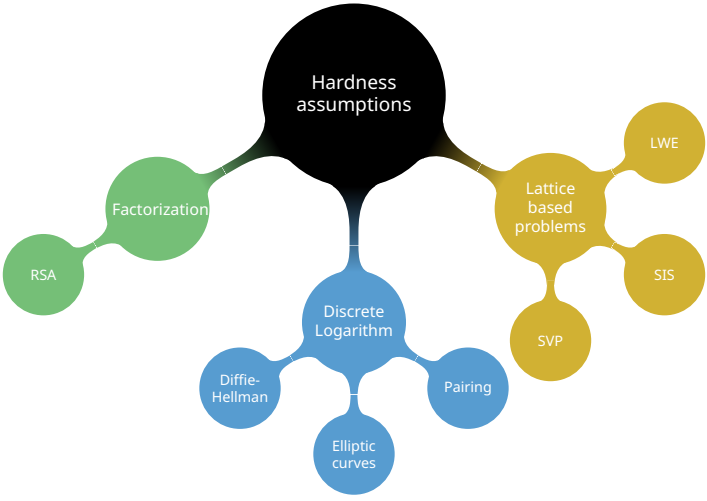
Léo Colisson Palais

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

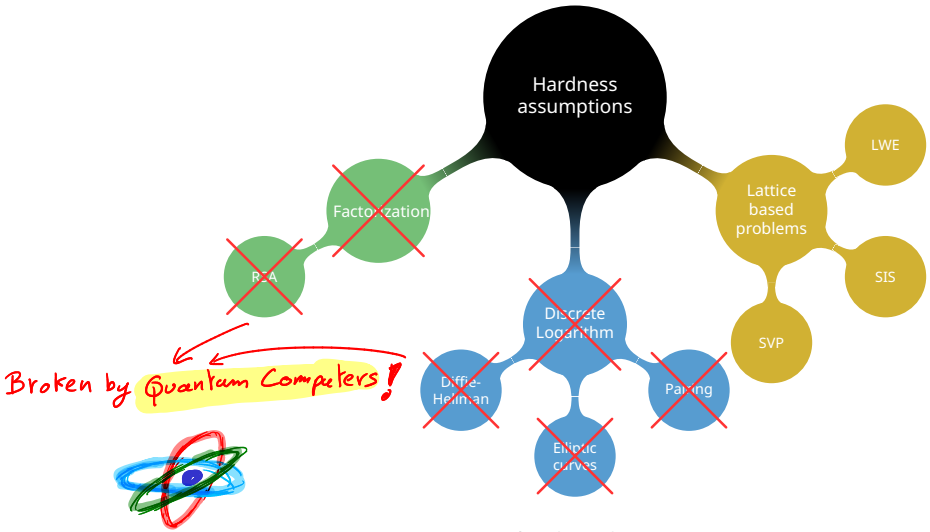
https://leo.colisson.me/teaching.html

Motivations

Motivations



Motivations



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?

  ⇒ **Cannot wait!** “Harvest now, decrypt later”

Motivations

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

  ⇒ **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?

  ⇒ **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?

  ⇒ **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)

  ⇒ Creation of a **standardization competition** by NIST!

Motivations

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

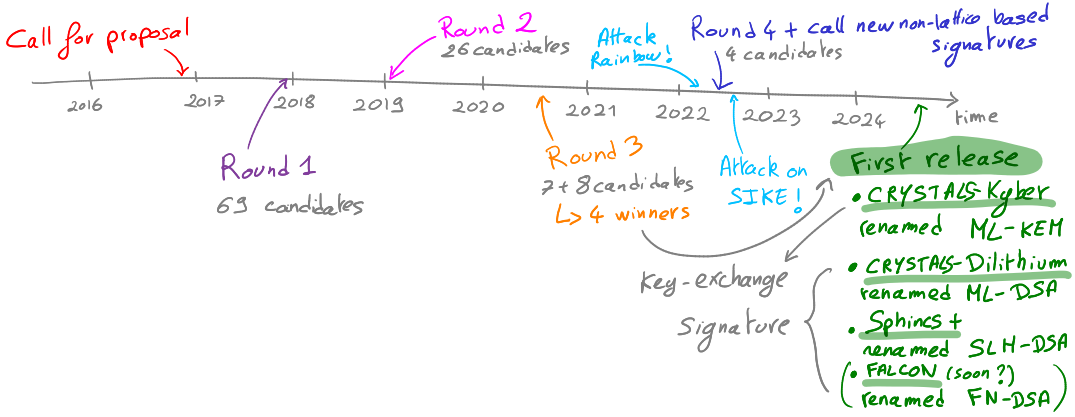
  ⇒ **Cannot wait!** “Harvest now, decrypt later”

- **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)

  ⇒ Creation of a **standardization competition** by NIST!
  ⇒ 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)

- CRYSTALS-Kyber
  renamed ML-KEM

Signature

- CRYSTALS-Dilithium
  renamed ML-DSA
- Sphincs+
  renamed SLH-DSA

Hardness Assumption (in bytes, Level 3)	Lattice-based	Lattice-based	Hash-based
$\lvert pk \rvert$:	1184	1952	48
$\lvert sk \rvert$:	2400	4032	96
$\lvert \text{cipher} \rvert$:	1088		
$\lvert \text{shared key} \rvert$:	32		
$\lvert \text{signature} \rvert$:		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!)

Hardness Assumption (in bytes, Level 3)	Lattice-based ✓	Elliptic-curves ✗
$\lvert p_k \rvert$:	1184 ✗	32 ✓
$\lvert s_k \rvert$:	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

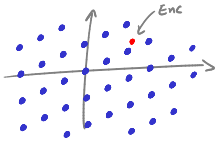
Compare with ECDH with Curve 25519
(Not post-quantum!)

Hardness Assumption (in bytes, Level 3)	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 ✓

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

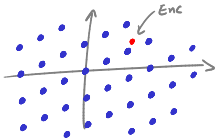
Famous post-quantum candidates

1. Lattice-based Crypto

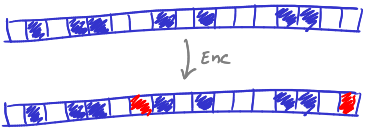


Famous post-quantum candidates

1 Lattice-based Crypto

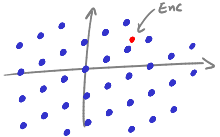


2 Code-based Crypto

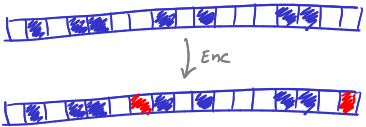


Famous post-quantum candidates

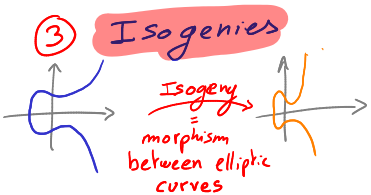
1 Lattice-based Crypto



2 Code-based Crypto

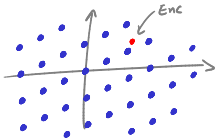


3 Isogenies

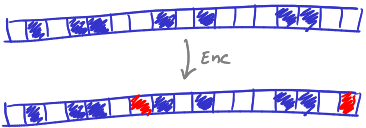


Famous post-quantum candidates

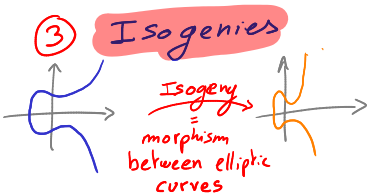
1 Lattice-based Crypto



2 Code-based Crypto



3 Isogenies

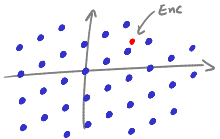


4 Multivariate Crypto

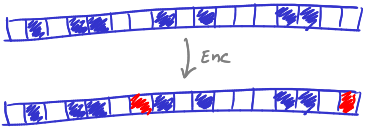
$$P_k := \begin{cases} \bullet\ 1 + x_1 + 2x_0x_3 \\ \bullet\ 4 + x_4 + 3x_1^2x_8 + x_9 \\ \bullet\ x_6 + x_2^3x_5 + x_7x_5 \end{cases} \xrightarrow{\text{Enc}} P_k(m)$$

Famous post-quantum candidates

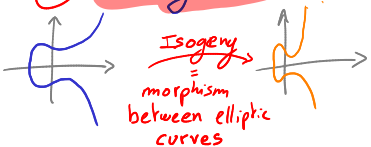
1. Lattice-based Crypto



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

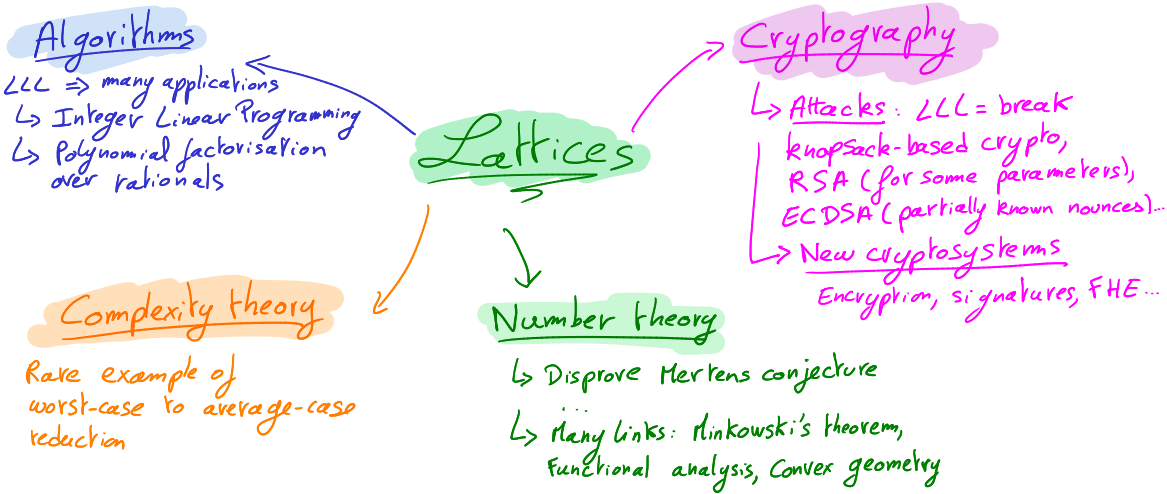
5 + Symmetric crypto (incl. signatures)

Introduction to lattices

References

- Great survey: *A Decade of Lattice Cryptography*, Chris Peikert
- Shorter survey with reduction proofs and great intuition regarding Ring-LWE: *The Learning with Errors Problem*, Oded Regev
  `https://cims.nyu.edu/~regev/papers/lwesurvey.pdf`
- 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



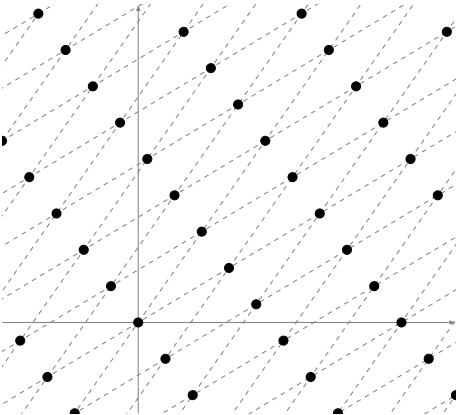
Lattices

Definition (Lattice)

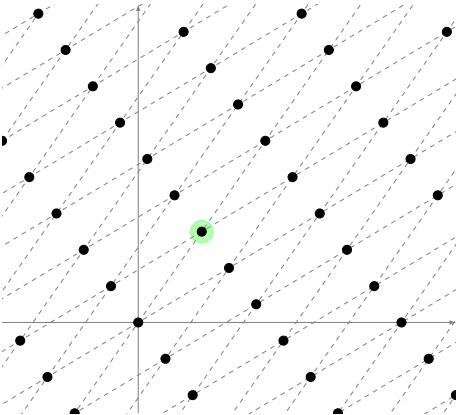
An n-dimensional *lattice* $\mathcal{L}$ is any subset of $\mathbb{R}^n$ that is both:

- 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

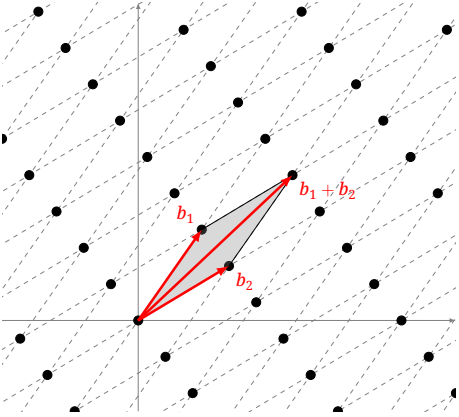
Lattice



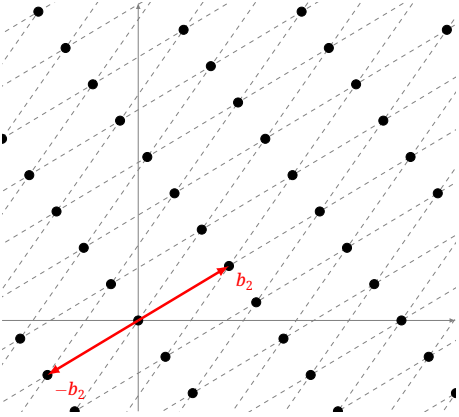
Lattice



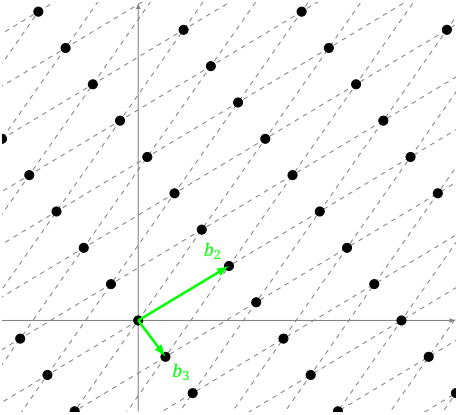
Lattice



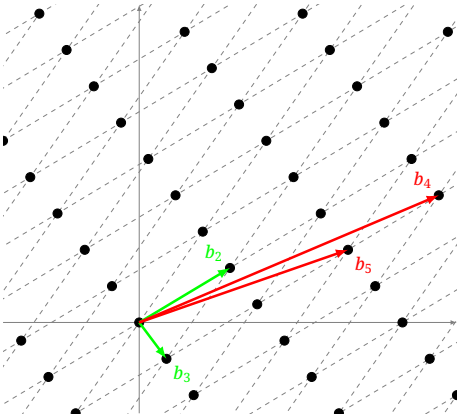
Lattice



Lattice

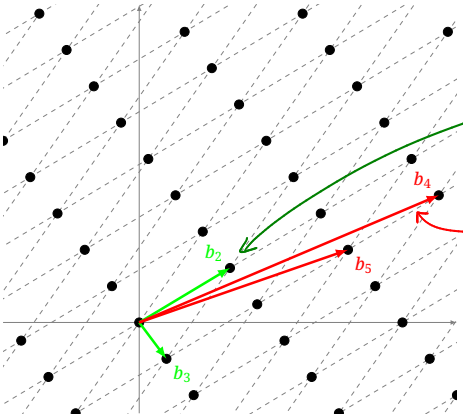


Lattice



Basis = not unique !

Lattice


b_4
b_2
b_5
b_3
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

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

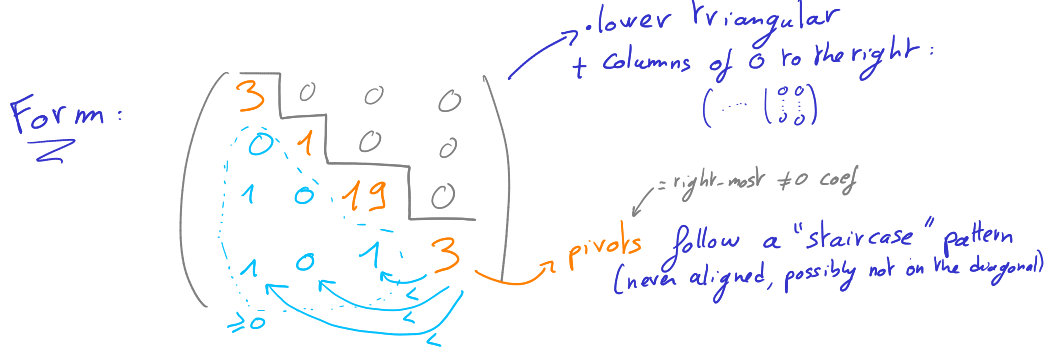
k is the **rank** of the lattice. If $k = n$, the lattice has **full-rank** (often the case).

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?

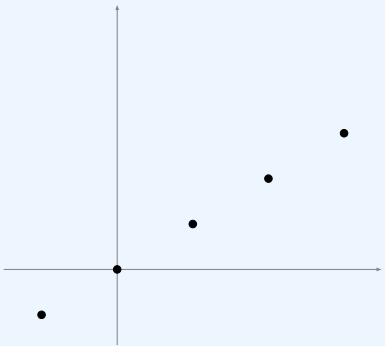
⇒ **Hermite normal form** can always be efficiently be computed and is unique: Good reference basis.



Lattice: basis

What is the dimension and rank of this lattice?

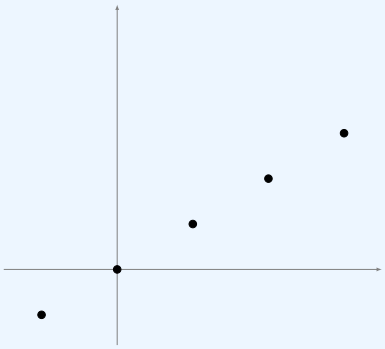
?



Lattice: basis

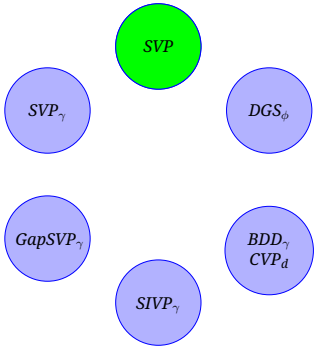
What is the dimension and rank of this lattice?

?

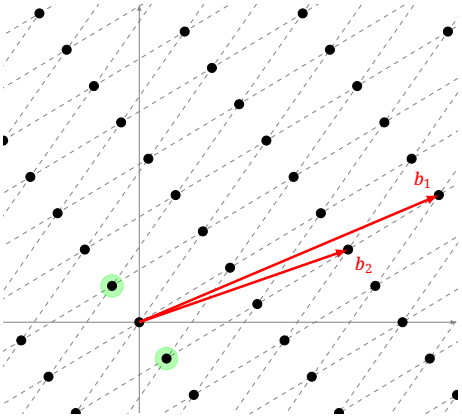


⇒ Dimension is 2, rank is 1

Lattice : what is hard to do?



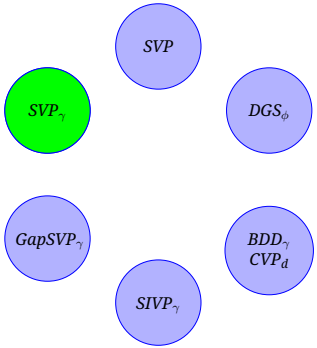
Shortest Vector Problem



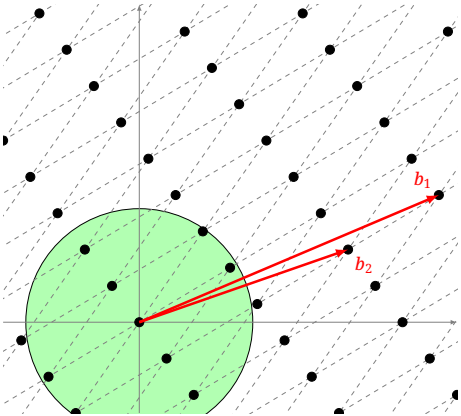
Compute basis B with the smallest length for each vector
λ_i = length of i-th largest vector in B

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?



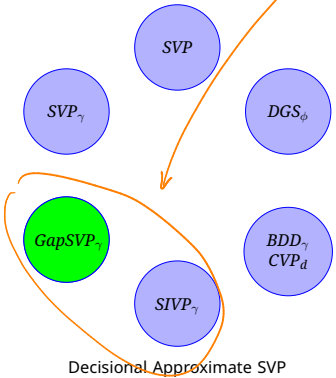
Approximate Shortest Vector Problem

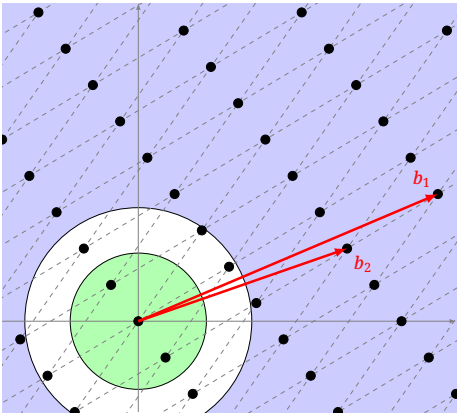


Goal: Given a basis B of a lattice $\mathcal{L}$, find a vector $x \in \mathcal{L} \setminus \{0\}$ s.t. $\|x\| \leq \gamma(n)\lambda_1(\mathcal{L})$.

Lattice : what is hard to do?

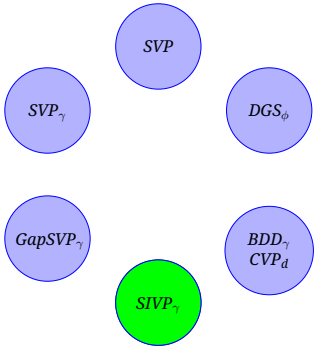
Hard to reduce to SVP/SVP$_\gamma$: most reductions reduce to GapSVP or SIVP



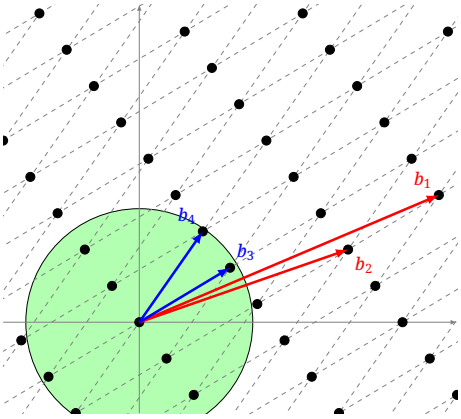


Goal: Given a basis B of a lattice $\mathcal{L}$, with the promise that $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma(n)$, determine which is the case.

Lattice : what is hard to do?

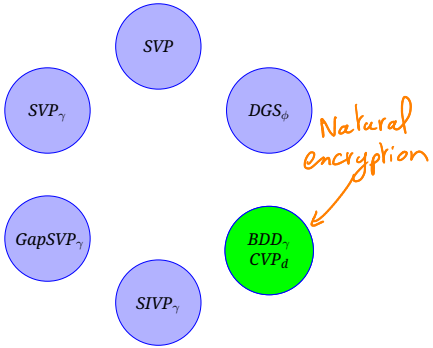


Approximate Shortests
Independent Vectors Problem

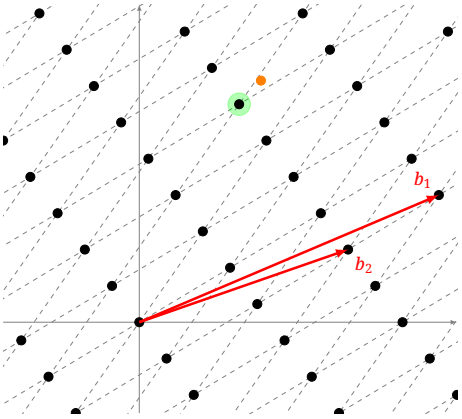


Goal: Given a basis B of a full-rank lattice $\mathcal{L}$, output a set $\{s_i\} \subset \mathcal{L}$ of n linearly independent lattice vectors where $\forall i, \|s_i\| \leq \gamma(n).\lambda_n(\mathcal{L})$.

Lattice : what is hard to do?



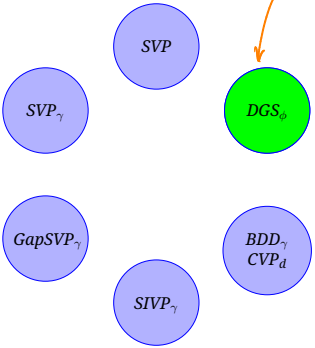
Bounded Distance Decoding Problem
and Closest Vector Problem



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

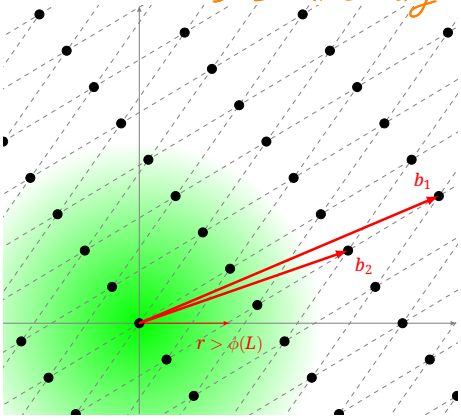
Lattice : what is hard to do?

Given used in proofs (e.g. LWE as hard as $GapSVP_\gamma$)



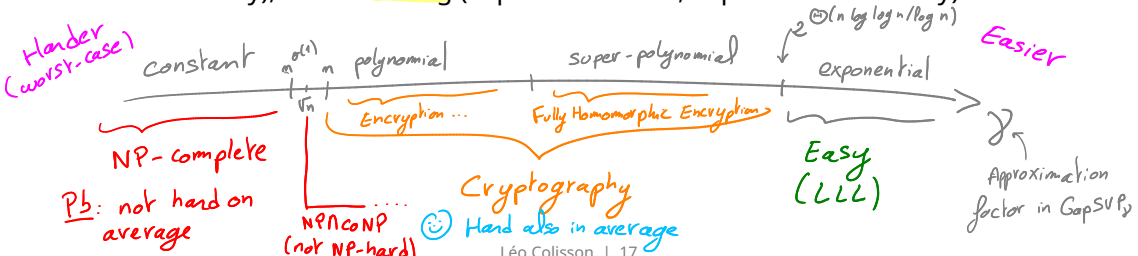
Discrete Gaussian Sampling problem

use DGS internally



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 ◀

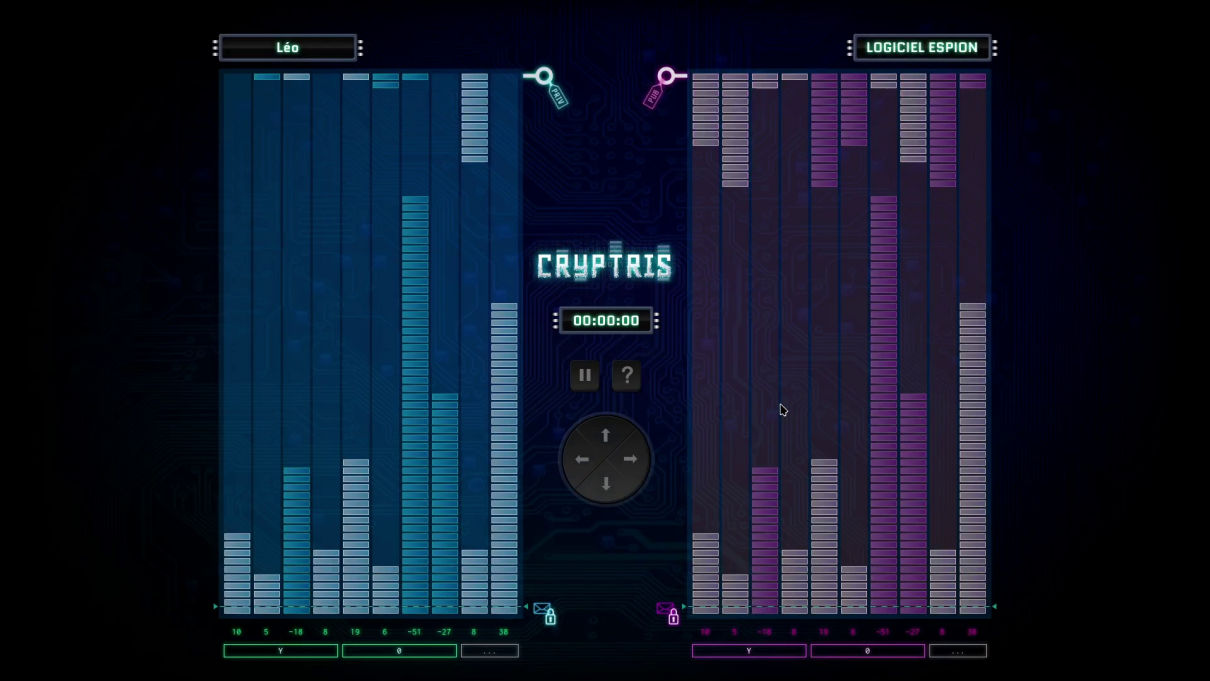
APPRENTI - 12 BLOCS

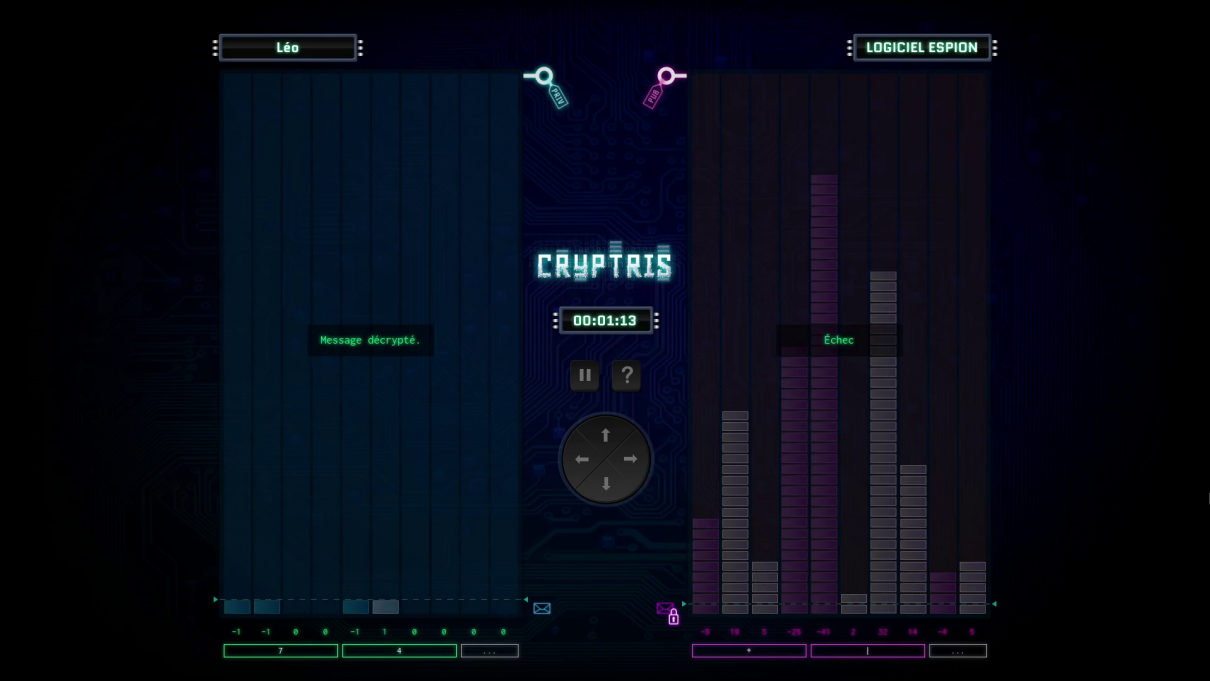
CHERCHEUR - 14 BLOCS

EXPERT - 16 BLOCS

JOUEUR : 

ADVERSAIRE : 

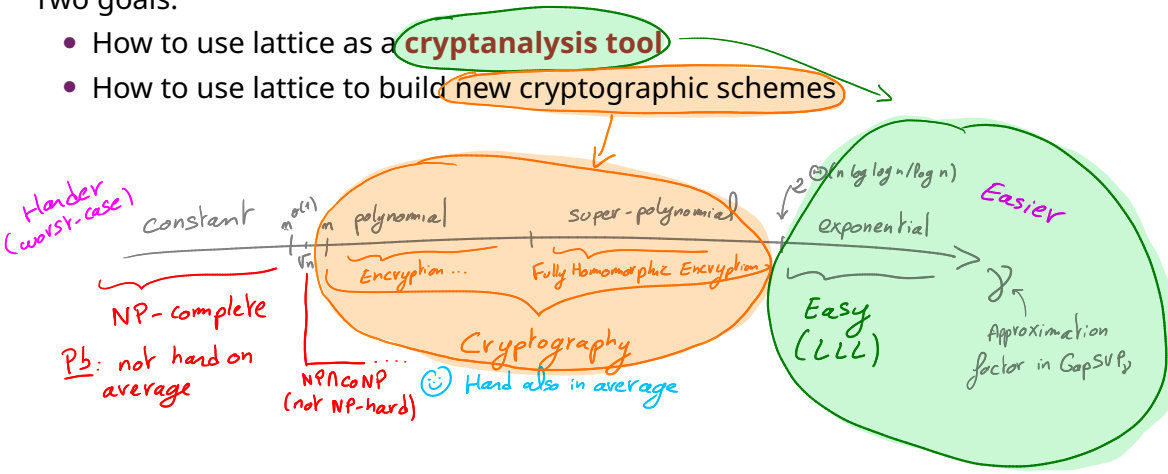

Léo
LOGICIEL ESPION
CRYPTRIS
00:00:00
?
10 5 -18 8 19 6 -51 -27 8 38
Y
0
10 5 -18 8 19 6 -51 -27 8 38
Y
0


Léo
LOGICIEL ESPION
CRYPTRIS
00:01:13
Message décrypté.
Échec

This course

Two goals:

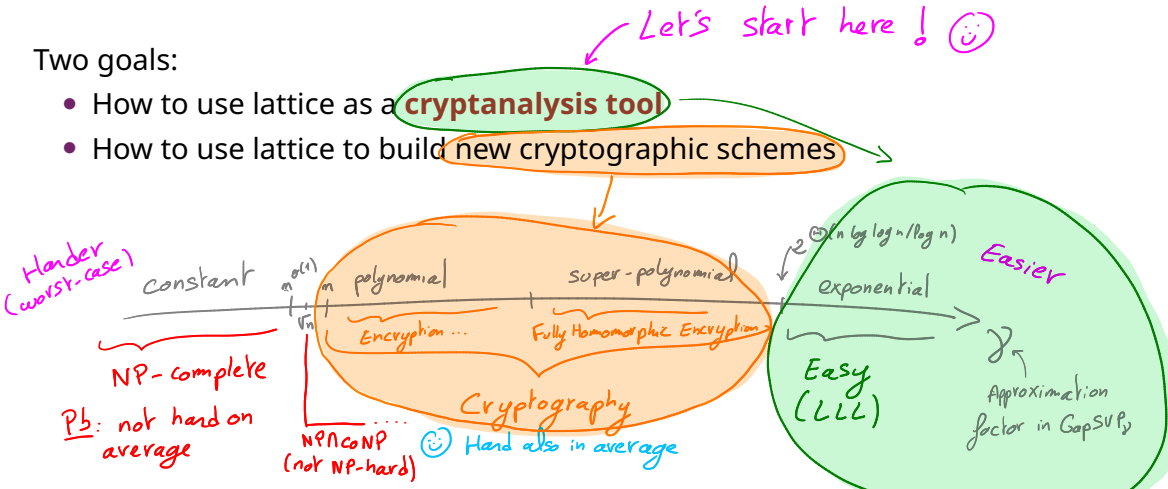
- How to use lattice as a **cryptanalysis tool**
- How to use lattice to build new cryptographic schemes



This course

Two goals:

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

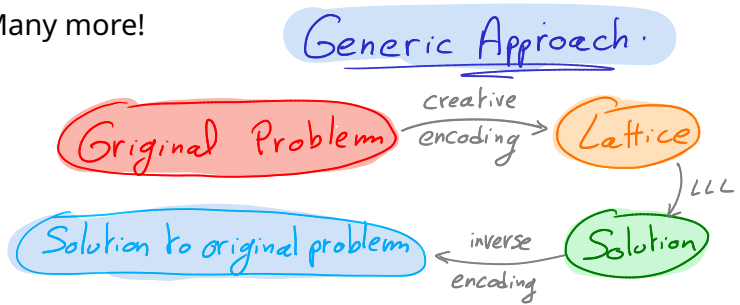


Cryptanalysis based on lattice

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!

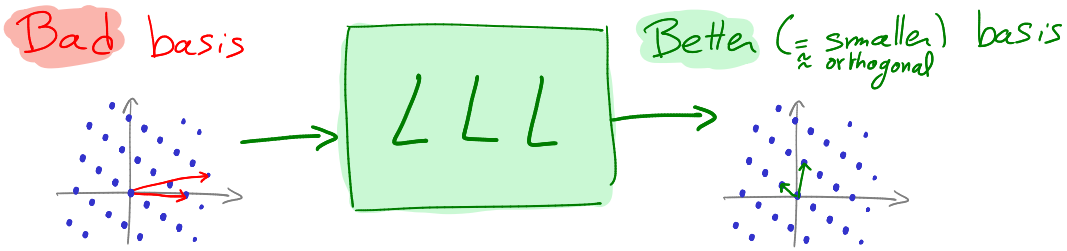


LLL

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

LLL

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



LLL

Generic idea :

analogue of Euclid's algorithm to compute GCD

- integers → vectors of integers
- similar operations, ≈ as efficient

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

$\gcd(100, 42) = \gcd(42, 100) = \gcd(42, 100 - \lfloor \frac{100}{42} \rfloor \times 42)$

shorten

"$\lfloor 2.38 \rfloor$"

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

$= 100 - 2 \times 42 = 16$

simplify

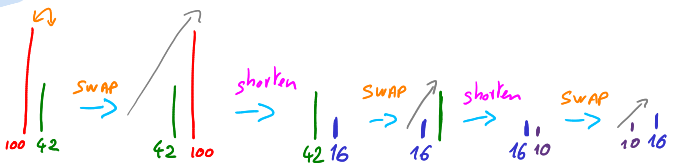
$= \gcd(42, 16) = \gcd(16, 42) = \gcd(16, 42 - \lfloor \frac{42}{16} \rfloor \times 16) = \dots$

$= 2$

10

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

In picture:



$\Rightarrow$ 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

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

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

$b \leftarrow b - \lfloor \frac{b}{a} \rceil a$ ⇝ 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 $a \not\leq b$ ⇝ 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

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

$\vec{b_i}^* = \vec{b_i} - \sum_{j=1}^{i-1} \mu_{ij} \vec{b_j}^*$

$\mu_{ij} = \frac{\langle \vec{b_i}, \vec{b_j}^* \rangle}{\| \vec{b_j}^* \|}$

- LLL(b_1, b_2, ..., b_{k-1}, b_k, b_{k+1}, ..., b_n):

$k-1$ already sorted vectors

vector to shorten

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 $a \leftrightarrow b$ ⇝ Lovasz condition: $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2) \langle b_{k-1}^*, b_{k-1}^* \rangle$

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

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

$\vec{b_i^*} = \vec{b_i} - \sum_{j=1}^{i-1} \mu_{ij} \vec{b_j^*}$

$\mu_{ij} = \frac{\langle \vec{b_i}, \vec{b_j^*} \rangle}{\| \vec{b_j^*} \|}$

- LLL(b_1, b_2, ..., b_{k-1}, b_k, ..., b_{k+1}, ..., b_n):

k-1 already sorted vectors

Step 1: shorten 1st non sorted vector

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

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 $a \neq b$ ⇝ Lovasz condition: $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2) \langle b_{k-1}^*, b_{k-1}^* \rangle$

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

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

$\vec{b_i^*} = \vec{b_i} - \sum_{j=1}^{i-1} \mu_{ij} \vec{b_j^*}$

$\mu_{ij} = \frac{\langle \vec{b_i}, \vec{b_j^*} \rangle}{\| \vec{b_j^*} \|}$

- LLL(b_1, b_2, ..., b_{k-1}, b_k, b_{k+1}, ..., b_n) :

k already sorted vectors !

Step 1: Shorten 1st non sorted vector

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

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 $a \neq b$ ⇝ Lovasz condition: $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^2) \langle b_{k-1}^*, b_{k-1}^* \rangle$
- ~~$b \leftarrow b - \lfloor \frac{b}{a} \rceil a$~~ ⇝ Gram Schmidt
- "Sized-reduce": $\forall i,j$: $|\mu_{ij}| \leq \frac{1}{2}$

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

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

- LLL($b_1, b_2, \dots, b_{k-1}, b_k, b_{k+1}, \dots, b_n$) :

$k-1$ already sorted vectors !

Step 1: Shorten 1st non sorted vector

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

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

STOP when sorted + small enough (sized-reduce)

LLL

INPUT
a lattice basis $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$ in $\mathbf{Z}^m$
a parameter δ 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 $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$
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

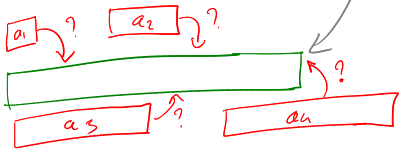
Contexte

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

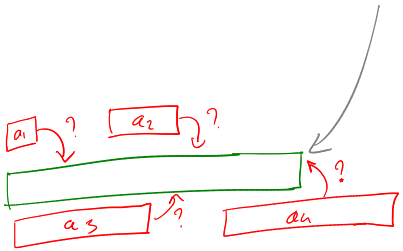


Goal: find subset of a_i's filling the bag

↳ 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 $\{a_1, \ldots, a_n\}$ (i.e. $\forall i, a_i > \sum_{j<i} a_j$)
- Let $N > \sum_i a_i$ and $A < N$, $\gcd(A, N) = 1$
- Public key: $\mathsf{pk} := \{b_i := Aa_i \pmod N\}$, private key: $\mathsf{sk} := (N, A, \{a_i\}_i)$

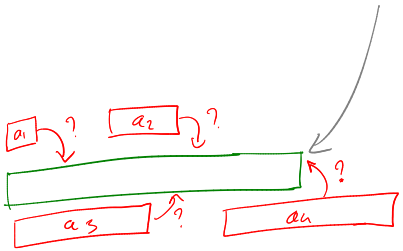
Encryption: message = subset of elements to take

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

Decryption: (not relevant, but based on $A^{-1}(\sum_i m_i b_i) \bmod N = \sum_i x_i a_i$) + 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 $\{a_1, \ldots, a_n\}$ (i.e. $\forall i, a_i > \sum_{j<i} a_j$)
- Let $N > \sum_i a_i$ and $A < N$, $\gcd(A, N) = 1$
- Public key: $\mathsf{pk} := \{b_i := A a_i \pmod N\}$, private key: $\mathsf{sk} := (N, A, \{a_i\}_i)$

Hide sequence (trivial otherwise)

Encryption: message = subset of elements to take

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

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

Merkle-Hellman

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

A. 16
B. 28
C. 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.

$\Rightarrow v \coloneqq (\sum_i m_i (b_i)) + 1 \times (-c) = (c - c) = (0)$

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.
$\Rightarrow v \coloneqq (\sum_i m_i (b_i)) + 1 \times (-c) = (c - c) = (0)$

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 $B \coloneqq \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ b_1 & b_2 & \cdots & b_n & -c \end{pmatrix}$ (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 $B \coloneqq \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ b_1 & b_2 & \cdots & b_n & -c \end{pmatrix}$ (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: $v \coloneqq B \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ 1 \end{pmatrix} = \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ \sum_i b_i m_i - c = 0 \end{pmatrix}$, and has norm $\|v\| \coloneqq \sqrt{|\{i \mid m_i = 1\}|} \leq \sqrt{n}$.

Merkle-Hellman attack

Let $B \coloneqq \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ b_1 & b_2 & \cdots & b_n & -c \end{pmatrix}$ (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: $v \coloneqq B \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ 1 \end{pmatrix} = \begin{pmatrix} m_1 \\ \vdots \\ m_n \\ \sum_i b_i m_i - c = 0 \end{pmatrix}$, and has norm

$\|v\| \coloneqq \sqrt{|\{i \mid m_i = 1\}|} \leq \sqrt{n}$.

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

knapsack_attack.ipynb

Code

[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

knapsack_attack.ipynb

Code

[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
```

Solution !
m = 1101

[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]
```

Cryptanalysis via LLL: conclusion

Take home message:

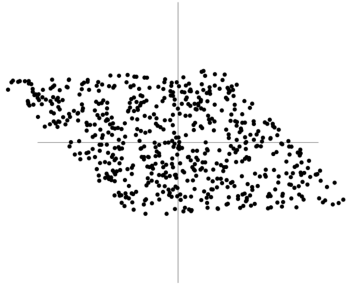
LLL reductions = very powerful tool to attack cryptosystems (and more!)

Lattice-based cryptography

Lattice-based cryptography: historical works

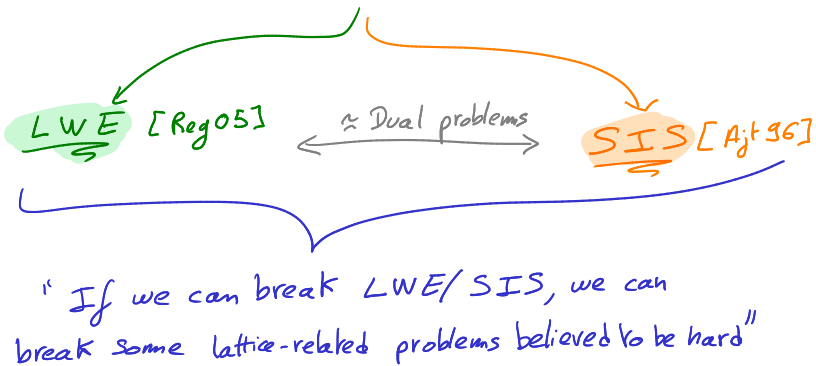
Lattice also used to **build new primitives**… but we don't use them directly anymore:

- First lattice-based public key encryption [Ajtai and Dwork, 1997]:
  - **breakthrough**: first crypto-system with a security proof with a reduction from worst case to average case
  - … but **very inefficient**: public key multiple gigabits!!
- NTRU [Hoffstein, Pipher, and Silverman, 1998]: quite efficient encryption, but no known reduction from worst case to average case
- Encryption + signature of [Goldreich, Goldwasser, and Halevi 1997]: close to our intuition "good basis/wrong basis":
  ↪ "to encrypt we encode the message in a vector and add some noise"
  ↪ "to sign we interpret the message as a point and find a close point in the lattice"
  Pb: No worst-case to average case reduction, signature **fully broken**, encryption severely attacked or inefficient [Ngu99, NR06]… but fundamental idea still used nowadays.



Lattice-based cryptography: modern approach

Modern approach: define problems that reduce to lattice-based problems



Lattice-based cryptography: SIS

Definition

SIS [Ajt 96]

n, q, β, m

- n: dimension $\in \mathbb{N}$
- q: modulus $\in \mathbb{N}$
- β: $\beta \ll q$
- m: $\approx$ number of samples $\in \mathbb{N}$

Given $A \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$

Find $z \in \mathbb{Z}_q^m \setminus \{\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}\}$

such that

- $\|z\| \leq \beta$
- $Az = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

Lattice-based cryptography: SIS

Definition

SIS [Ajt 96]

n, q, β, m

- n: dimension $\in \mathbb{N}$
- q: modulus $\in \mathbb{N}$
- β: $\beta \ll q$
- m: $\approx$ number of samples $\in \mathbb{N}$

Given $A \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$

Find $z \in \mathbb{Z}_q^m \setminus \{\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}\}$

such that

- $\|z\| \leq \beta$
- $Az = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

? why do we require $\|z\| \leq \beta$?

Lattice-based cryptography: SIS

Definition

SIS n, q, β, m [Ajt 96]

- dimension $\in \mathbb{N}$
- modulus $\in \mathbb{N}$
- $\beta \ll q$
- $\approx$ number of samples $\in \mathbb{N}$

Given $A \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$

Find $z \in \mathbb{Z}_q^m \setminus \{\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}\}$

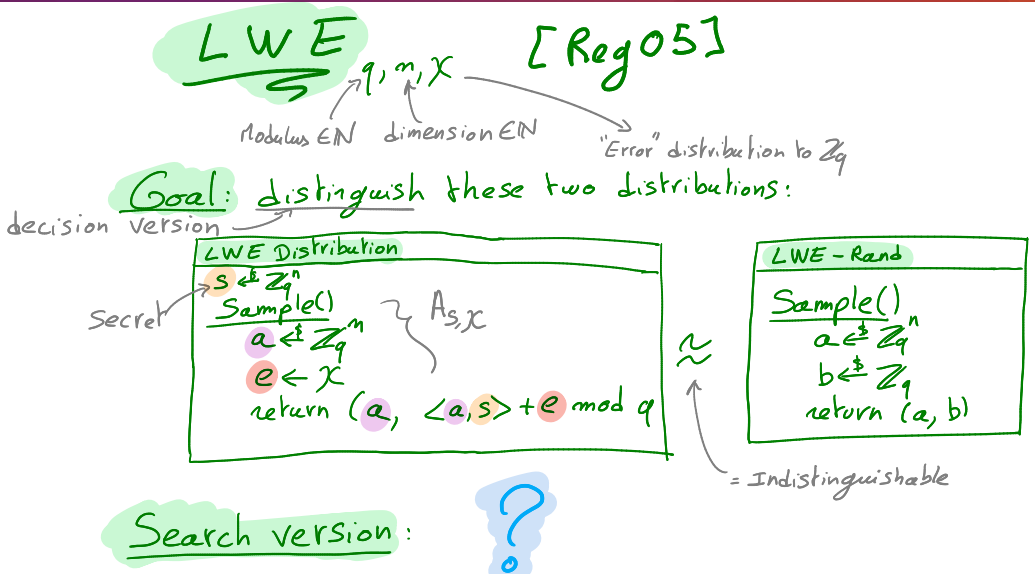
such that

- $\|z\| \leq \beta$
- $Az = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

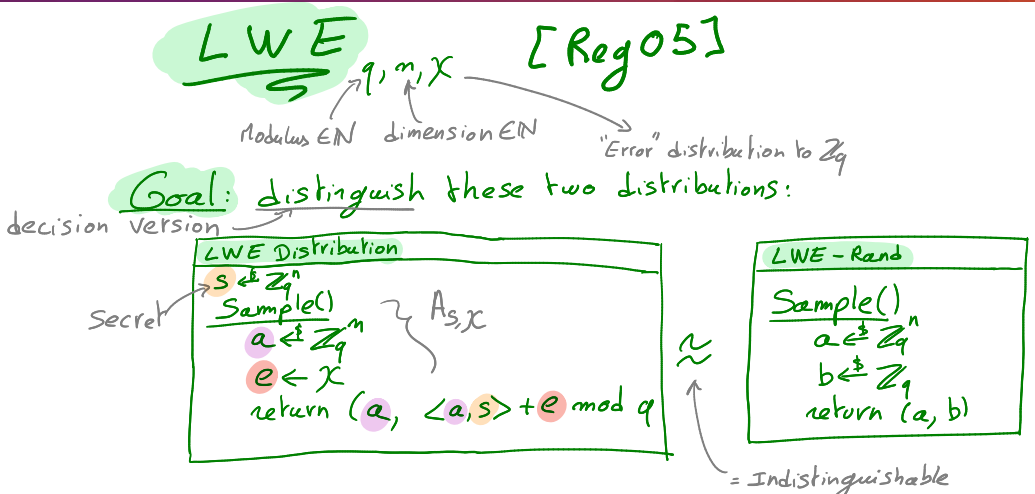
? why do we require $\|z\| \leq \beta$?

→ Otherwise easy to solve (Gaussian elimination)

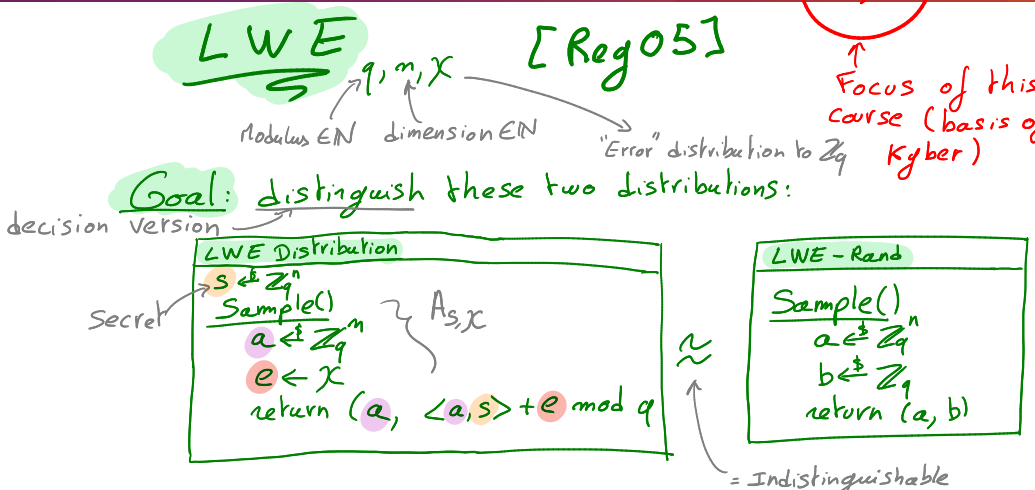
Lattice-based cryptography: LWE



Lattice-based cryptography: LWE



Lattice-based cryptography: LWE



Lattice-based cryptography: LWE

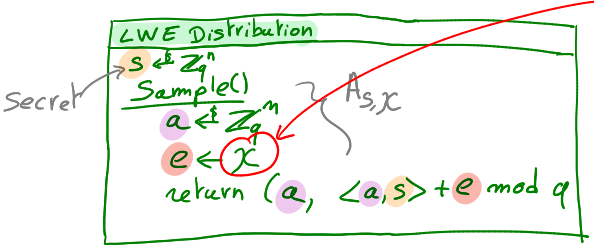
LWE q, n, χ [Reg05]

q: Modulus $\in \mathbb{N}$ — n: dimension $\in \mathbb{N}$ — χ: "Error" distribution to $\mathbb{Z}_q$

Focus of this course (basis of Kyber)

How to choose error χ ?

- If χ outputs 0: easy



Search version: find s from samples from $A_{s,\chi}$.

Lattice-based cryptography: LWE

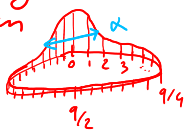
LWE q, n, χ [Reg05]

- q: Modulus $\in \mathbb{N}$
- n: dimension $\in \mathbb{N}$
- χ: "Error" distribution to $\mathbb{Z}_q$

Focus of this course (basis of Kyber)

How to choose error χ ?

- If χ outputs 0: easy (Gaussien elimination)
- χ usually a discrete Gaussian



LWE Distribution

Secret: $s \xleftarrow{\$} \mathbb{Z}_q^n$

Sample():

$a \xleftarrow{\$} \mathbb{Z}_q^n$

$e \leftarrow \chi$

return $(a, \langle a, s \rangle + e \bmod q)$

$A_{s,\chi}$

Search version: find s from samples from $A_{s,\chi}$.

Lattice-based cryptography: LWE

Definition (Discrete Gaussian)

Let $\rho_{\alpha q}(x) \coloneqq \exp\left(\frac{-\pi\|x\|^2}{(\alpha q)^2}\right)$. Then the discrete Gaussian distribution $\chi_{\alpha q}$ on $\mathbb{Z}_q$ is such that:

$$\forall x \in \mathbb{Z}_q, \Pr\left[\chi_{\alpha q} = x\right] \propto \rho_{\alpha q}(x)$$

LWE: matrix form

If we restrict the number of samples[1] to m, we can define the LWE in a matrix form:

Definition (Decision $\mathrm{LWE}_{n,q,\chi,m}$, matrix form)

Let $A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, $s \xleftarrow{\$} \mathbb{Z}_q^n$, $e \xleftarrow{\$} \chi^m$, and $b \xleftarrow{\$} \mathbb{Z}_q^m$, the goal is to distinguish $(A, As + e)$ from (A, b).

[1] This makes the problem even harder… but not much as we can simulate more samples from enough samples.

Lattice-based cryptography: Hardness of LWE

If we can solve LWE, we can solve GapSVP$_\gamma$ (so **LWE = more secure than lattices**):

Theorem ([Reg05])

For any $m = \text{poly}(n)$, modulus $q \leq 2^{\text{poly}(n)}$, and discrete Gaussian distribution $\chi_{\alpha q}$ with $\alpha q \geq 2\sqrt{n}$ and $0 < \alpha < 1$, solving the decision LWE$_{n,q,\chi,m}$ problem is at least as hard as quantumly[a] solving GapSVP$_\gamma$ and SIVP$_\gamma$ on arbitrary n-dimensional lattices, for $\gamma = \tilde{O}(n/\alpha)$.

[a] See also [Pei09] for a classical reduction.

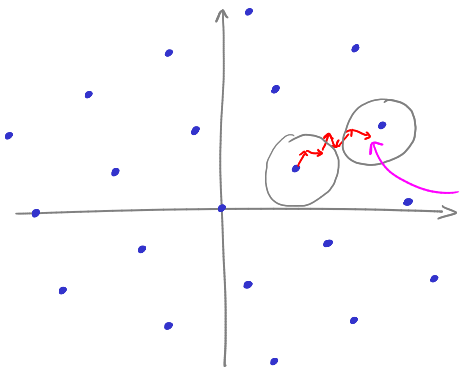
NB: The best polynomial algorithm to solve GapSVP$_\gamma$ work only for $\gamma = 2^{\Theta(n \log\log n / \log n)}$, so GapSVP$_\gamma$ believe to be secure for:

- $\gamma = \text{poly}(n)$ (polynomial noise ratio): preferred assumption, enough for encryption
- super-polynomial (but sub-exponential) γ: try to avoid it when possible (less secure, less efficient), but sometimes needed (e.g. most Fully Homomorphic Encryption schemes, but efforts to avoid it have been made [BS23] (not simulation-secure))

Polynomial vs super-polynomial γ

Less secure $\Leftrightarrow \gamma := \tilde{O}(n/\alpha)$ "big" $\Leftrightarrow \alpha$ "small" → Useful as summing the errors keep the error small (= possible to decrypt)

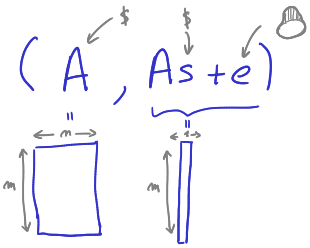
size error



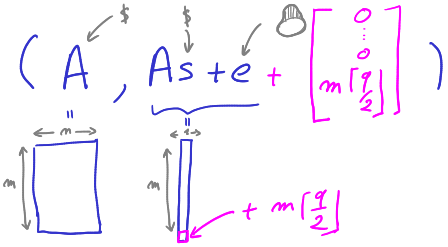
Encryption from LWE

Regev's encryption

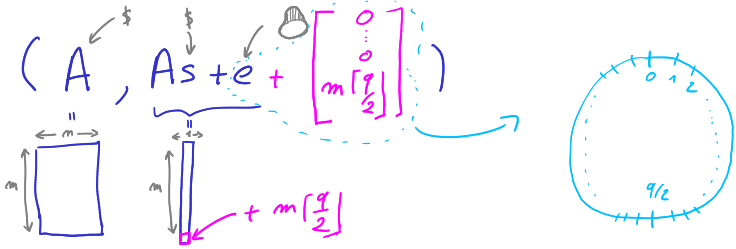
Reminder LWE:



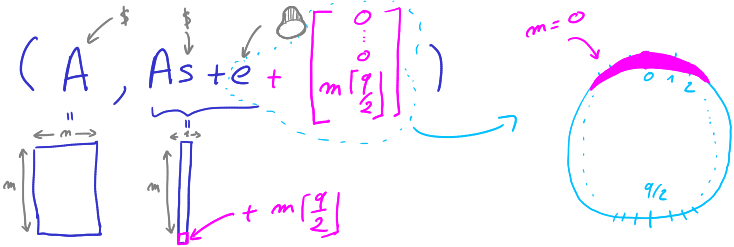
Regev's encryption



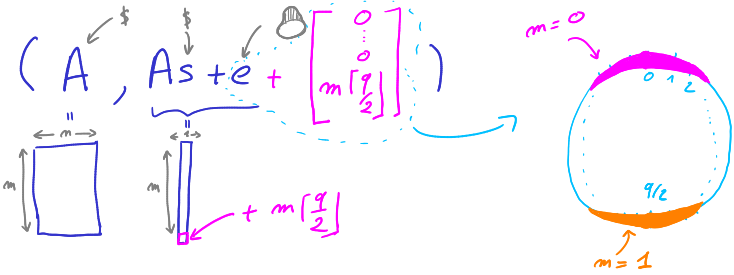
Regev's encryption



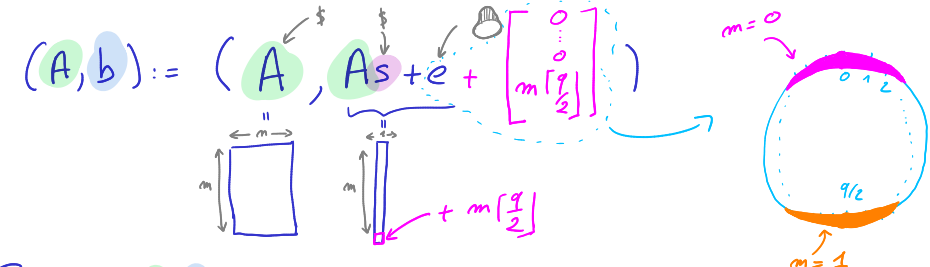
Regev's encryption



Regev's encryption



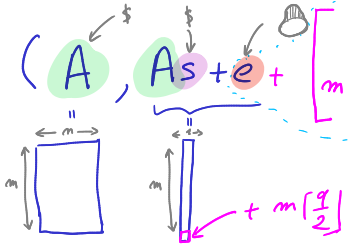
Regev's encryption

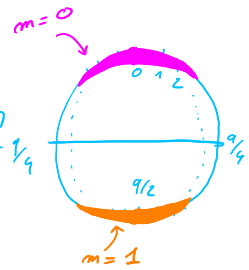


Claim 1: Given (A, b) and s, I can recover m

Regev's encryption

$$(A, b) := \left(A \xleftarrow{\$}, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rfloor \end{bmatrix} \xleftarrow{\$}\right)$$





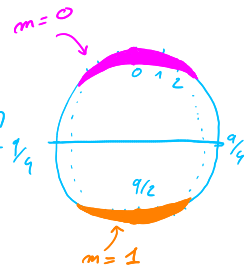
Claim 1: Given (A, b) and s, I can recover m

$\rightarrow$ $b - As = e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\,q/2 \end{bmatrix}$ $\rightsquigarrow$ if norm last coordinate $< \frac{q}{4}$ $m = 0$

else $m = 1$

Regev's encryption

$$(A, b) := \left(A \xleftarrow{\$},\ As + e \xleftarrow{\$} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rfloor \end{bmatrix}\right)$$

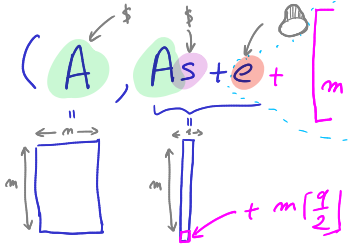


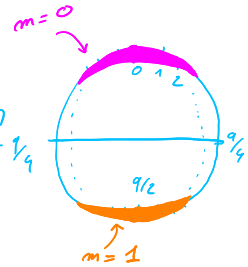
Claim 1: Given (A, b) and s, I can recover m

Claim 2: Given (A, b), hard to guess m

Regev's encryption

$$(A, b) := \left(A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rfloor \end{bmatrix}\right)$$





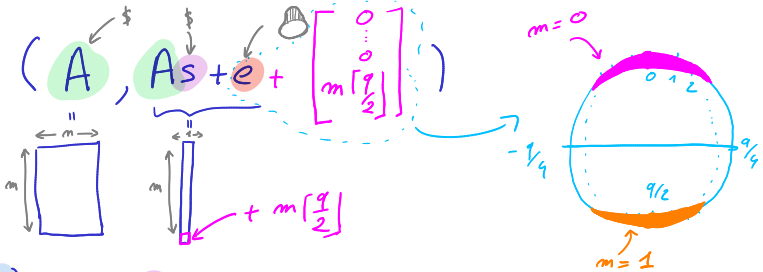
Claim 1: Given (A, b) and s, I can recover m

Claim 2: Given (A, b), hard to guess m

→ Decision-LWE: $(A, As + e) \approx (A, b)$

Regev's encryption

$(A, b) := (A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rceil \end{bmatrix})$



Claim 1: Given (A, b) and s, I can recover m

Claim 2: Given (A, b), hard to guess m

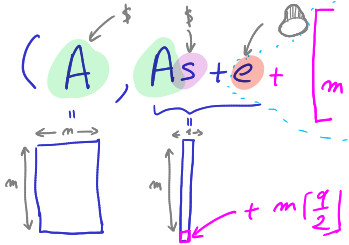
→ Decision-LWE: $(A, As + e) \approx (A, b)$

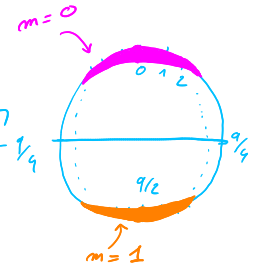
So $(A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil q/2 \rfloor \end{bmatrix}) \approx (A, b + \text{cte})$

cte

Regev's encryption

$$(A, b) := \left(A \xleftarrow{\$},\ As + e \xleftarrow{\$} + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rceil \end{bmatrix}\right)$$





Claim 1: Given (A, b) and s, I can recover m

Claim 2: Given (A, b), hard to guess m

→ Decision-LWE: $(A, As + e) \approx (A, b)$

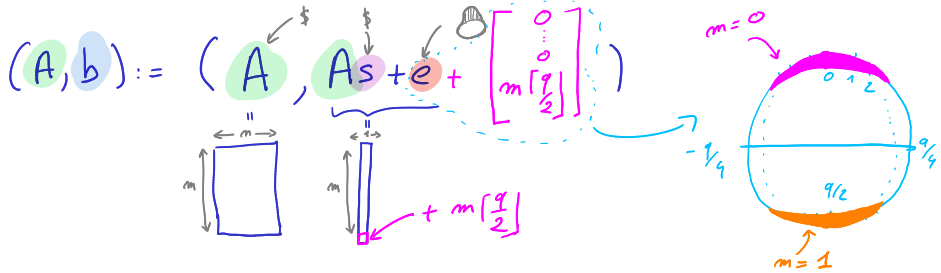
So $(A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil q/2 \rfloor \end{bmatrix}) \approx (A, b + \text{cte}) \approx (A, b)$

cte

Uniform distribution + cte = uniform dist.

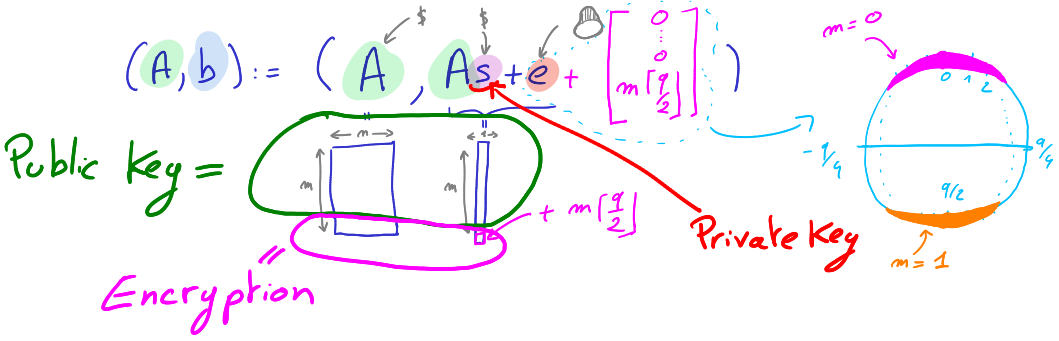
No info on m

Regev's encryption



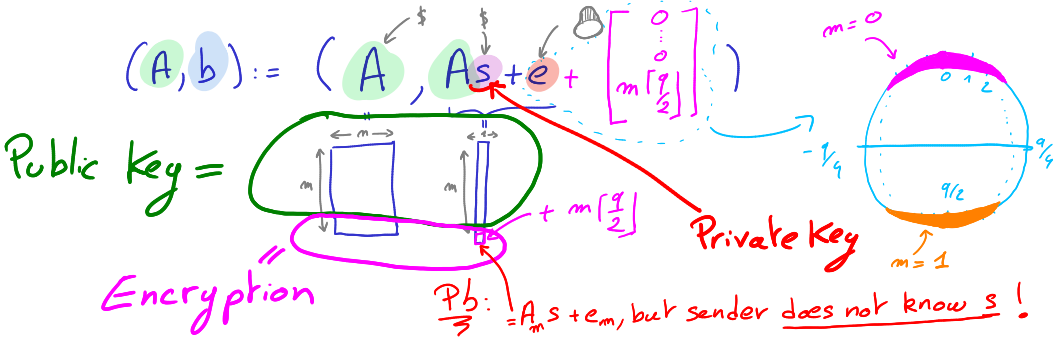
How to turn it into an encryption?

Regev's encryption



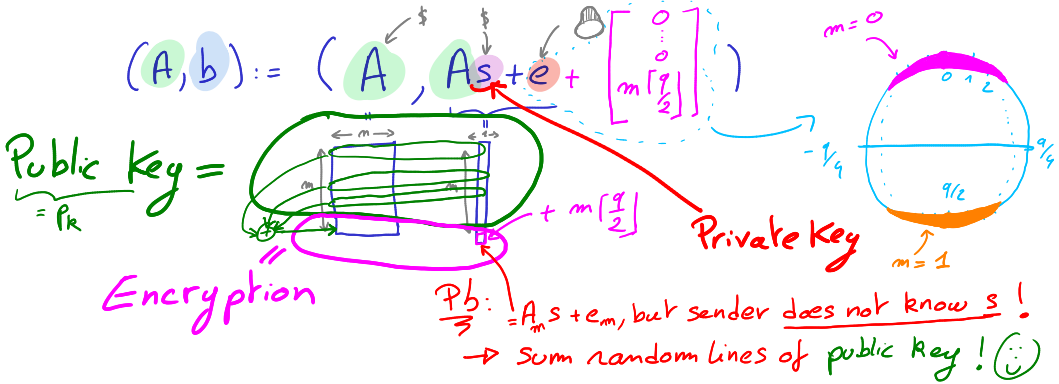
How to turn it into an encryption?

Regev's encryption



How to turn it into an encryption ?

Regev's encryption

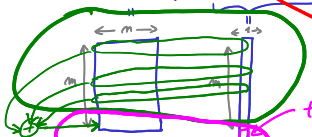


How to turn it into an encryption ?

Regev's encryption

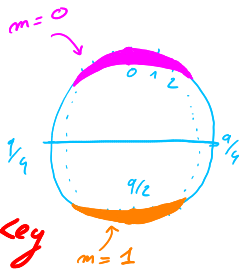
$(A, b) := (A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rceil \end{bmatrix})$

Public Key = P_k



$+ m\lceil \frac{q}{2} \rceil$

Private Key



Encryption

$= \left(x \cdot P_k + (0, \dots, 0, m\lceil \frac{q}{2} \rceil) \right)$

$x \xleftarrow{\$} \{0,1\}^{m-1}$ = choice of lines

New LWE sample !

$P_b := A_m s + e_m$, but sender does not know s !

→ Sum random lines of public key ! ☺

How to turn it into an encryption ?

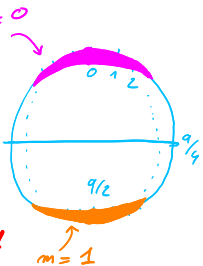
Regev's encryption

$(A, b) := (A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rceil \end{bmatrix})$

Public key = $= p_k$

$+ m\lceil \frac{q}{2} \rceil$

Private key



Encryption

$p_b : = A_m s + e_m$, but sender does not know s !

→ Sum random lines of public key ! ☺

$= x \, p_k + (0, \dots, 0, m\lceil \frac{q}{2} \rceil) = \sum_{i / x_i = 1} (A_i, A_i s + e_i)$

New LWE sample !

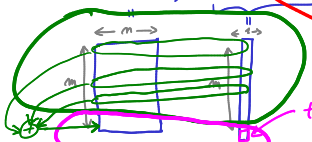
$x \in \{0,1\}^{m-1}$ = choice of lines

How to turn it into an encryption ?

Regev's encryption

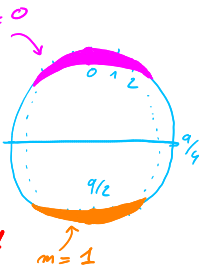
$(A, b) := (A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rceil \end{bmatrix})$

Public key = $= p_k$



$+ m\lceil \frac{q}{2} \rceil$

Private key



Encryption

$p_b := A_m s + e_m$, but sender does not know s !

→ Sum random lines of public key ! 🙂

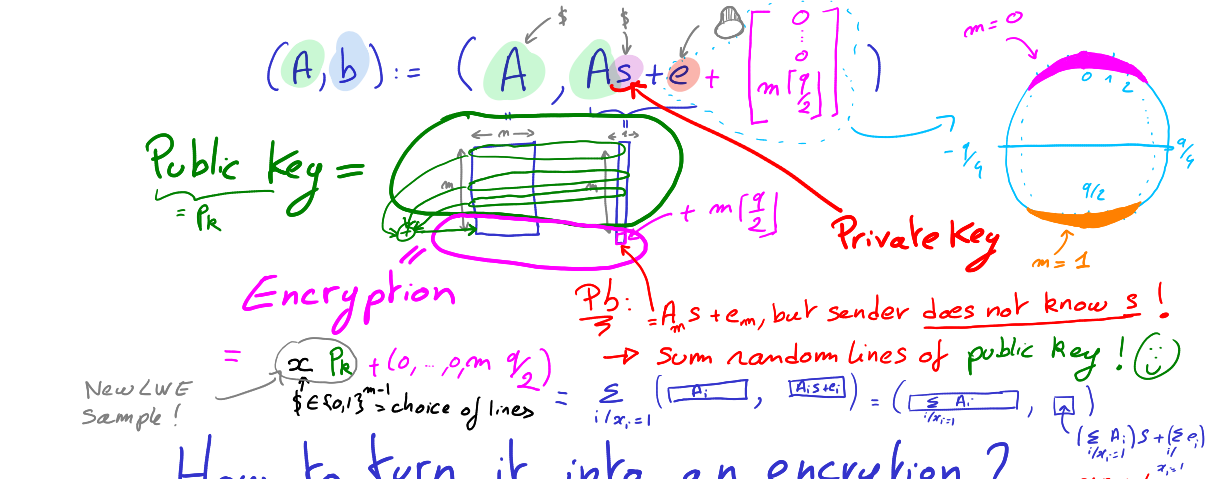
$= x \, p_k + (0, \dots, 0, m\lfloor \frac{q}{2} \rfloor) = \sum_{i / x_i = 1} (A_i, A_i s + e_i) = (\sum_{i/x_i=1} A_i, \sum_{i/x_i=1} A_i s + \sum_{i/x_i=1} e_i)$

$x \in \{0,1\}^{m-1}$ = choice of lines

New LWE sample !

How to turn it into an encryption ?

Regev's encryption

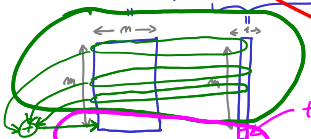


How to turn it into an encryption ?

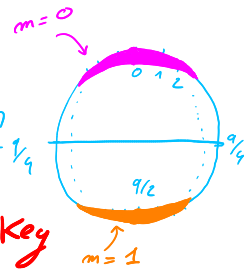
Regev's encryption

$(A, b) := (A, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m\lceil \frac{q}{2} \rceil \end{bmatrix})$

Public key = $= p_k$



Private key



Encryption

$= \left(x \cdot p_k + (0, \dots, 0, m\lceil \frac{q}{2} \rceil)\right)$

$x \in \{0,1\}^{m-1}$ = choice of lines

$= \sum_{i / x_i = 1} (A_i, A_i s + e_i) = \left(\sum_{i/x_i=1} A_i, \sum_{i/x_i=1} A_i s + \sum_{i/x_i=1} e_i\right)$

New LWE sample!

$p_b := A_m s + e_m$, but sender does not know s!

→ Sum random lines of public key! 🙂

How to turn it into an encryption?

error a bit bigger

⇒ To have $\sum_{i/x_i=1} e_i < q/4$ we must have α small enough

Regev's encryption (summary)

Definition (Regev's encryption)

Given $(n, m, q) \in \mathbb{N}^3$, $\alpha \in (0, 1)$, and χ an error distribution:

Regev's encryption

$\textsc{Gen}(1^\lambda)$:
$A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$
$s \xleftarrow{\$} \mathbb{Z}_q^n$
$e \leftarrow \chi^m$
return $(p_k := (A, As + e), s_k := s)$

$\textsc{Enc}(p_k := (A, b), m \in \{0, 1\})$:
$x \xleftarrow{\$} \{0, 1\}^m$
return $(x^T A, x^T b + m\lceil q/2 \rfloor)$

$\textsc{Dec}(s_k := s, (a, y))$:
if $y - as \mod q < q/4$ or $y - as \mod q > 3q/4$: **return** 0
else: **return** 1

Regev's encryption: security

How to choose the parameters?

m must be large enough. Can you find an attack if $m = O(\log \lambda)$?

Regev's encryption: security

How to choose the parameters?

m must be large enough. Can you find an attack if $m = O(\log \lambda)$?
$\hookrightarrow$ If $m = O(\log \lambda)$, a malicious eavedropper trying to decrypt (a, y) can bruteforce x (we have the good x when $a = x^T A$), taking time $2^m = \text{poly}(\lambda)$. Then, $m = \frac{y - x^T b}{\lceil q/2 \rfloor}$.

Regev's encryption: security

How to choose the parameters?

Should α be:

1. As small as possible
2. As large as possible
3. Neither too small nor too large

Regev's encryption: security

How to choose the parameters?

Should α be:

1. As small as possible
2. As large as possible
3. Neither too small nor too large ✓
   ↪ If α is too small = not secure (e.g. $\alpha = 0$ solve with Gaussian elimination).
   ↪ Too big = impossible to decrypt (modulus norm of the noise must be smaller than $q/4$)

Regev's encryption: security

Theorem (Security Regev's encryption)

Let ε be a negligible function. Regev's encryption is secure and $(1 - 2\varepsilon$, i.e. overwhelmingly) correct, if $\mathsf{LWE}_{n,q,\chi,m}$ is hard and if

- $\alpha q \sqrt{m \ln(1/\varepsilon)/\pi} \leq \frac{q}{4}$,
- $2^{(n+1)q-m} = \mathsf{negl}(\lambda)$.

In particular, we can choose the following parameters for large enough[a] $\lambda = n \geq 730$:

- $\varepsilon = \frac{1}{2^\lambda}$,
- $q = n \log n$,
- $m = 2(n+1) \log q$,
- $\alpha = \frac{1}{n \log n}$.

[a] No attempt is made to optimize the efficiency or security, here 730 is just a lower-bound so that the first condition applies. We can also get better asymptotic security with $\varepsilon = 1/2^{\log n}$.

Regev's encryption: security

Proof idea: Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ $b \in \mathbb{Z}_q^m$

= random

$\mathrm{Enc}_{pk}(m) := \left(x^T A, x^T (As+e) + m\lceil \frac{q}{2} \rfloor \right)$

Public key pk

Public key = pk

$+ m\lceil \frac{q}{2} \rfloor$

Private key

Encryption

Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($b \in \mathbb{Z}_q^m$) (no trapdoor!)

$\rightsquigarrow$ We can replace the public key $(A, As+e)$ with (A, b)

= random

$$\mathrm{Enc}_{pk}(m) := \left(x^T A,\ x^T(As+e) + m\left\lceil \tfrac{q}{2} \right\rfloor\right)$$

Public key pk

Public key = pk = Encryption

$+ m\left\lceil \frac{q}{2} \right\rfloor$

Private key

Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($b \in \mathbb{Z}_q^m$) (no trapdoor!)

$\rightsquigarrow$ We can replace the public key $(A, As+e)$ with (A, b)

= random

$$\mathrm{Enc}_{pk}(m) := \left(x^T A, x^T (As+e) + m\lceil \tfrac{q}{2} \rfloor \right)$$

Public key pk

Public key = pk

$= pk \cdot x + e + m\lceil \frac{q}{2} \rfloor$

Encryption

Private key

Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($b \in \mathbb{Z}_q^m$, $\$$, $\approx$, no trapdoor!)

$\rightsquigarrow$ We can replace the public key $(A, As+e)$ with (A, b)

Step 2: (A, b) is lossy: i.e. $(x^T A,\ x^T b) \approx (a', y')$ if $m \geq (n+1)\log q$ by leftover hash lemma.

= random

$$Enc_{pk}(m) := \left(x^T A,\ x^T(As+e) + m\left\lceil \tfrac{q}{2} \right\rfloor\right)$$

Public key pk

Public key = pk = A | $As+e$

$+ m\lceil \frac{q}{2} \rfloor$

Private Key

Encryption

Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($b \in \mathbb{Z}_q^m$) (no trapdoor!)

$\rightsquigarrow$ We can replace the public key $(A, As+e)$ with (A, b)

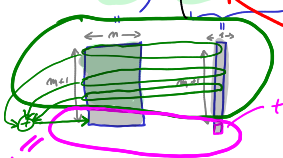
Step 2: (A, b) is lossy: i.e. $(x^T A,\ x^T b) \approx (a', y')$ if $m \geq (n+1)\log q$ by leftover hash lemma.

= random

$$Enc_{pk}(m) := \left(x^T A,\ x^T(As+e) + m\left\lceil \frac{q}{2} \right\rfloor\right)$$

Public key pk

Public key = pk =



$+ m\left\lceil \frac{q}{2} \right\rfloor$

Private Key

Encryption

Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$, $s \xleftarrow{\$} \mathbb{Z}_q^n$, $e \xleftarrow{\chi} \mathbb{Z}_q^m$, $b \xleftarrow{\$} \mathbb{Z}_q^m$) (no trapdoor!)

⇝ We can replace the public key $(A, As+e)$ with (A, b)

Step 2: (A, b) is lossy: i.e. $(x^T A,\ x^T b) \approx (a', y')$ if $m \geq (n+1)\log q$ by leftover hash lemma.

⇝ statistically impossible to learn m from $(x^T A,\ x^T b + m\lceil \frac{q}{2} \rfloor)$

= random

$$\mathrm{Enc}_{pk}(m) := \left(x^T A,\ x^T (As+e) + m\lceil \tfrac{q}{2} \rfloor\right)$$

Public key pk

Public key = pk = Encryption

$+ m\lceil \frac{q}{2} \rfloor$

Private key

Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($b \in \mathbb{Z}_q^m$) (no backdoor!)

⇝ We can replace the public key $(A, As+e)$ with (A, b)

Step 2: (A, b) is lossy: i.e. $(x^T A,\ x^T b) \approx (a', y')$ if $m \geq (n+1)\log q$ by leftover hash lemma.

⇝ statistically impossible to learn m from $(x^T A,\ x^T b + m\lceil \frac{q}{2} \rfloor) \approx (a', y' + m\lceil \frac{q}{2} \rfloor)$

= random

$$\mathrm{Enc}_{pk}(m) := \left(x^T A,\ x^T(As+e) + m\lceil \tfrac{q}{2} \rfloor\right)$$

Public key pk

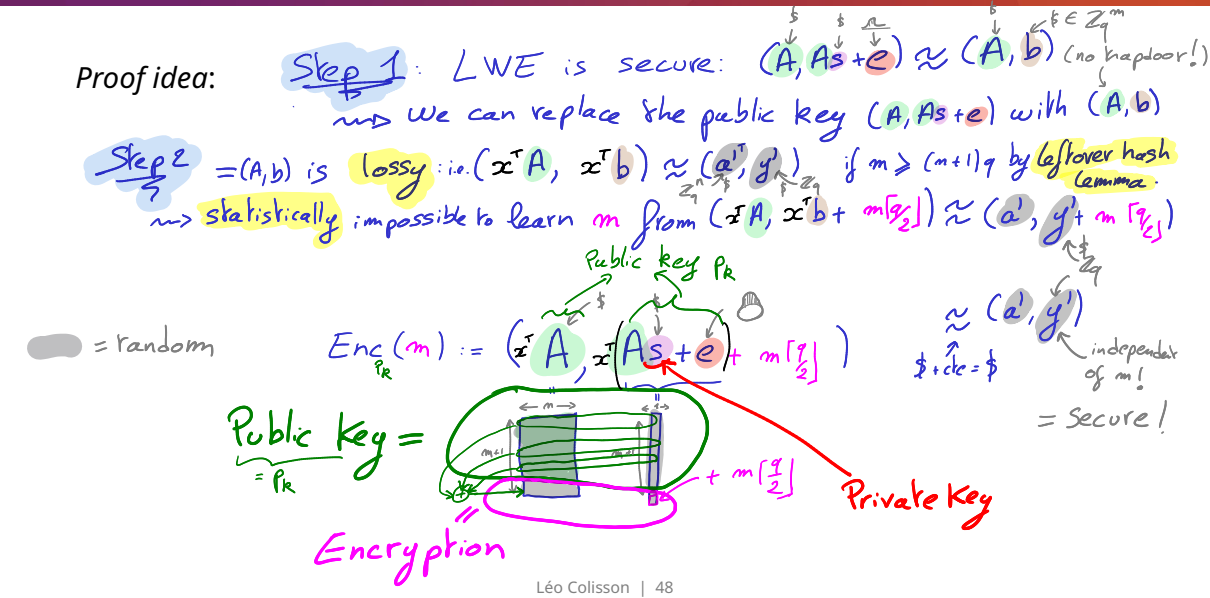
Public key = pk = Encryption

$+ m\lceil \frac{q}{2} \rfloor$

Private key

Regev's encryption: security

Proof idea:



Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($b \in \mathbb{Z}_q^m$) (no backdoor!)

⇝ We can replace the public key $(A, As+e)$ with (A, b)

Step 2: (A, b) is lossy: i.e. $(x^T A,\ x^T b) \approx (a', y')$ if $m \geq (n+1)\log q$ by leftover hash lemma.

⇝ statistically impossible to learn m from $(x^T A,\ x^T b + m\lceil q/2 \rfloor) \approx (a', y' + m\lceil q/2 \rfloor)$

$\approx (a', y')$

$\$ + \text{cte} = \$$

y' independent of m!

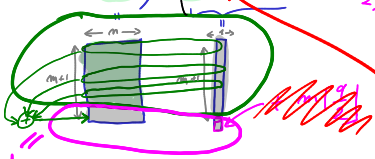
= secure!

= random

$\text{Enc}_{pk}(m) := \left(x^T A,\ x^T(As+e) + m\lceil q/2 \rfloor\right)$

Public key pk

Public key = pk



Encryption

Private key

Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As+e) \approx (A, b)$ ($\$ \in \mathbb{Z}_q^m$) (no trapdoor!)

⇝ We can replace the public key $(A, As+e)$ with (A, b)

Step 2: (A, b) is lossy: i.e. $(x^T A,\ x^T b) \approx (a', y')$ if $m \geq (n+1)\log q$ by leftover hash lemma.

⇝ statistically impossible to learn m from $(x^T A,\ x^T b + m\lceil q/2 \rfloor) \approx (a', y' + m\lceil q/2 \rfloor)$

$\approx (a', y')$

$\$ + c\$ = \$$

independent of m!

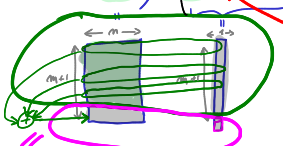
= secure!

= random

$\mathrm{Enc}_{pk}(m) := \left(x^T A,\ x^T(As+e) + m\lceil q/2 \rfloor\right)$

Public key pk

Public key = pk



Encryption

Private key

Leftover hash lemma

Useful and common lemma to quantify randomness of output. **Applications**:

- Prove security of cryptographic schemes (here Regev's encryption)
- Extract randomness from a somewhat imperfect random source (increase entropy)
- Obtain a better (more uniform but shorter) key from a longer key that partially leaked to an adversary:
  $\hookrightarrow$ Used a lot in Quantum Key Distribution

Leftover hash lemma informally: if a possibly unbounded adversary has partial arbitrary knowledge about a key, we can hash this key (with salting and a special kind of hash functions called 2-universal) to extract from it a shorter key looking completely uniform to the adversary.

= Entropy

Leftover hash lemma (particular case)

Particular case of leftover hash lemma for Regev's proof (perfect entropy, but function not strictly speaking 2-universal as it fails in the unlikely event where input is 0^m):

Theorem (Particular case of Leftover hash lemma, rewriting of [Reg05, Claim 5.3])

Let G be an Abelian group, and let $m \in \mathbb{N}$. If $\sqrt{\frac{|G|}{2^m}} = \mathsf{negl}(\lambda)$, then:

Leftover-real		Leftover-rand
$(g_1, \dots, g_m) \xleftarrow{\$} G^m$ GetElements(): **return** $(g_1, \dots, g_m)$ Sample(): $x \xleftarrow{\$} \{0,1\}^m$ **return** $\sum_i x_i g_i$	$\approx$	$(g_1, \dots, g_m) \xleftarrow{\$} G^m$ GetElements(): **return** $(g_1, \dots, g_m)$ Sample(): $r \xleftarrow{\$} G$ **return** r

Note: the distribution of samples is even **statistically close**, hence even unbounded adversaries have no significant advantage to distinguish a single sample.

Regev's encryption: formal proof

?

Using the previous lemmas, write a formal proof (using the formalism of *Joy of cryptography*) proving the security of Regev's encryption, based on the proof idea saw before.

Hint: $\frac{b}{u}\mathbb{Z} = G$ reɔisnoƆ.

Regev’s encryption: formal proof

Correction: on board

LWE in real life

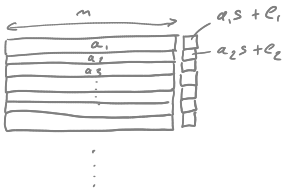
LWE in real life

LWE = basis of **Kyber**, the new post-quantum encryption standard, but with some improvements:

- Use the more efficient **ring**-LWE: like LWE, but replace each of the m lines in $\mathbb{Z}_q^n$ with polynomials in $\mathbb{F}_q[X]/\phi(x)$ where $\phi(x)$ is an irreducible degree n polynomial used as a modulo ($\phi(x) = 0$), more details next slide.
- Multiplication is done more efficiently via **Number-Theoretic Transform** (NTT, equivalent of Fourier Transform)
- Encode **multiple bits** at once
- Just implements a **Key-Exchange Mechanism** (KEM, basically encrypt a random key) instead of encryption (use symmetric cryptography like AES to encrypt = more efficient)
- All details in **FIPS 203**: `https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.203.pdf`

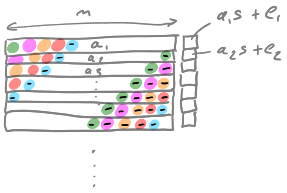
Intuition efficiency ring-LWE

LWE



Intuition efficiency ring-LWE

LWE ⟶ Ring-LWE



Idea: for increased efficiency, use correlated samples like

$a_1 = [x_1 | x_2 | \cdots | x_n]$

$a_2 = [x_2 | x_3 | \cdots | x_n | -x_1]$

$a_3 = [x_3 | x_4 | \cdots | -x_1 | -x_2]$

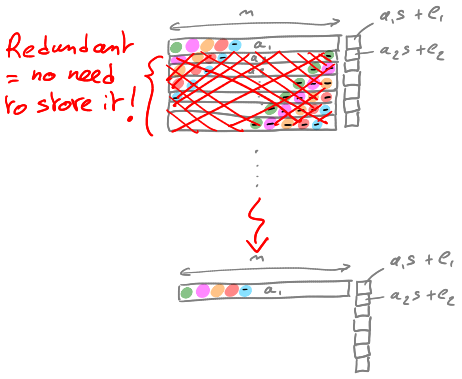
$\vdots$

$a_n = [x_n | -x_1 | \cdots | -x_{n-1}]$

by grouping samples n-by-n.

Intuition efficiency ring-LWE

LWE ⟶ Ring-LWE



Idea: for increased efficiency, use correlated samples like

$a_1 = [x_1 | x_2 | \cdots | x_n]$

$a_2 = [x_2 | x_3 | \cdots | x_n | -x_1]$

$a_3 = [x_3 | x_4 | \cdots | -x_1 | -x_2]$

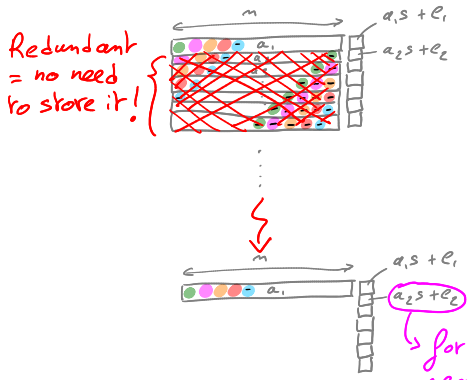
$\vdots$

$a_n = [x_n | -x_1 | \cdots | -x_{n-1}]$

by grouping samples n-by-n.

Intuition efficiency ring-LWE

LWE ⟶ Ring-LWE



for efficiency reasons, represent a_1, s and e with polynomials!

Idea: for increased efficiency, use correlated samples like

$a_1 = [x_1 | x_2 | \cdots | x_n]$

$a_2 = [x_2 | x_3 | \cdots | x_n | -x_1]$

$a_3 = [x_3 | x_4 | \cdots | -x_1 | -x_2]$

$\vdots$

$a_n = [x_n | -x_1 | \cdots | -x_{n-1}]$

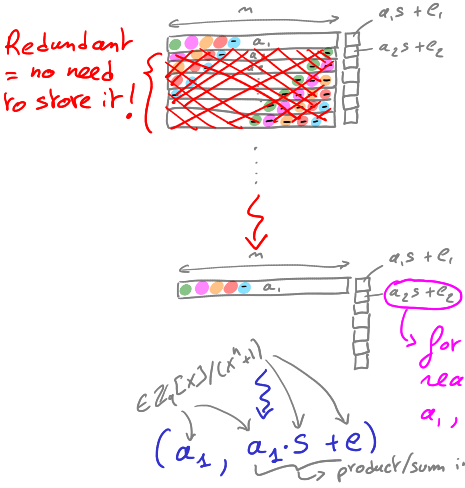
by grouping samples n-by-n.

$$\begin{cases} a_1 = x_1 + x_2 X + \cdots + x_n X^{n-1} \\ a_2 = a_1 X \mod (X^n + 1) \\ \vdots \\ a_n = a_1 X^{n-1} \mod (X^n + 1) \end{cases}$$

↑ i.e. $X^n = -1$

Intuition efficiency ring-LWE

LWE ⟶ Ring-LWE



Idea: for increased efficiency, use correlated samples like

$a_1 = [x_1 | x_2 | \cdots | x_n]$

$a_2 = [x_2 | x_3 | \cdots | x_n | -x_1]$

$a_3 = [x_3 | x_4 | \cdots | -x_1 | -x_2]$

$\vdots$

$a_n = [x_n | -x_1 | \cdots | -x_{n-1}]$

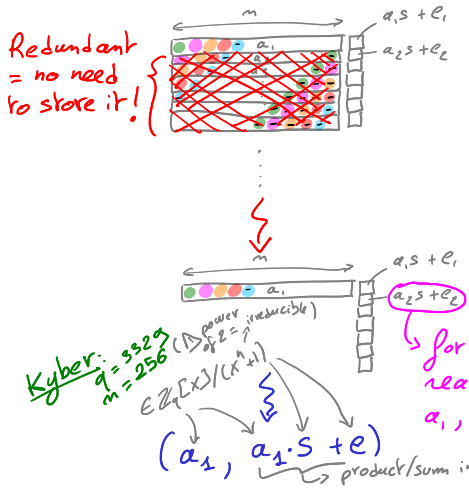
by grouping samples n-by-n.

$$\begin{cases} a_1 = x_1 + x_2 X + \cdots + x_n X^{n-1} \\ a_2 = a_1 X \mod (X^n+1) \\ \vdots \\ a_n = a_1 X^{n-1} \mod (X^n+1) \end{cases}$$

(↑ i.e. $X^n = -1$)

Intuition efficiency ring-LWE

LWE ⟶ Ring-LWE



for efficiency reasons, represent a_1, s and e with polynomials!

Idea: for increased efficiency, use correlated samples like

$a_1 = [x_1 | x_2 | \cdots | x_n]$

$a_2 = [x_2 | x_3 | \cdots | x_n | -x_1]$

$a_3 = [x_3 | x_4 | \cdots | -x_1 | -x_2]$

$\vdots$

$a_n = [x_n | -x_1 | \cdots | -x_{n-1}]$

by grouping samples n-by-n.

$$\begin{cases} a_1 = x_1 + x_2 X + \cdots + x_n X^{n-1} \\ a_2 = a_1 X \mod (X^n+1) \\ \vdots \\ a_n = a_1 X^{n-1} \mod (X^n+1) \end{cases}$$

↑ i.e. $X^n = -1$

Quotient ring

In the quotient ring $\mathbb{Z}_{10}[X]/(X^4+1)$, what is the value of

$$P := (4 + 2X + X^2)(3X + 2X^3)$$

?

(Quotient by X^4+1 means $X^4+1=0$ i.e. $X^4=-1$)

Quotient ring

In the quotient ring $\mathbb{Z}_{10}[X]/(X^4+1)$, what is the value of

?

$$
\begin{aligned}
P &:= (4+2X+X^2)(3X+2X^3) \\
&= 12X+8X^3+6X^2+4X^4+3X^3+2X^5 \\
&= 2X+8X^3+6X^2+4X^4+3X^3+2X^5 \\
&= 2X+8X^3+6X^2-4+3X^3+2XX^4 \\
&= 2X+8X^3+6X^2-4+3X^3-2X \\
&= 11X^3+6X^2-4 \\
&= X^3+6X^2-4
\end{aligned}
$$

(Quotient by X^4+1 means $X^4+1=0$ i.e. $X^4=-1$)

decision$-R_q-$LWE$_{\chi,m}$

Let R_q be a quotient ring (e.g. Kyber uses $\mathbb{Z}_q[X]/(X^n+1)$, note that n must be a power of 2 to have X^n+1 irreducible). Then, for any $s \in R_q$ and distribution χ on R_q, we define the distribution $A^{\text{ring}}_{s,\chi}$ as follows: first we sample a uniformly random $a \xleftarrow{\$} R_q$, we sample $e \leftarrow \chi$, and we output

$$(a, s \cdot a + e)$$

where the operations are performed in R_q. The decision$-R_q-$LWE$_{\chi,m}$ **problem** is defined as follows: given m independent samples $(a_i, b_i) \in R_q \times R_q$ from either $A^{\text{ring}}_{s,\chi}$ or from the uniform distribution, the goal is to distinguish in which case we are with non-negligible advantage.

Harness of LWE

Security reductions **more complicated**!
We reduce LWE to special lattices called **ideal lattices** (with additional structure following the ones in the ring), but **GapSVP is easy** (!) in these lattices. Instead, we rely on the hardness of approximate SIVP/SVP in these lattices, and **some subtleties arise with the noise distribution** (hardness independent of number of sample known only if the noise distribution is not symmetric):

Harndess Ring-LWE [LPR10]

For any $m = \text{poly}(n)$, cyclotomic ring R of degree n over $\mathbb{Z}$, and "appropriate choices" of modulus q and error distribution χ of error rate $\alpha < 1$, solving decision$-R_q-\text{LWE}_{\chi,m}$ is at least as hard as quantumly solving SVP_γ on arbitrary ideal lattices in R, for some $\gamma = \text{poly}(n)/\alpha$.