

Advanced Crypto 2024

Lattice-based cryptography

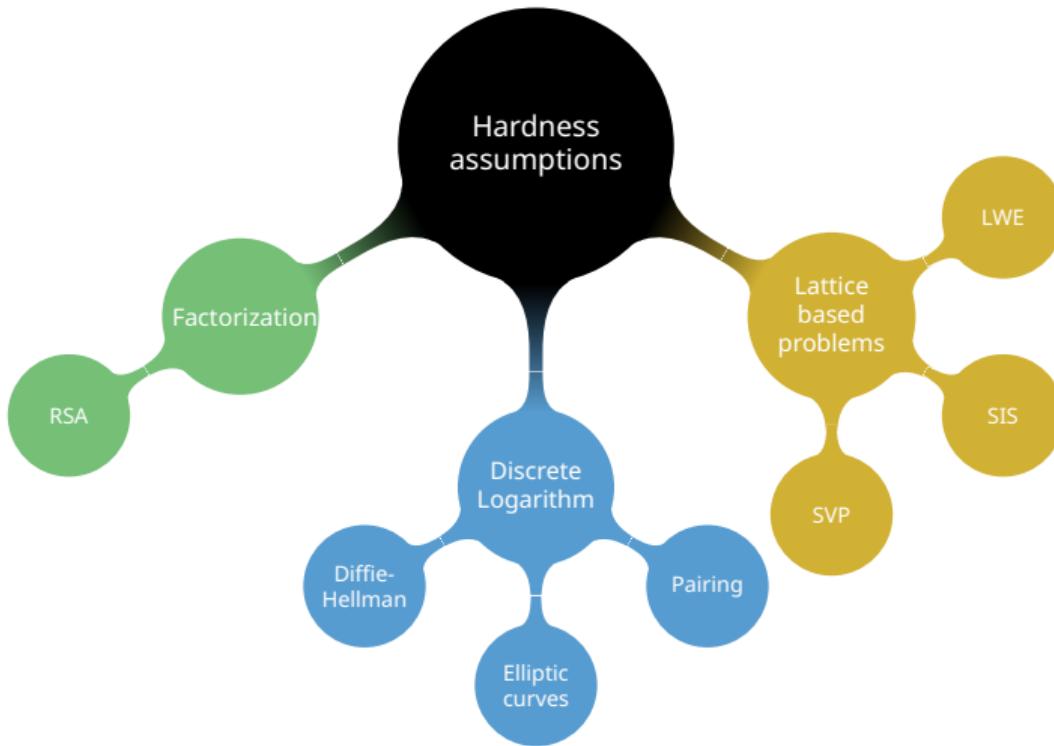
Léo COLISSON PALAIS

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

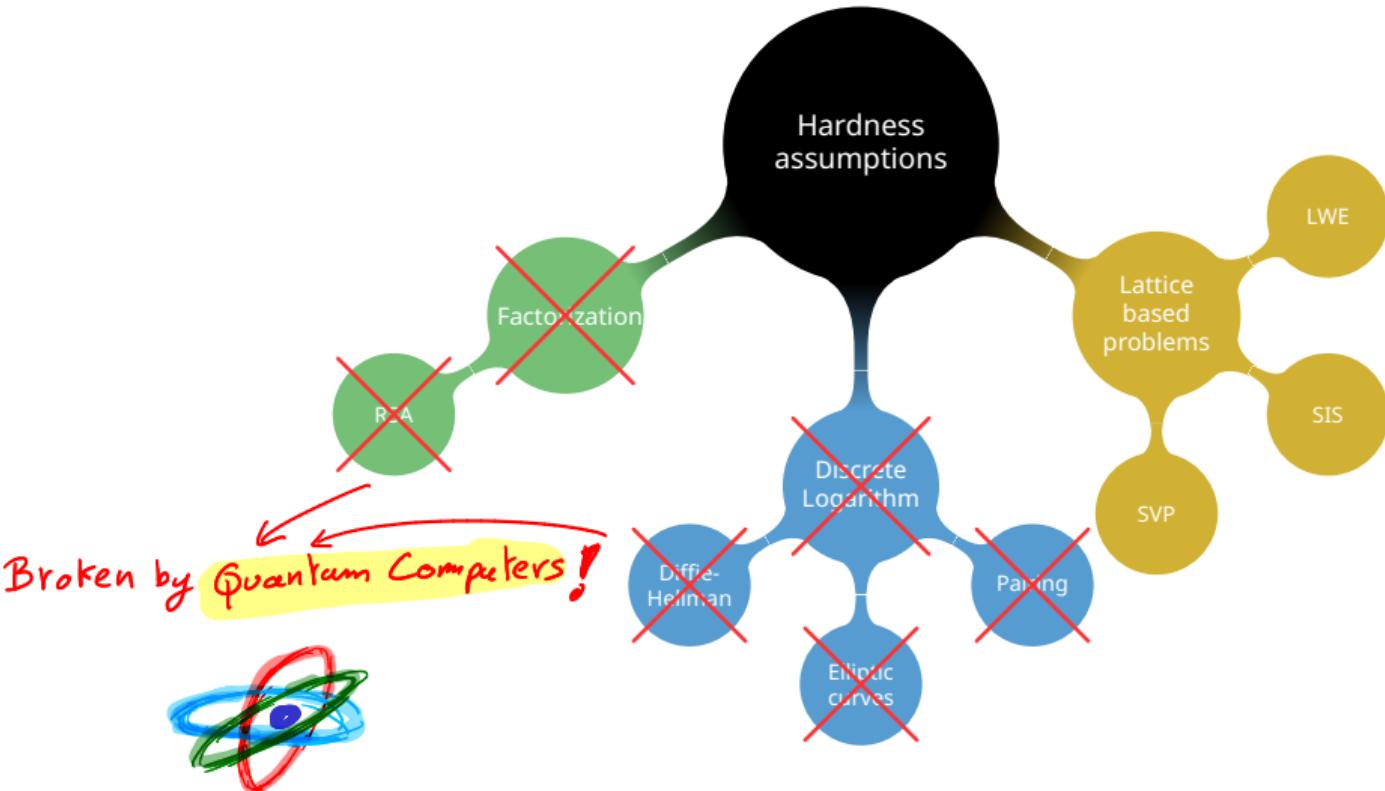
<https://leo.colisson.me/teaching.html>

Motivations

Motivations



Motivations



Motivations



imgflip.com

Motivations



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



JAKE-CLARK.TUMBLR

imgflip.com

Motivations



- Should we change technology **now or** can we **wait** until quantum computers arrive?
⇒ **Cannot wait!** "Harvest now, decrypt later"



JAKE-CLARK.TUMBLR

imgflip.com

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



imgflip.com

JAKE-CLARK.TUMBLR

Motivations



imgflip.com
JAKE-CLARK.TUMBLR

- 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



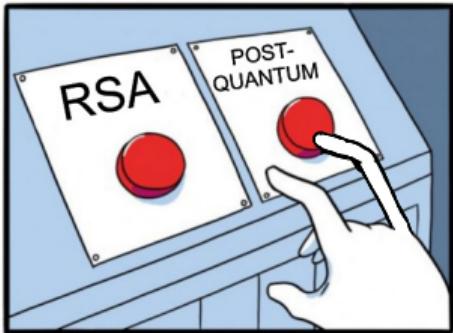
- Should we change technology **now or** can we **wait** until quantum computers arrive?
⇒ **Cannot wait!** "Harvest now, decrypt later"
- **Warning:** can't change too quickly; **need enough time** to analyse the new candidate



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

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

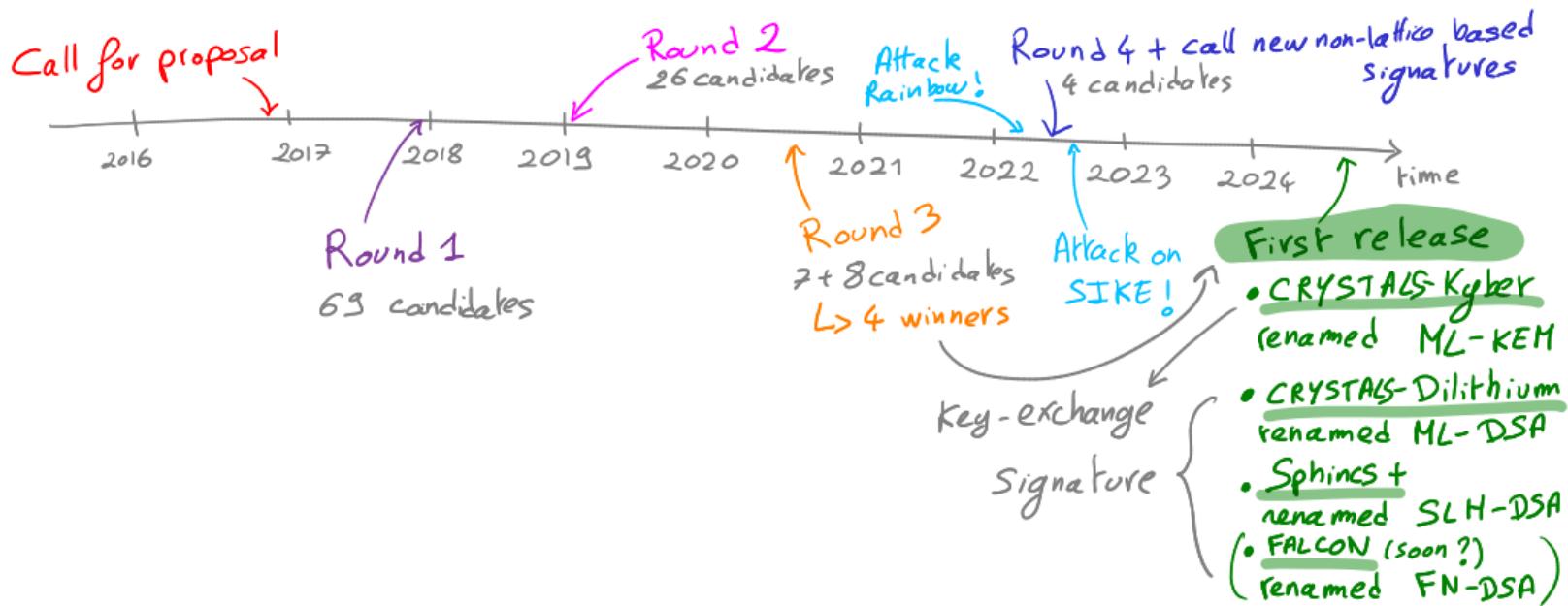
Motivations



imgflip.com

- 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)	Signature	
• <u>CRYSTALS-Kyber</u> renamed ML-KEM	Lattice-based	Lattice-based	Hash-based
P_K : 1184 S_K : 2400 cipher : 1088 shared key : 32	1184 2400 1088 32	1952 4032 3309	48 96 16224
Hardness Assumption In bytes, Level 3			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 X
$ P_R :$	1184 X	32 ✓
$ S_K :$	2400 X	32 ✓
$ \text{cipher} :$	1088 X	64 (2x32) ✓
$ \text{shared key} :$	32 X	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	✓
$ P_K $:	1184	X
$ S_K $:	2400	X
$ cipher $:	1088	X
$ shared\ key $:	32	X

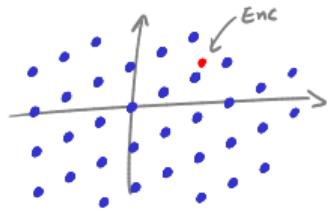
Elliptic-curves X

32	✓
32	✓
64 (2x32)	✓
32	✓

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

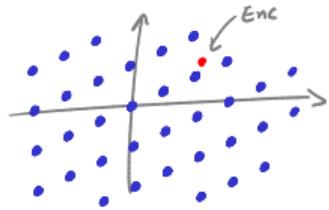
Famous post-quantum candidates

① Lattice-based Crypto

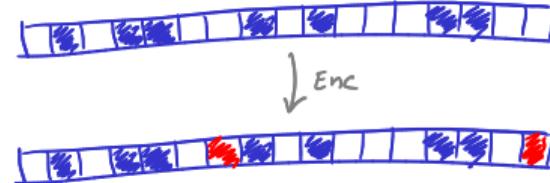


Famous post-quantum candidates

① Lattice-based Crypto

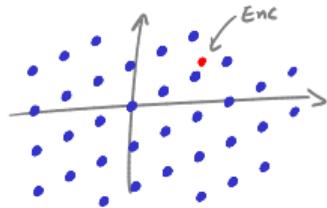


② Code-based Crypto

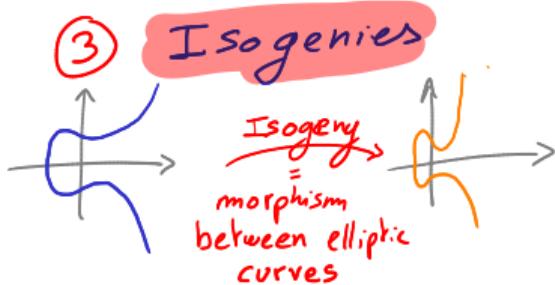
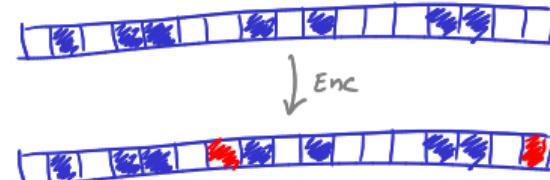


Famous post-quantum candidates

① Lattice-based Crypto

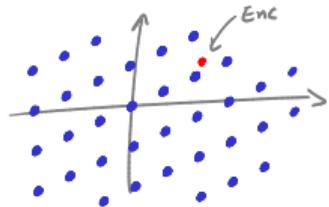


② Code-based Crypto

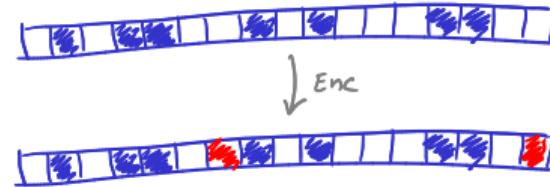


Famous post-quantum candidates

① Lattice-based Crypto



② Code-based Crypto



③ Isogenies

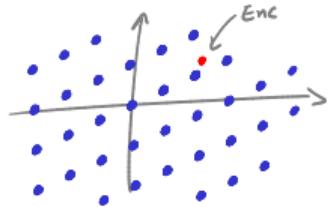
Isogeny
morphism
between elliptic
curves

④ Multivariate Crypto

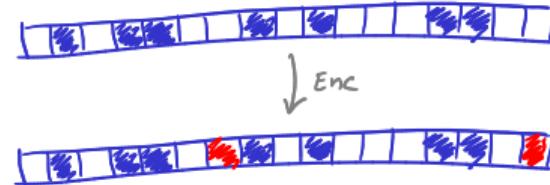
$$P_k := \begin{cases} \cdot 1 + x_1 + 2x_0x_3 \\ \cdot 4 + x_4 + 3x_1^2x_8 + x_9 \\ \cdot x_6 + x_2^3x_5 + x_7x_5 \end{cases} \xrightarrow{\text{Enc}} P_k(m)$$

Famous post-quantum candidates

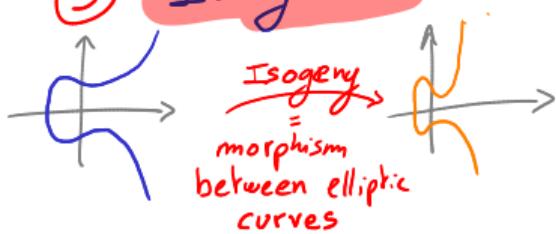
① Lattice-based Crypto



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 !

③ Isogenies

- ✗ • SIDH broken \Rightarrow lost confidence
- ✗ • complicated

② Code-based Crypto

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

④ Multivariate Crypto

- ✗ • many candidates were broken
 \Rightarrow 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 !

③ Isogenies

- ✗ • SIDH broken \Rightarrow lost confidence
- ✗ • complicated

② Code-based Crypto

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

④ Multivariate Crypto

- ✗ • many candidates were broken
 \Rightarrow 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

Algorithms

- LLL \Rightarrow many applications
 - ↳ Integer Linear Programming
 - ↳ Polynomial factorisation over rationals

Complexity theory

Rare example of worst-case to average-case reduction

Lattices

Number theory

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

Cryptography

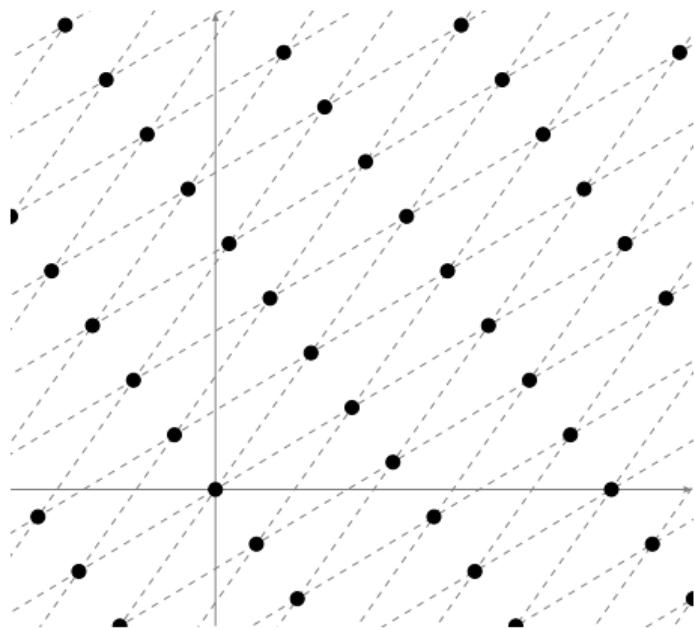
- ↳ Attacks: LLL = break knapsack-based crypto, RSA (for some parameters), ECDSA (partially known nonces)
- ↳ New cryptosystems
Encryption, signatures, FHE...

Definition (Lattice)

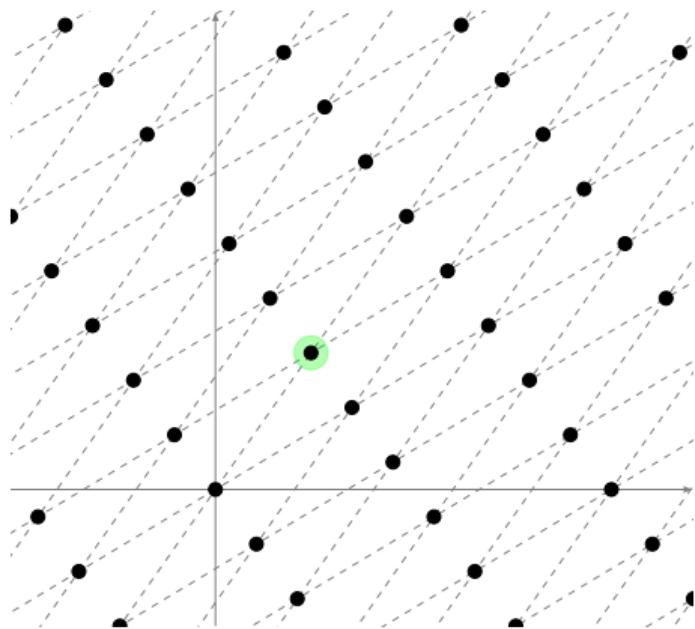
An n -dimensional *lattice* \mathcal{L} is any subset of \mathbb{R}^n that is both:

- 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

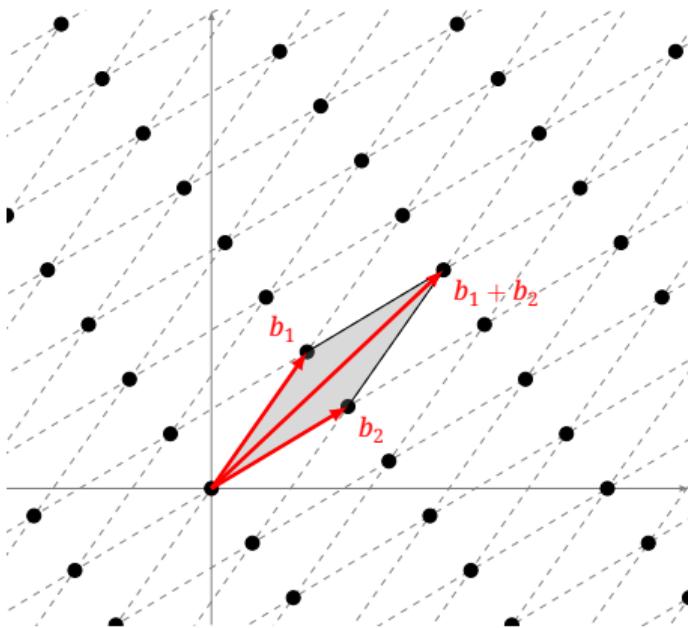
Lattice



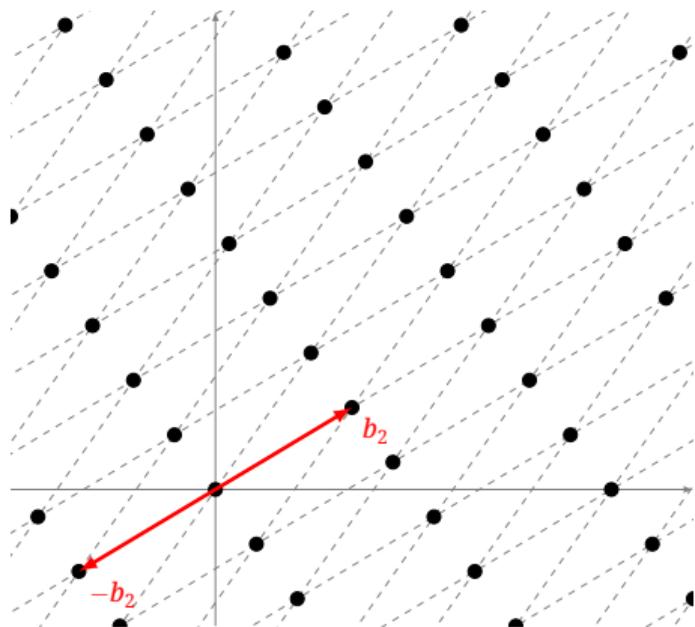
Lattice



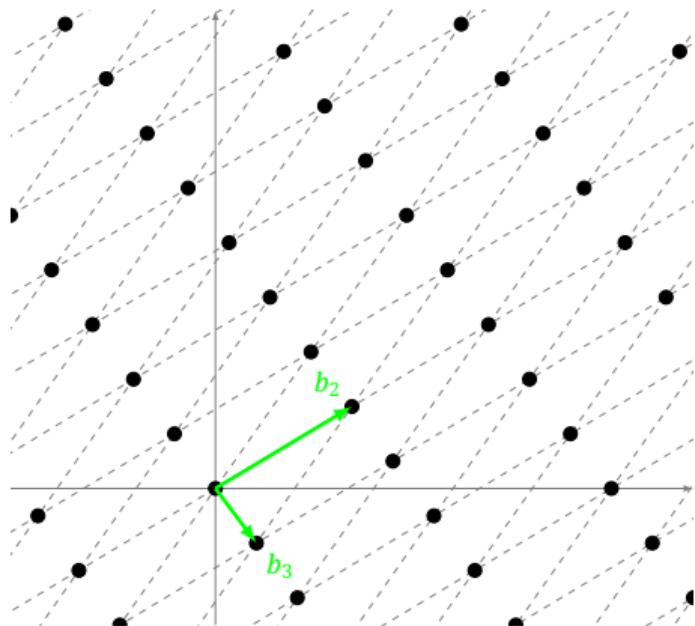
Lattice



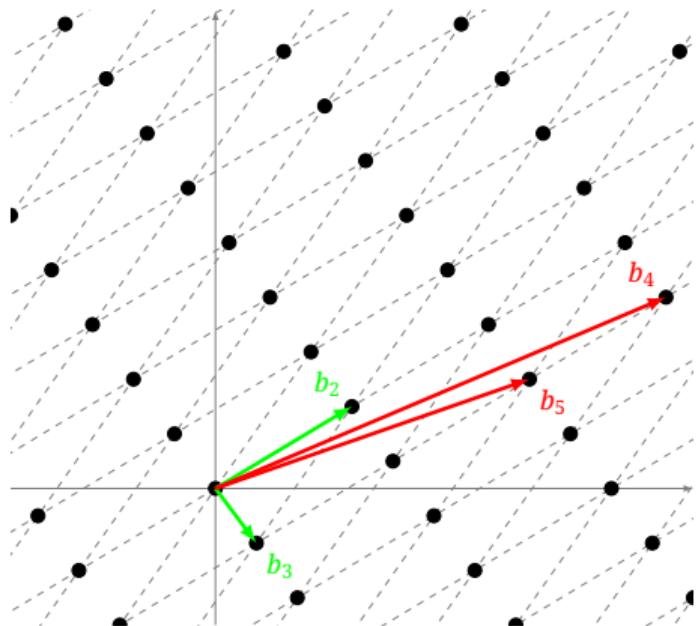
Lattice



Lattice

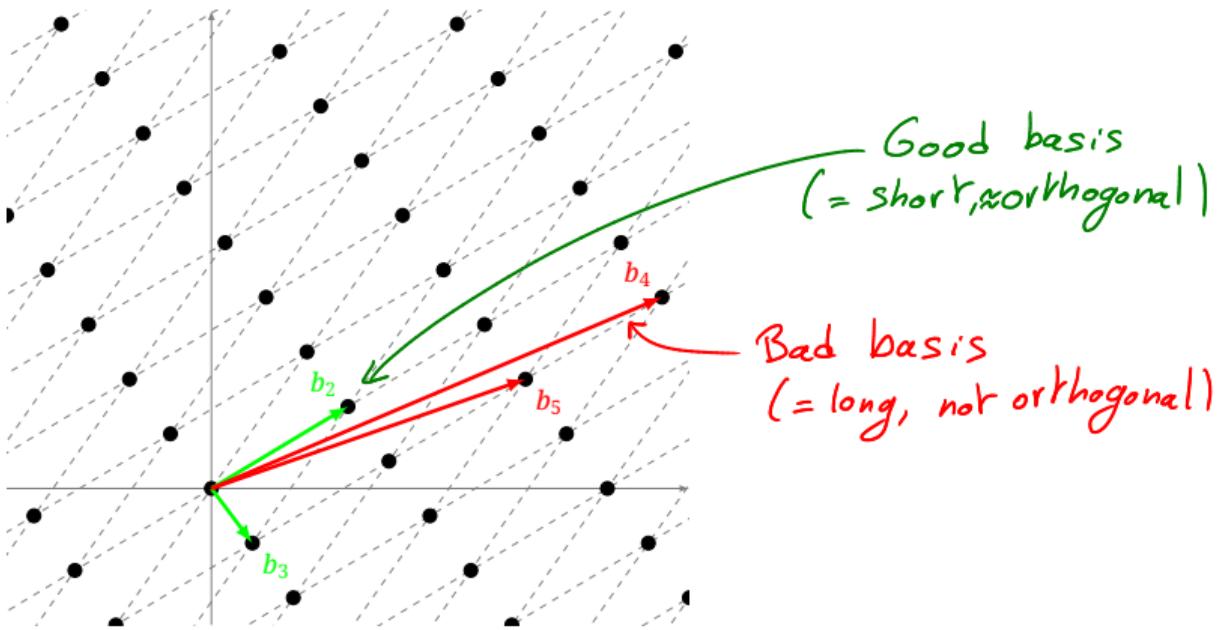


Lattice



Basis = not unique !

Lattice



Lattice: basis

Definition (Basis)

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

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

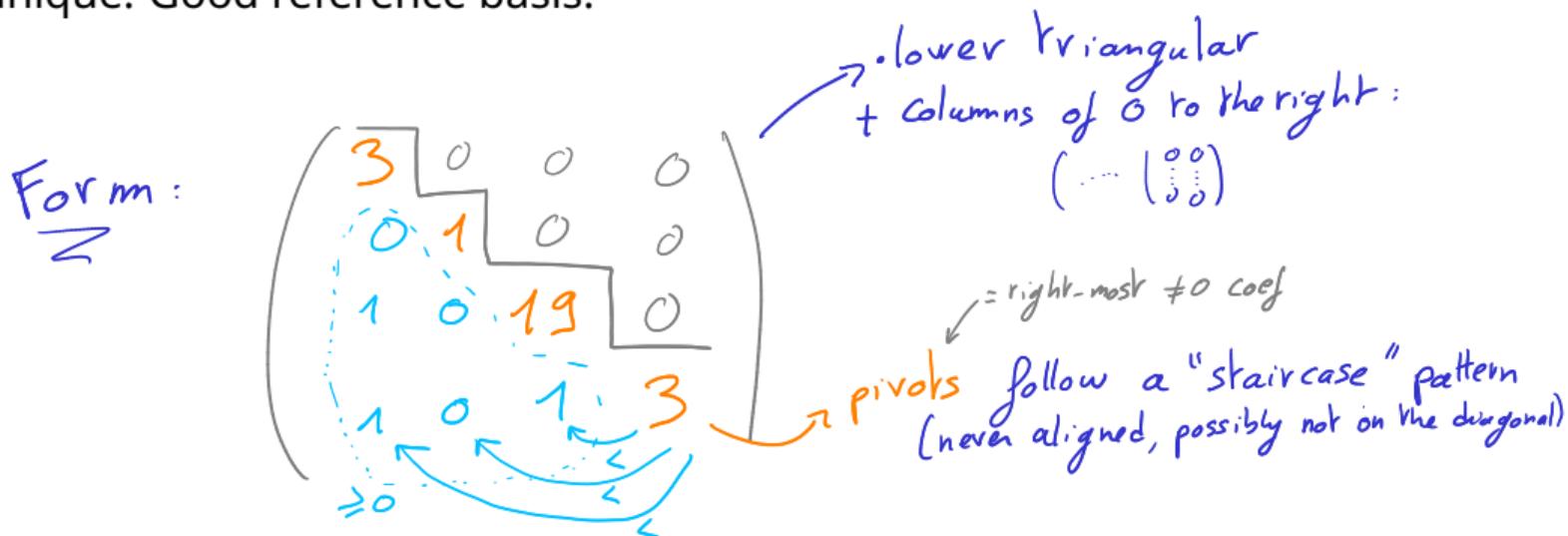
k is the **rank** of the lattice. If $k = n$, the lattice has **full-rank** (often the case).

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?

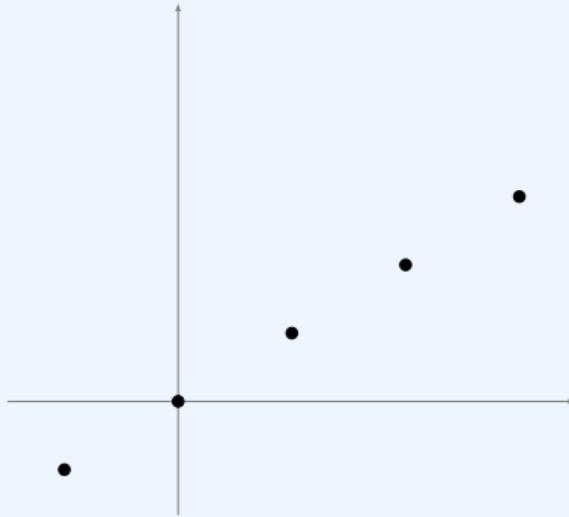
⇒ **Hermite normal form** can always be efficiently be computed and is unique: Good reference basis.



Lattice: basis

What is the dimension and rank of this lattice?

?



Lattice: basis

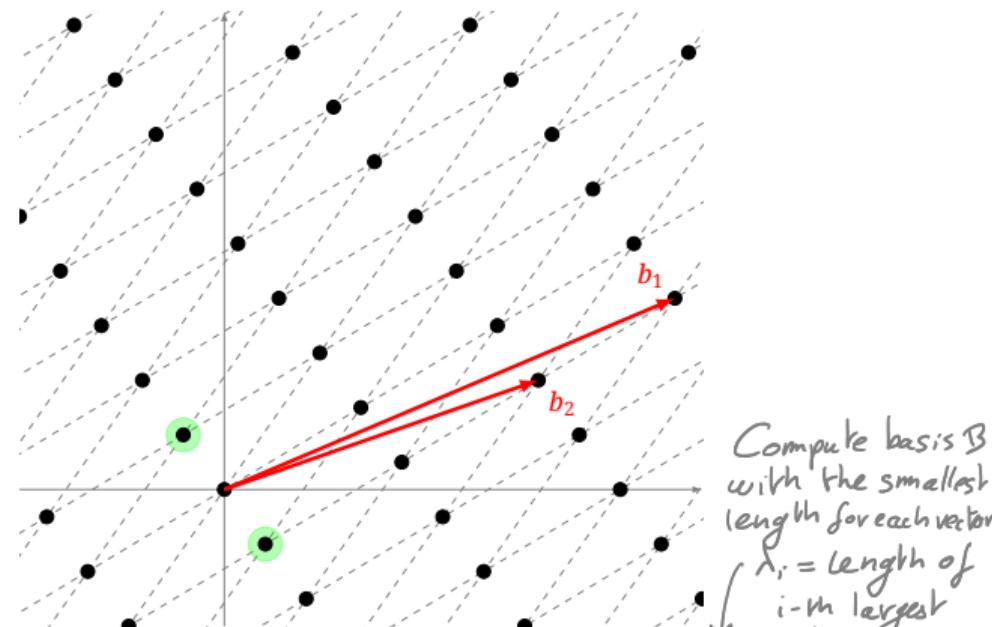
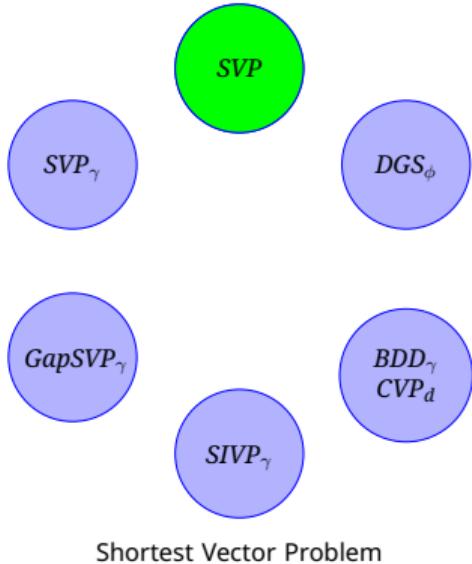
What is the dimension and rank of this lattice?

?



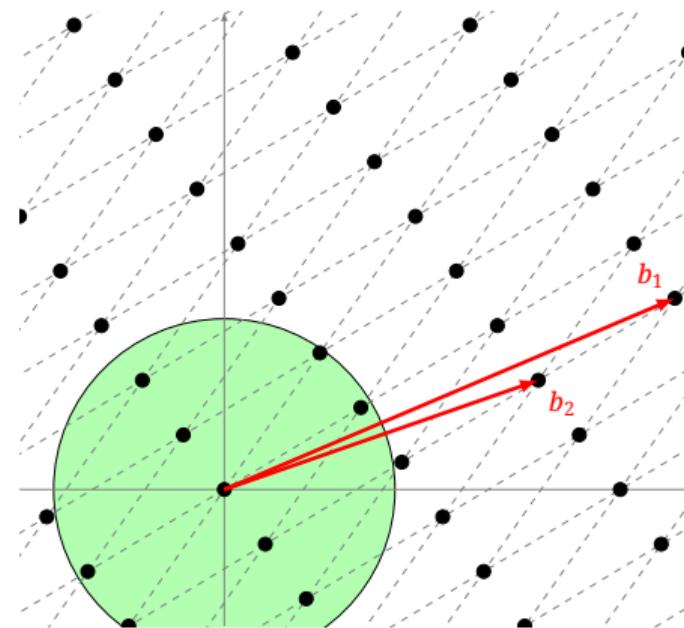
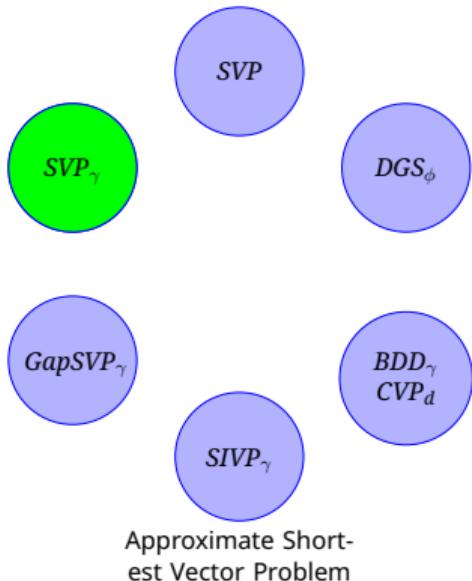
⇒ Dimension is 2, rank is 1

Lattice : what is hard to do?



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

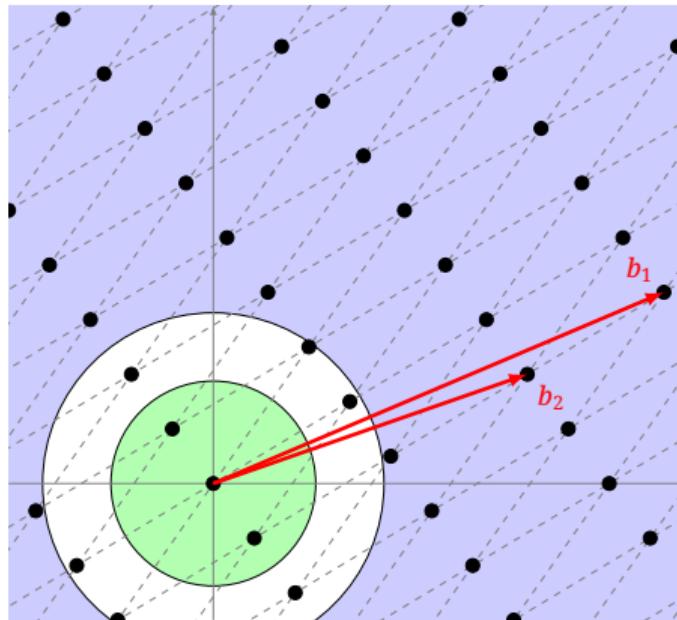
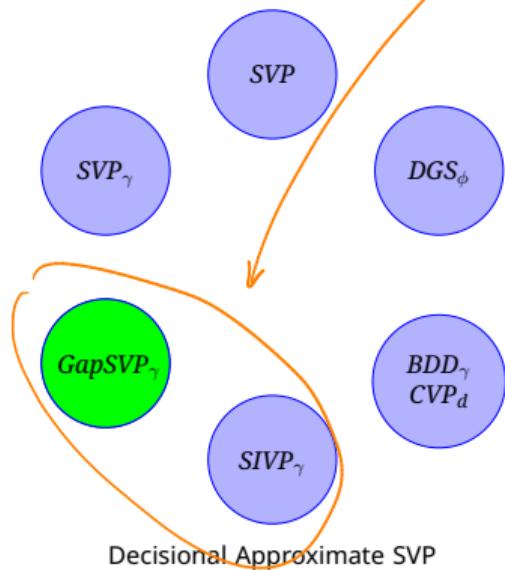
Lattice : what is hard to do?



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

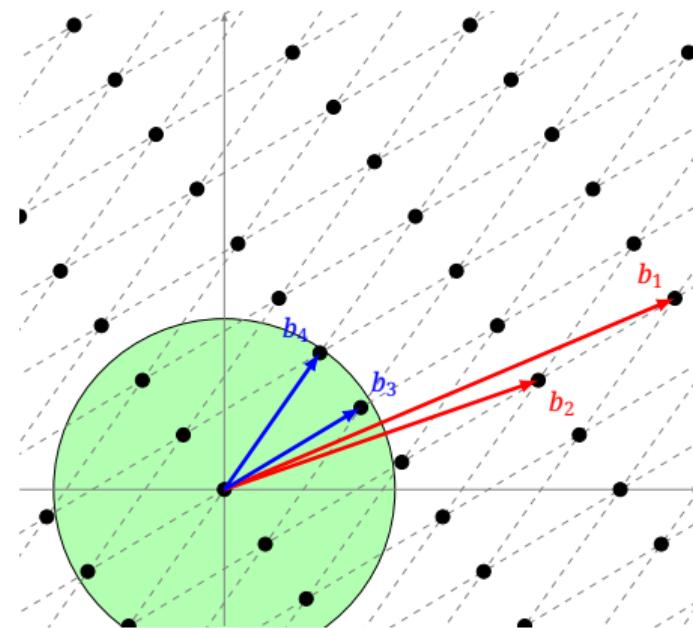
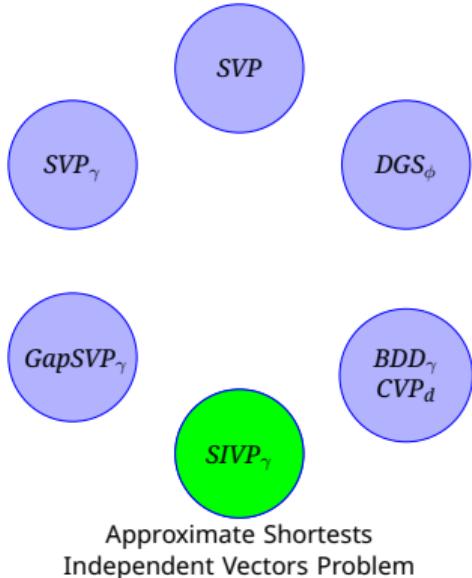
Lattice : what is hard to do?

Hard to reduce to SVP/SVP_{γ} : most reductions reduce to
 GapSVP or SIVP



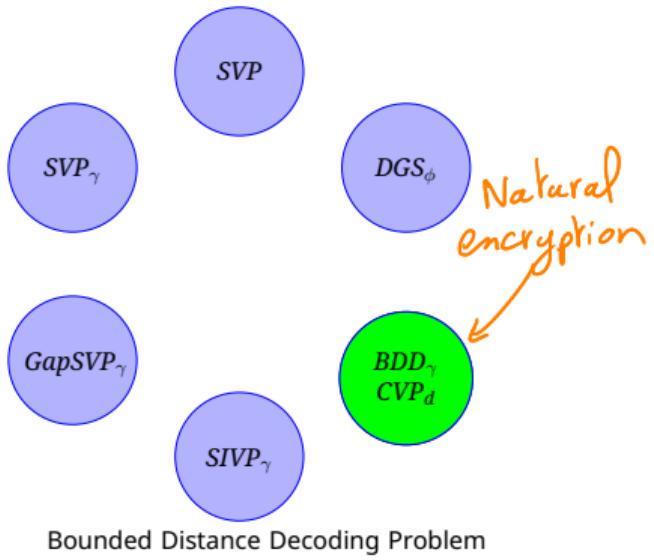
Goal: Given a basis B of a lattice \mathcal{L} , with the promise that $\lambda_1(\mathcal{L}) \leq 1$ or $\lambda_1(\mathcal{L}) > \gamma(n)$, determine which is the case.

Lattice : what is hard to do?

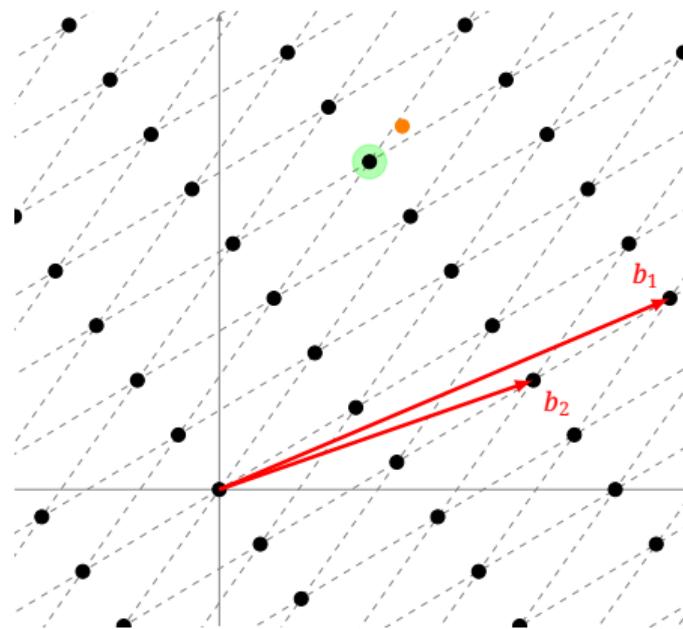


Goal: Given a basis B of a full-rank lattice \mathcal{L} , output a set $\{s_i\} \subset \mathcal{L}$ of n linearly independent lattice vectors where $\forall i, \|s_i\| \leq \gamma(n) \cdot \lambda_n(\mathcal{L})$.

Lattice : what is hard to do?

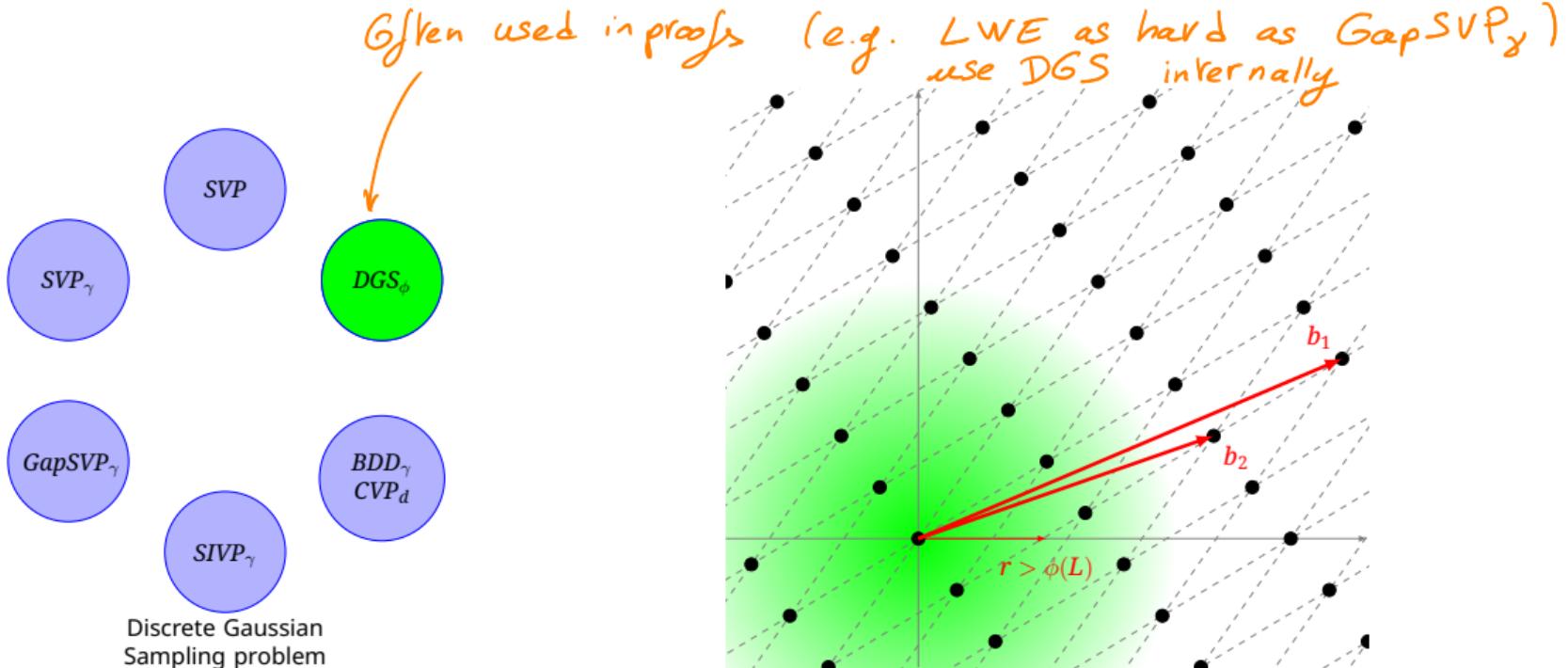


Bounded Distance Decoding Problem
and Closest Vector Problem



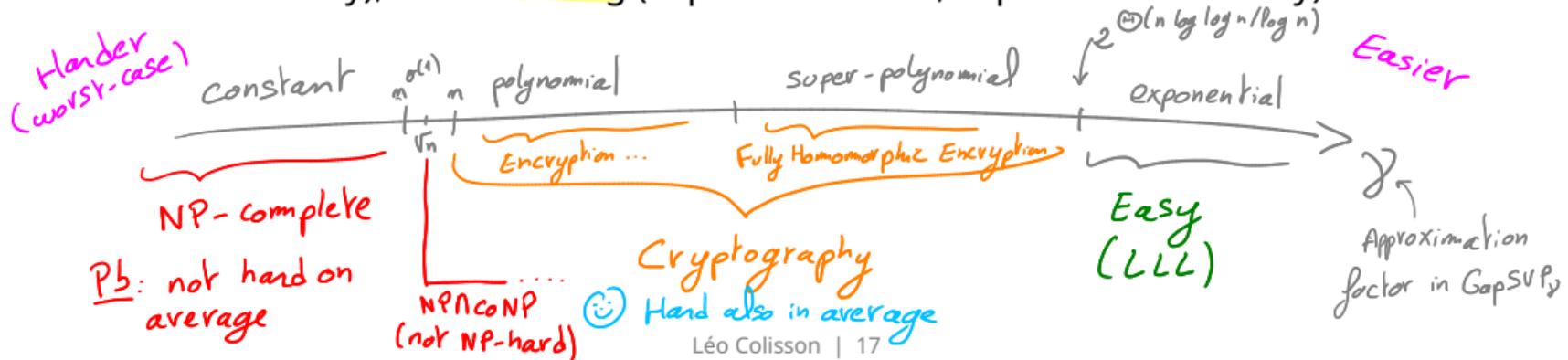
Goal: Given a basis B of a lattice \mathcal{L} and a target $t \in \mathbb{R}^n$ s.t. $\text{dist}(t, \mathcal{L}) < d := \lambda_1(\mathcal{L})/(2\gamma(n))$, find the unique v s.t. $\|t - v\| < d$.

Lattice : what is hard to do?



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/!](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 : CLÉ PRIVÉE

ROVERSIAIRE : CLÉ PUBLIQUE

Léo

LOGICIEL ESPION

CRYPTIS

00:00:00



10 5 -18 8 19 6 -51 -27 8 38

10 5 -18 8 19 6 -51 -27 8 38

Y

0

...

Y

0

...

Léo

LOGICIEL ESPION

CRYPTRIS

00:01:13



Message décrypté.

Échec

-1 -1 0 0 -1 1 0 0 0

7 4 ...

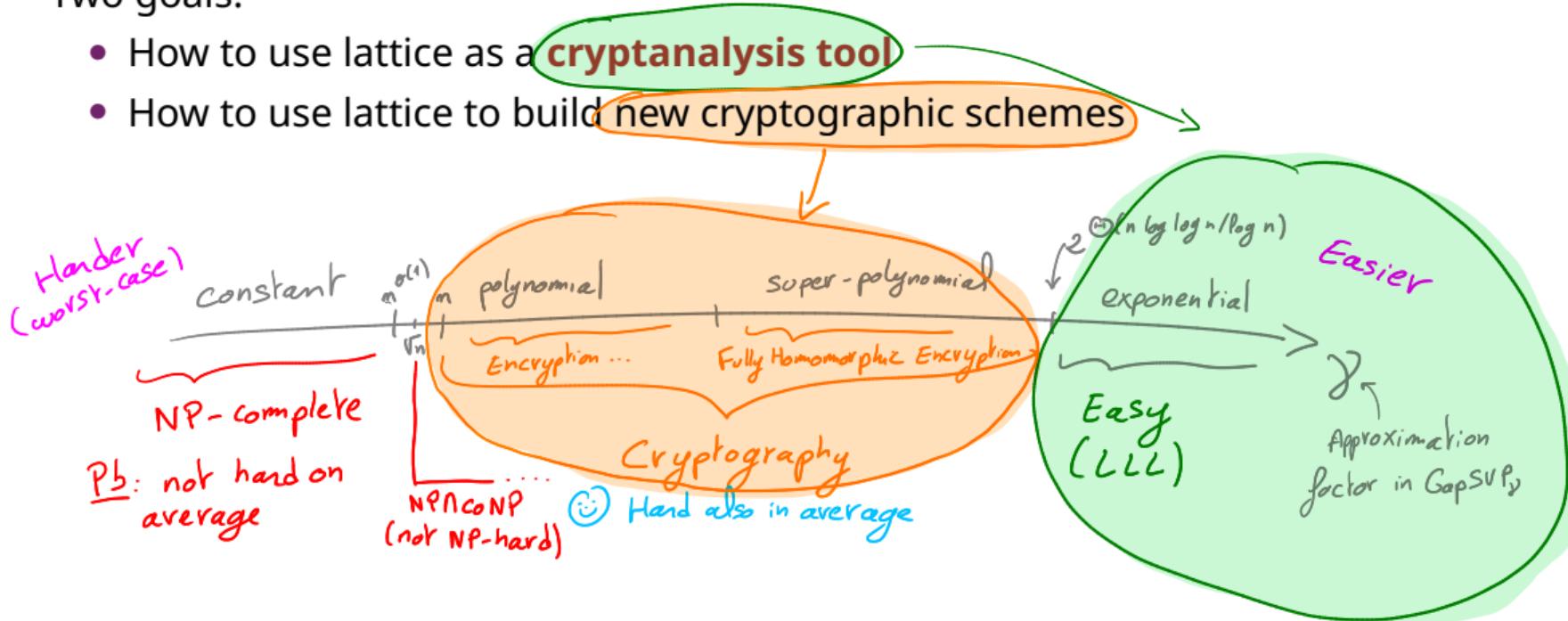
-8 16 3 -25 -41 2 32 16 -8 5

+ 1 ...

This course

Two goals:

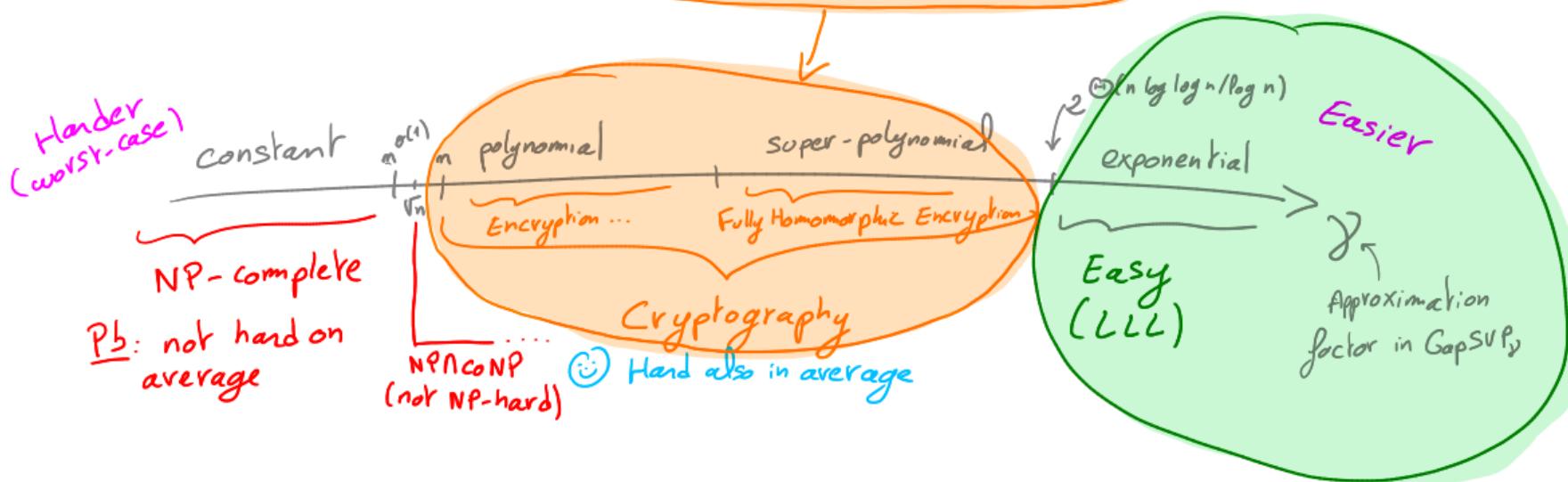
- How to use lattice as a **cryptanalysis tool**
- How to use lattice to build new cryptographic schemes



This course

Two goals:

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

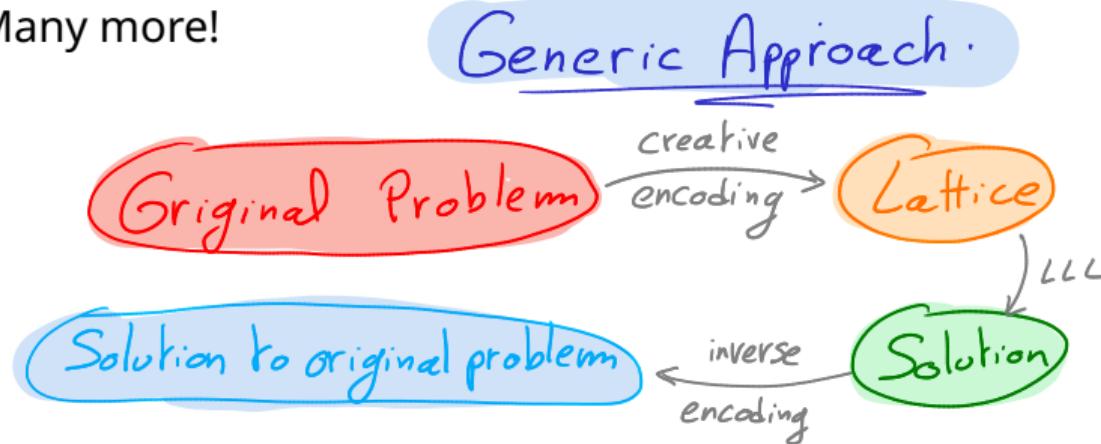


Cryptanalysis based on lattice

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!



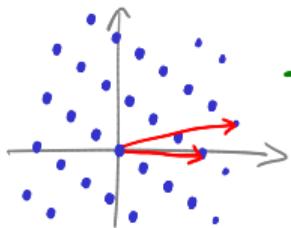
LLL

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

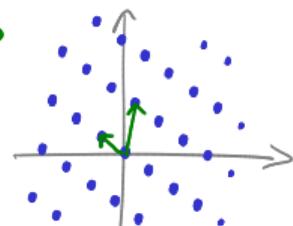
LLL

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

Bad basis



Better (\approx smaller
 \approx orthogonal) basis





Generic idea :

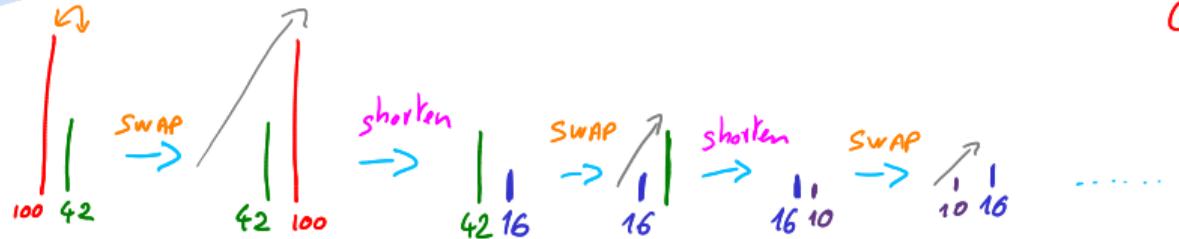
analogue of Euclid's algorithm to compute GCD

- integers \rightarrow vectors of integers
- Similar operations, \approx as efficient

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

$$\begin{aligned} \text{simplify} \\ \gcd(100, 42) &= \gcd(42, 100) = \gcd(42, 100 - \left\lfloor \frac{100}{42} \right\rfloor \times 42) \\ &\quad \xrightarrow{\text{SWAP: we want ordered list: } 42 < 100.} \\ &= \gcd(42, 16) = \gcd(16, 42) = \gcd(16, 42 - \left\lfloor \frac{42}{16} \right\rfloor \times 16) = \dots \end{aligned}$$

In picture:



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



\Rightarrow only 2 operations: SWAP and shorten until small enough

LLL

LLL algo

(param $\frac{1}{2} < \delta < 1$, e.g. $\frac{3}{4}$: time/quality trade-off)

SWAP and shorten until small enough

LLL

LLL algo

(param $\frac{1}{2} < \delta < 1$, e.g. $\exists \epsilon$: time/quality trade-off)
SWAP and shorten until small enough

$\rightarrow b - \left\lfloor \frac{b}{a} \right\rfloor a \rightsquigarrow$ Gram Schmidt

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} \mu_{ij} \vec{b}_j^*$$
$$\mu_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

LLL

LLL algo

(param $\frac{1}{2} < \delta < 1$, e.g. $\exists \epsilon$: time/quality trade-off)

SWAP and shorten until small enough

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

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

$$\vec{b}_i^* = \vec{b}_i - \sum_{j=1}^{i-1} p_{ij} \vec{b}_j^*$$
$$p_{ij} = \frac{\langle \vec{b}_i, \vec{b}_j^* \rangle}{\| \vec{b}_j^* \|}$$

LLL

LLL algo

(param $\frac{1}{2} < \delta < 1$, e.g. $\exists \gamma$: time/quality trade-off)

SWAP and shorten until small enough

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

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

$$b - \lfloor \frac{b}{a} \rfloor a \rightsquigarrow$$

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

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 $\frac{1}{2} < \delta < 1$, e.g. $\exists \gamma$: time/quality trade-off)

SWAP and shorten until small enough

if $a \times b \rightsquigarrow$ Lovasz condition:
 $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k, k-1}^c) \langle b_{k-1}^*, b_{k-1}^* \rangle$

$b - \lfloor \frac{b}{a} \rfloor a \rightsquigarrow$

"Sized-reduce": $\frac{\|v_{ij}\|}{\|v_{ij}\|} \leq \frac{1}{2}$

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

$k-1$ already sorted vectors

vector to shorten

Step 1:

shorten 1st non sorted vector



For $j = k-1 \rightarrow 1$:

$$b_k \leftarrow b_k - [\mu_{kj}] b_j$$

LLL

LLL algo

(param $\frac{1}{2} < \delta < 1$, e.g. $\exists \gamma$: time/quality trade-off)

SWAP and shorten until small enough

~~$a \times b \rightsquigarrow$~~ Lovasz condition:
 $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k, k-1}^c) \langle b_{k-1}^*, b_{k-1}^* \rangle$

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

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

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 (

$k-1$ already sorted vectors

Step 1: shorten 1st non sorted vector

For $j = k-1 \text{ to } 1:$
 $b_k \leftarrow b_k - [\mu_{kj}] b_j$

LLL

LLL algo

(param $\frac{1}{2} < \delta < 1$, e.g. $3\sqrt{d}$: time/quality trade-off)

SWAP and shorten until small enough

~~if $a \gg b \rightsquigarrow$~~ Lovasz condition:
 $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k,k-1}^c) \langle b_{k-1}^*, b_{k-1}^* \rangle$

~~$b = \lfloor \frac{b}{a} \rfloor a \rightsquigarrow$~~

"Sized-reduce": $\frac{\pi_{ij}}{\|b_j^*\|} \leq \frac{1}{2}$

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 () :

k already sorted vectors !

Step 1:

shorten 1st non sorted vector

For $j = k-1 \text{ to } 1$:
 $| b_k \leftarrow b_k - [\mu_{kj}] b_j |$

Step 2:

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

LLL algo(param $\frac{1}{2} < \delta < 1$, e.g. $3\sqrt{e}$: time/quality trade-off)

SWAP and shorten until small enough

~~$a \propto b \rightsquigarrow$~~ Lovasz condition:
 $\langle b_k^*, b_k^* \rangle > (\delta - \mu_{k, k-1}^c) \langle b_{k-1}^*, b_{k-1}^* \rangle$

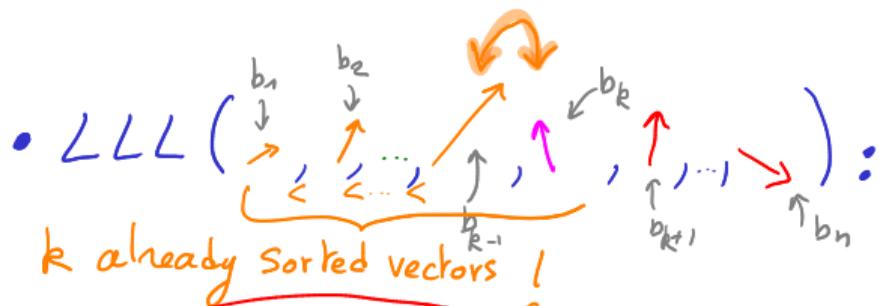
~~$b = \lfloor \frac{b}{a} \rfloor a$~~

"Sized-reduce": $\frac{\pi_{ij}}{\|b_j^*\|} \leq \frac{1}{2}$

Gram Schmidt

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

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



~~STOP when sorted + small enough (sized-reduce)~~

(3) (2) (1)

For $j = k-1 \text{ to } 1$:

$$b_k \leftarrow b_k - \lfloor \mu_{kj} \rfloor b_j$$

Step 1:

shorten 1st non sorted vector

Step 2:

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

LLL

Summary of

```
INPUT
    a lattice basis  $b_1, b_2, \dots, b_n$  in  $\mathbb{Z}^m$ 
    a parameter  $\delta$  with  $1/4 < \delta < 1$ , most commonly  $\delta = 3/4$ 

PROCEDURE
     $B^* \leftarrow \text{GramSchmidt}(\{b_1, \dots, b_n\}) = \{b_1^*, \dots, b_n^*\}$ ; and do not normalize
     $\mu_{l,j} \leftarrow \text{InnerProduct}(b_l, b_j^*) / \text{InnerProduct}(b_j^*, b_j^*)$ ; using the most current values of  $b_l$ 
    and  $b_j^*$ 
     $k \leftarrow 2$ ;
    while  $k \leq n$  do
        for  $j$  from  $k-1$  to  $1$  do
            if  $|\mu_{k,j}| > 1/2$  then
                 $b_k \leftarrow b_k - |\mu_{k,j}| b_j$ ;
                Update  $B^*$  and the related  $\mu_{l,j}$ 's as needed.
                (The naive method is to recompute  $B^*$  whenever  $b_i$  changes:
                 $B^* \leftarrow \text{GramSchmidt}(\{b_1, \dots, b_n\}) = \{b_1^*, \dots, b_n^*\}$ )
            end if
        end for
        if  $\text{InnerProduct}(b_k^*, b_k^*) > (\delta - \mu_{k,k-1}^2) \text{InnerProduct}(b_{k-1}^*, b_{k-1}^*)$  then
             $k \leftarrow k + 1$ ;
        else
            Swap  $b_k$  and  $b_{k-1}$ ;
            Update  $B^*$  and the related  $\mu_{l,j}$ 's as needed.
             $k \leftarrow \max(k-1, 2)$ ;
        end if
    end while
    return  $B$  the LLL reduced basis of  $\{b_1, \dots, b_n\}$ 

OUTPUT
    the reduced basis  $b_1, b_2, \dots, b_n$  in  $\mathbb{Z}^m$ 
```

Theorem (LLL)

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

- 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

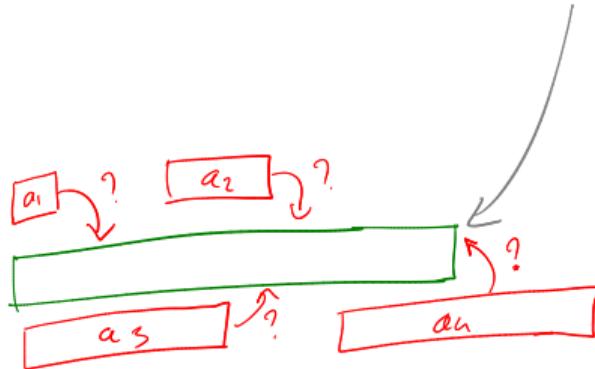
Contexte

Merkle-Hellman:

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

Merkle-Hellman

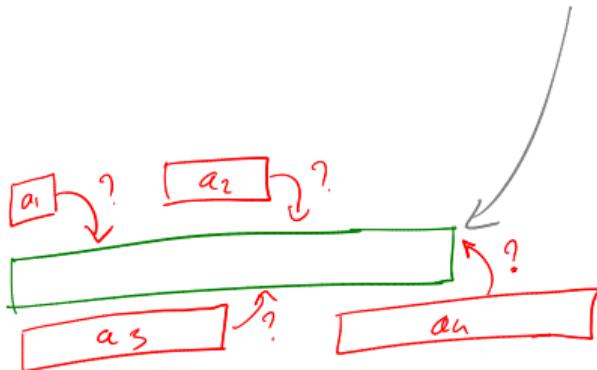
Based on knapsack problem + trapdoor



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

Merkle-Hellman

Based on knapsack problem + trapdoor



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

Key generation:

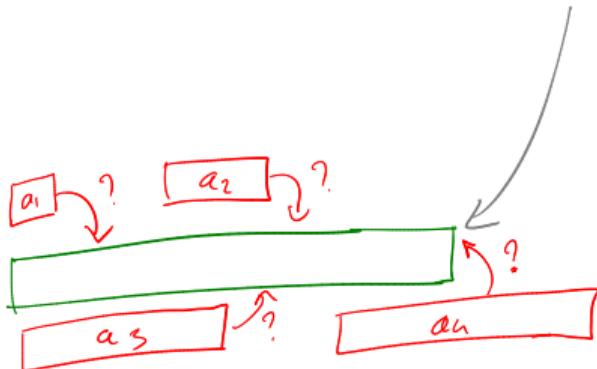
- super-increasing sequence $\{a_1, \dots, a_n\}$
(i.e. $\forall i, a_i > \sum_{j < i} a_j$)
- Let $N > \sum_i a_i$ and $A < N$, $\gcd(A, N) = 1$
- Public key: $\text{pk} := \{b_i := Aa_i \pmod{N}\}$,
private key: $\text{sk} := (N, A, \{a_i\}_i)$

Encryption: message = subset of elements
 $\text{Enc}_{\text{pk}}(m := (m_1, \dots, m_n)) = \sum_i m_i b_i$ to take

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

Merkle-Hellman

Based on knapsack problem + trapdoor



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

Key generation:

- super-increasing sequence $\{a_1, \dots, a_n\}$
(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: $\text{pk} := \{b_i := Aa_i \pmod{N}\}$,
private key: $\text{sk} := (N, A, \{a_i\}_i)$

Encryption: message = subset of elements
 $\text{Enc}_{\text{pk}}(m := (m_1, \dots, m_n)) = \sum_i m_i b_i$ to take

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

Merkle-Hellman



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

- A 16
- B 28
- C 44



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

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

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

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

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

Merkle-Hellman attack

Attack against Merkle-Hellman:

- ① **run LLL on \mathbf{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

The screenshot shows a Jupyter Notebook interface with the title "knapsack_attack.ipynb". The code cell [6] contains Python code for generating a knapsack problem and encoding a message. The code defines two functions: `gen_knapsack` and `enc`. The `gen_knapsack` function generates a public key `pk` consisting of a list of weights [10, 3, 16, 15]. The `enc` function encodes a message `m` using the public key `pk`. The code cell [11] shows the execution of `gen_knapsack(4)`, resulting in `pk` being [10, 3, 16, 15]. The code cell [12] shows the execution of `enc(pk, [1, 1, 0, 1])`, resulting in 28. The code cell [18] shows the construction of a matrix `B` from the public key `pk` and its transpose, followed by an LLL reduction. The resulting matrix is:

```
[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  X +  
File Edit Insert Cell Kernel Help  
Not  
[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  
  
[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!  
m = 1101
```

Cryptanalysis via LLL: conclusion

Take home message:

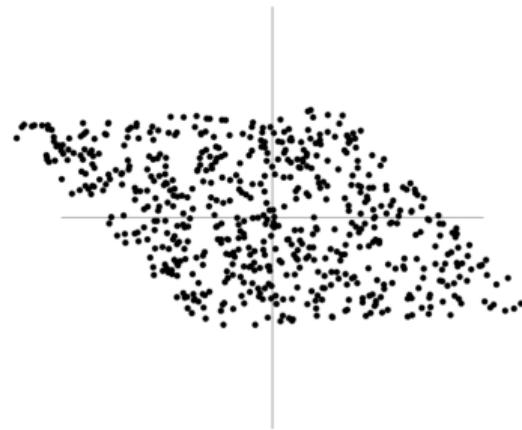
**LLL reductions = very powerful tool to
attack cryptosystems (and more!)**

Lattice-based cryptography

Lattice-based cryptography: historical works

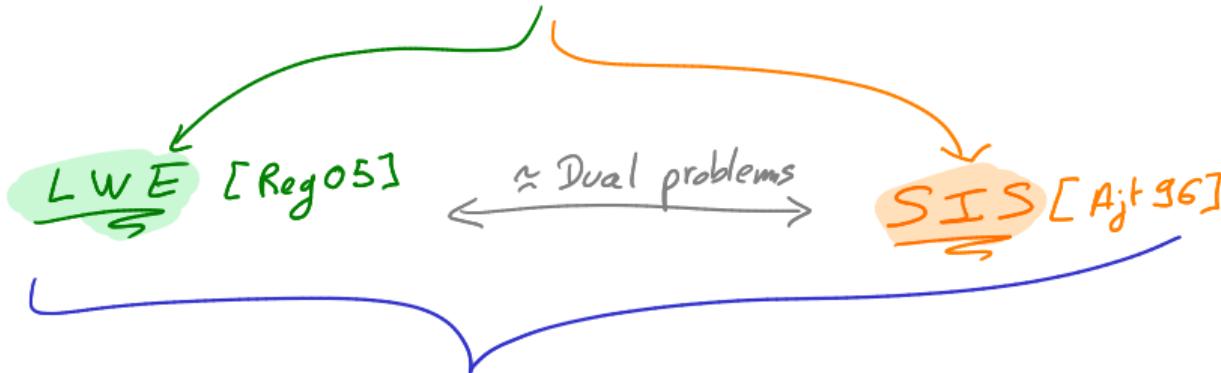
Lattice also used to **build new primitives**... but we don't use them directly anymore:

- 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



"If we can break LWE/SIS, we can
break some lattice-related problems believed to be hard"

Lattice-based cryptography: SIS

Definition

SIS n, q, β, m [Ajt96]

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

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

Find $z \in \mathbb{Z}_q^m \setminus \{[0]\}$

such that

- $\|z\| \leq \beta$
- $Az = [0]$

Lattice-based cryptography: SIS

Definition

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

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

Find $z \in \mathbb{Z}_q^m \setminus \{[0]\}$

such that

- $\|z\| \leq \beta$
- $Az = [0]$

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

Lattice-based cryptography: SIS

Definition

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

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

Find $z \in \mathbb{Z}_q^m \setminus \{[0]\}$

such that

- $\|z\| \leq \beta$
- $Az = [0]$

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

→ Otherwise easy to solve
(Gaussian elimination)

Lattice-based cryptography: LWE

LWE

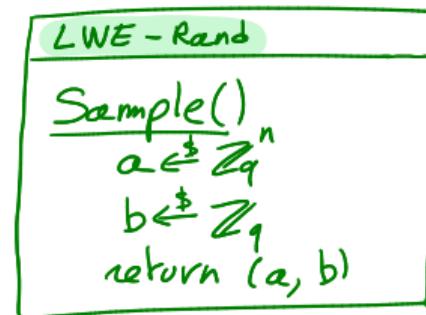
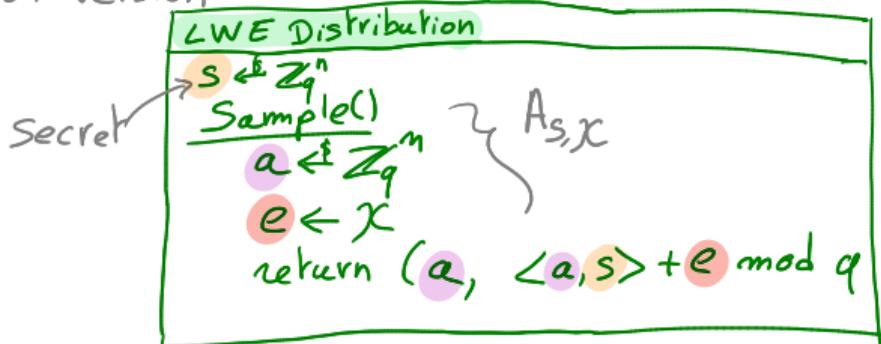
Modulus EN
 q, m, x

[Reg05]

"Error" distribution to \mathbb{Z}_q

Goal: distinguish these two distributions:

decision version



= Indistinguishable

Search version:



Lattice-based cryptography: LWE

LWE

[Reg05]

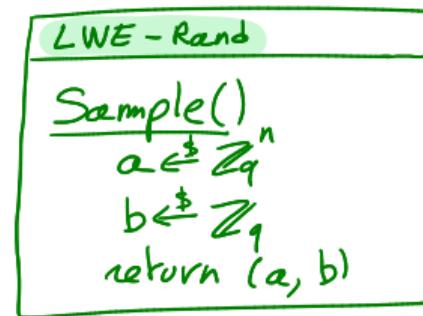
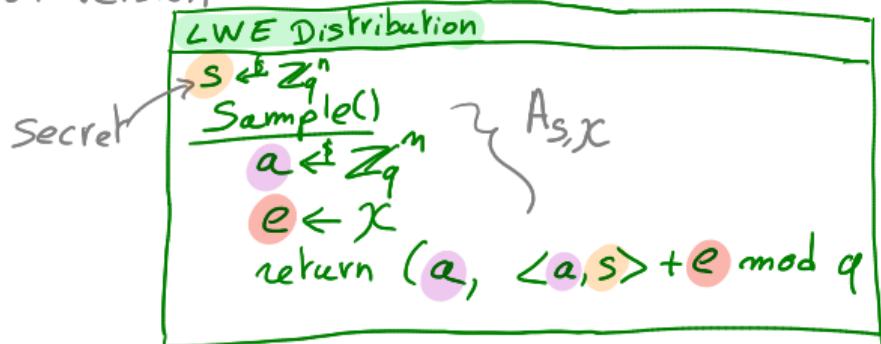
Modulus EN

dimension EN

"Error" distribution to \mathbb{Z}_q

Goal: distinguish these two distributions:

decision version



= Indistinguishable

Search version: find s from samples from $A_{S,X}$.

Lattice-based cryptography: LWE

LWE

Modulus EN
 q, m, x

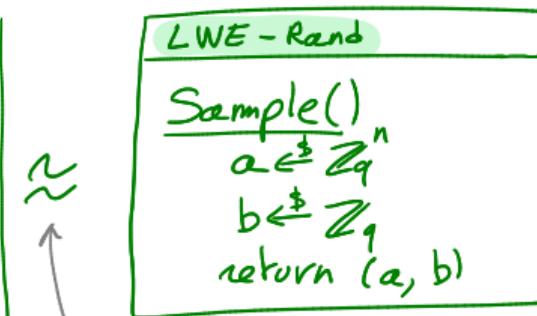
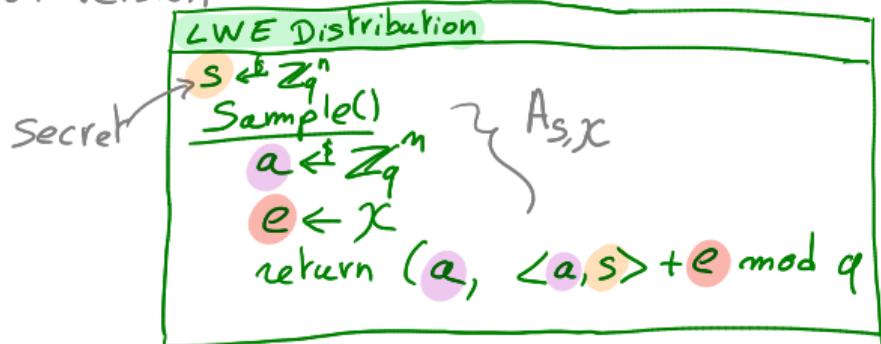
[Reg05]

"Error" distribution to \mathbb{Z}_q Kyber

Focus of this course (basis of Kyber)

Goal: distinguish these two distributions:

decision version



= Indistinguishable

Search version: find s from samples from $A_{S,X}$.

Lattice-based cryptography: LWE

LWE
≡

LWE

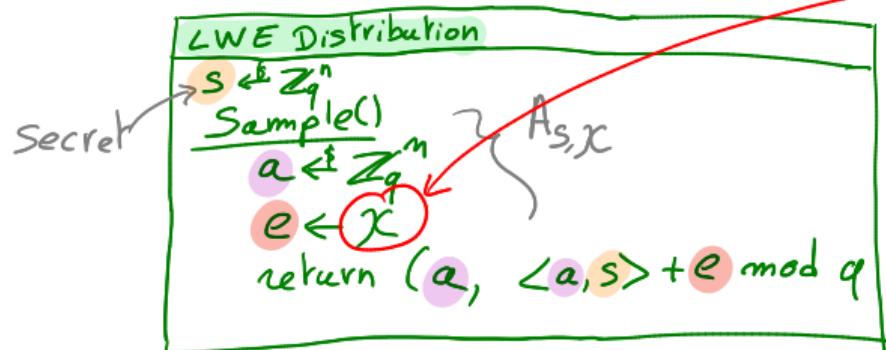
Modulus EN
 $\rightarrow q, m$

[Reg05]

dimension EN

Focus of this course (basis of Kyber)

"Error" distribution to \mathbb{Z}_q



How to choose error X ?

- If X outputs 0: easy

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

Lattice-based cryptography: LWE

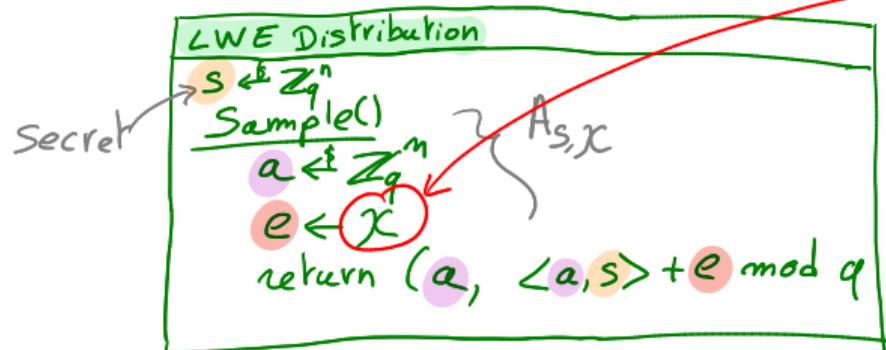
LWE
≡

LWE [Reg05]

Modulus EN dimension EN

"Error" distribution to \mathbb{Z}_q

Focus of this course (basis of Kyber)



How to choose error X ?

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



Search version: find s from samples from As, X .

Lattice-based cryptography: LWE

Definition (Discrete Gaussian)

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

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

LWE: matrix form

If we restrict the number of samples¹ to m , we can define the LWE in a matrix form:

Definition (Decision $\text{LWE}_{n,q,\chi,m}$, 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) .

¹This 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_γ (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 $\text{LWE}_{n,q,\chi,m}$ problem is at least as hard as quantumly^a solving GapSVP_γ and SIVP_γ on arbitrary n -dimensional lattices, for $\gamma = \tilde{O}(n/\alpha)$.

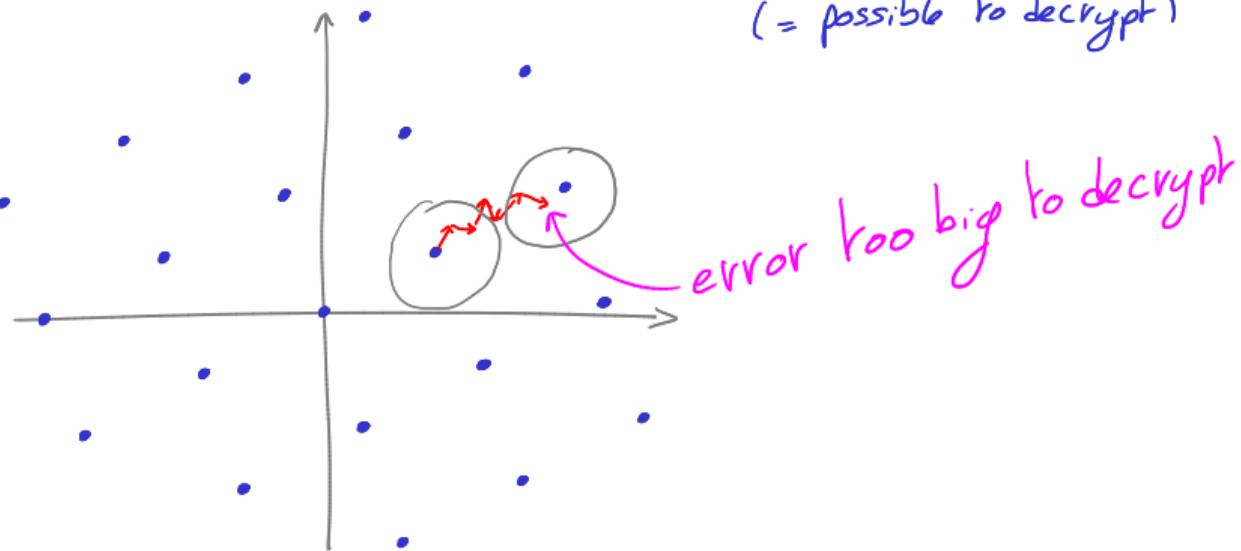
^aSee also [Pei09] for a classical reduction.

NB: The best polynomial algorithm to solve GapSVP_γ work only for $\gamma = 2^{\Theta(n \log \log n / \log n)}$, so GapSVP_γ 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" → Usefull as summing the errors
keep the error small
(= possible to decrypt)



Encryption from LWE

Regev's encryption

Reminder LWE:

$$(A', As + e)$$

The diagram shows two rectangles. The left rectangle is a square with its side length labeled m . The right rectangle has its width labeled m and its height labeled n . Both rectangles have double-headed arrows indicating their respective dimensions.

Regev's encryption

$$(A^{\$}, As + e + \begin{bmatrix} 0 \\ \vdots \\ 0 \\ m \begin{bmatrix} q \\ 2 \end{bmatrix} \end{bmatrix})$$

Diagram illustrating the components:

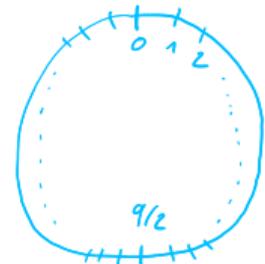
- A large square labeled $m \times m$ representing the matrix A .
- A tall vertical rectangle labeled $m \times 1$ representing the vector e .
- A small square at the bottom right labeled $+ m \begin{bmatrix} q \\ 2 \end{bmatrix}$ representing the additional term.

Regev's encryption

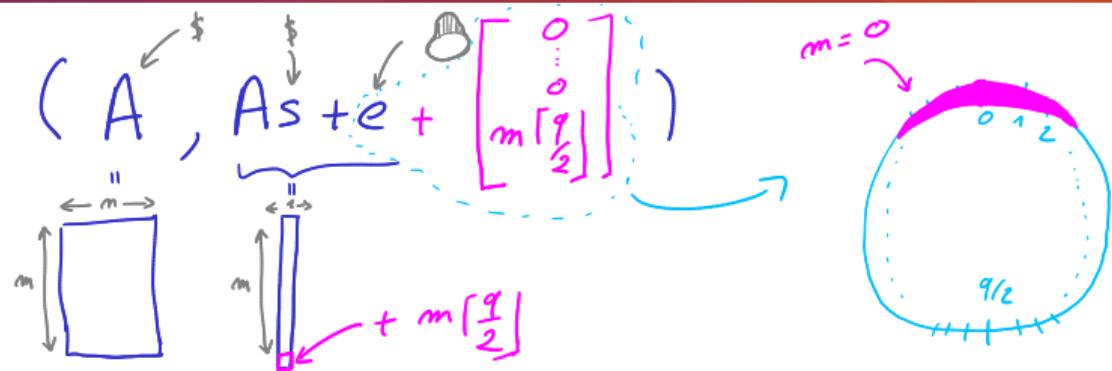
$$(A^{\$}, As + e + \begin{bmatrix} 0 \\ 0 \\ m \begin{bmatrix} q \\ 2 \end{bmatrix} \end{bmatrix})$$

Diagram illustrating the components of Regev's encryption:

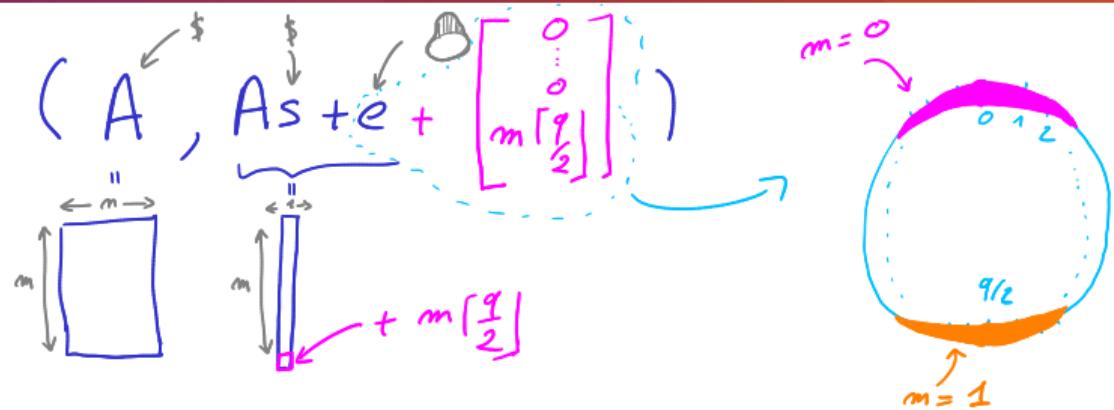
- A large square labeled m represents the public key matrix A .
- A smaller vertical rectangle labeled m represents the secret key matrix s .
- A pink bracket indicates the addition of e (noise) to the sum of As and e .
- A pink bracket indicates the addition of $m \begin{bmatrix} q \\ 2 \end{bmatrix}$ to the result.



Regev's encryption



Regev's encryption



Regev's encryption

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

Diagram illustrating the components of the encryption tuple:

- A green square labeled m represents the message space.
- A green circle labeled A represents the public key matrix.
- A purple circle labeled s represents a secret key vector.
- A blue circle labeled e represents an error vector.
- A pink bracketed expression $+ m \left[\frac{q}{2} \right]$ indicates the addition of a specific value to the message component.
- A pink bracketed expression $\begin{bmatrix} 0 \\ \vdots \\ 0 \\ m \left[\frac{q}{2} \right] \end{bmatrix}$ represents the final vector added to the product As .
- A pink circle labeled $m=0$ shows the result of the encryption for $m=0$.
- A pink circle labeled $m=1$ shows the result of the encryption for $m=1$, with a pink shaded region indicating the range of possible values.

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



Regev's encryption

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

Diagram illustrating the components of the encryption tuple:

- A green square matrix A of size $m \times m$.
- A green vector s of length m .
- A red vector e of length m .
- A pink vector $\begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix}$ representing a message m encoded as a coordinate vector.
- The resulting vector $As + e + \begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix}$.

On the right, a circle represents the domain of the function. The horizontal axis is labeled $\frac{q}{2}$. The upper half-plane is divided into two regions:

- The left region is labeled $m=0$.
- The right region is labeled $m=1$.

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

$$\rightarrow b - As = e + \begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix} \rightsquigarrow \begin{cases} \text{if norm last coordinate} < \frac{q}{2} \\ \quad m=0 \\ \text{else} \\ \quad m=1 \end{cases}$$

Regev's encryption

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

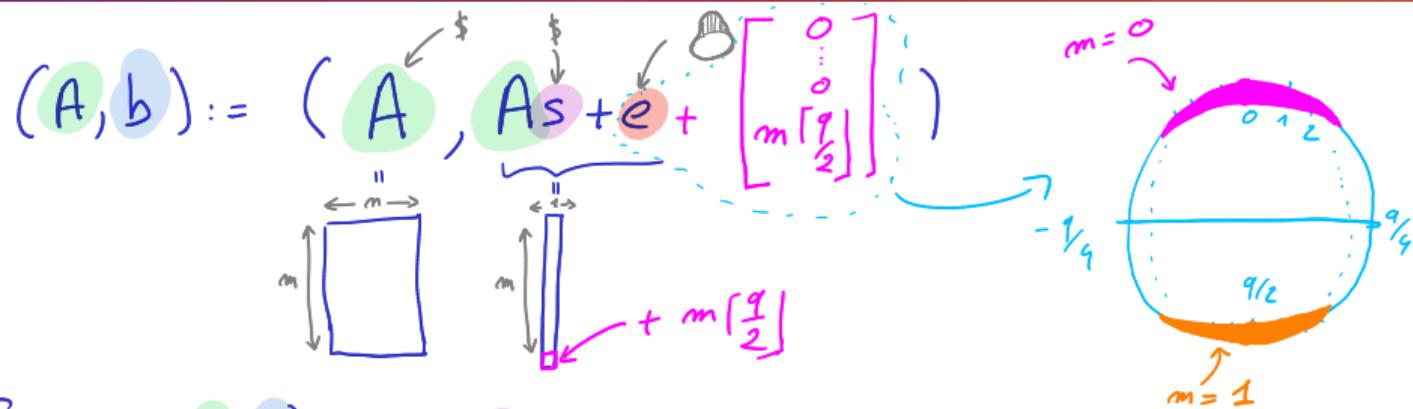
Diagram illustrating the construction of the encryption tuple (A, b) :

- A green square matrix A is shown with dimension $m \times m$.
- A green vector s is shown with dimension m .
- A red vector e is shown with dimension m .
- A pink matrix $\begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix}$ is shown.
- The resulting vector $As + e + \begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix}$ is labeled b .
- To the right, a circle represents the domain of the function. It is divided into two regions:
 - The upper-left region is shaded pink and labeled $m=0$.
 - The lower-right region is shaded orange and labeled $m=1$.

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

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

Regev's encryption

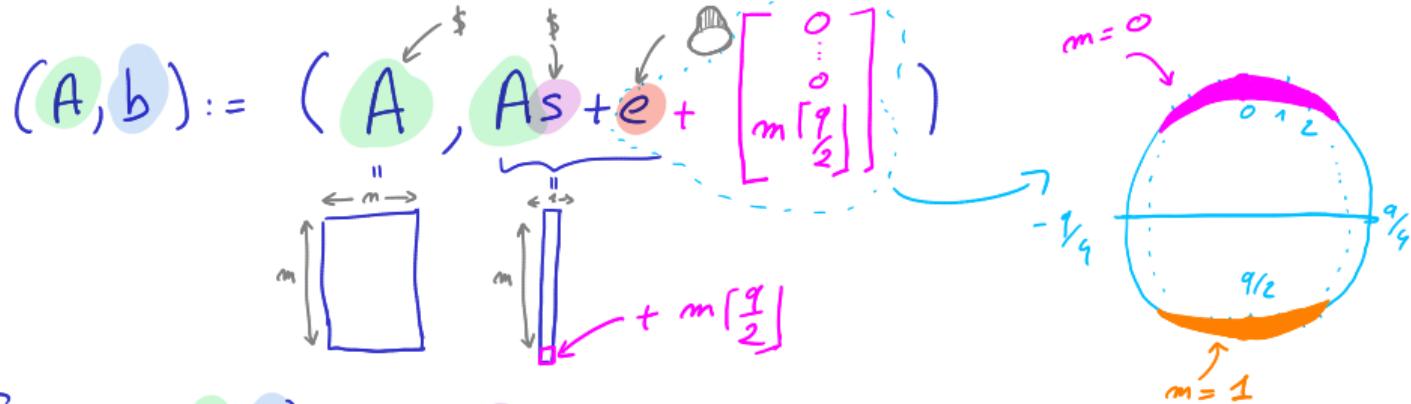


Claim 1: Given (A, b) and s , I can recover m

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

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

Regev's encryption



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 $(A, As + e + \begin{bmatrix} 0 \\ \vdots \\ m\left[\frac{q}{2}\right] \end{bmatrix}) \approx (A, b + cte)$

Regev's encryption

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

Diagram illustrating the construction:

- A green square labeled m represents the message m .
- A blue vertical rectangle labeled m represents the message m .
- The term $+ m \left[\frac{q}{2} \right]$ is shown below the blue rectangle.
- The term $\begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix}$ is shown next to the blue rectangle.
- The term $As + e$ is shown next to the blue rectangle.
- The term A is shown next to the green square.
- The final expression is enclosed in parentheses: $(A, As + e + \begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix})$.
- To the right, a circle is divided into two halves by a horizontal axis. The top half is shaded pink and labeled $m=0$. The bottom half is shaded orange and labeled $m=1$. The center of the circle is labeled $\frac{q}{2}$.

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

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

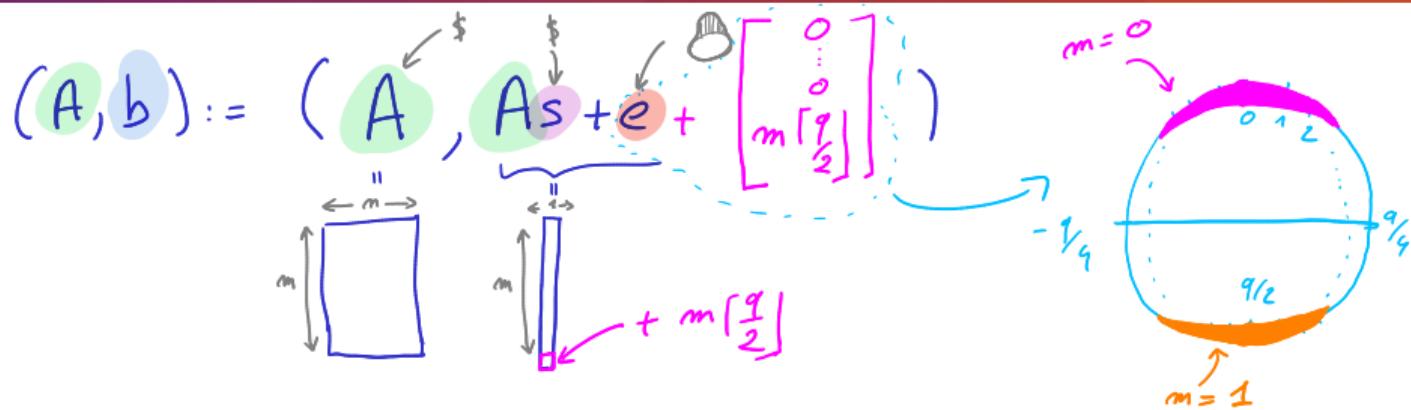
→ Decision-LWE: $(A, As + e) \approx (A, b)$

so $(A, As + e + \begin{bmatrix} 0 \\ \vdots \\ m \left[\frac{q}{2} \right] \end{bmatrix}) \approx (A, b + cte) \approx (A, b)$

Uniform distribution
+ cte = uniform dist.

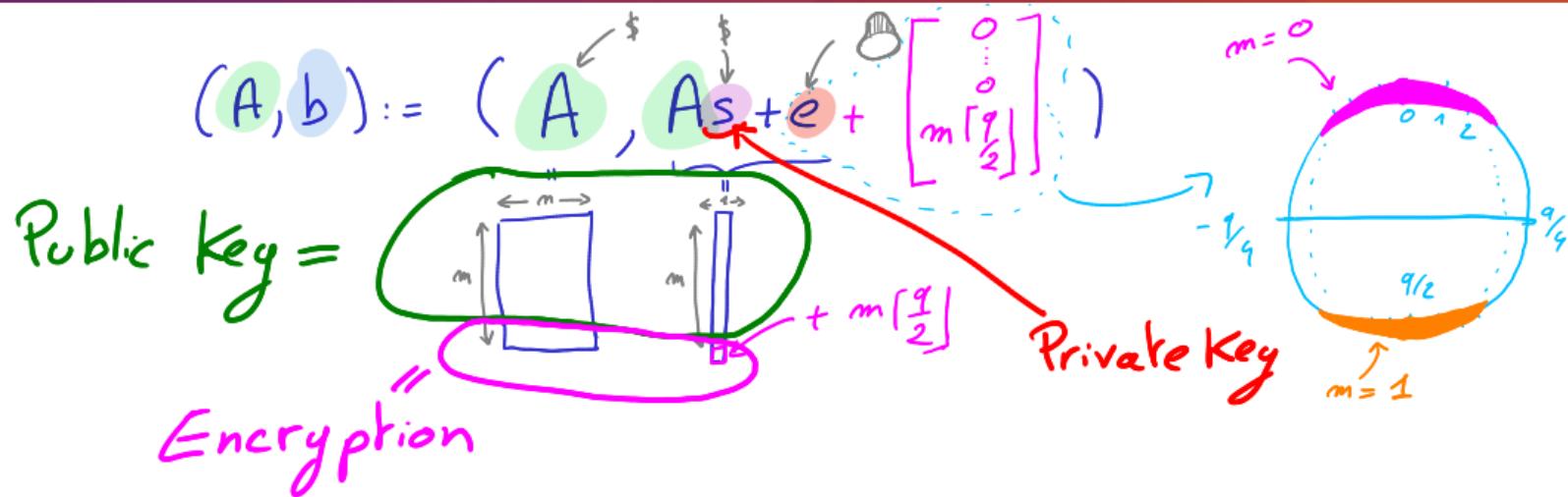
No info on m

Regev's encryption

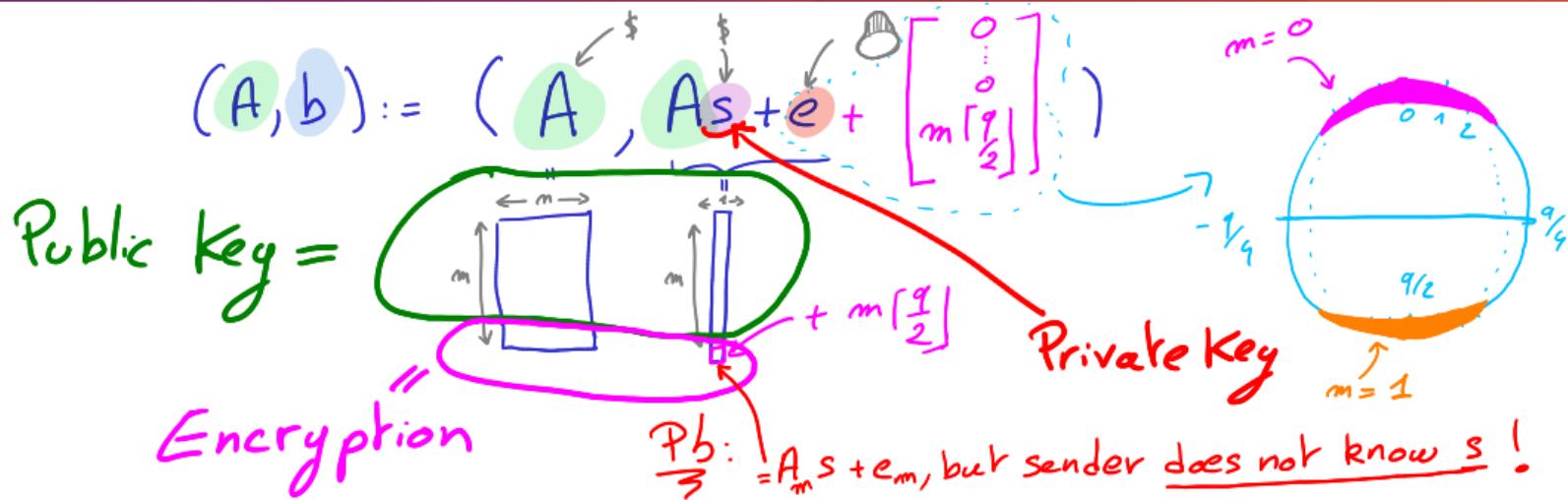


How to turn it into an encryption?

Regev's encryption

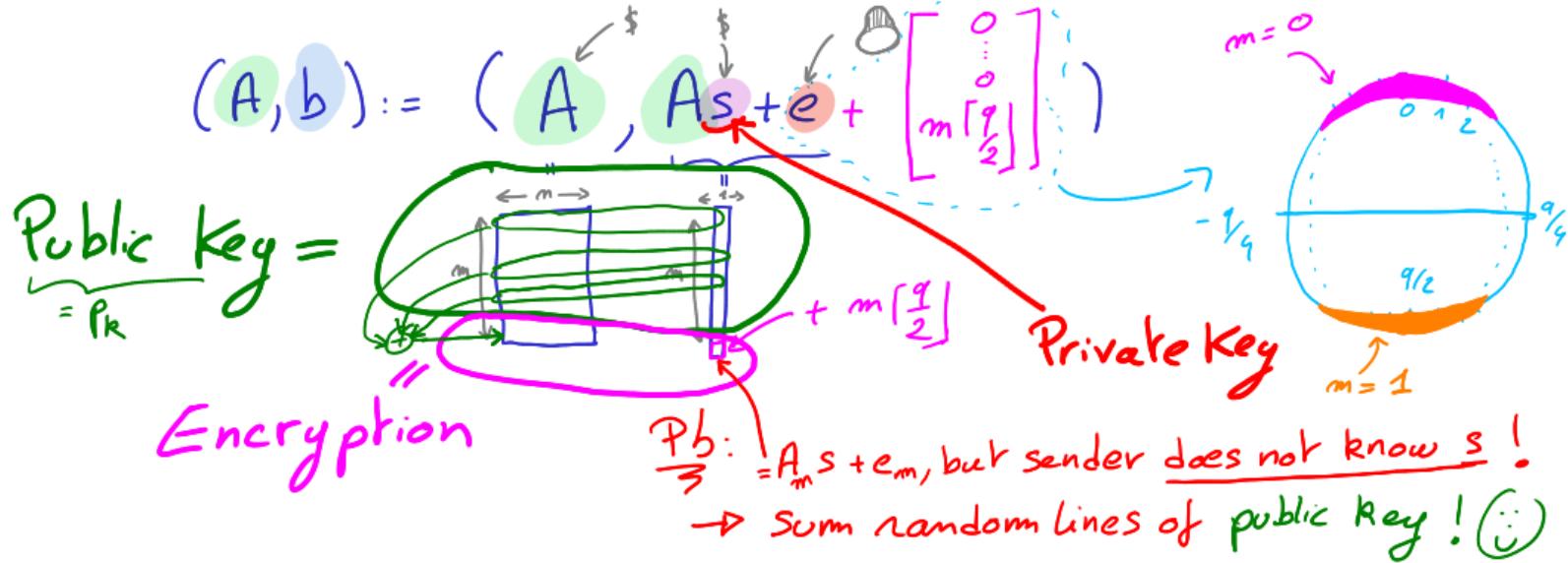


Regev's encryption



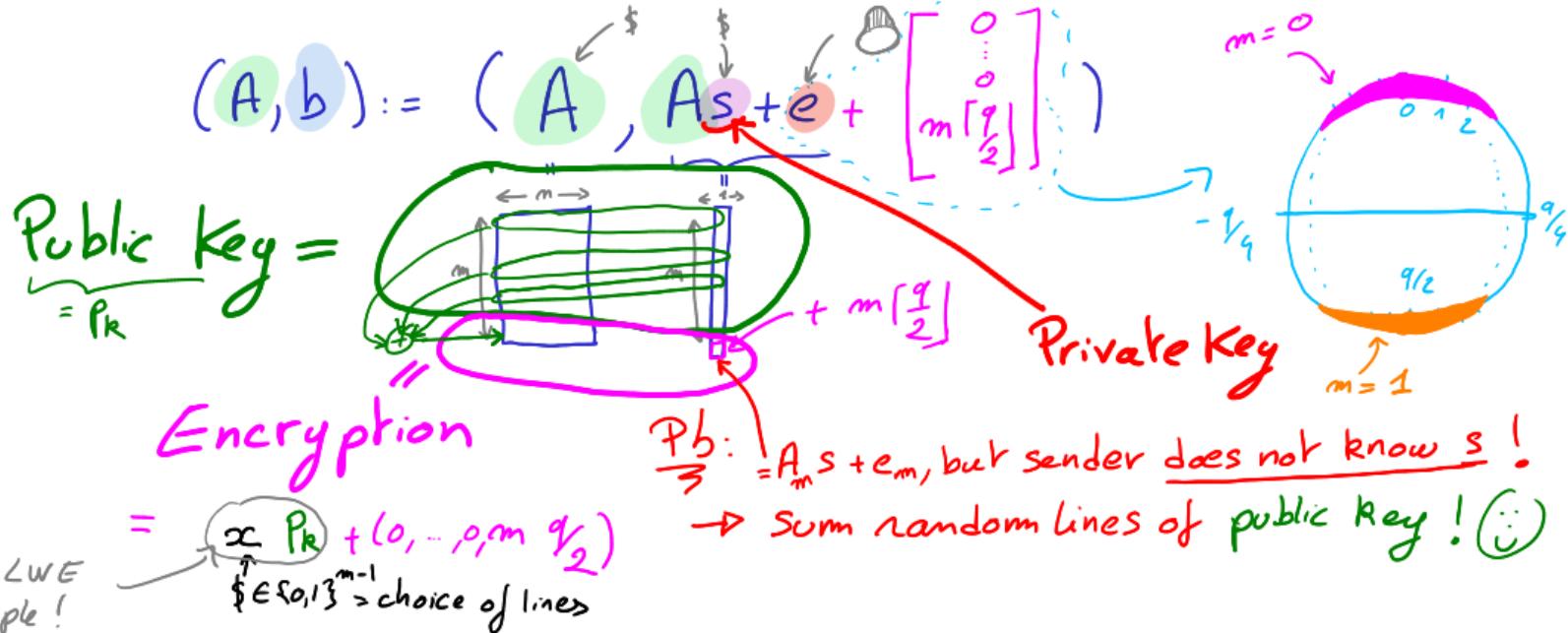
How to turn it into an encryption?

Regev's encryption



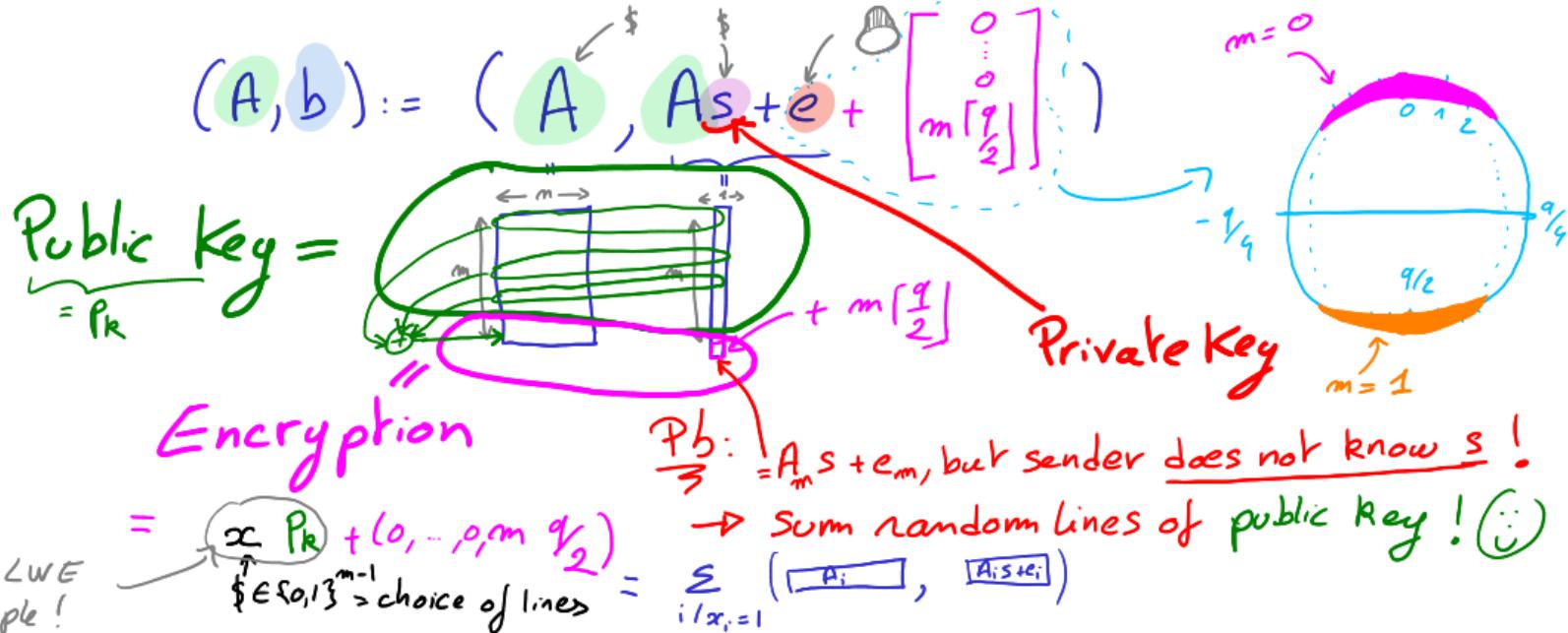
How to turn it into an encryption?

Regev's encryption



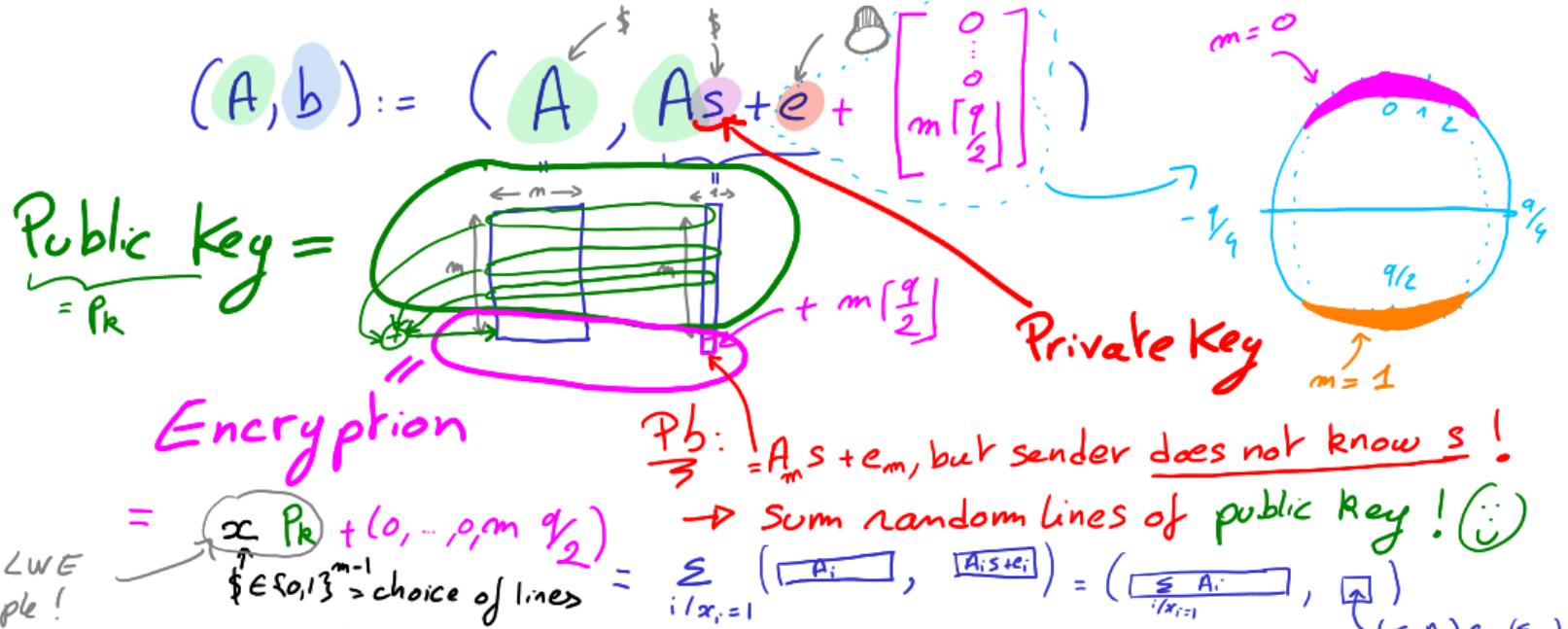
How to turn it into an encryption?

Regev's encryption



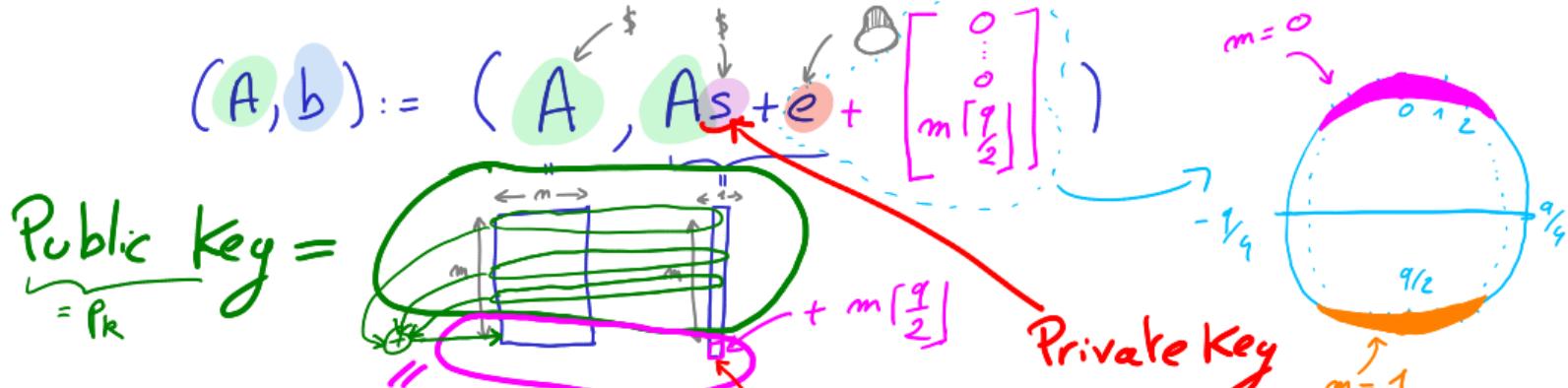
How to turn it into an encryption?

Regev's encryption



How to turn it into an encryption?

Regev's encryption



Encryption

$\frac{Pb}{\exists} = As + e_m$, but sender does not know s!

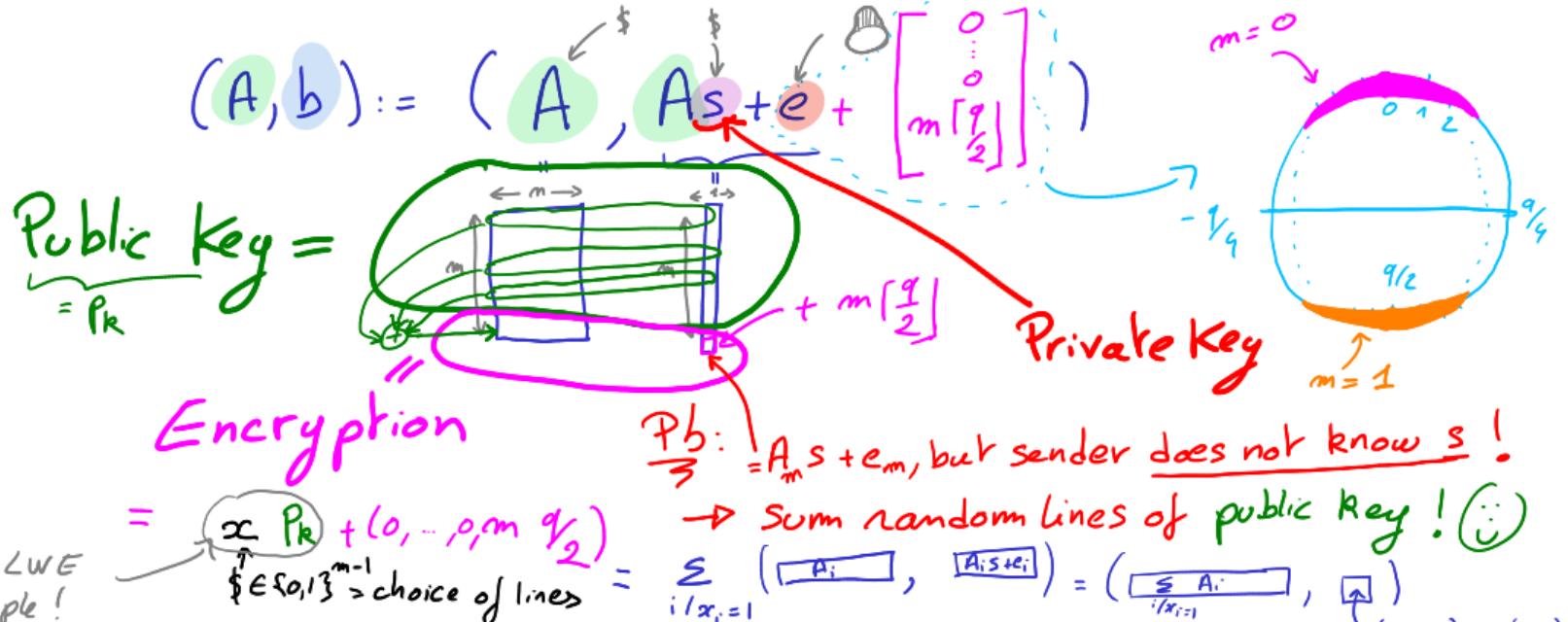
$= \sum_{x_i=1}^{\infty} P_k + (0, \dots, 0, m \begin{bmatrix} q \\ 2 \end{bmatrix}) \rightarrow \text{Sum random lines of public key!} \quad (\text{choice of lines})$

New LWE Sample!

How to turn it into an encryption?

$(\sum_{i/x_i=1} A_i)s + (\sum_{i/x_i=1} e_i)$
error ↴
a bit bigger

Regev's encryption



How to turn it into an encryption?

\Rightarrow To have $\sum_i e_i < q_{\ell_q}$, we must have α small enough

a bit bigger

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

GEN(1^λ):

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

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

$$e \leftarrow \chi^m$$

return $(p_k := (A, As + e), s_k := s)$

ENC($p_k := (A, b)$, $m \in \{0, 1\}$):

$$x \xleftarrow{\$} \{0, 1\}^m$$

return $(x^T A, x^T b + m \lceil q/2 \rceil)$

DEC($s_k := s$, (a, y)):

if $y - as \bmod q < q/4$ or $y - as \bmod q > 3q/4$: **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 $m = O(\log \lambda)$?

→ If $m = O(\log \lambda)$, a malicious eavesdropper 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, $m = \frac{y - x^T b}{\lceil q/2 \rceil}$.

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 $\text{LWE}_{n,q,\chi,m}$ is hard and if

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

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

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

^aNo 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:

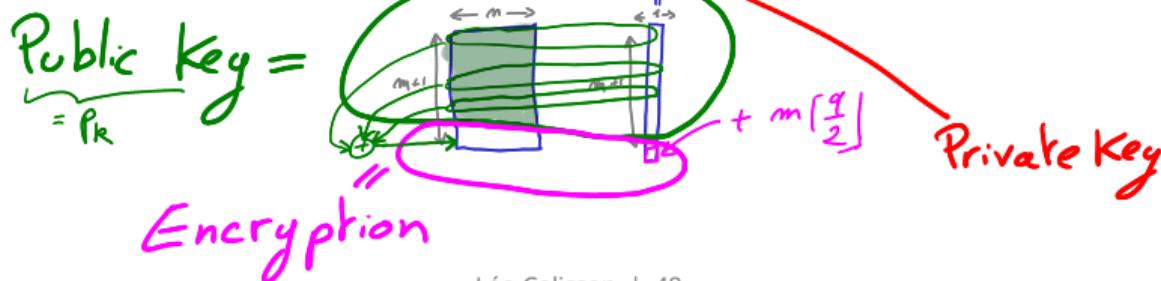
Step 1: LWE is secure:

$$(A, As + e) \approx (A, b)$$

$\begin{matrix} s \\ \$ \\ \$ \\ r \\ b \end{matrix}$ $\leftarrow f \in \mathbb{Z}_q^m$

= random

$$\text{Enc}_{P_k}(m) := (x^T A, x^T (As + e) + m \begin{bmatrix} q \\ 2 \end{bmatrix})$$



Regev's encryption: security

Proof idea:

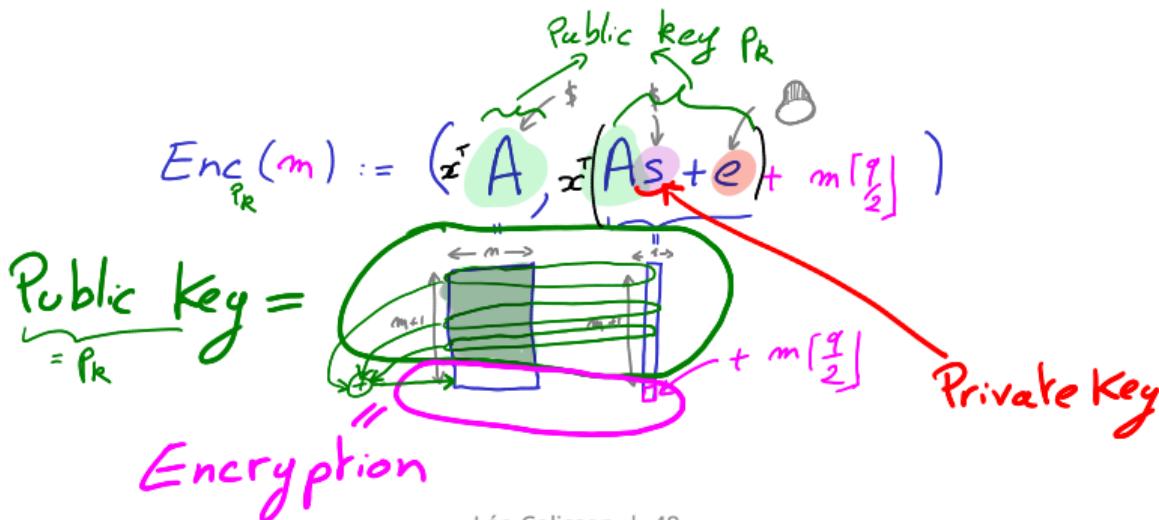
Step 1:

LWE is secure: $(A, As + e) \approx (A, b)$

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

$$\begin{array}{c} \$ \\ \downarrow \\ s \\ \$ \\ \downarrow \\ r \\ \downarrow \\ b \\ \leftarrow f \in \mathbb{Z}_q^m \end{array}$$

$(no \text{ klapdor!})$



Regev's encryption: security

Proof idea:

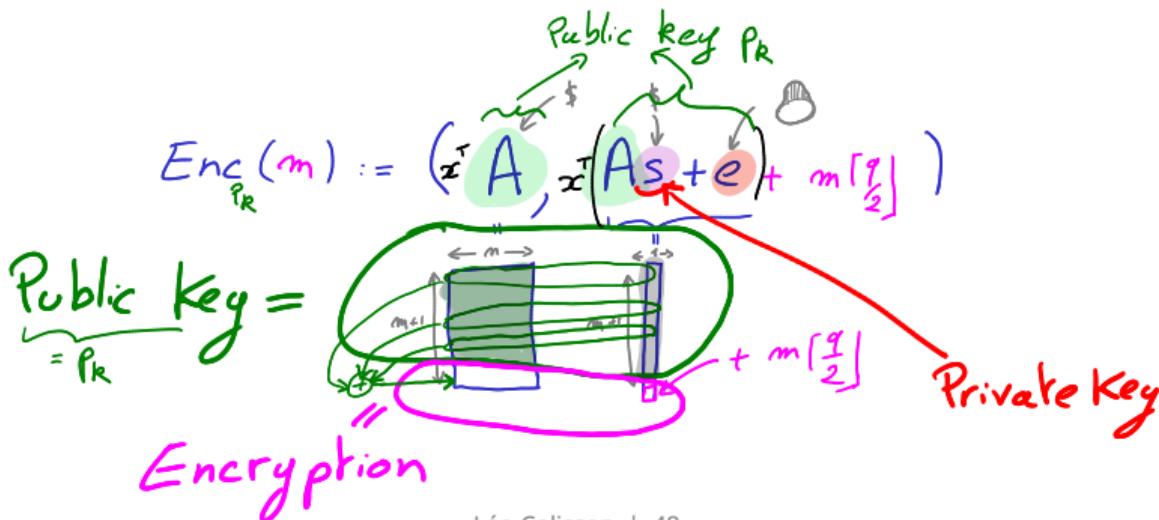
Step 1:

LWE is secure: $(A, As + e) \approx (A, b)$

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

$$\begin{array}{c} \$ \\ \downarrow \\ s \\ \$ \\ \downarrow \\ r \\ \downarrow \\ b \\ \leftarrow f \in \mathbb{Z}_q^m \end{array}$$

$(no \text{ klapdor!})$



Regev's encryption: security

Proof idea:

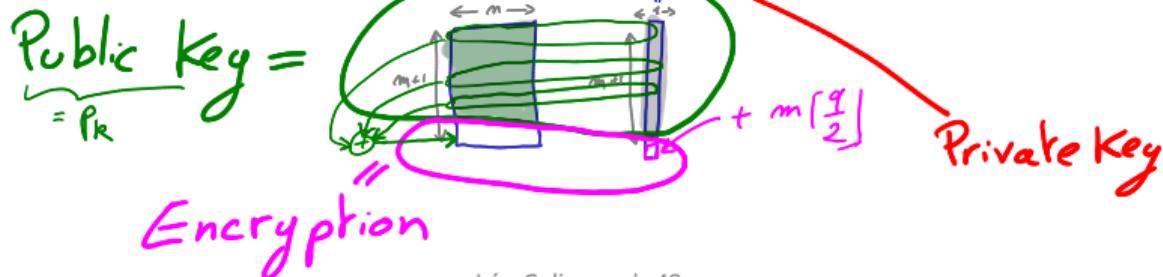
Step 1: LWE is secure: $(A, As + e) \approx (A, b)$ $\leftarrow b \in \mathbb{Z}_q^m$ (no trapdoor!)

→ we can replace the public key $(A, As + e)$ with (A, b)

Step 2: (A, b) is lossy: i.e. $(x^T A, x^T b) \approx (\alpha^T, y^T)$ if $m \geq (m+1)q$ by leftover hash lemma.

= random

$$\text{Enc}_{P_k}(m) := (x^T A, x^T (As + e) + m \begin{bmatrix} q \\ 2 \end{bmatrix})$$



Regev's encryption: security

Proof idea:

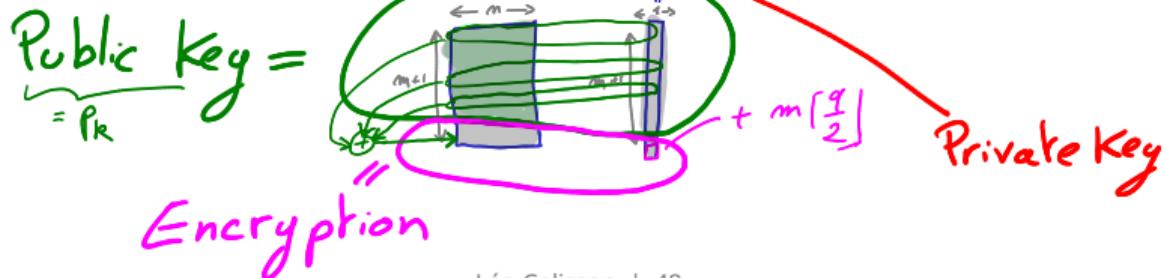
Step 1: LWE is secure: $(A, As + e) \approx (A, b)$ $\xrightarrow{b \in \mathbb{Z}_q^m}$ (no trapdoor!)

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

Step 2: (A, b) is lossy: i.e. $(x^T A, x^T b) \approx (\alpha^T, y^T)$ if $m \geq (m+1)q$ by leftover hash lemma.

= random

$$\text{Enc}_{P_k}(m) := (x^T A, x^T (As + e) + m[\frac{q}{2}])$$



Regev's encryption: security

Proof idea:

Step 1

LWE is secure: $(A, As + e) \approx (A, b)$
 \Downarrow We can replace the public key $(A, As + e)$ with (A, b)

Step 2

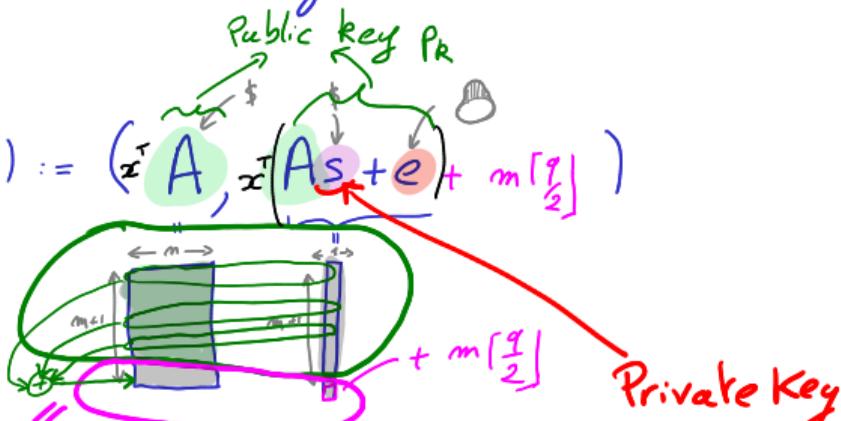
$= (A, b)$ is lossy: i.e. $(x^T A, x^T b) \approx (a^T, g^T)$ if $m \geq (m+1)q$ by leftover hash lemma.
 \Rightarrow statistically impossible to learn m from $(x^T A, x^T b + m[\frac{q}{2}])$

= random

$$\text{Enc}_{P_k}(m) := (x^T A, x^T (As + e + m[\frac{q}{2}]))$$

Public key =
 $= P_k$

Encryption



Regev's encryption: security

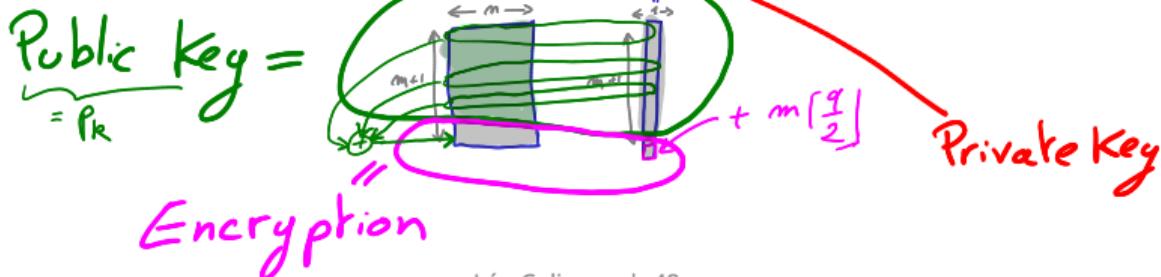
Proof idea:

Step 1: LWE is secure: $(A, As + e) \approx (A, b)$
↳ We can replace the public key $(A, As + e)$ with (A, b)

Step 2: $= (A, b)$ is lossy: i.e. $(x^T A, x^T b) \approx (a^T, y^T)$ if $m \geq (m+1)q$ by leftover hash lemma.
↳ statistically impossible to learn m from $(x^T A, x^T b + m[\frac{q}{2}]) \approx (a^T, y^T + m[\frac{q}{2}])$

= random

$$\text{Enc}_{P_k}(m) := (x^T A, x^T (As + e + m[\frac{q}{2}]))$$

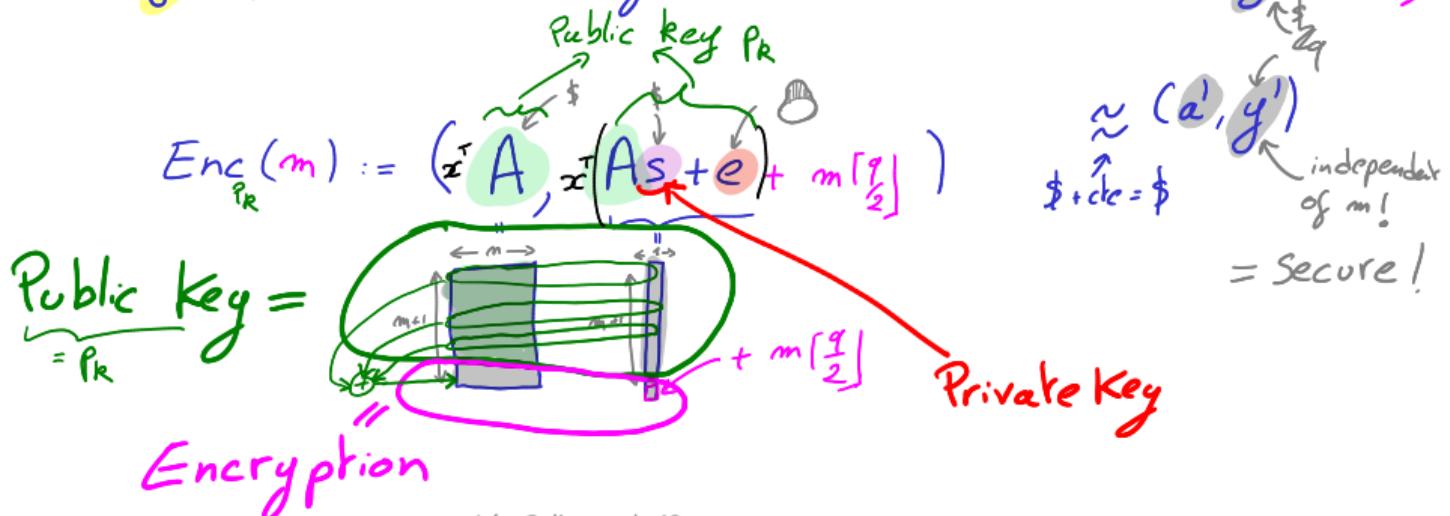


Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As + e) \approx (A, b)$
↳ 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 $m \geq (m+1)q$ by leftover hash lemma.
↳ statistically impossible to learn m from $(x^T A, x^T b + m[\frac{q}{2}]) \approx (a', y' + m[\frac{q}{2}])$



Regev's encryption: security

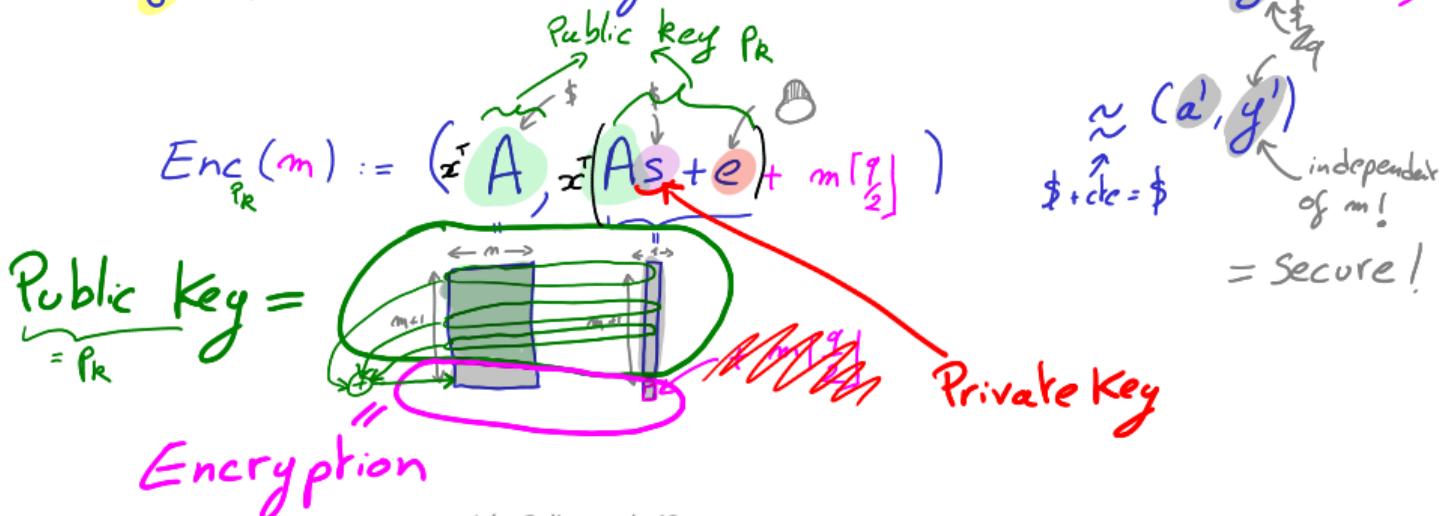
Proof idea:

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

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

Step 2: (A, b) is lossy: i.e. $(x^T A, x^T b) \approx (a^T, y^T)$ if $m \geq (m+1)q$ by leftover hash lemma.

\rightsquigarrow statistically impossible to learn m from $(x^T A, x^T b + m[\frac{q}{2}]) \approx (a^T, y^T + m[\frac{q}{2}])$

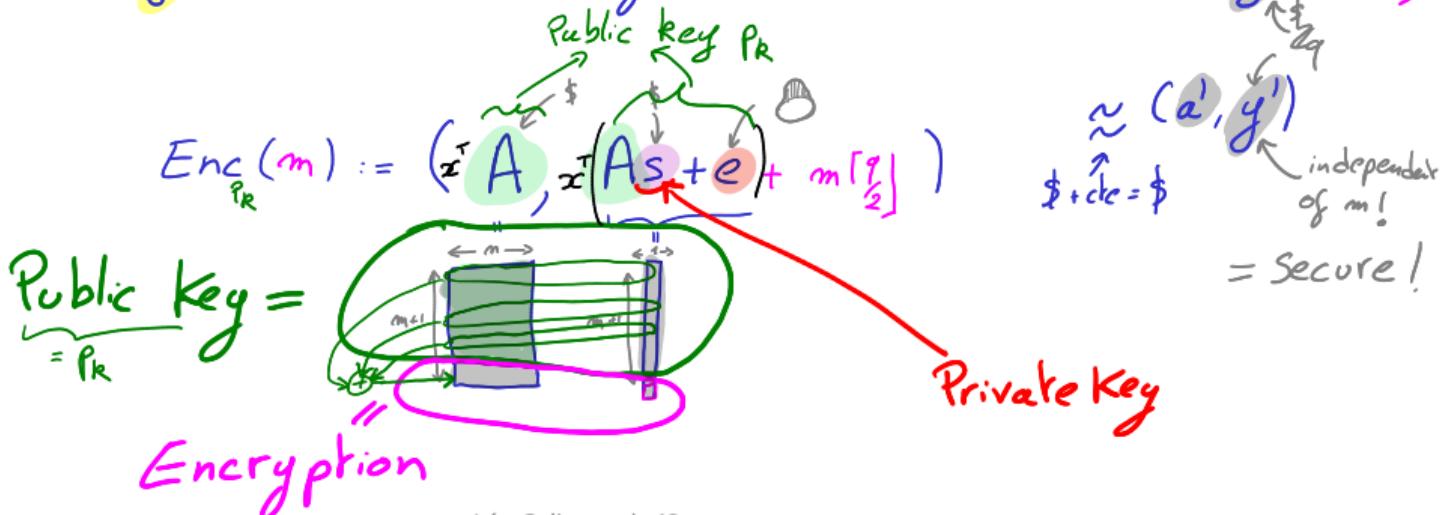


Regev's encryption: security

Proof idea:

Step 1: LWE is secure: $(A, As + e) \approx (A, b)$
↳ We can replace the public key $(A, As + e)$ with (A, b)

Step 2: $= (A, b)$ is lossy: i.e. $(x^T A, x^T b) \approx (a^T, y^T)$ if $m \geq (m+1)q$ by leftover hash lemma.
↳ statistically impossible to learn m from $(x^T A, x^T b + m[\frac{q}{2}]) \approx (a^T, y^T + m[\frac{q}{2}])$



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:
→ 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 $\sqrt{\frac{|G|}{2^m}} = \text{negl}(\lambda)$, then:

Leftover-real

$(g_1, \dots, g_m) \xleftarrow{\$} G^m$

GETELEMENTS():

return (g_1, \dots, g_m)

SAMPLE():

$x \xleftarrow{\$} \{0, 1\}^m$

return $\sum_i x_i g_i$

Leftover-rand

$(g_1, \dots, g_m) \xleftarrow{\$} G^m$

GETELEMENTS():

return (g_1, \dots, g_m)

SAMPLE():

$r \xleftarrow{\$} G$

return r

\approx

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: Consider $G = \mathbb{Z}_n^b$.

Regev's encryption: formal proof

Correction: on board

LWE in real life

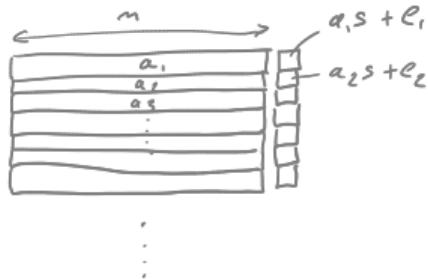
LWE in real life

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>

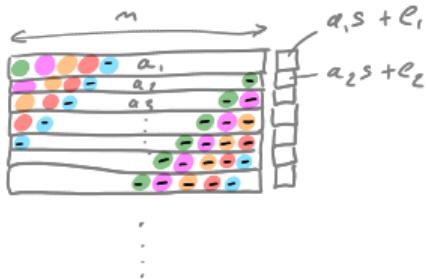
Intuition efficiency ring-LWE

LWE



Intuition efficiency ring-LWE

LWE \longrightarrow Ring-LWE



Idea: for increased efficiency,
use correlated samples like

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

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

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

:

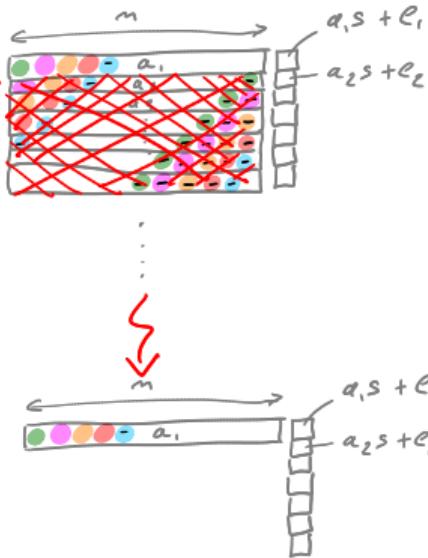
$$a_n = [x_n | -x_1 | \dots | -x_{n-1}]$$

by grouping samples n -by- n .

Intuition efficiency ring-LWE

LWE → Ring-LWE

Redundant
= no need
to store it!



Idea: for increased efficiency,
use correlated samples like

$$a_1 = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$$

$$a_2 = \begin{bmatrix} x_2 & x_3 & \dots & x_n - x_1 \end{bmatrix}$$

$$a_3 = \begin{bmatrix} x_3 & x_4 & \dots & -x_1 - x_2 \end{bmatrix}$$

⋮

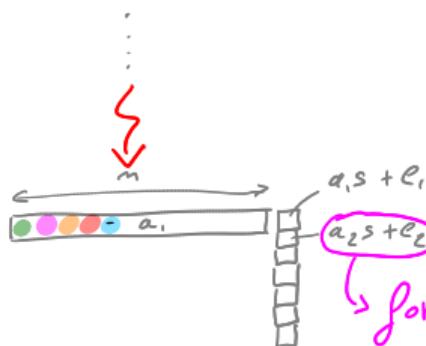
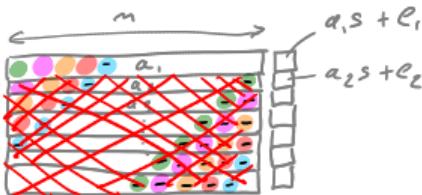
$$a_n = \begin{bmatrix} x_n - x_1 & \dots & -x_{n-1} - x_n \end{bmatrix}$$

by grouping samples n -by- n .

Intuition efficiency ring-LWE

LWE → Ring-LWE

Redundant
= no need
to store it!



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

Idea: for increased efficiency,
use correlated samples like

$$\alpha_1 = [x_1 \ | \ x_2 \ | \ \dots \ | \ x_n]$$

$$\alpha_2 = [x_2 \ | \ x_3 \ | \ \dots \ | \ x_n - x_1]$$

$$\alpha_3 = [x_3 \ | \ x_4 \ | \ \dots \ | \ -x_1 \ | \ -x_2]$$

:

$$\alpha_n = [x_n \ | \ -x_1 \ | \ \dots \ | \ -x_{n-1}]$$

by grouping samples n -by- n .

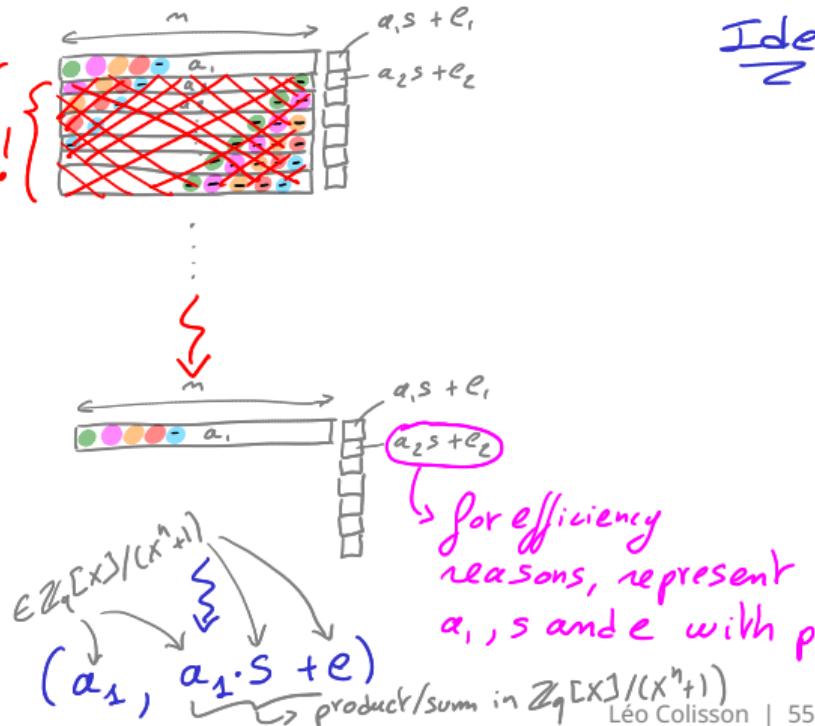
$$\begin{cases} a_1 = x_1 + x_2 X + \dots + x_n X^{n-1} \\ a_2 = a_1 X \pmod{(X^n + 1)} \\ \vdots \\ a_n = a_1 X^{n-1} \pmod{(X^n + 1)} \end{cases}$$

↑ i.e. $X^n = -1$

Intuition efficiency ring-LWE

LWE → Ring-LWE

Redundant
= no need
to store it!



Idea: for increased efficiency,
use correlated samples like

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

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

$$a_3 = [x_3 | x_4 | \dots | -x_1 | -x_n]$$

:

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

by grouping samples n -by- n .

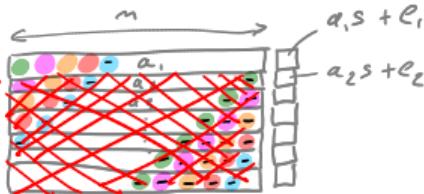
$$\begin{cases} a_1 = x_1 + x_2 X + \dots + x_n X^{n-1} \\ a_2 = a_1 X \pmod{(X^n+1)} \\ \vdots \\ a_n = a_1 X^{n-1} \pmod{(X^n+1)} \end{cases}$$

↑ i.e. $X^n = -1$

Intuition efficiency ring-LWE

LWE → Ring-LWE

Redundant
= no need
to store it!



$(a_1, a_1 s + e)$
product/sum in $\mathbb{Z}_q[X]/(X^n + 1)$

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

Idea: for increased efficiency,
use correlated samples like

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

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

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

:

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

by grouping samples n -by- n .

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

↑ 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



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

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

Ring-LWE

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 $A_{s,\chi}^{\text{ring}}$ 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 $A_{s,\chi}^{\text{ring}}$ 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 -LWE $_{\chi,m}$ is at least as hard as quantumly solving SVP $_\gamma$ on arbitrary ideal lattices in R , for some $\gamma = \text{poly}(n)/\alpha$.