

# An LLL algorithm for module lattices

Changmin Lee<sup>1</sup>, **Alice Pellet-Mary**<sup>2</sup>, Damien Stehlé<sup>1</sup>  
and Alexandre Wallet<sup>3</sup>

<sup>1</sup> ENS de Lyon, <sup>2</sup> KU Leuven, <sup>3</sup> NTT Tokyo

Asiacrypt,  
December 10, 2019

<https://eprint.iacr.org/2019/1035>



**KU LEUVEN**



ENS DE LYON



# What is this talk about?

First full-fledged LLL-type algorithm for modules over the ring of integers of a number field  $K$

- all number fields
- approx-factor  $\approx$  exponential in module rank
- quantum poly-time ... given a CVP oracle depending on  $K$

# Structured lattices

## Motivation

Improve efficiency of lattice-based schemes using structured lattices

# Structured lattices

## Motivation

Improve efficiency of lattice-based schemes using structured lattices

**Example:** NIST post-quantum standardization process

- 26 remaining candidates (2nd round)
- 12 lattice-based
- 11 using structured lattices

# Structured lattices

## Motivation

Improve efficiency of lattice-based schemes using structured lattices

**Example:** NIST post-quantum standardization process

- 26 remaining candidates (2nd round)
- 12 lattice-based
- 11 using structured lattices

	Frodo (unstructured lattices)	Kyber (structured lattices)
public key size (in Bytes)	9 616	800
ciphertexts size (in Bytes)	9 720	736

(CCA2 KEMs, round 2 version, security level 1)

# Structured lattices

## Motivation

Improve efficiency of lattice-based schemes using **structured lattices**

**Example:** NIST post-quantum standardization process

- 26 remaining candidates (2nd round)
- 12 lattice-based
- 11 using structured lattices

	Frodo (unstructured lattices)	Kyber (structured lattices)
public key size (in Bytes)	9 616	800
ciphertexts size (in Bytes)	9 720	736

(CCA2 KEMs, round 2 version, security level 1)

All 11 schemes use **module lattices**

# Module

- $K = \mathbb{Q}[X]/P(X)$
- $R = \mathbb{Z}[X]/P(X)$  (or  $R = \mathcal{O}_K$ )

with  $P$  monic and irreducible, degree  $d$

## Module

A (free) module  $M$  is a subset of  $K^k$  of the form  $M = \{B\vec{x} \mid \vec{x} \in R^k\}$ , with  $B \in K^{k \times k}$  invertible.  $B$  is a **basis** of  $M$ ;  $k$  is the **rank** of  $M$ .

# Module lattice

## Reminder:

- $K = \mathbb{Q}[X]/P(X)$  and  $R = \mathbb{Z}[X]/P(X)$
- $M = \{B\vec{x} : \vec{x} \in R^d\}$ , with  $B = (\vec{b}_1, \dots, \vec{b}_k)$  linearly independent



# Module lattice

## Reminder:

- $K = \mathbb{Q}[X]/P(X)$  and  $R = \mathbb{Z}[X]/P(X)$   
with  $\alpha_1, \dots, \alpha_d$  roots of  $P$  in  $\mathbb{C}$
- $M = \{B\vec{x} : \vec{x} \in R^d\}$ , with  $B = (\vec{b}_1, \dots, \vec{b}_k)$  linearly independent

## Canonical embedding

$$\sigma : K \rightarrow \mathbb{C}^d$$

$$x \mapsto (x(\alpha_1), \dots, x(\alpha_d))$$

# Module lattice

## Reminder:

- $K = \mathbb{Q}[X]/P(X)$  and  $R = \mathbb{Z}[X]/P(X)$   
with  $\alpha_1, \dots, \alpha_d$  roots of  $P$  in  $\mathbb{C}$
- $M = \{B\vec{x} : \vec{x} \in R^d\}$ , with  $B = (\vec{b}_1, \dots, \vec{b}_k)$  linearly independent

## Canonical embedding

$$\sigma : K \rightarrow \mathbb{C}^d$$

$$x \mapsto (x(\alpha_1), \dots, x(\alpha_d))$$

$$\sigma : K^k \rightarrow \mathbb{C}^{kd}$$

$$(x_1, \dots, x_k) \mapsto (\sigma(x_1), \dots, \sigma(x_k))$$

## Module lattice

### Reminder:

- $K = \mathbb{Q}[X]/P(X)$  and  $R = \mathbb{Z}[X]/P(X)$   
with  $\alpha_1, \dots, \alpha_d$  roots of  $P$  in  $\mathbb{C}$
- $M = \{B\vec{x} : \vec{x} \in R^d\}$ , with  $B = (\vec{b}_1, \dots, \vec{b}_k)$  linearly independent

### Canonical embedding

$$\sigma : K \rightarrow \mathbb{C}^d$$

$$x \mapsto (x(\alpha_1), \dots, x(\alpha_d))$$

$$\sigma : K^k \rightarrow \mathbb{C}^{kd}$$

$$(x_1, \dots, x_k) \mapsto (\sigma(x_1), \dots, \sigma(x_k))$$

$$\mathcal{L}(M) = \{\sigma(\vec{x}) : \vec{x} \in M\} \subset \mathbb{C}^{kd}$$

$\mathcal{L}(M)$  is a lattice of rank  $kd$ , spanned by  $(\sigma(\vec{b}_1), \sigma(x\vec{b}_1), \dots, \sigma(x^{d-1}\vec{b}_k))$

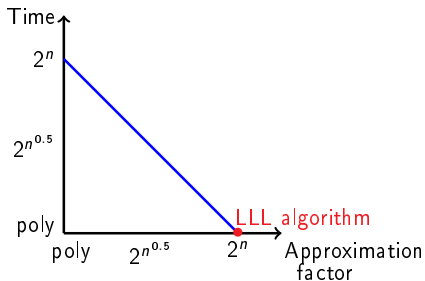
## Module lattices

- dimension  $n = kd$  over  $\mathbb{Z}$
- dimension  $k$  over  $R$

## Module lattices

- dimension  $n = kd$  over  $\mathbb{Z}$   
typically  $500 \leq kd \leq 1500$
- dimension  $k$  over  $R$   
typically  $k \leq 10$

# Motivation



Lattice reduction over  $\mathbb{Z}$

(in blue: BKZ trade-offs [Sch87, SE94])

## Module lattices

- dimension  $n = kd$  over  $\mathbb{Z}$   
typically  $500 \leq kd \leq 1500$
- dimension  $k$  over  $R$   
typically  $k \leq 10$

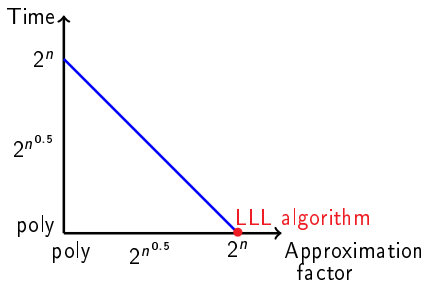
---

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. TCS.

[SE94] C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Mathematical programming.

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

# Motivation



Lattice reduction over  $\mathbb{Z}$   
(in blue: BKZ trade-offs [Sch87, SE94])

## Module lattices

- dimension  $n = kd$  over  $\mathbb{Z}$   
typically  $500 \leq kd \leq 1500$
- dimension  $k$  over  $R$   
typically  $k \leq 10$

Can we extend the LLL algorithm to lattices over  $R$ ?

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. TCS.

[SE94] C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Mathematical programming.

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

## Previous works and result

[Nap96] LLL for some specific number fields  
no bound on quality / run-time

---

[Nap96] H. Napias. A generalization of the LLL-algorithm over Euclidean rings or orders. Journal de théorie des nombres de Bordeaux.



## Previous works and result

- [Nap96] LLL for some specific number fields  
no bound on quality / run-time
- [FP96] LLL for any number fields  
no bound on quality / run-time  
bound on run-time for specific number fields

---

[FP96] C. Fieker, M. E. Pohst. Lattices over number fields. ANTS.

## Previous works and result

- [Nap96] LLL for some specific number fields  
no bound on quality / run-time
- [FP96] LLL for any number fields  
no bound on quality / run-time  
bound on run-time for specific number fields
- [KL17] LLL for norm-Euclidean fields  
bound on run-time but not on quality  
bound on quality for biquadratic fields

---

[KL17] T. Kim, C. Lee. Lattice reductions over euclidean rings with applications to cryptanalysis. IMACC.

## Previous works and result

- [Nap96] LLL for some specific number fields  
no bound on quality / run-time
- [FP96] LLL for any number fields  
no bound on quality / run-time  
bound on run-time for specific number fields
- [KL17] LLL for norm-Euclidean fields  
bound on run-time but not on quality  
bound on quality for biquadratic fields
- [LPSW19] LLL for any number field  
bound on quality and run-time if oracle solving CVP in a  
fixed lattice (depending only on  $R$ )

---

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

# Outline of the talk

- 1 The LLL algorithm
- 2 The Lagrange-Gauss algorithm
- 3 Computing the relaxed Euclidean division

# Outline of the talk

- 1 The LLL algorithm
- 2 The Lagrange-Gauss algorithm
- 3 Computing the relaxed Euclidean division

# High-level overview of LLL

## LLL over $\mathbb{Z}$

- $\gamma'$ -SVP in dim  $k$   
     $\leq$  1-SVP in dim 2
  - ▶  $\gamma' = 2^{O(k)}$
  - ▶ poly time

---

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

# High-level overview of LLL

## LLL over $\mathbb{Z}$

- $\gamma'$ -SVP in dim  $k$   
 $\leq$  1-SVP in dim 2
  - ▶  $\gamma' = 2^{O(k)}$
  - ▶ poly time
  
- Lagrange-Gauss algo  
for SVP in dim 2
  - ▶ poly time

---

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

# High-level overview of LLL

## LLL over $\mathbb{Z}$

- $\gamma'$ -SVP in dim  $k$   
 $\leq$  1-SVP in dim 2
  - ▶  $\gamma' = 2^{O(k)}$
  - ▶ poly time
  
- Lagrange-Gauss algo  
for SVP in dim 2
  - ▶ poly time

## Our contribution:

### LLL over $R = \mathbb{Z}[X]/(X^d + 1)$

- $\gamma'$ -SVP in rank- $k$   
 $\leq$   $\gamma$ -SVP in rank-2
  - ▶  $\gamma' = (\gamma d)^{O(k)}$
  - ▶ poly time

---

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



# High-level overview of LLL

## LLL over $\mathbb{Z}$

- $\gamma'$ -SVP in dim  $k$   
 $\leq$  1-SVP in dim 2
  - ▶  $\gamma' = 2^{O(k)}$
  - ▶ poly time
- Lagrange-Gauss algo  
for SVP in dim 2
  - ▶ poly time

## Our contribution:

### LLL over $R = \mathbb{Z}[X]/(X^d + 1)$

- $\gamma'$ -SVP in rank- $k$   
 $\leq$   $\gamma$ -SVP in rank-2
  - ▶  $\gamma' = (\gamma d)^{O(k)}$
  - ▶ poly time
- Algorithm for  $\gamma$ -SVP in rank-2
  - ▶  $\gamma = 2^{(\log d)^{O(1)}}$
  - ▶ heuristic, quantum
  - ▶ poly time if oracle solving CVP in a fixed lattice  
(depending only on  $R$ )

---

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

# High-level overview of LLL

## LLL over $\mathbb{Z}$

- $\gamma'$ -SVP in dim  $k$   
 $\leq$  1-SVP in dim 2
  - ▶  $\gamma' = 2^{O(k)}$
  - ▶ poly time
- Lagrange-Gauss algo for SVP in dim 2
  - ▶ poly time

## Our contribution:

### LLL over $R = \mathbb{Z}[X]/(X^d + 1)$

- $\gamma'$ -SVP in rank- $k$   
 $\leq \gamma$ -SVP in rank- $k$

needs QR-factorisation

- ▶ poly time

- Algorithm for  $\gamma$ -SVP in rank-2

- ▶  $\gamma = 2^{(\log d) \cdot O(1)}$

- ▶ heuristic
- ▶ poly oracle solving CVP in a fixed lattice  
(depending only on  $R$ )

next section

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

## Inner product over $R$

For  $\vec{a} = (a_1, \dots, a_k) \in K^k$  and  $\vec{b} = (b_1, \dots, b_k) \in K^k$ ,

$$\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \bar{b}_i \in K$$

## Inner product over $R$

For  $\vec{a} = (a_1, \dots, a_k) \in K^k$  and  $\vec{b} = (b_1, \dots, b_k) \in K^k$ ,

$$\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \bar{b}_i \in K$$

$$\|\vec{a}\|_K := \sqrt{\langle \vec{a}, \vec{a} \rangle_K} \in K \text{ (or } K_{\mathbb{R}})$$

## Inner product over $R$

For  $\vec{a} = (a_1, \dots, a_k) \in K^k$  and  $\vec{b} = (b_1, \dots, b_k) \in K^k$ ,

$$\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \bar{b}_i \in K$$

$$\|\vec{a}\|_K := \sqrt{\langle \vec{a}, \vec{a} \rangle_K} \in K \text{ (or } K_{\mathbb{R}})$$

### Properties

- $\text{Tr}(\|\vec{a}\|_K) = \|\sigma(\vec{a})\|_2$

$$\text{Tr}(x) = \sum_{i=1}^d \sigma(x)_i$$

## Inner product over $R$

For  $\vec{a} = (a_1, \dots, a_k) \in K^k$  and  $\vec{b} = (b_1, \dots, b_k) \in K^k$ ,

$$\langle \vec{a}, \vec{b} \rangle_K = \sum_i a_i \bar{b}_i \in K$$

$\|\vec{a}\|_K := \sqrt{\langle \vec{a}, \vec{a} \rangle_K} \in K$  (or  $K_{\mathbb{R}}$ )

### Properties

- $\text{Tr}(\|\vec{a}\|_K) = \|\sigma(\vec{a})\|_2$
- $\mathcal{N}(\|\vec{a}\|_K) = \Delta_K^{-1/2} \cdot \det(\mathcal{L}(\vec{a}))$

$$\mathcal{N}(x) = \prod_{i=1}^d \sigma(x)_i$$

## QR factorization over $R$

Let  $B = (b_1, \dots, b_k) \in K^k$ , define

$$b_i^* = b_i - \sum_{j < i} \mu_{ij} b_j^*, \quad \text{with } \mu_{ij} = \frac{\langle b_i, b_j^* \rangle_K}{\langle b_j^*, b_j^* \rangle_K}$$

## QR factorization over $R$

Let  $B = (b_1, \dots, b_k) \in K^k$ , define

$$b_i^* = b_i - \sum_{j < i} \mu_{ij} b_j^*, \quad \text{with } \mu_{ij} = \frac{\langle b_i, b_j^* \rangle_K}{\langle b_j^*, b_j^* \rangle_K}$$

**QR-factorisation:**  $B = QR$ , with

- $r_{ii} = \|b_i^*\|_K$ ,  $r_{ij} = \mu_{ji} r_{ii}$  for  $i < j$  and  $r_{ij} = 0$  otherwise
- columns of  $Q$  are  $b_i^* / \|b_i^*\|_K$



## QR factorization over $R$

Let  $B = (b_1, \dots, b_k) \in K^k$ , define

$$b_i^* = b_i - \sum_{j < i} \mu_{ij} b_j^*, \quad \text{with } \mu_{ij} = \frac{\langle b_i, b_j^* \rangle_K}{\langle b_j^*, b_j^* \rangle_K}$$

**QR-factorisation:**  $B = QR$ , with

- $r_{ii} = \|b_i^*\|_K$ ,  $r_{ij} = \mu_{ji} r_{ii}$  for  $i < j$  and  $r_{ij} = 0$  otherwise
- columns of  $Q$  are  $b_i^* / \|b_i^*\|_K$

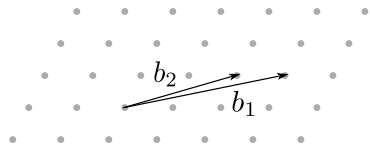
### Properties

- $\overline{Q}^T Q = I_k$
- $\det_K(B) = \prod_i r_{ii}$  and  $\det(\mathcal{L}(B)) = \prod_i \mathcal{N}(r_{ii})$
- $\forall v \in \mathcal{L}(B), \mathcal{N}(\|v\|_K) \geq \min \mathcal{N}(r_{ii})$

# Outline of the talk

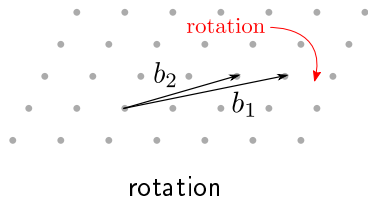
- 1 The LLL algorithm
- 2 The Lagrange-Gauss algorithm
- 3 Computing the relaxed Euclidean division

## Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} 10 & 7 \\ 2 & 2 \end{pmatrix}$$

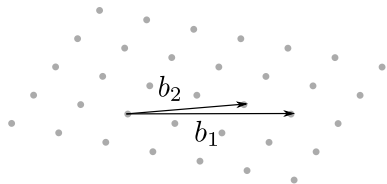
# Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} 10 & 7 \\ 2 & 2 \end{pmatrix}$$

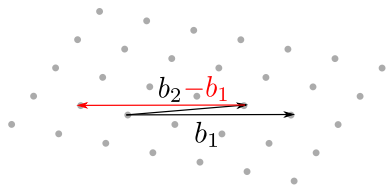
Compute QR factorization

## Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} 10.2 & 7.3 \\ 0 & 0.6 \end{pmatrix}$$

# Lagrange-Gauss algorithm (over $\mathbb{Z}$ )

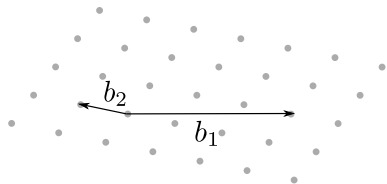


reduce  $b_2$  with  $b_1$

$$M = \begin{pmatrix} 10.2 & 7.3 \\ 0 & 0.6 \end{pmatrix}$$

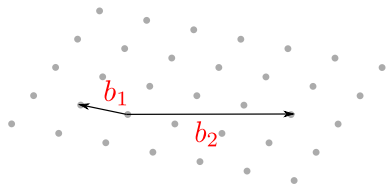
“Euclidean division” (over  $\mathbb{R}$ )  
of 7.3 by 10.2

## Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} 10.2 & -2.9 \\ 0 & 0.6 \end{pmatrix}$$

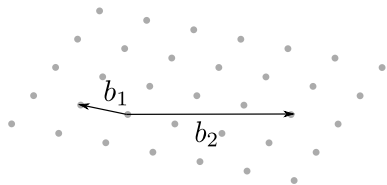
# Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} -2.9 & 10.2 \\ 0.6 & 0 \end{pmatrix}$$



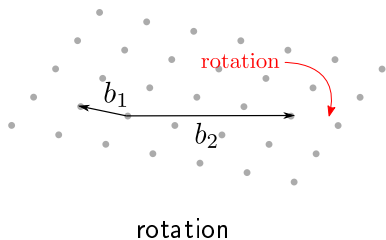
## Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



start again

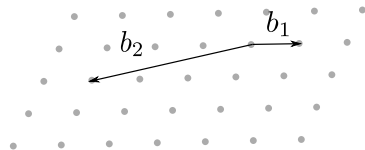
$$M = \begin{pmatrix} -2.9 & 10.2 \\ 0.6 & 0 \end{pmatrix}$$

# Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} -2.9 & 10.2 \\ 0.6 & 0 \end{pmatrix}$$

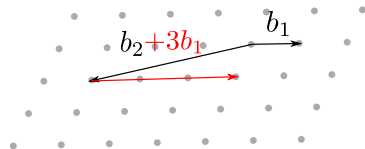
## Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



rotation

$$M = \begin{pmatrix} 3 & -10 \\ 0 & -2 \end{pmatrix}$$

# Lagrange-Gauss algorithm (over $\mathbb{Z}$ )

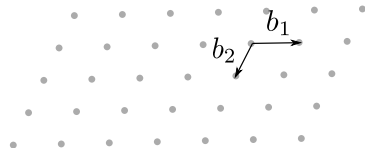


reduce  $b_2$  with  $b_1$

$$M = \begin{pmatrix} 3 & -10 \\ 0 & -2 \end{pmatrix}$$

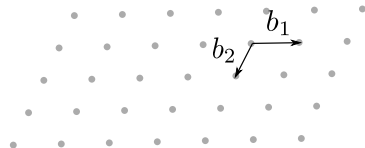
“Euclidean division” (over  $\mathbb{R}$ )  
of  $-10$  by  $3$

# Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} 3 & -1 \\ 0 & -2 \end{pmatrix}$$

## Lagrange-Gauss algorithm (over $\mathbb{Z}$ )

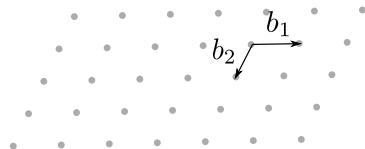


$$M = \begin{pmatrix} 3 & -1 \\ 0 & -2 \end{pmatrix}$$

For the Lagrange-Gauss algorithm over  $R$ , we need

- Rotation
- Euclidean division

## Lagrange-Gauss algorithm (over $\mathbb{Z}$ )



$$M = \begin{pmatrix} 3 & -1 \\ 0 & -2 \end{pmatrix}$$

For the Lagrange-Gauss algorithm over  $R$ , we need

- Rotation  $\Rightarrow$  ok
- Euclidean division  $\Rightarrow$  ?

# Euclidean division

**Over  $\mathbb{Z}$**

**Input:**  $a, b \in \mathbb{Z}, a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$



# Euclidean division

Over  $\mathbb{Z}$

**Input:**  $a, b \in \mathbb{Z}, a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .



# Euclidean division

**Over  $\mathbb{Z}$**

**Input:**  $a, b \in \mathbb{Z}$ ,  $a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .

**Over  $R$**

CVP in  $R$  with target  $-b/a$

$\Rightarrow$  output  $r \in R$



# Euclidean division

## Over $\mathbb{Z}$

**Input:**  $a, b \in \mathbb{Z}$ ,  $a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .



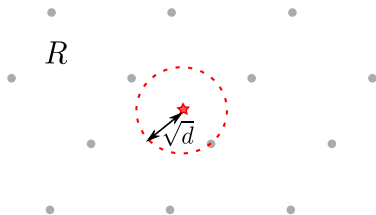
## Over $R$

CVP in  $R$  with target  $-b/a$

$\Rightarrow$  output  $r \in R$

**Difficulty:** Typically

$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$ .



# Euclidean division

## Over $\mathbb{Z}$

**Input:**  $a, b \in \mathbb{Z}, a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .

## Over $R$

CVP in  $R$  with target  $-b/a$

$\Rightarrow$  output  $r \in R$

**Difficulty:** Typically

$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|.$

## Relax the requirement

Find  $x, y \in R$  such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$

# Euclidean division

## Over $\mathbb{Z}$

**Input:**  $a, b \in \mathbb{Z}$ ,  $a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .

## Over $R$

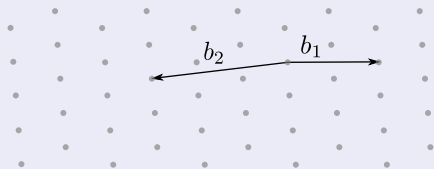
CVP in  $R$  with target  $-b/a$   
 $\Rightarrow$  output  $r \in R$

**Difficulty:** Typically  
 $\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$ .

## Relax the requirement

Find  $x, y \in R$  such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$



# Euclidean division

## Over $\mathbb{Z}$

**Input:**  $a, b \in \mathbb{Z}$ ,  $a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .

## Over $R$

CVP in  $R$  with target  $-b/a$

$\Rightarrow$  output  $r \in R$

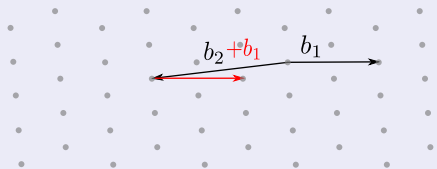
**Difficulty:** Typically

$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$ .

## Relax the requirement

Find  $x, y \in R$  such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$



# Euclidean division

## Over $\mathbb{Z}$

**Input:**  $a, b \in \mathbb{Z}$ ,  $a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .

## Over $R$

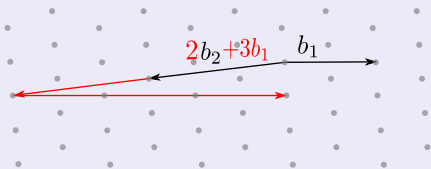
CVP in  $R$  with target  $-b/a$   
 $\Rightarrow$  output  $r \in R$

**Difficulty:** Typically  
 $\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$ .

## Relax the requirement

Find  $x, y \in R$  such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$



# Euclidean division

## Over $\mathbb{Z}$

**Input:**  $a, b \in \mathbb{Z}$ ,  $a \neq 0$

**Output:**  $r \in \mathbb{Z}$

such that  $|b + ra| \leq |a|/2$

CVP in  $\mathbb{Z}$  with target  $-b/a$ .

## Over $R$

CVP in  $R$  with target  $-b/a$

$\Rightarrow$  output  $r \in R$

**Difficulty:** Typically

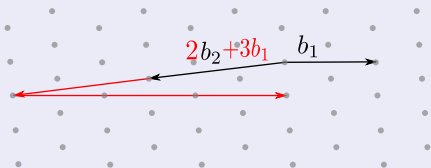
$\|b + ra\| \approx \sqrt{d} \cdot \|a\| \gg \|a\|$ .

## Relax the requirement

Find  $x, y \in R$  such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$

$\Rightarrow$  sufficient for Gauss' algo





# Outline of the talk

- 1 The LLL algorithm
- 2 The Lagrange-Gauss algorithm
- 3 Computing the relaxed Euclidean division

# The Log space

**Reminder:**  $\sigma(r) = (r(\alpha_1), \dots, r(\alpha_d))^T$   
(multiplication is coefficient-wise)

# The Log space

**Reminder:**  $\sigma(r) = (r(\alpha_1), \dots, r(\alpha_d))^T$

(multiplication is coefficient-wise)

$$\text{Log}(r) = (\log |r(\alpha_1)|, \dots, \log |r(\alpha_d)|)^T$$

# The Log space

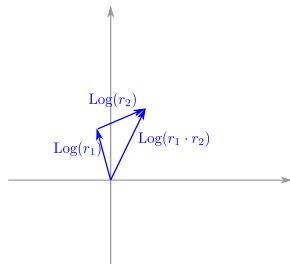
**Reminder:**  $\sigma(r) = (r(\alpha_1), \dots, r(\alpha_d))^T$

(multiplication is coefficient-wise)

$$\text{Log}(r) = (\log |r(\alpha_1)|, \dots, \log |r(\alpha_d)|)^T$$

## Properties of Log

- $\text{Log}(r_1 \cdot r_2) = \text{Log}(r_1) + \text{Log}(r_2)$



# The Log space

**Reminder:**  $\sigma(r) = (r(\alpha_1), \dots, r(\alpha_d))^T$

(multiplication is coefficient-wise)

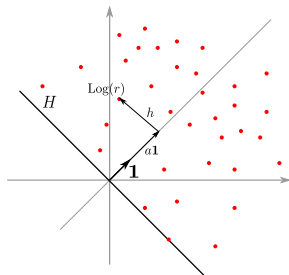
$$\text{Log}(r) = (\log |r(\alpha_1)|, \dots, \log |r(\alpha_d)|)^T$$

Let  $\mathbf{1} = (1, \dots, 1)$  and  $H = \mathbf{1}^\perp$

## Properties of Log

$\text{Log } r = h + a\mathbf{1}$ , with  $h \in H$

- $\text{Log}(r_1 \cdot r_2) = \text{Log}(r_1) + \text{Log}(r_2)$
- $a \geq 0$  if  $r \in R$



# The Log space

**Reminder:**  $\sigma(r) = (r(\alpha_1), \dots, r(\alpha_d))^T$

(multiplication is coefficient-wise)

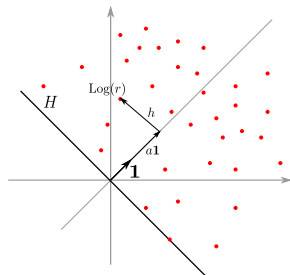
$$\text{Log}(r) = (\log |r(\alpha_1)|, \dots, \log |r(\alpha_d)|)^T$$

Let  $\mathbf{1} = (1, \dots, 1)$  and  $H = \mathbf{1}^\perp$

## Properties of Log

$\text{Log } r = h + a\mathbf{1}$ , with  $h \in H$

- $\text{Log}(r_1 \cdot r_2) = \text{Log}(r_1) + \text{Log}(r_2)$
- $a \geq 0$  if  $r \in R$
- $\|r\| \simeq 2^{\|\text{Log } r\|_\infty}$



## Using the Log space

Objective: find  $x, y \in R$  such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$

## Using the Log space

Objective: find  $x, y \in R$  such that

- $\|xa + yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$

**Difficulty:** Log works well with  $\times$ , but not with  $+$



## Using the Log space

Objective: find  $x, y \in R$  such that

- $\|xa - yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$

**Difficulty:** Log works well with  $\times$ , but not with  $+$

## Using the Log space

Objective: find  $x, y \in R$  such that

- $\|xa - yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$

**Difficulty:** Log works well with  $\times$ , but not with  $+$

**Solution:** If  $\|\text{Log}(u) - \text{Log}(v)\| \leq \varepsilon$   
then  $\|u - v\| \lesssim \varepsilon \cdot \min(\|u\|, \|v\|)$   
(requires to extend Log to take arguments into account)

## Using the Log space

Objective: find  $x, y \in R$  such that

- $\|xa - yb\| \leq \|a\|/2$
- $\|y\| \leq \text{poly}(d)$

**Difficulty:** Log works well with  $\times$ , but not with  $+$

**Solution:** If  $\|\text{Log}(u) - \text{Log}(v)\| \leq \varepsilon$   
then  $\|u - v\| \lesssim \varepsilon \cdot \min(\|u\|, \|v\|)$   
(requires to extend Log to take arguments into account)

### New objective

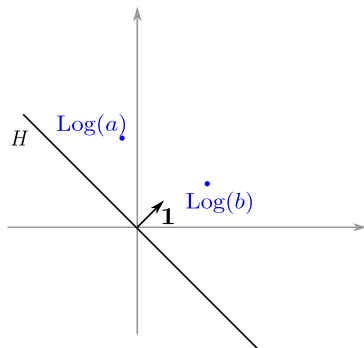
Find  $x, y \in R$  such that

- $\|\text{Log}(xa) - \text{Log}(yb)\| \leq \varepsilon$
- $\|\text{Log}(y)\|_\infty \leq O(\log d)$

# Idea

Objective: find  $x, y \in R$  s.t.

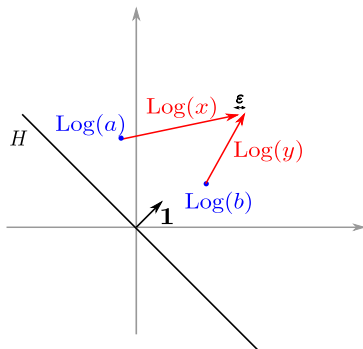
- $\| \text{Log}(xa) - \text{Log}(yb) \| \leq \varepsilon$
- $\| \text{Log}(y) \|_\infty \leq O(\log d)$



# Idea

Objective: find  $x, y \in R$  s.t.

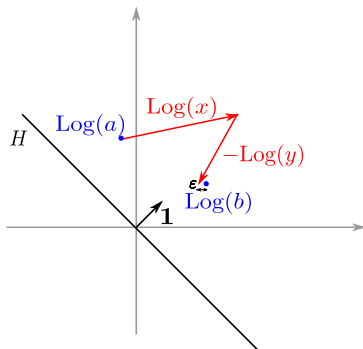
- $\| \text{Log}(xa) - \text{Log}(yb) \| \leq \varepsilon$
- $\| \text{Log}(y) \|_\infty \leq O(\log d)$



# Idea

Objective: find  $x, y \in R$  s.t.

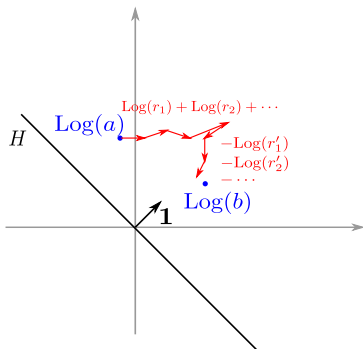
- $\|(\text{Log}(x) - \text{Log}(y)) - \text{Log}(b/a)\| \leq \varepsilon$
- $\|\text{Log}(y)\|_\infty \leq O(\log d)$



# Idea

Objective: find  $x, y \in R$  s.t.

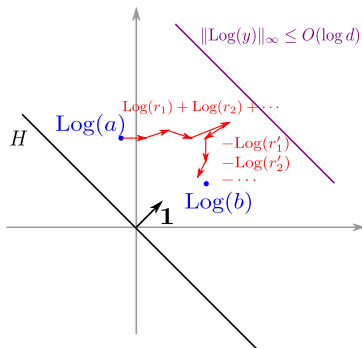
- $\|(\text{Log}(x) - \text{Log}(y)) - \text{Log}(b/a)\| \leq \varepsilon$
- $\|\text{Log}(y)\|_\infty \leq O(\log d)$



# Idea

Objective: find  $x, y \in R$  s.t.

- $\|(\text{Log}(x) - \text{Log}(y)) - \text{Log}(b/a)\| \leq \varepsilon$
- $\|\text{Log}(y)\|_\infty \leq O(\log d)$





# Idea

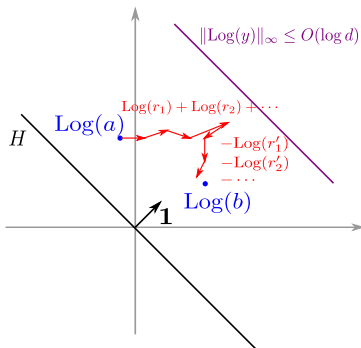
Objective: find  $x, y \in R$  s.t.

- $\|(\text{Log}(x) - \text{Log}(y)) - \text{Log}(b/a)\| \leq \varepsilon$
- $\|\text{Log}(y)\|_\infty \leq O(\log d)$

Solve exact CVP in  $L$  with target  $t$

$$L = \begin{pmatrix} \text{Log } r_1 & \cdots & \text{Log } r_{d^2} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \text{Log}(b/a) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

( $L$  is fixed and independent of  $a$  and  $b$ )



# Idea

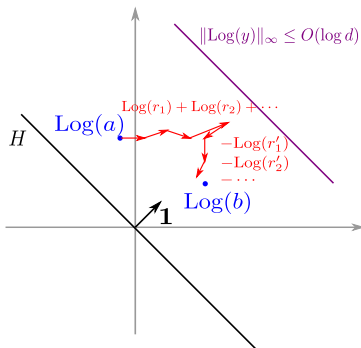
Objective: find  $x, y \in R$  s.t.

- $\|(\text{Log}(x) - \text{Log}(y)) - \text{Log}(b/a)\| \leq \varepsilon$
- $\|\text{Log}(y)\|_\infty \leq O(\log d)$

Solve exact CVP in  $L$  with target  $t$   
with an oracle

$$L = \begin{pmatrix} \text{Log } r_1 & \cdots & \text{Log } r_{d^2} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \text{Log}(b/a) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

( $L$  is fixed and independent of  $a$  and  $b$ )



# Idea

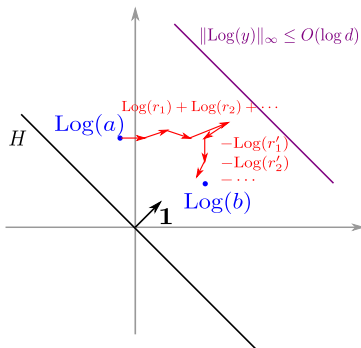
Objective: find  $x, y \in R$  s.t.

- $\|(\text{Log}(x) - \text{Log}(y)) - \text{Log}(b/a)\| \leq \varepsilon$
- $\|\text{Log}(y)\|_\infty \leq O(\log d)$

Solve exact CVP in  $L$  with target  $t$   
with an oracle

$$L = \begin{pmatrix} \text{Log } r_1 & \cdots & \text{Log } r_{d^2} \\ 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}, \quad t = \begin{pmatrix} \text{Log}(b/a) \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

( $L$  is fixed and independent of  $a$  and  $b$ )



Complexity

Quantum poly time  
(with the oracle)

# Under the carpet

- Heuristics
  - ▶ maths justification
  - ▶ numerical experiments (in very small dimension)
- Any module / ideal
  - ▶ use pseudo-basis
  - ▶ add units and class group to  $L$  (cf [Buc88])
- Full LLL algo over  $R$ 
  - ▶ Lovász' swap condition
  - ▶ switch between  $\mathcal{N}(\cdot)$  and  $\|\cdot\|$
  - ▶ handling bit sizes

---

[Buc88] J. Buchmann. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. Séminaire de théorie des nombres.

## Summary and impact

### LLL algorithm for power-of-two cyclotomic fields

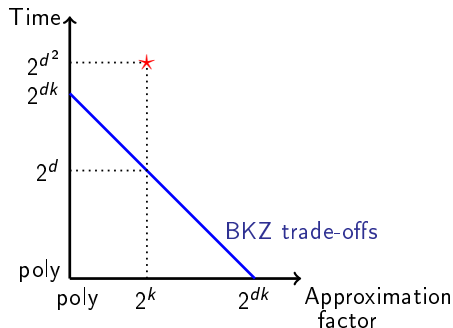
- Approx: quasi-poly( $d$ ) $^{O(k)} = 2^{(\log d)^{O(1)} \cdot k}$
- Time: quantum polynomial time  
if oracle solving CVP in  $L$  (of dim  $O(d^{2+\epsilon})$ )

# Summary and impact

## LLL algorithm for power-of-two cyclotomic fields

- Approx: quasi-poly( $d$ ) $^{O(k)} = 2^{(\log d)^{O(1)} \cdot k}$
- Time: quantum polynomial time  
if oracle solving CVP in  $L$  (of dim  $O(d^{2+\epsilon})$ )

In practice?  $\Rightarrow$  replace the oracle by a CVP solver

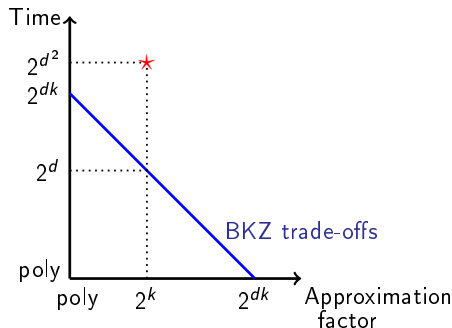


# Summary and impact

## LLL algorithm for power-of-two cyclotomic fields

- Approx: quasi-poly( $d$ ) $^{O(k)} = 2^{(\log d)^{O(1)} \cdot k}$
- Time: quantum polynomial time  
if oracle solving CVP in  $L$  (of dim  $O(d^{2+\epsilon})$ )

In practice?  $\Rightarrow$  replace the oracle by a CVP solver



$\Rightarrow$  theoretical result  
(not practical)

# Conclusion

Open problems:

- Better understanding of the lattice  $L$ 
  - ▶ reduce its dimension to  $\tilde{O}(d)$ ?
  - ▶ prove the heuristics?
  - ▶ better CVP solver for  $L$ ?

---

[MS19] T. Mukherjee, N. Stephens-Davidowitz. Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP, ePrint.



# Conclusion

Open problems:

- Better understanding of the lattice  $L$ 
  - ▶ reduce its dimension to  $\tilde{O}(d)$ ?
  - ▶ prove the heuristics?
  - ▶ better CVP solver for  $L$ ?
- Generalizing LLL to all the BKZ trade-offs?
  - ▶ reduction rank  $k$  to rank  $\beta$  (cf [MS19])
  - ▶ sieving/enumeration in modules?

---

[MS19] T. Mukherjee, N. Stephens-Davidowitz. Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP, ePrint.

# Conclusion

Open problems:

- Better understanding of the lattice  $L$ 
  - ▶ reduce its dimension to  $\tilde{O}(d)$ ?
  - ▶ prove the heuristics?
  - ▶ better CVP solver for  $L$ ?
- Generalizing LLL to all the BKZ trade-offs?
  - ▶ reduction rank  $k$  to rank  $\beta$  (cf [MS19])
  - ▶ sieving/enumeration in modules?

Thank you

---

[MS19] T. Mukherjee, N. Stephens-Davidowitz. Lattice Reduction for Modules, or How to Reduce ModuleSVP to ModuleSVP, ePrint.