Reductions from module lattices to free module lattices, and application to dequantizing module-LLL

Gabrielle De Micheli, Daniele Micciancio, Alice Pellet-Mary, Nam Tran

Crypto 2023

Motivation and main result

We focus on **algebraically structured lattices**.

More precisely: lattice-based algorithmic problems using **module** lattices.
motivation: efficiency of lattice-based schemes, NIST finalists.

Main question

Restricting to module lattices with specific structure (*i.e.*, **free modules**), do standard algorithmic problems (SVP, CVP ...) become easier ?

Motivation and main result

We focus on **algebraically structured lattices**.

More precisely: lattice-based algorithmic problems using **module** lattices.

motivation: efficiency of lattice-based schemes, NIST finalists.

Main question

Restricting to module lattices with specific structure (*i.e.*, **free modules**), do standard algorithmic problems (SVP, CVP ...) become easier ?

We show that: free modules are no weaker than arbitrary modules: there exist probabilistic polynomial time reductions from

$$\text{module-}\mathcal{P} \leq \textbf{free}\text{-module-}\mathcal{P}.$$

Mathematical background

A **module** is an algebraic object. It lives in a **number field**.

- Number field:

$$K \simeq \mathbb{Q}[X]/(p(X)),$$

$p(X) \in \mathbb{Q}[X]$ irreducible of degree d.

- Ring of integers:

$$\mathcal{O}_K := \{x \in K : g(x) = 0 \text{ for some } g(X) \text{ monic in } \mathbb{Z}[X]\}$$

Examples

- $K = \mathbb{Q}$, $\mathcal{O}_K = \mathbb{Z}$
- $K = \mathbb{Q}(\mathbf{i})$, $\mathcal{O}_K = \mathbb{Z}[\mathbf{i}] = \{a + b \cdot \mathbf{i} : a, b \in \mathbb{Z}\}$ (Gauss integers)

Modules

More concretely, what is a **module**?

Definition ((Finitely-generated) module)

A module is defined as

$$M = \left\{ \sum_{i=1}^{t} \alpha_i v_i : \alpha_i \in \mathcal{O}_K \right\}$$

with generators $v_1, \ldots, v_t \in K^m$.

The linear combinations are ring combinations.

- If $M \subset K$ $(m = 1)$ then M is a fractional ideal of K

A notion of (pseudo)-basis

- There exists ideals $I_1, \ldots, I_n$ and K-linearly independent vectors $b_1, \ldots, b_n$ such that

$$M = I_1 b_1 + \ldots + I_n b_n$$

A notion of (pseudo)-basis

- There exists ideals $I_1, \ldots, I_n$ and K-linearly independent vectors $b_1, \ldots, b_n$ such that

$$M = I_1 b_1 + \ldots + I_n b_n$$

- $((I_i, b_i))_{1 \leqslant i \leqslant n}$ is the **pseudo-basis** of M and n is the **rank** of M.

A notion of (pseudo)-basis

- There exists ideals $I_1, \ldots, I_n$ and K-linearly independent vectors $b_1, \ldots, b_n$ such that

$$M = I_1 b_1 + \ldots + I_n b_n$$

- $((I_i, b_i))_{1 \leqslant i \leqslant n}$ is the **pseudo-basis** of M and n is the **rank** of M.

Definition (Free module)

When $I_i = \mathcal{O}_K$, M is a **free** module.

Embedding and Geometry

How can we add some **geometry** to modules ?

- There exists d *embeddings* (injective homomorphisms) from K to $\mathbb{C}$:

$$\underbrace{\sigma_1, \ldots, \sigma_{r_1},}_{r_1 \text{ real embeddings}} \underbrace{\sigma_{r_1+1}, \overline{\sigma_{r_1+1}}, \ldots, \sigma_{r_1+r_2}, \overline{\sigma_{r_1+r_2}}}_{2r_2 \text{ complex embeddings}}$$

Embedding and Geometry

How can we add some **geometry** to modules ?

- There exists d *embeddings* (injective homomorphisms) from K to $\mathbb{C}$:

$$\underbrace{\sigma_1, \dots, \sigma_{r_1},}_{r_1 \text{ real embeddings}} \underbrace{\sigma_{r_1+1}, \overline{\sigma_{r_1+1}}, \dots, \sigma_{r_1+r_2}, \overline{\sigma_{r_1+r_2}}}_{2r_2 \text{ complex embeddings}}$$

- K can be embedded into $\mathbb{R}^d$ by the **canonical embedding**:

$$\sigma(x) = (\sigma_1(x), \cdots, \sigma_{r_1}(x), \mathrm{Re}(\sigma_{r_1+1}(x)), \mathrm{Im}(\sigma_{r_1+1}(x)), \cdots, \mathrm{Im}(\sigma_{r_1+r_2}(x)$$

Embedding and Geometry

How can we add some **geometry** to modules ?

- There exists d *embeddings* (injective homomorphisms) from K to $\mathbb{C}$:

$$\underbrace{\sigma_1, \ldots, \sigma_{r_1},}_{r_1 \text{ real embeddings}} \underbrace{\sigma_{r_1+1}, \overline{\sigma_{r_1+1}}, \ldots, \sigma_{r_1+r_2}, \overline{\sigma_{r_1+r_2}}}_{2r_2 \text{ complex embeddings}}$$

- K can be embedded into $\mathbb{R}^d$ by the **canonical embedding**:

$$\sigma(x) = (\sigma_1(x), \cdots, \sigma_{r_1}(x), \mathrm{Re}(\sigma_{r_1+1}(x)), \mathrm{Im}(\sigma_{r_1+1}(x)), \cdots, \mathrm{Im}(\sigma_{r_1+r_2}(x)$$

$\rightarrow$ Inducing a **geometry** over K: for $x \in K$, define $\|x\| = \|\sigma(x)\|$

Embedding and Geometry

How can we add some **geometry** to modules ?

- There exists d *embeddings* (injective homomorphisms) from K to $\mathbb{C}$:

$$\underbrace{\sigma_1, \ldots, \sigma_{r_1},}_{r_1 \text{ real embeddings}} \quad \underbrace{\sigma_{r_1+1}, \overline{\sigma_{r_1+1}}, \ldots, \sigma_{r_1+r_2}, \overline{\sigma_{r_1+r_2}}}_{2r_2 \text{ complex embeddings}}$$

- K can be embedded into $\mathbb{R}^d$ by the **canonical embedding**:

$$\sigma(x) = (\sigma_1(x), \cdots, \sigma_{r_1}(x), \mathrm{Re}(\sigma_{r_1+1}(x)), \mathrm{Im}(\sigma_{r_1+1}(x)), \cdots, \mathrm{Im}(\sigma_{r_1+r_2}(x)$$

$\rightarrow$ Inducing a **geometry** over K: for $x \in K$, define $\|x\| = \|\sigma(x)\|$

- Embedding M to $\mathbb{R}^{dn}$ gives a rank-dn **lattice.**
  (K^n can be embedded to $\mathbb{R}^{dn}$ by σ).

Lattice Problems

We focus on the following lattice problems.

- **Shortest Vector Problem (SVP)**: find $v \in \mathcal{L}\setminus\{0\}$ such that $\|v\| = \lambda_1(\mathcal{L})$. $\left(\lambda_1(\mathcal{L}) = \min_{v\in\mathcal{L}\setminus\{0\}} \|v\|\right)$.
- **Closest Vector Problem (CVP)**: given $\mathcal{L}$ and t, find $v \in \mathcal{L}$ such that $\|v - t\| = \mathrm{dist}(t, \mathcal{L})$. $\left(\mathrm{dist}(t,\mathcal{L}) = \min_{v\in\mathcal{L}} \|v - t\|\right)$.

Lattice Problems

We focus on the following lattice problems.

- **Shortest Vector Problem (SVP)**: find $v \in \mathcal{L}\backslash\{0\}$ such that $\|v\| = \lambda_1(\mathcal{L})$. $(\lambda_1(\mathcal{L}) = \min_{v \in \mathcal{L}\backslash\{0\}} \|v\|)$.
- **Closest Vector Problem (CVP)**: given $\mathcal{L}$ and t, find $v \in \mathcal{L}$ such that $\|v - t\| = \mathrm{dist}(t, \mathcal{L})$. $(\mathrm{dist}(t, \mathcal{L}) = \min_{v \in \mathcal{L}} \|v - t\|)$.

Approximate variants:

- γ-**SVP**: given $\mathcal{L}$, find $v \in \mathcal{L}\backslash\{0\}$ such that $\|v\| \leqslant \gamma \cdot \lambda_1(\mathcal{L})$.
- **Hermite SVP (HSVP)**: find $v \in \mathcal{L}\backslash\{0\}$ such that $\|v\| \leqslant \gamma \cdot \mathrm{vol}(\mathcal{L})^{1/n}$.
- γ-**CVP**: find $v \in \mathcal{L}$ such that $\|v - t\| \leqslant \gamma \cdot \mathrm{dist}(t, \mathcal{L})$.

Detailed contribution

We show that for $\mathcal{P}$ (either SVP, HSVP or CVP) and $n \geq 2$, there exist probabilistic polynomial time reductions from

$$n\text{-module-}\mathcal{P} \leq n\text{-}\textbf{free}\text{-module-}\mathcal{P}.$$

Detailed contribution

We show that for $\mathcal{P}$ (either SVP, HSVP or CVP) and $n \geq 2$, there exist probabilistic polynomial time reductions from

$$n\text{-module-}\mathcal{P} \leq n\text{-}\textbf{free}\text{-module-}\mathcal{P}.$$

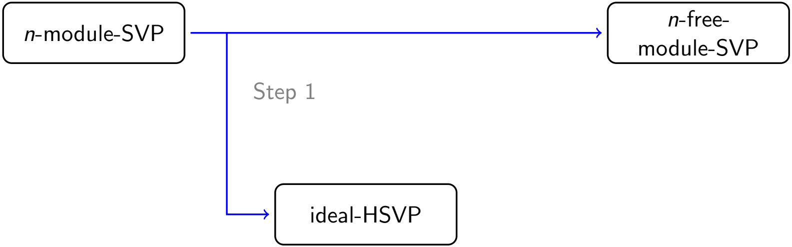
- Similar technique for all three algorithmic problems: focus on **SVP** in this presentation.
- Three subreductions.

Main application: provides a *fully classical* LLL algorithm for module lattices.

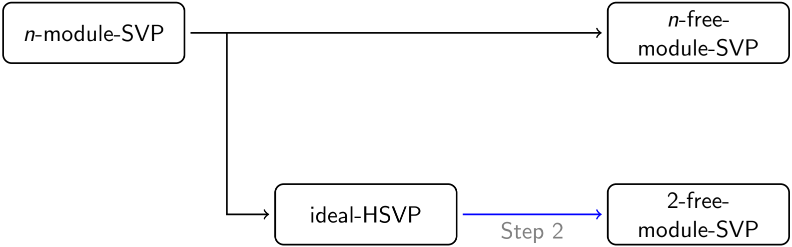
Layout of the reductions

n-module-SVP

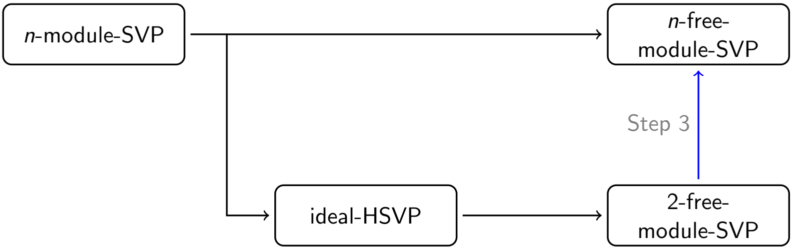
Layout of the reductions



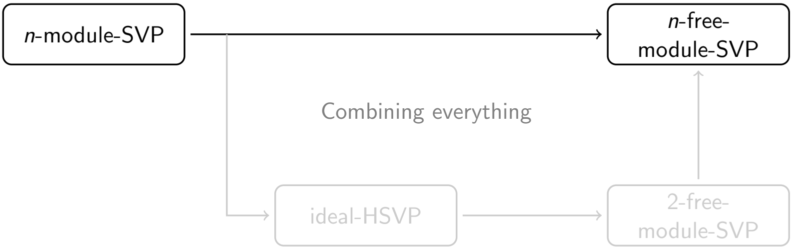
Layout of the reductions



Layout of the reductions



Layout of the reductions



Step 1: from module-SVP to free-module-SVP and ideal-HSVP

Goal: Find a reduction from module-SVP to free-module-SVP and ideal-HSVP.

Main technique: use an *almost-free representation* of the input module.

Theorem (informal)

A rank-n module always admits an *almost-free* representation, which is a pseudo-basis of the form

$$(\mathcal{O}_K, b_1); \ldots; (\mathcal{O}_K, b_{n-1}); (I, b_n).$$

Such representation can be computed in polynomial time.

Step 1: high level idea

Input: a rank-n module M, an oracle for ideal-HSVP and an oracle for free-module-SVP.
Output: a solution to SVP for module M.

- Compute an almost-free basis of M, *i.e.,* $(b_1, \cdots, b_{n-1}, (I, b_n))$.
- Solve HSVP in ideal I to find short element $\alpha \in I \setminus \{0\}$.
- Construct free submodule N of M spanned by $b_1, \cdots, b_{n-1}, \alpha b_n$.
- Solve free-module-SVP with input N.

$N \subset M$ so a solution to SVP in N is also a solution to SVP in M.

Step 2: from ideal-HSVP to free-module-SVP in rank 2.

Goal: Find a reduction from ideal-HSVP to free-module-SVP in rank 2.

Main technique: use a *two-element representation* of the input ideal I.

Theorem (informal)

Every ideal I in a number field has a two-element representation

$$I = (a) + (b),$$

where $a, b \in I$. There is a probabilistic algorithm computing it in expected polynomial time.

Step 2: high level idea

Input: an ideal I and an oracle for free-module-SVP in rank 2.
Output: a solution to ideal-HSVP for ideal I.

- Transform input ideal into *free* rank-2 module M_2.
- Solve free-module-SVP on M_2.

$$I \longrightarrow I = (a) + (b) \longrightarrow M_2 \subset K^2 \text{ with basis } \begin{pmatrix} a \\ 0 \end{pmatrix}, \begin{pmatrix} b \\ \varepsilon \end{pmatrix}.$$

- Solving SVP on M_2 should produce short vector of the form

$$\begin{pmatrix} ua + vb \\ v\varepsilon \end{pmatrix};$$

  where $u, v \in \mathcal{O}_K$. We need to make sure $ua + vb$ is small and non-zero.
- The quantity $ua + vb$ is a solution to ideal-HSVP in I.

Step 3: from free-module-SVP in rank 2 to free-module-SVP in rank n.

Goal: Find a reduction from free-module-SVP in rank 2 to free-module-SVP in rank n.

Main technique: Embed the rank 2 input module into a larger rank module, and pad the extra dimensions with dummy vectors.

Step 3: high level idea

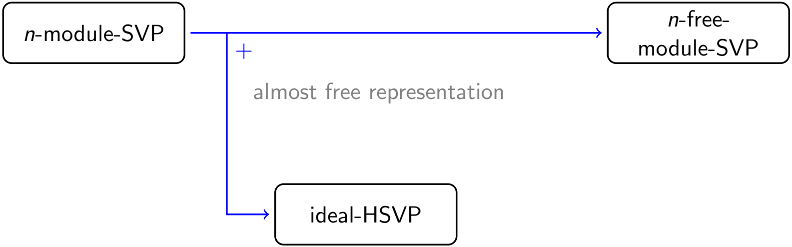
Input: A rank-2 free module $M_2 \subset K^2$ with basis $\tilde{\mathbf{B}}$, and an oracle for free-module-SVP in rank n.
Output: a solution to free-module-SVP for rank-2 module M_2.

- Construct a rank-n free module $M \subset K^n$ generated by the columns of the block matrix:
$$\left(\begin{array}{c|c} \tilde{\mathbf{B}} & 0 \\ \hline 0 & \delta I_{n-2} \end{array}\right)$$
- Solve SVP on free module M. The output is a vector $s = (s_1, s_2, \cdots, s_n)$.
- The vector $\tilde{s} = (s_1, s_2)$ is a solution to free-module-SVP for M_2. The δ value needs to be appropriately chosen.

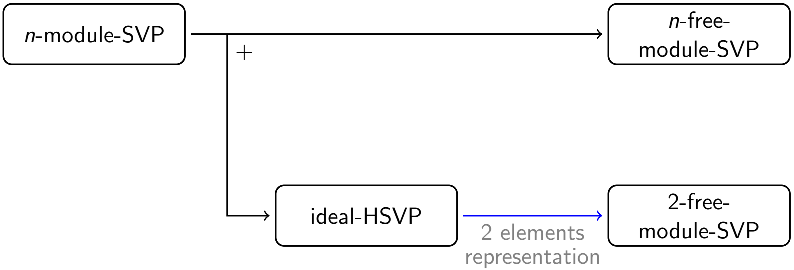
Summary of the reductions

n-module-SVP

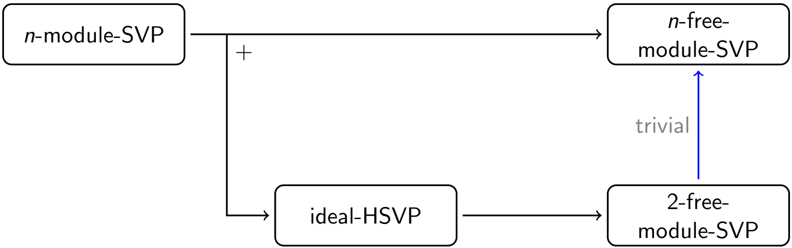
Summary of the reductions



Summary of the reductions



Summary of the reductions



Summary of the reductions

Combining the reductions

Theorem

*Let $\gamma \geq 1$ and $n \geq 2$ be an integer. For any $\gamma' \geq 2 \cdot \gamma^3 \cdot \Delta_K^{3/2d}$, there is a probabilistic polynomial-time reduction from solving (γ', n)-module-SVP in K^n to solving (γ, n)-***free***-module-SVP in K^n.*

Take-away messages:

- Free modules are no weaker than arbitrary modules for standard cryptographic algorithmic problems (SVP, HSVP, CVP).
- The framework seems quite flexible: it could be used for reduction for another algorithmic problem (e.g., SIVP, uSVP, BDD).

One application: de-quantising module-LLL

Module-LLL: provide an extension of the LLL algorithm to lattices over $\mathcal{O}_K$.

- *Quantum* algorithm for any number field in [LPSW19]:
  - heuristic *quantum* algorithm for γ-SVP in rank-2 modules in polynomial-time **if** access to a CVP-oracle in a fixed lattice depending only on $\mathcal{O}_K$.
  - *classical* algorithm **if** the input module is free.
- *Classical* algorithm using the reductions provided in this work!

One application: de-quantising module-LLL

Module-LLL: provide an extension of the LLL algorithm to lattices over $\mathcal{O}_K$.

- *Quantum* algorithm for any number field in [LPSW19]:
  - heuristic *quantum* algorithm for γ-SVP in rank-2 modules in polynomial-time **if** access to a CVP-oracle in a fixed lattice depending only on $\mathcal{O}_K$.
  - *classical* algorithm **if** the input module is free.
- *Classical* algorithm using the reductions provided in this work!

Comment: the oracle-call still remains !

One application: de-quantising module-LLL

Module-LLL: provide an extension of the LLL algorithm to lattices over $\mathcal{O}_K$.

- *Quantum* algorithm for any number field in [LPSW19]:
  - heuristic *quantum* algorithm for γ-SVP in rank-2 modules in polynomial-time **if** access to a CVP-oracle in a fixed lattice depending only on $\mathcal{O}_K$.
  - *classical* algorithm **if** the input module is free.
- *Classical* algorithm using the reductions provided in this work!

Comment: the oracle-call still remains !

Thank you!