SVP algorithms. BKZ

Elena Kirshanova

Technology Innovation Institute, Abu Dhabi, UAE

CIMPA Summer School, Rabat

Links

These slides are available here:

https://crypto-kantiana.com/elena.kirshanova/teaching/ssRabat/SVP_Rabat.pdf

Links II

Exercises, labs are available on the webpage:

https://crypto-kantiana.com/elena.kirshanova/teaching/summerschoolRabat2023.html

Agenda

- Today: Lectures
- Tomorrow: Exercises
- Friday: Labs

Labs

- For Lab1 you need to install FPyLLL
  `https://github.com/fplll/fpylll`
- It is available via SageMathCell and CoCalc (select a Jupyter notebook with a Sage kernel)
- For Lab2 and Lab3 you need Sage on your machine (Lab2 is checked via automated tests)
- Labs can be solved in teams of max 3 people
- Try to install FPyLLL or play with it in CoCalc

Prize

The fastest team to obtain correct* solutions/implementations gets an unforgettable prize!

*The correctness will be judged by the lecturer

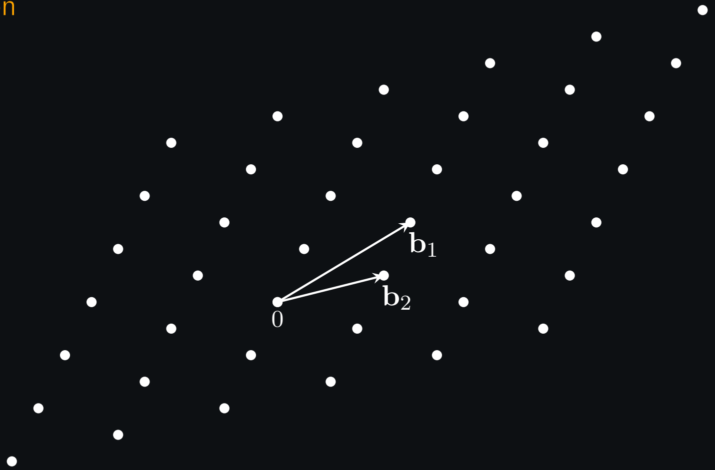
Content of the lectures

1. The shortest vector problem
2. Kannan-Finke-Pohst Enumeration algorithm
3. Sieving algorithm
4. Block Korkine-Zolotarev reduction
5. Solving LWE with BKZ

Part I

The shortest vector problem

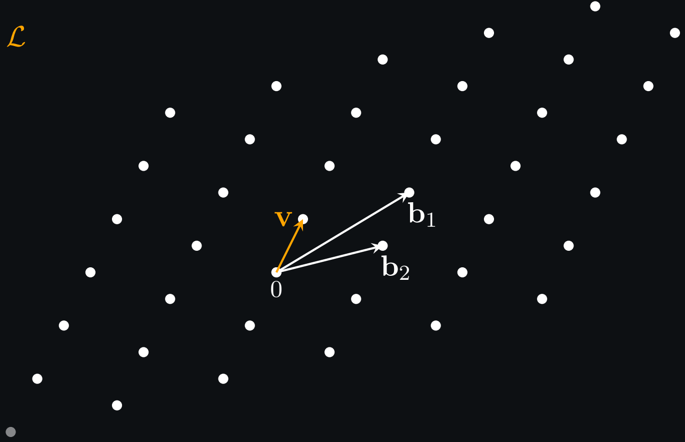
A lattice: definition



A lattice is the set $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for some linearly independent $\mathbf{b}_i$'s.

For us, $\mathbf{b}_i \in \mathbb{Z}^n$ and $\mathcal{L}$ is full-rank.

Short vectors in $\mathcal{L}$



The Shortest Vector Problem (SVP) asks to find non-zero $\mathbf{v}$ of minimal Euclidean length.

We do not know $||\mathbf{v}||$ in general, but for any n-rank $\mathcal{L}$:

$$||\mathbf{v}_{\text{shortest}}|| \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n} \quad \text{(Minkowski's bound)}$$

Hardness of SVP (small-order terms are omitted)

$$||\mathbf{v}_{\text{shortest}}|| \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

Approximate SVP asks to find $\mathbf{v}_{\text{short}}$:

$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

Hardness of SVP (small-order terms are omitted)

$$||\mathbf{v}_{\text{shortest}}|| \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

Approximate SVP asks to find $\mathbf{v}_{\text{short}}$:

$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

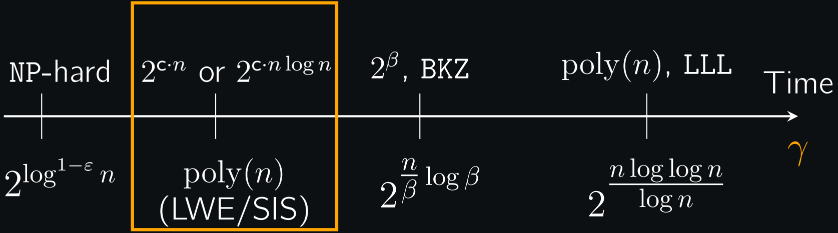
Time	NP-hard	$2^{c \cdot n}$ or $2^{c \cdot n \log n}$	2^{β}, BKZ	$\mathrm{poly}(n)$, LLL
γ	$2^{\log^{1-\varepsilon} n}$	$\mathrm{poly}(n)$ (LWE/SIS)	$2^{\frac{n}{\beta}\log\beta}$	$2^{\frac{n \log\log n}{\log n}}$

Hardness of SVP (small-order terms are omitted)

$$||\mathbf{v}_{\text{shortest}}|| \leq \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

Approximate SVP asks to find $\mathbf{v}_{\text{short}}$:

$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$



Practical Algorithms for SVP

- Enumeration
- Sieving (Provable/Heuristic)

Practical Algorithms for SVP

- Enumeration

$$\text{Time} = 2^{((1/2e)+o(1))n\log n} \qquad \text{Memory} = \mathrm{poly}(n)$$

✓ Lots of improvements for the $o(n\log n)$-term
✓ (Somewhat) easy to parallelize

- Sieving (Provable/Heuristic)

Practical Algorithms for SVP

- Enumeration

$$\text{Time} = 2^{((1/2e)+o(1))n\log n} \qquad \text{Memory} = \mathrm{poly}(n)$$

✓ Lots of improvements for the $o(n\log n)$-term
✓ (Somewhat) easy to parallelize

- Sieving (Provable/Heuristic)

$$\text{Time} = 2^{(2.465+o(1))n} \qquad \text{Memory} = 2^{(1.325+o(1))n}$$

$$\text{Time} = 2^{(0.292+o(1))n} \qquad \text{Memory} = 2^{(0.2075+o(1))n}$$

✓ Big $o(n)$-factors
✓ Parallelization is painful
✓ Time-memory trade-offs exist

Part II

Kannan-Finke-Pohst Enumeration algorithm

Enumeration algorithm for SVP: main idea

Idea: enumerate all lattice vector within a ball of certain radius k.

1. INPUT: basis $B = QR$, $R \in \mathbb{R}^{n \times n}$ – R-factor

Enumeration algorithm for SVP: main idea

Idea: enumerate all lattice vector within a ball of certain radius k.

1. INPUT: basis $B = QR$, $R \in \mathbb{R}^{n \times n}$ – R-factor
2. Set $k = \|\mathbf{b}_1\|$ – a bound
3. Let $\mathbf{x} \in \mathbb{Z}^n$ be the coefficient vector of $\mathbf{b} = B\mathbf{x}$. Then

$$\|B\mathbf{x}\|^2 = \|R\mathbf{x}\|^2 = \| \left(\sum_{i=1}^{n} r_{1,i} x_i, \sum_{i=2}^{n} r_{2,i} x_i, \ldots, r_{n,n} x_n \right) \|^2 = \sum_{j=1}^{n} \left(\sum_{i \geq j} r_{j,i} x_i \right)^2 .$$

Enumeration algorithm for SVP: main idea

Idea: enumerate all lattice vector within a ball of certain radius k.

1. INPUT: basis $B = QR$, $R \in \mathbb{R}^{n \times n}$ – R-factor
2. Set $k = \|\mathbf{b}_1\|$ – a bound
3. Let $\mathbf{x} \in \mathbb{Z}^n$ be the coefficient vector of $\mathbf{b} = B\mathbf{x}$. Then

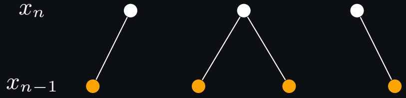
$$\|B\mathbf{x}\|^2 = \|R\mathbf{x}\|^2 = \| \left(\sum_{i=1}^{n} r_{1,i}x_i, \sum_{i=2}^{n} r_{2,i}x_i, \ldots, r_{n,n}x_n \right) \|^2 = \sum_{j=1}^{n} \left(\sum_{i \geq j} r_{j,i}x_i \right)^2.$$

We are going to enumerate x_i's for $i = n, \ldots, 1$, keeping the value $\sum_{j=1}^{n} \left(\sum_{i \geq j} r_{j,i}x_i \right)^2$ bounded.

Enumeration algorithm for SVP

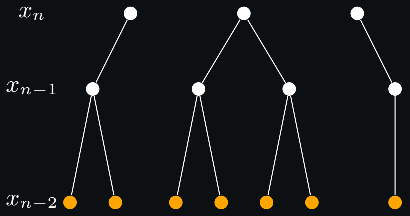
1. Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| < \frac{k}{r_{n,n}}$.

Enumeration algorithm for SVP



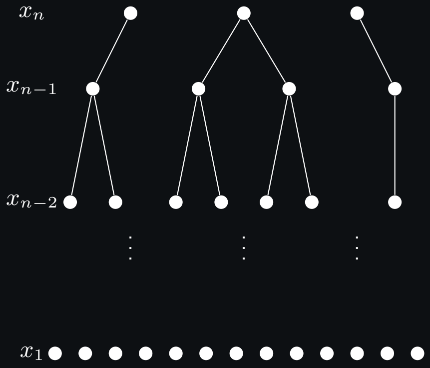
1. Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| < \frac{k}{r_{n,n}}$.
2. Fix x_n. Take all $x_{n-1} \in \mathbb{Z}$ s.t.
   $\left| x_{n-1} + \frac{r_{n-1,n}}{r_{n-1,n-1}} x_n \right| < \left(\frac{k^2 - (r_{n,n} x_n)^2}{r_{n-1,n-1}} \right)^{1/2}$

Enumeration algorithm for SVP



1. Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| < \frac{k}{r_{n,n}}$.
2. Fix x_n. Take all $x_{n-1} \in \mathbb{Z}$ s.t.
   $\left| x_{n-1} + \frac{r_{n-1,n}}{r_{n-1,n-1}} x_n \right| < \left(\frac{k^2 - (r_{n,n} x_n)^2}{r_{n-1,n-1}} \right)^{1/2}$
3. For fixed x_n, x_{n-1}, take all 'legitimate' $x_{n-2} \in \mathbb{Z}$

Enumeration algorithm for SVP



1. Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| < \frac{k}{r_{n,n}}$.
2. Fix x_n. Take all $x_{n-1} \in \mathbb{Z}$ s.t.
$\left| x_{n-1} + \frac{r_{n-1,n}}{r_{n-1,n-1}} x_n \right| < \left(\frac{k^2 - (r_{n,n} x_n)^2}{r_{n-1,n-1}} \right)^{1/2}$
3. For fixed x_n, x_{n-1}, take all 'legitimate' $x_{n-2} \in \mathbb{Z}$
4. Continue all the way to x_1's.

Complexity

Theorem

The size of the enumeration tree of the above algorithm that receives on input an LLL-reduced basis B of an n-dimensional lattice is $2^{(n^2)}$. It can be traversed using $\mathrm{poly}(n)$ memory.

A proof to be shown in TD.

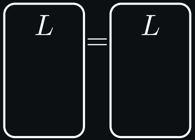
One can tweak the algorithm by making the smallest $r_{i,i}$'s larger. This gives the enumeration tree to the size $2^{\frac{n \log n}{2e} + o(n)}$, [Kan83,HanSte07]

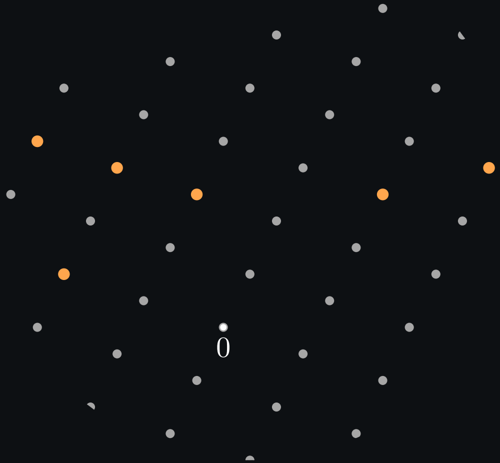
Sieving

https://sites.google.com/a/x.bestledlights.cf/a223/1-32772214555

Basic 2-Sieve (Nguyen-Vidick sieve)

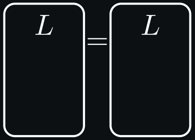
Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors

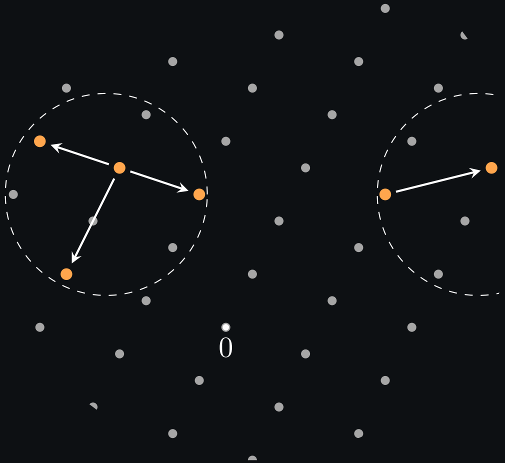




Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors

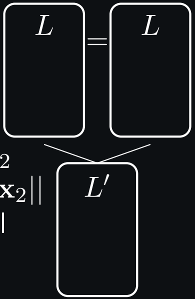


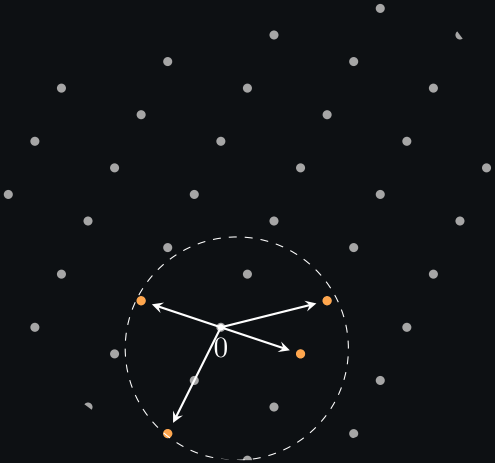


Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors

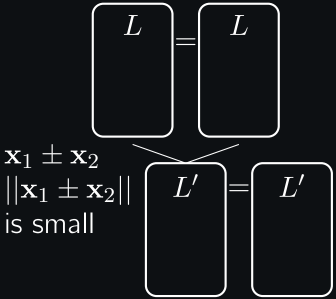
$\mathbf{x}_1 \pm \mathbf{x}_2$
$||\mathbf{x}_1 \pm \mathbf{x}_2||$
is small

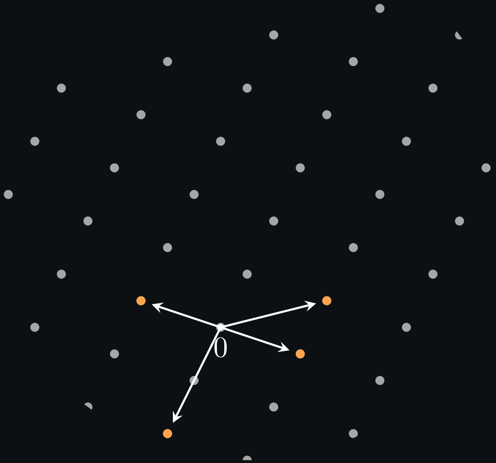




Basic 2-Sieve (Nguyen-Vidick sieve)

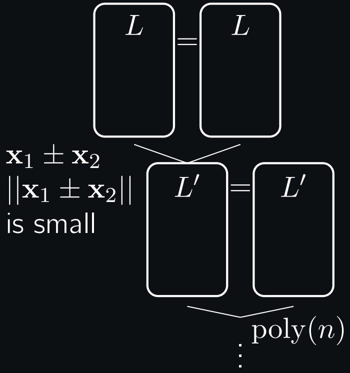
Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors

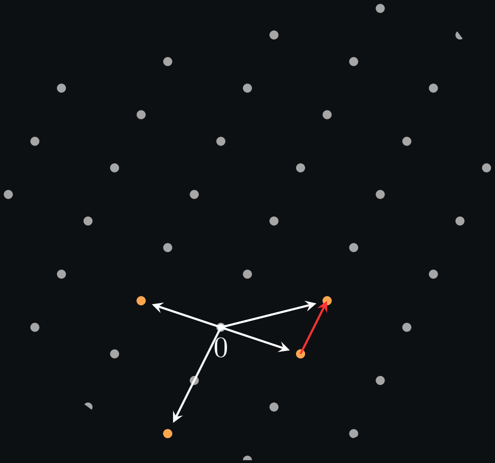




Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors





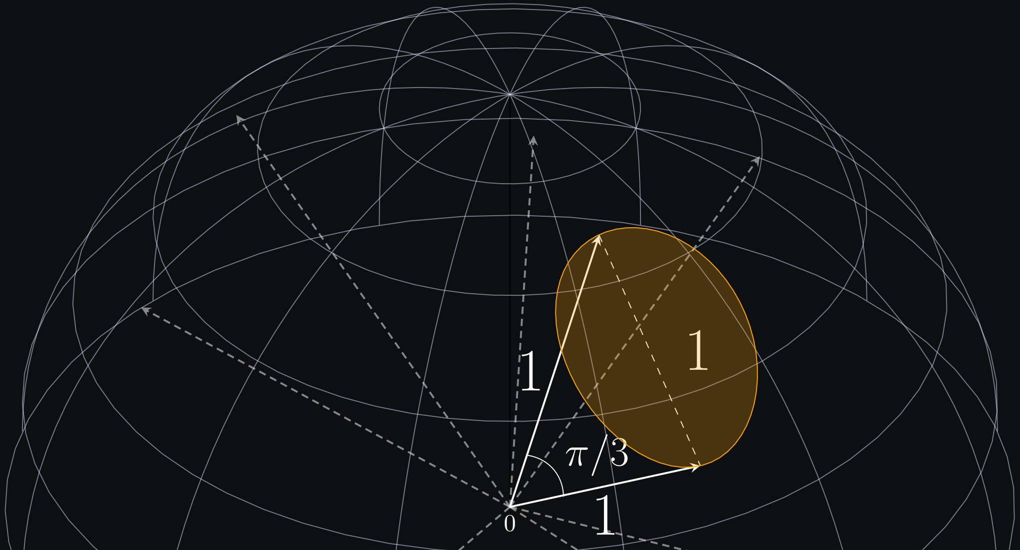
How large $|L|$ should be?

How large $|L|$ should be?

Assumption: vectors (normalized) in $|L|$ are uniform iid on S^{n-1}.

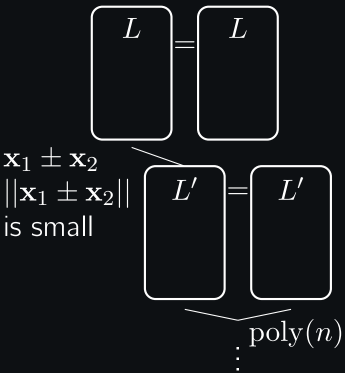
How large $|L|$ should be?

Assumption: vectors (normalized) in $|L|$ are uniform iid on S^{n-1}.



Basic 2-Sieve (Nguyen-Vidick sieve)

Main idea in all sieving algorithms: saturate space with enough lattice vectors so that their sums give short(er) vectors



$$|L| = \left(\sqrt{\frac{3}{4}}\right)^{-n} = 2^{0.2075n}$$

$$T(\text{2-Sieve}) = |L|^2 = 2^{0.415n}$$

0

SVP: conclusions

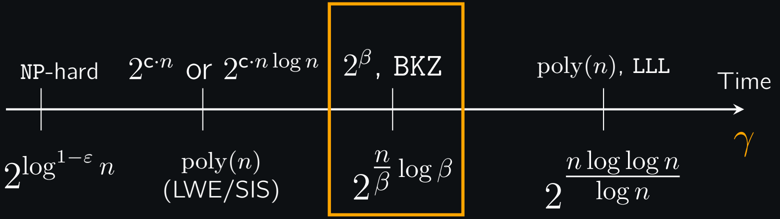
- Best known SVP algorithm require at least exponential (in lattice dimension) time
- We do not know how to use the additional structure to significantly speed up SVP algorithms for algebraic lattices

Open questions

- SVP in ℓ_∞ norm, algebraic SVP
- Precise complexity of SVP taking into account memory costs
- Quantum speed ups for SVP/LWE/SIS?

Part IV

Block Korkine-Zolotarev (BKZ) algorithm



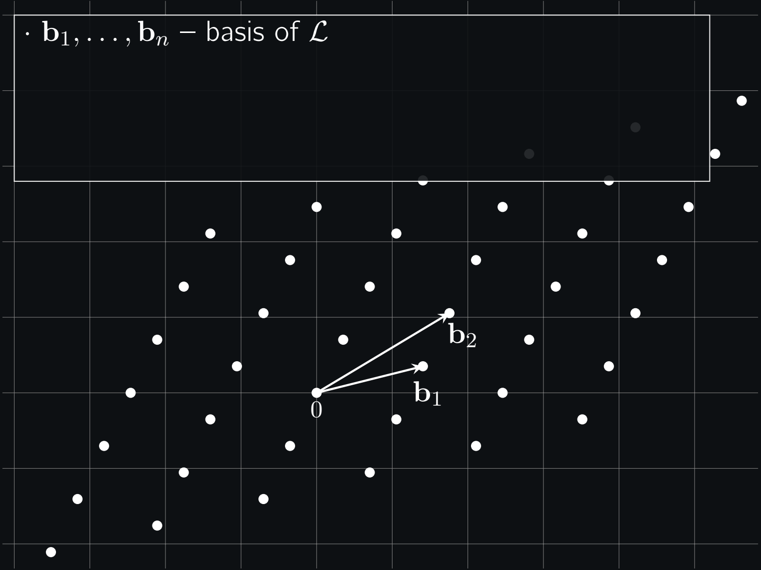
Small improvement at a time

- We never call an SVP oracle on an non-preprocessed basis
- Having a “better quality” basis of $\mathcal{L}$ is beneficial for most (all?) algorithms
- We try to gradually improve the “quality” of a basis

Quality - length of Gram-Schmidt vectors

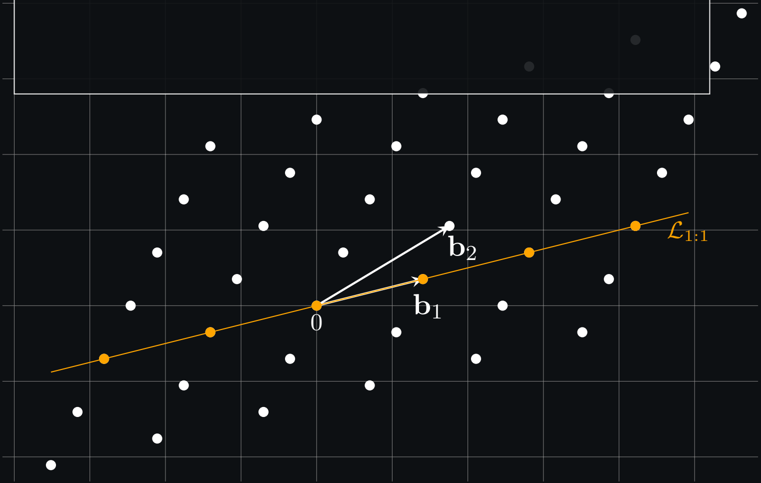
Projected lattice

- $\mathbf{b}_1, \ldots, \mathbf{b}_n$ – basis of $\mathcal{L}$



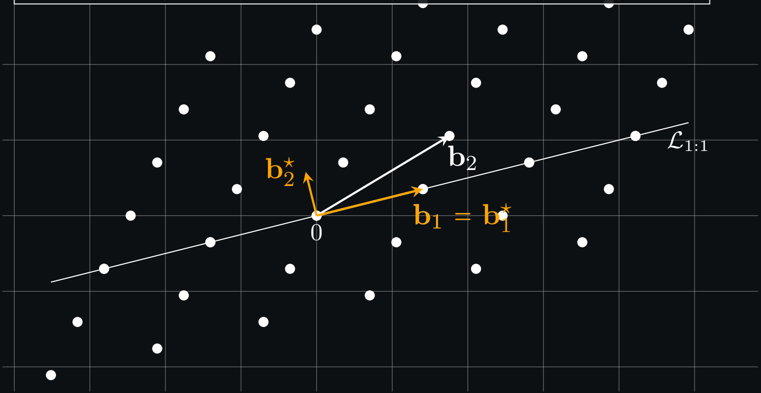
Projected lattice

- $\mathbf{b}_1, \ldots, \mathbf{b}_n$ – basis of $\mathcal{L}$
- $\mathcal{L}_{1:j}$ is the lattice spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_j$



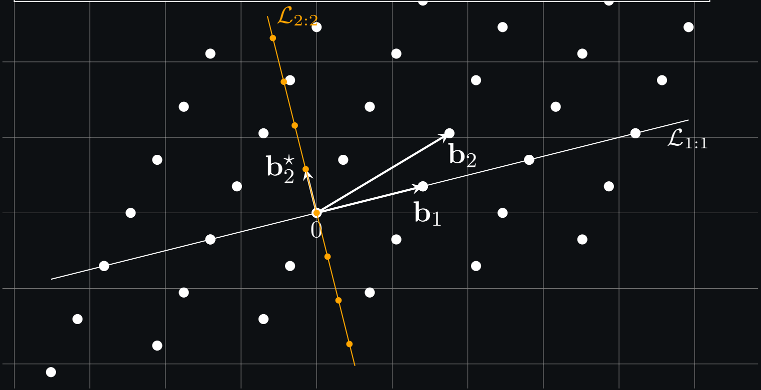
Projected lattice

- $\mathbf{b}_1, \ldots, \mathbf{b}_n$ – basis of $\mathcal{L}$
- $\mathcal{L}_{1:j}$ is the lattice spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_j$.
- $\mathbf{b}_i^\star$ is the projection of $\mathbf{b}_i$ on $\mathcal{L}_{1:i-1}^\perp$. (GSO)



Projected lattice

- $\mathbf{b}_1, \ldots, \mathbf{b}_n$ – basis of $\mathcal{L}$
- $\mathcal{L}_{1:j}$ is the lattice spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_j$.
- $\mathbf{b}_i^\star$ is the projection of $\mathbf{b}_i$ on $\mathcal{L}_{1:i-1}^\perp$ (GSO)
- $\mathcal{L}_{i:j}$ is the orthogonal projection of $\mathcal{L}_{1:j}$ on $\mathcal{L}_{1:i-1}^\perp$.



BKZ (simplified)

Notation: $\mathcal{L}_{[\ell\,;\,r]}$ - orthogonal projection of $\mathcal{L}_{1:r}$ on $\mathcal{L}^{\perp}_{1:\ell-1}$

Input: $B = (\mathbf{b}_i), \beta$

  for $k = 2 \ldots n-1$ **do**

    $\mathbf{b} \leftarrow \mathsf{SVP}(\mathcal{L}_{[k\,;\,\min\{k+\beta-1,n\}]})$

  end for

  if $\mathbf{b}$ is "short enough" **then**

    Insert $\mathbf{b}$ into B

    Remove lin. dependencies

  end if

$$\begin{pmatrix} | & | & | & & | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \ldots & \mathbf{b}_\beta & \mathbf{b}_{\beta+1} & \ldots & \mathbf{b}_n \\ | & | & | & & | & | & & | \end{pmatrix}$$

BKZ (simplified)

Notation: $\mathcal{L}_{[\ell\,;\,r]}$ - orthogonal projection of $\mathcal{L}_{1:r}$ on $\mathcal{L}_{1:\ell-1}^{\perp}$

Input: $B = (\mathbf{b}_i), \beta$
 for $k = 2$ **do**
  $\mathbf{b} \leftarrow \mathsf{SVP}(\mathcal{L}_{[2\,;\,\min\{\beta+1,n\}]})$
 end for
 if $\mathbf{b}$ is "short enough" **then**
  Insert $\mathbf{b}$ into B
  Remove lin. dependencies
 end if

$$\begin{pmatrix} | & | & | & & | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \dots & \mathbf{b}_\beta & \mathbf{b}_{\beta+1} & \dots & \mathbf{b}_n \\ | & | & | & & | & | & & | \end{pmatrix}$$

$$\underbrace{\qquad\qquad\qquad}_{\text{SVP}}$$

- BKZ runs this FOR-loop while there has been a change in the basis
- one run of this FOR-loop is called a tour

BKZ (simplified)

Notation: $\mathcal{L}_{[\ell\,;\,r]}$ - orthogonal projection of $\mathcal{L}_{1:r}$ on $\mathcal{L}^{\perp}_{1:\ell-1}$

Input: $B = (\mathbf{b}_i), \beta$
  for $k = 3$ **do**
    $\mathbf{b} \leftarrow \text{SVP}(\mathcal{L}_{[3\,;\,\min\{\beta+2,n\}]})$
  end for
  if $\mathbf{b}$ is "short enough" **then**
    Insert $\mathbf{b}$ into B
    Remove lin. dependencies
  end if

$$\begin{pmatrix} | & | & | & & | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \dots & \mathbf{b}_\beta & \mathbf{b}_{\beta+1} & \dots & \mathbf{b}_n \\ | & | & | & & | & | & & | \end{pmatrix}$$

SVP

- BKZ runs this FOR-loop while there has been a change in the basis
- one run of this FOR-loop is called a tour

BKZ (simplified)

Notation: $\mathcal{L}_{[\ell\ ;\ r]}$ - orthogonal projection of $\mathcal{L}_{1:r}$ on $\mathcal{L}^{\perp}_{1:\ell-1}$

Input: $B = (\mathbf{b}_i), \beta$

  for $k = 4$ **do**

    $\mathbf{b} \leftarrow \mathsf{SVP}(\mathcal{L}_{[4\ ;\ \min\{\beta+3, n\}]})$

  end for

  if $\mathbf{b}$ is "short enough" **then**

    Insert $\mathbf{b}$ into B

    Remove lin. dependencies

  end if

$$\left(\begin{array}{c} | \\ \mathbf{b}_1 \\ | \end{array}\ \underbrace{\begin{array}{cccccc} | & | & & | & | \\ \mathbf{b}_2 & \mathbf{b}_3 & \dots & \mathbf{b}_\beta & \mathbf{b}_{\beta+1} \\ | & | & & | & | \end{array}}_{\text{SVP}}\ \begin{array}{cc} & | \\ \dots & \mathbf{b}_n \\ & | \end{array}\right)$$

- BKZ runs this FOR-loop while there has been a change in the basis
- one run of this FOR-loop is called a tour

BKZ (simplified)

Notation: $\mathcal{L}_{[\ell\,;\,r]}$ - orthogonal projection of $\mathcal{L}_{1:r}$ on $\mathcal{L}_{1:\ell-1}^{\perp}$

Input: $B = (\mathbf{b}_i), \beta$

for $k = 1 \ldots n-1$ **do**

  $\mathbf{b} \leftarrow \mathsf{SVP}(\mathcal{L}_{[k\,;\,\min\{k+\beta-1,n\}]})$

end for

if $\mathbf{b}$ is "short enough" **then**

  Insert $\mathbf{b}$ into B

  Remove lin. dependencies

end if

$$\begin{pmatrix} | & | & | & & | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \ldots & \mathbf{b}_\beta & \mathbf{b}_{\beta+1} & \ldots & \mathbf{b}_n \\ | & | & | & & | & | & & | \end{pmatrix}$$

SVP

- BKZ runs this FOR-loop while there has been a change in the basis
- one run of this FOR-loop is called a tour

BKZ: Output Quality and Runtime

- The running time of the algorithm is dominated by the SVP calls if we bound the number of tours by $\mathrm{poly}(n)$.
- This leads to the complexity $2^{\mathcal{O}(\beta)}$ when sieving is used for SVP and $2^{\mathcal{O}(\beta \log \beta)}$. Question: memory?
- The approximation factor achieved by BKZ is (see TD):

$$\|\mathbf{b}_1\| \leq \beta^{\frac{n-1}{\beta-1}} \lambda_1(L).$$

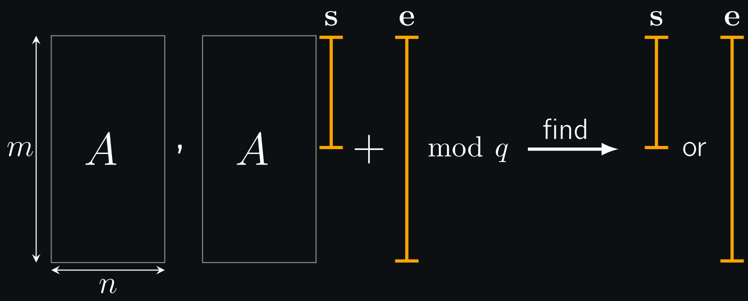
Time to show the demo...

TODO

Part V

Solving LWE with BKZ

LWE is BDD



- A defines the Construction-A lattice

$$\mathcal{L}_q(A) = A\mathbb{Z}_q^n + q\mathbb{Z}^m$$

- W.h.p., $\mathcal{L}_q(A)$ is of dim. m and $\det(\mathcal{L}_q(A)) = q^{m-n}$.
- $As + e \bmod q$ is a point near $\mathcal{L}_q(A)$ at distance $\Theta(\sqrt{m}\alpha q)$
- $(A, As + e)$ is a BDD instance on $\mathcal{L}_q(A)$ with $\gamma = \frac{q^{1-n/m}}{\alpha q}$

How do we solve BDD? Use an approxSVP algorithm! Kannan's Embedding

For a BDD instance $(\mathcal{L}, \mathbf{t})$, where B is a basis of $\mathcal{L}$, and c is a constant, let

$$B' = \begin{bmatrix} B & \mathbf{t} \\ \mathbf{0} & \mathsf{c} \end{bmatrix}$$

- Columns of B' are linearly independent
- Let $B\mathbf{x}$ be the solution
- For "properly" chosen c and $\mathbf{t}$ - sufficiently close to $\mathcal{L}$,

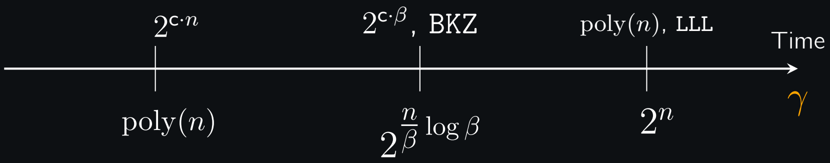
$$\begin{bmatrix} B & \mathbf{t} \\ \mathbf{0} & \mathsf{c} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = \begin{bmatrix} B\mathbf{x} - \mathbf{t} \\ -\mathsf{c} \end{bmatrix}$$

– is the shortest vector in $\mathcal{L}(B')$ (much shorter than any other $\mathbf{v} \in \mathcal{L}(B')$ non-parallel to it).

Kannan's embedding in pictures

Kannan's embedding in pictures

Hardness of LWE



For LWE parameters (n, m, q, α), $\gamma = \frac{q^{1-n/m}}{\alpha q}$

$$T(\mathsf{LWE}) = \exp\left(\mathsf{c}\cdot\frac{\lg q}{\lg^2\alpha}\lg\left(\frac{n\lg q}{\lg^2\alpha}\right)\cdot n\right),$$

where c is the constant in the exponent of SVP complexity, i.e., $T((\mathsf{SVP}))^{2^{\mathsf{c}\beta}}$.

This complexity is obtained by solving for β

$$2^{\frac{m}{\beta}\log\beta} = \frac{q^{1-n/m}}{\alpha q}$$

and choosing $m = \Omega(n)$ that minimizes the solution.

References

- [BDGL16] A. Becker, L. Ducas, N. Gama, T. Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving.
- [FP83] U. Fincke and M. Pohst. A procedure for determining algebraic integers of given norm.
- [HS07] Guillaume Hanrot and Damien Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm.
- [Kan83] Ravi Kannan. Improved algorithms for integer programming and related lattice problems.
- [Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming.
- [NV08] P. Nguyen, T. Vidick. Sieve algorithms for the shortest vector problem are practical.
- [S87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms.