Cryptanalysis of lattice-based constructions

Elena Kirshanova

at Post-Quantum Cryptography Summer School
Hammamet, Tunisia

Logistics

There will be two lectures on lattices:

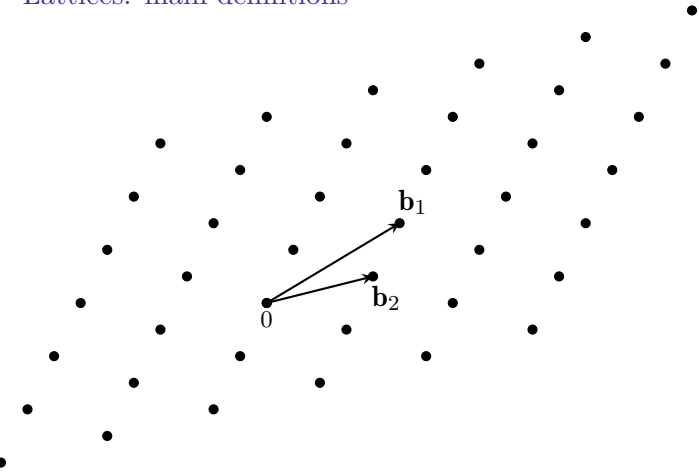
1. Lattice-based cryptographic constructions (yesterday)
2. Cryptanalysis of lattice-based constructions (now)

Lecture materials are available at
`https://elenakirshanova.github.io/teaching/summerschoolTunis2026.html`

Agenda

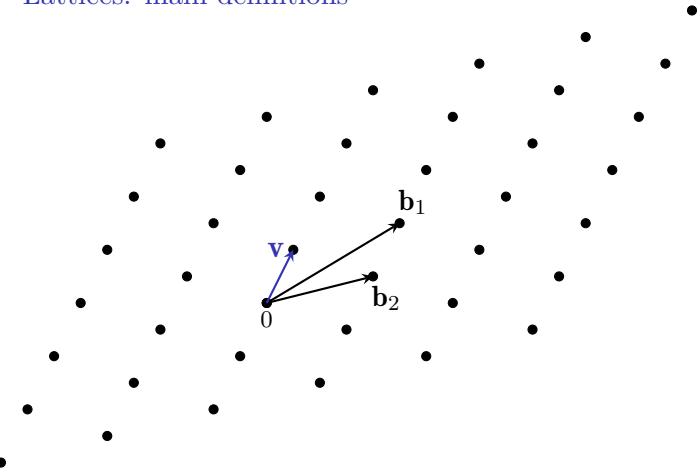
1. Lattices: basic definitions and hard problems
2. LWE as a lattice problem
3. SVP hardness
4. Demo
5. A few words about SIS

Lattices: main definitions



A lattice is a set $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for linearly independent $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– a basis of $\mathcal{L}$

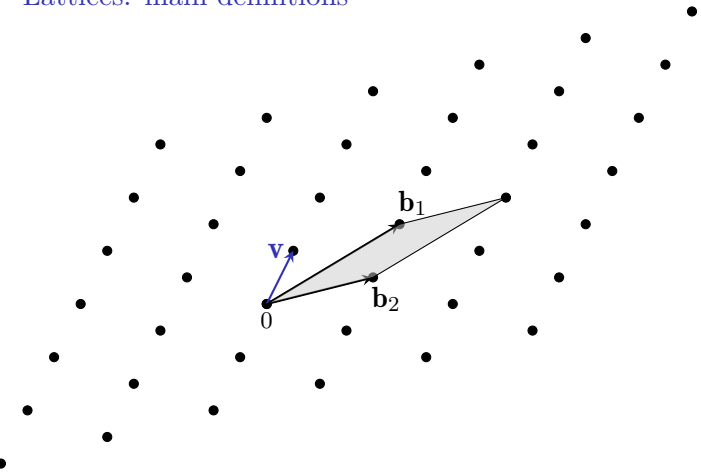


Minimum

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \mathbf{0}} \|\mathbf{v}\|$$

A lattice is a set $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for linearly independent $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– a basis of $\mathcal{L}$



Minimum

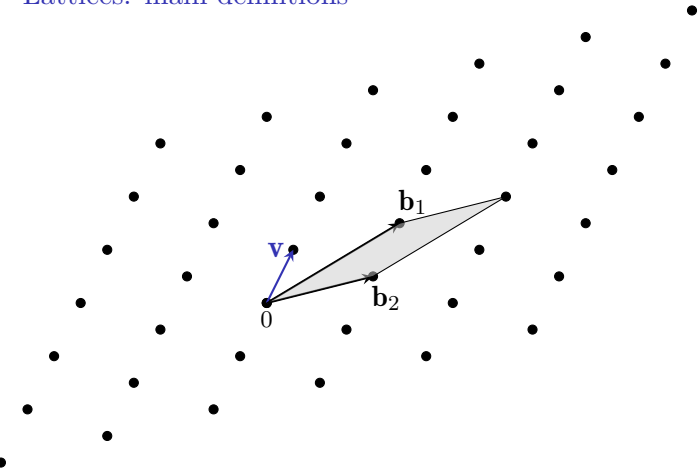
$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \mathbf{0}} \|\mathbf{v}\|$$

Determinant

$$\det(\mathcal{L}) = |\det(\mathbf{b}_i)_i|$$

A lattice is a set $\mathcal{L} = \{\sum_{i \le n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for linearly independent $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– a basis of $\mathcal{L}$



Minimum

$$\lambda_1(\mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L} \setminus \mathbf{0}} \|\mathbf{v}\|$$

Determinant

$$\det(\mathcal{L}) = |\det(\mathbf{b}_i)_i|$$

Minkowski bound

$$\lambda_1(\mathcal{L}) \leq \sqrt{n} \cdot \det(\mathcal{L})^{\frac{1}{n}}$$

A lattice is a set $\mathcal{L} = \{\sum_{i \leq n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for linearly independent $\mathbf{b}_i \in \mathbb{R}^n$

$\{\mathbf{b}_i\}_i$– a basis of $\mathcal{L}$

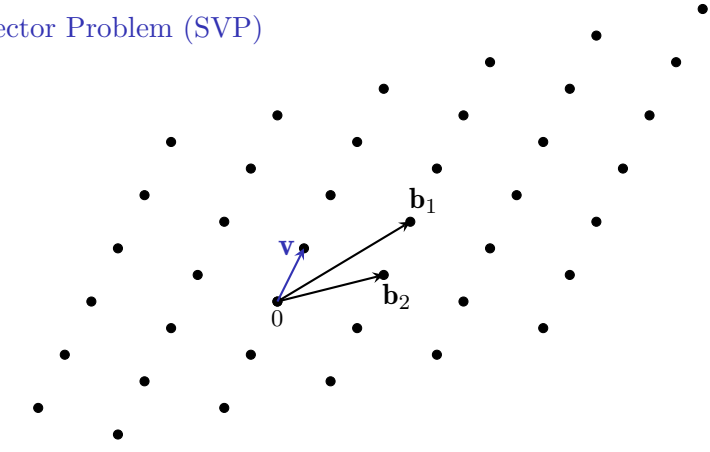
We write lattice basis $\{\mathbf{b}_i\}_i-$ into a matrix with $\{\mathbf{b}_i\}_i$ as columns

$$\begin{bmatrix} | & | & \dots & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \ddots & \mathbf{b}_n \\ | & | & \dots & | \end{bmatrix}$$

Bases are not unique but they are related via unitary transformations $B' = B \cdot U$ for $\det(U) = \pm 1$. Example:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} 4 & -3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$$

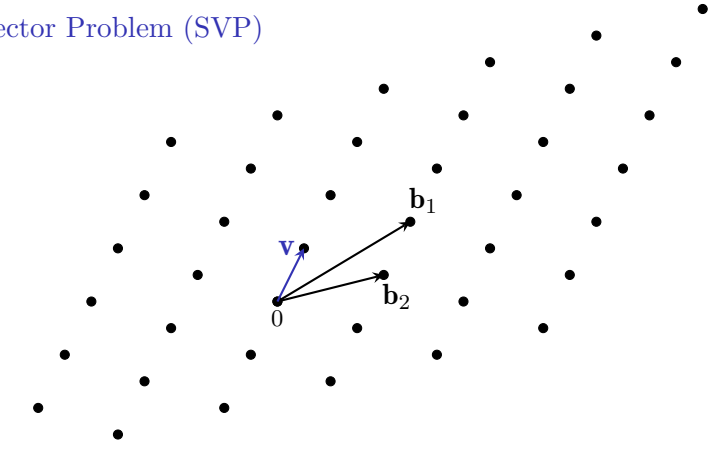
Shortest Vector Problem (SVP)



The Shortest Vector Problem (SVP) asks to find $\mathbf{v}_{\text{shortest}} \in \mathcal{L}$:

$$\|\mathbf{v}_{\text{shortest}}\| = \lambda_1(\mathcal{L})$$

Shortest Vector Problem (SVP)

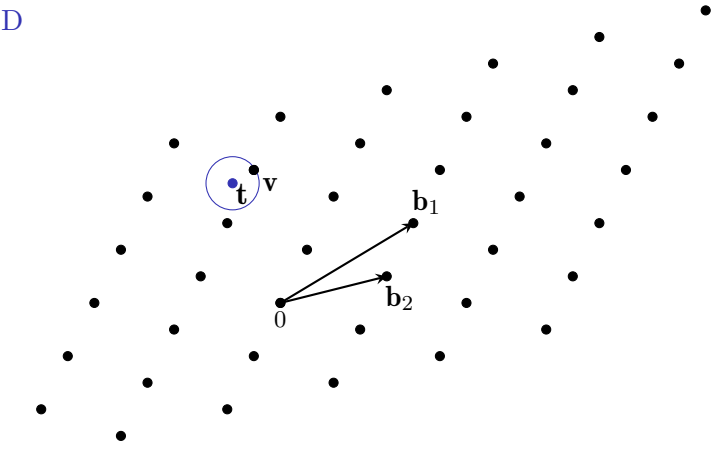


The Shortest Vector Problem (SVP) asks to find $\mathbf{v}_{\text{shortest}} \in \mathcal{L}$:

$$\|\mathbf{v}_{\text{shortest}}\| = \lambda_1(\mathcal{L})$$

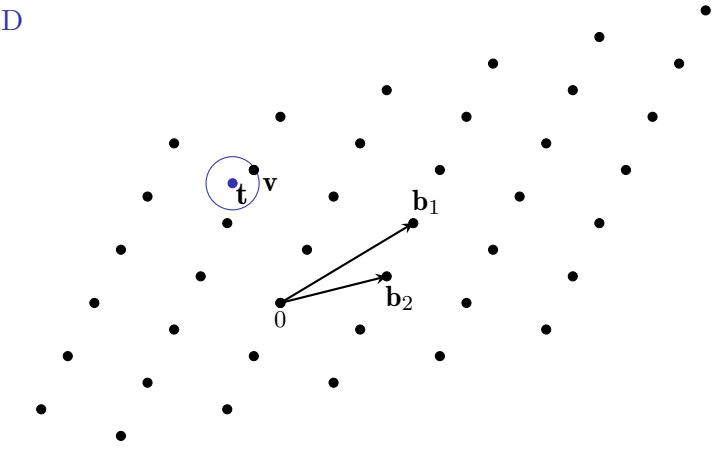
Often we are satisfied with an approximation (γ-SVP) to $\mathbf{v}_{\text{shortest}}$:

$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \lambda_1(\mathcal{L})$$



The Closest Vector Problem (CVP), given $\mathbf{t} \notin \mathcal{L}$ asks to find $\mathbf{v} \in \mathcal{L}$ s.t.

$$\|\mathbf{v} - \mathbf{t}\| \text{ is minimized over all } \mathbf{v} \in \mathcal{L}$$



The Closest Vector Problem (CVP), given $\mathbf{t} \notin \mathcal{L}$ asks to find $\mathbf{v} \in \mathcal{L}$ s.t.

$$\|\mathbf{v} - \mathbf{t}\| \text{ is minimized over all } \mathbf{v} \in \mathcal{L}$$

Often we have $\text{dist}(t, \mathcal{L}) \leq \frac{1}{\gamma}\lambda_1(\mathcal{L})$. This is the Bounded Distance Decoding Problem γ-BDD

Given a BDD instance $(\mathcal{L}, \mathbf{t})$, where $\mathcal{L}$ has basis B, consider for a constant c

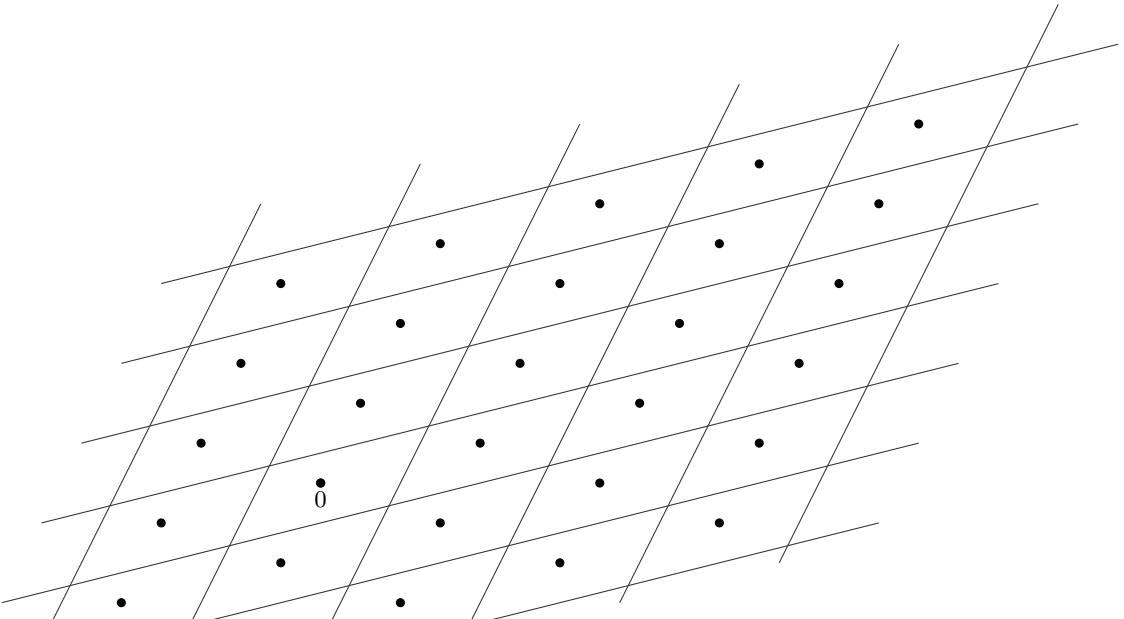
$$B' = \begin{bmatrix} B & \mathbf{t} \\ \mathbf{0} & \mathrm{c} \end{bmatrix}.$$

- the columns of B' are lin. independent
- If c is appropriately chosen and $\mathbf{t}$ is close enough to $\mathcal{L}$,

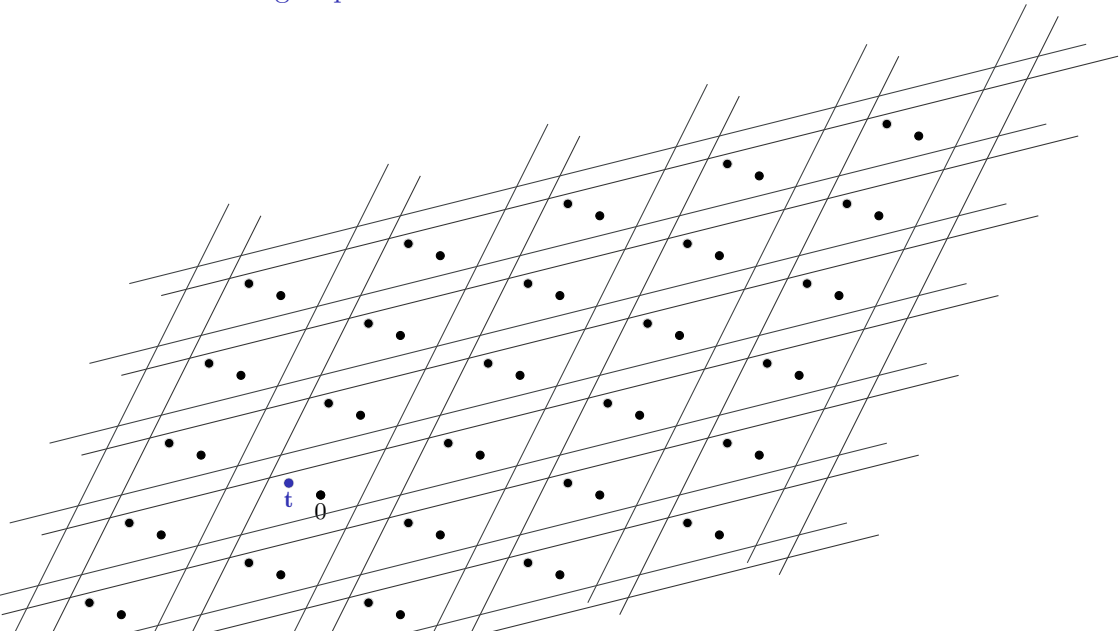
$$\begin{bmatrix} B & \mathbf{t} \\ \mathbf{0} & \mathrm{c} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x} \\ -1 \end{bmatrix} = \begin{bmatrix} B\mathbf{x} - \mathbf{t} \\ -\mathrm{c} \end{bmatrix}$$

is short (much shorter than any $\mathbf{v} \in \mathcal{L}(B')$ not parallel to it) in $\mathcal{L}(B')$.

Kannan's embedding in picture



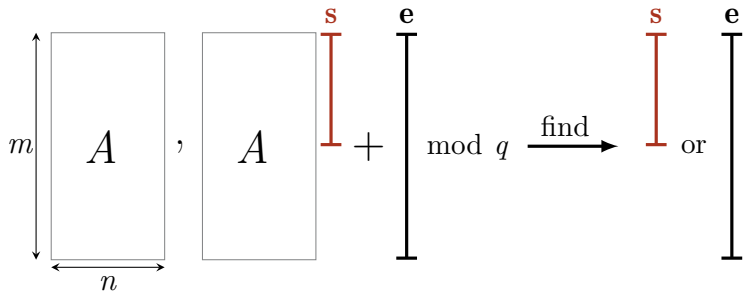
Kannan's embedding in picture



Why do we care about SVP?

- Combinatorial optimization (integer linear programming)
- Number theory
- Communication theory (solving BDD/CVP with Gaussian error)
- Cryptanalysis (RSA, side-channel attacks on ECSDA)
- Best known attacks (both in theory and in practice) against LWE and SIS invoke an SVP solver

LWE: remainder

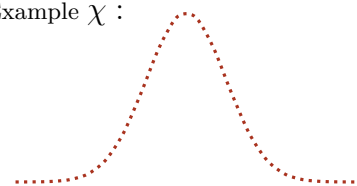


$$A \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$$

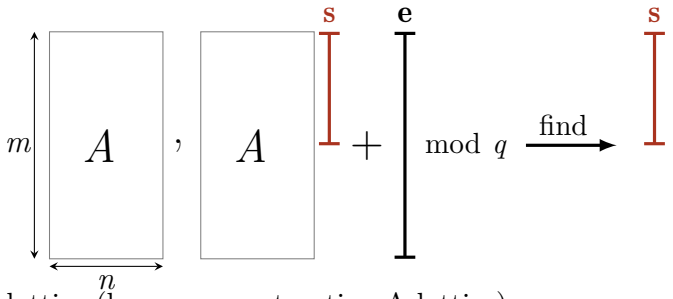
$$\mathbf{s} \leftarrow \chi^n$$

$$\mathbf{e} \leftarrow \chi^m$$

Example χ :



LWE as BDD



- A defines a lattice (know as 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 $\det(\mathcal{L}_q(A)) = q^{m-n}$
- $A\mathbf{s} + \mathbf{e} \bmod q$ is a point near $\mathcal{L}_q(A)$ at distance $\Theta(\|\mathbf{e}\|)$ (even though we do not know $\mathbf{e}$, we can approximate well $\|\mathbf{e}\|$)
- $(A, A\mathbf{s} + \mathbf{e})$ is a γ-BDD instance on $\mathcal{L}_q(A)$ with $\gamma = \frac{\sqrt{m}q^{1-n/m}}{\|\mathbf{e}\|}$

LWE as BDD

A basis of $\mathcal{L}_q(A)$ for $A \in \mathbb{Z}_q^{m \times n}$ is generated by the columns of

$$\begin{bmatrix} q\mathrm{Id}_m & -A \\ 0 & \mathrm{Id}_n \end{bmatrix}.$$

LWE as BDD

A basis of $\mathcal{L}_q(A)$ for $A \in \mathbb{Z}_q^{m \times n}$ is generated by the columns of

$$\begin{bmatrix} q\mathrm{Id}_m & -A \\ 0 & \mathrm{Id}_n \end{bmatrix}.$$

Kannan's embedding for $A, \mathbf{b}$ is a lattice generated by

$$\begin{bmatrix} q\mathrm{Id}_m & -A & \mathbf{b} \\ 0 & \mathrm{Id}_n & \mathbf{0} \\ 0 & \mathbf{0} & 1 \end{bmatrix}$$

Indeed, for $\mathbf{b} = A\mathbf{s} + \mathbf{e} - \mathbf{k}q$ for some $\mathbf{k} \in \mathbb{Z}^m$:

$$\begin{bmatrix} q\mathrm{Id}_m & -A & \mathbf{b} \\ 0 & \mathrm{Id}_n & \mathbf{0} \\ 0 & \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{k} \\ \mathbf{s} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{k} - A\mathbf{s} + b \\ \mathbf{s} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{e} \\ \mathbf{s} \\ 1 \end{bmatrix}$$

SVP hardness (small-order terms are omitted)

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

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

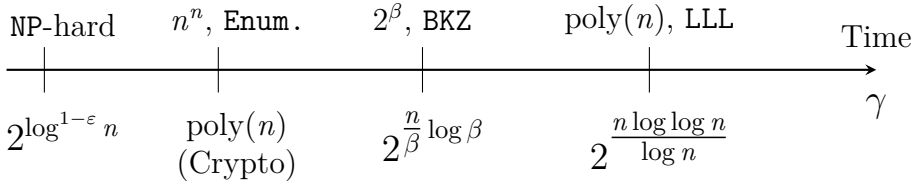
$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$

SVP hardness (small-order terms are omitted)

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

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

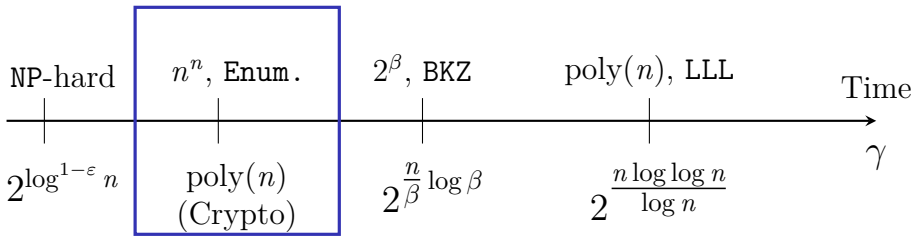
$$||\mathbf{v}_{\text{short}}|| \leq \gamma \cdot \sqrt{n} \cdot \det(\mathcal{L})^{1/n}$$



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

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

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



QR factorisation

For any matrix $B \in \mathbb{R}^{n \times m}$, there exists an orthogonal $Q \in \mathbb{R}^{n \times n}$ and an upper triangular $R \in \mathbb{R}^{m \times m}$ such that

$$B = Q \cdot \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Such decomposition is unique.

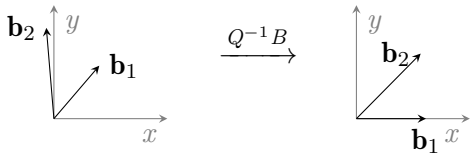


Figure: Q 'rotates' the basis B s.t. $\mathbf{b}_1$ is aligned with x-axis

Effectively, Q translates a basis B into a triangular 'basis' R.

QR factorisation

For any matrix $B \in \mathbb{R}^{n\times m}$, there exists an orthogonal $Q \in \mathbb{R}^{n\times n}$ and an upper triangular $R \in \mathbb{R}^{m\times m}$ such that

$$B = Q \cdot \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Such decomposition is unique.

Example:

$$\begin{bmatrix} -2 & 1 \\ 10 & 6 \end{bmatrix} = \begin{bmatrix} -0.196116135 & 0.980581 \\ 0.980580676 & 0.1961161 \end{bmatrix} \cdot \begin{bmatrix} 10.19803903 & 5.6873679 \\ 0 & 2.15727749 \end{bmatrix}$$

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

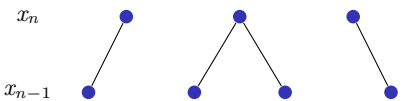
$$\|B\mathbf{x}\|^2 = \|R\mathbf{x}\|^2 = \| \left(\sum_{i=1}^{n} r_{1,i}x_i, \sum_{i=2}^{n} r_{2,i}x_i, \ldots, r_{n,n}x_n \right) \|^2 = \sum_{j=1}^{n} \left(\sum_{i \geq j} r_{j,i}x_i \right)^2 .$$

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

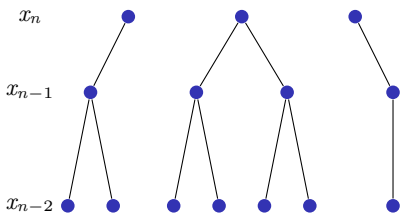
1. Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| < \frac{k}{r_{n,n}}$.

Enumeration algorithm for SVP



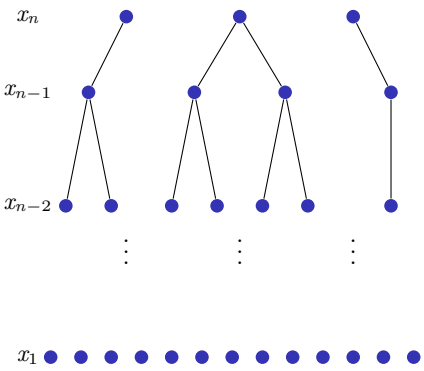
1. Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| < \frac{k}{r_{n,n}}$.
2. Fix x_n. Take all $x_{n-1} \in \mathbb{Z}$ s.t. $\left| x_{n-1} + \frac{r_{n-1,n}}{r_{n-1,n-1}} x_n \right| < \left(\frac{k^2 - (r_{n,n} x_n)^2}{r_{n-1,n-1}} \right)^{1/2}$

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



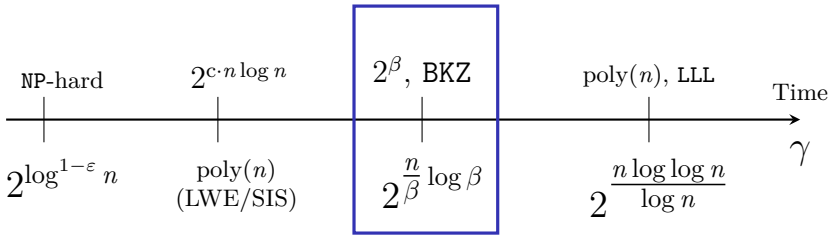
1. Take all $x_n \in \mathbb{Z}$ s.t. $|x_n| < \frac{k}{r_{n,n}}$.
2. Fix x_n. Take all $x_{n-1} \in \mathbb{Z}$ s.t. $\left| x_{n-1} + \frac{r_{n-1,n}}{r_{n-1,n-1}} x_n \right| < \left(\frac{k^2 - (r_{n,n} x_n)^2}{r_{n-1,n-1}} \right)^{1/2}$
3. For fixed x_n, x_{n-1}, take all 'legitimate' $x_{n-2}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

Complexity
[Kan83,HanSte07]

Theorem

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

Block Korkine-Zolotarev (BKZ) algorithm

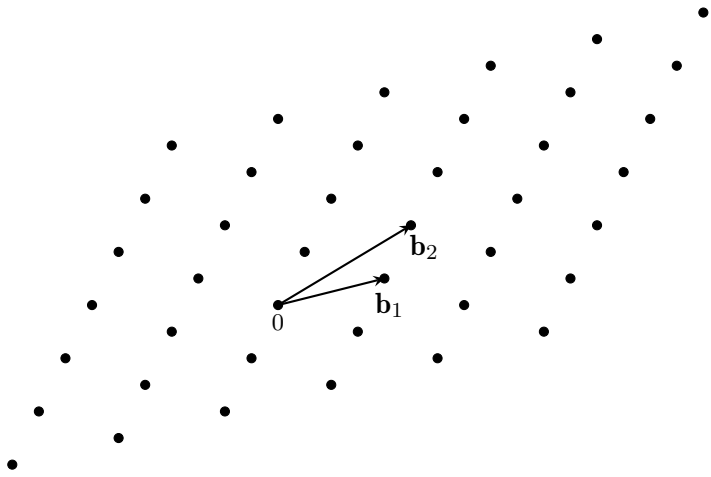


Small improvement at a time

- We never call an SVP oracle on an non-preprocessed basis
- Having a "better quality" basis of $\mathcal{L}$ is beneficial for all algorithms
- We try to gradually improve the "quality" of a basis, where "quality" means the decay rate of $r_{i,i}$ (the slower, the better the quality)

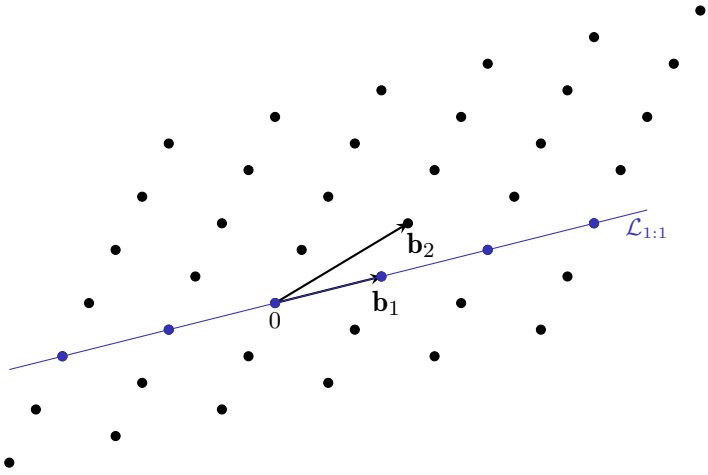
Projected lattice

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



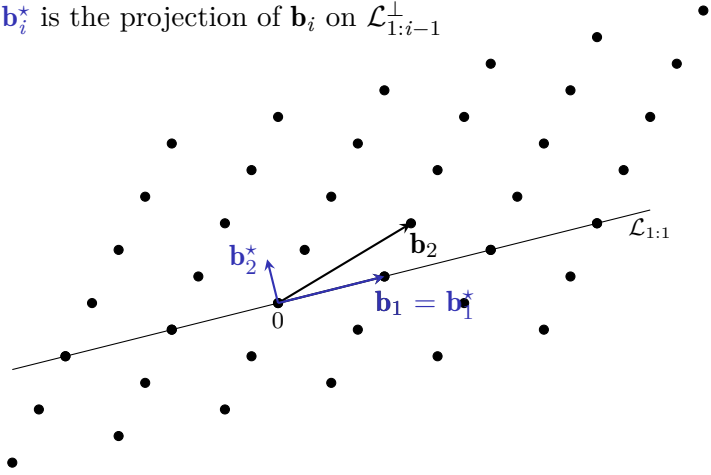
Projected lattice

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



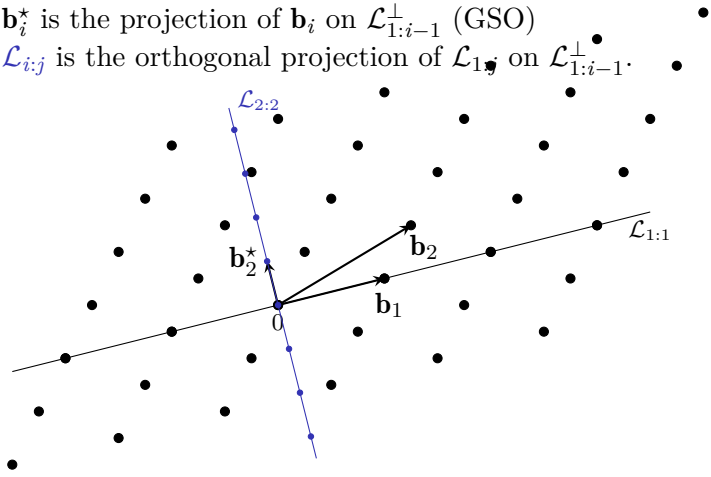
Projected lattice

· $\mathbf{b}_1, \ldots, \mathbf{b}_n$ – 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 $\mathcal{L}_{1:i-1}^\perp$



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 $\mathcal{L}_{1:\ell-1}^{\perp}$

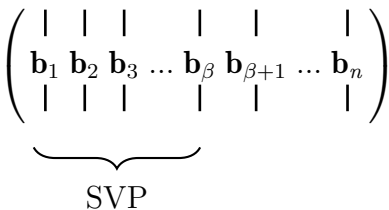
Input: $B = (\mathbf{b}_i), \beta$
  for $k = 2 \ldots n-1$ do
    $\mathbf{b} \leftarrow \mathrm{SVP}(\mathcal{L}_{[k\,;\,\min\{k+\beta-1,n\}]})$
    if $\mathbf{b}$ is "short enough" then
      Insert $\mathbf{b}$ into B
      Remove lin. dependencies
    end if
  end for

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

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

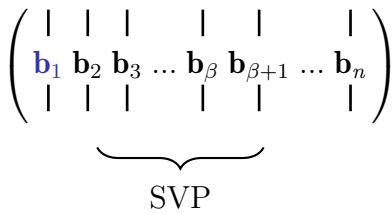


- BKZ runs this FOR-loop while there has been a change in the basis
- one run of this FOR-loop is called a tour

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

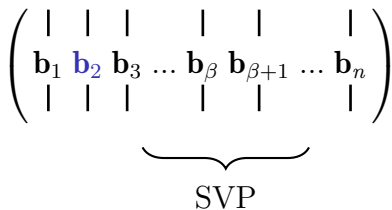


- BKZ runs this FOR-loop while there has been a change in the basis
- one run of this FOR-loop is called a tour

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 $k = 4$ do
    $\mathbf{b} \leftarrow \mathrm{SVP}(\mathcal{L}_{[4\,;\,\min\{\beta+3,n\}]})$
    if $\mathbf{b}$ is "short enough" then
      Insert $\mathbf{b}$ into B
      Remove lin. dependencies
    end if
  end for

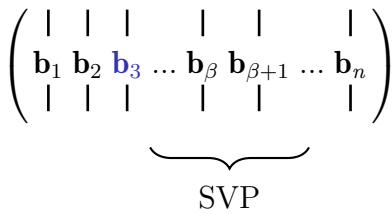


- BKZ runs this FOR-loop while there has been a change in the basis
- one run of this FOR-loop is called a tour

BKZ (simplified)

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

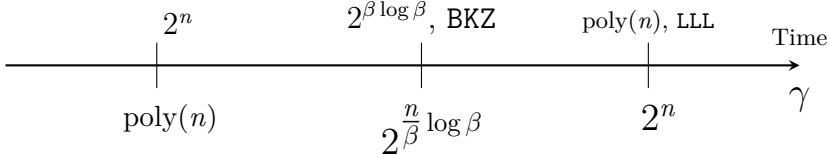
Input: $B = (\mathbf{b}_i), \beta$
  for $k = 1 \ldots n-1$ do
    $\mathbf{b} \leftarrow \mathrm{SVP}(\mathcal{L}_{[k\,;\,\min\{k+\beta-1,n\}]})$
  end for
  if $\mathbf{b}$ is "short enough" then
    Insert $\mathbf{b}$ into B
    Remove lin. dependencies
  end if



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

- The running time of the algorithm is dominated by the SVP calls if we bound the number of tours by $\text{poly}(n)$.
- This leads to the complexity $2^{\mathcal{O}(\beta \log \beta)}$ when we use Enumeration as SVP oracle.
- The approximation factor achieved by BKZ is:

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

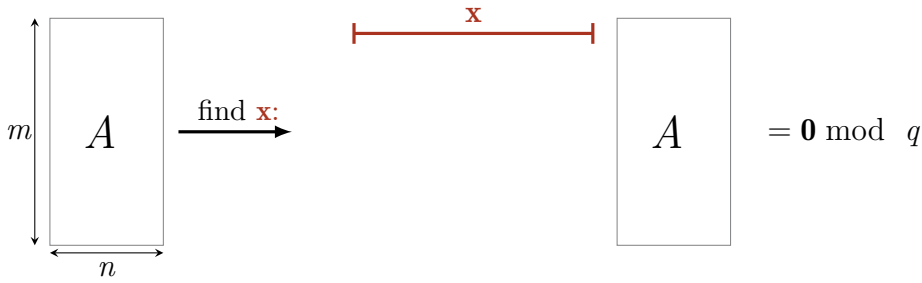


For LWE parameters (n, m, q), $\gamma = \frac{\sqrt{m} q^{1-n/m}}{\|\mathbf{e}\|}$, we find the appropriate β by solving

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

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

What about SIS?



SIS is (an average-case) approximate SVP

$$A \in \mathbb{Z}_q^{m \times n}, \quad \text{find a short } \mathbf{x}: \quad \mathbf{x}A = 0 \bmod q, \quad \|\mathbf{x}\| < B$$

- A defines the $\mathcal{L}_q^{\perp}(A)$ lattice:

$$\mathcal{L}_q^{\perp}(A) = \{\mathbf{y} \in \mathbb{Z}^m \ : \ \mathbf{y}A = 0 \bmod q\}$$

- SIS asks to find a short vector in $\mathcal{L}_q^{\perp}(A)$
- $\dim(\mathcal{L}_q^{\perp}(A)) = m$, $\det(\mathcal{L}_q^{\perp}(A)) = q^{\mathrm{rank}(A)}$
- $\lambda_1(\mathcal{L}_q^{\perp}(A)) = \sqrt{m} q^{n/m}$
- SIS is approx-SVP with approximation factor $\gamma = \frac{\lambda_1(\mathcal{L}_q^{\perp}(A))}{B}$

Resources

- On complexity of BKZ `https://blog.simons.berkeley.edu/2020/04/lattice-blog-reduction-part-i-bkz/`
- Implementations of lattice reductions:
  - FPLLL/FPyLLL `https://github.com/fplll/fplll`
  - Flatter `https://github.com/keeganryan/flatter`
  - BLASTer `https://github.com/ludopulles/BLASter`
- Some LWE challenge to solve `https://bochum-challeng.es/challenges/kyber`