Lattices&Codes: Algorithmic Connections and New Constructions

Elena Kirshanova

Technology Innovation Institute, Abu Dhabi, UAE

MWCC 2024

Agenda

Part I. Intro: Lattices&Codes

Part II. Sieving for codes

Part III (if time). Lattice constructions from codes

Part I

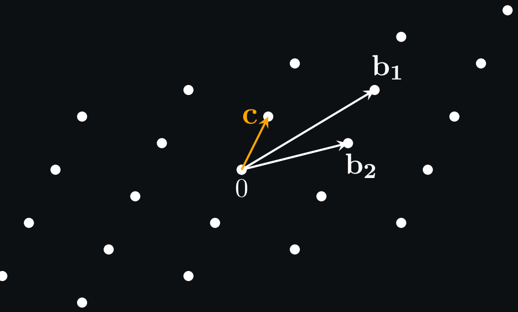
Intro: Lattices & Codes

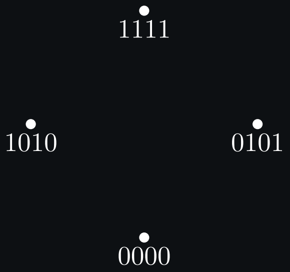
Lattices&Codes: definitions

$\mathcal{L}$	$\mathcal{C}$
Lattice $\mathcal{L}$ – additive group in $\mathbb{R}^n$	Code $\mathcal{C}$ – additive group in $\mathbb{F}_p^n$
Euclidean metric (ℓ_2)	ℓ_1 - metric
$\|\mathbf{v}\|_2$	$wt(\mathbf{v}) = \|\{i : \mathbf{v}[i] > 0\}\|$- Hamming weight

Lattices&Codes: definitions

$\mathcal{L}$	$\mathcal{C}$
Lattice $\mathcal{L}$ – additive group in $\mathbb{R}^n$	Code $\mathcal{C}$ – additive group in $\mathbb{F}_p^n$
Euclidean metric (ℓ_2)	ℓ_1 - metric
$\|\mathbf{v}\|_2$	$wt(\mathbf{v}) = \lvert\{i : \mathbf{v}[i] > 0\}\rvert$- Hamming weight
$\lambda_1(\mathcal{L})$ - shortest vector	$d(\mathcal{C})$-min. distance
Minkowski bound on $\lambda_1(\mathcal{L})$	Gilbert-Varshamov bound





Lattices&Codes: hard problems

$\mathcal{L}$ | $\mathcal{C}$

Finding a short vector

Given $A \in \mathbb{Z}_q^{(n-k)\times n}$, find $\mathbf{x} \in \mathbb{Z}_q^n$
s.t. $\|\mathbf{x}\| < B$ and $A\mathbf{x} = \mathbf{0} \bmod q$

Given $H \in \mathbb{F}_p^{(n-k)\times n}$, $\mathbf{s} \in \mathbb{F}_p^{n-k}$, find $\mathbf{e} \in \mathbb{F}_p^n$
s.t. $wt(\mathbf{e}) = \omega$ and $H\mathbf{e} = \mathbf{s}$

$$\boxed{A} \; \mathbf{x} = \mathbf{0} \bmod q \qquad \boxed{H} \; \mathbf{e} = \mathbf{s}$$

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

aka the SIS problem

Lattices&Codes: hard problems

$\mathcal{L}$ | $\mathcal{C}$

Finding a short vector

Given $A \in \mathbb{Z}_q^{(n-k)\times n}$, find $\mathbf{x} \in \mathbb{Z}_q^n$
s.t. $\|\mathbf{x}\| < B$ and $A\mathbf{x} = \mathbf{0} \bmod q$

Given $H \in \mathbb{F}_p^{(n-k)\times n}, \mathbf{s} \in \mathbb{F}_p^{n-k}$, find $\mathbf{e} \in \mathbb{F}_p^n$
s.t. $wt(\mathbf{e}) = \omega$ and $H\mathbf{e} = \mathbf{s}$

$$\boxed{A} \; \mathbf{x} = \mathbf{0} \bmod q$$

$$\boxed{H} \; \mathbf{s} \; \mathbf{e} = \mathbf{0}$$

-1

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

aka the SIS problem

$\mathcal{L}$	$\mathcal{C}$
Algorithms for finding a short vector:	
Enumeration algorithms	ISD algorithms
Sieving for lattice vectors	Sieving for codes[1]

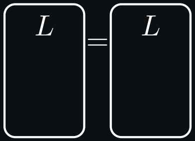
[1] Guo, Q., Johansson, T., Nguyen, V.: A new sieving-style information-set decoding algorithm. Ducas L, Esser A., Etinksi S., Kirshanova E.: Asymptotics and omprovements of sieving for codes

Part II

Sieving for codes

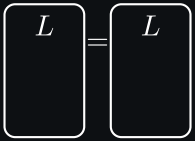
Idea of sieving in lattices

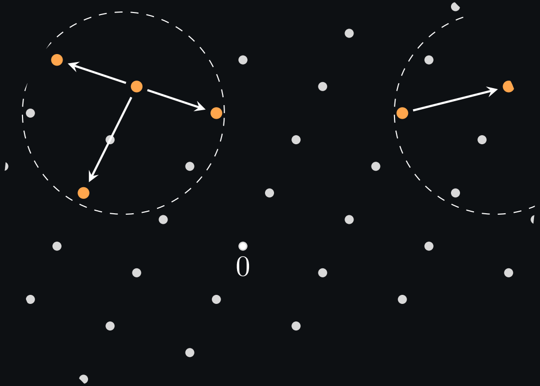
Saturate space with enough lattice vectors so that their sums give short(er) vectors



Idea of sieving in lattices

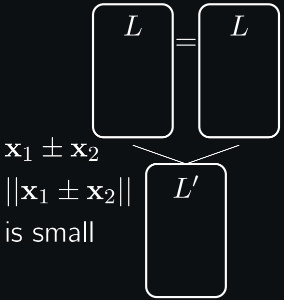
Saturate space with enough lattice vectors so that their sums give short(er) vectors

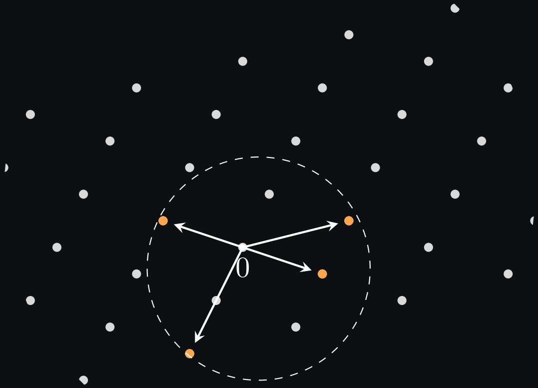




Idea of sieving in lattices

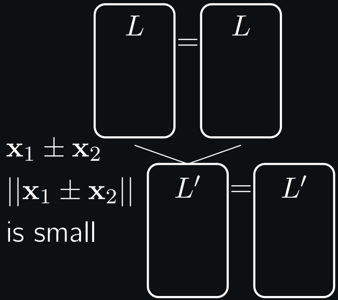
Saturate space with enough lattice vectors so that their sums give short(er) vectors





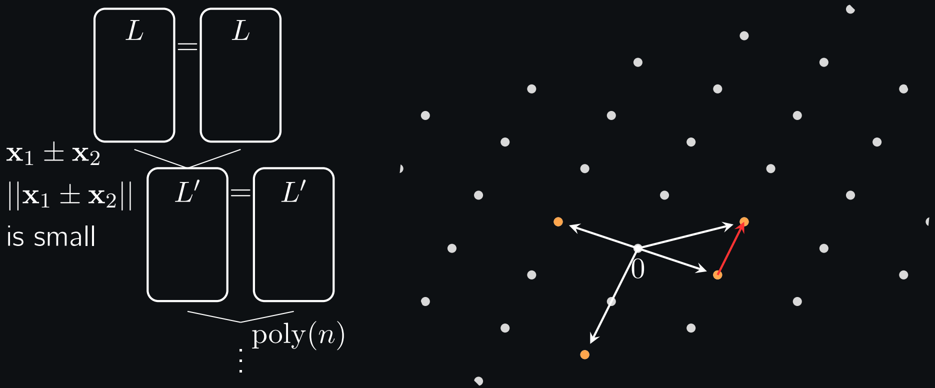
Idea of sieving in lattices

Saturate space with enough lattice vectors so that their sums give short(er) vectors



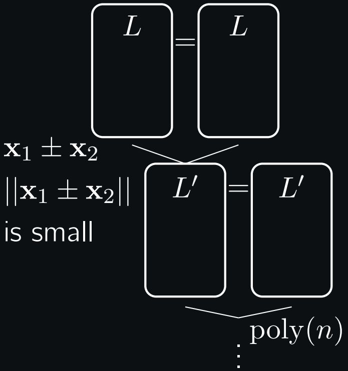
Idea of sieving in lattices

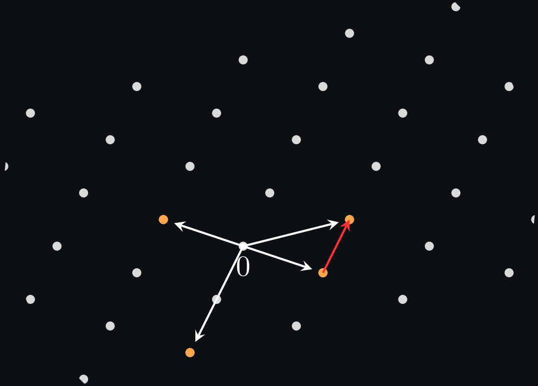
Saturate space with enough lattice vectors so that their sums give short(er) vectors



Idea of sieving in lattices

Saturate space with enough lattice vectors so that their sums give short(er) vectors

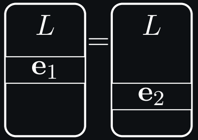




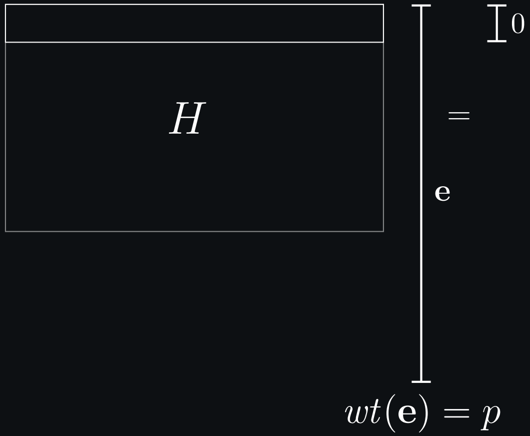
Questions to be addressed: size of L (memory), time to find all close pairs (complexity). Best known algorithm for the short vector problem.

Idea of sieving in binary codes

Keep weight constant, move gradually from subcodes $\mathcal{C}_i := \{\mathbf{e} \ : \ (H\mathbf{e})[0:i] = 0\}$ to the code $\mathcal{C}$: $\mathcal{C}_1 \subset \mathcal{C}_2 \ldots \subset \mathcal{C}$. Choose some weight $p \leq \omega$.

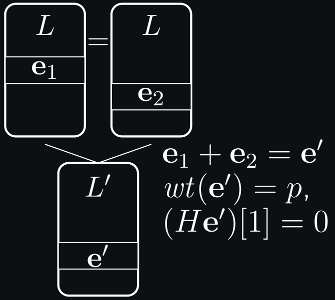


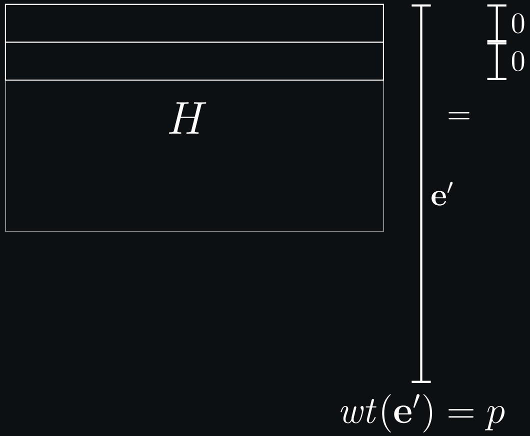
$wt(\mathbf{e}_i) = p$
$(H\mathbf{e}_i)[0] = 0$



Idea of sieving in binary codes

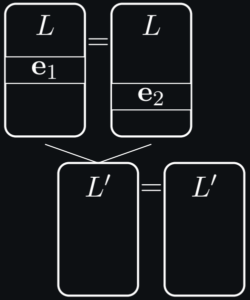
Keep weight constant, move gradually from subcodes $\mathcal{C}_i := \{\mathbf{e} : (H\mathbf{e})[0:i] = 0\}$ to the code $\mathcal{C}$: $\mathcal{C}_1 \subset \mathcal{C}_2 \ldots \subset \mathcal{C}$. Choose some weight $p \leq \omega$.

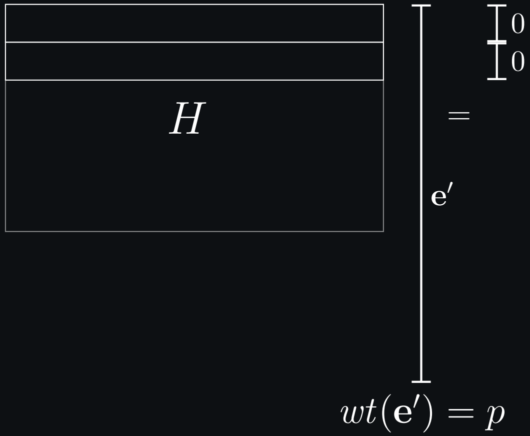




Idea of sieving in binary codes

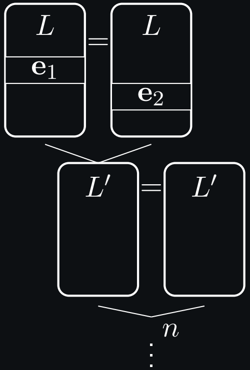
Keep weight constant, move gradually from subcodes $\mathcal{C}_i := \{\mathbf{e} \ : \ (H\mathbf{e})[0 : i] = 0\}$ to the code $\mathcal{C}$: $\mathcal{C}_1 \subset \mathcal{C}_2 \ldots \subset \mathcal{C}$. Choose some weight $p \leq \omega$.

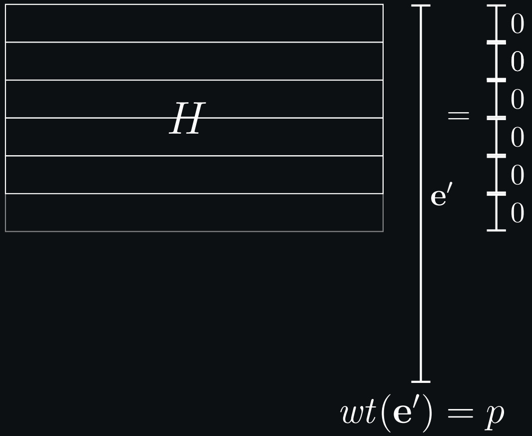




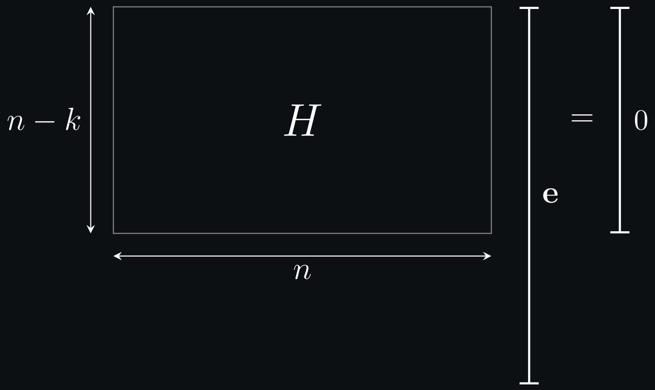
Idea of sieving in binary codes

Keep weight constant, move gradually from subcodes $\mathcal{C}_i := \{\mathbf{e} : (H\mathbf{e})[0:i] = 0\}$ to the code $\mathcal{C}$: $\mathcal{C}_1 \subset \mathcal{C}_2 \ldots \subset \mathcal{C}$. Choose some weight $p \leq \omega$.

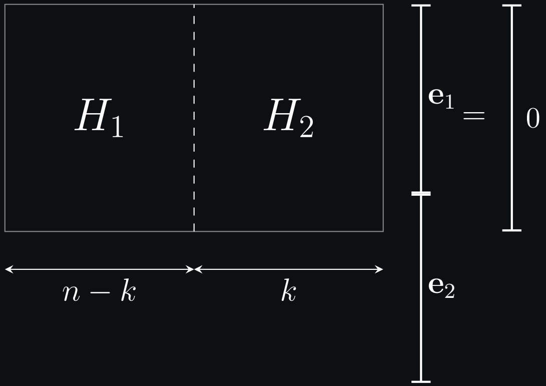




Setting up ISD with Sieving: systematic form



Setting up ISD with Sieving: systematic form



Setting up ISD with Sieving: systematic form

$$\left[\begin{array}{c|c} \mathbf{I}_{n-k} & H' \end{array}\right] \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}$$

$$\underbrace{\hphantom{\mathbf{I}_{n-k}}}_{n-k}\ \underbrace{\hphantom{H'}}_{k}$$

Setting up ISD with Sieving: systematic form

$$\begin{pmatrix} \underbrace{\mathbf{I}_{n-k-\ell}}_{} & H' \\ \underbrace{0}_{n-k-\ell} & \underbrace{H''}_{k+\ell} \end{pmatrix} \begin{pmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}$$

Setting up ISD with Sieving: systematic form

$$\begin{array}{cc} \left[\begin{array}{c:c} \mathbf{I}_{n-k-\ell} & H' \\ \hdashline 0 & H'' \end{array}\right] \\ \begin{array}{cc} \quad n-k-\ell \quad & \quad k+\ell \quad \end{array} \end{array} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix} \implies \begin{cases} H'\mathbf{e}_2 + \mathbf{e}_1 = 0 \\ H''\mathbf{e}_2 = 0 \end{cases}$$

Setting up ISD with Sieving: systematic form

$$\begin{array}{cc} \mathbf{I}_{n-k-\ell} & H' \\ 0 & H'' \\ \xleftrightarrow{\;n-k-\ell\;} & \xleftrightarrow{\;k+\ell\;} \end{array} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix} \implies \begin{cases} H'\mathbf{e}_2 + \mathbf{e}_1 = 0 \\ H''\mathbf{e}_2 = 0 \end{cases}$$

Sieve for $\mathbf{e}_2$

$$wt(\mathbf{e}_2) = p,\ wt(\mathbf{e}_1) = \omega - p$$

Setting up ISD with Sieving: systematic form

$$\begin{array}{cc} \left[\begin{array}{c:c} \mathbf{I}_{n-k-\ell} & H' \\ \hdashline 0 & H'' \end{array}\right] \\ \xleftrightarrow{\;n-k-\ell\;} \xleftrightarrow{\;k+\ell\;} \end{array} \begin{bmatrix} \mathbf{e}_1 \\ \mathbf{e}_2 \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix} \implies \begin{cases} H'\mathbf{e}_2 + \mathbf{e}_1 = 0 \\ H''\mathbf{e}_2 = 0 \end{cases}$$

Sieve for $\mathbf{e}_2$

$$wt(\mathbf{e}_2) = p,\, wt(\mathbf{e}_1) = \omega - p$$

Apply permutations on H until achieve the correct weight distribution on $\mathbf{e}_1, \mathbf{e}_2$.

Sieving for codes: the algorithm

1. Randomly permute H and compute H''.

Sieving for codes: the algorithm

1. Randomly permute H and compute H''.
2. Construct L_0, $|L_0| = N$ with vectors $\mathbf{v}$ s.t. $wt(\mathbf{v}) = p$ and $(H''\mathbf{v})[0] = 0$.

Sieving for codes: the algorithm

1. Randomly permute H and compute H''.
2. Construct L_0, $|L_0| = N$ with vectors $\mathbf{v}$ s.t. $wt(\mathbf{v}) = p$ and $(H''\mathbf{v})[0] = 0$.
3. For $i = 1, \ldots, n$:
   - 3.1 Find all pairs $\mathbf{v}, \mathbf{v} \in L_{i-1}$ with $wt(\mathbf{v} + \mathbf{v}') = p$, store them in L_i
   - 3.2 Discard all $\mathbf{v} \in L_i$ s.t. $\mathbf{v} \notin \mathcal{C}_i$

Sieving for codes: the algorithm

1. Randomly permute H and compute H''.
2. Construct L_0, $|L_0| = N$ with vectors $\mathbf{v}$ s.t. $wt(\mathbf{v}) = p$ and $(H''\mathbf{v})[0] = 0$.
3. For $i = 1, \ldots, n$:
   3.1 Find all pairs $\mathbf{v}, \mathbf{v} \in L_{i-1}$ with $wt(\mathbf{v} + \mathbf{v}') = p$, store them in L_i
   3.2 Discard all $\mathbf{v} \in L_i$ s.t. $\mathbf{v} \notin \mathcal{C}_i$
4. Check all $\mathbf{v} \in L_n$ for $\mathbf{v} \stackrel{?}{=} \mathbf{e}''$

Sieving for codes: the algorithm

1. Randomly permute H and compute H''.
2. Construct L_0, $|L_0| = N$ with vectors $\mathbf{v}$ s.t. $wt(\mathbf{v}) = p$ and $(H''\mathbf{v})[0] = 0$.
3. For $i = 1, \ldots, n$:
   3.1 Find all pairs $\mathbf{v}, \mathbf{v} \in L_{i-1}$ with $wt(\mathbf{v} + \mathbf{v}') = p$, store them in L_i
   3.2 Discard all $\mathbf{v} \in L_i$ s.t. $\mathbf{v} \notin \mathcal{C}_i$
4. Check all $\mathbf{v} \in L_n$ for $\mathbf{v} \overset{?}{=} \mathbf{e}''$

Runtime

Success Probability:

$$\underbrace{\Pr[wt(\mathbf{e}'') = p]}_{\frac{\binom{n-k-\ell}{w-p}\binom{k+\ell}{p}}{\binom{n}{w}}} \cdot \underbrace{\Pr[\mathbf{e}'' \in L_n]}_{\frac{N}{\binom{k+\ell}{p}/2^\ell}}$$

Time per iteration (Steps 1–4)

$n \cdot T_{\mathcal{NN}}$

$T_{\mathcal{NN}}$ – runtime of Near Neighbor search (Step 3.1)

Glimpse of the analysis

How large is N?

How large is $T_{\mathcal{N}\mathcal{N}}$?

Glimpse of the analysis

How large is N?

- Want: $\left|\mathbf{w} \in L_i \colon \mathbf{w} \in \mathcal{C}_i\right| \geq N$
- Each new parity-check equation eliminates half of the list elements:

$$\Pr[\mathbf{w} \in \mathcal{C}_i \mid \mathbf{w} \in L_i = \Pr[\mathbf{w} \in \mathcal{C}_i \mid \mathbf{w} \in \mathcal{C}_{i-1}] = \frac{|\mathcal{C}_i|}{|\mathcal{C}_{i-1}|} = 1/2$$

- We want to keep (asymptotically) the same list sizes:

$$\mathbb{E}\big[|\mathbf{w} \in L_i \colon \mathbf{w} \in \mathcal{C}_i|\big] = \mathbb{E}\big[|L_i|\big]/2 \overset{!}{\geq} |L_{i-1}| =: N$$

- $\mathbb{E}\big[|L_i|\big] = |L_{i-1}|^2 \cdot \Pr[wt(\mathbf{v}+\mathbf{v}') = p \colon wt(\mathbf{v}) = wt(\mathbf{v}') = p] =$

$$= N^2 \cdot \frac{\binom{k+\ell}{p} \cdot \binom{p}{p/2}\binom{k+\ell-p}{p/2}}{\binom{k+\ell}{p}^2} \overset{!}{=\geq} 2N \quad \Leftrightarrow \quad N \geq \frac{2\binom{k+\ell}{p}}{\binom{p}{p/2}\binom{k+\ell-p}{p/2}}$$

How large is $T_{\mathcal{NN}}$?

Glimpse of the analysis

How large is N?

- Want: $|\mathbf{w} \in L_i \colon \mathbf{w} \in \mathcal{C}_i| \geq N$
- Each new parity-check equation eliminates half of the list elements:

$$\Pr[\mathbf{w} \in \mathcal{C}_i \mid \mathbf{w} \in L_i = \Pr[\mathbf{w} \in \mathcal{C}_i \mid \mathbf{w} \in \mathcal{C}_{i-1}] = \frac{|\mathcal{C}_i|}{|\mathcal{C}_{i-1}|} = 1/2$$

- We want to keep (asymptotically) the same list sizes:

$$\mathbb{E}\big[|\mathbf{w} \in L_i \colon \mathbf{w} \in \mathcal{C}_i|\big] = \mathbb{E}\big[|L_i|\big]/2 \overset{!}{\geq} |L_{i-1}| =: N$$

- $\mathbb{E}\big[|L_i|\big] = |L_{i-1}|^2 \cdot \Pr[wt(\mathbf{v} + \mathbf{v}') = p \colon wt(\mathbf{v}) = wt(\mathbf{v}') = p] =$

$$= N^2 \cdot \frac{\binom{k+\ell}{p} \cdot \binom{p}{p/2}\binom{k+\ell-p}{p/2}}{\binom{k+\ell}{p}^2} \overset{!}{=\geq} 2N \quad \Leftrightarrow \quad N \geq \frac{2\binom{k+\ell}{p}}{\binom{p}{p/2}\binom{k+\ell-p}{p/2}}$$

How large is $T_{\mathcal{N}\mathcal{N}}$?

Depends on the algorithm...

ISD with Sieving: asymptotics (worst-case rate, GV bound error)

- $wt(\mathbf{v}_1) = wt(\mathbf{v}_2) = wt(\mathbf{v}_1 + \mathbf{v}_2) = p \;\Rightarrow\; wt(\mathbf{v}_1 \wedge \mathbf{v}_2) = p/2$
- Idea: Enumerate potential overlap for each vector

ISD with Sieving: asymptotics (worst-case rate, GV bound error)

- $wt(\mathbf{v}_1) = wt(\mathbf{v}_2) = wt(\mathbf{v}_1 + \mathbf{v}_2) = p \Rightarrow wt(\mathbf{v}_1 \wedge \mathbf{v}_2) = p/2$
- Idea: put another random code on top, decode all $\mathbf{v}_i$'s w.r.t. code. Close $\mathbf{v}_i$'s will decode to the same codeword(s).

ISD with Sieving: asymptotics (worst-case rate, GV bound error)

- $wt(\mathbf{v}_1) = wt(\mathbf{v}_2) = wt(\mathbf{v}_1 + \mathbf{v}_2) = p \Rightarrow wt(\mathbf{v}_1 \wedge \mathbf{v}_2) = p/2$
- Idea: a Random Product Code (RPC) on top, decode all $\mathbf{v}_i$'s w.r.t. code. Close $\mathbf{v}_i$'s will decode to the same codeword(s). But now we can find *all of them faster*.

ISD with Sieving: asymptotics (worst-case rate, GV bound error)

- $wt(\mathbf{v}_1) = wt(\mathbf{v}_2) = wt(\mathbf{v}_1 + \mathbf{v}_2) = p \Rightarrow wt(\mathbf{v}_1 \wedge \mathbf{v}_2) = p/2$
- Idea: a Random Product Code (RPC) on top, decode all $\mathbf{v}_i$'s w.r.t. code. Close $\mathbf{v}_i$'s will decode to the same codeword(s). But now we can find *all of them faster*.

Conclusions

- Take-away: lattice sieving including NN technique translate to codes in Hamming metric

Conclusions

- Take-away: lattice sieving including NN technique translate to codes in Hamming metric
- Open research direction: k-sieve (time-memory trade-offs)
- Full version: `https://eprint.iacr.org/2023/1577`
- Slides: `https://crypto-kantiana.com/elena.kirshanova/talks/MWCC24.pdf`

Conclusions

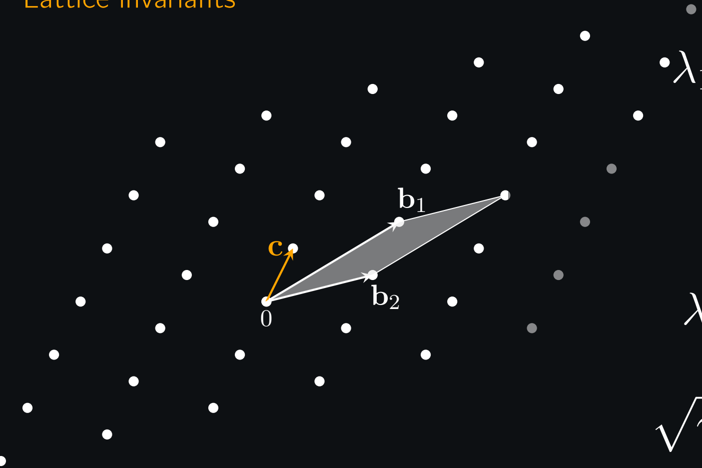
- Take-away: lattice sieving including NN technique translate to codes in Hamming metric
- Open research direction: k-sieve (time-memory trade-offs)
- Full version: `https://eprint.iacr.org/2023/1577`
- Slides: `https://crypto-kantiana.com/elena.kirshanova/talks/MWCC24.pdf`

Q?

Part III

From codes to lattices: dense lattice construction

Lattice invariants



Minimum

$$\lambda_1(\Lambda) = \min_{\mathbf{v} \in \Lambda \setminus \mathbf{0}} \|\mathbf{v}\|_2$$

Determinant

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

Minkowski bound

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

Normalized min. distance

$$\sqrt{\gamma(\Lambda)} = \lambda_1(\Lambda) / \det(\Lambda)^{\frac{1}{n}}$$

A lattice is a set $\Lambda = \{\sum_{i \leq n} x_i \mathbf{b}_i \ : \ x_i \in \mathbb{Z}\}$ for linearly independent $\mathbf{b}_i \in \mathbb{R}^n$.
$\{\mathbf{b}_i\}_i$ is a basis of Λ

Our goal

$$\sqrt{\gamma(\Lambda)} = \lambda_1(\Lambda) / \det(\Lambda)^{\frac{1}{n}} \leq \sqrt{n}$$

We are interested in

1. explicit construction of a lattice with as large $\gamma(\Lambda)$ as possible
2. with an efficient (list-) decoding algorithm (runtime at most $\mathrm{poly}(n)$).

Our goal

$$\sqrt{\gamma(\Lambda)} = \lambda_1(\Lambda)/\det(\Lambda)^{\frac{1}{n}} \leq \sqrt{n}$$

We are interested in

1. explicit construction of a lattice with as large $\gamma(\Lambda)$ as possible
2. with an efficient (list-) decoding algorithm (runtime at most $\mathrm{poly}(n)$).

Why? We might want to use lattice as codes, hence we care about their decoding properties.

A 'random' lattice (an example will given later) is expected to achieve $\sqrt{\gamma(\Lambda)} \sim \sqrt{n}$, but we do not know how to efficiently decode them.

State-of-the art on $\sqrt{\gamma(\Lambda)}$ ($\Omega()$ for $\sqrt{\gamma(\Lambda)}$ is omitted)

Lattice Λ	$\sqrt{\gamma(\Lambda)}$
Barnes-Wall lattice [BW]	$n^{1/4}$

Defined by the rows of

$$\mathrm{BW}^k = \begin{bmatrix} 1 & 1 \\ 0 & \phi \end{bmatrix}^{\otimes k} \subset \mathbb{C}^{2^k},$$

where $\phi = 1 + i$

State-of-the art on $\sqrt{\gamma(\Lambda)}$ ($\Omega()$ for $\sqrt{\gamma(\Lambda)}$ is omitted)

Lattice Λ	$\sqrt{\gamma(\Lambda)}$
Barnes-Wall lattice [BW]	$n^{1/4}$
Discrete Logarithm Lattices [DP]	$\frac{\sqrt{n}}{\log n}$

For $(\mathbb{Z}/m\mathbb{Z})^\star$,

p_i – primes, $1 \leq i \leq n$

$\phi : \mathbb{Z}^n \to (\mathbb{Z}/m\mathbb{Z})^\star$

$(x_1, \ldots, x_n) \mapsto \prod_{i=1}^n p_i^{x_i}$

$\Lambda_{\mathsf{dlog}} = \ker \phi$.

State-of-the art on $\sqrt{\gamma(\Lambda)}$ ($\Omega()$ for $\sqrt{\gamma(\Lambda)}$ is omitted)

Lattice Λ	$\sqrt{\gamma(\Lambda)}$
Barnes-Wall lattice [BW]	$n^{1/4}$
Discrete Logarithm Lattices [DP]	$\frac{\sqrt{n}}{\log n}$
Construction-A lattice from Reed-Solomon codes [BP]	$\sqrt{\frac{n}{\log n}}$

Constriction-A:

Take $B \in (\mathbb{Z}/q\mathbb{Z})^{n\times m}$ – a generator matrix of a code.

$\Lambda_{\mathsf{A}} = \mathbb{Z}^n B + q\mathbb{Z}^m \subset \mathbb{Z}^m$

is a construction-A lattice.

State-of-the art on $\sqrt{\gamma(\Lambda)}$ ($\Omega()$ for $\sqrt{\gamma(\Lambda)}$ is omitted)

Lattice Λ	$\sqrt{\gamma(\Lambda)}$
Barnes-Wall lattice [BW]	$n^{1/4}$
Discrete Logarithm Lattices [DP]	$\frac{\sqrt{n}}{\log n}$
Construction-A lattice from Reed-Solomon codes [BP]	$\sqrt{\frac{n}{\log n}}$
Construction-D lattice from BCH codes [MP]	$\sqrt{\frac{n}{\log n}}$

Lifting sequences of codes to lattices

State-of-the art on $\sqrt{\gamma(\Lambda)}$ ($\Omega()$ for $\sqrt{\gamma(\Lambda)}$ is omitted)

Lattice Λ	$\sqrt{\gamma(\Lambda)}$
Barnes-Wall lattice [BW]	$n^{1/4}$
Discrete Logarithm Lattices [DP]	$\frac{\sqrt{n}}{\log n}$
Construction-A lattice from Reed-Solomon codes [BP]	$\sqrt{\frac{n}{\log n}}$
Construction-D lattice from BCH codes [MP]	$\sqrt{\frac{n}{\log n}}$
Construction-D lattice from subfield subcodes of Garcia-Stichtenoth codes [KM]	$\frac{\sqrt{n}}{(\log n)^{\varepsilon+o(1)}}$

Kirshanova-Malygina'23.
This talk

Main result

Theorem: For a constant $\varepsilon > 0$, there is a family of lattices $\mathcal{L} \subset \mathbb{R}^n$ with normalized minimum distance

$$\frac{\lambda_1(\Lambda)}{\det(\Lambda)^{1/n}} = \Omega\left(\frac{\sqrt{n}}{(\log n)^{\varepsilon+o(1)}}\right).$$

These lattices are list decodable to within distance $\sqrt{1/2} \cdot \lambda_1(\Lambda)$ in $\mathrm{poly}(n)$ time.

Construction-D lattice: simplified definition

- Fix an integer $L \geq 0$, let
$$C_L \subseteq C_{L-1} \subseteq \ldots \subseteq C_1 \subseteq C_0 = \mathbb{F}_p^n$$
be a tower of p-ary codes of length n, where $\dim(C_i) = k_i$.

Construction-D lattice: simplified definition

- Fix an integer $L \geq 0$, let
$$C_L \subseteq C_{L-1} \subseteq \ldots \subseteq C_1 \subseteq C_0 = \mathbb{F}_p^n$$
be a tower of p-ary codes of length n, where $\dim(C_i) = k_i$.
- Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a basis of $\mathbb{F}_p^n$ s.t.
$$\mathbf{b}_1, \ldots, \mathbf{b}_{k_i} \text{ is a basis of } C_i \text{ for all } i = 0, \ldots, L.$$
- Define a set of distinguished $\mathbb{Z}^n$ representatives of $\mathbf{c}_i = \sum_{j=1}^{k_i} a_j \mathbf{b}_j \in C_i$ as
$$\overline{\mathbf{c}}_i = \sum_{j=1}^{k_i} \overline{a}_j \overline{\mathbf{b}}_j \in \mathbb{Z}^n \quad \text{where } \overline{a}_j \in \{0, \ldots p-1\} \subset \mathbb{Z}.$$

Construction-D lattice: simplified definition

- Fix an integer $L \geq 0$, let
$$C_L \subseteq C_{L-1} \subseteq \ldots \subseteq C_1 \subseteq C_0 = \mathbb{F}_p^n$$
be a tower of p-ary codes of length n, where $\dim(C_i) = k_i$.
- Let $\mathbf{b}_1, \ldots, \mathbf{b}_n$ be a basis of $\mathbb{F}_p^n$ s.t.
$$\mathbf{b}_1, \ldots, \mathbf{b}_{k_i} \text{ is a basis of } C_i \text{ for all } i = 0, \ldots, L.$$
- Define a set of distinguished $\mathbb{Z}^n$ representatives of $\mathbf{c}_i = \sum_{j=1}^{k_i} a_j \mathbf{b}_j \in C_i$ as
$$\overline{\mathbf{c}}_i = \sum_{j=1}^{k_i} \overline{a}_j \overline{\mathbf{b}}_j \in \mathbb{Z}^n \quad \text{where } \overline{a}_j \in \{0, \ldots p-1\} \subset \mathbb{Z}.$$
- Let $\mathcal{L}_0 = \mathbb{Z}^n$, and for each $i = 1, \ldots, L$ define
$$\Lambda_i = \overline{C}_i + p\Lambda_{i-1}, \quad \overline{C}_i = \{\overline{c}_i \ : \ c_i \in C_i\}.$$
- The construction-D for the tower $\{C_i\}$ is $\Lambda = \Lambda_L$.

Main idea

- Construct a sequence of codes $C_L \subseteq C_{L-1} \subseteq \ldots \subseteq C_1 \subseteq C_0 = \mathbb{F}_q^n$, each C_i is an algebraic-geometric code from a specific function field, the Garcia-Stichtenoth field.
- Such AG-codes are defined over $\mathbb{F}_{p^h}$ for an even h, hence go to subfield-subcodes:

$$C_L \cap \mathbb{F}_p^n \subseteq C_{L-1} \cap \mathbb{F}_p^n \subseteq \ldots C_0 \cap \mathbb{F}_p^n \ = \mathbb{F}_p^n,$$

- We know $\dim(C_i \cap \mathbb{F}_p^n)$ and minimal distance of $C_i \cap \mathbb{F}_p^n$ for all i.
- Compute the minimum $\lambda_1(\Lambda_L)$ of the construction-D lattice Λ_L.
- Compute (an upper bound on) $\det(\Lambda_L)$.
- Conclude on $\gamma(\Lambda) = \lambda_1(\Lambda)/\det(\Lambda_L)$.
- For efficient decoding, adapt soft decision decoding algorithm of Koetter-Vardy.

Conclusions

- Take you favourite code (may be an AG code) with a poly-time decoding algorithm.
- Construct a sequence of codes with a lower bound on min. distance and on dimension.
- These suffice to derive $\lambda_1(\Lambda)$ and (a lower bound) on $\det(\Lambda)$.
- Check if you beat the state-of-the-art.
- Interesting candidate: Bassa-Ritzenthaler towers, (arXiv:1807.05714)