Crypto 2024

August 21, 2024 — Santa Barbara, United States

FuLeakage:

Breaking FuLeeca by Learning Attacks

Felicitas Hörmann[1,2] – felicitas.hoermann@dlr.de

joint work with Wessel van Woerden[3]

[1]Institute of Communications and Navigation – German Aerospace Center (DLR), Germany

[2]School of Computer Science – University of St. Gallen, Switzerland

[3]Institut de Mathématiques de Bordeaux – University of Bordeaux, France

DLR

Motivation and Overview

FuLeeca

ia.cr/2023/377

FuLeeca

- is a code-based signature scheme,
- uses quasi-cyclic codes in the Lee metric, and
- was presented at CBCrypto 2023 and submitted to NIST's additional call for signatures [Ritterhoff et al., 2023].

Motivation and Overview

FuLeeca

`ia.cr/2023/377`

FuLeeca

- is a code-based signature scheme,
- uses quasi-cyclic codes in the Lee metric, and
- was presented at CBCrypto 2023 and submitted to NIST's additional call for signatures [Ritterhoff et al., 2023].

FuLeakage

`ia.cr/2024/353`

	few signatures ($\ll 100$)	**many signatures** ($\leq 175{,}000$)
classical attack	leaked-sublattice attack (reduced security)	learning attack (full break)
quantum attack	ideal-structure attack (full break)	$\leftarrow$ see this attack

Motivation and Overview

FuLeeca

- is a code-based signature scheme,
- uses quasi-cyclic codes in the Lee metric, and
- was presented at CBCrypto 2023 and submitted to NIST's additional call for signatures [Ritterhoff et al., 2023].

FuLeeca
`ia.cr/2023/377`

FuLeakage
`ia.cr/2024/353`

	few signatures ($\ll 100$)	**many signatures** ($\leq 175{,}000$)
classical attack	leaked-sublattice attack (reduced security)	learning attack (full break)
quantum attack	ideal-structure attack (full break)	← see this attack

Outline

1. FuLeeca, Codes, and Lattices
2. Leaked-Sublattice Attack
3. Learning Attack

Lee-Metric Codes and Euclidean Lattices

We use the representation $\mathbb{F}_p = \left\{-\frac{p-1}{2}, \ldots, \frac{p-1}{2}\right\}$.

Lee-Metric Codes and Euclidean Lattices

We use the representation $\mathbb{F}_p = \left\{-\frac{p-1}{2}, \dots, \frac{p-1}{2}\right\}$.

Lee-metric codes	Euclidean lattices
Linear code $\mathcal{C} = \mathbb{F}_p^k \cdot \boldsymbol{G}$ for a full-rank generator matrix $\boldsymbol{G} \in \mathbb{F}_p^{k \times n}$.	Lattice $\mathcal{L} = \mathbb{Z}^k \cdot \boldsymbol{B}$ for a full-rank basis $\boldsymbol{B} \in \mathbb{R}^{k \times n}$.

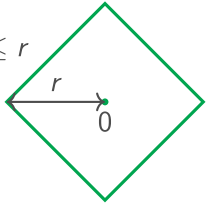
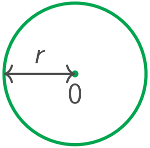
Lee-Metric Codes and Euclidean Lattices

We use the representation $\mathbb{F}_p = \left\{-\frac{p-1}{2}, \dots, \frac{p-1}{2}\right\}$.

Lee-metric codes	Euclidean lattices
Linear code $\mathcal{C} = \mathbb{F}_p^k \cdot \boldsymbol{G}$ for a full-rank generator matrix $\boldsymbol{G} \in \mathbb{F}_p^{k \times n}$.	Lattice $\mathcal{L} = \mathbb{Z}^k \cdot \boldsymbol{B}$ for a full-rank basis $\boldsymbol{B} \in \mathbb{R}^{k \times n}$.
$\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_p^n$ has Lee weight $\mathrm{wt}_L(\boldsymbol{x}) = \sum_i \lvert x_i \rvert$.	$\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ has Euclidean norm $\lVert \boldsymbol{x} \rVert_2 = \sqrt{\sum_i x_i^2}$.

Lee-Metric Codes and Euclidean Lattices

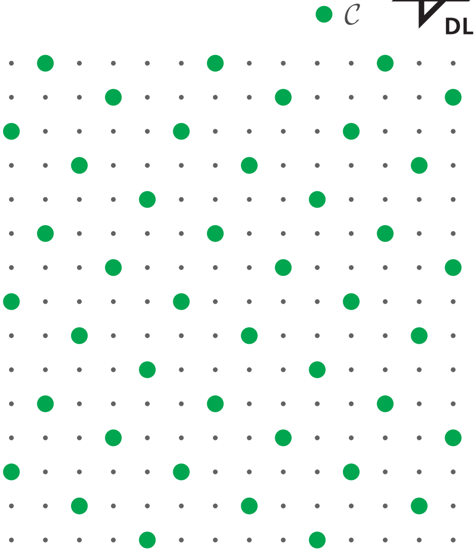
We use the representation $\mathbb{F}_p = \left\{-\frac{p-1}{2}, \dots, \frac{p-1}{2}\right\}$.

Lee-metric codes	Euclidean lattices
Linear code $\mathcal{C} = \mathbb{F}_p^k \cdot \boldsymbol{G}$ for a full-rank generator matrix $\boldsymbol{G} \in \mathbb{F}_p^{k \times n}$.	Lattice $\mathcal{L} = \mathbb{Z}^k \cdot \boldsymbol{B}$ for a full-rank basis $\boldsymbol{B} \in \mathbb{R}^{k \times n}$.
$\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_p^n$ has Lee weight $\mathrm{wt}_L(\boldsymbol{x}) = \sum_i \lvert x_i \rvert$.	$\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ has Euclidean norm $\lVert \boldsymbol{x} \rVert_2 = \sqrt{\sum_i x_i^2}$.
$\mathrm{wt}_L(\boldsymbol{x}) \leq r$ 	 $\lVert \boldsymbol{x} \rVert_2 \leq r$

Ball of radius r

Hash-and-Sign Signatures

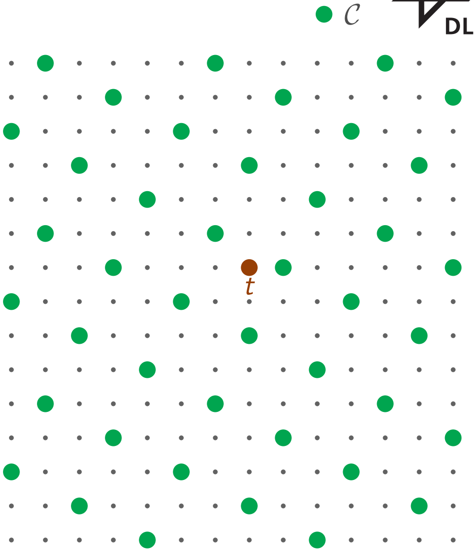
A hash-and-sign scheme based on a code $\mathcal{C}$ signs a message $\boldsymbol{m}$ as follows:



Hash-and-Sign Signatures

A hash-and-sign scheme based on a code $\mathcal{C}$ signs a message $\boldsymbol{m}$ as follows:

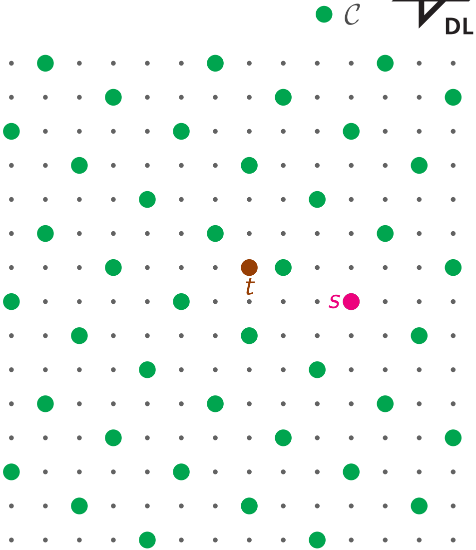
1. Hash the message $\boldsymbol{m}$ to a target $\boldsymbol{t}$.



Hash-and-Sign Signatures

A hash-and-sign scheme based on a code $\mathcal{C}$ signs a message $\boldsymbol{m}$ as follows:

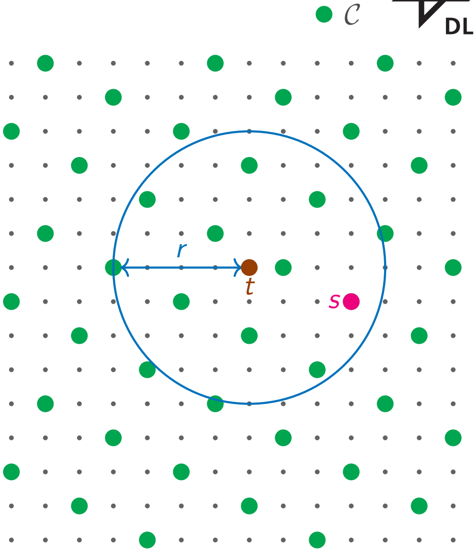
1. Hash the message $\boldsymbol{m}$ to a target $\boldsymbol{t}$.
2. The signature $\boldsymbol{s}$ is
   - a codeword
   - close to the target $\boldsymbol{t}$.



Hash-and-Sign Signatures

A hash-and-sign scheme based on a code $\mathcal{C}$ signs a message $\boldsymbol{m}$ as follows:

1. Hash the message $\boldsymbol{m}$ to a target $\boldsymbol{t}$.
2. The signature $\boldsymbol{s}$ is
   - a codeword
     $\rightarrow$ check if $\boldsymbol{s} \in \mathcal{C}$
   - close to the target $\boldsymbol{t}$.
     $\rightarrow$ check $\mathrm{wt}_L(\boldsymbol{s} - \boldsymbol{t}) \leq r$

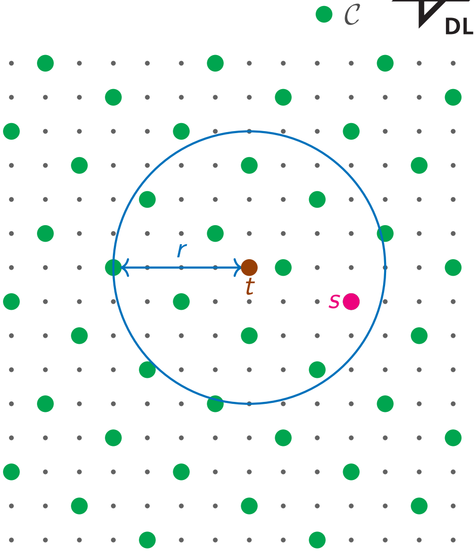


Hash-and-Sign Signatures

A hash-and-sign scheme based on a code $\mathcal{C}$ signs a message $\boldsymbol{m}$ as follows:

1. Hash the message $\boldsymbol{m}$ to a target $\boldsymbol{t}$.
2. The signature $\boldsymbol{s}$ is
   - a codeword
     $\rightarrow$ check if $\boldsymbol{s} \in \mathcal{C}$
   - close to the target $\boldsymbol{t}$.
     $\rightarrow$ check $\mathrm{wt}_L(\boldsymbol{s} - \boldsymbol{t}) \leq r$

The signer needs a good description of the code but any description works for the verifier.



FuLeeca Key Generation [Ritterhoff et al., 2023]

Generate a secret vector $\boldsymbol{g} = (\boldsymbol{a} \mid \boldsymbol{b}) \in \mathbb{F}_p^n$ with $n = 2k$ by drawing $\boldsymbol{a}$ and $\boldsymbol{b}$ uniformly at random from $\{\boldsymbol{x} \in \mathbb{F}_p^k : \mathrm{wt}_L(\boldsymbol{x}) = w_{\mathrm{key}}\}$.

FuLeeca Key Generation [Ritterhoff et al., 2023]

Generate a secret vector $\boldsymbol{g} = (\boldsymbol{a} \mid \boldsymbol{b}) \in \mathbb{F}_p^n$ with $n = 2k$
by drawing $\boldsymbol{a}$ and $\boldsymbol{b}$ uniformly at random from $\{\boldsymbol{x} \in \mathbb{F}_p^k : \mathrm{wt}_L(\boldsymbol{x}) = w_{\mathrm{key}}\}$.

Secret key:

$$\boldsymbol{G}_{\mathrm{sec}} = (\boldsymbol{A} \mid \boldsymbol{B}) = \left(\begin{array}{cccc|cccc} a_1 & a_2 & \dots & a_k & b_1 & b_2 & \dots & b_k \\ a_k & a_1 & \dots & a_{k-1} & b_k & b_1 & \dots & b_{k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & \dots & a_1 & b_2 & b_3 & \dots & b_1 \end{array}\right) = \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_k \end{pmatrix}$$

FuLeeca Key Generation [Ritterhoff et al., 2023]

Generate a secret vector $\boldsymbol{g} = (\boldsymbol{a} \mid \boldsymbol{b}) \in \mathbb{F}_p^n$ with $n = 2k$
by drawing $\boldsymbol{a}$ and $\boldsymbol{b}$ uniformly at random from $\{\boldsymbol{x} \in \mathbb{F}_p^k : \mathrm{wt}_L(\boldsymbol{x}) = w_{\text{key}}\}$.

Secret key:

$$\boldsymbol{G}_{\text{sec}} = (\boldsymbol{A} \mid \boldsymbol{B}) = \left(\begin{array}{cccc|cccc} a_1 & a_2 & \dots & a_k & b_1 & b_2 & \dots & b_k \\ a_k & a_1 & \dots & a_{k-1} & b_k & b_1 & \dots & b_{k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & \dots & a_1 & b_2 & b_3 & \dots & b_1 \end{array}\right) = \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_k \end{pmatrix}$$

Public key:

$$\boldsymbol{G}_{\text{pub}} = \left(\boldsymbol{I}_k \mid \boldsymbol{A}^{-1}\boldsymbol{B}\right)$$

FuLeeca Key Generation [Ritterhoff et al., 2023]

Generate a secret vector $\boldsymbol{g} = (\boldsymbol{a} \mid \boldsymbol{b}) \in \mathbb{F}_p^n$ with $n = 2k$
by drawing $\boldsymbol{a}$ and $\boldsymbol{b}$ uniformly at random from $\{\boldsymbol{x} \in \mathbb{F}_p^k : \mathrm{wt}_L(\boldsymbol{x}) = w_{\text{key}}\}$.

Secret key:

$$\boldsymbol{G}_{\text{sec}} = (\boldsymbol{A} \mid \boldsymbol{B}) = \left(\begin{array}{cccc|cccc} a_1 & a_2 & \dots & a_k & b_1 & b_2 & \dots & b_k \\ a_k & a_1 & \dots & a_{k-1} & b_k & b_1 & \dots & b_{k-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_2 & a_3 & \dots & a_1 & b_2 & b_3 & \dots & b_1 \end{array}\right) = \begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \\ \vdots \\ \boldsymbol{g}_k \end{pmatrix}$$

Public key:

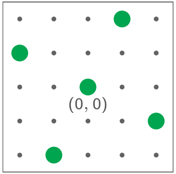
$$\boldsymbol{G}_{\text{pub}} = \left(\boldsymbol{I}_k \mid \boldsymbol{A}^{-1}\boldsymbol{B}\right)$$

Goal: Recover the secret vector $\boldsymbol{g}$ (or any of its quasi-circular shifts $\boldsymbol{g}_1, \dots, \boldsymbol{g}_k$).

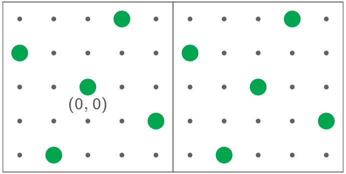
Lattices from Codes: Construction A

$\mathbb{F}_p^2$

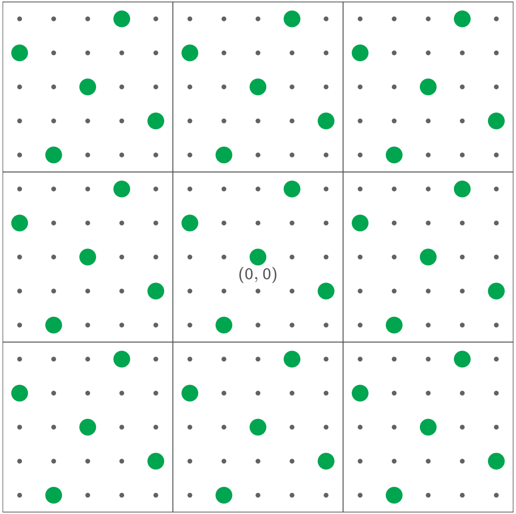
● $\mathcal{C}$



Lattices from Codes: Construction A

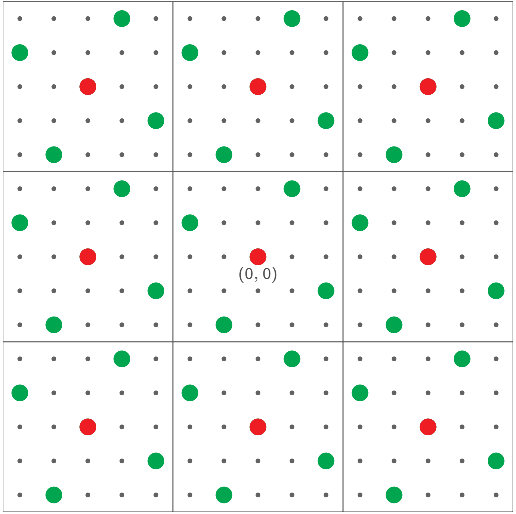


Lattices from Codes: Construction A



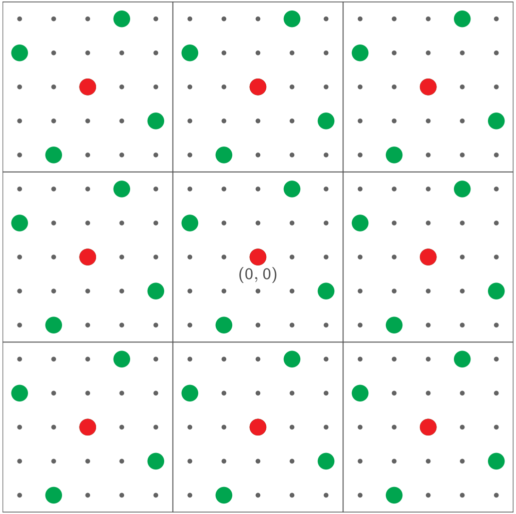
$$\mathcal{L}_{\mathrm{A}}(\mathcal{C}) := \{\boldsymbol{v} \in \mathbb{Z}^n : \boldsymbol{v} \pmod{p} \in \mathcal{C}\}$$

Lattices from Codes: Construction A



$$\mathcal{L}_{\mathsf{A}}(\mathcal{C}) = \mathcal{C} + p\mathbb{Z}^n$$

Lattices from Codes: Construction A

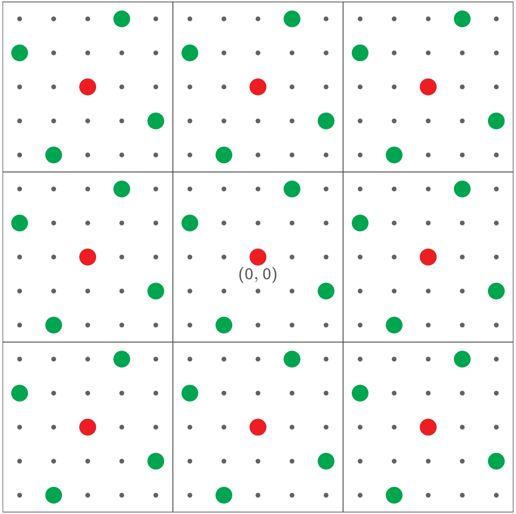


$$\mathcal{L}_{\mathrm{A}}(\mathcal{C}) = \mathcal{C} + p\mathbb{Z}^n$$

$\mathcal{L}_{\mathrm{A}}(\mathcal{C})$ is a rank-n lattice with basis

$$\boldsymbol{B}_{\mathrm{pub}} = \begin{pmatrix} \mathbf{I}_k & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \mathbf{0} & p\,\mathbf{I}_{n-k} \end{pmatrix}.$$

Lattices from Codes: Construction A



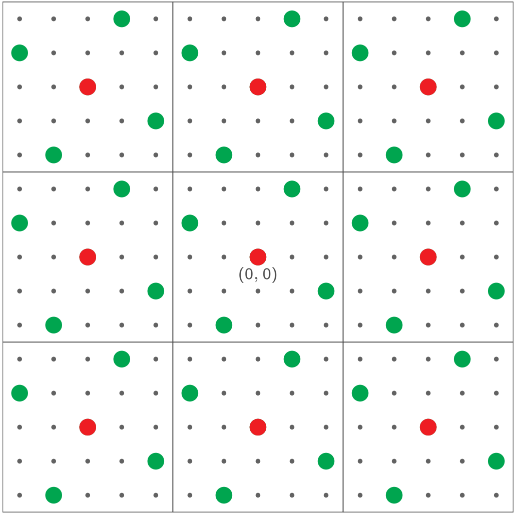
$$\mathcal{L}_{\mathrm{A}}(\mathcal{C}) = \mathcal{C} + p\mathbb{Z}^n$$

$\mathcal{L}_{\mathrm{A}}(\mathcal{C})$ is a rank-n lattice with basis

$$\boldsymbol{B}_{\text{pub}} = \begin{pmatrix} \mathbf{I}_k & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \mathbf{0} & p\,\mathbf{I}_{n-k} \end{pmatrix}.$$

Lattice techniques allow to recover a short vector in $O(2^{0.292n})$

Lattices from Codes: Construction A



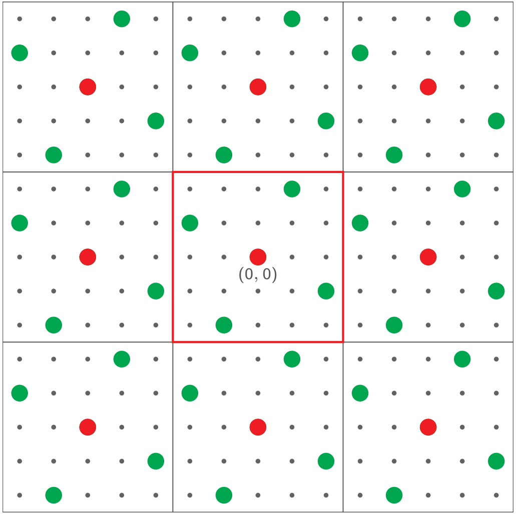
$$\mathcal{L}_{\mathrm{A}}(\mathcal{C}) = \mathcal{C} + p\mathbb{Z}^n$$

$\mathcal{L}_{\mathrm{A}}(\mathcal{C})$ is a rank-n lattice with basis

$$\boldsymbol{B}_{\text{pub}} = \begin{pmatrix} \mathbf{I}_k & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \mathbf{0} & p\,\mathbf{I}_{n-k} \end{pmatrix}.$$

Lattice techniques allow to recover a short vector in $O(2^{0.292n})$ but ISD attacks are more efficient.

Lattices from Codes: Construction A



$$\mathcal{L}_{\mathrm{A}}(\mathcal{C}) = \mathcal{C} + p\mathbb{Z}^n$$

$\mathcal{L}_{\mathrm{A}}(\mathcal{C})$ is a rank-n lattice with basis

$$\boldsymbol{B}_{\mathrm{pub}} = \begin{pmatrix} \mathbf{I}_k & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \mathbf{0} & p\,\mathbf{I}_{n-k} \end{pmatrix}.$$

Lattice techniques allow to recover a short vector in $O(2^{0.292n})$ but ISD attacks are more efficient.

Observe: In the central square, the Lee metric corresponds to the ℓ_1-norm.

Outline

1. FuLeeca, Codes, and Lattices
2. Leaked-Sublattice Attack
3. Learning Attack

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\boldsymbol{v} = \texttt{simpleSign}(\boldsymbol{c}, \boldsymbol{G}_{\text{sec}})$

2. $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\boldsymbol{v} = \texttt{simpleSign}(\boldsymbol{c}, \boldsymbol{G}_{\text{sec}})$
   Choose $\boldsymbol{v}$ *as a codeword close to* $\boldsymbol{c}$.

2. $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\boldsymbol{v} =$ `simpleSign` $(\boldsymbol{c}, \boldsymbol{G}_{\text{sec}})$

   Choose $\boldsymbol{v}$ *as a codeword close to* $\boldsymbol{c}$.

   Set $\boldsymbol{v} = \boldsymbol{x}\boldsymbol{G}_{\text{sec}} \bmod p$ with $x_i = \text{trunc}\left(\frac{s}{2} \cdot \langle \boldsymbol{c}, \text{sgn}(g_i)\rangle\right)$ for $i = 1, \ldots, k$.

2. $\boldsymbol{v} =$ `concentrate` $(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\boldsymbol{v} = \texttt{simpleSign}(\boldsymbol{c}, \boldsymbol{G}_{\text{sec}})$
   Choose $\boldsymbol{v}$ *as a codeword close to* $\boldsymbol{c}$.

   Set $\boldsymbol{v} = \boldsymbol{x}\boldsymbol{G}_{\text{sec}} \bmod p$ with $x_i = \operatorname{trunc}\left(\frac{s}{2} \cdot \langle \boldsymbol{c}, \operatorname{sgn}(g_i) \rangle\right)$ for $i = 1, \ldots, k$.

2. $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$
   Adapt $\boldsymbol{v}$ *to get its Lee weight and the sign matches with* $\boldsymbol{c}$ *to prescribed intervals.*

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\boldsymbol{v} =$ `simpleSign` $(\boldsymbol{c}, \boldsymbol{G}_{\text{sec}})$
   Choose $\boldsymbol{v}$ *as a codeword close to* $\boldsymbol{c}$.
   Set $\boldsymbol{v} = \boldsymbol{x}\boldsymbol{G}_{\text{sec}} \bmod p$ with $x_i = \text{trunc}\left(\frac{s}{2} \cdot \langle \boldsymbol{c}, \text{sgn}(g_i)\rangle\right)$ for $i = 1, \ldots, k$.

2. $\boldsymbol{v} =$ `concentrate` $(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$
   Adapt $\boldsymbol{v}$ *to get its Lee weight and the sign matches with* $\boldsymbol{c}$ *to prescribed intervals.*
   Iteratively improve $\boldsymbol{v}$ by successively adding $\pm \boldsymbol{g}_i$ for $i = 1, \ldots, k$.

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\boldsymbol{v} =$ `simpleSign` $(\boldsymbol{c}, \boldsymbol{G}_{\text{sec}})$
   Choose $\boldsymbol{v}$ *as a codeword close to* $\boldsymbol{c}$.
   Set $\boldsymbol{v} = \boldsymbol{x}\boldsymbol{G}_{\text{sec}} \bmod p$ with $x_i = \text{trunc}\left(\frac{s}{2} \cdot \langle \boldsymbol{c}, \text{sgn}(g_i) \rangle\right)$ for $i = 1, \ldots, k$.
2. $\boldsymbol{v} =$ `concentrate` $(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$
   Adapt $\boldsymbol{v}$ *to get its Lee weight and the sign matches with* $\boldsymbol{c}$ *to prescribed intervals.*
   Iteratively improve $\boldsymbol{v}$ by successively adding $\pm \boldsymbol{g}_i$ for $i = 1, \ldots, k$.

Note: The entries of $\boldsymbol{x}$ and $\boldsymbol{G}_{\text{sec}}$ are small but $p = 65{,}565$ is large!

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\boldsymbol{v} \in \mathbb{F}_p^n$ for a message hash $\boldsymbol{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\boldsymbol{v} = \texttt{simpleSign}(\boldsymbol{c}, \boldsymbol{G}_{\text{sec}})$
   Choose $\boldsymbol{v}$ as a codeword close to $\boldsymbol{c}$.
   Set $\boldsymbol{v} = \boldsymbol{x}\boldsymbol{G}_{\text{sec}} \bmod p$ with $x_i = \operatorname{trunc}\left(\frac{s}{2} \cdot \langle \boldsymbol{c}, \operatorname{sgn}(g_i)\rangle\right)$ for $i = 1, \dots, k$.

2. $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$
   Adapt $\boldsymbol{v}$ to get its Lee weight and the sign matches with $\boldsymbol{c}$ to prescribed intervals.
   Iteratively improve $\boldsymbol{v}$ by successively adding $\pm \boldsymbol{g}_i$ for $i = 1, \dots, k$.

Note: The entries of $\boldsymbol{x}$ and $\boldsymbol{G}_{\text{sec}}$ are small but $p = 65{,}565$ is large!

$\Longrightarrow$ No wrapping modulo p takes place, i.e., all computations are in fact over $\mathbb{Z}$,

FuLeeca Signature Generation [Ritterhoff et al., 2023]

The signature $\mathbf{v} \in \mathbb{F}_p^n$ for a message hash $\mathbf{c} \in \{\pm 1\}^n$ is computed as follows:

1. $\mathbf{v} =$ `simpleSign` $(\mathbf{c}, \mathbf{G}_{\text{sec}})$
   Choose $\mathbf{v}$ *as a codeword close to* $\mathbf{c}$.

   Set $\mathbf{v} = \mathbf{x}\mathbf{G}_{\text{sec}} \bmod p$ with $x_i = \text{trunc}\left(\frac{s}{2} \cdot \langle \mathbf{c}, \text{sgn}(g_i) \rangle\right)$ for $i = 1, \ldots, k$.

2. $\mathbf{v} =$ `concentrate` $(\mathbf{c}, \mathbf{v}, \mathbf{G}_{\text{sec}})$
   Adapt $\mathbf{v}$ *to get its Lee weight and the sign matches with* $\mathbf{c}$ *to prescribed intervals.*

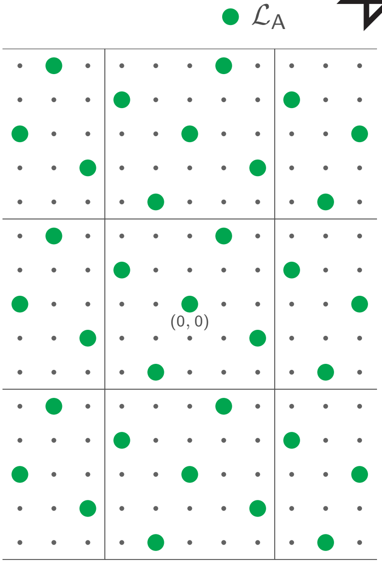
   Iteratively improve $\mathbf{v}$ by successively adding $\pm\mathbf{g}_i$ for $i = 1, \ldots, k$.

Note: The entries of $\mathbf{x}$ and $\mathbf{G}_{\text{sec}}$ are small but $p = 65{,}565$ is large!

$\Longrightarrow$ No wrapping modulo p takes place, i.e., all computations are in fact over $\mathbb{Z}$, and all signatures lie in the central square of $\mathcal{L}_{\text{A}}$.

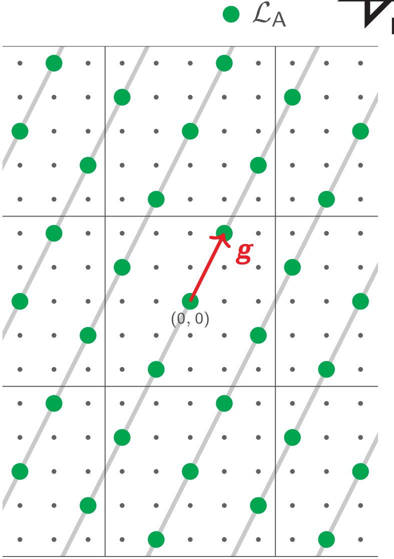
A Lower-Rank Sublattice

$$\mathcal{L}_{\mathrm{A}} = \mathbb{Z}^n \cdot \begin{pmatrix} \mathbf{I}_k & \boldsymbol{A}^{-1}\boldsymbol{B} \\ \mathbf{0} & p\,\mathbf{I}_{n-k} \end{pmatrix} \subset \mathbb{R}^n$$



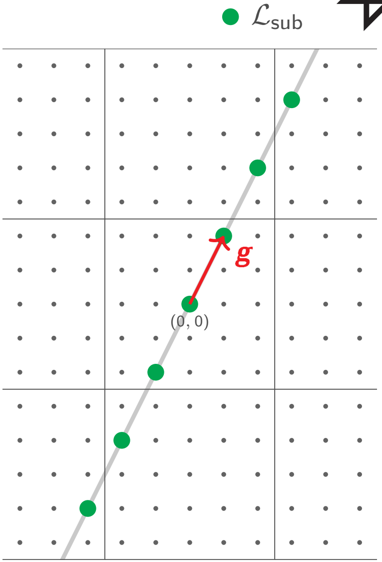
A Lower-Rank Sublattice

$$\mathcal{L}_\mathsf{A} = \mathbb{Z}^n \cdot \begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{0} & p\,\mathbf{I}_{n-k} \end{pmatrix} \subset \mathbb{R}^n$$



A Lower-Rank Sublattice

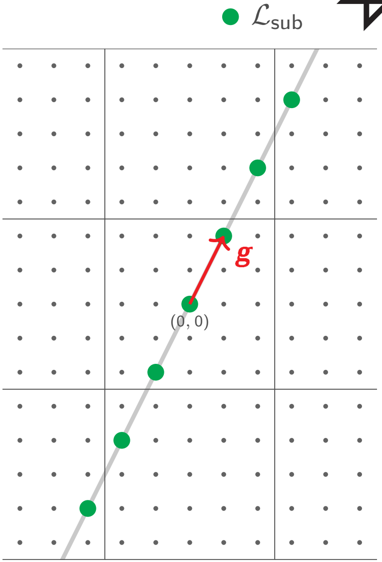
$$\mathcal{L}_{\text{sub}} := \mathbb{Z}^k \cdot \underbrace{\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \end{pmatrix}}_{=\boldsymbol{G}_{\text{sec}}} \subset \mathbb{R}^n$$



A Lower-Rank Sublattice

$$\mathcal{L}_{\text{sub}} := \mathbb{Z}^k \cdot \underbrace{\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \end{pmatrix}}_{= \boldsymbol{G}_{\text{sec}}} \subset \mathbb{R}^n$$

$\mathcal{L}_{\text{sub}}$ has rank $k = n/2$ and lattice techniques can recover short vectors with complexity $O\left(2^{\frac{0.292n}{2}}\right)$.

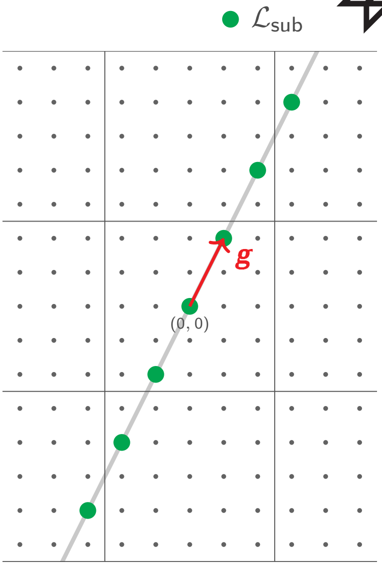


A Lower-Rank Sublattice

$$\mathcal{L}_{\text{sub}} := \mathbb{Z}^k \cdot \underbrace{\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \end{pmatrix}}_{=\boldsymbol{G}_{\text{sec}}} \subset \mathbb{R}^n$$

$\mathcal{L}_{\text{sub}}$ has rank $k = n/2$ and lattice techniques can recover short vectors with complexity $O\left(2^{\frac{0.292n}{2}}\right)$.

In fact, $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$ are unusually short in $\mathcal{L}_{\text{sub}}$ which reduces the complexity to $O\left(2^{\frac{0.292n}{4}}\right)$!



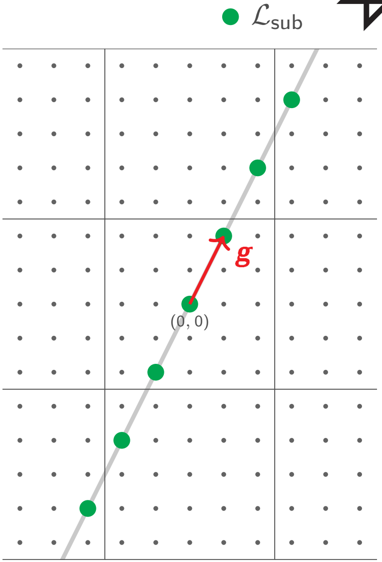
A Lower-Rank Sublattice

$$\mathcal{L}_{\text{sub}} := \mathbb{Z}^k \cdot \underbrace{\begin{pmatrix} \boldsymbol{A} & \boldsymbol{B} \end{pmatrix}}_{=\boldsymbol{G}_{\text{sec}}} \subset \mathbb{R}^n$$

$\mathcal{L}_{\text{sub}}$ has rank $k = n/2$ and lattice techniques can recover short vectors with complexity $O\left(2^{\frac{0.292n}{2}}\right)$.

In fact, $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_k$ are unusually short in $\mathcal{L}_{\text{sub}}$ which reduces the complexity to $O\left(2^{\frac{0.292n}{4}}\right)$!

Remark: The quasicyclic structure of $\mathcal{L}_{\text{sub}}$ enables a polynomial-time quantum attack.



Updated Security Levels for FuLeeca

But $\boldsymbol{G}_{\text{sec}}$ is secret and we need a basis of $\mathcal{L}_{\text{sub}}$ to apply lattice techniques !

Updated Security Levels for FuLeeca

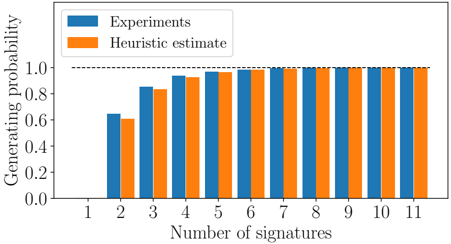
But $\boldsymbol{G}_{\text{sec}}$ is secret and we need a basis of $\mathcal{L}_{\text{sub}}$ to apply lattice techniques!

$\Longrightarrow$ Get a set of generating vectors from signatures and extract a basis.

Updated Security Levels for FuLeeca

But $\boldsymbol{G}_{\text{sec}}$ is secret and we need a basis of $\mathcal{L}_{\text{sub}}$ to apply lattice techniques!

$\Longrightarrow$ Get a set of generating vectors from signatures and extract a basis.

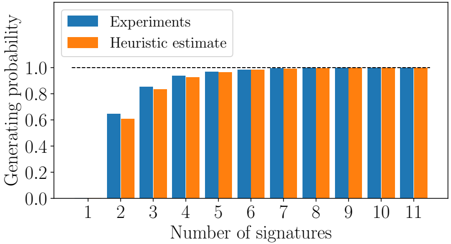


Probability to generate $\mathcal{L}_{\text{sub}}$.

Updated Security Levels for FuLeeca

But $\boldsymbol{G}_{\text{sec}}$ is secret and we need a basis of $\mathcal{L}_{\text{sub}}$ to apply lattice techniques!

$\Longrightarrow$ Get a set of generating vectors from signatures and extract a basis.



Probability to generate $\mathcal{L}_{\text{sub}}$.

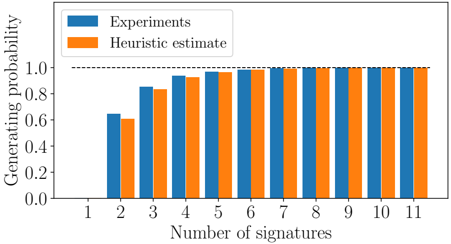
Our leaked-sublattice attack reduces FuLeeca's security levels as follows:

Parameter Set	**Security Level (in bits)**	
	Claimed	Updated
FuLeeca-I	160	111
FuLeeca-III	224	155
FuLeeca-V	288	199

Updated Security Levels for FuLeeca

But $\boldsymbol{G}_{\text{sec}}$ is secret and we need a basis of $\mathcal{L}_{\text{sub}}$ to apply lattice techniques!

$\Longrightarrow$ Get a set of generating vectors from signatures and extract a basis.



Probability to generate $\mathcal{L}_{\text{sub}}$.

Our leaked-sublattice attack reduces FuLeeca's security levels as follows:

Parameter Set	**Security Level (in bits)** Claimed	Updated
FuLeeca-I	160	111
FuLeeca-III	224	155
FuLeeca-V	288	199

$\Longrightarrow$ The NIST standards are not met. ✖

Outline

1. FuLeeca, Codes, and Lattices
2. Leaked-Sublattice Attack
3. Learning Attack

Observed Bias in FuLeeca

Recall: The method $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$ in the signature generation tries to improve $\boldsymbol{v}$ by successively adding $\pm\boldsymbol{g}_i$ for $i = 1, \ldots, k$.

Observed Bias in FuLeeca

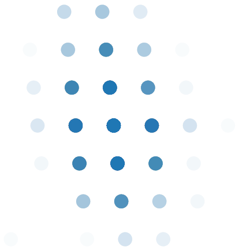
Recall: The method $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$ in the signature generation tries to improve $\boldsymbol{v}$ by successively adding $\pm\boldsymbol{g}_i$ for $i = 1, \ldots, k$.

This process is not properly randomized and the order is always $\pm\boldsymbol{g}_1, \pm\boldsymbol{g}_2, \ldots, \pm\boldsymbol{g}_k$.

Observed Bias in FuLeeca

Recall: The method $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$ in the signature generation tries to improve $\boldsymbol{v}$ by successively adding $\pm \boldsymbol{g}_i$ for $i = 1, \dots, k$.

This process is not properly randomized and the order is always $\pm \boldsymbol{g}_1, \pm \boldsymbol{g}_2, \dots, \pm \boldsymbol{g}_k$.

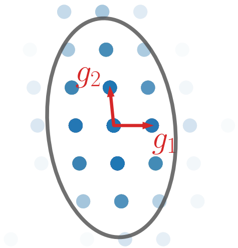


Observed bias for signatures.

Observed Bias in FuLeeca

Recall: The method $\boldsymbol{v} = $ `concentrate` $(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$ in the signature generation tries to improve $\boldsymbol{v}$ by successively adding $\pm\boldsymbol{g}_i$ for $i = 1, \ldots, k$.

This process is not properly randomized and the order is always $\pm\boldsymbol{g}_1, \pm\boldsymbol{g}_2, \ldots, \pm\boldsymbol{g}_k$.

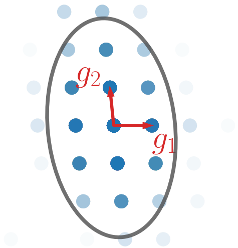


Observed bias for signatures.

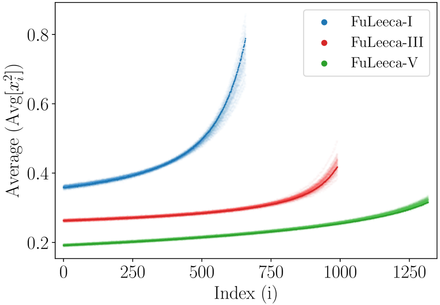
Observed Bias in FuLeeca

Recall: The method $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$ in the signature generation tries to improve $\boldsymbol{v}$ by successively adding $\pm\boldsymbol{g}_i$ for $i = 1, \ldots, k$.

This process is not properly randomized and the order is always $\pm\boldsymbol{g}_1, \pm\boldsymbol{g}_2, \ldots, \pm\boldsymbol{g}_k$.



Observed bias for signatures.

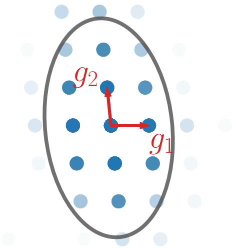


Observed average of x_i^2.

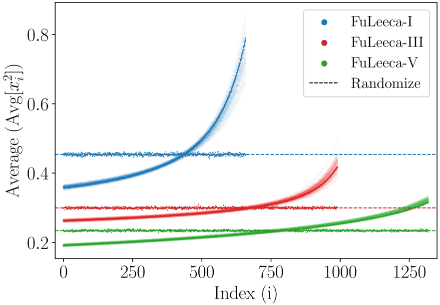
Observed Bias in FuLeeca

Recall: The method $\boldsymbol{v} = \texttt{concentrate}(\boldsymbol{c}, \boldsymbol{v}, \boldsymbol{G}_{\text{sec}})$ in the signature generation tries to improve $\boldsymbol{v}$ by successively adding $\pm\boldsymbol{g}_i$ for $i = 1, \ldots, k$.

This process is not properly randomized and the order is always $\pm\boldsymbol{g}_1, \pm\boldsymbol{g}_2, \ldots, \pm\boldsymbol{g}_k$.



Observed bias for signatures.



Observed average of x_i^2.

Exploiting the Bias

Collect FuLeeca signatures $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ with $\boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{G}_{\text{sec}} = (\underbrace{\boldsymbol{x}_i \boldsymbol{A}}_{=:\boldsymbol{w}_i} \mid \boldsymbol{x}_i \boldsymbol{B})$ for $i = 1, \ldots, N$.

Exploiting the Bias

Collect FuLeeca signatures $\boldsymbol{v}_1, \dots, \boldsymbol{v}_N$ with $\boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{G}_{\text{sec}} = (\underbrace{\boldsymbol{x}_i \boldsymbol{A}}_{=:\boldsymbol{w}_i} \mid \boldsymbol{x}_i \boldsymbol{B})$ for $i = 1, \dots, N$.

$$\operatorname{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_i^\top \boldsymbol{w}_i$$

Exploiting the Bias

Collect FuLeeca signatures $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ with $\boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{G}_{\text{sec}} = (\underbrace{\boldsymbol{x}_i \boldsymbol{A}}_{=:\boldsymbol{w}_i} \mid \boldsymbol{x}_i \boldsymbol{B})$ for $i = 1, \ldots, N$.

$$\mathrm{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_i^\top \boldsymbol{w}_i \quad \approx \quad \mathbb{E}\left[\boldsymbol{w}^\top \boldsymbol{w}\right]$$

Exploiting the Bias

Collect FuLeeca signatures $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ with $\boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{G}_{\text{sec}} = (\underbrace{\boldsymbol{x}_i \boldsymbol{A}}_{=:\boldsymbol{w}_i} \mid \boldsymbol{x}_i \boldsymbol{B})$ for $i = 1, \ldots, N$.

$$\operatorname{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_i^\top \boldsymbol{w}_i \quad \approx \quad \mathbb{E}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \cdot \underbrace{\mathbb{E}\left[\boldsymbol{x}^\top \boldsymbol{x}\right]}_{=\boldsymbol{D}+\boldsymbol{R}} \cdot \boldsymbol{A}$$

with $\boldsymbol{D} = \operatorname{diag}\left(\mathbb{E}\left[x_i^2\right]\right)_{i=1}^{k}$

Exploiting the Bias

Collect FuLeeca signatures $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ with $\boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{G}_{\text{sec}} = (\underbrace{\boldsymbol{x}_i \boldsymbol{A}}_{=:\boldsymbol{w}_i} \mid \boldsymbol{x}_i \boldsymbol{B})$ for $i = 1, \ldots, N$.

$$\text{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_i^\top \boldsymbol{w}_i \approx \mathbb{E}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \cdot \underbrace{\mathbb{E}\left[\boldsymbol{x}^\top \boldsymbol{x}\right]}_{=\boldsymbol{D}+\boldsymbol{R}} \cdot \boldsymbol{A} \quad \text{with } \boldsymbol{D} = \text{diag}\left(\mathbb{E}\left[x_i^2\right]\right)_{i=1}^{k}$$

$$\downarrow$$

$$\text{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \boldsymbol{D} \boldsymbol{A} + \boldsymbol{E}$$

Exploiting the Bias

Collect FuLeeca signatures $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$ with $\boldsymbol{v}_i = \boldsymbol{x}_i \boldsymbol{G}_{\text{sec}} = (\underbrace{\boldsymbol{x}_i \boldsymbol{A}}_{=:\boldsymbol{w}_i} \mid \boldsymbol{x}_i \boldsymbol{B})$ for $i = 1, \ldots, N$.

$$\text{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] := \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{w}_i^\top \boldsymbol{w}_i \approx \mathbb{E}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \cdot \underbrace{\mathbb{E}\left[\boldsymbol{x}^\top \boldsymbol{x}\right]}_{=\boldsymbol{D}+\boldsymbol{R}} \cdot \boldsymbol{A}$$

with $\boldsymbol{D} = \text{diag}\left(\mathbb{E}[x_i^2]\right)_{i=1}^{k}$

$$\text{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \boldsymbol{D} \boldsymbol{A} + \boldsymbol{E}$$

Secret-Key Recovery

$$\operatorname{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \boldsymbol{D} \boldsymbol{A} + \boldsymbol{E}$$

Secret-Key Recovery

$$\mathrm{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \boldsymbol{D}\boldsymbol{A} + \boldsymbol{E}$$

$\boldsymbol{E} = \boldsymbol{0}$	$\boldsymbol{E} \neq \boldsymbol{0}$

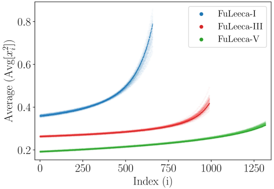
Secret-Key Recovery

$$\mathrm{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \boldsymbol{D} \boldsymbol{A} + \boldsymbol{E}$$

$\boldsymbol{E} = \mathbf{0}$	$\boldsymbol{E} \neq \mathbf{0}$

We can provably recover $\boldsymbol{a}$ since

- $\boldsymbol{A} = \mathrm{Shift}(\boldsymbol{a})$ is circulant and
- $\boldsymbol{D}$ has an increasing diagonal.



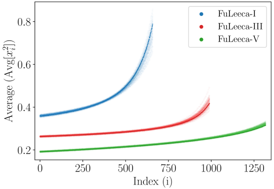
Secret-Key Recovery

$$\mathrm{Avg}\left[\boldsymbol{w}^\top \boldsymbol{w}\right] = \boldsymbol{A}^\top \boldsymbol{D}\boldsymbol{A} + \boldsymbol{E}$$

$\boldsymbol{E} = \mathbf{0}$	$\boldsymbol{E} \neq \mathbf{0}$

We can provably recover $\boldsymbol{a}$ since

- $\boldsymbol{A} = \mathrm{Shift}(\boldsymbol{a})$ is circulant and
- $\boldsymbol{D}$ has an increasing diagonal.

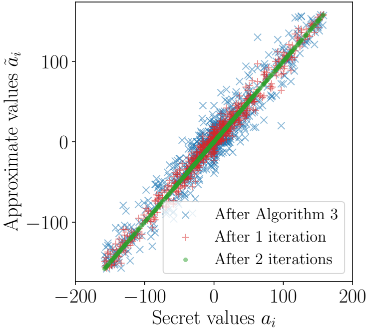


We recover $\boldsymbol{a}$ with high probability:

1. Get an approximation of $\boldsymbol{a}$.
2. Turn the guess into an exact solution iteratively.

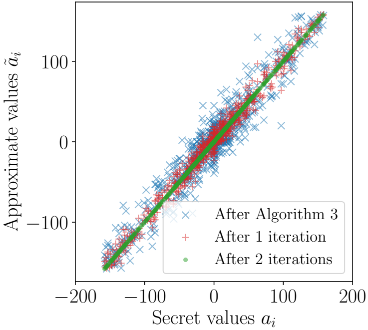
Success of the Learning Attack

From approximation to exact solution.



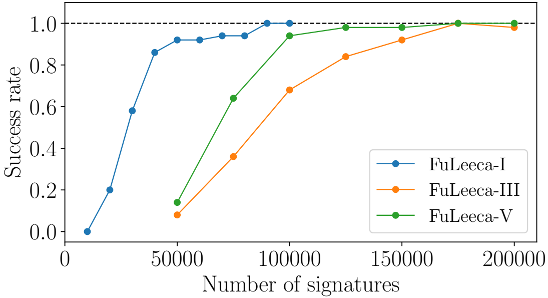
Success of the Learning Attack

From approximation to exact solution.



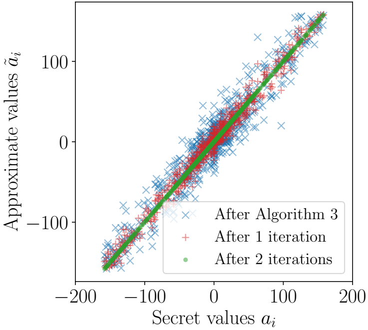
Success rate of the learning attack.

Averaged over 50 keys for each parameter set.



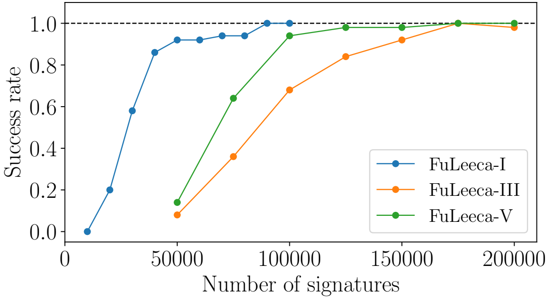
Success of the Learning Attack

From approximation to exact solution.



Success rate of the learning attack.

Averaged over 50 keys for each parameter set.



$\Longrightarrow$ 175,000 signatures are enough to fully break FuLeeca!

Summary

FuLeeca:

Summary

FuLeeca: is broken. 💣

Summary

FuLeeca: is broken.

	few signatures ($\ll$ 100)	**many signatures** ($\leq$ 175,000)
classical attack	leaked-sublattice attack (reduced security)	learning attack (full break)
quantum attack	ideal-structure attack (full break)	$\leftarrow$ see this attack

Summary

FuLeeca: is broken. 💣

- A too large p prevents wrapping modulo p and thus leaks a lower-rank sublattice.
- Signatures leak information about $\boldsymbol{G}_{\text{sec}}$.

	few signatures ($\ll 100$)	**many signatures** ($\leq 175{,}000$)
classical attack	leaked-sublattice attack (reduced security)	learning attack (full break)
quantum attack	ideal-structure attack (full break)	← see this attack

Summary

FuLeeca: is broken. 💣

- A too large p prevents wrapping modulo p and thus leaks a lower-rank sublattice.
- Signatures leak information about $\boldsymbol{G}_{\text{sec}}$.

	few signatures ($\ll 100$)	**many signatures** ($\leq 175{,}000$)
classical attack	leaked-sublattice attack (reduced security)	learning attack (full break)
quantum attack	ideal-structure attack (full break)	← see this attack

The bigger picture:

- Codes and lattices might be closer than you think, especially for the Lee metric.
  $\Longrightarrow$ Take this into account for the design of new schemes.
- How can non-leakage be provably achieved for the Lee metric?

Summary

FuLeeca: is broken.

- A too large p prevents wrapping modulo p and thus leaks a lower-rank sublattice.
- Signatures leak information about $\boldsymbol{G}_{\text{sec}}$.

	few signatures ($\ll 100$)	**many signatures** ($\leq$ 175,000)
classical attack	leaked-sublattice attack (reduced security)	learning attack (full break)
quantum attack	ideal-structure attack (full break)	$\leftarrow$ see this attack

The bigger picture:

- Codes and lattices might be closer than you think, especially for the Lee metric.
  $\Longrightarrow$ Take this into account for the design of new schemes.
- How can non-leakage be provably achieved for the Lee metric?

Find FuLeakage here:

- `ia.cr/2024/353`
- </> `artifacts.iacr.org/crypto/2024/a12`

References

[Ritterhoff et al., 2023] Ritterhoff, S., Maringer, G., Bitzer, S., Weger, V., Karl, P., Schamberger, T., Schupp, J., and Wachter-Zeh, A. (2023).
FuLeeca: A Lee-based signature scheme.
In *Code-Based Cryptography - 11th International Workshop, CBCrypto 2023*, volume 14311 of *Lecture Notes in Computer Science*, pages 56–83. Springer.
See `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/FuLeeca-spec-web.pdf` for the submission to NIST's call for additional digital signature schemes.