Algebraic and Combinatorial Algorithms for the Matrix Code Equivalence Problem

Monika Trimoska

including multiple joint works with the MEDS team

Rank-Metric Codes and Network Coding, SIAM-AG 2025
July 11, Madison, Wisconsin

TU/e

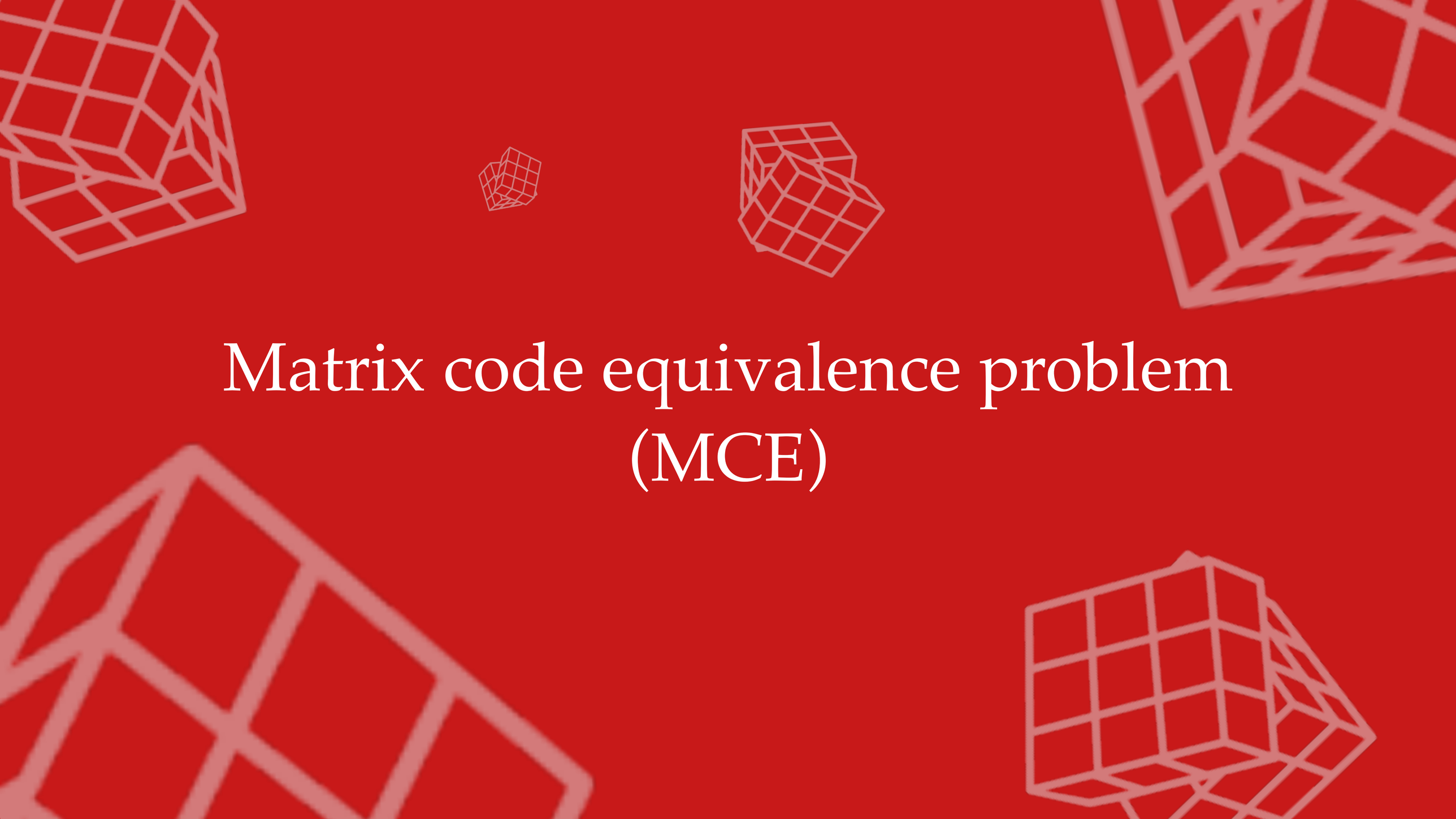
*Animated version at: https://mtrimoska.com/slides/SIAM25/index.html

The MEDS team

CD MEDS CD

Matrix Equivalence Digital Signature Scheme

Tung Chou, Academia Sinica, Taipei, Taiwan
Ruben Niederhagen, Academia Sinica, Taipei, Taiwan, and University of Southern Denmark, Odense, Denmark
Edoardo Persichetti, Florida Atlantic University, Boca Raton, USA, and Sapienza University, Rome, Italy
Lars Ran, Radboud Universiteit, Nijmegen, The Netherlands
Tovohery Hajatiana Randrianarisoa, Umeå University, Umeå, Sweden
Krijn Reijnders, Radboud Universiteit, Nijmegen, The Netherlands
Simona Samardjiska, Radboud Universiteit, Nijmegen, The Netherlands
Monika Trimoska, Radboud Universiteit, Nijmegen, The Netherlands



Matrix code equivalence problem (MCE)

Matrix (rank-metric) codes

Matrix code

A matrix code $\mathscr{C}$ over $\mathbb{F}_q$ is a k-dimensional $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^{m\times n}$.

Matrix (rank-metric) codes

Matrix code

A matrix code $\mathscr{C}$ over $\mathbb{F}_q$ is a k-dimensional $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^{m\times n}$.

Basis of a matrix code

The basis of a matrix code $\mathscr{C}$ is given by the k-tuple $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$.

Matrix (rank-metric) codes

Matrix code

A matrix code $\mathscr{C}$ over $\mathbb{F}_q$ is a k-dimensional $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^{m\times n}$.

Basis of a matrix code

The basis of a matrix code $\mathscr{C}$ is given by the k-tuple $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$.

Rank metric

For $\mathbf{C} \in \mathbb{F}_q^{m\times n}$, the rank weight of $\mathbf{C}$ is given by the rank of $\mathbf{C}$, aka.

$$\mathrm{wt}(\mathbf{C}) = \mathrm{rk}(\mathbf{C}).$$

Matrix (rank-metric) codes

Example. $q = 13, \quad m = 4, \quad n = 6, \quad k = 5$

$$\mathbf{C} = \lambda_1 \cdot \begin{pmatrix} 2 & 8 & 10 & 4 & 5 & 7 \\ 1 & 11 & 7 & 9 & 6 & 12 \\ 3 & 0 & 13 & 5 & 4 & 8 \\ 9 & 6 & 3 & 2 & 10 & 11 \end{pmatrix} + \lambda_2 \cdot \begin{pmatrix} 12 & 0 & 4 & 11 & 9 & 3 \\ 5 & 6 & 8 & 13 & 2 & 1 \\ 10 & 7 & 3 & 9 & 4 & 6 \\ 2 & 5 & 11 & 8 & 1 & 10 \end{pmatrix} + \lambda_3 \cdot \begin{pmatrix} 5 & 2 & 9 & 11 & 4 & 8 \\ 3 & 7 & 1 & 10 & 12 & 0 \\ 6 & 9 & 2 & 13 & 11 & 8 \\ 1 & 5 & 6 & 3 & 10 & 7 \end{pmatrix} + \lambda_4 \cdot \begin{pmatrix} 9 & 4 & 6 & 1 & 13 & 2 \\ 8 & 0 & 5 & 12 & 6 & 11 \\ 3 & 7 & 10 & 9 & 4 & 5 \\ 2 & 8 & 11 & 3 & 7 & 1 \end{pmatrix} + \lambda_5 \cdot \begin{pmatrix} 7 & 10 & 4 & 6 & 8 & 3 \\ 1 & 5 & 2 & 11 & 9 & 0 \\ 13 & 7 & 6 & 4 & 12 & 2 \\ 8 & 3 & 1 & 9 & 5 & 10 \end{pmatrix} \qquad \lambda_i \in \mathbb{F}_q$$

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

→ Multiply a codeword on the right by any $\mathbf{M} \in \mathbb{F}_q^{n \times r}$

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

→ Multiply a codeword on the right by any $\mathbf{M} \in \mathbb{F}_q^{n \times r}$ 

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

- Multiply a codeword on the right by any $\mathbf{M} \in \mathbb{F}_q^{n \times r}$ ✗
- Multiply a codeword on the right by $\mathbf{B} \in \mathrm{GL}_n$

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

- Multiply a codeword on the right by any $\mathbf{M} \in \mathbb{F}_q^{n \times r}$ ✗
- Multiply a codeword on the right by $\mathbf{B} \in \mathrm{GL}_n$ ✓

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

- Multiply a codeword on the right by any $\mathbf{M} \in \mathbb{F}_q^{n \times r}$ ✗
- Multiply a codeword on the right by $\mathbf{B} \in \mathrm{GL}_n$ ✓
- Multiply a codeword on the left by $\mathbf{A} \in \mathrm{GL}_m$

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

- Multiply a codeword on the right by any $\mathbf{M} \in \mathbb{F}_q^{n \times r}$ ✗
- Multiply a codeword on the right by $\mathbf{B} \in \mathrm{GL}_n$ ✓
- Multiply a codeword on the left by $\mathbf{A} \in \mathrm{GL}_m$ ✓

Matrix code equivalence

Isometry

An isometry (for our purposes) between two codes $\mathscr{C}$ and $\mathscr{D}$ is a linear map $\mu : \mathscr{C} \to \mathscr{D}$ that preserves the metric.

In this case: an isometry preserves the rank weight of codewords.

Which linear transformations preserve the rank?

- Multiply a codeword on the right by any $\mathbf{M} \in \mathbb{F}_q^{n \times r}$ ✗
- Multiply a codeword on the right by $\mathbf{B} \in \mathrm{GL}_n$ ✓
- Multiply a codeword on the left by $\mathbf{A} \in \mathrm{GL}_m$ ✓
- Take the transposition of a codeword (only when $m = n$, does not make the equivalence problem harder) ✓

Matrix code equivalence

The Matrix Code Equivalence (MCE) problem

Input: Two k-dimensional matrix codes $\mathscr{C}, \mathscr{D} \subset \mathbb{F}_q^{m \times n}$ for two matrix codes $\mathscr{C}$ and $\mathscr{D}$.
Question: Find - if any - a map $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A} \in \mathrm{GL}_m(\mathbb{F}_q)$ and $\mathbf{B} \in \mathrm{GL}_n(\mathbb{F}_q)$ such that for all $\mathbf{C} \in \mathscr{C}$, it holds that $\mathbf{ACB} \in \mathscr{D}$.

Matrix code equivalence

The MCE problem in matrix form

Let $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ be a basis of code $\mathscr{C}$ and let $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ be a basis of code $\mathscr{D}$. Find $\mathbf{A} \in \mathrm{GL}_m(\mathbb{F}_q)$, $\mathbf{B} \in \mathrm{GL}_n(\mathbb{F}_q)$ and $\mathbf{T} \in \mathrm{GL}_k(\mathbb{F}_q)$ such that

$$\mathbf{D}^{(i)} = \sum_{1 \leq j \leq k} t_{j,i} \mathbf{A} \mathbf{C}^{(j)} \mathbf{B}, \quad \forall 1 \leq i \leq k$$

Matrix code equivalence

The MCE problem in matrix form

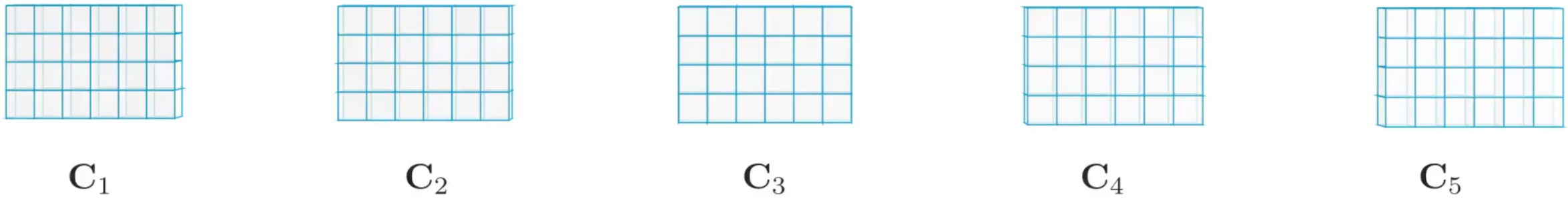
Let $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ be a basis of code $\mathscr{C}$ and let $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ be a basis of code $\mathscr{D}$. Find $\mathbf{A} \in \mathrm{GL}_m(\mathbb{F}_q)$, $\mathbf{B} \in \mathrm{GL}_n(\mathbb{F}_q)$ and $\mathbf{T} \in \mathrm{GL}_k(\mathbb{F}_q)$ such that

$$\mathbf{D}^{(i)} = \sum_{1 \le j \le k} t_{j,i} \mathbf{A} \mathbf{C}^{(j)} \mathbf{B}, \quad \forall 1 \le i \le k$$

change of basis

From matrix codes to 3-tensors

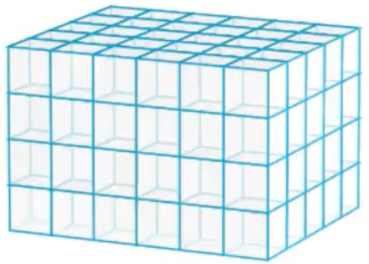
We can think of a matrix code as a 3-tensor over $\mathbb{F}_q$.



From matrix codes to 3-tensors

We can think of a matrix code as a 3-tensor over $\mathbb{F}_q$.

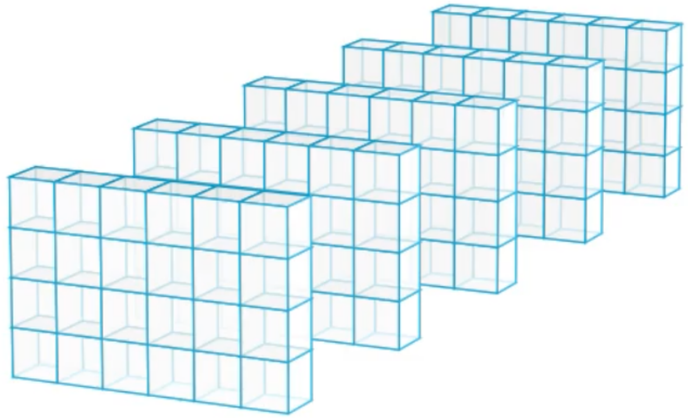
$$\mathcal{C} \subseteq \mathbb{F}_q^{m \times n \times k}$$



From matrix codes to 3-tensors

Viewed as a 3-tensor, we can see $\mathscr{C}$ from three directions

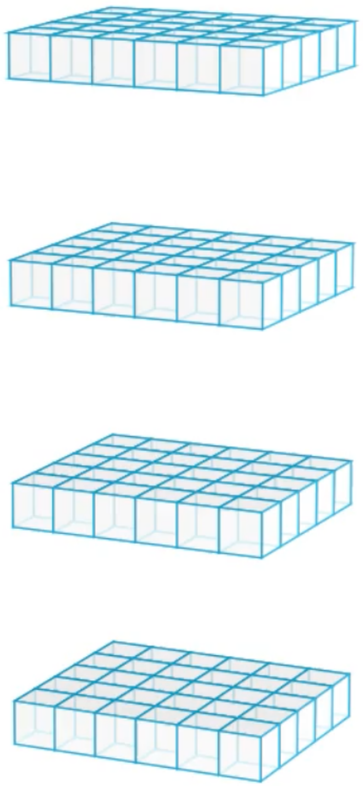
- a k-dimensional code in $\mathbb{F}_q^{m\times n}$
- an m-dimensional code in $\mathbb{F}_q^{n\times k}$
- an n-dimensional code in $\mathbb{F}_q^{m\times k}$



From matrix codes to 3-tensors

Viewed as a 3-tensor, we can see $\mathscr{C}$ from three directions

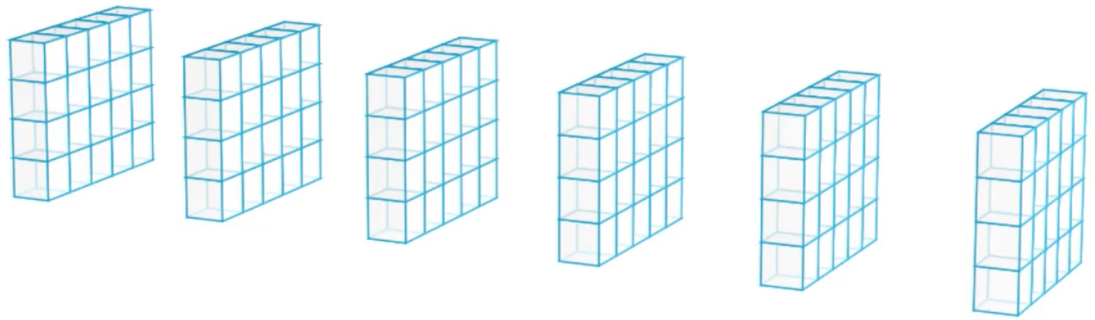
- a k-dimensional code in $\mathbb{F}_q^{m\times n}$
- an m-dimensional code in $\mathbb{F}_q^{n\times k}$
- an n-dimensional code in $\mathbb{F}_q^{m\times k}$



From matrix codes to 3-tensors

Viewed as a 3-tensor, we can see $\mathscr{C}$ from three directions

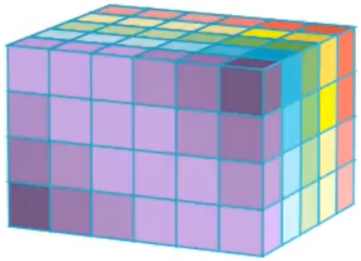
- a k-dimensional code in $\mathbb{F}_q^{m \times n}$
- an m-dimensional code in $\mathbb{F}_q^{n \times k}$
- an n-dimensional code in $\mathbb{F}_q^{m \times k}$



Tensor isomorphism

The equivalence then becomes tensor isomorphism.

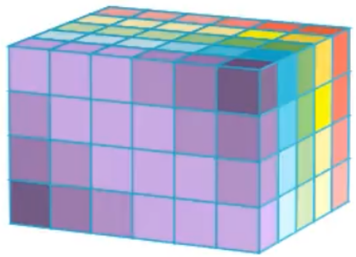
$$\mathcal{C} \subseteq \mathbb{F}_q^{m \times n \times k}$$



Tensor isomorphism

The equivalence then becomes tensor isomorphism.

$\mathbf{T} \in \mathrm{GL}_k(q)$



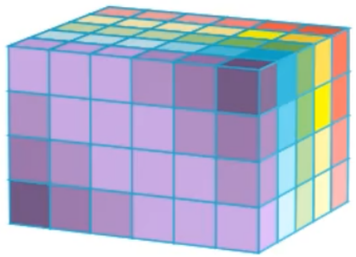
$\mathbf{A} \in \mathrm{GL}_m(q)$

$\mathbf{B} \in \mathrm{GL}_n(q)$

Tensor isomorphism

The equivalence then becomes tensor isomorphism.

$\mathbf{T} \in \mathrm{GL}_k(q)$



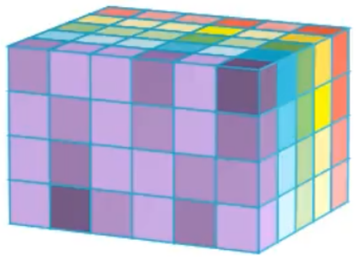
$\mathbf{A} \in \mathrm{GL}_m(q)$

$\mathbf{B} \in \mathrm{GL}_n(q)$

Tensor isomorphism

The equivalence then becomes tensor isomorphism.

$\mathbf{T} \in \mathrm{GL}_k(q)$



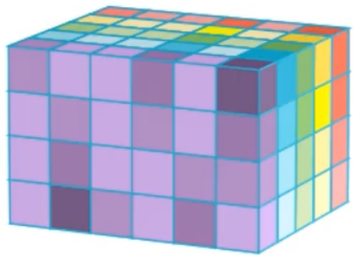
$\mathbf{A} \in \mathrm{GL}_m(q)$

$\mathbf{B} \in \mathrm{GL}_n(q)$

Tensor isomorphism

The equivalence then becomes tensor isomorphism.

$\mathbf{T} \in \mathrm{GL}_k(q)$



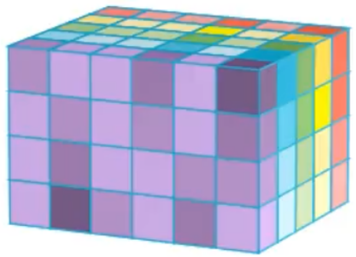
$\mathbf{A} \in \mathrm{GL}_m(q)$

$\mathbf{B} \in \mathrm{GL}_n(q)$

Tensor isomorphism

The equivalence then becomes tensor isomorphism.

$\mathbf{T} \in \mathrm{GL}_k(q)$



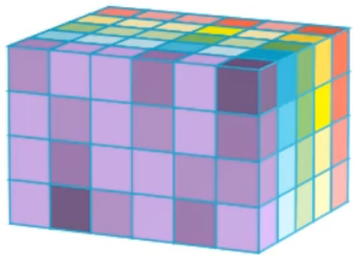
$\mathbf{A} \in \mathrm{GL}_m(q)$

$\mathbf{B} \in \mathrm{GL}_n(q)$

Tensor isomorphism

The equivalence then becomes tensor isomorphism.

$\mathbf{T} \in \mathrm{GL}_k(q)$



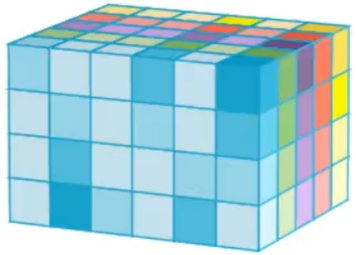
$\mathbf{A} \in \mathrm{GL}_m(q)$

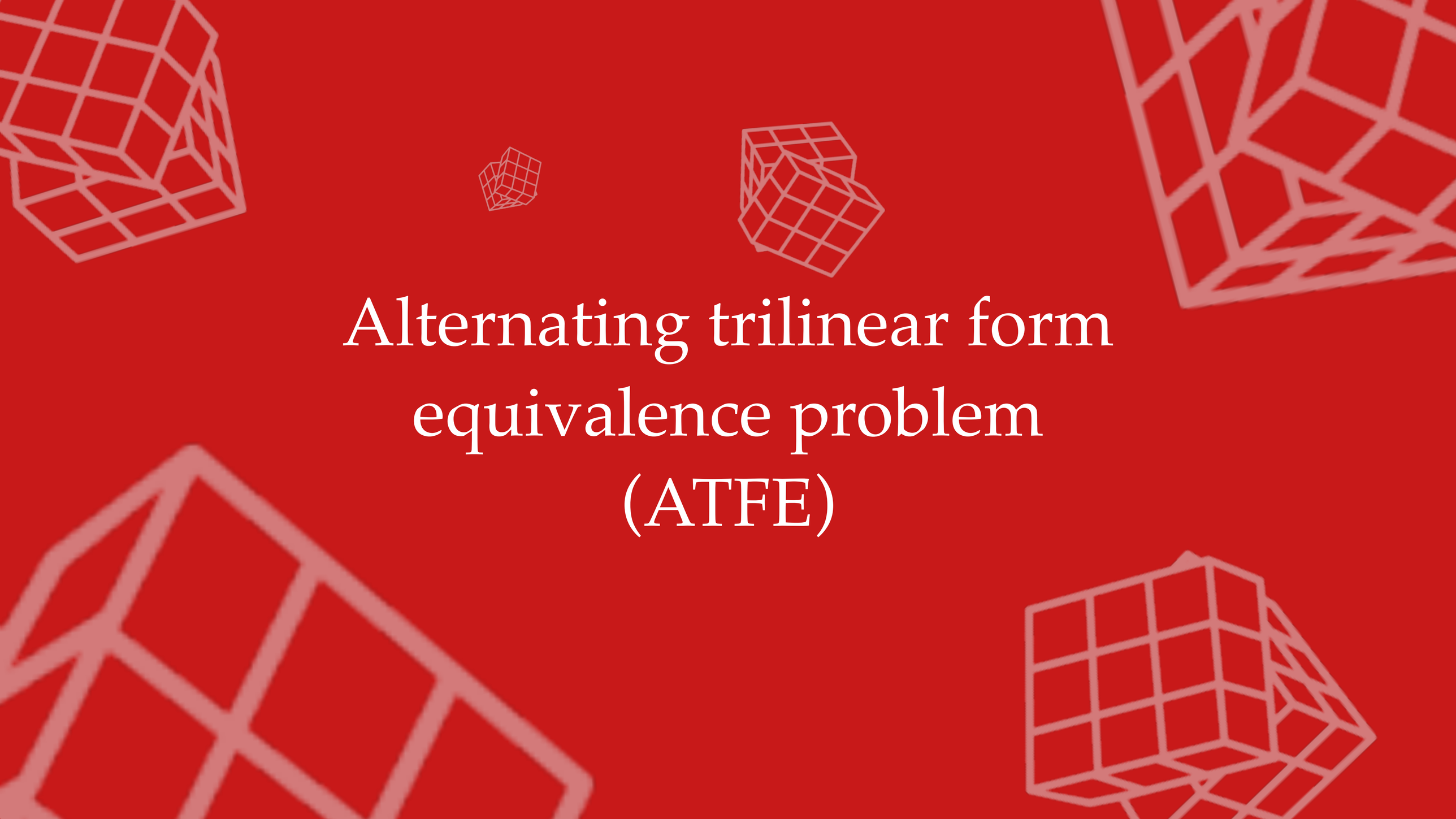
$\mathbf{B} \in \mathrm{GL}_n(q)$

Tensor isomorphism

The equivalence then becomes tensor isomorphism.

$$\mathcal{D} \subseteq \mathbb{F}_q^{m \times n \times k}$$





Alternating trilinear form equivalence problem (ATFE)

Multilinear forms

k-linear form

A k-linear form is a function $\phi : \mathbb{F}_q^n \times \ldots \times \mathbb{F}_q^n \to \mathbb{F}_q$ that is linear in each argument.

if we fix $k-1$ arguments, it is linear in the remaining argument.

Multilinear forms

k-linear form

A k-linear form is a function $\phi : \mathbb{F}_q^n \times \ldots \times \mathbb{F}_q^n \to \mathbb{F}_q$ that is linear in each argument.

if we fix $k-1$ arguments, it is linear in the remaining argument.

Alternating property

ϕ is alternating : $\phi(\mathbf{x}_1, \ldots, \mathbf{x}_k) = 0$ whenever $\mathbf{x}_i = \mathbf{x}_j$ for some $i \neq j$.

Multilinear forms

k-linear form

A k-linear form is a function $\phi : \mathbb{F}_q^n \times \ldots \times \mathbb{F}_q^n \to \mathbb{F}_q$ that is linear in each argument.

↳ if we fix $k-1$ arguments, it is linear in the remaining argument.

Alternating property

ϕ is alternating : $\phi(\mathbf{x}_1, \ldots, \mathbf{x}_k) = 0$ whenever $\mathbf{x}_i = \mathbf{x}_j$ for some $i \neq j$.

→ We focus on trilinear forms: $\phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$.

Array representation

Bilinear form:

$$\phi(\mathbf{x}, \mathbf{y}) = \sum \gamma_{i,j} x_i y_j$$

$\mathbf{x}$

x_1	x_2	x_3	x_4	x_5	x_6

$\mathbf{B}$

$\gamma_{1,1}$	$\gamma_{1,2}$	$\gamma_{1,3}$	$\gamma_{1,4}$	$\gamma_{1,5}$	$\gamma_{1,6}$
$\gamma_{2,1}$	$\gamma_{2,2}$	$\gamma_{2,3}$	$\gamma_{2,4}$	$\gamma_{2,5}$	$\gamma_{2,6}$
$\gamma_{3,1}$	$\gamma_{3,2}$	$\gamma_{3,3}$	$\gamma_{3,4}$	$\gamma_{3,5}$	$\gamma_{3,6}$
$\gamma_{4,1}$	$\gamma_{4,2}$	$\gamma_{4,3}$	$\gamma_{4,4}$	$\gamma_{4,5}$	$\gamma_{4,6}$
$\gamma_{5,1}$	$\gamma_{5,2}$	$\gamma_{5,3}$	$\gamma_{5,4}$	$\gamma_{5,5}$	$\gamma_{5,6}$
$\gamma_{6,1}$	$\gamma_{6,2}$	$\gamma_{6,3}$	$\gamma_{6,4}$	$\gamma_{6,5}$	$\gamma_{6,6}$

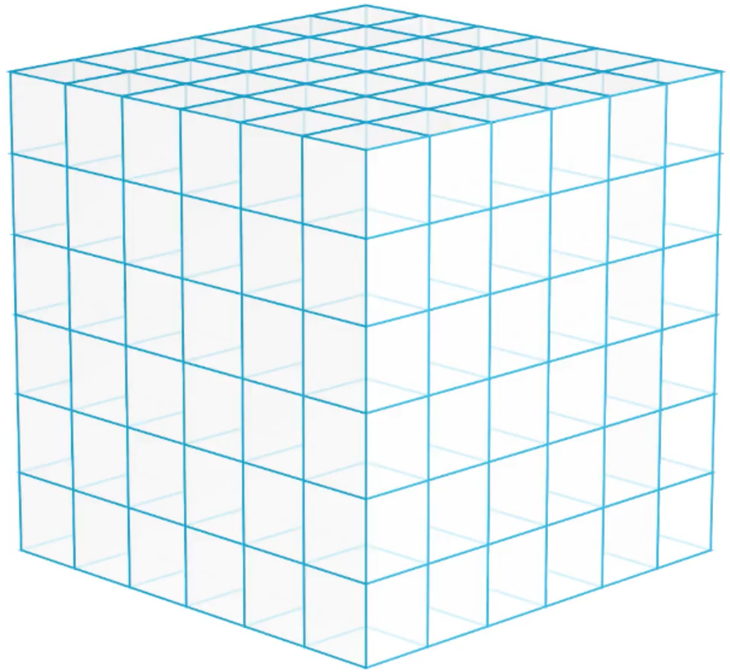
$\mathbf{y}$

y_1
y_2
y_3
y_4
y_5
y_6

so with $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$, we get $\phi(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{B} \mathbf{y}$.

Array representation

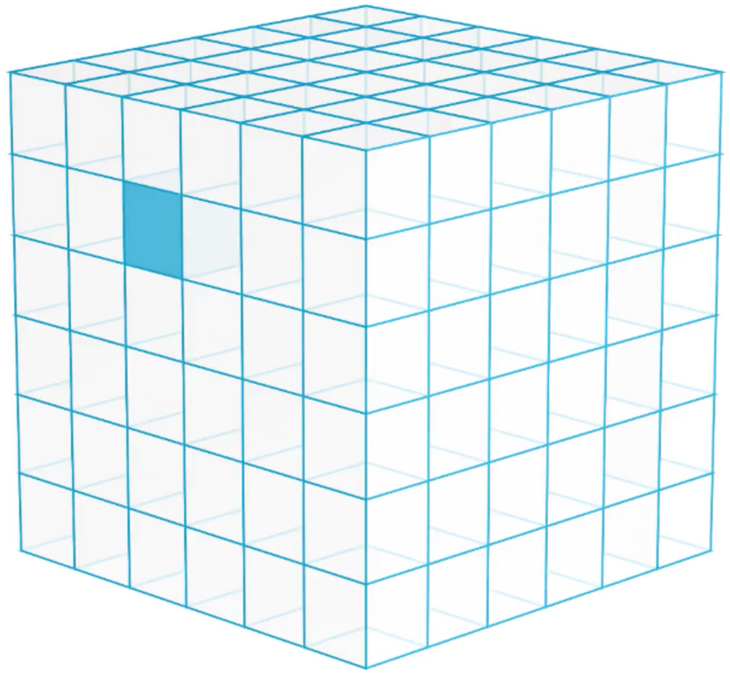
Trilinear form:



Array representation

Trilinear form:

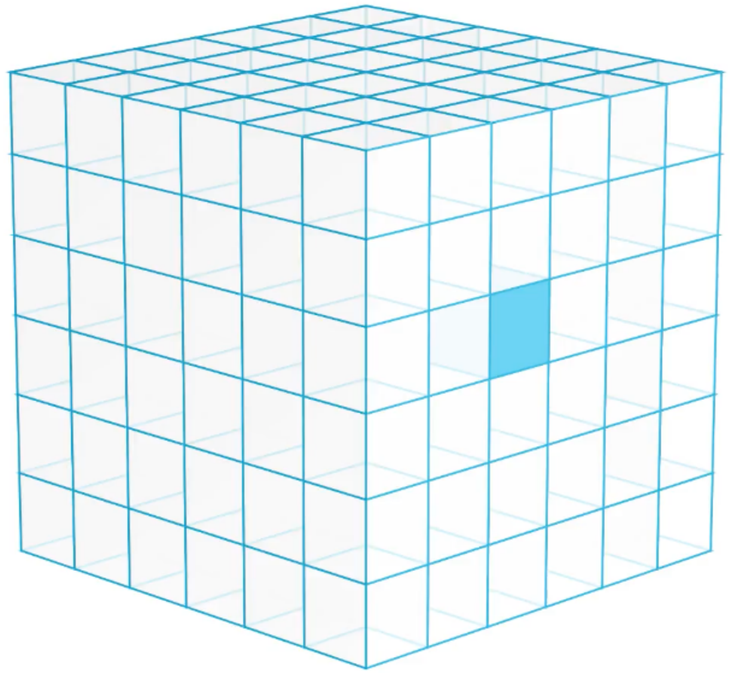
$\hookrightarrow \quad \gamma_{2,3,1}$



Array representation

Trilinear form:

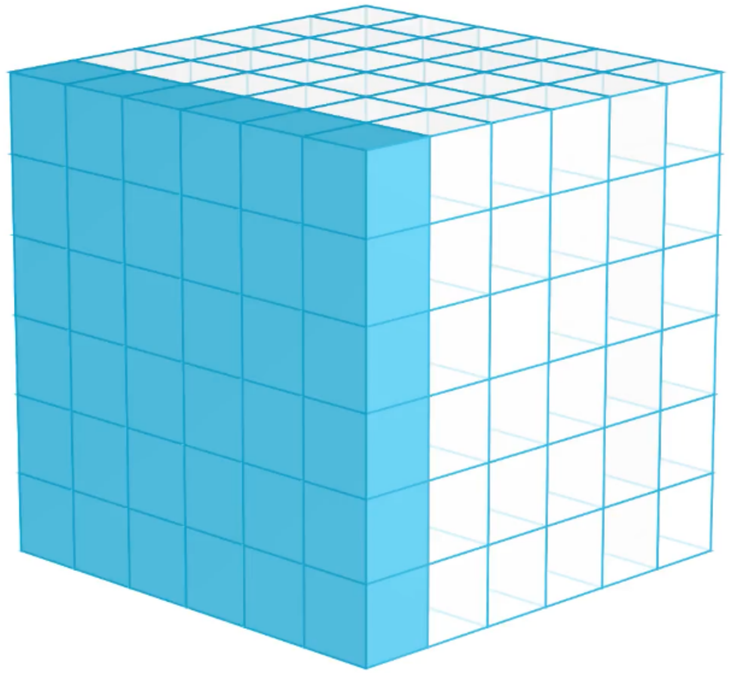
$\hookrightarrow \quad \gamma_{3,6,3}$



Array representation

Trilinear form:

$\hookrightarrow \quad \gamma_{*,*,1}$



Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs} (\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs}(\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

$$\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^* \text{ sends } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ to } \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_s & y_s & z_s \end{vmatrix}$$

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs}(\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

$$\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^* \text{ sends } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ to } \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_s & y_s & z_s \end{vmatrix} = x_i y_j z_s - x_i y_s z_j + x_s y_i z_j - x_j y_i z_s + x_j y_s z_i - x_s y_j z_i$$

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs}(\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

* when $\mathbf{x} = \mathbf{y}$

$$\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^* \text{ sends } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ to } \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_s & y_s & z_s \end{vmatrix} = x_i y_j z_s - x_i y_s z_j + x_s y_i z_j - x_j y_i z_s + x_j y_s z_i - x_s y_j z_i$$

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs}(\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

* when $\mathbf{y} = \mathbf{z}$

$$\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^* \text{ sends } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ to } \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_s & y_s & z_s \end{vmatrix} = x_i y_j z_s - x_i y_s z_j + x_s y_i z_j - x_j y_i z_s + x_j y_s z_i - x_s y_j z_i$$

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs}(\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

* when $\mathbf{x} = \mathbf{z}$

$$\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^* \text{ sends } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ to } \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_s & y_s & z_s \end{vmatrix} = x_i y_j z_s - x_i y_s z_j + x_s y_i z_j - x_j y_i z_s + x_j y_s z_i - x_s y_j z_i$$

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs}(\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

$$\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^* \text{ sends } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ to } \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_s & y_s & z_s \end{vmatrix} = x_i y_j z_s - x_i y_s z_j + x_s y_i z_j - x_j y_i z_s + x_j y_s z_i - x_s y_j z_i$$

Alternating trilinear form

The alternating property

$\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0$ whenever $\mathbf{x} = \mathbf{y}$ or $\mathbf{x} = \mathbf{z}$ or $\mathbf{y} = \mathbf{z}$.

Compact representation

$\sum_{1 \leqslant i < j < s \leqslant n} \gamma_{ijs} (\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^*)$, where $\wedge$ denotes the wedge product.

$$\mathbf{e}_i^* \wedge \mathbf{e}_j^* \wedge \mathbf{e}_s^* \text{ sends } (\mathbf{x}, \mathbf{y}, \mathbf{z}) \text{ to } \begin{vmatrix} x_i & y_i & z_i \\ x_j & y_j & z_j \\ x_s & y_s & z_s \end{vmatrix} = x_i y_j z_s - x_i y_s z_j + x_s y_i z_j - x_j y_i z_s + x_j y_s z_i - x_s y_j z_i$$

↳ Stored using $\binom{n}{3}$ entries, instead of n^3.

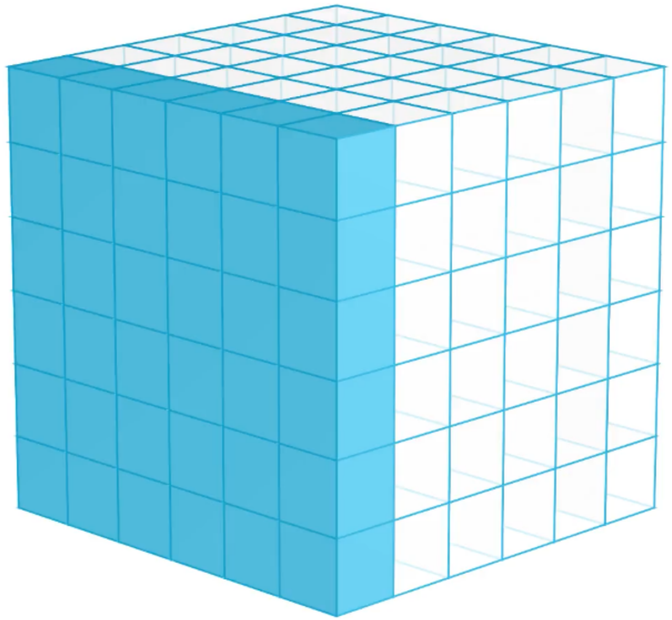
Alternating trilinear form equivalence

The Alternating Trilinear Form Equivalence (ATFE) problem

Input: Two alternating trilinear forms ϕ, ψ.
Question: Find - if any - $\mathbf{A} \in \mathrm{GL}_n(\mathbb{F}_q)$ such that $\phi(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \psi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{Az})$.

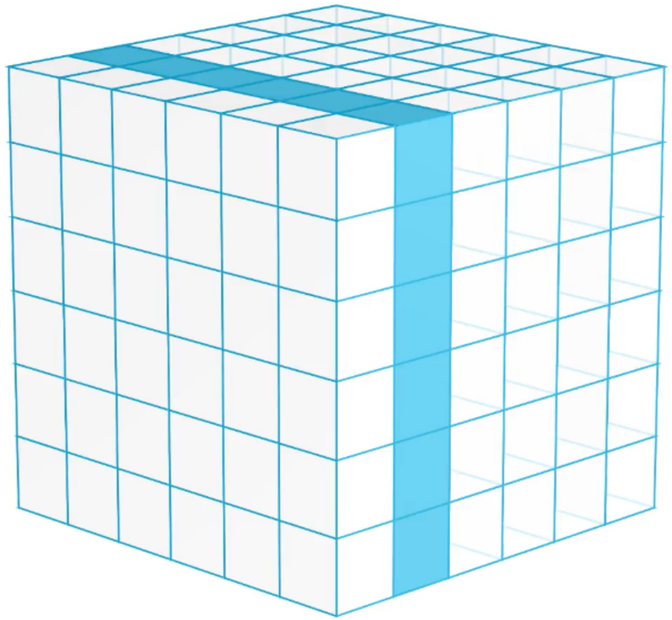
(Alternating) trilinear form $\longrightarrow$ matrix code

Let $\phi^{(1)}(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}, \mathbf{y}, \mathbf{e}_1)$



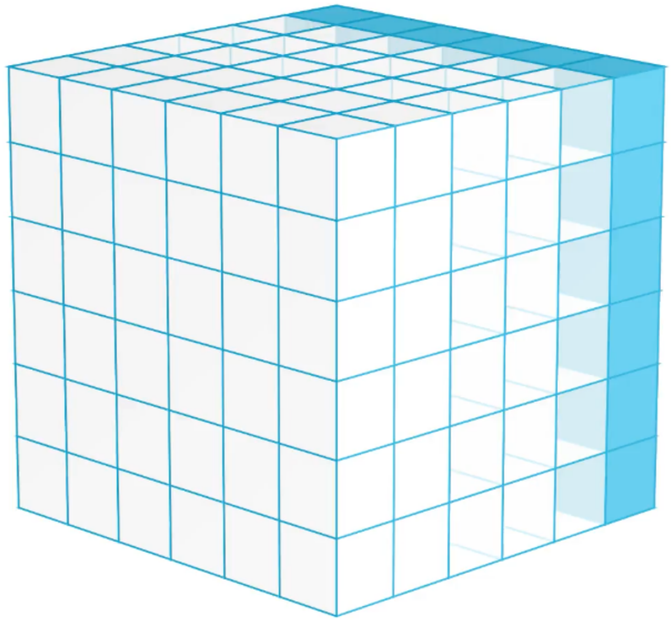
(Alternating) trilinear form $\longrightarrow$ matrix code

Let $\phi^{(2)}(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}, \mathbf{y}, \mathbf{e}_2)$

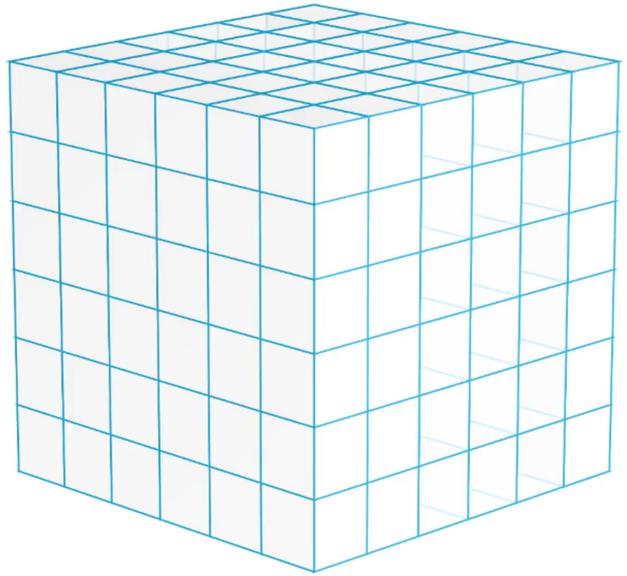


(Alternating) trilinear form $\longrightarrow$ matrix code

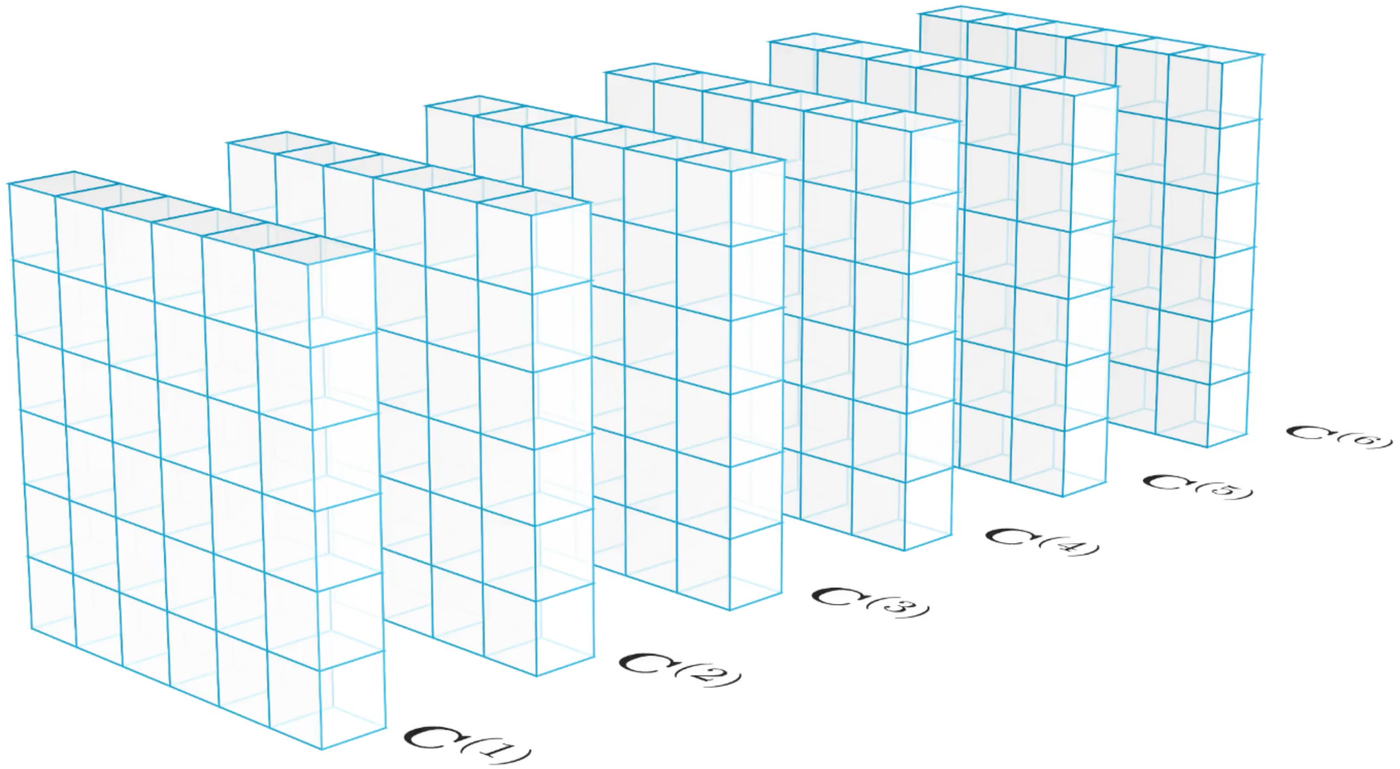
..Let $\phi^{(n)}(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}, \mathbf{y}, \mathbf{e}_n)$



(Alternating) trilinear form $\longrightarrow$ matrix code



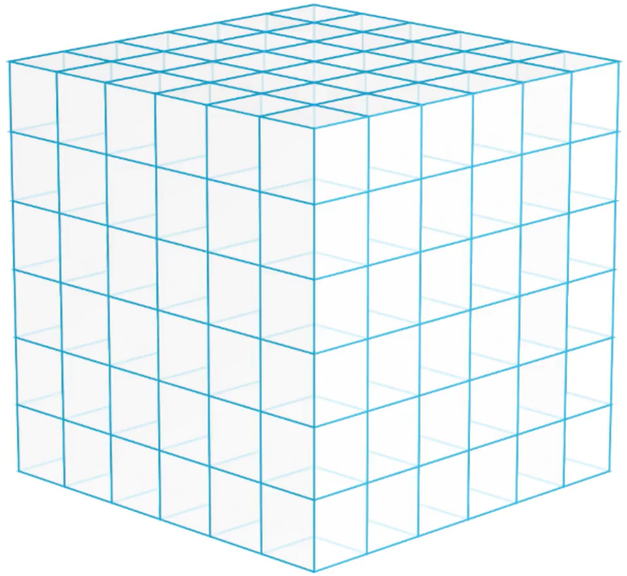
(Alternating) trilinear form ⟶ matrix code



Take $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(n)})$ as a basis of an n-dimensional matrix code

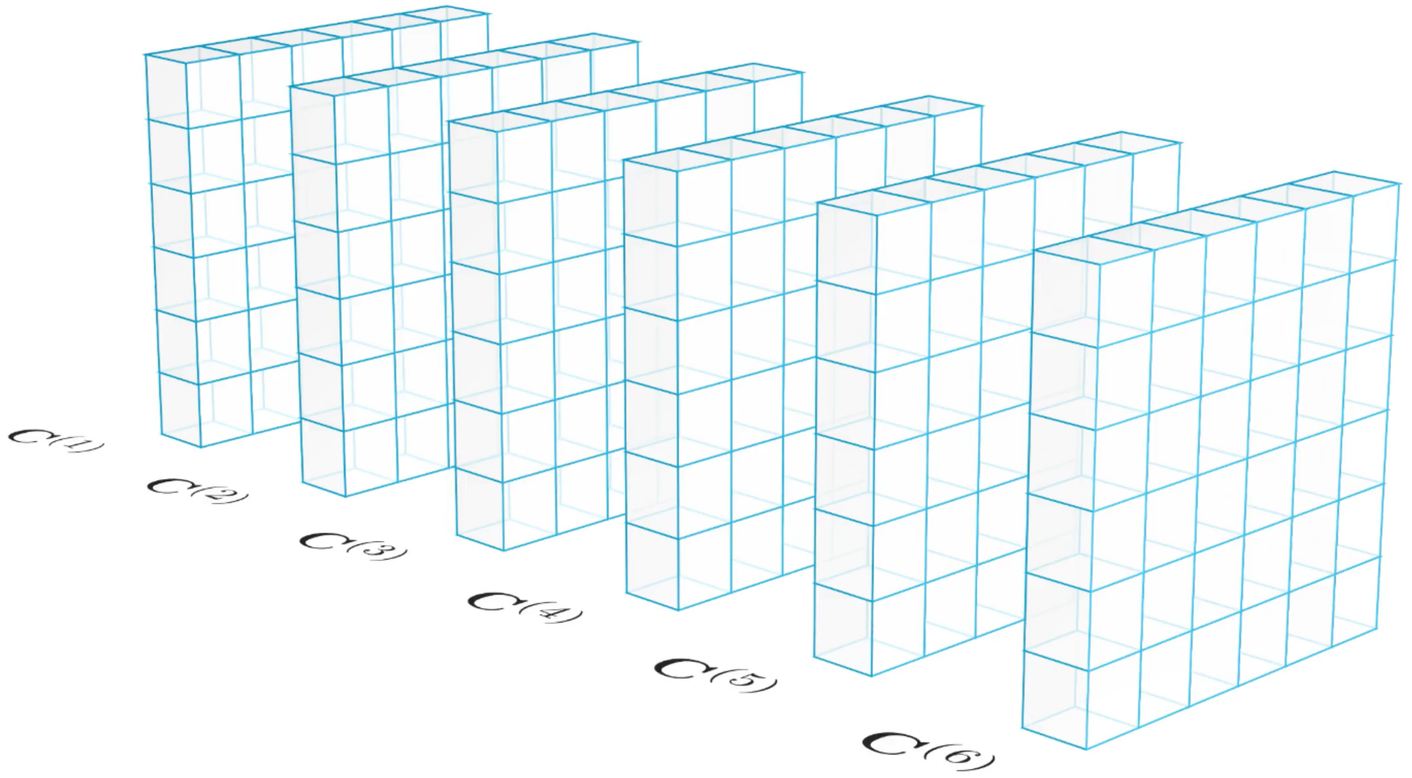
(Alternating) trilinear form $\longrightarrow$ matrix code

Let $\phi^{(i)}(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}, \mathbf{e}_i, \mathbf{z})$



(Alternating) trilinear form ⟶ matrix code

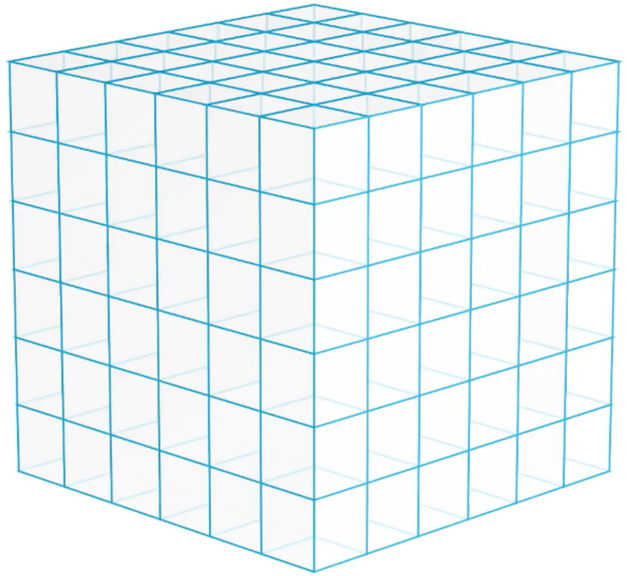
Let $\phi^{(i)}(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x}, \mathbf{e}_i, \mathbf{z})$



Take $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(n)})$ as a basis of an n-dimensional matrix code

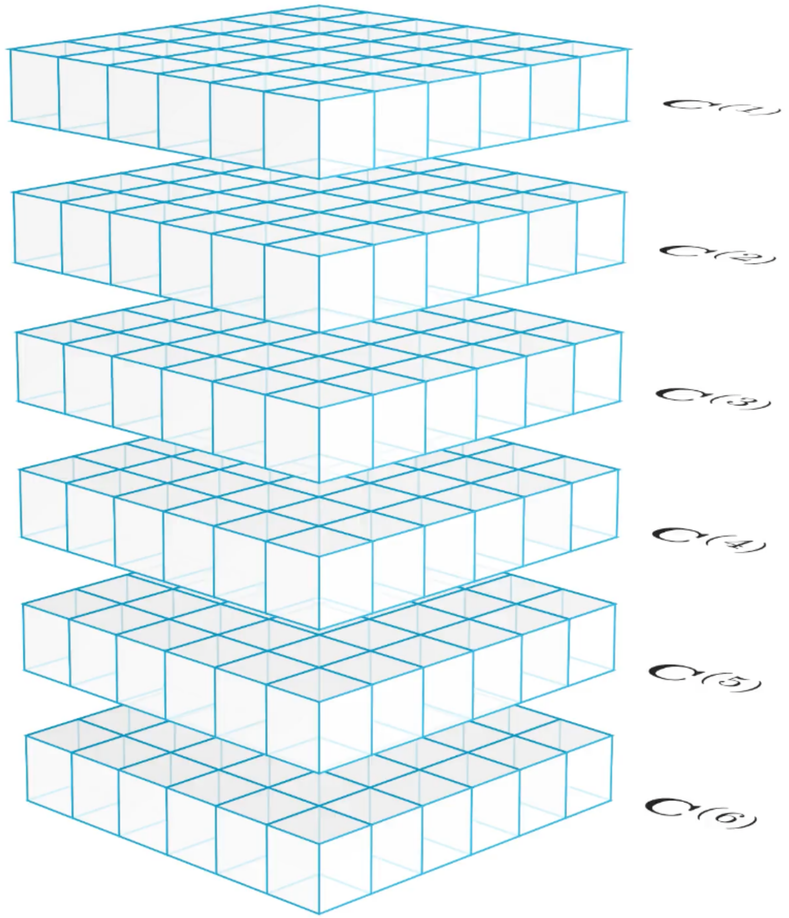
(Alternating) trilinear form $\longrightarrow$ matrix code

Let $\phi^{(i)}(\mathbf{y}, \mathbf{z}) = \phi(\mathbf{e}_i, \mathbf{y}, \mathbf{z})$



(Alternating) trilinear form ⟶ matrix code

Let $\phi^{(i)}(\mathbf{y}, \mathbf{z}) = \phi(\mathbf{e}_i, \mathbf{y}, \mathbf{z})$



Take $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(n)})$ as a basis of an n-dimensional matrix code

ATFE $\longrightarrow$ MCE

- Let (n, ϕ, ψ) be a positive ATFE instance.

ATFE $\longrightarrow$ MCE

Let (n, ϕ, ψ) be a positive ATFE instance.

$$\psi^{(i)}(\mathbf{x}, \mathbf{y}) =$$
$$= \psi(\mathbf{x}, \mathbf{y}, \mathbf{e}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{Ae}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, a_{1i}\mathbf{e}_1 + \ldots + a_{ni}\mathbf{e}_n) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{e}_j) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi^{(j)}(\mathbf{Ax}, \mathbf{Ay})$$

ATFE $\longrightarrow$ MCE

Let (n, ϕ, ψ) be a positive ATFE instance.

$$\psi^{(i)}(\mathbf{x}, \mathbf{y}) =$$
$$= \psi(\mathbf{x}, \mathbf{y}, \mathbf{e}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{Ae}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, a_{1i}\mathbf{e}_1 + \ldots + a_{ni}\mathbf{e}_n) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{e}_j) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi^{(j)}(\mathbf{Ax}, \mathbf{Ay})$$

ATFE $\longrightarrow$ MCE

Let (n, ϕ, ψ) be a positive ATFE instance.

$$\psi^{(i)}(\mathbf{x}, \mathbf{y}) =$$
$$= \psi(\mathbf{x}, \mathbf{y}, \mathbf{e}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{Ae}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, a_{1i}\mathbf{e}_1 + \ldots + a_{ni}\mathbf{e}_n) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{e}_j) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi^{(j)}(\mathbf{Ax}, \mathbf{Ay})$$

Rewrite in matrix form:

$$\mathbf{x}^\top \mathbf{D}^{(i)} \mathbf{y} = \sum_{1 \leqslant j \leqslant n} a_{ji}(\mathbf{Ax})^\top \mathbf{C}^{(j)}(\mathbf{Ay}), \quad \forall i, 1 \leqslant i \leqslant n$$

ATFE $\longrightarrow$ MCE

Let (n, ϕ, ψ) be a positive ATFE instance.

$$\psi^{(i)}(\mathbf{x}, \mathbf{y}) =$$
$$= \psi(\mathbf{x}, \mathbf{y}, \mathbf{e}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{Ae}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, a_{1i}\mathbf{e}_1 + \ldots + a_{ni}\mathbf{e}_n) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{e}_j) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi^{(j)}(\mathbf{Ax}, \mathbf{Ay})$$

Rewrite in matrix form:

$$\mathbf{D}^{(i)} = \sum_{1 \leqslant j \leqslant n} a_{ji}\mathbf{A}^\top\mathbf{C}^{(j)}\mathbf{A}, \quad \forall i, 1 \leqslant i \leqslant n$$

ATFE ⟶ MCE

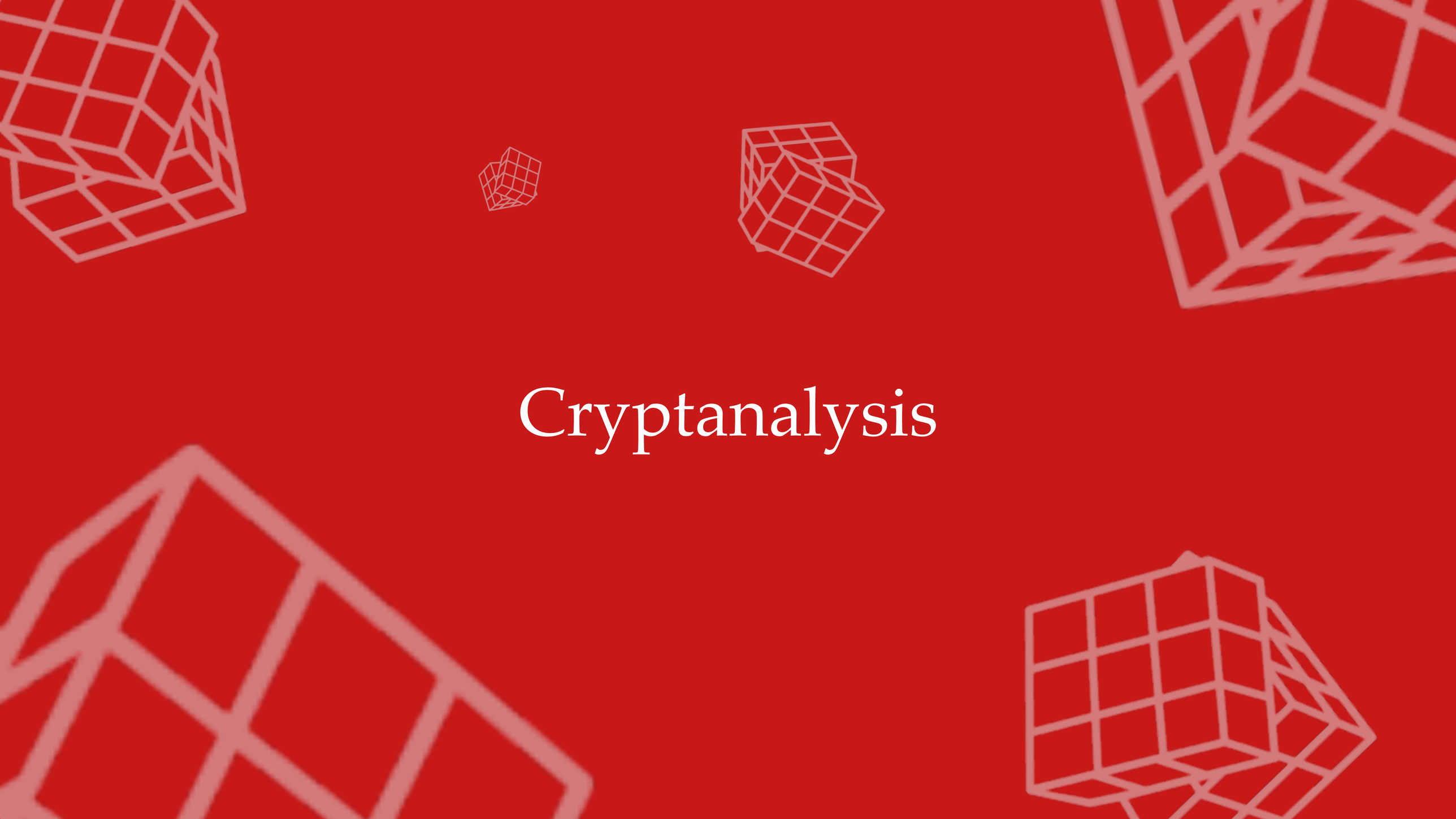
Let (n, ϕ, ψ) be a positive ATFE instance.

$$\psi^{(i)}(\mathbf{x}, \mathbf{y}) =$$
$$= \psi(\mathbf{x}, \mathbf{y}, \mathbf{e}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{Ae}_i) =$$
$$= \phi(\mathbf{Ax}, \mathbf{Ay}, a_{1i}\mathbf{e}_1 + \ldots + a_{ni}\mathbf{e}_n) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi(\mathbf{Ax}, \mathbf{Ay}, \mathbf{e}_j) =$$
$$= \sum_{1 \leqslant j \leqslant n} a_{ji}\phi^{(j)}(\mathbf{Ax}, \mathbf{Ay})$$

Rewrite in matrix form:

$$\mathbf{D}^{(i)} = \sum_{1 \leqslant j \leqslant n} a_{ji}\mathbf{A}^\top\mathbf{C}^{(j)}\mathbf{A}, \quad \forall i, 1 \leqslant i \leqslant n$$

$(\mathbf{A}^\top, \mathbf{A})$ is a solution to the MCE instance $(n, n, n, \mathscr{C}, \mathscr{D})$.


Cryptanalysis

Cryptanalysis

Cryptanalysis

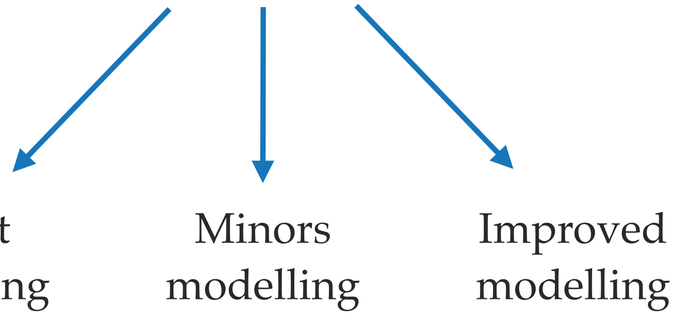
Algebraic attacks

Attacks reducing MCE/ATFE to the problem of solving a system of polynomial equations.

Cryptanalysis

Algebraic attacks

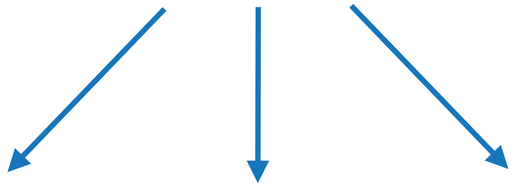
Attacks reducing MCE / ATFE to the problem of solving a system of polynomial equations.



Cryptanalysis

Algebraic attacks

Attacks reducing MCE/ATFE to the problem of solving a system of polynomial equations.



Direct modelling

Minors modelling

Improved modelling

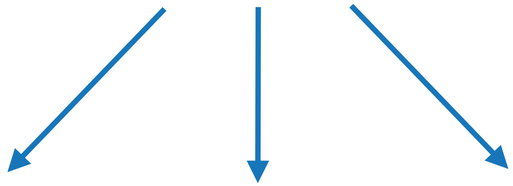
Combinatorial attacks

Collision search attacks using isometry-invariant substructures

Cryptanalysis

Algebraic attacks

Attacks reducing MCE/ATFE to the problem of solving a system of polynomial equations.



Direct modelling

Minors modelling

Improved modelling

Combinatorial attacks

Collision search attacks using isometry-invariant substructures

Graph-based algorithm

Leon-like algorithm

Algebraic attacks

Direct algebraic attack

The MCE problem in matrix form

Let $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ be a basis of code $\mathscr{C}$ and let $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ be a basis of code $\mathscr{D}$. Find $\mathbf{A} \in \mathrm{GL}_m(\mathbb{F}_q)$, $\mathbf{B} \in \mathrm{GL}_n(\mathbb{F}_q)$ and $\mathbf{T} \in \mathrm{GL}_k(\mathbb{F}_q)$ such that

$$\mathbf{D}^{(i)} = \sum_{1 \leq j \leq k} t_{j,i} \mathbf{A}\mathbf{C}^{(j)}\mathbf{B}, \quad \forall 1 \leq i \leq k$$

Direct algebraic attack

The MCE problem in matrix form

Let $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ be a basis of code $\mathscr{C}$ and let $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ be a basis of code $\mathscr{D}$. Find $\mathbf{A} \in \mathrm{GL}_m(\mathbb{F}_q)$, $\mathbf{B} \in \mathrm{GL}_n(\mathbb{F}_q)$ and $\mathbf{T} \in \mathrm{GL}_k(\mathbb{F}_q)$ such that

$$\mathbf{D}^{(i)} = \sum_{1 \leq j \leq k} t_{j,i} \mathbf{A}\mathbf{C}^{(j)}\mathbf{B}, \quad \forall 1 \leq i \leq k$$

Alternatively, this gives a better modelling:

$$\sum_{1 \leq j \leq k} t_{j,i} \mathbf{D}^{(j)} = \mathbf{A}\mathbf{C}^{(i)}\mathbf{B}, \quad \forall 1 \leq i \leq k$$

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \end{pmatrix}$$

For a matrix $\mathbf{C} \in \mathscr{M}_{m,n}(\mathbb{F}_q)$, let Vec be a mapping that sends a matrix $\mathbf{C}$ to the vector $\mathrm{Vec}(\mathbf{C}) \in \mathbb{F}_q^{mn}$ obtained by 'flattening' $\mathbf{C}$:

$$\mathrm{Vec} : \mathbf{C} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & \ddots & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{pmatrix} \mapsto \mathrm{Vec}(\mathbf{C}) = (c_{1,1}, \ldots, c_{1,n}, \ldots, c_{m,1}, \ldots, c_{m,n})$$

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

→ $\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}$ is a codeword in $\mathscr{D}$.

↳ All the minors of $\mathbf{G}$ are zero.

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

→ $\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}$ is a codeword in $\mathscr{D}$.

↳ All the minors of $\mathbf{G}$ are zero.

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

→ $\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}$ is a codeword in $\mathscr{D}$.

↳ All the minors of $\mathbf{G}$ are zero.

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

→ $\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}$ is a codeword in $\mathscr{D}$.

↳ All the minors of $\mathbf{G}$ are zero.

Minors modelling

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

→ $\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}$ is a codeword in $\mathscr{D}$.

↳ All the minors of $\mathbf{G}$ are zero.

Minors modelling: complexity

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

MCE

ATFE

Minors modelling: complexity

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

MCE

Bilinear system of

- $k(nm - k)$ equations
- $n^2 + m^2$ variables

ATFE

Minors modelling: complexity

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

MCE

Bilinear system of

- $k(nm - k)$ equations
- $n^2 + m^2$ variables

Algebraic solver for bilinear systems
(Gröbner basis)

ATFE

Minors modelling: complexity

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

MCE

Bilinear system of

- $k(nm - k)$ equations
- $n^2 + m^2$ variables

Algebraic solver for bilinear systems
(Gröbner basis)

ATFE

Quadratic system of

- $n(\binom{n}{2} - n)$ equations
- n^2 variables

Minors modelling: complexity

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

MCE

Bilinear system of

- $k(nm - k)$ equations
- $n^2 + m^2$ variables

Algebraic solver for bilinear systems
(Gröbner basis)

ATFE

Quadratic system of

- $n(\binom{n}{2} - n)$ equations
- n^2 variables

MQ solver
(Gröbner basis)

Minors modelling: complexity

$$\mathbf{G} = \begin{pmatrix} \mathrm{Vec}(\mathbf{A}^\top \mathbf{C}^{(i)} \mathbf{B}) \\ \mathrm{Vec}(\mathbf{D}^{(1)}) \\ \vdots \\ \mathrm{Vec}(\mathbf{D}^{(n)}) \end{pmatrix}$$

MCE

Bilinear system of

- $k(nm - k)$ equations
- $n^2 + m^2$ variables

Algebraic solver for bilinear systems
(Gröbner basis)

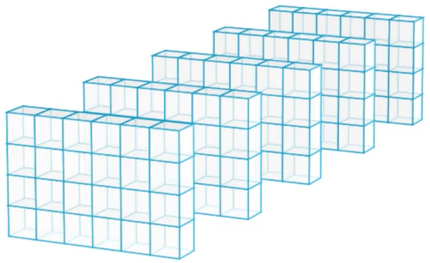
NIST Sec. level	n	q	Tang et al. [TDJ+22]	Beullens [Beu22]	ALTEQ specs. [BDN+23]	**Our work**
—	9	$2^{18}-1$	133	38	—	99
—	10	$2^{17}-1$	133	122	—	105
—	11	$2^{16}-5$	138	85	—	109
I	13	$2^{32}-5$	—	—	143[a]	**120**
III	20	$2^{32}-5$	—	—	219[a]	**165**
V	25	$2^{32}-5$	—	—	252[b]	**203**

MQ solver
(Gröbner basis)

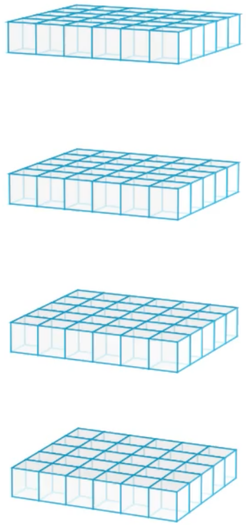
Improved modelling

(Recall) we can see $\mathscr{C}$ from three directions

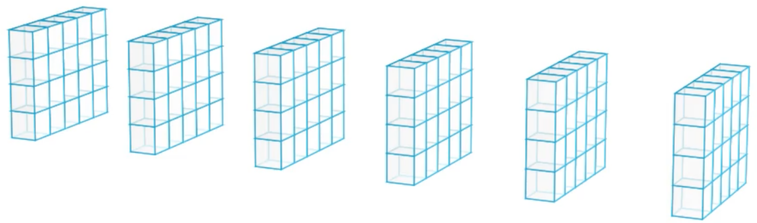
- a k-dimensional code in $\mathbb{F}_q^{m \times n}$



- an m-dimensional code in $\mathbb{F}_q^{n \times k}$



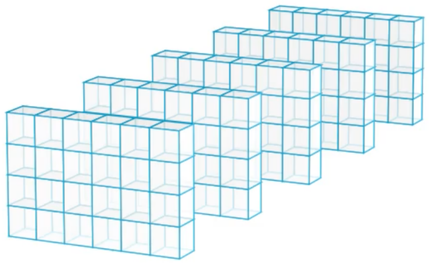
- an n-dimensional code in $\mathbb{F}_q^{m \times k}$

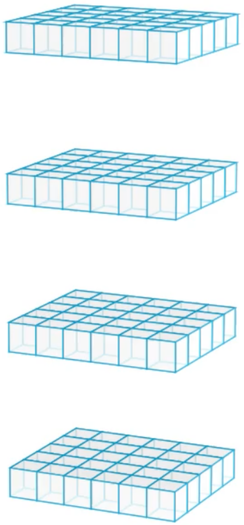


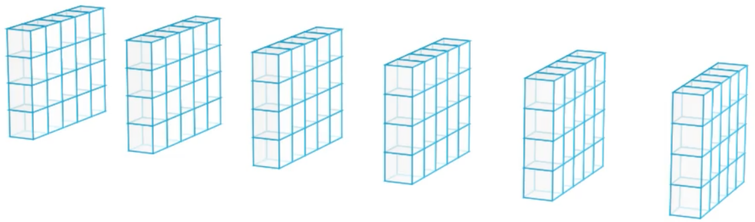
Improved modelling

(Recall) we can see $\mathscr{C}$ from three directions

- a k-dimensional code in $\mathbb{F}_q^{m\times n}$
- an m-dimensional code in $\mathbb{F}_q^{n\times k}$
- an n-dimensional code in $\mathbb{F}_q^{m\times k}$





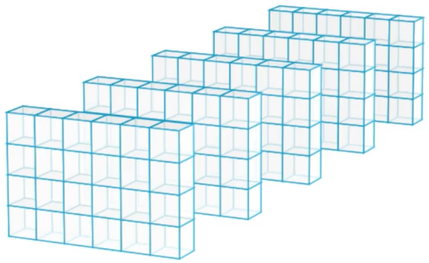


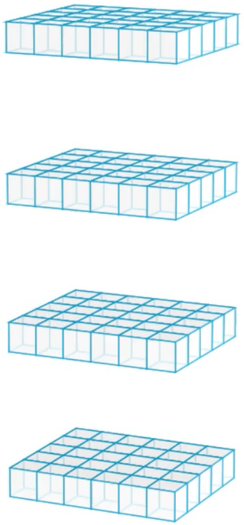
The complexity of the minors modelling is
$\min(\mathrm{GB}(n^2+m^2, k(nm-k), \mathrm{GB}(n^2+k^2, m(nk-m), \mathrm{GB}(k^2+m^2, n(km-n))$

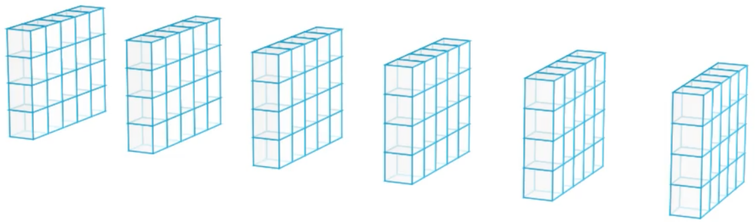
Improved modelling

(Recall) we can see $\mathscr{C}$ from three directions

- a k-dimensional code in $\mathbb{F}_q^{m\times n}$
- an m-dimensional code in $\mathbb{F}_q^{n\times k}$
- an n-dimensional code in $\mathbb{F}_q^{m\times k}$







The complexity of the minors modelling is
$\min(\mathrm{GB}(n^2+m^2, k(nm-k), \mathrm{GB}(n^2+k^2, m(nk-m), \mathrm{GB}(k^2+m^2, n(km-n))$

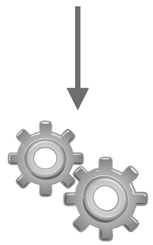
Include all three packets of constraints !

Improved modelling: complexity

MCE

Tri-homogeneous system of

- $k(nm-k)$ equations of tri-degree (1,1,0)
  $m(nk-m)$ equations of tri-degree (1,0,1)
  $n(km-n)$ equations of tri-degree (0,0,1)
- $n^2+m^2+k^2$ variables



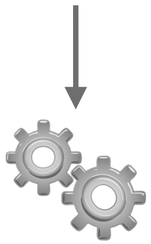
Algebraic solver
(Gröbner basis)

Improved modelling: complexity

MCE

Tri-homogeneous system of

- $k(nm-k)$ equations of tri-degree (1,1,0)
  $m(nk-m)$ equations of tri-degree (1,0,1)
  $n(km-n)$ equations of tri-degree (0,0,1)
- $n^2+m^2+k^2$ variables



Algebraic solver
(Gröbner basis)

ATFE

No added constraints.


Combinatorial attack

Collision

We have a collision when we know a codeword **C** in $\mathscr{C}$ that maps to a codeword **D** in $\mathscr{D}$.

$$\mathbf{D} = \mathbf{ACB}$$

Collision

We have a collision when we know a codeword $\mathbf{C}$ in $\mathscr{C}$ that maps to a codeword $\mathbf{D}$ in $\mathscr{D}$.

$$\mathbf{D} = \mathbf{ACB}$$

We can then infer linear constraints from

$$\mathbf{A}^{-1}\mathbf{D} = \mathbf{CB}$$

Collision

With two collisions, we get the following system

$$\mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B}$$
$$\mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B}$$

Collision

With two collisions, we get the following system

$$\mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B}$$
$$\mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B}$$

→ When $n = m = k$, results in a linear system with the same number of variables and equations.

Collision

With two collisions, we get the following system

$$\mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B}$$
$$\mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B}$$

- When $n = m = k$, results in a linear system with the same number of variables and equations.
  - If $\mathbf{C}_1, \mathbf{C}_2, \mathbf{D}_1, \mathbf{D}_2$ are all full rank, we should have a unique solution.
  - We can easily recover $\mathbf{A}$ from $\mathbf{A}^{-1}$.

Collision

We have a collision when we know a codeword $\mathbf{C}$ in $\mathscr{C}$ that maps to a codeword $\mathbf{D}$ in $\mathscr{D}$.

$$\mathbf{D} = \mathbf{ACB}$$

We can then infer linear constraints from

$$\mathbf{A}^{-1}\mathbf{D} = \mathbf{CB}$$

Collision

We have a collision when we know a codeword $\mathbf{C}$ in $\mathscr{C}$ that maps to a codeword $\mathbf{D}$ in $\mathscr{D}$.

$$\mathbf{D} = \mathbf{ACB}$$

We can then infer linear constraints from

$$\mathbf{A}^{-1}\mathbf{D} = \mathbf{CB}$$

If we add these linear constraints to the system obtained from the algebraic attack, we can (sometimes) efficiently solve the system of equations and recover the isometry.

The birthday paradox

The birthday problem in collision search algorithms

We draw, randomly, elements from a set of size N.
How many times do we expect to draw an element before we get the same element twice?

The birthday paradox

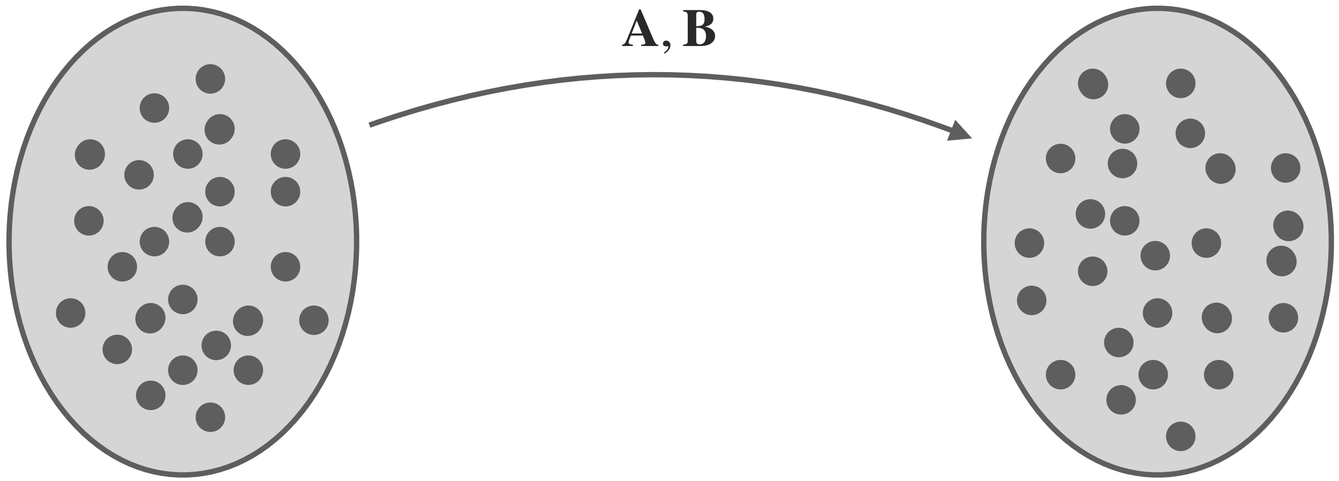
The birthday problem in collision search algorithms

We draw, randomly, elements from a set of size N.
How many times do we expect to draw an element before we get the same element twice?

$\approx c\sqrt{N}$, for c a small constant.

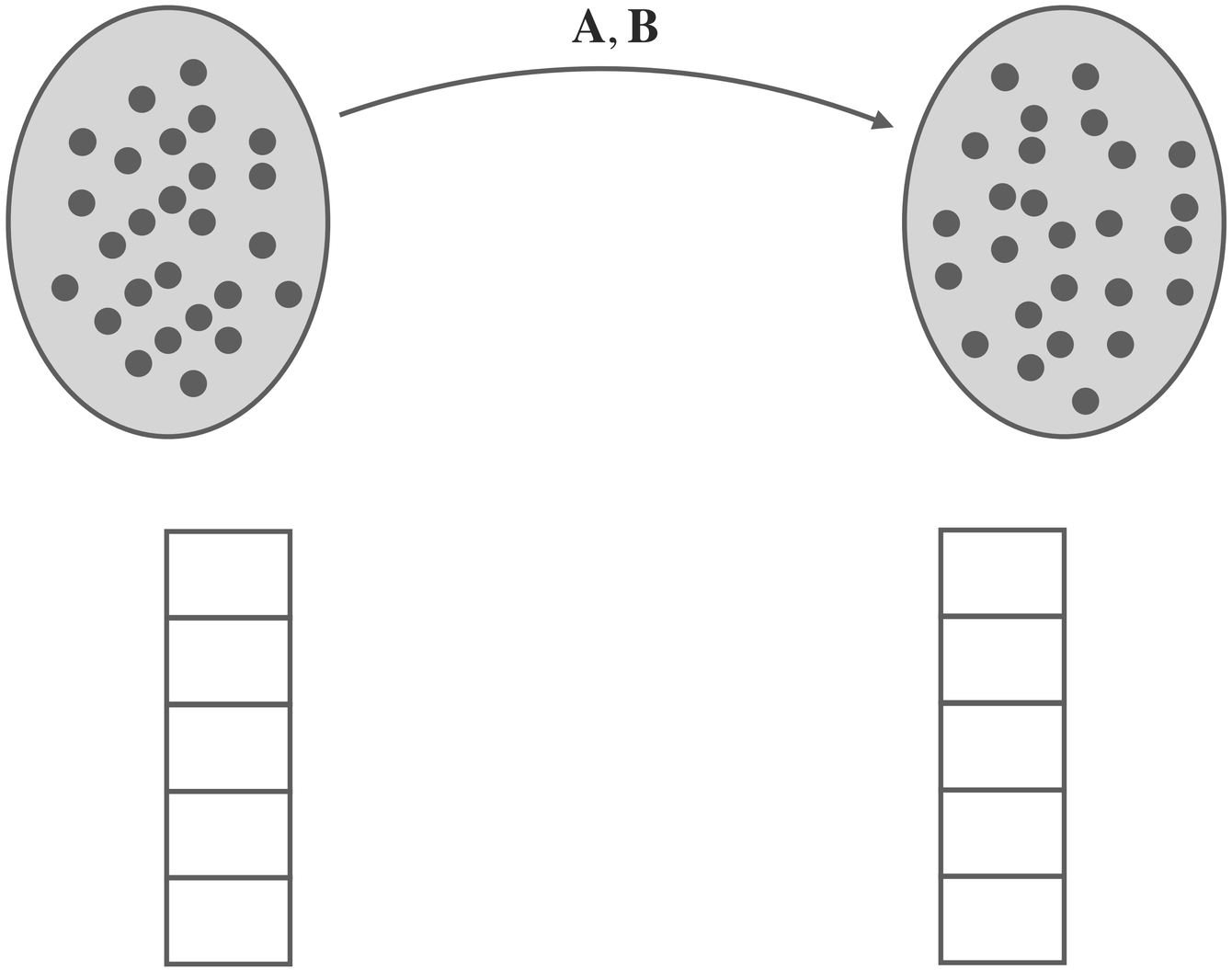
General collision attack

N - number of codewords



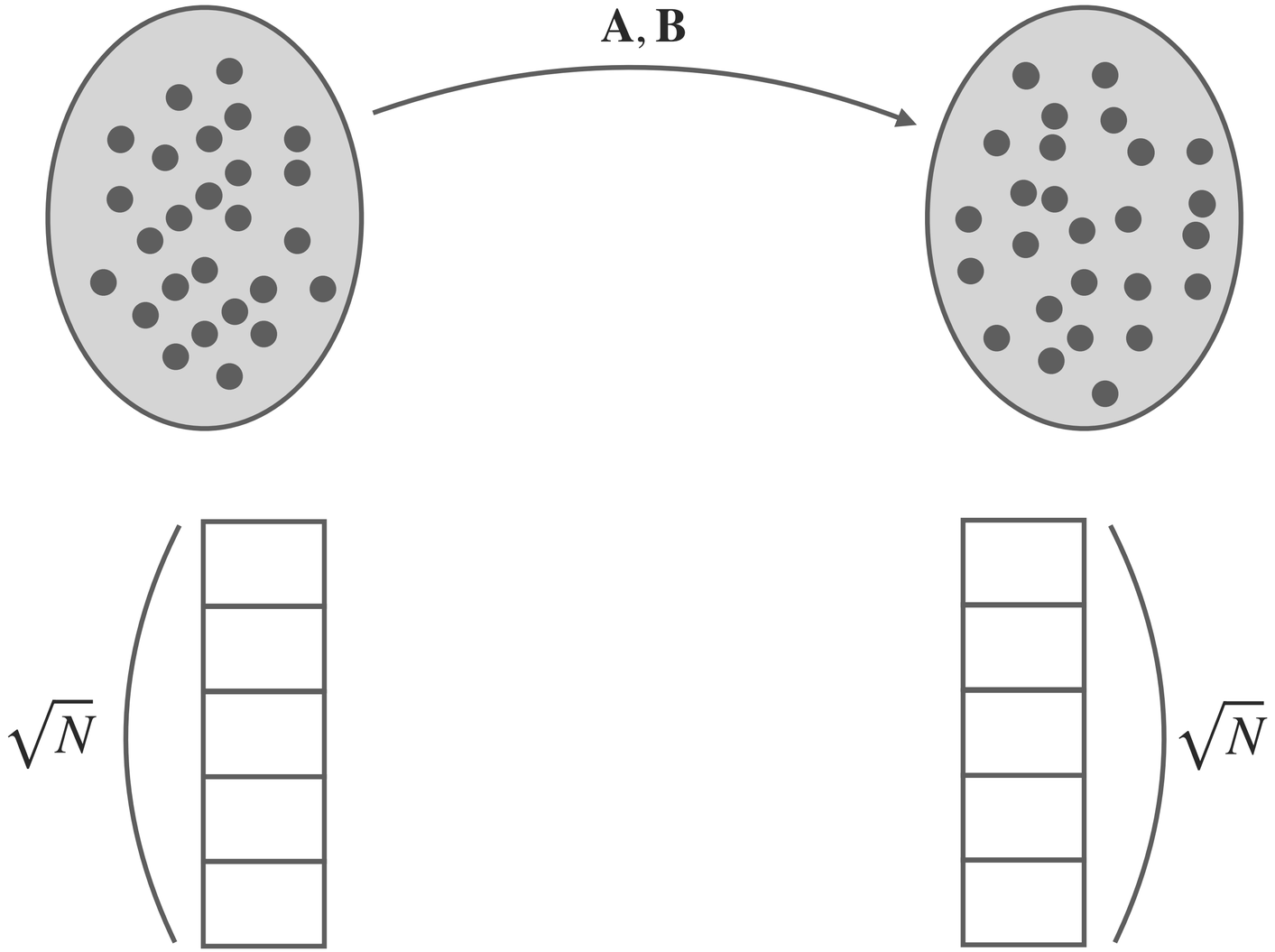
General collision attack

N - number of codewords



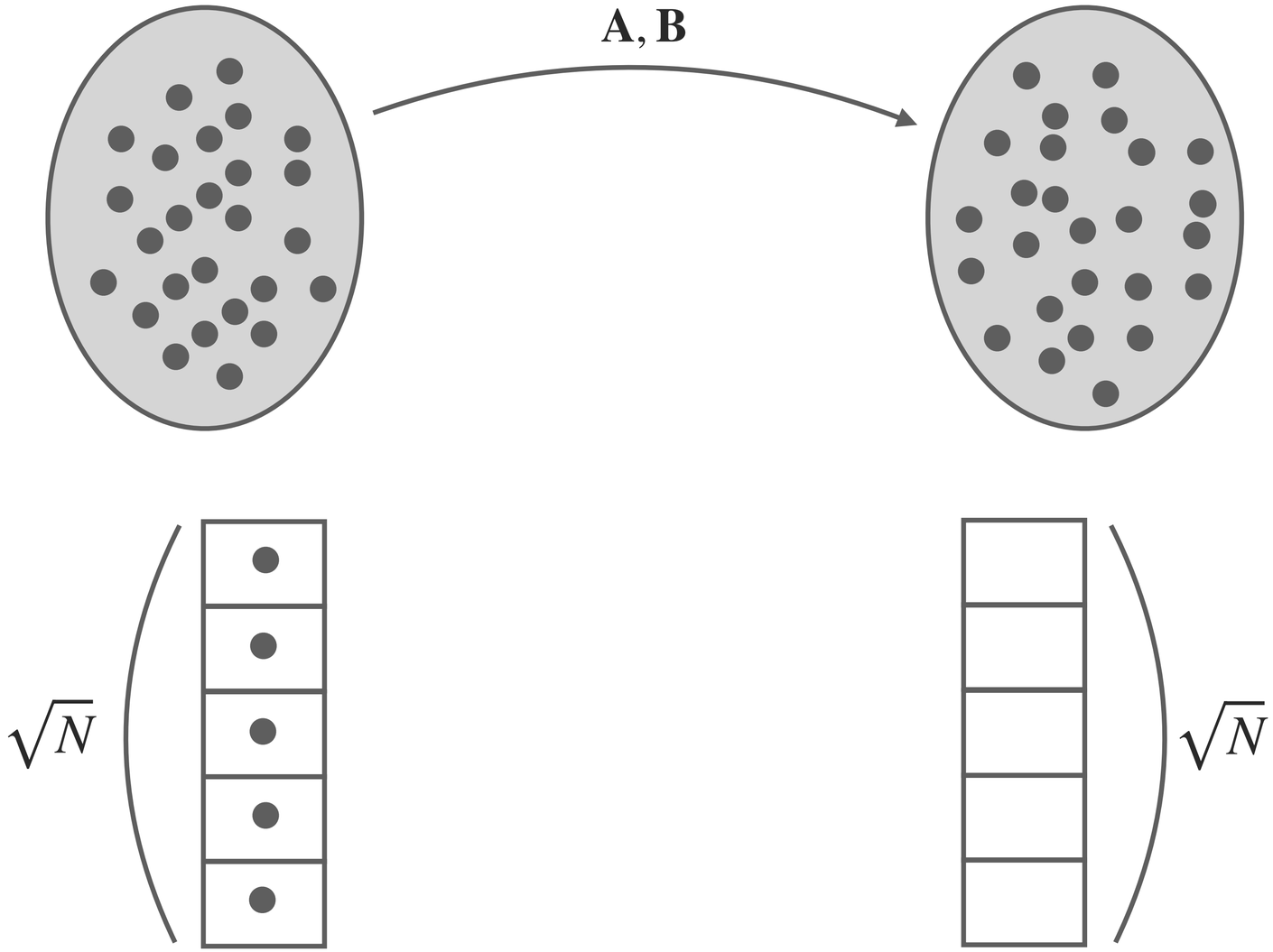
General collision attack

N - number of codewords



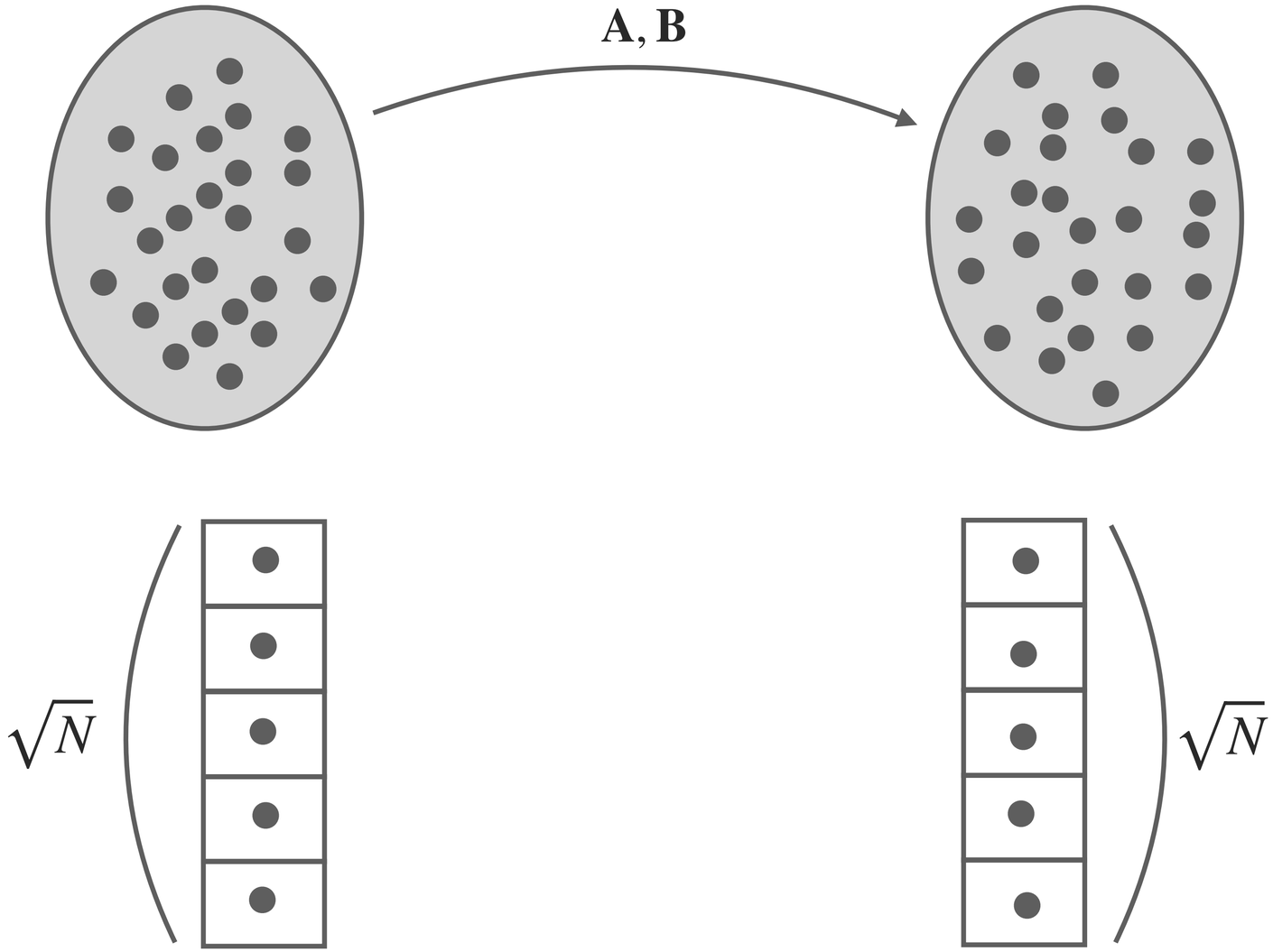
General collision attack

N - number of codewords



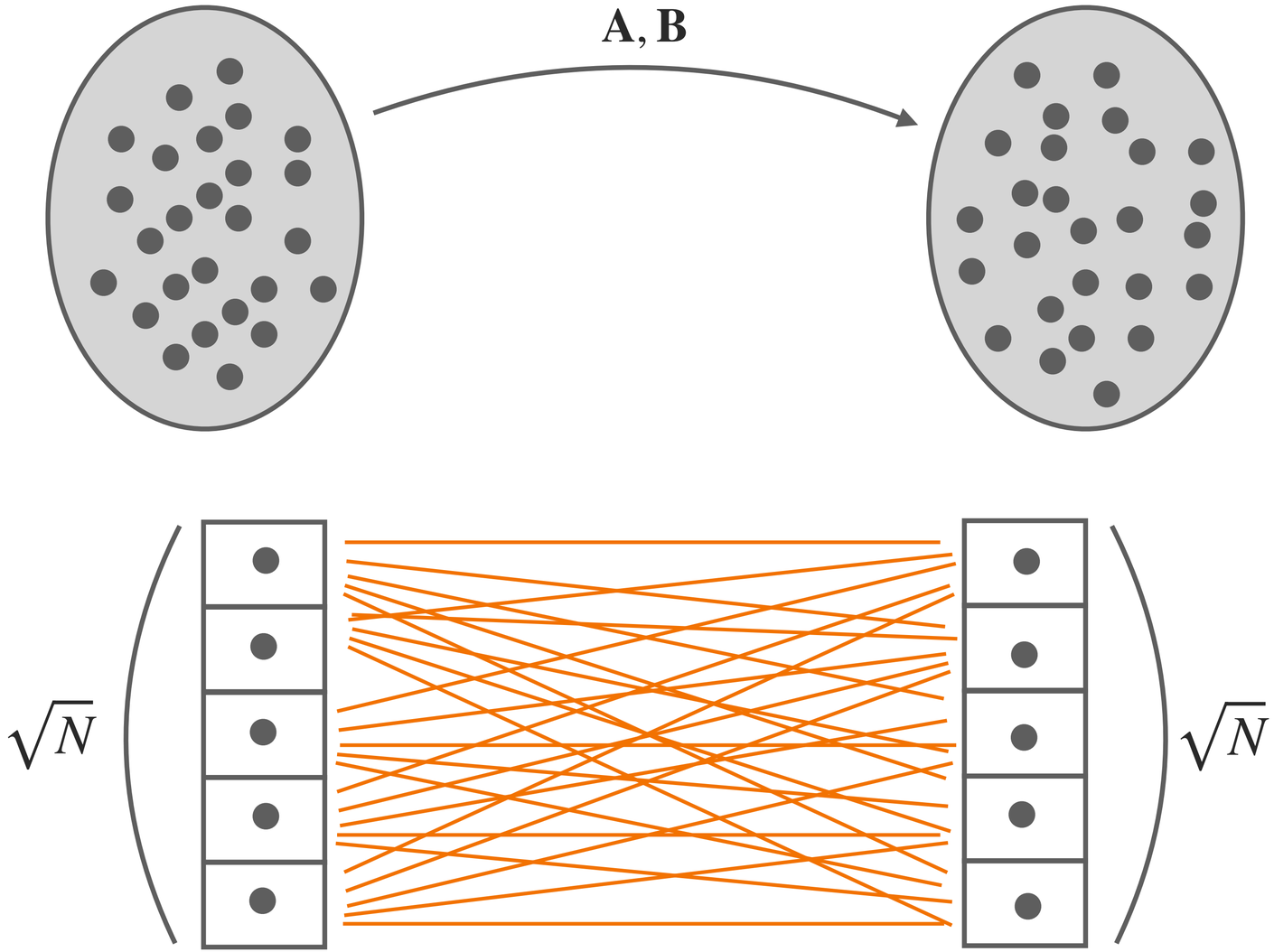
General collision attack

N - number of codewords



General collision attack

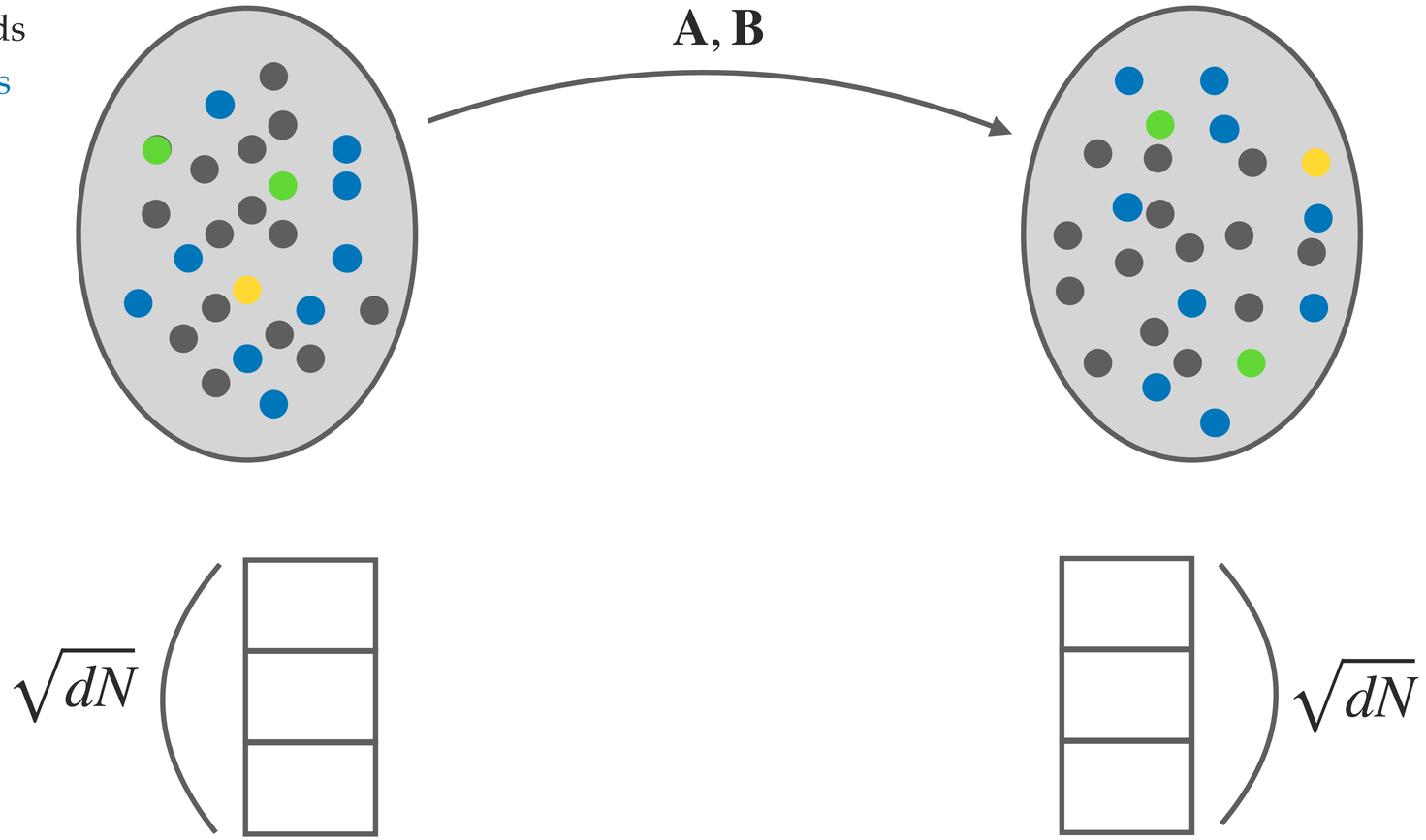
N - number of codewords



General collision attack

N - number of codewords

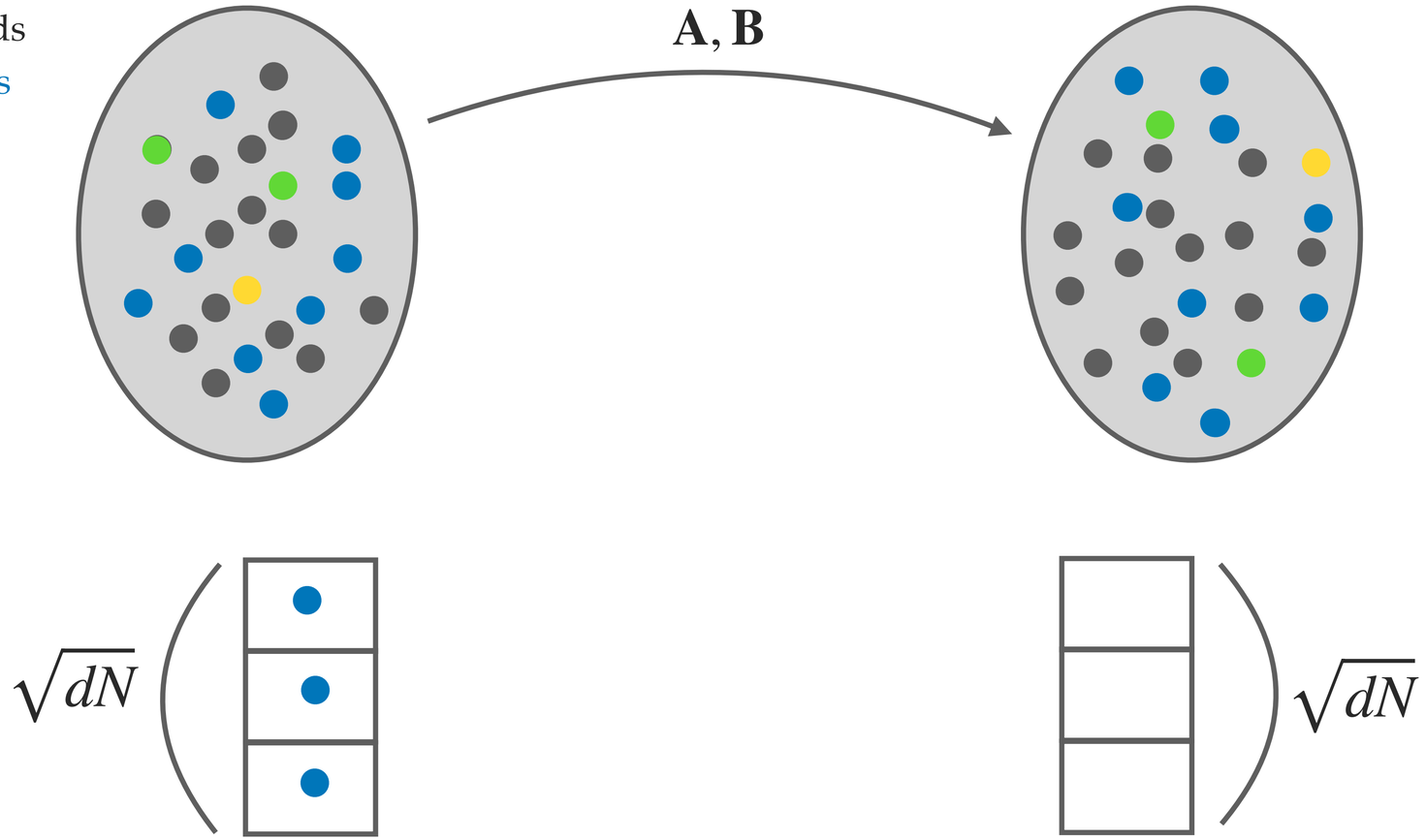
d - density of codewords of rank r



General collision attack

N - number of codewords

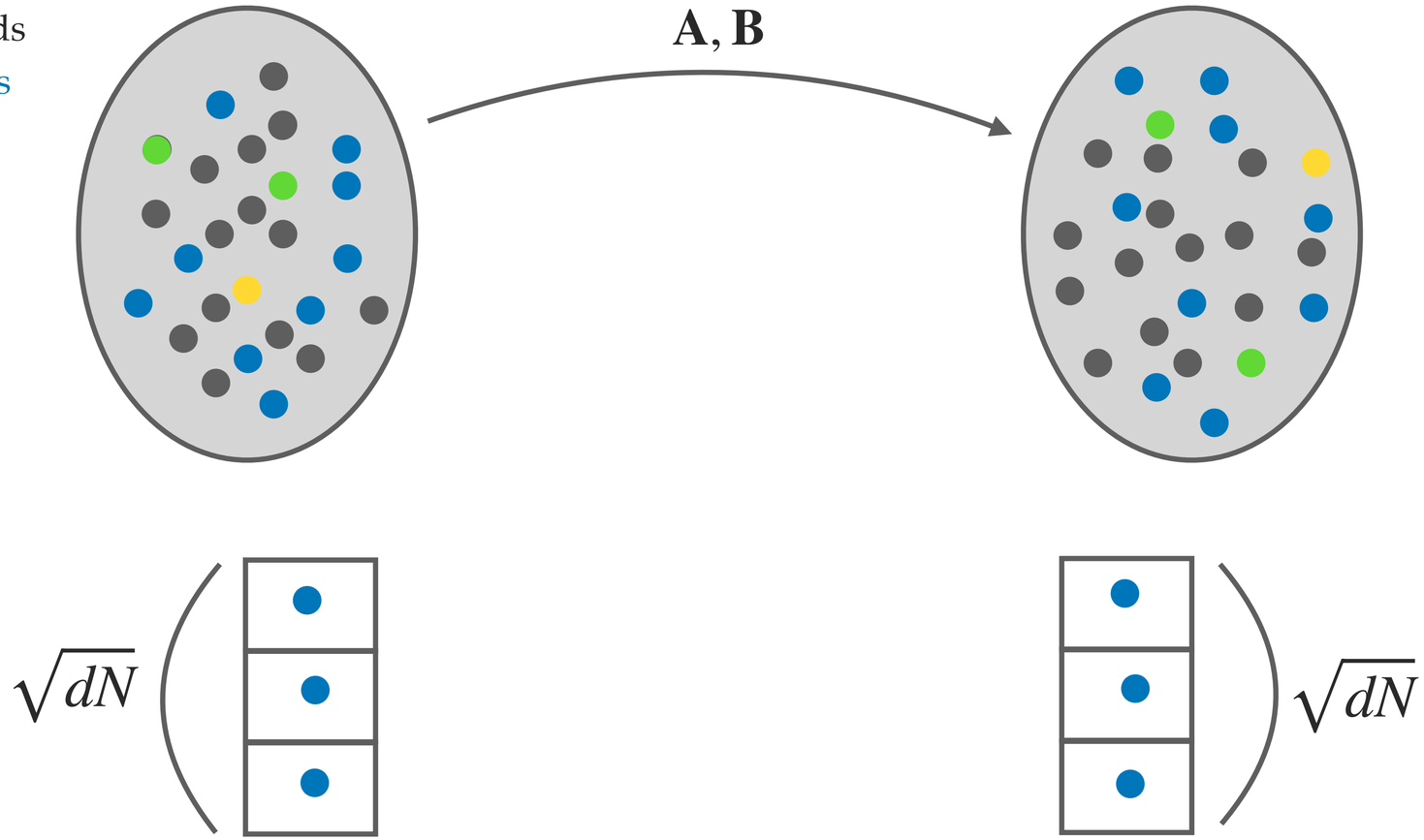
d - density of codewords of rank r



General collision attack

N - number of codewords

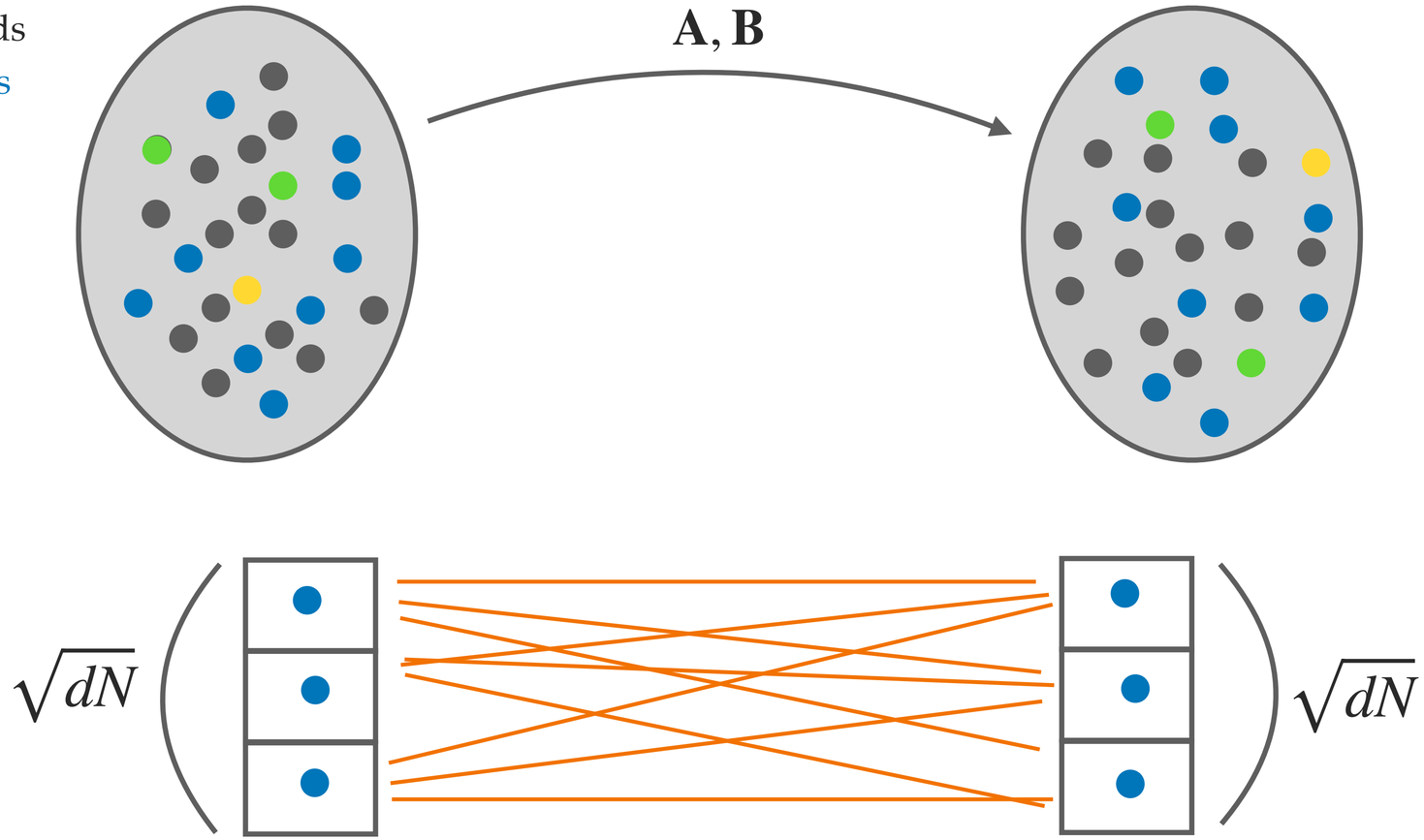
d - density of codewords of rank r



General collision attack

N - number of codewords

d - density of codewords of rank r



check if it's a collision

General collision attack

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then L ← L ∪ {a}
6:         end if
7:     until ⌊L⌋ = ℓ
8:     return L
9: end function

10: function CollisionFind(S_1, S_2)
11:     L_1 ← SampleSet(S_1, ℙ, ℓ)
12:     L_2 ← SampleSet(S_2, ℙ, ℓ)
13:     for all (a, b) ∈ L_1 × L_2 do
14:         φ ← FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

General collision attack

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if Rank(L ∪ {a}) > |L| then
6:             L ← L ∪ {a}
7:         end if
8:     until |L| = ℓ
9:     return L
10: end function
```

MinRank

```
10: function CollisionFind(S1, S2)
11:     L1 ← SampleSet(S1, ℙ, ℓ)
12:     L2 ← SampleSet(S2, ℙ, ℓ)
13:     for all (a, b) ∈ L1 × L2 do
14:         φ ← FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

General collision attack

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if Rank(L ∪ {a}) > Rank(L) then
6:             L ← L ∪ {a}
7:         end if
8:     until |L| = ℓ
9:     return L
10: end function
11: function CollisionFind(S1, S2)
12:     L1 ← SampleSet(S1, ℙ, ℓ)
13:     L2 ← SampleSet(S2, ℙ, ℓ)
14:     for all (a, b) ∈ L1 × L2 do
15:         φ ← FindFunction(a, b)
16:         if φ ≠ ⊥ then
17:             return solution φ
18:         end if
19:     end for
20:     return ⊥
21: end function
```

MinRank

$$\mathbf{AC}_1 = \mathbf{D}_1\mathbf{B}^{-1}$$
$$\mathbf{AC}_2 = \mathbf{D}_2\mathbf{B}^{-1}$$
$$\mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B}$$
$$\mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B}$$

General collision attack

Leon-like algorithm

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if Rank(L ∪ {a}) = |L| + 1 then
6:             L ← L ∪ {a}
7:         end if
8:     until |L| = ℓ
9:     return L
10: end function
11: function CollisionFind(S1, S2)
12:     L1 ← SampleSet(S1, ℙ, ℓ)
13:     L2 ← SampleSet(S2, ℙ, ℓ)
14:     for all (a, b) ∈ L1 × L2 do
15:         φ ← FindFunction(a, b)
16:         if φ ≠ ⊥ then
17:             return solution φ
18:         end if
19:     end for
20:     return ⊥
21: end function
```

MinRank

$$\mathbf{AC}_1 = \mathbf{D}_1\mathbf{B}^{-1}$$
$$\mathbf{AC}_2 = \mathbf{D}_2\mathbf{B}^{-1}$$
$$\mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B}$$
$$\mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B}$$

Collision attack : complexity

Algorithm 1 General Birthday-based Equivalence Finder

```
1:  function SampleSet(S, ℙ, ℓ)
2:      L ← ∅
3:      repeat
4:          a ←$ S
5:          if ℙ(a) then L ← L ∪ {a}
6:          end if
7:      until ⌊L⌋ = ℓ
8:      return L
9:  end function
10: function CollisionFind(S_1, S_2)
11:     L_1 ← SampleSet(S_1, ℙ, ℓ)
12:     L_2 ← SampleSet(S_2, ℙ, ℓ)
13:     for all (a, b) ∈ L_1 × L_2 do
14:         φ ←FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

Collision attack : complexity

N - total number of elements in S_* (number of codewords)

d - proportion of elements in S_* that satisfy $\mathbb{P}$ (density of codewords of rank r)

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then L ← L ∪ {a}
6:         end if
7:     until |L| = ℓ
8:     return L
9: end function
```

$$\frac{1}{d} \cdot \sqrt{dN} \cdot \mathrm{C}_{\mathbb{P}}$$

```
10: function CollisionFind(S_1, S_2)
11:     L_1 ← SampleSet(S_1, ℙ, ℓ)
12:     L_2 ← SampleSet(S_2, ℙ, ℓ)
13:     for all (a, b) ∈ L_1 × L_2 do
14:         φ ← FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

Collision attack : complexity

N - total number of elements in S_* (number of codewords)

d - proportion of elements in S_* that satisfy $\mathbb{P}$ (density of codewords of rank r)

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then L ← L ∪ {a}
6:         end if
7:     until |L| = ℓ
8:     return L
9: end function
```

$$\frac{1}{d} \cdot \sqrt{dN} \cdot \mathrm{C}_{\mathbb{P}}$$

Expected time to find one element that satisfies $\mathbb{P}$

```
10: function CollisionFind(S_1, S_2)
11:     L_1 ← SampleSet(S_1, ℙ, ℓ)
12:     L_2 ← SampleSet(S_2, ℙ, ℓ)
13:     for all (a, b) ∈ L_1 × L_2 do
14:         φ ← FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

Collision attack : complexity

N - total number of elements in S_* (number of codewords)

d - proportion of elements in S_* that satisfy $\mathbb{P}$ (density of codewords of rank r)

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then L ← L ∪ {a}
6:         end if
7:     until |L| = ℓ
8:     return L
9: end function
```

$$\frac{1}{d} \cdot \sqrt{dN} \cdot \mathrm{C}_{\mathbb{P}}$$

```
10: function CollisionFind(S_1, S_2)
11:     L_1 ← SampleSet(S_1, ℙ, ℓ)
12:     L_2 ← SampleSet(S_2, ℙ, ℓ)
13:     for all (a, b) ∈ L_1 × L_2 do
14:         φ ← FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

Expected time to find one element that satisfies $\mathbb{P}$

Number of elements we want to find

Collision attack : complexity

N - total number of elements in S_* (number of codewords)

d - proportion of elements in S_* that satisfy $\mathbb{P}$ (density of codewords of rank r)

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then L ← L ∪ {a}
6:     end if
7:     until |L| = ℓ
8:     return L
9: end function
```

```
10: function CollisionFind(S1, S2)
11:     L1 ← SampleSet(S1, ℙ, ℓ)
12:     L2 ← SampleSet(S2, ℙ, ℓ)
13:     for all (a, b) ∈ L1 × L2 do
14:         φ ← FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

$$\frac{1}{d} \cdot \sqrt{dN} \cdot \mathrm{C}_{\mathbb{P}}$$

Expected time to find one element that satisfies $\mathbb{P}$

Number of elements we want to find

Cost to check whether an element satisfies $\mathbb{P}$

Collision attack : complexity

N - total number of elements in S_* (number of codewords)

d - proportion of elements in S_* that satisfy $\mathbb{P}$ (density of codewords of rank r)

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then L ← L ∪ {a}
6:         end if
7:     until |L| = ℓ
8:     return L
9: end function
```

$$\frac{1}{d} \cdot \sqrt{dN} \cdot \mathrm{C}_{\mathbb{P}}$$

```
10: function CollisionFind(S_1, S_2)
11:     L_1 ← SampleSet(S_1, ℙ, ℓ)
12:     L_2 ← SampleSet(S_2, ℙ, ℓ)
13:     for all (a, b) ∈ L_1 × L_2 do
14:         φ ← FindFunction(a, b)
15:         if φ ∉ L then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

$$dN \cdot \mathrm{C}_{\mathrm{FF}}$$

Expected time to find one element that satisfies $\mathbb{P}$

Number of elements we want to find

Cost to check whether an element satisfies $\mathbb{P}$

Collision attack : complexity

N - total number of elements in S_* (number of codewords)

d - proportion of elements in S_* that satisfy $\mathbb{P}$ (density of codewords of rank r)

Algorithm 1 General Birthday-based Equivalence Finder

1: **function** SAMPLESET($S, \mathbb{P}, \ell$)
2: $L \leftarrow \emptyset$
3: **repeat**
4: $a \xleftarrow{\$} S$
5: **if** $\mathbb{P}(a)$ **then** $L \leftarrow L \cup \{a\}$
6: **end if**
7: **until** $\mathsf{L} = \ell$
8: **return** L
9: **end function**

10: **function** COLLISIONFIND(S_1, S_2)
11: $L_1 \leftarrow$ SAMPLESET($S_1, \mathbb{P}, \ell$)
12: $L_2 \leftarrow$ SAMPLESET($S_2, \mathbb{P}, \ell$)
13: **for all** $(a, b) \in L_1 \times L_2$ **do**
14: $\phi \leftarrow$FINDFUNCTION(a, b)
15: **if** $\phi \neq \perp$ **then**
16: **return** solution ϕ
17: **end if**
18: **end for**
19: **return** $\perp$
20: **end function**

$$\frac{1}{d} \cdot \sqrt{dN} \cdot \mathrm{C}_{\mathbb{P}}$$

$$dN \cdot \mathrm{C}_{\mathrm{FF}}$$

Expected time to find one element that satisfies $\mathbb{P}$

Number of elements we want to find

Cost to check whether an element satisfies $\mathbb{P}$

Cartesian product between two lists of size $\sqrt{dN}$

Collision attack : complexity

N - total number of elements in S_* (number of codewords)

d - proportion of elements in S_* that satisfy $\mathbb{P}$ (density of codewords of rank r)

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet(S, ℙ, ℓ)
2:     L ← ∅
3:     repeat
4:         a ←$ S
5:         if ℙ(a) then L ← L ∪ {a}
6:         end if
7:     until |L| = ℓ
8:     return L
9: end function
```

```
10: function CollisionFind(S1, S2)
11:     L1 ← SampleSet(S1, ℙ, ℓ)
12:     L2 ← SampleSet(S2, ℙ, ℓ)
13:     for all (a, b) ∈ L1 × L2 do
14:         φ ← FindFunction(a, b)
15:         if φ ≠ ⊥ then
16:             return solution φ
17:         end if
18:     end for
19:     return ⊥
20: end function
```

$$\frac{1}{d} \cdot \sqrt{dN} \cdot \mathrm{C}_{\mathbb{P}}$$

$$dN \cdot \mathrm{C}_{\mathrm{FF}}$$

Expected time to find one element that satisfies $\mathbb{P}$

Number of elements we want to find

Cost to check whether an element satisfies $\mathbb{P}$

Cartesian product between two lists of size $\sqrt{dN}$

Cost to check whether two elements form a collision

Collision attack : complexity

Depends on the choice of the predicate $\mathbb{P}$. The choice is made such that we obtain the optimal balance between the two parts of the algorithm, aka. they take approximately the same time (whenever possible).

$$\frac{1}{d}\sqrt{dN}\,\mathrm{C}_{\mathbb{P}} \approx dN\,\mathrm{C}_{\mathrm{FF}}$$

Collision attack : complexity

Depends on the choice of the predicate $\mathbb{P}$. The choice is made such that we obtain the optimal balance between the two parts of the algorithm, aka. they take approximately the same time (whenever possible).

$$\frac{1}{d}\sqrt{dN}\,\mathrm{C}_{\mathbb{P}} \approx dN\,\mathrm{C}_{\mathrm{FF}}$$

Best trade-off: when $d \approx N^{-\frac{1}{3}}$ (assuming $\mathrm{C}_{\mathbb{P}}$ and C_{FF} are poly-time and comparable).

Collision attack : complexity

Depends on the choice of the predicate $\mathbb{P}$. The choice is made such that we obtain the optimal balance between the two parts of the algorithm, aka. they take approximately the same time (whenever possible).

$$\frac{1}{d}\sqrt{dN}\,\mathrm{C}_{\mathbb{P}} \approx dN\,\mathrm{C}_{\mathrm{FF}}$$

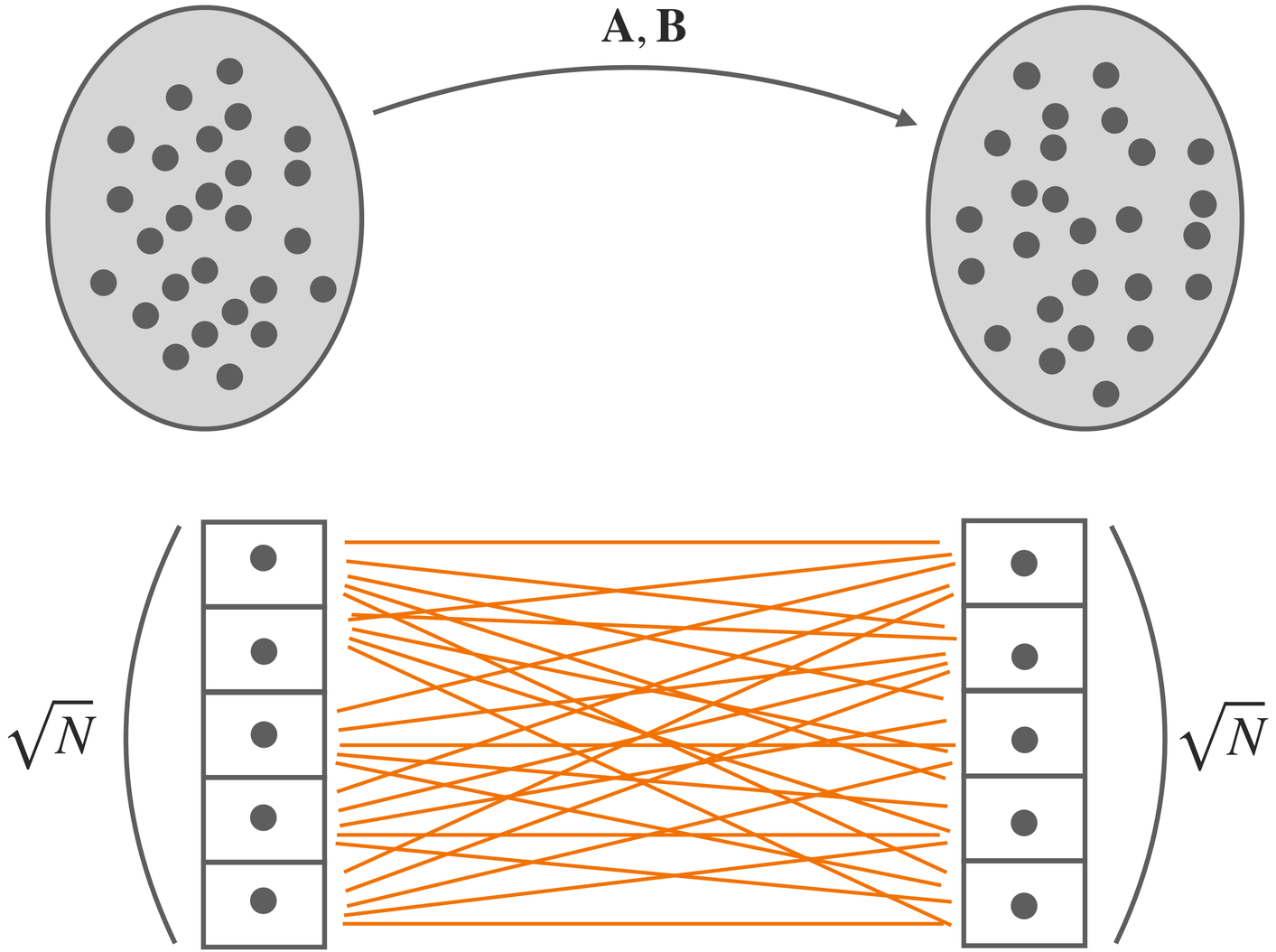
Best trade-off: when $d \approx N^{-\frac{1}{3}}$ (assuming $\mathrm{C}_{\mathbb{P}}$ and C_{FF} are poly-time and comparable).

Time complexity $\mathcal{O}(N^{\frac{2}{3}})$

Memory complexity $\mathcal{O}(N^{\frac{1}{3}})$

Why not $\sqrt{}$?

N - number of codewords



Distinguishing isomorphism invariants

A distinguishing invariant for QMLE (a variant of the isomorphism of polynomials problem) over $\mathbb{F}_2$.

[BFV] Bouillaguet, Fouque, Véber. Graph-Theoretic Algorithms for the Isomorphism of Polynomials Problem. (2012)

A distinguishing invariant for ATFE with parameter $n = 9$.

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

A distinguishing invariant for MCE and ATFE.

[NQT] Narayanan, Qiao, Tang. Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants. (2024)

A distinguishing invariant for MCE and ATFE.

[RS] Ran, Samardjiska. Rare structures in tensor graphs - Bermuda triangles for cryptosystems based on the Tensor Isomorphism problem. (2024)

Motivation & recap

Signatures from equivalence problems

Equivalence-based digital signature schemes in the NIST competition (and elsewhere):

LESS: Linear code equivalence

MEDS Matrix code equivalence

ALTEQ Alternating trilinear form equivalence

Patarin's signature scheme: Isomorphism of polynomials

SeaSign, SQISign: Isogeny between elliptic curves

HAWK: Lattice isomorphism

…

Signatures from equivalence problems

Equivalence-based digital signature schemes in the NIST competition (and elsewhere):

LESS: Linear

MEDS M

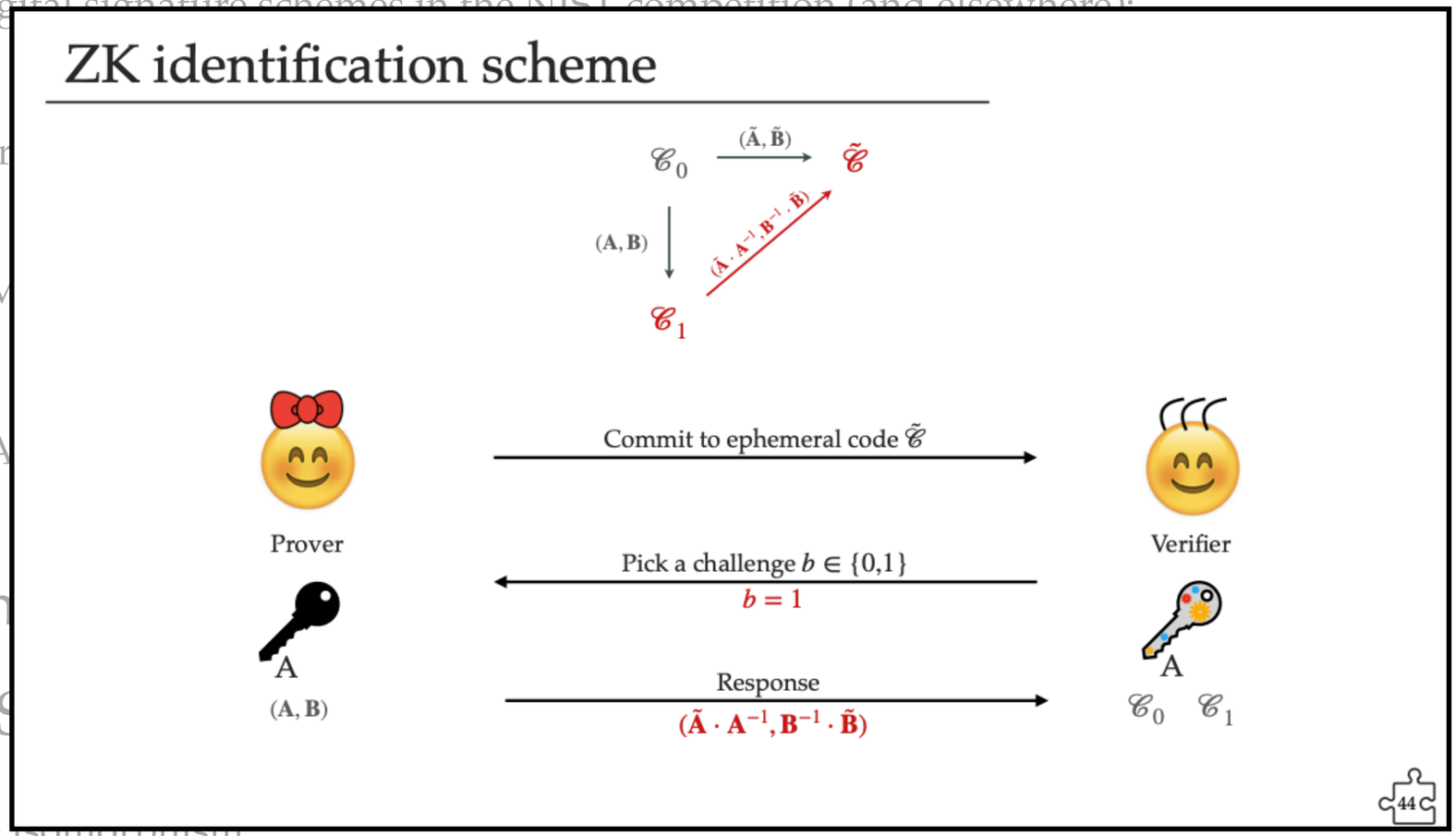
ALTEQ A

Patarin's sign

SeaSign, SQIS

HAWK: Lattice isomorphism

…



Signatures from equivalence problems

Equivalence-based digital signature schemes in the NIST competition (and elsewhere):

LESS: Linear code equivalence

MEDS Matrix code equivalence

ALTEQ Alternating trilinear form equivalence

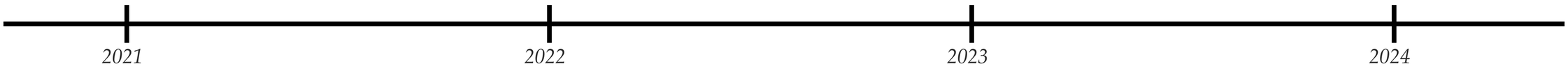
Patarin's signature scheme: Isomorphism of polynomials

SeaSign, SQISign: Isogeny between elliptic curves

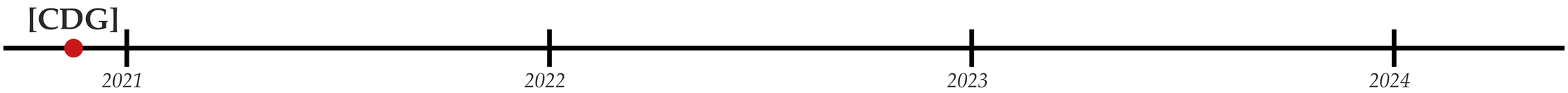
HAWK: Lattice isomorphism

…

Cryptanalysis timeline

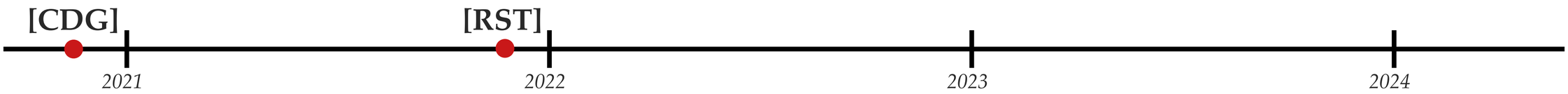


Cryptanalysis timeline



[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in rank metric. (2020)

Cryptanalysis timeline



[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in rank metric. (2020)

[RST] Reijnders, Samardjiska, Trimoska. Hardness estimates of the Code Equivalence Problem in the Rank Metric. (2021)

Cryptanalysis timeline

[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in rank metric. (2020)

[RST] Reijnders, Samardjiska, Trimoska. Hardness estimates of the Code Equivalence Problem in the Rank Metric. (2021)

[CNPRRST] Chou, Niederhagen, Persichetti, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Take your MEDS: Digital Signatures from Matrix Code Equivalence. (2022)

Cryptanalysis timeline

[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in rank metric. (2020)

[RST] Reijnders, Samardjiska, Trimoska. Hardness estimates of the Code Equivalence Problem in the Rank Metric. (2021)

[CNPRRST] Chou, Niederhagen, Persichetti, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Take your MEDS: Digital Signatures from Matrix Code Equivalence. (2022)

[CNPRRRST] Chou, Niederhagen, Persichetti, Ran, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Matrix Equivalence Digital Signature Scheme. (2023)

Cryptanalysis timeline

[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in rank metric. (2020)

[RST] Reijnders, Samardjiska, Trimoska. Hardness estimates of the Code Equivalence Problem in the Rank Metric. (2021)

[CNPRRST] Chou, Niederhagen, Persichetti, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Take your MEDS: Digital Signatures from Matrix Code Equivalence. (2022)

[CNPRRRST] Chou, Niederhagen, Persichetti, Ran, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Matrix Equivalence Digital Signature Scheme. (2023)

[NQT] Narayanan, Qiao, Tang. Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants. (2024)

Cryptanalysis timeline

[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in rank metric. (2020)

[RST] Reijnders, Samardjiska, Trimoska. Hardness estimates of the Code Equivalence Problem in the Rank Metric. (2021)

[CNPRRST] Chou, Niederhagen, Persichetti, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Take your MEDS: Digital Signatures from Matrix Code Equivalence. (2022)

[CNPRRRST] Chou, Niederhagen, Persichetti, Ran, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Matrix Equivalence Digital Signature Scheme. (2023)

[NQT] Narayanan, Qiao, Tang. Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants. (2024)

[RS] Ran, Samardjiska. Rare structures in tensor graphs - Bermuda triangles for cryptosystems based on the Tensor Isomorphism problem. (2024)

Cryptanalysis timeline

Cryptanalysis timeline

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. (2022)

Cryptanalysis timeline

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. (2022)

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

Cryptanalysis timeline

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. (2022)

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

[RST] Ran, Samardjiska, Trimoska. Algebraic Algorithm for the Alternating Trilinear Form Equivalence Problem. (2023)

Cryptanalysis timeline

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. (2022)

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

[RST] Ran, Samardjiska, Trimoska. Algebraic Algorithm for the Alternating Trilinear Form Equivalence Problem. (2023)

[BDNPQST]. Bläser, Duong, Narayanan, Plantard, Qiao, Sipasseuth, Tang. The ALTEQ Signature Scheme. (2023)

Cryptanalysis timeline

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. (2022)

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

[RST] Ran, Samardjiska, Trimoska. Algebraic Algorithm for the Alternating Trilinear Form Equivalence Problem. (2023)

[BDNPQST]. Bläser, Duong, Narayanan, Plantard, Qiao, Sipasseuth, Tang. The ALTEQ Signature Scheme. (2023)

[NQT] Narayanan, Qiao, Tang. Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants. (2024)

Cryptanalysis timeline

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. (2022)

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

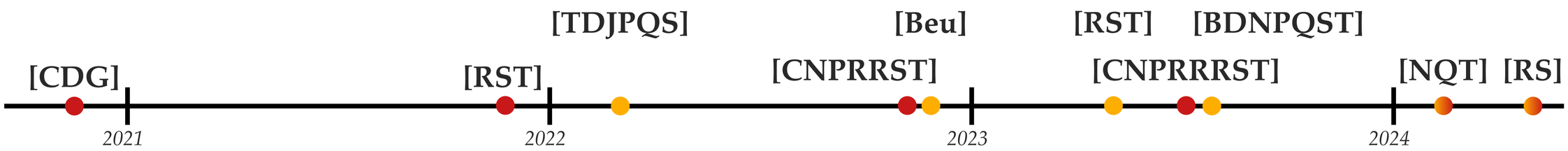
[RST] Ran, Samardjiska, Trimoska. Algebraic Algorithm for the Alternating Trilinear Form Equivalence Problem. (2023)

[BDNPQST]. Bläser, Duong, Narayanan, Plantard, Qiao, Sipasseuth, Tang. The ALTEQ Signature Scheme. (2023)

[NQT] Narayanan, Qiao, Tang. Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants. (2024)

[RS] Ran, Samardjiska. Rare structures in tensor graphs - Bermuda triangles for cryptosystems based on the Tensor Isomorphism problem. (2024)

Cryptanalysis timeline



[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in rank metric. (2020)

[RST] Reijnders, Samardjiska, Trimoska. Hardness estimates of the Code Equivalence Problem in the Rank Metric. (2021)

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. (2022)

[CNPRRST] Chou, Niederhagen, Persichetti, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Take your MEDS: Digital Signatures from Matrix Code Equivalence. (2022)

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

[RST] Ran, Samardjiska, Trimoska. Algebraic Algorithm for the Alternating Trilinear Form Equivalence Problem. (2023)

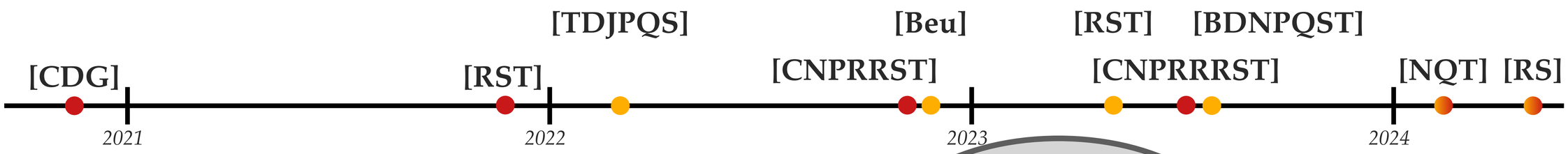
[CNPRRRST] Chou, Niederhagen, Persichetti, Ran, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Matrix Equivalence Digital Signature Scheme. (2023)

[BDNPQST]. Bläser, Duong, Narayanan, Plantard, Qiao, Sipasseuth, Tang. The ALTEQ Signature Scheme. (2023)

[NQT] Narayanan, Qiao, Tang. Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants. (2024)

[RS] Ran, Samardjiska. Rare structures in tensor graphs - Bermuda triangles for cryptosystems based on the Tensor Isomorphism problem. (2024)

Cryptanalysis timeline



Thank you !

[CDG] Couvreur, Debris-Alazard, Gaborit. On the hardness of code equivalence problems in

[RST] Reijnders, Samardjiska, Trimoska. Hardness estimates of the Code Equivalence Problem in

[TDJPQS] Tang, Duong, Joux, Plantard, Qiao, Susilo. Practical post-quantum signature schem om isomorphism problems of trilinear forms. (2022)

[CNPRRST] Chou, Niederhagen, Persichetti, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Take your MEDS: Digital Signatures from Matrix Code Equivalence. (2022)

[Beu] Beullens. Graph-Theoretic Algorithms for the Alternating Trilinear Form Equivalence Problem. (2022)

[RST] Ran, Samardjiska, Trimoska. Algebraic Algorithm for the Alternating Trilinear Form Equivalence Problem. (2023)

[CNPRRRST] Chou, Niederhagen, Persichetti, Ran, Randrianarisoa, Reijnders, Samardjiska, Trimoska. Matrix Equivalence Digital Signature Scheme. (2023)

[BDNPQST]. Bläser, Duong, Narayanan, Plantard, Qiao, Sipasseuth, Tang. The ALTEQ Signature Scheme. (2023)

[NQT] Narayanan, Qiao, Tang. Algorithms for Matrix Code and Alternating Trilinear Form Equivalences via New Isomorphism Invariants. (2024)

[RS] Ran, Samardjiska. Rare structures in tensor graphs - Bermuda triangles for cryptosystems based on the Tensor Isomorphism problem. (2024)