Digital signatures from equivalence problems - A closer look at MEDS and ALTEQ

Simona Samardjiska  Radboud University, Netherlands
Youming Qiao  University of Technology Sydney, Australia
NIST PQC Seminar
Acknowledgement

- The rest of the MEDS team: Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Lars Ran, Krijn Reijnders, Monika Trimoska
- The rest of the ALTEQ team: Markus Bläser, Dung Hoang Duong, Anand Kumar Narayanan, Thomas Plantard, Arnaud Sipasseuth, Gang Tang.
Interesting case - when problem is hard! What can we do with it? Turns out - a lot!

- Zero-Knowledge protocols
- Identification schemes (IDS)
- Digital Signatures via Fiat-Shamir transform

• F-S is a common strategy for PQ signatures

- Dilithium, MQDSS, Picnic in first 3 rounds of NIST competition
- More than 15 in the additional round!

Motivation

**Generic hard equivalence problem** $\text{EQ}(\mathcal{O}_0, \mathcal{O}_1)$:

Given $\mathcal{O}_0$ and $\mathcal{O}_1$, find (if any) an isomorphism $\phi$ s.t. \( \mathcal{O}_1 = \phi(\mathcal{O}_0) \)
Motivation

**Generic hard equivalence problem** $\text{EQ}(O_0, O_1)$:
Given $O_0$ and $O_1$, find (if any) an isomorphism $\phi$ s.t. $O_1 = \phi(O_0)$

Interesting case - **when problem is hard!** What can we do with it?
Motivation

Generic hard equivalence problem $\text{EQ}(\mathcal{O}_0, \mathcal{O}_1)$:
Given $\mathcal{O}_0$ and $\mathcal{O}_1$, find (if any) an isomorphism $\phi$ s.t. $\mathcal{O}_1 = \phi(\mathcal{O}_0)$

Interesting case - when problem is hard! What can we do with it? Turns out - a lot!
Motivation

**Generic hard equivalence problem** \( \text{EQ}(\mathcal{O}_0, \mathcal{O}_1) \): Given \( \mathcal{O}_0 \) and \( \mathcal{O}_1 \), find (if any) an isomorphism \( \phi \) s.t. \( \mathcal{O}_1 = \phi(\mathcal{O}_0) \)

Interesting case - when problem is **hard**! What can we do with it? Turns out - a lot!

- Zero-Knowledge protocols
Motivation


**Generic hard equivalence problem** $\text{EQ}(O_0, O_1)$:
Given $O_0$ and $O_1$, find (if any) an isomorphism $\phi$ s.t. $O_1 = \phi(O_0)$

Interesting case - **when problem is hard!** What can we do with it? Turns out - a lot!

- Zero-Knowledge protocols
- Identification schemes (IDS)
Motivation

**Generic hard equivalence problem** \( \text{EQ}(\mathcal{O}_0, \mathcal{O}_1) \):

Given \( \mathcal{O}_0 \) and \( \mathcal{O}_1 \), find (if any) an isomorphism \( \phi \) s.t. \( \mathcal{O}_1 = \phi(\mathcal{O}_0) \)

Interesting case - when problem is **hard**! What can we do with it? Turns out - a lot!

- Zero-Knowledge protocols
- Identification schemes (IDS)
- Digital Signatures via Fiat-Shamir transform
  - F-S is a common strategy for PQ signatures
    - Dilithium, MQDSS, Picnic in first 3 rounds of NIST competition
**Generic hard equivalence problem** \( \text{EQ}(O_0, O_1) \):

Given \( O_0 \) and \( O_1 \), find (if any) an isomorphism \( \phi \) s.t. \( O_1 = \phi(O_0) \)

Interesting case - when problem is **hard**! What can we do with it? Turns out - a lot!

- **Zero-Knowledge** protocols
- **Identification schemes (IDS)**
- **Digital Signatures via Fiat-Shamir transform**
  - F-S is a common strategy for PQ signatures
    - Dilithium, MQDSS, Picnic in first 3 rounds of NIST competition
    - More than 15 in the additional round!
Zero-Knowledge Interactive Proof of knowledge from Equivalence Problems

[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.
Zero-Knowledge Interactive Proof of knowledge from Equivalence Problems

[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.

\[ P(O_0, O_1, \phi) \quad \quad V(O_0, O_1) \]
Zero-Knowledge Interactive Proof of knowledge from Equivalence Problems

[Goldreich–Micali–Wigderson ’91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.
Zero-Knowledge Interactive Proof of knowledge from Equivalence Problems

[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it

\[ P(O_0, O_1, \phi) \quad \quad V(O_0, O_1) \]

\[ \text{com} \leftarrow O' \]
Zero-Knowledge Interactive Proof of knowledge from Equivalence Problems

[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.

$$
\begin{array}{c}
O_0 \\
\downarrow \phi \\
O_1
\end{array}
\quad
\begin{array}{c}
O' \\
\downarrow \phi
\end{array}
$$

$$
\frac{P(O_0, O_1, \phi)}{P(O_0, O_1, \phi)}
\frac{\text{com} \leftarrow O'}{V(O_0, O_1)}
$$

com

com
Zero-Knowledge Interactive Proof of knowledge from Equivalence Problems

[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.

$$P(O_0, O_1, \phi) \quad \quad V(O_0, O_1)$$

$$\text{com} \leftarrow O'$$

$$\text{ch} \leftarrow_R \{0, 1\}$$
[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.
[Goldreich–Micali–Wigderson ’91]:
Let \( \phi \) be an isomorphism s.t. \( \mathcal{O}_1 = \phi(\mathcal{O}_0) \).
Given \( \mathcal{O}_0, \mathcal{O}_1 \), the prover \( \mathcal{P} \) wants to prove to the verifier \( \mathcal{V} \) knowledge of \( \phi \) without revealing any information about it.
[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.

\[
\begin{align*}
\mathcal{P}(O_0, O_1, \phi) & \quad \mathcal{V}(O_0, O_1) \\
\text{com} \leftarrow O' & \quad \text{com} \\
\text{ch} \leftarrow R \{0, 1\} & \\
\text{resp} \leftarrow \phi_{\text{ch}} & \quad \text{resp}
\end{align*}
\]
Zero-Knowledge Interactive Proof of knowledge from Equivalence Problems

[Goldreich–Micali–Wigderson ´91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.

$$\begin{align*}
\mathcal{P}(O_0, O_1, \phi) & \quad \mathcal{V}(O_0, O_1) \\
\text{com} & \leftarrow O' \\
\text{resp} & \leftarrow \phi_{\text{ch}} \\
\text{ch} & \leftarrow_R \{0, 1\}
\end{align*}$$
[Goldreich–Micali–Wigderson '91]:
Let $\phi$ be an isomorphism s.t. $O_1 = \phi(O_0)$.
Given $O_0, O_1$, the prover $P$ wants to prove to the verifier $V$ knowledge of $\phi$ without revealing any information about it.

$$P(O_0, O_1, \phi) \quad \text{V}(O_0, O_1)$$

$\text{com} \leftarrow O'$

$\phi$ ch $\leftarrow R \{0, 1\}$

$\text{resp} \leftarrow \phi_{\text{ch}}$

$O' = \phi_{\text{ch}}(O_{\text{ch}})$
Digital Signatures via the Fiat-Shamir transform

\[ \mathcal{P}(O_0, O_1, \phi) \quad \mathcal{V}(O_0, O_1) \]

\[
\begin{align*}
\text{com} & \leftarrow O' \\
\text{resp} & \leftarrow \phi_{ch} \\
\text{ch} & \leftarrow R \{0, 1\} \\
\text{resp} & \leftarrow \phi_{ch} \\
\end{align*}
\]

\[ O' \equiv \phi_{ch}(O_{ch}) \]
**Digital Signatures via the Fiat-Shamir transform**

<table>
<thead>
<tr>
<th>$\mathcal{P}(O_0, O_1, \phi)$</th>
<th>$\mathcal{V}(O_0, O_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>com $\leftarrow O’, O''$, ..., $O^{(r)}$</td>
<td>com $\leftarrow O_0, O_1$</td>
</tr>
<tr>
<td>res $\leftarrow \phi_{ch_1}, \phi_{ch_2}$, ..., $\phi_{ch_r}$</td>
<td>res $\leftarrow \phi_{ch_1}(O_{ch_1})$, ..., $\phi_{ch_r}(O_{ch_r})$</td>
</tr>
</tbody>
</table>

**IDS**

$O’ \overset{?}{=} \phi_{ch_1}(O_{ch_1}), ..., O^{(r)} \overset{?}{=} \phi_{ch_r}(O_{ch_r})$
Digital Signatures via the Fiat-Shamir transform

**IDS**

\[ P(O_0, O_1, \phi) \quad \heartsuit \quad V(O_0, O_1) \]

- `com` ← \( O', O'', \ldots, O^{(r)} \)
- `ch` ← \( R \{ 0, 1 \}^r \)
- `resp` ← \( \phi_{ch_1}, \phi_{ch_2}, \ldots, \phi_{ch_r} \)

\[ O' \overset{?}{=} \phi_{ch_1}(O_{ch_1}), \ldots, O^{(r)} \overset{?}{=} \phi_{ch_r}(O_{ch_r}) \]

**FS signature**

\[ \text{Signer}(pk, sk) \]

- `com` ← \( (O', O'', \ldots, O^{(r)}) \)
- `ch` ← \( H(m, com) \)
- `resp` ← \( (\phi_{ch_1}, \phi_{ch_2}, \ldots, \phi_{ch_r}) \)

**Verifier(pk)**

- `ch` ← \( H(m, com) \)
- `b` ← \( Vf(pk, com, ch, resp) \)

**output** : \( b \)
The basic protocol is not very efficient

▶ Challenge space is of size 2 ⇒ Soundness error is $1/2$
The basic protocol is not very efficient

\[
\begin{array}{c}
\mathcal{O}_0 \xrightarrow{\psi_0} \mathcal{O}' \\
\phi \\
\mathcal{O}_1 \xrightarrow{\psi_1} \\
\end{array}
\]

- **Challenge space is of size** \(2\) \(\Rightarrow\) Soundness error is \(1/2\)
- For security of \(\lambda\) bits, **needs to be repeated** \(r = \lambda\) times!
The basic protocol is not very efficient

\[ \mathcal{O}_0 \xrightarrow{\psi_0} \mathcal{O}' \]

\[ \mathcal{O}_1 \xrightarrow{\psi_1} \mathcal{O}' \]

- **Challenge space is of size 2** \(\Rightarrow\) Soundness error is \(1/2\)
- For security of \(\lambda\) bits, **needs to be repeated** \(r = \lambda\) times!
- \(\Rightarrow\) Signature contains \(\lambda\) isometries (from \(\lambda\) rounds)
The basic protocol is not very efficient

\[
\begin{align*}
\mathcal{O}_0 & \xrightarrow{\phi} \mathcal{O}_1 \\
& \xrightarrow{\psi_0} \mathcal{O}' \\
& \xrightarrow{\psi_1} \mathcal{O}'
\end{align*}
\]

\[
\begin{align*}
\mathcal{P}(\mathcal{O}_0, \mathcal{O}_1, \phi) & \quad \mathcal{V}(\mathcal{O}_0, \mathcal{O}_1) \\
\text{com} & \leftarrow \mathcal{O}', \mathcal{O}'', \ldots, \mathcal{O}^{(r)} \\
\text{ch} & \leftarrow_R \{0, 1\}^r \\
\text{resp} & \leftarrow \psi_{\text{ch}_1}, \psi_{\text{ch}_2}, \ldots, \psi_{\text{ch}_r} \\
\text{com} & \rightarrow \text{resp} \\
\mathcal{O}' & \overset{?}{=} \psi_{\text{ch}_1}(\mathcal{O}_{\text{ch}_1}) \\
, \ldots, \mathcal{O}^{(r)} & \overset{?}{=} \psi_{\text{ch}_r}(\mathcal{O}_{\text{ch}_r})
\end{align*}
\]

- **Challenge space is of size 2** $\Rightarrow$ Soundness error is $1/2$
- **For security of $\lambda$ bits, needs to be repeated** $r = \lambda$ **times!**
- $\Rightarrow$ Signature contains $\lambda$ isometries (from $\lambda$ rounds)
- $\Rightarrow$ All operations in signing and verification need to be repeated $\lambda$ times
Optimization 1: Make the challenge space bigger (Multiple public keys)

- Challenge space is now of size $N \Rightarrow$ Soundness error is $1/N$
Optimization 1: Make the challenge space bigger (Multiple public keys)

▶ Challenge space is now of size $N \Rightarrow$ Soundness error is $1/N$

▶ For security of $\lambda$ bits, needs to be repeated $r = \frac{\lambda}{\log N}$ times!
Optimization 1: Make the challenge space bigger (Multiple public keys)

- **Challenge space is now of size** $N \Rightarrow$ Soundness error is $1/N$
- **For security of** $\lambda$ **bits, needs to be repeated** $r = \frac{\lambda}{\log N}$ **times!**
- $\Rightarrow$ Signature contains $\frac{\lambda}{\log N}$ isometries
**Optimization 1: Make the challenge space bigger (Multiple public keys)**

![Diagram showing challenge space expansion]

- **Challenge space is now of size** $N \Rightarrow$ Soundness error is $\frac{1}{N}$
- For security of $\lambda$ bits, **needs to be repeated** $r = \frac{\lambda}{\log N}$ times!
- $\Rightarrow$ Signature contains $\frac{\lambda}{\log N}$ isometries
- $\Rightarrow$ All operations in signing and verification need to be repeated $\frac{\lambda}{\log N}$ times
Optimization 1: Make the challenge space bigger (Multiple public keys)

- Challenge space is now of size $N \Rightarrow$ Soundness error is $1/N$
- For security of $\lambda$ bits, needs to be repeated $r = \frac{\lambda}{\log N}$ times!
- $\Rightarrow$ Signature contains $\frac{\lambda}{\log N}$ isometries
- $\Rightarrow$ All operations in signing and verification need to be repeated $\frac{\lambda}{\log N}$ times
- There is a cost - $N$-fold increase in public and private key
Optimization 1: Make the challenge space bigger (Multiple public keys)

- **Challenge space is now of size** \( N \) \( \Rightarrow \) **Soundness error is** \( 1/N \)
- **For security of** \( \lambda \) **bits, needs to be repeated** \( r = \frac{\lambda}{\log N} \) **times!**
- \( \Rightarrow \) Signature contains \( \frac{\lambda}{\log N} \) isometries
- \( \Rightarrow \) All operations in signing and verification need to be repeated \( \frac{\lambda}{\log N} \) times
- **There is a cost -** \( N \)-fold increase in public and private key
- **Always necessary to find the best trade-off**
Optimization 2: Reduce signature size by using seeds

- The map $\psi_0$ is chosen at random $\Rightarrow$ one can include only seed in signature
Optimization 2: Reduce signature size by using seeds

- The map \( \psi_0 \) is chosen at random \( \Rightarrow \) one can include only seed in signature
  - \( \psi_0 \) can be reconstructed from the seed
Optimization 2: Reduce signature size by using seeds

The map \( \psi_0 \) is chosen at random \( \Rightarrow \) one can include only seed in signature

- \( \psi_0 \) can be reconstructed from the seed

Problem: This works only for \( \text{ch} = 0 \), and probability of choosing challenge 0 is \( 1/N \)
Optimization 2: Reduce signature size by using seeds

- The map $\psi_0$ is chosen at random $\Rightarrow$ one can include only seed in signature
  - $\psi_0$ can be reconstructed from the seed
- Problem: This works only for $\text{ch} = 0$, and probability of choosing challenge 0 is $1/N$
  - $\Rightarrow$ not a big benefit in general
Optimization 2: Reduce signature size by using seeds

- The map \( \psi_0 \) is chosen at random \( \Rightarrow \) one can include only seed in signature
  - \( \psi_0 \) can be reconstructed from the seed
- **Problem:** This works only for \( ch = 0 \), and probability of choosing challenge 0 is \( 1/N \)
  - \( \Rightarrow \) not a big benefit in general
  - \( \Rightarrow \) signature is not of constant size
Optimization 2: Reduce signature size by using seeds

The map $\psi_0$ is chosen at random ⇒ one can include only seed in signature
  - $\psi_0$ can be reconstructed from the seed

Problem: This works only for $ch = 0$, and probability of choosing challenge 0 is $1/N$
  - ⇒ not a big benefit in general
  - ⇒ signature is not of constant size

Idea: Always have a fixed number $M$ of 0 challenges
Optimization 2: Reduce signature size by using seeds

The map $\psi_0$ is chosen at random $\Rightarrow$ one can include only seed in signature
- $\psi_0$ can be reconstructed from the seed

Problem: This works only for $ch = 0$, and probability of choosing challenge 0 is $1/N$
- $\Rightarrow$ not a big benefit in general
- $\Rightarrow$ signature is not of constant size

Idea: Always have a fixed number $M$ of 0 challenges
- We need a special hash function that always produces fixed weight outputs
Optimization 2: Reduce signature size by using seeds

The map $\psi_0$ is chosen at random $\Rightarrow$ one can include only seed in signature
- $\psi_0$ can be reconstructed from the seed

Problem: This works only for $ch = 0$, and probability of choosing challenge 0 is $1/N$
- $\Rightarrow$ not a big benefit in general
- $\Rightarrow$ signature is not of constant size

Idea: Always have a fixed number $M$ of 0 challenges
- We need a special hash function that always produces fixed weight outputs
- Always necessary to find the best trade-off
Optimization 2: Reduce signature size by using seeds

The map $\psi_0$ is chosen at random $\Rightarrow$ one can include only seed in signature

- $\psi_0$ can be reconstructed from the seed

Problem: This works only for $ch = 0$, and probability of choosing challenge 0 is $1/N$

- $\Rightarrow$ not a big benefit in general
- $\Rightarrow$ signature is not of constant size

Idea: Always have a fixed number $M$ of 0 challenges

- We need a special hash function that always produces fixed weight outputs
- Always necessary to find the best trade-off
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** $\text{EQ}(\mathcal{O}_0, \mathcal{O}_1)$:
Given $\mathcal{O}_0$ and $\mathcal{O}_1$, find (if any) an isomorphism $\phi$ s.t. $\mathcal{O}_1 = \phi(\mathcal{O}_0)$
Interesting concrete hard equivalence problems?

<table>
<thead>
<tr>
<th>Isomorphism of polynomials</th>
<th>Patarin's signature, 1998</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quasigroup isotopy</td>
<td>Identifcation scheme, [Denes 2001]</td>
</tr>
<tr>
<td>Isogeny on elliptic curves</td>
<td>SeaSign 2018, SqiSign 2020, [De Feo et al.]</td>
</tr>
<tr>
<td>Code equivalence</td>
<td>LESS - [Biasse et al. 2020], LESS-FM - [Barenghi et al. 2021]</td>
</tr>
<tr>
<td>Alternate trilinear form equivalence</td>
<td>[Tang et al. 2022]</td>
</tr>
<tr>
<td>Lattice isomorphism</td>
<td>[Ducas–van Woerden 2022]</td>
</tr>
<tr>
<td>Matrix code equivalence</td>
<td>[Reijnders–Samardjiska–Trimoska 2022]</td>
</tr>
</tbody>
</table>

**Generic hard equivalence problem** $\text{EQ}(\mathcal{O}_0, \mathcal{O}_1)$:

Given $\mathcal{O}_0$ and $\mathcal{O}_1$, find (if any) an isomorphism $\phi$ s.t. $\mathcal{O}_1 = \phi(\mathcal{O}_0)$
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** $\text{EQ}(O_0, O_1)$:
Given $O_0$ and $O_1$, find (if any) an isomorphism $\phi$ s.t. $O_1 = \phi(O_0)$

- **Isomorphism of polynomials** - Patarin’s signature, 1998
Interesting concrete hard equivalence problems?

Generic hard equivalence problem $\text{EQ}(O_0, O_1)$:
Given $O_0$ and $O_1$, find (if any) an isomorphism $\phi$ s.t. $O_1 = \phi(O_0)$

- **Isomorphism of polynomials** - Patarin’s signature, 1998
- **Quasigroup isotopy** - Identification scheme, [Denes 2001]
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** \( EQ(O_0, O_1) \):
Given \( O_0 \) and \( O_1 \), find (if any) an isomorphism \( \phi \) s.t. \( O_1 = \phi(O_0) \)

- **Isomorphism of polynomials** - Patarin’s signature, 1998
- **Quasigroup isotopy** - Identification scheme, [Denes 2001]
- **Isogeny on elliptic curves** - SeaSign 2018, SqiSign 2020, [De Feo et al.]
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** $\text{EQ}(\mathcal{O}_0, \mathcal{O}_1)$:
Given $\mathcal{O}_0$ and $\mathcal{O}_1$, find (if any) an isomorphism $\phi$ s.t. $\mathcal{O}_1 = \phi(\mathcal{O}_0)$

- **Isomorphism of polynomials** - Patarin’s signature, 1998
- **Quasigroup isotopy** - Identification scheme, [Denes 2001]
- **Isogeny on elliptic curves** - SeaSign 2018, SqiSign 2020, [De Feo et al.]
- **Code equivalence** - LESS - [Biasse et al. 2020], LESS-FM - [Barenghi et al. 2021]
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** $EQ(O_0, O_1)$:
Given $O_0$ and $O_1$, find (if any) an isomorphism $\phi$ s.t. $O_1 = \phi(O_0)$

- **Isomorphism of polynomials** - Patarin’s signature, 1998
- **Quasigroup isotopy** - Identification scheme, [Denes 2001]
- **Isogeny on elliptic curves** - SeaSign 2018, SqiSign 2020, [De Feo et al.]
- **Code equivalence** - LESS - [Biasse et al. 2020], LESS-FM - [Barenghi et al. 2021]
- **Alternate trilinear form equivalence** - [Tang et al. 2022]
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** $\text{EQ}(\mathcal{O}_0, \mathcal{O}_1)$:
Given $\mathcal{O}_0$ and $\mathcal{O}_1$, find (if any) an isomorphism $\phi$ s.t. $\mathcal{O}_1 = \phi(\mathcal{O}_0)$

- **Isomorphism of polynomials** - Patarin’s signature, 1998
- **Quasigroup isotopy** - Identification scheme, [Denes 2001]
- **Isogeny on elliptic curves** - SeaSign 2018, SqiSign 2020, [De Feo et al.]
- **Code equivalence** - LESS - [Biasse et al. 2020], LESS-FM - [Barenghi et al. 2021]
- **Alternate trilinear form equivalence** - [Tang et al. 2022]
- **Lattice isomorphism** - [Ducas–van Woerden 2022]
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** $\text{EQ}(O_0, O_1)$:
Given $O_0$ and $O_1$, find (if any) an isomorphism $\phi$ s.t. $O_1 = \phi(O_0)$

- **Isomorphism of polynomials** - Patarin’s signature, 1998
- **Quasigroup isotopy** - Identification scheme, [Denes 2001]
- **Isogeny on elliptic curves** - SeaSign 2018, SqiSign 2020, [De Feo et al.]
- **Code equivalence** - LESS - [Biasse et al. 2020], LESS-FM - [Barenghi et al. 2021]
- **Alternate trilinear form equivalence** - [Tang et al. 2022]
- **Lattice isomorphism** - [Ducas–van Woerden 2022]
- **Matrix code equivalence** - [Reijnders–Samardjiska–Trimoska 2022]
Interesting concrete hard equivalence problems?

**Generic hard equivalence problem** $\text{EQ}(O_0, O_1)$:

Given $O_0$ and $O_1$, find (if any) an isomorphism $\phi$ s.t. $O_1 = \phi(O_0)$

- **Isomorphism of polynomials** - Patarin’s signature, 1998
- **Quasigroup isotopy** - Identification scheme, [Denes 2001]
- **Isogeny on elliptic curves** - SeaSign 2018, SqiSign 2020, [De Feo et al.]
- **Code equivalence** - LESS - [Biasse et al. 2020], LESS-FM - [Barenghi et al. 2021]
- **Alternate trilinear form equivalence** - [Tang et al. 2022]
- **Lattice isomorphism** - [Ducas–van Woerden 2022]
- **Matrix code equivalence** - [Reijnders–Samardjiska–Trimoska 2022]
- ...
Equivalence problems for MEDS and ALTEQ
Matrix Code Equivalence (MCE) problem

\[ \text{Input: Two } k \text{-dimensional matrix codes } C, D \subset M_{m \times n}(F_q) \]

\[ \text{Question: Find – if any – } A \in \text{GL}_m(F_q), B \in \text{GL}_n(F_q) \text{ s.t. for all } C \in C, \text{ it holds that } ACB \in D \]

MEDS: Matrix Code Equivalence

- MEDS is based on the following equivalence problem.
- **Matrix code** - a subspace of \( \mathcal{M}_{m \times n}(F_q) \) of dimension \( k \) endowed with rank metric.
MEDS: Matrix Code Equivalence

- MEDS is based on the following equivalence problem.
- **Matrix code** - a subspace of $\mathcal{M}_{m \times n}(\mathbb{F}_q)$ of dimension $k$ endowed with **rank metric**.

Matrix Code Equivalence (MCE) problem [Berger, 2003]

MCE$(k, n, m, q, C, D)$:

**Input**: Two $k$-dimensional matrix codes $C, D \subset \mathcal{M}_{m,n}(q)$

**Question**: Find – if any – $A \in \text{GL}_m(q), B \in \text{GL}_n(q)$ s.t. for all $C \in C$, it holds that $ACB \in D$
ALTEQ: Alternating Trilinear Form Equivalence

- ALTEQ is based on the following equivalence problem.
- **Alternating trilinear form** - a map $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ that
  (1) is linear in each argument, and
  (2) evaluates to 0 whenever two arguments are the same.
ALTEQ: Alternating Trilinear Form Equivalence

ALTEQ is based on the following equivalence problem.

Alternating trilinear form - a map $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ that

1. is linear in each argument, and
2. evaluates to 0 whenever two arguments are the same.

---

**Alternating Trilinear Form Equivalence (ATFE)** [Grochow-Qiao-Tang, 2021]

**ALTEQ**$(n, q, \phi, \psi)$:

**Input:** Two alternating trilinear forms $\phi, \psi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$.

**Question:** Find – if any – $A \in \text{GL}_n(q)$ s.t. for any $u, v, w \in \mathbb{F}_q^n$, $\phi(u, v, w) = \psi(A^t(u), A^t(v), A^t(w))$. 
MCE and ATFE look very similar!

Matrix codes:

\[ \begin{align*}
\mathbf{D}_1 & \rightarrow \mathbf{D}_2 \rightarrow \cdots \rightarrow \mathbf{D}_k \\
\mathbf{C}_1 & \rightarrow \mathbf{C}_2 \rightarrow \cdots \rightarrow \mathbf{C}_k 
\end{align*} \]
MCE and ATFE look very similar!

Matrix codes:

MCE:

- matrix codes of rectangular matrices
MCE and ATFE look very similar!

Matrix codes:

MCE:
- matrix codes of rectangular matrices
- isometry \((A, B)\)
MCE and ATFE look very similar!

- An alternating trilinear form is $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$.
- We can record $\phi$ as an $n \times n \times n$ 3-way array $C = [c_{i,j,k}]$, where $c_{i,j,k} = \phi(e_i, e_j, e_k)$.
  - Note that $c_{i,j,k} = -c_{j,i,k} = -c_{k,j,i} = -c_{i,k,j} = c_{j,k,i} = c_{k,i,j}$.
- A 3-way array $C$ can also be represented as a matrix tuple $(C_1, \ldots, C_n)$, $C_i \in \mathcal{M}_n(q)$. 

MCE and ATFE look very similar!

- An alternating trilinear form is $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$.
- We can record $\phi$ as an $n \times n \times n$ 3-way array $C = [c_{i,j,k}]$, where $c_{i,j,k} = \phi(e_i, e_j, e_k)$.
  - Note that $c_{i,j,k} = -c_{j,i,k} = -c_{k,j,i} = -c_{i,k,j} = c_{j,k,i} = c_{k,i,j}$.
- A 3-way array $C$ can also be represented as a matrix tuple $(C_1, \ldots, C_n)$, $C_i \in \mathcal{M}_n(q)$.

ATFE:

- matrix codes with “symmetries in the three directions”.
- isometry $(A, A^\top)$ and $A$ on the third direction too.
MCE and ATFE are polynomial-time equivalent

- The objects in MCE and ATFE are both 3-way arrays.
  - A 2-way array, $[c_{i,j}]$, is a matrix.
  - A 3-way array, $[c_{i,j,k}]$, is sometimes called a 3-tensor.
  - The 3-way arrays from ATFE are subject to certain structural constraints.
Theorem ([Grochow-Qiao-Tang, 2023])

MCE and ATFE are polynomial-time equivalent.

The objects in MCE and ATFE are both 3-way arrays.
- A 2-way array, $[c_{i,j}]$, is a matrix.
- A 3-way array, $[c_{i,j,k}]$, is sometimes called a 3-tensor.
- The 3-way arrays from ATFE are subject to certain structural constraints.

The isomorphisms in MCE and ATFE are both invertible matrices.
- $L, R \in \text{GL}_n(q)$ sends $C \in \mathcal{M}_n(q)$ to $L^t C R$.
- $L, R, T = (t_{i,j}) \in \text{GL}_n(q)$ sends $(C_1, \ldots, C_n) \in \mathcal{M}_n(q)^n$ to $(L^t C'_1 R, \ldots, L^t C'_n R)$, where $C'_i = \sum_j t_{i,j} C_j$.
- The isomorphism in ATFE imposes that $L = R = T$. 

14
The objects in MCE and ATFE are both 3-way arrays.
- A 2-way array, $[c_{i,j}]$, is a matrix.
- A 3-way array, $[c_{i,j,k}]$, is sometimes called a 3-tensor.
- The 3-way arrays from ATFE are subject to certain structural constraints.

The isomorphisms in MCE and ATFE are both invertible matrices.
- $L, R \in \text{GL}_n(q)$ sends $C \in \mathcal{M}_n(q)$ to $L^t CR$.
- $L, R, T = (t_{i,j}) \in \text{GL}_n(q)$ sends $(C_1, \ldots, C_n) \in \mathcal{M}_n(q)^n$ to $(L^t C_1' R, \ldots, L^t C_n' R)$, where $C_i' = \sum_j t_{i,j} C_j$.
- The isomorphism in ATFE imposes that $L = R = T$.

Theorem ([Grochow-Qiao-Tang, 2023])

MCE and ATFE are polynomial-time equivalent.
A complexity class for isomorphism problems of algebraic structures

- Relations between isomorphism problems for some algebraic structures are studied in [Reijnders–Samardjiska–Trimoska, Grochow–Qiao–Tang, D’Alconzo, Couvreur–Debris-Alazard–Gaborit... ]
A complexity class for isomorphism problems of algebraic structures

- Relations between isomorphism problems for some algebraic structures are studied in [Reijnders–Samardjiska–Trimoska, Grochow–Qiao–Tang, D’Alconzo, Couvreur–Debris-Alazard–Gaborit...]

- The complexity class $\text{Tl}$ was defined in [Grochow-Qiao], consisting of problems polynomial-time reducible to MCE.
  - MCE was called 3-Tensor Isomorphism in [Grochow-Qiao].
  - In analogy with the complexity class GI for Graph Isomorphism.

- MCE and ATFE are $\text{Tl}$-complete.
A complexity class for isomorphism problems of algebraic structures

- Relations between isomorphism problems for some algebraic structures are studied in [Reijnders–Samardjiska–Trimoska, Grochow–Qiao–Tang, D’Alconzo, Couvreur–Debris-Alazard–Gaborit…]

- The complexity class $\text{Tl}$ was defined in [Grochow-Qiao], consisting of problems polynomial-time reducible to MCE.
  - MCE was called 3-Tensor Isomorphism in [Grochow-Qiao].
  - In analogy with the complexity class GI for Graph Isomorphism.

- MCE and ATFE are $\text{Tl}$-complete.

- $\text{Tl}$-complete problems include isomorphism problems for tensors, finite groups, (associative and Lie) algebras, (systems of) polynomials…
Relations with other isomorphism problems

- TI-complete problems appear in computational group theory, multivariate cryptography, and quantum information.
  - Experiences from these areas suggest that TI-complete problems are difficult to solve in practice.
Relations with other isomorphism problems

- **TI-complete** problems appear in computational group theory, multivariate cryptography, and quantum information.
  - Experiences from these areas suggest that TI-complete problems are difficult to solve in practice.
- Isomorphism problems for cubic forms and quadratic polynomial systems, as studied since 1996 [Patarin], are TI-complete.
  - Results from the study of polynomial isomorphism are valuable for MCE and ATFE.
Relations with other isomorphism problems

- **TI**-complete problems appear in computational group theory, multivariate cryptography, and quantum information.
  - Experiences from these areas suggest that TI-complete problems are difficult to solve in practice.

- Isomorphism problems for cubic forms and quadratic polynomial systems, as studied since 1996 [Patarin], are TI-complete.
  - Results from the study of polynomial isomorphism are valuable for MCE and ATFE.

- Linear code monomial equivalence and graph isomorphism are in TI [Couvreur–Debris-Alazard–Gaborit, Grochow–Qiao].
  - Linear code monomial equivalence supports LESS.
Why use MCE and ATFE in post-quantum cryptography?

- A natural development of Shor’s quantum algorithms for integer factorisation and discrete logarithm is the hidden subgroup problem framework.
- MCE and ATFE can be cast in this framework for general linear groups.
Why use MCE and ATFE in post-quantum cryptography?

- A natural development of Shor’s quantum algorithms for integer factorisation and discrete logarithm is the hidden subgroup problem framework.
- MCE and ATFE can be cast in this framework for general linear groups.
- A strong negative evidence for the “standard technique” to work in this setting [Hallgren-Moore-Rötteler-Russell-Sen, 2010].

[Moore-Russell-Vazirani] ... the strongest such insights we have about the limits of quantum algorithms.
Cryptanalysis for MCE and ATFE
We will introduce three approaches.

• Direct Gröbner basis attack.

• Hybrid Gröbner basis: $q^n \cdot \text{poly}(n, \log q)$.

• Utilising low-rank points (via birthday paradox and invariants).

Consider 3-way arrays of size $n \times n \times n$ over $\mathbb{F}_q$ under the action of $(L, R, T)$ or $(T, T, T) \in \text{GL}_n(q) \times \text{GL}_n(q) \times \text{GL}_n(q)$.

Brute-force algorithm: $q^{n^2} \cdot \text{poly}(n, \log q)$.

• After fixing $T$, to recover $L$ and $R$ can be done in time $\text{poly}(n, \log q)$. 
Consider 3-way arrays of size $n \times n \times n$ over $\mathbb{F}_q$ under the action of $(L, R, T)$ or $(T, T, T) \in \text{GL}_n(q) \times \text{GL}_n(q) \times \text{GL}_n(q)$.

Brute-force algorithm: $q^{n^2} \cdot \text{poly}(n, \log q)$.

- After fixing $T$, to recover $L$ and $R$ can be done in time $\text{poly}(n, \log q)$.

We will introduce three approaches.

- Direct Gröbner basis attack.
- Hybrid Gröbner basis: $q^n \cdot \text{poly}(n, \log q)$.
- Utilising low-rank points (via birthday paradox and invariants).
Let $C = [c_{i,j,k}]$ and $D = [d_{i,j,k}]$ be two $n \times n \times n$ 3-way arrays over $\mathbb{F}_q$. We view $C$ as a matrix tuple $(C_1, \ldots, C_n)$, $C_i \in \mathcal{M}_n(q)$. 

Direct Gröbner basis attack: the basic idea

- View the entries of $L$, $R$, and $T$ as variables.
- The question is whether $(L_t C'_1 R, \ldots, L_t C'_n R) = (D_1, \ldots, D_n)$.
- This amounts to $n^3$ cubic polynomials in $3n^2$ variables.

Let $C = [c_{i,j,k}]$ and $D = [d_{i,j,k}]$ be two $n \times n \times n$ 3-way arrays over $\mathbb{F}_q$. We view $C$ as a matrix tuple $(C_1, \ldots, C_n)$, $C_i \in \mathcal{M}_n(q)$. 

Direct Gröbner basis attack: the basic idea

- View the entries of $L$, $R$, and $T$ as variables.
- The question is whether $(L_t C'_1 R, \ldots, L_t C'_n R) = (D_1, \ldots, D_n)$.
- This amounts to $n^3$ cubic polynomials in $3n^2$ variables.
Let $C = [c_{i,j,k}]$ and $D = [d_{i,j,k}]$ be two $n \times n \times n$ 3-way arrays over $\mathbb{F}_q$.

We view $C$ as a matrix tuple $(C_1, \ldots, C_n)$, $C_i \in M_n(q)$.

Recall that $L, R, T = (t_{i,j}) \in \text{GL}_n(q)$ sends $(C_1, \ldots, C_n) \in M_n(q)^n$ to $(L^t C_1^t R, \ldots, L^t C_n^t R)$, where $C_i' = \sum_j t_{i,j} C_j$. 

Direct Gröbner basis attack: the basic idea
Direct Gröbner basis attack: the basic idea

- Let $\mathbf{C} = [c_{i,j,k}]$ and $\mathbf{D} = [d_{i,j,k}]$ be two $n \times n \times n$ 3-way arrays over $\mathbb{F}_q$.
- We view $\mathbf{C}$ as a matrix tuple $(C_1, \ldots, C_n)$, $C_i \in \mathcal{M}_n(q)$.
- Recall that $L, R, T = (t_{i,j}) \in \text{GL}_n(q)$ sends $(C_1, \ldots, C_n) \in \mathcal{M}_n(q)^n$ to $(L^t C'_1 R, \ldots, L^t C'_n R)$, where $C'_i = \sum_j t_{i,j} C_j$.
- Viewing the entries of $L$, $R$ and $T$ as variables, the question is whether $(L^t C'_1 R, \ldots, L^t C'_n R) = (D_1, \ldots, D_n)$.
  - This amounts to $n^3$ cubic polynomials in $3n^2$ variables.
Quadratic inverse modelling

For ATFE, let \( T' = \begin{bmatrix} t'_{i,j} \end{bmatrix} \). Then set \((T^t C_1 T, \ldots, T^t C_n T) = (D'_1, \ldots, D'_n)\) where \( D'_i = \sum_j t_{i,j} C_j \) and \( TT' = I_n \).

\[ \text{This is by [Bouillaguet-Faugère-Fouque-Perret, 2010].} \]

\[ \text{n} \cdot \frac{n^2}{2} + n^2 \text{ quadratic polynomials in } 2n^2 \text{ variables.} \]

Quadratic dual modelling

Use the dual space of \( D \) to express that \( L^t C_i R \in D \).

\[ \text{This is by [Chou-Niederhagen-Persichetti-Randrianarisoa-Reijnders-Samardjiska-Trimoska].} \]

\[ \text{This gives rise to } n \cdot (n^2 - n) \text{ homogeneous quadratic polynomials in } 2n^2 \text{ variables for MCE.} \]

\[ \text{And } n \cdot (n^2 - n) \text{ quadratic polynomials in } n^2 \text{ variables for ATFE.} \]

Cubic modelling

\( (L^t C'_1 R, \ldots, L^t C'_n R) = (D_1, \ldots, D_n) \) where \( C'_i = \sum_j t_{i,j} C_j \).

\[ \text{This gives rise to } n^3 \text{ cubic polynomials in } 3n^2 \text{ variables for MCE.} \]

\[ \text{And } \binom{n}{3} \text{ cubic polynomials in } n^2 \text{ variables for ATFE.} \]
Quadratic dual modelling

Use the dual space of $D$ to express that $L^t C_i R \in D$.

▶ This is by [Chou-Niederhagen-Persichetti-Randrianarisoa-Reijnders-Samardjiska-Trimoska].

▶ This gives rise to $n \cdot (n_2) = n \cdot (n_2)$ homogeneous quadratic polynomials in $2n^2$ variables for MCE.

▶ And $(n^3)$ cubic polynomials in $n^2$ variables for ATFE.

Cubic modelling $(L^t C_1^1 R, \ldots, L^t C_n^1 R) = (D_1, \ldots, D_n)$ where $C_i^1 = \sum_j t_{ij} C_j$.

▶ This gives rise to $n^3$ cubic polynomials in $3n^2$ variables for MCE.

▶ And $(n^3)$ cubic polynomials in $n^2$ variables for ATFE.

Quadratic inverse modelling For ATFE, let $T' = [t'_{ij}]$. Then set

$$(T^t C_1^1 T, \ldots, T^t C_n^1 T) = (D'_1, \ldots, D'_n)$$

where $D'_i = \sum_j t'_{ij} D_j$, and $TT' = I_n$.

▶ This is by [Bouillaguet-Faugère-Fouque-Perret, 2010].

▶ $n \cdot (n^2) + n^2$ quadratic polynomials in $2n^2$ variables.
Direct Gröbner basis attack: more efficient modellings

Cubic modelling \( (L^t C'_1 R, \ldots, L^t C'_n R) = (D_1, \ldots, D_n) \) where \( C'_i = \sum_j t_{i,j} C_j \).

- This gives rise to \( n^3 \) cubic polynomials in \( 3n^2 \) variables for MCE.
- And \( \binom{n}{3} \) cubic polynomials in \( n^2 \) variables for ATFE.

Quadratic inverse modelling For ATFE, let \( T' = [t'_{i,j}] \). Then set

\[
(T^t C_1 T, \ldots, T^t C_n T) = (D'_1, \ldots, D'_n)
\]

where \( D'_i = \sum_j t'_{i,j} D_j \), and \( TT' = I_n \).

- This is by [Bouillaguet-Faugère-Fouque-Perret, 2010].
- \( n \cdot \binom{n}{2} + n^2 \) quadratic polynomials in \( 2n^2 \) variables.

Quadratic dual modelling Use the dual space of \( D \) to express that \( L^t C_i R \in D \).

- This is by [Chou-Niederhagen-Persichetti-Randrianarisoa-Reijnders-Samardjiska-Trimoska].
- This gives rise to \( n \cdot (n^2 - n) \) homogeneous quadratic polynomials in \( 2n^2 \) variables for MCE.
- And \( n \cdot \left( \binom{n}{2} - n \right) \) quadratic polynomials in \( n^2 \) variables for ATFE.
- Note that some syzygies arise, complicating the analysis [MEDS spec].
Hybrid Gröbner basis attacks

- We set up $n \times n$ variable matrices $L$ and $R$ for MCE (or $T$ and $T'$ for ATFE).
- In [Faugère-Perret, 2006], it was discovered that Gröbner basis runs in polynomial time, provided that one (or two) rows of $L$ are known.
Further observations from [Beullens, 2023]:

- Knowing one row of $T$ up to scalar is enough.
- For low-rank points, the kernel information can be incorporated.

- We set up $n \times n$ variable matrices $L$ and $R$ for MCE (or $T$ and $T'$ for ATFE).
- In [Faugère-Perret, 2006], it was discovered that Gröbner basis runs in polynomial time, provided that one (or two) rows of $L$ are known.
- For ATFE, knowing one row of $T$ is enough, leading to a $q^n \cdot \text{poly}(n, \log q)$-time algorithm.
- For MCE, knowing two rows of $L$ is enough, leading to an $q^{2n} \cdot \text{poly}(n, \log q)$-time algorithm.
Hybrid Gröbner basis attacks

- We set up $n \times n$ variable matrices $L$ and $R$ for MCE (or $T$ and $T'$ for ATFE).
- In [Faugère-Perret, 2006], it was discovered that Gröbner basis runs in polynomial time, provided that one (or two) rows of $L$ are known.
- For ATFE, knowing one row of $T$ is enough, leading to a $q^n \cdot \text{poly}(n, \log q)$-time algorithm.
- For MCE, knowing two rows of $L$ is enough, leading to an $q^{2n} \cdot \text{poly}(n, \log q)$-time algorithm.
- Further observations from [Beullens, 2023]:
  - Knowing one row of $T$ up to scalar is enough.
  - For low-rank points, the kernel information can be incorporated.
Utilising low-rank points

- Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ be an alternating trilinear form.
- For $u \in \mathbb{F}_q^n$, let $\phi_u : \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ by $\phi_u(v, w) = \phi(u, v, w)$.
- An isomorphism invariant for $u$: $r = \text{Rank}(\phi_u)$. 
Utilising low-rank points

- Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ be an alternating trilinear form.
- For $u \in \mathbb{F}_q^n$, let $\phi_u : \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ by $\phi_u(v, w) = \phi(u, v, w)$.
- An isomorphism invariant for $u$: $r = \text{Rank}(\phi_u)$.
- Algorithms based on birthday paradox and hybrid Gröbner basis
  [Bouillaguet-Fouque-Véber, 2013; Beullens, 2023].
  - Suppose there exist $\approx q^k$-many rank-$r$ points for a random $\phi$.
  1. Sample $q^{k/2}$-many rank-$r$ points for $\phi$ and $\psi$, respectively.
  2. For every pair, use hybrid Gröbner basis to find a “matched” pair.
- Algorithm cost: $O(q^{k/2} \cdot \text{samp-cost} + q^k \cdot \text{gb-cost})$. 
Utilising low-rank points

Let $\phi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ be an alternating trilinear form.

For $u \in \mathbb{F}_q^n$, let $\phi_u : \mathbb{F}_q^n \times \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ by $\phi_u(v, w) = \phi(u, v, w)$.

An isomorphism invariant for $u$: $r = \text{Rank}(\phi_u)$.

Algorithms based on birthday paradox and hybrid Gröbner basis

[Bouillaguet-Fouque-Véber, 2013; Beullens, 2023].

- Suppose there exist $\approx q^k$-many rank-$r$ points for a random $\phi$.

(1) Sample $q^{k/2}$-many rank-$r$ points for $\phi$ and $\psi$, respectively.

(2) For every pair, use hybrid Gröbner basis to find a “matched” pair.

- Algorithm cost: $O(q^{k/2} \cdot \text{samp-cost} + q^k \cdot \text{gb-cost})$.

Sampling step: min-rank or graph-walking [Beullens, 2023]
Utilising low-rank points, cont’d

- Algorithms based on distinguishing isomorphism invariants with low-rank points [Bouillaguet-Fouque-Véber, 2013; Beullens, 2023].
  - Suppose there exist $\approx q^k$-many rank-$r$ points for a random $\phi$.
  - Suppose there exist distinguishing isomorphism invariants associated with such points.
Utilising low-rank points, cont’d

- Algorithms based on distinguishing isomorphism invariants with low-rank points
  [Bouillaguet-Fouque-Véber, 2013; Beullens, 2023].

  - Suppose there exist $\approx q^k$-many rank-$r$ points for a random $\phi$.
  - Suppose there exist distinguishing isomorphism invariants associated with such points.

  1. Sample $q^{k/2}$-many rank-$r$ points for $\phi$ and $\psi$, respectively.
  2. For every point, compute the isomorphism invariant.
  3. By birthday paradox, there exists a pair of points of the same invariant. Use hybrid Gröbner basis to complete.
Algorithms based on distinguishing isomorphism invariants with low-rank points
[Bouillaguet-Fouque-Véber, 2013; Beullens, 2023].

- Suppose there exist \( \approx q^k \)-many rank-\( r \) points for a random \( \phi \).
- Suppose there exist distinguishing isomorphism invariants associated with such points.

1. Sample \( q^{k/2} \)-many rank-\( r \) points for \( \phi \) and \( \psi \), respectively.
2. For every point, compute the isomorphism invariant.
3. By birthday paradox, there exists a pair of points of the same invariant. Use hybrid Gröbner basis to complete.

- Algorithm cost: \( O(q^{k/2} \cdot (\text{samp-cost} + \text{inv-cost}) + \text{gb-cost}) \).
Utilising low-rank points, cont’d

▶ Algorithms based on distinguishing isomorphism invariants with low-rank points [Bouillaguet-Fouque-Véber, 2013; Beullens, 2023].

- Suppose there exist \( \approx q^k \)-many rank-\( r \) points for a random \( \phi \).
- Suppose there exist distinguishing isomorphism invariants associated with such points.

1. Sample \( q^{k/2} \)-many rank-\( r \) points for \( \phi \) and \( \psi \), respectively.
2. For every point, compute the isomorphism invariant.
3. By birthday paradox, there exists a pair of points of the same invariant. Use hybrid Gröbner basis to complete.

- Algorithm cost: \( O(q^{k/2} \cdot (\text{samp-cost} + \text{inv-cost}) + \text{gb-cost}) \).

▶ Distinguishing isomorphism invariant candidates: ranks of the neighbours of low-rank points, and more [Narayanan-Qiao-Tang].
Parameters and performances of MEDS and ALTEQ
Parameters and performance of MEDS

<table>
<thead>
<tr>
<th>Level</th>
<th>param. set</th>
<th>public key size (KB)</th>
<th>signature size (KB)</th>
<th>key gen (ms)</th>
<th>sign (ms)</th>
<th>verify (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>MEDS-9923</td>
<td>9.9</td>
<td>9.9</td>
<td>1</td>
<td>272</td>
<td>271</td>
</tr>
<tr>
<td></td>
<td>MEDS-13220</td>
<td>13.2</td>
<td>13</td>
<td>1.3</td>
<td>46.7</td>
<td>46</td>
</tr>
<tr>
<td>III</td>
<td>MEDS-41711</td>
<td>41.7</td>
<td>41</td>
<td>5.1</td>
<td>779</td>
<td>762</td>
</tr>
<tr>
<td></td>
<td>MEDS-69497</td>
<td>55.6</td>
<td>54.7</td>
<td>6.7</td>
<td>203.8</td>
<td>200.4</td>
</tr>
</tbody>
</table>

**Table:** An overview of the parameters and performance of MEDS.

Optimizations:

- **Standard:** Multiple Public Keys + Fixed-Weight Challenge Strings + Seed tree
- **New:** Public Key Compression
  - generate public key partially from seed ⇒ signature size reduction
  - **Work in progress:** use similar idea during signing
### Parameters and performance of ALTEQ

<table>
<thead>
<tr>
<th>Level</th>
<th>mode</th>
<th>public key size (KB)</th>
<th>signature size (KB)</th>
<th>key gen (ms)</th>
<th>sign (ms)</th>
<th>verify (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Balanced</td>
<td>8</td>
<td>16</td>
<td>0.093</td>
<td>0.629</td>
<td>0.496</td>
</tr>
<tr>
<td></td>
<td>ShortSig</td>
<td>512</td>
<td>10</td>
<td>1.902</td>
<td>0.194</td>
<td>0.092</td>
</tr>
<tr>
<td>III</td>
<td>Balanced</td>
<td>32</td>
<td>48</td>
<td>0.582</td>
<td>6.986</td>
<td>6.483</td>
</tr>
<tr>
<td></td>
<td>ShortSig</td>
<td>1024</td>
<td>24</td>
<td>5.152</td>
<td>1.705</td>
<td>1.304</td>
</tr>
</tbody>
</table>

#### Table: An overview of the parameters and performance of ALTEQ.

#### Optimizations:

- **Standard**: Multiple Public Keys + Fixed-Weight Challenge Strings (+ Seed tree)
- **New**: Invertible matrix decomposition
  - Represent an invertible matrix as a product of column matrices for faster signing and verification
Digital signature based on equivalence problems: design and optimisations

Matrix code equivalence (MCE) and alternating trilinear form equivalence (ATFE)

Algorithms for MCE and ATFE

MEDS and ALTEQ: parameters and performances
Thank you for listening!