A New Formulation of the Linear Equivalence Problem and Shorter LESS Signatures

Asiacrypt 2023

Edoardo Persichetti

7 December 2023

FAU Department of Mathematical Sciences
Charles E. Schmidt College of Science
Florida Atlantic University

In This Talk

Roadmap

- Background
- Signatures from Code Equivalence
- A New Formulation
- Conclusions

Roadmap

- Background
- Signatures from Code Equivalence
- A New Formulation
- Conclusions

$[n, k]$ Linear Code over $\mathbb{F}_q$

A subspace of dimension k of $\mathbb{F}_q^n$. Value n is called length.

Hamming Metric

$wt(x) = |\{i : x_i \neq 0, 1 \leq i \leq n\}|$, $d(x, y) = wt(x - y)$.
Minimum distance (of $\mathfrak{C}$): $\min\{d(x, y) : x, y \in \mathfrak{C}\}$.

Generator Matrix

$G \in \mathbb{F}_q^{k \times n}$ defines the code as : $x{\in}\mathfrak{C} \iff x = uG$ for $u \in \mathbb{F}_q^k$.
Not unique: $SG, S \in \mathrm{GL}_k(q)$; Systematic form: $(I_k \mid M)$.

Parity-check Matrix

$H \in \mathbb{F}_q^{(n-k) \times n}$ defines the code as: $x{\in}\mathfrak{C} \iff Hx^T = 0$ (syndrome).
Not unique: $SH, S \in \mathrm{GL}_{n-k}(q)$; Systematic form: $(M^T \mid I_{n-k})$.

Information Set: set of columns carrying information symbols (G_J is invertible).

w-error correcting: $\exists$ algorithm that corrects up to w errors.

Decoding Problems

In general, it is hard to decode random codes.

General Decoding Problem (GDP)

Given: $G \in \mathbb{F}_q^{k \times n}$, $y \in \mathbb{F}_q^n$ and $w \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq w$ such that $y - e = x \in \mathfrak{C}_G$.

Easy to see this is equivalent to the following.

Syndrome Decoding Problem (SDP)

Given: $H \in \mathbb{F}_q^{(n-k) \times n}$, $y \in \mathbb{F}_q^{(n-k)}$ and $w \in \mathbb{N}$.
Goal: find a word $e \in \mathbb{F}_q^n$ with $wt(e) \leq w$ such that $He^T = y$.

NP-Complete (Berlekamp, McEliece and Van Tilborg, 1978; Barg, 1994).

Unique solution when w is below a certain threshold (GV Bound).

Very well-studied, solid security understanding Information-Set Decoding (ISD) solvers.

Use hard problems from coding theory, such as SDP in the Hamming metric.

Use hard problems from coding theory, such as SDP in the Hamming metric.

For encryption, one can obtain a trapdoor by masking the private code.

Use hard problems from coding theory, such as SDP in the Hamming metric.

For encryption, one can obtain a trapdoor by masking the private code.

Example (McEliece/Niederreiter):
use change of basis S and permutation P to obtain equivalent code.

Traditional Code-Based Cryptography

Use hard problems from coding theory, such as SDP in the Hamming metric.

For encryption, one can obtain a trapdoor by masking the private code.

Example (McEliece/Niederreiter):
use change of basis S and permutation P to obtain equivalent code.

Hardness is an assumption which depends on chosen code family.

Traditional Code-Based Cryptography

Use hard problems from coding theory, such as SDP in the Hamming metric.

For encryption, one can obtain a trapdoor by masking the private code.

Example (McEliece/Niederreiter):
use change of basis S and permutation P to obtain equivalent code.

Hardness is an assumption which depends on chosen code family.

This works well for encryption...

(Classic McEliece, BIKE, HQC)

Use hard problems from coding theory, such as SDP in the Hamming metric.

For encryption, one can obtain a trapdoor by masking the private code.

Example (McEliece/Niederreiter):
use change of basis S and permutation P to obtain equivalent code.

Hardness is an assumption which depends on chosen code family.

This works well for encryption...
(Classic McEliece, BIKE, HQC)

...far less so for signature schemes.
(CFS, KKS,...)

Traditional Code-Based Cryptography

Use hard problems from coding theory, such as SDP in the Hamming metric.

For encryption, one can obtain a trapdoor by masking the private code.

Example (McEliece/Niederreiter):
use change of basis S and permutation P to obtain equivalent code.

Hardness is an assumption which depends on chosen code family.

This works well for encryption...
(Classic McEliece, BIKE, HQC)

...far less so for signature schemes.
(CFS, KKS,...)

History suggest that we have to do things a little differently.

Roadmap

- Background
- Signatures from Code Equivalence
- A New Formulation
- Conclusions

Group Action

Let $\mathcal{X}$ be a set and $(\mathcal{G}, \cdot)$ be a group. A group action is a mapping

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \to & \mathcal{X} \\ & (g, x) & \mapsto & g \star x \end{array}$$

such that, for all $x \in \mathcal{X}$ and $g_1, g_2 \in \mathcal{G}$, $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$.

Group Action

Let $\mathcal{X}$ be a set and $(\mathcal{G}, \cdot)$ be a group. A group action is a mapping

$$\begin{aligned} \star : \quad \mathcal{G} \times \mathcal{X} & \to \quad \mathcal{X} \\ (g, x) & \mapsto \quad g \star x \end{aligned}$$

such that, for all $x \in \mathcal{X}$ and $g_1, g_2 \in \mathcal{G}$, $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$.

The word cryptographic means that it has some properties of interest in cryptography, e.g.:

Group Action

Let $\mathcal{X}$ be a set and $(\mathcal{G}, \cdot)$ be a group. A group action is a mapping

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \to & \mathcal{X} \\ & (g, x) & \mapsto & g \star x \end{array}$$

such that, for all $x \in \mathcal{X}$ and $g_1, g_2 \in \mathcal{G}$, $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$.

The word cryptographic means that it has some properties of interest in cryptography, e.g.:

- Efficient evaluation, sampling and membership testing algorithms.

Group Action

Let $\mathcal{X}$ be a set and $(\mathcal{G}, \cdot)$ be a group. A group action is a mapping

$$\begin{array}{rrcl} \star : & \mathcal{G} \times \mathcal{X} & \to & \mathcal{X} \\ & (g, x) & \mapsto & g \star x \end{array}$$

such that, for all $x \in \mathcal{X}$ and $g_1, g_2 \in \mathcal{G}$, $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$.

The word cryptographic means that it has some properties of interest in cryptography, e.g.:

- Efficient evaluation, sampling and membership testing algorithms.
- A hard vectorization problem.

Group Action

Let $\mathcal{X}$ be a set and $(\mathcal{G}, \cdot)$ be a group. A group action is a mapping

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \to & \mathcal{X} \\ & (g, x) & \mapsto & g \star x \end{array}$$

such that, for all $x \in \mathcal{X}$ and $g_1, g_2 \in \mathcal{G}$, $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$.

The word cryptographic means that it has some properties of interest in cryptography, e.g.:

- Efficient evaluation, sampling and membership testing algorithms.
- A hard vectorization problem.

Group Action Vectorization Problem

Given the pair $x_1, x_2 \in \mathcal{X}$, find, if any, $g \in \mathcal{G}$ such that $g \star x_1 = x_2$.

Group Action

Let $\mathcal{X}$ be a set and $(\mathcal{G}, \cdot)$ be a group. A group action is a mapping

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \to & \mathcal{X} \\ & (g, x) & \mapsto & g \star x \end{array}$$

such that, for all $x \in \mathcal{X}$ and $g_1, g_2 \in \mathcal{G}$, $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$.

The word cryptographic means that it has some properties of interest in cryptography, e.g.:

- Efficient evaluation, sampling and membership testing algorithms.
- A hard vectorization problem.

Group Action Vectorization Problem

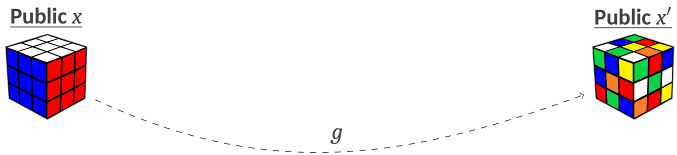
Given the pair $x_1, x_2 \in \mathcal{X}$, find, if any, $g \in \mathcal{G}$ such that $g \star x_1 = x_2$.

Most famous example: exactly DLP!

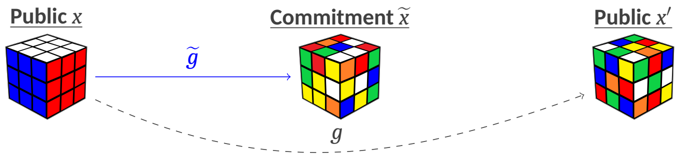
There is a standard way to obtain a simple 3-pass Sigma protocol from group actions.

Zero-Knowledge from Group Actions

There is a standard way to obtain a simple 3-pass Sigma protocol from group actions.

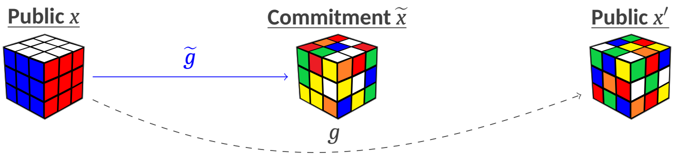


There is a standard way to obtain a simple 3-pass Sigma protocol from group actions.



$\widetilde{g}$ is a random element from $\mathcal{G}$.

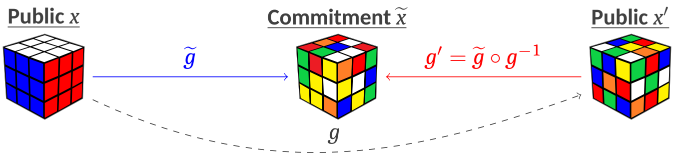
There is a standard way to obtain a simple 3-pass Sigma protocol from group actions.



$\widetilde{g}$ is a random element from $\mathcal{G}$.

If $ch = 0$: reveal $\widetilde{g}$

There is a standard way to obtain a simple 3-pass Sigma protocol from group actions.

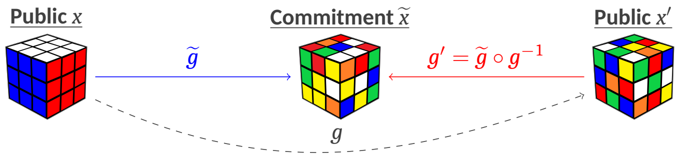


$\widetilde{g}$ is a random element from $\mathcal{G}$.

If $ch = 0$: reveal $\widetilde{g}$

If $ch = 1$: reveal g'

There is a standard way to obtain a simple 3-pass Sigma protocol from group actions.



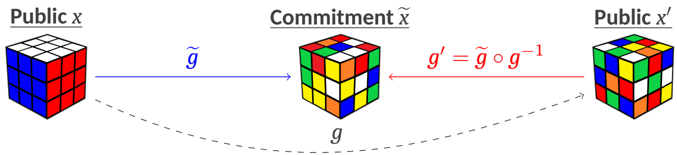
$\widetilde{g}$ is a random element from $\mathcal{G}$.

If $ch = 0$: reveal $\widetilde{g}$

If $ch = 1$: reveal g'

This naturally yields signatures (via Fiat-Shamir) but, in the DLP setting, these are obviously not post-quantum.

There is a standard way to obtain a simple 3-pass Sigma protocol from group actions.



$\widetilde{g}$ is a random element from $\mathcal{G}$.

If $ch = 0$: reveal $\widetilde{g}$

If $ch = 1$: reveal g'

This naturally yields signatures (via Fiat-Shamir) but, in the DLP setting, these are obviously not post-quantum.

What about group actions from coding theory?

Maps which preserve the distances (weights).

Isometries in the Hamming Metric

Maps which preserve the distances (weights).

- Permutations: $\pi\big((a_1, a_2, \dots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \dots, a_{\pi(n)}\big)$.

Maps which preserve the distances (weights).

- Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
- Monomials: permutations + scaling factors: $\mu = (v; \pi)$, with $v \in (\mathbb{F}_q^*)^n$

$$\mu\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

Monomial matrix: permutation $\times$ diagonal.

Maps which preserve the distances (weights).

- Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
- Monomials: permutations + scaling factors: $\mu = (v; \pi)$, with $v \in (\mathbb{F}_q^*)^n$

  $$\mu\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

  Monomial matrix: permutation $\times$ diagonal.
- Monomials + field automorphism.

Maps which preserve the distances (weights).

- Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
- Monomials: permutations + scaling factors: $\mu = (v; \pi)$, with $v \in (\mathbb{F}_q^*)^n$

$$\mu\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

  Monomial matrix: permutation $\times$ diagonal.
- Monomials + field automorphism.

Two codes are equivalent if they are connected by an isometry.

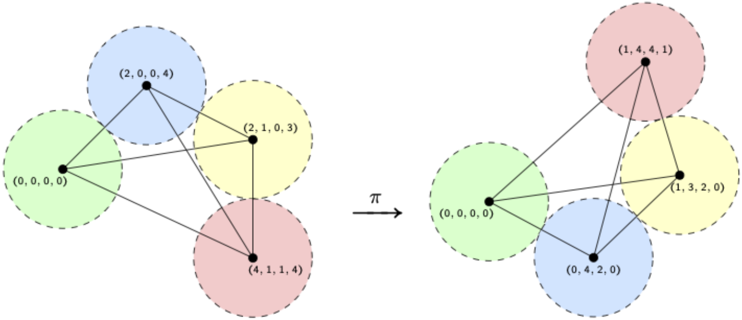
Maps which preserve the distances (weights).

- Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
- Monomials: permutations + scaling factors: $\mu = (v; \pi)$, with $v \in (\mathbb{F}_q^*)^n$

  $$\mu\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

  Monomial matrix: permutation $\times$ diagonal.
- Monomials + field automorphism.

Two codes are equivalent if they are connected by an isometry.



Maps which preserve the distances (weights).

- Permutations: $\pi\big(\,(a_1, a_2, \ldots, a_n)\,\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
- Monomials: permutations + scaling factors: $\mu = (v; \pi)$, with $v \in (\mathbb{F}_q^*)^n$

$$\mu\big(\,(a_1, a_2, \ldots, a_n)\,\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

  Monomial matrix: permutation $\times$ diagonal.

- Monomials + field automorphism.

Two codes are equivalent if they are connected by an isometry.

We talk about permutation, linear and semilinear equivalence, respectively.

Maps which preserve the distances (weights).

- Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
- Monomials: permutations + scaling factors: $\mu = (v; \pi)$, with $v \in (\mathbb{F}_q^*)^n$

$$\mu\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

  Monomial matrix: permutation $\times$ diagonal.
- Monomials + field automorphism.

Two codes are equivalent if they are connected by an isometry.

We talk about permutation, linear and semilinear equivalence, respectively.

Can easily be described using representatives, i.e. generator (or parity-check) matrices. Clearly:

Maps which preserve the distances (weights).

- Permutations: $\pi\big((a_1, a_2, \ldots, a_n)\big) = \big(a_{\pi(1)}, a_{\pi(2)}, \ldots, a_{\pi(n)}\big)$.
- Monomials: permutations + scaling factors: $\mu = (v; \pi)$, with $v \in (\mathbb{F}_q^*)^n$

  $$\mu\big((a_1, a_2, \ldots, a_n)\big) = \big(v_1 \cdot a_{\pi(1)}, v_2 \cdot a_{\pi(2)}, \ldots, v_n \cdot a_{\pi(n)}\big)$$

  Monomial matrix: permutation $\times$ diagonal.
- Monomials + field automorphism.

Two codes are equivalent if they are connected by an isometry.

We talk about permutation, linear and semilinear equivalence, respectively.

Can easily be described using representatives, i.e. generator (or parity-check) matrices. Clearly:

$$\mathfrak{C} \overset{\mathsf{PE}}{\sim} \mathfrak{C}' \iff \exists (S, P) \in \mathsf{GL}_k(q) \times \mathsf{S}_n \ \text{ s.t. } \ G' = SGP,$$
$$\mathfrak{C} \overset{\mathsf{LE}}{\sim} \mathfrak{C}' \iff \exists (S, Q) \in \mathsf{GL}_k(q) \times \mathsf{M}_n(q) \ \text{ s.t. } \ G' = SGQ,$$

where P is a permutation matrix, and Q a monomial matrix.

Can be seen as a group action of $\mathcal{G} = \mathsf{GL}_k(q) \times \mathsf{M}_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k \times n}$.

Can be seen as a group action of $\mathcal{G} = \mathrm{GL}_k(q) \times \mathrm{M}_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k\times n}$.

Code-based Group Action

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \to & \mathcal{X} \\ & ((S,Q),G) & \mapsto & SGQ \end{array}$$

Can be seen as a group action of $\mathcal{G} = \mathrm{GL}_k(q) \times \mathrm{M}_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k\times n}$.

Code-based Group Action

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \rightarrow & \mathcal{X} \\ & ((S,Q),G) & \mapsto & SGQ \end{array}$$

$\mathcal{G}$ acts on the entire codes if we choose canonical representation, e.g. systematic form.

Can be seen as a group action of $\mathcal{G} = \mathrm{GL}_k(q) \times \mathrm{M}_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k\times n}$.

Code-based Group Action

$$\begin{aligned} \star : \quad \mathcal{G} \times \mathcal{X} &\rightarrow \mathcal{X} \\ ((S,Q),G) &\mapsto SGQ \end{aligned}$$

$\mathcal{G}$ acts on the entire codes if we choose canonical representation, e.g. systematic form.

Note that here $\mathcal{X}$ is not a group, and the action is also non-commutative, which prevents both Shor's and Kuperberg's algorithms.

Can be seen as a group action of $\mathcal{G} = \mathsf{GL}_k(q) \times \mathsf{M}_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k\times n}$.

Code-based Group Action

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \rightarrow & \mathcal{X} \\ & ((S,Q),G) & \mapsto & SGQ \end{array}$$

$\mathcal{G}$ acts on the entire codes if we choose canonical representation, e.g. systematic form.

Note that here $\mathcal{X}$ is not a group, and the action is also non-commutative, which prevents both Shor's and Kuperberg's algorithms.

The problem of deciding if two codes are equivalent is well-known in coding theory; the computational version is the vectorization problem for our action.

Can be seen as a group action of $\mathcal{G} = \mathrm{GL}_k(q) \times \mathrm{M}_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k\times n}$.

Code-based Group Action

$$\begin{array}{rccc} \star: & \mathcal{G} \times \mathcal{X} & \to & \mathcal{X} \\ & ((S,Q),G) & \mapsto & SGQ \end{array}$$

$\mathcal{G}$ acts on the entire codes if we choose canonical representation, e.g. systematic form.

Note that here $\mathcal{X}$ is not a group, and the action is also non-commutative, which prevents both Shor's and Kuperberg's algorithms.

The problem of deciding if two codes are equivalent is well-known in coding theory; the computational version is the vectorization problem for our action.

Linear Equivalence Problem (LEP)

Given $\mathfrak{C}, \mathfrak{C}' \subseteq \mathbb{F}_q^n$, find μ such that $\mu(\mathfrak{C}) = \mathfrak{C}' \iff$ Given (systematic) generator matrices $G, G' \in \mathbb{F}_q^{k\times n}$, find $Q \in \mathrm{M}_n(q)$ such that $G' = SF(GQ)$.

Can be seen as a group action of $\mathcal{G} = \mathrm{GL}_k(q) \times \mathrm{M}_n(q)$ on full-rank matrices in $\mathbb{F}_q^{k \times n}$.

Code-based Group Action

$$\begin{array}{rccc} \star : & \mathcal{G} \times \mathcal{X} & \rightarrow & \mathcal{X} \\ & ((S, Q), G) & \mapsto & SGQ \end{array}$$

$\mathcal{G}$ acts on the entire codes if we choose canonical representation, e.g. systematic form.

Note that here $\mathcal{X}$ is not a group, and the action is also non-commutative, which prevents both Shor's and Kuperberg's algorithms.

The problem of deciding if two codes are equivalent is well-known in coding theory; the computational version is the vectorization problem for our action.

Linear Equivalence Problem (LEP)

Given $\mathfrak{C}, \mathfrak{C}' \subseteq \mathbb{F}_q^n$, find μ such that $\mu(\mathfrak{C}) = \mathfrak{C}' \iff$ Given (systematic) generator matrices $G, G' \in \mathbb{F}_q^{k \times n}$, find $Q \in \mathrm{M}_n(q)$ such that $G' = SF(GQ)$.

Note that the permutation case (PEP) is just a special case, and for practical applications, we are not interested in the semilinear version of the problem.

The LESS ZKID

A ZK protocol based exclusively on the hardness of the code equivalence problem.

(Biasse, Micheli, P., Santini, 2020)

The LESS ZKID

A ZK protocol based exclusively on the hardness of the code equivalence problem.

(Biasse, Micheli, P., Santini, 2020)

Key Generation

- Input public parameters, hash function $\mathbf{H}$.
- Choose random q-ary code $\mathfrak{C}$, given by generator matrix G.
- sk: monomial matrix Q.
- pk: matrix $G' = SF(GQ)$.

The LESS ZKID

A ZK protocol based exclusively on the hardness of the code equivalence problem.

(Biasse, Micheli, P., Santini, 2020)

Key Generation

- Input public parameters, hash function $\mathbf{H}$.
- Choose random q-ary code $\mathfrak{C}$, given by generator matrix G.
- sk: monomial matrix Q.
- pk: matrix $G' = SF(GQ)$.

Prover		Verifier
Choose random monomial matrix $\tilde{Q} \in \mathrm{M}_n(q)$.		
Compute $\tilde{G} = SF(G\tilde{Q})$.		
Set $cmt = \mathbf{H}(\tilde{G})$	$\xrightarrow{cmt}$	
	$\xleftarrow{b}$	Select random $ch \in \{0, 1\}$.
If $ch = 0$ set $rsp = \tilde{Q}$	$\xrightarrow{rsp}$	Verify $\mathbf{H}(SF(G \cdot rsp)) = cmt$.
If $ch = 1$ set $rsp = Q^{-1}\tilde{Q}$		Verify $\mathbf{H}(SF(G' \cdot rsp)) = cmt$.

It is easy to prove completeness, 2-special soundness and honest-verifier zero-knowledge.

It is easy to prove completeness, 2-special soundness and honest-verifier zero-knowledge.

Before Fiat-Shamir, reduce soundness error $1/2 \implies t = \lambda$ parallel repetitions.

It is easy to prove completeness, 2-special soundness and honest-verifier zero-knowledge.

Before Fiat-Shamir, reduce soundness error $1/2 \implies t = \lambda$ parallel repetitions.

The protocol can be greatly improved with the following modifications:

("LESS-FM", Barenghi, Biasse, P., Santini, 2021)

It is easy to prove completeness, 2-special soundness and honest-verifier zero-knowledge.

Before Fiat-Shamir, reduce soundness error $1/2 \implies t = \lambda$ parallel repetitions.

The protocol can be greatly improved with the following modifications:
("LESS-FM", Barenghi, Biasse, P., Santini, 2021)

- Use multiple public keys and non-binary challenges.

+ Lower soundness error: $1/2 \rightarrow 1/2^{\ell}$.

− Rapid increase in public key size.

It is easy to prove completeness, 2-special soundness and honest-verifier zero-knowledge.

Before Fiat-Shamir, reduce soundness error $1/2 \implies t = \lambda$ parallel repetitions.

The protocol can be greatly improved with the following modifications:
("LESS-FM", Barenghi, Biasse, P., Santini, 2021)

- Use multiple public keys and non-binary challenges.
+ Lower soundness error: $1/2 \to 1/2^\ell$.
– Rapid increase in public key size.

- Use a challenge string with fixed weight ω.
+ Exploits imbalance in cost of response: seed vs monomial.
– Larger number of iterations.

It is easy to prove completeness, 2-special soundness and honest-verifier zero-knowledge.

Before Fiat-Shamir, reduce soundness error $1/2 \implies t = \lambda$ parallel repetitions.

The protocol can be greatly improved with the following modifications:
("LESS-FM", Barenghi, Biasse, P., Santini, 2021)

- Use multiple public keys and non-binary challenges.

+ Lower soundness error: $1/2 \to 1/2^{\ell}$.
− Rapid increase in public key size.

- Use a challenge string with fixed weight ω.

+ Exploits imbalance in cost of response: seed vs monomial.
− Larger number of iterations.

Such modifications do not affect security, only requiring small tweaks in proofs or switching to equivalent security assumptions.

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Design Considerations

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n, k, q, w) > 2^{\lambda}.$$

Design Considerations

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n, k, q, w) > 2^{\lambda}.$$

For example for NIST Category 1 (≈ 128 sec. bits) we have $(n, k, q) = (252, 126, 127)$.

Design Considerations

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n, k, q, w) > 2^{\lambda}.$$

For example for NIST Category 1 (≈ 128 sec. bits) we have $(n, k, q) = (252, 126, 127)$.

Protocol parameters (t, ω, s) infer performance profile:

Design Considerations

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n,k,q,w) > 2^{\lambda}.$$

For example for NIST Category 1 (≈ 128 sec. bits) we have $(n,k,q) = (252, 126, 127)$.

Protocol parameters (t, ω, s) infer performance profile:

- $pk = (s-1)\underbrace{k(n-k)\lceil \log_2(q) \rceil}_{G}/8 + \mathit{seed}$ bytes

Design Considerations

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n, k, q, w) > 2^{\lambda}.$$

For example for NIST Category 1 (≈ 128 sec. bits) we have $(n, k, q) = (252, 126, 127)$.

Protocol parameters (t, ω, s) infer performance profile:

- $pk = (s-1)\underbrace{k(n-k)\lceil \log_2(q) \rceil}_{G}/8 + seed$ bytes
- $sig = \omega \cdot \underbrace{n(\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)}_{iso}/8 + \{seeds, digest, salt\}$ bytes

Design Considerations

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n, k, q, w) > 2^\lambda.$$

For example for NIST Category 1 (≈ 128 sec. bits) we have $(n, k, q) = (252, 126, 127)$.

Protocol parameters (t, ω, s) infer performance profile:

- $pk = (s-1)\underbrace{k(n-k)\lceil \log_2(q) \rceil}_{G}/8 + seed$ bytes
- $sig = \omega \cdot \underbrace{n(\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)}_{iso}/8 + \{seeds, digest, salt\}$ bytes

Runtime is dominated by SF computation, for both Keygen and Sign/Verify.

Design Considerations

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n, k, q, w) > 2^\lambda.$$

For example for NIST Category 1 (≈ 128 sec. bits) we have $(n, k, q) = (252, 126, 127)$.

Protocol parameters (t, ω, s) infer performance profile:

- $pk = (s-1)\underbrace{k(n-k)\lceil\log_2(q)\rceil}_{G}/8 + seed$ bytes
- $sig = \omega \cdot \underbrace{n(\lceil\log_2(n)\rceil + \lceil\log_2(q-1)\rceil)}_{iso}/8 + \{seeds, digest, salt\}$ bytes

Runtime is dominated by SF computation, for both Keygen and Sign/Verify.

The protocol shows a high degree of flexibility, to cater for different priorities.

Design Considerations

2 Signatures from Code Equivalence

Best *generic* LEP solvers (i.e. no weak instances) are combinatorial and reduce to SDP.

(Leon, 1982; Beullens, 2020; Barenghi, Biasse, P., Santini, 2023)

Code parameters chosen using according to this, following conservative criterion. Namely, pick n, k, q so that, for any d and any w:

$$\sqrt{N_d(w)} \cdot C_{\mathsf{ISD}}^{(d)}(n, k, q, w) > 2^{\lambda}.$$

For example for NIST Category 1 (≈ 128 sec. bits) we have $(n, k, q) = (252, 126, 127)$.

Protocol parameters (t, ω, s) infer performance profile:

- $pk = (s-1)\underbrace{k(n-k)\lceil\log_2(q)\rceil}_{G}/8 + seed$ bytes
- $sig = \omega \cdot \underbrace{n(\lceil\log_2(n)\rceil + \lceil\log_2(q-1)\rceil)}_{iso}/8 + \{seeds, digest, salt\}$ bytes

Runtime is dominated by SF computation, for both Keygen and Sign/Verify.

The protocol shows a high degree of flexibility, to cater for different priorities.

Can we compress signatures?

Roadmap

- Background
- Signatures from Code Equivalence
- A New Formulation
- Conclusions

Parsing the Information

3 A New Formulation

Let us consider for simplicity the permutation case.

Parsing the Information
3 A New Formulation

Let us consider for simplicity the permutation case.

Information contained in P:

- which columns are moved to the k leftmost coordinates
- how the k columns on the left are sorted
- how the $n-k$ columns on the right are sorted

Parsing the Information

Let us consider for simplicity the permutation case.

Information contained in P:

- which columns are moved to the k leftmost coordinates
- how the k columns on the left are sorted
- how the $n-k$ columns on the right are sorted

Such information is represented by three permutation matrices:

- $n \times n$ permutation matrix $P_{\mathrm{is}} \in \mathsf{S}_{n,k}$
- $k \times k$ permutation matrix $P_{\mathrm{rows}} \in \mathsf{S}_{k}$
- $(n-k) \times (n-k)$ permutation matrix $P_{\mathrm{cols}} \in \mathsf{S}_{n-k}$

Parsing the Information

Let us consider for simplicity the permutation case.

Information contained in P:

- which columns are moved to the k leftmost coordinates
- how the k columns on the left are sorted
- how the $n-k$ columns on the right are sorted

Such information is represented by three permutation matrices:

- $n \times n$ permutation matrix $P_{\text{is}} \in \mathsf{S}_{n,k}$
- $k \times k$ permutation matrix $P_{\text{rows}} \in \mathsf{S}_k$
- $(n-k) \times (n-k)$ permutation matrix $P_{\text{cols}} \in \mathsf{S}_{n-k}$

In particular, for any P:

$$P = P_{\text{is}} \cdot \begin{pmatrix} P_{\text{rows}}^{-1} & 0 \\ 0 & P_{\text{cols}} \end{pmatrix}$$

Splitting the Action

Let $J :=$ set of coordinates that are moved in first k positions; then

$$G \cdot P_{\text{is}} = \Big(\underbrace{G_J}_{k \text{ columns}} , \underbrace{G_{\{1,\cdots,n\} \setminus J}}_{n-k \text{ columns}} \Big).$$

Splitting the Action

Let $J :=$ set of coordinates that are moved in first k positions; then

$$G \cdot P_{\mathrm{is}} = (\underbrace{G_J}_{k \text{ columns}}, \underbrace{G_{\{1,\cdots,n\}\setminus J}}_{n-k \text{ columns}}).$$

Applying P we get:

$$\begin{aligned} G \cdot P &= G \cdot P_{\mathrm{is}} \cdot \begin{pmatrix} P_{\mathrm{rows}}^{-1} & 0 \\ 0 & P_{\mathrm{cols}} \end{pmatrix} \\ &= (G_J, G_{\{1,\cdots,n\}\setminus J}) \cdot \begin{pmatrix} P_{\mathrm{rows}}^{-1} & 0 \\ 0 & P_{\mathrm{cols}} \end{pmatrix} = (G_J \cdot P_{\mathrm{rows}}^{-1}, G_{\{1,\cdots,n\}\setminus J} \cdot P_{\mathrm{cols}}). \end{aligned}$$

Splitting the Action

Let $J :=$ set of coordinates that are moved in first k positions; then

$$G \cdot P_{\text{is}} = (\underbrace{G_J}_{k \text{ columns}}, \underbrace{G_{\{1,\cdots,n\}\setminus J}}_{n-k \text{ columns}}).$$

Applying P we get:

$$\begin{aligned} G \cdot P &= G \cdot P_{\text{is}} \cdot \begin{pmatrix} P_{\text{rows}}^{-1} & 0 \\ 0 & P_{\text{cols}} \end{pmatrix} \\ &= (G_J, G_{\{1,\cdots,n\}\setminus J}) \cdot \begin{pmatrix} P_{\text{rows}}^{-1} & 0 \\ 0 & P_{\text{cols}} \end{pmatrix} = (G_J \cdot P_{\text{rows}}^{-1}, G_{\{1,\cdots,n\}\setminus J} \cdot P_{\text{cols}}). \end{aligned}$$

Then, for any $S \in \mathrm{GL}_k(q)$:

$$\begin{aligned} \mathsf{SF}(SGP) &= \mathsf{SF}\big((S \cdot G_J \cdot P_{\text{rows}}^{-1}, S \cdot G_{\{1,\cdots,n\}\setminus J} \cdot P_{\text{cols}})\big) \\ &= \big(I_k, (S \cdot G_J \cdot P_{\text{rows}}^{-1})^{-1} \cdot S \cdot G_{\{1,\cdots,n\}\setminus J} \cdot P_{\text{cols}}\big) \\ &= \big(I_k, P_{\text{rows}} \cdot G_J^{-1} \cdot G_{\{1,\cdots,n\}\setminus J} \cdot P_{\text{cols}}\big). \end{aligned}$$

Let $\tilde{G} = \mathsf{SF}(G \cdot \tilde{P})$ sent during commitment and $\tilde{P}$ decomposed as before; then

$$\begin{aligned}\tilde{G} &= \left(I_k\ ,\ \tilde{P}_{\text{rows}} \cdot G_J^{-1} \cdot G_{\{1,\cdots,n\}\setminus J} \cdot \tilde{P}_{\text{cols}}\right)\\ &= \left(I_k\ ,\ A\right).\end{aligned}$$

Splitting the Action (cont.)

Let $\tilde{G} = \mathsf{SF}(G \cdot \tilde{P})$ sent during commitment and $\tilde{P}$ decomposed as before; then

$$\begin{aligned}\tilde{G} &= \left(I_k \ , \ \tilde{P}_{\text{rows}} \cdot G_J^{-1} \cdot G_{\{1,\cdots,n\}\setminus J} \cdot \tilde{P}_{\text{cols}}\right) \\ &= \left(I_k \ , \ A\right).\end{aligned}$$

Consider $P^* = \tilde{P}_{\text{is}} \cdot \begin{pmatrix} \tilde{P}_{\text{rows}}^{-1} & 0 \\ 0 & I_{n-k} \end{pmatrix}$; then

$$\begin{aligned}\mathsf{SF}(G \cdot P^*) &= \left(I_k \ , \ \tilde{P}_{\text{rows}} \cdot G_J^{-1} \cdot G_{\{1,\cdots,n\}\setminus J}\right) \\ &= \left(I_k \ , \ A \cdot \tilde{P}_{\text{cols}}^{-1}\right).\end{aligned}$$

Let $\tilde{G} = \mathsf{SF}(G \cdot \tilde{P})$ sent during commitment and $\tilde{P}$ decomposed as before; then

$$\begin{aligned}\tilde{G} &= \left(I_k\ ,\ \tilde{P}_{\text{rows}} \cdot G_J^{-1} \cdot G_{\{1,\cdots,n\}\setminus J} \cdot \tilde{P}_{\text{cols}}\right)\\ &= \left(I_k\ ,\ A\right).\end{aligned}$$

Consider $P^* = \tilde{P}_{\text{is}} \cdot \begin{pmatrix} \tilde{P}_{\text{rows}}^{-1} & 0 \\ 0 & I_{n-k} \end{pmatrix}$; then

$$\begin{aligned}\mathsf{SF}(G \cdot P^*) &= \left(I_k\ ,\ \tilde{P}_{\text{rows}} \cdot G_J^{-1} \cdot G_{\{1,\cdots,n\}\setminus J}\right)\\ &= \left(I_k\ ,\ A \cdot \tilde{P}_{\text{cols}}^{-1}\right).\end{aligned}$$

Thus, we obtain an invariant up to a column permutation.

We can modify the Commit procedure:

We can modify the Commit procedure:

1. Agree on an ordering (e.g. lexicographic).

A New Notion of Equivalence

We can modify the Commit procedure:

1. Agree on an ordering (e.g. lexicographic).
2. Respond to challenge using P^*.

A New Notion of Equivalence

3 A New Formulation

We can modify the Commit procedure:

1. Agree on an ordering (e.g. lexicographic).
2. Respond to challenge using P^*.

Transmitting P^* only takes $k\Big(\lceil \log_2(n) \rceil\Big) \implies \approx 1/2$ space saving.

A New Notion of Equivalence

We can modify the Commit procedure:

1. Agree on an ordering (e.g. lexicographic).
2. Respond to challenge using P^*.

Transmitting P^* only takes $k\left(\lceil \log_2(n) \rceil\right) \implies \approx 1/2$ space saving.

Extends naturally to linear equivalence: remaining coordinates identical up to a monomial.

We can modify the Commit procedure:

1. Agree on an ordering (e.g. lexicographic).
2. Respond to challenge using P^*.

Transmitting P^* only takes $k\left(\lceil \log_2(n) \rceil\right) \implies \approx 1/2$ space saving.

Extends naturally to linear equivalence: remaining coordinates identical up to a monomial.

Information Set Linear Equivalence Problem (IS-LEP)

Given $\mathfrak{C}, \mathfrak{C}' \subseteq \mathbb{F}_q^n$, find monomials μ, ζ and an information set J' such that for every $c \in \widetilde{\mathfrak{C}} = \mu(\mathfrak{C})$ there exists $c' \in \mathfrak{C}'$ with $\widetilde{c}_{J'} = c'_{J'}$ and $\widetilde{c}_{\{1,\cdots,n\}\setminus J'} = \zeta\left(c'_{\{1,\cdots,n\}\setminus J'}\right)$. Equivalently, given generators $\widetilde{G}, G' \in \mathbb{F}_q^{k\times n}$, it must be that

$$\widetilde{G}_{J'}^{-1}\widetilde{G}_{\{1,\cdots,n\}\setminus J'} = \zeta\left(G'^{-1}_{J'}G'_{\{1,\cdots,n\}\setminus J'}\right).$$

A New Notion of Equivalence

We can modify the Commit procedure:

1. Agree on an ordering (e.g. lexicographic).
2. Respond to challenge using P^*.

Transmitting P^* only takes $k\Big(\lceil \log_2(n) \rceil\Big) \implies \approx 1/2$ space saving.

Extends naturally to linear equivalence: remaining coordinates identical up to a monomial.

Information Set Linear Equivalence Problem (IS-LEP)

Given $\mathfrak{C}, \mathfrak{C}' \subseteq \mathbb{F}_q^n$, find monomials μ, ζ and an information set J' such that for every $c \in \widetilde{\mathfrak{C}} = \mu(\mathfrak{C})$ there exists $c' \in \mathfrak{C}'$ with $\widetilde{c}_{J'} = c'_{J'}$ and $\widetilde{c}_{\{1,\cdots,n\}\setminus J'} = \zeta\left(c'_{\{1,\cdots,n\}\setminus J'}\right)$.
Equivalently, given generators $\widetilde{G}, G' \in \mathbb{F}_q^{k\times n}$, it must be that

$$\widetilde{G}_{J'}^{-1}\widetilde{G}_{\{1,\cdots,n\}\setminus J'} = \zeta\left(G'^{-1}_{J'}G'_{\{1,\cdots,n\}\setminus J'}\right).$$

We prove that this is equivalent to LEP (reduction in both ways).

Roadmap

- Background
- Signatures from Code Equivalence
- A New Formulation
- Conclusions

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

- Ring signatures.

  (Barenghi, Biasse, Ngo, P., Santini, 2022)

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

- Ring signatures.

  (Barenghi, Biasse, Ngo, P., Santini, 2022)

- Threshold signatures.

  (Battagliola, Borin, Meneghetti, P., preprint)

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

- Ring signatures.
  (Barenghi, Biasse, Ngo, P., Santini, 2022)
- Threshold signatures.
  (Battagliola, Borin, Meneghetti, P., preprint)
- Blind signatures.
  (Kuchta, LeGrow, P., preprint)

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

- Ring signatures.
  (Barenghi, Biasse, Ngo, P., Santini, 2022)
- Threshold signatures.
  (Battagliola, Borin, Meneghetti, P., preprint)
- Blind signatures.
  (Kuchta, LeGrow, P., preprint)
- ...

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

- Ring signatures.
  (Barenghi, Biasse, Ngo, P., Santini, 2022)
- Threshold signatures.
  (Battagliola, Borin, Meneghetti, P., preprint)
- Blind signatures.
  (Kuchta, LeGrow, P., preprint)
- ...

Our work is able to reduce signature size by half, compared to LESS-FM.

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

- Ring signatures.
  (Barenghi, Biasse, Ngo, P., Santini, 2022)
- Threshold signatures.
  (Battagliola, Borin, Meneghetti, P., preprint)
- Blind signatures.
  (Kuchta, LeGrow, P., preprint)
- ...

Our work is able to reduce signature size by half, compared to LESS-FM.

Current work: extend this result to generic notion of canonical forms, further compress to $\approx 1/3$ of reported sizes.

Conclusions, Current and Future Work

The introduction of the LESS scheme opened the way to a new, interesting approach for designing code-based cryptographic schemes.

The group action structure is particularly suitable to develop protocols with advanced functionalities, e.g.:

- Ring signatures.
  (Barenghi, Biasse, Ngo, P., Santini, 2022)
- Threshold signatures.
  (Battagliola, Borin, Meneghetti, P., preprint)
- Blind signatures.
  (Kuchta, LeGrow, P., preprint)
- ...

Our work is able to reduce signature size by half, compared to LESS-FM.

Current work: extend this result to generic notion of canonical forms, further compress to $\approx 1/3$ of reported sizes.

Future work includes more performance improvements (e.g. Gaussian elimination, pk size), implementation (e.g. AVX2, hardware) and other applications.

Thank you for listening!
Any questions?

$<$

https://www.less-project.com

References

E. Berlekamp, R. McEliece, and H. Van Tilborg
On the inherent intractability of certain coding problems.
IEEE Transactions on Information Theory 24.3, 1978.

S. Barg
Some new NP-complete coding problems.
Problemy Peredachi Informatsii, 1994.

N. Courtois, M. Finiasz and N. Sendrier
How to Achieve a McEliece-Based Digital Signature Scheme.
ASIACRYPT 2001.

G. Kabatianskii, E. Krouk and B. Smeets
A digital signature scheme based on random error-correcting codes.
Crytography and Coding: 6th IMA International Conference, 1997.

J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini
LESS is More: Code-Based Signatures Without Syndromes.
AFRICACRYPT 2020.

A. Barenghi, J.-F. Biasse, E. Persichetti, and P. Santini
LESS-FM: Fine-Tuning Signatures from the Code Equivalence Problem.
PQCRYPTO 2021.

References

J. Leon
Computing automorphism groups of error-correcting codes.
IEEE Transactions on Information Theory, 28(3):496–511, 1982.

W. Beullens
Not Enough LESS: An Improved Algorithm for Solving Code Equivalence Problems over $\mathbb{F}_q$.
SAC 2020.

A. Barenghi, J.-F. Biasse, E. Persichetti, and P. Santini
On the Computational Hardness of the Code Equivalence Problem in Cryptography.
Advances in Mathematics of Communications, 17(1):23–55, 2023.

A. Barenghi, J.-F. Biasse, T. Ngo, E. Persichetti, and P. Santini
Advanced Signature Functionalities from the Code Equivalence Problem.
International Journal of Computer Mathematics: Computer Systems Theory, 2022.

M. Battagliola, G. Borin, A. Meneghetti and E. Persichetti
Cutting the GRASS: Threshold GRoup Action Signature Schemes.
preprint, available at https://eprint.iacr.org/2023/859.

V. Kuchta, J. LeGrow, and E. Persichetti
Code-Based Blind Signatures.
preprint, to appear.

T. Chou, E. Persichetti, and P. Santini
On Linear Equivalence, Canonical Forms, and Digital Signatures.
preprint, available at https://eprint.iacr.org/2023/1533.