LESS is More: Code-Based Signatures without Syndromes

J.-F. Biasse, G. Micheli, E. Persichetti and P. Santini

20 July 2020

Charles E. Schmidt College of Science
Florida Atlantic University

Traditional Code-based Approach

McEliece: first cryptosystem using error correcting codes (1978).

TRADITIONAL CODE-BASED APPROACH

McEliece: first cryptosystem using error correcting codes (1978).

Based on the hardness of decoding random linear codes.

McEliece: first cryptosystem using error correcting codes (1978).

Based on the hardness of decoding random linear codes.

Important that the chosen code is indistinguishable from random.

TRADITIONAL CODE-BASED APPROACH

McEliece: first cryptosystem using error correcting codes (1978).

Based on the hardness of decoding random linear codes.

Important that the chosen code is indistinguishable from random.

$\rightarrow$ the Code Equivalence Problem.

Code Equivalence Notions

Permutation Code Equivalence

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *permutationally equivalent*, or $\mathfrak{C} \stackrel{\mathsf{PE}}{\sim} \mathfrak{C}'$, if there is a permutation $\pi \in S_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}$, i.e.

$$\mathfrak{C}' = \{\pi(x), \;\; x \in \mathfrak{C}\}.$$

Code Equivalence Notions

Permutation Code Equivalence

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *permutationally equivalent*, or $\mathfrak{C} \overset{\mathsf{PE}}{\sim} \mathfrak{C}'$, if there is a permutation $\pi \in S_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}$, i.e.

$$\mathfrak{C}' = \{\pi(x), \;\; x \in \mathfrak{C}\}.$$

This notion can be extended using linear isometries.

Code Equivalence Notions

Permutation Code Equivalence

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *permutationally equivalent*, or $\mathfrak{C} \overset{\mathsf{PE}}{\sim} \mathfrak{C}'$, if there is a permutation $\pi \in S_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}$, i.e.

$$\mathfrak{C}' = \{\pi(x), \;\; x \in \mathfrak{C}\}.$$

This notion can be extended using linear isometries.

Linear Code Equivalence

Two codes $\mathfrak{C}$ and $\mathfrak{C}'$ are *linearly equivalent*, or $\mathfrak{C} \overset{\mathsf{LE}}{\sim} \mathfrak{C}'$, if there is a linear isometry $\mu = (v, \pi) \in {\mathbb{F}_q^*}^n \rtimes S_n$ such that $\mathfrak{C}' = \mu(\mathfrak{C})$, i.e.

$$\mathfrak{C}' = \{\mu(x), \;\; x \in \mathfrak{C}\}.$$

Code equivalence can be described using generator matrices.
Clearly:

Code equivalence can be described using generator matrices. Clearly:

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

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

The Code Equivalence Problem

Code equivalence can be described using generator matrices. Clearly:

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

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

Permutation (Linear) Code Equivalence Problem

Let $\mathfrak{C}$ and $\mathfrak{C}'$ be two $[n, k]$ linear codes over $\mathbb{F}_q$, having generator matrices G and G', respectively. Determine whether the two codes are permutationally (linearly) equivalent, i.e. if there exist matrices $S \in \mathrm{GL}$ and $P \in S_n$ ($Q \in M_n(q)$) such that $G' = SGP$ ($G' = SGQ$).

Hardness at a Glance

Studied for a very long time.

HARDNESS AT A GLANCE

Studied for a very long time.

Unlikely to be NP-complete (unless polynomial hierarchy collapses).
(Petrank and Roth, 1997)

HARDNESS AT A GLANCE

Studied for a very long time.

Unlikely to be NP-complete (unless polynomial hierarchy collapses).
(Petrank and Roth, 1997)

Existing algorithms efficiently attack particular cases, however...

Hardness at a Glance

Studied for a very long time.

Unlikely to be NP-complete (unless polynomial hierarchy collapses).
(Petrank and Roth, 1997)

Existing algorithms efficiently attack particular cases, however...

...underlying exponential complexity makes it easy to find intractable instances.

Could Code Equivalence be used as a stand-alone problem?

Could Code Equivalence be used as a stand-alone problem?

The situation for linear isometries recalls that of DLP (although without commutativity).

Applications in Cryptography

Could Code Equivalence be used as a stand-alone problem?

The situation for linear isometries recalls that of DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence, with evident computational advantages.

Applications in Cryptography

Could Code Equivalence be used as a stand-alone problem?

The situation for linear isometries recalls that of DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence, with evident computational advantages.

In this work, we construct a ZK protocol based exclusively on the hardness of the linear code equivalence problem.

APPLICATIONS IN CRYPTOGRAPHY

Could Code Equivalence be used as a stand-alone problem?

The situation for linear isometries recalls that of DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence, with evident computational advantages.

In this work, we construct a ZK protocol based exclusively on the hardness of the linear code equivalence problem.

This can be then transformed into a full-fledged signature scheme via Fiat-Shamir.

Applications in Cryptography

Could Code Equivalence be used as a stand-alone problem?

The situation for linear isometries recalls that of DLP (although without commutativity).

This means several existing constructions could be adapted to be based on Code Equivalence, with evident computational advantages.

In this work, we construct a ZK protocol based exclusively on the hardness of the linear code equivalence problem.

This can be then transformed into a full-fledged signature scheme via Fiat-Shamir.

Since the scheme does not rely on decoding hardness, very small codes can be employed, leading to very practical instances.

LESS Identification Scheme

Key Generation

- Choose linear code $\mathfrak{C}$ with generator matrix G.
- SK: invertible matrix S and monomial matrix Q.
- PK: matrix $G' = SGQ$.

Prover's Computation

- Choose random monomial matrix $\tilde{Q}$
- Set $\tilde{G} = G\tilde{Q}$ and $h = Hash(SystForm(\tilde{G}))$.

  (After receiving challenge bit b)
- If $b = 0$ respond with $\mu = \tilde{Q}$.
- If $b = 1$ respond with $\mu = Q^{-1}\tilde{Q}$.

Verifier's Computation

- If $b = 0$ verify that $Hash(SystForm(G\mu)) = h$.
- If $b = 1$ verify that $Hash(SystForm(G'\mu)) = h$.

Security Requirements

The three main security aspects of a zero-knowledge identification scheme are easily proved (sketch below).

The three main security aspects of a zero-knowledge identification scheme are easily proved (sketch below).

- Completeness: this is immediate, and it is possible thanks to the use of the systematic form.

The three main security aspects of a zero-knowledge identification scheme are easily proved (sketch below).

- Completeness: this is immediate, and it is possible thanks to the use of the systematic form.
- Zero-Knowledge: the produced responses do not leak information about the private key. In fact, in both cases, the response is distributed uniformly at random over the set of all monomial matrices.

SECURITY REQUIREMENTS

The three main security aspects of a zero-knowledge identification scheme are easily proved (sketch below).

- Completeness: this is immediate, and it is possible thanks to the use of the systematic form.
- Zero-Knowledge: the produced responses do not leak information about the private key. In fact, in both cases, the response is distributed uniformly at random over the set of all monomial matrices.
- Soundness: the protocol is 2-special sound (cheating probability 1/2). In fact, an extractor algorithm that finds a witness, would need to either be able to find a collision for the hash function, or solve an instance of the linear equivalence problem.

Introduced in 1982 as a method to find the automorphism group of a code.

Attacks: Leon's Algorithm

Introduced in 1982 as a method to find the automorphism group of a code.

Can be adapted to solve Permutation Equivalence by analyzing the action of the permutation on a subset of fixed-weight codewords.

Attacks: Leon's Algorithm

Introduced in 1982 as a method to find the automorphism group of a code.

Can be adapted to solve Permutation Equivalence by analyzing the action of the permutation on a subset of fixed-weight codewords.

Weight, say ω, is usually set $\geq$ GV bound. This is likely the best choice (big enough but not too big).

Attacks: Leon's Algorithm

Introduced in 1982 as a method to find the automorphism group of a code.

Can be adapted to solve Permutation Equivalence by analyzing the action of the permutation on a subset of fixed-weight codewords.

Weight, say ω, is usually set $\geq$ GV bound. This is likely the best choice (big enough but not too big).

Bottleneck: it requires enumerating the codewords of weight ω.

Attacks: Leon's Algorithm

Introduced in 1982 as a method to find the automorphism group of a code.

Can be adapted to solve Permutation Equivalence by analyzing the action of the permutation on a subset of fixed-weight codewords.

Weight, say ω, is usually set $\geq$ GV bound. This is likely the best choice (big enough but not too big).

Bottleneck: it requires enumerating the codewords of weight ω.

Complexity can be upper-bounded as:

$$O\left(4(n-k)\sum_{\delta=1}^{\omega}(\delta-1)\binom{k}{\delta}(q-1)^{\delta-1}\right).$$

Attacks: Leon's Algorithm

Introduced in 1982 as a method to find the automorphism group of a code.

Can be adapted to solve Permutation Equivalence by analyzing the action of the permutation on a subset of fixed-weight codewords.

Weight, say ω, is usually set $\geq$ GV bound. This is likely the best choice (big enough but not too big).

Bottleneck: it requires enumerating the codewords of weight ω.

Complexity can be upper-bounded as:

$$O\left(4(n-k)\sum_{\delta=1}^{\omega}(\delta-1)\binom{k}{\delta}(q-1)^{\delta-1}\right).$$

Only efficient for codes of small dimension over small finite fields.

Introduced by Sendrier in 2000 as a dedicated algorithm for Permutation Equivalence, uses the following concept.

Introduced by Sendrier in 2000 as a dedicated algorithm for Permutation Equivalence, uses the following concept.

Signature Function

Let $\mathfrak{C}$ be a linear code of length n; we say that a function S is a signature function over a set F if it maps $\mathfrak{C}$ and a position $i \in [0; n-1]$ to F and is such that

$$\mathsf{S}(\mathfrak{C}, i) = \mathsf{S}\big(\pi(\mathfrak{C}), \pi(i)\big), \quad \forall \pi \in S_n.$$

A signature function is fully discriminant if $\mathsf{S}(\mathfrak{C}, i) \neq \mathsf{S}(\mathfrak{C}, j), \forall i \neq j$.

ATTACKS: SUPPORT SPLITTING ALGORITHM

Introduced by Sendrier in 2000 as a dedicated algorithm for Permutation Equivalence, uses the following concept.

SIGNATURE FUNCTION

Let $\mathfrak{C}$ be a linear code of length n; we say that a function S is a signature function over a set F if it maps $\mathfrak{C}$ and a position $i \in [0; n-1]$ to F and is such that

$$\mathsf{S}(\mathfrak{C}, i) = \mathsf{S}\big(\pi(\mathfrak{C}), \pi(i)\big), \quad \forall \pi \in S_n.$$

A signature function is fully discriminant if $\mathsf{S}(\mathfrak{C}, i) \neq \mathsf{S}(\mathfrak{C}, j), \forall i \neq j$.

Then clearly $\mathsf{S}(\mathfrak{C}, i) = \mathsf{S}(\mathfrak{C}', j) \iff j = \pi(i)$, which allows to reconstruct the permutation.

Complexity of SSA

Finding a fully discriminant signature is not obvious.

Complexity of SSA

Finding a fully discriminant signature is not obvious.

Sendrier proposes to build them from the hull of the code, i.e. $\mathfrak{C} \cap \mathfrak{C}^{\perp}$ (via puncturing and computing the weight enumerator).

Complexity of SSA

Finding a fully discriminant signature is not obvious.

Sendrier proposes to build them from the hull of the code, i.e. $\mathfrak{C} \cap \mathfrak{C}^\perp$ (via puncturing and computing the weight enumerator).

Complexity scales accordingly, and it is given by:

$$O\big(n^3 + n^2 q^{d_{hull}} \log n\big)$$

Complexity of SSA

Finding a fully discriminant signature is not obvious.

Sendrier proposes to build them from the hull of the code, i.e. $\mathfrak{C} \cap \mathfrak{C}^\perp$ (via puncturing and computing the weight enumerator).

Complexity scales accordingly, and it is given by:

$$O\left(n^3 + n^2 q^{d_{hull}} \log n\right)$$

Algorithm is efficient when hull is small - but not trivial (empty).

(Bardet, Otmani and Saeed-Taha, 2019)

COMPLEXITY OF SSA

Finding a fully discriminant signature is not obvious.

Sendrier proposes to build them from the hull of the code, i.e. $\mathfrak{C} \cap \mathfrak{C}^{\perp}$ (via puncturing and computing the weight enumerator).

Complexity scales accordingly, and it is given by:

$$O\left(n^3 + n^2 q^{d_{hull}} \log n\right)$$

Algorithm is efficient when hull is small - but not trivial (empty).
(Bardet, Otmani and Saeed-Taha, 2019)

Worst-case: weakly self-dual codes ($\mathfrak{C} \subseteq \mathfrak{C}^{\perp}$).

Solving Linear Equivalence

Both algorithms can be extended to work on the Linear Equivalence version, using *closures*.

Solving Linear Equivalence

Both algorithms can be extended to work on the Linear Equivalence version, using *closures*.

Closure of a Code

Let $\mathbb{F}_q = \{a_0 = 0, a_1, \cdots, a_{q-1}\}$, and $a = (a_1, \cdots, a_{q-1})$. We define the *closure* of a linear code $\mathfrak{C}$, defined over $\mathbb{F}_q$, as the $[n(q-1), k]$ linear code

$$\tilde{\mathfrak{C}} = \{c \otimes a, \;\; c \in \mathfrak{C}\}.$$

Both algorithms can be extended to work on the Linear Equivalence version, using *closures*.

CLOSURE OF A CODE

Let $\mathbb{F}_q = \{a_0 = 0, a_1, \cdots, a_{q-1}\}$, and $a = (a_1, \cdots, a_{q-1})$. We define the *closure* of a linear code $\mathfrak{C}$, defined over $\mathbb{F}_q$, as the $[n(q-1), k]$ linear code

$$\tilde{\mathfrak{C}} = \{c \otimes a, \;\; c \in \mathfrak{C}\}.$$

THEOREM 1

Let $\mathfrak{C}, \mathfrak{C} \subseteq \mathbb{F}_q^n$*; then,* $\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}'$ *if and only if* $\tilde{\mathfrak{C}} \overset{\text{PE}}{\sim} \tilde{\mathfrak{C}}'$.

Solving Linear Equivalence

Both algorithms can be extended to work on the Linear Equivalence version, using *closures*.

Closure of a Code

Let $\mathbb{F}_q = \{a_0 = 0, a_1, \cdots, a_{q-1}\}$, and $a = (a_1, \cdots, a_{q-1})$. We define the *closure* of a linear code $\mathfrak{C}$, defined over $\mathbb{F}_q$, as the $[n(q-1), k]$ linear code

$$\tilde{\mathfrak{C}} = \{c \otimes a, \;\; c \in \mathfrak{C}\}.$$

Theorem 1

Let $\mathfrak{C}, \mathfrak{C} \subseteq \mathbb{F}_q^n$; then, $\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}'$ if and only if $\tilde{\mathfrak{C}} \overset{\text{PE}}{\sim} \tilde{\mathfrak{C}}'$.

Leon's algorithm needs to enumerate fixed-weight codewords in the closure.

Both algorithms can be extended to work on the Linear Equivalence version, using *closures*.

CLOSURE OF A CODE

Let $\mathbb{F}_q = \{a_0 = 0, a_1, \cdots, a_{q-1}\}$, and $a = (a_1, \cdots, a_{q-1})$. We define the *closure* of a linear code $\mathfrak{C}$, defined over $\mathbb{F}_q$, as the $[n(q-1), k]$ linear code

$$\tilde{\mathfrak{C}} = \{c \otimes a, \;\; c \in \mathfrak{C}\}.$$

THEOREM 1

Let $\mathfrak{C}, \mathfrak{C} \subseteq \mathbb{F}_q^n$*; then,* $\mathfrak{C} \overset{\text{LE}}{\sim} \mathfrak{C}'$ *if and only if* $\tilde{\mathfrak{C}} \overset{\text{PE}}{\sim} \tilde{\mathfrak{C}}'$.

Leon's algorithm needs to enumerate fixed-weight codewords in the closure.

SSA applies directly to the closure; however, when $q \geq 5$, this is always weakly self-dual.

Grover's Algorithm

We can expect that a Grover search would provide the usual speedup to Leon's algorithm.

We can expect that a Grover search would provide the usual speedup to Leon's algorithm.

However, a Grover search over all possible secrets (i.e. $P \in S_n$) would not outperform the classical SSA, because of the size of S_n.

Grover's Algorithm

We can expect that a Grover search would provide the usual speedup to Leon's algorithm.

However, a Grover search over all possible secrets (i.e. $P \in S_n$) would not outperform the classical SSA, because of the size of S_n.

Alternatively, could use Grover's *within* SSA.

Grover's Algorithm

We can expect that a Grover search would provide the usual speedup to Leon's algorithm.

However, a Grover search over all possible secrets (i.e. $P \in S_n$) would not outperform the classical SSA, because of the size of S_n.

Alternatively, could use Grover's *within* SSA.

Searching for $j = \pi(i)$ corresponds to $f(j) = 1$ for

$$f(j) = \begin{cases} 1 & \text{if } \mathsf{S}(\mathfrak{C}', j) = \mathsf{S}(\mathfrak{C}, i) \\ 0 & \text{otherwise} \end{cases}$$

We can expect that a Grover search would provide the usual speedup to Leon's algorithm.

However, a Grover search over all possible secrets (i.e. $P \in S_n$) would not outperform the classical SSA, because of the size of S_n.

Alternatively, could use Grover's *within* SSA.

Searching for $j = \pi(i)$ corresponds to $f(j) = 1$ for

$$f(j) = \begin{cases} 1 & \text{if } \mathsf{S}(\mathfrak{C}', j) = \mathsf{S}(\mathfrak{C}, i) \\ 0 & \text{otherwise} \end{cases}$$

Due to the short search space and expensive oracle, we have a total cost of

$$\tilde{O}(n^{5/2} q^{d_{\mathrm{Hull}}} \log n).$$

We can expect that a Grover search would provide the usual speedup to Leon's algorithm.

However, a Grover search over all possible secrets (i.e. $P \in S_n$) would not outperform the classical SSA, because of the size of S_n.

Alternatively, could use Grover's *within* SSA.

Searching for $j = \pi(i)$ corresponds to $f(j) = 1$ for

$$f(j) = \begin{cases} 1 & \text{if } \mathsf{S}(\mathfrak{C}', j) = \mathsf{S}(\mathfrak{C}, i) \\ 0 & \text{otherwise} \end{cases}$$

Due to the short search space and expensive oracle, we have a total cost of

$$\tilde{O}(n^{5/2} q^{d_{\mathrm{Hull}}} \log n).$$

Once again, this does not outperform the classical SSA.

Quantum Fourier Sampling

Search for a secret subgroup H within a known "control group" G.

Search for a secret subgroup H within a known "control group" G.

In our case, we have $G = (\mathrm{GL}_k(2) \times S_n) \rtimes \mathbb{Z}_2$.
(Dinh, Moore and Russell, 2011)

Search for a secret subgroup H within a known "control group" G.

In our case, we have $G = (\mathrm{GL}_k(2) \times S_n) \rtimes \mathbb{Z}_2$.
(Dinh, Moore and Russell, 2011)

In some cases, this leads to an upper bound on the sampling probability.

Quantum Fourier Sampling

Search for a secret subgroup H within a known "control group" G.

In our case, we have $G = (\mathrm{GL}_k(2) \times S_n) \rtimes \mathbb{Z}_2$.
(Dinh, Moore and Russell, 2011)

In some cases, this leads to an upper bound on the sampling probability.

This does not necessarily imply any form of hardness.

PARAMETERS

We consider here simple repetition of the protocol, over 128 rounds, without optimizations relative to signature scheme.

PARAMETERS

We consider here simple repetition of the protocol, over 128 rounds, without optimizations relative to signature scheme.

LESS parameters for 128 bits of security.

n	k	q	Type	PK Size (bits)	Signature Size (Kb)
54	27	53	MONO	8,748	12.43
106	45	7	MONO	14,310	19.02
60	25	31	PERM	7,500	7.82

We consider here simple repetition of the protocol, over 128 rounds, without optimizations relative to signature scheme.

LESS parameters for 128 bits of security.

n	k	q	Type	PK Size (bits)	Signature Size (Kb)
54	27	53	MONO	8,748	12.43
106	45	7	MONO	14,310	19.02
60	25	31	PERM	7,500	7.82

The third parameter sets uses permutations instead of monomials, and therefore employs weakly self-dual codes.

CONCLUSIONS

We have presented a new innovative method for designing code-based primitives, based on the Code Equivalence problem as a standalone security assumption.

Conclusions

We have presented a new innovative method for designing code-based primitives, based on the Code Equivalence problem as a standalone security assumption.

We designed LESS, a zero-knowledge identification scheme, that can be converted to a signature scheme by standard means.

CONCLUSIONS

We have presented a new innovative method for designing code-based primitives, based on the Code Equivalence problem as a standalone security assumption.

We designed LESS, a zero-knowledge identification scheme, that can be converted to a signature scheme by standard means.

Our design performs better than previous protocols based on identification schemes (e.g. Stern, Veron, etc.) and compares well with other code-based signature schemes (e.g. Wave, Durandal).

CONCLUSIONS

We have presented a new innovative method for designing code-based primitives, based on the Code Equivalence problem as a standalone security assumption.

We designed LESS, a zero-knowledge identification scheme, that can be converted to a signature scheme by standard means.

Our design performs better than previous protocols based on identification schemes (e.g. Stern, Veron, etc.) and compares well with other code-based signature schemes (e.g. Wave, Durandal).

We expect excellent performance from a computational point of view, due to the simplicity of the underlying arithmetic (no decoding).

Thank you