Cryptanalysis of the Code Equivalence Problem

Edoardo Persichetti

24 February 2020

Charles E. Schmidt College of Science
Florida Atlantic University

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

Code-based PKE Schemes

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

Based on the hardness of decoding random linear codes.

Code-based PKE Schemes

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.

CODE-BASED PKE SCHEMES

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 $\mathfrak{C} \overset{\text{PE}}{\sim} \mathfrak{C}'$, 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:

The Code Equivalence Problem

Code equivalence can be described using generator matrices. Clearly:

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

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

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?

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).

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.

For example, a ZK protocol can be obtained, which can be then transformed into a 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.

For example, a ZK protocol can be obtained, which can be then transformed into a signature scheme via Fiat-Shamir.

It may also be possible to obtain signatures by following El Gamal's framework, leading to even more efficient instantiations.

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.

For example, a ZK protocol can be obtained, which can be then transformed into a signature scheme via Fiat-Shamir.

It may also be possible to obtain signatures by following El Gamal's framework, leading to even more efficient instantiations.

It could also be possible to devise a Diffie-Hellman-like non-interactive key exchange.

Leon's Algorithm

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

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.

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).

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 all the codewords of weight ω.

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 all the codewords of weight ω.

Complexity can be estimated as:

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

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 all the codewords of weight ω.

Complexity can be estimated 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 $\mathsf{S}(\mathfrak{C}, i) \neq \mathsf{S}(\mathfrak{C}, j), \forall i \neq j$.

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:

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

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:

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

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

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

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

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

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 all 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,* $\mathfrak{C} \overset{\mathsf{LE}}{\sim} \mathfrak{C}'$ *if and only if* $\tilde{\mathfrak{C}} \overset{\mathsf{PE}}{\sim} \tilde{\mathfrak{C}}'$*.*

Leon's algorithm needs to enumerate all 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.

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.

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.

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

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

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

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

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.

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)

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.

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.

Thank you