How easy is code equivalence over Fq?

Nicolas Sendrier, Dimitrios E. Simos

Noname manuscript No.
(will be inserted by the editor)

How easy is code equivalence over $\mathbb{F}_q$?

Nicolas Sendrier · Dimitris E. Simos

Received: date / Accepted: date

Abstract The linear code equivalence problem is to decide whether two linear codes over $\mathbb{F}_q$ are identical up to a linear isometry of the Hamming space. The support splitting algorithm [23] runs in polynomial time for all but a negligible proportion of all linear codes, and solves the latter problem by recovering the isometry when it is just a permutation of the code support. While for a binary alphabet isometries are exactly the permutations, this is not true for $q \geq 3$. We give in this paper, a generalization of the support splitting algorithm where we aim to retrieve any isometry between equivalent codes. Our approach is twofold; first we reduce the problem of deciding the equivalence of linear codes to an instance of permutation equivalence. To this end, we introduce the notion of the closure of a code and give some of its properties. In the aftermath, we exhibit how this algorithm can be adapted for $q \in \{3, 4\}$, where its complexity is polynomial for almost all of its instances. Although the aforementioned reduction seems attractive, when $q \geq 5$ the closure reduces the instances of the linear code equivalence problem to exactly those few instances of permutation equivalence that were hard for the support splitting algorithm. Finally, we argue that for $q \geq 5$ the linear code equivalence problem might be hard for almost all instances and we elaborate on the various complexity problems.

Keywords Equivalence · Isometry · Closure of a Code · Linear Codes

Mathematics Subject Classification (2000) 94B05 · 05E20

Nicolas Sendrier[1]
[1] INRIA Paris-Rocquencourt, Project-Team SECRET
78153 Le Chesnay Cedex, France
E-mail: nicolas.sendrier@inria.fr

Dimitris E. Simos[1,2]
[1] INRIA Paris-Rocquencourt, Project-Team SECRET
78153 Le Chesnay Cedex, France
E-mail: dimitrios.simos@inria.fr
[2] SBA Research, 1040 Vienna, Austria
E-mail: dsimos@sba-research.org

1 Introduction

The purpose of this work is to examine the worst-case and average-case hardness of the Linear Code Equivalence problem. That is, given the generator matrices of two q-ary linear codes, how hard is it to decide whether or not these codes are identical up to a linear isometry of the Hamming space? The computational version of this problem, is to retrieve the linear isometry.

The Permutation Code Equivalence problem is the restriction of the above problem when the isometries are limited to permutations of the code support[1]. Petrank and Roth proved [20] that the worst-case was not easier than for the Graph Isomorphism problem. On the other hand, the support splitting algorithm [23] solves the computational version of the problem in time polynomial for all but an exponentially small proportion of the instances.

For a more general notion of code equivalence which includes all linear isometries, the situation seems to change drastically. In practice, the support splitting algorithm can be extended for $q \in \{3, 4\}$, and similarly solves all but an exponentially small proportion of the instances in polynomial time. However, for any fixed $q \geq 5$, the computational and the decisional problem seems to be intractable for almost all instances.**TODO:** emphasize more on the results

The paper is structured as follows. In section 2, we present the different notions of code equivalence induced by isometries of the Hamming space, while in section 3, we define in formal terms all decisional and computational problems related to code equivalence and mention the most significant contributions in terms of complexity and algorithms. In section 5, we illustrate a reduction of the Linear Code Equivalence problem as an instance of the Permutation Code Equivalence, and its efficiency is analyzed in the following section. Finally, we elaborate on the hardness of these computational and decisional problems and mention possible implications, in the concluding discussion.**TODO:** Structure needs update

2 Equivalence of linear codes

Code equivalence is a basic concept in coding theory. However, the equivalence of linear codes has met a few different definitions in the literature, often without motivation. We review the concept of what it means for codes to be "essentially different" by considering the metric Hamming space together with its isometries, which are the maps preserving the metric structure. This in turn will lead to a rigorous definition of equivalence of linear codes. In fact, we will call codes isometric if they are equivalent as subspaces of the Hamming space.

Let $\mathbb{F}_q$ be a finite field of cardinality $q = p^r$, where the prime number p is its characteristic, and r is a positive integer. As usual, a linear $[n, k]$ code C is a k-dimensional subspace of the finite vector space $\mathbb{F}_q^n$ and its elements are called codewords. We consider all vectors, as row vectors. Therefore, an element v of $\mathbb{F}_q^n$ is of the form $v := (v_1, \ldots, v_n)$. It can also be regarded as

[1] except for $q = 2$ the isometries are not limited to permutations.

the mapping υ from the set $\mathcal{I}_n = \{1, \ldots, n\}$ to $\mathbb{F}_q$ defined by $\upsilon(i) := \upsilon_i$. The Hamming distance (metric) on $\mathbb{F}_q^n$ is the following mapping,

$$d : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{N} : (x, y) \mapsto d(x, y) :=\mid \{i \in \{1, 2, \ldots, n\} \mid x_i \neq y_i\} \mid .$$

The pair $(\mathbb{F}_q^n, d)$ is a metric space, called the Hamming space of dimension n over $\mathbb{F}_q$, denoted by $H(n, q)$. The Hamming weight $w(x)$ of a codeword $x \in C$ is simply the number of its non-zero coordinates, i.e. $w(x) := d(x, 0)$.

Two codes C, C' are of the same quality if there exists a mapping $\iota : \mathbb{F}_q^n \mapsto \mathbb{F}_q^n$ with $\iota(C) = C'$ which preserves the Hamming distance, i.e. $d(\upsilon, \upsilon') = d(\iota(\upsilon), \iota(\upsilon'))$, for all $\upsilon, \upsilon' \in \mathbb{F}_q^n$. Mappings with the latter property are called the isometries of $H(n, q)$, and the two codes C and C' will be called isometric. It is well-known due to a theorem of MacWilliams that any linear[2] isometry between linear codes preserving the weight of the codewords induces an equivalence for codes [17]. Clearly, isometric codes have the same error-correction capabilities. We write $\mathcal{S}_n$ for the symmetric group acting on the set $\mathcal{I}_n$, equipped with the composition of permutations.

If $q = p^r$ is not a prime, then the Frobenius automorphism $\tau : \mathbb{F}_q \to \mathbb{F}_q, x \mapsto x^p$ applied on each coordinate of $\mathbb{F}_q^n$ preserves the Hamming distance, too. Moreover, for $n \geq 3$, the isometries of $\mathbb{F}_q^n$ which map subspaces onto subspaces are exactly the semilinear mappings[3][4] of the form $(\upsilon; (\alpha, \pi))$, where $(\upsilon; \pi)$ is a linear isometry and α is a field automorphism, i.e. $\alpha \in \mathrm{Aut}(\mathbb{F}_q)$ (c.f. [2,14]). All these mappings form the group of semilinear isometries of $H(n, q)$ which is isomorphic to the semidirect product $\mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$, where the multiplication of elements is given by

$$(\upsilon; (\alpha, \pi))(\varphi; (\beta, \sigma)) := (\upsilon \cdot \alpha(\varphi_\pi); (\alpha\beta, \pi\sigma)) \tag{1}$$

where, in detail we have $(\upsilon \cdot \alpha(\varphi_\pi))_i := \upsilon_i \alpha(\varphi_{\pi^{-1}(i)})$ for $i = 1, \ldots, n$. Furthermore, there is a description of $\mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$ as a generalized wreath product $\mathbb{F}_q^* \wr_n (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$, see [2,9,14]. Clearly, the notion of semilinear isometry which can be expressed as a group action on the set of linear subspaces gives rise to the most general notion of equivalence for linear codes. The action of the latter group in an element of $\mathbb{F}_q^n$ is translated into an equivalence for linear codes. Equivalence can also be induced by arbitrary isometries of $H(n, q)$, but such mappings may destroy linearity and we are only interested in isometries that map linear subspaces to linear subspaces.

Definition 1 Two linear codes $C, C' \subseteq \mathbb{F}_q^n$ will be called semilinearly equivalent, and will be denoted as $C \overset{\mathbf{SLE}}{\sim} C'$, if there exists a semilinear isometry $(\upsilon; (\alpha, \sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$ that maps C onto C', i.e. $C' = (\upsilon; (\alpha, \sigma))(C) = \{(\upsilon; (\alpha, \sigma))(x) \mid (x_i)_{i \in \mathcal{I}_n} \in C\}$ where $(\upsilon; (\alpha, \sigma))(x_1, \ldots, x_n) = (\upsilon_1 \alpha(x_{\sigma^{-1}(1)}), \ldots, \upsilon_n \alpha(x_{\sigma^{-1}(n)}))$.

[2] For all $u, v \in \mathbb{F}_q^n$ we have $\iota(u + v) = \iota(u) + \iota(v)$, $\iota(uv) = u\iota(v)$ and $\iota(0) = 0$.

[3] $\sigma : \mathbb{F}_q^n \to \mathbb{F}_q^n$ is semilinear if there exists $\alpha \in \mathrm{Aut}(\mathbb{F}_q)$ such that for all $u, v \in \mathbb{F}_q^n$ and $k \in \mathbb{F}_q$ we have $\sigma(u + v) = \sigma(u) + \sigma(v)$ and $\sigma(ku) = \alpha(k)\sigma(u)$.

[4] The action of the semilinear and linear group in an element of $\mathbb{F}_q^n$ can be seen at definitions 1 and 2, respectively.

The group of semilinear isometries of $H(n,q)$ reduces to the group of linear isometries if and only if q is a prime (since $\mathrm{Aut}(\mathbb{F}_q)$ is trivial if and only if q is a prime). The latter group corresponds to the semidirect product of $\mathbb{F}_q^{*n}$ and $\mathcal{S}_n$, $\mathbb{F}_q^{*n} \rtimes \mathcal{S}_n = \{(\upsilon;\pi) \mid \upsilon : \mathcal{I}_n \mapsto \mathbb{F}_q^*, \pi \in \mathcal{S}_n\}$, called the monomial group of degree n over $\mathbb{F}_q^*$. Note that, some authors [2,8,10], describe this group as the wreath product $\mathbb{F}_q^* \wr_n \mathcal{S}_n$. Therefore, by restricting the group of semilinear isometries to the group of linear isometries we have another notion of equivalence for linear codes.

Definition 2 Two linear codes $C, C' \subseteq \mathbb{F}_q^n$ will be called linearly or monomially equivalent, and will be denoted as $C \overset{\mathbf{LE}}{\sim} C'$, if there exists a linear isometry $\iota = (\upsilon;\sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ that maps C onto C', i.e. $C' = (\upsilon;\sigma)(C) = \{(\upsilon;\sigma)(x) \mid (x_1,\ldots,x_n) \in C\}$ where $(\upsilon;\sigma)(x_1,\ldots,x_n) := (\upsilon_1 x_{\sigma^{-1}(1)},\ldots,\upsilon_n x_{\sigma^{-1}(n)})$.

In addition, when $\mathbb{F}_q = \mathbb{F}_2$ the group of linear isometries of $H(n,2)$ is isomorphic to $\mathcal{S}_n$, and these isometries correspond to permutation of coordinates.

Definition 3 Two linear codes $C, C' \subseteq \mathbb{F}_q^n$ will be called permutationally equivalent and will be denoted as $C \overset{\mathbf{PE}}{\sim} C'$, if there exists a permutation $\sigma \in \mathcal{S}_n$ that maps C onto C', i.e. $C' = \sigma(C) = \{\sigma(x) \mid x = (x_1,\ldots,x_n) \in C\}$ where $\sigma(x) = \sigma(x_1,\ldots,x_n) := (x_{\sigma^{-1}(1)},\ldots,x_{\sigma^{-1}(n)})$.

Moreover, there is a particular subgroup of $\mathcal{S}_n$ that maps C onto itself, the permutation group of C defined as $\mathrm{PAut}(C) := \{C = \sigma(C) \mid \sigma \in \mathcal{S}_n\}$. $\mathrm{PAut}(C)$ always contains the identity permutation. If it does not contain any other element, we will say that it is trivial. Finally, we can define the monomial group of C as $\mathrm{MAut}(C) := \{C = (\upsilon;\sigma)(C) \mid (\upsilon;\sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n\}$ and the automorphism group of C as $\mathrm{Aut}(C) := \{C = (\upsilon;(\alpha,\sigma))(C) \mid (\upsilon;(\alpha,\sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)\}$ where their elements map each codeword of C to another codeword of C, under the respective actions of the involved groups. For more details, on automorphism groups of linear codes we refer to [13].

3 Previous work

For efficient computation of codes we represent them with generator matrices. A $k \times n$ matrix G over $\mathbb{F}_q$, is called a generator matrix for the $[n,k]$ linear code C if the rows of G form a basis for C, so that $C = \{xG \mid x \in \mathbb{F}_q^k\}$. In that case, we denote the code C that is spanned by the generator matrix G, as $C = \langle G \rangle$. In general, a linear code possess many different bases, and it is clear from linear algebra that the set of all generator matrices for C can be reached by $\{SG \mid S \in \mathrm{GL}_k(q)\}$, where $\mathrm{GL}_k(q)$ is the group of all $k \times k$ invertible matrices over $\mathbb{F}_q$.

For any $\sigma \in \mathcal{S}_n$ associate by $P_\sigma = [p_{i,j}]$ the $n \times n$ matrix such that $p_{i,j} = 1$ if $\sigma(i) = j$ and $p_{i,j} = 0$ otherwise, therefore P_σ is a permutation matrix. Note that, the action of $\sigma \in \mathcal{S}_n$ on $x \in \mathbb{F}_q^n$ agrees with the ordinary matrix multiplication. The permutation matrices form a subgroup of $M_n(q)$, the set

of all $n \times n$ monomial matrices over $\mathbb{F}_q$, that is, matrices with exactly one nonzero entry per row and column from $\mathbb{F}_q$. If $M = [m_{i,j}] \in M_n(q)$, then $M = DP$, where P is a permutation matrix and $D = [d_{i,j}] = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix with $d_i = d_{i,i} = m_{i,j}$ if $m_{i,j} \neq 0$ and $d_{i,j} = 0$ if $i \neq j$. There is an isomorphism between diagonal matrices and $\mathbb{F}_q^{*n}$, therefore we associate $D_v = \mathrm{diag}(v_1, \ldots, v_n)$ for $v = (v_i)_{i \in \mathcal{I}_n} \in \mathbb{F}_q^{*n}$. Hence, we can map any linear isometry $(v; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ to a monomial matrix $M_{(v;\sigma)} = D_v P_\sigma \in M_n(q)$, and this mapping is an isomorphism between $\mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ and $M_n(q)$. Therefore, we can express the equivalence between linear codes in terms of their generator matrices. As we have three different notions of equivalence, we define the respective decisional problems, below. The first one is w.r.t. the semilinear equivalence.

Problem 1 (Semilinear Code Equivalence (SLCE))

Parameters: n, k, q.
Instance: two matrices $G, G' \in \mathbb{F}_q^{k \times n}$.
Question: are $\langle G \rangle \overset{\mathbf{SLE}}{\sim} \langle G' \rangle$?

In a similar manner, we can define decisional problems related to linear and permutation equivalence.

Problem 2 (Linear Code Equivalence (LCE))

Parameters: n, k, q.
Instance: two matrices $G, G' \in \mathbb{F}_q^{k \times n}$.
Question: are $\langle G \rangle \overset{\mathbf{LE}}{\sim} \langle G' \rangle$?

Problem 3 (Permutation Code Equivalence (PCE))

Parameters: n, k, q.
Instance: two matrices $G, G' \in \mathbb{F}_q^{k \times n}$.
Question: are $\langle G \rangle \overset{\mathbf{PE}}{\sim} \langle G' \rangle$?

The computational versions of all three previous decisional problems, is to retrieve the equivalence mapping between the codes. Again, we begin with the semilinear equivalence.

Problem 4 (Computational Semilinear Code Equivalence (CSLCE))

Parameters: n, k, q.
Instance: two matrices $G, G' \in \mathbb{F}_q^{k \times n}$.
Problem: Find a semilinear isometry $(v; (\alpha, \sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$ such that $\langle G' \rangle = (v; (\alpha, \sigma))(\langle G \rangle)$.

Finally, we define the computational versions of the LCE and PCE problems.

Problem 5 (Computational Linear Code Equivalence (CLCE))

Parameters: n, k, q.
Instance: two matrices $G, G' \in \mathbb{F}_q^{k\times n}$.
Problem: Find a linear isometry $(v;\sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ such that $\langle G' \rangle = (v;\sigma)(\langle G \rangle)$.

Problem 6 (Computational Permutation Code Equivalence (CPCE))

Parameters: n, k, q.
Instance: two matrices $G, G' \in \mathbb{F}_q^{k\times n}$.
Problem: Find a permutation $\sigma \in \mathcal{S}_n$ such that $\langle G' \rangle = \sigma(\langle G \rangle)$.

One of our goals is to explore the hardness of the LCE and CLCE problems, therefore we deem necessary to briefly mention the most significant results in terms of complexity, for deciding them, and algorithms, for computing them.

3.1 Past complexity results

The PCE problem, was introduced in [20], who showed that if $\mathbb{F}_q = \mathbb{F}_2$ then it is harder than the Graph Isomorphism (GI) problem, there exists a polynomial time reduction, but not NP-complete unless P = NP. A different proof of this reduction is also given in [14]. Recently, the reduction of [20] was generalized in [12] over any field $\mathbb{F}_q$, hence the PCE problem is harder than the GI problem, for any field $\mathbb{F}_q$. An important complexity result concerning the LCE problem has been presented in [7]. In particular, a polynomial time reduction from the LCE problem to the GI problem is given, over any field $\mathbb{F}_q$. This implies that the LCE problem is again harder than the GI problem, for any field $\mathbb{F}_q$ and hard instances must be expected. Later on we also show in §7, that the LCE problem cannot be easier than the GI problem over $\mathbb{F}_q$. The latter problem, has been extensively studied for decades, but until now there is no polynomial-time algorithm for solving all of its instances.

Last but not least, we would like to mention that the McEliece public-key cryptosystem [18] is related to the hardness of permutationally equivalent binary linear codes. Towards this direction, another important complexity result was shown in [5], that the Hidden Subgroup problem also reduces to the PCE problem for any field $\mathbb{F}_q$. In detail, in [5] it was defined that a linear code C will be called HSP-hard if strong quantum Fourier sampling, reveals negligible information about the permutation $\sigma \in \mathcal{S}_n$ of permutationally equivalent codes, i.e. $C' = \sigma(C)$. This result was further extended in [24] where the instances of codes that are HSP-hard for the PCE problem, remain hard for the LCE problem.

3.2 Related algorithms for code equivalence

Due to its relation to the GI problem, some researchers have tried to solve the CPCE problem by interpreting graph isomorphism algorithms to codes. This

approach, was followed in [4] using the fact that orbits under edge local complementation of a bipartite graph correspond to equivalence classes of binary linear codes. Mapping codes to graphs and using the software NAUTY by B. D. McKay has been used in [19], for binary, ternary and quaternary codes where the permutation, linear and semi-linear equivalence was considered, respectively. Moreover, an adaptation of Luks's algorithm for hypergraph isomorphism for solving the CPCE problem over any $\mathbb{F}_q$ was presented in [1], whose complexity is simply-exponential in the length n of a code $C \subseteq \mathbb{F}_q^n$. Another approach using bipartite graphs for the CLCE problem over small fields was given in [3], where code equivalence is reduced to a decision problem regarding isomorphism of binary matrices. Note also, that in this work also the semi-linear equivalence was considered for $\mathbb{F}_4$. Computation of canonical forms for generator matrices of linear codes for the CSLCE problem over $\mathbb{F}_q$ by formulating the equivalence classes of codes as orbits of a group action from the left on the set of generator matrices was given in [6]. It is worthwhile also to mention the algorithm of J. Leon for computing the automorphism group of a code [15], which is available for many computer algebra systems like GAP and MAGMA, and is used for also for testing code equivalence. More specifically, in GAP, it is implemented for solving the CPCE problem over the binary field, while in MAGMA the implementation works for the CLCE problem, for small prime fields and for $\mathbb{F}_4$. However, Leon's algorithm requires a time exponential in the code dimension since it computes the set of all codewords of minimum weight.

Finally, we would like to remark that, to the best of our knowledge there is no efficient algorithm for solving the CLCE problem for any field $\mathbb{F}_q$.

3.3 The Support splitting algorithm

The support splitting algorithm ($\mathcal{SSA}$) is intended to solve the computational permutation code equivalence problem. It does so in polynomial time for all but a small fraction of linear codes. The algorithm uses the notions of invariant and signature.

Definition 4
- An *invariant* $\mathcal{R}$ over a set E maps a linear code C of length n on an element of E and is such that for any permutation σ of n elements we have $\mathcal{R}(\sigma(C)) = \mathcal{R}(C)$.
- A signature S over E is defined for any length n and maps a linear code C of length n and one of its positions $i \in \mathcal{I}_n$ on an element of E and is such that for any permutation σ of n elements we have $S(\sigma(C), \sigma(i)) = S(C, i)$.
- A signature S is *discriminant for the code* C if there exists i and j in $\mathcal{I}_n$ such that $S(C, i) \neq S(C, j)$, it is *fully discriminant for* C if all the $S(C, i), i \in \mathcal{I}_n$ are distinct.

For instance the Hamming weight enumerator of a code is an invariant. A signature can be obtained by applying an invariant on a punctured (or shortened) code. An abstract version of the support splitting algorithm is given in

Table 1 An abstract version of the support splitting algorithm [23]

Notations and definitions:

- A partition of $\mathcal{I}_n$ is denoted $\mathcal{P} = (\mathcal{P}_s)_{s\in E}$ with some index set E. The set E can be infinite but only a finite number of cells are non empty. We denote $|\mathcal{P}|$ the number of non empty elements and $|\mathcal{P}_s|$ the cardinality of the cell $\mathcal{P}_s$.
- Let $\mathcal{P}' = (\mathcal{P}'_s)_{s\in E}$ be another partition of $\mathcal{I}_n$ with the same index set, it is *equivalent* to $\mathcal{P}$, we denote $\mathcal{P} \equiv \mathcal{P}'$, if for all $s \in E$ we have $|\mathcal{P}_s| = |\mathcal{P}'_s|$.
- Let $\mathcal{Q} = (\mathcal{Q}_u)_{u\in F}$ be a partition of $\mathcal{I}_n$ indexed by a set F, the *product* of $\mathcal{P}$ and $\mathcal{Q}$ is defined as $\mathcal{P} \times \mathcal{Q} = (\mathcal{P}_s \cap \mathcal{Q}_u)_{(s,u)\in E\times F}$. It is a partition of $\mathcal{I}_n$ indexed by $E \times F$.

function `SSA`
`input:` $G, G' \in \mathbb{F}_q^{k\times n}$
`output:` two partitions of $\mathcal{I}_n$ with an identical index set
  $\mathcal{P} \leftarrow$ `SSA_step`(G) ; $\mathcal{P}' \leftarrow$ `SSA_step`(G')
  `while` $\mathcal{P} \equiv \mathcal{P}'$ `and` $|\mathcal{P}| < n$ *// repeat at most $O(\log n)$ times*
    // both $\mathcal{P}$ and $\mathcal{P}'$ are indexed by the same set F
    $s \leftarrow \{s \in F \mid \mathcal{P}_s \neq \emptyset\}$ *// at random or according to some heuristic*
    $\mathcal{P} \leftarrow$ `SSA_refine`$(G, \mathcal{P}, s)$; $\mathcal{P}' \leftarrow$ `SSA_refine`$(G', \mathcal{P}', s)$
  `return` $\mathcal{P}, \mathcal{P}'$

Parameter: a signature S over E.

function `SSA_step`
`input:` $G \in \mathbb{F}_q^{k\times n}$
`output:` a partition of $\mathcal{I}_n$ indexed by E
  `for` $i \in \mathcal{I}_n$
    $s \leftarrow S(\langle G\rangle, i)$; $\mathcal{P}_s \leftarrow \mathcal{P}_s \cup \{i\}$ *// all $\mathcal{P}_s$ are initially empty*
  `return` $(\mathcal{P}_s)_{s\in E}$

function `SSA_refine`
`input:` $G \in \mathbb{F}_q^{k\times n}$, $\mathcal{P}$ a partition of $\mathcal{I}_n$ indexed by F, $s \in F$
`output:` a partition of $\mathcal{I}_n$ indexed by $F \times E$
  $\mathcal{Q} \leftarrow$ `SSA_step`$(G_{\mathcal{P}_s})$ *// columns of G indexed by $\mathcal{P}_s$ are replaced by zeroes*
  `return` $\mathcal{P} \times \mathcal{Q}$

Table 1. It takes as parameter a signature S. The behavior of the algorithm can be predicted regardless of S.

Proposition 1 *Let $G, G' \in \mathbb{F}_q^{k\times n}$, $C = \langle G\rangle$, and $C' = \langle G'\rangle$. We run the algorithm of Table 1 with input G, G' and obtain as output $\mathcal{P} = (\mathcal{P}_s)_{s\in F}$ and $\mathcal{P}' = (\mathcal{P}'_s)_{s\in F}$. If $C' = \sigma(C)$, we have $\mathcal{P} \equiv \mathcal{P}'$ and $\mathcal{P}'_s = \sigma(\mathcal{P}_s)$ for all $s \in F$.*

Proof Because of the definition of a signature, when two calls to `SSA_step` are made on equivalent codes C and $C' = \sigma(C)$ the returned partitions $\mathcal{P}$ and $\mathcal{P}'$ are equivalent. Moreover, if $i \in \mathcal{P}_s$ we have $s = S(C, i) = S(C', \sigma(i))$ and $\sigma(i) \in \mathcal{P}'_s$. And conversly.

At every step of the algorithm $\mathcal{P}$ and $\mathcal{P}'$ are always the result of calls to `SSA_step` with equivalent inputs. □

The above proposition implies in particular that when the output partitions only have singleton cells, only one permutation can match, and its value is immediately available from the partitions. If the signature S is fully discriminant

for either one of its outputs the algorithm will succeed in a single step. If it is somewhat discriminant –but not fully– it can usually be transformed into a fully discriminant signature after a logarithmic number of refinement steps. In both cases, we can either conclude on the non equivalence, or the two outputs are partitions which only contain singletons and at most one permutation is compatible. This permutation can be checked in polynomial time and the decision can be made. Finally, if the partitions are equivalent but the signature is poorly discriminant or not discriminant at all, no decision can be made.

Discriminant signature from the hull. For all $\beta \in \mathbb{F}_q$, we define $C_{\beta \cdot i}$ as the code C in which the i-th coordinate is multiplied by β. Thus $C_{1 \cdot i}$ is the code itself and $C_{0 \cdot i}$ is the code punctured in i. In addition, we conventionally define $C_{\infty \cdot i} = ((C^\perp)_{0 \cdot i})^\perp$ which is (essentially, but not exactly) the code shortened in i. It is proven in [23] that the intersections $C_{\beta \cdot i} \cap C^\perp$ are equal to the hull $\mathcal{H}(C) = C \cap C^\perp$ shortened in i, except for exactly one value, say $\beta \in \mathbb{F}_q \cup \{\infty\}$, for which the intersection has one dimension more or less. The $\mathcal{SSA}$ uses as signature

$$S(C, i) = \left(\beta, \mathcal{W}(C_{\beta \cdot i} \cap C^\perp)\right)$$

where $\mathcal{W}(C)$ denotes the Hamming weight enumerator of a code C. It is a signature because the hull commutes with the permutation of the support[5] ($\mathcal{H}(\sigma(C)) = \sigma(\mathcal{H}(C))$). It is discriminant enough to allow the algorithm to succeed most of the time with a logarithmic number of refinement steps. The most notable exceptions are codes with a non trivial permutation group, those have to be handled differently (see below). More details can be found in [23].

Codes with a Non-trivial group. A consequence of Proposition 1 is that the output partitions of `SSA` cannot be finer than the orbits of the permutation groups of the respective input codes. In particular, no signature can be discriminant for a code with a transitive permutation group. When this happens we obtain two coarse but equivalent partitions. We puncture the codes on positions with identical signature and repeat this process until the permutation group vanishes. An extended version of the algorith is given in Table 2. A sufficient condition for this extension to work properly is that the cells of the partitions returned by the function `SSA` of Table 1 are the orbits of the coordinates under the action of the permutation group. In that case, the recursive calls to `SSA_group` will stop as soon as the set of all indices i chosen to puncture the code form a base of the permutation group (in the sense of [25]). Unfortunately, it seems impossible to prove the correct behavior of the algorithm, we cannot exclude the possibility that the partition is coarser though we never observed a counterexample.

[5] The hull does not always commute with isometries and this will have important consequences as we will see later

Table 2 An extension of $\mathcal{SSA}$ to handle permutation groups

function `SSA_group`
`input:` $G, G' \in \mathbb{F}_q^{k\times n}$
`output:` two partitions of $\mathcal{I}_n$ with an identical index set
$\mathcal{P}, \mathcal{P}' \leftarrow$ `SSA`(G, G')
`if` $\mathcal{P} \equiv \mathcal{P}'$ `and` $|\mathcal{P}| < n$
$s \leftarrow \{s \in F \mid \mathcal{P}_s \neq \emptyset\}$ *// $\mathcal{P}, \mathcal{P}'$ both indexed by F*
$i \leftarrow \mathcal{P}_s$; $j \leftarrow \mathcal{P}'_s$
$\mathcal{P}, \mathcal{P}' \leftarrow$ `SSA_group`(G_i, G'_j) *// column i or j resp. is set to zero*
`return` $\mathcal{P}, \mathcal{P}'$

After several recursive calls to `SSA_group` the matrices will contain several zero columns and the algorithm will be unable to discriminate between those. This would lead to an inacurate result. In reality, we define two partitions $\mathcal{Q}$ and $\mathcal{Q}'$ over $\{0,1\}$ with $\mathcal{Q}_0 = \{i\}$, $\mathcal{Q}_1 = \mathcal{I}_n \setminus \{i\}$, $\mathcal{Q}'_0 = \{j\}$, and $\mathcal{Q}'_1 = \mathcal{I}_n \setminus \{j\}$. Just after the call to `SSA_group` we set $\mathcal{P} \leftarrow \mathcal{P} \times \mathcal{Q}$ and $\mathcal{P}' \leftarrow \mathcal{P}' \times \mathcal{Q}'$. We feel that this would needlessly obscure the pseudocode and we prefer to leave it as a comment.

Hard instances and algorithmic complexity. We make the empirical assumption that the support splitting algorithm (Table 1) always returns partitions whose cells are the orbits under the action of the permutation group. In that case the number of calls to the elementary function `SSA_step` is logarithmic in the code length. The algorithm will make the following computations:

- once, it computes the hull of the codes, this is essentially a matrix inversion, we assumme a cost of $O(n^3)$ and we denote h the dimension of the hulls[6]
- for each of the $O(\log n)$ `SSA_step`, the algorithm computes $O(n)$ signatures, that is
  - find the value of β, which has a constant cost as it is a coefficient of a transition matrix computed with the hull.
  - compute the intersection $C_{\beta\cdot i} \cap C^\perp$ and adjust the transition matrix for a cost $O(hn)$
  - compute the weight enumerator of a code of dimension (close to) h for a cost $O(q^h n)$

The total order of magnitude of the algorithmic cost is $O(n^3 + q^h n^2 \log n)$ operation in $\mathbb{F}_q$.

1. If the hulls of the input codes behave like those of random codes, they have constant size (see the Proposition at the end of the section) and the algorithm cost is domanate by the initial matrix inversion, that is at most $O(n^3)$.
2. If the input codes have hulls of large dimension, the computation of the weight enumerator dominates and the cost which becomes $O(q^h n^2 \log n)$ elementary operations in $\mathbb{F}_q$ where h is the hull dimension.

The hard instances are easy to identify and to exhibit. When the inputs are weakly self-dual codes the algorithm has a running time exponential in the code dimension.

[6] If the input codes have hulls of different dimension they are not permutation equivalent.

Average size of the hull.

Proposition 2 *The hull of a linear code has cardinality 2 asymptotically on average.*

Proof Asymptotically, regardless of their dimension, q-ary linear codes have a hull of dimension i and thus cardinality q^i with probability R_i where (see [22])

$$R_i = \frac{R_{i-1}}{q^i - 1}, i \geq 1 \text{ and } R_0 = \prod_{i \geq 1} \left(1 - \frac{1}{q^i + 1}\right).$$

The asymptotic average cardinality is then

$$\sum_{i \geq 0} q^i R_i = \sum_{i \geq 0} (q^i - 1) R_i + \sum_{i \geq 0} R_i = \sum_{i \geq 1} R_{i-1} + \sum_{i \geq 0} R_i = 2.$$

4 Generalization of the support splitting algorithm to linear and Semi-linear code equivalence

4.1 Semi-linear code equivalence *vs.* linear code equivalence

The finite field $\mathbb{F}_q$ with $q = p^m$ and p a prime admits exactly m distinct field automorphisms. Thus if one has access to a program able to either decide or compute linear equivalence, calling that program m times will be enough to solve the corresponding semi-linear equivalence problem.

4.2 Linear code equivalence and the $\mathcal{SSA}$

Partial reconstruction. Let C and C' be two q-ary linear $[n, k]$ code and assume we somehow know a permutation σ such that $C' = (\upsilon; \sigma)(C)$. The scalars $\upsilon = (\upsilon_1, \ldots, \upsilon_n)$ are by recovered by solving a linear system.

Proposition 3 *Let C and C' be two q-ary linear $[n, k]$ code which admits systematic generator matrices of the form*

$$G = \left(\begin{array}{ccc|c} 1 & & & \\ & \ddots & & \\ & & 1 & \end{array} \right. \left. \begin{array}{c} \\ g_{i,j} \\ \\ \end{array} \right) \text{ and } G' = \left(\begin{array}{ccc|c} 1 & & & \\ & \ddots & & \\ & & 1 & \end{array} \right. \left. \begin{array}{c} \\ g'_{i,j} \\ \\ \end{array} \right)$$

where i ranges from 1 to k and j from $k+1$ to n. We have $C' = (\upsilon, 1)(C)$ if and only if $\upsilon = (\upsilon_1, \ldots, \upsilon_n)$ is a non-zero solution of the system

$$\upsilon_i g'_{i,j} = \upsilon_j g_{i,j}, 1 \leq i \leq k, k < j \leq n.$$

Proof If $C' = (v, 1)(C)$ then

$$G \cdot \begin{pmatrix} v_1 & & \\ & \ddots & \\ & & v_n \end{pmatrix} = \left(\begin{array}{ccc|c} v_1 & & & \\ & \ddots & & v_j g_{i,j} \\ & & v_k & \end{array}\right)$$

is a generator matrix of C'. Because of the uniqueness of the systematic form, we must have

$$\left(\begin{array}{ccc|c} v_1 & & & \\ & \ddots & & v_j g_{i,j} \\ & & v_k & \end{array}\right) = \begin{pmatrix} v_1 & & \\ & \ddots & \\ & & v_n \end{pmatrix} \cdot G' = \left(\begin{array}{ccc|c} v_1 & & & \\ & \ddots & & v_i g'_{i,j} \\ & & v_k & \end{array}\right)$$

which implies that v must be a solution of the system given in the statement. Conversely, if v is a non-zero solution to the system, the above identities imply $C' = (v, 1)(C)$. □

Invariants and signatures. The notions of invariants and signatures generalize naturally with other notions of equivalence. For any linear or semilinear isometry Ψ, we will denote $\mathrm{perm}(\Psi)$ its permutation component.

Definition 5 – A mapping is an invariant for the linear (respectively semilinear) equivalence if it takes the same value for any linearly (respectively semilinearly) equivalent linear codes.
– A signature S over E for the linear (respectively semilinear) equivalence is defined for any length n and maps a linear code C of length n and one of its positions $i \in \mathcal{I}_n$ on an element of E and is such that for any linear (respectively semilinear) isometry Ψ we have $S(\Psi(C), \mathrm{perm}(\Psi)(i)) = S(C, i)$ for all $i \in \mathcal{I}_n$.

This notion of invariant is consistent with the folklore of algebraic coding theory. It means that codes that are identical up to an isometry share a common property. For instance, the weight enumerator of a code is an invariant for the semilinear equivalence. As before, a convenient way to construct signatures will be to apply an invariant to punctured or shortened codes.

However, the above notion is more general, and there exists mappings which are invariant for permutation equivalence but not for linear equivalence. An importance instance is the weight enumerator of the hull. It is an invariant for permutations (the $\mathcal{SSA}$ is built on that) but it is not an invariant in general because the hull does not commute with isometries in general.

Generalized $\mathcal{SSA}$. Generalizing the support splitting algorithm is very simple. In Table 1, if the signature S used as parameter of `SSA_step` is for linear isometries, the Proposition 1 is easily generalized.

Proposition 4 *Let $G, G' \in \mathbb{F}_q^{k\times n}$, $C = \langle G\rangle$, and $C' = \langle G'\rangle$. We run the algorithm of Table 1, using as parameter a signature for linear (resp. semilinear) equivalence, with input G, G' and obtain as output $\mathcal{P} = (\mathcal{P}_s)_{s\in F}$ and $\mathcal{P}' = (\mathcal{P}'_s)_{s\in F}$. If $C' = \Psi(C)$ for some linear (resp. semilinear) isometry Ψ, we have $\mathcal{P} \equiv \mathcal{P}'$ and $\mathcal{P}'_s = \sigma(\mathcal{P}_s)$ for all $s \in F$ where $\sigma = \mathrm{perm}(\Psi)$.*

The proof is the same as for Proposition 1. It follows that if we somehow manage to build a fully (or somewhat) discriminant signature *for the linear equivalence*, the output of the same algorithm as for permutation equivalence will provide fine partitions, consisting of singletons, that will reveal the permutation part of the isometry we seek. From the partial reconstruction property we stated earlier in this section, this will be enough to recover the whole isometry. The case where the code has a non trivial automorphism group is treated as in Table 2 with a similar effect.

Ternary and quaternary linear codes

Lemma 1 *Let C be a ternary linear code, for any linear isometry Ψ we have $\Psi(C^\perp) = \Psi(C)^\perp$. Let C be a linear code over $\mathbb{F}_4$, for any linear isometry Ψ we have $\Psi(C^\perp) = \Psi(C)^\perp$ when duality is defined with repect to the Hermitian scalar product.*

Proof
- $\mathbb{F}_3$: for any isometry $\Psi = (\upsilon;\sigma)$ with $\upsilon = (\upsilon_1, \ldots, \upsilon_n)$, we have $\Psi(C)^\perp = (\upsilon^{-1};\sigma)(C^\perp)$ where $\upsilon^{-1} = (\upsilon_1^{-1}, \ldots, \upsilon_n^{-1})$. In $\mathbb{F}_3$, any non zero element is its own inverse and thus we have $\Psi(C^\perp) = \Psi(C)^\perp$
- $\mathbb{F}_4$: the Hermitian scalar product is defined as $\langle x, y\rangle_{\mathrm{H}} = \sum_{i=1}^n x_i y_i^2$. Let $\Psi = (\upsilon;\sigma)$ with $\upsilon = (\upsilon_1, \ldots, \upsilon_n)$. The dual of $\Psi(C)$ with respect to $\langle\cdot\rangle_{\mathrm{H}}$ is $\Psi(C)^\perp = (\upsilon^{-2};\sigma)(C^\perp)$ where $\upsilon^{-2} = (\upsilon_1^{-2}, \ldots, \upsilon_n^{-2}) = (\upsilon_1, \ldots, \upsilon_n)$ in $\mathbb{F}_4$. Again we have $\Psi(C^\perp) = \Psi(C)^\perp$. □

Because of this lemma, $\mathcal{H}(\Psi(C)) = \Psi(\mathcal{H}(C))$ for ternary and quaternary codes. It follows that the signature used in $\mathcal{SSA}$ is valid for linear equivalence and the algorithms of Table 1 and Table 2 perform correctly: as for permutation equivalence, it will decide equivalence and compute the permutation part of the isometry when the code are equivalent in polynomial time, unless the hull of the code is large.

Larger alphabet. When $q \geq 5$ the problem of code equivalence seems to become difficult for almost all instances. In fact, the $\mathcal{SSA}$ is still available and can be instantiated with any invariant for linear equivalence. Unfortunately, all invariants that lead to, even mildly, discriminant signatures require a computation time exponential in the code dimension making the algorithm unpractical except for small codes. For instance, with the weight enumerator, the $\mathcal{SSA}$ has complexity at least $O(q^k n^2)$ for q-ary (n, k) codes.

5 Reduction of linear code equivalence to permutation code equivalence

Hence, we have at our disposal an algorithm, the support splitting algorithm, that solves the PCE and CPCE problems in (almost) polynomial time. Therefore, it is natural to investigate a reduction of the LCE problem as an instance of the PCE problem. To this end, we introduce the *closure* of a linear code. We mention, that a similar approach was given in [26].

5.1 The closure of a linear code

Definition 6 The *closure* of a q-ary code C of length n is a code of length $n(q-1)$ defined as

$$\widetilde{C} = \{(\alpha x_i)_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*} \mid (x_i)_{i\in\mathcal{I}_n} \in C\}.$$

Definition 7 A monomial permutation of $\mathcal{I}_n \times \mathbb{F}_q^*$ is a permutation π such that for all $(i,\alpha) \in \mathcal{I}_n \times \mathbb{F}_q^*$ we have $\pi(i,\alpha) = (\sigma^{-1}(i), \alpha\upsilon_i)$ for some $\sigma \in \mathcal{S}_n$ and some $\upsilon = (\upsilon_1, \ldots, \upsilon_n) \in (\mathbb{F}_q^*)^n$. We will denote $\pi = \sigma \star \upsilon$.

The closure duplicate each code coordinate $(q-1)$ times with a different scaling. The effect on any generator matrix is the same. We omit the proof of the following lemma.

Lemma 2 *Let $\widetilde{G} = (g_{i,\alpha})_{\mathcal{I}_n\times\mathbb{F}_q^*}$ denote the generator matrix of a closure. For all $(i,\alpha) \in \mathcal{I}_n \times \mathbb{F}_q^*$ We have $g_{i,\alpha} = \alpha g_{i,1}$.*

Lemma 3 *Let $G = (g_i)_{i\in\mathcal{I}_n} \in \mathbb{F}_q^{k\times n}$ denote a q-ary matrix (the g_i are column vectors of size k). We denote $\widetilde{G} = (\alpha g_i)_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*} \in \mathbb{F}_q^{k\times(q-1)n}$ (αg_i is the column g_i multiplied by the scalar α). For $\sigma \in \mathcal{S}_n$ and $\upsilon = (\upsilon_1, \ldots, \upsilon_n) \in (\mathbb{F}_q^*)^n$, we denote $G' = (\upsilon_i g_{\sigma^{-1}(i)})_{i\in\mathcal{I}_n} \in \mathbb{F}_q^{k\times n}$ and $\widetilde{G}' = (\alpha\upsilon_i g_{\sigma^{-1}(i)})_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*} \in \mathbb{F}_q^{k\times(q-1)n}$. We have*

$$(\langle G'\rangle = (\upsilon;\sigma)(C)) \Leftrightarrow (\langle G\rangle = C) \Leftrightarrow \left(\langle\widetilde{G}\rangle = \widetilde{C}\right) \Leftrightarrow \left(\langle\widetilde{G}'\rangle = (\sigma\star\upsilon)(\widetilde{C})\right).$$

Proof The first equivalence derives directly from the definition of the monomial isometries and the second from the definition of the closure. For the last equivalence, we denote $\pi = \sigma \star \upsilon$ and we write $\widetilde{G} = (\tilde{g}_{i,\alpha})_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*}$. Since $\widetilde{G}$ is a generator matrix of $\widetilde{C}$, the matrix $(\tilde{g}_{\pi(i,\alpha)})_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*}$ is generator of the code $\pi(\widetilde{C})$. We have $\tilde{g}_{\pi(i,\alpha)} = \tilde{g}_{\sigma^{-1}(i),\alpha\upsilon_i} = \alpha\upsilon_i g_{\sigma^{-1}(i)}$ that is precisely the (i,α)-th column of $\widetilde{G}'$ which is thus a generator matrix of $\pi(\widetilde{C}) = (\sigma\star\upsilon)(\widetilde{C})$. □

Corollary 1 (of Lemma 3) *Let C and C' denote two q-ary linear codes of length n and let $\widetilde{C}$ and $\widetilde{C}'$ denote their closures. For all $\sigma \in \mathcal{S}_n$ and all $\upsilon = (\upsilon_1, \ldots, \upsilon_n) \in (\mathbb{F}_q^*)^n$ we have*

$$C' = (\upsilon;\sigma)(C) \Leftrightarrow \widetilde{C}' = (\sigma\star\upsilon)(\widetilde{C})$$

Proof Let $G = (g_i)_{i\in\mathcal{I}_n}$ denote a generator matrix of C. We use the notations of Lemma 3. If we denote $g'_i = \upsilon_i g_{\sigma^{-1}(i)}$, then we have $G' = (g'_i)_{i\in\mathcal{I}_n}$ and $\widetilde{G}' = (\alpha g'_i)_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*}$. Consequently the code spanned by $\widetilde{G}'$ is the closure of the code spanned by G'. From Lemma 3 again

$$(C' = \langle G'\rangle = (\upsilon;\sigma)(C)) \Leftrightarrow \left(\widetilde{C}' = \langle \widetilde{G}'\rangle = (\sigma\star\upsilon)(\widetilde{C})\right).$$

□

Lemma 4 *Let C and C' denote two q-ary linear codes of length n whose closures $\widetilde{C}$ and $\widetilde{C}'$ are permutation equivalent, say $\widetilde{C}' = \pi(\widetilde{C})$. We define σ and υ by $(\sigma^{-1}(i), \upsilon_i) = \pi(i,1)$ for all $i \in \mathcal{I}_n$, and we set $\pi' = \sigma\star\upsilon$. We have $\widetilde{C}' = \pi'(\widetilde{C})$ and $C' = (\upsilon;\sigma)(C)$.*

Proof Let $G = (g_i)_{i\in\mathcal{I}_n}$ denote a generator matrix of C. The matrix $\widetilde{G} = (\tilde{g}_{i,\alpha})_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*} = (\alpha g_i)_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*}$ is a generator matrix of $\widetilde{C}$. Because $\widetilde{C}' = \pi(\widetilde{C})$ the matrix $X = (x_{i,\alpha})_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*} = (\tilde{g}_{\pi(i,\alpha)})_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*}$ is a generator matrix of $\widetilde{C}'$.

By definition $\pi' = \sigma\star\upsilon$ is the only monomial permutation such that $\pi(i,1) = \pi'(i,1)$ for all $i \in \mathcal{I}_n$. We have $x_{i,1} = \tilde{g}_{\pi(i,1)} = \tilde{g}_{\sigma^{-1}(i),\upsilon_i} = \tilde{g}_{\pi'(i,1)}$. Because X is the generator matrix of a closure we have $x_{i,\alpha} = \alpha x_{i,1}$ (Lemma 2) and, because $\widetilde{G}$ is the generator matrix of a closure, we have $\alpha\tilde{g}_{\sigma^{-1}(i),\upsilon_i} = \tilde{g}_{\sigma^{-1}(i),\alpha\upsilon_i} = \tilde{g}_{\pi'(i,\alpha)}$, that is $x_{i,\alpha} = \tilde{g}_{\pi(i,\alpha)} = \tilde{g}_{\pi'(i,\alpha)}$ for all (i,α).

Finally, even if we may have $\pi \neq \pi'$, the actions of π and π' on the columns of a generator matrix of $\widetilde{C}$ leads to the same result. It follows that $\widetilde{C}' = \pi(\widetilde{C}) = (\sigma\star\upsilon)(\widetilde{C})$. From Corollary 1, we deduce that C and C' are linearly equivalent and $C' = (\upsilon;\sigma)(C)$. □

If the permutation between two closures is not monomial, it means that in a generator matrix $\widetilde{G} = (\tilde{g}_{i,\alpha})_{(i,\alpha)\in\mathcal{I}_n\times\mathbb{F}_q^*}$ of any of those closure, we can find two identical columns $\tilde{g}_{i,\alpha} = \tilde{g}_{j,\beta}$ with $i \neq j$. This can only mean that the i-th and j-th coordinate of the original code were proportional. We state this as a remark for future reference.

Remark 1 If neither C nor C' have no duplicate coordinate, any permutation between their closures is monomial.

The main results of this section are gathered in Theorem 1, below. This theorem is of great importance, because it realizes a reduction from the LCE problem to the PCE problem through the closure. Thus, we are able to decide if the codes C and C' are linearly equivalent by checking their closures for permutation equivalence. In addition, it is shown that a solution to the PCE problem for closures implies a solution to the LCE problem for codes. Furthermore, the monomial automorphism group of a code is contained to the permutation automorphism group of its closure, and in some cases these two groups are essentially the same. Most important, if the closures are permutation equivalent we exhibit in §5.3 an algorithmic procedure that will allow

us to recover the initial isometry between C and C'. However, as we shall see shortly after in §5.2, the closure reduces an instance of the CLCE problem to exactly those instances that were hard for the support splitting algorithm for tackling the CPCE problem over $\mathbb{F}_q$, $q \geq 5$.

Theorem 1 *Let $C, C' \subseteq \mathbb{F}_q^n$.*

(i) Then C and C' are linearly equivalent, i.e. $C \overset{\mathbf{LE}}{\sim} C'$, if and only if $\widetilde{C}$ and $\widetilde{C}'$ are permutationally equivalent, i.e. $\widetilde{C} \overset{\mathbf{PE}}{\sim} \widetilde{C}'$.

(ii) If $\widetilde{C}' = \pi(\widetilde{C})$ for some permutation π of degree $(q-1)n$ then there exists a linear isometry $\Psi = (\upsilon; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ such that $C' = \Psi(C)$.

(iii) Then $\mathrm{MAut}(C) \subseteq \mathrm{PAut}(\widetilde{C}) \cap \mathcal{S}(q-1, n)$ where by $\mathcal{S}(q-1, n)$ we denote the generalized symmetric group. In addition, if the code C has no identical coordinates (up to a scalar) the two groups are the same, i.e. $\mathrm{MAut}(C) = \mathrm{PAut}(\widetilde{C}) \cap \mathcal{S}(q-1, n)$.

Proof (i) ($\Longrightarrow$) Assume that C and C' are linearly equivalent under an action of a linear isometry $(\upsilon; \sigma)$ with corresponding closures $\widetilde{C}$ and $\widetilde{C}'$. By Corollary 1, the closures of C and C' are mapped one into another under an action of a monomial permutation and we obtain the result.
($\Longleftarrow$) The converse follows directly from Lemma 4.

(ii) Assume that the two closures $\widetilde{C}$ and $\widetilde{C}'$ of the codes C and C' are permutation equivalent under the action of the permutation π. By Lemma 4 there exists a monomial permutation π' in terms of $\sigma \star \upsilon$ for some $\sigma \in \mathcal{S}_n$ and $\upsilon = (\upsilon_1, \ldots, \upsilon_n) \in (\mathbb{F}_q^*)^n$ defined as in the same lemma. We have that $\widetilde{C}' = \pi(\widetilde{C}) = (\sigma \star \upsilon)(\widetilde{C})$ and from the one-to-one correspondence between monomial permutations and linear isometries the result follows.

(iii) If we consider the cyclic group $\mathcal{C}_{q-1}$ of order $q-1$ there is a natural isomorphism between $\mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ and $\mathcal{C}_{q-1} \wr_n \mathcal{S}_n$, the semidirect product of n copies of $\mathcal{C}_{q-1}$ and $\mathcal{S}_n$, called also the generalized symmetric group which we denote by $\mathcal{S}(q-1, n)$. Its order is $(q-1)^n n!$ and its elements are permutations that can also appear as permutations of permutationally equivalent closures. Now, take an element of $\mathrm{MAut}(C)$, i.e. a linear isometry $(\upsilon; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ such that $C = (\upsilon; \sigma)(C)$. From Corollary 1, we have that $\widetilde{C}' = (\sigma \star \upsilon)(\widetilde{C})$ where the permutation $\pi = (\sigma \star \upsilon) \in \mathrm{PAut}(\widetilde{C})$ and the inclusion follows. When the code C has no identical coordinates (up to a scalar), by Remark 1 every element of $\mathrm{PAut}(\widetilde{C})$ consists of monomial permutations which are in one-to-one correspondence with all linear isometries of $\mathrm{MAut}(C)$ and in this case we have that $\mathrm{MAut}(C) = \mathrm{PAut}(\widetilde{C}) \cap \mathcal{S}(q-1, n)$. □

Weight properties of the closure. In this paragraph, we consider some properties related to the weight enumerator of the closure and related them to those of the weight enumerator of the original code.

Proposition 5 *Let a code $C \subseteq \mathbb{F}_q^n$. The weight enumerator of the closure $\widetilde{C}$ of the code C is a multiple of the weight enumerator of C. In particular, it holds $W_X(\widetilde{C}) = X^{q-1} \mathcal{W}_X(C)$.*

Proof Since, the code and its closure have the same dimension, therefore they contain the same number of codewords, it will be suficient to show that any codeword $x \in C$ of weight $w(x) = w$ is mapped to a codeword $\widetilde{x} \in \widetilde{C}$ of weight $w(\widetilde{x}) = (q-1)w$.

Recall, that the closure duplicate each code coordinate $(q-1)$ times with a different scaling. Therefore, we have w non-zero code coordinates of x replicated $(q-1)$ times in a codeword $\widetilde{x}$ of the closure and the result follows. □

This result in addition implies that the number of codewords of weight i in C are equal to the number of codewords of weight $(q-1)i$ in $\widetilde{C}$. In particular, if the weight spectrum of $C \subseteq \mathbb{F}_q^n$ is $\mathrm{spec}(C) = [1, A_1, \ldots, A_i, \ldots, A_n]$ then the spectrum of the closure $\widetilde{C}$ is given by the following formula:

$$\mathrm{spec}(\widetilde{C}) = [1, \underbrace{0, \ldots, 0}_{q-2}, A_1, \underbrace{0, \ldots, 0}_{q-2}, \ldots, \underbrace{0, \ldots, 0}_{q-2}, A_i, \underbrace{0, \ldots, 0}_{q-2}, \ldots, \underbrace{0, \ldots, 0}_{q-2}, A_n]$$

Corollary 2 *If C is an $[n, k, d]$ code over $\mathbb{F}_q$ then the closure $\widetilde{C}$ of C is an $[(q-1)n, k, (q-1)d]$ code over $\mathbb{F}_q$.*

Proof Since the minimum distance of a code is the lowest weight of its non-zero codewords the result follows from Proposition 5. □

We conclude this paragraph with the following remark, which states the relationship between the weight enumerator of the hull of a code and its closure.

Remark 2 Let $C \subseteq \mathbb{F}_q^n$ with closure $\widetilde{C} \subseteq \mathbb{F}_q^{(q-1)n}$. We have that if $\dim(\mathcal{H}(C)) = h$ then $\dim\left(\widetilde{\mathcal{H}(C)}\right) = h$ and $\mathcal{W}_X(\widetilde{\mathcal{H}(C)}) = X^{q-1}\mathcal{W}_X(\mathcal{H}(C))$.

5.2 Efficiency of the reduction

The $\mathcal{SSA}$ used as an invariant the hull $\mathcal{H}(C)$ of a code. In order to explore possible extensions of $\mathcal{SSA}$ we have to determine the quality of the hull of the closure $\mathcal{H}(\widetilde{C}) = \widetilde{C} \cap \widetilde{C}^{\perp}$, where the dual of the closure is defined according to some inner product. We consider two inner products, the Euclidean and Hermitian inner product, defined below:

- $\langle x, y\rangle_{\mathrm{E}} = \sum_{i=1}^{n} \langle x_i, y_i\rangle_{\mathrm{E}} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + \ldots + x_n y_n \in \mathbb{F}_q$. If q is a square, $\langle x, y\rangle_{\mathrm{H}}$ (below) is generally preferred to $\langle x, y\rangle_{\mathrm{E}}$.
- $\langle x, y\rangle_{\mathrm{H}} = \sum_{i=1}^{n} \langle x_i, y_i\rangle_{\mathrm{H}} = \sum_{i=1}^{n} x_i \overline{y}_i = x_1 \overline{y}_1 + \ldots + x_n \overline{y}_n \in \mathbb{F}_q$, where q is an even power of a prime with $\overline{x} = x^{\sqrt{q}}$ for $x \in \mathbb{F}_q$ (cf. [21]). Note that, for $x, y \in \mathbb{F}_q$,
$$(x + y)^{\sqrt{q}} = x^{\sqrt{q}} + y^{\sqrt{q}}, \quad x^q = x.$$

When a code C is contained in its dual (defined according to some inner product), i.e. $C \subset C^{\perp}$, we say that the code C is a weakly self-dual code.

Proposition 6 *Let $C \subseteq \mathbb{F}_q^n$ be a linear code. The closure $\widetilde{C}$ of C is a weakly self-dual code over $\mathbb{F}_q$, for every $q \geq 5$ considering both Euclidean and Hermitian duals.*

Proof Now, consider two codewords $\widetilde{x}, \widetilde{y}$ of the closure $\widetilde{C}$ of $C \subseteq \mathbb{F}_q^n$. Then their Euclidean and Hermitian inner product is given by $\langle \widetilde{x}, \widetilde{y} \rangle_{\mathrm{E}} = \left(\sum_{i=1}^{q-1} a_i^2 \right) \langle x, y \rangle_{\mathrm{E}}$ and $\langle \widetilde{x}, \widetilde{y} \rangle_{\mathrm{H}} = \left(\sum_{i=1}^{q-1} a_i \overline{a}_i \right) \langle x, y \rangle_{\mathrm{H}}$, respectively, where $\mathbb{F}_q = \{a_0 = 0, a_1, \ldots, a_{q-1}\}$. Using lemma 7.3. of [16] which states that $a_0, a_1, \ldots, a_{q-1}$ are distinct if and only if $\sum_{i=0}^{q-1} a_i^t = 0$ for $t = 0, 1, \ldots, q-2$ and $\sum_{i=0}^{q-1} a_i^t = -1$ for $t = q-1$, we can show that,

$$\langle \widetilde{x}, \widetilde{y} \rangle_{\mathrm{E}} = \begin{cases} 0 & \text{for} \quad q \geq 4 \\ -\langle x, y \rangle_{\mathrm{E}} & \text{for} \quad q = 3. \end{cases} \text{ and } \langle \widetilde{x}, \widetilde{y} \rangle_{\mathrm{H}} = \begin{cases} 0 & \text{for} \quad q > 4 \\ -\langle x, y \rangle_{\mathrm{H}} & \text{for} \quad q = 4. \end{cases}$$

This concludes the proof. □

This means, that the closure $\widetilde{C}$ is a weakly self-dual code for every $q \geq 5$, considering both Euclidean and Hermitian duals, which is exactly the hard instances of $\mathcal{SSA}$. Moreover, for $\mathbb{F}_3$ and $\mathbb{F}_4$ equipped with the Euclidean and Hermitian inner product, respectively, the distribution of the dimension of $\mathcal{H}(\widetilde{C})$ follows the distribution of the dimension $\mathcal{H}(C)$, since the closure has the same dimension as C, and will be on average a small constant, [22], except in the cases where C is also a weakly self-dual code.

5.3 An extension of $\mathcal{SSA}$ to $\mathbb{F}_3$ and $\mathbb{F}_4$

As we have witnessed earlier, the support splitting algorithm can be adapted for treating the LCE problem over $\mathbb{F}_q$, as long as we provide an invariant for the linear code equivalence in order to be able to build discriminant signatures.

However, the hull is not an invariant in general for the LCE problem and this served as an additional motivation for the introduction of the closure of a code. In particualr, we have to mention that due to Theorem 1 we can consider permutationally equivalent closures and linearly equivalent closures, interchangeably. Thus, any invariant derived from the hull of the closures for the PCE problem will serve as an invariant for the LCE problem for the original codes. Moreover, the fact that the closure is a weakly self-dual code over $\mathbb{F}_q$, for $q \geq 5$ provides a negative result but also renders visible the cases where an extension of $\mathcal{SSA}$ via the closure is feasible, specifically for $\mathbb{F}_3$ and $\mathbb{F}_4$.

For these two cases, of ternary and quaternary linear codes there exists an additional relation between the hull of the closure and the closure of the hull, which we present below.

Proposition 7 *Let $C \subseteq \mathbb{F}_q^n$ with closure $\widetilde{C} \subseteq \mathbb{F}_q^{(q-1)n}$. Then $\widetilde{\mathcal{H}(C)} = \mathcal{H}(\widetilde{C})$ when*

(i) $\mathbb{F}_q = \mathbb{F}_3$ equipped with the Euclidean inner product.
(ii) $\mathbb{F}_q = \mathbb{F}_4$ equipped with the Hermitian inner product.

Proof We will prove the necessary inclusions. Recall that $\widetilde{x}$ denotes the image of a codeword $x \in C$ in the closure $\widetilde{C}$.
($\subseteq$) In order to show that $\widetilde{\mathcal{H}(C)} \subseteq \mathcal{H}(\widetilde{C})$, is sufficient to show that for all $x \in \mathcal{H}(C) = C \cap C^\perp$ we have $\widetilde{x} \in \mathcal{H}(\widetilde{C}) = \widetilde{C} \cap \widetilde{C}^\perp$.

By the definition of the closure, it holds $\widetilde{x} \in \widetilde{C}$. Since, $x \in \mathcal{H}(C)$ therefore also $x \in C^\perp \Leftrightarrow \forall y \in C$, we have $\langle x, y\rangle = 0$. We have also that for all $\widetilde{y} \in \widetilde{C}$ it holds $\langle \widetilde{x}, \widetilde{y}\rangle = 0 \Leftrightarrow \widetilde{x} \in \widetilde{C}^\perp$. Finally, we obtain that $\widetilde{x} \in \widetilde{C} \cap \widetilde{C}^\perp = \mathcal{H}(\widetilde{C})$.
($\supseteq$) Now, we have to show that $\mathcal{H}(\widetilde{C}) \subseteq \widetilde{\mathcal{H}(C)} = \widetilde{C \cap C^\perp}$. It will be sufficient to prove that for $\widetilde{x} \in \mathcal{H}(\widetilde{C}) = \widetilde{C} \cap \widetilde{C}^\perp$ then also $x \in \mathcal{H}(C)$.

Since,$\widetilde{x} \in \widetilde{C}^\perp$ we have that for all $\widetilde{y} \in \widetilde{C}$ it holds $\langle \widetilde{x}, \widetilde{y}\rangle = 0$. However, from the proof of Proposition 6 we have that,

$$\left.\begin{array}{lll} \langle \widetilde{x}, \widetilde{y}\rangle_{\mathrm{E}} = -\langle x, y\rangle_{\mathrm{E}} & \text{for} & q = 3 \\ \langle \widetilde{x}, \widetilde{y}\rangle_{\mathrm{H}} = -\langle x, y\rangle_{\mathrm{H}} & \text{for} & q = 4 \end{array}\right\} \implies \begin{array}{lll} \langle x, y\rangle_{\mathrm{E}} = 0 & \text{for} & q = 3 \\ \langle x, y\rangle_{\mathrm{H}} = 0 & \text{for} & q = 4 \end{array}$$

This implies that $\forall\ y \in C$ we have $\langle x, y\rangle = 0$ when $\langle , \rangle$ is the Euclidean or the Hermitian inner product over $\mathbb{F}_3$ or $\mathbb{F}_4$, respectively, therefore $x \in C^\perp$. Moreover, $\widetilde{x}$ is also in $\widetilde{C}$ and hence $x \in C$. This means that $x \in \mathcal{H}(C)$ and the proof is concluded. □

The previous result is of importance in terms of computing the various properties of the hull of the closure, since for $\mathbb{F}_3$ and $\mathbb{F}_4$ (equipped with $\langle , \rangle_{\mathrm{H}}$) these can be derived directly from the respective properties of the closure of the hull. In particular, from Remark 2 and Proposition 7 we obtain the following result which relates the weight enumerator of a code with the one of the hull of its closure.

Corollary 3 *Let $C \subseteq \mathbb{F}_q^n$ with closure $\widetilde{C} \subseteq \mathbb{F}_q^{(q-1)n}$ where $q \in \{3, 4\}$. Then if $\dim(\mathcal{H}(C)) = h$ then $\dim(\mathcal{H}(\widetilde{C})) = h$ and $\mathcal{W}_X(\mathcal{H}(\widetilde{C})) = X^{q-1}\mathcal{W}_X(\mathcal{H}(C))$.*

Invariants and signatures from the hull of the closure. It is worth mentioning that permutationally equivalent closures have permutationally equivalent hulls. In particular, the following result is implied.

Corollary 4 *For any invariant $\mathcal{R}$ and closure $\widetilde{C} \subseteq \mathbb{F}_q^{(q-1)n}$, the mapping $\widetilde{C} \mapsto \mathcal{R}(\mathcal{H}(\widetilde{C}))$ is an invariant.*

Then, a signature for an extension of $\mathcal{SSA}$ can be built from the weight enumerator of the $\mathcal{H}(\widetilde{C})$. Following the notations of §3.3 we derive the following signature for an extension of $\mathcal{SSA}$ through its closure $\widetilde{C}$ as follows,

$$S(\widetilde{C}, i) = \left(\beta, \mathcal{W}_X(\widetilde{C}_{\beta \cdot i} \cap \widetilde{C}^{\perp})\right)$$

where for all $\beta \in \mathbb{F}_q$, we define $C_{\beta \cdot i}$ as the code C in which the i-th coordinate is multiplied by β and $\mathcal{W}_X(\widetilde{C})$ denotes the (Hamming) weight enumerator of the closure. Due to Corollary 3 this signature will have the same discrimancy as if it was used for the original code and in addition we can derive the spectrum values for the closure by performing the computation of the weight distribution of the code.

The extension to $\mathbb{F}_3$ *and* $\mathbb{F}_4$*.* The LCE and CLCE problems can be decided (and computed) in polynomial time using $\mathcal{SSA}$ only in $\mathbb{F}_3$ and $\mathbb{F}_4$, as long as the hull of the given code is small (the worst-case being a weakly self-dual code), using the results of Theorem 1 as these are illustrated in the following diagram.

$$\begin{array}{ccc} C & \xrightarrow{\Psi=(\upsilon;\sigma)} & C' \\ \downarrow & & \downarrow \\ \widetilde{C} & \xrightarrow{\pi=(\sigma\star\upsilon)} & \widetilde{C}' \\ \downarrow & & \downarrow \\ \mathcal{H}(\widetilde{C}) & \xrightarrow{\pi=(\sigma\star\upsilon)} & \mathcal{H}(\widetilde{C}') \end{array}$$

The CLCE problem for the codes C and C' can be solved if we can retrieve the linear isometry $(\upsilon; \sigma)$ from the monomial permutation of the hull of closures $\widetilde{C}$ and $\widetilde{C}'$. This is ensured from Theorem 1 and Corollary 4. Moreover, the $\mathcal{SSA}$ algorithm given in Table 1 can easily be extended for $\mathbb{F}_3$ and $\mathbb{F}_4$ by using the generator matrices of the closures and operating on a larger index set. In particular, we replace $\mathcal{I}_n$ with $\mathcal{I}_n \times \mathbb{F}_q^*$ and the `SSA_extension` presented below in Table 3 is a combined version of the `SSA` and `SSA_group` algorithms presented in Tables 1 and 2, respectively.

We note that, the permutation group of the closure is always non-trivial over $\mathbb{F}_q$ since it contains at least $(q-1)$ monomial permutations. In practice, the output partitions of `SSA_closure` could not be finer than the orbits of PAut($\widetilde{C}$ thus it was necessary the process in `SSA_extension` using the mechanics of the `SSA_group` until the group vanishes. Finally, if we manage to build a fully discriminant signature after a logarithmic number of steps the desired monomial permutation for the closures will be obtained with its corresponding values retrieved from the partitions. There is one additional step needed, to solve the CLCE problem which is converting the retrieved monomial permutation to the original linear isometry. This can be done easily using Lemma 4.

Table 3 An abstract extension of $\mathcal{SSA}$ using the closure

function `SSA_extension`
`input:` $\widetilde{G}, \widetilde{G}' \in \mathbb{F}_q^{k\times(q-1)n}$
`output:` two partitions of $\mathcal{I}_n \times \mathbb{F}_q^*$ with an identical index set
  $\mathcal{P}, \mathcal{P}' \leftarrow$ `SSA_closure`$(\widetilde{G}, \widetilde{G}')$
  `if` $\mathcal{P} \equiv \mathcal{P}'$ `and` $|\mathcal{P}| < n$
    $s \leftarrow \{s \in F \mid \mathcal{P}_s \neq \emptyset\}$ *// $\mathcal{P}, \mathcal{P}'$ both indexed by F*
    $i \leftarrow \mathcal{P}_s$; $j \leftarrow \mathcal{P}'_s$
    $\mathcal{P}, \mathcal{P}' \leftarrow$ `SSA_extension`$(\widetilde{G}_i, \widetilde{G}'_j)$ *// column i or j resp. is set to zero*
  `return` $\mathcal{P}, \mathcal{P}'$

function `SSA_closure`
`input:` $\widetilde{G}, \widetilde{G}' \in \mathbb{F}_q^{k\times(q-1)n}$
`output:` two partitions of $\mathcal{I}_n \times \mathbb{F}_q^*$ with an identical index set
  $\mathcal{P} \leftarrow$ `SSA_step`$(\widetilde{G})$; $\mathcal{P}' \leftarrow$ `SSA_step`$(\widetilde{G}')$
  `while` $\mathcal{P} \equiv \mathcal{P}'$ `and` $|\mathcal{P}| < n$ *// repeat at most $O(\log(q-1)n)$ times*
    // both $\mathcal{P}$ and $\mathcal{P}'$ are indexed by the same set F
    $s \leftarrow \{s \in F \mid \mathcal{P}_s \neq \emptyset\}$ *// at random or according to some heuristic*
    $\mathcal{P} \leftarrow$ `SSA_refine`$(\widetilde{G}, \mathcal{P}, s)$; $\mathcal{P}' \leftarrow$ `SSA_refine`$(\widetilde{G}', \mathcal{P}', s)$
  `return` $\mathcal{P}, \mathcal{P}'$

The functions `SSA_step` and `SSA_refine` are used as in Table 1 for the closure and $\mathcal{I}_n \times \mathbb{F}_q^*$ and are not detailed here. The algorithm uses as a parameter the signature S defined for the closure $\widetilde{C}$.

Algorithmic complexity for $\mathcal{SSA}$ extension. We conclude this section, by giving the (heuristic) algorithmic complexity of the $\mathcal{SSA}$ extension for $\mathbb{F}_3$ and $\mathbb{F}_4$. This extension of the $\mathcal{SSA}$ is polynomial for $\mathbb{F}_3$ and $\mathbb{F}_4$ but unfortunately exponential for $\mathbb{F}_q$, $q \geq 5$. A thorough algorithmic analysis has been made already in §3.3 and we just mention here that the computational cost of the weight distribution for an $[n, k]$ code over $\mathbb{F}_q$ is proportional to nq^k operations in $\mathbb{F}_q$ and for random instances the average number of operations is proportional to $2n$, see [23]. The closure is a code of length $(q-1)n$ and of the same dimension k as the original code, therefore the number of operations earned are $(q-1)nq^k - nq^k = (q-2)nq^k$ and on average $2(q-1)n - 2n = 2(q-2)n$. Thus for $q \in \{3, 4\}$ the difference is negligible.

Table 4 Heuristic complexity for $\mathcal{SSA}$ and its extension over $\mathbb{F}_q$

Algorithm	Field (alphabet)	Random codes (average-case)	Weakly self-dual codes (worst-case)
$\mathcal{SSA}$	$\mathbb{F}_2$	$\mathcal{O}(n^3)$	$\mathcal{O}(2^k n^2 \log n)$
$\mathcal{SSA}$ extension	$\mathbb{F}_3$	$\mathcal{O}(n^3)$	$\mathcal{O}(3^k n^2 \log n)$
$\mathcal{SSA}$ extension	$\mathbb{F}_4$	$\mathcal{O}(n^3)$	$\mathcal{O}(2^{2k} n^2 \log n)$
$\mathcal{SSA}$ extension	$\mathbb{F}_q$, $q \geq 5$	$\mathcal{O}(q^k n^2 \log n)$	$\mathcal{O}(q^k n^2 \log n)$

6 Hardness of linear code equivalence for $q \geq 5$

As we mentioned earlier, the LCE and CLCE problems can be decided (and computed) in polynomial time using $\mathcal{SSA}$ only in $\mathbb{F}_3$ and $\mathbb{F}_4$, as long as the hull of the given code is small (the worst-case being a weakly self-dual code). It does not seem possible to extend this result to larger alphabet. Even though, the closure was introduced as a mean to provide an invariant for the LCE problem over $\mathbb{F}_q$ it turned out that any invariant built on top of the hull of the closure, is not an easy computable invariant unless $q \in \{3, 4\}$. In addition, recall that the hull of the code is not an invariant for the LCE and SLCE problems except for $\mathbb{F}_3$ and $\mathbb{F}_4$. We conclude by posing the following conjecture.

Conjecture 1 For a given $q \geq 5$, the LCE and CLCE problems over $\mathbb{F}_q$ are hard for almost all instances.

In fact, due to the recent reductions of the LCE problem to the GI problem (see [7] and §7) hard instances had to be expected. However, our conjectured statement is apparently quite stronger which is further supported by a similar negative complexity result due to Dirk Vertigan [27].

Vertigan's result is given for graphs, but, translated for codes, it states that evaluating the (homogeneous) weight enumerator polynomial of a linear code over $\mathbb{F}_q$ for $q \geq 5$ on any point of the complex unit circle is always difficult except for a constant number of trivial points. The evaluation of the weight enumerator in those points essentially provide the code cardinality. There is an additional point easy to evaluate for $q \in \{2, 3, 4\}$. The evaluation in this point essentially provides the cardinality of the hull of the code. For $q = 4$ the hull is defined according to the hermitian inner product. There is possibly more than just a coincidence here, but the connection with code equivalence is not obvious to establish. Doing so would certainly be enlightening. We conclude the section by stating some open problems related to the LCE problem and its hardness.

- Do we already know all easy (computable) invariants for code equivalence?
- (Semi)-linear isometries of $H(n, q)$ that can be expressed as a group action on the set of linear subspaces preserve the weight of codewords and induce natural equivalences for codes. Therefore, from an algorithmic point of view any counting function must be somehow related to the weight of codewords. It is thus intriguing to ask, whether if all invariants are related to the weight enumerator of a code?
- Do there exist similar subcodes of a code, like the hull, which capture the characteristics and inherit the properties of the original code in terms of invariants for the LCE problem?

Last but not least, we would like to note that the apparent hardness of the LCE problem over F_q, for $q \geq 5$ arises from the absence of an easy computable invariant and not the inexistence of an algorithm.

7 Relationships between the various problems

We first remark that, as stated in §4.1, and because there are few field automorphisms, we have a polynomial time reduction between linear code equivalence and semilinear code equivalence problems. This hold for the decisional as well as the computational problems. We thus exclude the semi-linear case and we examine in this section the relationships between the various other code equivalence problem. The arguments we present here are sometimes formal, sometimes less formal, leading to yet unsolved open problems. We start with a discussion about the computation of the automorphism group of a code. Next, the questions we will answer or discuss relate with the relationships between decisional and computational then with the relationships between monomial and permutation code equivalence.

7.1 Computing automorphism groups

Let C denote a q-ary linear code of length n. We denote $\mathrm{PAut}(C)$ its permutation group and $\mathrm{MAut}(C)$ its monomial group (see §2 for definitions).

7.1.1 Permutation group versus monomial group

As previously remarked by Leon in [15], computing the monomial group of a code can be reduced to computing the permutation group of another longer code. In fact, every time the code has no duplicate coordinate (up to a scalar), from Remark 1 any element in the permutation group of its closure is a monomial permutation and monomial permutations are in one-to-one correspondance with linear isometries. When there are duplicate coordinates, they can first be removed, the group computed then adjusted by taking into account the duplicates. We do not detail this here.

7.1.2 Computing permutation groups

The computation of the permutation group of a code was described by Leon [15] based on concepts developed by Sims [25]. Let $\mathcal{G}$ denote subgroup of $\mathcal{S}_n$. A *base* of $\mathcal{G}$ is a subset $B = (b_1, \ldots, b_\ell)$ of $\mathcal{I}_n$ such that at most one element of $\mathcal{G}$ has prescribed values on B. For any i, $1 \leq i \leq \ell$, we denote $B_i = (b_1, \ldots, b_i)$ and $\mathcal{G}_{B_i}$ the subgroup of $\mathcal{G}$ stabilizing B_i. Effective computation in permutation group will use an increasing sequence of group

$$\{1\} = \mathcal{G}_B \subset \mathcal{G}_{B_{\ell-1}} \subset \cdots \subset \mathcal{G}_{B_1} \subset \mathcal{G}.$$

A strong generating set S_B according to B is a set of generators of $\mathcal{G}$ such that for all i, $S_{B_i} = S \cap \mathcal{G}_{B_i}$ is a set of generators of $\mathcal{G}_{B_i}$. A strong generating set is a convenient object which allows the easy computation of all sorts of properties (membership, orbits, ...). The set S_B is computed recursively, and

to expand S_{B_i} into $S_{B_{i-1}}$, it is enough find one or several permutation stabilizing B_{i-1} and such that the union of all orbits of b_i under the action of those permutations is maximal. Table 5 present a simplified version of an algorithm computing the permutation group of a code. To instantiate this algorithm, we

Table 5 A simplified algorithm to compute the permutation group of a code

function `PAut`
`input:` $G \in \mathbb{F}_q^{k\times n}$, $B \subset \mathcal{I}_n$
`output:` a strong generating set according to B
    `if` $B = \mathcal{I}_n$ `then return` $\{1\}$
    $i \leftarrow \mathcal{I}_n \setminus B$ *// chosen arbitrarily*
    $S \leftarrow$ `PAut`$(G, B \cup \{i\})$
    `for` $j \in \mathcal{I}_n \setminus (B \cup \{i\})$
(*) $\sigma \leftarrow \{\sigma \in \mathrm{PAut}(\langle G\rangle) \mid \sigma(i) = j, \sigma(b) = b, \forall b \in B\}$
        `if` $\sigma \neq$ FAIL and $\sigma \notin \langle S\rangle$ `then`
            $S \leftarrow S \cup \{\sigma\}$
    `return` S

In practice the stopping condition and the control of the `for` loop can be improved, but the general idea is that after picking a polynomial number of permutations in step (*) the algorithm returns a strong generating set of the permutation group. The set returned by the call `PAut`(G, B) is a strong generating set of the stabilizer of B in $\mathrm{PAut}(\langle G\rangle)$. The initial call will be `PAut`$(G, \emptyset)$.

need to specify the step (*) which is indeed the difficult part. If this can be achieved accurately and efficiently the algorithm will be efficient, else ...

7.1.3 Reducing the computation of permutation groups to the computational permutation code equivalence problem?

We use the notations of Table 5 and denote $C = \langle G\rangle$ the target code. All elements of the set $\{\sigma \in \mathrm{PAut}(C) \mid \sigma(i) = j, \sigma(b) = b, \forall b \in B\}$ in which we wish to pick σ are solutions to $\mathrm{CPE}(C_{B\cup\{i\}}, C_{B\cup\{j\}})$ the computational permutation code equivalence problem applied on C punctured in $B\cup\{i\}$ and C punctured in $B\cup\{j\}$. This would hold also with shortened codes or for any sequence of puncturing and shortening, as long as the sequence is the same for both arguments of CPE. Unfortunately, it may happen that there are other solutions to $\mathrm{CPE}(C_{B\cup\{i\}}, C_{B\cup\{j\}})$ and thus replacing the (*) step with

$$(**) \qquad \sigma \leftarrow \mathrm{CPE}(C_{B\cup\{i\}}, C_{B\cup\{j\}})$$

will not work for all code and thus do not provide a reduction. For most codes, the permutation group of the code punctured in i will be equal to the stabilizer of $\{i\}$. However the permutation group may very well increase when the code is punctured.

It is possible that a more elaborate call to CPE, or a polynomial number of such calls, allows the computation of a suitable σ. We found no solution though and so far a polynomial reduction from CPE the PAut remains an open problem.

7.2 Decision *vs.* computation

If two codes of length n are equivalent for some isometry Ψ, with $\sigma = \mathrm{perm}(\Psi)$, then for all $i \in \mathcal{I}_n$ the codes punctured (or shortened) in i and $j = \sigma(i)$ are also equivalent. This holds for any notion of equivalence (permutation, linear, semi-linear). To exploit this property, we may wish to use the algorithm of

Table 6 Reduction from decisional code equivalence to computational code equivalence

Parameter: an oracle `equiv` which tells us whether its arguments span equivalent codes. It can be defined for any form of equivalence (permutation, linear, semilinear).

---

function `find_perm`
`input:` $G, G' \in \mathbb{F}_q^{k\times n}$
  `if not equiv`(G, G') `then return` NOT EQUIVALENT
  `for all` $(i, j) \in \mathcal{I}_n^2$
    `if equiv`(G_i, G'_j) *// in fact testing equivalence of punctured codes*
      $\sigma(i) \leftarrow j$; $c_i \leftarrow c_i + 1$ *// $c_i = 0$ initially*
  `if` $\forall i, c_i = 1$ `then return` σ `else return` FAIL

Table 6 which tries to find matching positions in the two codes. Note that for one call to `find_perm` only a polynomial number of calls to `equiv` (at most $n^2 + 1$) have to be performed. The principle of Table 6 is used in $\mathcal{SSA}$ and works very in many situation. In fact, for any code verifying the following property, we can guaranty the good behavior of the reduction.

Property 1 Let C be a q-ary linear code of length n. The property $P_1(C)$ holds if for all $i \neq j$ in $\mathcal{I}_n$, we have $C_i \not\sim C_j$.

Proposition 8 *We consider an intance of the algorithm of Table 6 with inputs G and G'. We denote $C = \langle G \rangle$ and $C' = \langle G' \rangle$.*

1. *The* `equiv` *parameter in Table 6 is an oracle for permutation equivalence. If $C \overset{\mathbf{PE}}{\sim} C'$ and $P_1(C)$ holds, then $P_1(C')$ holds and* `find_perm` *returns a permutation σ such that $\sigma(C) = C'$.*
2. *The* `equiv` *parameter in Table 6 is an oracle for linear equivalence. If $C \overset{\mathbf{LE}}{\sim} C'$ and $P_1(C)$ holds, then $P_1(C')$ holds and* `find_perm` *returns a permutation σ such that $\Psi(C) = C'$ with* $\mathrm{perm}(\Psi) = \sigma$.

Proof First, we remark that if $C' = \psi(C)$ for some isometry Ψ with $\sigma = \mathrm{perm}(\Psi)$ then for all i we have $C_i \sim C'_{\sigma(i)}$. It follow that if either $P_1(C)$ or $P_1(C')$ holds, the other also holds. We use below the noations of Table 6.

1. Let $C' = \pi(C)$. For all i, we have $C_i \sim C_{\pi(i)}$ thus $c_i \geq 1$. If we had $C_i \sim C'_j$ for some $j \neq \pi(i)$ then we would also have $C_i \sim C_{\pi^{-1}(j)} \sim C'_j$ and $P_1(C)$ would not hold. Thus $c_i = 1$ for all i and the algorithm returns the permutation $\sigma = \pi$.

2. Let $C' = \Psi(C)$. The proof is identical, we must have $c_i = 1$ for all i and the permutation returned by the algorithm can only be $\mathrm{perm}(\Psi)$.

Finally, note that if $C' = \Psi(C)$ and $\sigma = \mathrm{perm}(\Psi)$ recovering the isometry Ψ can be achieved in polynomial time (see Proposition 3).

The Property 1 does not hold for all codes. In particular, when its automorphism group is non trivial, the code C punctured in i is equivalent the code C punctured on any element of the orbit of i under the action of $\mathrm{MAut}(C)$. We briefly consider this case in the next section.

It is worth noting however that even with a trivial group a code may not verify the Property 1. There are various interesting questions related to that. For instance, what are exactly the codes which codes do not verify the property? If we can describe them, can we adjust the reduction and keep it polynomial?

7.2.1 Non trivial automorphism group

When the automorphism group is non trivial, it is probably also possible to obtain a (limited) reduction from the decisional to the computational problem. When a code is punctured, its automorphism group will often reduce to the subgroup which stabilizes this position. If we limit the reduction to codes that behave well in that respect we may find a sequence of puncturing (in the spirit of Table 2 for the $\mathcal{SSA}$ when we are confronted to a pair of codes) such that the automorphism group gradually vanishes. At this point, we may apply the reduction of Table 6.

7.3 Linear code equivalence *vs.* permutation code equivalence

Because of the properties of the closure, namely Lemma 4, there is a reduction from LCE to PCE for both the decisional and the computational problems. The reduction in the other way does not seem to be easy to obtain and remains an open problem.

This is very paradoxal. On one hand it seems that for $q \geq 5$, solving the linear code equivalence problem is hard of almost all instances while solving the permutation equivalence problem is easy for almost all instances. On the other hand, when we consider the worst-case complexity, the linear code equivalence is provably not harder than the permutation equivalence problem. Somehow, the closure seems to map linear equivalence problems precisely onto some of the few hard instances of the permutation equivalence problem (namely weakly self-dual codes).

Finally, note that the reduction from LCE to PCE leads by transitity to a proof that the Graph Isomorphism (GI) problem is not harder than the Linear Code Equivalence (LCE) problem. This last reduction was already obtained, but only recently, by Thomas Feulner [7] with a different approach.

8 Conclusion

In this paper, we explored the hardness of the (Computational) Linear Code Equivalence problem(s) over $\mathbb{F}_q$. We showed that an extension of $\mathcal{SSA}$ for solving the latter problems when $q \in \{3, 4\}$ is possible, in (almost) polynomial time, however for $q \geq 5$ its complexity growth becomes exponential for all instances. Moreover, we conjectured that, for $q \geq 5$, the computational and decisional version of linear code equivalence are hard for almost all instances. Our argument, is supported by some impossibility results on the Tutte polynomial of a graph which corresponds to the weight enumerator of a code. On the bright side, the negativity of our claim, might lead to some interesting features for applications. For example, in cryptography, zero-knowledge protocols have been designed in the past, based on the hardness of the Permutation Code Equivalence problem [11]. Moreover, the relation of the automorphism groups of the code and its closure might be of cryptographic interest. The context of the framework built in [5] suggests that codes with large automorphism groups resist quantum Fourier sampling as long as permutation equivalence is considered. It would thus be intriguing to investigate, if this result can also be extended for the linear and semilinear code equivalence.

Acknowledgements The authors are grateful to the reviewers for their comments and suggestions that improved the quality and the presentation of the paper. The work of the second author was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 246016.

References

1. Babai, L., Codenotti, P., Grochow, J.A., Y.Qiao: Code equivalence and group isomorphism. In: Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '11, pp. 1395–1408. SIAM (2011)
2. Betten, A., Braun, M., Fripertinger, H., Kerber, A., Kohnert, A., Wassermann, A.: Error-Correcting Linear Codes: Classification by Isometry and Applications, *Algorithms and Computation in Mathematics*, vol. 18. Springer, Berlin, Heidelberg (2006)
3. Bouyukliev, I.: About the code equivalence. Ser. Coding Theory Cryptol. **3**, 126–151 (2007)
4. Danielsen, L.E., Parker, M.G.: Edge local complementation and equivalence of binary linear codes. Des. Codes Cryptography **49**, 161–170 (2008)
5. Dinh, H., Moore, C., Russell, A.: Mceliece and niederreiter cryptosystems that resist quantum fourier sampling attacks. In: Proceedings of the 31st annual conference on Advances in cryptology, CRYPTO'11, pp. 761–779. Springer-Verlag, Berlin, Heidelberg (2011)
6. Feulner, T.: The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes. Adv. Math. Commun. **3**, 363–383 (2009)
7. Feulner, T.: Kanonische reprsentanten äquivalenter codes computergesttzte berechnung und ihre anwendung in der codierungstheorie, kryptographie und geometrie. Ph.D. thesis, University of Bayreuth (2013)
8. Fripertinger, H.: Enumeration of linear codes by applying methods from algebraic combinatorics. Grazer Math. Ber. **328**, 31–42 (1996)

9. Fripertinger, H.: Enumeration of the semilinear isometry classes of linear codes. Bayrether Mathematische Schriften **74**, 100–122 (2005)
10. Fripertinger, H., Kerber, A.: Isometry classes of indecomposable linear codes. In: Proceedings of the 11th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, AAECC-11, pp. 194–204. Springer-Verlag, London, UK (1995)
11. Girault, M.: A (non-practical) three-pass identification protocol using coding theory. In: J. Seberry, J. Pieprzyk (eds.) Advances in Cryptology AUSCRYPT '90, *Lecture Notes in Computer Science*, vol. 453, pp. 265–272. Springer Berlin Heidelberg (1990)
12. Grochow, J.A.: Matrix lie algebra isomorphism. Tech. Rep. TR11-168, Electronic Colloquium on Computational Complexity (2011). Also available as arXiv:1112.2012. To appear, IEEE Conference on Computational Complexity, 2012.
13. Huffman, W.C.: Codes and groups. In: V. Pless, W.C. Huffman (eds.) Handbook of Coding Theory, pp. 1345–1440. Elsevier, North-Holland, Amsterdam (1998)
14. Kaski, P., Östergård, P.R.J.: Classification Algorithms for Codes and Designs, *Algorithms and Computation in Mathematics*, vol. 15. Springer, Berlin, Heidelberg (2006)
15. Leon, J.S.: Computing automorphism groups of error-correcting codes. IEEE Trans. Inform. Theory **28**, 496–511 (1982)
16. Lidl, R., Niederreiter, H.: Finite Fields, *Encyclopedia of Mathematics and its Applications*, vol. 20, 2nd edn. Cambridge University Press (1997)
17. MacWilliams, F.J.: Error-correcting codes for multiple-level transmission. Bell. Syst. Tech. J. **40**, 281–308 (1961)
18. McEliece, R.J.: A public-key cryptosystem based on algebraic coding theory. Tech. Rep. DSN Progress Report 42–44, California Institute of Technology, Jet Propulsion Laboratory, Pasadena, CA (1978)
19. Östergård, P.R.J.: Classifying subspaces of hamming spaces. Des. Codes Cryptography **27**, 297–305 (2002)
20. Petrank, E., Roth, R.M.: Is code equivalence easy to decide? IEEE Trans. Inform. Theory **43**, 1602–1604 (1997)
21. Rains, E.M., Sloane, N.J.A.: Self-dual codes. In: V. Pless, W.C. Huffman (eds.) Handbook of Coding Theory, pp. 177–294. Elsevier, North-Holland, Amsterdam (1998)
22. Sendrier, N.: On the dimension of the hull. SIAM J. Discrete Math. **10**(2), 282–293 (1997)
23. Sendrier, N.: Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Trans. Inform. Theory **26**, 1193–1203 (2000)
24. Sendrier, N., Simos, D.E.: The hardness of code equivalence over $\mathbb{F}_q$ and its application to code-based cryptography. In: PQCrypto '13: Proceedings of the Fifth International Conference on Post-Quantum Cryptography, Lecture Notes in Computer Science, vol. 7932, pp. 203–216 (2013)
25. Sims, C.: Computational methods in the study of permutation groups. In: J. Leech (ed.) Computational Problems in Abstract Algebra, pp. 169–183. Pergamon Press (1970)
26. Skersys, G.: Calcul du groupe d'automorphisme des codes. détermination de l' equivalence des codes. Thèse de doctorat, Université de Limoges (1999)
27. Vertigan, D.: Bicycle dimension and special points of the Tutte polynomial. Journal of Combinatorial Theory, Series B **74**, 378–396 (1998)