How easy is code equivalence over F_q?

Nicolas Sendrier, Simos Dimitrios

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

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

*Extended Abstract

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 [25] 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 explore 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.

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 [22] that the worst-case was not easier than for the Graph Isomorphism problem. On the other hand, the support splitting algorithm [25] 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.

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.

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 [18]. 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. [3,14]). 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 [3,9,14]. 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 [3,8,10], 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 $\mathbb{F}_q$.
Since every linear code can be represented with a generator matrix, we 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 $(\upsilon; (\alpha, \sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$ such that $\langle G'\rangle = (\upsilon; (\alpha, \sigma))(\langle G\rangle)$.

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 $(\upsilon; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ such that $\langle G'\rangle = (\upsilon; \sigma)(\langle G\rangle)$.

Problem 6 (Computational Permutation Code Equivalence2 (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 [22], 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 [22] was generalized in [12] over any field $\mathbb{F}_q$, hence the PCE problem is harder than the GI problem, for any field $\mathbb{F}_q$. 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 [19] is related to the hardness of permutationally equivalent binary linear codes. Towards this direction, another important complexity result was shown in [6], that the Hidden Subgroup problem also reduces to the PCE problem for any field $\mathbb{F}_q$.

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 [5] 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 [20], 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 [2], 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 [4], 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 [7]. 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 $\mathbb{F}_q$.

The support splitting algorithm can be used as an oracle to decide whether two binary codes are permutationally equivalent [25], as well as to retrieve the equivalence mapping. The main idea is to partition the support $\mathcal{I}_n$ of a code $C \subseteq \mathbb{F}_2^n$, into small sets that are fixed under operations of $\mathrm{PAut}(C)$. The algorithm employs the concept of invariants and signatures, defined below.

Let $\mathcal{L}_{n,k}$ denote the set of all linear codes of length n and dimension k, and let $\mathcal{L} = \bigcup_{n,k>0} \mathcal{L}_{n,k}$ be the set of all such codes.

Definition 4 An invariant $\mathcal{R}$ over a set E is defined to be a mapping $\mathcal{R} : \mathcal{L} \mapsto E$ such that any two permutation equivalent codes take the same value, i.e. if $C \overset{\mathbf{PE}}{\sim} C' \Longrightarrow \mathcal{R}(C) = \mathcal{R}(C')$.

For instance, the Hamming weight enumerator is an invariant over the polynomials with integer coefficients. Applying an invariant, for instance the weight enumerator, may help us to decide whether two codes are equivalent or not.

Definition 5 A signature S over a set F maps a code $C \subseteq \mathbb{F}_q^n$ and an element $i \in \mathcal{I}_n$ into an element of F and is such that for all $\sigma \in \mathcal{S}_n$, $S(C, i) = S(\sigma(C), \sigma(i))$. Moreover, S is called discriminant for C if there exist $i, j \in \mathcal{I}_n$ such that $S(C, i) \neq S(C, j)$ and fully discriminant if this holds $\forall\ i, j \in \mathcal{I}_n$.

If S is fully discriminant for C, and $C' = \sigma(C)$ for $\sigma \in \mathcal{S}_n$, we are able to retrieve σ. The support splitting algorithm ($\mathcal{SSA}$) takes as an argument a generator matrix G for a code C and returns a labeled partition $\Pi = \{(\Pi_j, j)\}_{j \in \mathcal{I}_n}$ of the code support. For any two linear codes C and C' with generator matrices G and G', let $\mathcal{SSA}(G) = \{(\Pi_j, j)\}_{j \in \mathcal{I}_n}$ and $\mathcal{SSA}(G') = \{(\Pi'_j, j)\}_{j \in \mathcal{I}_n}$. The fundamental property of $\mathcal{SSA}$ is that if

$$C' = \sigma(C) \quad \Longrightarrow \quad \forall j \in \mathcal{I}_n, \quad \Pi'_j = \sigma(\Pi_j) \tag{2}$$

and implies in particular that the output of $\mathcal{SSA}$ is independent of the choice of G. The converse of relation (2) is not necessarily true, but satisfied in practice under the assumption that the cells of the output of $\mathcal{SSA}$ achieve the orbits of the elements of the code support w.r.t. the action of $\mathrm{PAut}(C)$ and constitute its finest obtainable partition [17,25]. The main difficulty of the algorithm, is to obtain a fully discriminant signature, for as many codes as possible. In [25] it was shown that such a signature, can be built from the weight enumerator of the hull of a code C, denoted by $\mathcal{H}(C)$, and defined as the intersection of the code with its dual, $\mathcal{H}(C) = C \cap C^{\perp}$ [1], because the hull commutes with permutations[5], $\mathcal{H}(\sigma(C)) = \sigma(\mathcal{H}(C))$, and therefore it is an invariant for permutation equivalence. The (heuristic) complexity of $\mathcal{SSA}$ for an $[n, k]$ code C is $\mathcal{O}(n^3 + 2^h n^2 \log n)$ where h is the dimension of the hull [21,25]. In practice, for random codes, the hull has a small dimension with overwhelming probability [24] and the dominant cost for the average case is $\mathcal{O}(n^3)$. Note that, the worst case occurs when the hull dimension is maximal; weakly self-dual codes ($C \subset C^{\perp}$) are equal to their hulls. Then the algorithm becomes intractable with a complexity equal to $\mathcal{O}(2^k n^2 \log n)$.

[5] No such property exists in general for linear codes when (semi)-linear equivalence is considered, see also lemma 2.

3.3 Computational vs. decisional code equivalence

The purpose of this section is to exhibit the relation of the worst-case complexities of the CPCE, CLCE and CSLCE problems. If one can explicitly compute the latter problems then one can also solve its corresponding decisional versions. We argue that the other direction is also possible, that is provided we have access to an oracle for deciding the PCE, LCE and SLCE problems we can build an algorithm for computing the CPCE, CLCE and CSLCE problems, respectively. Therefore, the computational and decisional problems related to code equivalence belong in the same complexity class.

In the case of the PCE problem, the oracle used is an abstract version of $\mathcal{SSA}$ denoted by $\mathbf{Or_{PCE}}(G, G') \in \{\textsc{True}, \textsc{False}\}$. This oracle takes as input two generator matrices G and G' of two q-ary linear codes and is ideal, in the sense that it always return true if the generator matrices span permutationally equivalent codes, and false otherwise. It is well known, that punctured codes of equivalent codes remain equivalent, when the first are punctured in the same position. For our result, we impose a stronger condition and state without proof the following lemma.

Lemma 1 *Let G and G' span two $[n, k]$ linear codes C and C' over $\mathbb{F}_q$. If $\mathbf{Or_{PCE}}(G, G')$ is* True *and $\mathbf{Or_{PCE}}(G_i, G'_j)$ is* True *for some $i, j \in \mathcal{I}_n$ then there exists $\sigma \in \mathcal{S}_n$ such that $C' = \sigma(C)$ and $j = \sigma(i)$.*

The previous lemma is the keystone for proving that the computational and decisional version of the permutation code equivalence are equally hard. A similar lemma can be stated for the LCE or SLCE problem where the previous oracle can be used as a building block of an algorithm that retrieves the permutational part of the linear or semilinear isometry of the CLCE and CSLCE problems. Hence, the computational and decisional problems of code equivalence are not essentially different. Now, consider an algorithm that solves the CSLCE or CLCE problem to which is given an instance of the CPCE problem. Due to the previous discussion, the expected output of the algorithm, a linear or semilinear isometry, is just a permutation.

3.4 The Support Splitting Algorithm

The support splitting algorithm ($\mathcal{SSA}$) is intended to solve the computational permutation 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 6
- 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 algorithm is given in Table 1. It takes

Table 1 An abstract version of the support splitting algorithm [25]

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:` $G, G' \in \{0,1\}^{k\times n}$
`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:` $G \in \{0,1\}^{k\times n}$
`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:` $G \in \{0,1\}^{k\times n}$, $\mathcal{P}$ 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 $\mathcal{P} \times \mathcal{Q}$

as parameter a signature S. The behavior of the algorithm can be predicted regardless of S.

Proposition 1 *Let $G, G' \in \{0,1\}^{k\times n}$, $C = \langle G\rangle$, 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 $\mathcal{P} \not\equiv \mathcal{P}'$ the codes C and C' are not permutation equivalent. Else, if $C' = \sigma(C)$, we must have $\mathcal{P}'_s = \sigma(\mathcal{P}_s)$ for all $s \in F$.*

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 [25] 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

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

where $\mathcal{W}(C)$ denotes the Hamming weight enumerator of a code C. This signature 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 [25].

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

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

function `SSA_extended`
`input:` $G, G' \in \{0,1\}^{k \times n}$
`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_extended`(G_i, G'_j) *// column i or j resp. is set to zero*
  `return` $\mathcal{P}, \mathcal{P}'$

The singletons $\{i\}$ and $\{j\}$ may have to be reintroduced into $\mathcal{P}$ and $\mathcal{P}'$ with the same fresh index after the recursive call to `SSA_extended`, for the sake of readability we do not detail that.

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_extended` will stop as soons as the set of all indices i chosen to puncture the code form a base of the permutation group (in the sense of [26]). 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.

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 the elementary function `SSA_step` is logarithmic in the code length. The algorithm first computes the hull of the code $C \cap C^{\perp}$, once this is done computing one signature involves very little linear algebra (adjusting the hull one dimension up or down) and a weight enumerator of some code of dimension close to that of the hull. To keep things simple:

- If the input codes have a hull of small dimension, the cost for computing $C \cap C^{\perp}$ dominates and the total cost does not exceed $O(n^3)$ elementary operations in $\mathbb{F}_q$.
- If the input codes have a hull of large dimension, the computation of the weight enumerator dominates and the total cost is $O(q^h n^2 \log n)$ elementary operations in $\mathbb{F}_q$ where h is the hull dimension.

Because random linear codes have a very small hull (see [24]) the $\mathcal{SSA}$ runs (heuristically) in polynomial time on average.

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.

4 Generalization of the Support Splitting Algorithm to Linear and Semi-linear Equivalence

4.1 Semi-linear Equivalence *vs.* Linear 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 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 2 *Let C and C' be two q-ary linear $[n,k]$ code which admits systematic generator matrices of the form*

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

where i ranges from 1 to k and j from $k+1$ to n. We have $C' = (v,1)(C)$ if and only if $v = (v_1, \ldots, v_n)$ is a non-zero solution of the system

$$v_i g'_{i,j} = v_j g_{i,j}, 1 \le i \le k, k < j \le n.$$

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 $C' = (v,1)(C)$.

4.3 Invariants and Signatures

Invariants and signatures were used above in the context of permutation equivalence. The notion of invariant generalizes naturally to any kind of equivalence

$$C \sim C' \Rightarrow \mathcal{R}(C) = \mathcal{R}(C')$$

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 [27].

Definition 7 Let $\mathbb{F}_q = \{a_0, a_1, \ldots, a_{q-1}\}$, with $a_0 = 0$, and a linear code $C \subseteq \mathbb{F}_q^n$. Define $\mathcal{I}_{q-1}^{(n)}$ as the cartesian product of $\mathcal{I}_{q-1} \times \mathcal{I}_n$. The closure $\widetilde{C}$ of the code C is a code of length $(q-1)n$ over $\mathbb{F}_q$ where,

$$\widetilde{C} = \{(a_k x_i)_{(k,i)\in\mathcal{I}_{q-1}^{(n)}} \mid (x_i)_{i\in\mathcal{I}_n} \in C\}.$$

Clearly, we see that every coordinate of the closure $\widetilde{C}$, corresponds to a coordinate position of a codeword of C multiplied by a nonzero element of $\mathbb{F}_q$. Since, the index $(k,i) \in \mathcal{I}_{q-1}^{(n)}$ of a position of a codeword of the closure means that $k \in \mathcal{I}_{q-1}$ and $i \in \mathcal{I}_n$, we have taken into account every possible multiplication of x_i with nonzero elements of $\mathbb{F}_q$, and it is easy for someone to show[6] the following:

Theorem 1 *Let $C, C' \subseteq \mathbb{F}_q^n$. If C and C' are linearly equivalent, i.e. $C \overset{\mathbf{LE}}{\sim} C'$ then $\widetilde{C}$ and $\widetilde{C'}$ are permutationally equivalent, i.e. $\widetilde{C} \overset{\mathbf{PE}}{\sim} \widetilde{C'}$.*

Theorem 1 is of great importance, because it realizes a reduction from the LCE problem to the PCE problem. Thus, we are able to decide if the codes C and C' are linearly equivalent by checking their closures for permutation equivalence. Moreover, if the closures are permutation equivalent there might be an algorithmic procedure that will allow us to recover the initial isometry between C and C'. However, as we shall see shortly after, 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 $\mathbb{F}_q$, $q \geq 5$.

We would also like to mention that this representation of the closure is not unique. In particular, it depends on a lexigographical ordering of $\mathbb{F}_q^*$.

For example, the ordering $(a_1, 1) < \ldots < (a_1, n) < (a_2, 1) < \ldots (a_2, n) < \ldots < (a_{q-1}, 1) < \ldots < (a_{q-1}, n)$ gives a total order for $\mathcal{I}_{q-1}^{(n)}$, and gives rise to the following closure,

$$\widetilde{C} = \{(a_1 x_1, \ldots, a_1 x_n, \ldots, a_{q-1} x_1, \ldots, a_{q-1} x_n) \mid (x_1, \ldots, x_n) \in C\}.$$

Note that, such an ordering can always be induced by a permutation of the symmetric group $\mathcal{S}_{\mathbb{F}_q^*}$ acting on $\mathbb{F}_q$ defined as $\mathcal{S}_{\mathbb{F}_q^*} := \{\rho \mid \rho : \mathbb{F}_q \to \mathbb{F}_q, \rho \text{ is a bijection and } \rho(0) = 0\}$.

Moreover, it is natural to ask which permutations can appear as permutations of the closures since $\mathcal{SSA}$ was designed exactly to retrieve the permutation between equivalent codes. If we assume that we are given a primitive element γ of $\mathbb{F}_q$, it is well-known that all of its permissible powers generate the multiplicative group of $\mathbb{F}_q = \{\gamma, \gamma^2, \ldots, \gamma^{q-2}, \gamma^{q-1} = 1\}$. Then an ordering according to a cyclic shift of a power of γ will produce a unique closure for the code C (consider the row echelon form on two generator matrices of the closures produced by such orderings).

[6] The detailed proof of this theorem and all subsequent results will appear in an extended version of this paper.

Since, such a closure can always be reached by a composition of permutations of $\mathcal{S}_{\mathbb{F}_q^*}$, we define a systematic form for the closure as follows,

$$\widetilde{C}_{\mathbf{sys}} = \{(x_1, \gamma x_1 \ldots, \gamma^{q-2} x_1, \ldots, x_n, \gamma x_n \ldots, \gamma^{q-2} x_n) \mid (x_1, \ldots, x_n) \in C\}.$$

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 and denoted by $\mathcal{S}(q-1, n)$. Its order is $(q-1)^n n!$ and its elements are exactly those permutations that can appear as permutations of permutationally equivalent closures. This reasoning is sufficient for one to show that for $C \subseteq \mathbb{F}_q^n$ we have $\mathrm{MAut}(C)$ to be isomorphic to $\mathrm{PAut}(\widetilde{C}) \cap \mathcal{S}(q-1, n)$.

Moreover, it further implies that the converse of theorem 1 also holds, and by involving the systematic forms of the closures as an intermediate step, after a non-trivial proof, we can show the following relation for equivalent codes and their closures.

Theorem 2 *Let $C, C' \subseteq \mathbb{F}_q^n$. 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. $\widetilde{C} \overset{\mathbf{PE}}{\sim} \widetilde{C'}$.*

6 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. [23]). Note that, for $x, y \in \mathbb{F}_q$,
$$(x + y)^{\sqrt{q}} = x^{\sqrt{q}} + y^{\sqrt{q}}, \quad x^q = x.$$

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 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, [24], except in the cases where C is also a weakly self-dual code.

It is worth mentioning that these are exactly the same cases where the hull of a code could be used as an invariant for (semi)-linear equivalence, because the duals of linear and semilinear codes remain equivalent with the same isometry of the original codes only in $\mathbb{F}_3$ and $\mathbb{F}_4$, due to the following relation (see [13,25]):

Lemma 2 *Let $C \subseteq \mathbb{F}_q^n$, and $(v;(\sigma,\alpha)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$. Then*

(i) $(v;(\sigma,\alpha))(C)^\perp = (v^{-1};(\sigma,\alpha))(C^\perp)$ *where $C^\perp$ is w.r.t. $\langle,\rangle_{\mathrm{E}}$.*
(ii) $(v;(\sigma,\alpha))(C)^\perp = ((\overline{v})^{-1};(\sigma,\alpha))(C^\perp)$ *where $C^\perp$ is w.r.t. $\langle,\rangle_{\mathrm{H}}$.*

Then, a signature for an extension of $\mathcal{SSA}$ can be built from the weight enumerator of the $\mathcal{H}(\widetilde{C})$. 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. 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.

Note that, there is a similar negative complexity result due to Dirk Vertigan [28]. The 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.

7 Computational vs. decisional code equivalence

The following proposition states that if

Proposition 3 *Let C be a q-ary linear code and $(v;(\alpha,\sigma))$ be an isometry of the code automorphism group* $\mathrm{Aut}(C)$. *For all $i \in \mathcal{I}_n$ we have $C_i \overset{\mathbf{SLE}}{\sim} C_{\sigma(i)}$ and ${C^\perp}_i \overset{\mathbf{SLE}}{\sim} {C^\perp}_{\sigma(i)}$.*

Corollary 1 *Let C and C' be two q-ary linear code. If $C \overset{\mathbf{SLE}}{\sim} C'$ with $C' = \Psi(C)$ and $\Psi = (\upsilon; (\alpha, \sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$, then for all $i \in \mathcal{I}_n$ we have ${C_i} \overset{\mathbf{SLE}}{\sim} {C'}_{\sigma(i)}$ and ${C^\perp}_i \overset{\mathbf{SLE}}{\sim} {(C')^\perp}_{\sigma(i)}$.*

The converse

Property 1 Let C be a q-ary linear code. If $C_i \overset{\mathbf{SLE}}{\sim} C_j$ for some i and j in $\mathcal{I}_n$ then there exist an isometry $(\upsilon; (\alpha, \sigma)) \in \mathrm{Aut}(C)$ such that $j = \sigma(i)$.

If the property holds for some code C,

Property 2 Let C be a q-ary linear code. If $C_i \overset{\mathbf{SLE}}{\sim} C_j$ and ${C^\perp}_i \overset{\mathbf{SLE}}{\sim} {C^\perp}_j$ for some i and j in $\mathcal{I}_n$ then there exist an isometry $(\upsilon; (\alpha, \sigma)) \in \mathrm{Aut}(C)$ such that $j = \sigma(i)$.

function `find_perm`
`input:` two q-ary (n, k) code C and C'
    `if` $\min(k, n-k) \le k_0$ `then` *// k_0 is some small constant*
        `return` $\mathcal{SSA}(C, C')$ *// or any other technique*
    `if` $C \not\sim C'$ `then` FAIL *// and exit*
    find i such that $C_n \sim C'_i$
    $\tau \leftarrow (n\ i)$ *// the transposition of n and i*
    $C'' \leftarrow \tau(C')$ *// so that $C_n \sim C''_n$*
    $\sigma \leftarrow$ `find_perm`(C_n, C''_n) *// implicitly $\sigma(n) = n$*
    `return` $\tau \circ \sigma$

7.1 From Permutation to Isometry

We show here if that if two codes are equivalent, say $C' = \Psi(C)$ for some isometry $\Psi = (\upsilon; (\alpha, \sigma)) \in \mathbb{F}_q^{*n} \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathcal{S}_n)$, then it is enough to know the permutation σ to recover the whole isometry with a polynomial complexity.

We first consider the case where C and C' are linearly equivalent codes. We denote $(\upsilon; \sigma) \in \mathbb{F}_q^{*n} \rtimes \mathcal{S}_n$ the corresponding isometry. and assume that the permutation σ is known. Then recovering υ is always easy. In fact, once C and C' are reordered, we may use the unicity of the systematic generator matrix to recover $\upsilon = (\upsilon_1, \ldots, \upsilon_n)$. Let

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

denote generator matrices of C and C' respectively. The following matrix

$$G = \left(\begin{array}{c|c} \upsilon_1 & \\ \ddots & \upsilon_j g_{i,j} \\ \upsilon_k & \end{array}\right)$$

must be a generator matrix of C' and thus we must have

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

which provides a system a linear equations relating the v_i

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

Any solution of that system provides a solution to the computational linear equivalence problem. In practice, this system can be solved with only $O(n)$ field operations.

First we remark that the field automorphism in the isometries is not a big issue as far as algorithmic complexity is concerned. If $q = p^r$ there are only r distinct field automorphisms (the powers of the Frobenius).

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 [6] 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. Assmus, E.F.J., Key, J.D.: Designs and their Codes, *Cambridge Tracts in Mathematics*, vol. 103. Cambridge University Press (1992). Second printing with corrections, 1993
2. 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)
3. 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)
4. Bouyukliev, I.: About the code equivalence. Ser. Coding Theory Cryptol. **3**, 126–151 (2007)
5. Danielsen, L.E., Parker, M.G.: Edge local complementation and equivalence of binary linear codes. Des. Codes Cryptography **49**, 161–170 (2008)
6. 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)
7. Feulner, T.: The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes. Adv. Math. Commun. **3**, 363–383 (2009)
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. Loidreau, P., Sendrier, N.: Weak keys in McEliece public-key cryptosystem. IEEE Trans. Inform. Theory **47**, 1207–1212 (2001)
18. MacWilliams, F.J.: Error-correcting codes for multiple-level transmission. Bell. Syst. Tech. J. **40**, 281–308 (1961)
19. 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)
20. Östergård, P.R.J.: Classifying subspaces of hamming spaces. Des. Codes Cryptography **27**, 297–305 (2002)
21. Overbeck, R., Sendrier, N.: Code-based cryptography. In: D. Bernstein, J. Buchmann, E. Dahmen (eds.) Post-Quantum Cryptography, pp. 95–145. Springer (2009)
22. Petrank, E., Roth, R.M.: Is code equivalence easy to decide? IEEE Trans. Inform. Theory **43**, 1602–1604 (1997)
23. 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)

24. Sendrier, N.: On the dimension of the hull. SIAM J. Discrete Math. **10**(2), 282–293 (1997)
25. Sendrier, N.: Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Trans. Inform. Theory **26**, 1193–1203 (2000)
26. 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)
27. Skersys, G.: Calcul du groupe d'automorphisme des codes. détermination de l' equivalence des codes. Thèse de doctorat, Université de Limoges (1999)
28. Vertigan, D.: Bicycle dimension and special points of the Tutte polynomial. Journal of Combinatorial Theory, Series B **74**, 378–396 (1998)

A Average Cardinality of the Hull

When n and k go to infinity, the proportion of q-ary linear $[n,k]$ codes with a hull of dimension i is denoted R_i and we have

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

Proposition 4 *The hull of a linear code has cardinality 2 asymptotically on average.*

Proof Asymptotically, q-ary linear codes have dimension i and thus cardinality q^i with probability R_i. The asymptotic average cardinality is then

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