On Linear Equivalence, Canonical Forms, and Digital Signatures

Tung Chou[1], Edoardo Persichetti[2], Paolo Santini[3]

[1]Academia Sinica, Taiwan.
[2]Florida Atlantic University, USA.
[3] Marche Polytechnic University, Italy.

Contributing authors: blueprint@crypto.tw; epersichetti@fau.edu; p.santini@staff.univpm.it;

Abstract

Given two linear codes, the code equivalence problem asks to find an isometry mapping one code into the other. The problem can be described in terms of group actions and, as such, finds a natural application in signatures derived from a Zero-Knowledge Proof system.

A recent paper, presented at Asiacrypt 2023, showed how a proof of equivalence can be significantly compressed by describing how the isometry acts only on an information set. Still, the resulting signatures are far from being optimal, as the size for a witness to this relation is still significantly larger than the theoretical lower bound, which is twice the security parameter.

In this paper, we fill this gap and propose a new notion of equivalence, which leads to a drastically reduced witness size. For many cases, the resulting size is exactly the optimal one given by the lower bound. We achieve this by introducing the framework of *canonical representatives*, that is, representatives for classes of codes which are equivalent under some notion of equivalence. We propose new notions of equivalence which encompass and further extend all the existing ones: this allows to identify broader classes of equivalent codes, for which the equivalence can be proved with a very compact witness. We associate these new notions to a specific problem, called Canonical Form Linear Equivalence Problem (CF-LEP), which we show to be as hard as the original one (when random codes are considered), providing reductions in both ways. As an added consequence, this reduction leads to a new solver for the code equivalence problem, which is the fastest solver when the finite field size is large enough. Finally, we show that our framework yields a remarkable reduction in signature size when compared to the LESS submission. Our variant is able to obtain very compact signatures, around 2 KB or less, which are among the smallest in the code-based setting.

1 Introduction

LESS is a post-quantum signature scheme, first introduced in [9], which relies on the idea of finding some kind of isomorphism between linear codes. This notion is well known in coding theory under the name of *code equivalence* and has been studied for a very long time. Indeed, determining whether two linear codes are equivalent is considered a hard task, in general, and thus constitutes a natural problem to construct cryptographic protocols. Two notions of isomorphisms are traditionally considered, for the Hamming metric: permutations and monomial maps. These yield the problems usually referred to as Permutation Equivalence Problem (PEP) and Linear Equivalence Problem (LEP), respectively.

The equivalence between codes can be seen as a *group action*, akin to the ubiquitous one behind the Discrete Logarithm Problem (DLP), although showing more similarities to settings such as the isomorphisms between polynomials, or graphs. It is in this way that LESS is constructed, following in the steps of well-trodden paths to construct a Sigma protocol based on the Code Equivalence Problem; this is then turned into a signature scheme using the Fiat-Shamir transform. The lion's share of the signature consists of the protocol responses that are provided by the prover. In the case of LESS these are, in principle, either ephemeral transformations, or proofs of equivalence between two codes. The former are conveniently communicated using seeds of length λ, where λ is the security parameter. For the latter type, instead, the prover must communicate either a length-n permutation or a length-n monomial map (depending on which notion of equivalence is employed); with the encoding employed in [9], this yields the following communication costs:

$$\text{PEP: } n\log_2(n), \quad \text{LEP: } n\big(\log_2(n)+\log_2(q-1)\big)$$

where n and q denote the code length and finite field size, respectively.

In [18], the authors propose a way to reduce the cost for communicating a proof of equivalence. The idea consists in describing how the isomorphism acts only on an information set, which is composed of k coordinates; thanks to this technique, [18] reduces the communication cost for communicating an isometry down to

$$\text{PEP: } Rn\log_2(n), \quad \text{LEP: } Rn\big(\log_2(n)+\log_2(q-1)\big)$$

where $R = k/n$ denotes the code rate. To verify that such a truncated description indeed leads to an isometry, [18] proposes to commit to an ad-hoc invariant function, whose role is basically that of compensating for the missing information. This requires to introduce new notions of equivalence, called Information-Set Permutation Equivalence and Information-Set Linear Equivalence, leading to the computational problems IS-PEP and IS-LEP. These problems turn out to be as hard as their traditional counterparts so that, in the end, one can rely on these new formulations without introducing additional security assumptions. In particular, IS-LEP has been used for the specification of LESS [1], as submitted to NIST's call for additional post-quantum signatures [17].

As is well known, a theoretical lower bound on the size of non-ephemeral responses in a ZK protocol based on group actions is 2λ. Despite the work in [18], the communication cost for the code equivalence group action is still far from optimal. This can be easily seen by looking at the proposed LESS instances. Since $q = 127$, $R = \frac{1}{2}$ and $n \approx 2\lambda$, isometries are communicated with approximately $\lambda\big(1 + \log_2(\lambda)\big)$ bits for IS-PEP and $\lambda\big(8 + \log_2(\lambda)\big)$ for IS-LEP.[1] One may argue that a reduction in communication cost could be obtained using an optimal encoding for permutations and monomials. For instance, considering only the case of PEP, from Stirling's approximation we know that permutations can be represented with $\approx 2\lambda \cdot \log_2(2\lambda/e) + \frac{1}{2} \cdot \log_2(4\pi\lambda)$ bits (this has been derived assuming $n \approx 2\lambda$). The dominant term is $2\lambda \cdot \log_2(2\lambda)$, which is exactly the communication cost according to [9]. For actual parameters, even the optimal encoding is pretty far from the theoretical lower bound: for instance, for $\lambda = 128$, we obtain 1687 bits which is 6.6 times larger than the theoretical lower bound 256. Analogous considerations hold for the other equivalence relations.

1.1 Our Contributions

We show how to drastically reduce the size of witnesses to the code equivalence problem: for both PEP and LEP, this becomes just $\log_2 \binom{n}{k}$ bits. Perhaps surprisingly, the size is the same regardless of which notion of equivalence is considered. For $k = Rn$, this corresponds to $n \cdot h(R) \cdot \big(1 + o(1)\big)$, where h denotes the binary entropy function. We show that, in many cases, this size corresponds to the optimal one given by the lower bound. Moreover, we propose a novel attack with running time $\tilde{O}\left(\sqrt{\binom{n}{k}}\right) = 2^{\frac{1}{2}n \cdot h(R) \cdot \big(1+o(1)\big)}$ which, when the size of the underlying finite field is large enough, turns out to be the fastest solver for code equivalence.[2] For a security parameter of λ, we set $\frac{1}{2}n \cdot h(R) = \lambda$ so that the bit size of a witness becomes $n \cdot h(R) \cdot \big(1 + o(1)\big) = 2\lambda\big(1 + o(1)\big)$ and matches the theoretical lower bound.

We apply this machinery to LESS signatures. As expected, the resulting scheme, which we call CF-LESS, achieves very compact signatures, much smaller than its predecessors. Indeed, LESS parameters have been chosen using a lower bound on the cost of attacks based on low-weight codewords finding, with resulting code lengths $n \approx 2\lambda$. In other words, these instances have been designed considering the cost of a potential attack which, however, does not exist right now: all known attacks have a somewhat higher cost. Since all LESS instances have $R = \frac{1}{2}$, our new attack runs in time $2^{\frac{1}{2}n \cdot h(\frac{1}{2}) \cdot \big(1+o(1)\big)} \approx 2^{\lambda\big(1+o(1)\big)}$ and thus represents the best currently known attack on LESS instances.[3] Moreover, in this regime, the size for a proof of equivalence is reduced to $n \cdot h(\frac{1}{2}) = n \approx 2\lambda$ bits. In practice, this implies that we can reduce LESS signature sizes as much as possible, ultimately reaching the theoretical lower bound. Considering the same code and protocol parameters as in the "balanced" parameter sets from the LESS submission [1] (which uses only 2 generator matrices and aims to

[1] In the LESS submission, the code lengths are $n = 252$, $n = 400$ and $n = 548$ for the three NIST security categories 1, 3 and 5.

[2] This formula holds only if the success probability of the employed canonical form function is non-negligible. This is exactly what happens for all cases which are relevant for cryptographic applications.

[3] This is true at least asymptotically. Indeed, the $o(1)$ is due to polynomial factors which, in the end, increase the cost by $20 \div 30$ bits.

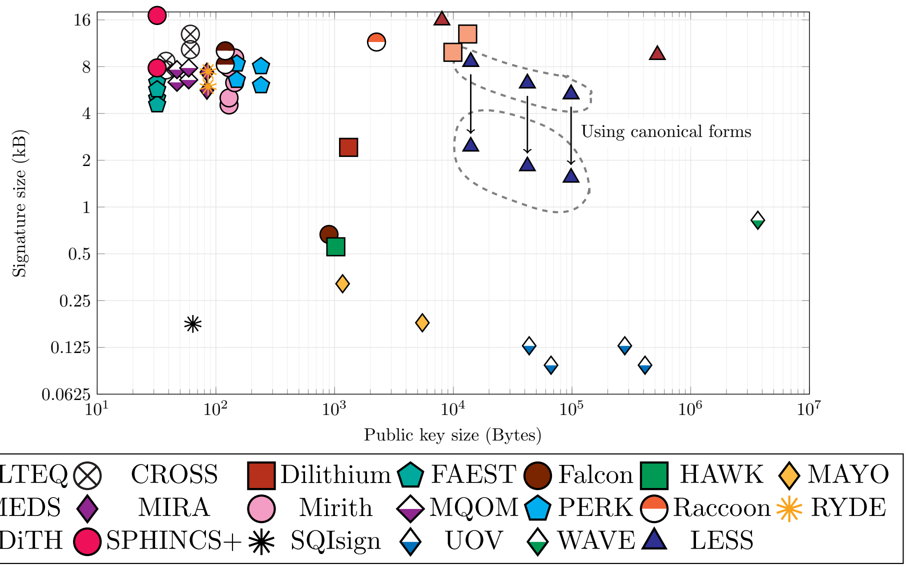

Fig. 1: Comparison between CF-LESS and some selected schemes from Round 1 of NIST's competition (including the current version of LESS), for NIST category 1.

minimize the public key size), we obtain signatures of only 2.4 KB, 5.7 KB, and 9.8 KB for NIST security categories 1, 3 and 5, respectively. If 4 generator matrices are used, these sizes are further reduced to 1.8 KB, 4.3 KB and 7.7 KB, respectively. We apply the same modifications to the ring signature scheme of [3], obtaining a comparable gain in signature size (depending on the amount of users in the ring).

In practice, the modification to existing schemes can be viewed as tweaks to how commitments are prepared and later verified: the prover commits to the *canonical representative* of a code where, by canonical representative, we refer to the representative of some equivalence class. We introduce canonical representative for codes, show how they can be computed efficiently and how they relate with existing notions of equivalence. In practice, the operations required to compute a canonical representative for a code boil down to computing a *canonical form* for a certain equivalence relation on $k \times (n-k)$ matrices. In this paper, we introduce efficient functions to compute such a canonical form for all relevant equivalence classes.

Our framework leads to a new notion of equivalence and to the associated computational problem which, in a nutshell, consists in finding an isometry so that the two input codes lead to the same canonical representative. We show that this new problem is as hard as the traditional ones, when random codes are employed (as in LESS). Thus, the new schemes we propose in this paper enjoy the same security guarantees as their predecessors. Moreover, the modifications we require lead to a computational overhead which, in the worst case, is comparable with the computational bottlenecks that these schemes already exhibit. Thus, in the end, our techniques allow to reduce signatures sizes without any important penalty, for what concerns both security and computational complexity.

The framework we describe in this paper generalizes the one introduced in [18]. Indeed, using the vocabulary of canonical representatives, we can say that IS-PEP and IS-LEP correspond to special cases of our more general framework. In particular, these notions are associated with equivalence classes which, however, are not as broad as they can be. In this paper we enlarge these classes as much as possible and define canonical representative functions having much stronger invariance properties with respect to those employed for IS-PEP and IS-LEP: this is a more challenging task than the one faced in [18], which we solve by providing efficient examples of such functions, for all relevant cases.

1.2 Paper Organization

The paper is organized as follows. Section 2 specifies our notation and summarizes some preliminary notions. Section 3 describes the notions of equivalence for codes, introducing a new formalism, as well as the concept of canonical representatives. This serves as an important basis for the discussions in the subsequent sections. Section 4 shows concrete ways to define new canonical representatives, expanding on the existing ones that were described in the previous section; in practice, this is achieved by defining canonical forms. In Section 5, we first briefly review the Sigma protocol underlying LESS, and then present a new Sigma protocol, which we refer to as the CF-LESS Sigma protocol, that makes use of canonical forms to reduce the communication size. We will see in Section 7 that this has a considerable impact on signature size, allowing for a drastic reduction which yields the smallest signature sizes among many post-quantum schemes, and in particular, code-based schemes based on zero-knowledge proofs. Finally, in Section 6, we discuss the security of our new technique: we first argue that CF-LESS is secure by showing a security reduction, and then discuss an application of canonical forms to cryptanalysis, which results in an intuitive attack against LEP.

2 Notation and Preliminaries

In this section, we settle the notation we are going to use throughout the paper and recall useful background concepts about linear codes and the code equivalence problem.

Finite Fields, Vectors and Matrices

As usual, $\mathbb{F}_q$ denotes the finite field with q elements and $\mathbb{F}_q^*$ stands for its multiplicative group. Then, vector and matrix spaces over this field are defined naturally as, respectively, $\mathbb{F}_q^n$ and $\mathbb{F}_q^{k\times n}$. We use bold uppercase (resp., lowercase) letters for matrices (resp., vectors). For a vector $\mathbf{v}$, v_i indicates the i-th element; for a matrix $\mathbf{A}$, $a_{i,j}$ indicates the element in the i-th row and j-th column. For a non ordered set $J \subseteq \{1, \cdots, n\}$ of size m and a matrix $\mathbf{A} \in \mathbb{F}_q^{k\times n}$, we use $\mathbf{A}_J$ to indicate the $k \times m$ submatrix formed by the columns of $\mathbf{A}$ that are indexed by J. The general linear group of non-singular $k \times k$ matrices over $\mathbb{F}_q$ is indicated as $\mathrm{GL}_k(q)$. We denote by $\overline{\mathbb{F}}_q^{k\times n} \subset \mathbb{F}_q^{k\times n}$, where $k < n$, the set of $k \times n$ matrices having full rank k. Finally, the identity matrix of size k is indicated as $\mathbf{I}_k$.

Permutation and Monomial Maps

We denote by S_n the symmetric group on n elements, comprised of *permutations* of n objects. The identity in the group S_n is indicated as id_n. Using the two-lines notation, a permutation π can be expressed as

$$\pi := \begin{pmatrix} 1 & 2 & \cdots & n \\ \pi(1) & \pi(2) & \cdots & \pi(n) \end{pmatrix},$$

meaning that π moves the j-th element to position $\pi(j)$. For a vector $\mathbf{v} = (v_1, \cdots, v_n)$, it holds that

$$\pi(\mathbf{v}) = \big(v_{\pi^{-1}(1)}, \cdots, v_{\pi^{-1}(n)}\big).$$

As it is well known, each permutation can be represented also as an $n \times n$ matrix such that every row and every column has a unique element equal to 1 while all the other elements are 0. If $\mathbf{P}$ is the permutation matrix associated to $\pi \in S_n$, we then have $\pi(\mathbf{v}) = \mathbf{v} \cdot \mathbf{P}$. Sometimes we slightly abuse notation and write $\mathbf{P} \in S_n$ to imply that $\mathbf{P}$ is a permutation matrix.

The symmetric group can be seen as a subgroup of a more general group M_n (formally, we express this as $S_n \leq M_n$) which corresponds to the group of *monomial maps*, that is, functions of the form $\mu := (\pi, \mathbf{a})$ with $\pi \in S_n$ and $\mathbf{a} \in \mathbb{F}_q^{*n}$, acting as follows

$$\mu(\mathbf{v}) = \pi(\mathbf{v}) \cdot \begin{pmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_n \end{pmatrix} = \big(a_1 \cdot v_{\pi^{-1}(1)}, \cdots, a_n \cdot v_{\pi^{-1}(n)}\big).$$

Similarly to the case of permutations, we can represent monomial transformation with matrices. In particular, each monomial can be associated with an $n \times n$ matrix $\mathbf{Q} \in \mathbb{F}_q^{n \times n}$ such that every row and column has a unique non null element which, differently from permutations, can be any element from $\mathbb{F}_q^*$. Again, for such matrices, we will sometimes be flexible with notation and write $\mathbf{Q} \in M_n$.

We naturally extend the action of monomials on matrices $\mathbf{A}$, i.e., $\mu(\mathbf{A})$ indicates the matrix resulting from the action of μ on the columns of $\mathbf{A}$. In other words, if $\mathbf{Q}$ is the monomial associated with μ, we have $\mu(\mathbf{A}) = \mathbf{G} \cdot \mathbf{Q}$. The group operation in M_n is indicated as $\circ$: for two monomials $\mu, \mu' \in M_n$, we write $\mu \circ \mu' \in M_n$ to denote the monomial resulting from their combination and have $(\mu' \circ \mu)(\mathbf{A}) = \mu'\big(\mu(\mathbf{A})\big) = \mathbf{A} \cdot \mathbf{Q} \cdot \mathbf{Q}'$, with $\mathbf{Q}$ and $\mathbf{Q}'$ being the matrix representations of μ and μ', respectively. The identity in M_n is denoted as id_n.

Finally, for our work, we introduce a special type of permutations: given $k < n$, we call $S_{k,n} \subset S_n$ the set of permutations such that

$$\pi^{-1}(1) < \pi^{-1}(2) < \cdots < \pi^{-1}(k), \qquad \pi^{-1}(k+1) < \pi^{-1}(k+2) < \cdots < \pi^{-1}(n).$$

In other words, each permutation $\pi \in S_{k,n}$ is uniquely associated with a size-k subset $J \subseteq \{1, \cdots, n\}$ such that indices in J are moved to the first k positions, while indices outside of J are moved to the last $n-k$ positions. The matrix representation of such a permutation would be such that:

- $i < i' \leq k$ and $p_{i,j} = p_{i',j'} = 1$ implies that $j < j'$, and
- $k < i < i'$ and $p_{i,j} = p_{i',j'} = 1$ implies that $j < j'$.

In fact, once the first k columns of a matrix in $S_{k,n}$ are defined, the whole matrix is defined. Note that $|S_{k,n}| = \binom{n}{k} \leq 2^n$. Examples of permutations from $S_{k,n}$ are shown in Figure 2.

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\pi := \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 4 & 2 & 5 & 3 & 6 \end{pmatrix}$$

(a)

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\pi := \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 4 & 1 & 5 & 2 & 3 & 6 \end{pmatrix}$$

(b)

Fig. 2: Examples of matrices from $S_{k,n}$ and their binary representation, for $n = 6$ and $k = 3$. The size-3 sets representing the permutations are $\{1, 3, 5\}$ for (a) and $\{2, 4, 5\}$ for (b).

Complexity

We denote the "small o" and "big O" Landau symbols as $o(\cdot)$ and $O(\cdot)$, respectively. We sometimes relax the notation and, as it is common, use the soft O symbol: we write $O\big(n^{e_1} \cdot (\log n)^{e_2}\big)$ as $\tilde{O}(n^{e_1})$ for any positive $e_1, e_2 \in \mathbb{R}$. Each occurrence of a field addition, a field subtraction, a field multiplication, and a field inversion is considered to take 1 field operation.

Orderings

We assume that there is a total ordering $\leq_{\mathbb{F}_q}$ defined on $\mathbb{F}_q$, and comparing two elements in $\mathbb{F}_q$ w.r.t. $\leq_{\mathbb{F}_q}$ is considered to take 1 field operation. For an integer d, we define $\leq_{\mathbb{F}_q^d}$ as the total ordering defined on $\mathbb{F}_q^d$, such that $\mathbf{v} \leq_{\mathbb{F}_q^d} \mathbf{v}'$ if and only if 1) $\mathbf{v} = \mathbf{v}'$ or 2) $\mathbf{v}_i \leq_{\mathbb{F}_q} \mathbf{v}'_i$ and $\mathbf{v}_i \neq \mathbf{v}'_i$ for some i and $\mathbf{v}_j = \mathbf{v}'_j$ for all $j < i$. We define a total ordering $\leq_{\mathbb{F}_q^{k \times (n-k)}}$ on $\mathbb{F}_q^{k \times (n-k)}$ in a similar way. Comparing two elements in $\mathbb{F}_q^d$ w.r.t. $\leq_{\mathbb{F}_q^d}$ is considered to take $O(d)$ field operations. Comparing two elements in $\mathbb{F}_q^{k \times (n-k)}$ w.r.t. $\leq_{\mathbb{F}_q^{k \times n-k}}$ is considered to take $O\big(k(n-k)\big)$ field operations.

For a vector $\mathbf{v}$, we use the notation $\mathsf{multiset}(\mathbf{v})$ to indicate the multiset formed by its entries. We define a partial ordering $\prec_{\mathbb{F}_q^d}$ on $\mathbb{F}_q^d$. For $\mathbf{v}, \mathbf{v}' \in \mathbb{F}_q^d$, we have $\mathbf{v} \prec_{\mathbb{F}_q^d} \mathbf{v}'$ if and only if $\mathbf{w} \leq_{\mathbb{F}_q^d} \mathbf{w}'$ and $\mathbf{w} \neq \mathbf{w}'$, where $\mathbf{w}, \mathbf{w}' \in \mathbb{F}_q^d$ are obtained by sorting entries in $\mathbf{v}, \mathbf{v}'$ w.r.t. $\leq_{\mathbb{F}_q}$, respectively. Comparing two elements[4] in $\mathbb{F}_q^d$ w.r.t. $\prec_{\mathbb{F}_q^d}$ is considered to take $\tilde{O}(d)$ field operations. One can view $\prec_{\mathbb{F}_q^d}$ as a way to compare vectors $\mathbf{v}, \mathbf{v}' \in \mathbb{F}_q^d$ by comparing $\mathsf{multiset}(\mathbf{v}), \mathsf{multiset}(\mathbf{v}')$ w.r.t. a total ordering defined on all size-d multisets of elements in $\mathbb{F}_q$. $\mathbf{v}$ and $\mathbf{v}'$ are considered incomparable if $\mathbf{v} \neq \mathbf{v}'$ and $\mathsf{multiset}(\mathbf{v}) = \mathsf{multiset}(\mathbf{v}')$, since neither of $\mathbf{v} = \mathbf{v}'$, $\mathbf{v} \prec_{\mathbb{F}_q^{n-k}} \mathbf{v}'$, or $\mathbf{v}' \prec_{\mathbb{F}_q^{n-k}} \mathbf{v}$ holds.

Linear Codes

A *linear code* $\mathscr{C}$ with *dimension* $k > 0$ and *length* $n \geq k$ is a k-dimensional linear subspace of $\mathbb{F}_q^n$. Linear codes are typically measured in the *Hamming metric*, which defines the *weight* of a word as the number of its non-zero positions; a notion of *distance* between words is then naturally defined as the weight of their difference (which corresponds to the number of positions in which they differ). The *minimum distance* of a code $\mathscr{C}$ is by definition the smallest distance among distinct codewords of $\mathscr{C}$, which can be easily seen to be equivalent to the smallest weight of a non null codeword of $\mathscr{C}$. Since linear codes are vector spaces, they can be represented compactly via a choice of basis; typically, such a basis consists of vectors, which are seen as rows of a full rank matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, so that $\mathscr{C} = \{\mathbf{u} \cdot \mathbf{G} \mid \mathbf{u} \in \mathbb{F}_q^k\}$. This matrix is called *generator matrix*. Note that there are several choices for generator matrices for the same code, corresponding to different choices of basis: any $\mathbf{G}$ and $\mathbf{S} \cdot \mathbf{G}$, where $\mathbf{S} \in \mathrm{GL}_k(q)$, generate the same code. Whenever a generator matrix is in the form $(\mathbf{I}_k \mid \mathbf{A})$, where $\mathbf{I}_k$ is the identity of size k, we say it is in *systematic form* (or *standard form*). When $\{1, \cdots, k\}$ is an *information set* (i.e. the first k columns of $\mathbf{G}$ are linearly independent), this form can be obtained by setting $\mathbf{S}$ as the inverse of the leftmost $k \times k$ submatrix in $\mathbf{G}$. Otherwise, one can first compute the *Reduced Row Echelon Form (RREF)*, i.e., the matrix $\mathbf{G}_J^{-1} \cdot \mathbf{G}$ with J being the first information set[5], and then eventually applying a column permutation so that the identity columns are moved from positions J to $\{1, \cdots, k\}$. In other words, the systematic form is defined as

$$\mathsf{SF}(\mathbf{G}) = \left(\mathbf{I}_k \mid \mathbf{G}_J^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus J}\right), \quad J \text{ is the first information set.}$$

Note that this also corresponds to $\mathsf{RREF}(\mathbf{G}) \cdot \mathbf{P}$ for some permutation matrix $\mathbf{P} \in S_{k,n}$, where $\mathsf{RREF} : \mathbb{F}_q^{k \times n} \to \mathbb{F}_q^{k \times n}$ is the function computing the RREF. In the following, we will use RREF^* to define the function returning both the systematic form, as well as the permutation $\mathbf{P} \in S_{k,n}$. If $\{1, \cdots, k\}$ is an information set, then SF and RREF coincide and $\mathbf{P} = \mathbf{I}_n$. Notice that SF and RREF are invariant under changes of basis: for any two generator matrices $\mathbf{G}$ and $\mathbf{S} \in \mathrm{GL}_k(q)$, it holds that $\mathsf{SF}(\mathbf{G}) = \mathsf{SF}(\mathbf{SG})$ and $\mathsf{RREF}(\mathbf{G}) = \mathsf{RREF}(\mathbf{S} \cdot \mathbf{G})$.

[4] By this we mean, given two elements $\mathbf{v}, \mathbf{v}' \in \mathbb{F}_q^d$, determining which of the following cases holds: 1) $\mathbf{v} = \mathbf{v}'$, 2)$\mathbf{v} \prec_{\mathbb{F}_q^d} \mathbf{v}'$, 3) $\mathbf{v}' \prec_{\mathbb{F}_q^d} \mathbf{v}$, or 4) neither of the 3 cases holds.

[5] We consider the natural ordering defined by the relations $\{1, \cdots, k-1, k\} < \{1, \cdots, k-1, k+1\} < \cdots < \{1, \cdots, k-1, n\} < \cdots < \{1, \cdots, n-1, n\} < \cdots < \{2, \cdots, k, k+1\} < \cdots < \{n-k+1, \cdots, n\}$

3 Notions of Equivalence for Codes

To describe the various notions of code equivalence, we begin by introducing a unified framework. To this end, let $(E, \circ)$ be a group of *isometries*, i.e. maps which preserve the distances. A map $\psi \in E$ acts on codewords, i.e. is an endomorphism of $\mathbb{F}_q^n$. When ψ is applied to all the codewords of a code $\mathscr{C}$, with some abuse of notation we write $\psi(\mathscr{C})$, i.e. $\psi(\mathscr{C}) = \{\psi(\mathbf{c}) \mid \mathbf{c} \in \mathscr{C}\}$. We then have the following definition.

Definition 1. *Two linear codes $\mathscr{C}$ and $\mathscr{C}'$ are E-*equivalent *if there exists $\psi \in E$ such that $\mathscr{C}' = \psi(\mathscr{C})$; in such a case, we write $\mathscr{C} \sim_E \mathscr{C}'$.*

In other words, two codes are E-equivalent (or simply "equivalent") whenever there is an isometry mapping one code into the other; this isometry is in fact a *witness* for the equivalence. Obviously, not all codes are equivalent (and in fact, the distribution of the weights of the codewords crucially affects the error-correcting capabilities of a code). It is then natural to ask whether two given codes are equivalent or not, which leads to the following problem.

Problem 1 (Code Equivalence Problem (CEP)). *Given linear codes $\mathscr{C}$ and $\mathscr{C}'$, determine if $\mathscr{C} \sim_E \mathscr{C}'$, i.e., if there exists $\psi \in E$ such that $\mathscr{C}' = \psi(\mathscr{C})$.*

The nature of the isometries depends, also, on which metric is considered. As our formulation is very generic, it could theoretically encompass various metrics besides the Hamming one, such as the *rank metric* or the *Lee metric*, and associated notions of equivalence. In our work, however, we focus exclusively on the Hamming metric; we refer the reader interested in a more generic characterization to [11], for example, where the topic is studied for any group action. In the Hamming metric, an isometry needs to preserve the number of non-zero positions in a word. In the simplest of cases, such a map consists of just a permutation, so that $E = S_n$, which leads to the notion of *permutation equivalence*; if instead the map is a monomial one, then $E = M_n$ and this is usually known as *linear equivalence*. The most general notion of *semilinear equivalence* also includes a field automorphism, i.e. $E = \mathrm{Aut}(\mathbb{F}_q) \times M_n$; however, this concept is not relevant for cryptographic applications[6], and we do not treat it here. Furthermore, it is immediate to notice that permutation equivalence is nothing but a special case of linear equivalence.

According to the above, we have that two codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ are M_n-equivalent, and write $\mathscr{C} \sim_{M_n} \mathscr{C}'$ if there exists some $\mu \in M_n$ such that $\mathscr{C}' = \mu(\mathscr{C})$. This is normally referred to as *linear equivalence* and, consequently, the codes are said to be linearly equivalent. The associated problem of determining whether two codes are linearly equivalent is normally called *Linear Equivalence Problem (LEP)*. In the special case in which the isometry is a permutation, we denote the equivalence by $\mathscr{C} \sim_{S_n} \mathscr{C}'$ and define the associated problem as *Permutation Equivalence Problem (PEP)*. The codes are, consequently, said to be *permutation equivalent*.

Problem 2 (Linear Equivalence Problem (LEP)). *Given linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ with length n and dimension k, determine if $\mathscr{C} \sim_{M_n} \mathscr{C}'$, i.e., if there exists $\mu \in M_n$ such that $\mathscr{C}' = \mu(\mathscr{C})$.*

---

[6] In the sense that it does not affect security (negatively or positively) and therefore does not influence choice of parameters.

Problem 3 (**Permutation Equivalence Problem (PEP)**)**.** *Given linear codes* $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ *with length* n *and dimension* k*, determine if* $\mathscr{C} \sim_{S_n} \mathscr{C}'$*, i.e., if there exists* $\pi \in S_n$ *such that* $\mathscr{C}' = \pi(\mathscr{C})$*.*

Now, recall that codes are typically represented through their generator matrices.[7] Consequently, the notion of code equivalence should, in principle, be defined on full-rank $k \times n$ matrices; since we know that each code admits multiple generator matrices, this means that one would have to account for possible change-of-basis matrices, as well. Indeed, we have that $\mathscr{C}$ and $\mathscr{C}'$ are linearly equivalent if, for any two generator matrices $\mathbf{G}, \mathbf{G}' \in \overline{\mathbb{F}}_q^{k \times n}$ for $\mathscr{C}$ and $\mathscr{C}'$, respectively, there exist $\mathbf{S} \in \mathrm{GL}_k(q)$ and $\mu \in M_n$ (represented via a monomial matrix $\mathbf{Q}$) such that

$$\mathbf{G}' = \mathbf{S} \cdot \mu(\mathbf{G}) = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{Q}.$$

Formally, this can be seen as the action of the group $\mathrm{GL}_k(q) \times M_n$ on the set of full-rank $k \times n$ matrices, mapping $\mathbf{G}$ to $\mathbf{G}'$ as described above. However, it is immediate to notice that, when one is not concerned with the specific choice of representative (i.e. generator matrix), such a view can be simplified, omitting the role of $\mathrm{GL}_k(q)$. In other words, if one were to choose a *canonical* representative for the codes, then linear equivalence could be fully described using only the associated monomial map. Indeed, one such representative already exists in coding theory, and it is typically the systematic form of a generator matrix; however, as it is possible that the leftmost k columns do not form an invertible submatrix, it is common to relax this notion and use the RREF instead. Importantly, this is always well defined: it can be computed efficiently from any given generator matrix and it uniquely identifies a code, in the sense that if two codes have the same generator matrix in RREF, then they must be the same code, and viceversa.

For the remainder of this work, we will show how this phenomenon can be generalized, and what kind of impact such a generalization has in cryptographic applications, in particular with respect to the LESS setting.

3.1 Equivalence Classes and Canonical Representatives

To begin, we introduce a subset $F \subseteq E$, to which we associate the equivalence relation $\sim_F$: we say that $\mathscr{C}$ and $\mathscr{C}'$ are F-equivalent, and write $\mathscr{C} \sim_F \mathscr{C}'$, if $\mathscr{C}' = \varphi(\mathscr{C})$ for some $\varphi \in F$. The associated problem, then, consists in determining whether, given two codes $\mathscr{C}$ and $\mathscr{C}'$, there exists some $\varphi \in F$ such that $\mathscr{C}' = \varphi(\mathscr{C})$. Solving this problem, i.e., deciding when two given codes are equivalent according to F, should ideally be computationally easy. We can then define the F*-equivalence class* of a code.

Definition 2 (F-Equivalence class)**.** *Let* $F \subseteq E$*. Given a code* $\mathscr{C} \subseteq \mathbb{F}_q^n$*, we define its equivalence class according to* F *(or* F*-equivalence class for short) as*

$$\mathfrak{C}_F(\mathscr{C}) = \{\varphi(\mathscr{C}) \mid \varphi \in F\}.$$

[7] Obviously, one may describe codes using parity-check matrices, instead; our theory applies in the same way also to this case.

At a first glance, we expect that the size of $\mathfrak{C}_F(\mathscr{C})$ would correspond to $|F|$. However, this is not guaranteed, as it may be that $|\mathfrak{C}_F(\mathscr{C})| < |F|$: this happens whenever two distinct isometries φ and φ' map $\mathscr{C}$ to the same code, which implies that $\varphi^{-1} \circ \varphi'$ and $\varphi'^{-1} \circ \varphi$ are automorphisms for $\mathscr{C}$.

If checking whether two codes are in the same F-equivalence class is computationally easy, and F-equivalence classes contain more than one element, it may be possible to reduce the size of a witness for the E-equivalence relation. Indeed, any isometry $\chi \in E$ sending $\mathscr{C}$ to the F-equivalence class $\mathfrak{C}_F(\mathscr{C}')$ will suffice, as it can be efficiently verified that $\mathscr{C}^* = \chi(\mathscr{C}) \sim_F \mathscr{C}'$ and this implies that $\mathscr{C}$ and $\mathscr{C}'$ are E-equivalent. Indeed, since any isometry in F is also an element of E, the three codes $\mathscr{C}$, $\mathscr{C}'$ and $\mathscr{C}^*$ are all equivalent according to E. Then, among all possibilities for χ, there may be some choices having a special representation, leading to a compact description.

As we show next, when F is a subgroup (and not a mere subset of E), we can formalize this framework through a group-theoretic point of view.

A Group-Theoretic Characterization

From now on, we focus on the case in which $F \leq E$ is a subgroup. For any $\psi \in E$, we define its (right)[8] coset as

$$F\psi = \{\varphi \circ \psi \mid \varphi \in F\}.$$

As is well known, cosets define a partition of E; the number of such cosets is called *index* of F in E and is typically indicated as $[E : F]$. By Lagrange's theorem, we get

$$[E : F] = \frac{|E|}{|F|}. \tag{1}$$

Now, let $\mathscr{C}' = \psi(\mathscr{C})$: to prove that $\mathscr{C}' \sim_E \mathscr{C}$, one can provide any $\chi \in F\psi$. Indeed, $\chi = \varphi \circ \psi$ with $\varphi \in F$ and

$$\mathscr{C}^* = \chi(\mathscr{C}) = \varphi \circ \psi(\mathscr{C}) \sim_F \psi(\mathscr{C}) = \mathscr{C}'.$$

Note that any two isometries in the same coset map $\mathscr{C}$ to the same F-equivalence class.

Assume there exists an efficient way to define a representative for each coset: then, the bitsize for a witness (recalling (1)) is reduced to

$$\log_2 [E : F] = \log_2 |E| - \log_2 |F|. \tag{2}$$

The above equation embeds the power of our framework. The larger F, the more compact is the description of a witness to the E-equivalence relation. Obviously, the goal here consists in identifying the largest F for which 1) F-equivalences can be easily verified, and 2) coset representatives can be efficiently computed. Later on, we provide examples that satisfy both requirements.

[8] Left cosets are defined analogously.

Canonical Representatives for F-equivalence Classes

To verify equivalences according to F, we propose to use *canonical representatives.* Namely, we want to define a way so that, given a code, one can efficiently compute a representative for its F-equivalence class. Below, we give the formal definition.

Definition 3 (Canonical Representatives)**.** *Let $F \leq E$ be a subgroup of isometries for k-dimensional linear codes. We say CR_F is a canonical representative function if:*

i) CR_F *takes as input a code $\mathscr{C} \subseteq \mathbb{F}_q^n$ with dimension k;*
ii) CR_F *returns either a k-dimensional linear code with length n, or a failure $\perp$;*
iii) for any input $\mathscr{C}$, CR_F *runs in time which is polynomial in n and $\log_2(q)$;*
iv) when the canonical representative is well defined, the output is F-equivalent to the input, i.e., $\mathsf{CR}_F(\mathscr{C}) \sim_F \mathscr{C}$ for any $\mathscr{C}$ such that $\mathsf{CR}_F(\mathscr{C}) \neq \perp$;
v) for any two F-equivalent codes, the output is the same, i.e., $\mathsf{CR}_F(\mathscr{C}) = \mathsf{CR}_F(\mathscr{C}')$ for any two codes $\mathscr{C}$ and $\mathscr{C}'$ such that $\mathscr{C} \sim_F \mathscr{C}'$.

Remark 1. *Requirement iii) is derived considering that linear codes can be represented using either a generator or a parity-check matrix, taking $O\big(n^2 \cdot \log_2(q)\big)$ bits. We observe that allowing for a time which is, instead, polynomial in q would have led to the possible existence of functions taking time which is exponential in the input size.*

Our definition for canonical representatives accounts for the possibility that, once a function CR_F is defined, for some F-equivalence classes a representative cannot be computed; this is captured by the function returning $\perp$ and, in such cases, we say that CR_F fails. Since CR_F can be computed in polynomial time, one can efficiently verify that two codes are in the same F-equivalence class (assuming CR_F does not fail), and thus that they are F-equivalent, by verifying that they lead to the same canonical representative (i.e., the same output of CR_F). A visualization of an F-equivalence class with a canonical representative is given in Figure 3.

Remark 2. *The whole framework can be described also in terms of group actions. Indeed, an equivalence class $\mathfrak{C}_F(\mathscr{C})$ basically corresponds to the orbit of $\mathscr{C}$ under the action of F; starting from this, many more analogies can be derived. We do not use such a description this paper and refer the interested reader to [11].*

3.2 Revisiting the Notion of Equivalence

Before introducing new concepts, we consider existing notions of equivalence and show how they fit in our framework.

Linear and Permutation Equivalence

We only treat explicitly the case $E = M_n$ as the case of S_n follows as a special case. Let $F = \{id_n\}$; then, equivalence classes contain a unique element, and hence a witness for the equivalence between two codes can only be the exact monomial mapping one code to the other. In this case, the definition of a canonical representative is trivial since CR_F is the identity. It is easy to see that the identity function satisfies all requirements from Definition 3 and is well defined for all input codes.

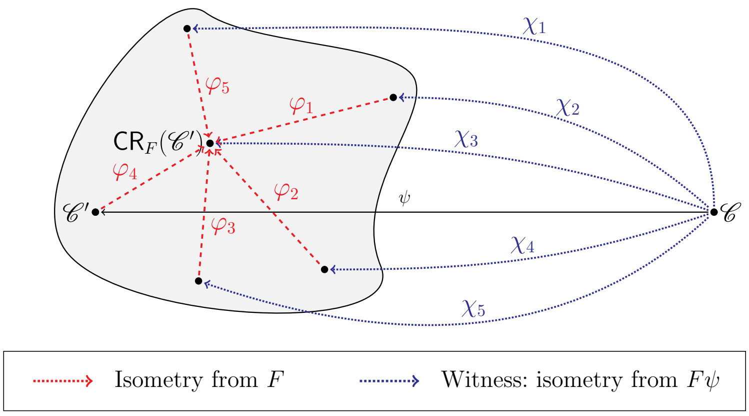

Fig. 3: Equivalence class for a code $\mathscr{C}' = \psi(\mathscr{C})$ and resulting witnesses. In this case, $F = \{id_n, \varphi_1, \cdots, \varphi_5\}$ hence $|\mathfrak{C}_F(\mathscr{C}')| \leq 6$; in the example, we have exactly $|\mathfrak{C}_F(\mathscr{C})| = |F| = 6$. The coset $F\psi$ contains six elements as well, namely, $F\psi = \{\psi, \chi_1, \cdots, \chi_5\}$ and each of them is a witness for the equivalence, according to E, between $\mathscr{C}$ and $\mathscr{C}'$. As shown in the figure, the canonical representative can be computed from each code in the equivalence class and is obtained via application of an isometry from F.

Since $|F| = 1$, we have $[E : F] = |E| = n!(q-1)^n$ and an element is typically encoded using $n\log_2(n) + n\log_2(q-1)$ bits. Incidentally, this is how witnesses were described in the original LESS formulation [4, 9].

Clearly, an application of our framework to this simple case would be excessively formal, yet this line of thought could be useful to lay the ground for less obvious (and more significant!) choices of F.

Remark 3. *A slight reduction in witness size can be obtained by considering that every code has, as (trivial) automorphisms, the transformations that scale all coordinates by the same non-zero value. It is easy to verify that all such transformations form a subgroup of M_n of order $q-1$: applying our machinery, we get*

$$[E : F] = \frac{|M_n|}{q-1} = n!(q-1)^{n-1}.$$

A convenient representative for each coset can be, for instance, the monomial transformation whose first coefficient is 1. We observe that each F-equivalence class contains a unique code also in this case.

Information-Set Linear and Permutation Equivalence

In [18], the authors propose two new notions of code equivalence with associated computational problems, IS-LEP and IS-PEP. Again, without loss of generality, we focus on the case of IS-LEP as IS-PEP follows as a special case. Let $J \subseteq \{1, \cdots, n\}$ be the size-k set corresponding to the pivoted columns in $\mathsf{RREF}(\mathscr{C}')$. Then, according to [18], equivalence can be verified via any monomial $\chi \in M_n$ such that $\mathscr{C}^* = \chi(\mathscr{C})$

and $\mathscr{C}' = \mu(\mathscr{C})$ differ only by a monomial map fixing the coordinates indexed by J. In other words, χ and μ act in the same way on the information set J. Verification is performed by computing RREFs for both codes $\mathscr{C}'$ and $\mathscr{C}^*$ and, then, permuting and scaling the columns of the non-systematic parts so that the new columns are in lexicographic ordering.

This new notion of equivalence can, again, be described using the machinery we have introduced in the previous section. Indeed, let $F \leq M_n$ be the subgroup of monomials that fix the coordinates indexed by J. This subgroup is isomorphic to $\{id_k\} \times M_{n-k}$ and so $|F| = (n-k)!(q-1)^{n-k}$. Recalling (2), the number of cosets is now given by

$$[E:F] = \frac{|M_n|}{|F|} = \frac{n!(q-1)^n}{(n-k)!(q-1)^{n-k}} = (q-1)^k \cdot \prod_{i=0}^{k-1}(n-i).$$

Each coset can be represented by the isometry χ that fixes the entries not indexed by J. Note that all codes in the equivalence class $\mathfrak{C}_F(\mathscr{C}')$ differ only by a monomial transformation in the coordinates not indexed by J. Then, we obtain a canonical representative by considering the code whose generator matrix in RREF is such that the columns which are not indexed by J are in lexicographic ascending ordering.

We observe that such a definition for a canonical representative satisfies all the requirements in Definition 3. Moreover, the canonical representative exists for all F-equivalence classes.

Remark 4. *Similarly to the cases of LEP and PEP, we can enrich F by considering the trivial automorphisms of the code. This would enlarge F by a factor $q-1$ and, consequently, would lead to a reduction in the number of cosets by the same factor.*

Going Beyond IS-PEP and IS-LEP

In principle, our framework can be applied to any choice of F. Obviously, the main goal is to find subgroups F which are as large as possible and, at the same time, guarantee that canonical representatives can be found efficiently. In this paper, we consider progressively larger choices of F until we obtain witnesses of size $[E:F] = \binom{n}{k}$. As we show later, this choice is optimal in many cases (when q is large enough), in the sense that any larger set would imply an attack on LEP. We start in the next section by describing how canonical representatives can be defined and computed.

4 Efficient Canonical Representatives

In this section, we introduce some new canonical representatives and describe how they can be computed efficiently. For the sake of simplicity, in this section, we focus only on the case of linear codes for which $\{1, \cdots, k\}$ is an information set. This assumption allows to simplify the description of how canonical representatives can be computed efficiently. It is important to point out that any code can be permuted so that $\{1, \cdots, k\}$ is an information set: hence, our simplification still allows to compute canonical representatives starting from any code.

4.1 Computing Canonical Representatives

The first thing to keep in mind, is that essentially all operations on linear codes boil down to linear algebra. Thus, it is not surprising that, operatively, we compute a canonical representative for a code $\mathscr{C}$ by performing some operations on a generator matrix for $\mathscr{C}$. These operations consist in first bringing the matrix in systematic form, and then scaling and permuting the rows and columns of the non systematic portion of the matrix. We formalize this with the notion of a *canonical form function*.

To begin, we fix some notation. Consider an isometry ψ, acting on n elements. As we said, when ψ acts on codewords, one can see it as a map $\psi : \mathbb{F}_q^n \to \mathbb{F}_q^n$; however, it is common to slightly abuse this notation to describe the application of this map to other objects. For instance, in the previous section we have used $\psi(\mathscr{C})$ to indicate the code obtained by applying ψ to all the codewords of $\mathscr{C}$. Furthermore, if $\mathbf{G}$ is a $k \times n$ generator matrix for a code $\mathscr{C}$, we have also used $\psi(\mathbf{G})$ to indicate the matrix obtained by applying ψ to its columns; formally then, we would have $\psi : (\mathbb{F}_q^k)^n \to (\mathbb{F}_q^k)^n$. Later, we will need to apply this map to just a selection of the columns of $\mathbf{G}$, which would again change its definition. All in all, to avoid confusion, we deem it easier, in certain situations, to represent each map via the associated matrix $\mathbf{M}$, and its action via multiplication (i.e. $\psi(\mathbf{G}) = \mathbf{G} \cdot \mathbf{M}$). This additionally allows us to operate smoothly on rows, as well as columns, by multiplying the matrix on the left or on the right[9].

We focus on subgroups F with the following property: there exist $F_r \le M_k$ and $F_c \le M_{n-k}$ such that any $\varphi \in F$ is associated to a unique pair $(\varphi_r, \varphi_c) \in F_r \times F_c$ as follows: φ_r acts only the first k elements and φ_c acts only on the last $n-k$ elements.[10] In practice we will often apply φ_r to the rows of a matrix, and φ_c to the columns, which explains the choice of subscripts. We will frequently represent φ_r and φ_c via their corresponding matrices, respectively $\mathbf{M}_r$ and $\mathbf{M}_c$. Each isometry in F is associated with a monomial matrix in the form

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_c \end{pmatrix}, \qquad \mathbf{M}_r \in F_r, \quad \mathbf{M}_c \in F_c.$$

It is easy to see that F is isomorphic to $F_r \times F_c$. Finally, we define the action of $F_r \times F_c$ on the set of $k \times (n-k)$ matrices as follows: for $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$ and $\psi := (\mathbf{M}_r, \mathbf{M}_c) \in F_r \times F_c$, we have $\psi(\mathbf{A}) = \mathbf{M}_r \cdot \mathbf{A} \cdot \mathbf{M}_c$.

Similarly to equivalence classes for codes, one may define equivalence classes (or orbits) induced by the action of $F_r \times F_c$ on $\mathbb{F}_q^{k\times(n-k)}$. In other words, for any $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$, the orbit would correspond to $\{\mathbf{M}_r \cdot \mathbf{A} \cdot \mathbf{M}_c \mid (\mathbf{M}_r, \mathbf{M}_c) \in F_r \times F_c\}$. Then, a canonical form for a matrix $\mathbf{A}$ can be defined to be nothing but a representantive for such a orbit.

[9] Provided the dimensions are correct, of course.

[10] Put it differently, each isometry in F is the combination of a unique pair of isometries, one acting as φ_c on the first k coordinates and fixing the last $n-k$ coordinates, the other one fixing the first k coordinates and acting as φ_c on the last $n-k$ coordinates.

Definition 4 (Canonical Form Function). *Let $F \leq M_n$ be a subgroup of isometries, isomorphic to $F_r \times F_c$ as above. We define a function $\mathsf{CF}_F : \mathbb{F}_q^{k\times(n-k)} \to \left\{\{\perp\} \cup \mathbb{F}_q^{k\times(n-k)}\right\}$ such that:*

i) *the running time is polynomial in n and $\log_2(q)$;*

ii) *for any $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$ such that $\mathsf{CF}_F(\mathbf{A}) \neq \perp$, then there exist some $\psi \in F_r \times F_c$ such that*

$$\mathsf{CF}_F(\mathbf{A}) = \psi(\mathbf{A});$$

iii) *for any $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$ such that $\mathsf{CF}_F(\mathbf{A}) \neq \perp$, then for all $\psi \in F_r \times F_c$ it holds that*

$$\mathsf{CF}_F(\mathbf{A}) = \mathsf{CF}_F\big(\psi(\mathbf{A})\big).$$

We naturally extend the notion of canonical forms to a generator matrix.

Definition 5 (Generator Matrix in Canonical Form). *Let $\mathscr{C} \subseteq \mathbb{F}_q^n$ be a linear code with dimension k. For $F \leq M_n$ being a group of isometries as above, let CF_F be a function satisfying the requirements in Definition 4. Then, we say that a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ is a generator matrix for $\mathscr{C}$* in canonical form *if:*

i) *$\mathbf{G}$ is in systematic form, i.e., $\mathbf{G} = (\mathbf{I}_k \mid \mathbf{A})$ with $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$;*

ii) *$\mathbf{A} = \mathsf{CF}_F(\mathbf{A})$.*

We now prove a fundamental result which shows that a well-defined canonical form function (as per Definition 4) immediately gives a well-defined canonical representative.

Theorem 1. *Let CF_F be a canonical form function according to Definition 4. For a k-dimensional code $\mathscr{C} \subseteq \mathbb{F}_q^n$ having $\{1, \cdots, k\}$ as an information set, we define the canonical representative $\mathsf{CR}_F(\mathscr{C})$ as the function that:*

i) *computes the systematic generator matrix for $\mathscr{C}$, i.e. the matrix $\mathbf{G} = (\mathbf{I}_k \mid \mathbf{A})$;*

ii) *if $\mathsf{CF}_F(\mathbf{A}) = \perp$, returns a failure; else, returns the code generated by $\big(\mathbf{I}_k \mid \mathsf{CF}_F(\mathbf{A})\big)$.*

Then, CR_F satisfies the requirements i) – v) from Definition 3.

Proof. First, observe that requirements *i) – iii)* from Definition 3 are trivially satisfied. We now show that requirement *iv)* is satisfied as well. To this end, we must show that the code $\mathsf{CR}_F(\mathscr{C})$ generated by $\big(\mathbf{I}_k \mid \mathsf{CF}_F(\mathbf{A})\big)$ is F-equivalent to $\mathscr{C}$. This is immediate since, by definition, there exists $\psi = (\mathbf{M}_r, \mathbf{M}_c) \in F_r \times F_c$ such that $\mathsf{CF}_F(\mathbf{A}) = \psi(\mathbf{A}) = \mathbf{M}_r \cdot \mathbf{A} \cdot \mathbf{M}_c$; then

$$\begin{aligned}\big(\mathbf{I}_k \mid \mathsf{CF}_F(\mathbf{A})\big) = \big(\mathbf{I}_k \mid \psi(\mathbf{A})\big) = \big(\mathbf{I}_k \mid \mathbf{M}_r \cdot \mathbf{A} \cdot \mathbf{M}_c\big) &= \mathbf{M}_r \cdot \big(\mathbf{M}_r^{-1} \cdot \mathbf{I}_k \mid \mathbf{A} \cdot \mathbf{M}_c\big) \\ &= \mathbf{M}_r \cdot (\mathbf{I}_k \mid \mathbf{A}) \cdot \overline{\mathbf{M}}\end{aligned}$$

where $\overline{\mathbf{M}} = \begin{pmatrix} \mathbf{M}_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_c \end{pmatrix}$.

Now, since $\mathbf{M}_r$ represents an invertible map applied to the rows of a matrix, we have that $\mathbf{M}_r \cdot (\mathbf{I}_k \mid \mathbf{A}) = \mathbf{M}_r \cdot \mathbf{G}$ generates the same code as $\mathbf{G}$; on the other hand, $\overline{\mathbf{M}}$ represents an isometry $\varphi = (\varphi_r^{-1}, \varphi_c) \in F$, and thus we obtain a code that is F-equivalent to the one generated by $(\mathbf{I}_k \mid \mathsf{CF}_F(\mathbf{A}))$, which is our thesis.

To conclude the proof, we need to show that for any two codes in the same F-equivalence class, $\mathsf{CR}_F(\mathscr{C})$ returns the same code. Now, any code $\mathscr{C}' \in \mathfrak{C}_F(\mathscr{C})$ admits a generator matrix in the form $\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{M}_\varphi$, where $\mathbf{G} = (\mathbf{I}_k \mid \mathbf{A})$, $\mathbf{S} \in \mathrm{GL}_k(q)$ and $\mathbf{M}_\varphi$ is the matrix associated to $\varphi = (\varphi_r, \varphi_c) \in F$. It follows that $\mathbf{G}' = (\mathbf{S} \cdot \mathbf{M}_r \mid \mathbf{S} \cdot \mathbf{A} \cdot \mathbf{M}_c)$. Bringing $\mathbf{G}'$ into systematic form, we obtain a matrix $(\mathbf{I}_k \mid \mathbf{A}')$ with

$$\mathbf{A}' = (\mathbf{S} \cdot \mathbf{M}_r)^{-1} \cdot \mathbf{S} \cdot \mathbf{A} \cdot \mathbf{M}_c = \mathbf{M}_r^{-1} \cdot \mathbf{A} \cdot \mathbf{M}_c.$$

Then, by the definition of CF_F we have that $\mathsf{CF}_F(\mathbf{A}') = \mathsf{CF}_F(\mathbf{A})$ as desired. □

4.2 Failure Probability

Recall that the definition of canonical representatives takes into account that, for some codes, a representative for the F-equivalence class may not be computed. When CR_F is defined according to Theorem 1, this happens whenever $\mathsf{CF}_F(\mathbf{A}) = \perp$, with $\mathbf{A}$ being the non-systematic part of the systematic generator matrix. In the rest of the paper, we denote with γ the probability that CF_F exists, when the input matrix is uniformly distributed over $\mathbb{F}_q^{k\times(n-k)}$. We use this probability to estimate the ratio between the number of codes for which the canonical representative exists and the overall number of codes. Put it differently, γ is an estimate for the probability that, for a uniformly random code, a canonical representative can be computed. This is coherent since, in practice, uniform sampling of random codes having $\{1, \cdots, k\}$ as an information set is achieved by just sampling their systematic generator matrix $(\mathbf{I}_k \mid \mathbf{A})$ which, in turns, implies sampling uniformly at random $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$.

We finally note that, in other contexts, canonical forms are defined similarly to Definition 4 (obviously, using different objects and notions of isometries) but are required to be defined for all inputs. This is not the case for the canonical forms we consider in this paper; hence, our canonical forms may be deemed as *improper*. Nevertheless, we show that for our canonical forms, $\gamma = \Omega(1)$ when q is sufficiently large (formally, grows with n): in practice, representatives can be found for the majority of codes hence our canonical forms are almost proper. For the purposes of this paper, this is sufficient.

It is not hard to see that there is an inherent difficulty in finding a proper canonical form for the cases we are interested in. For instance, for the very first case (which we will present in the next subsection), finding a proper canonical form would imply existence of a polynomial-time algorithm deciding Graph Isomorphism. In other words, in such a case, finding a canonical form function with $\gamma = 0$ is a GI-hard problem.

The following sections show canonical form functions for several F's. The remaining sections of the paper explain how to make use of the resulting canonical representatives.

4.3 Case 1

We start with the case $F_1 \simeq S_k \times S_{n-k}$. The isometries in this group are of the form $\varphi = (\varphi_r, \varphi_c) = (\mathbf{P}_r, \mathbf{P}_c)$, where $\mathbf{P}_r \in S_k$ and $\mathbf{P}_c \in S_{n-k}$ are respectively row permutations, and column permutations. Then, we have

$$[M_n : F_1] = \frac{|M_n|}{|F_1|} = \frac{n!(q-1)^n}{k!(n-k)!} = \binom{n}{k}(q-1)^n.$$

We simplify notation and refer to the canonical form function for this case as $\mathsf{CF}^{(1)}$.[11] The function outputs either $\perp$, or a matrix in $\mathbb{F}_q^{k\times(n-k)}$ such that the rows are sorted w.r.t. $\prec_{\mathbb{F}_q^{n-k}}$ and the columns are sorted w.r.t. $\leq_{\mathbb{F}_q^k}$, if it exists in the orbit of the input matrix. Note that such a matrix might not exist in the orbit. This happens only when there are two distinct rows that cannot be compared (because they lead to the same multiset).

Algorithm 1: $\mathsf{CF}^{(1)}$: Canonical form computation for F_1

Input: $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$
Output: $\mathbf{A}' \in \mathbb{F}_q^{k\times(n-k)}$ or $\perp$

```
1 Set A' = A;
  /* Sort rows                                                    */
2 Sort the rows in A' w.r.t. ≺_{F_q^{n-k}}; if two rows v and v' are such that v ≠ v'
    and multiset(v) = multiset(v'), return ⊥;
  /* Sort columns                                                 */
3 Sort the columns in A' w.r.t. ≤_{F_q^k};
4 return A'
```

For any matrix, one can derive the corresponding canonical form by sorting first the rows using the partial ordering, and then the columns using the total ordering. It is easy to detect whether the corresponding canonical form exists or not, while sorting rows. The pseudocode of the algorithm is shown in Algorithm 1. Similarly, one can instead define the canonical form as the result of sorting first the columns using $\prec_{\mathbb{F}_q^k}$, and then sorting the rows using $\leq_{\mathbb{F}_q^{n-k}}$, which will lead to a different canonical form function. An example of how this canonical form is computed is shown in Figure 4. We analyze the main aspects of this canonical form below.

Correctness

The function is invariant under row and column permutations: if $\mathsf{CF}^{(1)}(\mathbf{A}) = \mathbf{A}' \neq \perp$, then $\mathbf{A}' = \mathsf{CF}^{(1)}(\mathbf{P}_r \cdot \mathbf{A} \cdot \mathbf{P}_c)$, for any $\mathbf{P}_r \in S_k$ and $\mathbf{P}_c \in S_{n-k}$. Indeed, $\mathsf{multiset}(\mathbf{v}) = \mathsf{multiset}(\mathbf{v} \cdot \mathbf{P}_c)$ for any $\mathbf{v} \in \mathbb{F}_q^{n-k}$ and any $\mathbf{P}_c \in S_{n-k}$. Thus, the multisets formed by the rows of $\mathbf{A}$ and $\mathbf{P}_r \cdot \mathbf{A} \cdot \mathbf{P}_c$ are the same, up to a different ordering due to $\mathbf{P}_r$.

[11] Following the notation we settled in the previous section, this should have been CF_{F_1}.

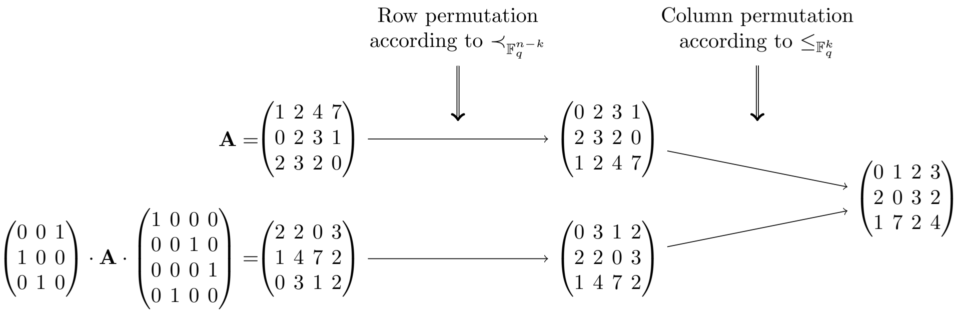

Fig. 4: Example of computation of $\mathsf{CF}^{(1)}$, for $n = 7$ and $k = 3$.

Sorting the rows of $\mathbf{A}$ and $\mathbf{P}_r \cdot \mathbf{A} \cdot \mathbf{P}_c$ with respect to $\prec_{\mathbb{F}_q^{n-k}}$ leads to two matrices which are equal up to a column permutation. Finally, sorting with respect to $\leq_{\mathbb{F}_q^k}$ is invariant under column permutation.

Computational Complexity

Algorithm 1 takes $\tilde{O}(n^2)$ field operations, assuming that the rows and columns are sorted using a comparison-based sorting algorithm that takes an essentially linear number of comparisons.

Success Probability

According to Appendix B, the success probability of $\mathsf{CF}^{(1)}$ is lower bounded by

$$\gamma_1^*(q, k, n-k) = 1 - \left(\prod_{i=1}^{k-1} 1 - \frac{i \cdot m}{q^{n-k}} \right) \tag{3}$$

where

$$m = \begin{cases} (n-k)! & \text{if } n-k \leq q, \\ \frac{(n-k)!}{\left(v!\right)^{q(v+1)-(n-k)} \left((v+1)!\right)^{n-k-qv}} & \text{if } n-k > q \end{cases}$$

and $v = \lfloor (n-k)/q \rfloor$. A proof for the bound can be found in Appendix B.

Theorem 2. *Let $k = R \cdot n$, with R being constant. Let S be a positive constant such that $q \geq S \cdot n$ (which implies that $q = \Omega(n)$) and $S/(1-R) > 1/e$. Then, $\gamma_1^*(q, k, n-k) = \Omega(1)$.*

Proof. Below, by $f \gtrsim g$, we mean $g = O(f)$. We also define $[f]$ as the minimum value between 1 and f. Consider that

$$\gamma_1^*(q, k, n-k) \geq \prod_{i=1}^{k-1} 1 - \left[\frac{i \cdot (n-k)!}{q^{n-k}} \right] \geq \left(1 - \left[\frac{k \cdot (n-k)!}{q^{n-k}} \right] \right)^k$$

$$\geq \left(1 - \left[\frac{Rn \cdot ((1-R)n)!}{(Sn)^{(1-R)n}}\right]\right)^{Rn}.$$

It is known that $n! < \sqrt{2\pi n} \cdot (\frac{n}{e})^n \cdot e^{\frac{1}{12n}}$ for all $n \geq 1$, hence we continue the above chain of inequalities as

$$\gamma_1^*(q, k, n-k) \gtrsim \left(1 - \frac{Rn \cdot \sqrt{2\pi(1-R)n} \cdot e^{\frac{1}{12(1-R)n}}}{\left(\frac{eS}{1-R}\right)^{(1-R)n}}\right)^{Rn} \gtrsim \left(\left(1 - \frac{1}{n}\right)^n\right)^R = \Omega(1),$$

where we have used that $\lim_{x\to\infty}((x-1)/x)^x = e^{-1}$. □

4.4 Case 2

We now move on to the case $F_2 \simeq M_k \times S_{n-k}$.[12] Here we have isometries of the form $\varphi = (\varphi_r, \varphi_c) = (\mathbf{Q}, \mathbf{P})$, where $\mathbf{Q} \in M_k$ and $\mathbf{P} \in S_{n-k}$. Following the same steps as in the previous case, we have

$$[M_n : F_2] = \frac{|M_n|}{|F_2|} = \frac{n!(q-1)^n}{k!(n-k)!(q-1)^k} = \binom{n}{k}(q-1)^{n-k}.$$

Compared to the previous case, $[E : F_2]$ is smaller than $[E : F_1]$ by a factor $(q-1)^k$, which is exactly the ratio between $|F_2|$ and $|F_1|$.

We now define the corresponding canonical form function $\mathsf{CF}^{(2)}$. The function outputs either $\perp$ or a matrix in $\mathbb{F}_q^{k\times(n-k)}$ with the following properties.

1. For each row $\mathbf{v}$ of the form $(\alpha, \ldots, \alpha)$, it must be $\alpha \in \{0, 1\}$.
2. For each row $\mathbf{v}$ not of the form $(\alpha, \ldots, \alpha)$, it must be either $\sum_i v_i = 1$ or $\sum_i v_i = 0$ and $\sum_i v_i^{q-2} = 1$ (here, v_i is the i-th element of $\mathbf{v}$).
3. The rows and columns are sorted as in $\mathsf{CF}^{(1)}$.

To derive such a canonical form, one can carry out one step to ensure that the first two constraints hold, and then another to ensure that the third constraint holds. The second step can be carried out by Algorithm 2. The first step can be carried out as follows. For each row $\mathbf{v} \in \mathbb{F}_q^{n-k}$, if $\mathbf{v}$ is of the form $(\alpha, \ldots, \alpha)$ where $\alpha \in \mathbb{F}_q^*$, replace the row by $(1, \ldots, 1)$. If $\mathbf{v}$ is not of the form $(\alpha, \ldots, \alpha)$, compute $(s, s') = (\sum_i v_i, \sum_i v_i^{q-2})$. If $s \neq 0$, replace the row by $s^{-1} \cdot \mathbf{v}$. If $s = 0$ and $s' \neq 0$, replace the row by $s' \cdot \mathbf{v}$. If $s = s' = 0$, return $\perp$.

Pseudocode of the algorithm is shown in Algorithm 2. Figure 5 shows an example of how the canonical form is computed.

[12] Note that the case $F_2 \simeq S_k \times M_{n-k}$ can be defined and treated analogously.

Algorithm 2: $\mathsf{CF}^{(2)}$: Canonical form computation for F_2

Input: $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$
Output: $\mathbf{C} \in \mathbb{F}_q^{k\times(n-k)}$ *or* $\perp$

Set $\mathbf{A}' = \mathbf{A}$;
for $i = 1$ **to** k **do**
  Set $\mathbf{v}$ as row i of $\mathbf{A}'$, compute $(s, s') = (\sum_\ell v_\ell, \sum_\ell v_\ell^{q-2})$;
  /* Scale row only if it is not $(0, \cdots, 0)$ */
  if $\mathbf{v} \neq (0, \ldots, 0)$ **then**
    if $\mathbf{v} = \alpha \cdot (1, \ldots, 1)$ for some $\alpha \in \mathbb{F}_q^*$ **then**
      replace row i of $\mathbf{A}'$ by $(1, \ldots, 1)$;
    else
      If $s \neq 0$, replace row i of $\mathbf{A}'$ by $s^{-1} \cdot \mathbf{v}$;
      If $s = 0$ and $s' \neq 0$, replace row i of $\mathbf{A}'$ by $s' \cdot \mathbf{v}$;
      If $s = s' = 0$, **return** $\perp$;
return $\mathsf{CF}^{(1)}(\mathbf{A}')$;

$$\mathbf{A} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 1 & 0 & 3 & 4 \end{pmatrix} \xrightarrow[\text{3rd row: } (s,s') = (3,2)]{\substack{\text{1st row: } (s,s') = (2,2) \\ \text{2nd row: } (s,s') = (0,1)}} \begin{pmatrix} 2^{-1} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3^{-1} \end{pmatrix} \cdot \begin{pmatrix} 2 & 3 & 1 & 1 \\ 2 & 1 & 1 & 1 \\ 1 & 0 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 3 \\ 2 & 1 & 1 & 1 \\ 2 & 0 & 1 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 4 & 3 & 3 \\ 2 & 1 & 1 & 1 \\ 2 & 0 & 1 & 3 \end{pmatrix} \xrightarrow{\mathsf{CF}^{(1)}} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 1 & 2 & 1 \\ 4 & 3 & 1 & 3 \end{pmatrix} \xleftarrow{\mathsf{CF}^{(1)}} \begin{pmatrix} 1 & 1 & 1 & 2 \\ 3 & 4 & 3 & 1 \\ 1 & 0 & 3 & 2 \end{pmatrix}$$

$$\mathbf{Q} \cdot \mathbf{A} \cdot \mathbf{P} = \begin{pmatrix} 1 & 1 & 1 & 2 \\ 4 & 2 & 4 & 3 \\ 4 & 0 & 2 & 3 \end{pmatrix} \xrightarrow[\text{3rd row: } (s,s') = (4,4)]{\substack{\text{1st row: } (s,s') = (0,1) \\ \text{2nd row: } (s,s') = (3,3)}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3^{-1} & 0 \\ 0 & 0 & 4^{-1} \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 2 \\ 4 & 2 & 4 & 3 \\ 4 & 0 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 2 \\ 3 & 4 & 3 & 1 \\ 1 & 0 & 3 & 2 \end{pmatrix}$$

Fig. 5: Example of computation of $\mathsf{CF}^{(2)}$, for $n = 7$ and $k = 3$, over $\mathbb{F}_5$. In the example, we have $\mathbf{Q} \in M_k$ given by $\left(\begin{smallmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right) \cdot \left(\begin{smallmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{smallmatrix}\right)$ and $\mathbf{P} \in S_{n-k}$ by $\left(\begin{smallmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{smallmatrix}\right)$.

Correctness

Let $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$ and $\widetilde{\mathbf{A}} = \mathbf{Q} \cdot \mathbf{A} \cdot \mathbf{P}$, with $\mathbf{Q} \in M_k$ and $\mathbf{P} \in S_{n-k}$. We show 1) that $\mathsf{CF}^{(2)}(\mathbf{A}) = \mathsf{CF}^{(2)}(\widetilde{\mathbf{A}})$ for any choice of $\mathbf{Q} \in M_k$ and $\mathbf{P} \in S_{n-k}$, and 2) that the canonical form is obtained by applying a monomial on the left and a permutation on the right. Actually, it suffices to show that, on input $\mathbf{A}$ and $\widetilde{\mathbf{A}}$, instructions 2–9 will return two matrices $\mathbf{A}'$ and $\widetilde{\mathbf{A}}'$, obtained just by scaling rows and such that $\widetilde{\mathbf{A}}' = \mathbf{P}_r \cdot \mathbf{A}' \cdot \mathbf{P}_c$ for some $\mathbf{P}_r \in S_k$ and $\mathbf{P}_c = \mathbf{P}$. In other words, we can reduce to the previous instance (Case 1) as by construction we have that $\mathsf{CF}^{(1)}(\mathbf{A}') = \mathsf{CF}^{(1)}(\widetilde{\mathbf{A}}')$.

Let $d_i \neq 0$ denote the i-th non-zero scaling factor of $\mathbf{Q}$. Let j be the index which is moved to position i by $\mathbf{P}_r$: then, if we denote by $\mathbf{v}$ and $\widetilde{\mathbf{v}}$ the j-th row of $\mathbf{A}$ and the i-th row of $\widetilde{\mathbf{A}}$, respectively, it holds that

$$\widetilde{\mathbf{v}} = d_i \cdot \mathbf{v} \cdot \mathbf{P}_c.$$

We now consider how instructions 2–9 would modify these two rows.

If $\mathbf{v}$ is null, then also $\widetilde{\mathbf{v}}$ is null and the rows are not updated. If $\mathbf{v} = (\alpha, \ldots, \alpha)$, then $\widetilde{\mathbf{v}} = d_i \cdot (\alpha, \ldots, \alpha)$. Both rows are replaced by $(1, \ldots, 1)$: for $\mathbf{v}$, this corresponds to a multiplication by α^{-1}, while for $\widetilde{\mathbf{v}}$ this is obtained via multiplication by $d_i^{-1} \cdot \alpha^{-1}$.

We now consider the case in which both $\mathbf{v}$ and $\widetilde{\mathbf{v}}$ do not fall into the two previous cases. Let (s, s') be the values computed in line 6 for row $\mathbf{v}$, and $(\widetilde{s}, \widetilde{s}')$ be the ones computed for $\widetilde{\mathbf{v}}$. We first consider the case in which $s \neq 0$, and observe that

$$\widetilde{s} = \sum_{\ell} \widetilde{v}_\ell = d_i \cdot \sum_{\ell} v_\ell = d_i \cdot s.$$

Note that, in the above equation, we have exploited the fact that $\mathbf{P}_c$ does not affect computation of $\mathbf{s}$ and $\widetilde{\mathbf{s}}$ (it only changes the order in which elements are summed). Since $s \neq 0$, also $\widetilde{s} \neq 0$. The row $\mathbf{v}$ is replaced by $s^{-1} \cdot \mathbf{v}$, while row $\widetilde{\mathbf{v}}$ is replaced by

$$\widetilde{s}^{-1} \cdot \widetilde{\mathbf{v}} = (d_i \cdot s)^{-1} \cdot d_i \cdot \mathbf{v} \cdot \mathbf{P}_c = s^{-1} \cdot \mathbf{v} \cdot \mathbf{P}_c.$$

Thus, $\mathbf{v}$ and $\widetilde{\mathbf{v}}$ lead to two rows which are equal, up to the permutation $\mathbf{P}_c$.
The case $s = 0$, $\widetilde{s} \neq 0$ is studied analogously, by noticing that

$$\widetilde{s}' = \sum_{\ell} \widetilde{v}_\ell^{q-2} = d_i^{q-2} \cdot \sum_{\ell} v_\ell^{q-2} = d_i^{q-2} \cdot s'$$

and

$$\widetilde{s}' \cdot \widetilde{\mathbf{v}} = d_i^{q-2} \cdot s' \cdot d_i \cdot \mathbf{v} \cdot \mathbf{P}_c = \underbrace{d_i^{q-1}}_{=1} \cdot s' \cdot \mathbf{v} \cdot \mathbf{P}_c = s' \cdot \mathbf{v} \cdot \mathbf{P}_c.$$

Note that $d_i^{q-1} = 1$, for any d_i, since we are working in a finite field with q elements. Thus, also in this case, $\mathbf{v}$ and $\widetilde{\mathbf{v}}$ lead to the same row, up to the permutation $\mathbf{P}_c$.

The above reasoning applies to all rows of $\mathbf{A}$ and $\widetilde{\mathbf{A}}$. Since row updates are always obtained by applying some scaling factor, overall the effect of instructions 2–9 is described by a diagonal acting on the left.

Finally, if some row of $\mathbf{A}$ has $s = s' = 0$, then also in $\widetilde{\mathbf{A}}$ there is a row yielding $\widetilde{s} = \widetilde{s}' = 0$ thus $\mathsf{CF}^{(2)}$ fails for both $\mathbf{A}$ and $\widetilde{\mathbf{A}}$.

Computational Complexity

The loop in Algorithm 2 takes $O(n^2)$ field operations. In particular, each iteration of Line 3 takes $O(n)$ field operations, as $x^{q-2} = x^{-1}$ for any $x \in \mathbb{F}_q^*$. Line 11 takes $\tilde{O}(n^2)$ field operations, so the whole algorithm takes $\tilde{O}(n^2)$ field operations.

Success Probability

We claim that the success probability of $\mathsf{CF}^{(2)}$ is lower bounded by

$$\gamma_2^*(q,k,n-k) = \left(1-\frac{1}{q}\right)^k \cdot \gamma_1'(q,k,n-k)$$

where $\gamma_1'(q,k,n-k) = \prod_{i=1}^{k-1} 1 - [\frac{i\cdot m\cdot(q-1)}{q^{n-k}}]$. The term $(1-\frac{1}{q})^k$ is a lower bound on the probability that every row $\mathbf{v}$ is either of the form $(\alpha,\dots,\alpha)$ or such that $\sum_i v_i \neq 0$ or $\sum_i v_i^{-1} \neq 0$. The term $\gamma_1'(q,k,n-k)$ is a lower bound on the probability that $\mathsf{CF}^{(1)}(\mathbf{A}') \neq \perp$: each row can be scaled in at most $q-1$ ways in the loop, so we add the term $q-1$ to account for the possibility that two rows lead to the same multiset after scaling.

Theorem 3. *When the same hypothesis as in Theorem 2 holds,* $\gamma_2^*(q,k,n-k) = \Omega(1)$*.*

Proof. We first observe that

$$\left(1-\frac{1}{q}\right)^k \gtrsim \left(\frac{Sn-1}{Sn}\right)^{Rn} = \left(\left(\frac{Sn-1}{Sn}\right)^{Sn}\right)^{\frac{R}{S}}.$$

Since $\lim_{x\to\infty}((x-1)/x)^x = 1/e$, we have $(1-\frac{1}{q})^k = \Omega(1)$. It is easy to show that $\gamma_1'(q,k,n-k) = \Omega(1)$, by mimicking the proof for $\gamma_1^*(q,k,n-k) = \Omega(1)$. □

$\gamma_2^*(q,k,n-k)$ is expected to be a loose bound: The probability that $\perp$ is returned in each iteration of the loop in Algorithm 2 should be close to $1/q^2$. Therefore, we estimate the success probability of Algorithm 2 as

$$\gamma_2(q,k,n-k) = \left(1-\frac{1}{q^2}\right)^k \cdot \gamma_1'(q,k,n-k).$$

4.5 Case 3

To conclude, we consider what we deem the most general case, which corresponds to the largest choice for F, that is $F_3 \simeq M_k \times M_{n-k}$. We now have isometries of the form $\varphi = (\varphi_r, \varphi_c) = (\mathbf{Q}_r, \mathbf{Q}_c)$, where $\mathbf{Q}_r \in M_k$ and $\mathbf{Q}_c \in M_{n-k}$[13]. The impact of this last choice for F is extremely significant, as now the witness size is given by

[13] In this section, it will be useful to remember that each monomial matrix $\mathbf{Q} \in M_r$ can be seen as a product $\mathbf{Q} = \mathbf{P} \cdot \mathbf{D}$ of a permutation $\mathbf{P} \in S_r$ and a diagonal matrix $\mathbf{D} \in D_r$.

$$[M_n : F_3] = \frac{|M_n|}{|F_3|} = \frac{n!(q-1)^n}{k!(n-k)!(q-1)^k(q-1)^{n-k}} = \binom{n}{k}.$$

We now given an intuition on how a canonical form for this case can be derived.

Compared to the previous case, we now have to deal with an additional scaling of columns, i.e., column i is now scaled by some non-null coefficient d_i. For each row $(a_{i,1}, \cdots, a_{i,n-k})$ which does not contain zeros (i.e., $a_{i,j} \neq 0$ for every $j \in \{1, \cdots, n-k\}$), one can scale column j by $a_{i,j}^{-1}$. The effect of extra scaling factors is canceled. To see this, let us assume that

$$\mathbf{A} = \mathbf{B} \cdot \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_{n-k} \end{pmatrix}$$

which implies $a_{i,j} = d_j \cdot b_{i,j}$. When we scale column j by $a_{i,j}^{-1} = d_j^{-1} \cdot b_{i,j}^{-1}$, we get rid of the dependence on d_j.

Let $\mathbf{A}^{(i)}$ be the matrix obtained after columns of $\mathbf{A}$ are scaled as we described above. Note that row i of $\mathbf{A}^{(i)}$ is $(1, \cdots, 1)$. Then, for each $\mathbf{A}^{(i)}$, Algorithm 2 is applied to obtain a matrix $\mathbf{C}^{(i)}$ (if $\perp$ is not returned). Finally, return the smallest $\mathbf{C}^{(i)}$ with respect to $<_{\mathbb{F}_q^{k\times(n-k)}}$. Pseudocode of the algorithm is shown in Algorithm 3. The resulting canonical form function is denoted as $\mathsf{CF}^{(3)}$.

Algorithm 3: $\mathsf{CF}^{(3)}$: Canonical form computation for F_3

Input: $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$
Output: $\mathbf{C} \in \mathbb{F}_q^{k\times(n-k)}$ *or* $\perp$

Set $T \leftarrow \varnothing$; // E is initialized as the empty set
for $i = 1$ **to** k **do**
  /* If row i does not have zeros, use the inverses of its coefficients as scaling factors and call $\mathsf{CF}^{(2)}$ */
  if $0 \notin \{a_{i,1}, \ldots, a_{i,n-k}\}$ **then**
    $\mathbf{A}^{(i)} \leftarrow \mathbf{A} \cdot \mathbf{D}$, where $\mathbf{D} \in D_{n-k}$ and $d_{j,j} = a_{i,j}^{-1}$ for $j \in \{1, \ldots, n-k\}$;
    $\mathbf{C}^{(i)} \leftarrow \mathsf{CF}^{(2)}(\mathbf{A}^{(i)})$;
    if $\mathbf{C}^{(i)} \neq \perp$ **then**
      $T \leftarrow T \cup \{\mathbf{C}^{(i)}\}$;
/* If $T = \varnothing$, return $\perp$; else, $\mathsf{CF}^{(3)}$ can be computed */
if $T = \varnothing$ **then**
  return $\perp$;
else
  return the smallest element in T w.r.t. $\leq_{\mathbb{F}_q^{k\times(n-k)}}$;

Correctness

To prove correctness of our algorithm, we need two preliminary results.

Proposition 4. *Given* $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{k\times(n-k)}$ *satisfying* $\mathbf{B} = \mathbf{D}_r \cdot \mathbf{A} \cdot \mathbf{D}_c$ *for some* $\mathbf{D}_r \in D_k$*,* $\mathbf{D}_c \in D_{n-k}$*, such that row* i *of* $\mathbf{A}$ *(and thus row* i *of* $\mathbf{B}$*) consists of only non-zero elements. Let*

$$\mathbf{A}^{(i)} = \mathbf{A} \cdot \begin{pmatrix} a_{i,1}^{-1} & & \\ & \ddots & \\ & & a_{i,n-k}^{-1} \end{pmatrix}, \qquad \mathbf{B}^{(i)} = \mathbf{B} \cdot \begin{pmatrix} b_{i,1}^{-1} & & \\ & \ddots & \\ & & b_{i,n-k}^{-1} \end{pmatrix}$$

Then $\mathbf{B}^{(i)} = \mathbf{D}'_r \cdot \mathbf{A}^{(i)}$ *for some* $\mathbf{D}'_r \in D_k$*.*

Proof. Let the elements on the main diagonal of $\mathbf{D}_r$ be $x_1, \ldots x_k$. Analogously, let those of $\mathbf{D}_c$ be $y_1, \ldots y_{n-k}$. Then, row j of $\mathbf{A}^{(i)}$ is given by $(a_{j,1} \cdot a_{i,1}^{-1}, \ldots, a_{j,n-k} \cdot a_{i,n-k}^{-1})$, while row j of $\mathbf{B}^{(i)}$ is just a scalar multiple of this row. Indeed, the elements of this row are given by:

$$\begin{aligned} &\big(\underbrace{x_j \cdot a_{j,1} \cdot y_1 \cdot (x_i \cdot a_{i,1} \cdot y_1)^{-1}}_{\text{Element } 1}, \cdots, \underbrace{x_j \cdot a_{j,n-k} \cdot y_{n-k} \cdot (x_i \cdot a_{i,n-k} \cdot y_{n-k})^{-1}}_{\text{Element } n-k}\big) \\ &= \big(x_j \cdot x_i^{-1} \cdot a_{j,1} \cdot a_{i,1}^{-1}, \cdots, x_j \cdot x_i^{-1} \cdot a_{j,n-k} \cdot a_{i,n-k}^{-1}\big) \\ &= (x_j \cdot x_i) \cdot \underbrace{\big(a_{j,1} \cdot a_{i,1}^{-1}, \cdots, a_{j,n-k} \cdot a_{i,n-k}^{-1}\big)}_{\text{Row } j \text{ of } \mathbf{A}^{(i)}}. \end{aligned}$$

□

Proposition 5. *Given* $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{k\times(n-k)}$ *satisfying* $\mathbf{A} = \mathbf{P}_r \cdot \mathbf{B} \cdot \mathbf{P}_c$ *for some* $\mathbf{P}_r \in S_k$*,* $\mathbf{P}_c \in S_{n-k}$*, such that row* i *of* $\mathbf{A}$ *consists of only non-zero elements. Let* i' *be the column index of* 1 *in row* i *of* $\mathbf{P}_r^{-1}$*. In other words, the row permutation represented by* $\mathbf{P}_r^{-1}$ *maps row* i' *to row* i*. Then* $\mathbf{A}^{(i)} = \mathbf{P}_r^{-1} \cdot \mathbf{B}^{(i')} \cdot \mathbf{P}_c^{-1}$*.*

The proof for Proposition 5 is immediate and therefore omitted in the interest of space. Now, combining the two above propositions, we have

$$\begin{aligned} \mathbf{B} = \mathbf{P}_r \cdot \mathbf{D}_r \cdot \mathbf{A} \cdot \mathbf{D}_c \cdot \mathbf{P}_c \implies \mathbf{P}_r^{-1} \cdot \mathbf{B} \cdot \mathbf{P}_c^{-1} = \mathbf{D}_r \cdot \mathbf{A} \cdot \mathbf{D}_c \\ \implies \mathbf{P}_r^{-1} \cdot \mathbf{B}^{(i')} \cdot \mathbf{P}_c^{-1} = \mathbf{D}'_r \cdot \mathbf{A}^{(i)} \implies \mathbf{B}^{(i')} = \mathbf{P}_r \cdot \mathbf{D}'_r \cdot \mathbf{A}^{(i)} \cdot \mathbf{P}_c. \end{aligned}$$

By applying the algorithm for Case 2 to $\mathbf{A}^{(i)}$ and $\mathbf{B}^{(i')}$, we obtain the same matrix. Thus, the set $\left\{\mathsf{CF}^{(2)}\big(\mathbf{A}^{(i)}\big) \mid i = 1, \cdots, k, \quad 0 \notin \{a_{i,1}, \cdots, a_{i,n-k}\}\right\}$ is equal to $\left\{\mathsf{CF}^{(2)}\big(\mathbf{B}^{(i)}\big) \mid i = 1, \cdots, k, \quad 0 \notin \{b_{i,1}, \cdots, b_{i,n-k}\}\right\}$, and we conclude that the algorithm leads to the same output for any $\mathbf{A}, \mathbf{B}$ in the same equivalence class.

Computational Complexity

The number of field operations taken by Algorithm 3 is dominated by the ones taken by iterations of Line 5. Therefore, the algorithm takes $n \cdot \tilde{O}(n^2) = \tilde{O}(n^3)$ field operations.

Success Probability

The success probability of $\mathsf{CF}^{(3)}$ is lower bounded by

$$\gamma_3^*(q, n, n) = \left(\frac{q-1}{q}\right)^{n-k} \cdot \gamma_2^*(q, k-1, n-k).$$

Here, $(\frac{q-1}{q})^{n-k}$ is the probability that the first row of $\mathbf{A}$ is in $(\mathbb{F}_q^*)^{n-k}$ and thus $\mathbf{A}^{(1)}$ is well-defined. $\gamma_2^*(q, k-1, n-k)$ is a lower bound on the probability that $\mathbf{C}^{(1)} \neq \perp$, under the condition that $\mathbf{A}^{(1)}$ is well-defined. We consider $\gamma_2^*(q, k-1, n-k)$ instead of $\gamma_2^*(q, k, n-k)$, as whether $\mathbf{C}^{(1)} = \perp$ depends only on entries not in the first row: The first row of $\mathbf{A}^{(1)}$ is $(1, \ldots, 1)$, so Line 10 of Algorithm 2 will be skipped when $i = 1$; Also, there cannot be a vector $\mathbf{v}$ such that $\mathbf{v} \neq (1, \ldots, 1)$ and $\text{multiset}(\mathbf{v}) = \{1, \ldots, 1\}$.

Theorem 6. *$\gamma_3^*(q, k, n-k) = \Omega(1)$, under the assumptions in Theorem 2.*

Proof. $(\frac{q-1}{q})^{n-k} = \Omega(1)$ can be proven by mimicking how we show that $(1-\frac{1}{q})^k = \Omega(1)$ in the proof of Theorem 3, and $\gamma_2^*(q, k-1, n-k) \geq \gamma_2^*(q, k, n-k) = \Omega(1)$. □

$\gamma_3^*(q, k, n-k)$ is expected to be a loose bound, as $\mathbf{C}^{(1)} \neq \perp$ is just a sufficient condition for Algorithm 3 to succeed. Algorithm 3 succeeds as long as there exists j such that $\mathbf{C}^{(j)} \neq \perp$. Therefore, we estimate the success probability of Algorithm 3 as

$$\gamma_3(q, n, k) \coloneqq 1 - \left(1 - \left(\frac{q-1}{q}\right)^{n-k} \cdot \gamma_2(q, k-1, n-k)\right)^k. \tag{4}$$

5 Application to LESS Signatures

In this section, we introduce the scheme resulting from the application of canonical forms to LESS, which we call CF-LESS. We first describe the main idea at a high level, then give full details about the case we are interested in, namely, $E = M_n$ and $F = F_3$.

Let $F \leq M_n$ and $\mathsf{CosetRep}$ be a function that, on input an isometry ψ, returns a representative for the coset $F\psi$. By definition, for any two distinct isometries which are in the same coset, the function returns the same element. The function $\mathsf{CosetRep}$ has at most $[M_n : F] = |M_n|/|F|$ distinct images, hence its output has bitsize $\log_2 |M_n| - \log_2 |F|$. Later on, for the case $F = F_3$, we give full details about how such a function can be computed efficiently and how its output can be encoded efficiently.

A graphical representation of how the framework developed in the previous sections can be incorporated in a Sigma protocol derived from code equivalence is shown in Figure 6. For the sake of simplicity, we consider only the case in which the challenge is binary, i.e., the prover is asked to prove either a path from $\mathscr{C}$ to cmt, or from $\mathscr{C}'$ to cmt; when more linear codes are employed, modifications are analogous.

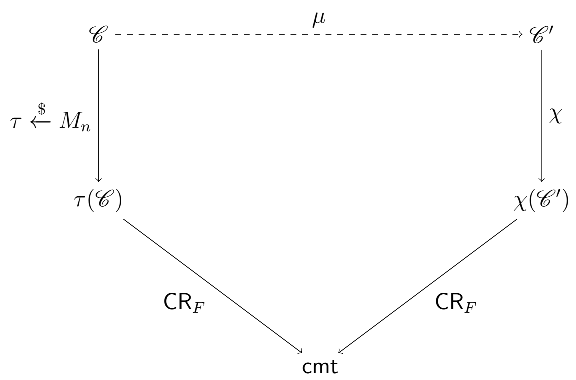

Fig. 6: Visualization of a ZK proof of knowledge based on canonical representatives. The public key is the pair $(\mathscr{C}, \mathscr{C}')$, while $\mu \in M_n$ is the secret key and $\chi = \mathsf{CosetRep}\left(\tau \circ \mu^{-1}\right)$.

As in traditional Sigma protocols derived from group actions, the prover samples a uniformly random isometry $\tau \in M_n$ and applies it to $\mathscr{C}$. The main difference is that in our case the prover commits to $\mathsf{CR}_F\left(\tau(\mathscr{C})\right)$. In practice, cmt is a commitment to its generator matrix in canonical form.

When the challenge is $\mathsf{ch} = 0$, the prover reveals τ; when $\mathsf{ch} = 1$, instead, the prover is asked to show how cmt can be computed, starting from $\mathscr{C}'$. In this case, thanks to our framework, the bitsize of the response can be greatly reduced. First, the isometry $\tau \circ \mu^{-1}$ maps $\mathscr{C}'$ to $\tau(\mathscr{C})$. Moreover, any isometry χ from the coset $F\left(\tau \circ \mu^{-1}\right)$ brings $\mathscr{C}'$ to the same F-equivalence class to which $\tau(\mathscr{C})$ belongs. Then, computation of the canonical representative will lead to the same commitment cmt. In particular, the prover can reply with $\chi = \mathsf{CosetRep}\big(F(\tau \circ \mu^{-1})\big)$, since this is enough to fully represent the coset.

We first recall how the traditional LESS Sigma protocol works then modifies it according to the above framework, for the case $F = F_3$.

5.1 The LESS Sigma Protocol

To begin we recall, in Figure 7, the Sigma protocol underlying the LESS signature scheme. As we have already said, there are variants with $s > 2$ generator matrices in the public key; for the sake of simplicity, Figure 7 is specific to $s = 2$.

As shown in [9], the protocol is *2-special sound*, with soundness error $\varepsilon = 1/2$. Note that, if the isometries μ and τ are both permutations, this protocol falls into a special case, in which security relies exclusively on PEP (as this is a special case of LEP); this may require some slight changes in how the protocol is actually deployed (for example, utilizing different parameters or particular choices of codes, such as self-orthogonal codes).

When $\mathsf{ch} = 0$, the verifier computes the very same matrix $\tau(\mathbf{G})$ used by the prover to generate the commitment. However, when $\mathsf{ch} = 1$, the verifier computes

Private Key: $\mu \in M_n$.
Public Key: Generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ for two linear codes $\mathscr{C}, \mathscr{C}' = \mu(\mathscr{C}) \subseteq \mathbb{F}_q^n$.

PROVER		VERIFIER
$\tau \xleftarrow{\$} M_n$		
$\widetilde{\mathbf{G}} \leftarrow \tau(\mathbf{G})$		
$\mathsf{cmt} \leftarrow \mathsf{Hash}\big(\mathsf{RREF}(\widetilde{\mathbf{G}})\big)$	$\xrightarrow{\mathsf{cmt}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \xleftarrow{\$} \{0,1\}$
If $\mathsf{ch} = 0$:		
$\mathsf{rsp} \leftarrow \tau$		
Else:		
$\mathsf{rsp} \leftarrow \tau \circ \mu^{-1}$	$\xrightarrow{\mathsf{rsp}}$	**If** $\mathsf{ch} = 0$:
		Verify $\mathsf{Hash}\Big(\mathsf{RREF}\big(\mathsf{rsp}(\mathbf{G})\big)\Big) = \mathsf{cmt}$
		Else:
		Verify $\mathsf{Hash}\Big(\mathsf{RREF}\big(\mathsf{rsp}(\mathbf{G}')\big)\Big) = \mathsf{cmt}$

Fig. 7: The original LESS Sigma protocol for linear equivalence. To generate the version for permutation equivalence, simply replace M_n by S_n.

$(\tau\circ\mu^{-1})(\mathbf{G}')$, which is equal to $\widetilde{\mathbf{G}}$ up to a change of a basis. This is why it is necessary to use RREF, which as we know is invariant under change of basis, to ensure verification works. Note that, in the case $\mathsf{ch} = 0$, the response consists of the randomly-generated isometry τ, and can thus be compressed by transmitting only a seed for a secure PRNG, as is common practice.

Remark 5. *The scheme presented in Figure 7 is simply the "core" element in the design of the LESS signature scheme. Indeed, to obtain a signature scheme, it is necessary to iterate the protocol, say t times, and apply the Fiat-Shamir transformation. Furthermore, a variety of optimizations are incorporated into the design, to improve the overall performance. For instance, the final signature scheme uses a variable number s of public keys (generating a tradeoff between public key and signature size); an "unbalanced" challenge string of fixed Hamming weight w (to maximize the reduction obtained by transmitting seeds for random objects) and a* seed tree *to compactly transmit seeds (as described in various previous works such as [7, 8]). It is worth clarifying that the Fiat-Shamir transformation directly yields EUF-CMA security [13], and the addition of such standard optimizations does not affect this claim, as shown for instance in [4, 12].*

Next, we show how the use of canonical forms can be embedded into the protocol. The high-level intuition is that using canonical forms, on top of the RREF computation, enriches the invariance properties we are able to achieve. In practice, we let the prover and the verifier end up in two codes which are F-equivalent. Leveraging this fact, the prover can provide multiple responses to verify the same commitment: among all such choices, we consider the one having the smallest communication cost.

5.2 The CF-LESS Sigma protocol

In this section we explain how the CF-LESS Sigma protocol operates. To do that, we first need to detail how CosetRep is defined; indeed, the availability of an efficient manner to implement this function is fundamental. We start by introducing a result which will be fundamental to understand how CosetRep works (the proof is omitted in the interest of space).

Theorem 7. *For every $\mu \in M_n$, there exists a unique pair $(\chi, \varphi) \in S_{k,n} \times F_3$ such that*

$$\mu = \varphi \circ \chi.$$

The above theorem has a very simple interpretation: every monomial map is essentially the combination of a permutation from $S_{k,n}$, which maps to the leftmost k coordinates, and a map from F_3 whose action is split into two sub-maps, one acting on the resulting first k coordinates, and the other on the rightmost $n-k$ coordinates. We use the above theorem to give a formal definition of the function CosetRep.

Definition 6 (Coset Representative)**.** *We define* CosetRep *as the function that, on input an isometry $\mu \in M_n$, returns $\chi \in S_{k,n}$, where χ is the unique isometry from $S_{k,n}$ such that, as in Theorem 7, we can write $\mu = \varphi \circ \chi$ for $\varphi \in F_3$.*

It is easy to see that CosetRep can be computed efficiently. We now show that the above definition is proper, i.e., that for any isometry in the same coset, the function outputs the same isometry.

Theorem 8. *For any $\mu \in M_n$, the function* CosetRep*, on input any isometry from the coset $F_3\mu$, returns the same value.*

Proof. We just need to show that, for every two isometries in the same coset, we obtain the same representative. To this end, let $\mu, \mu' \in M_n$ such that $\mu' = \varphi^* \circ \mu$ for some $\varphi^* \in F_3$. Since $\mu = \varphi \circ \chi$ (as in Theorem 7), we have also

$$\mu' = \varphi^* \circ \mu = \underbrace{\varphi^* \circ \varphi}_{\varphi'} \circ \chi = \varphi' \circ \chi. \tag{5}$$

Since F_3 is a subgroup, then $\varphi' \in F_3$. Then, thanks to Theorem 7, the decomposition in (5) is unique, hence $\mathsf{CosetRep}(\mu') = \mathsf{CosetRep}(\mu) = \chi$. □

Remark 6. *If one considers S_n instead of M_n and F_1 instead of F_3, the above theorems remain valid. Moreover, the function* CosetRep *is defined in analogous way.*

As we have already specified in Section 2, every element of $S_{k,n}$ can be encoded as a set $J \subseteq \{1, \cdots, n\}$ of size k. This is exactly how the output of CosetRep can be encoded: on input some monomial, it decomposes it according to Theorem 7 and then returns the set J which encodes $\chi \in S_{k,n}$.

We are now ready to show how the protocol depicted in Figure 6 can be turned into a practical ZK protocol. The formal description of the resulting Sigma protocol is given in Figure 8. The properties of the protocol (completeness, zero-knowledge and soundness) are analyzed in the next section.

Private Key: $\mu \in M_n$.

Public Key: Generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ for two linear codes $\mathscr{C}, \mathscr{C}' = \mu(\mathscr{C}) \subseteq \mathbb{F}_q^n$.

PROVER		VERIFIER
do:		
$\tau \xleftarrow{\$} M_n$		
$\big((\mathbf{I}_k \mid \mathbf{A}), \pi\big) \leftarrow \mathsf{RREF}^*\big(\tau(\mathbf{G})\big)$		
while $\mathsf{CF}^{(3)}(\mathbf{A}) = \perp$		
$\mathsf{cmt} \leftarrow \mathsf{Hash}\big(\mathsf{CF}^{(3)}(\mathbf{A})\big)$	$\xrightarrow{\mathsf{cmt}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \xleftarrow{\$} \{0,1\}$
If $\mathsf{ch} = 0$:		
$\mathsf{rsp} := \tau$		
Else:		
$\mathsf{rsp} := J = \mathsf{CosetRep}(\pi \circ \tau \circ \mu^{-1})$	$\xrightarrow{\mathsf{rsp}}$	**If** $\mathsf{ch} = 0$:
		$(\mathbf{I}_k \mid \mathbf{A}) \leftarrow \mathsf{SF}\big(\tau(\mathbf{G})\big)$
		Else:
		$\mathbf{A} \leftarrow \mathbf{G}'^{-1}_J \cdot \mathbf{G}'_{\{1,\cdots,n\}\setminus J}$
		Verify $\mathsf{Hash}\big(\mathsf{CF}^{(3)}(\mathbf{A})\big) = \mathsf{cmt}$

Fig. 8: The CF-LESS Sigma protocol for linear equivalence. To generate the version for permutation equivalence, simply replace M_n by S_n.

Remark 7. *For $k = Rn$, we have $\log_2\binom{n}{k} = n \cdot h(R) \cdot \big(1 + o(1)\big)$ and, in particular, $\log_2\binom{n}{k} \le n \cdot h(R)$, with the bound being asymptotically tight. If $R = \frac{1}{2}$, one has $h(R) = 1$, hence $\log_2\binom{n}{k}$ asymptotically approaches n (from below). In such a case, one can simplify the encoding of cosets and utilize just a binary string of length n, where the ones correspond to the coordinates that are moved to the leftmost k coordinates. This allows to consider a very simple and asymptotically optimal encoding for cosets as a binary string with length n and weight $n/2$, with the positions corresponding to the coordinates that are moved to the k leftmost positions.*

5.3 Properties of the CF-LESS Sigma Protocol

We are now ready to show that the CF-LESS Sigma protocol achieves the three fundamental properties for a ZK proof of knowledge, that is, completeness, zero-knowledge and special soundness. The first two properties are immediate; nonetheless, we prove them for the sake of absolute clarity. For what concerns special soundness, we show that it reduces to finding solutions to the following problem.

Problem 4 (**Canonical Forms Linear Equivalence Problem (CF-LEP)**)**.** *Let CR_{F_3} and CF_{F_3} be canonical representative function and canonical form functions, respectively. Given two linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$, find $\chi, \chi' \in S_{k,n}$ such that*

$$\mathsf{CR}_{F_3}\big(\chi(\mathscr{C})\big) = \mathsf{CR}_{F_3}\big(\chi'(\mathscr{C})\big).$$

Equivalently, given two generator matrices $\mathbf{G}, \mathbf{G}' \in \overline{\mathbb{F}}_q^{k\times n}$*, find two size-k sets* $J, J' \subseteq \{1, \cdots, n\}$ *such that*

$$\mathsf{CF}_{F_3}\left(\mathbf{G}_J^{-1} \cdot \mathbf{G}_{\{1,\cdots,n\}\setminus J}\right) = \mathsf{CF}_{F_3}\left(\mathbf{G}'^{-1}_{J'} \cdot \mathbf{G}'_{\{1,\cdots,n\}\setminus J'}\right)$$

For analogy with the traditional case, we refer to the above problem as *CF-LEP*; we will carefully analyze its hardness in Section 6.

Completeness and Zero-Knowledge

Zero-knowledge follows immediately from the fact that τ is uniformly distributed over M_n. We now proceed by showing that the protocol is complete.

The prover commits to the generator matrix (in canonical form) of the canonical representative of the F_3-equivalence class $\mathfrak{C}_{F_3}(\widetilde{\mathscr{C}})$, with $\widetilde{\mathscr{C}} = \pi \circ \tau(\mathscr{C})$. Notice that if $\{1, \cdots, k\}$ is an information set for $\tau(\mathscr{C})$, then $\pi = id_n$ otherwise $\pi \neq id_n$.

When $\mathsf{ch} = 0$, the prover responds with τ so the verifier repeats the very same operations performed by the prover. When instead $\mathsf{ch} = 1$, the prover replies with the representative of the coset $F_3(\pi \circ \tau \circ \mu^{-1})$. Observe that $\tau' = \pi \circ \tau \circ \mu^{-1}$ is such that $\tau'(\mathscr{C}') = \widetilde{\mathscr{C}}$. The prover responds with a representative χ for the coset $F_3\tau'$: the verifier computes $\chi(\mathscr{C}')$ which is F_3-equivalent to $\widetilde{\mathscr{C}}$ hence has the very same canonical representative.

In the protocol, all the above operations are detailed in terms of linear algebra. For the sake of completeness, we review completeness also using linear algebra. Let $\widetilde{\mathbf{G}} = \pi \circ \tau(\mathbf{G})$ and write $\mathbf{G}' = \mathbf{S} \cdot \mu(\mathbf{G})$ for some $\mathbf{S} \in \mathrm{GL}_k(q)$. Let $\chi \in S_{k,n}$ be the permutation represented by the set J; then,

$$(\mathbf{G}'_J \mid \mathbf{G}'_{\{1,\cdots,n\}\setminus J}) = \chi(\mathbf{G}') = \mathbf{S} \cdot \widetilde{\mathbf{G}} \cdot \begin{pmatrix} \mathbf{Q}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_c \end{pmatrix} = \mathbf{S} \cdot (\mathbf{I}_k \mid \mathbf{A}) \cdot \begin{pmatrix} \mathbf{Q}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_c \end{pmatrix},$$

with $\mathbf{Q}_r \in M_k$ and $\mathbf{Q}_c \in M_{n-k}$. Then

$$\mathbf{G}'^{-1}_J \cdot \mathbf{G}'_{\{1,\cdots,n\}\setminus J} = (\mathbf{S} \cdot \mathbf{Q}_r)^{-1} \cdot \mathbf{S} \cdot \mathbf{A} \cdot \mathbf{Q}_c = \mathbf{Q}_r^{-1} \cdot \mathbf{A} \cdot \mathbf{Q}_c.$$

Then, $\mathsf{CF}^{(3)}\left(\mathbf{G}'^{-1}_J \cdot \mathbf{G}'_{\{1,\cdots,n\}\setminus J}\right) = \mathsf{CF}^{(3)}(\mathbf{A})$.

Special Soundness

We show that the protocol is 2-special sound, i.e., that there exists a polynomial time algorithm that, on input two accepting transcripts with same commitment but different challenge, computes the secret. Given that we are considering a Sigma protocol with binary challenge (i.e., the challenge space has size 2), it follows that it has soundness error $\varepsilon = 1/2$.

Proposition 9. *The protocol in Figure 8 is 2-special sound.*

Proof. We consider two accepting transcripts $\tau \in M_n$ and $J \subseteq \{1, \cdots, n\}$, respectively for $\mathsf{ch} = 0$ and $\mathsf{ch} = 1$, and commitment cmt.

We first focus on the transcript for $\mathsf{ch} = 0$. From the knowledge of τ, one can obtain $\pi \in S_{k,n}$ such that $\mathsf{SF}\big(\tau(\mathbf{G})\big) = \mathsf{RREF}\Big(\pi\big(\tau(\mathbf{G})\big)\Big)$. Indeed, π is just the permutation that moves the information set used for RREF computation to the first k coordinates. Let $J^* = \mathsf{CosetRep}(\pi \circ \tau)$: then, $\mathbf{A}^* = \mathbf{G}_{J^*}^{-1} \cdot \mathbf{G}_{\{1,\cdots,n\}\setminus J^*}$ is such that $\mathsf{Hash}\big(\mathsf{CF}^{(3)}(\mathbf{A}^*)\big) = \mathsf{cmt}$.

We now focus on the transcript for $\mathsf{ch} = 1$. Since it is accepting, then $\mathbf{A} = \mathbf{G}_J^{-1} \cdot \mathbf{G}_{\{1,\cdots,n\}\setminus J}$ is such that $\mathsf{Hash}\big(\mathsf{CF}^{(3)}(\mathbf{A})\big) = \mathsf{cmt}$. Either $\mathsf{CF}^{(3)}(\mathbf{A}) \neq \mathsf{CF}^{(3)}(\mathbf{A}')$ and we found a hash collision, or $\mathsf{CF}^{(3)}(\mathbf{A}) = \mathsf{CF}^{(3)}(\mathbf{A}')$ and we found a solution to Problem 4. In particular, the solution corresponds to the two permutations from $S_{k,n}$ which are represented by J^* and J. □

Remark 8. *The protocol is zero-knowledge , complete and 2-special sound also for the permutation case since, again, this is just a special case of our analysis. The corresponding hard problem is defined as Problem 4, with the only difference that one should employ functions computing canonical representatives and forms for case F_1.*

5.4 Computational Complexity

We briefly comment on the computational cost of the protocol in Figure 8. To this end, we rely on a heuristic which is commonly employed when studying code-based problems (e.g., in papers about information-set decoding). Most importantly, numerical simulations confirm the heuristic.

Heuristic 1. *Let $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ be the generator matrix for a code with dimension k and length n. For any set $J \subseteq \{1, \cdots, n\}$ of size k, we consider that $\mathbf{G}_J$ is a $k \times k$ matrix sampled according to the uniform distribution over $\mathbb{F}_q$. Analogously, also $\mathbf{G}_{\{1,\cdots,n\}\setminus J}$ is a $k \times (n-k)$ matrix sampled according to the uniform distribution over $\mathbb{F}_q$.*

Under this heuristic, we have that the average number of isometries τ the prover has to test, before a valid matrix is found, corresponds to $1/\gamma$. Note that the heuristic is here employed since we consider that, for each choice of τ, $\tau(\mathbf{G})$ behaves as a uniformly random matrix. For each τ the prover executes RREF^* and then computes $\mathsf{CF}^{(3)}$.Let T_{RREF^*} and T_{CF} be the costs of these functions, respectively; then, computing the commitment comes with cost $\frac{T_{\mathsf{RREF}^*}+T_{\mathsf{CF}}}{\gamma}$. As we have already seen, γ is in practice very high, so that $1/\gamma \approx 1$: the first choice of $\widetilde{\mathbf{Q}}$ is successful with overwhelming probability.

Computing the response takes a much smaller time so, for simplicity, we do not consider it. Analogously, verification is predominated by performing Gaussian elimination and then computing the canonical form. So, on the verifier's side, the cost can be estimated as $T_{\mathsf{RREF}^*} + T_{\mathsf{CF}}$.

Whenever T_{CF} has a cost which is less than or, at the very least, comparable with T_{RREF^*}, the use of canonical forms does not lead to significant computational overhead. Indeed, as it is well known, a crude but realistic estimate for T_{RREF^*} is $O\big(n^3\big)$ field operations. As we have already seen in Section 4, it is possible to define canonical forms whose time complexity is much better than or comparable with that of T_{RREF^*}. Indeed, among the functions we have defined, the most time consuming one is that for Case 5, taking $\tilde{O}(n^3)$ field operations.

6 Hardness Analysis and Implications

In this section we provide strong evidences that the new formulation of code equivalence, using canonical forms, still leads to a hard problem. We give all our reductions considering the most general case of LEP and $F = F_3$, but the reductions trivially extend to other choices for F. We provide reductions between LEP and CF-LEP which hold given that:

- canonical forms can be computed in polynomial time;
- from the computation of $\mathsf{CF}_{F_3}(\mathbf{A})$, one also obtains (in polynomial time) transformations $\mathbf{M}_r, \mathbf{M}_c \in F_r \times F_c$ such that $\mathsf{CF}_{F_3}(\mathbf{A}) = \mathbf{M}_r \cdot \mathbf{A} \cdot \mathbf{M}_c$;
- for the pair of considered codes, canonical representatives are well defined.

The first two conditions are trivially verified. For instance, the function $\mathsf{CF}^{(3)}$ runs in polynomial time and implicitly builds the transformations $\mathbf{M}_r$ and $\mathbf{M}_c$. In the following, to make this requirement clear, we indicate by $\mathsf{CF}^*_{F_3}$ a canonical form function that returns both the canonical form and the pair $(\mathbf{M}_r, \mathbf{M}_c)$.

For what concerns the last requirement, we prove that, under Heuristic 1, all but a negligible portion of random codes do not admit a canonical representative if γ is large enough (say, it does not decrease exponentially with n).

As we show later, for parameters that are relevant for cryptographic applications, the fraction of codes for which the reduction fails is negligible, hence, our reductions are applicable and CF-LEP is indeed as hard as LEP.

We furthermore show that the reductions may be used to mount a practical attack on code equivalence. Given access to canonical forms that are efficiently computable (as those in Section 4) and have sufficiently low failure probability, we can exploit the birthday paradox and devise a simple attack running in time $\tilde{O}\left(\sqrt{\binom{n}{k}}\right)$. In some regimes (e.g., when q is large enough), this attack appears to be faster than all previously known solvers. Moreover, it does not depend on some code properties such as the hull dimension, differently from [19] and [2], or the minimum distance [6, 16]. Regardless of the type of equivalence considered (permutation vs linear), the procedure remains exactly the same, with the only difference being in the employed canonical form function. Finally, we note that the only dependence on the finite field size is in the cost and success probability of the employed canonical form function, which are expected to be very mild. This is another remarkable difference with other solvers, for instance, those based on finding low weight codewords [6, 16]: when q increases, the minimum distance of random codes increases as well, so these attacks become slower.

6.1 Reductions between LEP and CF-LEP

In this section we show that, whenever a solution to CF-LEP exist (i.e., whenever computation of canonical representative functions do not fail), LEP and CF-LEP are equivalent. Later on we deal with failures and show that, for random codes, canonical representatives exist with extremely large probability. Putting everything together, the reductions we give in this section show that, for random codes, LEP and CF-LEP are basically the same problem (as solving one problem allows to solve the other).

We first show that CF-LEP reduces to LEP. Let $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ be the two codes defining the LEP instance and let $\mu \in M_n$ be a solution. Then, obtaining a solution for CF-LEP is trivial. Let $\pi \in S_{k,n}$ be the permutation provided as output by RREF^* on input any generator matrix for $\mathscr{C}$. Then, a solution for CF-LEP can be obtained as $\chi \coloneqq \pi$ and $\chi' = \mathsf{CosetRep}(\pi \circ \mu^{-1})$. Indeed,

$$\chi'(\mathscr{C}') \sim_{F_3} \pi \circ \mu^{-1} \circ \mu(\mathscr{C}) = \pi(\mathscr{C}) = \chi(\mathscr{C}).$$

If, after computation of systematic forms, canonical forms can be computed, then the reduction is done. Otherwise, one can row reduce with respect to a different information set (and update accordingly π), until a canonical form can be computed.

The other direction, i.e., showing that LEP reduces to CF-LEP is more interesting. The way to map a CF-LEP solution into a LEP solution is described in Algorithm 4; its correctness is detailed in the next Proposition.

Proposition 10. *If $(\mathscr{C}, \mathscr{C}')$ admits a solution for CF-LEP, then a solution for LEP on input $\mathscr{C}, \mathscr{C}'$,can be found in polynomial time.*

Proof. Algorithm 4 obviously takes polynomial time, since all it does is computing canonical forms (which takes polynomial time by hypothesis), performing matrix multiplications/inversions and computing RREFs. Hence, we only need to show that the algorithm is correct, i.e., that the output $\mu \in M_n$ is indeed an isometry between $\mathscr{C}$ and $\mathscr{C}'$.

Let $\widetilde{\mathbf{G}} = \mathbf{G} \cdot \mathbf{P}$ and $\widetilde{\mathbf{G}}' = \mathbf{G}' \cdot \mathbf{P}'$, and

$$\big(\mathbf{I}_k \mid \mathbf{A}\big) = \mathsf{RREF}(\widetilde{\mathbf{G}}), \quad \big(\mathbf{I}_k \mid \mathbf{A}'\big) = \mathsf{RREF}(\widetilde{\mathbf{G}}').$$

Notice that this means there exist $\mathbf{S}, \mathbf{S}' \in \mathrm{GL}_k(q)$ such that

$$\big(\mathbf{I}_k \mid \mathbf{A}\big) = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}, \qquad \big(\mathbf{I}_k \mid \mathbf{A}'\big) = \mathbf{S}' \cdot \mathbf{G}' \cdot \mathbf{P}'$$

Let $(\mathbf{Q}_r, \mathbf{Q}_c) \in M_k \times M_{n-k}$ such that

$$\mathsf{CF}_{F_3}(\mathbf{A}) = \mathbf{Q}_r \cdot \mathbf{A} \cdot \mathbf{Q}_c,$$

and $(\mathbf{Q}_r', \mathbf{Q}_c') \in M_k \times M_{n-k}$ such that

$$\mathsf{CF}_{F_3}(\mathbf{A}') = \mathbf{Q}_r' \cdot \mathbf{A}' \cdot \mathbf{Q}_c'.$$

Algorithm 4: Building LEP solution from CF-LEP solution

Data: $\mathsf{CF}^*_{F_3} : \mathbb{F}_q^{k\times(n-k)} \mapsto \big\{\{\perp\} \cup \mathbb{F}_q^{k\times(n-k)}\big\}$: a canonical form function for F_3
Input: matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$, solution $\chi, \chi' \in S_{k,n}$ for CF-LEP
Output: solution $\mu \in M_n$ for LEP

`/*` Apply χ and χ' so that both matrices have $\{1, \cdots, k\}$ as information set, compute systematic forms `*/`

Set $\mathbf{P}, \mathbf{P}' \in S_{k,n}$ as the permutation matrices associated to χ and χ', respectively;

Compute $\widetilde{\mathbf{G}} = \mathbf{G} \cdot \mathbf{P}$ and $\widetilde{\mathbf{G}}' = \mathbf{G}' \cdot \mathbf{P}'$;

Compute $\big(\mathbf{I}_k \mid \mathbf{A}\big) = \mathsf{SF}(\widetilde{\mathbf{G}})$ and $\big(\mathbf{I}_k \mid \mathbf{A}'\big) = \mathsf{SF}(\widetilde{\mathbf{G}}')$;

`/*` Compute the transformations bringing $\mathbf{A}$ and $\mathbf{A}'$ to the canonical forms `*/`

Compute $\mathbf{B}, (\mathbf{Q}_r, \mathbf{Q}_c) = \mathsf{CF}^*_{F_3}(\mathbf{A})$;

Compute $\mathbf{B}', (\mathbf{Q}'_r, \mathbf{Q}'_c) = \mathsf{CF}^*_{F_3}(\mathbf{A}')$;

`/*` Build solution for LEP `*/`

Set $\widetilde{\mathbf{Q}}_r = \mathbf{Q}'^{-1}_r \cdot \mathbf{Q}_r$;

Set $\widetilde{\mathbf{Q}}_c = \mathbf{Q}'_c \cdot \mathbf{Q}_c^{-1}$;

Set $\mu \in M_n$ as the monomial associated with the matrix

$$\mathbf{P}' \cdot \begin{pmatrix} \widetilde{\mathbf{Q}}_r & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Q}}_c \end{pmatrix} \cdot \mathbf{P}^{-1}$$

return μ

Since $\mathsf{CF}_{F_3}(\mathbf{A}) = \mathsf{CF}_{F_3}(\mathbf{A}')$, it holds that

$$\mathbf{Q}_r \cdot \mathbf{A} \cdot \mathbf{Q}_c = \mathbf{Q}'_r \cdot \mathbf{A}' \cdot \mathbf{Q}'_c \implies \mathbf{A} = \underbrace{\mathbf{Q}_r^{-1} \cdot \mathbf{Q}'_r}_{\widetilde{\mathbf{Q}}_r^{-1}} \cdot \mathbf{A}' \cdot \underbrace{\mathbf{Q}'_c \cdot \mathbf{Q}_c^{-1}}_{\widetilde{\mathbf{Q}}_c}.$$

Then

$$\begin{aligned}
\mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P} &= \big(\mathbf{I}_k \mid \mathbf{A}\big) \\
&= \big(\mathbf{I}_k \mid \widetilde{\mathbf{Q}}_r^{-1} \cdot \mathbf{A}' \cdot \widetilde{\mathbf{Q}}_c\big) \\
&= \widetilde{\mathbf{Q}}_r^{-1} \cdot \big(\mathbf{I}_k \mid \mathbf{A}'\big) \cdot \begin{pmatrix} \widetilde{\mathbf{Q}}_r & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Q}}_c \end{pmatrix} \\
&= \widetilde{\mathbf{Q}}_r^{-1} \cdot \mathbf{S}' \cdot \mathbf{G}' \cdot \mathbf{P}' \cdot \begin{pmatrix} \widetilde{\mathbf{Q}}_r & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Q}}_c \end{pmatrix}.
\end{aligned}$$

Hence

$$\mathbf{G} = \underbrace{\mathbf{S}^{-1} \cdot \widetilde{\mathbf{Q}}_r^{-1} \cdot \mathbf{S}'}_{\in \mathrm{GL}_k(q)} \cdot \mathbf{G}' \cdot \underbrace{\mathbf{P}' \cdot \begin{pmatrix} \widetilde{\mathbf{Q}}_r & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{Q}}_c \end{pmatrix} \cdot \mathbf{P}^{-1}}_{\in M_n}.$$

□

6.2 Existence of solutions to CF-LEP for random codes

We proceed by analyzing the probability that a random code admits a canonical representative, assuming we have a canonical form function that exists with probability $\gamma < 1$. We are able to show that, unless γ is negligible, canonical representatives exist with overwhelming probability.

For a set $J \subseteq \{1, \cdots, k\}$, we denote by ζ the probability that J is an information set, that is, the probability that the columns indexed by J form a non singular matrix. Under Heuristic 1, we study this matrix as it is uniformly random over $\mathbb{F}_q$. Then, the probability ζ is the same for all sets J and, moreover (see e.g [14, Section 2]), it holds that $\zeta \geq 1 - 1/q - 1/q^2$. According to the heuristic, after RREF, the non systematic part behaves as a uniformly random $k \times (n-k)$ matrix over $\mathbb{F}_q$, so it admits a canonical form with probability γ. Consequently, the probability that canonical forms cannot be defined for all sets J is $(1 - \zeta\gamma)^{\binom{n}{k}}$. Taking the logarithm of this quantity and considering that $\log_2(x) \leq \frac{1}{\ln(2)}(x-1)$ for all positive $x \in \mathbb{R}$, we further get

$$\log_2 (1 - \zeta\gamma)^{\binom{n}{k}} \leq \binom{n}{k} \frac{(1-\zeta\gamma) - 1}{\ln(2)} = -\binom{n}{k} \frac{\zeta\gamma}{\ln(2)} \leq -\gamma \binom{n}{k} \frac{(1 - 1/q - 1/q^2)}{\ln(2)}.$$

Thus, according to our analysis, the probability that for a random code a canonical representative cannot be defined is less than $2^{-\gamma\binom{n}{k}\frac{(1-1/q-1/q^2)}{\ln(2)}}$: this is always negligible, unless γ is negligible, as well.

Remark 9. *For CF-LESS, we consider instances having rate* $1/2$ *and canonical forms that succeeds with probability at least* $1 - 2^{-83}$*. In such a case, the probability that a canonical representative does not exist is well approximated by* $2^{-\frac{1}{\ln(2)}2^n}$ *: the probability that the reduction does not apply is negligible.*

Remark 10. *For (some of) the canonical forms we defined in Section 4, the success probability gets smaller when* q *gets lower. In such a regime, there may exist different ways to define canonical forms with sufficiently large success probability. We view the task of finding canonical forms that work better, even when* q *is smaller, as an interesting open question.*

6.3 Canonical Forms as a Solver for LEP

We now show how the reduction in Algorithm 4 can be used to mount a practical attack on code equivalence. Again, we focus on the case of LEP but the attack obviously works also when considering PEP. The core of our proposed procedure is shown in Algorithm

5. Essentially, the procedure first solves CF-LEP using a meet-in-the-middle strategy; then, it calls the reduction in Algorithm 5 to reconstruct the equivalence between $\mathbf{G}$ and $\mathbf{G}'$.

Algorithm 5: Solving code equivalence via canonical forms

Input: matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$, lists size m
Output: equivalence between $\mathbf{G}$ and $\mathbf{G}'$, or failure

Set $\mathcal{L} = \varnothing$, $\mathcal{L}' = \varnothing$;

// Populate first list
while $|\mathcal{L}| < m$ **do**
  Sample $\chi \xleftarrow{\$} S_{k,n}$;
  Compute $\widetilde{\mathbf{G}} = \chi(\mathbf{G})$;
  if $\widetilde{\mathbf{G}}_{\{1,\cdots,k\}}$ has rank k **then**
    Compute $\mathbf{A} = \widetilde{\mathbf{G}}^{-1}_{\{1,\cdots,k\}} \cdot \widetilde{\mathbf{G}}_{\{k+1,\cdots,n\}}$;
    Compute $\mathbf{B} = \mathsf{CF}_{F_3}(\mathbf{A})$;
    if $\mathbf{B} \neq \perp$ **then**
      Add $\big(\chi, \mathbf{B}\big)$ to $\mathcal{L}$;

// Populate second list
while $|\mathcal{L}'| < m$ **do**
  Sample $\chi' \xleftarrow{\$} S_{k,n}$;
  Compute $\widetilde{\mathbf{G}}' = \chi'(\mathbf{G}')$;
  if $\widetilde{\mathbf{G}}'_{\{1,\cdots,k\}}$ has rank k **then**
    Compute $\mathbf{A}' = \widetilde{\mathbf{G}}'^{-1}_{\{1,\cdots,k\}} \cdot \widetilde{\mathbf{G}}'_{\{k+1,\cdots,n\}}$;
    Compute $\mathbf{B}' = \mathsf{CF}_{F_3}(\mathbf{A}')$;
    if $\mathbf{B}' \neq \perp$ **then**
      Add $\big(\chi', \mathbf{B}'\big)$ to $\mathcal{L}'$;

// Find solution for CF-LEP, then reconstruct the equivalence
Search for collisions, i.e., pairs $(\chi, \mathbf{B}) \in \mathcal{L}$, $(\chi', \mathbf{B}') \in \mathcal{L}'$ such that $\mathbf{B} = \mathbf{B}'$;
If a collision is found, call Algorithm 4 on input $\mathbf{G}, \mathbf{G}'$ and χ, χ'

The analysis of the resulting time complexity is very simple. First, because of the birthday paradox, the algorithm has success probability which is approximately $1/2$. For each candidate χ (resp., χ'), the probability that it corresponds to a computable canonical form is $\zeta\gamma$. Thus, the number of distinct χ (and χ') leading to a canonical form can be estimated as $\gamma\zeta\binom{n}{k}$. Exploiting the birthday paradox, we can set the lists size as $m = \sqrt{\gamma\zeta\binom{n}{k}}$. Since each candidate for χ (and χ') leads to a failure in the canonical form computation with probability $\gamma\zeta$, the average number of candidates we have to test, for each list, is given by $\frac{1}{\sqrt{\gamma\zeta}} \cdot \sqrt{\binom{m}{k}}$. Indeed, on average, this yields lists of size

$$\gamma\zeta \cdot \frac{1}{\sqrt{\gamma\zeta}} \cdot \sqrt{\binom{n}{k}} = \sqrt{\gamma\zeta} \cdot \sqrt{\binom{n}{k}}.$$

Thus, we get an overall cost of

$$O\left(\frac{1}{\sqrt{\gamma\zeta}} \cdot T_{\mathsf{CF}} \cdot \sqrt{\binom{n}{k}}\right).$$

As we have already seen, ζ is lower bounded by a constant which increases with q. Hence, using canonical forms that can be computed in polynomial time, asymptotically, we get an overall cost of

$$\tilde{O}\left(\frac{1}{\sqrt{\gamma}} \cdot \sqrt{\binom{n}{k}}\right) = 2^{\frac{1}{2}n\cdot h(R)\cdot\big(1+o(1)\big)+\frac{1}{2}\log_2(1/\gamma)}.$$

In all the cases in which γ is non-negligible in n, the factor $\log_2(1/\gamma)$ gets absorbed by the $o(1)$ term and we get an attack with complexity $2^{n\cdot h(R)\cdot\big(1+o(1)\big)}$.

Appendix A briefly compares the cost of Algorithm 5 with other known attacks. For solvers based on short codewords [5,6,16], we use a rough but simplified analysis: essentially, we consider that finding a single codeword with minimum weight is enough. These preliminary results show that, when q is large enough, Algorithm 4 becomes faster than all the other state-of-the-art attacks.

The existence of some non-null failure rate for canonical forms impacts the complexity of the attack very mildly. Indeed, the factor $\log_2(1/\gamma)$ becomes relevant only when γ is negligible in n (i.e., when $1/\gamma$ is exponential in n). This is very unlikely.

For the regime in which q is large enough, we have already shown that canonical forms that can be computed in polynomial time and have a sufficiently large success probability exist. The canonical forms we have defined may not be the best choice for small q but we believe it is very plausible that, in this regime, other good canonical forms exist. We leave this as an interesting open question.

In particular, our analysis holds for the CF-LESS instances that we propose in the next section since we show that, for all of them, the success probability is always at least $1-2^{-83}$.

7 Concrete Instantiations

In this section, we discuss the practical impact of our technique on concrete instances.

7.1 Optimal Signature Sizes

In all the cases in which the collision attack we presented in the previous section is the fastest solver for code equivalence, our framework allows to achieve optimal signature sizes when $F \simeq S_k \times S_{n-k}$, i.e. Case 1, if permutation equivalence is considered, or $F \simeq M_k \times M_{n-k}$, i.e. Case 3, if linear equivalence is considered. To guarantee that the collision attack has the desired complexity, we must choose the code length so that $\frac{1}{2} \cdot n \cdot h(R) = \lambda$, that is, $n \cdot h(R) = 2\lambda$. Since $S_{k,n}$ has size $\binom{n}{k}$, its elements are represented with binary size

$$\log_2 \binom{n}{k} = \log_2 \binom{n}{Rn} = n \cdot h(R) \cdot \big(1 + o(1)\big) = 2\lambda \cdot \big(1 + o(1)\big).$$

In particular, this applies to the instances that we recommend for CF-LESS. Indeed, the code parameters are inherited from those of the LESS NIST submission [1], for which $R = \frac{1}{2}$ and the resulting n is approximately equal to 2λ. As we shall see next, our collision attack has an asymptotic running time $\approx 2^{2\lambda} \approx 2^{\frac{1}{2}n}$. This even means that, as we already said, the encoding of permutations from $S_{k,n}$ as binary vector of length n and weight $n/2$, despite being very simple and efficient, yields an optimal strategy for representing elements of $S_{k,n}$.

7.2 CF-LESS Instances

Table 1 shows the results obtained when applying canonical forms to the LESS parameters, for the case $s = 2$; the original is included in the top row of each cell, for ease of comparison. The parameter t stands for the total number of rounds. The parameter w stands for the number of rounds where the challenge is nonzero. Whenever the challenge is zero, the response is just a short seed, so keeping w small compared to t helps to save signature size.

The main purpose of this table is to illustrate the impact of our technique; therefore, we report sizes corresponding to the various choices of F defined in our work, indicating which one was considered in the column "Case". The column "Attack Factor" indicates $\log_2$ of the largest factor $\sqrt{\binom{n}{k}}$ in the cost of the attack in Section 6.3. Note that the number of bit operations taken by the attack is more than $\sqrt{\binom{n}{k}}$, as there are other nontrivial factors such as T_{CF}. Giving the exact bit operation counts is out of the scope of this paper.

NIST Cat.	Type	Code Params n	k	q	Prot. Params s	t	w	Attack Factor	Case	pk (B)	sig (B)	Failure Rate
1	Mono	252	126	127	2	247	30	123.84	[18]	13939	8624	0
	Mono								Case 3		2481	$\approx 10^{-24}$
	Perm								Case 1		2481	$\approx 10^{-49}$
3	Mono	400	200	127	2	759	33	197.67	[18]	35074	17208	0
	Mono								Case 3		5658	$\approx 10^{-19}$
	Perm								Case 1		5658	$\approx 10^{-63}$
5	Mono	548	274	127	2	1352	40	271.56	[18]	65792	30586	0
	Mono								Case 3		10036	$\approx 10^{-14}$
	Perm								Case 1		10036	$\approx 10^{-69}$

Table 1: Parameter sets for CF-LESS with $s = 2$. All sizes in bytes (B).

"Failure Rate" indicates the probability that the corresponding canonical form function returns $\perp$, and numbers for Case 1 and Case 3 are derived using Equation 3 and Equation 4, respectively. Note that the numbers for Case 1 are actually proven upper bounds on the failure rates.

The signature sizes in the column "sig" are computed as

$$w \cdot \lceil n/8 \rceil + \mathcal{N}(t, w) \cdot \ell_{\text{tree\_seed}} + \ell_{\text{salt}} + \ell_{\text{digest}}.$$

The value $\mathcal{N}(t, w)$ indicates the number of seeds (in the tree) that need to be released, and is estimated by $2^{\lceil \log_2 w \rceil} + w \cdot (\lceil \log_2 t \rceil - \lceil \log_2 w \rceil - 1)$, as in [10, 15]. The symbols $\ell_{\text{tree\_seed}}$, ℓ_{salt} and ℓ_{digest} stand for the respective lengths of seeds, salt and digest. These values have been specified in [1, Table 2].

Of course, one can achieve even smaller signature sizes by increasing s, at the cost of larger public keys. We report these in Table 2.

NIST Cat.	Type	Code Params n	k	q	Prot. Params s	t	w	Attack Factor	Case	pk (B)	sig (B)	Failure Rate
1	Mono	252	126	127	4	244	20	123.84	[18]	41785	5941	0
	Mono								Case 3		1846	$\approx 10^{-24}$
	Perm								Case 1		1846	$\approx 10^{-49}$
3	Mono	400	200	127	4	895	24	197.67	[18]	105174	12768	0
	Mono								Case 3		4368	$\approx 10^{-19}$
	Perm								Case 1		4368	$\approx 10^{-63}$
5	Mono	548	274	127	4	907	34	271.56	[18]	197312	25237	0
	Mono								Case 3		7769	$\approx 10^{-14}$
	Perm								Case 1		7769	$\approx 10^{-69}$

Table 2: Parameter sets for CF-LESS with $s = 4$. All sizes in bytes (B).

7.3 Advanced Signatures.

It is important to point out that our technique can, in principle, be applied to any other scheme based on code equivalence. To this end, we report in Table 3 the results concerning the ring signature scheme presented in [3]; as above, the original value is included in the top row of each cell. This scheme was built on top of the original LESS-FM protocol, and therefore did not feature any optimization for representing matrices, which explains the "-" in this row. Also, the authors in [3] only propose parameters for the permutation case (to minimize signature size), and for the lowest security level, roughly equivalent to NIST Category 1. The column r indicates the size of the ring of users.

It is worth noting that the scheme of [3] already compares extremely well with the rest of the literature for post-quantum ring signatures, due to the logarithmic signature size and reasonable computation cost. Thanks to the use of canonical forms, the scheme is able to beat even isogeny-based protocols such as Calamari [7], which is quite a remarkable feat. For example, for $r = 2^3$, the Calamari scheme yields a signature size of 5.4 KB, which is more than the 4522 bytes reported above.

NIST Cat.	Type	Code Params n	k	q	Attack Factor	pk (B)	Prot. Params t	w	r	Case	sig (B)	Failure Rate
1	Perm	230	115	127	112.87	11571	233	31	2^3	-	10761	0
										Case 1	4522	$\approx 10^{-49}$
									2^6	-	13737	0
										Case 1	7498	$\approx 10^{-49}$
									2^{12}	-	19689	0
										Case 1	13450	$\approx 10^{-49}$
									2^{21}	-	28617	0
										Case 1	22378	$\approx 10^{-49}$

Table 3: Parameter sets for ring signatures using canonical forms, and resulting sizes in bytes (B).

Acknowledgements

The authors wish to thank the anonymous reviewers for their insightful comments and suggestions which have greatly helped us in improving the paper.

The work of Paolo Santini was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU. The work of Edoardo Persichetti was partially supported via NSA grant H98230-22-1-0328. The work of Tung Chou was partially supported by Academia Sinica Grand Challenge Program Project AS-GCP-114-M01.

References

[1] Baldi, M., Barenghi, A., Beckwith, L., Biasse, J.F., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.J.O., Santini, P., Wallace, R.: LESS: Linear Equivalence Signature Scheme (2023), https://www.less-project.com/LESS-2023-08-18.pdf

[2] Bardet, M., Otmani, A., Saeed-Taha, M.: Permutation code equivalence is not harder than graph isomorphism when hulls are trivial. In: 2019 IEEE International Symposium on Information Theory (ISIT). pp. 2464–2468. IEEE (2019). https://doi.org/10.1109/ISIT.2019.8849855

[3] Barenghi, A., Biasse, J.F., Ngo, T., Persichetti, E., Santini, P.: Advanced signature functionalities from the code equivalence problem. International Journal of Computer Mathematics: Computer Systems Theory **7**(2), 112–128 (2022)

[4] Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: LESS-FM: Fine-Tuning Signatures from the Code Equivalence Problem. In: Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12. pp. 23–43. Springer (2021). https://doi.org/10.1007/978-3-030-81293-5

[5] Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: On the computational hardness of the code equivalence problem in cryptography. Advances in Mathematics of Communications **17**(1), 23–55 (2023). https://doi.org/10.3934/amc.2022064

[6] Beullens, W.: Not enough less: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In: Selected Areas in Cryptography: 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised

Selected Papers. pp. 387–403. Springer (2021). https://doi.org/10.1007/978-3-030-81652-0
[7] Beullens, W., Katsumata, S., Pintore, F.: Calamari and Falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 464–492. Springer (2020). https://doi.org/10.1007/978-3-030-64834-3
[8] Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-Fish: efficient isogeny based signatures through class group computations. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 227–247. Springer (2019). https://doi.org/10.1007/978-3-030-34578-5
[9] Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: LESS is More: Code-based Signatures Without Syndromes. In: Progress in Cryptology - AFRICACRYPT 2020: 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20–22, 2020, Proceedings 12. pp. 45–65. Springer (2020). https://doi.org/10.1007/978-3-030-51938-4
[10] Boyar, J., Erfurth, S., Larsen, K.S., Niederhagen, R.: Quotable signatures for authenticating shared quotes. In: Progress in Cryptology – LATINCRYPT 2023. pp. 273–292. Springer (2023). https://doi.org/10.1007/978-3-031-44469-2, https://arxiv.org/pdf/2212.10963.pdf
[11] D'Alconzo, G., Meneghetti, A., Signorini, E.: Group Factorisation for Smaller Signatures from Cryptographic Group Actions. Cryptology ePrint Archive, Paper 2024/1510 (2024), https://eprint.iacr.org/2024/1510
[12] De Feo, L., Galbraith, S.: SeaSign: Compact Isogeny Signatures from Class Group Actions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019. Lecture Notes in Computer Science, vol. 11478, pp. 759–789. Springer (2019). https://doi.org/10.1007/978-3-030-17659-4
[13] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Advances in Cryptology — CRYPTO' 86. pp. 186–194. Springer (1986). https://doi.org/10.1007/3-540-47721-7
[14] FULMAN, J., GOLDSTEIN, L.: Stein's method and the rank distribution of random matrices over finite fields. The Annals of Probability **43**(3), 1274–1314 (2015). https://doi.org/10.1214/13-AOP889
[15] Gueron, S., Persichetti, E., Santini, P.: Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. Cryptography **6**(1), 5 (2022). https://doi.org/10.3390/cryptography6010005
[16] Leon, J.: Computing automorphism groups of error-correcting codes. IEEE Transactions on Information Theory **28**(3), 496–511 (1982). https://doi.org/10.1109/TIT.1982.1056498
[17] NIST: Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process (2023), https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals
[18] Persichetti, E., Santini, P.: A New Formulation of the Linear Equivalence Problem and Shorter LESS Signatures. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 351–378. Springer (2023)
[19] Sendrier, N.: Finding the permutation between equivalent linear codes: The

support splitting algorithm. IEEE Transactions on Information Theory **46**(4), 1193–1203 (2000). https://doi.org/10.1109/18.850662
[20] Sendrier, N., Simos, D.E.: The Hardness of Code Equivalence over $\mathbb{F}_q$ and Its Application to Code-Based Cryptography. In: Post-Quantum Cryptography: 5th International Workshop, PQCrypto 2013, Limoges, France, June 4-7, 2013. Proceedings 5. pp. 203–216. Springer (2013). https://doi.org/10.1007/978-3-642-38616-9

A Comparison with Other Solvers

In this section, we consider various solvers for the code equivalence problem, and compare their running time with the one of our algorithm from Section 6.3.

SSA, [19]: this algorithm can efficiently solve PEP when the hull of the considered codes is small. However, the attack takes exponential time when the hull is large, as is the case for self-orthogonal codes (i.e. contained in their dual); in such a case, it has time complexity $T_{SSA} = O(q^k) = O(2^{Rn\cdot\log_2(q)})$. Thanks to a reduction in [20], SSA can also be used to solve LEP; however, whenever $q \geq 5$, the reduction maps any code into a self-orthogonal code with dimension k (so, it has time complexity $O(q^k)$).

BOS, [2]: this algorithm reduces PEP to graph isomorphism. While the technique is efficient for codes whose hull is either trivial or has small dimension, it yields super-exponential running time $T_{\text{BOS}} = O(n^{Rn})$ when self-orthogonal codes are considered.

Leon, *Beullens*, *BBPS*, [5,6,16]: each of these algorithms exhibits some peculiar aspects and may work only in certain regimes. For instance, while Leon's algorithm works regardless of q, Beullens' algorithm is very likely to fail when q is too small. Both of these algorithms can solve both PEP and LEP, while the BBPS algorithm improves upon Beullens' LEP algorithm by exploiting short codewords instead of subcodes. A precise estimate for the time complexity of each of these algorithms would depend on several factors which are sometimes hard to take into account. For instance, Leon requires to find all codewords whose weight is not greater than some value w which (heuristically) can be set slightly larger than the minimum distance: however, to the best of our knowledge, a formula to set w a priori is not known. In any case, these three algorithms follows a common principle, since they do not depend on the hull dimension and require to find a sufficiently large number of short codewords (or subcodes). For the sake of simplicity, for these three algorithms we consider the cost of finding a unique low-weight codeword using Prange's algorithm[14][15] Hence, for these algorithms

[14] The choice of Prange's ISD is meaningful since, for large finite fields, modern algorithms such as Lee-Brickell and Stern seem to perform worse.

[15] Even though this provides only a very broad estimate of the actual time complexity, this allows us to compare with these algorithms in a simple and concise way. We point out that cryptanalysis is not the focus of this paper and the aim of this section is merely to show that canonical forms can be a useful tool not only for the design of cryptographic schemes, but also for the cryptanalysis of the code equivalence problem.

we consider a time complexity given by

$$T = O\left(2^{\tau_{\text{Prange}}(R,q)\left(1+o(1)\right)}\right)$$

where

$$\tau_{\text{Prange}}(R,q) = h_2(R) - \left(1 - h_q^{-1}(1-R)\right) \cdot h_2\left(\frac{R}{1-h_q^{-1}(1-R)}\right)$$

and h_q denotes the q-ary entropy function.

We are now ready to compare the above algorithms with Algorithm 5; to this end, consider Figure 9. We are considering code equivalence instances for which both SSA and BOS are no efficient (i.e., PEP with self-orthogonal codes or LEP with $q \geq 5$). Note that SSA and BOS have been omitted from the comparison since their performance would have not been competitive: BOS runs in time which is super-exponential in the code length n while SSA is sometimes faster than our algorithm only if $q \leq 7$. We see that, when q is small, our algorithm is significantly slower than those based on codeword finding. Instead, when q grows, our algorithm becomes much more competitive and becomes faster then Prange.

We remark that this analysis holds given that efficiently computable canonical forms are considered. The ones introduced in this paper work whenever q is large enough, while they may yield a small success probability when q gets lower: this may make our attack slower. We have not analyzed how these canonical forms work when q gets lower; we see this, and even the question of whether new canonical forms may exist, as interesting research perspectives.

Observe that the time complexity of Prange deteriorates quickly. This is due to the fact that, as q grows, the minimum distance approaches $n-k$ (since random codes meet the Singleton's bound with high probability). Hence, there is a unique information set which would result in a success for Prange's ISD: this is corroborated by the fact that $h_q^{-1}(1-R) \to 1-R$ as q grows and $\tau_{\text{Prange}}(R,q) \to h_2(R)$. Note that this complexity coefficient is twice the one which is achieved by our algorithm.

Asymptotic cost of Prange's ISD. A random code of length n and rate R has with overwhelming probability minimum distance $d = \delta n$, where $\delta = h_q^{-1}(1-R)$ (where h_q is the q-ary entropy function). The average number of iterations which are performed by the algorithm is

$$\frac{\binom{n}{k}}{\binom{n-d}{k}} = \frac{\binom{n}{Rn}}{\binom{n(1-\delta)}{Rn}} = 2^{n\cdot\left(h_2(R)-(1-\delta)\cdot h_2\left(\frac{R}{1-\delta}\right)\right)\left(1+o(1)\right)}.$$

The cost of each iteration is that of one Gaussian elimination: this is a polynomial term so we do not consider it. Then, for the algorithm we assume a complexity coefficient given by

$$\tau_{\text{Prange}}(R,q) = h_2(R) - (1-\delta) \cdot h_2\left(\frac{R}{1-\delta}\right).$$

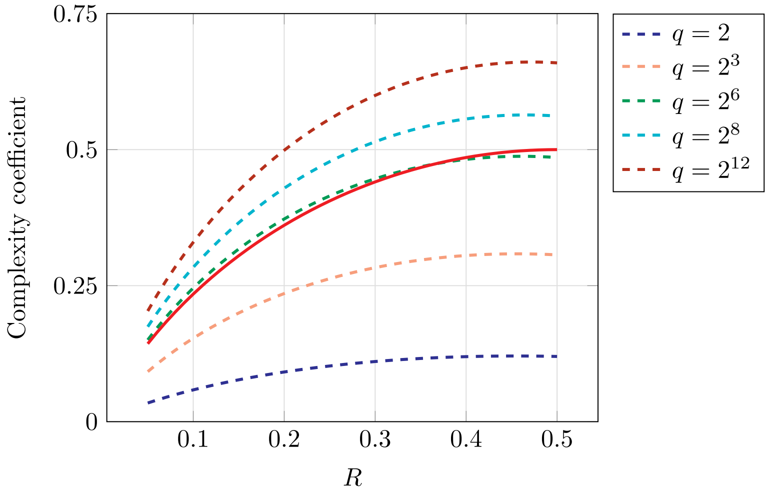

Fig. 9: Comparison between the complexity coefficients for Prange (dashed lines) and Algorithm 5 (continuous red line), as a function of the code rate.

B A Lower Bound on the Success Probability of the Canonical Form for Case 3

We derive a closed form, lower bound for the success probability of the canonical from Section 4, case 3.

Proposition 11. *For* $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$ *chosen uniformly at random, the canonical form for case* F_1 *exists with probability at least* $\prod_{i=1}^{k-1} 1 - \frac{im}{q^{n-k}}$*, where*

$$m = \begin{cases} (n-k)! & \text{if } n-k \leq q, \\ \frac{(n-k)!}{\left(v!\right)^{q(v+1)-(n-k)}\left((v+1)!\right)^{n-k-qv}} & \text{if } n-k > q, \end{cases}$$

where $v = \lfloor (n-k)/q \rfloor$.

Proof. We use $\mathbf{a}_i$ to indicate the i-th row of $\mathbf{A}$ and $\mathcal{S}(\mathbf{a}_i)$ to denote the set of vectors whose multiset is equal to that of $\mathbf{a}_i$. In other words, $\mathcal{S}(\mathbf{a}_i)$ contains all vectors that one can obtain by permuting the entries of $\mathbf{a}_i$. Remember that the canonical form computation, in this case, does not fail if the multisets of the rows $\mathbf{A}$ are all distinct. We now lower bound this probability with a simple iterative reasoning.

Let us consider $\mathbf{a}_1$ and $\mathbf{a}_2$ (the first two rows of $\mathbf{A}$): the probability that this pair of rows is valid is

$$\Pr\left[\{\mathbf{a}_1, \mathbf{a}_2\} \text{ is valid}\right] = \sum_{\mathbf{a}_i \in \mathbb{F}_q^n} \Pr\left[\mathbf{a}_2 \text{ is valid} \mid \mathbf{a}_1\right] \cdot \Pr\left[\mathbf{a}_1\right]$$

$$= \frac{1}{q^{n-k}} \sum_{\mathbf{a}_i \in \mathbb{F}_q^n} \left(1 - \frac{|\mathcal{S}(\mathbf{a}_1)|}{q^{n-k}}\right)$$

where $\Pr[\mathbf{a}_1]$ is the probability that the first row is equal to $\mathbf{a}_1$ and is equal to $q^{-(n-k)}$ for each $\mathbf{a}_1$ (since $\mathbf{A}$ is sampled according to the uniform distribution). Now, let m such that $|\mathcal{S}(\mathbf{a}_1)| \leq m$ for each possible $\mathbf{a}_1$: we get

$$\Pr[\{\mathbf{a}_1, \mathbf{a}_2\} \text{ is valid}] \geq \frac{1}{q^{n-k}} \sum_{\mathbf{a}_i \in \mathbb{F}_q^n} \left(1 - \frac{m}{q^{n-k}}\right) = 1 - \frac{m}{q^{n-k}}.$$

We now consider $\mathbf{a}_3$ and, with analogous reasoning, get that for any valid pair $\{\mathbf{a}_1, \mathbf{a}_2\}$, a new vector $\mathbf{a}_3$ is valid only if it does not belong to $\mathcal{S}(\mathbf{a}_1) \cup \mathcal{S}(\mathbf{a}_2)$. Using the upper bound m for both sets, we get that $\mathbf{a}_3$ is valid with probability at least $1 - \frac{2m}{q^{n-k}}$. If we iterate the reasoning up to the k-th row, we obtain the following probability:

$$\prod_{i=1}^{k-1} 1 - \frac{im}{q^{n-k}}.$$

Now we just need to derive useful values for m. To this end, we consider that, when $n-k \geq q$, then we can set $m = (n-k)!$: indeed, $|\mathcal{S}(a_1)| = (n-k)!$ holds only if $\mathbf{a}_1$ has all distinct entries while, otherwise $|\mathcal{S}(a_1)|$ contains fewer vectors. When $n-k > q$, we can refine the bound by taking into account that each $\mathbf{a}_i$ must necessarily have some repeated entries. The proof on how m is derived, in this case, is reported below. □

Maximum Number of Permutations for Vectors with Repeated Entries.

We study the following problem: find the maximal value that $|\mathcal{S}(\mathbf{a})|$ can have, when $\mathbf{a}$ is a length-z vector over $\mathbb{F}_q$. Let ℓ_i denote the number of entries of $\mathbf{a}$ with value equal to $x_i \in \mathbb{F}_q$ (we are writing the field as $\{x_0 = 0, x_1 = 1, x_2, \cdots, x_{q-1}\}$); note that it must be $\sum_{i=0}^{q-1} \ell_i = z$. The values ℓ_i allow us to take into account the number of permutations with repetitions, so that

$$|\mathcal{S}(\mathbf{a})| = \frac{z!}{\prod_{i=0}^{q-1} \ell_i!} = \frac{z!}{f(\ell_0, \ldots, \ell_{q-1})}.$$

Maximizing $|\mathcal{S}(\mathbf{a})|$ means minimizing $f(\ell_0, \ldots, \ell_{q-1})$: as we show next, this is achieved when all values ℓ_i are balanced, i.e., the difference between any pair of values ℓ_i, ℓ_j is not greater than 1.

Proposition 12. *For any $(\ell_0, \cdots, \ell_{q-1}) \in \mathbb{N}^q$ such that $\sum_{i=0}^{q-1} \ell_i = z$, it holds that*

$$f(\ell_0, \cdots, \ell_{q-1}) \geq (v!)^{q(v+1)-z} \left((v+1)!\right)^{z-qv},$$

where $v = \left\lfloor \frac{z}{q} \right\rfloor$.

Proof. The proof is crucially based on the simple observation that

$$\forall x, y \in \mathbb{N}, \text{ it holds } \quad y!x! > (y-1)!(x+1)! \text{ if } y - x > 1.$$

Let us consider an arbitrary tuple $(\ell_0, \cdots, \ell_{q-1})$, summing to z, and assume there are two values ℓ_j, ℓ_u such that $\ell_j - \ell_u > 1$. Then, there exists a new tuple $(\ell'_0, \cdots, \ell'_{q-1})$ such that $\ell'_i = \ell_i$ for any $i \neq j, u$, $\ell'_j = \ell_j - 1$ and $\ell'_u = \ell_u + 1$. First, this configuration is valid since the sum of all the ℓ'_i is still equal to z. Also, because of (B), we have that

$$\frac{f(\ell_0, \cdots, \ell_{q-1})}{f(\ell'_0, \cdots, \ell'_{q-1})} = \frac{\prod_{i=0}^{q-1} \ell_i!}{\prod_{i=0}^{q-1} \ell'_i!} = \frac{\ell_j! \ell_u!}{\ell'_j! \ell'_u!} = \frac{\ell_j! \ell_u!}{(\ell_j - 1)!(\ell_u + 1)!} > 1.$$

We can iterate the procedure until we end up with a tuple where, for each pair of values, the difference is at most 1. This implies that there are only two possible values in the tuple, $v = \left\lfloor \frac{z}{q} \right\rfloor$ and $v + 1$. Let w denote the number of entries with value v: since it must be $vw + (q - w)(v + 1) = z$, we find $w = q(v + 1) - z$. So, the number of entries with value equal to $v + 1$ is $q - w = z - qv$. □

It follows that

$$\forall \mathbf{a} \in \mathbb{F}_q^z, \ |\mathcal{S}(\mathbf{a})| \leq \frac{z!}{\left(\lfloor z/q \rfloor !\right)^{q\left(\left\lfloor \frac{z}{q} \right\rfloor + 1\right) - z} \left((\lfloor z/q \rfloor + 1)!\right)^{z - q\left\lfloor \frac{z}{q} \right\rfloor}}.$$