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 of $\mathbf{2\lambda}$.

In this paper, we fill this gap and propose a new notion of equivalence, which leads to a drastically reduced witness size. In particular, for many cases, the size obtained is exactly the optimal one given by the lower bound. We do this by introducing the framework of *canonical forms*, 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, providing reductions in both ways. As an added consequence, we show how 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 analyze the impact of our technique, showing that it 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 post-quantum ecosystem.

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 (which are conveniently communicated using seeds of length λ) or proofs of equivalence between two codes. For the latter type, the prover must communicate either a length-n permutation or a length-n monomial map (depending on which notion of equivalence is employed), yielding 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.

In [17], 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; this yields a significant reduction as the cost for communicating an isometry is reduced 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, [17] 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 the NIST's call for additional post-quantum signatures [16].

As is well known, a theoretical lower bound on the size of non-ephemeral messages in a ZK protocol based on group actions is 2λ. Despite the work in [17], the communication cost for the code equivalence group action is still far from being optimal. This can be easily acknowledged by looking at the proposed LESS instances. Since $q = 127$, $R = \frac{1}{2}$ and $n \approx 2\lambda$, we have that 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]

1.1 Our Contributions

We show how to drastically reduce the size of witnesses to the code equivalence problem: for both PEP and LEP, we bring it down to $\log_2 \binom{n}{k}$ bits. For $k = Rn$, this corresponds to $n \cdot h(R) \cdot \big(1 + o(1)\big)$, where h denotes the binary entropy function. Perhaps surprisingly, the size is the same regardless of which notion of equivalence is considered. 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 $\tilde{O}\left(\sqrt{\binom{n}{k}}\right) = 2^{\frac{1}{2}n \cdot h(R) \cdot \left(1+o(1)\right)}$ 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 $n \cdot h(R) \cdot \big(1 + o(1)\big) = 2\lambda\big(1 + o(1)\big)$.

We apply this machinery to LESS signatures. As expected, the resulting scheme, which we call CF-LESS, achieves extremely 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 $2^{\frac{1}{2}n \cdot h(\frac{1}{2}) \cdot \left(1+o(1)\right)} \approx 2^{\lambda\left(1+o(1)\right)}$ 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 down to $n \cdot h(\frac{1}{2}) = n \approx 2\lambda$ bits. In practice, this implies that we can reduce signature sizes of LESS 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 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 proposed in [3], obtaining a comparable gain in signature size (depending on the amount of users in the ring).

[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.

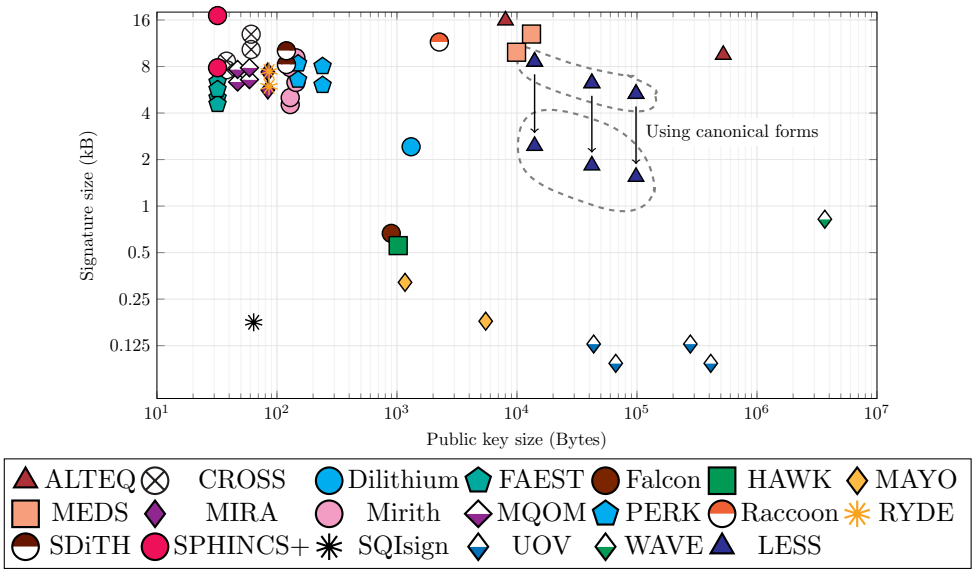

Fig. 1: Comparison between CF-LESS and some selected schemes from the NIST competition (including the current version of LESS), for NIST category 1.

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 form* of a code where, by canonical forms, we refer to the representative of some equivalence class. We introduce canonical forms for codes, show how they can be computed efficiently and how they relate with existing notions of equivalence. This even 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 form. 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 [17]. Indeed, using the vocabulary of canonical forms, we can say that IS-PEP and IS-LEP correspond to specific cases of equivalences based on canonical forms. 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, by defining canonical form functions having much stronger invariance properties: this is a more challenging task than the one faced in [17], which we solve by providing efficient examples of such functions, for all relevant cases.

1.2 Technical Overview

The main theoretical contribution of this paper is in the introduction of canonical forms for codes. The concept of canonical forms is already widely employed in the study of other mathematical problems, e.g., graph theory, in which a canonical form is assigned to a graph by application of an isomorphism. All the graphs in the same class have the same canonical form, while graphs belonging to different classes (i.e., which are not isomorphic) have different canonical forms.

We define and employ canonical forms in an analogous way, by deriving equivalence classes from the traditional notions of code equivalence, and defining canonical forms as representatives of the classes.

Equivalence Classes and Canonical Forms for Codes

Let S_m denote the symmetric group on m objects and D_m the group of non-singular $m \times m$ diagonal matrices. Let $F \in \tilde{D}_k \times \tilde{S}_k \times \tilde{D}_{n-k} \times \tilde{S}_{n-k}$, where

$$\tilde{D}_m \in \big\{\{\mathbf{I}_m\}\ ,\ D_m\big\}, \quad \tilde{S}_m = \big\{\{\mathbf{I}_m\}\ ,\ S_m\big\}, \forall m \in \mathbb{Z}.$$

Let $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ be the generator matrix of a code $\mathscr{C} \subseteq \mathbb{F}_q^{k\times n}$ with dimension k. Once F is defined, we obtain the *equivalence class* of $\mathbf{G}$ as follows

$$\mathfrak{C}_F(\mathbf{G}) = \left\{\mathbf{S}\cdot\mathbf{G}\cdot\mathbf{Q}\ :\ \mathbf{S}\in \mathrm{GL}_k,\ \mathbf{Q} = \begin{bmatrix}\mathbf{D}_r^{-1}\cdot\mathbf{P}_r^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_c\cdot\mathbf{P}_c\end{bmatrix},\ (\mathbf{D}_r,\mathbf{P}_r,\mathbf{D}_c,\mathbf{P}_c)\in F\right\}$$

where GL_k is the general linear group of $k \times k$ matrices. For two matrices $\mathbf{G}$, $\mathbf{G}'$ in the same class, we write $\mathbf{G} \sim_{\mathsf{R}'_F} \mathbf{G}'$.[4]

By definition, the equivalence class is invariant under changes of basis, that is, $\mathfrak{C}_F(\mathbf{G}) = \mathfrak{C}_F(\mathbf{U}\cdot\mathbf{G})$ for any $\mathbf{U} \in \mathrm{GL}_k$. Since $\mathbf{G}$ and $\mathbf{U}\cdot\mathbf{G}$ generate the same code, we can associate the equivalence class with the code $\mathscr{C}$, as follows:

$$\mathfrak{C}_F(\mathscr{C}) = \left\{\mathbf{c}\cdot\mathbf{Q}\ :\ \mathbf{c}\in\mathscr{C},\ \mathbf{Q} = \begin{bmatrix}\mathbf{D}_r^{-1}\cdot\mathbf{P}_r^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_c\cdot\mathbf{P}_c\end{bmatrix},\ (\mathbf{D}_r,\mathbf{P}_r,\mathbf{D}_c,\mathbf{P}_c)\in F\right\}.$$

Since the matrix $\mathbf{Q}$ defines a monomial map μ, the above class contains codes which are linearly equivalent. However, since the allowed transformations have a very specific structure, $\mathfrak{C}_F(\mathscr{C})$ contains only a subset of the codes which are linearly equivalent with $\mathscr{C}$.[5] Depending on how F is defined, the number and structure of transformations which are allowed in (1.2) varies. For instance, if $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times S_{n-k}$, the transformation fixes the first k coordinates.

To each class, we assign a representative which we call *canonical form*. For our purposes, this is defined as a function $\mathsf{CF}_{\mathsf{R}'_F} : \mathrm{GL}_{k,n} \to \big\{\{\perp\} \cup \mathrm{GL}_{k,n}\big\}$, where $\mathrm{GL}_{k,n}$ denotes the set of full rank $k \times n$ matrices and $\perp$ the failure symbol, such that:

[4] At this stage, the notation may seem unnecessarily complex. The reasons behind this choice will be clarified in the remainder of the paper.

[5] Using the group action terminology, $\mathfrak{C}_F(\mathscr{C})$ is contained in the orbit of $\mathscr{C}$ under the action of M_n.

i) for any $\mathbf{G} \in \mathrm{GL}_{k,n}$, $\mathsf{CF}_{\mathsf{R}'_F}(\mathbf{G}) \neq \perp \implies \mathbf{G} \sim_{\mathsf{R}'_F} \mathsf{CF}_{\mathsf{R}'_F}(\mathbf{G})$;
ii) for any $\mathbf{G}, \mathbf{G}' \in \mathrm{GL}_{k,n}$, $\mathbf{G} \sim_{\mathsf{R}'_F} \mathbf{G}' \implies \mathsf{CF}_{\mathsf{R}'_F}(\mathbf{G}) = \mathsf{CF}_{\mathsf{R}'_F}(\mathbf{G}')$.

We say that $\mathsf{CF}_{\mathsf{R}'_F}$ has *success probability* γ if the canonical form is well-defined for a fraction γ of the possible $q^{k(n-k)}$ input matrices.

In this paper we are interested only in canonical forms that can be computed *efficiently*, that is, in time which is polynomial in the input size. Note that, for some choices for F, we are able to identify functions having a null failure probability (i.e., $\gamma = 1$): we emphasize this by referring to such functions as *proper canonical forms*. We are not able to define proper canonical forms for all choices for F; actually, we argue that to have $\gamma = 1$ one would probably need to get rid of the polynomial time requirement, which however is fundamental for the applications we consider. Still, we show that when γ is high enough this is not an issue as the impact on both security and efficiency of the resulting schemes is very mild.

Since the canonical form is an element of the class, we view it as a "special" generator for a "special" code: it generates a special code in the class and, among all generator matrices for the same code, we select a very specific one. In particular, we define $\mathsf{CF}_{\mathsf{R}'_F}$ so that, on input some $\mathbf{G}$ (for which the canonical form exists), the output is a matrix of the form $[\mathbf{I}_k \mid \mathbf{A}]$ with $\mathbf{A}$ being a $k \times (n-k)$ matrix. In the end, the algorithmic task of computing canonical forms will boil down to that of finding representatives for another equivalence class, defined on $k \times (n-k)$ matrices and induced by another equivalence relation R_F which depends on F. We do not give more details here but simply mention that the whole theory of canonical forms applies also to this new equivalence relation; in the following, we will denote by $\mathsf{CF}_{\mathsf{R}_F}$ a canonical form function for this class.

Proving Equivalence between Codes

To show that two codes $\mathscr{C}$ and $\mathscr{C}'$ are equivalent according to permutation or linear equivalence, one can consider the procedure defined in Figure 2. The proof for the equivalence is provided by communicating τ, which may be different from μ^{-1}: to verify that $\mathscr{C}$ and $\mathscr{C}'$ are equivalent, one verifies that $\mathscr{C}$ and $\widetilde{\mathscr{C}} = \tau(\mathscr{C}')$ are in the same equivalence class. This can be done by checking that $\mathsf{CF}_{\mathsf{R}'_F}(\mathscr{C}) = \mathsf{CF}_{\mathsf{R}'_F}(\widetilde{\mathscr{C}})$.

Note that τ must be some special transformation, with a structure that depends on how F is chosen. The general idea is that one can relax the requirements on τ, allowing it to bring only *a portion of the information contained* in μ^{-1}. This lack of information is compensated by the invariance properties of the canonical form $\mathsf{CF}_{\mathsf{R}'_F}$: as τ brings less information, the associated equivalence class becomes wider. Reducing the information represented by τ is exactly what allows for reducing the binary size for witnesses to the equivalence problem. The operations in the top part of Figure 2 can be composed, obtaining a map from $\mathscr{C}$ to $\mathscr{C}'$ which may be different from μ. For each choice of F, this map corresponds to a specific notion of equivalence. Interestingly, this framework encompasses all existing notions of equivalence and, more interestingly, allows to derive more powerful notions of equivalence.

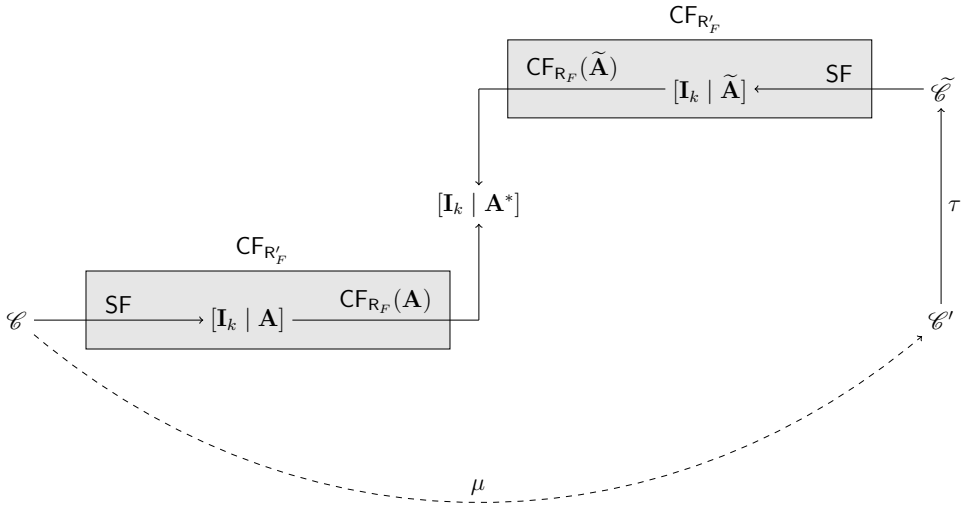

Fig. 2: Graphical representation of how to prove equivalence between two codes $\mathscr{C}$, $\mathscr{C}'$. The figure emphasizes that the canonical form of an equivalence class is just a generator matrix in systematic form and that computing $\mathsf{CF}_{\mathsf{R}'_F}$ boils down to first compute a systematic form and then apply $\mathsf{CF}_{\mathsf{R}_F}$ to the non-systematic part.

PEP and LEP

We first consider the case in which all components of F are trivial, i.e., $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times \{\mathbf{I}_{n-k}\} \times \{\mathbf{I}_{n-k}\}$. In such a case, the canonical form $\mathsf{CF}_{\mathsf{R}'_F}$ is trivial since, by the requirements of the canonical form, it must be $\mathsf{CF}_{\mathsf{R}'_F}(\mathbf{G}) = \mathsf{CF}_{\mathsf{R}'_F}(\mathbf{G}')$ (where $\mathbf{G}$ and $\mathbf{G}'$ denote any two generator matrices of $\mathscr{C}$ and $\mathscr{C}'$, respectively). Thus, each equivalence class contains $\mathfrak{C}_F(\mathbf{G})$ boils down to the set of all generator matrices for the same code. The equivalence between $\mathscr{C}$ and $\widetilde{\mathscr{C}}$ is verified by checking that the two codes have the same systematic form. This requires that $\widetilde{\mathscr{C}}$ and $\mathscr{C}$ are the same code, thus, it must be $\tau = \mu^{-1}$: the employed notion of equivalence is the standard one.

IS-PEP and IS-LEP

We focus on IS-LEP, as IS-PEP trivially follows by restricting to all the monomials with all scaling factors set to 1. In this case, not all components of F are trivial since $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times S_{n-k}$. All codes in the equivalence class $\mathfrak{C}_F(\mathscr{C})$ are equal up to a monomial map that fixes the first k coordinates, i.e., permutes and scales only the last $n-k$ coordinates. In other words, the equivalence class is derived by considering all monomial maps acting only on the coordinates that are outside of the information set $\{1, \cdots, k\}$. Thus, τ can be obtained from μ^{-1} by simply omitting how μ^{-1} acts outside of the information set: $\mathscr{C}$ and $\widetilde{\mathscr{C}}$ are equal in the first k coordinates, thus $\mathfrak{C}_F(\mathscr{C}) = \mathfrak{C}_F(\widetilde{\mathscr{C}})$ and they possess the same canonical form.

This is, essentially, a translation of the treatment in [17], using the vocabulary of canonical forms. The authors even define an ad-hoc function (called Lex) which is, de facto, an example of a proper canonical form. The associated equivalence notion differs from the traditional LEP, as τ sends $\mathscr{C}'$ to a code which is different from $\mathscr{C}$. The authors of [17] refer to the corresponding notions as IS-LEP and IS-PEP and show that they are just a different formulation of the traditional notions.

Going Beyond IS-LEP and IS-PEP

The theory of canonical forms allows to generalize the framework in [17], by defining wider equivalence classes associated with shorter witnesses to the equivalence problem. We show this by directly considering the most general case, in which all the four components of F are non-trivial. Let us factor μ^{-1} as

$$\mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{D}_r^{-1} \cdot \mathbf{P}_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c \cdot \mathbf{P}_c \end{bmatrix}$$

where $\mathbf{P}_{\text{is}} \in S_n$ is a special permutation telling only which coordinates are moved to the leftmost k coordinates. In practice, this permutation brings only information about which coordinates shall be used to constitute the information set which is later used for the systematic form. We define the corresponding subset of S_n as $S_{k,n}$, which has size $\binom{n}{k} = \binom{n}{Rn} = 2^{n \cdot h(R) \cdot \left(1+o(1)\right)}$.

If τ corresponds to $\mathbf{P}_{\text{is}}$ then $\widetilde{\mathscr{C}}$ is in the same class of $\mathscr{C}$, since for each codeword in $\widetilde{\mathscr{C}}$ there is a codeword in $\mathscr{C}$ which is equal, up to a monomial acting on the first k coordinates and another acting on the last $n-k$. Assuming we have an efficiently computable function $\mathsf{CF}_{\mathsf{R}'_F}$, we can acknowledge that indeed $\mathfrak{C}_F(\mathscr{C}) = \mathfrak{C}_F(\widetilde{\mathscr{C}})$.

Efficient Canonical Form Functions and Dealing with Failures

Finding efficient canonical form functions seems to be a rather challenging task, at least for broad classes of equivalence. As we already mentioned, we circumvent the issue by allowing for canonical forms that admit a (small) *failure probability*. In other words, we relax the requirements for canonical forms and accept that the function may not be defined for some input matrices. We measure the failure probability as the ratio between the number of inputs leading to a failure and the overall number of inputs.

In this paper we introduce canonical form functions for five cases; the first two cases are trivial and already considered in [17], while the new functions we propose are for the following three cases:

- Case 3: $F = \{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$
- Case 4: $F = D_k \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$
- Case 5: $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$

For all three cases, we find functions that run in worst-case polynomial time, with the case 5 function being the slowest, requiring $\tilde{O}(n^3)$ in the worst cases. In particular, the function for case 5 calls, as fundamental subroutines, the canonical forms defined for cases 3 and 4.

We take into account the existence of failures and, for all the canonical form functions we define, assess such probability with theoretical arguments. Note that our theoretical assessment of failure probabilities relies on heuristics which are expected to hold for random codes. This is exactly the setting which is considered in LESS, as the public key is generated by first sampling at random $\mathscr{C}$ and μ, and then setting $\mathscr{C}' = \mu(\mathscr{C})$. In particular, for all LESS instances, the resulting failure probabilities are always very small (e.g., less than 2^{-83} for NIST category 1). To further verify these numbers, we ran intensive numerical simulations testing, for each LESS instance, 2^{20} different pairs of codes: in all our tests, we never witnessed a failure. Even in the (unlikely) event of failure, one can simply restart the signing procedure, so having a non-zero failure rate does not really hurt.

New Notions of Equivalence

Analogously to the treatment in [17], using canonical forms leads to new notions of equivalence. We show that, for each of these new notions, there exists bidirectional links to the traditional notions. We provide reductions considering the most general notion of equivalence (in which all four components of F are non-trivial); all other notions follow as special cases. We call the associated problem CF-LEP and show that, given a solution to CF-LEP, one can compute a solution to LEP via simple linear algebra manipulations.

Note that the existence of failures also affects this reduction, as we cannot guarantee that CF-LEP admits a solution for every LEP instance. Still, for random codes, we can estimate the probability that at least one solution to CF-LEP exists as $1-2^{-\gamma\binom{n}{k}\frac{(1-1/q-1/q^2)}{\ln(2)}} = 1-2^{\psi}$, where γ denotes the success probability of the employed canonical form. Unless γ is exponentially small (say, comparable with $1/\binom{n}{k}$), the previous probability is very close to 1.

The quantity $1-2^{\psi}$ corresponds to the expected fraction of codes for which CF-LEP cannot be solved. It is clear that, if this number is close to 1, this may imply that CF-LEP is much easier than LEP (as an algorithm solving CF-LEP would solve only a small fraction of LEP instances). For the considered LESS instances, we always have $\psi \ll 0$: for instance, taking into account the failure probabilities of the canonical form functions we define in this paper, for all NIST Category 1 instances we have $\psi \leq -252$.

Finally, the reduction from LEP to CF-LEP can also be used, constructively, to build a new attack on the equivalence problem. Our solver first solves CF-LEP thanks to a birthday attack, then reconstruct the solution to LEP. The birthday attack phase is done considering collisions on canonical forms sampling, for each code, a sufficiently large number of permutations from $S_{k,n}$. Since the solution space of CF-LEP is $S_{k,n}$, thanks to the birthday paradox we are able to solve the problem in time $\widetilde{O}\left(\frac{1}{\sqrt{\gamma}} \cdot \sqrt{\binom{n}{k}}\right)$. Whenever γ is large enough (namely, when it is not negligible in n), this yields an attack running in time $2^{\frac{1}{2}n \cdot h(R) \cdot \left(1+o(1)\right)}$.

1.3 Paper Organization

The paper is organized as follows. Section 2 specifies our notation and summarizes some preliminary notions. Section 3 describes the equivalence relations that we consider for matrices over $\mathbb{F}_q$, and introduces the concept of canonical forms. This serves as an important basis for the discussions in the sections that follow. Section 4 shows concrete ways to define canonical forms, starting from existing ones and including some entirely new formulations. 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. 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.

Matrices and Maps

As usual, $\mathbb{F}_q$ denotes the finite field with q elements and $\mathbb{F}_q^*$ stands for its multiplicative group. We use bold uppercase (resp., lowercase) letter for matrices (resp., vectors). Given a vector $\mathbf{v}$, we define $\mathsf{multiset}(\mathbf{v})$ as the multiset formed by the entries of $\mathbf{v}$. We use three main matrix groups in our work, which are all subgroups of the general linear group GL_n of non-singular matrices over $\mathbb{F}_q$. With some abuse of notation, we denote by $\mathrm{GL}_{k,n} \subset \mathbb{F}_q^{k\times n}$, with $k < n$, the set of matrices where the first k columns form an element in GL_k. The first, is the symmetric group, i.e. the group of permutations on n objects, represented as binary matrices with a single 1 in each row and column; this is denoted by S_n. Second, we consider the group comprised of diagonal matrices over $\mathbb{F}_q$, such that all elements on the main diagonal are non-zero; we denote this by D_n. Finally, we consider a group which is a generalization of S_n, where each permutation is scaled with non-zero factors from $\mathbb{F}_q$; we denote this by M_n. In other words, for each $\mathbf{Q} \in M_n$, we have that $\mathbf{Q} = \mathbf{P} \cdot \mathbf{D}$, where $\mathbf{P} \in S_n$ and $\mathbf{D} \in D_n$. Such matrices are known as *monomial matrices*, and we will thus refer to the group M_n as the *monomial group* over $\mathbb{F}_q$.

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, for every $\mathbf{P} \in S_{k,n}$,

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

In fact, once the first k columns of a matrix in $S_{k,n}$ is defined, the whole matrix is defined. Applying a matrix from $S_{k,n}$ can be seen as dividing coordinates into two

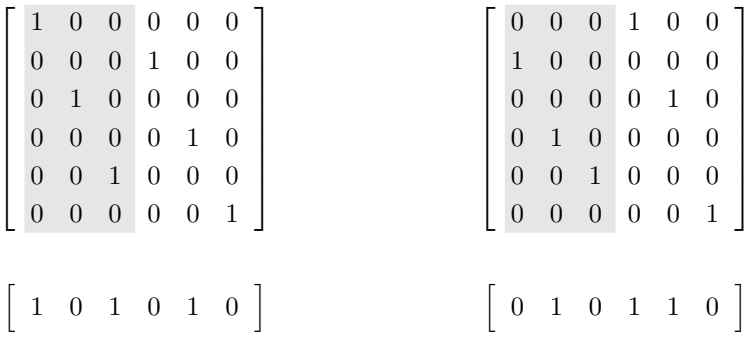

Fig. 3: Examples of matrices from $S_{k,n}$ and their binary representation, for $n = 6$ and $k = 3$. The coordinates that are moved to the leftmost positions are $\{1, 3, 5\}$ for Figure 3a and $\{2, 4, 5\}$ for Figure 3b.

groups: k coordinates that are moved to the leftmost positions, and $n-k$ coordinates that are moved to the rightmost positions. Note that $|S_{k,n}| = \binom{n}{k} \leq n$. Every matrix in $S_{k,n}$ admits a very simple binary representation: if $k \leq n/2$, it can be conveniently represented as a binary vector of length n and Hamming weight k, with the ones corresponding to the coordinates that are moved to the leftmost positions (i.e., the row indices of the 1's in the first k columns are exactly the indices of the 1's in the vector). If $k > n/2$, we employ the same representation but revert the role of zeros and ones. Examples of matrices from $S_{k,n}$ are shown in Figure 3.

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 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. Given $d \in \{k, n-k\}$, 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.

We define a strict partial ordering $\prec_{\mathbb{F}_q^d}$ on $\mathbb{F}_q^d$. Given $\mathbf{v}, \mathbf{v}' \in \mathbb{F}_q^d$, we consider $\mathbf{v} \prec_{\mathbb{F}_q^d} \mathbf{v}'$ 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 in $\mathbb{F}_q^d$ w.r.t. $\prec_{\mathbb{F}_q^d}$ is considered to take $\tilde{O}(d)$ field operations. [6] 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

The matrices considered in this work are mostly treated in the context of coding theory. A *linear code* $\mathscr{C}$ with *dimension* $k > 0$ and *length* $n > k$ is a k-dimensional linear subspace of $\mathbb{F}_q^n$ and, as such, can be represented compactly by a full rank matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ which serves as a basis, i.e., $\mathscr{C} = \left\{\mathbf{u} \cdot \mathbf{G} : \mathbf{u} \in \mathbb{F}_q^k\right\}$. 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 $\mathbf{S} \in \mathrm{GL}_k$, 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 corresponding 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, let $\mathbf{G}_J$ be the submatrix of $\mathbf{G}$ with columns indexed by J. 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[7], 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 form. In the following, we will sometime use RREF^* to define the function returning both the systematic form, as well as the permutation $\mathbf{P} \in S_{k,n}$. SF and RREF are invariant under changes of basis: for any two generator matrices $\mathbf{G}$ and $\mathbf{S} \in \mathrm{GL}_k$, it holds that $\mathsf{SF}(\mathbf{G}) = \mathsf{SF}(\mathbf{SG})$ and $\mathsf{RREF}(\mathbf{G}) = \mathsf{RREF}(\mathbf{S} \cdot \mathbf{G})$. If $\{1, \cdots, k\}$ is an information set, then SF and RREF coincide and $\mathbf{P} = \mathbf{I}_n$.

The Code Equivalence Problem

Linear codes are typically measured in the *Hamming metric*, which defines the *weight* of a (code)word as the number of its non-zero positions. Indeed, the precise distribution of weights of the various codewords, and the *distance* between them, is exactly what

[6] By comparing two elements $\mathbf{v}, \mathbf{v}' \in \mathbb{F}_q^d$ we mean 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.

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

characterizes the code, enabling features such as error detection and correction. With this in mind, it is then natural to consider *equivalent* two codes having an identical weight distribution. Equivalent codes are obtained from one another via an *isometry*, i.e. a map preserving distances. In the simplest of cases, such a map consists of just a permutation, which leads to the notion of permutation equivalence; if instead the map is a monomial one, this is usually known as *linear equivalence*. The more general notion of *semilinear equivalence* refers to the addition of a field automorphism; this concept is not relevant for cryptographic applications, and we do not treat it here. It is immediate to note that permutation equivalence is nothing but a special case of linear equivalence. Equivalent codes can be easily represented via their generator matrices, so that, if $\mathscr{C}$ is permutationally (resp. linearly) equivalent to $\mathscr{C}'$, then there exist a change-of-basis matrix $\mathbf{S} \in \mathrm{GL}_k$ and a permutation $\mathbf{P} \in S_n$ (resp. monomial $\mathbf{Q} \in M_n$) such that $\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}$, where $\mathbf{G}$ and $\mathbf{G}'$ are any two generator matrices for $\mathscr{C}$ and $\mathscr{C}'$, respectively.

Determining whether two codes are equivalent is often a challenging task. In cryptography, one is usually interested in the computational version of the problem, which we report below.

Problem 1 (**Code Equivalence**). *Given linear codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ with dimension k, defined by generator matrices $\mathbf{G}, \mathbf{G}'$ respectively, decide if $\mathscr{C}$ is equivalent to $\mathscr{C}'$, i.e., if there exists $\mathbf{S} \in \mathrm{GL}_k$ and $\mathbf{P} \in S_n$ (resp. $\mathbf{Q} \in M_n$) such that $\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{P}$ (resp. $\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \mathbf{Q}$).*

The problem is known respectively as *Permutation Equivalence Problem (PEP)* or *Linear Equivalence Problem (LEP)*, depending on which kind of equivalence is involved. The hardness of the code equivalence problem has been studied in detail through several works, and we point the interested reader to [5,6,19] for an extensive treatment.

3 Equivalence Relations on Matrices and Codes

Let $n, k, q \in \mathbb{Z}$ be the typical coding theory parameters, as defined above. Let $\tilde{D}_m \in \{\{\mathbf{I}_m\}, D_m\}$ and let $\tilde{S}_m \in \{\{\mathbf{I}_m\}, S_m\}$ for any positive integer m. The equivalence relations introduced in this sections are defined using a set $F \in \tilde{D}_k \times \tilde{S}_k \times \tilde{D}_{n-k} \times \tilde{S}_{n-k}$. To each of the 16 cases for F, we associate a notion of equivalence relation on $\mathbb{F}_q^{k\times(n-k)}$ and $\mathrm{GL}_{k,n}$.

3.1 New Equivalence Relations on Matrices

Definition 1 (**Equivalence Relations on** $\mathrm{GL}_{k,n}$). *Given $F \in \tilde{D}_k \times \tilde{S}_k \times \tilde{D}_{n-k} \times \tilde{S}_{n-k}$, we say that two matrices $\mathbf{A}, \mathbf{B} \in \mathrm{GL}_{k,n}$ are equivalent with respect to R'_F (denoted as $\mathbf{A} \sim_{\mathsf{R}'_F} \mathbf{B}$) if*

$$\mathbf{B} = \mathbf{T} \cdot \mathbf{A} \cdot \begin{bmatrix} \mathbf{D}_r^{-1}\mathbf{P}_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c \end{bmatrix}$$

for some $\mathbf{T} \in \mathrm{GL}_k$ and $(\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c) \in F$.

Remark 1. *Equivalently, one can view* R'_F *as an equivalence relation between codes formed by row spaces of matrices in* $\mathrm{GL}_{k,n}$*. In other words, two codes* $\mathscr{C}$ *and* $\mathscr{C}'$*, defined by generator matrices* $\mathbf{G}, \mathbf{G}'$ *respectively, are considered equivalent under* R'_F *if*

$$\mathbf{G}' = \mathbf{S} \cdot \mathbf{G} \cdot \begin{bmatrix} \mathbf{D}_r^{-1}\mathbf{P}_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c \end{bmatrix}$$

for some $\mathbf{S} \in \mathrm{GL}_k$ *and* $(\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c) \in F$*.*

R'_F *is clearly a subcase of linear equivalence since the monomial in the above definition has a very special structure: it never mixes the first* k *columns with the last* $n-k$ *ones.*

Replacing $\mathbf{D}_r^{-1}\mathbf{P}_r^{-1}$ by $\mathbf{D}_r\mathbf{P}_r$ in the above definition does not change R'_F. We used $\mathbf{D}_r^{-1}\mathbf{P}_r^{-1}$ because this is convenient for explaining the relationship between R'_F and an equivalence relation on $\mathbb{F}_q^{k\times(n-k)}$ which is defined below.

Definition 2 (**Equivalence Relations on** $\mathbb{F}_q^{k\times(n-k)}$)**.** *Given* $F \in \tilde{D}_k \times \tilde{S}_k \times \tilde{D}_{n-k} \times \tilde{S}_{n-k}$*, we say that two matrices* $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{k\times(n-k)}$ *are equivalent with respect to* R_F *(denoted as* $\mathbf{A} \sim_{\mathsf{R}_F} \mathbf{B}$*) if*

$$\mathbf{B} = \mathbf{P}_r \cdot \mathbf{D}_r \cdot \mathbf{A} \cdot \mathbf{D}_c \cdot \mathbf{P}_c$$

for some $(\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c) \in F$[8] *.*

Note that replacing $\mathbf{P}_r \cdot \mathbf{D}_r$ by $\mathbf{D}_r \cdot \mathbf{P}_r$ or replacing $\mathbf{D}_c \cdot \mathbf{P}_c$ by $\mathbf{P}_c \cdot \mathbf{D}_c$ does not change the definition of R_F. The following theorem explains the relationship between R'_F and R_F.

Theorem 1. *Let* $\mathbf{A}, \mathbf{B} \in \mathrm{GL}_{k,n}$*. Let* $[\mathbf{I}_k \mid \widehat{\mathbf{A}}]$ *and* $[\mathbf{I}_k \mid \widehat{\mathbf{B}}]$ *be the respective systematic generator matrices. Then*

$$\mathbf{A} \sim_{\mathsf{R}'_F} \mathbf{B} \iff \widehat{\mathbf{A}} \sim_{\mathsf{R}_F} \widehat{\mathbf{B}}.$$

Moreover,

$$\mathbf{B} = \mathbf{T} \cdot \mathbf{A} \cdot \begin{bmatrix} \mathbf{D}_r^{-1}\mathbf{P}_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c \end{bmatrix} \iff \widehat{\mathbf{B}} = \mathbf{P}_r \cdot \mathbf{D}_r \cdot \widehat{\mathbf{A}} \cdot \mathbf{D}_c \cdot \mathbf{P}_c$$

where $(\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c) \in F$ *and* $\mathbf{T} \in \mathrm{GL}_k$*.*

Proof. Let $\mathbf{A} = (\mathbf{A}_1 \mid \mathbf{A}_2)$, with $\mathbf{A}_1 \in \mathrm{GL}_k$ and $\mathbf{A}_2 \in \mathbb{F}_q^{k\times(n-k)}$. Since by hypothesis $\{1, \cdots, k\}$ is an information set, the systematic form for $\mathbf{A}$ is $\mathbf{A}_1^{-1}\mathbf{A}$, so that $\widehat{\mathbf{A}} = \mathbf{A}_1^{-1} \cdot \mathbf{A}_2$. Using analogous notation for $\mathbf{B}$, we get $\widehat{\mathbf{B}} = \mathbf{B}_1^{-1} \cdot \mathbf{B}_2$. For the direction $\Longrightarrow$, it is enough to consider that $\mathbf{B}_1 = \mathbf{T} \cdot \mathbf{A}_1 \cdot \mathbf{D}_r^{-1} \cdot \mathbf{P}_r^{-1}$ and $\mathbf{B}_2 = \mathbf{T} \cdot \mathbf{A}_2 \cdot \mathbf{D}_c \cdot \mathbf{P}_c$,

[8] The subscripts r and c stand for row and column, respectively. We use these subscripts because in Definition 2, $\mathbf{P}_r, \mathbf{D}_r$ can be considered as row operations and $\mathbf{P}_c, \mathbf{D}_c$ can be considered as column operations.

which implies

$$\begin{aligned}\widehat{\mathbf{B}} &= \underbrace{(\mathbf{T}\cdot\mathbf{A}_1\cdot\mathbf{D}_r^{-1}\cdot\mathbf{P}_r^{-1})^{-1}}_{\mathbf{B}_1^{-1}}\cdot\underbrace{\mathbf{T}\cdot\mathbf{A}_2\cdot\mathbf{D}_c\cdot\mathbf{P}_c}_{\mathbf{B}_2}\\ &= \mathbf{P}_r\cdot\mathbf{D}_r\cdot\underbrace{\mathbf{A}_1^{-1}\cdot\mathbf{A}_2}_{\widehat{\mathbf{A}}}\cdot\mathbf{D}_c\cdot\mathbf{P}_c.\end{aligned}$$

For the direction $\Longleftarrow$, we have

$$\begin{aligned}\mathbf{B}_1^{-1}\underbrace{\begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_1\cdot\widehat{\mathbf{B}}\end{bmatrix}}_{\mathbf{B}} &= \begin{bmatrix}\mathbf{I}_k & \widehat{\mathbf{B}}\end{bmatrix} = \begin{bmatrix}\mathbf{I}_k & \mathbf{P}_r\cdot\mathbf{D}_r\cdot\widehat{\mathbf{A}}\cdot\mathbf{D}_c\cdot\mathbf{P}_c\end{bmatrix}\\ &= \mathbf{P}_r\cdot\mathbf{D}_r\begin{bmatrix}\mathbf{I}_k & \widehat{\mathbf{A}}\end{bmatrix}\begin{bmatrix}\mathbf{D}_r^{-1}\cdot\mathbf{P}_r^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_c\cdot\mathbf{P}_c\end{bmatrix}\end{aligned}$$

which implies

$$\underbrace{\begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_1\cdot\widehat{\mathbf{B}}\end{bmatrix}}_{\mathbf{B}} = \mathbf{B}_1\cdot\mathbf{P}_r\cdot\mathbf{D}_r\cdot\mathbf{A}_1^{-1}\underbrace{\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_1\cdot\widehat{\mathbf{A}}\end{bmatrix}}_{\mathbf{A}}\begin{bmatrix}\mathbf{D}_r^{-1}\mathbf{P}_r^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c\end{bmatrix}.$$

□

3.2 Representatives for Equivalence Classes

In the previous section, we defined equivalence relations R_F and R'_F. For the purpose of this paper, we are interested in specifying representatives for some (not necessarily all[9]) equivalence classes. As a representative of an equivalence class, we use the *canonical form* of any element in the class. If the representative is not specified for a class, then the canonical form does not exist for its elements. The mapping between any element and the canonical form of the equivalence class of the element can be formalized via a *canonical form function*.

Definition 3 (**Canonical Form Function**). *For an equivalence relation* R *defined on a set* E*, we define a canonical form function as a function* $\mathsf{CF}_\mathsf{R} : E \mapsto \{\{\perp\}\cup E\}$ *satisfying the following properties:*

i) *for any* $\mathbf{A}\in E$, $\mathsf{CF}_\mathsf{R}(\mathbf{A})\neq\perp \implies \mathbf{A}\sim_\mathsf{R}\mathsf{CF}_\mathsf{R}(\mathbf{A})$;
ii) *for any* $\mathbf{A},\mathbf{B}\in E$, $\mathbf{A}\sim_\mathsf{R}\mathbf{B} \implies \mathsf{CF}_\mathsf{R}(\mathbf{A})=\mathsf{CF}_\mathsf{R}(\mathbf{B})$.

We say CF_R *has success probability* γ *if* $\mathsf{CF}_\mathsf{R}(\mathbf{A})\neq\perp$ *with probability* γ*, when* $\mathbf{A}$ *is drawn uniformly at random from* E.

Remark 2. *Note that the biconditional* $\mathbf{A}\sim_\mathsf{R}\mathbf{B} \iff \mathsf{CF}_\mathsf{R}(\mathbf{A})=\mathsf{CF}_\mathsf{R}(\mathbf{B})$ *is hindered by the fact that the canonical form may not be defined for some matrices (i.e., the output of* CF_R *is* $\perp$*), and holds only if* $\gamma=1$ *(i.e., it is defined for all elements in* E*).*

[9] Usually people try to specify a representative for every single equivalence class. For some choices of F, it seems hard to specify a *representative* for every equivalence class of R_F and R'_F while achieving some nice properties we want, such as having an efficient canonical-form-deriving algorithm. Therefore, we relaxed the constraint and simply require that the representative exist for a sufficiently large number of classes.

As we shall see next, these properties are crucial to have a tight correspondence between the code equivalence problem and the problems we rely on in this paper.

Proposition 2. *Assume that* $\mathsf{CF}_{\mathsf{R}_F}$ *is a canonical form function with* $\mathsf{R} = \mathsf{R}_F$ *and* $E = \mathbb{F}_q^{k\times(n-k)}$. *For any* $\mathbf{A} = [\mathbf{A}_1 \mid \mathbf{A}_2] \in \mathrm{GL}_{k,n}$ *where* $\mathbf{A}_1 \in \mathrm{GL}_k$, *we define* $\mathsf{CF}_{\mathsf{R}'_F}(\mathbf{A})$ *as* $\left[\mathbf{I}_k \mid \mathsf{CF}_{\mathsf{R}_F}(\mathbf{A}_1^{-1}\mathbf{A}_2)\right]$ *if* $\mathsf{CF}_{\mathsf{R}_F}(\mathbf{A}_1^{-1}\mathbf{A}_2) \neq \perp$, *or* $\perp$ *otherwise. Then* $\mathsf{CF}_{\mathsf{R}'_F}$ *is a canonical form function with* $\mathsf{R} = \mathsf{R}'_F$ *and* $E = \mathrm{GL}_{k,n}$.

Proof. According to Theorem 1, $\mathsf{CF}_{\mathsf{R}'_F}(\mathbf{A}) \neq \perp$ implies

$$\mathsf{CF}_{\mathsf{R}'_F}(\mathbf{A}) = \left[\mathbf{I}_k \mid \mathsf{CF}_{\mathsf{R}_F}(\mathbf{A}_1^{-1}\mathbf{A}_2)\right] \sim_{\mathsf{R}'_F} [\mathbf{A}_1 \mid \mathbf{A}_2]$$

and

$$\begin{aligned}
[\mathbf{A}_1 \mid \mathbf{A}_2] \sim_{\mathsf{R}'_F} [\mathbf{B}_1 \mid \mathbf{B}_2] &\implies \mathbf{A}_1^{-1}\mathbf{A}_2 \sim_{\mathsf{R}_F} \mathbf{B}_1^{-1}\mathbf{B}_2 \\
&\implies \mathsf{CF}_{\mathsf{R}_F}(\mathbf{A}_1^{-1}\mathbf{A}_2) = \mathsf{CF}_{\mathsf{R}_F}(\mathbf{B}_1^{-1}\mathbf{B}_2) \\
&\implies \mathsf{CF}_{\mathsf{R}'_F}([\mathbf{A}_1 \mid \mathbf{A}_2]) = \mathsf{CF}_{\mathsf{R}'_F}([\mathbf{B}_1 \mid \mathbf{B}_2])
\end{aligned}$$

so the properties in Definition 3 are satisfied. □

In the next section we show practical examples for $\mathsf{CF}_{\mathsf{R}_F}$. This means we will focus on equivalence classes defined on the set $E = \mathbb{F}_q^{k\times(n-k)}$ in the next section. Note that each example of $\mathsf{CF}_{\mathsf{R}_F}$ immediately gives us a canonical form function $\mathsf{CF}_{\mathsf{R}'_F}$, as shown in the proposition above. In the remainder of the paper, we will often omit the subscripts R_F and R'_F, as it will always be clear from the context which equivalence relation we are considering.

Definition 4 (Proper Canonical Form Function). $\mathsf{CF}_{\mathsf{R}_F}$ *(or* $\mathsf{CF}_{\mathsf{R}'_F}$*) is said to be a* proper *canonical form function if the following two properties hold.*

1. *There exists a polynomial-time algorithm that, given any* $\mathbf{A} \in E$, *outputs* $\mathsf{CF}(\mathbf{A})$. *When* $\mathsf{CF}(\mathbf{A}) \neq \perp$, *the algorithm also outputs* $(\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c) \in F$ *mapping* $\mathbf{A}$ *into* $\mathsf{CF}(\mathbf{A})$.
2. $\Pr(\mathsf{CF}(\mathbf{A}) \neq \perp) = 1$ *for* $\mathbf{A}$ *sampled uniform randomly from* E.

It is not hard to see that, when $\tilde{S}_k$ and $\tilde{S}_{n-k}$ are not trivial, having a proper canonical form function $\mathsf{CF}_{\mathsf{R}_F}$ implies that there is a polynomial-time algorithm that determines whether any two bipartite graphs are isomorphic. Unfortunately, the problem is GI-hard, so it would be very surprising if there exists a proper canonical form function in this case. Nevertheless, we show in the next section that canonical form functions that achieve the first property and $\Pr(\mathsf{CF}(\mathbf{A}) \neq \perp) = \Omega(1)$ when q is "not too small" compared to n. The next section also shows canonical form functions for some R_F's where one of $\tilde{S}_k$ and $\tilde{S}_{n-k}$ is trivial, and these canonical form functions are proper.

4 Canonical Form Functions for R_F

This section shows how one can define canonical form functions for R_F for different choices of F. Any canonical function $\mathsf{CF} : \mathbb{F}_q^{k\times(n-k)} \mapsto \left\{\{\perp\} \cup \mathbb{F}_q^{k\times(n-k)}\right\}$ for R_F satisfies the properies in Definition 3 and the first property in Definition 4. For simplicity, we only present algorithms for deriving $\mathsf{CF}(\mathbf{A})$ from $\mathbf{A}$. It is easy to see how these algorithms can be modified to output also $(\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c) \in F$.

We will consider 5 cases for F and, for each of them, provide and analyze examples of canonical form functions. Note that the first 2 canonical form functions have been implicitly introduced in [17], and they are proper canonical form functions. The remaining 3 canonical form functions are entirely new, but they only satisfy the first property of proper canonical form functions. Nevertheless, we show that if $\frac{q}{n(1-R)} > 1/e$, where R is the code rate, the success probability of each of the remaining 3 canonical form functions is $\Omega(1)$. Note that parameter sets proposed in [1] do satisfy $\frac{q}{n(1-R)} > 1/e$.

The work of [17] studied the problem within a specific framework, which required the introduction of a new and ad-hoc problem called Information Set Linear-Equivalence Problem (IS-LEP). Thanks to the use of canonical forms, our treatment is more general as it encompasses all the already existing notions of code equivalences. For the sake of clarity, for the last three cases (i.e. the new ones), we will accompany the description of each case with a pseudocode to compute the corresponding canonical form.

4.1 Case 1: $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$

Let $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$. In this case, one can define the canonical form of any equivalence class as the matrix in which the columns are sorted with respect to $<_{\mathbb{F}_q^k}$. This way, the canonical form is well-defined for every equivalence class, that is, $\gamma = 1$. This canonical form is proper. From any matrix, the canonical form can be derived easily by simply sorting the columns.

4.2 Case 2: $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times S_{n-k}$

Let $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times S_{n-k}$. Here, one can define the canonical form as the unique matrix such that 1) the first non-zero entry of each non-zero column is 1 and 2) the columns are sorted with respect to $<_{\mathbb{F}_q^k}$. Again, this guarantees that the canonical form is well-defined for every equivalence class, so $\gamma = 1$. Given any matrix, to compute the canonical form, one can first scale each non-zero column to turn the first non-zero entry into 1, and then sort the columns.As before, this canonical form function is proper.

4.3 Case 3: $F = \{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$

Let $F = \{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$. In this case, for any equivalence class, we define the canonical form as the unique matrix of which 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. For some equivalence classes, such a matrix might not exist. This happens only when there are two distinct rows that cannot be compared (because they lead to the same multiset). This canonical form function is denoted as $\mathsf{CF}^{(3)}$.

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, during the step of sorting rows. The pseudocode of the algorithm is shown in Algorithm 3. 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.

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

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 $\mathbf{A}' = \mathbf{A}$;
2 Sort the rows in $\mathbf{A}'$ w.r.t. $\prec_{\mathbb{F}_q^{n-k}}$ using a comparison-based sorting algorithm. When comparing two rows $\mathbf{v}, \mathbf{v}'$ in $\mathbf{A}'$, if multiset($\mathbf{v}$) = multiset($\mathbf{v}'$) and $\mathbf{v} \neq \mathbf{v}'$, **return** $\perp$;
3 Sort the columns in $\mathbf{A}'$ w.r.t. $\leq_{\mathbb{F}_q^k}$ using a comparison-based sorting algorithm;
4 **return** $\mathbf{A}'$

An example of how this canonical form is computed is shown in Figure 4.

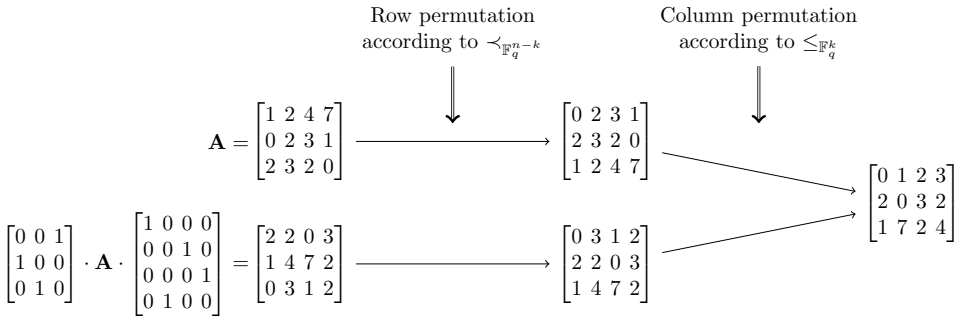

Fig. 4: Example of computation of $\mathsf{CF}^{(3)}$, for $n = 7$ and $k = 3$.

Correctness

Whenever the output is different from $\perp$, the algorithm satisfies the requirements of Definition 3. Indeed, the output matrix $\mathbf{A}'$ is obtained by applying row and column permutations, so $\mathbf{A} \sim_{\mathsf{R}_F} \mathbf{A}'$.

The function is invariant under row and column permutations: if $\mathsf{CF}^{(3)}(\mathbf{A}) = \mathbf{A}' \neq \perp$, then $\mathbf{A}' = \mathsf{CF}^{(3)}(\mathbf{P}_r \cdot \mathbf{A} \cdot \mathbf{P}_c)$, 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 $\mathbf{P}_r$. 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 3 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 $\mathsf{CF}^{(3)}$ (the probability that it does not return $\perp$) is lower bounded by

$$\Gamma_3(q, k, n-k) \coloneqq \prod_{i=1}^{k-1} 1 - \frac{i \cdot m}{q^{n-k}}$$

where

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

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

Theorem 3. *Assume that $k = R \cdot n$ and $q \geq S \cdot n$, where S satifies $S/(1-R) > 1/e$, for all sufficiently large n (which implies that $q = \Omega(n)$). Then, $\Gamma_3 = \Omega(1)$.*

Proof. Below, by $f \gtrsim g$, we mean $g = O(f)$. We define $[f/g]$ as f/g if $f \leq g$, or 1 otherwise. We define $R' \coloneqq 1 - R$.

$$\begin{aligned} \Gamma_3 &\geq \prod_{i=1}^{k-1} 1 - [\frac{i \cdot (n-k)!}{q^{n-k}}] \geq \left(1 - [\frac{k \cdot (n-k)!}{q^{n-k}}]\right)^k \gtrsim \left(1 - [\frac{Rn \cdot (R'n)!}{(Sn)^{R'n}}]\right)^{Rn} \\ &\gtrsim \left(1 - [\frac{Rn \cdot \sqrt{2\pi R'n} \cdot (R'n)^{R'n} \cdot 2}{(e \cdot \frac{S}{R'})^{R'n} \cdot (R'n)^{R'n}}]\right)^{Rn} = \left(1 - [\frac{Rn \cdot \sqrt{2\pi R'n} \cdot 2}{(e \cdot \frac{S}{R'})^{R'n}}]\right)^{Rn} = \Omega(1). \end{aligned}$$

Note that we make use of Stirling's approximation here. □

4.4 Case 4: $F = D_k \times S_k \times \{\mathbf{I}_k\} \times S_{n-k}$

Let $F = D_k \times S_k \times \{\mathbf{I}_k\} \times S_{n-k}$. We define the canonical form of an equivalence class as the matrix (if it exists) such that

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 $\sum_i \mathbf{v}_i = 1$ or $\sum_i \mathbf{v}_i = 0$ and $\sum_i \mathbf{v}_i^{q-2} = 1$.
3. the rows and columns are sorted as in $\mathsf{CF}^{(3)}$.

This canonical form function is denoted as $\mathsf{CF}^{(4)}$.

To derive the canonical form of a matrix, one can carry out one step to ensure that the first two constraints above hold and then another to ensure that the third constraint holds. The second step can be carried out by Algorithm 3. The first step can be carried out as follows. For each row $\mathbf{v} \in \mathbb{F}_q^k$, 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 $(s, s') = (\sum_i \mathbf{v}_i, \sum_i \mathbf{v}_i^{q-2})$. 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 4.

Algorithm 4: $\mathsf{CF}^{(4)}$: Canonical form computation for Case 4.

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 $\mathbf{A}'$;
    If $\mathbf{v} = (0, \ldots, 0)$, continue;
    If $\mathbf{v} = \alpha \cdot (1, \ldots, 1)$ for $\alpha \in \mathbb{F}_q^*$, replace the row by $(1, \ldots, 1)$ and continue;
    Compute $(s, s') = (\sum_\ell \mathbf{v}_\ell, \sum_\ell \mathbf{v}_\ell^{q-2})$;
    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** $\perp$;
return $\mathsf{CF}^{(3)}(\mathbf{A}')$;

Figure 5 shows an example of how the canonical form is computed; in the example, we have $\mathbf{P}_r = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{D}_r = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}$ and $\mathbf{P}_c = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$.

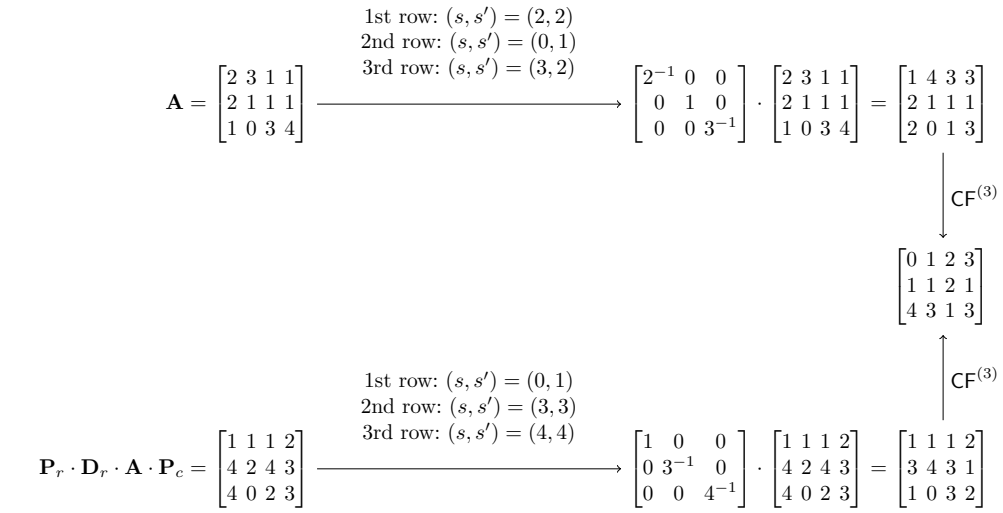

Fig. 5: Example of computation of $\mathsf{CF}^{(4)}$, for $n = 7$ and $k = 3$, over $\mathbb{F}_5$.

Correctness

Let $\mathbf{A} \in \mathbb{F}_q^{k \times (n-k)}$ and $\widetilde{\mathbf{A}} = \mathbf{P}_r \cdot \mathbf{D}_r \cdot \mathbf{A} \cdot \mathbf{P}_c$, with $\mathbf{D}_r \in D_k$, $\mathbf{P}_r \in S_k$ and $\mathbf{P}_c \in S_{n-k}$. We show that 1) $\mathsf{CF}^{(4)}(\mathbf{A}) = \mathsf{CF}^{(4)}(\widetilde{\mathbf{A}})$ for any $\mathbf{D}_r, \mathbf{P}_r$ and $\mathbf{P}_c$, 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 $\widetilde{\mathbf{A}}' = \mathbf{P}_r \cdot \mathbf{A}' \cdot \mathbf{P}_c$: both matrices are in the same equivalence class we have already considered for case 3, thus $\mathsf{CF}^{(3)}(\mathbf{A}') = \mathsf{CF}^{(3)}(\widetilde{\mathbf{A}}')$.

Let $d_i \neq 0$ denote the non null element in the i-th row and i-th column of $\mathbf{D}_r$. 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

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

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

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

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 for some row of $\mathbf{A}$ one has $s = s' = 0$, then there is a row in $\widetilde{\mathbf{A}}$ yielding $\widetilde{s} = \widetilde{s}' = 0$ thus $\mathsf{CF}^{(4)}$ fails for both $\mathbf{A}$ and $\widetilde{\mathbf{A}}$.

Computational Complexity

The loop in Algorithm 4 takes n^2 field operations. In particular, each iteration of Line 6 takes $O(n)$ filed operations, as $x^{q-2} = x^{-1}$ for any $x \in \mathbb{F}_q^*$. Line 8 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 $\mathsf{CF}^{(4)}$ is lower bounded by

$$\Gamma_4(q, k, n-k) \coloneqq \left(1 - \frac{1}{q}\right)^k \cdot \Gamma_3'(q, k, n-k)$$

where $\Gamma_3'(q, k, n-k) \coloneqq \prod_{i=1}^{k-1} 1 - [\frac{i \cdot m \cdot (q-1)}{q^{n-k}}]$. The term $(1 - \frac{1}{q})^k$ is a lower bound on the probability that $\mathbf{v}$ is of the form $(\alpha, \ldots, \alpha)$ or $\sum_i \mathbf{v}_i \neq 0$ or $\sum_i \mathbf{v}_i^{-1} \neq 0$ for every row $\mathbf{v}$. The term $\Gamma_3'(q, k, n-k)$ is a lower bound on the probability that $\mathsf{CF}^{(3)}(\mathbf{A}') \neq \perp$: 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. We estimate the success probability of Algorithm 4 as

$$\gamma_4(q, k, n-k) \coloneqq \left(1 - \frac{1}{q^2}\right)^k \cdot \Gamma_3'(q, k, n-k).$$

Theorem 4. $\Gamma_4(q,k,n-k) = \Omega(1)$ *under the assumptions in Theorem 3.*

Proof.

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

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 $\Gamma'_3(q,k,n-k) = \Omega(1)$, by mimicking how we show $\Gamma_3(q,k,n-k) = \Omega(1)$. □

4.5 Case 5: $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$

Let $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$. The way we define canonical forms in this case is a little less intuitive, so we limit ourselves to describe here just the algorithm for deriving the canonical form from any matrix. Consider an input matrix $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$. For each j such that row j consists of only non-zero elements, we define $\mathbf{A}^{(j)}$ as the result of scaling the columns of $\mathbf{A}$ so that row j of $\mathbf{A}^{(j)}$ becomes $(1,1,\ldots,1)$. If every column contains 0, return $\perp$. Then, for each $\mathbf{A}^{(j)}$, Algorithm 4 is applied to obtain a matrix $\mathbf{A}^{[j]}$ (if $\perp$ is not returned). Finally, return the smallest $\mathbf{A}^{[j]}$ with respect to $<_{\mathbb{F}_q^{k\times(n-k)}}$. Pseudocode of the algorithm is shown in Algorithm 5. The resulting canonical form function is denoted as $\mathsf{CF}^{(5)}$.

Algorithm 5: $\mathsf{CF}^{(5)}$: Canonical form computation for Case 5.

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 $E \leftarrow \emptyset$;
for $j = 1$ **to** k **do**
  $\mathbf{A}^{[j]} \leftarrow \perp$;
  if $(\mathbf{A}_{j,1},\ldots,\mathbf{A}_{j,n-k}) \in (\mathbb{F}_q^*)^{n-k}$ **then**
    Set $\mathbf{A}^{(j)} \leftarrow \mathbf{A}\cdot\mathbf{D}$, where $\mathbf{D} \in D_{n-k}$ and $\mathbf{D}_{i,i} = \mathbf{A}_{j,i}^{-1}$ for $i \in \{1,\ldots,n-k\}$;
    Set $\mathbf{A}^{[j]} \leftarrow \mathsf{CF}^{(4)}(\mathbf{A}^{(j)})$;
  if $\mathbf{A}^{[j]} \neq \perp$ **then**
    $E \leftarrow E \cup \mathbf{A}^{[j]}$;
if $E = \emptyset$ **then**
  return $\perp$;
else
  Compute $\mathbf{C}$ as the smallest element in E w.r.t. $\leq_{\mathbb{F}_q^{k\times(n-k)}}$;
  return $\mathbf{C}$;

Correctness

It is easy to see that Algorithm 5 leads to a canonical form function satisfying the first property in Definition 3. To show that the second property is also satisfied, we introduce two results.

Proposition 5. *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* j *of* $\mathbf{A}$ *(and thus row* j *of* $\mathbf{B}$*) consists of only non-zero elements. Then* $\mathbf{B}^{(j)} = \mathbf{D}'_r \cdot \mathbf{A}^{(j)}$ *for some* $\mathbf{D}'_r \in D_k$.

Proof. Let the elements on the main diagonal of $\mathbf{D}_r$ be $x_1, \dots x_k$. Analogously, let the elements on the main diagonal of $\mathbf{D}_c$ be $y_1, \dots y_{n-k}$. Then, row i of $\mathbf{A}^{(j)}$ is given by $(\mathbf{A}_{i,1} \cdot \mathbf{A}_{j,1}^{-1}, \dots, \mathbf{A}_{i,n-k} \cdot \mathbf{A}_{j,n-k}^{-1})$, while column i of $\mathbf{B}^{(j)}$ is given by

$$\left(\frac{x_i \cdot \mathbf{A}_{i,1} \cdot y_1}{x_j \cdot \mathbf{A}_{j,1} \cdot y_1}, \dots, \frac{x_i \cdot \mathbf{A}_{i,n-k} \cdot y_{n-k}}{x_j \cdot \mathbf{A}_{j,n-k} \cdot y_{n-k}}\right).$$

□

Proposition 6. *Given* $\mathbf{A}, \mathbf{B} \in \mathbb{F}_q^{k\times(n-k)}$ *satisfying* $\mathbf{A} = \mathbf{P}_r^{-1} \cdot \mathbf{B} \cdot \mathbf{P}_c^{-1}$ *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 6 is immediate and therefore omitted in the interest of space. Now, combining the two 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 4.4 to $\mathbf{A}^{(i)}$ and $\mathbf{B}^{(i')}$, we obtain $\mathbf{A}^{[i]} = \mathbf{B}^{[i']}$. Thus, the set of $\mathbf{A}^{[j]}$'s is the same as the set of $\mathbf{B}^{[j]}$'s, 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 5 is dominated by the ones taken by iterations of Line 6. Therefore, the algorithm takes $n \cdot \tilde{O}(n^2) = \tilde{O}(n^3)$ field operations.

Success Probability

The success probability of $\mathsf{CF}^{(5)}$ is lower bounded by

$$\Gamma_5(q, n, k) \coloneqq \left(\frac{q-1}{q}\right)^{n-k} \cdot \Gamma_4(q, k-1, n-k).$$

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. $\Gamma_4(q, k-1, n-k)$ is a lower bound on the probability that $\mathbf{A}^{[1]} \neq \perp$,

under the condition that $\mathbf{A}^{(1)}$ is well-defined. We have $\Gamma_4(q, k-1, n-k)$ instead of $\Gamma_4(q, k, n-k)$, as whether $\mathsf{CF}^{(4)}(\mathbf{A}^{(1)}) = \perp$ depends only on entries not in the first row: The first row of $\mathbf{A}^{(1)}$ is $(1, \ldots, 1)$, so Line 9 of Algorithm 4 will be skipped when $r = 1$; Also, there cannot be a vector $\mathbf{v}$ such that $\mathbf{v} \neq (1, \ldots, 1)$ and $\mathrm{multiset}(\mathbf{v}) = \{1, \ldots, 1\}$.

$\Gamma_5(q, k, n-k)$ is a loose bound, as $\mathbf{A}^{[1]} \neq \perp$ is just a sufficient condition for Algorithm 5 to succeed. Algorithm 5 succeeds as long as there is some j such that $\mathbf{A}^{[j]} \neq \perp$. Therefore, we estimate the success probability of Algorithm 5 as

$$\gamma_5(q, n, k) \coloneqq 1 - (1 - \Gamma_5'(q, n, k))^k$$

where $\Gamma_5'(q, n, k) \coloneqq \left(\frac{q-1}{q}\right)^{n-k} \cdot \gamma_4(q, k-1, n-k)$.

Theorem 7. *$\Gamma_5(q, k, n-k) = \Omega(1)$, under the assumptions in Theorem 3.*

Proof. $(\frac{q-1}{q})^{n-k} = \Omega(1)$ can be proven by mimicking how we show $(1 - \frac{1}{q})^k = \Omega(1)$ in the proof of Theorem 4, and $\Gamma_4(q, k-1, n-k) \geq \Gamma_4(q, k, n-k) = \Omega(1)$. □

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.

5.1 The LESS Sigma Protocol

First we recall, in Figure 6, the Sigma protocol underlying the LESS signature scheme. Note that there are variants with more than 2 generator matrices in the public key. The number of generator matrices in the public key is denoted as s ($s = 2$ in Figure 6).

Private Key: Matrix $\mathbf{Q} \in M_n$
Public Key: Generator Matrices $\mathbf{G}, \mathbf{G}' = \mathsf{RREF}(\mathbf{GQ}) \in \mathbb{F}_q^{k \times n}$

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

Fig. 6: The original LESS Sigma protocol for linear equivalence. To generate the version for permutation equivalence, simply replace M_n by S_n.

As shown in [9], the protocol is *2-special sound*, with soundness error $\varepsilon = \frac{1}{2}$. Note that, if the matrices $\mathbf{Q}$ and $\widetilde{\mathbf{Q}}$ 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 $\mathbf{G} \cdot \widetilde{\mathbf{Q}}$, which is the very same matrix $\widetilde{\mathbf{G}}$ computed by the prover. However, when $\mathsf{ch} = 1$, the verifier computes $\mathbf{G}' \cdot \mathbf{Q}'$, which is equal to $\widetilde{\mathbf{G}}$ up to a change of a basis. To avoid this discrepancy, and enable verification, the protocol computes the RREF, which is used as an invariant under change of basis.Note that, in the case $\mathsf{ch} = 0$, the response consists of the randomly-generated matrix $\widetilde{\mathbf{Q}}$, and can thus be compressed by transmitting only a seed for a secure PRNG, as is common practice.

Remark 3. *The scheme presented in Figure 6 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 [12], and the addition of such standard optimizations does not affect this claim, as shown for instance in [4, 11].*

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. Technically, we let the prover and the verifier end up in two generator matrices which are equal up to an equivalence R'_F; in other words, we allow the verifier to compute a code which is equivalent to the one generated by $\widetilde{\mathbf{G}}$, but not exactly the same. 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

At a high level, canonical forms can be incorporated in the LESS ZK protocol as shown in Figure 7. The idea is that, as in the LESS scheme, the commitment is generated by applying a random transformation τ to $\mathscr{C}$. However, in CF-LESS, the prover commits to the canonical form of the equivalence class $\mathfrak{C}_F\big(\tau(\mathscr{C})\big)$. When the challenge is 0, the prover reveals τ. Instead, when the challenge is 1, the prover reveals a transformation $\pi \neq \tau \circ \mu^{-1}$ such that $\tau(\mathscr{C})$ and $\pi(\mathscr{C}')$ are in the same equivalence class, but are not necessarily equal. In this case, the commitment is verified by checking that it corresponds to the canonical form of $\pi(\mathscr{C}')$. The formal description of the resulting Sigma protocol is given in Figure 8.

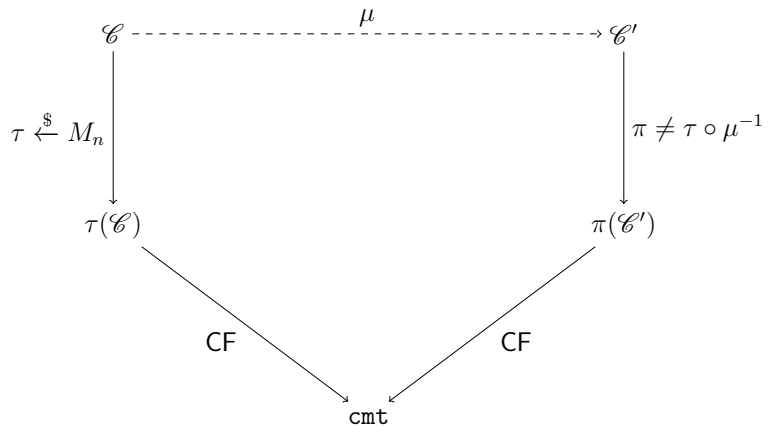

Fig. 7: Visualization of ZK proof of knowledge based on canonical forms. The public key is constituted by the pair $(\mathscr{C}, \mathscr{C}')$, while $\mu \in M_n$ is the secret key.

Private Key: Monomial $\mathbf{Q} \in M_n$
Public Key: Generator Matrices $\mathbf{G}, \mathbf{G}' = \mathsf{RREF}(\mathbf{GQ}) \in \mathbb{F}_q^{k \times n}$

PROVER		VERIFIER
do:		
$\widetilde{\mathbf{Q}} \xleftarrow{\$} M_n$		
$\widetilde{\mathbf{G}}, \mathbf{P} \leftarrow \mathsf{RREF}^*(\mathbf{G} \cdot \widetilde{\mathbf{Q}})$		
while $\mathsf{CF}(\widetilde{\mathbf{G}}) = \perp$		
$\mathsf{cmt} \leftarrow \mathsf{Hash}(\mathsf{CF}(\widetilde{\mathbf{G}}))$	$\xrightarrow{\mathsf{cmt}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \xleftarrow{\$} \{0,1\}$
$\overline{\mathbf{Q}} \leftarrow \widetilde{\mathbf{Q}} \cdot \mathbf{P}$		
If $\mathsf{ch} = 0$:		
$\mathsf{rsp} \leftarrow \widetilde{\mathbf{Q}}$		
Else:		
$\mathsf{rsp} \leftarrow \mathsf{Compress}(\mathbf{Q}' := \mathbf{Q}^{-1}\overline{\mathbf{Q}})$	$\xrightarrow{\mathsf{rsp}}$	**If** $\mathsf{ch} = 0$:
		$\widetilde{\mathbf{G}}, \mathbf{P} \leftarrow \mathsf{RREF}^*(\mathbf{G} \cdot \mathsf{rsp})$
		Verify $\mathsf{Hash}(\mathsf{CF}(\widetilde{\mathbf{G}}))$=cmt
		Else:
		If $\mathbf{G}' \cdot \mathsf{rsp} \notin \mathrm{GL}_{k,n}$:
		Reject
		Verify $\mathsf{Hash}(\mathsf{CF}(\mathbf{G}' \cdot \mathsf{rsp}))$=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 and set $D_k = \{\mathbf{I}_k\}$ and $D_{n-k} = \{\mathbf{I}_{n-k}\}$.

It is easy to see that the protocol is complete if Compress is omitted. The function Compress is used to reduce the size of the response when $\mathsf{ch} = 1$ (and thus the signature size). Before specifying how Compress works, we show a useful decomposition for matrices in S_n and M_n.

Theorem 8. *For every* $\mathbf{P} \in S_n$*, we have*

$$\mathbf{P} = \mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_c \end{bmatrix}$$

for a unique tuple $(\mathbf{P}_{\text{is}}, \mathbf{P}_r, \mathbf{P}_c) \in S_{k,n} \times S_k \times S_{n-k}$.[10]

Theorem 9. *For every* $\mathbf{M} \in M_n$*, we have*

$$\mathbf{M} = \mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{D}_r \cdot \mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c \cdot \mathbf{P}_c \end{bmatrix}$$

for a unique tuple $(\mathbf{P}_{\text{is}}, \mathbf{P}_r, \mathbf{P}_c, \mathbf{D}_r, \mathbf{D}_c) \in S_{k,n} \times S_k \times S_{n-k} \times D_k \times D_{n-k}$.

The proof of the theorem is trivial and is hence omitted. Next, we specify how the function Compress operates.

Definition 5. *Let* $\overline{F} \in \tilde{D}_k \times \tilde{S}_k \times \tilde{D}_{n-k} \times \tilde{S}_{n-k}$ *such that each of its components is trivial if the corresponding component in* F *is non trivial, and vice versa. When linear equivalence is considered, the function* Compress*:* $M_n \mapsto M_n$ *decomposes the input matrix into the form shown in Theorem 9 and replaces* $\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c$*, or* $\mathbf{P}_c$ *by an identity matrix if the corresponding component of* $\overline{F}$ *is trivial. When permutation equivalence is considered, the function* Compress*:* $S_n \mapsto S_n$ *operates in a similar way using the decomposition in Theorem 8 and only replaces* $\mathbf{P}_r$ *or* $\mathbf{P}_c$.

Let us make an example by considering $F = D_k \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$. In this case, $\overline{F} = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times \{\mathbf{I}_{n-k}\}$ and Compress replaces $\mathbf{P}_r$ by $\mathbf{I}_k$ and $\mathbf{D}_c, \mathbf{P}_c$ by $\mathbf{I}_{n-k}$. Therefore, we have

$$\mathsf{Compress}(\mathbf{Q}') = \mathsf{Compress}\left(\mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{D}_r\mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c \end{bmatrix}\right) = \mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c \end{bmatrix}.$$

Without Compress, the response size for $\mathsf{ch} = 1$ would be $n(\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)$ bits. Instead, as it turns out, $\mathsf{Compress}(\mathbf{Q}')$ takes only about $\lceil \log_2 \binom{n}{k} \rceil + (n-k)\log_2(q-1)$ bits. It is not hard to see that for each of the 16 cases of F, Compress reduces response size in a similar way.

Communication Cost

We now consider how the response size for $\mathsf{ch} = 1$ changes for several choices of F. We denote it by isobits and give some meaningful examples in Table 1.
For Case 2, rsp can be written as

$$\mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{D}_r\mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-k} \end{bmatrix} = \mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{P}_r\mathbf{D}'_r & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-k} \end{bmatrix} = \underbrace{\mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-k} \end{bmatrix}}_{\mathbf{P}} \cdot \begin{bmatrix} \mathbf{D}'_r & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{n-k} \end{bmatrix}.$$

[10] The subscript "is" stands for information set.

The matrix $\mathbf{P}$ takes $k \cdot \lceil \log_2(n) \rceil$ bits, while $\mathbf{D}'_r$ takes $k \cdot \lceil \log_2(q-1) \rceil$ bits. Therefore, we have $\mathsf{isobits} = k \cdot (\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)$. This corresponds to the same setting as in [17].

For Case 5 and linear equivalence, $\mathsf{isobits} = \lceil \log_2 \binom{n}{k} \rceil$ as rsp can be represented as an element in $S_{k,n}$. For case 3 we get the same communication cost, if permutation equivalence is considered.

Type	F	**Case**	isobits
Mono	$\{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times S_{n-k}$	2	$k \cdot (\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)$
Mono	$D_k \times S_k \times D_{n-k} \times S_{n-k}$	5	$\lceil \log_2 \binom{n}{k} \rceil \leq n$
Perm	$\{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$	3	$\lceil \log_2 \binom{n}{k} \rceil \leq n$

Table 1: isobits, the number of bits required to represent $\mathsf{Compress}(\mathbf{Q}')$.

Remark 4. *For $k = Rn$, we have*

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

and, in particular, $\log_2 \binom{n}{k} \leq n \cdot h(R)$, with the bound being asymptotically tight since $\log_2 \binom{n}{k} = n \cdot h(R) - c \cdot \log_2(n)$ for some positive constant c. In particular, if $R = \frac{1}{2}$, one has $h(R) = 1$ hence $\log_2 \binom{n}{k}$ is tightly upper bounded by n and asymptotically tends (from below) to n. In such a case we simplify the expression for the communication cost and write $\mathsf{isobits} = n$. This allows to consider a very simple and asymptotically optimal encoding for rsp 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 completeness. For what concerns special soundness, we show that it reduces to finding solutions to the following problem.

Problem 2 (Canonical Forms Code Equivalence). *Let $F \in \tilde{D}_k \times \tilde{S}_k \times \tilde{D}_{n-k} \times \tilde{S}_{n-k}$. Given $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$, find*

$$\mathbf{P}_{\text{is}}, \mathbf{P}'_{\text{is}} \in S_{k,n}, \quad (\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c), (\mathbf{D}'_r, \mathbf{P}'_r, \mathbf{D}'_c, \mathbf{P}'_c) \in \overline{F}$$

such that

$$\mathsf{CF}\left(\mathbf{G} \cdot \mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{D}_r\mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c \end{bmatrix}\right) = \mathsf{CF}\left(\mathbf{G}' \cdot \mathbf{P}'_{\text{is}} \cdot \begin{bmatrix} \mathbf{D}'_r\mathbf{P}'_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}'_c\mathbf{P}'_c \end{bmatrix}\right).$$

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 $\widetilde{\mathbf{Q}}$ is uniformly distributed over M_n. We now show why the protocol is complete. To this end, we consider that when $\mathsf{ch} = 0$, the verifier receives $\widetilde{\mathbf{Q}}$ and executes the very same operations which have been performed by the prover. When $\mathsf{ch} = 1$, let us take $F = D_k \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$ for example. In this case, the verifier receives $\mathsf{Compress}(\mathbf{Q}')$, which is built such that

$$\mathbf{G}'\mathsf{Compress}(\mathbf{Q}') = \mathbf{G}'\mathbf{P}_{\text{is}} \begin{bmatrix} \mathbf{I}_k & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c \end{bmatrix} \sim_{\mathsf{R}'_F} \mathbf{G}'\mathbf{P}_{\text{is}} \begin{bmatrix} \mathbf{D}_r\mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c \end{bmatrix} = \mathbf{G}'\mathbf{Q}'$$
$$\implies \mathsf{CF}(\mathbf{G}' \cdot \mathsf{Compress}(\mathbf{Q}')) = \mathsf{CF}(\mathbf{G}' \cdot \mathbf{Q}').$$

Therefore, the CF-LESS protocol is complete for the specific F. Similarly, the protocol can be shown to be complete for $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times S_{n-k}$ [11] and $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$. It is also easy to see that the protocol is zero-knowledge and complete for the special case of permutations, when $F = \{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$.

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

Proof. We consider two accepting transcripts $\widetilde{\mathbf{Q}}$ and $\mathbf{Q}'$, respectively for $\mathsf{ch} = 0$ and $\mathsf{ch} = 1$, and commitment cmt. From the knowledge of $\widetilde{\mathbf{Q}}$, one can obtain $\mathbf{P}$ and compute $\overline{\mathbf{Q}} = \widetilde{\mathbf{Q}} \cdot \mathbf{P}$. This implies that

$$\mathsf{Hash}\left(\mathsf{CF}(\mathbf{G} \cdot \overline{\mathbf{Q}})\right) = \mathsf{Hash}\big(\mathsf{CF}(\mathbf{G}' \cdot \mathbf{Q}')\big).$$

Either $\mathsf{CF}(\mathbf{G} \cdot \overline{\mathbf{Q}}) \neq \mathsf{CF}(\mathbf{G}' \cdot \mathbf{Q}')$ and a hash collision has been found, or $\mathsf{CF}(\mathbf{G} \cdot \overline{\mathbf{Q}}) = \mathsf{CF}(\mathbf{G}' \cdot \mathbf{Q}')$. Note that $\mathbf{Q}'$ is the output of $\mathsf{Compress}$, so it is constituted by $\mathbf{P}'_{\text{is}} \in S_{k,n}$ and $(\mathbf{D}'_r, \mathbf{P}'_r, \mathbf{D}'_c, \mathbf{P}'_c) \in \overline{F}$. One can compute $\mathsf{Compress}$, on input $\overline{\mathbf{Q}}$, and find $\mathbf{P}_{\text{is}} \in S_{k,n}$ and $(\mathbf{D}_r, \mathbf{P}_r, \mathbf{D}_c, \mathbf{P}_c) \in \overline{F}$ such that

$$\mathsf{CF}(\mathbf{G} \cdot \overline{\mathbf{Q}}) = \mathsf{CF}\left(\mathbf{G} \cdot \mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \mathbf{D}_r\mathbf{P}_r & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c \end{bmatrix}\right).$$

[11] The papers [1, 4] make use of the same technique for $F = \{\mathbf{I}_k\} \times \{\mathbf{I}_k\} \times D_{n-k} \times S_{n-k}$.

Thus, we have obtained

$$\mathsf{CF}\left(\mathbf{G}\cdot\mathbf{P}_{\text{is}}\cdot\begin{bmatrix}\mathbf{D}_r\mathbf{P}_r & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_c\mathbf{P}_c\end{bmatrix}\right)=\mathsf{CF}\left(\mathbf{G}'\cdot\mathbf{P}'_{\text{is}}\cdot\begin{bmatrix}\mathbf{D}'_r\mathbf{P}'_r & \mathbf{0}\\ \mathbf{0} & \mathbf{D}'_c\mathbf{P}'_c\end{bmatrix}\right)$$

which implies that we have found a solution for CF-code equivalence, on input $\mathbf{G}\cdot\overline{\mathbf{Q}}$ and $\mathbf{G}'$. Starting from this solution, one can easily recover the equivalence between $\mathbf{G}$ and $\mathbf{G}'$.

Note that the proof depends on the possibility to compute efficiently, from any $\mathbf{A}\in\mathbb{F}_q^{k\times n}$ such that the first k columns form an information set, $\mathsf{CF}(\mathbf{A})$ along $(\mathbf{P}_r,\mathbf{D}_r,\mathbf{P}_c,\mathbf{D}_c)\in F$ mapping $\mathbf{A}$ into $\mathsf{CF}(\mathbf{A})$. As we have already specified (see Section 4), the canonical form functions we consider satisfy this requirement. So, if computing CF takes polynomial time, then all the operations we have considered in this proof take polynomial time, as well. □

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, and recalling Definition 3, we have that the average number of matrices $\widetilde{\mathbf{Q}}$ one 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 $\widetilde{\mathbf{Q}}$, $\widetilde{\mathbf{G}}$ behaves as a uniformly random matrix. For each $\widetilde{\mathbf{Q}}$ the prover executes RREF^* and then computes CF. 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 $\frac{1}{\gamma}\approx 1$: 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 $O(n^3)$ 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 $\tilde{O}(n^3)$ operations on average.

6 Hardness Analysis and Implications

In this section we provide strong evidence that the new formulation of code equivalence, using canonical forms, still leads to a hard problem. Namely, we prove that there exists a polynomial-time reduction from code equivalence to Problem 2, given that canonical forms can be computed in polynomial time. If the considered canonical forms have $\gamma = 1$ (i.e., they never fail), we obtain that the reduction works for every code equivalence instance. When instead $\gamma < 1$ we prove that, under Heuristic 1, the reduction is successful for all but a negligible portion of codes 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.

The reduction in the other way is trivial (we briefly sketch it, even though it has already been shown implicitly in the protocol of Figure 8). In practice, this means that the code equivalence problem with canonical forms is as hard as in its traditional formulation.

We furthermore show that the reduction 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 [18] and [2], or the minimum distance [6,15]. Moreover, regardless of the considered type of equivalence (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, 15]: when q increases, the minimum distance of random codes increases as well, so these attacks become slower.

To be as general as possible, in all the reductions we consider the LEP version of code equivalence and the case in which $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$, which is the most general case. All the other cases (e.g., PEP and $F = \{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$) follow as special cases.

6.1 Reductions between LEP and CF-LEP

Showing that CF-LEP reduces to LEP is trivial and has already been done, implicitly, in the description of the CF-LESS Sigma protocol. Namely, for a pair $\mathbf{G}, \mathbf{G}'$ for which a solution $\mathbf{Q} = \mathbf{PD}$ is known, it would be enough to continue sampling matrices $\mathbf{P}_{\text{is}} \in S_{k,n}$ until $\mathsf{CF}(\mathbf{G} \cdot \mathbf{P}_{\text{is}}) \neq \perp$. Then, from the knowledge of $\mathbf{P}$, one can easily find $\mathbf{P}'_{\text{is}}$ such that $\mathsf{CF}(\mathbf{G} \cdot \mathbf{P}_{\text{is}}) = \mathsf{CF}(\mathbf{G}' \cdot \mathbf{P}'_{\text{is}})$.

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 6;

its correctness, together with the probability that the reduction succeeds, is detailed in the next Proposition.

Algorithm 6: Building LEP solution from CF-LEP solution

Data: $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$, canonical form function $\mathsf{CF} : \mathbb{F}_q^{k\times(n-k)} \mapsto \left\{\{\perp\} \cup \mathbb{F}_q^{k\times(n-k)}\right\}$
Input: matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$, solution $\mathbf{P}_{\text{is}}, \mathbf{P}'_{\text{is}}$ for IS-LEP
Output: solution $\mathbf{S} \in \mathrm{GL}_k$, $\mathbf{Q} \in M_n$ for LEP

1 Compute $\mathsf{RREF}(\mathbf{G} \cdot \mathbf{P}_{\text{is}}) = \left[\mathbf{I}_k \mid \mathbf{A}\right] = \mathbf{U} \cdot \mathbf{G} \cdot \mathbf{P}_{\text{is}}$;
2 Compute $\mathsf{RREF}(\mathbf{G}' \cdot \mathbf{P}'_{\text{is}}) = \left[\mathbf{I}_k \mid \mathbf{A}'\right] = \mathbf{U}' \cdot \mathbf{G}' \cdot \mathbf{P}'_{\text{is}}$;
3 Compute $\mathbf{B} = \mathsf{CF}(\mathbf{A}) = \mathbf{P}_r \cdot \mathbf{D}_r \cdot \mathbf{A} \cdot \mathbf{P}_c \cdot \mathbf{D}_c$;
4 Compute $\mathbf{B}' = \mathsf{CF}(\mathbf{A}') = \mathbf{P}'_r \cdot \mathbf{D}'_r \cdot \mathbf{A}' \cdot \mathbf{P}'_c \cdot \mathbf{D}'_c$;
5 Set $\widetilde{\mathbf{P}}_r \cdot \widetilde{\mathbf{D}}_r = (\mathbf{P}'_r \cdot \mathbf{D}'_r)^{-1} \cdot \mathbf{P}_r \cdot \mathbf{D}_r$;
6 Set $\widetilde{\mathbf{P}}_c \cdot \widetilde{\mathbf{D}}_c = \mathbf{P}_c \cdot \mathbf{D}_c \cdot (\mathbf{P}'_c \cdot \mathbf{D}'_c)^{-1}$;
7 Compute $\mathbf{Q} = \mathbf{P}_{\text{is}} \cdot \begin{bmatrix} \widetilde{\mathbf{P}}_r \cdot \widetilde{\mathbf{D}}_r & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{P}}_c \cdot \widetilde{\mathbf{D}}_c \end{bmatrix} \cdot \mathbf{P}'^{-1}_{\text{is}}$;
8 Compute $\mathbf{S} = \mathbf{U}'^{-1} \cdot (\widetilde{\mathbf{P}}_r \cdot \widetilde{\mathbf{D}}_r)^{-1} \cdot \mathbf{U}$;
9 **return** $\mathbf{S}, \mathbf{Q}$

Proposition 11. *Let $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$ and CF be a canonical form for F, computable in polynomial time. If $(\mathbf{G}, \mathbf{G}')$ admits a solution for CF-LEP, then a solution for LEP, on input $(\mathbf{G}, \mathbf{G}')$,can be found in polynomial time.*

Proof. Let us denote by $\mathbf{P}_{\text{is}}, \mathbf{P}'_{\text{is}} \in S_{k,n}$ the solution for the CF-LEP instance $(\mathbf{G}, \mathbf{G}')$. Now, consider Algorithm 6, on input $\mathbf{G}, \mathbf{G}'$ and $\mathbf{P}_{\text{is}}, \mathbf{P}_{\text{is}}$. The algorithm obviously takes polynomial time, since all it does is computing canonical forms (which takes polynomial time by hypothesis) and matrix multiplications/inversions. We only need to show it is correct, i.e., that the returned $\mathbf{S}$, $\mathbf{Q}$ indeed solves LEP.

Since $\mathbf{P}_{\text{is}}, \mathbf{P}'_{\text{is}}$ is a solution for CF-LEP, this means that $\mathsf{CF}(\mathbf{G} \cdot \mathbf{P}_{\text{is}}) = \mathsf{CF}(\mathbf{G} \cdot \mathbf{P}'_{\text{is}})$. Recalling how canonical forms are defined, this is equivalent to first computing RREF and then computing CF on the non systematic part. Let $\mathbf{U}$ be the matrix that brings $\mathbf{G}\cdot\mathbf{P}_{\text{is}}$ into RREF form, that is, $\mathbf{U}\cdot\mathbf{G}\cdot\mathbf{P}_{\text{is}} = [\mathbf{I}_k \mid \mathbf{A}]$. So $\mathsf{CF}\big(\mathbf{G}\cdot\mathbf{P}_{\text{is}}\big) = \mathsf{CF}([\mathbf{I}_k \mid \mathsf{CF}(\mathbf{A})]\big)$ and, by definition of canonical forms,

$$\mathsf{CF}(\mathbf{A}) = \mathbf{P}_r \cdot \mathbf{D}_r \cdot \mathbf{A} \cdot \mathbf{P}_c \cdot \mathbf{D}_c.$$

We use analogous notation for $\mathbf{G}'$ (note that we are following the notation of Algorithm 6), and get $\mathbf{U}' \cdot \mathbf{G}' \cdot \mathbf{P}'_{\text{is}} = (\mathbf{I}_k \mid \mathbf{A}')$ and

$$\mathsf{CF}(\mathbf{A}') = \mathbf{P}'_r \cdot \mathbf{D}'_r \cdot \mathbf{A}' \cdot \mathbf{P}'_c \cdot \mathbf{D}'_c.$$

Since $\mathsf{CF}(\mathbf{G}\cdot\mathbf{P}_{\text{is}}) = \mathsf{CF}(\mathbf{G}'\cdot\mathbf{P}'_{\text{is}})$, we have $\mathsf{CF}(\mathbf{A}) = \mathsf{CF}(\mathbf{A}')$, i.e.,

$$\mathbf{A}' = \underbrace{(\mathbf{P}'_r\cdot\mathbf{D}'_r)^{-1}\cdot\mathbf{P}_r\cdot\mathbf{D}_r}_{\widetilde{\mathbf{P}}_r\cdot\widetilde{\mathbf{D}}_r}\cdot\mathbf{A}\cdot\underbrace{\mathbf{P}_r\cdot\mathbf{D}_r\cdot(\mathbf{P}'_c\cdot\mathbf{D}'_c)^{-1}}_{\widetilde{\mathbf{P}}_c\cdot\widetilde{\mathbf{D}}_c}.$$

So, we can write

$$\begin{aligned}\mathbf{U}'\cdot\mathbf{G}'\cdot\mathbf{P}'_{\text{is}} = [\mathbf{I}_k \mid \mathbf{A}'] &= (\widetilde{\mathbf{P}}_r\cdot\widetilde{\mathbf{D}}_r)^{-1}\cdot\underbrace{[\mathbf{I}_k \mid \mathbf{A}]}_{\mathbf{U}\cdot\mathbf{G}\cdot\mathbf{P}_{\text{is}}}\cdot\begin{bmatrix}\widetilde{\mathbf{P}}_r\widetilde{\mathbf{D}}_r & \mathbf{0}\\ \mathbf{0} & \widetilde{\mathbf{P}}_c\widetilde{\mathbf{D}}_c\end{bmatrix}\\ &= (\widetilde{\mathbf{P}}_r\cdot\widetilde{\mathbf{D}}_r)^{-1}\cdot\mathbf{U}\cdot\mathbf{G}\cdot\mathbf{P}_{\text{is}}\cdot\begin{bmatrix}\widetilde{\mathbf{P}}_r\widetilde{\mathbf{D}}_r & \mathbf{0}\\ \mathbf{0} & \widetilde{\mathbf{P}}_c\widetilde{\mathbf{D}}_c\end{bmatrix}.\end{aligned}$$

From the above, with simple algebraic manipulations, we obtain the desired expressions for $\mathbf{S}$ and $\mathbf{Q}$. □

We proceed by analyzing how many LEP instances admit also a solution for CF-LEP. While the case $\gamma = 1$ is trivial, the case $\gamma < 1$ requires some attention. Still, we are able to show that, unless γ is negligible, an arbitrary LEP instance admits at least a solution for CF-LEP with overwhelming probability.

Case $\gamma = 1$

We consider that an arbitrary LEP instance $(\mathbf{G}, \mathbf{G}')$ admits a solution for CF-LEP in case there exists at least one matrix $\mathbf{P}_{\text{is}} \in S_{k,n}$ for which $\mathsf{CF}(\mathbf{G}\cdot\mathbf{P}_{\text{is}}) \neq \perp$. This corresponds to the complementary of the probability that, for all possible RREF forms for $\mathbf{G}$ and $\mathbf{G}'$, the canonical form is not defined. Whenever $\gamma = 1$, then canonical forms can always be defined and the above reduction applies to any LEP instance.

Case $\gamma < 1$

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 [13, 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 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

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

So, the probability that, for a random code, canonical forms cannot be defined is less than

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

It is easy to see that it is always negligible, unless γ is negligible, as well.

Remark 5. *For CF-LESS, we consider instances having rate* $1/2$ *and canonical forms that succeeds with probability at least* $1 - 2^{20}$ *(which we verified experimentally). In such a case, the coefficient* ψ *is well approximated by* $-\frac{1}{\ln(2)}2^n$*: the probability that the reduction does not apply is negligible.*

Remark 6. *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.*

6.2 Canonical Forms as a Solver for LEP

We now show how the reduction in Algorithm 6 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 7. Essentially, the procedure first solves CF-LEP using a meet-in-the-middle strategy; then, it calls the reduction in Algorithm 7 to reconstruct the equivalence between $\mathbf{G}$ and $\mathbf{G}'$.

Algorithm 7: 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 $\mathbf{P}_{\text{is}} \xleftarrow{\$} S_{k,n}$;
  if $\mathbf{G} \cdot \mathbf{P}_{\text{is}} \in \mathrm{GL}_{k,n}$ **then**
    Compute $\mathbf{B} = \mathsf{CF}(\mathbf{G} \cdot \mathbf{P}_{\text{is}})$;
    if $\mathbf{B} \neq \perp$ **then**
      Add $(\mathbf{P}_{\text{is}}, \mathbf{B})$ to $\mathcal{L}$;
// Populate second list
while $|\mathcal{L}'| < m$ **do**
  Sample $\mathbf{P}'_{\text{is}} \xleftarrow{\$} S_{k,n}$;
  if $\mathbf{G}' \cdot \mathbf{P}'_{\text{is}} \in \mathrm{GL}_{k,n}$ **then**
    Compute $\mathbf{B}' = \mathsf{CF}(\mathbf{G}' \cdot \mathbf{P}'_{\text{is}})$;
    if $\mathbf{B}' \neq \perp$ **then**
      Add $(\mathbf{P}'_{\text{is}}, \mathbf{B}')$ to $\mathcal{L}'$;
// Find solution for CF-LEP, then reconstruct the equivalence
Search for collisions, i.e., pairs $(\mathbf{P}_{\text{is}}, \mathbf{B}) \in \mathcal{L}$, $(\mathbf{P}'_{\text{is}}, \mathbf{B}') \in \mathcal{L}'$ such that $\mathbf{B} = \mathbf{B}'$;
If a collision is found, call Algorithm 6 on input $\mathbf{G}, \mathbf{G}'$ and $\mathbf{P}_{\text{is}}$, $\mathbf{P}'_{\text{is}}$

The analysis of the resulting time complexity is very simple. First, because of the birthday paradox, the algorithm has constant success probability which is approximately 1/2. For each candidate $\mathbf{P}_{\text{is}}$ (resp., $\mathbf{P}'_{\text{is}}$), the probability that it corresponds to a computable canonical form is $\zeta\gamma$. Thus, the number of different distinct $\mathbf{P}_{\text{is}}$ (and $\mathbf{P}'_{\text{is}}$) 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 $\mathbf{P}_{\text{is}}$ (and $\mathbf{P}'_{\text{is}}$) 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

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

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 $2^{n \cdot h(R) \cdot \left(1+o(1)\right)}$.

Appendix A briefly compares the cost of Algorithm 7 with other known attacks. For solvers based on short codewords [5,6,15], 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 6 becomes faster than all the other state-of-the-art attacks.

Remark 7. *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 $F = \{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$ and permutation equivalence is considered, or $F = D_k \times S_k \times D_{n-k} \times S_{n-k}$ and linear equivalence is considered. Indeed, to guarantee that the collision attack is slow enough, 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 we recommend for CF-LESS. Indeed, their 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 2 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 6.2. 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	4.2	13939	8624	0
	Mono								4.5		2481	$\approx 10^{-24}$
	Perm								4.3		2481	$\approx 10^{-49}$
3	Mono	400	200	127	2	759	33	197.67	4.2	35074	17208	0
	Mono								4.5		5658	$\approx 10^{-19}$
	Perm								4.3		5658	$\approx 10^{-63}$
5	Mono	548	274	127	2	1352	40	271.56	4.2	65792	30586	0
	Mono								4.5		10036	$\approx 10^{-14}$
	Perm								4.3		10036	$\approx 10^{-69}$

Table 2: 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 4.3 and 4.5 are derived using Equation 4 and 13, respectively. Note that the numbers for Case 4.3 are actually proven upper bounds on the failure rates.

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

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

The value of $\mathsf{isobits}$ has been shown in Table 1. 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 [10,14]. $\ell_{\text{tree\_seed}}$, 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 3.

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	4.2	41785	5941	0
	Mono								4.5		1846	$\approx 10^{-24}$
	Perm								4.3		1846	$\approx 10^{-49}$
3	Mono	400	200	127	4	895	24	197.67	4.2	105174	12768	0
	Mono								4.5		4368	$\approx 10^{-19}$
	Perm								4.3		4368	$\approx 10^{-63}$
5	Mono	548	274	127	4	907	34	271.56	4.2	197312	25237	0
	Mono								4.5		7769	$\approx 10^{-14}$
	Perm								4.3		7769	$\approx 10^{-69}$

Table 3: 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 4 for instance the results concerning the ring signature scheme presented in [3]; as above, the original 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.

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	- 4.3	10761 4522	0 $\approx 10^{-49}$
									2^6	- 4.3	13737 7498	0 $\approx 10^{-49}$
									2^{12}	- 4.3	19689 13450	0 $\approx 10^{-49}$
									2^{21}	- 4.3	28617 22378	0 $\approx 10^{-49}$

Table 4: Parameter sets for ring signatures using canonical forms, and resulting sizes in bytes (B).

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.

References

[1] Baldi, M., Barengh, 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] Feo, L.D., 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

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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

[17] Persichetti, E., Santini, P.: A new formulation of the linear equivalence problem and shorter LESS signatures (2023), https://eprint.iacr.org/2023/847

[18] 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
[19] 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 6.2.

SSA, [18]: 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 [19], 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*, [5,6,15]: 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[12][13] Hence, for these algorithms we consider a time complexity given by

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

[12] 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.
[13] 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.

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 7; 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.

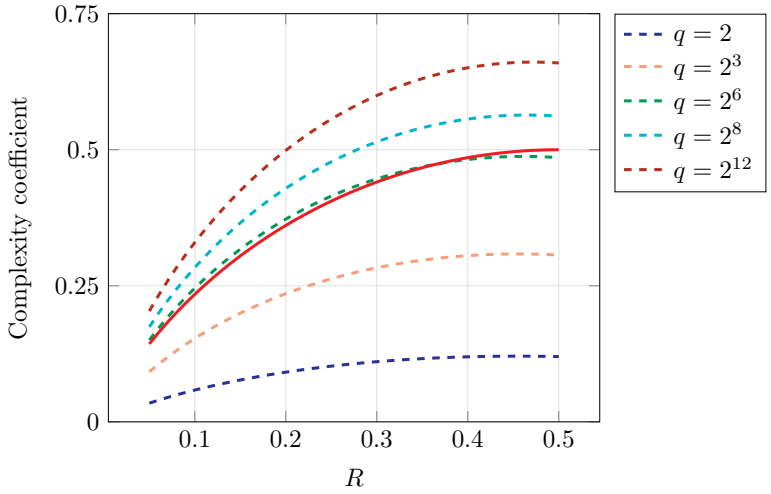

Fig. 9: Comparison between the complexity coefficients for Prange (dashed lines) and Algorithm 7 (continuous red line), as a function of the code rate.

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

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

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, which is defined for $F = \{\mathbf{I}_k\} \times S_k \times \{\mathbf{I}_{n-k}\} \times S_{n-k}$.

Proposition 12. *For $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$ chosen uniformly at random, the canonical form defined as in Section 4, Case 3 exists with probability at least $\prod_{i=1}^{k-1} 1 - \frac{im}{q^{n-k}}$, where*

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

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 $\mathbf{a}_i$. Note that $\mathcal{S}(\mathbf{a}_i)$ contains all vectors that one can obtain by permuting the entries of $\mathbf{a}_i$. The probability that computational of canonical form does not fail can be lower bounded with a simple iterative reasoning.

Let us consider the first two rows of $\mathbf{A}$: regardless of $\mathbf{a}_1$, they will have different multisets if $\mathsf{multiset}(\mathbf{a}_2) \neq \mathsf{multiset}(\mathbf{a}_1)$. So, the probability that this pair of rows is valid is

$$\begin{aligned} \Pr\left[\{\mathbf{a}_1, \mathbf{a}_2\} \text{ is valid}\right] &= \sum_{\mathbf{a}_i \in \mathbb{F}_q^n} \Pr\left[\mathbf{a}_2 \text{ is valid} \mid \mathbf{a}_1\right] \cdot \Pr\left[\mathbf{a}_1\right] \\ &= \frac{1}{q^{n-k}} \sum_{\mathbf{a}_i \in \mathbb{F}_q^n} \left(1 - \frac{|\mathcal{S}(a_1)|}{q^{n-k}}\right) \end{aligned}$$

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 13. *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

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