A New Formulation of the Linear Equivalence Problem and Shorter LESS Signatures

Edoardo Persichetti[1] , Paolo Santini[2]

[1]Florida Atlantic University, Sapienza University of Rome
[2]Marche Polytechnic University

Abstract. The Linear Equivalence Problem (LEP) asks to find a linear isometry between a given pair of linear codes; in the Hamming weight this is known as a *monomial map*. LEP has been used in cryptography to design the family of LESS signatures, which includes also some advanced schemes, such as ring and identity-based signatures. All of these schemes are obtained applying the Fiat-Shamir transformation to a Sigma protocol, in which the prover's responses contain a description of how the monomial map acts on all code coordinates; such a description constitutes the vast majority of the signature size. In this paper, we propose a new formulation of LEP, which we refer to as Information-Set (IS)-LEP. Exploiting IS-LEP, it is enough for the prover to provide the description of the monomial action only on an information set, instead of all the coordinates. Thanks to this new formulation, we are able to drastically reduce signature sizes for all LESS signature schemes, without any relevant computational overhead. We prove that IS-LEP and LEP are completely equivalent (indeed, the same problem), which means that improvement comes with no additional security assumption, either.

1 Introduction

The Code Equivalence Problem (CEP) is a traditional problem of coding theory, which asks to determine whether two given linear codes are equivalent to each other. For the canonical (and most studied) case of isometries in the Hamming metric, the notion of equivalence is linked to the existence of a generalized permutation (i.e. with non-unitary scaling factors), also known as *monomial* transformation. In such a setting, the problem is normally referred to as the Linear Equivalence Problem (LEP).

The computational version of LEP, which is of interest in cryptography, may appear to be somewhat less secure than other problems from coding theory such as the well-known Syndrome Decoding Problem (SDP); unlike SDP, in fact, LEP is probably not NP-hard, since this would imply the collapse of the polynomial hierarchy [PR97]. Nevertheless, perhaps surprisingly, the best known algorithms for LEP (at least, for the regime of interest) utilize an SDP solver as a subroutine. Moreover, the application of an isometry to a linear code can be described as a (non-commutative) group action with certain nice properties, which is exactly the key for its use in cryptographic applications.

1.1 Related Works

The first cryptosystem built on LEP was presented in 2020 as LESS, acronym for Linear Equivalence Signature Scheme [BMPS20]. The paper describes a simple 3-pass Zero-Knowledge Identification (ZK-ID) protocol, following in the footsteps of [GMW19], and then shows how this can be transformed into a full-fledged signature scheme via Fiat-Shamir. It is worth noting that LESS is part of a collection of schemes leveraging this framework, relying on tools from a wide variety of setting, including polynomials [Pat96], isogenies [FG19], lattices [DvW22], matrix codes [CNP+23], trilinear forms [TDJ+22,DG22] etc.

In a follow-up work [BBPS21], the authors refine the scheme using some familiar protocol-level techniques such as the use of multiple keys (to amplify soundness) and fixed-weight challenge strings (to reduce signature length), as seen for instance in [BKV19]; the work also features new parameters, adjusted to withstand a novel LEP solver introduced by Beullens [Beu21]. In fact, a comprehensive study of solvers for LEP was subsequently put together in [BBPS23], with the aim of presenting a clear picture of the best attack techniques, and a tool for selecting secure parameters. The group action structure connected to LEP proved to be appealing as a potential building block in many constructions, developed in ensuing works: some successfully, such as the ring and identity-based signatures proposed in [BBN+22], some unsuccessfully (e.g. [ZZ21,PRS22]).

The essential structure of the LESS protocol is as follows. Starting from a public code $\mathscr{C}$, the prover generates their public key as $\mathscr{C}' = \mu(\mathscr{C})$ where μ is a linear isometry. Then, the protocol goes as in Figure 1.

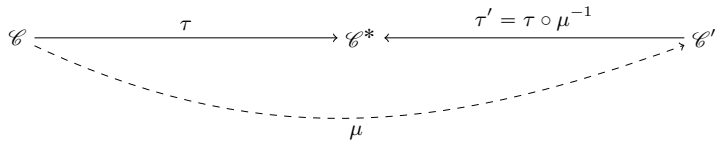


Fig. 1: Representation of the proof of knowledge structure in LESS

In each execution of the protocol, the prover samples an ephemeral map τ and commits to $\mathscr{C}^* = \tau(\mathscr{C})$. The verifier then asks to disclose one of the following two maps: the one *on the left* between $\mathscr{C}$ and $\mathscr{C}^*$, or the one *on the right* between $\mathscr{C}^*$ and $\mathscr{C}'$. The honest prover is always able to provide both maps, i.e., can always construct a graph like the one in Figure 1. A cheating prover, instead, can only craft one of the two maps at a given time and try to guess which one is going to be asked; he cannot, however, reproduce the full graph, without knowing the secret key (which requires to solve a LEP instance). This informal argument of witness extractability intuitively leads to a soundness error of 1/2; to achieve λ bits of security, it is then necessary to utilize standard error amplification techniques such as parallel repetitions.

Linear codes are customarily represented through their generator matrices, whereas isometries consist of column transformations together with a change of basis. This means that, in practice, $\mathscr{C}'$ is represented as $\mathbf{S}\mu(\mathbf{G})$, where $\mathbf{G} \in \mathbb{F}_q^{k\times n}$ is a generator matrix for $\mathscr{C}$ and $\mathbf{S}$ is non-singular of size k. To check that two codes are equal, one can compute a special generator matrix, say, the one in systematic form, which can be naturally obtained with one Gaussian elimination. This allows to greatly reduce the communication cost, because the prover can commit to the hash of the systematic generator of $\mathscr{C}^*$: the verifier will recompute such a matrix, hash it, and check consistence with the commitment. Without this consideration, the LESS scheme would not be practical, since the size of commitments would be gigantic.

Two meaningful improvements appeared in [BBPS21]. The first one consists in allowing for more than two equivalent codes in the public key, which allows to enrich the graph in Figure 1 with some additional maps on the right. The corresponding graph is reported in Figure 2; in the figure, we are denoting $\tau'_i = \tau \circ \mu_i^{-1}$ and are using s for the number of codes.

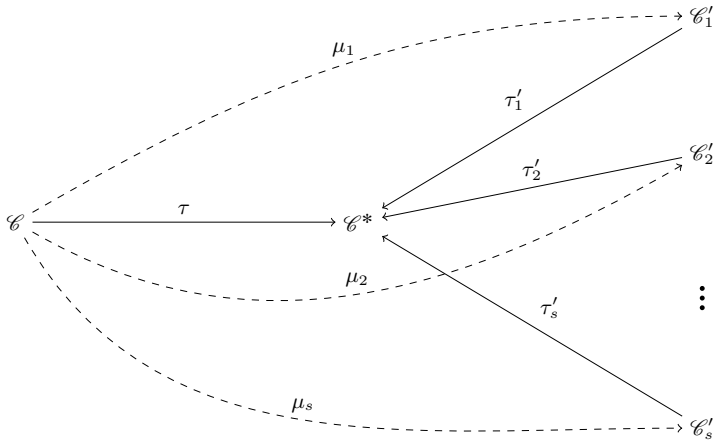


Fig. 2: The LESS-FM proof of knowledge with multiple keys

With this variant, the verifier will choose either the map on the left or one of the $s-1$ maps on the right. It is easy to see that an adversary can reply correctly only by guessing, in advance, which instance will be selected by the prover. This leads to an amplified soundness error of $\frac{1}{s}$ and, consequently, only $\frac{\lambda}{\log_2(s)}$ repetitions are required. With respect to the LESS scheme, this leads to an improvement for what concerns both the signature size and the computational overhead on the verifier's side, since the number of parallel repetitions is reduced by a factor $\log_2(s)$. Obviously, the price to pay is a steep increase in public key size.

The second optimization introduced in LESS-FM consists of using challenges with a non-uniform distribution, so that the map on the left is the one queried most frequently. This is because such a map, being entirely random, can be represented compactly by the seed used to generate it. To preserve the soundness error, which now behaves like a binomial coefficient, one needs to increase the number of overall repetitions; however, the number of maps on the right which are verified (which cannot be compressed with seeds) is much smaller. This yields a significant reduction in signature size, which is further improved by utilizing a seed tree [BKP20] to efficiently transmit the seeds.

1.2 Our Contributions

In this work, we describe a new technique which greatly improves the performance of the scheme. Unlike the ones described in LESS-FM, which are somewhat standard techniques applicable to any Sigma protocol with the same structure, our improvement is specific to the LEP setting.

A new method for verification. In a nutshell, our technique consists of a compact way to verify the maps of the graph in Figure 2. The main idea is based on the following key observation: if two linear codes are linearly equivalent, once two information sets are mapped to each other, the remaining coordinates are identical up to a linear isometry (i.e. a monomial), whose existence can be efficiently checked. This means that the prover does not need to include the entire map in his response, since a description of how the map acts on an information set would be enough. This brings to a direct improvement in the signature size of LESS signature schemes: instead of $n \log_2(n) + n \log_2(q-1)$ bits, the binary size for equivalences on the right is reduced to only $k \log_2(n) + k \log_2(q-1)$ bits. This reduces the size of responses by a factor equal to the code rate k/n: since the codes employed in LESS have all rate $\approx 1/2$, we essentially halve the communication cost (factoring out the cost of the small overhead due to seeds, salts and other minor items).

A new notion of equivalence. Our improvement comes with several technical caveats. The main concern is that we have to make sure that this novel way to verify that two codes are indeed equivalent, does not introduce vulnerabilities. To do this, we introduce a new notion of equivalence, which we call Information Set (IS) - linear equivalence, to emphasize that the focus is on how the linear map acts on an information set. We then show that the associated decisional problem, which we call IS-LEP, is literally the same as LEP: any solver for IS-LEP can in fact be used to solve LEP, and viceversa. Formally, what we prove is something stronger, namely that any "YES" (resp. "NO") instance for LEP is also a "YES" (resp. "NO") instance for IS-LEP: this implies that IS-LEP and LEP are actually the same problem. The definition of IS-LEP is the focus of section 4.

Application to proofs of knowledge. In Section 5 we deal with the practical problem of embedding the verification of IS-LEP into proof-of-knowledge protocols as in Figures 1 and 2. Indeed, unlike the existing schemes, in this case the prover cannot commit anymore to the systematic form of $\mathscr{C}^*$. The issue is that now the prover provides only a *truncated* representation for the maps on the right: the verifier computes a code which is identical to $\mathscr{C}^*$ only in k out of n coordinates, so its systematic form will be different from the one of $\mathscr{C}^*$. To circumvent this issue, we modify the verification procedure and require that, after the computation of the systematic form, both the verifier and the prover execute an ad-hoc function which is an invariant under truncated monomial maps. We show that these extra steps have a cost which is much smaller than that of Gaussian elimination so that, in practice, the overall computational cost is only slightly affected. We also address the problem of communicating the information set which is used for verification: with a proper way to represent the truncated map, this cost can be entirely removed.

Practical outlook. Finally, in Section 6 we present some new instances of LESS signatures. These include new instances also for the ring signatures described in [BBN+22]. These are formulated with additional constraints and guidelines in mind, oriented at providing the best performance for the intended use case, and desired security level. Indeed, after recalling the state-of-the-art attacks on LEP, we propose a simple procedure to design secure LEP instances. This leads to parameters that are slightly larger than those employed in [BBPS21,BBN+22] but are more conservative. To be sure, this new procedure not only rules out the best attacks, i.e. the ones based on finding low-weight codewords (which is computationally equivalent to SDP), but also possible improvements to such attacks. This provides a very high level of confidence on the new parameters: new attacks, in order to significantly lower the security level, would need to be radically different from those based on low-weight codeword finding.

As mentioned before, the sizes resulting from this process are nearly half of those that would be obtained without our improvement. To be precise, we are able to produce signature sizes that range between 5 and 8.5 KiB, for NIST's security category 1. We also propose parameters for categories 3 and 5, ranging respectively between 14 and 18.5 KiB for the former, and 26 and 32.5 KiB for the latter. In all cases except one, the sum of our public keys and signatures is below 100KiB. To complete the picture, we include also some implementation figures, that we obtain by a reference implementation in ANSI C. While these numbers are far from optimized, they are still useful to show that the scheme is practical: indeed, the number of cycles is comparable with that obtained measuring the reference code of e.g. SPHINCS+.

2 Notation and Background

In this section we establish the notation that we will use throughout the paper, as well as recall basic concepts about linear codes.

2.1 Notation

As usual, we use $\mathbb{F}_q$ to indicate the finite field with q elements and $\mathbb{F}_q^*$ to indicate its multiplicative group. Given a matrix $\mathbf{A}$ over $\mathbb{F}_q$, we write $\mathbf{a}_i$ to indicate its i-th column. The general linear group formed by the non singular $k \times k$ matrices over $\mathbb{F}_q$ is indicated as GL_k. For an ordered set J, we write $\mathbf{A}_J$ to indicate the matrix formed by the columns of $\mathbf{A}$ that are indexed by the elements in J; equivalent notation is adopted for vectors. The identity with size k is indicated as $\mathbf{I}_k$, while $\mathbf{0}$ denotes the null-matrix (its dimensions will always be clear from the context). The standard matrix product between $\mathbf{A}$ and $\mathbf{B}$ is indicated as $\mathbf{AB}$, i.e., without any operator. In some cases, to avoid confusion with other operations, we will make it explicit and write the product as $\mathbf{A} \cdot \mathbf{B}$.

We denote by S_n the symmetric group on n elements, and consider its elements as permutations of n objects. We represent permutations in one-line notation, as n-tuples of the form $\pi := (i_1, i_2, \cdots, i_n)$, so that $\pi(j) = i_j$, i.e., π moves the j-th element to position i_j. For a vector $\mathbf{a} = (a_1, \cdots, a_n)$, it holds that

$$\pi(\mathbf{a}) = \left(a_{\pi^{-1}(1)}, \cdots, a_{\pi^{-1}(n)}\right).$$

We denote by M_n the set of monomial transformations, that is, transformations of the form $\mu := (\pi, \mathbf{v})$ with $\pi \in S_n$ and $\mathbf{v} \in \mathbb{F}_q^{*n}$, acting as follows

$$\mu(\mathbf{a}) = \pi(\mathbf{a}) \begin{pmatrix} v_1 & & & \\ & v_2 & & \\ & & \ddots & \\ & & & v_n \end{pmatrix} = \left(v_1 a_{\pi^{-1}(1)}, \cdots, v_n a_{\pi^{-1}(n)}\right).$$

We naturally extend the action of monomials on matrices $\mathbf{A}$, i.e., $\mu(\mathbf{A})$ indicates the matrix resulting from the action of μ on the columns of $\mathbf{A}$. For two monomials $\mu, \mu' \in M_n$, we write $\mu \circ \mu'$ to denote the monomial resulting from their combination.

2.2 Linear Codes

A linear code $\mathscr{C} \subseteq \mathbb{F}_q^n$ is a k-dimensional subspace of $\mathbb{F}_q^n$. The quantity $R = k/n$ is called *code rate*, and any vector $\mathbf{c} \in \mathscr{C}$ is called *codeword*. A canonical representation for a code is through a *generator matrix*, that is, a full-rank matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ such that $\mathscr{C} = \left\{\mathbf{uG} \mid \mathbf{u} \in \mathbb{F}_q^k\right\}$. Codes admit multiple generator matrices: for any $\mathbf{S} \in \mathrm{GL}_k$, which can be seen as a change of basis, it holds that $\mathbf{SG}$ and $\mathbf{G}$ generate the same code. The dual code $\mathscr{C}^\perp$ is the set of all vectors that are orthogonal to codewords in $\mathscr{C}$, that is, $\mathscr{C}^\perp = \left\{\mathbf{v} \in \mathbb{F}_q^n \mid \mathbf{cv}^\top = 0,\ \forall \mathbf{c} \in \mathscr{C}\right\}$. It is easy to see that $\mathscr{C}^\perp$ is a linear subspace of $\mathbb{F}_q^n$ with dimension $r = n - k$ (which is normally called *redundancy*). The dual code is generated by a full-rank matrix $\mathbf{H} \in \mathbb{F}_q^{r \times n}$, which is called *parity-check matrix* and is such that $\mathbf{GH}^\top = \mathbf{0}$. Obviously, for any $\mathbf{S} \in \mathrm{GL}_r$, $\mathbf{H}$ and $\mathbf{SH}$ are parity-check matrices for the same code.

For $J \subseteq \{1, \cdots, n\}$, we write $\mathscr{C}_J := \{\mathbf{c}_J \mid \mathbf{c} \in \mathscr{C}\}$. We say that a set J with size k is an *information set* for a code $\mathscr{C}$ if, for any two distinct $\mathbf{c}, \mathbf{c}' \in \mathscr{C}$, it holds that $\mathbf{c}_J \neq \mathbf{c}'_J$, which implies that $\mathscr{C}_J$ contains q^k elements. Equivalently, J is an information set if, for $\mathbf{G}$ being a generator matrix for $\mathscr{C}$, it holds that $\mathbf{G}_J$ is non singular. Normally, we say that a generator matrix $\mathbf{G}$ is in *systematic form* if $\mathbf{G} = \big(\mathbf{I}_k, \mathbf{V}\big)$, where $\mathbf{I}_k$ is the identity matrix of size k and $\mathbf{V} \in \mathbb{F}_q^{k \times (n-k)}$. This matrix exists whenever $J = \{1, \cdots, k\}$ is an information set: starting from any generator matrix $\mathbf{G}$, we obtain the one in systematic form as $\mathbf{G}_J^{-1}\mathbf{G}$. Also, the systematic matrix is an invariant under changes of basis: if $\mathbf{G}' = \mathbf{S}\mathbf{G}$, then its systematic form is $\mathbf{G}'^{-1}_J\mathbf{G}' = \mathbf{G}_J^{-1}\mathbf{S}^{-1}\mathbf{S}\mathbf{G} = \mathbf{G}_J^{-1}\mathbf{G}$.

In principle, there is no guarantee that $\{1, \cdots, k\}$ is an information set. Thus, sometimes one considers a slightly more general definition: given a matrix $\mathbf{G}$, its systematic form is $\mathbf{G}_J^{-1}\mathbf{G}$, where J is the first (according to some lexicographic ordering) subset of $\{1, \cdots, n\}$ of size k and such that $\mathbf{G}_J$ is non-singular. We refer to this operation as Row Reduced Echelon Form (RREF) with respect to J. To encompass the canonical definition of systematic matrix, we impose that the lexicographically first set is $\{1, \cdots, k\}$. It is easy to see that also this generalized definition is invariant under changes of basis: to emphasize this property, we will write $\mathsf{SF}(\mathscr{C})$ to denote the function that, on input a linear code, returns its systematic form.

Finally, we summarize here the traditional notion of equivalence between two codes, in the Hamming metric. To do this, we first clarify that we indicate with $\mu(\mathscr{C})$ the linear code obtained by applying the monomial transformation μ to all the codewords $\mathbf{c} \in \mathscr{C}$.

Definition 1 (Linear Equivalence). *We say that two codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ are* linearly equivalent*, and write $\mathscr{C} \sim \mathscr{C}'$, if there exists a monomial transformation $\mu \in M_n$ such that $\mathscr{C}' = \mu(\mathscr{C})$. That is, given generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ for $\mathscr{C}$ and $\mathscr{C}'$, respectively, the two codes are linearly equivalent if $\mathbf{G}' = \mathbf{S}\mu(\mathbf{G})$ for some non-singular matrix $\mathbf{S} \in \mathrm{GL}_k$, or analogously, if $\mathsf{SF}(\mathscr{C}') = \mathsf{SF}(\mu(\mathscr{C}))$.*

The above definition encompasses the weaker notion of *permutation equivalence*, which is the particular case where the monomial μ is a permutation.

3 The Code Equivalence Problem

The code equivalence problem generically asks, on input two codes $\mathscr{C}$ and $\mathscr{C}'$, to find a linear isometry mapping one code into the other. The problem is sometimes distinguished into two versions, depending on the type of isometry that one desires to identify. We present here only the more general one.

Problem 1 (Linear Equivalence Problem (LEP)) *Given $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ with dimension k, decide if $\mathscr{C} \sim \mathscr{C}'$, i.e., if there exists $\mu \in M_n$ such that $\mathscr{C}' = \mu(\mathscr{C})$. Equivalently, given $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ (generators for $\mathscr{C}$ and $\mathscr{C}'$, respectively), decide whether there exist $\mu \in M_n$ and $\mathbf{S} \in \mathrm{GL}_k$ such that $\mathbf{G}' = \mathbf{S}\mu(\mathbf{G})$.*

The *Permutation Equivalence Problem (PEP)* is just a special case of LEP, since any permutation is a monomial with scalar factors equal to 1.

Avoiding Weak Instances. Given its importance in coding theory, LEP has been studied for decades. As we have already mentioned, a well-known result states that the NP-completeness of LEP would imply a collapse of the polynomial hierarchy [PR97]. For PEP, there exist certain algorithms that can have a polynomial running time [Sen00,BOS19]. Namely, these attacks take times in $O\left(n^3 + q^{\widetilde{k}}\right)$ and $O\left(n^{2.3+\widetilde{k}}\right)$, respectively, where $\widetilde{k}$ is the dimension of the hull, that is, the linear code $\mathscr{C} \cap \mathscr{C}^{\perp}$. For random codes, the size of the hull tends to a small constant [Sen97], so that the above attacks become essentially polynomial in the code length. To counter these attacks, it suffices to use codes with large enough hull, or even *self-orthogonal codes*, that is, codes such that $\mathscr{C} \subseteq \mathscr{C}^{\perp}$. In this extreme case, in fact, the hull is equal to the code itself, so that $\widetilde{k} = k = Rn$, and the attacks in [Sen00,BOS19] take exponential time.

For LEP, however, it is still safe to use random codes, provided that the underlying finite field is sufficiently large. Indeed, there exists a polynomial time map that takes any LEP instance into a PEP instance, so that any solver for PEP can be used to solve LEP. However, when $q \geqslant 5$, this reduction always ends in a self-dual code [SS13]. This guarantees that the algorithms in [Sen00,BOS19] have maximum, exponential running time.

Attacks based on Low-weight Codeword Finding. The other class of attacks against PEP and LEP is characterized by the search for codewords with low Hamming weight (or subcodes with small support) [Leo82,Beu21,BBPS23]. The description of these attacks requires several technicalities which, due to lack of space, we cannot report here. Yet, they all share the common principle of looking at a small set of codewords (or subcodes) from which the action of μ can be recovered. For instance, Leon's algorithm [Leo82] requires to find, for each code, all codewords with weight $\leqslant w$, that is,

$$A = \{\mathbf{c} \in \mathscr{C} \mid \mathrm{wt}(\mathbf{c}) \leqslant w\}\,, \quad A' = \left\{\mathbf{c}' \in \mathscr{C}' \mid \mathrm{wt}(\mathbf{c}') \leqslant w\right\}.$$

This guarantees that $\mu(A) = A'$ and, when $w \ll n$, we have $|A| \ll |\mathscr{C}| = q^k$: roughly, since A and A' contain a few codewords, reconstructing μ gets easy. Modern algorithms relax the requirements of Leon and, instead, aim to find a sufficiently large number of *collisions*. This idea has been first proposed in [Beu21] and then refined in [BBPS23]. Here, by collision, we refer to a pair of codewords $\mathbf{c} \in \mathscr{C}$, $\mathbf{c}' \in \mathscr{C}'$ such that $\mu(\mathbf{c}) = \mathbf{c}'$. When the Hamming weights of $\mathbf{c}$ and $\mathbf{c}'$ are sufficiently small, collisions can be determined efficiently. The gain with respect to Leon's algorithm depends on several technicalities but, as a rule of thumb, it is enough to consider that this attack outperforms Leon's only if q is sufficiently large.

Note that the attack in [Beu21] actually uses two-dimensional subcodes instead of codewords. However, as observed in [BBPS23], this attack can be im-

proved by first finding low-weight codewords and then using them to build subcodes. This allows to improve upon the attack in [Beu21] since the component codewords have a much smaller support size than the resulting subcode, hence finding them is much easier.

Conservative Design Criteria. In practice, once weak instances are excluded, the best attacks against LEP are those based on low-weight codeword finding. Fortunately, this is one of the oldest and most studied problems in coding theory and we have a pretty consolidated picture about the cost of the best solvers, which are Information-Set Decoding (ISD) algorithms. In particular, for non-binary fields, the state-of-the-art is Peters' ISD [Pet10]. In the following, we will denote by $C_{\mathrm{ISD}}(q, n, k, w)$ the cost of finding a single codeword with weight w, in a code defined over $\mathbb{F}_q$, with length n and dimension k.

Looking at the attacks summarized above, we see that they all follow a general model, where an attacker always pursues the following strategy: i) produce two lists L_1 and L_2 with short codewords, ii) find collisions, i.e., pairs of elements in L_1 and L_2 that are presumably mapped by μ, and iii) use collisions to reconstruct the secret monomial. To obtain a conservative point of view on these attacks, we make the following choices:

- we assume that the technique employed to find collisions has no cost;
- we assume that the attacker never finds *fake collisions*, i.e., never considers $(\mathbf{c}, \mathbf{c}')$ as a collision even if $\mathbf{c} \neq \mu(\mathbf{c}')$. Note that fake collisions may make the monomial reconstruction unfeasible or, at the very least, much more complicated. In fact, the possibility of fake collisions is exactly the reason why the attacks in [Beu21,BBPS23] focus only on short codewords and, most importantly, work only when the finite field is large enough;
- we assume that knowing one collision is enough to retrieve significant and useful information about the secret monomial. Notice that all attacks, instead, require to find a sufficiently large number of collisions. For instance, Leon's algorithm requires to determine all codewords with some bounded weight. Analogously, the attacks in [Beu21,BBPS23] reconstruct exactly the secret monomial only if a sufficiently large number of (not fake) collisions is available. Yet, there may be ways to improve the monomial reconstruction phase (i.e., efficient techniques that require a smaller number of collisions), or to make use of some partial information. To show why this a concrete possibility, consider the case in which $\mathscr{C}$ contains only one minimum weight codeword $\mathbf{c}$. This gets mapped into $\mathbf{c}' = \mu(\mathbf{c}) \in \mathscr{C}'$. The pair $(\mathbf{c}, \mathbf{c}')$ already provides some information about μ: for instance, if $c_i = 0$ and $c'_j \neq 0$, we learn that μ does not move i in position j.

Taking into account the above three conservative assumptions, we use the following criterion to select secure LEP instances.

Criterion 1 *Let q, n, k denote, respectively, the finite field size, code length and dimension. We consider only $q \geqslant 5$ and random codes. We select n, k, q so*

that, for any $w \in \{1, \cdots, n\}$, finding lists $L_1 \subseteq \mathscr{C}$ and $L_2 \subseteq \mathscr{C}'$ with weight-w codewords and such that $L_2 \cap \mu(L_1)$ is non empty (where $\{\mu(\mathbf{c}) \mid \mathbf{c} \in L_1\}$), takes time greater than 2^λ.

This translates into a very simple way to select parameters. Indeed, let L_1 and L_2 have the same size ℓ. Then, the cost to produce these lists is

$$f\big(\ell, N(w)\big) \cdot \frac{C_{\mathrm{ISD}}(n,k,q,w)}{N(w)},$$

where $f(\ell, N(w))$ counts the number of ISD calls to find ℓ distinct codewords. The term $N(w)$ accounts for the number of codewords with weight w: since codes are random, this is well estimated as

$$N(w) = \binom{n}{w}(q-1)^w q^{-(n-k)}.$$

The cost of each ISD call is divided by $N(w)$ to take into account existence of multiple solutions.

We now observe that, on average, we have

$$|L_2 \cap \mu(L_1)| = \frac{|L_1| \cdot |L_2|}{N(w)} = \frac{\ell^2}{N(w)}.$$

Indeed, for each codeword in L_2, there is only one good collision among the $N(w)$ codewords in $\mathscr{C}$. Since we populate L_1 with ℓ random codewords, the probability that such a codeword is indeed in L_1 is $\frac{\ell}{N(w)}$. It follows that, in order to have at least one collision in the expectation, it must hold that $\ell^2 \geqslant N(w)$, which implies $\ell \geqslant \sqrt{N(w)}$. So, $\ell \ll N(w)$ and ℓ calls to ISD return, with high probability, ℓ distinct codewords [Beu21]. Consequently, we have

$$f\big(\ell, N(w)\big) \approx \ell = \sqrt{N(w)}.$$

Consequently, Criterion 1 translates into the following criterion.

Criterion 2 *We consider random codes defined over $\mathbb{F}_q$ with $q \geqslant 5$, and choose q, n, k so that, for any w, it holds that*

$$\frac{1}{\sqrt{N(w)}} \cdot C_{\mathrm{ISD}}(n,k,q,w) > 2^\lambda.$$

The above criterion emphasizes the fact that, when weak instances are avoided and in light of existing attacks, solving LEP reduces to finding low-weight codewords.

At the end of the day, as we will see in Section 6, the parameters we consider in this paper are only slightly bigger than those previously proposed in [BBPS21].

4 A New Formulation

In this section we show that LEP can be reformulated using a more convenient notion of equivalence, which allows for a much more compact representation for the solution to the equivalence problem. We introduce a new definition of equivalence between codes, which we call *Information Set (IS) - Linear Equivalence*, and then define the associated decisional problem, which we call IS-LEP. The main difference between LEP and IS-LEP is in that, for the latter, one is interested only in how the linear map acts on an information set (i.e., on k positions instead of n). We then show that IS-LEP is effectively the same as LEP, namely, that any "YES" (resp., "NO") instance of IS-LEP is also a "YES" (resp., "NO") instance for LEP.

4.1 Splitting Monomials with respect to Information Sets

To begin, we introduce some additional notation, which will help improve the readability of the next topics.

Definition 2. *Let* $\mathbf{G} = (\mathbf{g}_1, \cdots, \mathbf{g}_n) \in \mathbb{F}_q^{k\times n}$, $\mu = (\pi, \mathbf{v}) \in M_n$ *and* $\mathbf{G}' = \mu(\mathbf{G})$. *For any* $J' = \{j'_1, \cdots, j'_k\} \subseteq \{1, \cdots, n\}$, *we define* $J = \pi^{-1}(J') = \left\{\pi^{-1}(j') \mid j' \in J'\right\}$. *We define* $\mu^{(J\mapsto J')} \in M_k$ *as the monomial transformation such that* $\mu^{(J\mapsto J')}(\mathbf{G}_J) = \mathbf{G}'_{J'}$. *Equivalently, we define* $\mu^{(\setminus J\mapsto \setminus J')} \in M_{n-k}$ *as the monomial transformation such that* $\mu^{(\setminus J\mapsto \setminus J')}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}) = \mathbf{G}'_{\{1,\cdots,n\}\setminus J'}$.

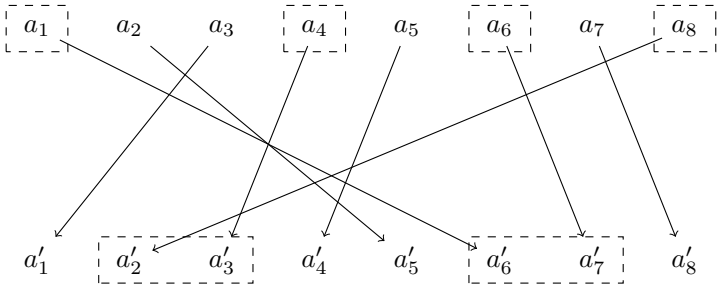
Determining $\mu^{(J\mapsto J')}$ from the knowledge of $\mu = (\pi, \mathbf{v})$ and J' is easy. Indeed, let us express $\mu^{(J\mapsto J')} = \left(\pi^{(J\mapsto J')}, \mathbf{v}^{(J\mapsto J')}\right)$. Then, it is enough to apply the following rule: if the i-th column of $\mathbf{G}'_{J'}$ corresponds to the j-th column of $\mathbf{G}_J$, multiplied by α, then we set $\pi^{(J\mapsto J')}(j) = i$ and $v_i^{(J\mapsto J')} = \alpha$. With analogous reasoning, one can compute $\mu^{(\setminus J\mapsto \setminus J')}$.

Splitting the action of a monomial with respect to a set J' is useful to understand how the map acts inside and outside an information set. Indeed, it is easy to verify that the following relation holds

$$\mathbf{G}' = \mathbf{S}\cdot\mu(\mathbf{G}) \implies \begin{cases} \mathbf{G}'_{J'} = \mathbf{S}\cdot\mu^{(J\mapsto J')}(\mathbf{G}_J), \\ \mathbf{G}'_{\{1,\cdots,n\}\setminus J'} = \mathbf{S}\cdot\mu^{(\setminus J\mapsto \setminus J')}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}). \end{cases} \tag{1}$$

We will frequently make use of the above relations to describe how monomial transformations act on specific sets of coordinates.

Example 1. Let us consider the example of $n = 8$ and $\mu = (\pi, \mathbf{v})$, with $\pi = (6, 5, 1, 3, 4, 7, 8, 2)$ and $\mathbf{v} = (2, 3, 1, 5, 3, 4, 6, 1)$ over $\mathbb{F}_7$. We describe how μ can be split, considering the set $J' = \{2, 3, 6, 7\}$. We observe that the permutation acts as follows (we are denoting $\mathbf{a}' = \pi(\mathbf{a})$):



We have $J = \{1, 4, 6, 8\}$, $\pi^{(J \mapsto J')} = (3, 2, 4, 1)$ and $\pi^{(\setminus J \mapsto \setminus J')} = (3, 1, 2, 4)$. Considering also the action of $\mathbf{v}$, we have that $\mu(\mathbf{a})$ is:

$$2a_3 \quad \boxed{3a_8 \quad 1a_4} \quad 5a_5 \quad 3a_2 \quad \boxed{4a_1 \quad 6a_6} \quad 1a_7$$

Hence, $\mathbf{v}^{(J \mapsto J')} = (3, 1, 4, 6)$ and $\mathbf{v}^{(\setminus J \mapsto \setminus J')} = (2, 5, 3, 1)$.

4.2 LEP with Information Sets

We are now ready to introduce the new notion of equivalence between codes which, at a first glance, may seem rather different from the traditional notion used to define LEP. Perhaps surprisingly, we are able to prove that the two notions are exactly the same.

Definition 3 (Information Set (IS) - Linear Equivalence). *We say that two codes $\mathscr{C}, \mathscr{C}' \subseteq \mathbb{F}_q^n$ are* Information Set (IS) linearly equivalent*, and write $\mathscr{C} \overset{*}{\sim} \mathscr{C}'$, if there exist monomial transformations $\widetilde{\mu} \in M_n$, $\zeta \in M_{n-k}$ and an information set J' for both $\mathscr{C}'$ and $\widetilde{\mathscr{C}} = \widetilde{\mu}(\mathscr{C})$ such that, for any codeword $\widetilde{\mathbf{c}} \in \widetilde{\mathscr{C}}$, there exists a codeword in $\mathbf{c}' \in \mathscr{C}'$ with*

i) $\widetilde{\mathbf{c}}_{J'} = \mathbf{c}'_{J'}$;
ii) $\widetilde{\mathbf{c}}_{\{1,\cdots,n\}\setminus J'} = \zeta\big(\mathbf{c}'_{\{1,\cdots,n\}\setminus J'}\big)$.

Equivalently, given generator matrices $\widetilde{\mathbf{G}}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ for $\widetilde{\mathscr{C}}$ and $\mathscr{C}'$, it must be

$$\widetilde{\mathbf{G}}_{J'}^{-1} \widetilde{\mathbf{G}}_{\{1,\cdots,n\}\setminus J'} = \zeta\big(\mathbf{G}'^{-1}_{J'} \mathbf{G}'_{\{1,\cdots,n\}\setminus J'}\big).$$

In other words, the two systematic generator matrices (computed with respect to the set J') have the non-systematic parts which are identical, up to a monomial transformation.

We associate this new notion of equivalence with the following decisional problem.

Problem 2 (Information Set-Linear Equivalence (IS-LEP)) *Given two linear codes $\mathscr{C}, \mathscr{C}'$, determine whether $\mathscr{C} \overset{*}{\sim} \mathscr{C}'$.*

In the next theorem we prove the core result of this section: LEP is equivalent to IS-LEP. Technically, we show that two codes are linearly equivalent if and only if they are also IS-linearly equivalent: this implies that any "YES" (resp. "NO") instance $(\mathscr{C},\mathscr{C}')$ for LEP is a "YES" (resp. "NO") instance for IS-LEP, and viceversa. Hence, IS-linear equivalence is merely a different formulation of the traditional notion of linear equivalence.

Before showing the full proof, as a warm up, we provide a small example which captures the essence of the relation between LEP and IS-LEP. Let $\mathscr{C}$ and $\mathscr{C}'$ be two equivalent codes and assume that $J' = \{1, \cdots, k\}$ is an information set for $\mathscr{C}'$. First, note that any solution μ for LEP is also a solution for IS-LEP: this corresponds to the special case of ζ in Definition 3 being the identity. Now, let $\widetilde{\mu}$ be a solution for IS-LEP and $\widetilde{\mathscr{C}} = \widetilde{\mu}(\mathscr{C})$: since we are considering RREF to the first k columns, we get

$$\mathsf{SF}(\mathscr{C}') = \big(\mathbf{I}_k, \mathbf{A}\big), \quad \mathsf{SF}(\widetilde{\mathscr{C}}) = \big(\mathbf{I}_k, \zeta(\mathbf{A})\big).$$

Then, $\widetilde{\mathscr{C}} \sim \mathscr{C}'$ are equivalent. But since $\widetilde{\mathscr{C}} \sim \mathscr{C}$, by transitive property we get that $\mathscr{C} \sim \mathscr{C}'$. The proof of the following theorem makes use of the above ideas, but does not restrict to a particular choice for J'; also, the proof is constructive, i.e., it shows explicit relations between solutions for LEP and IS-LEP.

Theorem 1 (Equivalence between IS-LEP and LEP). *For any pair of linear codes $\mathscr{C},\mathscr{C}' \subseteq \mathbb{F}_q^n$, it holds $\mathscr{C} \sim \mathscr{C}' \iff \mathscr{C} \overset{*}{\sim} \mathscr{C}'$.*

Proof. We first prove that $\mathscr{C} \sim \mathscr{C}'$ implies $\mathscr{C} \overset{*}{\sim} \mathscr{C}'$. Let us consider two generator matrices $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ for two equivalent codes $\mathscr{C}$ and $\mathscr{C}'$. Consequently, it holds $\mathbf{G}' = \mathbf{S}\mu(\mathbf{G})$ for some non-singular $\mathbf{S} \in \mathbb{F}_q^{k\times k}$ and $\mu = (\pi, \mathbf{v}) \in M_n$. We now show that $\mathscr{C}$ and $\mathscr{C}'$ are also IS-linearly equivalent. Let J' be an information set for $\mathscr{C}'$, and $J = \pi^{-1}(J')$ (recall Definition 2). Because of (1), we have

$$\mathbf{G}'_{J'} = \mathbf{S}\mu^{(J\mapsto J')}(\mathbf{G}_J), \qquad \mathbf{G}'_{\{1,\cdots,n\}\setminus J'} = \mathbf{S}\mu^{(\setminus J\mapsto \setminus J')}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}).$$

Representing the action of monomials through matrices, we rewrite the above relations as

$$\mathbf{G}'_{J'} = \mathbf{S}\mathbf{G}_J\mathbf{M}', \qquad \mathbf{G}'_{\{1,\cdots,n\}\setminus J'} = \mathbf{S}\mathbf{G}_{\{1,\cdots,n\}\setminus J}\mathbf{M}'',$$

with $\mathbf{M}' \in \mathbb{F}_q^{k\times k}$ and $\mathbf{M}'' \in \mathbb{F}_q^{(n-k)\times(n-k)}$. Reducing $\mathbf{G}'$ with respect to J', and considering only the non-systematic part, we obtain the matrix

$$\begin{aligned}\mathbf{A} &= \mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'} \\ &= \big(\mathbf{S}\mathbf{G}_J\mathbf{M}'\big)^{-1}\mathbf{S}\mathbf{G}_{\{1,\cdots,n\}\setminus J}\mathbf{M}'' \\ &= \mathbf{M}'^{-1}\mathbf{G}_J^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus J}\mathbf{M}''.\end{aligned}$$

Let $\widetilde{\mu} \in M_n$ be an arbitrary monomial such that $\widetilde{\mu}^{(J\mapsto J')} = \mu^{(J\mapsto J')}$ and, generically, $\widetilde{\mu}^{(\setminus J\mapsto \setminus J')} = \mu^{(\setminus J\mapsto \setminus J')} \circ \zeta$, where $\zeta \in M_{n-k}$ can be any monomial transformation. Using again matrices to represent monomials, we associate

$\widetilde{\mu}^{(J\mapsto J')} = \mu^{(J\mapsto J')}$ with $\mathbf{M}'$ and $\widetilde{\mu}^{(\setminus J\mapsto \setminus J')}$ with $\widetilde{\mathbf{M}}''$ which, in general, is different from $\mathbf{M}''$. Let $\widetilde{\mathbf{G}} = \widetilde{\mu}(\mathbf{G})$, and consider the non-systematic part of the matrix we obtain by row reducing with respect to J'. Taking again (1) into account, we have

$$\begin{aligned}\widetilde{\mathbf{G}}_{J'}^{-1}\widetilde{\mathbf{G}}_{\{1,\cdots,n\}\setminus J'} &= \big(\mathbf{G}_J\mathbf{M}'\big)^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus J}\widetilde{\mathbf{M}}'' \\ &= \underbrace{\mathbf{M}'^{-1}\mathbf{G}_J^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus J}}_{\mathbf{A}\mathbf{M}''^{-1}}\widetilde{\mathbf{M}}'' \\ &= \mathbf{A}\mathbf{M}''^{-1}\widetilde{\mathbf{M}}'' = \zeta(\mathbf{A})\end{aligned}$$

with $\zeta \in M_{n-k}$ being the monomial associated to $\mathbf{M}''^{-1}\widetilde{\mathbf{M}}''$. This proves that $\mathscr{C}$ and $\mathscr{C}'$ are indeed IS-linearly equivalent.

We now show the other way around, i.e., that two codes that are IS-linearly equivalent are also linearly equivalent. We consider again two generator matrices $\mathbf{G}$ and $\mathbf{G}'$ and assume we know an information set J' and monomials $\widetilde{\mu} \in M_n$, $\zeta \in M_{n-k}$ that satisfy the requirements for IS-linear equivalence. Let $\widetilde{\mathbf{G}} = \widetilde{\mu}(\mathbf{G})$. The non-systematic parts of $\mathbf{G}'$ and $\widetilde{\mathbf{G}}$, when reducing with respect to J', are

$$\mathbf{A}' = \mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'}, \qquad \widetilde{\mathbf{A}} = \widetilde{\mathbf{G}}_{J'}^{-1}\widetilde{\mathbf{G}}_{\{1,\cdots,n\}\setminus J'}.$$

By definition of IS-linear equivalence, we have that $\mathbf{A}'$ and $\widetilde{\mathbf{A}}$ are linearly equivalent, i.e.,

$$\widetilde{\mathbf{A}} = \zeta(\mathbf{A}') = \mathbf{A}'\mathbf{Z} = \mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'}\mathbf{Z},$$

for some monomial $\zeta \in M_{n-k}$ associated with the matrix $\mathbf{Z} \in \mathbb{F}_q^{(n-k)\times(n-k)}$. Let us again split the action of $\widetilde{\mu}$, using the information set J'. We write $\widetilde{\mu} = (\widetilde{\pi}, \widetilde{\mathbf{v}})$, set $J = \widetilde{\pi}^{-1}(J')$ and represent $\widetilde{\mu}^{(J\mapsto J')}$ and $\widetilde{\mu}^{(\setminus J\mapsto \setminus J')}$ through the monomial matrices $\widetilde{\mathbf{M}}' \in \mathbb{F}_q^{k\times k}$ and $\widetilde{\mathbf{M}}'' \in \mathbb{F}_q^{(n-k)\times(n-k)}$. We then have

$$\widetilde{\mathbf{G}}_{J'} = \widetilde{\mu}^{(J\mapsto J')}(\mathbf{G}_J) = \mathbf{G}_J\widetilde{\mathbf{M}}',$$

$$\widetilde{\mathbf{G}}_{\{1,\cdots,n\}\setminus J'} = \widetilde{\mu}^{(\setminus J\mapsto \setminus J')}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}) = \mathbf{G}_{\{1,\cdots,n\}\setminus J}\widetilde{\mathbf{M}}''.$$

Thus

$$\begin{aligned}\widetilde{\mathbf{A}} &= \widetilde{\mathbf{G}}_{J'}^{-1}\widetilde{\mathbf{G}}_{\{1,\cdots,n\}\setminus J'} \\ &= \big(\mathbf{G}_J\widetilde{\mathbf{M}}'\big)^{-1}\big(\mathbf{G}_{\{1,\cdots,n\}\setminus J}\widetilde{\mathbf{M}}''\big).\end{aligned}$$

Recalling that $\widetilde{\mathbf{A}} = \mathbf{A}'\mathbf{Z} \implies \widetilde{\mathbf{A}}\mathbf{Z}^{-1} = \mathbf{A}'$, we get

$$\underbrace{\big(\mathbf{G}_J\widetilde{\mathbf{M}}'\big)^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus J}}_{\widetilde{\mathbf{A}}}\widetilde{\mathbf{M}}''\mathbf{Z}^{-1} = \underbrace{\mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'}}_{\mathbf{A}'},$$

from which

$$\mathbf{G}_{\{1,\cdots,n\}\setminus J}\widetilde{\mathbf{M}}''\mathbf{Z}^{-1} = \mathbf{G}_J\widetilde{\mathbf{M}}'\mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'}. \tag{2}$$

We are now finally ready to determine the $\mathbf{S} \in \mathrm{GL}_k$ and $\mu \in M_n$ that would solve LEP on $\mathbf{G}$ and $\mathbf{G}'$. Indeed, let μ such that $\mu^{(J \mapsto J')}$ corresponds to $\widetilde{\mathbf{M}}'$ and $\mu^{(\setminus J \mapsto \setminus J')}$ corresponds to $\widetilde{\mathbf{M}}''\mathbf{Z}^{-1}$, and $\mathbf{S} = \mathbf{G}'_{J'}\widetilde{\mathbf{M}}'^{-1}\mathbf{G}_J^{-1}$. In the positions of $\mathbf{S}\mu(\mathbf{G})$ which are indexed by J', we have

$$\begin{aligned}\mathbf{S}\mu^{(J \mapsto J')}(\mathbf{G}_J) &= \mathbf{S}\mathbf{G}_J\widetilde{\mathbf{M}}' \\ &= \mathbf{G}'_{J'}\widetilde{\mathbf{M}}'^{-1}\mathbf{G}_J^{-1}\mathbf{G}_J\widetilde{\mathbf{M}}' = \mathbf{G}'_{J'},\end{aligned}$$

while in the positions which are not indexed by J',

$$\begin{aligned}\mathbf{S}\mu^{(\setminus J \mapsto \setminus J')}(\mathbf{G}) &= \mathbf{S}\mathbf{G}_{\{1,\cdots,n\}\setminus J}\widetilde{\mathbf{M}}''\mathbf{Z}^{-1} \\ &= \underbrace{\mathbf{G}'_{J'}\widetilde{\mathbf{M}}'^{-1}\mathbf{G}_J^{-1}}_{\mathbf{S}}\underbrace{\mathbf{G}_J\widetilde{\mathbf{M}}'\mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'}}_{\mathbf{G}_{\{1,\cdots,n\}\setminus J}\widetilde{\mathbf{M}}''\mathbf{Z}^{-1}} \qquad \text{(Using (2))} \\ &= \mathbf{G}'_{\{1,\cdots,n\}\setminus J'}.\end{aligned}$$

□

We conclude this section by showing how, from the knowledge of a solution for LEP, one can derive a solution to IS-LEP, which is more convenient in terms of communication cost. This depends on how the action of a monomial can be represented. To this end, we introduce the following functions, which we will use to represent the action of $\widetilde{\mu}$.

Definition 4. *Let $\mu = (\pi, \mathbf{v}) \in M_n$. For $J' = \{j'_1, \cdots, j'_k\} \subseteq \{1, \cdots, n\}$ with size k, we define*

$$\begin{aligned}\mathsf{Trunc}(\mu, J') &= \big(\pi^*, \mathbf{v}^*\big) \\ &= \Big(\big(\pi^{-1}(j'_1), \pi^{-1}(j'_2), \cdots, \pi^{-1}(j'_k)\big) \ , \ \big(v_{j'_1}, v_{j'_2}, \cdots, v_{j'_k}\big)\Big).\end{aligned}$$

Notice that π^* is an ordered subset of $\{1, \cdots, n\}$ with size k, that is, $\pi^* = (j_1^*, \cdots, j_k^*)$. Also, $\mathbf{v}^* = (v_1^*, \cdots, v_k^*) = \mathbf{v}_{J'}$ is represented as a length-k vector over $\mathbb{F}_q^*$.

Definition 5. *We define $\mathsf{Apply}\big((\pi^*, \mathbf{v}^*), \mathbf{G}\big)$ as the function that outputs the matrix $\mathbf{U} \in \mathbb{F}_q^{k \times k}$ such that, if the i-th entry of π^* is $j \in \{1, \cdots, n\}$, has i-th column $\mathbf{u}_i = v_i^*\mathbf{g}_j$, where $\mathbf{g}_j$ denotes the j-column of $\mathbf{G}$.*

Remark 1. The elements of π^* and $J = \pi^{-1}(J')$ are the same, but have a different order. While J represents an information set (and, coherently with the notation we are using, is a non-ordered set), π^* is meant to describe how π acts on the coordinates which are moved to J'. Consequently, it is important that π^* is seen as an ordered set. Notice that π^* describes the action of π only on k coordinates (hence, the function is called Trunc, which stand for truncated).

Remark 2. If J' is the information set that has been used to compute $(\pi^*, \mathbf{v}^*)$ using the monomial μ and $J = \pi^{-1}(J')$, then $\mathbf{U} = \mu^{(J \mapsto J')}(\mathbf{G})$.

We first observe that representing π^* requires $k\log_2(n)$ bits while $\mathbf{v}^*$ takes $k\log_2(q-1)$ bits. The implication on signatures based on LEP is easy to see. In fact, all existing schemes communicate monomial transformations using $n\log_2(n)+n\log_2(q-1)$ bits, i.e., the action of the monomial is fully represented. We are aiming at reducing this size thanks to the convenient representation we have defined above. However, this requires some additional technical steps (e.g. modifications in how commitments are computed). Thus, we postpone this discussion to the next section, and we conclude the current one by showing that communicating $\mathsf{Trunc}(\mu, J')$ is enough to verify a solution to IS-LEP, in a time which is essentially not modified with respect to LEP.

Proposition 1 *Let $\mathscr{C}$ and $\mathscr{C}'$ be two linearly equivalent codes, i.e., there exists $\mu = (\pi, \mathbf{v}) \in M_n$ such that $\mathscr{C}' = \mu(\mathscr{C})$. Let $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ be generator matrices for such codes. To show that $\mathscr{C}$ and $\mathscr{C}'$ are IS-linearly equivalent, it is enough to provide J' and $\mathsf{Trunc}(\mu, J')$. Verifying the solution for IS-LEP takes a time which is polynomial in n and, in practice, is the same as computing two RREFs.*

Proof. Since the codes are linearly equivalent, there exists $\mathbf{S} \in \mathrm{GL}_k$ such that $\mathbf{G}' = \mathbf{S}\mu(\mathbf{G})$. Let us indicate $J = \pi^{-1}(J')$; notice that J is known because of π^* (see Remark 1). Thanks to (1), we can write

$$\mathbf{G}'_{J'} = \mathbf{S}\mu^{(J\mapsto J')}(\mathbf{G}_J),$$

$$\mathbf{G}'_{\{1,\cdots,n\}\setminus J'} = \mathbf{S}\mu^{(\setminus J\mapsto \setminus J')}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}).$$

Let $\mathbf{U} = \mathsf{Apply}\big((\pi^*, \mathbf{v}^*), \mathbf{G}\big)$, and notice that

$$\mathbf{U} = \mathbf{G}_{\{1,\cdots,n\}\setminus \pi^*} = \mathbf{G}_{\{1,\cdots,n\}\setminus J}.$$

Indeed, we consider that π^* is identical to J, up to a reordering of the elements, hence $\{1,\cdots,n\}\setminus\{j_1^*,\cdots,j_k^*\} = \{1,\cdots,n\}\setminus J$. Let $\widetilde{\mu} \in M_n$ be any monomial such that $\mathsf{Trunc}(\mu, J') = \mathsf{Trunc}(\widetilde{\mu}, J')$ and $\widetilde{\mathbf{G}} = \widetilde{\mu}(\mathbf{G})$. We now compute the RREFs of both $\mathbf{G}'$ and $\widetilde{\mathbf{G}}$ with respect to J'. The non-systematic parts of the two matrices are, respectively,

$$\begin{aligned}
\mathbf{A}' &= \mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'} \\
&= \big(\mathbf{S}\cdot\mu^{(J\mapsto J')}(\mathbf{G}_J)\big)^{-1}\cdot\mathbf{S}\cdot\mu^{(\setminus J\mapsto\setminus J')}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}) \\
&= \big(\mu^{(J\mapsto J')}(\mathbf{G}_J)\big)^{-1}\cdot\mu^{(\setminus J\mapsto\setminus J')}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}) \\
&= \big(\mu^{(J\mapsto J')}(\mathbf{G}_J)\big)^{-1}\cdot\mathbf{G}_{\{1,\cdots,n\}\setminus J}\mathbf{Z},
\end{aligned}$$

and

$$\begin{aligned}
\widetilde{\mathbf{A}} &= \widetilde{\mathbf{G}}^{-1}_{J'}\widetilde{\mathbf{G}}_{\{1,\cdots,n\}\setminus J'} \\
&= \big(\mu^{(J\mapsto J')}(\mathbf{G}_J)\big)^{-1}\cdot\mathbf{G}_{\{1,\cdots,n\}\setminus J} \\
&= \mathbf{A}'\cdot\mathbf{Z}^{-1},
\end{aligned}$$

where $\mathbf{Z}$ is the matrix associated to $\mu^{(\setminus J \mapsto \setminus J')}$. To conclude verification, one should acknowledge that indeed $\mathbf{A}'$ and $\widetilde{\mathbf{A}}$ are identical, up to a monomial transformation. This can be easily verified. For instance, it is enough to consider scalar multiples of the columns in $\mathbf{A}'$ and search whether $\widetilde{\mathbf{A}}$ contains an identical column. Namely, we start with $i = 1$ (i.e, consider the first column $\mathbf{a}'_i = \mathbf{a}'_1$): if, in $\widetilde{\mathbf{A}}$, we find a column in position $j \in \{1, \cdots, n-k\}$ and such that $z\mathbf{a}'_1 = \widetilde{\mathbf{a}}'_j$, then we know that ζ moves the first coordinate in position j, and scales it by z. We then repeat the reasoning, considering $i = 2$ and searching for the matching column in the positions $\{1, \cdots, n-k\} \setminus \{j\}$. Iterating this procedure, we have that we successfully end the search (i.e., we find a match for all columns of $\mathbf{A}'$) if and only if there indeed exists such a monomial $\mathbf{Z}$. The cost of this procedure is $O(n^2)$, which is smaller than that of computing the RREFs, which is in $O(n^3)$. □

In the next section, we show how IS-LEP can be employed to build ZK proofs. We anticipate that, in such applications, we will not need to transmit J'; furthermore, we will use a different approach to determine if the non systematic parts of two matrices are equal up to a monomial transformation (based on the computation of an ad-hoc invariant).

Remark 3. Even though we stated LEP and IS-LEP as decisional problems, they can be reformulated as search problems. The proofs of Theorem 1 and Proposition 1 show, constructively, reductions in both ways, even for the search versions.

Example 2. Let $q = 11$, $n = 5$ and $k = 2$. Let $\mathscr{C}$ be the code generated by

$$\mathbf{G} = \begin{pmatrix} 9 & 3 & 1 & 1 & 4 \\ 2 & 5 & 5 & 10 & 1 \end{pmatrix}.$$

Let $\mu = (\pi, \mathbf{v})$ with $\pi = (2, 1, 5, 3, 4)$ and $\mathbf{v} = (5, 6, 8, 9, 3)$ and $\mathscr{C}' = \mu(\mathscr{C})$. To represent the code, we use the generator matrix $\mathbf{G}' = \mathbf{S}\mu(\mathbf{G})$ with $\mathbf{S} = \begin{pmatrix} 0 & 4 \\ 4 & 10 \end{pmatrix}$, so that

$$\mathbf{G}' = \begin{pmatrix} 1 & 4 & 1 & 3 & 5 \\ 2 & 6 & 7 & 3 & 8 \end{pmatrix}.$$

Let $J' = \{1, 4\}$, which is an information set for $\mathscr{C}'$ since $\mathbf{G}'_{J'} = \begin{pmatrix} 1 & 3 \\ 2 & 3 \end{pmatrix}$ is non-singular (its determinant is 8). We have $\mathsf{Trunc}(\mu, J') = (\pi^*, \mathbf{v}^*)$ where $\pi^* = \{\pi^{-1}(1), \pi^{-1}(4)\} = \{j_1^*, j_2^*\} = \{2, 5\}$ and $\mathbf{v}^* = \mathbf{v}_{J'} = (v_1^*, v_2^*) = (5, 9)$. We now consider $\mathbf{U} = \mathsf{Apply}\big((\pi^*, \mathbf{v}^*), \mathbf{G}\big)$ and have

$$\mathbf{U} = \left(v_1^* \mathbf{g}_{j_1^*}\ ,\ v_2^* \mathbf{g}_{j_2^*}\right) = \left(5\mathbf{g}_2\ ,\ 9\mathbf{g}_5\right) = \left(5 \cdot \begin{pmatrix} 3 \\ 5 \end{pmatrix}\ ,\ 9 \cdot \begin{pmatrix} 4 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} 4 & 3 \\ 3 & 9 \end{pmatrix}.$$

Since $\{1, \cdots, n\} \setminus \{j_1^*, j_2^*\} = \{1, 3, 4\}$, we have

$$\mathbf{G}_{\{1,\cdots,n\}\setminus\pi^*} = (\mathbf{g}_1, \mathbf{g}_3, \mathbf{g}_4) = \begin{pmatrix} 9 & 1 & 1 \\ 2 & 5 & 10 \end{pmatrix}.$$

We now compute the non-systematic part of $\mathbf{G}'$, after RREF with respect to J', and obtain

$$\begin{aligned}\mathbf{A}' &= \mathbf{G}'^{-1}_{J'}\mathbf{G}'_{\{1,\cdots,n\}\setminus J'} = (\mathbf{g}'_1, \mathbf{g}'_4)^{-1}(\mathbf{g}'_2, \mathbf{g}'_3, \mathbf{g}'_5) \\ &= \begin{pmatrix} 1 & 3 \\ 2 & 3 \end{pmatrix}^{-1} \begin{pmatrix} 4 & 1 & 5 \\ 6 & 7 & 8 \end{pmatrix} = \begin{pmatrix} 2 & 6 & 3 \\ 8 & 2 & 8 \end{pmatrix}.\end{aligned}$$

Finally, we have

$$\widetilde{\mathbf{A}} = \mathbf{U}^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus\pi^*} = \begin{pmatrix} 4 & 3 \\ 3 & 9 \end{pmatrix}^{-1} \begin{pmatrix} 9 & 1 & 1 \\ 2 & 5 & 10 \end{pmatrix} = \begin{pmatrix} 4 & 1 & 9 \\ 5 & 10 & 3 \end{pmatrix}.$$

Now, we observe that

$$\mathbf{a}'_1 = 6 \cdot \widetilde{\mathbf{a}}_1, \qquad \mathbf{a}'_2 = 8 \cdot \widetilde{\mathbf{a}}_3, \qquad \widetilde{\mathbf{a}}_3 = 8 \cdot \mathbf{a}'_2.$$

This confirms that $\mathbf{A}'$ and $\widetilde{\mathbf{A}}$ are equal, up to a monomial transformation.

5 Compact Proofs of Equivalence from IS-LEP

Recall that, in a proof-of-knowledge constructed from LEP, the protocol goes as follows (see Figure 1):

- there are two equivalent (public) codes $\mathcal{C}$ and $\mathcal{C}'$, with $\mathcal{C}' = \sigma(\mathcal{C})$ for some (secret) map μ;
- the prover samples a random transformation $\tau \in M_n$ and commits to $\mathcal{C}^* = \tau(\mathcal{C})$; this is done by applying a function $\mathsf{Commit}(\mathcal{C}^*)$ whose output is $h \in \{0;1\}^{2\lambda}$;
- the verifier either asks for the random map (i.e., a proof that $\mathcal{C} \sim \mathcal{C}^*$) and receives τ, or for the one involving the public code (i.e., a proof that $\mathcal{C}^* \sim \mathcal{C}'$) and receives $\tau' = \tau \circ \mu^{-1}$;
- the verifier either checks that $h = \mathsf{Commit}\big(\tau(\mathcal{C})\big)$, or that $h = \mathsf{Commit}\big(\tau'(\mathcal{C}')\big)$.

Note that we must necessarily assume that the commitment is obtained via a hash function, since otherwise one would need to publish $\mathcal{C}^*$, which requires at least $k(n-k)\log_2(q)$ bits (assuming a generator matrix in systematic form is employed). Currently, the commitment function is implemented as

$$\mathsf{Commit} = \mathsf{Hash}\big(\mathsf{SF}(\mathcal{C})\big) : \mathcal{C} \mapsto \{0,1\}^{2\lambda}.$$

This works well since it satisfies two fundamental properties:

i) the systematic generator matrix is an invariant of the code;
ii) the commitment function is relatively easy to compute.

The second property is obviously necessary to have a practical scheme, while the first one is crucial to guarantee verification, when the verifier asks for the equivalence on the right. Indeed, in this case he computes $\tau'(\mathbf{G}')$, which generates the same code as $\tau(\mathbf{G})$. However, the two generator matrices are not the same: generically, it holds that $\tau(\mathbf{G}') = \mathbf{S} \cdot \tau'(\mathbf{G})$ for some non-singular $\mathbf{S} \in \mathrm{GL}_k$. Thanks to use of the systematic form, we get rid of this discrepancy.

To put it differently, the systematic form is used as an easy-to-compute representative for a code[1]. As we have seen in the previous section, with the IS-LEP formulation we can reduce significantly the communication cost. However, the commitment function which is currently employed will not work anymore, since the prover provides only a portion of τ'. In this section we describe an efficient solution to circumvent this issue. This requires to modify the commitment function and use a new invariant which, fortunately, can be computed with a cost which is comparable with that of a RREF. This leads to a direct improvement in all schemes based on LEP, for what concerns all relevant aspects: we reduce the communication cost (in practice $k \approx 0.5n$ so we almost halve it) and essentially keep the computational cost unchanged. Also, we do not introduce a new security assumption since, as we showed in the previous section, IS-LEP and LEP are two different formulations of the very same problem.

5.1 A New Invariant for Codes

Let us recall the concept of *lexicographic ordering* for vectors and matrices over a finite field.

Definition 6 (Lexicographic Ordering). *We define a lexicographic ordering over* $\mathbb{F}_q = \{x_1, x_2, \cdots, x_q\}$ *as*

$$x_1 \overset{\mathsf{Lex}}{<} x_2 \overset{\mathsf{Lex}}{<} \cdots \overset{\mathsf{Lex}}{<} x_q.$$

For two vectors $\mathbf{a}, \mathbf{b}$*, we write* $\mathbf{a} \overset{\mathsf{Lex}}{<} \mathbf{b}$ *if there exists an* i *such that* $a_j = b_j$ *for all* $j < i$*, and* $a_i \overset{\mathsf{Lex}}{<} b_i$*. Analogously, for two matrices* $\mathbf{A}$ *and* $\mathbf{B}$*, we write* $\mathbf{A} \overset{\mathsf{Lex}}{<} \mathbf{B}$ *if there exists an* i *such that* $\mathbf{a}_j = \mathbf{b}_j$ *for all* $j < i$ *and* $\mathbf{a}_i \overset{\mathsf{Lex}}{<} \mathbf{b}_i$*, where* $\mathbf{a}_i$ *and* $\mathbf{b}_i$ *denote the* i*-th columns of* $\mathbf{A}$ *and* $\mathbf{B}$*, respectively. We write* $\mathbf{A} \overset{\mathsf{Lex}}{\leqslant} \mathbf{B}$ *if either* $\mathbf{A} = \mathbf{B}$ *or* $\mathbf{A} \overset{\mathsf{Lex}}{<} \mathbf{B}$*.*

Using the notion of lexicographic ordering defined above, we can define a representative for the orbit of a matrix, under the action of monomial transformations[2].

[1] There exist other invariants, but their computation is much harder. For instance, the prover may commit to the hash of the weight enumerator function. However, its computation requires $O(q^k)$ operations and is obviously unfeasible.

[2] In the context of code linear equivalence, these concepts have been first used by Beullens [Beu21].

Definition 7 (First Lexicographic Matrix). *Given* $\mathbf{A} \in \mathbb{F}_q^{m \times u}$*, we denote its orbit under the action of* M_u *as* $M_u(\mathbf{A}) = \{\tau(\mathbf{A}) \,|\tau \in M_u\}$*. Then, we define* $\mathsf{MinLex}(\mathbf{A})$ *as the function that returns the first lexicographic matrix in the orbit, that is*

$$\mathsf{MinLex}(\mathbf{A}) = \mathbf{A}^* \iff \mathbf{A}^* \overset{\mathsf{Lex}}{\leqslant} \widehat{\mathbf{A}}, \ \ \forall \widehat{\mathbf{A}} \in M_u(\mathbf{A}).$$

Note that the above definitions hold for any arbitrary choice of lexicographic ordering. However, since we are mostly interested in prime finite fields, from now on we focus on the simplest and most natural ordering, that is $0 \overset{\mathsf{Lex}}{<} 1 \overset{\mathsf{Lex}}{<} 2 \overset{\mathsf{Lex}}{<} \cdots \overset{\mathsf{Lex}}{<} q-1$. If $\mathbf{A}$ has m rows and u columns, computing MinLex takes in the worst case $O(um)$ operations over $\mathbb{F}_q$: indeed, it is enough to first scale each column so that the first non null element is 1 and then sort the columns so that they are in ascending lexicographic ordering. An example is given in Figure 3.

$$\begin{pmatrix} 5 & 0 & 3 & 8 \\ 10 & 5 & 7 & 0 \end{pmatrix} \xrightarrow{\text{Scale columns}} \begin{pmatrix} 1 & 0 & 1 & 1 \\ 2 & 1 & 6 & 0 \end{pmatrix} \xrightarrow{\text{Reorder columns}} \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 6 \end{pmatrix}$$

Fig. 3: Example of computation of MinLex, for a matrix with $m = 2$ rows, $u = 4$ columns, with values over $\mathbb{F}_{11}$.

We finally have all the necessary tools to define our proposed invariant function, which we call SF^*. Details about how the function operates are given in Algorithm 1. Basically, it computes the systematic form and then computes MinLex on the non systematic part. Since computing MinLex is much easier than a RREF, computing SF^* comes with a cost which is slightly larger than that of SF. We observe that, in the wide majority of cases, the employed information set is $J^* = \{1, \cdots, k\}$ (i.e., the one that is tested first). Indeed, the probability that this set is valid can be estimated by considering the probability that a random $k \times k$ matrix over $\mathbb{F}_q$ is non-singular, that is

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

For instance, for $q = 127$, this is approximately 0.992.

To conclude this section, we show that the function SF^* possesses exactly the invariance properties we need.

Proposition 2 *Let* $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ *be the generator matrices of two linearly equivalent codes, i.e.,* $\mathbf{G}' = \mathbf{S}\mu(\mathbf{G})$ *for some* $\mathbf{S} \in \mathrm{GL}_k$ *and* $\mu \in M_n$*. Let* $J^*, \mathbf{A}^* = \mathsf{SF}^*(\mathbf{G}')$*. Let* $(\pi^*, \mathbf{v}^*) = \mathsf{Trunc}(\mu, J^*)$ *and* $\mathbf{U} = \mathsf{Apply}\big((\pi^*, \mathbf{v}^*), \mathbf{G}\big)$*. Then, for any* μ *and any* $\mathbf{S}$*, it holds that*

$$\mathbf{A}^* = \mathsf{MinLex}(\mathbf{U}^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus\pi^*}).$$

Proof. Let J be the set of columns that get moved to J^*. Because of RREF, the effect of $\mathbf{S}$ gets canceled. So, RREF with respect to J^* yields

$$\mathbf{A}' = \mathbf{U}^{-1}\mathbf{G}'_{\{1,\cdots,n\}\setminus J^*} = \mathbf{U}^{-1}\mu^{(J\mapsto J^*)}(\mathbf{G}_{\{1,\cdots,n\}\setminus J}),$$

which is identical (up to a monomial transformation) to

$$\mathbf{A}'' = \mathbf{U}^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus \pi^*} = \mathbf{U}^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus J}.$$

This means that they are in the same orbit, i.e., $\mathbf{A}' \in \mathsf{M}_{n-k}(\mathbf{A}'')$: computation of MinLex returns the same matrix. □

Algorithm 1: Function SF^*

Input: matrix $\mathbf{G} \in \mathbb{F}_q^{k\times n}$
Output: set $J^* \subseteq \{1,\cdots,n\}$, matrix $\mathbf{A}^* \in \mathbb{F}_q^{k\times(n-k)}$

1 Find the first $J^* \subseteq \{1,\cdots,n\}$ of size k and such that $\mathrm{Rank}(\mathbf{G}_{J^*}) = k$;
2 Set $\mathbf{A} = \mathbf{G}_{J^*}^{-1}\mathbf{G}_{\{1,\cdots,n\}\setminus J^*}$;// Non systematic part after RREF
3 Compute $\mathbf{A}^* = \mathsf{MinLex}(\mathbf{A})$;// Compute first lexicographic matrix
4 Return J^*, $\mathbf{A}^*$.

5.2 Proof-of-knowledge with IS-LEP

We now describe how the proof-of-knowledge protocol used in the family of LESS schemes [BMPS20,BBPS21,BBN+22] can be reformulated to take into account IS-LEP. In Figure 4 we have reported the description of one round of the LESS-FM protocol, taking into account verification based on IS-LEP.

The protocol possesses all the properties that are required by a ZK proof of knowledge. Completeness holds because of Proposition 2, while Zero-Knowledge is guaranteed by the fact that $(\pi^*, \mathbf{v}^*)$ is a truncated representation of τ', which is uniformly distributed over M_n. The only property which is not obvious is special soundness; for this reason, we present a detailed analysis next.

Proposition 3 *The protocol of Figure 4 is 2-special sound.*

Proof. Let us consider two accepting transcripts, associated with the same commitment h and two different challenges b and $\widetilde{b}$. We assume that both b and $\widetilde{b}$ are different from 0 (the case where one of the challenges is 0 trivially follows and is therefore omitted). We denote by $(\pi^*, \mathbf{v}^*)$ the response for challenge b, and by $(\widetilde{\pi}^*, \widetilde{\mathbf{v}}^*)$ the one for challenge $\widetilde{b}$. We now show that, from the knowledge of these two accepting transcripts, either a hash collision has been found, or a monomial map from $\mathscr{C}_b$ to $\mathscr{C}_{\widetilde{b}}$ can be computed in polynomial time.

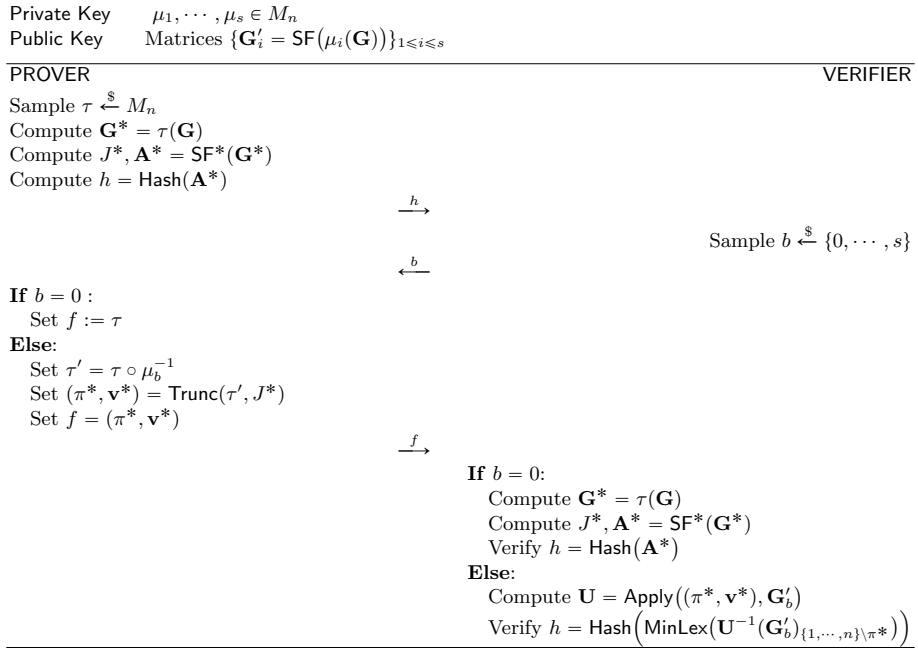


Fig. 4: One round of LESS-FM using IS-LEP

Let $\mathbf{U} = \mathsf{Apply}\big((\pi^*, \mathbf{v}^*,)\mathbf{G}_b\big)$ and $\widetilde{\mathbf{U}} = \mathsf{Apply}\big((\widetilde{\pi}^*, \widetilde{\mathbf{v}}^*), \mathbf{G}_{\widetilde{b}}\big)$. Since both are accepting transcripts, it follows that either a hash collision has been found, or

$$\mathsf{MinLex}\big(\underbrace{\mathbf{U}^{-1}(\mathbf{G}'_b)_{\{1,\cdots,n\}\setminus\pi^*}}_{\mathbf{A}}\big) = \mathsf{MinLex}\big(\underbrace{\widetilde{\mathbf{U}}^{-1}(\mathbf{G}'_{\widetilde{b}})_{\{1,\cdots,n\}\setminus\widetilde{\pi}^*}}_{\widetilde{\mathbf{A}}}\big).$$

This means that one knows two monomial transformations $\zeta, \widetilde{\zeta} \in M_{n-k}$ such that $\zeta(\mathbf{A}) = \widetilde{\zeta}(\widetilde{\mathbf{A}}) = \mathbf{A}^*$.

Remember that what Apply does is applying a monomial transformation that modifies only the k coordinates which are included in π^*. In other words, starting from $\mathbf{G}_b$, one possesses the generator matrix for an equivalent code, in the form $\big(\mathbf{U}, \mathbf{G}'_{b\{1,\cdots,n\}\setminus\pi^*}\big)$. Let us denote by $\sigma \in M_n$ the monomial such that $\sigma(\mathbf{G}'_b) = \big(\mathbf{U}, \mathbf{G}'_{b\{1,\cdots,n\}\setminus\pi^*}\big)$. Doing RREF with respect to the first k positions, we find a generator matrix for the same code, in the form $\big(\mathbf{I}_k, \mathbf{A}\big)$. If we now apply another monomial transformation $\sigma' \in M_n$, acting as the identity in the first k positions and as ζ in the last $n-k$ positions, we end up with $\big(\mathbf{I}_k, \zeta(\mathbf{A})\big) = \big(\mathbf{I}_k, \mathbf{A}^*\big)$. This means that $\mathscr{C}_b$, the code generated by $\mathbf{G}_b$, is equivalent to the one $\mathscr{C}^*$ generated by $\big(\mathbf{I}_k, \mathbf{A}^*\big)$: the equivalence between the two codes is given by $\sigma' \circ \sigma$.

The same chain of transformations can be applied to $\mathbf{G}'_{\widetilde{b}}$, and would bring us to the code generated by $\big(\mathbf{I}_k, \widetilde{\zeta}(\widetilde{\mathbf{A}})\big) = \big(\mathbf{I}_k, \mathbf{A}^*\big)$. To summarize all the transformations we used, see Figure 5.

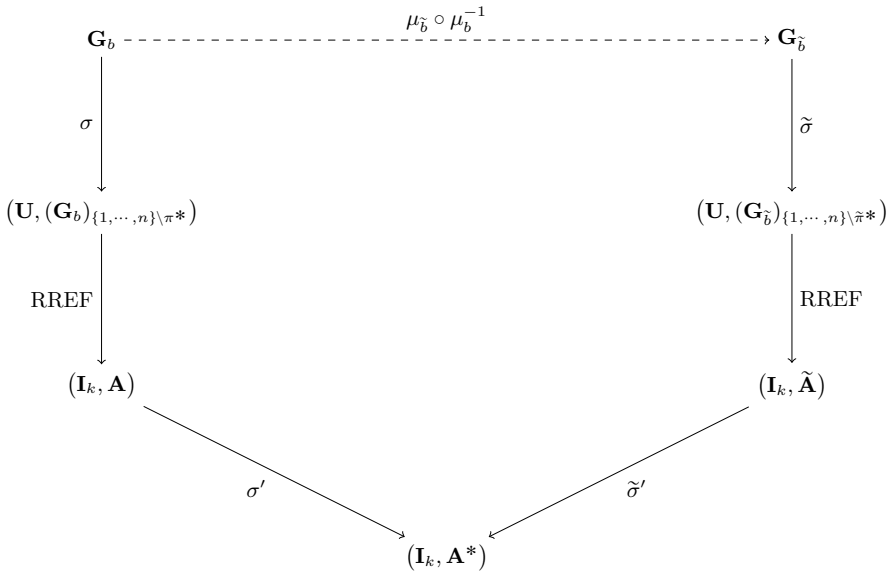


Fig. 5: Transformations from $\mathscr{C}_b$ and $\mathscr{C}_{\tilde{b}}$ to a common code $\mathscr{C}^*$

In the end, we found a code $\mathscr{C}^*$ which is equivalent to both $\mathscr{C}_b$ and $\mathscr{C}_{\tilde{b}}$, and we also know the transformations that map $\mathscr{C}_b$ into $\mathscr{C}^*$ and $\mathscr{C}_{\tilde{b}}$ into $\mathscr{C}^*$. Combining such transformations, we are able to find a map between $\mathscr{C}_b$ and $\mathscr{C}_{\tilde{b}}$. □

For what concerns computational complexity, as in LESS-FM, the most time consuming operation remains the systematic form computation.

6 New Instances for LESS Signatures

In this section, we report on the practical impact of our new techniques, in the context of LESS, as well as schemes derived from it. To begin with, we recall the parameters that were proposed in LESS-FM. Table 1, below, is an excerpt from [BBPS21].

Optimization Criterion	**Type**	**Code Params** n	k	q	**Prot. Params** t	ω	s	**pk** (KiB)	**sig** (KiB)
Min. pk size	Mono	198	94	251	283	28	2	9.77	15.2
Min. sig size	Perm	235	108	251	66	19	16	205.74	5.25
Min. pk + sig size	Perm	230	115	127	233	31	2	11.57	10.39

Table 1: Parameter sets for LESS-FM, for $\lambda = 128$ classical bits of security.

At this point, a few comments on these parameters are due. First, note that two out of three parameter sets use permutation equivalence, namely those which aim at minimizing the signature in some way. This make sense, since a permutation can be described utilizing only $n \log_2(n)$ bits, as opposed to the $n \log_2(n) + n \log_2(q-1)$ necessary for a monomial matrix; the latter term includes in fact the cost of storing the non-zero scaling factors. However, in this work (as well as subsequent ones) we will focus mainly on the monomial case. In fact, using permutations requires additional care in the definition of the protocol. For instance, as we have seen in Section 3, it makes the scheme vulnerable to certain types of algebraic attacks, so that it is not safe to use random codes. This presents a challenge in practice, as generating self-orthogonal codes can be quite expensive.

Secondly, as mentioned at the end of Section 3, we adopt a new, conservative criterion for choosing parameters with respect to best attacks, which leads to different choices for code lengths and dimensions. Furthermore, we include in our thought process some considerations connected to implementation efficiency, which were absent in the LESS-FM work: for instance, we restrict our attention to the value $q = 127$, which is optimal in this sense, and avoid parameters which would yield excessive data sizes. With respect to the latter, we decide then to remain within the psychological threshold of 100 kB.

Finally, as we transition from a mostly theoretical design, to one with a practical outlook, we provide parameters for higher security levels. For this, we follow NIST's guidance and align with their proposed definitions for categories 1, 3 and 5. We report the new data in Table 2, with a slightly different layout. Indeed, we no longer need to specify the type of equivalence considered, since this is always monomial. Also, the optimization criterion is no longer purely aimed at "minimizing" quantities. Instead, we use the nomenclature LESS-$\alpha\beta$ which recalls simultaneously the security level achieved (via the number $\alpha \in \{1, 3, 5\}$), and the characteristics of the resulting choice (via the letter β). To be precise, we use "b" for "balanced", i.e. a set which yields similar sizes for public key and signature; "s" for "short", i.e. a set which sacrifices public-key size in favor of signature; and "i", only for category 1, for an "intermediate" set.

To illustrate the advantage of our technique, in Table 2 we have reported signature sizes for both the scheme with, and without the new technique; to do so, we have use the format $x(y)$ where x is the optimized signature size, and y the unoptimized one.

Next, we report some timings. We start with those obtained for an unoptimized reference implementation in ANSI C, which are to be considered purely in the spirit of exemplification. The values are collected on an Intel Core i7-12700K, on a P-core, clocked at 4.9 GHz. Clock cycle values collected via `rtdscp`, as averages of 100 primitive runs. The computer is endowed with 64 GiB of PC5-19200 DDR5 and is running Debian 11. The source was compiled with `gcc` 10.2.1-20210110 (version packaged with the distribution), with `-O3 -march=native` compilation options.

NIST Cat.	Parameter Set	Code Params n	k	q	Prot. Params t	ω	s	pk (KiB)	sig (KiB)
1	LESS-1b	252	126	127	247	30	2	13.6	8.4 (15.3)
	LESS-1i				244	20	4	40.8	5.8 (10.7)
	LESS-1s				198	17	8	95.2	5.0 (9.2)
3	LESS-3b	400	200	127	759	33	2	34.2	16.8 (30.5)
	LESS-3s				895	26	3	68.5	13.4 (24.2)
5	LESS-5b	548	274	127	1352	40	2	64.2	29.8 (53.8)
	LESS-5s				907	37	3	128.5	26.6 (48.8)

Table 2: New parameter sets for LESS, for different security categories.

NIST Cat.	Parameter Set	KeyGen (Mcycles)	Sign (Mcycles)	Verify (Mcycles)
1	LESS-1b	3.4	878.7	890.8
	LESS-1i	9.8	876.6	883.6
	LESS-1s	23.0	703.6	714.7
3	LESS-3b	9.3	7224.1	7315.8
	LESS-3s	18.3	8527.4	8608.6
5	LESS-5b	24.4	33787.7	34014.0
	LESS-5s	48.0	22621.5	22703.3

Table 3: Timings for the reference implementation of LESS.

To provide a hint at the improved performance that we can obtain by leveraging more advanced tools, we report below the results of an additional implementation. Since, as explained above, the RREF computation is by far the most expensive operation, this implementation is realized by amending the ANSI C reference code with Gaussian Elimination code implemented using AVX2 C intrinsics. The test system was a Dell OptiPlex XE4, a mid-range 2022 desktop system with Intel Core i7-12700 CPU running at 2.1 GHz. The test programs were executed on a single CPU thread with frequency scaling disabled. The system has 64GB of physical RAM and was running Ubuntu 22.04.2 LTS Linux operating system, and the C test code was compiled with gcc 11.3.0 packaged in that operating system. Compilation and optimization flags were \verb|-Wall -Wextra -Ofast -march=native|.

To complete our showcase, we report below the data obtained while applying our technique to the LESS-based ring signature scheme.

Table 5 is an excerpt from [BBN+22], with some caveats. First, note that the parameter s is missing, as the optimization involving multiple codes was not used; instead, we have a new parameter r corresponding to the size of the

NIST Cat.	Parameter Set	KeyGen (Mcycles)	Sign (Mcycles)	Verify (Mcycles)
1	LESS-1b	0.9	263.6	271.4
	LESS-1i	2.3	254.3	263.4
	LESS-1s	5.1	206.6	213.4
3	LESS-3b	2.8	2446.9	2521.4
	LESS-3s	5.2	2984.3	3075.1
5	LESS-5b	6.4	10212.6	10458.8
	LESS-5s	11.7	6763.2	7016.5

Table 4: Timings for the additional implementation of LESS.

Parameter Set	Code Params			Prot. Params			pk (kB)	sig (kB)
	n	k	q	t	ω	r		
I	230	115	127	233	31	2^3	11.6	8.6 (10.8)
II						2^6		11.6 (13.8)
III						2^{12}		17.5 (19.7)
IV						2^{21}		26.5 (28.7)

Table 5: Parameter sets for ring signatures based on LESS, for $\lambda = 128$ classical bits of security.

ring of users. Secondly, all the instances presented in [BBN+22] were based on permutation equivalence (and thus the "Type" column is omitted). In this case, rather than presenting entirely new parameters based on (IS-)LEP, we simply calculate the sizes that we would obtain applying our technique to PEP, i.e. replacing $n \log_2(n)$ bits with $k \log_2(n)$ bits whenever a permutation needs to be transmitted. We use the same $x(y)$ format as above, where now the unoptimized value y corresponds to the sizes appearing in [BBN+22].

Note that, compared to the reduction obtained for LESS, in the case of ring signature the improvement is considerably less relevant. This is mainly because a large part of the signature size, in such a scheme, is comprised of the cost of transmitting a Merkle proof, which is proportional to the (logarithm) of the number of users in the ring. It is worth considering, however, that this is exactly the feature that makes the scheme appealing in the first place, and so we are satisfied with our improvement being less impactful in this case.

Acknowledgements

The work of the first author is generously sponsored by NSF grant 1906360 and NSA grant H98230-22-1-0328.

References

BBN+22. Alessandro Barenghi, Jean-François Biasse, Tran Ngo, Edoardo Persichetti, and Paolo Santini. Advanced signature functionalities from the code equivalence problem. *International Journal of Computer Mathematics: Computer Systems Theory*, 0(ja):1–0, 2022.

BBPS21. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. LESS-FM: fine-tuning signatures from the code equivalence problem. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *PQCrypto 2021*, volume 12841 of *LNCS*, pages 23–43. Springer, 2021.

BBPS23. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. On the computational hardness of the code equivalence problem in cryptography. *Advances in Mathematics of Communications*, 17(1):23–55, 2023.

Beu21. Ward Beullens. 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*, pages 387–403. Springer, 2021.

BKP20. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and falafl: logarithmic (linkable) ring signatures from isogenies and lattices. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part II*, pages 464–492. Springer, 2020.

BKV19. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. Csi-fish: efficient isogeny based signatures through class group computations. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 227–247. Springer, 2019.

BMPS20. Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. LESS is more: Code-based signatures without syndromes. In Abderrahmane Nitaj and Amr Youssef, editors, *AFRICACRYPT*, pages 45–65. Springer, 2020.

BOS19. M. Bardet, A. Otmani, and M. Saeed-Taha. Permutation code equivalence is not harder than graph isomorphism when hulls are trivial. In *IEEE ISIT 2019*, pages 2464–2468, July 2019.

CNP+23. Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Take your MEDS: Digital signatures from matrix code equivalence. In *International Conference on Cryptology in Africa*, pages 28–52. Springer, 2023.

DG22. Giuseppe D'Alconzo and Andrea Gangemi. Trifors: Linkable trilinear forms ring signature. *Cryptology ePrint Archive*, 2022.

DvW22. Léo Ducas and Wessel van Woerden. On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part III*, pages 643–673. Springer, 2022.

FG19. L. De Feo and S. Galbraith. Seasign: Compact isogeny signatures from class group actions. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT*

2019, volume 11478 of *Lecture Notes in Computer Science*, pages 759–789. Springer, 2019.

GMW19. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 285–306. 2019.

Leo82. J. Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3):496–511, May 1982.

Pat96. Jacques Patarin. Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 33–48. Springer, 1996.

Pet10. C. Peters. Information-set decoding for linear codes over $\mathbb{F}_q$. In *International Workshop on Post-Quantum Cryptography*, pages 81–94. Springer, 2010.

PR97. E. Petrank and R. M. Roth. Is code equivalence easy to decide? *IEEE Transactions on Information Theory*, 43(5):1602–1604, Sep. 1997.

PRS22. Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, and Paolo Santini. An attack on a non-interactive key exchange from code equivalence. *Tatra Mountains Mathematical Publications*, 82(2):53–64, 2022.

Sen97. Nicolas Sendrier. On the dimension of the hull. *SIAM Journal on Discrete Mathematics*, 10(2):282–293, 1997.

Sen00. Nicolas Sendrier. The support splitting algorithm. *Information Theory, IEEE Transactions on*, pages 1193 – 1203, 08 2000.

SS13. Nicolas Sendrier and Dimitris E Simos. The hardness of code equivalence over 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*, pages 203–216. Springer, 2013.

TDJ+22. Gang Tang, Dung Hoang Duong, Antoine Joux, Thomas Plantard, Youming Qiao, and Willy Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In *Advances in Cryptology–EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30–June 3, 2022, Proceedings, Part III*, pages 582–612. Springer, 2022.

ZZ21. Zhuoran Zhang and Fangguo Zhang. Code-based non-interactive key exchange can be made. Cryptology ePrint Archive, Report 2021/1619, 2021.