On Linear Equivalence, Canonical Forms, and Digital Signatures

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

[1] Academia Sinica, Taipei, Taiwan
[2] Florida Atlantic University, Boca Raton, USA and Sapienza University, Rome, Italy
[3] Marche Polytechnic University, Ancona, Italy

Abstract. In this preliminary version of the work, we sketch our ideas regarding canonical forms for matrices, and how they can be used to obtain compact signatures from the linear equivalence problem.

1 Introduction

LESS is a post-quantum signature scheme which was first introduced in [6]. The core of LESS is a Sigma protocol based on the permutation equivalence problem (PEP) or the linear equivalence problem (LEP); the Sigma protocol is then turned into a signature scheme using the Fiat-Shamir transform. Several improvements followed in subsequent works: for instance, the authors in [3] show that the signature size can be reduced by having more than 2 generator matrices in the public key, as well as by selecting challenges according to a fixed-weight distribution. In [10], it is shown that it is possible to further reduce the signature size, by a large factor, as some pieces of information in the commitments are redundant. The idea of [10] is later used for the specification of LESS [1], a project submitted to NIST's call for additional post-quantum signatures [9]. The specification shows that the smallest signature sizes are around 5.0 KB, 13.4 KB, and 26.6 KB for security categories 1, 3 and 5, respectively. These sizes are achieved by using more than 2 generator matrices in each public key, which increases the public key sizes by a large factor.

In this paper, we show that the idea of [10] can be described using the concept of equivalent relations and canonical form of matrices. Using such a concept, we are able to generalize this approach and exploit various types of canonical forms; thus, we are able propose a new family of LESS-like signature schemes which we collectively call

LEQ, of which LESS serves as a special case. With LEQ, it is possible to achieve even smaller signatures: for instance, the signature sizes are only around 2.4 KB, 5.7 KB, and 9.8 KB for category 1, 3 and 5, respectively, for one version of LEQ, with only 2 generator matrices in each public key.

2 Notation and Preliminaries

We will denote by q a prime power, e.g., an integer of the form $q = p^m$ with p prime and $m \in \mathbb{N}$. As usual, $\mathbb{F}_q$ will denote the finite field with q elements. We will use $\mathsf{poly}(n, q)$ to indicate the set of all positive-valued functions of n and q which can be bounded by a polynomial in n and q.

We utilize three matrix groups in our work, which are all in fact subgroups of the general linear group, $\mathsf{GL}_n(\mathbb{F}_q)$. The first, is the symmetric group, i.e. the group of permutations on n objects, represented as binary matrices with exactly a single 1 in each row and column; this is denoted by P_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 $\mathsf{D}_n(\mathbb{F}_q)$. Finally, we define a group which is a generalization of P_n, where each permutation is scaled with non-zero factors from $\mathbb{F}_q$; we denote this by $\mathsf{M}_n(\mathbb{F}_q)$. In other words, for each $Q \in \mathsf{M}_n(\mathbb{F}_q)$, we have that $Q = P \cdot D$, where $P \in \mathsf{P}_n$ and $D \in \mathsf{D}_n(\mathbb{F}_q)$; it is easy to see, then, that this group is isomorphic to $\mathsf{P}_n \times \mathsf{D}_n(\mathbb{F}_q)$. Such matrices are known as *monomial matrices*, and we will thus refer to the group $\mathsf{M}_n(\mathbb{F}_q)$ as the *monomial group* over $(\mathbb{F}_q)$. For the remainder of the paper, we will often omit $\mathbb{F}_q$, when clear from the context, and simply write $\mathsf{P}_n, \mathsf{D}_n, \mathsf{M}_n$.

The matrices considered in this work are mostly treated in the context of coding theory. Indeed, each matrix, defined over a finite field, potentially defines a linear code, as follows. Let $q, n, k \in \mathbb{Z}$ such that q is a prime power and $n > k > 0$. The value n is called *length* of the code, while k is its *dimension*. This terminology stems from linear algebra, as indeed a linear code is nothing but a vector space, and more precisely, a subspace of $\mathbb{F}_q^n$ having dimension k as a vector space, i.e. admits a basis with k linearly independent vectors. When these basis vectors are collected as row vectors, they form a

$k \times n$ matrix G known as *generator matrix*, since it generates the code in the vector-space fashion, i.e. via linear combinations of the basis elements. In other words, a linear code $\mathcal{C}$ is defined as the set $\left\{uG \mid u \in \mathbb{F}_q^k\right\}$. Note that there are several choices for generator matrices for the same code, corresponding to different choices of basis, which can be obtained by applying a non-singular change-of-basis matrix $S \in \mathsf{GL}_k(\mathbb{F}_q)$; in particular, when S is the inverse of the leftmost $k \times k$ submatrix, one obtains a generator in the form $(I_k \mid M)$, where I_k is the identity matrix of size k, which is known as *systematic form*.

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 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 weights. 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*[4]. 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 $\mathcal{C}_0$ is permutationally (resp. linearly) equivalent to $\mathcal{C}_1$, then there exist a change-of-basis matrix $S \in \mathsf{GL}_k(\mathbb{F}_q)$ and a permutation $P \in \mathsf{P}_n$ (resp. monomial $Q \in \mathsf{M}_n$) such that $G_1 = S \cdot G_0 \cdot P$, where G_0 and G_1 are the respective generator matrices.

Determining whether two codes are equivalent is not an easy task. In cryptography, one is often more interested in the computational version of the problem, which we report below.

Problem 1 (Code Equivalence) *Given $\mathcal{C}_0, \mathcal{C}_1 \subseteq \mathbb{F}_q^n$ with dimension k, having generator matrices G_0, G_1 respectively, decide if $\mathcal{C}_0$ is*

[4] 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.

equivalent to $\mathcal{C}_1$, i.e., if there exists $S \in \mathsf{GL}_k(\mathbb{F}_q)$ and $P \in \mathsf{P}_n$ (resp. $Q \in \mathsf{M}_n$) such that $G_1 = SG_0P$ (resp. $G_1 = SG_0Q$).

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 PEP and LEP has been studied in detail through several works, and we point the interested reader to [4, 5, 12] for an extensive treatment.

3 The LESS Signature Scheme

We will now briefly illustrate the Sigma protocol, based on the LEP problem, underlying the LESS signature scheme. To begin, the prover publishes as public key a pair of equivalent codes $(\mathcal{C}_0, \mathcal{C}_1)$ defined by respective generator matrices (G_0, G_1), storing the equivalence map as its private key. It then proceeds using the following steps:

1. The prover first generates a random matrix $\tilde{Q}$ in M_n (or P_n in the permutation version) such that $\mathsf{SF}(G_0\tilde{Q}) \neq \perp$. Then, the commitment cmt is computed as $\mathsf{Hash}(\mathsf{SF}(G_0\tilde{Q}))$.
2. The verifier generates the challenge $\mathsf{ch} \in \{0, 1\}$.
3. The prover generates the response rsp as $(1 - \mathsf{ch}) \cdot \tilde{Q} + \mathsf{ch} \cdot Q^{-1}\tilde{Q}$.
4. The verifier accepts the response if $\mathsf{Hash}(\mathsf{SF}(G_{\mathsf{ch}} \cdot \mathsf{rsp})) = \mathsf{cmt}$.

4 Equivalence Relations on Matrices

Let $q, n, k \in \mathbb{Z}$ be the typical coding theory parameters (see Section 2). This section introduces and discusses some equivalence relations on $\mathbb{F}_q^{k\times(n-k)}$ and $\mathbb{F}_q^{k\times n}$. The equivalence relations introduced in this sections are defined using 4 sets of matrices:

- The set $\pi_r \in \{\{I_k\}, \mathsf{P}_k(\mathbb{F}_q)\}$.
- The set $\pi_c \in \{\{I_{n-k}\}, \mathsf{P}_{n-k}(\mathbb{F}_q)\}$.
- The set $\delta_r \in \{\{I_k\}, \mathsf{D}_k(\mathbb{F}_q)\}$.
- The set $\delta_c \in \{\{I_{n-k}\}, \mathsf{D}_{n-k}(\mathbb{F}_q)\}$.

4.1 Equivalence Relations on $\mathbb{F}_q^{k\times(n-k)}$

Definition 1. *Given $F = \pi_r \times \pi_c \times \delta_r \times \delta_c$, we define ϵ_F as the equivalence relation such that any two matrices $A, B \in \mathbb{F}_q^{k\times(n-k)}$ are equivalent under ϵ_F (denoted as $A \sim_{\epsilon_F} B$) if and only if*

$$B = P_r \cdot D_r \cdot A \cdot D_c \cdot P_c,$$

for some $(P_r, P_c, D_r, D_c) \in F$.

Notice that the type of equivalence between two matrices depends on how the sets $\pi_r, \pi_c, \delta_r, \delta_c$ are defined. For instance, if $\delta_r = \{I_k\}$ and $\delta_c = \{I_{n-k}\}$, then $A \sim_{\epsilon_F} B$ implies that $A = P_r \cdot B \cdot P_c$, since $D_r, D_c \in \delta_r \times \delta_c = \{I_k\} \times \{I_{n-k}\}$, which means $D_r = I_k$ and $D_c = I_{n-k}$.

4.2 Equivalence Relations on Systematic Matrices in $\mathbb{F}_q^{k\times n}$

Definition 2. *Given $F = \pi_r \times \pi_c \times \delta_r \times \delta_c$, we define $\mathcal{E}_F$ as the equivalence relation such that any two systematic matrices $A, B \in \mathbb{F}_q^{k\times n}$ are equivalent under $\mathcal{E}_F$ (denoted as $A \sim_{\mathcal{E}_F} B$) if and only if*

$$B = T \cdot A \cdot \begin{bmatrix} D_r^{-1}P_r^{-1} & 0 \\ 0 & D_cP_c \end{bmatrix},$$

for some $(P_r, P_c, D_r, D_c) \in F$ and $T \in GL_k(\mathbb{F}_q)$.

4.3 Relationship between ϵ and $\mathcal{E}$

Theorem 1. *Let $\hat{A} = (I_k \mid A) \in \mathbb{F}_q^{k\times n}$ and $\hat{B} = (I_k \mid B) \in \mathbb{F}_q^{k\times n}$,*

$$\hat{A} \sim_{\mathcal{E}_F} \hat{B} \iff A \sim_{\epsilon_F} B.$$

Moreover,

$$\hat{B} = T \cdot \hat{A} \cdot \begin{bmatrix} D_r^{-1}P_r^{-1} & 0 \\ 0 & D_cP_c \end{bmatrix} \iff B = P_r \cdot D_r \cdot A \cdot D_c \cdot P_c.$$

Proof.

$$\begin{aligned}
\underbrace{(I_k \mid B)}_{=\hat{B}} &= T \cdot \underbrace{(I_k \mid A)}_{=\hat{A}} \cdot \begin{bmatrix} D_r^{-1}P_r^{-1} & 0 \\ 0 & D_cP_c \end{bmatrix} \\
&= \underbrace{T}_{=P_r \cdot D_r} \cdot ((P_rD_r)^{-1} \mid A) \cdot \begin{bmatrix} I_k & 0 \\ 0 & D_cP_c \end{bmatrix} \\
&= (P_rD_r) \cdot ((P_rD_r)^{-1} \mid A) \cdot \begin{bmatrix} I_k & 0 \\ 0 & D_cP_c \end{bmatrix} \\
&= \left[I_k \mid (P_rD_r) \cdot A \cdot (D_cP_c) \right]
\end{aligned}$$

4.4 Representatives in Equivalence Classes

Once F is defined, ϵ_F, $\mathcal{E}_F$, and the corresponding equivalence classes are also defined. For the purpose of this paper, we are interested in specifying representatives for some (not necessarily all) equivalence classes. We call the representative of an equivalence class the *canonical form* of any matrix in the equivalence class. For the purpose of this paper, it would be sufficient if

1. for "most" of the matrices, the canonical form is well-defined, and

2. there is an "efficient" algorithm that, given any matrix in $\mathbb{F}_q^{k\times(n-k)}$, outputs the canonical form if it is well-defined or $\perp$ otherwise.

In the following, we formalize such properties.

Definition 3. Canonical form
Let $F = \pi_r \times \pi_c \times \delta_r \times \delta_c$. Let $\mathsf{CF} : \mathbb{F}_q^{k\times(n-k)} \mapsto \left\{\{\perp\} \cup \mathbb{F}_q^{k\times(n-k)}\right\}$; we say that CF is a canonical form for F *if it satisfies the following properties:*

i) for any $A \in \mathbb{F}_q^{k\times(n-k)}$, computing $\mathsf{CF}(A)$ takes time which is in $\mathsf{poly}(n,q)$;
ii) let $\gamma(n,q)$ denote the success probability, i.e., the probability that $\mathsf{CF}(A)$ does not return a failure when A is uniformly random over $\mathbb{F}_q$; then, it must be $\frac{1}{\gamma(n,q)} \in \mathsf{poly}(n,q)$;

iii) *if* $B = \mathsf{CF}(A)$*, then it must be* $A \sim_{\varepsilon_F} B$*, i.e.,*

$$\exists(P_r, P_c, D_r, D_c) \in F \text{ such that } B = D_r \cdot P_r \cdot A \cdot P_c \cdot D_c.$$

Also, computing P_r, P_c, D_r, D_c *should take time in* $\mathsf{poly}(n, q)$*;*

iv) *for any two matrices* $A \sim_{\varepsilon_F} A'$*, it must be* $\mathsf{CF}(A) = \mathsf{CF}(A')$*.*

As we shall see in the following, the above properties are crucial to both dispose of a useful (i.e., efficiently computable) canonical form and, at the same time, allow for a tight correspondence between the classical code equivalence problem and the problems we rely on in this paper.

Due to Theorem 1, for any $\hat{A} = (I_k \mid A) \in \mathbb{F}_q^{k\times n}$, we define $\mathrm{CF}(\hat{A})$ as $(I_k \mid \mathrm{CF}(A))$ if $\mathrm{CF}(A) \neq \perp$, or $\perp$ otherwise.

5 Using $\boldsymbol{\mathcal{E}_F}$ to Reduce LESS Signature Size: LEQ

Given $F = \pi_r \times \pi_c \times \delta_r \times \delta_c$, the underlying Sigma protocol in the LEQ signature scheme is essentially the same as the one shown in Section 3, except that and $\mathsf{SF}(G_0 \cdot \tilde{Q})$ is replaced by $\mathsf{CF}(\mathsf{SF}(G_0 \cdot \tilde{Q}))$ for commitment, and $\mathsf{SF}(G_{\mathrm{ch}} \cdot \mathsf{rsp})$ is replaced by $\mathsf{CF}(\mathsf{SF}(G_{\mathrm{ch}} \cdot \mathsf{rsp}))$ for verification. Interestingly, this simple change allows us to compress rsp in various ways, depending on how F is defined.

5.1 Compressing rsp

Given q, k, n, let $\mathsf{P}_{k,n}$ be the subset of P_n such that

- in the first k columns, the row indices of 1's are in increasing order.
- in the last $n-k$ columns, the row indices of 1's are in increasing order.

In fact, once the first k columns of a matrix in $\mathsf{P}_{k,n}$ is defined, the whole matrix is defined. Therefore, every matrix in $\mathsf{P}_{k,n}$ can be represented as a vector in $\mathbb{F}_2^n$ of Hamming weight k, such that the row indices of the 1's in the first k columns are exactly the indices of the 1's in the vector.

Theorem 2. *For every $P \in \mathsf{P}_n$, we have*

$$P = P_{is} \cdot \begin{bmatrix} P_r^{-1} & 0 \\ 0 & P_c \end{bmatrix},$$

for some $(P_{is}, P_r, P_c) \in \mathsf{P}_{k,n} \times \mathsf{P}_k \times \mathsf{P}_{n-k}$. For every $M \in \mathsf{M}_n$, we have

$$M = P_{is} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & D_c P_c \end{bmatrix},$$

for some $(P_{is}, P_r, P_c, D_r, D_c) \in \mathsf{P}_{k,n} \times \mathsf{P}_k \times \mathsf{P}_{n-k} \times \mathsf{D}_k \times \mathsf{D}_{n-k}$.

One can make use of Theorem 2 to reduce the size of rsp. We explain the idea using some examples.

Example: PEP with $(\delta_r, \pi_r, \delta_c, \pi_c) = (\{I_k\}, \mathsf{P}_k, \{I_{n-k}\}, \mathsf{P}_{n-k})$. In this case, rsp can be compressed into the corresponding P_{is} matrix of $\tilde{Q}$ or $Q^{-1} \cdot \tilde{Q}$. Indeed, the verifier can verify rsp since

$$\begin{aligned} \mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}})) &= \mathsf{CF}(T \cdot G_{\text{ch}} \cdot P_{\text{is}}) \\ &= \mathsf{CF}(P_r \cdot T \cdot G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} P_r^{-1} & 0 \\ 0 & P_c \end{bmatrix}) \\ &= \mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} P_r^{-1} & 0 \\ 0 & P_c \end{bmatrix})). \end{aligned}$$

Example: LEP with $(\delta_r, \pi_r, \delta_c, \pi_c) = (\{I_k\}, \{I_k\}, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$. In this case, rsp can be compressed into the corresponding P_{is}, P_r, D_r matrices of $\tilde{Q}$ or $Q^{-1} \cdot \tilde{Q}$. Indeed, the verifier can verify rsp since

$$\begin{aligned} \mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & I_{n-k} \end{bmatrix}) &= \mathsf{CF}(T \cdot G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & I_{n-k} \end{bmatrix}) \\ &= \mathsf{CF}(T \cdot G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & D_c P_c \end{bmatrix}) \\ &= \mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & D_c P_c \end{bmatrix})). \end{aligned}$$

Note that this is exactly the technique used in [1, 10].

Example: LEP with $(\delta_r, \pi_r, \delta_c, \pi_c) = (\{I_k\}, \mathsf{P}_k, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$. In this case, rsp can be compressed into the corresponding P_{is}, D_r matrices of $\tilde{Q}$ or $Q^{-1} \cdot \tilde{Q}$. Indeed, the verifier can verify rsp since

$$\begin{aligned}
\mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} & 0 \\ 0 & I_{n-k} \end{bmatrix}) &= \mathsf{CF}(T \cdot G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} & 0 \\ 0 & I_{n-k} \end{bmatrix}) \\
&= \mathsf{CF}(P_r \cdot T \cdot G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & D_c P_c \end{bmatrix}) \\
&= \mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & D_c P_c \end{bmatrix})).
\end{aligned}$$

Example: LEP with $(\delta_r, \pi_r, \delta_c, \pi_c) = (\mathsf{D}_k, \mathsf{P}_k, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$. In this case, rsp can be compressed into the corresponding P_{is} matrix of $\tilde{Q}$ or $Q^{-1} \cdot \tilde{Q}$. Indeed, the verifier can verify rsp since

$$\begin{aligned}
\mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}})) &= \mathsf{CF}(T \cdot G_{\text{ch}} \cdot P_{\text{is}}) \\
&= \mathsf{CF}(P_r \cdot D_r \cdot T \cdot G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & D_c P_c \end{bmatrix}) \\
&= \mathsf{CF}(\mathsf{SF}(G_{\text{ch}} \cdot P_{\text{is}} \cdot \begin{bmatrix} D_r^{-1} P_r^{-1} & 0 \\ 0 & D_c P_c \end{bmatrix})).
\end{aligned}$$

6 Attacks against PEP and LEP

Disposing of efficiently computable canonical forms may be also help in cryptanalysing the code equivalence problem. Indeed, consider the (high level) procedure depicted in Algorithm 1.

Proposition 1. *Algorithm 1 has constant success probability* ≈ 0.5 *and time complexity* $T = O\left((T_{CF} + n^3) \cdot \sqrt{\binom{n}{k}}\right)$, *where* T_{CF} *is the time to compute one canonical form. For* $k = Rn$, *with* R *a constant in* $[0; 1]$, *it has asymptotic time complexity* $2^{n \cdot \tau(R) \cdot \left(1 + o(1)\right)}$, *where* $\tau(R) = \frac{1}{2} \cdot h_2(R)$ *and* h_2 *is the binary entropy function.*

Proof. Because of the birthday paradox, the success probability of the algorithm is approximately 0.5. The time complexity is derived by considering that, for each candidate J (resp., J') the probability that SF fails is constant (and, for growing 1, tends to 1). Also,

Algorithm 1: Using canonical forms to attack code equivalence

Input: matrices $G_0, G_1 \in \mathbb{F}_q^{k\times n}$
Output: failure or equivalence between G_0 and G_1

// Create first list
1 Set $\mathcal{L} = \varnothing$;
2 **while** $|\mathcal{L}| < \sqrt{\binom{n}{k}}$ **do**
3   Sample $J \subseteq \{1, \cdots, n\}$ of size k;
4   Compute $A = \mathrm{SF}(G_0, J)$;
5   **if** $A \neq \perp$ **then**
6   Compute $B = \mathsf{CF}(A)$;
7   **if** $B \neq \perp$ **then**
8   Add $\big(J, B\big)$ to $\mathcal{L}$;

// Create second list
9 Set $\mathcal{L}' = \varnothing$;
10 **while** $|\mathcal{L}'| < \sqrt{\binom{n}{k}}$ **do**
11   Sample $J' \subseteq \{1, \cdots, n\}$ of size k;
12   Compute $A' = \mathrm{SF}(G_1, J')$;
13   **if** $A' \neq \perp$ **then**
14   Compute $B' = \mathsf{CF}(A')$;
15   **if** $B' \neq \perp$ **then**
16   Add $\big(J', B'\big)$ to $\mathcal{L}'$;

// Find collisions and reconstruct the equivalence
17 Search for collisions, i.e., pairs $(J, B) \in \mathcal{L}$, $(J', B') \in \mathcal{L}'$ such that $B = B'$;
18 If a collision is found, reconstruct the equivalence

because of the properties in Definition 3, the probability $\gamma(n, k, q)$ that CF fails can be is extremely small and it can be neglected. So, in practice, the amount of times instructions 3–7 are executed is essentially equal to $\sqrt{\binom{n}{k}}$. For each candidate J, we compute one systematic form (with cost n^3) and one canonical form (with cost T_{CF}). For the asymptotics, we consider that

$$\frac{1}{n}\log_2(T) = \frac{1}{2n}\log_2\left(\binom{n}{k}\right) + \frac{1}{n}\log_2(n^3 + T_{\mathsf{CF}}).$$

Since T_{CF} is a polynomial function of both n and q, the quantity $\frac{1}{n}\log_2(n^3 + T_{\mathsf{CF}})$ is in $o(1)$. Finally, we consider that $\binom{n}{k} = \binom{n}{Rn} = 2^{n\cdot h(R)\big(1+o(1)\big)}$.

Remark 1. To store each list element, several approaches. For instance, each J (resp., J') can be represented with n bits or (if sets are generated using a PRG), we can store the employed seed. For each B (resp., B'), we can use $\mathsf{poly}(n, q)$ memory (since a canonical form is a matrix) or, more conveniently, we can store only its digest (and, then, search for collisions in the digests values).

Interestingly, our algorithm does not depend significantly on the size of the finite field: the only effect of the finite field size is on the cost of computing Gaussian eliminations and canonical forms. So, the dependence is only marginal and entirely disappears when considering the asymptotic regime.

Appendix A compares Algorithm 1 with other known attacks.

7 Defining Canonical Forms for ϵ

This section shows how one can define the canonical forms for different cases of $(\delta_r, \pi_r, \delta_c, \pi_c)$. The approaches in Section 7.1 and Section 7.2 have been implicitly used in [10] and [1], while the approaches in other subsections are new. The number of field operations taken by each algorithm presented in this section is apparently polynomial in q, k, and n in the worst case.

7.1 When $(\delta_r, \pi_r, \delta_c, \pi_c) = (\{I_k\}, \{I_k\}, \{I_{n-k}\}, \mathsf{P}_{n-k})$

One can define the canonical form of any equivalence class as the matrix in which the columns are sorted w.r.t. a total ordering defined on $\mathbb{F}_q^k$. For example, the total ordering can be the lexicographic order. In this way, for every equivalence class the canonical form is well-defined, and from any matrix the canonical form can be derived easily by simply sorting the columns.

7.2 When $(\delta_r, \pi_r, \delta_c, \pi_c) = (\{I_k\}, \{I_{n-k}\}, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$

One can define the canonical form as the unique matrix of which 1) the first nonzero entry of each nonzero column is 1 and 2) the columns are sorted w.r.t. a total ordering defined on $\mathbb{F}_q^k$. In this way, for every equivalence class the canonical form is well-defined. Given any matrix, to compute the canonical form, one can first scale each nonzero column to make the first nonzero entry become 1 and then sort the columns.

7.3 When $(\boldsymbol{\delta_r}, \boldsymbol{\pi_r}, \boldsymbol{\delta_c}, \boldsymbol{\pi_c}) = (\{\boldsymbol{I_k}\}, \mathsf{P}_{\boldsymbol{k}}, \{\boldsymbol{I_{n-k}}\}, \mathsf{P}_{\boldsymbol{n-k}})$

Given a vector v, we define multiset(v) as the multiset formed by the entries in v. Consider a partial ordering defined on $\mathbb{F}_q^{n-k}$ such that

- for any distinct $v, v' \in \mathbb{F}_q^{n-k}$, either $v \leq v'$ or $v' \leq v$ if multiset$(v) \neq$ multiset(v'), and
- for any distinct $v, v' \in \mathbb{F}_q^{n-k}$, neither $v \leq v'$ or $v' \leq v$ holds if multiset$(v) =$ multiset(v').

Given such a partial ordering on $\mathbb{F}_q^{n-k}$ and a total ordering defined on $\mathbb{F}_q^k$, for any equivalence class, we define the canonical form as the matrix of which the rows are sorted w.r.t. to the partial ordering and the columns are sorted w.r.t. the total ordering, 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) using the partial ordering in all matrices in the equivalence class.

Given any matrix, one can derive the corresponding canonical form by first sorting the rows using the partial ordering and then sorting the columns using the total ordering. Whether the corresponding canonical form exists can be easily detected during the step of sorting rows.

Assume that there is a total ordering defined over the set of all possible multisets of size $n-k$. One can define the partial ordering as follows: for any distinct $v, v' \in \mathbb{F}_q^{n-k}$,

$$v \leq v' \iff \text{multiset}(v) \leq \text{multiset}(v').$$

There are multiple ways to define the total ordering for multisets. One reasonably efficient approach is as follows: for any distinct multiset S and S', let u be the vector obtained by sorting the elements in S, let u' be the vector obtained by sorting the elements in S', then

$$S \leq S' \iff u \leq u',$$

where u and u' are compared using a total ordering defined on $\mathbb{F}_q^{n-k}$.

Similarly, one can also define the canonical form as the result of first sorting the columns using a partial ordering defined on $\mathbb{F}_q^k$ and then sorting the rows using a total ordering defined on $\mathbb{F}_q^{n-k}$.

A lower bound on the success probability of the algorithm above is shown in Appendix E.

7.4 When $(\delta_r, \pi_r, \delta_c, \pi_c) = (\{I_k\}, \mathsf{P}_k, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$

Let $p_0, \ldots, p_{d-1} \in \mathbb{Z}$ be d distinct integers such that

- $0 < p_0 < p_1 < \cdots < p_{d-1} < q - 1$,
- $\gcd(p_i, q-1) = 1$ for all i, and
- $\mathrm{char}(\mathbb{F}_q)$ is not a factor of any p_i.

For efficiency of the algorithm that will be introduced below, it is reasonable to select p_i's as the smallest d integers that satisfies the criteria. We define the canonical form of a equivalence class as the matrix such that

1. for each nonzero column v, the first nonzero entry of the vector
$$(\sum_i v_i^{p_0}, \ldots, \sum_i v_i^{p_{d-1}})$$
is 1, and
2. the rows and columns are sorted in the way described in Section 7.3,

if such a matrix exists.

To derive the systematic form of a matrix, one can carry out one step to ensure that the first constraint above holds and then another step to ensure that the second constraint holds. The second step simply runs the algorithm in Section 7.3. The first step can be carried out by scaling each nonzero column independently. For each nonzero column $v \in \mathbb{F}_q^k$, we compute a list of d vectors $v^{(0)}, \ldots, v^{(d-1)}$, where each $v^{(j)}$ is the unique multiple of v such that $\sum_i (v_i^{(j)})^{p_j} = 1$, or 0 if $\sum_i v_i^{p_j} = 0$. If $v^{(j)} = 0$ for all j, $\perp$ will be returned. Otherwise, the column is replaced by the first $v^{(j)}$ which is non-zero.

7.5 When $(\boldsymbol{\delta_r}, \boldsymbol{\pi_r}, \boldsymbol{\delta_c}, \boldsymbol{\pi_c}) = (\mathsf{D}_k, \mathsf{P}_k, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$

To derive the canonical form given a matrix A, for each j such that column j consists of only nonzero elements, we define $A^{(j)}$ as the result of scaling rows of A so that column j of $A^{(j)}$ is $(1, 1, \ldots, 1)$. If every column contains 0, $\perp$ will be returned. Then, for each $A^{(j)}$, the algorithm in Section 7.4 is applied to obtain a matrix $A^{[j]}$ if $\perp$ is not returned. Finally, the canonical form is simply defined as the smallest $A^{[j]}$ w.r.t. a total ordering defined on $\mathbb{F}_q^{k\times(n-k)}$.

Proposition 2. *Given $A, B \in \mathbb{F}_q^{k\times(n-k)}$ satisfying $B = D_r \cdot A \cdot D_c$ for some $D_r, D_c \in \mathsf{D}_{n-k}$, such that column j of A (and B) consists of only nonzero elements. Then $B^{(j)} = A^{(j)} \cdot D'_c$ for some $D'_c \in \mathsf{D}_{n-k}$.*

Proof. Let the elements on the main diagonal of D_r be $x_0, \ldots x_k$. Let the elements on the main diagonal of D_c be $y_0, \ldots y_{n-k}$. Then column i of $A^{(j)}$ is $(A_{0,i} \cdot A_{0,j}^{-1}, \ldots, A_{k-1,i} \cdot A_{k-1,j}^{-1})$, while column i of $B^{(j)}$ is

$$(x_0 \cdot A_{0,i} \cdot y_i \cdot A_{0,j}^{-1} \cdot x_0^{-1} \cdot y_j^{-1}, \ldots, x_{k-1} \cdot A_{k-1,i} \cdot y_i \cdot A_{k-1,j}^{-1} \cdot x_{k-1}^{-1} \cdot y_j^{-1}).$$

Proposition 3. *Given $A, B \in \mathbb{F}_q^{k\times(n-k)}$ satisfying $A = P_r \cdot B \cdot P_c$ for some $P_r, P_c \in \mathsf{P}_{n-k}$, such that column j of A consists of only nonzero elements. Let j' be the row index of 1 in column j of P_c. In other words, the column permutation represented by P_c maps column j' to column j. Then $A^{(j)} = P_r \cdot B^{(j')} \cdot P_c$.*

Combining the two propositions, we have

$$\begin{aligned} B = P_r \cdot D_r \cdot A \cdot D_c \cdot P_c &\implies P_r^{-1} \cdot B \cdot P_c^{-1} = D_r \cdot A \cdot D_c \\ &\implies P_r^{-1} \cdot B^{(j')} \cdot P_c^{-1} = A^{(j)} \cdot D'_c \\ &\implies B^{(j')} = P_r \cdot A^{(j)} \cdot D'_c \cdot P_c. \end{aligned}$$

Therefore, the set of $A^{[j]}$'s is the same as the set of $B^{[j]}$'s, and we conclude that the algorithm leads to the same output for any input matrix in a equivalence class.

8 Potential Parameter Sets for LEQ

The crucial quantity to calculate, in this work, is the size of the signature for the scheme. Hence, we describe it carefully as follows. First, note that, per Fiat-Shamir, a LEQ signature is composed of two separate pieces: the responses for each round of the Sigma protocol, and the hash digest which yields the challenges. The former can be separated into two subcases, depending whether the response is truly random or not, since in the former case, it can be compactly represented via a short seed. Thus, as a first formula, we describe the size of the signature as

$$C_{\text{iso}} + C_{\text{seeds}} + \ell_{\text{salt}} + \ell_{\text{digest}}, \tag{1}$$

where $C_{\text{iso}}, C_{\text{seeds}}, \ell_{\text{salt}}$ and ℓ_{digest} stand for, respectively, the cumulative costs of transmitting the isometries, the seeds, the salt and the digest. ℓ_{salt} and ℓ_{digest} are specified in Table 4. The other two costs depend heavily on how the objects are represented, according to what algorithm is used for canonical form, and what structure is used to communicate the seeds. For instance, the original LESS signature scheme [3] transmitted entire monomial matrices, while the LESS specification document [1] uses a structure similar to the one described in Section 7.2, and so on. These are summarized below, keeping in mind that the total number of isometries to be transmitted is given by the scheme parameter w.

Method	Cost
[3]	$\lceil (w \cdot n \cdot (\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil))/8 \rceil$
7.2 ([1, 10])	$\lceil (w \cdot k \cdot (\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil))/8 \rceil$
7.4	$\lceil (w \cdot (n + k \cdot \lceil \log_2(q-1) \rceil))/8 \rceil$
7.5	$\lceil (w \cdot n)/8 \rceil$

Table 1. C_{iso}, cost of transmitting isometries, in bytes.

Regarding the transmission of seeds, this hinges mainly on whether a "seed tree" is used or not. The cost is summarized in Table 2.

Summing up the results from the tables above, we are able to propose a variety of practical instances for our scheme.

Method	Cost
No seed tree	$(t-w)\cdot\ell_{\text{seed}}$
Seed tree	$(2^{\lceil\log_2 w\rceil}+w\cdot(\lceil\log_2 t\rceil-\lceil\log_2 w\rceil-1))\cdot\ell_{\text{seed}}$

Table 2. C_{seeds}, cost of transmitting seeds, in bytes. The formula in the last row is from [7, Proposition 1]. ℓ_{seed} is specified in Table 4.

q	k	n	s	t	w	$\log_2\sqrt{\binom{n}{k}}$	alg	pk	sig	tree	type
127	126	252	2	247	30	123.84	7.3	13939	2481	yes	P
							7.5				L
							7.4		5789	yes	L
							7.2 ($[1,10]$)		8624	yes	L
127	200	400	2	759	33	197.67	7.3	35074	5658	yes	P
							7.5				L
							7.4		11433	yes	L
							7.2 ($[1,10]$)		17208	yes	L
127	274	548	2	1352	40	271.56	7.3	65792	10036	yes	P
							7.5				L
							7.4		19626	yes	L
							7.2 ($[1,10]$)		30586	yes	L

Table 3. Potential parameter sets for LEQ.

Table 3 shows some potential parameter sets for LEQ, along with the parameter sets with $s=2$ of LESS; the latter are reported in the bottom row of each cell. The column "alg" indicates which choice of $(\pi_r,\pi_c,\delta_r,\delta_c)$ is used, along with how the canonical form is defined. The columns "pk" and "sig" indicate the corresponding sizes for public key and signature, in bytes, respectively. The column "tree" indicates whether a seed tree is used or not. The column "type" indicates whether the parameter set is based on (CF-)PEP or (CF-)LEP.

The column $\log_2\sqrt{\binom{n}{k}}$ indicates the largest factor in the cost of of the attack in Section 6. 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.

	Category 1	Category 3	Category 5
ℓ_{digest}	32	48	64
ℓ_{salt}	32	48	64
ℓ_{seed}	16	24	32

Table 4. The lengths in bytes of the digest, the salt, and each seed in the seed tree in [1, Table 2].

References

1. Marco Baldi, Alessandro Barengh, Luke Beckwith, Jean-François Biasse, Andre Esser, Kris Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani O. Saarinen, Paolo Santini, and Robert Wallace. LESS: Linear equivalence signature scheme, 2023. URL: `https://www.less-project.com/LESS-2023-08-18.pdf`.
2. Magali Bardet, Ayoub Otmani, and Mohamed Saeed-Taha. Permutation code equivalence is not harder than graph isomorphism when hulls are trivial. In *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 2464–2468. IEEE, 2019.
3. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. 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*, pages 23–43. Springer, 2021.
4. 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.
5. 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.
6. Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. 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*, pages 45–65. Springer, 2020.
7. Joan Boyar, Simon Erfurth, Kim S. Larsen, and Ruben Niederhagen. Quotable signatures for authenticating shared quotes, 2023. URL: `https://arxiv.org/pdf/2212.10963.pdf`.
8. Jeffrey Leon. Computing automorphism groups of error-correcting codes. *IEEE Transactions on Information Theory*, 28(3):496–511, 1982.
9. NIST. Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process, 2023. URL: `https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals`.
10. Edoardo Persichetti and Paolo Santini. A new formulation of the linear equivalence problem and shorter less signatures, 2023. URL: `https://eprint.iacr.org/2023/847`.
11. Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, 2000.

12. Nicolas Sendrier and Dimitris E Simos. 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*, pages 203–216. Springer, 2013.

A Comparison with other solvers

We now consider other solvers for the code equivalence problems, and compare their running time with the one of our algorithm.

- *SSA*, [11]: this algorithm can efficiently solve PEP when the hull of the considered codes is small. Instead, it takes exponential time when self dual codes (i.e., codes whose hull is equal to the code itself) are considered; 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 [12], SSA can also be used to solve LEP; however, whenever $q \geq 5$, the reduction maps any code into a self-dual code with dimension k (so, it has time complexity $O(q^k)$);
- *BOS*, [2]: this algorithm reduces PEP to graph isomorphism. While it is efficient for codes with small hull, it has super-exponential running time $T_{\text{BOS}} = O(n^{Rn})$ when self-dual codes are considered;
- *Leon*, *Beullens*, *BBPS*, [4, 5, 8]: 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, at 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[5][6] Hence, for these algorithms we consider a time complexity given by $T = O\left(2^{\tau_{\text{Prange}}(R,q)\left(1+o(1)\right)}\right)$, where

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

where h_q denotes the q-ary entropy function (for details on how this is derived, see Appendix C).

We are now ready to compare the above algorithms with Algorithm 1; to this end, consider Figure 1. First, SSA and BOS have been omitted from the comparison since their performances 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 codewords finding. Instead, when q grows, our algorithm becomes much more competitive and becomes faster then Prange.

We observe that the time complexity of Prange deteriorates. This is due to the fact that, as q grows, the minimum distance tends to $n-k$ (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 $\tau_{\text{Prange}}(R,q) \to h_2(R)$. Notice that this complexity coefficient is twice the one which is achieved by our algorithm.

B Asymptotic cost of Prange's ISD

A random code of length n and rate R has minimum distance $d = \delta n$, where $\delta = h_q^{-1}(1-R)$ (where h_q is the q-ary entropy function). The

[5] 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.

[6] Even though this provides only a very broad estimate of the actual time complexity, this allows us to compare with these algorithms in simple and concise way. Remember that cryptanalysis is not the focus of this paper and the aim of this section is 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.

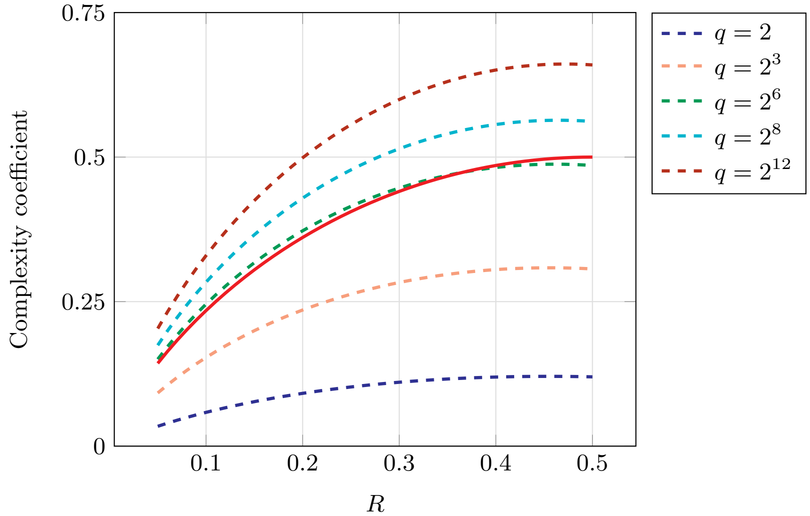

Fig. 1. Comparison between the complexity coefficients for Prange (dashed lines) and Algorithm 1 (continuous red line), as a function of the code rate.

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 do not consider it. Then, for the algorithm we assume a complexity coefficient given by

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

C Maximum number of permutations associated to multiset

We study the following problem: find the maximal value that $|\mathcal{M}(a)|$ can have, when a is a length-m vector over $\mathbb{F}_q$. Let ℓ_i denote the number of entries of a with value equal to $x_i \in \mathbb{F}_q$ (we are writing the field as $\{x_0 = 0, x_1 = 1, x_2, \cdots, x_{q-1}\}$); notice that it must be

$\sum_{i=0}^{q-1} \ell_i = m$. The values ℓ_i allow us to take into account the number of permutations with repetitions, so that

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

Maximizing $|\mathcal{M}(a)|$ implies 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 4. *For any $(\ell_0, \cdots, \ell_{q-1}) \in \mathbb{N}^q$ such that $\sum_{i=0}^{q-1} \ell_i = m$, it holds that*

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

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

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

$$\forall x, y \in \mathbb{N}, \text{ it holds} \quad y!x! > (y-1)!(x+1)! \text{ if } y - x > 2. \tag{2}$$

Indeed, let us consider an arbitrary tuple $(\ell_0, \cdots, \ell_{q-1})$, summing to m, and assume there are two values ℓ_j, ℓ_u such that $\ell_j - \ell_u > 2$. Then, there exists a new tuple $(\ell'_0, \cdots, \ell'_{q-1})$ such that $\ell'_u = \ell_u$ if $i \neq j, u$, $\ell'_j = \ell_j - 1$ and $\ell'_u = \ell_u + 1$. First, this configuration is valid since the sum of all the ℓ'_i is still equal to m. Also, because of (2), 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, $v = \left\lfloor \frac{m}{q} \right\rfloor$ and $v + 1$. Let z denote the number of entries with value v: since it must be $vz + (q - z)(v + 1) = m$, we find $z = q(v + 1) - m$. So, the number of entries with value equal to $v + 1$ is $q - z = m - qv$. $\square$

It follows that

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

D Other ways to define canonical forms

This section shows some other ways to define canonical forms.

D.1 When $(\delta_r, \pi_r, \delta_c, \pi_c) = (\{I_k\}, \mathsf{P}_k, \{I_{n-k}\}, \mathsf{P}_{n-k})$

Algorithm 2: Canonical form

Input: matrix $A \in \mathbb{F}_q^{k\times(n-k)}$
Output: canonical form $B \in \mathbb{F}_q^{k\times(n-k)}$

for $i = 1, \cdots, k$ **do**
  Compute $M_i = \text{multiset}(v_i)$; `/* Multiset of i-th row */`
Compute $J = \{i \in [1; k] \mid M_i \text{ is unique}\}$;
`// Split rows into two groups, and sort them`
Set $C :=$ matrix whose rows are indexed by J;
Set $C' :=$ matrix whose rows are not indexed by J;
Sort C and C' so that the multisets of the rows are sorted;
`// Apply column permutation`
Let $P_c =$ column permutation so that columns of C are lexicographically sorted Sort C' according to P_c;
`// Sort rows of C' and output canonical form`
Sort the rows of C' so that they are in lexicograph order;
Set $B = \begin{bmatrix} C \\ C' \end{bmatrix}$;
return B;

In Algorithm 2 we describe a novel computation of canonical forms. We illustrate the functioning of the algorithm with an example.

Example 1. Let $A = \begin{bmatrix} 5&7&8&2&0&1&3 \\ 1&0&0&4&5&9&2 \\ 0&0&1&9&2&4&5 \\ 2&3&0&1&8&2&6 \\ 8&7&5&3&1&0&2 \\ 6&3&4&5&2&3&1 \end{bmatrix}$. We compute the multisets, and get

$$M_1 = \{0, 1, 2, 3, 5, 7, 8\},$$
$$M_2 = \{0, 0, 1, 2, 4, 5, 9\},$$

$$M_3 = \{0, 0, 1, 2, 4, 5, 9\},$$

$$M_4 = \{0, 1, 2, 2, 3, 6, 8\},$$

$$M_5 = \{0, 1, 2, 3, 5, 7, 8\},$$

$$M_6 = \{1, 2, 3, 3, 4, 5, 6\}.$$

First, we observe that $M_2 = M_3$ and $M_1 = M_5$, while M_4 and M_6 are unique. So, $J = \{4, 6\}$ and $C = \begin{bmatrix} v_4 \\ v_6 \end{bmatrix}$ and $C' = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_5 \end{bmatrix}$. We now sort the rows of C: since $M_4 < M_6$, the sorted C is identical to the initial C, hence

$$C = \begin{bmatrix} v_4 \\ v_6 \end{bmatrix} = \begin{bmatrix} 2\,3\,0\,1\,8\,2\,6 \\ 6\,3\,4\,5\,2\,3\,1 \end{bmatrix}.$$

For C', we notice that $M_2 = M_3 < M_1 = M_5$, so we sort the rows of C' as

$$C' = \begin{bmatrix} v_2 \\ v_3 \\ v_1 \\ v_5 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,4\,5\,9\,2 \\ 0\,0\,1\,9\,2\,4\,5 \\ 5\,7\,8\,2\,0\,1\,3 \\ 8\,7\,5\,3\,1\,0\,2 \end{bmatrix}.$$

Now, we sort the columns of C and C', applying the following permutation:

$$P_c = \begin{pmatrix} 1\,2\,3\,4\,5\,6\,7 \\ 4\,5\,1\,2\,7\,3\,6 \end{pmatrix}.$$

This way, we get

$$C = \begin{bmatrix} 0\,1\,2\,2\,3\,6\,8 \\ 4\,5\,3\,6\,3\,1\,2 \end{bmatrix}, \qquad C' = \begin{bmatrix} 0\,4\,9\,1\,0\,2\,5 \\ 1\,9\,4\,0\,0\,5\,2 \\ 8\,2\,1\,5\,7\,3\,0 \\ 5\,3\,0\,8\,7\,2\,1 \end{bmatrix}.$$

Now, we conclude by sorting the rows of C': it is enough to swap rows 3 and 4. Appending C and C', we get the canonical form of A.

D.2 When $(\boldsymbol{\delta_r}, \boldsymbol{\pi_r}, \boldsymbol{\delta_c}, \boldsymbol{\pi_c}) = (\{\boldsymbol{I_k}\}, \mathsf{P}_k, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$

Given a matrix A, one can define the canonical form as the output of the following two-step algorithm.

1. The first step works as follows. First, let S be the set of row indices such that the rows consist of only nonzero elements. If $S = \emptyset$, $\perp$ will be returned. Otherwise, for each $i \in S$, compute $A^{(i)}$ as the result of scaling columns in A, such that row i of $A^{(i)}$ is $(1, 1, \ldots, 1)$.
2. The second step simply applies the algorithm in Section 7.3 to each $A^{(i)}$ to obtain $A^{[i]}$. Then, the smallest $A^{[i]}$ w.r.t. a total ordering on $\mathbb{F}_q^{k\times(n-k)}$ is returned as the canonical form.

D.3 When $(\boldsymbol{\delta_r}, \boldsymbol{\pi_r}, \boldsymbol{\delta_c}, \boldsymbol{\pi_c}) = (\mathsf{D}_k, \mathsf{P}_k, \mathsf{D}_{n-k}, \mathsf{P}_{n-k})$

Given a matrix A, one can define the canonical form as the output of the following two-step algorithm.

1. The first step works as follows. First, let S_r be the set of row indices such that the rows consist of only nonzero elements, and let S_c be the set of column indices such that the columns consist of only nonzero elements. If $S_r = \emptyset$ or $S_c = \emptyset$, $\perp$ will be returned. Otherwise, for each $(i, j) \in S_r \times S_c$, compute $A^{(i,j)}$ as the result of scaling columns and rows in A, such that row i and column j of $A^{(i,j)}$ are both $(1, 1, \ldots, 1)$.
2. The second step simply applies the algorithm in Section 7.3 to each $A^{(i,j)}$ to obtain $A^{[i,j]}$. Then, the smallest $A^{[i,j]}$ w.r.t. a total ordering on $\mathbb{F}_q^{k\times(n-k)}$ is returned as the canonical form.

E A Lower Bound on Success Probability

Proposition 5. *Let $A \xleftarrow{\$} \mathbb{F}_q^{k\times(n-k)}$; then, the canonical form of A exists with probability is at least $\gamma(n,k,q) = \prod_{i=1}^{k-1} 1 - \frac{iM}{q^{n-k}}$, 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 will use a_i to indicate the i-th row of A and $\mathcal{M}(a_i)$ to denote the set of vectors whose multiset is equal to that of a_i. The probability that A admits the canonical form considered in this section can be lower bounded with a simple iterative reasoning.

Let us consider the first two rows of A: regardless of a_1, they will have different multisets if multiset$(a_2) \neq$ multiset(a_1). So, the probability that this pair of rows is valid is

$$\begin{aligned}\Pr\left[\{a_1, a_2\} \text{ is valid}\right] &= \sum_{a_i \in \mathbb{F}_q^n} \Pr\left[a_2 \text{ is valid} \mid a_1\right] \cdot \Pr\left[a_1\right] \\ &= \frac{1}{q^{n-k}} \sum_{a_i \in \mathbb{F}_q^n} \left(1 - \frac{|\mathcal{M}(a_1)|}{q^{n-k}}\right),\end{aligned}$$

where $\Pr\left[a_1\right]$ is the probability that the first row is equal to a_1 and is equal to $q^{-(n-k)}$ for each a_1 (since A is sampled according to the uniform distribution). Now, let M such that $|\mathcal{M}(a_1) \leq M|$ for each possible a_1: we get

$$\Pr\left[\{a_1, a_2\} \text{ is valid}\right] \geq \frac{1}{q^{n-k}} \sum_{a_i \in \mathbb{F}_q^n} \left(1 - \frac{M}{q^{n-k}}\right) = 1 - \frac{M}{q^{n-k}}.$$

We now consider a_3 and, with analogous reasoning, get that for any valid pair $\{a_1, a_2\}$, a new vector a_3 is valid only if it does not belong to $\mathcal{M}(a_1) \cup \mathcal{M}(a_2)$. Using the upper bound M for both sets, we get that 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:

$$\gamma(n, k, q) = \prod_{i=1}^{k-1} 1 - \frac{iM}{q^{n-k}}.$$

Now, we simply 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, $|\mathcal{M}(a_1)| = (n - k)!$ holds only if a_1 has all distinct entries while, otherwise $|\mathcal{M}(a_1)|$ contains fewer vectors. When $n - k > q$, we can refine the bound by taking into account that each a_i must necessarily have some repeated entries. The proof on how M is derived, in this case, is reported in Appendix C. $\square$