An Improved Algorithm for Code Equivalence

Julian Nowakowski

Ruhr University Bochum, Bochum, Germany

Abstract. We study the linear code equivalence problem (LEP) for linear $[n, k]$-codes over finite fields $\mathbb{F}_q$. Recently, Chou, Persichetti and Santini gave an elegant heuristic algorithm that solves LEP over *large* finite fields (with $q = \Omega(n)$) in time $2^{\frac{1}{2}\,\mathrm{H}\left(\frac{k}{n}\right)n}$, where $\mathrm{H}(\cdot)$ denotes the binary entropy function. However, for *small* finite fields, their algorithm can be significantly slower. In particular, for fields of constant size $q = \mathcal{O}(1)$, its runtime increases by an exponential factor $2^{\Theta(n)}$.
We present an improved and provably correct version of their algorithm, which achieves the desired runtime of $2^{\frac{1}{2}\,\mathrm{H}\left(\frac{k}{n}\right)n}$ for *all* finite fields of size $q \geq 7$. For a wide range of parameters, this improves over the runtime of all previously known algorithms by an exponential factor.

Keywords: Linear Code Equivalence Problem · Canonical Form Functions

1 Introduction

Digital signatures schemes based on so-called *equivalence problems* have recently emerged as promising candidates for post-quantum security. Examples of such schemes include LESS [BMPS20], HAWK [DPPv22] and MEDS [CNP+23], which are based on the *linear code equivalence problem*, the *lattice isomorphism problem*, and the *matrix code equivalence problem*, respectively. In this work, we focus on the linear code equivalence problem (LEP).

LEP is an important problem in coding theory. With the recent introduction of LESS, LEP has gained significant interest in cryptography [Beu20, PS23, BBPS23, CPS23]. In a nutshell, the problem is defined as follows: Given generator matrices $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_q^{k \times n}$ of two linear $[n, k]$-codes $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$, one is asked to compute a linear, Hamming weight preserving map $\mathbf{Q}$ that bijectively maps $\mathcal{C}_1$ to $\mathcal{C}_2$ (provided such a map exists). Such maps $\mathbf{Q}$ are precisely those linear maps, that permute the coordinates of the codewords $\mathbf{c} \in \mathcal{C}_1$, and additionally multiply them by units from the underlying field $\mathbb{F}_q$. These maps are called *monomials.*

1.1 Previous Work

Support Splitting. The *permutation equivalence problem* (PEP) is a variant of LEP, in which one is asked to find a permutation, mapping $\mathcal{C}_1$ to $\mathcal{C}_2$ (again, provided it exists). Curiously, PEP is easy on average, but seems to be hard in the worst case. Indeed, Sendrier's famous *support splitting algorithm* (SSA) [Sen00] solves random PEP instances in polynomial time. However, there are worst-case instances (in which $\mathcal{C}_1$ and $\mathcal{C}_2$ are so-called *weakly self-dual codes*), for which SSA requires exponential time.

Since there is a reduction from LEP to PEP [SS13], one can try solving LEP by first reducing it to PEP and then using SSA. For fields of size $q \leq 4$ this approach works just fine. Hence, random LEP instances over $\mathbb{F}_2$ $\mathbb{F}_3$ and $\mathbb{F}_4$ are easy. However, for fields of size $q \geq 5$, the reduction results in weakly self-dual codes, and thus in an exponential

E-mail: julian.nowakowski@rub.de (Julian Nowakowski)

runtime for SSA. It is conjectured that this state-of-the-art of SSA cannot improved, and that random LEP instances over fields of size $q \geq 5$ are hard.

Finding Low-weight Codewords. An alternative approach for solving LEP is based on computing low-weight codewords. It was first suggested by Leon [Leo82], and is based on the following simple observation: Let us fix some parameter w, and let $L_1(w) \subset \mathcal{C}_1$ and $L_2(w) \subset \mathcal{C}_2$ denote sets of all codewords in $\mathcal{C}_1$ and $\mathcal{C}_2$ of weight at most w. Since monomials preserve Hamming weight, any monomial that maps $\mathcal{C}_1$ to $\mathcal{C}_2$ has to map $L_1(w)$ to $L_2(w)$. Conversely, if w is only slightly larger than the weight of the minimal-weight codeword in $\mathcal{C}_1$, then any monomial that maps $L_1(w)$ to $L_2(w)$ will – with decent probability – map $\mathcal{C}_1$ to $\mathcal{C}_2$. To solve LEP, Leon thus suggests the following simple two step approach: First compute the sets $L_1(w)$ and $L_2(w)$. Then compute a monomial $\mathbf{Q}$, mapping $L_1(w)$ to $L_2(w)$. Computing $L_1(w)$ and $L_2(w)$ takes time exponential in n, computing $\mathbf{Q}$ can be done in time polynomial in $|L_1(w)| = |L_2(w)|$.

Recently, first Beullens [Beu20], and afterwards Barenghi, Biasse, Persichetti and Santini (BBPS) [BBPS23] have introduced significantly improved variants of Leon's algorithm, following a similar two-step, low-weight codeword finding based approach. In many parameter regimes, Beullens and BBPS improve over Leon the runtime by an exponential factor. As a result, up until very recently, BBPS was in most parameter regimes the fastest algorithm for solving LEP.

Canonical Form Functions. A very recent work by Chou, Persichetti and Santini (CPS) [CPS23] introduced a completely different approach for solving LEP, based on so-called *canonical form functions*.

In their work, CPS define a novel equivalence relation for linear codes, which we denote by $\overset{\mathsf{LRL}}{\sim}$. Suppose we have two codes $\mathcal{C}_1$ and $\mathcal{C}_2$ with generator matrices $\mathbf{G}_1 = [\mathbf{I}_k \mid \mathbf{A}_1], \mathbf{G}_2 = [\mathbf{I}_k \mid \mathbf{A}_2] \in \mathbb{F}_q^{k \times n}$, where $\mathbf{I}_k$ denotes the k-dimensional idenity matrix. We call $\mathcal{C}_1$ and $\mathcal{C}_2$ equivalent with respect to $\overset{\mathsf{LRL}}{\sim}$, if there exist monomials $\mathbf{Q}_r$, $\mathbf{Q}_c$ such that $\mathbf{A}_2 = \mathbf{Q}_r \cdot \mathbf{A}_1 \cdot \mathbf{Q}_c$. (Here we identify the monomials with their corresponding transformation matrices.) In a nutshell, a canonical form function for $\overset{\mathsf{LRL}}{\sim}$ is an algorithm that takes a generator matrix $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}]$ of some code $\mathcal{C}$ as input, and outputs a generator matrix $\mathbf{G}^* = [\mathbf{I}_k \mid \mathbf{A}^*]$ of a *canonical representative* $\mathcal{C}^*$ of the equivalence class of $\mathcal{C}$ (with respect to $\overset{\mathsf{LRL}}{\sim}$).

Importantly, CPS allow canonical form functions to *fail*. That is, instead of *always* outputting the desired matrix $\mathbf{G}^*$, a canonical form function may (with some failure probability) also output an error symbol $\perp$.

Suppose we have a canonical form function CF for $\overset{\mathsf{LRL}}{\sim}$. Let γ denote the success probability of CF, i.e., let γ denote the probability that, on input $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}] \in \mathbb{F}_q^{k \times n}$ with uniformly random $\mathbf{A} \in \mathbb{F}_q^{k \times (n-k)}$, CF does not output $\perp$. CPS give a transformation, that (heuristically) turns any canonical form function CF into an LEP algorithm with runtime $\gamma^{-1/2} \cdot 2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})n}$, where $\mathrm{H}(\cdot)$ denotes the binary entropy function. In particular, for canonical form functions with (at least) constant success probability $\gamma = \Omega(1)$, the CPS transformation yields a heuristic LEP algorithm with runtime roughly $2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})n}$.

As Figure 1 shows, if such a canonical form function with (at least) constant success probability exists, then the resulting LEP algorithm would – for all fields of size $q \geq 2^7$ – improve over the previously best algorithm of BBPS by an exponential factor.[1]

Unfortunately, finding canonical form functions with (at least) constant success probability is challenging: CPS give a canonical form function that achieves constant success probability only for large $q = \Omega(n)$. However, for constant $q = \mathcal{O}(1)$, its success probability

[1]We computed the runtime T_{BBPS} in Figure 1 using the estimator from `https://github.com/paolo-santini/LESS_project/blob/main/attacks/LEP/cost.sage`.

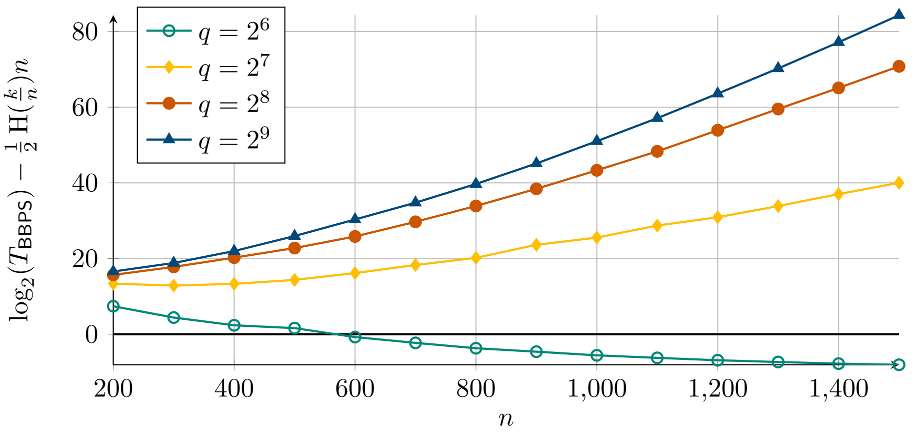

Figure 1: Comparison between runtime T_{BBPS} of BPPS' algorithm and the canonical form function based algorithm – assuming the underlying canonical form function has (at least) constant success probability. Results are for codes of rate $\frac{k}{n} = \frac{1}{2}$ over various finite fields $\mathbb{F}_q$. For codes of rate $\frac{k}{n} \neq \frac{1}{2}$, the improvements of the canonical form function based algorithm over BPPS would be even higher.

is exponentially small in n.[2] Hence, for constant $q = \mathcal{O}(1)$, the runtime of the resulting LEP algorithm increases by a factor exponential in n.

1.2 Our Contributions

We introduce a novel canonical form function $\mathsf{CF}_{\mathsf{New}}$ that – for all finite fields of size $q \geq 7$ and codes of constant rate[3] – has success probability $1 - \mathcal{O}(n^{-1})$. Together with the seminal results of CPS, this immediately results in a $2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})n}$-time algorithm for LEP. As shown in Figure 1, we thus improve over the previously fastest known LEP algorithm known by an exponential factor. Importantly, while the original analysis by CPS was only heuristic, we show that our LEP algorithm is *provably correct*. Furthermore, as an additional side result, we briefly show that the algorithm naturally admits a quantum variant with time and memory $2^{\frac{1}{3}\,\mathrm{H}(\frac{k}{n})n}$.

On the technical side, our novel canonical form function re-uses many of ideas of the original canonical form function by CPS. However, we enhance their ideas via novel techniques, which allow us to circumvent the failure conditions of CPS' algorithm. Thereby, we significantly increase its success probability to $1 - \mathcal{O}(n^{-1})$.

Impact for LESS. The suggested LESS parameters use $q = \Omega(n)$. Hence, for these parameters, already the original canonical form function by CPS has constant success probability. Thus, for the LESS parameters, our novel algorithm does not improve substantially over the LEP algorithm introduced by CPS in [CPS23]. In particular, our novel results do not invalidate the security analysis of LESS.

[2]For all inputs $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}] \in \mathbb{F}_q^{k\times n}$, in which every row of $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$ contains at least one zero entry, the canonical form of CPS outputs $\perp$. For uniformly random $\mathbf{A}$, the probability that all k rows contain only non-zero entries is at most $k \cdot (1 - 1/q)^{n-k}$. For constant $q = \mathcal{O}(1)$, this is exponentially small in n.

[3]An $[n, k]$-code $\mathcal{C}$ has constant rate, if the *code dimension* grows as $k = c \cdot n$, where n is the *code length* and c is a constant with $0 < c < 1$. In other words, $\mathcal{C}$ has constant rate if $\frac{k}{n} \neq o(1)$, and $\frac{k}{n} \neq 1 - o(1)$. This is the most important setting in practice.

Implementation. A proof-of-concept SageMath implementation of our novel LEP algorithm is available at

https://github.com/juliannowakowski/lep-cf

1.3 Open Questions

Our novel canonical form function $\mathsf{CF}_{\mathsf{New}}$ has success probability close to 1 for all finite fields of size $q \geq 7$. It is a natural open question, whether one can also achieve this success probability for fields of size $q < 7$. A perhaps even more interesting question is whether one can hybridize the ideas behind the low-weight codeword finding based algorithms with the ideas behind the canonical form function based algorithm to obtain an even faster algorithm. Lastly, it would be interesting to explore, whether one can adapt the canonical form function based algorithm to other equivalence problems, such as the lattice isomorphism problem, or the matrix code equivalence problem.

2 Preliminaries

2.1 Notations

We frequently use soft-$\mathcal{O}$ and soft-Θ notaions, i.e., $\widetilde{\mathcal{O}}(\cdot)$ and $\widetilde{\Theta}(\cdot)$, which suppress polynomial factors. The finite field with q elements is denoted by $\mathbb{F}_q$. Its unit group is $\mathbb{F}_q^* := \mathbb{F}_q \setminus \{0\}$. The group of invertible $(k \times k)$-dimensional matrix over $\mathbb{F}_q$ is denoted by $\mathrm{GL}(\mathbb{F}_q^k)$. We define the set of natural numbers as $\mathbb{N} := \{0, 1, 2, \ldots\}$. For a natural number $n \geq 1$, we define $[n] := \{1, 2, \ldots, n\}$. For a subset $J \subseteq [n]$, we denote its complement by $\overline{J} := [n] \setminus J$. All vectors $\mathbf{v} \in \mathbb{F}_q^n$ are row vectors. The i-th unit vector is denoted by $\mathbf{e}_i$, e.g., $\mathbf{e}_1 = (1, 0, \ldots, 0)$. Let $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ be a matrix. The transpose of $\mathbf{G}$ is denoted by $\mathbf{G}^T$. For $J \subseteq [n]$, we denote by $\mathbf{G}^J$ the submatrix of $\mathbf{G}$ formed by the columns indexed by J. We call J an *information set* of $\mathbf{A}$, if $|J| = k$ and the matrix $\mathbf{G}^J \in \mathbb{F}_q^{k \times k}$ is invertible. We denote by $\mathsf{RREF}(\mathbf{G})$ the row-reduced echelon form of $\mathbf{G}$. If $\mathbf{G}$ is of the form $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}]$, then we say that $\mathbf{G}$ is in *systematic form.* A linear $[n, k]$-code $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q^n$. The *rate* of an $[n, k]$-code is $\frac{k}{n}$. The binary entropy function is denoted by $\mathrm{H}(x) := -x \log_2(x) - (1 - x) \log_2(1 - x)$. We frequently make use of the approximation $\binom{n}{k} = \widetilde{\Theta}(2^{\mathrm{H}(\frac{k}{n})} n)$, which is a direct consequence of Stirling's formula.

2.2 Permutations and Monomials

We denote by Σ_n the group of permutations on n letters. For $\mathbf{P} \in \Sigma_n$, the image of $j \in [n]$ under $\mathbf{P}$ is denoted by $\mathbf{P}[j]$. More generally, for a set $J \subseteq [n]$, we define $\mathbf{P}[J] := \{\mathbf{P}[j] \mid j \in J\}$. We view permutations $\mathbf{P} \in \Sigma_n$ as $(n \times n)$-matrices with columns $\mathbf{e}^T_{\mathbf{P}^{-1}[1]}, \ldots, \mathbf{e}^T_{\mathbf{P}^{-1}[n]}$. As a consequence, multiplying a vector $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{F}_q^n$ by $\mathbf{P}$ gives $\mathbf{v} \cdot \mathbf{P} = (v_{\mathbf{P}^{-1}[1]}, \ldots, v_{\mathbf{P}^{-1}[n]})$. In other words, multiplying $\mathbf{v}$ by $\mathbf{P}$ permutes the entries of $\mathbf{v}$ according to $\mathbf{P}$. It is easy to see that inverse of $\mathbf{P}$ is given by the transpose $\mathbf{P}^T$. Hence, if we have a column vector $\mathbf{w}^T = (w_1, \ldots, w_n)^T$, then $\mathbf{P} \cdot \mathbf{w}^T$ is equal to the vector obtained by permuting the entries of $\mathbf{w}^T$ according to $\mathbf{P}^{-1}$, i.e., $\mathbf{P} \cdot \mathbf{w}^T = (w_{\mathbf{P}[1]}, \ldots, w_{\mathbf{P}[n]})^T$. For $J \subseteq [n]$ with $|J| = k$, we denote by $\mathbf{P}^J \in \Sigma_n$ a permutation with $\mathbf{P}[J] = \{1, 2, \ldots, k\}$. Hence, for all matrices $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ it holds that $\mathbf{G} \cdot \mathbf{P}^J = [\mathbf{G}^J \mid \mathbf{G}^{\overline{J}}]$.

The group of $(n \times n)$ diagonal matrices over $\mathbb{F}_q^*$ is denoted by $\mathcal{D}_{n,q}$. For a diagonal matrix $\mathbf{D} \in \mathcal{D}_{n,q}$ with diagonal entries $d_1, \ldots, d_n \in \mathbb{F}_q^*$ and a permutation $\mathbf{P} \in \Sigma_n$, the matrix $\mathbf{P}^{-1} \cdot \mathbf{D} \cdot \mathbf{P}$ is a diagonal matrix with diagonal entries $d_{\mathbf{P}^{-1}[1]}, \ldots, d_{\mathbf{P}^{-1}[n]}$.

The group of n-dimensional monomials over $\mathbb{F}_q$ is defined as

$$\mathcal{M}_{n,q} := \{\mathbf{P} \cdot \mathbf{D} \mid \mathbf{P} \in \Sigma_n, \mathbf{D} \in \mathcal{D}_{n,q}\} = \{\mathbf{D} \cdot \mathbf{P} \mid \mathbf{P} \in \Sigma_n, \mathbf{D} \in \mathcal{D}_{n,q}\}. \tag{1}$$

The fact that we can swap the order $\mathbf{P}$ and $\mathbf{D}$ in Equation (1) follows from the facts that $\mathbf{D} \cdot \mathbf{P} = \mathbf{P} \cdot (\mathbf{P}^{-1} \cdot \mathbf{D} \cdot \mathbf{P})$, and that $\mathbf{P}^{-1} \cdot \mathbf{D} \cdot \mathbf{P}$ is a diagonal matrix. Let $\mathbf{Q} \in \mathcal{M}_{n,q}$ be a monomial, and let $k \in [n]$. As first noted in [PS23], we can *factor* $\mathbf{Q}$ as

$$\mathbf{Q} = \mathbf{P}^J \cdot \begin{bmatrix} \mathbf{Q}_r & \\ & \mathbf{Q}_c \end{bmatrix},$$

where $J \subseteq [n]$ with $|J| = k$, $\mathbf{Q}_r \in \mathcal{M}_{k,q}$ and $\mathbf{Q}_c \in \mathcal{M}_{n-k,q}$. Considering such factorizations of monomials can be helpful when studying the action of monomials on matrices. Indeed, for every matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, it holds that

$$\mathbf{G} \cdot \mathbf{Q} = [\mathbf{G}^J \cdot \mathbf{Q}_r \mid \mathbf{Q}^{\overline{J}} \cdot \mathbf{Q}_c].$$

Moreover, if J is an information set of $\mathbf{G}$, then $\mathbf{G}^J \cdot \mathbf{Q}_r$ is invertible, and it holds that

$$\mathsf{RREF}(\mathbf{G} \cdot \mathbf{Q}) = [\mathbf{I}_k \mid \mathbf{Q}_r^{-1} \cdot (\mathbf{G}^J)^{-1} \cdot \mathbf{G}^{\overline{J}} \cdot \mathbf{Q}_c].$$

2.3 Linear Code Equivalence Problem

Two linear $[n, k]$-codes $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$ are called *linearly equivalent*, if there exists a monomial $\mathbf{Q} \in \mathcal{M}_{n,q}$ such that $\mathcal{C}_2 = \mathcal{C}_1 \cdot \mathbf{Q}$, i.e., $\mathcal{C}_2 = \{\mathbf{c}_1 \cdot \mathbf{Q} \mid \mathbf{c}_1 \in \mathcal{C}_1\}$. Equivalently, $\mathcal{C}_1$ and $\mathcal{C}_2$ are linearly equivalent, if generator matrices $\mathbf{G}_1, \mathbf{G}_2$ of $\mathcal{C}_1, \mathcal{C}_2$ satisfy the following equivalence relation:

Definition 2.1 (Linear Equivalence)**.** *Generator matrices* $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_q^{k \times n}$ *are called* linearly equivalent*, if there exist* $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$ *and* $\mathbf{Q} \in \mathcal{M}_{n,q}$*, such that* $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q}$*. In that case, we write* $\mathbf{G}_1 \sim \mathbf{G}_2$*.*

It is straight-forward to verify that $\sim$ indeed defines an equivalence relation on the set of all $(k \times n)$ matrices over $\mathbb{F}_q$. Definition 2.1 now suggests the following computational problem:

Definition 2.2 (LEP)**.** *The* linear code equivalence problem (LEP) *with parameters* (n, k, q) *is defined as follows:*

- ***Given:*** *Linearly equivalent generator matrices* $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_q^{k \times n}$.
- ***Find:*** *Matrices* $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$ *and* $\mathbf{Q} \in \mathcal{M}_{n,q}$ *such that* $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q}$.

In cryptography, one usually considers an average case variant of LEP, where the matrices $\mathbf{G}_1$ and $\mathbf{G}_2$ are sampled from the following distribution.

Definition 2.3 (Average Case LEP Distribution)**.** *For parameters* n, k, q*, the* average case LEP distribution $D_{n,k,q}^{\mathsf{LEP}}$ *is defined as follows: Sample a uniformly random matrix* $\mathbf{G}_1 \in \mathbb{F}_q^{k \times n}$*, and a uniformly random monomial* $\mathbf{Q} \in \mathcal{M}_{n,q}$*. Compute* $\mathbf{G}_2 := \mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{Q})$*, and output the tuple* $(\mathbf{G}_1, \mathbf{G}_2)$*.*

Formally, the average case variant of LEP ($-LEP) is defined as follows.

Definition 2.4 ($-LEP)**.** *The* average case linear code equivalence problem ($-LEP) *with parameters* (n, k, q) *is defined as follows:*

- ***Given:*** *Linearly equivalent generator matrices* $\mathbf{G}_1$*,* $\mathbf{G}_2$ *sampled from* $D_{n,k,q}^{\mathsf{LEP}}$.
- ***Find:*** *Matrices* $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$ *and* $\mathbf{Q} \in \mathcal{M}_{n,q}$ *such that* $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q}$.

2.4 Probabilities

We need the following concentration bound for the sum of (possibly dependent) $\{0,1\}$-valued random variables $X_1, \ldots, X_n$. inequality.

Lemma 2.5. *Let $X_1, \ldots, X_n \in \{0,1\}$ denote (possibly dependent) random variables. Let $p \in [0,1]$, such that $\Pr[X_i = 1] > p$ for every $i \in [n]$. Then for $X := \sum_{i=1}^n X_i$ it holds that*

$$\Pr\left[X > \frac{p}{2} \cdot n\right] > \frac{p}{2}.$$

A proof for Lemma 2.5 is given in Appendix A.1. We note that for *independent* random variables $X_1, \ldots, X_n$, Lemma 2.5 is significantly inferior to more standard concentration bounds, such as the Chernoff bound (which states that $\Pr\left[X > \frac{p}{2} \cdot n\right] > 1 - e^{-\Omega(p \cdot n)}$). However, a major advantage of Lemma 2.5 is that it also applies to *dependent* random variables.

Lemma 2.6. *Let q be a prime power and let $k \in \mathbb{N}$. If we sample a matrix $\mathbf{A}$ uniformly at random from $\mathbb{F}_q^{k \times k}$, then $\mathbf{A}$ is invertible with probability greater than $\frac{1}{4}$.*

A proof for Lemma 2.6 is given in Appendix A.2.

3 CPS Revisited

In this section, we revisit the original work of Chou, Persichetti and Santini (CPS) [CPS23]. We start by recalling the definition of so-called *canonical form functions* in Section 3.1. While CPS initially introduced these functions to improve the efficiency of the LESS signature scheme, they come with a surprising destructive application: CPS showed that any canonical form function can be transformed into a LEP algorithm. We revisit this transformation in Sections 3.2 and 3.3. While the original CPS analysis of the transformation was heuristic, we show – as a novel result – that, under certain conditions, the transformation can be made provably correct. This observation then serves as the foundation for our novel, provably correct LEP algorithm, which we introduce in Section 4.

3.1 Canonical Form Functions

LRL Equivalence. CPS introduce a novel framework for studying equivalence relations for linear codes. While CPS use their framework to study five different equivalence relations, we need only one out of these five. CPS call this equivalence relation *Case 5*. However, we choose the more descriptive name *left-right linear equivalence*, or *LRL equivalence*, for short.

Definition 3.1 (LRL Equivalence). *Two generator matrices in systematic form $\mathbf{G}_1 = [\mathbf{I}_k \mid \mathbf{A}_1], \mathbf{G}_2 = [\mathbf{I}_k \mid \mathbf{A}_2] \in \mathbb{F}_q^{k \times n}$ are called* left-right linearly equivalent *or* LRL equivalent, *if and only if there exist $\mathbf{Q}_r \in \mathcal{M}_{k,q}$ and $\mathbf{Q}_c \in \mathcal{M}_{n-k,q}$ such that*

$$\mathbf{A}_2 = \mathbf{Q}_r \cdot \mathbf{A}_1 \cdot \mathbf{Q}_c.$$

In that case, we write $\mathbf{G}_1 \overset{\mathsf{LRL}}{\sim} \mathbf{G}_2$. The equivalence class of a generator matrix in systematic form $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}]$ is denoted by $[\mathbf{G}]_{\overset{\mathsf{LRL}}{\sim}}$.

Notice that $\overset{\mathsf{LRL}}{\sim}$ indeed defines an equivalence relation on the set of $(k \times n)$-matrices over $\mathbb{F}_q$ in systematic form. We point out that the original definition by CPS is slightly more general than ours, as it also considers generator matrices that are not in systematic form. However, for our purposes, the simplified definition above suffices.

Additionally, we like to point out that LRL equivalence is a special case of linear equivalence: If $\mathbf{G}_1 = [\mathbf{I}_k \mid \mathbf{A}_1]$ and $\mathbf{G}_2 = [\mathbf{I}_k \mid \mathbf{A}_2]$ are LRL equivalent, i.e., $\mathbf{A}_2 = \mathbf{Q}_r \cdot \mathbf{A}_1 \cdot \mathbf{Q}_c$, then for

$$\mathbf{Q} := \begin{bmatrix} \mathbf{Q}_r^{-1} & \\ & \mathbf{Q}_c \end{bmatrix} \in \mathcal{M}_{n,q}, \quad \text{and} \quad \mathbf{U} := \mathbf{Q}_r \in \mathcal{M}_{k,q} \subseteq \mathrm{GL}(\mathbb{F}_q^k),$$

it holds that $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q}$. Hence, the codes $\mathcal{C}_1$ and $\mathcal{C}_2$ generated by $\mathbf{G}_1$ and $\mathbf{G}_2$ are linearly equivalent

Some Background. Definition 3.1 stems from the following scenario arising in the LESS signature scheme: Let $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathbb{F}_q^n$ be linearly equivalent $[n, k]$-codes. Suppose Alice and Bob know generator matrices $\mathbf{G}_1$, $\mathbf{G}_2$ of $\mathcal{C}_1$ and $\mathcal{C}_2$. Additionally, suppose Alice knows a monomial $\mathbf{Q} \in \mathcal{M}_{n,q}$ such that $\mathcal{C}_2 = \mathcal{C}_1 \cdot \mathbf{Q}$. In LESS, Alice wants to prove to Bob that $\mathcal{C}_1$ and $\mathcal{C}_2$ are indeed linearly equivalent. A simple way to do this, would be for Alice to simply send $\mathbf{Q}$ to Bob. However, in the LESS setting, Alice would like to make the proof as *memory-efficient* as possible. To this end, CPS suggest the following approach:

Let us factor $\mathbf{Q}$ as

$$\mathbf{Q} = \mathbf{P}^J \cdot \begin{bmatrix} \mathbf{Q}_r & \\ & \mathbf{Q}_c \end{bmatrix},$$

for some $J \subseteq [n]$ with $|J| = k$, $\mathbf{Q}_r \in \mathcal{M}_{k,q}$ and $\mathbf{Q}_c \in \mathcal{M}_{n-k,q}$. Let us define $\mathbf{G}_1' := \mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^J)$, and $\mathbf{G}_2' := \mathsf{RREF}(\mathbf{G}_2)$. For simplicity, let us assume that J is an information set of $\mathbf{G}_1$. Then it holds that

$$\mathbf{G}_1' = [\mathbf{I}_k \mid (\mathbf{G}_1^J)^{-1} \cdot \mathbf{G}_1^{\overline{J}}].$$

Since $\mathcal{C}_2 = \mathcal{C}_1 \cdot \mathbf{Q}$, we have $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q}$ for some $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$. Together with the fact that RREF is invariant under invertible transformations from the left, this implies

$$\mathbf{G}_2' = \mathsf{RREF}(\mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q}) = \mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{Q}) = [\mathbf{I}_k \mid \mathbf{Q}_r^{-1} \cdot (\mathbf{G}_1^J)^{-1} \cdot \mathbf{G}_1^{\overline{J}} \cdot \mathbf{Q}_c].$$

The crucial observation is now that $\mathbf{G}_1'$ and $\mathbf{G}_2'$ satisfy Definition 3.1, i.e., the matrices $\mathbf{G}_1'$ and $\mathbf{G}_2'$ are LRL equivalent.

Assume for a moment that Bob has an efficient algorithm for deciding whether two matrices are LRL equivalent. In such a scenario, CPS suggest instead of Alice sending $\mathbf{Q}$ to Bob, to send only J. To verify that $\mathcal{C}_1$ and $\mathcal{C}_2$ are linearly equivalent, Bob can then proceed as follows: Bob computes $\mathbf{G}_2'$, and uses J to compute $\mathbf{G}_1'$. After that, he tests whether $\mathbf{G}_1'$ and $\mathbf{G}_2'$ are LRL equivalent. If so, he accepts that $\mathcal{C}_1$ and $\mathcal{C}_2$ are linearly equivalent.

As shown by CPS, this approach is *sound*, i.e., Bob accepts only if $\mathcal{C}_1$ and $\mathcal{C}_2$ are indeed linearly equivalent. Since storing J requires significantly less memory than storing $\mathbf{Q}$, this approach greatly improves the memory-complexity of the proof. However, it requires access to an efficient algorithm for deciding, whether to matrices are LRL equivalent. For certain parameters of n, k and q, CPS can indeed give such an algorithm. It is based on so-called *canonical form functions*, which, we formally define below.

Canonical Form Functions. In a nuthsell, a canonical form function for $\overset{\mathsf{LRL}}{\sim}$ is an efficient algorithm CF that takes a generator matrix $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}]$, and outputs a canonical representative $\mathbf{G}^* = [\mathbf{I}_k \mid \mathbf{A}^*]$ of the equivalence class $[\mathbf{G}]_{\underset{\sim}{\mathsf{LRL}}}$. Additionally, CF outputs monomials $\mathbf{Q}_r$ and $\mathbf{Q}_c$, such that $\mathbf{A}^* = \mathbf{Q}_r \cdot \mathbf{A} \cdot \mathbf{Q}_c$. More precisely, a canonical form function is defined as follows:

Definition 3.2 (Canonical Form Function)**.** *A* canonical form function *(for LRL equivalence) is a polynomial time algorithm* CF*, that on input of a generator matrix in systematic form* $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}] \in \mathbb{F}_q^{k \times n}$ *either outputs*

- *a tuple* $(\mathbf{G}^*, \mathbf{Q}_r, \mathbf{Q}_c) \in [\mathbf{G}]_{\underset{\sim}{\mathsf{LRL}}} \times \mathcal{M}_{k,q} \times \mathcal{M}_{n-k,q}$*, where* $\mathbf{G}^* = [\mathbf{I}_k \mid \mathbf{A}^*]$ *is a representative of the equivalence class* $[\mathbf{G}]_{\underset{\sim}{\mathsf{LRL}}}$*, such that* $\mathbf{A}^* = \mathbf{Q}_r \cdot \mathbf{A} \cdot \mathbf{Q}_c$*,*

- *or an error symbol* $\perp$*.*

Furthermore, we require the representative $\mathbf{G}^*$ *to be* canonical*. That is, for all* $\mathbf{G}_1 \overset{\mathsf{LRL}}{\sim} \mathbf{G}_2$ *with* $\mathsf{CF}(\mathbf{G}_1) \neq \perp$*, we require* $\mathsf{CF}(\mathbf{G}_1)$ *and* $\mathsf{CF}(\mathbf{G}_2)$ *to output the same representative of the equivalence class* $[\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}} = [\mathbf{G}_2]_{\underset{\sim}{\mathsf{LRL}}}$*. For a canonical form function* CF*, we define its* success probability *as*

$$\gamma_{\mathsf{CF}}(n,k,q) := \Pr_{\mathbf{A} \leftarrow \mathbb{F}_q^{k \times (n-k)}} \Big[\mathsf{CF}\big([\mathbf{I}_k \mid \mathbf{A}]\big) \neq \perp \Big].$$

As with our definition of LRL equivalence (Definition 3.1), we point out that also the original CPS definition for canonical form functions is more general than ours, as it also considers matrices that are not in systematic form. However, again our simplified definition suffices.

The Dark Side of CF. While CPS introduced canonical form functions with a *constructive* application in mind, they have a surprising *destructive* application: CPS give an elegant transformation that turns any canonical form function CF into a algorithm for solving LEP in time $\widetilde{\mathcal{O}}\left(\gamma_{\mathsf{CF}}(n,k,q)^{-1/2} \cdot 2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})n}\right)$. In particular, for canonical form functions with (at least) constant success probability, the transformation results in an LEP algorithm with runtime $\widetilde{\mathcal{O}}\left(2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})n}\right)$. Unfortunately, as discussed in the introduction, finding canonical form functions with constant success probability is challenging: CPS give a canonical form function that achieves constant success probability only for large $q = \Omega(n)$. However, for constant $q = \mathcal{O}(1)$, its success probability is exponentially small – leading to an LEP algorithm that requires time exponentially higher than $2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})n}$.

In Section 4, we will introduce a novel canonical form function, that has success probability probability $1 - \mathcal{O}(n^{-1})$ for all $q \geq 7$. By combining our canonical function with the CPS transformation, this immediately implies our novel $\widetilde{\mathcal{O}}\left(2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})n}\right)$-time LEP algorithm.

Importantly, the original analysis of the CPS transformation is only *heuristic*. However, as we show in the following two sections, for our setting of a canonical form function with (at least) constant success probability, we can even make the transformation *provably correct*.

3.2 LEP as a Collision Finding Problem

The main idea behind the CPS transformation for turning a canonical form function into an LEP algrithm is to view LEP as a collision finding problem: The transformation turns any canonical form function into a meet-in-the-middle algorithm, that on input of a LEP instance $\mathbf{G}_1 \sim \mathbf{G}_2$ tries to find CF-*colliding* information sets J_1, J_2, as defined below.

Definition 3.3 (CF-colliding)**.** *Let* $\mathbf{G}_1 \sim \mathbf{G}_2$ *be an LEP instance, and let* CF *be a canonical form function. We call two information sets* J_1, J_2 *of* $\mathbf{G}_1$ *and* $\mathbf{G}_2$ CF-colliding *for* $(\mathbf{G}_1, \mathbf{G}_2)$*, if*

$$\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1}) \overset{\mathsf{LRL}}{\sim} \mathsf{RREF}(\mathbf{G}_2 \cdot \mathbf{P}^{J_2}),$$

and additionally

$$\mathsf{CF}(\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1})) \neq \perp, \quad \mathsf{CF}(\mathsf{RREF}(\mathbf{G}_2 \cdot \mathbf{P}^{J_2})) \neq \perp.$$

As the following lemma shows, once CF-colliding information sets J_1 and J_2 are found, solving LEP becomes easy:

Lemma 3.4 (Adapted from Proposition 11 in [CPS23]). *Let $\mathbf{G}_1 \sim \mathbf{G}_2$ be an LEP instance, and let CF be a canonical form function. Let J_1, J_2 be CF-colliding information sets for $(\mathbf{G}_1, \mathbf{G}_2)$. On input $\mathbf{G}_1, \mathbf{G}_2, J_1, J_2$, algorithm $\mathsf{RecoverMon}^{\mathsf{CF}(\cdot)}$ (Algorithm 1) computes a solution $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$, $\mathbf{Q} \in \mathcal{M}_{n,q}$ to the LEP instance defined by $\mathbf{G}_1$ and $\mathbf{G}_2$ in polynomial time.*

For completeness, we recall the proof of Lemma 3.4 in Appendix A.3.

Algorithm 1: $\mathsf{RecoverMon}^{\mathsf{CF}(\cdot)}$

```
Input:   LEP instance $\mathbf{G}_1 \sim \mathbf{G}_2 \in \mathbb{F}_q^{k \times n}$,
         $\mathsf{CF}$-colliding information sets $J_1, J_2$ for $(\mathbf{G}_1, \mathbf{G}_2)$.
Output:  Solution $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k), \mathbf{Q} \in \mathcal{M}_{n,q}$ with $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q}$.
1 Compute $\mathbf{G}_i' := \mathsf{RREF}(\mathbf{G}_i \cdot \mathbf{P}^{J_i})$ for $i \in \{1, 2\}$.
2 Compute $\mathsf{CF}(\mathbf{G}_i') = (\mathbf{G}_i^*, \mathbf{Q}_{r,i}, \mathbf{Q}_{c,i})$ for $i \in \{1, 2\}$.
3 Compute $\mathbf{U} := \mathbf{G}_2^{J_2} \cdot \mathbf{Q}_{r,2}^{-1} \cdot \mathbf{Q}_{r,1} \cdot (\mathbf{G}_1^{J_1})^{-1}$.
4 Compute
    $$\mathbf{Q} := \mathbf{P}^{J_1} \cdot \begin{bmatrix} \mathbf{Q}_{r,1}^{-1} \cdot \mathbf{Q}_{r,2} & \\ & \mathbf{Q}_{c,1} \cdot \mathbf{Q}_{c,2}^{-1} \end{bmatrix} \cdot (\mathbf{P}^{J_2})^{-1}.$$
5 return $\mathbf{U}, \mathbf{Q}$
```

To see that CF-colliding information sets actually exist, we need the following lemma.

Lemma 3.5. *Let $\mathbf{G}_1 \sim \mathbf{G}_2 \in \mathbb{F}_q^{k \times n}$ be linearly equivalent matrices, where*

$$\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{P} \cdot \mathbf{D},$$

for some $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$, $\mathbf{P} \in \Sigma_n$ and $\mathbf{D} \in \mathcal{D}_{n,q}$. Let J_1 be an information set of $\mathbf{G}_1$. Then $J_2 := \mathbf{P}[J_1]$ is an information set of $\mathbf{G}_2$, and it holds that

$$\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1}) \overset{\mathsf{LRL}}{\sim} \mathsf{RREF}(\mathbf{G}_2 \cdot \mathbf{P}^{J_2}).$$

In the original CPS paper, Lemma 3.5 is not stated explicitly, but only hinted at. For completeness, we give a formal proof in Appendix A.4.

3.3 A Provably Correct Variant of the CPS Transformation

We are now ready to describe the CPS transformation for converting a canonical form function CF into a LEP algorithm. It is depicted in Algorithm 2. For simplicity, we give a variant of the transformation that only works well for canonical form functions CF with (at least) constant success probability $\gamma_{\mathsf{CF}}(n, k, q) = \Omega(1)$. As we show below, a major advantage of this variant is that it can be shown to be provably correct.

The Algorithm. In a nutshell, Algorithm 2 samples on input of a LEP instance $\mathbf{G}_1 \sim \mathbf{G}_2$ sufficiently many random information sets J_1, J_2 of $\mathbf{G}_1$ and $\mathbf{G}_2$, with the hope of sampling at least one CF-colliding pair (see Definition 3.3). If it finds such a pair, it uses

Algorithm 2: $\mathsf{LEP\text{-}Coll\text{-}Search}^{\mathsf{CF}(\cdot)}$

```
Input:   LEP instance $\mathbf{G}_1 \sim \mathbf{G}_2 \in \mathbb{F}_q^{k\times n}$.
Output:  Solution $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k), \mathbf{Q} \in \mathcal{M}_{n,q}$ with $\mathbf{G}_2 = \mathbf{U}\cdot\mathbf{G}_1\cdot\mathbf{Q}$,
         or error symbol $\perp$.

1  Initialize empty list $L$.
2  repeat $\left\lfloor\sqrt{\frac{1}{2}\binom{n}{k}}\right\rfloor$ times
3      Sample uniformly random size-$k$ subset $J_1$ of $[n]$.
4      $\mathbf{G}_1' := \mathsf{RREF}(\mathbf{G}_1\cdot\mathbf{P}^{J_1})$.
5      if $\mathbf{G}_1'$ is in systematic form then                      ▷ Is $J_1$ information set?
6          if $\mathsf{CF}(\mathbf{G}_1') \neq \perp$ then
7              Parse the first component of $\mathsf{CF}(\mathbf{G}_1')$'s output as $\mathbf{G}_1^* \in [\mathbf{G}_1']_{\underset{\sim}{\mathsf{LRL}}}$.
8              Store $(\mathbf{G}_1^*, J_1)$ in $L$.
9  Sort $L$ by the second component.
10 repeat $\left\lfloor\sqrt{\frac{1}{2}\binom{n}{k}}\right\rfloor$ times
11     Sample uniformly random size-$k$ subset $J_2$ of $[n]$.
12     $\mathbf{G}_2' := \mathsf{RREF}(\mathbf{G}_2\cdot\mathbf{P}^{J_2})$.
13     if $\mathbf{G}_2'$ is in systematic form then                      ▷ Is $J_2$ information set?
14         if $\mathsf{CF}(\mathbf{G}_2') \neq \perp$ then
15             Parse the first component of $\mathsf{CF}(\mathbf{G}_2')$'s output as $\mathbf{G}_2^* \in [\mathbf{G}_2']_{\underset{\sim}{\mathsf{LRL}}}$.
16             if $(\mathbf{G}_2^*, J_1) \in L$ for some $J_1$ then          ▷ Are $J_1, J_2$ CF-colliding?
17                 return $\mathsf{RecoverMon}^{\mathsf{CF}(\cdot)}(\mathbf{G}_1, \mathbf{G}_2, J_1, J_2)$
18 return $\perp$
```

$\mathsf{RecoverMon}^{\mathsf{CF}(\cdot)}$ (Algorithm 1) as a subroutine to easily solve the LEP instance. More precisely, it works as follows:

On input $\mathbf{G}_1 \sim \mathbf{G}_2$, Algorithm 2 picks $\left\lfloor\sqrt{\frac{1}{2}\binom{n}{k}}\right\rfloor$ random size-k subsets J_1 of $[n]$, and computes $\mathbf{G}_1' := \mathsf{RREF}(\mathbf{G}_1\cdot\mathbf{P}^{J_1})$, for every J_1. If $\mathbf{G}_1'$ is in systematic form (or equivalently, if J_1 is an information set of $\mathbf{G}_1$), the algorithm runs CF on $\mathbf{G}_1'$. If CF does not return $\perp$, CF returns a canonical representative $\mathbf{G}_1^*$ of the equivalence class $[\mathbf{G}_1']_{\underset{\sim}{\mathsf{LRL}}}$. Algorithm 2 then stores $\mathbf{G}_1^*$ along with J_1 in some list L.

Next, the algorithm tries to find an information set J_2 of $\mathbf{G}_2$, that together with some previously sampled information set J_1 of $\mathbf{G}_1$ is CF-colliding for $(\mathbf{G}_1, \mathbf{G}_2)$. The algorithm can easily detect such a J_2 by simply testing if $\mathbf{G}_2' := \mathsf{RREF}(\mathbf{G}_2\cdot\mathbf{P}^{J_2})$ is in systematic form, and, additionally, if the computation of $\mathsf{CF}(\mathbf{G}_2')$ yields a canonical representative identical to one of the $\mathbf{G}_1^*$'s, that it has stored in L before (see Definitions 3.2 and 3.3). Once it finds such a J_2, it can easily solve the LEP instance via algorithm $\mathsf{RecoverMon}^{\mathsf{CF}(\cdot)}$ (see Lemma 3.4).

Runtime and Success Probability. The first repeat-loop in Algorithm 2 clearly runs in time $T := \widetilde{\Theta}\left(\sqrt{\binom{n}{k}}\right)$. Sorting L can be done in time T as well. After sorting L, testing for membership in L can be done in time $\widetilde{\Theta}(1)$. Thus, also the second repeat-loop runs in time T. Hence, we obtain an overall runtime of $T = \widetilde{\Theta}\left(\sqrt{\binom{n}{k}}\right) = \widetilde{\Theta}\left(2^{\frac{1}{2}\,\mathrm{H}\left(\frac{k}{n}\right)n}\right)$ for Algorithm 2.

As we show below, for canonical form functions with (at least) constant success probability, the algorithm *provably* solves the average case variant of LEP ($-LEP) with constant success probability:

Theorem 3.6 (Correctness CPS Transformation)**.** *Let $\mathbf{G}_1 \sim \mathbf{G}_2 \in \mathbb{F}_q^{k\times n}$ be a \$-LEP instance, and let* CF *be a canonical form function with (at least) constant success probability. On input $\mathbf{G}_1, \mathbf{G}_2$, Algorithm* $\mathsf{LEP\text{-}Coll\text{-}Search}^{\mathsf{CF}(\cdot)}$ *(Algorithm 2) outputs a solution to the \$-LEP instance defined by $\mathbf{G}_1$ and $\mathbf{G}_2$ in time $\widetilde{\Theta}\left(2^{\frac{1}{2}\,\mathrm{H}\left(\frac{k}{n}\right)n}\right)$, and with constant success probability.*

The proof of Theorem 3.6 is based on the following novel technical lemma.

Lemma 3.7. *Let $\mathbf{G}_1 \sim \mathbf{G}_2 \in \mathbb{F}_q^{k\times n}$ be a \$-LEP instance, and let* CF *be a canonical form function with (at least) constant success probability. If we run* $\mathsf{LEP\text{-}Coll\text{-}Search}^{\mathsf{CF}(\cdot)}$ *(Algorithm 2) on input $\mathbf{G}_1, \mathbf{G}_2$, then with constant probability, the list L computed by* $\mathsf{LEP\text{-}Coll\text{-}Search}^{\mathsf{CF}(\cdot)}$ *contains more than*

$$\frac{\gamma_{\mathsf{CF}}(n,k,q)}{8} \cdot \left\lfloor \sqrt{\frac{1}{2} \cdot \binom{n}{k}} \right\rfloor$$

distinct elements.

Proof. Let $T := \left\lfloor \sqrt{\frac{1}{2} \cdot \binom{n}{k}} \right\rfloor$, and $\gamma := \gamma_{\mathsf{CF}}(n,k,q)$. We denote by $J_{1,1}, \dots, J_{1,T}$ the T sets J_1, that algorithm Algorithm 2 samples in its first repeat-loop. For every i, we define an indicator variable $X_i \in \{0,1\}$, that is equal to 1, if and only if $J_{1,i}$ gets stored in L. Let E_i denote the event that $J_{1,i}$ is an information set of $\mathbf{G}_1$. Looking at Lines 5 and 6 of Algorithm 2, it follows that

$$\Pr[X_i = 1] = \Pr[E_i] \cdot \Pr[\mathsf{CF}(\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_{1,i}})) \neq \perp \mid E_i].$$

The set $J_{1,i}$ is an information set of $\mathbf{G}_1$, if and only if $\mathbf{G}_1^{J_{1,i}} \in \mathbb{F}_q^{k\times k}$ is invertible. Since in \$-LEP, the matrix $\mathbf{G}_1$ is uniformly random, also $\mathbf{G}_1^{J_{1,i}}$ uniformly random. Hence, by Lemma 2.6, we have $\Pr[E_i] > \frac{1}{4}$, and thus

$$\begin{aligned}\Pr[X_i = 1] &> \frac{1}{4} \cdot \Pr[\mathsf{CF}(\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_{1,i}})) \neq \perp \mid E_i] \\ &= \frac{1}{4} \cdot \Pr[\mathsf{CF}([\mathbf{I}_k \mid (\mathbf{G}_1^{J_{1,i}})^{-1} \cdot \mathbf{G}_1^{\overline{J_{1,i}}}]) \neq \perp \mid E_i] = \frac{\gamma}{4},\end{aligned}$$

where the last equality follows from Definition 3.2 and the fact that in \$-LEP, the matrix $\mathbf{G}_1^{\overline{J_{1,i}}}$ is uniformly random.

Applying Lemma 2.5 to $|L| = \sum_{i=1}^{T} X_{J_{1,i}}$, we obtain $\Pr\left[|L| > \frac{\gamma}{8} \cdot T\right] > \frac{\gamma}{8} = \Omega(1)$. This already shows that, with constant probability, the list L contains more than $\frac{\gamma_{\mathsf{CF}}(n,k,q)}{8} \cdot \left\lfloor \sqrt{\frac{1}{2} \cdot \binom{n}{k}} \right\rfloor$ elements. To finish the proof, we have to show that with constant probability these elements are distinct. To this end, we simply note that the probability that the i-th sampled set $J_{1,i}$ is equal to a previously sampled set $J_{1,1}, \dots, J_{1,i-1}$ is $(i-1)/\binom{n}{k}$. Thus, the probability that all sets $J_{1,i}$ are distinct is

$$\prod_{i=1}^{T}\left(1 - \frac{i-1}{\binom{n}{k}}\right) \geq \left(1 - \frac{T}{\binom{n}{k}}\right)^{T} \geq 1 - \frac{T^2}{\binom{n}{k}} \geq 1 - \frac{\frac{1}{2}\binom{n}{k}}{\binom{n}{k}} = \frac{1}{2} = \Omega(1).$$

This shows that with constant probability all elements in L are distinct, and thus concludes the proof. □

Using Lemma 3.7, we now prove Theorem 3.6.

Proof (Theorem 3.6). We have to show that Algorithm 2 samples in its second repeat-loop with constant probability an information set J_2 of $\mathbf{G}_2$, that, together with some information set J_1 stored in the list L, is CF-colliding for $(\mathbf{G}_1, \mathbf{G}_2)$.

Since $\mathbf{G}_1 \sim \mathbf{G}_2$, we can write $\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{P} \cdot \mathbf{D}$, for some $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$, $\mathbf{P} \in \Sigma_n$ and $\mathbf{D} \in \mathcal{D}_{n,q}$. Let $\mathcal{I}_1$ denote the set of all information sets J_1 that Algorithm 2 stores in the list L, and let $\mathcal{I}_2 := \{\mathbf{P}[J_1] \mid J_1 \in \mathcal{I}_1\}$. By Lemma 3.5, every pair $(J_1, \mathbf{P}[J_1]) \in \mathcal{I}_1 \times \mathcal{I}_2$ is CF-colliding for $(\mathbf{G}_1, \mathbf{G}_2)$. Thus, it suffices to show that Algorithm 2 samples at least one set J_2 with $J_2 \in \mathcal{I}_2$.

Let $\gamma := \gamma_{\mathsf{CF}}(n,k,q)$ and $T := \left\lfloor \sqrt{\frac{1}{2}\binom{n}{k}} \right\rfloor$. By Lemma 3.7, we have with constant probability that $|\mathcal{I}_2| = |\mathcal{I}_1| = |L| > \frac{\gamma}{8} \cdot T$. If indeed $|\mathcal{I}_2| > \frac{\gamma}{8} \cdot T$, then Algorithm 2 samples $J_2 \in \mathcal{I}_2$ with probability at least

$$1 - \left(1 - \frac{\frac{\gamma}{8} \cdot T}{\binom{n}{k}}\right)^T \geq 1 - \exp\left(-\frac{\frac{\gamma}{8} \cdot T^2}{\binom{n}{k}}\right) \geq 1 - e^{-\gamma/16} \geq \frac{\gamma}{32}.$$

Hence, the overall success probability of Algorithm 2 is lower bounded by $\Omega(1) \cdot \frac{\gamma}{32} = \Omega(1)$, as desired.

$\square$

A Memoryless Variant. We note that the memory consumption of Algorithm 2 is quite excessive, as (by Lemma 3.7) it requires storing a list of size roughly $\sqrt{\binom{n}{k}}$. However, this issue can easily be avoided via a standard Van-Oorschot-Wiener-like collision-finding algorithm [vW99].

A Quantum Variant. For canonical form functions with (at least) constant success probability, Algorithm 2 naturally gives rise to quantum variant with time and memory $\widetilde{\Theta}\left(2^{\frac{1}{3}\,\mathrm{H}(\frac{k}{n})n}\right)$: Instead of sampling roughly $\sqrt{\binom{n}{k}}$ sets J_1 in the algorithms for first repeat-loop, we sample only $\binom{n}{k}^{1/3}$ such sets. By slightly adapting the proofs of Lemma 3.7 and Theorem 3.6, one can easily show that the probability that a single iteration of the second repeat-loop finds a CF-colliding pair J_1, J_2 then drops from roughly $\binom{n}{k}^{-1/2}$ to roughly $\binom{n}{k}^{-2/3}$. Hence, by replacing the second repeat-loop by Grover search / amplitude amplification, we immediately obtain a quantum algorithm with the desired runtime and memory consumption.

Comparison with Original CPS Analysis. Our novel Theorem 3.6 shows that Algorithm 2 solves LEP in time roughly $\sqrt{\binom{n}{k}} \approx 2^{\frac{1}{2}\,\mathrm{H}(\frac{k}{n})}$, and with constant success probability – provided that the underlying canonical form function CF has constant success probability $\gamma_{\mathsf{CF}}(n,k,q) = \Omega(1)$. CPS, on the other hand, give a more general, yet heuristic, variant of Algorithm 2, that is supposed to work for *any* canonical form function CF. For their variant, CPS claim runtime roughly $\sqrt{\frac{1}{\gamma_{\mathsf{CF}}(n,k,q)}\binom{n}{k}}$, and "constant success probability which is approximately 1/2".

Let $\zeta > \frac{1}{4}$ denote the probability that a uniformly random matrix $\mathbf{A} \in \mathbb{F}_q^{k \times k}$ is invertible (see Lemma 2.6). The only difference between Algorithm 2 and the original CPS algorithm is that instead of sampling $\left\lfloor \sqrt{\frac{1}{2}\binom{n}{k}} \right\rfloor$ random index sets J_1, J_2, CPS suggest to sample roughly $\sqrt{\frac{1}{\zeta \cdot \gamma_{\mathsf{CF}}(n,k,q)}\binom{n}{k}}$ such sets. Since each pair J_1, J_2 is CF-colliding with probability at least $\zeta \cdot \gamma_{\mathsf{CF}}(n,k,q) \cdot \binom{n}{k}^{-1}$, CPS then sample on expectation at least one CF-colliding pair. Since Algorithm 2 is successful, if and only if it samples at least one such pair, it follows that *on expectation*, CPS indeed solve LEP.

Unfortunately, sampling one such pair *on expectation* does not necessarily imply that one actually samples one such pair *with decent probability.*[4] To overcome this issue, CPS heuristically assume that for any pair of index sets J_1, J_1' the probabilities $\Pr[\mathsf{CF}(\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1})) \neq \bot]$ and $\Pr[\mathsf{CF}(\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1'})) \neq \bot]$ can be treated as independent.[5] Under this assumption, standard concentration bounds (e.g., the Chernoff bound) indeed imply that that the original CPS algorithm solves LEP with constant probability. However, in reality, these probabilities are of course not perfectly independent, and it is unclear how much of an issue this is in practice.

To circumvent this issue, we resort in our proof of Lemma 3.7 to the concentration bound from Lemma 2.5. We use Lemma 2.5 to show that when sampling $T \in \mathbb{N}$ random index sets J_1, then with probability at least $\gamma_{\mathsf{CF}}(n, k, q)/8$ more than $\gamma_{\mathsf{CF}}(n, k, q)/8 \cdot T$ of these sets satisfy $\mathsf{CF}(\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1})) \neq \bot$. For our setting of canonical form functions with constant success probability $\gamma_{\mathsf{CF}}(n, k, q) = \Omega(1)$, this is good enough to conclude constant success probability for Algorithm 2. However, for canonical form functions with exponentially small success probability, Lemma 2.5 is too weak to make any meaningful conclusion about the success probability of the original CPS algorithm.

4 A Novel Canonical Form Function

Now that we have formally defined canonical form functions in the previous Section 3, we are ready to introduce our novel canonical form function, which we denote by $\mathsf{CF}_{\mathsf{New}}$. As we will show below, $\mathsf{CF}_{\mathsf{New}}$ has over all fields of size $q \geq 7$ success probability $1 - \mathcal{O}(n^{-1})$. Together with Theorem 3.6 from the previous section, this immediately yields our novel $\widetilde{\mathcal{O}}\left(2^{\frac{1}{2}\,\mathrm{H}\left(\frac{k}{n}\right)n}\right)$-time LEP algorithm.

Road Map. For ease of exposition, we break $\mathsf{CF}_{\mathsf{New}}$ into four steps. While descirbing these steps, we prove the correctness of $\mathsf{CF}_{\mathsf{New}}$ along the way. Let us briefly outline our road map for our proof of correctness. To this end, let $\mathbf{G}_1 = [\mathbf{I}_k \mid \mathbf{A}_1] \overset{\mathsf{LRL}}{\sim} \mathbf{G}_2 = [\mathbf{I}_k \mid \mathbf{A}_2] \in \mathbb{F}_q^{k \times n}$ be any pair of LRL equivalent matrices. To prove that our novel canonical form function $\mathsf{CF}_{\mathsf{New}}$ is correct, we have to show that running $\mathsf{CF}_{\mathsf{New}}$ on inputs $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively, returns the same representative of the equivalence class $[\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}} = [\mathbf{G}_2]_{\underset{\sim}{\mathsf{LRL}}}$. To this end, we proceed as follows:

On input $\mathbf{G}_1$, $\mathsf{CF}_{\mathsf{New}}$ computes in the i-th of its four steps a matrix $\mathbf{G}_1^{(i)} = [\mathbf{I}_k \mid \mathbf{A}_1^{(i)}] \in [\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}}$. Analogously, on input $\mathbf{G}_2$, $\mathsf{CF}_{\mathsf{New}}$ computes in its i-th step a matrix $\mathbf{G}_2^{(i)} = [\mathbf{I}_k \mid \mathbf{A}_2^{(i)}] \in [\mathbf{G}_2]_{\underset{\sim}{\mathsf{LRL}}} = [\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}}$. We show that as the steps progress, the matrices $\mathbf{A}_1^{(i)}$, $\mathbf{A}_2^{(i)}$ become increasingly *similar*. Ultimately, after the fourth step, we end up with $\mathbf{A}_1^{(4)} = \mathbf{A}_2^{(4)}$. The final matrices $\mathbf{G}_1^{(4)} = \mathbf{G}_2^{(4)}$ then serve as our canonical representative of the equivalence class $[\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}} = [\mathbf{G}_2]_{\underset{\sim}{\mathsf{LRL}}}$.

Comparison with CPS. Before we finally begin, we would like to give credit and note that our novel canonical form function $\mathsf{CF}_{\mathsf{New}}$ re-uses many of the original ideas by CPS: In Steps 1 to 3, we run essentially an improved variant of the original CPS canonical form

[4]Consider a random variable X with $\Pr[X = 2^n] = 2^{-n}$ and $\Pr[X = 0] = 1 - 2^{-n}$. Then $\mathbb{E}[X] = 1$, but $\Pr[X \geq 1] = 2^{-n}$ is negligible.

[5]More precisely, CPS assume that for any given matrix $\mathbf{G}$ with information set J, the matrix $(\mathbf{G}^J)^{-1} \cdot \mathbf{G}^{\overline{J}}$ obtained from $\mathsf{RREF}(\mathbf{G} \cdot \mathbf{P}^J) = [\mathbf{I}_k \mid (\mathbf{G}^J)^{-1} \cdot \mathbf{G}^{\overline{J}}]$ can be treated as a freshly sampled uniformly random matrix, see [CPS23, Heuristic 1].

function [CPS23] on well-chosen submatrices of our inputs $\mathbf{A}_1$ and $\mathbf{A}_2$. By restricting ourselves to these submatrices, we can circumvent some of the abort conditions of CPS. The fourth step of $\mathsf{CF}_{\mathsf{New}}$ is, however, completely different from the original CPS canonical form function.

4.1 Step 1

Let $\mathbf{G}_1 = [\mathbf{I}_k \mid \mathbf{A}_1] \overset{\mathsf{LRL}}{\sim} \mathbf{G}_2 = [\mathbf{I}_k \mid \mathbf{A}_2] \in \mathbb{F}_q^{k\times n}$ be the inputs to our canonical form function $\mathsf{CF}_{\mathsf{New}}$. By definition of LRL equivalence, we can write

$$\mathbf{A}_2 = \mathbf{P}_r \cdot \mathbf{D}_r \cdot \mathbf{A}_1 \cdot \mathbf{P}_c \cdot \mathbf{D}_c, \tag{2}$$

for some permutations $\mathbf{P}_r \in \Sigma_k$, $\mathbf{P}_c \in \Sigma_{n-k}$ and diagonal matrices $\mathbf{D}_r \in \mathcal{D}_{k,q}$, $\mathbf{D}_c \in \mathcal{D}_{n-k,q}$.

The first step of $\mathsf{CF}_{\mathsf{New}}$ is given in Algorithm $\mathsf{CF}^{(1)}_{\mathsf{New}}$ (Algorithm 3). On inputs $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively, our canonical form function starts by computing $(\mathbf{A}_1^{(1)}, w_1) := \mathsf{CF}^{(1)}_{\mathsf{New}}(\mathbf{A}_1, i_1)$ and $(\mathbf{A}_2^{(1)}, w_2) := \mathsf{CF}^{(1)}_{\mathsf{New}}(\mathbf{A}_2, i_2)$, respectively, where $i_1, i_2 \in [k]$ are some well-chosen parameters. For ease of exposition, we defer the exact description of the selection process for i_1 and i_2 to later. For the moment, it suffices to know that i_1 and i_2 satisfy $i_2 = \mathbf{P}_r[i_1]$, where $\mathbf{P}_r$ is the permutation from Equation (2).

Algorithm 3: $\mathsf{CF}^{(1)}_{\mathsf{New}}$

```
   Input: $\mathbf{A} \in \mathbb{F}_q^{k\times(n-k)}$, index $i \in [k]$.
   Output: $\mathbf{A}^{(1)} \in \mathbb{F}_q^{k\times(n-k)}$, parameter $w \in [n-k]$.
1  $\mathbf{A}^{(1)} := \mathbf{A}$
2  $\mathcal{J} := \emptyset$
3  Parse the $i$-th row of $\mathbf{A}^{(1)}$ as $(a_{i,1}, \ldots, a_{i,n-k})$.
4  for $j = 1, \ldots, n-k$ do
5      if $a_{i,j} \neq 0$ then
6          Divide all entries in the $j$-th column of $\mathbf{A}^{(1)}$ by $a_{i,j}$.
7      else
8          $\mathcal{J} := \mathcal{J} \cup \{j\}$.
9  $w := n - k - |\mathcal{J}|$          ▷ Number of non-zero entries in the $i$-th row of $\mathbf{A}$.
10 Move all columns of $\mathbf{A}^{(1)}$ indexed by $\mathcal{J}$ to the right of the matrix.
11 Swap the first row of $\mathbf{A}^{(1)}$ with the $i$-th row.
12 return $(\mathbf{A}^{(1)}, w)$
```

Relating $\mathbf{A}_1^{(1)}$ and $\mathbf{A}_2^{(1)}$. Let us define $w := w_1$. It is straight-forward to verify that the matrix $\mathbf{A}_1^{(1)}$ is of the shape

$$\mathbf{A}_1^{(1)} = \begin{array}{r} \\ 1\{ \\ k-1\{ \end{array} \begin{bmatrix} \overbrace{1, 1, \ldots, 1}^{w} & \overbrace{0, 0, \ldots, 0}^{n-k-w} \\ \mathbf{A}_{1,1}^{(1)} & \mathbf{A}_{1,2}^{(1)} \end{bmatrix},$$

where $\mathbf{A}_{1,1}^{(1)}$ and $\mathbf{A}_{1,2}^{(1)}$ are some matrices. Furthermore, for our choice of $i_2 = \mathbf{P}_r[i_1]$, it is straight-forward to verify that $w_1 = w_2$, and that

$$\mathbf{A}_2^{(1)} = \begin{array}{r} 1\{ \\ k-1\{ \end{array} \left[\begin{array}{cc} \overbrace{1, 1, \dots, 1}^{w} & \overbrace{0, 0, \dots, 0}^{n-k-w} \\ \mathbf{Q}_r^{(1)} \cdot \mathbf{A}_{1,1}^{(1)} \cdot \mathbf{P}_c^{(1)} & \mathbf{Q}_r^{(1)} \cdot \mathbf{A}_{1,2}^{(1)} \cdot \mathbf{Q}_c^{(1)} \end{array} \right], \quad (3)$$

for some monomials $\mathbf{Q}_r^{(1)}$, $\mathbf{Q}_c^{(1)}$ and a permutation $\mathbf{P}_c^{(1)}$.

Staying in the Equivalence Class. Since the output matrices $\mathbf{A}_i^{(1)}$, $i \in \{1, 2\}$, are obtained by simply permuting and scaling the input matrices $\mathbf{A}_i$, it is clear that the corresponding matrices $\mathbf{G}_i^{(1)} := [\mathbf{I}_k \mid \mathbf{A}_i^{(1)}]$ satisfy $\mathbf{G}_i^{(1)} \in [\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}} = [\mathbf{G}_2]_{\underset{\sim}{\mathsf{LRL}}}$, as required.

As we will see below, the remaining three steps of $\mathsf{CF}_{\mathsf{New}}$ also work by simply permuting and scaling the corresponding input matrices. Thus, throughout the execution of $\mathsf{CF}_{\mathsf{New}}$, we will only compute matrices $\mathbf{G}_i^{(1)}, \dots, \mathbf{G}_i^{(4)}$ from the equivalence class $[\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}} = [\mathbf{G}_2]_{\underset{\sim}{\mathsf{LRL}}}$.

The Value of w. As noted in Line 9 of Algorithm 3, the parameter $w = w_1 = w_2$ is equal to the number of non-zero entries in the i-th row of our input matrix $\mathbf{A}_1$. For uniformly random $\mathbf{A}_1$ (as in \$-LEP) we thus have $\mathbb{E}[w] = (1 - \frac{1}{q})(n - k)$. By the Chernoff bound, w meets its expected value up to a small $(1 \pm \delta)$-factor with overwhelming probability $1 - e^{-\Omega(n-k)}$. In particular, for all $q > 2$ (and large enough $n - k$), we can safely assume that $w \geq \frac{n-k}{2}$.

4.2 Step 2

Step 2 of $\mathsf{CF}_{\mathsf{New}}$ is described in Algorithm $\mathsf{CF}_{\mathsf{New}}^{(2)}$ (Algorithm 4). After computing in Step 1 the matrix $\mathbf{A}_i^{(1)}$ and the parameter w_i, where $i \in \{1, 2\}$, our canonical form function $\mathsf{CF}_{\mathsf{New}}$ proceeds to compute $(\mathbf{A}_i^{(2)}, h_i) := \mathsf{CF}_{\mathsf{New}}^{(2)}(\mathbf{A}_i^{(1)}, w_i)$.

Algorithm 4: $\mathsf{CF}_{\mathsf{New}}^{(2)}$

```
   Input: $\mathbf{A}^{(1)} \in \mathbb{F}_q^{k \times (n-k)}$, parameter $w \in [n-k]$.
   Output: $\mathbf{A}^{(2)} \in \mathbb{F}_q^{k \times (n-k)}$, parameter $h \in [k-1]$.
1  $\mathbf{A}^{(2)} := \mathbf{A}^{(1)}$
2  $\mathcal{I} := \emptyset$
3  for $i = 2, \dots, k$ do
4      Parse the $i$-th row of $\mathbf{A}^{(2)}$ as $(a_{i,1}, \dots, a_{i,n-k})$.
5      $s_i := \sum_{j=1}^{w} a_{i,j}$
6      if $s_i \neq 0$ then
7          Divide all entries in the $i$-th row of $\mathbf{A}^{(2)}$ by $s_i$.
8      else
9          $\mathcal{I} := \mathcal{I} \cup \{i\}$
10 $h := k - |\mathcal{I}| - 1$                ▷ Number of rows of $\mathbf{A}^{(1)}$, for which $s_i \neq 0$.
11 Move all rows of $\mathbf{A}^{(2)}$ indexed by $\mathcal{I}$ to the bottom of the matrix.
12 return $(\mathbf{A}^{(2)}, h)$
```

Relating $\mathbf{A}_1^{(1)}$ and $\mathbf{A}_2^{(1)}$. Let us introduce some notation: Let $s_2, \dots, s_k$ denote the values computed in Line 5 of Algorithm 4, when running the algorithm on input $(\mathbf{A}_1^{(1)}, w_1)$. Analogously, let $\widetilde{s}_2, \dots, \widetilde{s}_k$ denote these values, when running the algorithm on input $(\mathbf{A}_2^{(1)}, w_2)$. Let us write the monomial $\mathbf{Q}_r^{(1)}$ from Equation (6) as $\mathbf{Q}_r^{(1)} = \mathbf{P}_r^{(1)} \cdot \mathbf{D}_r^{(1)}$ for some permutation $\mathbf{P}_r^{(1)}$ and a diagonal matrix $\mathbf{D}_r^{(1)}$. Let $d_1, \dots, d_{k-1}$ denote the diagonal entries of $\mathbf{D}_r^{(1)}$. For $i \in \{2, \dots, k\}$, let $\mathbf{a}_i := (a_{i,1}, \dots, a_{i,n-k})$ denote the i-th row $\mathbf{A}_1^{(1)}$. Let $\widetilde{\mathbf{a}}_i$ denote the i-th row of $\mathbf{A}_2^{(1)}$.

From Equation (3) it follows that

$$\widetilde{\mathbf{a}}_{\mathbf{P}_r^{(1)}[i]} = d_{i-1} \cdot \mathbf{a}_i \cdot \begin{bmatrix} \overbrace{\mathbf{P}_c^{(1)}}^{w} & \overbrace{\phantom{\mathbf{Q}_c^{(1)}}}^{n-k-w} \\ & \mathbf{Q}_c^{(1)} \end{bmatrix}.$$

Thus,

$$\widetilde{s}_{\mathbf{P}_r^{(1)}[i]} = \sum_{j=1}^{w} d_{i-1} \cdot a_{i,(\mathbf{P}_c^{(1)})^T[j]} = d_{i-1} \cdot \sum_{j=1}^{w} a_{i,j} = d_{i-1} \cdot s_i. \tag{4}$$

Hence, if $s_i = 0$, then $\mathsf{CF}_{\mathsf{New}}^{(2)}$ moves both the i-th row of $\mathbf{A}_1^{(1)}$ and the $\mathbf{P}_r^{(1)}[i]$-th row of $\mathbf{A}_2^{(1)}$ to the bottom of $\mathbf{A}_1^{(2)}$ and $\mathbf{A}_2^{(2)}$, respectively. On the other hand, if $s_i \neq 0$, then the i-th row $\mathbf{a}_i$ of $\mathbf{A}_1^{(1)}$ gets replaced by $\frac{1}{s_i} \cdot \mathbf{a_i}$, whereas the $\mathbf{P}_r^{(1)}[i]$-th row of $\mathbf{A}_2^{(1)}$ gets replaced by

$$\frac{1}{\widetilde{s}_{\mathbf{P}_r^{(1)}[i]}} \cdot \widetilde{\mathbf{a}}_{\mathbf{P}_r^{(1)}[i]} = \frac{1}{s_i} \cdot \mathbf{a}_i \cdot \begin{bmatrix} \overbrace{\mathbf{P}_c^{(1)}}^{w} & \overbrace{\phantom{\mathbf{Q}_c^{(1)}}}^{n-k-w} \\ & \mathbf{Q}_c^{(1)} \end{bmatrix}.$$

This shows that for all i with $s_i \neq 0$ in the resulting matrices $\mathbf{A}_1^{(2)}$ and $\mathbf{A}_2^{(2)}$, the first w entries of the rows obtained from $\mathbf{a}_i$ and $\widetilde{\mathbf{a}}_{\mathbf{P}_r^{(1)}[i]}$ are *identical up to permutation.*

Let us define $h := h_1$. Since h_1 is the number of rows of $\mathbf{A}^{(1)}$, for which $s_i \neq 0$ (see Line 5 of Algorithm 4), we have by Equation (4) that $h = h_1 = h_2$. Let us write $\mathbf{A}_2^{(2)}$ as

$$\mathbf{A}_1^{(2)} = \begin{array}{r} \\ 1\{ \\ h\{ \\ k-h-1\{ \end{array} \begin{bmatrix} \overbrace{}^{w} & \overbrace{}^{n-k-w} \\ 1,1,\dots,1 & 0,0,\dots,0 \\ \mathbf{A}_{1,1}^{(2)} & \mathbf{A}_{1,2}^{(2)} \\ \mathbf{A}_{1,3}^{(2)} & \mathbf{A}_{1,4}^{(2)} \end{bmatrix}, \tag{5}$$

for some matrices $\mathbf{A}_{1,1}^{(2)}, \dots, \mathbf{A}_{1,4}^{(2)}$. By the discussion above, we have

$$\mathbf{A}_2^{(2)} = \begin{array}{r} \\ 1\{ \\ h\{ \\ k-h-1\{ \end{array} \begin{bmatrix} \overbrace{\phantom{\mathbf{P}_r^{(2)} \cdot \mathbf{A}_{1,1}^{(2)} \cdot \mathbf{P}_c^{(2)}}}^{w} & \overbrace{\phantom{\mathbf{P}_r^{(2)} \cdot \mathbf{A}_{1,2}^{(2)} \cdot \mathbf{Q}_c^{(2)}}}^{n-k-w} \\ 1,1,\dots,1 & 0,0,\dots,0 \\ \mathbf{P}_r^{(2)} \cdot \mathbf{A}_{1,1}^{(2)} \cdot \mathbf{P}_c^{(2)} & \mathbf{P}_r^{(2)} \cdot \mathbf{A}_{1,2}^{(2)} \cdot \mathbf{Q}_c^{(2)} \\ \mathbf{Q}_r^{(2)} \cdot \mathbf{A}_{1,3}^{(2)} \cdot \mathbf{P}_c^{(2)} & \mathbf{Q}_r^{(2)} \cdot \mathbf{A}_{1,4}^{(2)} \cdot \mathbf{Q}_c^{(2)} \end{bmatrix}, \tag{6}$$

for some monomials $\mathbf{Q}_r^{(2)}$, $\mathbf{Q}_c^{(2)}$ and permutations $\mathbf{P}_r^{(2)}$, $\mathbf{P}_c^{(2)}$. In particular, the upper left $((h+1) \times w)$-blocks of $\mathbf{A}_1^{(2)}$ and $\mathbf{A}_2^{(2)}$ are identical up to row and column permutation.

The Value of h. As noted in Line 5 of Algorithm 4, the parameter $h = h_1 = h_2$ is equal to the number of rows of $\mathbf{A}^{(1)}$, for which $s_i \neq 0$. It is easy to see that for uniformly random inputs $\mathbf{A}_1$ to $\mathsf{CF}_{\mathsf{New}}^{(1)}$, the second to k-th rows of the outputs $\mathbf{A}_1^{(1)}$ are still uniformly random. Hence, the s_i's are uniformly random over $\mathbb{F}_q$, and we have $\mathbb{E}[h] = (1 - \frac{1}{q})(k-1)$. Arguing exactly as for the parameter w in the previous section, it follows that for all $q > 2$ (and large enough k), we can safely assume that $h \geq \frac{k-1}{2}$.

4.3 Step 3

As shown in the previous section, the upper left $((h+1) \times w)$-blocks of the matrices $\mathbf{A}_1^{(2)}$ and $\mathbf{A}_2^{(2)}$ obtained from $\mathsf{CF}_{\mathsf{New}}^{(2)}$ are identical up to row and column permutation. This observation lets us now easily transform $\mathbf{A}_1^{(2)}$ and $\mathbf{A}_2^{(2)}$ via a simple sorting procedure into matrices $\mathbf{A}_1^{(3)}$ and $\mathbf{A}_2^{(3)}$, in which the upper left $((h+1) \times w)$-blocks are *identical.* More precisely, we can easily compute matrices of the forms

$$\mathbf{A}_1^{(3)} = \begin{array}{r} 1\{ \\ h\{ \\ k-h-1\{ \end{array} \begin{bmatrix} \overbrace{1,1,\ldots,1}^{w} & \overbrace{0,0,\ldots,0}^{n-k-w} \\ \mathbf{A}_{1,1}^{(3)} & \mathbf{A}_{1,2}^{(3)} \\ \mathbf{A}_{1,3}^{(3)} & \mathbf{A}_{1,4}^{(3)} \end{bmatrix}, \tag{7}$$

$$\mathbf{A}_2^{(3)} = \begin{array}{r} 1\{ \\ h\{ \\ k-h-1\{ \end{array} \begin{bmatrix} \overbrace{1,1,\ldots,1}^{w} & \overbrace{0,0,\ldots,0}^{n-k-w} \\ \mathbf{A}_{1,1}^{(3)} & \mathbf{A}_{1,2}^{(3)} \cdot \mathbf{Q}_c^{(3)} \\ \mathbf{Q}_r^{(3)} \cdot \mathbf{A}_{1,3}^{(3)} & \mathbf{Q}_r^{(3)} \cdot \mathbf{A}_{1,4}^{(3)} \cdot \mathbf{Q}_c^{(3)} \end{bmatrix}, \tag{8}$$

Our sorting procedure is described in Algorithm $\mathsf{CF}_{\mathsf{New}}^{(3)}$ (Algorithm 5). From Equation (6) it immediately follows that for $\mathbf{A}_i^{(3)} := \mathsf{CF}_{\mathsf{New}}^{(3)}(\mathbf{A}_i^{(2)}, w_i, h_i)$, $i \in \{1,2\}$ the outputs $\mathbf{A}_i^{(3)}$ indeed have the desired shape as in Equations (7) and (8) – provided, of course, that $\mathsf{CF}_{\mathsf{New}}^{(3)}$ does not return $\perp$ in Line 9 . Fortunately, as we show below, for all fields of size of $q \geq 7$, the probability of $\mathsf{CF}_{\mathsf{New}}^{(3)}$ not returning $\perp$ is close to 1.

Success Probability. The probability that $\mathsf{CF}_{\mathsf{New}}^{(3)}$ returns $\perp$ on input $\mathbf{A}_i^{(2)}$ is the probability that the matrix $\mathbf{A}_{1,1}^{(2)}$ from Equation (5) contains two rows or two columns, that are identical up to permutation. To compute this probability we introduce two novel technical lemmas.

Lemma 4.1. *Let $\mathbf{B} \in \mathbb{F}_q^{h \times w}$ be a uniformly random matrix. Let P denote the probability that two rows of $\mathbf{B}$ are identical up to permutation. If $q \geq 7$, then $P = \mathcal{O}(h^2 \cdot w^{-3})$.*

Proof. The larger q, the smaller P. Thus, to prove our upper bound of $P = \mathcal{O}(h^2 \cdot w^{-3})$ for all $q \geq 7$, it suffices to prove it for the special case of $q = 7$.

Let $\mathbf{v} = (v_0, v_1, \ldots, v_6) \in \mathbb{N}^7$, such that $\sum_{i=0}^{7} v_i = w$. For $i_1, i_2 \in [h]$ with $i_1 \neq i_2$, let $P_{i_1,i_2,\mathbf{v}}$ denote the probability that the i_1-th row, the i_2-th row, and the vector $(0^{v_0}, 1^{v_1}, \ldots, 6^{v_6}) \in \mathbb{F}_7^w$ are up to identical permutation. Then the probability that the i_1-th and i_2-th row are identical up to permutation is given by

$$P_{i_1,i_2} := \sum_{v_0+v_1+\ldots+v_6=w} P_{i_1,i_2,\mathbf{v}} = \sum_{v_0+v_1+\ldots+v_6=w} \left(\frac{1}{7^w} \cdot \binom{w}{v_0, v_1, \ldots, v_6} \right)^2 .$$

Algorithm 5: $\mathsf{CF}^{(3)}_{\mathsf{New}}$

```
    Input: $\mathbf{A}^{(2)} \in \mathbb{F}_q^{k\times(n-k)}$, parameters $w \in [n-k]$, $h \in [k-1]$.
    Output: $\mathbf{A}^{(3)} \in \mathbb{F}_q^{k\times(n-k)}$, or error symbol $\perp$.
 1  $\mathbf{A}^{(3)} := \mathbf{A}^{(2)}$
 2  for $i = 2, \dots, h+1$ do
 3      Parse the $i$-th row of $\mathbf{A}^{(3)}$ as $(a_{i,1}, \dots, a_{i,n-k})$.
 4      Let $R_i$ denote the multiset $(a_{i,1}, \dots, a_{i,w})$.
 5  for $j = 1, \dots, w$ do
 6      Parse the $j$-th column of $\mathbf{A}^{(3)}$ as $(a_{1,j}, \dots, a_{k,j})^T$.
 7      Let $C_j$ denote the multiset $(a_{2,j}, \dots, a_{h+1,j})$.
 8  if the $R_i$'s or the $C_j$'s are not pairwise distinct then
 9      return $\perp$
10  Sort the 2nd to $(h+1)$-th rows of $\mathbf{A}^{(3)}$ according to an lexicographic ordering of
      the multisets $R_2, \dots, R_{h+1}$.
11  Sort the 1st to $w$-th columns of $\mathbf{A}^{(3)}$ according to an lexicographic ordering of the
      multisets $C_1, \dots, C_w$.
12  return $\mathbf{A}^{(3)}$
```

As shown in [RS09, Theorem 4], it holds that

$$\sum_{v_0+v_1+\ldots+v_6=w} \binom{w}{v_0, v_1, \dots, v_6}^2 \sim 7^{2w+7/2}(4\pi w)^{(1-7)/2}.$$

Thus,

$$P_{i_1,i_2} \sim 7^{7/2} \cdot (4\pi w)^{-3} = \mathcal{O}(w^{-3}),$$

Taking a union bound over the $\Theta(h^2)$ pairs (i_1, i_2), the lemma follows. ☐

We like to point out that in [CPS23, Theorem 3], CPS already showed that for $q = \Omega(n)$, a uniformly random matrix from $\mathbb{F}_q^{k\times(n-k)}$ has a constant probability $\Omega(1)$ of having no two rows identical up to permutation. Our novel Lemma 4.1 significantly improves upon this CPS result.

Applying Lemma 4.1 to the transpose of a uniformly random matrix $\mathbf{B} \in \mathbb{F}_q^{h\times w}$ immediately yields the following lemma.

Lemma 4.2. *Let $\mathbf{B} \in \mathbb{F}_q^{h\times w}$ be a uniformly random matrix. Let P denote the probability that two columns of $\mathbf{B}$ are identical up to permutation. If $q \geq 7$, then $P = \mathcal{O}(w^2 \cdot h^{-3})$.*

Carefully inspecting Algorithms 3 and 4, it easily follows that for a uniformly random input $\mathbf{A}_1 \in \mathbb{F}_q^{k\times(n-k)}$ to $\mathsf{CF}_{\mathsf{New}}$, the matrix $\mathbf{A}^{(2)}_{1,1} \in \mathbb{F}_q^{h\times w}$ from Equation (5), obtained after running $\mathsf{CF}^{(1)}_{\mathsf{New}}$ and $\mathsf{CF}^{(2)}_{\mathsf{New}}$, is still uniformly random. Hence, Lemmas 4.1 and 4.2 show that for $q \geq 7$, the probability that $\mathsf{CF}^{(3)}_{\mathsf{New}}$ returns $\perp$ is $\mathcal{O}(h^2 \cdot w^{-3}) + \mathcal{O}(w^2 \cdot h^{-3})$. By construction, we can upper bound w and h by $w \leq n-k$ and $h \leq k-1$. As discussed in the previous two sections, with overwhelming probabilities $1 - e^{-\Omega(n-k)}$ and $1 - e^{-\Omega(k)}$, we can lower bound w and h by $w \geq \frac{n-k}{2}$ and $h \geq \frac{k-1}{2}$. Thus, for the typical parameter setting of constant rate, we have $w = \Theta(n)$ and $h = \Theta(n)$. Hence, the probability that $\mathsf{CF}^{(3)}_{\mathsf{New}}$ returns $\perp$ is $\mathcal{O}(n^2 \cdot n^{-3}) = \mathcal{O}(n^{-1})$ – showing that with probability $1 - \mathcal{O}(n^{-1})$ the output of $\mathsf{CF}^{(3)}_{\mathsf{New}}$ indeed has the desired shape.

4.4 Step 4

By Equations (7) and (8), the upper left $((h+1) \times w)$-blocks of the matrices $\mathbf{A}_1^{(3)}$ and $\mathbf{A}_2^{(3)}$ obtained from $\mathsf{CF}_{\mathsf{New}}^{(3)}$ are identical up to row and column permutation. Additionally, the upper right $((h+1) \times (n-k-w))$-blocks are identical up to a monomial transformation *from the right.* The lower left $((k-h-1) \times w)$-blocks are identical up to a monomial transformation *from the left.* Dealing with monomials that act *only on one side* of the matrix is much easier, than dealing with monomials that act on both sides (as we had to in the previous three sections). Indeed, if we now appropriately divide the $(w+1)$-th to $(n-k)$-th columns, and the $(h+2)$-th to k-th rows of our matrices, we can easily turn them into matrices that are identical up to row and column permutation. After that, we simply invoke our algorithm from Step 3 once more to sort our matrices. Thereby, we finally obtain *identical* matrices $\mathbf{A}_2^{(4)} = \mathbf{A}_1^{(1)}$. Our approach is formally described in Algorithm $\mathsf{CF}_{\mathsf{New}}^{(4)}$ (Algorithm 6). It is easy to see that its output has the desired shape, i.e., for $\mathbf{A}_i^{(4)} := \mathsf{CF}_{\mathsf{New}}^{(4)}(\mathbf{A}_i^{(3)}, w_i, h_i)$ we have $\mathbf{A}_2^{(4)} = \mathbf{A}_1^{(1)}$ – provided that the algorithm does not return $\perp$.

Success Probability. The probability that Algorithm 6 aborts in Line 17 is exponentially small. (The algorithm aborts here only if one of the uniformly random matrices $\mathbf{A}_{1,2}^{(3)}$, $\mathbf{A}_{1,3}^{(3)}$ from Equation (7) contains an all-zero row or column.) Furthermore, as detailed in the previoius section, the probability that Algorithm 6 aborts in Line 18 is upper bounded by $\mathcal{O}(n^{-1})$. Hence, with probability $1 - \mathcal{O}(n^{-1})$ the output of $\mathsf{CF}_{\mathsf{New}}^{(4)}$ indeed has the desired shape.

Algorithm 6: $\mathsf{CF}_{\mathsf{New}}^{(4)}$

```
Input: $\mathbf{A}^{(3)} \in \mathbb{F}_q^{k \times (n-k)}$, parameters $w \in [n-k]$, $h \in [k-1]$.
Output: $\mathbf{A}^{(4)} \in \mathbb{F}_q^{k \times (n-k)}$, or error symbol $\perp$.
1  $\mathbf{A}^{(4)} := \mathbf{A}^{(3)}$
2  $\mathcal{J} := \{w+1, w+2, \dots, n-k\}$
3  $\mathcal{I} := \{h+2, h+3, \dots, k\}$
4  for $i = 2, \dots, h+1$ do
5      Parse the $i$-th row of $\mathbf{A}^{(4)}$ as $(a_{i,1}, \dots, a_{i,n-k})$.
6      for $j \in \mathcal{J}$ do
7          if $a_{i,j} \neq 0$ then
8              Divide the $j$-th column of $\mathbf{A}^{(4)}$ by $a_{i,j}$.
9              Remove $j$ from $\mathcal{J}$.
10 for $j = 1, \dots, w$ do
11     Parse the $j$-th column of $\mathbf{A}^{(4)}$ as $(a_{1,j}, \dots, a_{k,j})^T$.
12     for $i \in \mathcal{I}$ do
13         if $a_{i,j} \neq 0$ then
14             Divide the $i$-th row of $\mathbf{A}^{(4)}$ by $a_{i,j}$.
15             Remove $i$ from $\mathcal{I}$.
16 if $\mathcal{J} \neq \emptyset$ or $\mathcal{I} \neq \emptyset$ then
17     return $\perp$
18 $\mathbf{A}^{(3)} := \mathsf{CF}_{\mathsf{New}}^{(3)}(\mathbf{A}^{(3)}, n-k, k)$
19 return $\mathbf{A}^{(3)}$
```

4.5 Putting Everything Together

We are now almost ready to fully describe our novel canonical form function. The only thing left to do, is describing how to pick the inputs i_1 and i_2, in Step 1. (Recall that in Step 1, we want compute $(\mathbf{A}_1^{(1)}, w_1) := \mathsf{CF}^{(1)}_{\mathsf{New}}(\mathbf{A}_1, i_1)$ and $(\mathbf{A}_2^{(1)}, w_2) := \mathsf{CF}^{(1)}_{\mathsf{New}}(\mathbf{A}_2, i_2)$, where $i_2 = \mathbf{P}_r[i_1]$, and $\mathbf{P}_r$ is the permutation from Equation (2).) To this end, we re-use an idea by CPS: On input $\mathbf{G}_1 = [\mathbf{I}_k \mid \mathbf{A}_1]$, we iterate over all values $i_1 = 1, 2, \dots, k$. For each i_1, we run Steps 1 to 4, such that in the end we obtain a list L_1 of (up to) k matrices from the equivalence class $[\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}}$. We sort L_1 lexicographically and then output the first entry. Analogously, on input $\mathbf{G}_2 = [\mathbf{I}_k \mid \mathbf{A}_2]$, we iterate over all values $i_2 = 1, 2, \dots, k$ to obtain a list of matrices L_2. For any i_1, the i_1-th entry in L_1 is then identical to the $\mathbf{P}_r[i_1]$-th entry in L_2. Hence, by lexicographically sorting L_1 and L_2, we output the same representative from the equivalence class $[\mathbf{G}_1]_{\underset{\sim}{\mathsf{LRL}}} = [\mathbf{G}_2]_{\underset{\sim}{\mathsf{LRL}}}$.

The full description of $\mathsf{CF}_{\mathsf{New}}$ is given in Algorithm 7. We remark that Definition 3.2 technically requires a canonical form function not only to output both a canonoical representative $\mathbf{G}^* = [\mathbf{I}_k \mid \mathbf{A}^*] \in [\mathbf{G}]_{\underset{\sim}{\mathsf{LRL}}}$, but also to output monomials $\mathbf{Q}_r, \mathbf{Q}_c$, satisfying $\mathbf{A}^* = \mathbf{Q}_r \cdot \mathbf{A} \cdot \mathbf{Q}_c$. For ease of notation, we omit these monomials in the description of Algorithm 7. In practice, the monomials can easily be computed. To this end, one simply has to keep track of the monomial transformations made by $\mathsf{CF}^{(1)}_{\mathsf{New}}, \dots, \mathsf{CF}^{(4)}_{\mathsf{New}}$. (See also our proof-of-concept implementation, available in GitHub.)

Algorithm 7: $\mathsf{CF}_{\mathsf{New}}$

```
   Input: $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}] \in \mathbb{F}_q^{k\times n}$
   Output: Canonical representative $\mathbf{G}^* = [\mathbf{I}_k \mid \mathbf{A}^*]$ of $[\mathbf{G}]_{\underset{\sim}{\mathsf{LRL}}}$, or error symbol $\perp$.
1  Initialize empty list $L$.
2  for $i = 1, \dots, k$ do
3      $(\mathbf{A}^{(1)}, w) := \mathsf{CF}^{(1)}_{\mathsf{New}}(\mathbf{A})$
4      $(\mathbf{A}^{(2)}, h) := \mathsf{CF}^{(2)}_{\mathsf{New}}(\mathbf{A}^{(1)}, w)$
5      $\mathbf{A}^{(3)} := \mathsf{CF}^{(3)}_{\mathsf{New}}(\mathbf{A}^{(2)}, w, h)$
6      if $\mathbf{A}^{(3)} \neq \perp$ then
7          $\mathbf{A}^{(4)} := \mathsf{CF}^{(4)}_{\mathsf{New}}(\mathbf{A}^{(3)}, w, h)$
8          if $\mathbf{A}^{(4)} \neq \perp$ then
9              Add $[\mathbf{I}_k \mid \mathbf{A}^{(4)}]$ to $L$.
10 if $L$ is not empty then
11     return the lexicographically first entry in $L$.
12 else
13     return $\perp$
```

Summarizing the above four sections, we finally obtain the following theorem:

Theorem 4.3 (Correctness $\mathsf{CF}_{\mathsf{New}}$). *For all $q \geq 7$ and constant rate $\frac{k}{n}$, the canonical form function $\mathsf{CF}_{\mathsf{New}}$ has success probability*

$$\gamma_{\mathsf{CF}_{\mathsf{New}}}(n,k,q) = \Pr_{\mathbf{A} \leftarrow \mathbb{F}_q^{k\times(n-k)}} \left[\mathsf{CF}_{\mathsf{New}}\big([\mathbf{I}_k \mid \mathbf{A}]\big) \neq \perp\right] \geq 1 - \mathcal{O}(n^{-1}).$$

Combining Theorems 3.6 and 4.3, our main result follows:

Theorem 4.4 (Main Result). *For all $q \geq 7$ and constant rate $\frac{k}{n}$, there is an algorithm that solves \$-LEP with parameters (n,k,q) in time $\widetilde{\Theta}\left(2^{\frac{1}{2}\mathrm{H}(\frac{k}{n})n}\right)$, and with constant success probability.*

References

[BBPS23] Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. On the computational hardness of the code equivalence problem in cryptography. *Adv. Math. Commun.*, 17(1):23–55, 2023. URL: https://doi.org/10.3934/amc.2022064, doi:10.3934/AMC.2022064.

[Beu20] Ward Beullens. Not enough LESS: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 387–403. Springer, Cham, October 2020. doi:10.1007/978-3-030-81652-0_15.

[BMPS20] Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. LESS is more: Code-based signatures without syndromes. In Abderrahmane Nitaj and Amr M. Youssef, editors, *AFRICACRYPT 20: 12th International Conference on Cryptology in Africa*, volume 12174 of *Lecture Notes in Computer Science*, pages 45–65. Springer, Cham, July 2020. doi:10.1007/978-3-030-51938-4_3.

[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 Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *AFRICACRYPT 23: 14th International Conference on Cryptology in Africa*, volume 14064 of *Lecture Notes in Computer Science*, pages 28–52. Springer, Cham, July 2023. doi:10.1007/978-3-031-37679-5_2.

[CPS23] Tung Chou, Edoardo Persichetti, and Paolo Santini. On linear equivalence, canonical forms, and digital signatures. *IACR Cryptol. ePrint Arch.*, page 1533, 2023. URL: https://eprint.iacr.org/2023/1533.

[DPPv22] Léo Ducas, Eamonn W. Postlethwaite, Ludo N. Pulles, and Wessel P. J. van Woerden. Hawk: Module LIP makes lattice signatures fast, compact and simple. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 65–94. Springer, Cham, December 2022. doi:10.1007/978-3-031-22972-5_3.

[Leo82] Jeffrey S. Leon. Computing automorphism groups of error-correcting codes. *IEEE Trans. Inf. Theory*, 28(3):496–510, 1982. doi:10.1109/TIT.1982.1056498.

[PS23] Edoardo Persichetti and Paolo Santini. A new formulation of the linear equivalence problem and shorter LESS signatures. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VII*, volume 14444 of *Lecture Notes in Computer Science*, pages 351–378. Springer, Singapore, December 2023. doi:10.1007/978-981-99-8739-9_12.

[RS09] L. Bruce Richmond and Jeffrey O. Shallit. Counting abelian squares. *Electron. J. Comb.*, 16(1), 2009. doi:10.37236/161.

[Sen00] Nicolas Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Trans. Inf. Theory*, 46(4):1193–1203, 2000. doi:10.1109/18.850662.

[SS13] Nicolas Sendrier and Dimitrios E. Simos. How easy is code equivalence over fq? 2013.

[vW99] Paul C. van Oorschot and Michael J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, January 1999. doi:10.1007/PL00003816.

A Appendix: Additional Proofs

A.1 Lemma 2.5

Lemma 2.5. *Let $X_1, \ldots, X_n \in \{0,1\}$ denote (possibly dependent) random variables. Let $p \in [0,1]$, such that $\Pr[X_i = 1] > p$ for every $i \in [n]$. Then for $X := \sum_{i=1}^{n} X_i$ it holds that*

$$\Pr\left[X > \frac{p}{2} \cdot n\right] > \frac{p}{2}.$$

Proof. Let us define $Y := n - X$. Using $\mathbb{E}[Y] = n - \mathbb{E}[X] < (1-p) \cdot n$, and applying Markov's inequality to the non-negative random variable Y, we obtain

$$\Pr\left[Y \geq \left(1 - \frac{p}{2}\right) \cdot n\right] \leq \Pr\left[Y \geq \frac{1 - \frac{p}{2}}{1-p} \cdot \mathbb{E}[Y]\right] \leq \frac{1-p}{1-\frac{p}{2}} = 1 - \frac{p}{2-p} < 1 - \frac{p}{2},$$

and conversely

$$\Pr\left[X > \frac{p}{2} \cdot n\right] = \Pr\left[Y < \left(1 - \frac{p}{2}\right) \cdot n\right] > \frac{p}{2},$$

as required. □

A.2 Lemma 2.6

Lemma 2.6. *Let q be a prime power and let $k \in \mathbb{N}$. If we sample a matrix $\mathbf{A}$ uniformly at random from $\mathbb{F}_q^{k \times k}$, then $\mathbf{A}$ is invertible with probability greater than $\frac{1}{4}$.*

Proof. A uniformly random matrix is invertible with probability

$$\prod_{i=0}^{k-1}(1 - q^{i-k}) > \prod_{i=1}^{\infty}(1 - q^{-i}) \geq \prod_{i=1}^{\infty}(1 - 2^{-i}).$$

Numerically evaluating the last of the above products shows that the probability is lower bounded by $0.288 > \frac{1}{4}$. □

A.3 Lemma 3.4

Lemma 3.4 (Adapted from Proposition 11 in [CPS23])**.** *Let $\mathbf{G}_1 \sim \mathbf{G}_2$ be an LEP instance, and let CF be a canonical form function. Let J_1, J_2 be CF-colliding information sets for $(\mathbf{G}_1, \mathbf{G}_2)$. On input $\mathbf{G}_1, \mathbf{G}_2, J_1, J_2$, algorithm $\mathsf{RecoverMon}^{\mathsf{CF}(\cdot)}$ (Algorithm 1) computes a solution $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$, $\mathbf{Q} \in \mathcal{M}_{n,q}$ to the LEP instance defined by $\mathbf{G}_1$ and $\mathbf{G}_2$ in polynomial time.*

Proof. Algorithm 1 starts by computing $\mathbf{G}'_i := \mathsf{RREF}(\mathbf{G}_i \cdot \mathbf{P}^{J_i}) \in \mathbb{F}_q^{k \times (n-k)}$. Since the J_i's are information sets, we have $\mathbf{G}'_i = [\mathbf{I}_k \mid \mathbf{A}_i]$, where

$$\mathbf{A}_i = (\mathbf{G}_i^{J_i})^{-1} \cdot \mathbf{G}_i^{\overline{J_i}} \in \mathbb{F}_q^{k \times (n-k)}. \tag{9}$$

In particular, since the $\mathbf{G}_i$'s are in systematic form, they are valid inputs for CF. (Recall that CF is only defined for inputs in systematic form, see Definition 3.2.) Since the J_i's are CF-colliding, we have $\mathsf{CF}(\mathbf{G}_i') \neq \perp$. Hence, in Line 2, Algorithm 1 indeed obtains a tuple $(\mathbf{G}_i^*, \mathbf{Q}_{r,i}, \mathbf{Q}_{c,i})$. By Definition 3.2, we have

$$\mathbf{G}_i^* = [\mathbf{I}_k \mid \mathbf{A}_i^*], \quad \text{for } \mathbf{A}_i^* = \mathbf{Q}_{r,i} \cdot \mathbf{A}_i \cdot \mathbf{Q}_{c,i},$$

Moreover, we have by Definition 3.2 that $\mathbf{A}_1^* = \mathbf{A}_2^*$, and thus

$$\mathbf{A}_2 = \mathbf{Q}_{r,2}^{-1} \cdot \mathbf{Q}_{r,1} \cdot \mathbf{A}_1 \cdot \mathbf{Q}_{c,1} \cdot \mathbf{Q}_{c,2}^{-1}. \tag{10}$$

Let $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$ and $\mathbf{Q} \in \mathcal{M}_{n,q}$ denote the matrices computed by Algorithm 1. (We have $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$, since the J_i's are information sets. Furthermore, we have $\mathbf{Q} \in \mathcal{M}_{n,q}$, since the $\mathbf{P}^{J_i} \in \Sigma_n \subseteq \mathcal{M}_{n,q}$'s are monomials.) A tedious but straight-forward computation shows that

$$\mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q} = \mathbf{G}_2^{J_2} \cdot [\mathbf{I}_k \mid \mathbf{Q}_{r,2}^{-1} \cdot \mathbf{Q}_{r,1} \cdot (\mathbf{G}_1^{J_1})^{-1} \cdot \mathbf{G}_1^{\overline{J_1}} \cdot \mathbf{Q}_{c,1} \cdot \mathbf{Q}_{c,2}^{-1}] \cdot (\mathbf{P}^{J_2})^{-1}.$$

Using Equations (9) and (10), one can simplify the above equation as

$$\mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{Q} = \mathbf{G}_2,$$

which shows that $\mathbf{U}$ and $\mathbf{Q}$ form a solution to the LEP instance defined by $\mathbf{G}_1$ and $\mathbf{G}_2$. Since Algorithm 1 clearly runs in polynomial time, this proves the lemma.

□

A.4 Lemma 3.5

Lemma 3.5. *Let $\mathbf{G}_1 \sim \mathbf{G}_2 \in \mathbb{F}_q^{k \times n}$ be linearly equivalent matrices, where*

$$\mathbf{G}_2 = \mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{P} \cdot \mathbf{D},$$

for some $\mathbf{U} \in \mathrm{GL}(\mathbb{F}_q^k)$, $\mathbf{P} \in \Sigma_n$ and $\mathbf{D} \in \mathcal{D}_{n,q}$. Let J_1 be an information set of $\mathbf{G}_1$. Then $J_2 := \mathbf{P}[J_1]$ is an information set of $\mathbf{G}_2$, and it holds that

$$\mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1}) \overset{\mathsf{LRL}}{\sim} \mathsf{RREF}(\mathbf{G}_2 \cdot \mathbf{P}^{J_2}).$$

Proof. By definition of J_2, we have

$$\mathbf{G}_1 \cdot \mathbf{P} \cdot \mathbf{Q} \cdot \mathbf{P}^{J_2} = \left[\mathbf{G}^{J_1} \cdot \mathbf{Q}_r \mid \mathbf{G}^{\overline{J_1}} \cdot \mathbf{Q}_c\right]$$

for some $\mathbf{Q}_r \in \mathcal{M}_{k,q}$ and $\mathbf{Q}_c \in \mathcal{M}_{n-k,q}$. Since RREF is invariant under invertible transformations from the left, this yields that

$$\begin{aligned}\mathsf{RREF}(\mathbf{G}_2 \cdot \mathbf{P}^{J_2}) &= \mathsf{RREF}(\mathbf{U} \cdot \mathbf{G}_1 \cdot \mathbf{P} \cdot \mathbf{D} \cdot \mathbf{P}^{J_2}) \\ &= \mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P} \cdot \mathbf{D} \cdot \mathbf{P}^{J_2}) \\ &= \mathsf{RREF}\left(\left[\mathbf{G}_1^{J_1} \cdot \mathbf{Q}_r \mid \mathbf{G}^{\overline{J_1}} \cdot \mathbf{Q}_c\right]\right).\end{aligned}$$

Since J_1 is an information set of $\mathbf{G}_1$, the matrix $\mathbf{G}_1^{J_1} \cdot \mathbf{Q}_r$ has full rank. Hence,

$$\mathsf{RREF}(\mathbf{G}_2 \cdot \mathbf{P}^{J_2}) = \left[\mathbf{I}_k \mid \mathbf{Q}_r^{-1} \cdot (\mathbf{G}_1^{J_1})^{-1} \cdot \mathbf{G}^{\overline{J_1}} \cdot \mathbf{Q}_c\right].$$

This shows that J_2 is an information set of $\mathbf{G}_2$ (since this shows that $\mathbf{G}_2^{J_2}$ is invertible). Furthermore, this shows that

$$\mathsf{RREF}(\mathbf{G}_2 \cdot \mathbf{P}^{J_2}) \overset{\mathsf{LRL}}{\sim} \left[\mathbf{I}_k \mid (\mathbf{G}_1^{J_1})^{-1} \cdot \mathbf{G}^{\overline{J_1}}\right] = \mathsf{RREF}(\mathbf{G}_1 \cdot \mathbf{P}^{J_1}),$$

and thus proves the lemma.

□