On the Sample Complexity of Linear Code Equivalence for All Code Rates

Alessandro Budroni[1*] and Andrea Natale[2]

[1*]Cryptography Research Center, Technology Innovation Institute, UAE.
[2]Department of Mathematics, University of Trento, Italy.[†]

*Corresponding author(s). E-mail(s): alessandro.budroni@tii.ae;
Contributing authors: natalea00@gmail.com;

Abstract

In parallel with the standardization of lattice-based cryptosystems, the research community in Post-quantum Cryptography focused on non-lattice-based hard problems for constructing public-key cryptographic primitives. The Linear Code Equivalence (LCE) Problem has gained attention regarding its practical applications and cryptanalysis. Recent advancements, including the LESS signature scheme and its candidacy in the NIST standardization for additional signatures, supported LCE as a foundation for post-quantum cryptographic primitives. However, recent cryptanalytic results have revealed vulnerabilities in LCE-based constructions when multiple related public keys are available for one specific code rate. In this work, we generalize the proposed attacks to cover all code rates. We show that the complexity of recovering the private key from multiple public keys is significantly reduced for any code rate scenario. Thus, we advise against constructing specific cryptographic primitives using LCE.

Keywords: Sample Complexity, Code Equivalence, Cryptanalysis, Post-quantum Cryptography

1 Introduction

In view of the advent of quantum computers, post-quantum cryptography is currently in the phase of being standardized for widespread use. In 2022, the National Institute

[†]This work was conducted during Andrea Natale's internship at the Cryptography Research Center, Technology Innovation Institute, UAE.

of Standards and Technology (NIST) announced the first set of post-quantum key-agreement and signature schemes selected for standardization [1]. Subsequently, NIST introduced a new competition to select additional signature schemes based on hard problems that are different from those already chosen [2].

The Linear Code Equivalence (LCE) Problem has emerged as a robust hard problem to rely on for building new post-quantum schemes. Notably, the LESS signature scheme [3], which relies on the computational hardness of solving LCE, is currently a running candidate to the NIST's ongoing standardization of additional digital signature schemes [4]. The development of LESS, along with advancements in group action-based cryptography, has inspired researchers to develop signature schemes with advanced functionalities based on LCE, such as threshold signatures [5], ring signatures, and identity-based signatures [6]. However, recent works [7, 8] have shown that LCE, due to its linear nature, is not suitable for securely constructing certain group action-based primitives, e.g., the group action-based updatable encryption proposed in [9]. They demonstrated that having access to several different LCE public keys corresponding to a single private key allows efficient recovery of it in polynomial time.

Additionally, a new polynomial-time algorithm was introduced in [8] that, when the code rate is $1/2$ (i.e., when the code length is double its dimension), can retrieve an LCE private key from just two related public keys. This result demonstrated a vulnerability in certain cryptographic constructions relying on this assumption. In particular, it impacted the key distribution of the threshold signature from [5]. Moreover, while [6] discussed the possibility of adding the linkability property to their ring signature scheme by relying on a variant of LCE, they explicitly refrained from proposing a concrete construction due to concerns about this variant. The work of [8] confirms that these concerns were well-founded, showing that such a construction is not secure when instantiated with codes of rate $1/2$. This leaves open the question of whether codes with a different code rate could still be suitable for constructing such primitives.

This work addresses this question by generalizing the polynomial-time algorithm from [8] to any code rate and any number of available LCE public keys. Then, we compute the complexity of such an algorithm and compare it against the state-of-art attack, i.e., the meet-in-the-middle attack introduced in [10] adapted to this case scenario. We show that, regardless of the code rate, whenever one has access to more than one LCE instance with the same secret, the complexity of retrieving it drops significantly compared to having only one instance.

1.1 Overview of the Contribution

Background

In what follows, a *monomial matrix* is a matrix formed by the multiplication of a permutation matrix and a full-rank diagonal matrix.

Definition 1 *Let $\mathcal{C}$ and $\mathcal{C}'$ two (n,k) linear codes over a finite field $\mathbb{F}_q$. We say that $\mathcal{C}$ and $\mathcal{C}'$ are* equivalent *if there exists a monomial matrix $\boldsymbol{Q} \in \mathbb{F}_q^{n\times n}$ such that*

$$\mathcal{C}' = \mathcal{C}\boldsymbol{Q}.$$

The Linear Code Equivalence (LCE) Problem consists of finding $\boldsymbol{Q}$ given two generators of $\mathcal{C}$ and $\mathcal{C}'$.

Consider the scenario where an adversary has access to (the generators of) t pairs of equivalent codes $(\mathcal{C}_i, \mathcal{C}'_i)_{i=1}^t$ via the same secret monomial matrix $\boldsymbol{Q}$, i.e.,

$$\mathcal{C}'_i = \mathcal{C}_i \boldsymbol{Q}, \qquad \text{for } i = 1, \dots, t.$$

We call the *sample complexity* of LCE the minimum t such that $\boldsymbol{Q}$ can be computed in polynomial time. Note that this number strongly depends on the code rate $r := k/n$. Indeed, it was shown in [8] that there exists an efficient algorithm able to recover the monomial matrix $\boldsymbol{Q}$ if

$$t \geq \left\lfloor \frac{n^2}{k(n-k)} \right\rfloor + 1 = \left\lfloor \frac{1}{r(1-r)} \right\rfloor + 1.$$

Notice that this result provides an upper bound for the sample complexity of LCE, as it identifies a potential minimum value of t that enables an efficient recovery of $\boldsymbol{Q}$. However, such an upper bound is not necessarily optimal: there is no guarantee that other polynomial-time algorithms capable of recovering $\boldsymbol{Q}$ for smaller values of t do not exist. In fact, another finding from [8] shows that, when $r = 1/2$, only $t = 2$ LCE samples are enough to efficiently recover $\boldsymbol{Q}$. This result was achieved by introducing a polynomial-time algorithm that exploits the permutation structure in $\boldsymbol{Q}$ and sequentially guesses the positions of its non-zero entries. However, for $r \neq 1/2$, this algorithm is no longer effective, leaving the question of whether the provided upper bound is optimal or not for other code rates.

Contribution

Our contribution to the topic is twofold:

i) we provide new tighter upper bounds for the sample complexity of LCE,

ii) we demonstrate that the complexity of solving LCE in the multiple samples setting is significantly reduced compared to the single-sample case, for any code-rate and number of samples available.

We achieve the above by generalizing the algorithm from [8] to make it effective also when $k \neq n/2$. Our generalization consists of relaxing both the condition of the number of simultaneous guesses ℓ allowed and the number t of LCE samples available. In particular, for a given rate $r = k/n$ and t samples, then one must make $\ell > n - tk$ simultaneous guesses at a time to make the algorithm successful. This increases the complexity with respect to the version presented in [8] where $\ell = 1$ always. However, we were able to isolate the scenarios for which such an algorithm still runs in polynomial time and make a comparison against the meet-in-the-middle attack introduced in [10] adapted to the multi-sample case scenario. Specifically, we discover that our algorithm is preferable when $t = n/k$, while the meet-in-the-middle attack is asymptotically faster for the other cases.

In general, our result strengthens the thesis from [7, 8] that LCE in the multiple sample setting is not secure and should not be used for cryptographic applications.

Organization

In Section 2, we give the notation and the necessary preliminaries. In Section 3, we present our new algorithm, which generalizes the one introduced in [8] for any code rate and give its analysis. The cryptographic implication of our work and the comparisons against the state-of-the-art attacks are discussed in Section 4. Finally, we give our conclusions in Section 5.

2 Preliminaries

2.1 Notation and Linear Algebra

We denote by $\mathbb{N}$ the set of natural numbers. Given $n \in \mathbb{N}$, we use $[n]$ to denote the set $1, 2, \dots, n$. Matrices are denoted by uppercase bold letters (e.g., $\boldsymbol{A}$), and column vectors by lowercase bold letters (e.g., $\boldsymbol{a}$). We denote by $\mathbb{F}_q$ a finite field of order q. The tensor product $(\boldsymbol{A} \otimes \boldsymbol{B}) \in \mathbb{F}_q^{mr \times ns}$ of two matrices $\boldsymbol{A} \in \mathbb{F}_q^{m \times n}$ and $\boldsymbol{B} \in \mathbb{F}_q^{r \times s}$ is defined as the Kronecker product of $\boldsymbol{A}$ and $\boldsymbol{B}$. We work with the following sets of matrices:

- $\mathsf{GL}_n(\mathbb{F}_q)$, the set of all $n \times n$ invertible matrices with elements in $\mathbb{F}_q$,
- $\mathsf{Perm}_n(\mathbb{F}_q)$, the set of all permutation matrices of dimension n,
- $\mathsf{Mono}_n(\mathbb{F}_q)$, the set of all $n \times n$ *monomial* matrices, i.e., matrices of the form $\boldsymbol{M} = \boldsymbol{D}\boldsymbol{P}$, where $\boldsymbol{D} \in \mathbb{F}_q^{n \times n}$ is diagonal with only non-zero entries, and $\boldsymbol{P} \in \mathsf{Perm}_n(\mathbb{F}_q)$.

The $n \times n$ identity matrix over $\mathbb{F}_q$ is denoted by $\boldsymbol{I}_n$. Given an $m \times n$ matrix $\boldsymbol{M}$, we denote its left and right kernels by $\mathsf{ker_L}(\boldsymbol{M}) = \{\boldsymbol{w} \in \mathbb{F}_q^m : \boldsymbol{w}^\top \boldsymbol{M} = \boldsymbol{0}\}$ and $\ker(\boldsymbol{M}) = \{\boldsymbol{v} \in \mathbb{F}_q^n : \boldsymbol{M}\boldsymbol{v} = \boldsymbol{0}\}$, respectively, and we denote its image by $\mathsf{im}(\boldsymbol{M}) = \{\boldsymbol{M}\boldsymbol{v} : \boldsymbol{v} \in \mathbb{F}_q^n\}$.

We assume that matrix multiplication and inversion can be performed using $O(n^\omega)$ field operations for some $\omega \in [2, 3]$ (for instance, using Strassen's algorithm, which gives $\omega = \log_2(7)$ for large n). Therefore, performing Gaussian elimination to solve a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, with $\boldsymbol{A} \in \mathbb{F}_q^{n \times n}$ and $\boldsymbol{b} \in \mathbb{F}_q^n$, or to compute its rank, costs $O(n^\omega)$ field operations (if the matrix is not square, we consider $n = \max\{n_{\text{rows}}, n_{\text{cols}}\}$).

Proposition 1 ([8, Prop. 1]) *Let $\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{B}_1, \boldsymbol{B}_2 \in \mathbb{F}_q^{k \times (2k-1)}$ be matrices, for $k \geq 2$. Then the rank of the matrix*

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \\ \boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \end{bmatrix}$$

is strictly smaller than $2k^2$.

2.2 Linear Code Equivalence

An (n, k)-linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional vector subspace of $\mathbb{F}_q^n$. We say that $\mathcal{C}$ has length n and dimension k. The *rate* of the code is the ratio $r := \frac{k}{n}$. A matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ is called a *generator matrix* of $\mathcal{C}$ if its rows form a basis of $\mathcal{C}$, that is, $\mathcal{C} = \{\boldsymbol{u}^T\boldsymbol{G} : \boldsymbol{u} \in \mathbb{F}_q^k\}$. We say that $\boldsymbol{G}$ is in *systematic form* if $\boldsymbol{G} = (\, \boldsymbol{I}_k \,|\, \boldsymbol{M} \,)$ for some $\boldsymbol{M} \in \mathbb{F}_q^{k \times (n-k)}$. The systematic form of a generator, when it exists, can be obtained in polynomial time by computing its row-echelon form, which provides a standard basis for the vector space it spans. We denote this operation by $\mathsf{SF}(\cdot)$. A *parity-check matrix* for a code $\mathcal{C}$ is a full-rank matrix $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$ such that for all $\boldsymbol{c} \in \mathcal{C}$, it holds that $\boldsymbol{Hc} = \mathbf{0}$.

Let $\boldsymbol{G}, \boldsymbol{G'}$ be the generator matrices of two (n, k)-linear codes $\mathcal{C}, \mathcal{C}'$. We say that $\mathcal{C}$ and $\mathcal{C}'$ are *equivalent* if there exist $\boldsymbol{S} \in \mathsf{GL}_k(\mathbb{F}_q)$ and $\boldsymbol{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ such that $\boldsymbol{G}' = \boldsymbol{SGQ}$.

Definition 2 (Linear Code Equivalence (LCE) Problem) *Let $\boldsymbol{G}, \boldsymbol{G'} \in \mathbb{F}_q^{k\times n}$ be the generator matrices of two (n, k)-linear codes $\mathcal{C}, \mathcal{C}'$, respectively. The Code Equivalence Problem is to find matrices (if they exist) $\boldsymbol{S} \in \mathsf{GL}_k(\mathbb{F}_q)$ and $\boldsymbol{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ such that $\boldsymbol{G}' = \boldsymbol{SGQ}$.*

If the secret matrix $\boldsymbol{Q}$ is specifically a permutation matrix, then the problem is known as **Permutation Code Equivalence** (PCE) **Problem**.

Remark 1 For efficiency reasons, LCE instances are given in practice with the generators in systematic form [4]

$$(\boldsymbol{G}, \boldsymbol{G}' = \mathsf{SF}(\boldsymbol{GQ})).$$

In this case, the secret invertible matrix $\boldsymbol{S}$ is redundant, and the problem reduces to find only $\boldsymbol{Q}$.

2.3 On the Sample Complexity of LCE

To study the stronger cryptographic properties of linear code equivalence, a more general definition of LCE where an adversary has access to multiple LCE samples for the same secret monomial $\boldsymbol{Q}$ was introduced in [8].

Definition 3 ([8, Def. 5]) *Let n, k, q be integers such that $k < n$ and q is prime. Let $\boldsymbol{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ be a* secret *monomial matrix. We denote by $\mathcal{L}_{n,k,q,\boldsymbol{Q}}$ the probability distribution on $\mathbb{F}_q^{k\times n} \times \mathbb{F}_q^{k\times n}$ obtained by sampling $\boldsymbol{M} \in \mathbb{F}_q^{k\times(n-k)}$ uniformly at random, setting $\boldsymbol{G} = (\, \boldsymbol{I}_k \,|\, \boldsymbol{M} \,) \in \mathbb{F}_q^{k\times n}$, and returning*

$$(\boldsymbol{G}, \boldsymbol{G}' = \mathsf{SF}(\boldsymbol{GQ})).$$

Given t independent samples from $\mathcal{L}_{n,k,q,\boldsymbol{Q}}$, the t-samples LCE problem*, denoted as t-LCE, is to find $\boldsymbol{Q}$.*

In [8], the authors make a study on the necessary number of samples t that makes t-LCE solvable in polynomial time, which we refer to as *sample complexity* of LCE. Given an LCE instance $(\boldsymbol{G}, \boldsymbol{G}')$, let $\boldsymbol{H}'$ be a parity-check matrix for the code generated by $\boldsymbol{G}'$. Then we have that

$$\boldsymbol{GQH}'^{\top} = \mathbf{0}.$$

Consequently, the following linear system of dimension $k(n-k)$ in n^2 variables

$$[\boldsymbol{G} \otimes \boldsymbol{H}']\boldsymbol{x} = \mathbf{0}. \tag{1}$$

has, among its solutions, the vector composed by the entries of $\boldsymbol{Q}$ unrolled row-by-row [11, Definition 1.1.3 and Corollary 3.2.13]. It is shown in [8] that by stacking

$$t \geq \left\lfloor \frac{n^2}{k(n-k)} \right\rfloor + 1 \tag{2}$$

systems as in Equation (1) derived from independent LCE samples (but with the same secret unimodular $\boldsymbol{Q}$), one obtains a (over)determined linear system allowing an efficient retrieval of the unknown $\boldsymbol{x}$ (and so of $\boldsymbol{Q}$) in polynomial time via Gaussian elimination. Notice that Equation (2) provides an upper bound for the sample complexity of LCE that is not necessarily optimal.

2.4 A Polynomial-time Algorithm for Solving 2-LCE when $k = n/2$

For the parameter setting of $k = n/2$, a polynomial-time algorithm was given in [8] that allows an efficient recovery of the secret monomial $\boldsymbol{Q}$ when only $t = 2$ samples are provided (i.e., solving 2-LCE). The idea is to construct a linear system from two LCE samples $(\boldsymbol{G}_\iota, \boldsymbol{G}'_\iota)$, for $\iota = 1, 2$, exploiting again Equation (1),

$$\mathsf{S} : \overbrace{\begin{bmatrix} \boldsymbol{G}_1 \otimes \boldsymbol{H}'_1 \\ \boldsymbol{G}_2 \otimes \boldsymbol{H}'_2 \end{bmatrix}}^{\boldsymbol{A}} \boldsymbol{x} = \boldsymbol{0}, \tag{3}$$

and using the structure of the secret $\boldsymbol{Q}$ to reduce the number of variables. Indeed, the $2k(n-k) \times n^2$ linear system in Equation (3) is underdetermined since $2k(n-k) < n^2$, and $\boldsymbol{A}$ is full-rank with probability $1 - 1/q$ [8, Proposition 4]. However, the solution vector corresponding to $\boldsymbol{Q}$ is significantly structured. In particular, every row and column in $\boldsymbol{Q}$ contains only one non-zero entry. Suppose we know that the entry (i, j) is non-zero. In that case, we also know that all the other entries on the i^{th} row and j^{th} column are zero, as well as the corresponding variables. Hence, by guessing the position of a non-zero entry, one can guess a total of $2n-1$ unknowns. It was shown in [8] that it is possible to distinguish correct from wrong guesses thanks to the following observation.

Let $\mathsf{S}_{i,j}$ be the k dimensional linear system with $(n-1)^2$ variables obtained when guessing the entry (i, j) of $\boldsymbol{Q}$ to be non-zero. Then we have that

$$\mathsf{S}_{i,j} : \overbrace{\begin{bmatrix} (\boldsymbol{G}_1)_{-i} \otimes (\boldsymbol{H}'_1)_{-j} \\ (\boldsymbol{G}_2)_{-i} \otimes (\boldsymbol{H}'_2)_{-j} \end{bmatrix}}^{\boldsymbol{A}_{i,j}} \boldsymbol{x} = \boldsymbol{b}_{i,j}, \tag{4}$$

where $(\boldsymbol{G}_\iota)_{-i}, (\boldsymbol{H}'_\iota)_{-j} \in \mathbb{F}_q^{k \times (n-1)}$ are the matrices $\boldsymbol{G}_\iota, \boldsymbol{H}'_\iota$ punctured at positions i and j respectively, for $\iota = 1, 2$, and $\boldsymbol{b}_{i,j} \in \mathbb{F}_q^{k(n-k)}$ is the column of the original system corresponding to the non-zero guessed position. Proposition 1 is used to show that, surprisingly, $\mathsf{rank}(\boldsymbol{A}_{i,j}) < 2k(n-k)$. So, one can distinguish correct from wrong guesses as follows:

- a correct guess is such that $\mathsf{rank}(\boldsymbol{A}_{i,j}) = \mathsf{rank}(\boldsymbol{A}_{i,j}|\boldsymbol{b}_{i,j})$ because, for the Rouché-Capelli Theorem, a solution corresponding to $\boldsymbol{Q}$ always exists,
- a wrong guess is such that $\mathsf{rank}(\boldsymbol{A}_{i,j}) \neq \mathsf{rank}(\boldsymbol{A}_{i,j}|\boldsymbol{b}_{i,j}) = 2k(n-k)$ with probability approximately $1 - 1/q$.

Following this observation, one defines Test 1 based on the Rouché-Capelli Theorem. This test allows to rule out bad guesses with probability $1 - 1/q$, and always accepts good guesses.

Test 1 ([8, Test 1]) *For the guess on the (i, j)-th entry of $\boldsymbol{Q}$ to be non-zero, construct a reduced system S_{ij} from S (as in Equation (4)) with $(n-1)^2$ variables by setting $\boldsymbol{Q}(i,j) = 1$ and $\boldsymbol{Q}(i,\mu), \boldsymbol{Q}(\eta,j) = 0$, for $\mu \in \{1 \dots n\} \setminus \{j\}$ and $\eta \in \{1 \dots n\} \setminus \{i\}$. Accept the guess if the system S_{ij} accepts at least one solution; reject otherwise.*

If a guess on the position (i, j) to be non-zero does not pass Test 1, then certainly that entry, and the corresponding variable in Equation (3), must be equal to zero. Performing Test 1 on all entries of $\boldsymbol{Q}$ allows to set to zero enough variables such that the remaining ones are $< 2k(n-k)$. Finally, one obtains a reduced and determined linear system $\mathsf{S}_{\mathsf{red}}$, which allows efficient recovery of the non-zero entries via Gaussian elimination. The algorithm summarizing such procedure is reported in Algorithm 1. The estimated complexity is $O(n^{2+2\omega})$.

Algorithm 1 Solving 2-LCE

Input: a 2-LCE instance

Output: a monomial matrix $\boldsymbol{Q}$

```
1: Construct $\mathsf{S}$ as in Equation (3)
2: Set $L = [x_{1,1}, x_{1,2}, ..., x_{n,n}]$ the list of variables in $\mathsf{S}$
3: for $i = 1$ to $n$ do
4:     for $j = 1$ to $n$ do
5:         if Test 1 on $(i, j)$ fails then
6:             Remove $x_{i,j}$ from $L$
7:         end if
8:     end for
9: end for
10: Construct the reduced system $\mathsf{S}_{\mathsf{red}}$ with only the variables in $L$
11: if $\mathsf{S}_{\mathsf{red}}$ is underdetermined then
12:     return $\perp$
13: end if
14: Compute a solution matrix $\boldsymbol{R}$ for $\mathsf{S}_{\mathsf{red}}$
15: return $\boldsymbol{R}$
```

3 A Generalized Algorithm for Solving t-LCE

The algorithm presented in [8, Section 4] allows the retrieval of a solution with a monomial structure from an underdetermined but structured linear system as detailed in Proposition 1. Specifically, when constructing a linear system as in Equation (3), the conditions in Proposition 1 are satisfied only when $k = n/2$. In this section, we generalize the approach from [8, Section 4] to any $k \in (0, n/2]$ and t not necessarily equal to 2. By making a relaxation on the number of simultaneous guesses ℓ allowed and on the number t of samples available, we derive a generalization of Proposition 1 that highlights the parameter settings for which the monomial solution can be retrieved in a similar fashion as in Algorithm 1.

We begin by giving the details on multiple guessing and the corresponding generalized test in Section 3.1 and Section 3.2. Then, we introduce our algorithm in Section 3.3.

3.1 Multiple Guessing

In this section, we explain how to extend Test 1 to the scenario of multiple guessing. Once again, we take advantage of the structure of $\boldsymbol{Q}$ as follows. Instead of guessing the position of only one non-zero entry, we consider guessing the position of ℓ non-zero entries of $\boldsymbol{Q}$ as follows.

Let us consider a generalized version of the system in Equation (3) for t samples of LCE

$$\mathsf{S}: \overbrace{\begin{bmatrix} \boldsymbol{G}_1 \otimes \boldsymbol{H}'_1 \\ \boldsymbol{G}_2 \otimes \boldsymbol{H}'_2 \\ \cdots \\ \boldsymbol{G}_t \otimes \boldsymbol{H}'_t \end{bmatrix}}^{\boldsymbol{A}} \mathbf{x} = \mathbf{0}, \tag{5}$$

where $t < \left\lfloor \frac{n^2}{k(n-k)} \right\rfloor + 1$, (i.e. below the bound given in Equation (2)), and $\mathbf{x} = (x_{1,1}, x_{1,2}, \ldots, x_{n,n})^\top$ is a vector of unknowns made by the unrolling of the entries of $\boldsymbol{Q}$.

Since every row and every column in $\boldsymbol{Q}$ has only one non-zero entry, if we know that a set of ℓ entries $\{(i_1, j_1), \ldots, (i_\ell, j_\ell)\}$ is non-zero, then all the other entries on the $i_1^{th}, \ldots, i_\ell^{th}$ rows and on the $j_1^{th}, \ldots, j_\ell^{th}$ columns are zero. Hence, we cannot have two guesses on the same row or column because it would yield a solution matrix with a row or a column made only of zeroes, which contradicts the structure of $\boldsymbol{Q}$. So, we need the indexes $i_1, \ldots, i_\ell$ to be pairwise different as well as the indexes $j_1, \ldots, j_\ell$.

Let us analyze how many variables get removed when making ℓ simultaneous guesses in a linear system as in Equation (5). Although with one guess we are able to remove $2n - 1$ variables, we do not remove $\ell \cdot (2n - 1)$ of them when we perform ℓ guesses. Indeed, the row-column sets of removed variables overlap. For instance, if we consider a guessing of two entries (i_1, j_1) and (i_2, j_2), they produce two sets of erased variables:

$$\{x_{i_1,\mu}, x_{\nu,j_1} : \mu, \nu \in [n]\} \qquad \text{and} \qquad \{x_{i_2,\mu}, x_{\nu,j_2} : \mu, \nu \in [n]\}.$$

However, the intersection of these two sets is $\{x_{i_1,j_2}, x_{i_2,j_1}\}$, so they overlap and we remove only $2(2n-1)-2$ variables. Every successive guessing decreases the number of variables we remove because we have more overlaps.

We must point out an observation before giving the explicit number of variables that we can remove. In the single guessing case, we are able to set the variable of the non-zero entry to be 1, only implying that the monomial matrix $\boldsymbol{Q}$ is a multiple of the solution of the system. This is not possible for the multiple guessing case. In fact, if we set all the non-zero entries to 1 we assume that all of the guessed non-zero entries have the same value in the monomial matrix, which is not necessarily true. Hence, it might yield reduced systems that do not admit the monomial matrix $\boldsymbol{Q}$ as a solution. However, we can still set one (and only one) of the non-zero entries to 1, say the one corresponding to x_{i_1,j_1}. In what follows, we call **extra columns** the other ones corresponding to the other guessed positions, in this case, $x_{i_2,j_2}, \ldots, x_{i_\ell,j_\ell}$.

In total, ℓ simultaneous guesses allow to remove $2(n-1)+2(n-2)+\ldots+2(n-\ell)+1 = 2\ell n - \ell(\ell+1)+1$ variables from the system S, obtaining a reduced linear system of the form

$$\mathsf{S}_L : \quad \boldsymbol{A}_L \boldsymbol{x} = \boldsymbol{b}_L,$$

where $\boldsymbol{b}_L$ is given by the column in $\boldsymbol{A}$ corresponding to the variable set to 1, and the total number of remaining unknowns is

$$n^2 - (2\ell n - \ell(\ell+1)+1) = (n-\ell)^2 + \ell - 1.$$

In Test 2, we formalize the procedure to make ℓ guesses simultaneously.

Test 2 *Let S be the system constructed from a t-LCE instance. For the multiple guess on the entries $L = \{(i_1,j_1),(i_2,j_2),\ldots,(i_\ell,j_\ell)\}$ we construct a reduced system S_L with $(n-\ell)^2+\ell-1$ variables by doing the following on the unknowns of S:*

- $x_{i_1,j_1} = 1$
- $x_{i_k,\mu} = x_{\nu,j_k} = 0$ *for every $\mu \in \{1,\ldots,n\} \setminus \{j_k\}$ and $\nu \in \{1,\ldots,n\} \setminus \{i_k\}$, for every $k = 1,\ldots,\ell$*
- $x_{i_2,j_2},\ldots,x_{i_\ell,j_\ell}$ *are left unchanged and the relative columns in the matrix are called "extra columns"*

Accept the guess on the entries in L if S_L admits at least one solution; reject otherwise.

In order to check whether the system admits a solution, we use the Rouché-Capelli Theorem, i.e., we accept the guess if $\mathsf{rank}(\boldsymbol{A}_L) = \mathsf{rank}(\boldsymbol{A}_L|\boldsymbol{b}_L)$, reject it otherwise.

Remark 2 In practice, we always consider $\ell < n-k$ because, as the authors of [8] already highlighted, guessing a non-zero entry corresponds to puncturing the two equivalent codes. In our framework, codes have length n and dimension k, meaning the redundancy is $n-k$. If we guess $n-k$ entries, then we are puncturing $n-k$ columns of the codes; in other words, we are removing its redundancy. Without redundancy, all linear codes collapse to the identity matrix and do not provide any useful information for distinguishing good from bad guesses. Therefore, guessing becomes equivalent to a brute-force attack on the permutation $\boldsymbol{P}$ such that $\boldsymbol{Q} = \boldsymbol{PD}$.

3.2 Analysis of Test 2

3.2.1 Robustness

Since we use the Rouché-Capelli Theorem to distinguish between good and bad guesses, we need to prove that the framework of Test 2 is robust. Recall that the test rejects a guess $L = \{(i_1, j_1), (i_2, j_2), \dots, (i_\ell, j_\ell)\}$ if and only if $\mathsf{rank}(\boldsymbol{A}_L) < \mathsf{rank}(\boldsymbol{A}_L|\boldsymbol{b}_L)$. This means that we need the rank of the matrix of coefficients to decrease after the guess, in particular

$$\mathsf{rank}(\boldsymbol{A}_L) < \mathsf{rank}(\boldsymbol{A}). \tag{6}$$

Indeed, if $\mathsf{rank}(\boldsymbol{A}_L) = \mathsf{rank}(\boldsymbol{A})$, then the span of the columns of $\boldsymbol{A}_L$ is the same one of $\boldsymbol{A}$. So, if we augment the matrix $\boldsymbol{A}_L$ of the vector $\boldsymbol{b}_L$, the rank cannot increase because $\boldsymbol{b}_L$ is a column of $\boldsymbol{A}$. Therefore, we always get $\mathsf{rank}(\boldsymbol{A}_L) = \mathsf{rank}(\boldsymbol{A}_L|\boldsymbol{b}_L)$ and the test is passed, giving us no meaningful information on the variable. This scenario is not functional since we want to erase as many variables as possible. As a matter of fact, Equation (6) is a necessary condition for the test to work, and we have to ensure that the rank of the matrix of coefficients **always** decreases after the guess. We prove that the condition is satisfied in two steps: (i) we prove that matrix $\boldsymbol{A}_L$ is never full-rank, and (ii) we prove that matrix $\boldsymbol{A}$ is full-rank with overwhelming probability.

(i) The first step is achieved by means of Proposition 2. If we have a t-LCE instance $\{(\boldsymbol{G}_i, \boldsymbol{G}'_i)\}_{i=1}^{t}$ with parameters n, k such that $t \geq 2, k \leq n/2$ and $n \geq tk$, then we can set $\ell = n - tk + 1$. If $\ell < n-k$, then the hypothesis $tk > n - \ell$ is satisfied, too. Indeed,
$$n - \ell = n - (n - tk + 1) = tk - 1 < tk.$$
Therefore, according to Proposition 2, the matrix $\boldsymbol{A}_L$ defined in Equation (5) is not full-rank.

Proposition 2 (Generalization of Proposition 1) *Let q be a prime and let n, k, t, ℓ be positive integers such that $t \geq 2$, $\ell < n-k$, $k \leq n/2$ and $n \geq tk$. Let us consider a set of t pairs of matrices $\boldsymbol{A}_i \in \mathbb{F}_q^{k\times(n-\ell)}, \boldsymbol{B}_i \in \mathbb{F}_q^{(n-k)\times(n-\ell)}$ for $i = 1, \dots, t$. If $tk > n - \ell$, then the rank of the matrix*
$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \\ \boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \\ \cdots \\ \boldsymbol{A}_t \otimes \boldsymbol{B}_t \end{bmatrix}$$
is strictly less than $tk(n-k)$ and
$$\dim\left(\mathsf{ker_L}(\boldsymbol{C})\right) \geq (tk - n + \ell)(\ell t + n - \ell - tk).$$

Proof First of all, let us define two matrices

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 \\ \boldsymbol{A}_2 \\ \cdots \\ \boldsymbol{A}_t \end{bmatrix} \in \mathbb{F}_q^{tk\times(n-\ell)} \qquad \text{and} \qquad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1 \\ \boldsymbol{B}_2 \\ \cdots \\ \boldsymbol{B}_t \end{bmatrix} \in \mathbb{F}_q^{t(n-k)\times(n-\ell)}.$$

We observe that $\boldsymbol{A}$ admits a vector in its left kernel because the number of its rows is larger than the number of its columns. This is ensured by the hypothesis $tk > n-\ell$. Let us call $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_t)$ a non-zero vector in $\mathsf{ker}_\mathsf{L}(\boldsymbol{A})$. Now, we want to prove that the intersection of the vector subspaces $\bigcap_{i=1}^t \mathsf{im}(\boldsymbol{B}_i)$ is positive dimensional. Towards this direction, we can assume that $\dim(\mathsf{im}(\boldsymbol{B}_i)) = n-k$, for any $i = 1, \dots, t$. Indeed, if it was less than $n-k$ for one of them, let us say $\boldsymbol{B}_j$, then $\boldsymbol{B}_j$ would not be full-rank, and neither would $\boldsymbol{C}$. We can bound from below the dimension of this intersection using Grassmann's formula[*]:

$$\begin{aligned} \dim\left(\bigcap_{i=1}^t \mathsf{im}(\boldsymbol{B}_i)\right) &\geq \sum_{i=1}^t \dim(\mathsf{im}(\boldsymbol{B}_i)) - (t-1)(n-\ell) \\ &= t(n-k) - (t-1)(n-\ell) = \ell t + n - \ell - tk. \end{aligned}$$

Thanks to the hypothesis $n \geq tk$, we can lower bound the last expression as

$$\ell t + n - \ell - tk \geq \ell t + tk - \ell - tk = \ell(t-1),$$

which is greater than zero because $t \geq 2$ and $\ell > 0$ by hypothesis. As a result, we have $\dim\left(\bigcap_{i=1}^t \mathsf{im}(\boldsymbol{B}_i)\right) > 0$ so there exists a non-zero vector $\mathbf{y} \in \mathbb{F}_q^{n-\ell}$ such that $\forall i = 1, \dots, t$ there is $\boldsymbol{\beta}_i \in \mathbb{F}_q^{n-k}$ satisfying $\boldsymbol{\beta}_i \boldsymbol{B}_i = \mathbf{y}$. Let us now define a new vector $\boldsymbol{v}$ as

$$\boldsymbol{v} = (\boldsymbol{\alpha}_1 \otimes \boldsymbol{\beta}_1, \boldsymbol{\alpha}_2 \otimes \boldsymbol{\beta}_2, \dots, \boldsymbol{\alpha}_t \otimes \boldsymbol{\beta}_t).$$

Then we have that

$$\begin{aligned} \boldsymbol{v}\boldsymbol{C} = (\boldsymbol{\alpha}_1 \otimes \boldsymbol{\beta}_1, \boldsymbol{\alpha}_2 \otimes \boldsymbol{\beta}_2, \dots, \boldsymbol{\alpha}_t \otimes \boldsymbol{\beta}_t) \begin{bmatrix} \boldsymbol{A}_1 \otimes \boldsymbol{B}_1 \\ \boldsymbol{A}_2 \otimes \boldsymbol{B}_2 \\ \cdots \\ \boldsymbol{A}_t \otimes \boldsymbol{B}_t \end{bmatrix} = \\ = \boldsymbol{\alpha}_1\boldsymbol{A}_1 \otimes \boldsymbol{\beta}_1\boldsymbol{B}_1 + \boldsymbol{\alpha}_2\boldsymbol{A}_2 \otimes \boldsymbol{\beta}_2\boldsymbol{B}_2 + \ldots + \boldsymbol{\alpha}_t\boldsymbol{A}_t \otimes \boldsymbol{\beta}_t\boldsymbol{B}_t = \\ = \boldsymbol{\alpha}_1\boldsymbol{A}_1 \otimes \mathbf{y} + \boldsymbol{\alpha}_2\boldsymbol{A}_2 \otimes \mathbf{y} + \ldots + \boldsymbol{\alpha}_t\boldsymbol{A}_t \otimes \mathbf{y} = \\ = \big(\boldsymbol{\alpha}_1\boldsymbol{A}_1 + \boldsymbol{\alpha}_2\boldsymbol{A}_2 + \ldots + \boldsymbol{\alpha}_t\boldsymbol{A}_t\big) \otimes \mathbf{y} = \underbrace{\boldsymbol{\alpha}\boldsymbol{A}}_{=0} \otimes \mathbf{y} = 0. \end{aligned}$$

As a matter of fact, $\boldsymbol{v} \in \mathsf{ker}_\mathsf{L}(\boldsymbol{C})$ which implies that the $\mathsf{rank}(\boldsymbol{C})$ must be strictly less than $tk(n-k)$. In particular, we have that

$$\mathsf{ker}_\mathsf{L}(\boldsymbol{A}) \otimes \bigcap_{i=1}^t \mathsf{im}(\boldsymbol{B}_i) \leq \mathsf{ker}_\mathsf{L}(\boldsymbol{C}).$$

By the property of the Kronecker product, the dimension of the tensor product of two vector spaces is the product of their dimensions, so

$$\dim\big(\mathsf{ker}_\mathsf{L}(\boldsymbol{A})\big) \cdot \dim\left(\bigcap_{i=1}^t \mathsf{im}(\boldsymbol{B}_i)\right) \leq \dim\big(\mathsf{ker}_\mathsf{L}(\boldsymbol{C})\big).$$

We conclude the proof by noting that $\dim\big(\mathsf{ker}_\mathsf{L}(\boldsymbol{A})\big) \geq tk - n + \ell$. $\square$

[*]Let V be a finite-dimensional vector space and $U, W \leq V$ two subspaces. Then, we have $\dim(U \cap W) = \dim(U) + \dim(W) - \dim(U+W) \geq \dim(U) + \dim(W) - \dim(V)$

Remark 3 An immediate consequence of choosing $\ell = n - tk + 1$ is that the multiple guessing approach does not affect the single sample case, i.e., LCE. Indeed, when $t = 1$, we should take $\ell = n-k+1$, which is strictly bigger than $n-k$. According to Remark 2, this is equivalent to a brute-force attack.

Remark 4 The hypothesis $k \leq n/2$ is used implicitly throughout the proof of Proposition 2. In fact, if we allow $k > n/2$, then t would necessarily be set to 1 so as not to clash with the hypothesis $n \geq tk$. When $t = 1$ is combined with the last hypothesis $tk > n - \ell$, it produces $\ell > n - k$ (which is pathological, as we discussed in Remark 2). However, the case $k > n/2$ can be covered as well. We just need to adapt the hypothesis: $n \geq tk$ becomes $n \geq t(n-k)$ and $tk > n - \ell$ becomes $t(n-k) > n - \ell$. The proof is almost identical to the one of Proposition 2, "swapping $\boldsymbol{A}$ with $\boldsymbol{B}$": we first show that there is a non-zero vector in $\mathsf{ker}_\mathsf{L}(\boldsymbol{B})$ and then we prove that $\bigcap_{i=1}^{t} \mathsf{im}(\boldsymbol{A}_i)$ is positive dimensional.

Now, we apply Proposition 2 to our scenario. We recall that performing a guess on an entry (i,j) is equivalent to reducing the system $(\boldsymbol{G} \otimes \boldsymbol{H}')\mathbf{x} = \mathbf{0}$ by puncturing the code generated by $\boldsymbol{G}$ in i and the code generated by $\boldsymbol{H}'$ in j. Similarly, performing a multiple guess is equivalent to puncturing the codes generated by $\boldsymbol{G}$ and $\boldsymbol{H}'$ in several entries $\forall i$. In particular, the first coordinate of the set of entries is the punctured columns of $\boldsymbol{G}$, while the second coordinate of the set of entries is the punctured columns of $\boldsymbol{H}'$.

So, let us define $I = \{i_1, \dots, i_\ell\}$ and $J = \{j_1, \dots, j_\ell\}$, and let us call $\dot{\boldsymbol{G}}_i$ the matrix $\boldsymbol{G}_i$ punctured in I and $\dot{\boldsymbol{H}}'_i$ the matrix $\boldsymbol{H}'_i$ punctured in J, $\forall i = 1, \dots, t$. If we call

$$\dot{\boldsymbol{A}} = \begin{bmatrix} \dot{\boldsymbol{G}}_1 \otimes \dot{\boldsymbol{H}}'_1 \\ \dot{\boldsymbol{G}}_2 \otimes \dot{\boldsymbol{H}}'_2 \\ \dots \\ \dot{\boldsymbol{G}}_t \otimes \dot{\boldsymbol{H}}'_t \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{M} = \begin{bmatrix} \dot{\boldsymbol{G}}_1 \\ \dot{\boldsymbol{G}}_2 \\ \dots \\ \dot{\boldsymbol{G}}_t \end{bmatrix},$$

then w.l.o.g. $\boldsymbol{A}_L$ can be considered as the matrix $\dot{\boldsymbol{A}}$ augmented of the *extra columns* i.e. $\boldsymbol{A}_L = (\dot{\boldsymbol{A}} | \boldsymbol{K})$, with $\boldsymbol{K}$ being a $tk(n-k) \times (\ell - 1)$ matrix obtained by sticking together the *extra columns.*

By applying Proposition 2 to $\dot{\boldsymbol{A}}$ we get that $\dim\left(\mathsf{ker}_\mathsf{L}(\dot{\boldsymbol{A}})\right) \geq (tk - n + \ell)(\ell t + n - \ell - tk)$. Since $\boldsymbol{A}_L = (\dot{\boldsymbol{A}} | \boldsymbol{K})$, we have that $\dim\left(\mathsf{ker}_\mathsf{L}(\boldsymbol{A}_L)\right) \geq \dim\left(\mathsf{ker}_\mathsf{L}(\dot{\boldsymbol{A}})\right) - (\ell - 1)$. Hence, the rank of matrix $\boldsymbol{A}_L$ is

$$\mathsf{rank}(\boldsymbol{A}_L) = \mathsf{rank}(\dot{\boldsymbol{A}}) + \ell - 1 \leq tk(n-k) - (tk - n + \ell)(\ell t + n - \ell - tk) + \ell - 1. \quad (7)$$

For our purposes, we need $\dim\left(\mathsf{ker}_\mathsf{L}(\boldsymbol{A}_L)\right) \geq 1$ which is ensured if $\dim\left(\mathsf{ker}_\mathsf{L}(\dot{\boldsymbol{A}})\right) \geq \ell$. Therefore, we conclude that the first step holds true if

$$\dim\left(\mathsf{ker}_\mathsf{L}(\dot{\boldsymbol{A}})\right) \geq \ell. \quad (8)$$

Under the hypothesis $\ell = n - tk + 1$, Equation (8) reduces to $\ell(t-1) \geq 1$, which always holds true since $\ell \geq 1$ and $t \geq 2$.

(ii) For the second step, we need to introduce one more result tailored for t-LCE instances.

Proposition 3 *Let $q \geq 2$ be a prime, let n, k, t be integers such that $n \geq 2$, $t \geq 2$ and $tk \leq n$ and let*

$$\{(\boldsymbol{G}_i, \boldsymbol{G}'_i)\}_{i=1}^t$$

be a t-LCE instance with secret monomial matrix $\boldsymbol{Q}$. Let us construct the matrices $\boldsymbol{A}$ and $\boldsymbol{M}$ as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{G}_1 \otimes \boldsymbol{H}'_1 \\ \boldsymbol{G}_2 \otimes \boldsymbol{H}'_2 \\ \cdots \\ \boldsymbol{G}_t \otimes \boldsymbol{H}'_t \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{M} = \begin{bmatrix} \boldsymbol{G}_1 \\ \boldsymbol{G}_2 \\ \cdots \\ \boldsymbol{G}_t \end{bmatrix},$$

$\boldsymbol{H}'_i$ being the parity check matrices of $\boldsymbol{G}'_i$.
Then, $\boldsymbol{A}$ is full-rank if and only if $\boldsymbol{M}$ is full-rank.

Proof See Appendix A. □

An immediate consequence of Proposition 3 is that we can reduce the study of $\mathsf{rank}(\boldsymbol{A})$ to $\mathsf{rank}(\boldsymbol{M})$, with $\boldsymbol{M}$ being the same matrix defined in the statement of Proposition 3. Since the matrices $\boldsymbol{G}_i$ are random $\forall i$, then we can consider $\boldsymbol{M}$ as a matrix sampled randomly in $\mathbb{F}_q^{tk \times n}$. Recalling that the probability of a random matrix in $\mathbb{F}_q^{r \times s}$ being full-rank is $1 - \frac{1}{q^{1+s-r}}$, we can conclude that

$$\mathcal{P}\big(\boldsymbol{A} \text{ is full-rank}\big) = \mathcal{P}\big(\mathsf{rank}(\boldsymbol{M}) = tk\big) = 1 - \frac{1}{q^{1+n-tk}}. \tag{9}$$

3.2.2 False Positives

Now that we have proved the robustness of Test 2, we can move to discuss its probability of giving false positives. From now on, if not specified, we assume the value of ℓ to be

$$\ell = n - tk + 1. \tag{10}$$

Indeed, according to Proposition 2, this is the **minimum** value of ℓ that ensures us that Test 2 works.

Similarly to Test 1, a correct multiple guess always passes Test 2 because the reduced system admits a solution by construction. On the other hand, a wrong multiple guess may or may not be accepted. Using an argument similar to the one used in [8, Section 4.1], we can estimate the probability of Test 2 giving a false positive. Let us consider a t-LCE instance, the relative linear system $\mathsf{S} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ and a set of ℓ entries $L = \{(i_1, j_1), \ldots, (i_\ell, j_\ell)\}$ supposedly **not** corresponding to a set of non-zero entries in the monomial matrix. If we perform the multiple guesses on the variables in L, then we obtain a reduced system

$$\mathsf{S}_L : \boldsymbol{A}_L \bar{\boldsymbol{x}} = \boldsymbol{b}_L,$$

where $\bar{\boldsymbol{x}}$ is the new vector of variables according to Test 2.

Since we use Rouché-Capelli Theorem as a distinguisher between good and bad guesses, we need to compute the probability that $\mathsf{rank}(\boldsymbol{A}_L) = \mathsf{rank}(\boldsymbol{A}_L|\boldsymbol{b}_L)$. Thanks to Proposition 3 and Equation (9), we know that $\mathsf{rank}(\boldsymbol{A}) = tk(n-k)$ with high probability. On the other hand, performing ℓ simultaneous guesses (with $\ell = n - tk + 1$) reduces the rank of the system by a certain value d as already explained above, i.e., $\mathsf{rank}(\boldsymbol{A}_L) = tk(n-k) - d$.

Let us now consider $X = \mathsf{ker}_\mathsf{L}(\boldsymbol{A}_L)$ and $Y = \mathsf{ker}_\mathsf{L}(\boldsymbol{b}_L)$. Their dimensions are

$$\dim(X) = tk(n-k) - \mathsf{rank}(\boldsymbol{A}_L) = d \qquad \text{and} \qquad \dim(Y) = tk(n-k) - 1.$$

Modelling X and Y as random vector sub-spaces, we can estimate the probability of Test 2 to pass as follows. According to Rouché-Capelli Theorem, the test passes if $\mathsf{rank}(\boldsymbol{A}_L|\boldsymbol{b}_L) = \mathsf{rank}(\boldsymbol{A}_L)$ which is equivalent to $X \subset Y$. In order to compute $\mathcal{P}(X \subset Y)$ we recall that given a (finite-dimensional) $\mathbb{F}_q$-vector space V and a subspace $W \leq V$, then a random vector $\boldsymbol{v}$ lies in W with probability $q^{\dim(W) - \dim(V)}$. If we consider a basis $\mathcal{B}_X = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_d\}$ for X then

$$\mathcal{P}(\boldsymbol{v}_i \in Y) = \frac{q^{tk(n-k)-1}}{q^{tk(n-k)}} = \frac{1}{q}, \qquad \forall i = 1, \dots, d.$$

Assuming that the events "$\boldsymbol{v}_i \in Y$" are independent $\forall i$, we conclude that the probability of a false positive for Test 2 is approximately the product of the probabilities above, which is $\frac{1}{q^d}$.

From Proposition 2, we know that $d = (tk - n + \ell)(\ell t + n - \ell - tk) - (\ell - 1)$, therefore we can state that for Test 2 we have the following probability

$$\mathcal{P}(\text{Test 2 gives a false positive}) = \frac{1}{q^{(tk-n+\ell)(\ell t+n-\ell-tk)-(\ell-1)}} = \frac{1}{q^d}. \tag{11}$$

Experimental Validation. We conducted two experiments. The first experiment validates our analysis of $\mathsf{rank}(\boldsymbol{A}_L)$. In each of the 1000 test trials, we generated random t-LCE instances and performed a random multiple guess of ℓ entries. We computed $\mathsf{rank}(\boldsymbol{A}_L)$ for each instance and averaged the results over all trials, as reported in Table 1. Sometimes, the average rank is slightly lower than the upper bound estimated in our analysis using Equation (7) as, for small q, some additional random linear dependencies appear between the rows.

The second experiment tests the probability of false positives given by Equation (11). Our experiment generates a t-LCE instance and then performs Test 2 on 10000 random guesses for some values of n, k, q, t, and ℓ. In addition, we discarded degenerate cases where the matrix $\boldsymbol{A}$ was not full-rank. Each of the random guesses was performed on the first ℓ rows of the monomial matrix. We do not perform the guesses on random rows because of the design of Algorithm 2 (defined below). In fact,

n	k	t	ℓ	q	**Average Rank**	**Predicted Rank (Eq.7)**
40	19	2	3	29	795.000	795
	12	2	3		997.999	998
60	19	3	4	29	2329.000	2329
	13	4	9		2417.000	2417
80	30	2	21	11	2978.935	2979
	21	3	18		3680.944	3681

Table 1: In each experiment, we compute 1000 t-LCE instances and a random multiple guess of ℓ entries and then we compute the rank of the matrix $\boldsymbol{A}_L$. The second-last column reports the arithmetic mean of the 1000 ranks. The last column reports the predicted rank according to the analysis following Proposition 2.

at line 2, a set I of ℓ rows is fixed, and then the guesses are performed only on entries lying on those rows. In Table 2, we compare the average number of false positives measured over 10 repetitions of the experiment against the expected number of false positives according to Equation (11). One can observe that the data in the last column follows the trend of the experimental data, meaning that our estimation matches the experiments. The code to reproduce the experiments is available at [12].

n	k	t	ℓ	q	**False Positives Found**	**False Positives Expected (Eq. 11)**
33	16	2	2	5	481	400
				7	197	204
				11	76.7	82.6
33	15	2	4	5	22.8	16.0
				7	3.80	4.16
				11	1.50	0.68
30	14	2	3	5	112	80.0
				7	37.5	29.1
				11	8.60	7.51
30	13	2	5	5	4.00	3.20
				7	0.50	0.59
				11	0.10	0.06

Table 2: In each experiment, we run 10000 times Test 2 and record how many false-positives are accepted. The second-last column reports the arithmetic mean of 10 experiments. The last column reports the expected number of false-positives according to Equation (11). ℓ was computed as $n - tk + 1$.

3.3 A Generalization of Algorithm 1

To make Test 2 viable, we must perform it on all the combinations produced by a set of ℓ rows (or, equivalently, ℓ columns). In other words, we have to choose a set of ℓ rows (or columns) of the monomial matrix and then try all the possible combinations of ℓ guesses that can be done on those rows (or columns). This means that for an ℓ-multiple guess we have to check $\frac{n!}{(n-\ell)!} \approx n^\ell$ combinations.

Algorithm 2 Procedure for ℓ multiple guess

Input: a t-LCE instance, an integer ℓ and a list V containing the variables of the system

Output: a list of surviving variables

```
1: Let $\mathcal{L} = \{(\iota_1, \iota_2, \dots, \iota_\ell) \in [n]^\ell : \iota_\mu \neq \iota_\nu,\ \forall \mu \neq \nu\}$ and choose $I = (i_1, \dots, i_\ell) \in \mathcal{L}$
2: Let W $= [x_{i_1,1}, \dots, x_{i_\ell,n}]$ be a list containing all the variables related to the $\ell$ rows $i_1, \dots, i_\ell$
3: for $\omega \in \mathcal{L}$ do
4:     if Test 2 with $L = \{(i_1, \omega[1]), \dots, (i_\ell, \omega[\ell])\}$ is accepted then
5:         Remove the variables $x_{i_1,\omega[1]}, \dots, x_{i_\ell,\omega[\ell]}$ from W
6:     end if
7: end for
8: Remove from V all the variables that are stored in W
9: return V
```

Moreover, we recall that the number of surviving variables depends on the probability of false positives. In particular,

$$\#\{\text{surviving variables}\} = \frac{n!}{(n-\ell)!}\mathcal{P}(\text{Test 2 gives a false positive}) =$$

$$= \frac{n!}{(n-l)!}\frac{1}{q^d} \approx \frac{n^\ell}{q^d}.$$

One should notice the following observation regarding this quantity: if it is too high, it might be that for every variable $x_{i_\alpha,\beta} \in$ W there exists at least one combination $\omega \in \mathcal{L}$ such that $x_{i_\alpha,\beta} \in L = \{(i_1, \omega[1]), \dots, (i_\ell, \omega[\ell])\}$ and L is accepted by Test 2. According to the design of Algorithm 2, this scenario does not allow the removal of any variable from V, resulting in a failure. On the other hand, the best scenario is when only one combination is accepted by Test 2: the correct one, i.e., the one corresponding only to the non-zero entries of the selected ℓ rows. This happens when the number of combinations is less than the inverse of the probability of a false positive, i.e., when

$$n^\ell < q^d. \tag{12}$$

As long as this last condition holds, the procedure leads to a very small number of surviving variables. Afterward, one can repeat the procedure on another set of rows (or columns) until the total number of surviving variables is less than the number of equations in the system and then solve it with Gaussian elimination. The whole strategy is outlined in Algorithm 3.

Algorithm 3 Solving t-LCE with multiple guess

```
Input: a t-LCE instance and an integer $\ell$
Output: a monomial matrix $\boldsymbol{M}$ that solve Equation (5)
1: Let $\mathsf{V} = \left[x_{i,j} : i, j \in [n]\right]$ be a list containing all the variables of the system
2: while $\#\mathsf{V} > tk(n-k)$ do
3:     Let $\mathsf{W}$ be the output of Algorithm 2 with inputs $\ell$ and $\mathsf{V}$.
4:     $\mathsf{V} \leftarrow \mathsf{W}$
5: end while
6: Let $\mathbf{x}$ be a column vector with the entries of $\mathsf{V}$
7: Solve $\boldsymbol{A}\mathbf{x} = \mathbf{0}$ and let $\boldsymbol{M}$ be a solution
8: return $\boldsymbol{M}$
```

Analysis of Algorithm 3. First, we prove that the algorithm terminates. The core of the algorithm is represented by the while loop at lines 2-5. The condition $n^\ell < q^d$ ensures us that the number of surviving variables at each execution of Algorithm 2 is approximately ℓ. Thus, every time one iteration of the while loop is executed, the number of variables decreases. So, the algorithm enters the loop only a finite number of times, ensuring it terminates within a finite amount of time.

We now give the complexity of Algorithm 3. If we assume that at every iteration of the while loop, the set I chosen in line 2 of Algorithm 2 contains only rows that have not been considered by previous iterations, then every iteration removes $\ell(n-1)$ variables. So, the number r of iterations of the while loop must be such that

$$tk(n-k) > n(n-r\ell) + r\ell.$$

We rearrange the equation as

$$tk(n-k) - n^2 > r\ell(1-n) \qquad \iff \qquad r < \frac{tk(n-k)-n^2}{\ell(1-n)},$$

which means that $r \in O(n)$. The complexity of Algorithm 2 is $O(n^{\ell+2\omega})$ because it consists of $O(n^\ell)$ rank computation, whose cost is $O(n^{2\omega})$ each. We conclude that the time complexity of Algorithm 3 is

$$O(n^{\ell+2\omega+1}).$$

This formula justifies Equation (10): choosing the minimum value for ℓ improves drastically the complexity of Algorithm 3.

Finally, given that it is possible to not store the whole list $\mathcal{L}$ in memory by computing it on the fly, then the algorithm is polynomial in memory.

In order to test the correctness of Algorithm 3, we implemented it in `SageMath` [13], and the code is available at [12]. As an example, we solved an instance with $n = 26,\ k = 12,\ q = 61,\ t = 2,\ \ell = 3$ in 162 seconds on a MacBook Pro with processor 2.4 GHz 8-Core Intel Core i9, 64 GB of RAM, and running it on a single core.

4 Cryptographic Implications

4.1 Implications on the Sample Complexity of LCE

We now discuss how Algorithm 3 improves the sample complexity of LCE given in [8, Section 3], i.e., the smallest t such that t-LCE is solvable in polynomial time. We recall that, for any integers n, k with $k \leq n$, it is always possible to recover efficiently the secret monomial matrix $\boldsymbol{Q}$ if the number of samples is

$$t \geq \left\lfloor \frac{n^2}{k(n-k)} \right\rfloor + 1.$$

Indeed, it was shown in [8, Section 3] that in this case, the system $\mathsf{S} : \boldsymbol{Ax} = \boldsymbol{0}$ in Equation (5) becomes determined, allowing us to retrieve the vector of entries of the monomial matrix $\boldsymbol{Q}$ via Gaussian elimination. However, for smaller values of t, the system is underdetermined, and this approach does not work anymore. In fact, in this case, the space of solutions has dimension larger than one, and retrieving the monomial solution out of a basis becomes hard: formally, the complexity is exponential in the dimension of $\ker(\boldsymbol{A})$. Therefore, solving a t-LCE instance with this approach would not be efficient.

Equation (2) gives us an upper bound on the number of variables sufficient to retrieve the monomial matrix $\boldsymbol{Q}$ in polynomial time. However, it was shown in [8, Section 4] that for $k = n/2$, there exists an algorithm that recovers the monomial matrix for $t = 2$ only and with polynomial-time complexity $O(n^{2+2\omega})$. Now, thanks to Algorithm 3, we improve the bound for a whole new class of parameters. Let $k = \frac{n-c}{t}$, with c being a constant integer. Given t samples, the complexity of Algorithm 3 is

$$O\big(n^{\ell+2\omega+1}\big) = O\big(n^{n-tk+2\omega+2}\big) = O\big(n^{n-t\frac{n-c}{t}+2\omega+2}\big) = O\big(n^{c+2\omega+2}\big).$$

Since c is constant, then the algorithm runs in polynomial time. In this case, the value of t is

$$t = \frac{n-c}{k} \leq \frac{n}{k} < \frac{n}{k} \cdot \underbrace{\frac{n}{n-k}}_{>1} = \frac{n^2}{k(n-k)}.$$

Thus, Algorithm 3 improves the sample complexity of LCE with respect to the bound in Equation (2). We can also notice that the case $k = n/2$ addressed in [8, Section 4] is no more than the specific case $t = 2$ and $c = 0$ of our class of parameters. To highlight such an improvement, we compare in Table 3 the sample complexity from [8] against the one provided by Algorithm 3 for a range of code rates.

k	**Sufficient t by [8]**	**Sufficient t by Algorithm 3**
128	2	2
127	5	2
126	5	2
85	5	3
84	5	3
83	5	3
64	6	4
63	6	4
62	6	4
51	7	5
50	7	5
49	7	5

Table 3: Comparison between sample complexity of Algorithm 3 and sample complexity resulting from [8] for $n = 256$.

4.2 Comparisons with CF-MitM

We now compare Algorithm 3 with state-of-the-art solvers for t-LCE. At the moment, the fastest solver for LCE is represented by the Canonical Form Meet-in-the-Middle (CF-MitM) attack from [10] whose complexity is

$$O\left(\sqrt{\binom{n}{n-k}}\right) = O\left(\sqrt{\binom{n}{k}}\right). \tag{13}$$

This complexity is referred to the scenario with only one sample. Yet, if we assume $k < n/2$, we can show that more samples can improve the complexity of the algorithm. Indeed, every sample is a pair of codes $\mathcal{C}, \mathcal{C}'$ with generating matrices $\boldsymbol{G}_i, \boldsymbol{G}_i' \in \mathbb{F}_q^{k\times n}$ such that $\boldsymbol{G}_i' = \boldsymbol{G}_i\boldsymbol{Q}$ for $i = 1, \dots, t$. Notice that if we join together two linear codes with parameters (n, k), their union is again a code but with parameters (n, k'), with k' being an integer in the interval $[k, 2k]$:

$$\begin{bmatrix}\boldsymbol{G}_1\\ \boldsymbol{G}_2\end{bmatrix}\boldsymbol{Q} = \begin{bmatrix}\boldsymbol{G}_1\boldsymbol{Q}\\ \boldsymbol{G}_2\boldsymbol{Q}\end{bmatrix} = \begin{bmatrix}\boldsymbol{G}_1'\\ \boldsymbol{G}_2'\end{bmatrix} \in \mathbb{F}_q^{2k\times n}.$$

Remark 5 The value of k' cannot be known *a priori* because it depends on the two original codes, which might have some linear dependency. However, for random codes, the dimension of the combined code is highly likely to equal the sum of the dimensions of the individual codes for large values of q, as overlaps in random spaces are rare. In fact, the probability of a linear dependence between the two linear codes is roughly

$$\mathcal{P}(k' < 2k) = \sum_{d=1}^{k} \mathcal{P}(k' = 2k - d) \approx \frac{1}{q^{n-2k+1}} + o\left(\frac{1}{q^{n-2k+2}}\right). \tag{14}$$

For the union of t codes, Equation (14) can be generalized by replacing 2 with t, and $k' < tk$. Since in applications (e.g., [4]) we deal with random codes and relatively big primes (for small values of q the LCE problem is vulnerable to other types of attack [14]), the probability of linear dependence is small – however, not always negligible. In the following, we assume that the dimension of the union of two random codes is the sum of their dimensions. However, we will also show how the complexity changes when this event occurs.

By induction, we can join together an arbitrary amount s of codes with $s \leq t$, thus producing a (n, sk) code. In this way, we have just proved that we can reduce t-LCE with parameters (n, k) to LCE with parameters (n, sk) for any $s \leq t$. Therefore, the complexity of CF-MitM becomes

$$\min\left\{O\left(\sqrt{\binom{n}{sk}}\right) : s = 1, \ldots, t \text{ and } sk < n\right\}. \tag{15}$$

Note that if we join together too many codes, then the reduction does not work. In fact, if $s \geq n/k$ then the union of s codes is (with high probability according to Remark 5) the whole vector space $\mathbb{F}_q^n$ since

$$s \cdot k \geq \frac{n}{k} \cdot k \geq n.$$

Therefore, we would reduce to a trivial instance of LCE whose solution does not give any information on the monomial matrix $\boldsymbol{Q}$. For this reason, Equation (15) also checks that sk is strictly less than n. Moreover, this also means that from the point of view of the CF-MitM attack, **although** the number of samples is greater than $\lfloor n/k \rfloor$, the hardness of the problem is the same as that of $\lfloor n/k \rfloor$-LCE. Hence, we restrict our analysis to the regime $t = \lfloor n/k \rfloor$.

The complexity of Algorithm 3 is $O\big(n^{n-tk+2\omega+2}\big)$ and we observe that it is significantly bigger than the complexity of CF-MitM in most of cases (see Table 4). Even in the case $k = \frac{n-c}{t}$ explained in Section 4.1, CF-MitM outperforms the polynomial-time complexity of Algorithm 3. Indeed, if $k = \frac{n-c}{t}$ with $c > 0$ being a constant, then the complexity of CF-MitM is

$$\min\left\{O\left(\sqrt{\binom{n}{sk}}\right) : s = 1, \ldots, t\right\} = O\left(\sqrt{\binom{n}{tk}}\right) =$$
$$O\left(\sqrt{\binom{n}{n-c}}\right) = O\left(\sqrt{\binom{n}{c}}\right) \approx O(n^{\frac{c}{2}}),$$

according to Equation (15). In contrast, the time complexity of Algorithm 3 when $k = \frac{n-c}{t}$ is $O(n^{c+2\omega+2})$.

4.2.1 Special case of $n = tk$

There is a case when Algorithm 3 is a better solver than CF-MitM. Indeed, if $n = tk$, the reduction from t-LCE to LCE with parameters (n, tk) does not work. In this case, if we join together all the generators $\boldsymbol{G}_i$ (resp. $\boldsymbol{G}'_i$), we end up with a (n, n) linear code, i.e., the whole vector space $\mathbb{F}_q^n$ (at least with high probability, as pointed out in Remark 5). Therefore, we would reduce to a trivial instance of LCE whose solution does not give any information on the monomial matrix $\boldsymbol{Q}$. Hence, since we cannot join together all the generators, we can select a number $s < t$ of generators. The complexity in this case is

$$\min\left\{O\left(\sqrt{\binom{n}{sk}}\right) : s = 1, \dots, t-1\right\} = O\left(\sqrt{\binom{n}{k}}\right).$$

The last equality holds because binomial coefficient $\binom{n}{x}$ is smaller when x is closer to either n or 0. So, the smallest values occur when $s = 1$ or $s = t - 1$. Since k divides n and, by the properties of the binomial coefficient, we have that

$$\binom{n}{(t-1)k} = \binom{n}{n-k} = \binom{n}{k}.$$

Thus, in the case k divides n, having more samples from the same monomial matrix does not decrease the cost of the CF-MitM attack. On the other hand, the case $n = tk$ is of the type $k = \frac{n-c}{t}$ in particular with $c = 0$, then Algorithm 3 runs in polynomial time and outperforms asymptotically CF-MitM.

Table 4 shows the comparison between the complexity in bits of Algorithm 3 and CF-MitM with $n = 256$ and several values of t and k. As we can see, the complexity of Algorithm 3 rapidly increases with ℓ while the complexity of CF-MitM spikes sharply when k divides n.

As highlighted in Remark 5, there is a non-negligible probability that the union of the codes is not full-rank. When this happens, the complexity of the CF-MitM attack changes. In the regime $k < n/t$ with t samples the cost **increases**, indeed

$$\sqrt{\binom{n}{kt-d}} > \sqrt{\binom{n}{tk}}, \qquad d \in \left\{0, 1, 2, \dots, \frac{n}{2}\right\}.$$

On the other hand, in the regime $k = n/t$ the cost **decreases** sharply because in this case the reduction leads to a non-trivial LCE instance. Table 4 reports only the most likely complexity, i.e. the full-rank case.

t	k	$\ell = n - tk + 1$	**Complexity of Algorithm 3**	**Complexity of CF-MitM**
2	128	1	**60.9**	**125.8**
	127	3	76.9	7.5
	126	5	92.9	13.7
	125	7	108.9	19.2
	124	9	124.9	24.3
	110	37	348.9	73.1
3	85	2	68.9	4
	84	5	92.9	13.7
	83	8	116.9	21.8
4	64	1	**60.9**	**101.8**
	63	5	76.9	13.7
	62	9	124.9	24.3
5	51	2	68.9	4
	50	7	108.9	19.2
	49	12	148.9	31.2

Table 4: Comparison between Algorithm 3 and CF-MitM [10]. The complexities of the two algorithms are given in $\log_2$ scale and $n = 256$.

5 Conclusions

In this work, we extended the Linear Code Equivalence problem analysis in the multiple sample setting, generalizing previous results to any code rate and number of available LCE public keys. In addition to providing new tighter upper bounds on the sample complexity of LCE, our results demonstrate that irrespective of the code rate, the complexity of solving LCE significantly decreases when multiple LCE instances with the same secret are available, compared to when only one instance is accessible. We introduced a generalized algorithm that adapts to different code rates and compared it against the CF-MitM algorithm. Despite our algorithm is more efficient in memory, CF-MitM has a better time complexity in most scenarios than our algorithm. Nevertheless, we discovered some cases in which our algorithm outperforms CF-MitM.

References

[1] National Institute of Standards and Technology: Post-Quantum Cryptography Standardization. https://csrc.nist.gov/projects/post-quantum-cryptography (2017)

[2] National Institute of Standards and Technology: Post-Quantum Cryptography: Digital Signature Schemes. Round 1 Additional Signatures (2023). https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

[3] Biasse, J.-F., Micheli, G., Persichetti, E., Santini, P.: Less is more: Code-based signatures without syndromes. In: Progress in Cryptology - AFRICACRYPT 2020: 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20 – 22, 2020, Proceedings, pp. 45–65. Springer, Berlin, Heidelberg (2020). https://doi.org/10.1007/978-3-030-51938-4_3 . https://doi.org/10.1007/978-3-030-51938-4_3

[4] Baldi, M., Beckwith, A.B.L., Biasse, J.-F., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.-J.O., Santini, P., Wallace, R.: LESS (version 1.1). Tech. rep., National Institute of Standards and Technology (2023). https://www.less-project.com/

[5] Battagliola, M., Borin, G., Meneghetti, A., Persichetti, E.: Cutting the grass: Threshold group action signature schemes. In: Oswald, E. (ed.) Topics in Cryptology – CT-RSA 2024, pp. 460–489. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-58868-6_18

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

[7] D'Alconzo, G., Di Scala, A.J.: Representations of group actions and their applications in cryptography. Finite Fields and Their Applications **99**, 102476 (2024) https://doi.org/10.1016/j.ffa.2024.102476

[8] Budroni, A., Chi-Domínguez, J.-J., D'Alconzo, G., Di Scala, A.J., Kulkarni, M.: Don't use it twice! solving relaxed linear equivalence problems. In: Advances in Cryptology – ASIACRYPT 2024: 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9–13, 2024, Proceedings. Part VIII, pp. 35–65. Springer, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-981-96-0944-4_2

[9] Leroux, A., Roméas, M.: Updatable encryption from group actions. In: International Conference on Post-Quantum Cryptography, pp. 20–53 (2024). Springer. https://doi.org/10.1007/978-3-031-62746-0_2

[10] Chou, T., Persichetti, E., Santini, P.: On Linear Equivalence, Canonical Forms, and Digital Signatures. Cryptology ePrint Archive, Paper 2023/1533. https://eprint.iacr.org/2023/1533 (2023). https://eprint.iacr.org/2023/1533

[11] Saeed, M.A.: Algebraic approach for code equivalence. PhD thesis, Normandie Université, University of Khartoum, (2017). Available at https://theses.hal.science/tel-01678829v2

[12] Our code is avaliable at https://github.com/andreanatalee/tLCE_solver

[13] The Sage Developers: SageMath, the Sage Mathematics Software System (Version

9.8). (2023). https://www.sagemath.org

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

A Proof of Proposition 3

Proposition 3 *Let $q \geq 2$ be a prime, let n, k, t be integers such that $n \geq 2$, $t \geq 2$ and $tk \leq n$ and let*

$$\left\{\left(\boldsymbol{G}_i, \boldsymbol{G}'_i\right)\right\}_{i=1}^{t}$$

be a t-LCE instance with secret monomial matrix $\boldsymbol{Q}$. Let us construct the matrices $\boldsymbol{A}$ and $\boldsymbol{M}$ as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{G}_1 \otimes \boldsymbol{H}'_1 \\ \boldsymbol{G}_2 \otimes \boldsymbol{H}'_2 \\ \cdots \\ \boldsymbol{G}_t \otimes \boldsymbol{H}'_t \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{M} = \begin{bmatrix} \boldsymbol{G}_1 \\ \boldsymbol{G}_2 \\ \cdots \\ \boldsymbol{G}_t \end{bmatrix},$$

$\boldsymbol{H}'_i$ being the parity check matrices of $\boldsymbol{G}'_i$.
Then, $\boldsymbol{A}$ is full-rank if and only if $\boldsymbol{M}$ is full-rank.

Proof For the sake of simplicity, we prove the result in the case that every matrix $\boldsymbol{G}_i$ and $\boldsymbol{H}'_i$ are in systematic form. This does not come with any loss of generality. Indeed, let us call $\boldsymbol{U}_i \in \mathsf{GL}_k(\mathbb{F}_q)$ and $\boldsymbol{V}_i \in \mathsf{GL}_{n-k}(\mathbb{F}_q)$ the matrices that turn $\boldsymbol{G}_i$ and $\boldsymbol{H}'_i$ in systematic form, respectively for $i = 1, \dots, t$. Then, the block diagonal matrix

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{U}_1 \otimes \boldsymbol{V}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{U}_2 \otimes \boldsymbol{V}_2 & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{U}_t \otimes \boldsymbol{V}_t \end{bmatrix}$$

turns every matrix in $\boldsymbol{A}$ in its systematic form, viz.

$$\tilde{\boldsymbol{A}} = \boldsymbol{S}\boldsymbol{A} = \begin{bmatrix} (\boldsymbol{U}_1 \otimes \boldsymbol{V}_1)(\boldsymbol{G}_1 \otimes \boldsymbol{H}'_1) \\ (\boldsymbol{U}_2 \otimes \boldsymbol{V}_2)(\boldsymbol{G}_2 \otimes \boldsymbol{H}'_2) \\ \vdots \\ (\boldsymbol{U}_t \otimes \boldsymbol{V}_t)(\boldsymbol{G}_t \otimes \boldsymbol{H}'_t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{U}_1\boldsymbol{G}_1 \otimes \boldsymbol{V}_1\boldsymbol{H}'_1 \\ \boldsymbol{U}_2\boldsymbol{G}_2 \otimes \boldsymbol{V}_2\boldsymbol{H}'_2 \\ \vdots \\ \boldsymbol{U}_t\boldsymbol{G}_t \otimes \boldsymbol{V}_t\boldsymbol{H}'_t \end{bmatrix} = \begin{bmatrix} (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_1}\,) \otimes (\,-\boldsymbol{M_1'}^\top \,|\, \boldsymbol{I}_{n-k}\,) \\ (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_2}\,) \otimes (\,-\boldsymbol{M_2'}^\top \,|\, \boldsymbol{I}_{n-k}\,) \\ \vdots \\ (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_t}\,) \otimes (\,-\boldsymbol{M_t'}^\top \,|\, \boldsymbol{I}_{n-k}\,) \end{bmatrix}.$$

Moreover, $\boldsymbol{S}$ is full rank since it is made out of blocks in the linear group. Hence, $\mathsf{rank}(\tilde{\boldsymbol{A}}) = \mathsf{rank}(\boldsymbol{A})$. We can continue the proof focusing on $\tilde{\boldsymbol{A}}$ instead of $\boldsymbol{A}$ because we have proved they have the same rank. In the following, every occurrence of $\boldsymbol{A}$ must be thought of as $\tilde{\boldsymbol{A}}$. With a similar argument, we can consider the matrices in $\boldsymbol{M}$ as in systematic form. Thus,

$$\boldsymbol{M} = \begin{bmatrix} (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_1}\,) \\ (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_2}\,) \\ \vdots \\ (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_t}\,) \end{bmatrix}.$$

Let us assume that $\boldsymbol{A}$ is not full-rank. So, $\mathsf{rank}(\boldsymbol{A}) < tk(n-k)$ there exists a vector $\boldsymbol{v} \in \mathbb{F}_q^{tk(n-k)}$ such that $\boldsymbol{vA} = \boldsymbol{0}$. We can think of it as a concatenation of vectors in $\mathbb{F}_q^{n-k}$

$$\boldsymbol{v} = (\boldsymbol{v}_1^1, \dots, \boldsymbol{v}_k^1, \boldsymbol{v}_1^2, \dots, \boldsymbol{v}_k^2, \dots, \boldsymbol{v}_1^t, \dots, \boldsymbol{v}_k^t).$$

If we focus only on the columns related to $(\,\boldsymbol{I}_k \,|\, \boldsymbol{M_i}\,) \otimes \boldsymbol{I}_{n-k}$ we get

$$(\boldsymbol{v}_1^1, \dots, \boldsymbol{v}_k^1, \boldsymbol{v}_1^2, \dots, \boldsymbol{v}_k^2, \dots, \boldsymbol{v}_1^t, \dots, \boldsymbol{v}_k^t) \begin{bmatrix} \boldsymbol{I}_{n-k} & \boldsymbol{0} & \dots & \boldsymbol{0} & | & m_{1,1}^1 \boldsymbol{I}_{n-k} & \dots & m_{1,n-k}^1 \boldsymbol{I}_{n-k} \\ \boldsymbol{0} & \boldsymbol{I}_{n-k} & \dots & \boldsymbol{0} & | & m_{2,1}^1 \boldsymbol{I}_{n-k} & \dots & m_{2,n-k}^1 \boldsymbol{I}_{n-k} \\ \dots & & & & & & & \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{I}_{n-k} & | & m_{k,1}^1 \boldsymbol{I}_{n-k} & \dots & m_{k,n-k}^1 \boldsymbol{I}_{n-k} \\ \boldsymbol{I}_{n-k} & \boldsymbol{0} & \dots & \boldsymbol{0} & | & m_{1,1}^2 \boldsymbol{I}_{n-k} & \dots & m_{1,n-k}^2 \boldsymbol{I}_{n-k} \\ \dots & & & & & & & \\ \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{I}_{n-k} & | & m_{k,1}^t \boldsymbol{I}_{n-k} & \dots & m_{k,n-k}^t \boldsymbol{I}_{n-k} \end{bmatrix} =$$

$$\left(\sum_i \boldsymbol{v}_1^i, \sum_i \boldsymbol{v}_2^i, \dots, \sum_i \boldsymbol{v}_t^i, \sum_{i,j} \boldsymbol{v}_i^j m_{i,1}^j, \sum_{i,j} \boldsymbol{v}_i^j m_{i,2}^j, \dots, \sum_{i,j} \boldsymbol{v}_i^j m_{i,n-k}^j \right) = 0,$$

which yields a vector in $\mathbb{F}_q^{tk}$ in the left kernel of $\boldsymbol{M}$ (for instance, one could take the first entry of each $\boldsymbol{v}_i^j$). So, $\mathsf{rank}(\boldsymbol{M}) < tk$.

For the second implication, let us assume that $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_t) \in \mathsf{ker_L}(\boldsymbol{M})$. First, we prove that there exists a vector $\boldsymbol{y} \in \mathbb{F}_q^{n-k}$ such that $\boldsymbol{y}(\,-\boldsymbol{M_1'}^\top \,|\, \boldsymbol{I}_{n-k}\,) = \boldsymbol{y}(\,-\boldsymbol{M_2'}^\top \,|\, \boldsymbol{I}_{n-k}\,) = \dots = \boldsymbol{y}(\,-\boldsymbol{M_t'}^\top \,|\, \boldsymbol{I}_{n-k}\,)$. We consider two cases:

Case $\frac{n}{t+1} < k < \frac{n}{t}$: let us consider the following kernels

$$\begin{aligned} \ker\big((\,-\boldsymbol{M_1'}^\top \,|\, \boldsymbol{I}_{n-k}\,) - (\,-\boldsymbol{M_2'}^\top \,|\, \boldsymbol{I}_{n-k}\,)\big) &= \ker\big(\boldsymbol{M_2'}^\top - \boldsymbol{M_1'}^\top\big), \\ \ker\big((\,-\boldsymbol{M_2'}^\top \,|\, \boldsymbol{I}_{n-k}\,) - (\,-\boldsymbol{M_3'}^\top \,|\, \boldsymbol{I}_{n-k}\,)\big) &= \ker\big(\boldsymbol{M_3'}^\top - \boldsymbol{M_2'}^\top\big), \\ &\vdots \\ \ker\big((\,-\boldsymbol{M_{t-1}'}^\top \,|\, \boldsymbol{I}_{n-k}\,) - (\,-\boldsymbol{M_t'}^\top \,|\, \boldsymbol{I}_{n-k}\,)\big) &= \ker\big(\boldsymbol{M_t'}^\top - \boldsymbol{M_{t-1}'}^\top\big). \end{aligned}$$

It is trivial to notice that any vector in the intersection of all kernels above is a valid $\boldsymbol{y}$. We can show that they have non-trivial intersection by looking at their dimensions. Each kernel is a vector subspace in $\mathbb{F}_q^{n-k}$ and has dimension at least $(n-k)-k = n-2k$. So,

$$\dim\left(\bigcap_{i=1}^{t-1} \ker(\boldsymbol{M_{i+1}}^\top - \boldsymbol{M_i}^\top)\right) \geq (n-2k)(t-1) - (n-k)(t-2) = n - kt.$$

Since we assumed $k < n/t$, we can conclude that the dimension is strictly positive. So, it contains at least one non-zero vector that can be the vector $\boldsymbol{y}$ we require.

Case $k = \frac{n}{t}$: let us consider the matrix

$$\boldsymbol{N} = \begin{bmatrix} (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_1'}\,) \\ (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_2'}\,) \\ \vdots \\ (\,\boldsymbol{I}_k \,|\, \boldsymbol{M_t'}\,) \end{bmatrix}.$$

We can observe that $\boldsymbol{N}$ can be obtained from $\boldsymbol{M}$ by means of multiplying by the monomial matrix $\boldsymbol{Q}$ and computing the systematic forms of the codes (which is multiplying each chunk of rows by an invertible matrix $\boldsymbol{X_i} \in \mathbb{F}_q^{n-k\times n-k}$):

$$\boldsymbol{N} = \begin{bmatrix} \boldsymbol{X_1}\left(\, \boldsymbol{I}_k \,\middle|\, \boldsymbol{M_1} \,\right) \\ \boldsymbol{X_2}\left(\, \boldsymbol{I}_k \,\middle|\, \boldsymbol{M_2} \,\right) \\ \vdots \\ \boldsymbol{X_t}\left(\, \boldsymbol{I}_k \,\middle|\, \boldsymbol{M_t} \,\right) \end{bmatrix} \boldsymbol{Q}.$$

Let $\boldsymbol{w} = (\boldsymbol{w}_1, \dots, \boldsymbol{w}_t) \in \mathbb{F}_q^{tk}$ be the vector such that $\boldsymbol{wM} = \boldsymbol{0}$, then the vector $\tilde{\boldsymbol{w}} = (\boldsymbol{w}_1\boldsymbol{X_1}^{-1}, \dots, \boldsymbol{w}_t\boldsymbol{X_t}^{-1}) \in \mathsf{ker}_{\mathsf{L}}(\boldsymbol{N})$. The details of the computation are left to the reader. Since we assumed $k = n/t$, then $\boldsymbol{N}$ is a square matrix, and it is not full-rank. Therefore, its transpose is not full-rank either, i.e., there is a vector $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_q^n$ such that

$$(\boldsymbol{x}, \boldsymbol{y}) \cdot \begin{bmatrix} \boldsymbol{I}_k & \boldsymbol{I}_k & \dots & \boldsymbol{I}_k \\ \boldsymbol{M'}_{\boldsymbol{1}}^{\top} & \boldsymbol{M'}_{\boldsymbol{2}}^{\top} & \dots & \boldsymbol{M'}_{\boldsymbol{t}}^{\top} \end{bmatrix} = \left(\boldsymbol{x} + \boldsymbol{yM'}_{\boldsymbol{1}}^{\top}, \boldsymbol{x} + \boldsymbol{yM'}_{\boldsymbol{2}}^{\top}, \dots, \boldsymbol{x} + \boldsymbol{yM'}_{\boldsymbol{t}}^{\top} \right) = \boldsymbol{0}.$$

Thus, $(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{y}\left(\, -\boldsymbol{M}_{\boldsymbol{1}}'^{\top} \,\middle|\, \boldsymbol{I}_{n-k} \,\right) = \boldsymbol{y}\left(\, -\boldsymbol{M}_{\boldsymbol{2}}'^{\top} \,\middle|\, \boldsymbol{I}_{n-k} \,\right) = \dots = \boldsymbol{y}\left(\, -\boldsymbol{M}_{\boldsymbol{t}}'^{\top} \,\middle|\, \boldsymbol{I}_{n-k} \,\right)$.
Now, by direct computations and exploiting the properties of tensor product and of the vectors, we get

$$\begin{aligned}(\boldsymbol{v} \otimes \boldsymbol{y})\boldsymbol{A} = \sum_{i=1}^{t} \boldsymbol{v}_i\left(\, \boldsymbol{I}_k \,\middle|\, \boldsymbol{M_i} \,\right) \otimes \boldsymbol{y}\left(\, -\boldsymbol{M}_{\boldsymbol{i}}'^{\top} \,\middle|\, \boldsymbol{I}_{n-k} \,\right) = \\ \left(\sum_{i=1}^{t} \boldsymbol{v}_i\left(\, \boldsymbol{I}_k \,\middle|\, \boldsymbol{M_i} \,\right) \right) \otimes \boldsymbol{x} = \underbrace{\boldsymbol{vM}}_{=0} \otimes \boldsymbol{x} = 0.\end{aligned}$$

Therefore, $\boldsymbol{v} \otimes \boldsymbol{y}$ lies in the left kernel of $\boldsymbol{A}$ whose rank is strictly less than $tk(n-k)$. $\square$