Two Is All It Takes: Asymptotic and Concrete Improvements for Solving Code Equivalence

Alessandro Budroni[1], Andre Esser[1*], Ermes Franch[2†], and Andrea Natale[3‡]

[1] Technology Innovation Institute, UAE
{alessandro.budroni,andre.esser}@tii.ae
[2] No affiliation
ermes.franch@gmail.com
[3] University of Trento, Italy
andrea.natale@unitn.it

Abstract. The Linear Code Equivalence (LCE) problem asks, for two given linear codes $\mathcal{C}, \mathcal{C}'$, to find a monomial $\mathbf{Q}$ mapping $\mathcal{C}$ into $\mathcal{C}'$. Algorithms solving LCE crucially rely on a (heuristic) subroutine, which recovers the secret monomial from $\Omega(\log n)$ pairs of codewords $(\mathbf{v}_i, \mathbf{w}_i) \in \mathcal{C} \times \mathcal{C}'$ satisfying $\mathbf{w}_i = \mathbf{v}_i\mathbf{Q}$. We greatly improve on this known bound by giving a constructive (heuristic) algorithm that recovers the secret monomial from any *two* pairs of such codewords for any $q \geq 23$. We then show that this reduction in the number of required pairs enables the design of a more efficient algorithm for solving the LCE problem. Our asymptotic analysis shows that this algorithm outperforms previous approaches for a wide range of parameters, including all parameters proposed across the literature. Furthermore, our concrete analysis reveals significant bit security reductions for suggested parameters. Most notably, in the context of the LESS signature scheme, a second-round contender in the ongoing NIST standardization effort for post-quantum secure digital signatures, we obtain bit security reductions of up to 24 bits.

Keywords: Code Equivalence, Cryptanalysis, Code-based Cryptography, Post-quantum Cryptography

1 Introduction

Code-based cryptography remains a strong contender for the next post-quantum digital signature standard. Historically, most code-based signature

*Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID MA 2536/12

†Part of this work was conducted during Ermes Franch staying at the Department of Informatics, University of Bergen, Norway, which was supported by the Research Council of Norway (No. 311646/O70)

‡Part of this work was conducted during Andrea Natale's internship at the Technology Innovation Institute, UAE.

schemes have based their security on the Syndrome Decoding Problem (SDP) [Ste94,Vér97,CVE11,DST19,FJR22]. While the SDP is well studied and provides a solid security foundation, its conservative nature often comes at the price of larger signatures, larger public keys, or slower signing and verification. In recent years, a promising alternative emerged with the LESS signatures scheme [BMPS20], which is based on the *Linear Code Equivalence* (LCE) problem. The collective improvements made to the scheme over the past years [BBPS21,PS23,CPS23] have positioned it as a strong candidate [BBB+25a] in the ongoing second round of the NIST standardization process for post-quantum digital signatures [Nat23]. Furthermore, due to its compatibility with the cryptographic group action framework, the LCE problem has served as the foundation of a variety of digital signature schemes with advanced functionalities, including threshold [BBMP24,BBC+25], (linkable) ring [BBN+22], and identity-based [BBN+22] signatures.

The Linear Code Equivalence problem is defined as follows: Given two $\mathbb{F}_q$ linear $[n, k]$ codes $\mathcal{C}, \mathcal{C}'$, find a monomial $\mathbf{Q}$ that maps $\mathcal{C}$ into $\mathcal{C}'$, i.e., $\mathcal{C}' = \mathcal{C}\mathbf{Q}$. The best algorithms for solving LCE rely on the computation of low-weight codewords $\mathbf{v} \in \mathcal{C}$, $\mathbf{w} \in \mathcal{C}'$ in both codes [Leo82,Beu20,BBPS23]. Among the found codewords, the algorithms then identify pairs of codewords equivalent under the secret monomial, i.e., codewords satisfying $\mathbf{w}_i = \mathbf{v}_i\mathbf{Q}$. Once enough such pairs have been identified, the secret monomial can be recovered in polynomial time. A central question that evolves is

How many such pairs are required to recover $\mathbf{Q}$ *in polynomial time?*

The currently best known (heuristic) algorithm for recovering $\mathbf{Q}$ from such information in polynomial time requires $\Omega(\log n)$ pairs of equivalent codewords [Beu20,BBPS23].[1] Note that this algorithm retrieves the secret monomial solely from the information provided by those pairs of equivalent codewords (or 2-dimensional subcodes). In such a scenario, a single pair of codewords cannot be sufficient to recover $\mathbf{Q}$ in polynomial time, as for two codewords of same weight $w \ll n$, there usually exist exponentially many monomials $\mathbf{Q}'$ satisfying the relation $\mathbf{w} = \mathbf{v}\mathbf{Q}'$. Moreover, we find that for constant field size q, the information provided by any constant amount of equivalent pairs is insufficient to uniquely identify $\mathbf{Q}$.

However, generally, the searched monomial is uniquely determined by the equivalence relationship $\mathcal{C}' = \mathcal{C}\mathbf{Q}$. By combining the equivalence identity with the information provided by equivalent codeword pairs, we are able to construct a (heuristic) algorithm that recovers the secret monomial from only *two* pairs of such codewords for any field of size $q \geq 23$ in polynomial time. As a second main contribution, we then show that this result allows us to overcome a central bottleneck of previous algorithms relying on finding low-weight codewords, resulting in the fastest known algorithm for solving LCE for most parameters. We provide a full asymptotic analysis showing superiority over previous approaches in theory as well as a concrete treatment with respect to bit complexities, reducing security levels of the LESS signatures scheme by up to 24 bits.

[1] More precisely, this algorithm require pairs of equivalent 2-dimensional subcodes, i.e., pairs of pairs of equivalent codewords.

Related Work For solving the LCE problem, there exist two main approaches: algorithms following the above concept of computing low-weight codewords in the underlying codes [Leo82,Beu20,BBPS23] and a recently introduced meet-in-the-middle technique relying on *canonical forms* [CPS23]. The latter has attracted increased attention since its introduction, with recent results improving its success probability [Now24] as well as the range of applicable parameters [BBB+25b]. Notably, the algorithm's complexity is (almost) independent of the field size, obtaining a time complexity of roughly $2^{c_1(R)n}$, where c_1 is a constant depending on the code rate $R = k/n$. For large choices of q, this algorithm becomes the preferred choice to solve LCE. However, with respect to suggested parameters, algorithms relying on finding low-weight codewords remain yet superior.

An early algorithm by Leon [Leo82] computes all low-weight codewords in both codes and recovers the secret monomial from the resulting sets provided those sets are sufficiently large. Beullens [Beu20] improved upon Leon's approach by increasing the weight of the codewords and computing only a subset of them for each code. More precisely, Beullens concentrates on finding small support 2-dimensional subcodes, i.e., pairs of codewords with a small joint support. He then defines a criterion to identify equivalent subcodes and, eventually, relies on a heuristic algorithm recovering the secret monomial from $\Omega(\log n)$ equivalent pairs. Most recently, Barenghi, Biasse, Santini and Persichetti (BBPS) [BBPS23] improved on Beullens' algorithm by modifying the subroutine to find small support subcodes, while relying on the same post-processing to recover the secret monomial.

Our Contribution Our contribution is twofold. First we show a fundamental result on the required number of pairs of equivalent codewords to recover the secret monomial $\mathbf{Q}$ of an LCE instance in polynomial time. Precisely, we construct a (heuristic) polynomial-time algorithm that recovers the secret monomial from any two pairs of equivalent codewords. Our second contribution is the design of the fastest known algorithm for solving LCE for most relevant parameters. Below we give more technical details on these contributions.

Recovering the secret monomial from two pairs of equivalent codewords. Our starting point forms a modeling proposed in [Sae17], representing the equivalence relationship $\mathcal{C}' = \mathcal{C}\mathbf{Q}$ as a linear system. More precisely, for $\mathbf{H}'$ being the parity-check matrix of $\mathcal{C}'$ and $\mathbf{G}$ the generator matrix of $\mathcal{C}$, the secret monomial $\mathbf{Q}$ satisfies the equation $\mathbf{G}\mathbf{Q}\mathbf{H}'^{\top} = \mathbf{0}$. This can be reformulated as a linear system of the form $\mathbf{A}\mathbf{x} = \mathbf{0}$ with $\mathbf{A} = \mathbf{G} \otimes \mathbf{H}'$ and a solution $\mathbf{x}$ corresponding to the vectorized version of $\mathbf{Q}$.

This linear system generally admits many solutions since it does not enforce the non-linear constraint that $\mathbf{Q}$ is a monomial matrix. However, we then show how to refine the system by incorporating information from pairs of equivalent codewords, which simplifies the system by eliminating variables. In most cases, this still results in an underdetermined system. In order to recover the solution, we then make educated guesses about the remaining variables, by exploiting the monomial structure of $\mathbf{Q}$ — an approach inspired by a technique used in [BCDD+25] and

further refined in [BN24] for solving specific variants of LCE. Due to the structured nature of the constructed system, wrong guesses often lead to an immediate contradiction, allowing to remove additional variables, until the system finally becomes determined.

Note that the very same structure enabling this last step of the algorithm makes a rigorous analysis challenging. However, we provide intuitive reasoning and a heuristic analysis showing that the algorithm recovers the secret monomial from the given information for sufficiently large fields $\mathbb{F}_q$, with $q \geq 23$, in polynomial time. We provide a full implementation of the algorithm and present extensive experimental evidence that demonstrates the effectiveness of the approach across all parameters. Specifically, using our implementation, we successfully recover the secret monomial for any code rate and $q \geq 23$ with an empirical success probability close to one. Additionally, we validate the practicality on cryptographic-sized parameters by recovering the secret monomial from two pairs of equivalent fixed-weight codewords following parameters suggested in the LESS signature scheme.

A faster algorithm solving LCE. As a second contribution, we show how to leverage the above result to design a new algorithm for solving LCE outperforming previous approaches. Conceptually, the algorithm follows the common framework of algorithms relying on finding low-weight codewords: We construct two lists of fixed weight codewords, with list sizes chosen to guarantee the existence of at least two pairs of equivalent codewords between them. In a first simplified asymptotic version of the algorithm, we compute all tuples of codeword pairs between those lists and initiate the recovery of the secret monomial for each of them.

Note that the efficiency of the overall procedure greatly relies on the fact that two codeword pairs are sufficient for monomial reconstruction. The final computation of all tuples of pairs between the lists comes at a cost of N^4, where N is the list size. More specifically, if c pairs of equivalent codewords are required, the computation of all possibly equivalent c-tuples between the lists requires time N^{2c}. Due to the exponential size of N, any non-constant c results in an algorithm with super-exponential complexity. On the other hand, an improvement of the constant c from two down-to one would result in an immediate improvement in the complexity of the algorithm. However, even for $c = 2$, the proposed algorithm already outperforms all previous algorithms based on low-weight codeword finding and remains superior to the canonical forms meet-in-the-middle approach as long as the field size does not become too large. More precisely, our asymptotic analysis shows that the algorithm achieves a lower time complexity than the canonical forms meet-in-the-middle approach for any field size $q \leq 202$. Meanwhile, our concrete analysis suggests a significantly higher break-even point, in the order of $q > 2^{12}$.

We then provide an improved version of the algorithm that incorporates multiple concrete complexity improvements. Most notably, we avoid the computation of all possibly equivalent pairs by defining a criterion for monomial compatibility that can be checked efficiently. Based on that criterion, we obtain an algorithm where the final step is dominated by the list of pairs, i.e., it can be performed in time N^2 rather than N^4, for large enough fields. We then show how to exploit the existence of scalar multiples of the solution to obtain a further polynomial speedup. Overall,

when applied to the parameters suggested in the LESS signature NIST second-round submission, the improved algorithm leads to a bit security reduction of up to 24 bits (compared to Table 1). With respect to other parameters throughout the literature, we achieve a bit security reduction of up to 36 bits.

	Previous	This work
LESS-I	139	**127**
LESS-III	214	**196**
LESS-V	290	**266**

Table 1: Bit complexity estimates for suggested LESS parameters.

Artifacts. We provide all source code used to conduct the practical experiments and to compute the asymptotic and concrete complexity estimations in an accompanying git repository [BEFN].

Outline. In Section 2 we define necessary notation as well as fundamentals, and give a formal definition of the LCE problem. In Section 3, we give the details on the algorithm to reconstruct the secret monomial from any two pairs of equivalent codes. Subsequently, in Section 4, we present the new algorithm for solving LCE including both an asymptotic as well as a concrete analysis along with a comparison to the state of the art.

2 Preliminaries

For any $n \in \mathbb{N}$, we let $[n] := \{1, \ldots, n\}$. We denote matrices with capital bold letters and vectors with lower case bold letters. All vectors are row-vectors if not stated otherwise. For a vector $\mathbf{v}$ we denote by $\mathsf{sort}(\mathbf{v})$ the vector obtained by sorting the entries of $\mathbf{v}$ lexicographically. For a matrix $\mathbf{A} \in \mathbb{F}_q^{n \times n}$, we refer to its entry in the i-th row and j-th column by $\mathbf{A}(i, j)$. Given a set of indices $I \subseteq [n]$ we denote with $\mathbf{v}_I$ the projection of the length-n vector $\mathbf{v}$ onto the indices in I. We extend this to matrices by letting $\mathbf{M}_I$ be the submatrix formed by the columns of $\mathbf{M}$ indexed by I. We define $\mathsf{GL}_n(\mathbb{F}_q)$ as the linear group of invertible matrices, $\mathsf{Perm}_n(\mathbb{F}_q)$ as the set of permutation matrices, and $\mathsf{Mono}_n(\mathbb{F}_q)$ as the set of monomial matrices, i.e., matrices $\mathbf{Q} = \mathbf{DP}$ with $\mathbf{D}$ being a full-rank diagonal matrix and $\mathbf{P} \in \mathsf{Perm}_n(\mathbb{F}_q)$. Given a matrix $\mathbf{M} \in \mathbb{F}_q^{m \times n}$, we denote with $\mathsf{ker}(\mathbf{M})$ its right kernel. We denote with $\mathsf{vec}(\mathbf{M})$ the vector of length mn formed by concatenating the rows of $\mathbf{M}$. For two matrices $\mathbf{M} \in \mathbb{F}_q^{m \times n}$ and $\mathbf{N} \in \mathbb{F}_q^{r \times s}$, we denote by $\mathbf{M} \otimes \mathbf{N}$ their Kronecker product in $\mathbb{F}_q^{mr \times ns}$.

Linear Code Equivalence An $[n, k]$-linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a subspace of dimension k of $\mathbb{F}_q^n$. $\mathcal{C}$ can be represented via a basis $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ called *generator matrix* or a *parity-check matrix* $\mathbf{H} \in \mathbb{F}_q^{n-k \times n}$ satisfying $\mathbf{Hc} = \mathbf{0} \Leftrightarrow \mathbf{c} \in \mathcal{C}$. We refer to n as the

length of $\mathcal{C}$ and to the quantity $R := k/n$ as the *code rate*. Given a codeword $\mathbf{c} \in \mathcal{C}$, we refer to the non-zero entries $\mathrm{Supp}(\mathbf{c}) := \{i \in [n] : c_i \neq 0\}$ of $\mathbf{c}$ as its support. The Hamming weight $\mathrm{wt}(\mathbf{c}) = |\mathrm{Supp}(\mathbf{c})|$ of $\mathbf{c}$ is the size of its support.

We denote by N_w the number of codewords of fixed weight w in $\mathcal{C}$, which for random codes is known to be of expected size

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

The minimum distance $\min_{\mathbf{c}\in\mathcal{C}}\{\mathrm{wt}(\mathbf{c})\}$ of a linear code is defined as the minimum weight over all its codewords. The minimum distance of a random $[n,k]$-linear code over $\mathbb{F}_q$ is known to (asymptotically) meet the Gilbert-Varshamov bound, which is the smallest w satisfying $N_w \geq 1$. We denote this w by w_{GV}. Furthermore, we define the relative weight coefficient $\omega_{\mathrm{GV}} := w_{\mathrm{GV}}/n$.

In the following we give a formal definition of the linear equivalence problem, which we refer to as LCE problem in the remainder of this work.

Definition 1 (Linear Code Equivalence (LCE)). *Given two $[n,k]$ linear codes $\mathcal{C},\mathcal{C}'$ over $\mathbb{F}_q$ with generator matrices $\mathbf{G}$ and $\mathbf{G}'$, respectively, the* Linear Code Equivalence *problem asks to find a monomial $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ such that there exists a matrix $\mathbf{S} \in \mathsf{GL}_n(\mathbb{F}_q)$ satisfying $\mathbf{G}' = \mathbf{SGQ}$. We call $(\mathcal{C},\mathcal{C}')$ an* instance *of the LCE problem.*

The special case where $\mathbf{Q} \in \mathsf{Perm}_n(\mathbb{F}_q)$ is known as the *Permutation Code Equivalence* (PCE) problem, and all results reported in this work for LCE also apply to it. In case for two given codes $\mathcal{C},\mathcal{C}'$ there exists a solution to the above problem, we call those codes *equivalent*, and shorthand write $\mathcal{C}' = \mathcal{C}\mathbf{Q}$. Motivated by cryptographic constructions, we restrict to instances of the LCE problem with unique solution $\mathbf{Q}$ (up to scalar multiples). Furthermore, we focus on code rates $R \leq \frac{1}{2}$ as the LCE problem with rate R is equivalent to the LCE problem with rate $1-R$ by switching to the dual codes. In particular for $\mathbf{G},\mathbf{G}'$ being the generator matrices of $\mathcal{C},\mathcal{C}'$ and $\mathbf{H},\mathbf{H}'$ the respective parity-check matrices, we have

$$\mathbf{G}' = \mathbf{SGQ} \Rightarrow \mathbf{0} = \mathbf{G}'\mathbf{H}'^{\top} = \mathbf{SGQH}'^{\top}$$

From the last equality it follows that $\mathbf{QH}'^{\top}$ is the transpose of a parity-check matrix of $\mathcal{C}$. We can conclude that $\mathbf{H}' = \mathbf{RH}(\mathbf{Q}^{\top})^{-1}$ for some invertible matrix $\mathbf{R} \in \mathbb{F}_q^{n-k\times n-k}$, implying that the dual codes are equivalent with the monomial matrix $(\mathbf{Q}^{\top})^{-1}$.

Our analysis makes use of the following result for the complexity of computing fixed-weight codewords in a given linear code.

Lemma 1 (Prange' ISD complexity [Pra62]). *The cost of computing m of the N_w codewords of weight w in a random $[n,k]$-linear code is*

$$\tilde{\mathcal{O}}\left(\frac{m}{N_w}\frac{\binom{n}{w}}{\binom{n-k}{w}}\right).$$

Furthermore we use the following standard approximation of binomial coefficients

$$\frac{2^{h(\frac{k}{n})n}}{n+1} \leq \binom{n}{k} \leq 2^{h(\frac{k}{n})n}, \tag{1}$$

where $h(x) := -x \log x - (1-x)\log(1-x)$ is the binary entropy function.

3 Solving Linear Code Equivalence with Hints

In this section, we provide a new heuristic algorithm for recovering the secret monomial of an LCE instance from only two pairs of codewords equivalent under the secret monomial. Let us start by defining the term of *equivalent codewords*.

Definition 2 (Equivalent Codewords). *Let $\mathcal{C}, \mathcal{C}'$ be two $[n,k]$ linear codes over $\mathbb{F}_q$, equivalent under some monomial transformation $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ and let $\mathbf{v} \in \mathcal{C}$ and $\mathbf{w} \in \mathcal{C}'$ be two nonzero codewords. We say that $(\mathbf{v}, \mathbf{w})$ is a pair of* equivalent *codewords with respect to $\mathcal{C}, \mathcal{C}'$ if $\mathbf{w} = \mathbf{vQ}$.*

In the following, the codes $\mathcal{C}$ and $\mathcal{C}'$ are usually clear from the context, hence, we drop the term *with respect to $\mathcal{C}, \mathcal{C}'$*. In terms of this definition, we propose an algorithm that recovers the secret monomial from two pairs of equivalent codewords. For comparison, the previous best results require $c = \Omega(\log n)$ equivalent 2-dimensional subcodes [Beu20,BBPS22], corresponding to about $2c$ equivalent codewords.

High-level Idea of the Algorithm For two equivalent codes $\mathcal{C}$ and $\mathcal{C}' = \mathcal{C}\mathbf{Q}$ where $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ it holds that

$$\mathbf{GQH}'^{\top} = 0, \tag{2}$$

where $\mathbf{G} \in \mathbb{F}_q^{k \times n-k}$ is a generator matrix of $\mathcal{C}$ and $\mathbf{H}' \in \mathbb{F}_q^{n-k \times n}$ is a parity-check matrix of $\mathcal{C}'$. From this equation, one can derive the following linear system (see [Sae17, Corollary 3.2.20] or [BCDD+25, Proposition 2])

$$\mathsf{S}: \quad \overbrace{[\mathbf{G} \otimes \mathbf{H}']}^{\mathbf{A}} \mathbf{x} = \mathbf{0}, \tag{3}$$

where $\mathbf{x}$ is the length-n^2 column-vector $\mathsf{vec}(\mathbf{Q})$ formed by the concatenation of the rows of $\mathbf{Q}$. In particular, we have that $\mathsf{rank}(\mathbf{A}) = \mathsf{rank}(\mathbf{G}) \cdot \mathsf{rank}(\mathbf{H}') = k(n-k)$. Notice that this linear system itself is underdetermined, since the number of variables is $n^2 > k(n-k)$ for any code of dimension k. However, the non-linear restriction that $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ uniquely characterizes the solution.

In the following, we first show how to extract information on the monomial $\mathbf{Q}$ from any pair of equivalent codewords. Put simply, we exploit the fact that the (overlap of the) support of the given codewords carries information on the monomial matrix. In turn, this information allows us to reduce the number of variables of the linear system S. We find that for some code rates the information we extract

from two pairs of equivalent codewords is already sufficient to obtain a determined system, i.e., to recover the secret monomial $\mathbf{Q}$. In the case the system remains underdetermined, we make guesses on the remaining variables, exploiting the monomial structure. Due to the structure of the system, incorrect guesses often lead to an immediate contradiction, which reveals further information on the monomial matrix and ultimately leads to a determined system.

3.1 Extracting Hints from Equivalent Codewords

Let us start by showing how to extract information on the monomial matrix from a single pair of equivalent codewords. For this purpose we define the concept of a *hint*, which essentially corresponds to a known variable in the linear system from Eq. (3).

Definition 3 (Hint). *Let $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ be the monomial representing the solution to the system in Eq. (3). We define a* hint *relative to $\mathbf{Q}$ to be a pair*

$$((i,j),a) \in ([n] \times [n]) \times \mathbb{F}_q, \quad \textit{where } \mathbf{Q}(i,j) = a.$$

Given two equivalent linear codes $\mathcal{C}, \mathcal{C}'$, consider a pair of equivalent codewords $(\mathbf{v}, \mathbf{w}) \in \mathcal{C} \times \mathcal{C}'$ of Hamming weight w. Let $I := \mathrm{Supp}(\mathbf{v}) \subset [n]$ and $J := \mathrm{Supp}(\mathbf{w}) \subset [n]$, where $|I| = |J| = w$. Denote by $\bar{I}$ and $\bar{J}$ the complement of I and J in $[n]$, since the monomial matrix $\mathbf{Q}$ sends nonzero (resp. zero) entries to nonzero (resp. zero) entries, it follows that $\mathbf{Q}(i,j) = 0$ for all $(i,j) \in (I \times \bar{J}) \cup (\bar{I} \times J)$. This is equivalent to the set of hints

$$\mathsf{Determine\text{-}1}(\mathbf{v}, \mathbf{w}) := \{((i,j), 0) \mid (i,j) \in (I \times \bar{J}) \cup (\bar{I} \times J)\}, \tag{4}$$

of size $2w(n-w)$ on the monomial $\mathbf{Q}$. We define the procedure that on input a pair of codewords outputs this set of hints by $\mathsf{Determine\text{-}1}$.

The same reasoning extends naturally to multiple pairs of equivalent codewords. Given two pairs $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2) \in \mathcal{C} \times \mathcal{C}'$ of equivalent codewors, we define

$$I_\iota := \mathrm{Supp}(\mathbf{v}_\iota) \qquad \text{and} \qquad J_\iota := \mathrm{Supp}(\mathbf{w}_\iota), \quad \text{for } \iota = 1, 2,$$

and let $\ell := |I_1 \cap I_2| = |J_1 \cap J_2|$ be the number of shared support positions between the two codewords. Then the application of $\mathsf{Determine\text{-}1}$ to both pairs results in a set of hints $H = \mathsf{Determine\text{-}1}(\mathbf{v}_1, \mathbf{w}_1) \cup \mathsf{Determine\text{-}1}(\mathbf{v}_2, \mathbf{w}_2)$, of size

$$|H| = n^2 - 2(w-\ell)^2 - (n-2w+\ell)^2 - \ell^2. \tag{5}$$

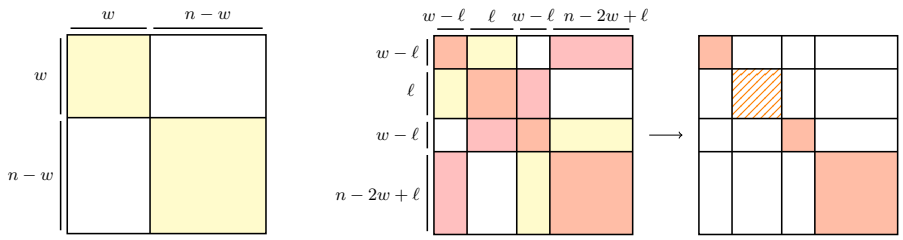
Therefore note that the structure of how indices are mapped under $\mathbf{Q}$ behaves as follows

- indices in $I_1 \cap I_2$ must be mapped to $J_1 \cap J_2$,
- indices in $I_1 \setminus I_2$ and $I_2 \setminus I_1$ are sent to $J_1 \setminus J_2$ and $J_2 \setminus J_1$ respectively, and
- indices outside both supports, i.e., in $[n] \setminus I_1 \cup I_2$ are sent to $[n] \setminus J_1 \cup J_2$.

Now the size of H follows by observing that the sizes of these sets are given by

$$|I_1 \cap I_2| = \ell, \quad |I_1 \setminus I_2| = |I_1 \setminus I_2| = w - \ell, \quad |[n] \setminus I_1 \cup I_2| = n - 2w + \ell.$$

We give a graphical illustration of the information derived on the monomial in Fig. 1 for one (left) and two pairs (right).



(a) Hints obtained from equivalent pair $\mathbf{v} = (\mathbf{v}', 0^{n-w})$, $\mathbf{w} = (\mathbf{w}', 0^{n-w})$, with $\mathbf{v}', \mathbf{w}' \in (\mathbb{F}_q^*)^w$

(b) Hints from two pairs $(\mathbf{v}_1, \mathbf{w}_1)$ (as on the left) and $\mathbf{v}_2 = (0^{w-\ell}, \mathbf{v}'', 0^{n-2w+\ell}), \mathbf{w}_2 = (0^{w-\ell}, \mathbf{w}'', 0^{n-2w+\ell})$, with $\mathbf{v}'', \mathbf{w}'' \in (\mathbb{F}_q^*)^w$

Fig. 1: Graphical representation of hints obtained from one (left) and two (right) pairs of equivalent codewords via Eq. (4). White regions correspond to zero entries, while colored regions contain the non-zero entries.

3.2 Extracting additional hints

Next, we show how to leverage the information two pairs of equivalent codewords carry on the monomial with respect to their support intersection to determine additional variables. More precisely, we recover information on the striped $\ell \times \ell$ submatrix in Figure 1b. The core characteristic we exploit is that codeword entries that are mapped under the monomial must differ by the same multiplicative shift in both given codeword pairs. As long as any of those entries is zero, this does not yield strong restrictions on the monomial. However, concentrating on the support intersection can reveal further information.

Restrictions on the monomial implied by the support intersection. For two pairs of equivalent codewords $(\mathbf{v}_\iota, \mathbf{w}_\iota) \in \mathcal{C} \times \mathcal{C}'$, $\iota = 1, 2$ let $I = \mathrm{Supp}(\mathbf{v}_1) \cap \mathrm{Supp}(\mathbf{v}_2)$ and $J = \mathrm{Supp}(\mathbf{w}_1) \cap \mathrm{Supp}(\mathbf{w}_2)$. From the argumentation in the previous section we know that a monomial with $\mathbf{v}_\iota = \mathbf{w}_\iota \mathbf{Q}$ must map indices in J to indices in I. This implies that for any $i \in I$ there exist a $j \in J$ such that

$$\mathbf{v}_1(i) = \alpha_{i,j} \cdot \mathbf{w}_1(j) \quad \text{and} \quad \mathbf{v}_2(i) = \alpha_{i,j} \cdot \mathbf{w}_2(j) \quad \text{for some } \alpha_{i,j} \in \mathbb{F}_q^*, \tag{6}$$

where $\alpha_{i,j} = \mathbf{Q}(i,j)$. Note that this implies that $\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)} = \frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}$. Therefore whenever we have that $\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)} \neq \frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}$ it follows that j is not mapped to i under the monomial and hence $\mathbf{Q}(i,j) = 0$. Furthermore, if for any i there is a unique j satisfying Eq. (6), we can conclude that $\mathbf{Q}(i,j) = \alpha_{i,j} = \frac{\mathbf{v}_1(i)}{\mathbf{w}_1(j)}$.

In Algorithm 1 we formalize the procedure of obtaining the additional hints related to the support intersection of both equivalent pairs.

Analysis of Algorithm 1 We start the analysis with the following lemma on the correctness and time complexity of Algorithm 1.

Algorithm 1 Determine-2$((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2))$

Input: Two pairs of equivalent codewords $(\mathbf{v}_i, \mathbf{w}_i), \in \mathcal{C} \times \mathcal{C}'$, $i = 1, 2$.
Output: A set of up to ℓ^2 hints $((i, j), \alpha)$ such that $\mathbf{Q}(i, j) = \alpha \in \mathbb{F}_q$, where $\ell := |\operatorname{Supp}(\mathbf{v}_1) \cap \operatorname{Supp}(\mathbf{v}_2)|$.

1: Let $I = \operatorname{Supp}(\mathbf{v}_1) \cap \operatorname{Supp}(\mathbf{v}_2)$ and $J = \operatorname{Supp}(\mathbf{w}_1) \cap \operatorname{Supp}(\mathbf{w}_2)$
2: $H \leftarrow \emptyset$
3: **for** i in I **do**
4: $\quad \mathsf{count} \leftarrow 0$
5: $\quad$ **for** j in J **do**
6: $\quad\quad$ **if** $\mathbf{v}_1(i)/\mathbf{v}_2(i) = \mathbf{w}_2(j)/\mathbf{w}_2(j)$ **then**
7: $\quad\quad\quad \alpha \leftarrow \mathbf{v}_1(i)/\mathbf{w}_1(j)$
8: $\quad\quad\quad \mathsf{col} \leftarrow j$
9: $\quad\quad\quad \mathsf{count} \leftarrow \mathsf{count} + 1$
10: $\quad\quad$ **else**
11: $\quad\quad\quad H \leftarrow H \cup \{((i, j), 0)\}$
12: $\quad$ **if** $\mathsf{count} = 1$ **then**
13: $\quad\quad H \leftarrow H \cup \{((i, \mathsf{col}), \alpha)\}$
14: **Return** H

Lemma 2 (Complexity of Algorithm 1). *Algorithm 1 returns a set H containing $|H| \leq \ell^2$ hints on the monomial $\mathbf{Q}$ in time $\mathcal{O}(\ell^2)$.*

Proof. The correctness of the algorithm follows from the argumentation above. Therefore, observe that whenever for $(i, j) \in I \times J$ it holds that $\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)} \neq \frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}$ the corresponding entry $\mathbf{Q}(i, j)$ is set to zero in Line 11. Furthermore, if for any i there is only a unique j satisfying Eq. (6), we set $\mathbf{Q}(i, j) = \alpha$ in Line 13.

The time complexity of the algorithm is $|I \times J| = \ell^2 \leq n^2$ times the time for a single iteration of the inner loop. Note that the time for one iteration of the inner loop is dominated by the field inversion in Line 6, leading to the stated running time in terms of field operations. □

Note that Lemma 2 only specifies an upper bound on the number of hints obtained. In following we analyze the expected amount of hints obtained over the random choice of equivalent pairs $(\mathbf{v}_i, \mathbf{w}_i)$, $i = 1, 2$.

Lemma 3 (Hints returned by Algorithm 1). *The set H of hints returned by Algorithm 1 is of expected size*

$$\mathbb{E}\big[|H|\big] = \ell^2 - \sum_{t=2}^{\ell} t^2 (q-1) \binom{\ell}{t} \left(\frac{1}{q-1}\right)^t \left(\frac{q-2}{q-1}\right)^{\ell - t}, \tag{7}$$

over the random choice of its inputs.

Proof. Consider the vectors $\mathbf{v} = \left(\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)}\right)_{i \in I}$ and $\mathbf{w} = \left(\frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}\right)_{j \in J}$. First note that Equation (6) implies that for any $\alpha \in \mathbf{v}$ there must be a corresponding $\alpha \in \mathbf{w}$. Moreover, the number of appearances of any $\alpha \in \mathbb{F}_q^*$ is the same in $\mathbf{v}$ and $\mathbf{w}$.

Now, observe that if all entries of $\mathbf{v}$ and $\mathbf{w}$ are unique, each iteration of the loop of Algorithm 1 contributes to the size of H and hence $|H| = \ell^2$. On the other hand if any entry α is repeated t times, with, e.g., $\mathbf{v}(i) = \mathbf{w}(j)$ for $(i,j) \in I_\alpha \times J_\alpha$, then all t^2 iterations $(i,j) \in I_\alpha \times J_\alpha$ will not contribute to H. In fact if there are N_t distinct entries in $\mathbf{v}$ that are all repeated t times, the searched expectation can be written as

$$\mathbb{E}\big[|H|\big] = \ell^2 - \sum_{t=2}^{\ell} t^2 \mathbb{E}[N_t].$$

To estimate the number of repeated entries within those vectors we model the construction of $\mathbf{v}$ as a balls into bins problem. Precisely, there are $q-1$ bins (elements in $\mathbb{F}_q^*$) and ℓ balls (components of $\mathbf{v}$). Since $\mathbf{v}_1, \mathbf{v}_2$ are drawn uniformly at random, so is $\mathbf{v}$, which implies that balls are thrown randomly into the bins. Let X_α be the random variable counting the load of bin $\alpha \in \mathbb{F}_q^*$. Then we expect that for any $t = 1, \ldots, \ell$, there are

$$\mathbb{E}[N_t] = \sum_{\alpha \in \mathbb{F}_q^*} \Pr\left[X_\alpha = t\right] = (q-1)\Pr\left[X_\alpha = t\right],$$

bins which hold exactly t balls. Now, the probability that any bin in such a setting contains exactly t balls is known to be

$$\Pr\left[X_\alpha = t\right] = \binom{\ell}{t} \left(\frac{1}{q-1}\right)^t \left(\frac{q-2}{q-1}\right)^{\ell-t},$$

which leads to the statement of the lemma. □

Moreover, we show in Appendix A that $\mathbb{E}\big[|H|\big] \approx \ell^2$ for sufficiently large $q > \ell$.

3.3 Solving Linear Code Equivalence with Hints

Let us analyze the impact of the hints provided by two pairs of equivalent codewords on the linear system S in Equation (3). Let $H_{\text{2-pairs}}$ be the set of hints obtained from two pairs $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2) \in \mathcal{C} \times \mathcal{C}'$ of equivalent codewords of Hamming weight w, i.e.,

$$H_{\text{2-pairs}} = \mathsf{Determine\text{-}1}(\mathbf{v}_1, \mathbf{w}_1) \cup \mathsf{Determine\text{-}1}(\mathbf{v}_2, \mathbf{w}_2) \cup \mathsf{Determine\text{-}2}\big((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)\big).$$

By combining Equation (5) and Lemma 3, we find that the expected size of $H_{\text{2-pairs}}$ is $\mathbb{E}\big[|H_{\text{2-pairs}}|\big] = n^2 - v(n, w, \ell, q)$, where

$$\begin{aligned} v(n,w,\ell,q) &:= 2(w-\ell)^2 + (n-2w+\ell)^2 + \sum_{t=2}^{\ell} t^2 (q-1) \binom{\ell}{t} \left(\frac{1}{q-1}\right)^t \left(\frac{q-2}{q-1}\right)^{\ell-t} \\ &\approx 2(w-\ell)^2 + (n-2w+\ell)^2, \quad \text{for large enough } q. \end{aligned} \tag{8}$$

Obtaining a reduced linear system. From here we construct the following linear system incorporating all hints from $H_{\text{2-pairs}}$ into S

$$\mathsf{S}_{\mathsf{red}}: \quad \begin{cases} \mathbf{Ax} = \mathbf{0} \\ \mathbf{x}(i \cdot n + j) = a, \quad \forall((i,j),a) \in H_{\text{2-pairs}}. \end{cases} \tag{9}$$

Note that this results in a non-homogeneous system of the form

$$\mathsf{S}_{\mathsf{red}}: \quad \mathbf{A}_{\mathsf{red}}\mathbf{x} = \mathbf{b}, \tag{10}$$

with $\mathbf{b} \neq \mathbf{0}$, as long as there is a hint $((i,j),a) \in H_{\text{2-pairs}}$ with $a \neq 0$. If $\mathsf{S}_{\mathsf{red}}$ behaves as a random system, we would expect that as long as

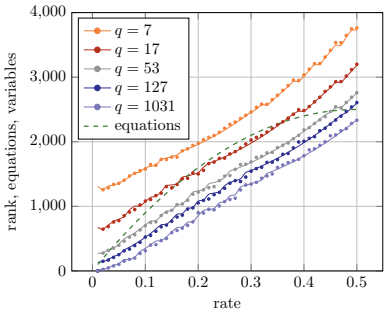
$$k(n-k) \geq v(n,w,\ell,q), \tag{11}$$

the system contains more equations than unknowns and is, hence, determined. If we assume the two pairs of weight-w equivalent codewords overlap in a number of coordinates $\ell = \frac{w^2}{n}$ equal to its expectation, we find that $v(n,w,\ell,q)$ for large enough q is strictly decreasing in w. We therefore assume in following $w = w_{\text{GV}}$ is minimal, i.e., equal to the minimum distance of $\mathcal{C}, \mathcal{C}'$, leading to the maximum amount of remaining variables, representing the worst case possible.

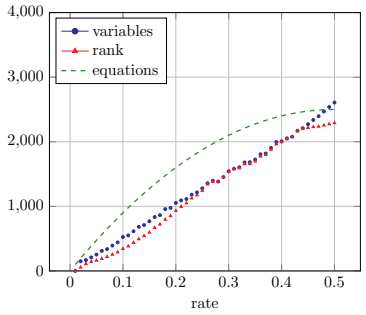
In Fig. 2 (on the left) we illustrate for $n = 100$ and multiple choices of q the (exact) value of $v(n,w,\ell,q)$ from Eq. (8) (solid lines) and the number of variables recorded when experimentally constructing the system $\mathsf{S}_{\mathsf{red}}$ (marks) from two pairs of equivalent codewords of weight-w_{GV} as a function of the code rate. Note that the amount of variables is decreasing with q since w_{GV} grows with q and, additionally, Determine-2 leads to a larger set of hints for growing q. We illustrate the number of equations $k(n-k)$ as a green dashed line. From this graphic, one would expect that the system should be determined for $q = 127$ (blue marks) for almost all rates since the number of variables lies below the number of equations. However, on the right, for the case of $q = 127$ we illustrate additionally the experimentally observed rank of the system. As can be observed, in some cases, the rank of the system is found maximal, i.e., equal to the number of variables, which allows to recover the solution to $\mathsf{S}_{\mathsf{red}}$ and, hence, the solution to the LCE problem. In many cases, however, the rank is lower than the number of variables, leaving the system underdetermined.

Recovering Q when $\mathsf{S}_{\mathsf{red}}$ is underdetermined The core idea of the following procedure is to make guesses on the positions of non-zero entries of $\mathbf{Q}$. By exploiting the monomial structure of $\mathbf{Q}$, guessing a single non-zero entry simultaneously eliminates multiple variables from $\mathsf{S}_{\mathsf{red}}$. If the resulting system no longer admits a solution, the initial guess was wrong, revealing further information on $\mathbf{Q}$. By repeating this procedure multiple times, we eventually obtain enough hints for $\mathsf{S}_{\mathsf{red}}$ to become determined.

The Rouché-Capelli test. To check if a linear system still admits a solution, we rely on a fundamental result in linear algebra, the Rouché–Capelli theorem. This result



(a) Number of variables vs. equations for different q and k.



(b) Expected number of variables vs rank of the system for $q = 127$.

Fig. 2: Number of equations, rank and amount of variables in the system $\mathsf{S}_{\mathsf{red}}$ for $n = 100$, $w = w_{\mathrm{GV}}$.

states that a linear system admits a solution if and only if the rank of its coefficients matrix equals the rank of its augmented matrix. In the context of the linear system $\mathsf{S}_{\mathsf{red}}$, this condition reads

$$\mathsf{rank}(\mathbf{A}_{\mathsf{red}}) = \mathsf{rank}(\mathbf{A}_{\mathsf{red}}|\mathbf{b}),$$

which is satisfied, as $\mathsf{S}_{\mathsf{red}}$ admits a solution by construction.

Exploiting the monomial structure. Let us assume that $\mathbf{Q}(i,j) \neq 0$ and that $((i,j),*) \notin H_{\text{2-pairs}}$, meaning that the entry (i,j) has not been determined by Determine-2. Then, considering the monomial structure of $\mathbf{Q}$, we know that all other entries along the i-th row and j-th column are equal to zero. Thus, we construct the following set of hints

$$H_{(i,j)} = \{((i,j'),0) \text{ for } j' \in [n] \setminus \{j\}\} \cup \{((i',j),0) \text{ for } i' \in [n] \setminus \{i\}\}. \tag{12}$$

Now define the following linear system by additionally incorporating the hints from $H_{(i,j)}$ into $\mathsf{S}_{\mathsf{red}}$

$$\mathsf{S}_{(i,j)}: \quad \begin{cases} \mathbf{Ax} = 0 \\ \mathbf{x}(n \cdot i + j) = a, \quad \forall((i,j),a) \in H_{\text{2-pairs}} \cup H_{(i,j)}. \end{cases} \tag{13}$$

Let us denote with $\mathbf{A}_{\mathsf{guess}}$ the resulting matrix of coefficients in Equation (13)

$$\mathsf{S}_{(i,j)}: \quad \mathbf{A}_{\mathsf{guess}}\mathbf{x} = \mathbf{b}. \tag{14}$$

Notice that $\mathbf{A}_{\mathsf{guess}}$ has $k(n-k)$ rows and an expected number of columns of

$$v(n,w,\ell,q) - |H_{(i,j)}| + c = v(n,w,\ell,q) - 2(n-1) + c,$$

where $c = |H_{(i,j)} \cap H_{\text{2-pairs}}|$ counts the number of hints with respect to the i-th row and j-th column already present in $H_{\text{2-pairs}}$.

Guessing entries of $\mathbf{Q}$. We define a *guess* on an entry (i, j) of $\mathbf{Q}$ to be the act of constructing the linear system $\mathsf{S}(i, j)$. A guess is *correct* if $\mathbf{Q}(i, j) \neq 0$, and *incorrect* if $\mathbf{Q}(i, j) = 0$. While correct guesses preserve the solution to the system, this is not the case for incorrect ones. Indeed, any incorrect guess on an entry (i, j), would lead to the wrong assignment of zero values to $\mathbf{Q}(i, j^*)$ and $\mathbf{Q}(i^*, j)$, where (i, j^*) and (i^*, j) are the positions of the non-zero entries in the i-th row and j-th column of $\mathbf{Q}$ respectively. Hence, the system $\mathsf{S}_{(i,j)}$ is not guaranteed to admit a solution anymore. Summarizing, we can draw the following conclusions

- If $\mathsf{S}_{(i,j)}$ *does not* admit a solution, which is equivalent to $\mathsf{rank}(\mathbf{A}_{\mathsf{guess}}) < \mathsf{rank}(\mathbf{A}_{\mathsf{guess}}|\mathbf{b})$, then the guess is incorrect, which implies $\mathbf{Q}(i, j) = 0$.
- If $\mathsf{S}_{(i,j)}$ admits a solution, or equivalently $\mathsf{rank}(\mathbf{A}_{\mathsf{guess}}) = \mathsf{rank}(\mathbf{A}_{\mathsf{guess}}|\mathbf{b})$, no conclusion can be reached on the correctness of the guess.

Based on those two cases we define in Algorithm 2 a procedure that identifies incorrect guesses and in that case returns a corresponding hint on the original system.

Algorithm 2 $\mathsf{RoucheCapelli\text{-}test}(\mathsf{S}, H, (i, j))$

Input: linear system S, set of hints H, pair of indices (i, j).
Output: a hint $((i, j), 0)$ or $\perp$.

1: Construct the linear system $\mathsf{S}_{(i,j)}$ from Eq. (14)
2: **if** $\mathsf{rank}(\mathbf{A}_{\mathsf{guess}}) < \mathsf{rank}(\mathbf{A}_{\mathsf{guess}}|\mathbf{b})$ **then**
3: **Return** $((i, j), 0)$ ▷ The guess is incorrect, implying $\mathbf{Q}(i, j) = 0$.
4: **Return** $\perp$ ▷ Result inconclusive, $\mathbf{Q}(i, j)$ might be nonzero.

Guesses leading to hints in Algorithm 2. We have already argued that incorrect guesses on an entry (i, j) lead to a system $\mathsf{S}_{(i,j)}$, which is not guaranteed to admit a solution, i.e., it does not contain the solution $\mathbf{x}$ related to the original monomial. However, there might exist other solutions $\mathbf{x}'$ (without monomial structure) satisfying $\mathbf{A}_{\mathsf{guess}}\mathbf{x}' = \mathbf{b}$. In such a case $\mathbf{b}$ is still in the span of $\mathbf{A}_{\mathsf{guess}}$, implying $\mathsf{rank}(\mathbf{A}_{\mathsf{guess}}) = \mathsf{rank}(\mathbf{A}_{\mathsf{guess}}|\mathbf{b})$, and, hence, no hint is returned. On the other hand, if no such $\mathbf{x}'$ exist, it follows $\mathsf{rank}(\mathbf{A}_{\mathsf{guess}}) < \mathsf{rank}(\mathbf{A}_{\mathsf{guess}}|\mathbf{b})$ and Algorithm 2 returns the respective hint. Since correct guesses always preserve $\mathbf{b}$ in the span of $\mathbf{A}_{\mathsf{guess}}$, hints can only be returned for incorrect guesses.

Now the probability that any random element $\mathbf{b}'$ in the span of $\mathbf{A}_{\mathsf{red}}$ also lies in the span of $\mathbf{A}_{\mathsf{guess}}$ is approximately $q^{\mathsf{rank}(\mathbf{A}_{\mathsf{guess}}) - \mathsf{rank}(\mathbf{A}_{\mathsf{red}})}$. Note that this probability is non-trivial only if the guess (and the related hints from Eq. (12)) result in a decrease of the rank of the coefficient matrix, i.e., if

$$\mathsf{rank}(\mathbf{A}_{\mathsf{guess}}) < \mathsf{rank}(\mathbf{A}_{\mathsf{red}}) \tag{15}$$

Recall that both matrices describe underdetermined linear systems, with the same amount of equations. Therefore, random systems would share the same rank with

high probability. However, in our experiments we find that the tensor structure of the system (compare to Eq. (2)) causes guesses to frequently reduce the rank of $\mathbf{A}_{\mathsf{guess}}$ compared to $\mathbf{A}_{\mathsf{red}}$ and in turn leads to the detection of incorrect guesses in Line 3 of Algorithm 2.

We note that, due to the tensor product structure, the matrix $\mathbf{A}_{\mathsf{guess}}$ can be constructed by removing $2(n-1)-c$ columns from $\mathbf{A}_{\mathsf{red}}$, where $c = |H_{(i,j)} \cap H_{2\text{-pairs}}|$. A rank decrease from $\mathbf{A}_{\mathsf{red}}$ to $\mathbf{A}_{\mathsf{guess}}$ therefore implies the existence of a codeword of weight $w < 2(n-1) - c$ in the code with generator matrix $\mathbf{A}_{\mathsf{red}}$, whose support lies entirely in the positions corresponding to the removed columns. Since this code is of length $v(n, \ell, w, q) \sim n^2$ such a codeword represents an unusually small weight relative to the code length. We give further theoretical evidence for the existence of an exponential amount of these codewords in form of an explicit construction in Appendix B.

The full procedure. We are now ready to give in Algorithm 3 the full procedure to recover the secret monomial from two pairs of equivalent codewords. The algorithm first constructs the hints via the Determine-1 and Determine-2 procedures to construct the reduced system $\mathsf{S}_{\mathsf{red}}$. In case $\mathsf{S}_{\mathsf{red}}$ remains underdetermined, the algorithm iterates through all guesses for uknown entries (i, j) of $\mathbf{Q}$, i.e., entries for which no corresponding hint is present in $H_{2\text{-pairs}}$. For each such guess we attempt to generate a new hint via RoucheCapelli-test and, if successful, incorporate the hint into $\mathsf{S}_{\mathsf{red}}$. If after all possible guesses, $\mathsf{S}_{\mathsf{red}}$ is finally determined, the secret monomial is returned.

Lemma 4 (Complexity of Algorithm 3). *Algorithm 3 runs in expected time* $\mathcal{O}(v(n, \ell, w, q)^4)$ *and memory* $\mathcal{O}(n^4)$*, for* $v(n, \ell, w, q)$ *as in Eq.* (8)*.*

Proof. The time complexity of Algorithm 3 is dominated by the **for**-loop in Line 5. The number of guesses to test via RoucheCapelli-test is equal to the number of unknowns of $\mathsf{S}_{\mathsf{red}}$, whose expected value is $v := v(n, w, \ell, q)$. The time complexity of RoucheCapelli-test is equal to two rank computations which can be performed in time $\mathcal{O}(v^\delta)$, where δ is the linear algebra constant. Therefore, the overall time complexity of the algorithm is $\mathcal{O}(v^{\delta+1}) = \mathcal{O}(v^4)$. □

Note that since $v(n, w, \ell, q) \leq n^2$ the algorithm runs in time polynomial in n. The following lemma shows the correctness of the algorithm in the sense that if $\mathsf{S}_{\mathsf{red}}$ at the end of the algorithm is a determined system, it leads to the retrieval of the secret monomial.

Lemma 5 (Correctness of Algorithm 3). *Algorithm 3 either returns a matrix* $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ *defining an equivalence between* $\mathcal{C}, \mathcal{C}'$ *or* $\perp$*.*

Proof. First, observe that the hints computed in line 2 are, by construction, consistent with the secret monomial $\mathbf{Q}$ given the pairs $(\mathbf{v}_i, \mathbf{w}_i)$, $i = 1, 2$ are equivalent. This follows from the correctness of the Determine-1 and Determine-2 procedures (compare to Section 3.1 and Lemma 2). In the **for**-loop in line 5 guesses on entries (i, j) of $\mathbf{Q}$ are preformed and it is verified whether those guesses lead to a contradiction using Algorithm 2. Recall that a guesses on entry (i, j) assumes $\mathbf{Q}(i, j) \neq 0$.

Algorithm 3 Compute-Monomial$(\mathbf{G}, \mathbf{H}', (\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2))$

Input: Generator $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ of linear code $\mathcal{C}$, parity-check $\mathbf{H}' \in \mathbb{F}_q^{(n-k) \times n}$ of linear code $\mathcal{C}'$, equivalent codewords $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2) \in \mathcal{C} \times \mathcal{C}'$ of Hamming weight w.

Output: A matrix $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ such that $\mathbf{G}\mathbf{Q}\mathbf{H}'^{\top} = \mathbf{0}$ or $\perp$.

1: Construct the linear system S from Equation (3)
2: Compute hints $H_{2\text{-pairs}} = \mathsf{Determine\text{-}1}(\mathbf{v}_1, \mathbf{w}_1) \cup \mathsf{Determine\text{-}1}(\mathbf{v}_2, \mathbf{w}_2) \cup \mathsf{Determine\text{-}2}((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2))$
3: Reduce S to $\mathsf{S}_{\mathsf{red}}$ using the hints in $H_{2\text{-pairs}}$ (see Equation (9))
4: **if** $\mathsf{S}_{\mathsf{red}}$ is underdetermined **then**
5: **for** $(i, j) \in [n] \times [n] \colon (i, j, *) \notin H_{2\text{-pairs}}$ **do**
6: $\mathsf{hint} \leftarrow \mathsf{RoucheCapelli\text{-}test}(\mathsf{S}, H_{2\text{-pairs}}, (i, j))$
7: **if** $\mathsf{hint} \neq \perp$ **then**
8: Update $\mathsf{S}_{\mathsf{red}}$ according hint and add hint to $H_{2\text{-pairs}}$
9: **if** $\mathsf{S}_{\mathsf{red}}$ is determined **then**
10: Compute solution $\mathbf{s}$ to $\mathsf{S}_{\mathsf{red}}$
11: Construct matrix $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ with $\mathbf{G}\mathbf{Q}\mathbf{H}'^{\top} = \mathbf{0}$ using $H_{2\text{-pairs}}$ and $\mathbf{s}$
12: **Return** $\mathbf{Q}$
13: **Return** $\perp$

Since RoucheCapelli-test only returns hints of the form $((i, j), 0)$ and only if the guesses was incorrect, it follows that such hints are consistent with the solution. Therefore, if $\mathsf{S}_{\mathsf{red}}$ becomes fully determined in line 10, by construction, the solution must correspond to the monomial $\mathbf{Q}$, defining the equivalence. On the other hand, if $\mathsf{S}_{\mathsf{red}}$ is still underdetermined after all guesses, the algorithm returns a failure. □

In order to rigorously prove the effectiveness of Algorithm 3 we would need to argue about its success probability. That is the probability that the system $\mathsf{S}_{\mathsf{red}}$ is determined at the end of the procedure, leading to the recovery of the secret monomial. However, the induced structure of the system and existing dependencies make a formal analysis challenging. While we offer additional theoretical insights and arguments supporting the algorithm's effectiveness in Appendix B, for now we resort to the following heuristic. We then provide extensive experimental evidence demonstrating its validity in the following section.

Heuristic 1 (Success Probability of Algorithm 3) *Let $n, k, q, w \in \mathbb{N}$ with $w_{GV} \leq w \leq n - k$ and $q \geq 23$. Then Algorithm 3 returns a monomial matrix $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ with high probability.*

3.4 Implementation and Experiments

We provide a full proof of concept implementation of Algorithm 3 in `SageMath` [The22] available at [BEFN]. Based on this implementation, we provide extensive

experiments confirming a high success probability of Algorithm 3 across all parameters for sufficiently large $q \geq 23$. All experiments were carried out on a server equipped with two `AMD EPYC 7763 64-Core Processors` and 1TB of RAM.

Sampling instances. In each experiment, we sample a random LCE instance, where the involved codes are guaranteed to include two codewords of minimum weight. Therefore we construct a basis of $\mathcal{C}$ consisting of $k-2$ random elements in $\mathbb{F}_q^n$ and two random weight-w_{GV} vectors $\mathbf{v}_i \in \mathbb{F}_q^n$.[2] We then sample a random monomial $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ and set $\mathcal{C}' = \mathcal{C}\mathbf{Q}$ and $\mathbf{w}_i = \mathbf{v}_i\mathbf{Q}$.

Practical optimizations. We implemented various practical improvements to speed up Algorithm 3 as well as to boost its success probability for smaller values of q. The time improvements include, an early abort of the loop whenever the system becomes determined after the inclusion of a new hint as well as further exploiting the monomial structure. For the latter we observe that, whenever all guesses along a row i^* of $\mathbf{Q}$ are found to be incorrect, except one in column j^*, it implies $\mathbf{Q}(i^*, j^*) \neq 0$. From the monomial structure of $\mathbf{Q}$ it then follows that all other entries along the same column $\mathbf{Q}(i, j^*)$, for $i \neq i^*$, are equal to zero. Furthermore we observe that if hints could be generated during the **for**-loop via RoucheCapelli-test but the system remains underdetermined at the end of the algorithm, a re-iteration of the **for**-loop may increase the success probability. In such cases the loop is re-applied with the updated set of hints $H_{\text{2-pairs}}$, leading to a simpler system $\mathsf{S}_{\mathsf{red}}$ and in turn to more incorrect guesses being detected by RoucheCapelli-test. However, we find that such a reapplication is only necessary for rather small choices of $q < 31$.

Testing the success probability. We report in Table 2 the empirical success probability for fixed $n = 100$ for different values of the rate $R = k/n$ and increasing values of q over a sample size of 100 for each set of parameters. We treated an experiment on a random LCE instance as successful if at the end of the algorithm the secret monomial $\mathbf{Q}$ could be recovered. We observe that already for $q \geq 23$ the success probability is found to be empirically equal to one in all cases analyzed.

		n = 100				
q	rate	0.1	0.2	0.3	0.4	0.5
7		0.02	0.37	0.34	0.48	0.07
11		0.38	0.74	0.96	0.76	0.76
13		0.56	0.84	0.98	0.90	0.86
17		0.90	1.00	1.00	1.00	0.95
19		0.99	1.00	1.00	1.00	1.00
23		1.00	1.00	1.00	1.00	1.00
127		1.00	1.00	1.00	1.00	1.00

Table 2: Empirical success probability of Algorithm 3 over 100 random trials for $n = 100$, different rates and q.

[2]Recall, that a weight of w_{GV} represents the worst case, as it leads to the largest number of remaining variables (compare to Eq. (8)).

Area of effectiveness for increasing q. To further demonstrate the effectiveness of Algorithm 3 across various parameters, we present experimental results in Fig. 3. Each dot represents a single run of Algorithm 3 on random LCE instance. *Green dots* indicate cases where $\mathsf{S}_{\mathsf{red}}$ was already determined based on the hints computed in Line 3. *Blue dots* represent instances where the **for**-loop in Line 4 successfully recovered the monomial. Unsuccessful runs, in which the monomial could not be recovered are marked as *red dots*. The *gray dots* denote codes of dimension 1 and 2. For dimension one codes, the experiment can not be conducted as two linearly independent codewords are required as input to the algorithm. For dimension two codes, on the other hand, there exist exponentially many solutions. Note that multiple solutions reduce the success probability of Algorithm 3 as incorrect guesses do not necessarily eliminate all solutions anymore. Note that, the red dots in case of $q = 23$ are related to instances which admit multiple solutions, which happens only for very small dimensions $k \leq 4$. Recall, that we generally restrict our attention to instances with unique solutions, for which the algorithm is found to be successful for any set of parameters with $q \geq 23$.

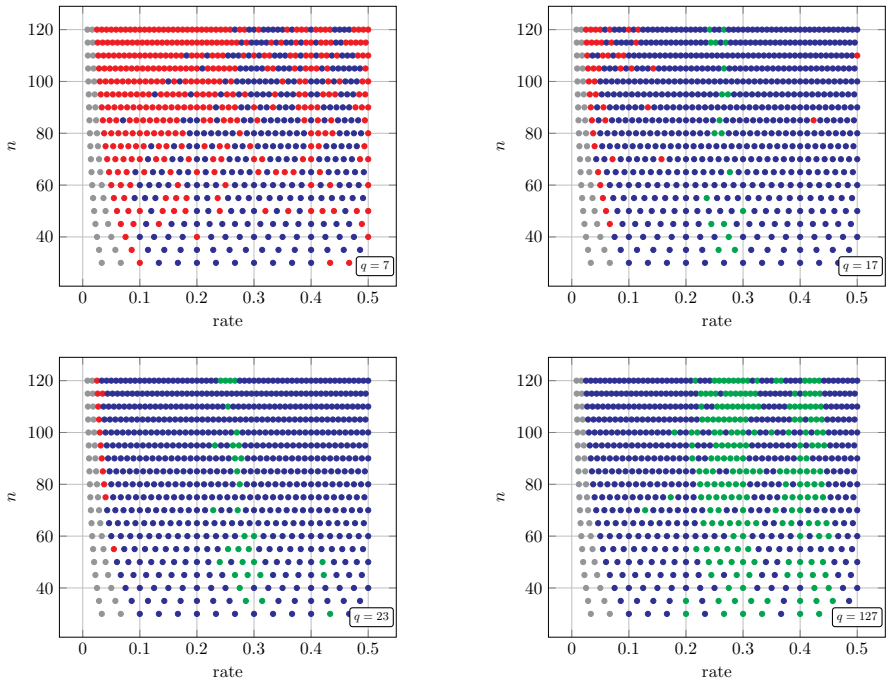


Fig. 3: Success and failure areas of Algorithm 3 for different n, q and code rates.

Tests on cryptographic parameters. To further enhance the confidence in the success probability for large parameters, we successfully retrieved the secret monomial from a random LCE instance and two pairs of equivalent codewords with parameters

$(n, k, q, w) = (252, 126, 127, 94)$, corresponding to a choice made in the LESS signature scheme in about 11.8 hours. A repeated experiment with $w = 107$ motivated by the application in the following section, let to the recovery of the secret monomial in about 8.5 hours.

4 A New Algorithm Solving Code Equivalence

In the following, we propose a new algorithm for solving Linear Code Equivalence, leading to significant concrete as well as asymptotic improvements over the state of the art. At the core of the proposed algorithm lies the observation from the previous section that two pairs of equivalent codewords are sufficient to recover the secret monomial (see Algorithm 3).

High level description. The algorithm first computes two sufficiently large lists of weight w codewords in both codes, where w is an optimization parameter. After the construction of those lists, pairs of codewords are formed. We then define a criterion to check the compatibility of pairs with respect to monomial transformations, where pairs are deemed compatible if there exists any monomial transformation (not necessarily $\mathbf{Q}$) between them. Eventually, for each pair of codewords found to be compatible, we initiate the reconstruction of the secret monomial via Algorithm 3. As shown in the previous section, this reconstruction is successful whenever the two codeword pairs were indeed equivalent under the searched monomial $\mathbf{Q}$. We summarize this procedure in Algorithm 4.

Algorithm 4 Solving LCE

Input: $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n-k}$

Output: $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ with $\mathbf{G} = \mathbf{S}\mathbf{G}'\mathbf{Q}$ for some invertible $\mathbf{S} \in \mathbb{F}_q^{k \times k}$ or $\perp$

choose w and N optimally
Compute a list L_1 of codewords of weight w in $\mathbf{G}$, with $|L_1| = N$
Compute a list L_2 of codewords of weight w in $\mathbf{G}'$, with $|L_2| = N$
Construct $V = L_1 \times L_1$ and $W = L_2 \times L_2$ containing pairs of codewords
$Z \leftarrow \mathsf{Compatible\text{-}Pairs}(V, W)$
for $\big((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)\big) \in Z$ **do**
    $\mathbf{Q} \leftarrow \mathsf{Compute\text{-}Monomial}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{w}_1, \mathbf{w}_2, \mathbf{G}, \mathbf{G}')$
    if $\mathbf{Q} \neq \perp$ **then**
        Return $\mathbf{Q}$
Return $\perp$

4.1 An Asymptotic Version of the Algorithm

We start the analysis of the algorithm from an asymptotic perspective. Therefore, for now we define

$$\mathsf{Compatible\text{-}Pairs}(V,W) := \{\big((\mathbf{v}_1,\mathbf{w}_1),(\mathbf{v}_2,\mathbf{w}_2)\big) \mid (\mathbf{v}_1,\mathbf{v}_2) \in V \wedge (\mathbf{w}_1,\mathbf{w}_2) \in W\}, \tag{16}$$

i.e., as all possible pairs constructed from the lists V, W. In Section 4.2, we introduce a more efficient procedure for finding compatible pairs, leading to significant *concrete* improvements. However, we then also prove that from an asymptotic perspective the procedure defined in Eq. (16) is optimal (up to polynomial factors) for any constant field size q. For clarity we, hence, stick with the simple procedure during the asymptotic analysis. The following theorem summarizes the asymptotic time and memory complexity of Algorithm 4.

Theorem 1 (Complexity of Algorithm 4). *Let $n, q \in \mathbb{N}$, $k = Rn$ for constant rate R, $w \leq n-k$. Then Algorithm 4 solves the* LCE *problem under Heuristic 1 in expected*

$$\textit{time } T = \tilde{\mathcal{O}}\left(\max\left(\frac{\binom{n}{w}}{\binom{n-k}{w}\sqrt{N_w}}, N_w^2\right)\right), \quad \textit{and} \quad \textit{memory } M = \tilde{\mathcal{O}}\left(\sqrt{N_w}\right),$$

with high probability.

Proof. We fix $N = c \cdot \sqrt{N_w}$ for some large enough constant c. The time complexity of the algorithm splits in the time complexity to compute two lists of weight-w codewords of size $|L_i| = N$ and the time complexity to apply the Compute-Monomial function to each element $v \in Z$, where $|Z| = |V \times W| = N^4$. Note that, since the recomputation of the solution via the Compute-Monomial function runs in polynomial-time (see Lemma 4), the complexity for the latter is

$$T_{\text{Rec}} = \tilde{\mathcal{O}}\left(|V \times W|\right) = \tilde{\mathcal{O}}\left(N^4\right) = \tilde{\mathcal{O}}\left(N_w^2\right)$$

The time to find N weight-w codewords in an $[n,k]$ linear code over $\mathbb{F}_q$ is given by Lemma 1 as

$$T_{\text{ISD}} = \tilde{\mathcal{O}}\left(\frac{\binom{n}{w}}{\binom{n-k}{w}\sqrt{N_w}}\right),$$

In summary, we obtain the claimed time complexity as $T = \max(T_{\text{ISD}}, T_{\text{Rec}})$. For the memory complexity, observe that we can check elements from Z on-the-fly and do not have to store those lists. Therefore the memory complexity is dominated by the size of L_1, L_2, giving $M = \tilde{\mathcal{O}}\left(N\right) = \tilde{\mathcal{O}}\left(\sqrt{N_w}\right)$.

For the correctness it remains to show that for $|L_i| = c\sqrt{N_w}$ there are at least two codewords $\mathbf{v}_1, \mathbf{v}_2 \in L_1$ and two codewords $\mathbf{w}_1, \mathbf{w}_2 \in L_2$ with $\mathbf{w}_i = \mathbf{v}_i\mathbf{Q}$. Under Heuristic 1, those pairs then lead to the reconstruction of the secret monomial via the Compute-Monomial function with high probability. Note that for any $\mathbf{c} \in \mathcal{C}$ there

is exactly one $\mathbf{w} \in \mathcal{C}'$ such that $\mathbf{c} = \mathbf{wQ}$ for the searched monomial. Therefore we expect that there are

$$\frac{|L_1 \times L_2|}{N_w} = c$$

equivalent codewords between the two lists. For large enough c a birthday paradox argument gives that there are at least two pairs of this form with high probability. □

In the following we use Eq. (1) to express the complexity of Algorithm 4 as a function $2^{\vartheta(R,q,\omega)n}$, with $\vartheta(R, q, \omega)$ being a constant that depends on the code rate R, the chosen relative weight ω, where $w = \omega n$ as well as the field size q.

Corollary 1. *Let $n \in \mathbb{N}$, $k = Rn$, $w = \omega n$ for constants $\omega, R \in [0, 1]$ and q be a constant integer. Then Algorithm 4 runs in expected time $T = \tilde{\mathcal{O}}\left(2^{\vartheta n}\right)$ and memory $M = \tilde{\mathcal{O}}\left(2^{\mu n}\right)$, with*

$$\vartheta = \max\left(h(w) - (1-R)h\left(\frac{\omega}{1-R}\right) - \frac{\gamma}{2},\ \ 2\gamma\right) \quad \textit{and} \quad \mu = \frac{\gamma}{2},$$

where $\gamma = h(\omega) + \omega \log(q-1) - (1-R)\log q$.

Proof. By Theorem 1 the time complexity of Algorithm 4 is $\tilde{\mathcal{O}}\left(\max(T_{\text{Rec}}, T_{\text{ISD}})\right)$, where

$$T_{\text{Rec}} = (N_w)^2 \quad \text{and} \quad T_{\text{ISD}} = \frac{\binom{n}{w}}{\binom{n-k}{w}\sqrt{N_w}},$$

with $N_w = \tilde{\Theta}\left(\frac{\binom{n}{w}(q-1)^w}{q^{n-k}}\right)$. Approximating the involved binomial coefficients via Eq. (1), and dropping Landau notation for convenience, we obtain

$$\log T_{\text{Rec}} = 2\log N_w \quad \text{and} \quad \log T_{\text{ISD}} = \left(h(w) - (1-R)h\left(\frac{\omega}{1-R}\right)\right)n - \frac{\log N_w}{2},$$

where $\log N_w = \left(h(\omega) + \omega\log(q-1) - (1-R)\log q\right)n$. Now by observing that $\gamma = \log N_w$ and recalling that the memory complexity of the algorithm is given as $M = \tilde{\mathcal{O}}\left(\sqrt{N_w}\right)$ the statement follows. □

For a given rate R and field size q we choose ω to minimize the expression ϑ from Corollary 1. Numerical optimization reveals that, apart from some edge cases when R is very close to 0 or 1, the optimal ω usually balances both factors in the maximum, i.e., it satisfies

$$h(\omega) - (1-R)h\left(\frac{\omega}{1-R}\right) - \frac{\gamma}{2} = 2\gamma.$$

Comparison to Existing Techniques In the following we provide an asymptotic comparison of Algorithm 4 to existing techniques, including Leon's algorithm [Leo82] as well as the canonical forms meet-in-the-middle approach [CPS23]. Note that for Beullens' [Beu20] as well as for the BBPS algorithm[BBPS22] no asymptotic analysis has been provided. We therefore postpone the comparison against those algorithms to the concrete analysis in Section 4.2.

Canonical forms. The CF-MITM algorithm produces two lists of size $\sqrt{\binom{n}{k}}$ where each list element requires the computation of a systematic form for an $[n, k]$ linear code over $\mathbb{F}_q$. Its running time therefore summarizes as

$$T_{\texttt{CF}} = \mathcal{O}\left(k^2 \cdot n\sqrt{\binom{n}{k}}\right) = \tilde{\mathcal{O}}\left(2^{h(k/n)n/2}\right). \tag{17}$$

Leon's algorithm. LEON requires to compute all codewords of minimum weight in the two codes, which has complexity (compare to Lemma 1)

$$T_{\text{Leon}} = \tilde{\mathcal{O}}\left(\binom{n}{w}/\binom{n-k}{w}\right) = \tilde{\mathcal{O}}\left(2^{n\cdot h\left(\frac{w}{n}\right)-(n-k)h\left(\frac{w}{n-k}\right)}\right).$$

First observe that for a choice of $\omega = \omega_{\text{GV}} = h_q^{-1}(1-R)$ equal to the GV-bound, implying $N_w = \mathcal{O}(1)$, Theorem 1 yields a time complexity equal to the time complexity of Leon's algorithm. This implies that Algorithm 4 is always at least as efficient as Leon's algorithm and strictly outperforms it for any optimal $\omega > \omega_{\text{GV}}$.

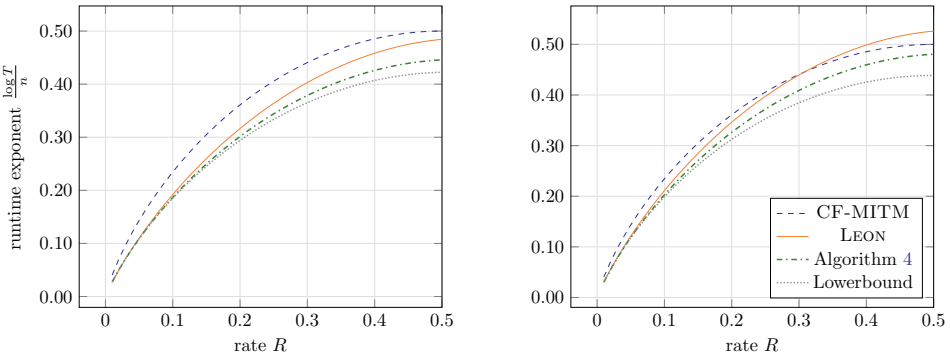
Comparison for fixed q. In Fig. 4 we compare the runtime exponent of Algorithm 4 against those of canonical forms as well as Leon's algorithm for fixed values of $q \in \{63, 127\}$. Recall that solving LCE with code rate R is equivalent to solving LCE with code rate $1-R$, by switching to the dual. We therefore report only the minimum exponent to solve either of those instances in the regime $R \in (0, 0.5]$. Additionally, we also include a known lower bound for low-weight codeword based approaches assuming (1) the secret monomial can be reconstructed from a single pair of equivalent codewords in negligible time and (2) this pair of codewords could be found in time linear in $|L_1| = \mathcal{O}\left(\sqrt{N_w}\right)$, leading to time complexity

$$T_{\text{Lower}} = \tilde{\mathcal{O}}\left(\max\left(\frac{\binom{n}{w}}{\binom{n-k}{w}\sqrt{N_w}}, \sqrt{N_w}\right)\right).$$

The runtime exponent of this lowerbound is illustrated in Fig. 4 as a dotted line.

We observe that across all code rates, Algorithm 4 achieves the smallest runtime exponent among any constructive algorithm. Moreover, for smaller values of q, the runtime exponent approaches the lower bound, whereas for larger q, it begins to diverge. This divergence is related to the fast growth of N_w as q increases, since the difference between the lower bound and the complexity given in Theorem 1 is a factor of $(N_w)^{3/2}$ in the second term of the maximum.

Comparison in the (q, R) space. In general, the time complexity of Algorithm 4 increases with q, while the runtime of the canonical forms approach remains independent of the field size. This suggests that, for sufficiently large q, the canonical forms method eventually outperforms Algorithm 4. However, we find that the break-even point with respect to Algorithm 4 requires $q > 200$ for all rates, while, in the case of Leon's algorithm, the break-even point is already reached at $q = 81$. Furthermore,



(a) $q = 63$ (b) $q = 127$

Fig. 4: Runtime exponent of different algorithms as a function of the rate R for different choices of q.

the farther the rate lies from $1/2$, the higher the break-even point with respect to q. In Fig. 5, we provide an illustration of the (q, R) space where the colored regions correspond to the respective algorithm performing best for those parameters, i.e., the algorithm having the lowest runtime exponent. Note that Algorithm 4 obtains the lowest runtime exponent in the entire green area. The striped area corresponds to the regime, where Leon's algorithm is favorable to CF-MITM. Therefore, the regime in which low-weight codeword finding algorithms are favorable is significantly enlarged with the introduction of Algorithm 4.

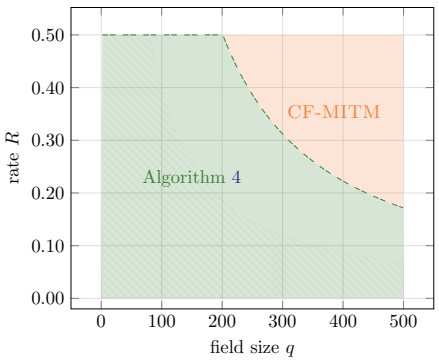


Fig. 5: Partition of the (q, R) space by optimal algorithms.

A further comparison of the lower bound for algorithms following a codeword-based approach against the time complexity $T_{\text{CF-MITM}}$ of CF-MITM shows that $T_{\text{Lower}} < T_{\text{CF-MITM}}$ for any (q, R). Moreover, we observe that T_{Lower} converges to $T_{\text{CF-MITM}}$ from below for $q \to \infty$. This is because for large q, we have that $w_{\text{GV}} \to n-k$, which fixes the optimal $w \in [w_{\text{GV}}, n-k]$ to $n-k$. In turn this implies

$N_w \to \binom{n}{n-k}$ and in summary leading to

$$T_{\text{Lower}} \to \tilde{\mathcal{O}}\left(\sqrt{\binom{n}{n-k}}\right) = T_{\text{CF-MITM}},$$

demonstrating the potential of codeword finding based approaches for arbitrary choices of q.

Comparison for the worst case rate. As can be observed from Fig. 4, the maximum runtime exponent for all algorithms is obtained at rate $R = 0.5$. For this reason cryptographic constructions, including the LESS signature scheme, usually resort to this choice for the code rate. To quantify the improvement for this specific choice, we compare in Fig. 6 the runtime exponent of the different approaches for fixed rate $R = 0.5$ as a function of q. As observed earlier, the runtime exponent of CF-MITM is constant in q. The intersection of the runtime exponents of Leon and Algorithm 4 with this constant lies at $q = 81$ and $q = 202$, respectively. Therefore at rate $\frac{1}{2}$ Algorithm 4 improves on the asymptotic complexity of previous algorithms for any $q \leq 202$.

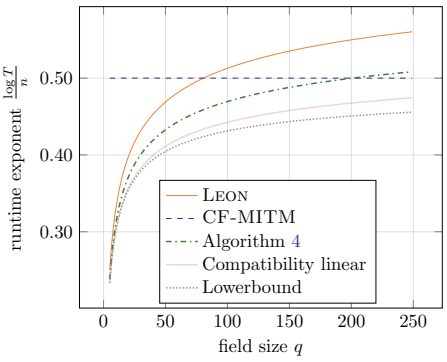


Fig. 6: Runtime exponents for fixed rate $R = \frac{1}{2}$ as a function of the field size q.

In Fig. 6 we again illustrate the lower bound for codeword finding based algorithms. Additionally, we provide a semi-lower bound (pink solid line), which refers to the running time under the assumption that the Compatible-Pairs function in Algorithm 4 runs in time linear in $|V| = |W| = N_w$, rather than quadratic time. In the following section, we show how to obtain a concrete algorithm whose complexity for growing choices of q converges to this optimistic bound.

4.2 An Improved Concrete Version of the Algorithm

We now introduce a series of improvements to Algorithm 4 that significantly reduce its concrete time complexity. In the corresponding analysis, we use the number of field operations as a measure for time complexity and the number of field elements that need to be stored as a memory unit.

Identifying Compatible Codewords In the following we derive a criterion to determine efficiently if two pairs of codewords can be mapped via a monomial transformation. Eventually, this criterion leads to a more efficient instantiation of the Compute-Monomial function in Algorithm 4. More precisely, given two pairs of codewords $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{C}$ and $\mathbf{w}_1, \mathbf{w}_2 \in \mathcal{C}'$, we say the pairs $(\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \mathbf{w}_2)$ are *compatible* if there exist a monomial transformation $\mathbf{Q}'$ such that $\mathbf{v}_1 = \mathbf{w}_1\mathbf{Q}'$ and $\mathbf{v}_2 = \mathbf{w}_2\mathbf{Q}'$, for some $\mathbf{Q}' \in \mathsf{Mono}_n(\mathbb{F}_q)$. Note that the difference to equivalent pairs (see Definition 2) is the fact that such a monomial $\mathbf{Q}'$ not necessarily defines a linear equivalence between $\mathcal{C}$ and $\mathcal{C}'$, i.e., it is not guaranteed to solve the code equivalence problem. However, any pair of equivalent codewords must necessarily be compatible according to this definition. We summarize this property in the following.

Definition 4 (Compatible Pairs). *Let $\mathcal{C}, \mathcal{C}'$ be two $[n, k]$ codes over $\mathbb{F}_q$. We call two pairs of codewords $(\mathbf{v}_1, \mathbf{w}_1)$, $(\mathbf{v}_2, \mathbf{w}_2)$ with $\mathbf{v}_i \in \mathcal{C}$ and $\mathbf{w}_i \in \mathcal{C}'$* compatible *if and only if there exists a monomial $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ such that*

$$\mathbf{w}_1 = \mathbf{v}_1\mathbf{Q} \quad \textit{and} \quad \mathbf{w}_2 = \mathbf{v}_2\mathbf{Q}.$$

We now derive a simple criterion to check the compatibility of pairs based on the argumentation about the restriction on the monomial imposed by the support intersection of the pairs discussed in Section 3.2. In that section, we already observed that a suitable monomial has to map indices in I to indices in J, where $I = \mathrm{Supp}(\mathbf{v}_1) \cap \mathrm{Supp}(\mathbf{v}_2)$ and $J = \mathrm{Supp}(\mathbf{w}_1) \cap \mathrm{Supp}(\mathbf{w}_2)$ are the respective support intersections. Therefore compatibility implies that for every $i \in I$ there exists a $j \in J$ satisfying (compare to Eq. (6))

$$\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)} = \frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}.$$

This translates into a simple criterion to check compatibility of pairs, which we summarize in the following lemma.

Lemma 6 (Compatibility Criterion). *Two pairs of codewords $(\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \mathbf{w}_2)$ of same Hamming weight $wt(\mathbf{v}_i) = wt(\mathbf{w}_i) = w$ are compatible if and only if the entries of $\mathbf{v}_{\mathrm{comp}}$ and $\mathbf{w}_{\mathrm{comp}}$ generate the same multiset, where*

$$\mathbf{v}_{\mathrm{comp}} := \left(\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)}\right)_{i \in I} \quad \textit{and} \quad \mathbf{w}_{\mathrm{comp}} := \left(\frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}\right)_{j \in J}$$

with $I = \mathrm{Supp}(\mathbf{v}_1) \cap \mathrm{Supp}(\mathbf{v}_2)$ and $J = \mathrm{Supp}(\mathbf{w}_1) \cap \mathrm{Supp}(\mathbf{w}_2)$.

Proof. Both vectors $\mathbf{v}_{\mathrm{comp}}$ and $\mathbf{w}_{\mathrm{comp}}$ sharing the same multiset of entries implies Eq. (6). This yields for $\mathbf{Q}(i, j) = \alpha_{i,j} = \mathbf{v}(i)/\mathbf{w}(j)$ a valid monomial transformation with respect to the intersection of the supports of those pairs. Note that for indices $i \in \mathrm{Supp}(\mathbf{v}_1) \setminus I$ a valid monomial can be obtained by mapping i to an arbitrary index $j \in \mathrm{Supp}(\mathbf{w}_1) \setminus J$ and setting the scaling factor to $\mathbf{v}_1(i)/\mathbf{w}_1(j)$ (analogously for $\mathbf{v}_2$ and $\mathbf{w}_2$). Lastly, indices where neither of the codewords have support can be mapped arbitrarily among each other.

On the other hand the existence of a monomial map $\mathbf{Q}$ such that $\mathbf{w}_1 = \mathbf{v}_1\mathbf{Q}$ and $\mathbf{w}_2 = \mathbf{v}_2\mathbf{Q}$ implies Eq. (6) which yields that $\mathbf{v}_{\mathrm{comp}}$ and $\mathbf{w}_{\mathrm{comp}}$ share the same multiset of entries. □

An algorithm finding compatible pairs. Lemma 6 now easily turns into a more efficient algorithm for the Compatible-Pairs function in Algorithm 4, which returns only those elements whose labels $\mathbf{v}_{\text{comp}}, \mathbf{w}_{\text{comp}}$ share the same multiset of their entries. More precisely, for each element $(\mathbf{v}_1, \mathbf{v}_2) \in V$ (resp. $(\mathbf{w}_1, \mathbf{w}_2) \in W$) we compute the corresponding labels $\mathbf{v}_{\text{comp}}$ (resp. $\mathbf{w}_{\text{comp}}$). Finally we compute elements between the lists satisfying equality with respect to the multiset of the labels via a sort-and-match procedure. We summarize this procedure in Algorithm 5.

Algorithm 5 Compatible-Pairs(V, W)

Input: lists $V, W \subset \mathbb{F}_q^n \times \mathbb{F}_q^n$ with $V = \{(\mathbf{v}_1, \mathbf{v}_2) \mid \mathrm{wt}(\mathbf{v}_1) = \mathrm{wt}(\mathbf{v}_2) = w\}$ and $W = \{(\mathbf{w}_1, \mathbf{w}_2) \mid \mathrm{wt}(\mathbf{w}_1) = \mathrm{wt}(\mathbf{w}_2) = w\}$
Output: list $Z = \{\big((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)\big) \mid \mathbf{v}_i \in V, \mathbf{w}_i \in W \wedge (\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \mathbf{w}_2)$ are compatible $\}$

1: $V' \leftarrow \{(\mathsf{sort}(\mathbf{v}_{\text{comp}}), (\mathbf{v}_1, \mathbf{v}_2)) \in (\mathbb{F}_q^*)^\ell \times V \mid \ell \geq w^2/n\}$
2: $W' \leftarrow \{(\mathsf{sort}(\mathbf{w}_{\text{comp}}), (\mathbf{w}_1, \mathbf{w}_2)) \in (\mathbb{F}_q^*)^\ell \times W \mid \ell \geq w^2/n\}$
3: $Z \leftarrow \{\big((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)\big) \mid (\mathbf{x}, \mathbf{v}_1, \mathbf{v}_2) \in V' \wedge (\mathbf{x}, \mathbf{w}_1, \mathbf{w}_2) \in W'\}$
4: **Return** Z

Lemma 7 (Finding Compatible Pairs). *Algorithm 5 runs in expected time and memory*

$$(\ell + 2n)\big(|V| + |W|\big) + 4n \cdot |V \times W| \cdot \frac{(w^2/n)!}{(q-1)^{w^2/n}},$$

and returns any compatible pair between V and W with probability at least $1/2$.

Proof. A sort-and-match procedure between two lists can be performed in time linear in the involved list sizes. It remains to determine the expected size of Z. Note that two random vectors $\mathbf{v}_{\text{comp}} \in (\mathbb{F}_q^*)^\ell, \mathbf{w}_{\text{comp}} \in (\mathbb{F}_q^*)^\ell$ are equal with probability $(q-1)^{-\ell}$. The algorithm, however matches equality on the sorted vectors $\mathsf{sort}(\mathbf{v}_{\text{comp}}), \mathsf{sort}(\mathbf{w}_{\text{comp}})$. Therefore labels match as long as $\mathbf{w}_{\text{comp}}$ forms any permutation of $\mathbf{v}_{\text{comp}}$, which gives

$$\begin{aligned} p :&= \Pr\left[\mathsf{sort}(\mathbf{v}_{\text{comp}}) = \mathsf{sort}(\mathbf{w}_{\text{comp}}) \mid \mathbf{v}_{\text{comp}}, \mathbf{w}_{\text{comp}} \in (\mathbb{F}_q^*)^\ell\right] \\ &= \frac{|\{\pi(\mathbf{v}_{\text{comp}}) \mid \pi \in \mathsf{Perm}_\ell\}|}{(q-1)^\ell} \leq \frac{|\{\pi(\mathbf{v}_{\text{comp}}) \mid \pi \in \mathsf{Perm}_{w^2/n}\}|}{(q-1)^{w^2/n}} \\ &\leq \frac{(w^2/n)!}{(q-1)^{w^2/n}}, \end{aligned} \tag{18}$$

where the first inequality follows from the fact that p is generally decreasing in $\ell \geq w^2/n$, while the second inequality follows from bounding the amount of permutations of $\mathbf{v}_{\text{comp}} \in (\mathbb{F}_q^*)^\ell$ as $\ell!$. Eventually, this yields the expected size of Z as $\mathbb{E}\big[|Z|\big] = p|V \times W|$. Note that the construction of list elements requires an amount of field operations bounded by the length of the included vectors, which is $\ell + 2n$

for V', W' and $4n$ for Z. Analogously the same amount of field elements need to be stored.

For the correctness note that $\mathsf{sort}(\mathbf{v}_{\text{comp}}) = \mathsf{sort}(\mathbf{w}_{\text{comp}})$ implies that the entries of $\mathbf{v}_{\text{comp}}$ and $\mathbf{w}_{\text{comp}}$ form the same multiset. Therefore Lemma 6 yields that those elements are compatible. It remains to analyze the probability that $\mathbf{v}_{\text{comp}}$ (resp. $\mathbf{w}_{\text{comp}}$) is of length $\ell \geq w^2 n$. Note that ℓ corresponds to the size of the intersection of support between $(\mathbf{v}_1, \mathbf{v}_2) \in V$, i.e., $\ell = \text{Supp}(\mathbf{v}_1) \cap \text{Supp}(\mathbf{v}_2)$. For random $\mathbf{v}_1, \mathbf{v}_2$, ℓ is distributed according to a hypergeometric distribution with population size n and parameters (w, w). For large n this distribution is closely approximated by a binomial distribution $\text{Bin}_{w,w/n}$, with mean w^2/n. For $X \sim \text{Bin}_{w,w/n}$ we have

$$\Pr\left[X \geq E[X] = w^2/n\right] \geq 1/2,$$

since the binomial distribution is symmetric around its mean. Since for compatible pairs the length of $\mathbf{v}_{\text{comp}}$ and $\mathbf{w}_{\text{comp}}$ is the same by definition, the probability for any compatible pair being returned by the algorithm is at least $1/2$. □

In the following we replace the routine Compatible-Pairs in Algorithm 4 with Algorithm 5, which leads to significant improvements with respect to concrete parameter choices. However, we first show that under the assumptions made in the asymptotic analysis, the procedure Compatible-Pairs from Eq. (16) is (up to polynomial factors) optimal with respect to the definition of compatibility from Definition 4. Precisely, the following lemma shows that for constant q computing all compatible pairs takes time at least $|V \times W|/\text{poly}(\ell)$.

Lemma 8 (Asymptotic Size of Z). *Let $n \in \mathbb{N}$. For any constant q the expected size of Z in Algorithm 4 is $\mathbb{E}\big[|Z|\big] = \frac{|V \times W|}{\text{poly}(\ell)}$.*

Proof. In Eq. (18) we observed that the probability of two pairs of random weight w vectors with same overlapping support size ℓ being compatible is

$$p = \frac{|\{\pi(\mathbf{v}_{\text{comp}}) \mid \pi \in \mathsf{Perm}_\ell\}|}{(q-1)^\ell} = \frac{\binom{\ell}{m_1, m_2, \ldots, m_{q-1}}}{(q-1)^\ell},$$

where m_i denotes the amount of appearances of $i \in \mathbb{F}_q^*$ in the vector $\mathbf{v}_{\text{comp}}$. The probability that each element appears an amount equal to its expectation $\ell/(q-1) > 1$ in $\mathbf{v}_{\text{comp}}$ is

$$\begin{aligned} p_2 := \Pr\left[m_i = \frac{\ell}{(q-1)}\right] &= \frac{\binom{\ell}{\frac{\ell}{(q-1)}, \frac{\ell}{(q-1)}, \ldots, \frac{\ell}{(q-1)}}}{(q-1)^\ell} = \frac{\ell!}{\left(\frac{\ell}{q-1}!\right)^{q-1} (q-1)^\ell} \\ &= \Theta(\ell^{(1-q)/2}) = \frac{1}{\text{poly}(\ell)}, \end{aligned}$$

which follows from approximating the factorials via Stirling's formula and the fact that q is a constant. Here we ignore rounding issues stemming from $\ell/(q-1)$ not being an integer for clarity.

Note that now the probability of the event E that two random pairs of vectors are compatible can be written as

$$p \geq \Pr\left[E \mid m_i = \frac{\ell}{q-1}\right] \cdot \Pr\left[m_i = \frac{\ell}{q-1}\right] = (p_2)^2 = \frac{1}{\mathrm{poly}(\ell)}.$$

Since the expected size of Z is $\mathbb{E}[Z] = p \cdot |V \times W|$ the statement follows. □

Exploiting Scalar Multiples Note that there exist $q-1$ solutions to the LCE problem, as any scaled version $\beta \cdot \mathbf{Q}$ of the secret monomial $\mathbf{Q}$ for any $\beta \in \mathbb{F}_q^*$ defines a linear equivalence between the two codes. Implicitly, for any equivalent pair $(\mathbf{v}_1, \mathbf{w}_1)$, equivalent under secret monomial $\mathbf{Q}$, the pair $(\mathbf{v}_1, \beta \cdot \mathbf{w}_1)$ is equivalent under secret monomial $\mathbf{Q}' = \beta\mathbf{Q}$.

In Section 4.1 we chose a list size $N = |L_1| = c\sqrt{N_w}$ in Algorithm 4 that ensures that between the lists V and W there are at least two pairs of codewords equivalent under the same secret monomial $\beta\mathbf{Q}$. In following we adapt this choice to still guarantee that there are two pairs of equivalent codewords between those lists, but not necessarily equivalent under the same scaled version of the secret monomial. In turn this allows us to start with smaller list sizes, while we have to adapt Algorithm 5 to identify pairs of codewords that are compatible under scaled versions of the same monomial transformation.

New starting list size. Let us start determining the necessary list size such that on expectation there are two pairs of codewords $(\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \mathbf{w}_2)$, $\mathbf{v}_i \in L_1$, $\mathbf{w}_i \in L_2$, that are equivalent under (potentially) different scaled versions of the secret monomial, i.e.,

$$\mathbf{w}_1 = \mathbf{v}_1(\gamma\mathbf{Q}) \quad \text{and } \mathbf{w}_2 = \mathbf{v}_2(\beta\mathbf{Q}), \quad \gamma, \beta \in \mathbb{F}_q^*. \tag{19}$$

In the following, we assume, without loss of generality, that $\gamma = 1$.[3] As observed, for every codeword $\mathbf{v} \in \mathcal{C}$, there are $q-1$ codewords equivalent under scaled versions of the secret monomial in $\mathcal{C}'$. Thus, the probability that a random pair $(\mathbf{v}, \mathbf{w}) \in \mathcal{C} \times \mathcal{C}'$ is equivalent under any of those scaled versions is $\frac{q-1}{N_w}$. Therefore, choosing

$$N = \sqrt{\frac{2N_w}{q-1}} \tag{20}$$

yields an expected amount of $\frac{q-1}{N_w} \cdot |L_1 \times L_2| = \frac{(q-1)N^2}{N_w} = 2$ pairs equivalent under (potentially) different scaled versions of the secret monomial. This corresponds to a saving of a factor of about $\sqrt{q-1}$ compared to the choice made in Theorem 1.

[3] Note that this is equivalent to redefining $\gamma\mathbf{Q}$ to be *the* secret monomial and updating the scaling factor β to β/γ.

Compatibility up to scalar factors. Note that given a pair of codewords $(\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \mathbf{w}_2)$ satisfying Eq. (19) (with $\gamma = 1$) for an arbitrary monomial $\mathbf{Q}$, we can still follow the same logic used to derive Eq. (6). Therefore, let $I = \mathrm{Supp}(\mathbf{v}_1) \cap \mathrm{Supp}(\mathbf{v}_2)$ and $J = \mathrm{Supp}(\mathbf{w}_1) \cap \mathrm{Supp}(\mathbf{w}_2)$. The two monomials still map elements from I to J, and moreover for any $i \in I$ there must be a $j \in J$ satisfying

$$\mathbf{v}_1(i) = \alpha_{i,j} \cdot \mathbf{w}_1(j) \quad \text{and} \quad \mathbf{v}_2(i) = \alpha_{i,j} \cdot \beta \cdot \mathbf{w}_2(j) \quad \text{for some } \alpha_{i,j}, \beta \in \mathbb{F}_q^*,$$

where $\alpha_{i,j} = \mathbf{Q}(i,j)$, which implies $\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)} = \beta^{-1} \cdot \frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}$. From here, following the logic of Lemma 6, we can deduce that, whenever the entries of the two labels $\mathbf{v}_{\mathrm{comp}}, \mathbf{w}_{\mathrm{comp}}$ generate the same multiset up to a multiplicative shift δ, there exists a monomial $\mathbf{Q}$ such that Eq. (19) is satisfied for $\beta = \delta^{-1}$. For finding those pairs for which the multiset of $\mathbf{v}_{\mathrm{comp}}$ and $\mathbf{w}_{\mathrm{comp}}$ differs by a shift δ we perform a standard meet-in-the-middle procedure.

Meeting the scalar in the middle. Assume we have two labels $\mathbf{v}_{\mathrm{comp}}, \mathbf{w}_{\mathrm{comp}}$ with $\mathsf{sort}(\mathbf{v}_{\mathrm{comp}}) = \delta \cdot \mathsf{sort}(\mathbf{w}_{\mathrm{comp}})$, $\delta \in \mathbb{F}_q$, i.e., the multiset of their entries differs by a factor of δ. Now we write $\delta = \delta_1 \cdot \delta_2, \delta_i \in S_i \subset \mathbb{F}_q$ and obtain

$$(\delta_1)^{-1} \cdot \mathsf{sort}(\mathbf{v}_{\mathrm{comp}}) = \delta_2 \cdot \mathsf{sort}(\mathbf{w}_{\mathrm{comp}}).$$

We then modify Algorithm 5 to include for every element V' a total of $|S_1|$ different labels $(\delta_1)^{-1} \cdot \mathsf{sort}(\mathbf{v}_{\mathrm{comp}})$ and analogously $|S_2|$ different labels $\delta_2 \cdot \mathsf{sort}(\mathbf{w}_{\mathrm{comp}})$ for each element in W'. This procedure is described in Algorithm 6.

Algorithm 6 Meet-Compatible-Pairs(V, W)

Input: lists $V, W \subset \mathbb{F}_q^n \times \mathbb{F}_q^n$ with $V = \{(\mathbf{v}_1, \mathbf{v}_2) \mid \mathrm{wt}(\mathbf{v}_1) = \mathrm{wt}(\mathbf{v}_2) = w\}$ and $W = \{(\mathbf{w}_1, \mathbf{w}_2) \mid \mathrm{wt}(\mathbf{w}_1) = \mathrm{wt}(\mathbf{w}_2) = w\}$

Output: list $Z = \{\big((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \beta\mathbf{w}_2)\big) \mid \mathbf{v}_i \in V, \mathbf{w}_i \in W \wedge \beta \in \mathbb{F}_q^* \wedge (\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \beta\mathbf{w}_2)$ are compatible $\}$

1: Let $m = \lceil \sqrt{q-1} \rceil$ and $\mathbb{F}_q^* = \langle g \rangle$
2: Let $S_1 = \{g^i \mid 0 \le i \le m\}$ and $S_2 = \{g^{i \cdot m} \mid 0 \le i \le m\}$
3: $V' \leftarrow \{((\delta_1)^{-1} \cdot \mathsf{sort}(\mathbf{v}_{\mathrm{comp}}), (\mathbf{v}_1, \mathbf{v}_2), \delta_1) \in (\mathbb{F}_q^*)^\ell \times V \times S_1 \mid \ell \ge w^2/n\}$
4: $W' \leftarrow \{(\delta_2 \cdot \mathsf{sort}(\mathbf{w}_{\mathrm{comp}}), (\mathbf{w}_1, \mathbf{w}_2), \delta_2) \in (\mathbb{F}_q^*)^\ell \times W \times S_2 \mid \ell \ge w^2/n\}$
5: $Z \leftarrow \{\big((\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2')\big) \mid (\mathbf{x}, (\mathbf{v}_1, \mathbf{v}_2), \delta_1) \in V', (\mathbf{x}, (\mathbf{w}_1, \mathbf{w}_2), \delta_2) \in W'\}$, with $\mathbf{w}_2' := (\delta_1\delta_2)^{-1} \cdot \mathbf{w}_2$
6: **Return** Z

Lemma 9 (Complexity of Algorithm 6). *Algorithm 6 runs in expected time and memory*

$$\sqrt{q-1} \cdot (\ell + 2n)\big(|V| + |W|\big) + 4n \cdot |V \times W| \cdot \frac{(w^2/n)!}{(q-1)^{w^2/n-1}},$$

and returns any pair between V and W satisfying Eq. (19) for an arbitrary monomial $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ with probability at least $1/2$.

Proof. The statement about the time and memory complexity follows along the lines of the proof of Lemma 7 by observing that the lists V' and W' are now of size $\lceil\sqrt{q-1}\rceil \cdot |V|$ and $\lceil\sqrt{q-1}\rceil \cdot |W|$ respectively.

The correctness follows from the argumentation above observing that any $\delta \in \mathbb{F}_q^*$ can be factored into $\delta = \delta_1\delta_2$ with $\delta_i \in S_i$. □

The Concrete Algorithm Eventually, we are ready to specify the improved concrete version of Algorithm 4. Precisely for this version, we use a starting list size N as specified in Eq. (20) and instantiate the Compute-Monomial function in Algorithm 4 via Algorithm 6.

We specify the concrete complexity of this algorithm in the following theorem.

Theorem 2 (Concrete Complexity of Algorithm 4). *Let $n, q, k, w \in \mathbb{N}$, $w < n-k$, $q \geq 23$. Then the* LCE *problem under Heuristic 1 can be solved in expected time*

$$4T_{\mathrm{ISD}}^{n,k,q,w} \cdot \sqrt{\frac{2N_w}{q-1}} + \frac{4N_w(\ell+2n)}{\sqrt{q-1}} + v(n,w,w^2/n,q)^4 \cdot \frac{8(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}$$

and expected memory $M_{\mathrm{ISD}}^{n,k,q,w} + \frac{2N_w(\ell+2n)}{\sqrt{q-1}} + \frac{4(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}$, with constant probability, where $T_{\mathrm{ISD}}^{n,k,q,w}$ (resp. $M_{\mathrm{ISD}}^{n,k,q,w}$) is the expected time (resp. memory) complexity for finding a weight-w codeword in an $[n,k]$ linear code over $\mathbb{F}_q$ and $v(n,w,\ell,q)$ as defined in Eq. (8).

Proof. For the correctness we have shown that the chosen list size leads on expectation to two pairs of codewords satisfying Eq. (19) for the secret monomial $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$. These pairs are returned in the list Z with constant probability $\frac{1}{2}$ according to Lemma 9. Now, Heuristic 1 ensures that from this pair the secret monomial is reconstructed via the Compute-Monomial function with high probability.

The time complexity of Algorithm 4 splits in the time to construct two lists of $N = \sqrt{\frac{2N_w}{q-1}}$ weight-w codewords, the call to the Compute-Monomial function and the final reconstruction of the monomial for each element in Z.

Note that the time to find $2N$ weight-w codewords is defined as $T_{\mathrm{ISD}}^{n,k,q,w} \cdot 2N = T_{\mathrm{ISD}}^{n,k,q,w} \cdot \sqrt{\frac{8N_w}{q-1}}$. Now, Lemma 9 for $|V| = |W| = N^2$ yields a running time of

$$\frac{4N_w(\ell+2n)}{\sqrt{q-1}} + \frac{16n(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}},$$

where the size of the returned list is $|Z| = \frac{4(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}$. Eventually, attempting to reconstruct the monomial for each element in this list has time complexity

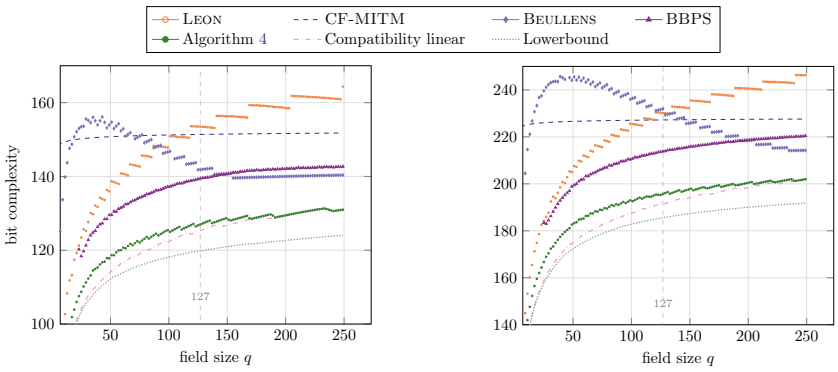


Fig. 7: Bitcomplexity of LCE for $n = 252$ (left) and $n = 400$ (right) with rate 0.5.

$v(n, w, w^2/n, q)^4 \cdot |Z|$ (compare to Lemma 4), as Algorithm 6 enforces $\ell \geq w^2/n$. Now the expected running time follows by observing that on expectation two applications of the procedure lead to the equivalent pair being returned by Algorithm 6. The memory complexity is derived equally by observing that the executions of the ISD algorithm are sequential and memory can be reused. □

Concrete Comparison We now compare the concrete performance of Algorithm 4 against previous approaches with respect to parameters suggested in the literature. For the estimation of the bitcomplexity of Leon, Beullens as well as BBPS we rely on the *CryptographicEstimators* library [EVZB24][4]. For the CF-MITM we use the formula provided in Eq. (17).

Application to LESS parameters. We first focus on the parameters used in the LESS signature scheme. The scheme provides three parameter sets for different security categories, which all use $q = 127$ and rate $\frac{1}{2}$. We refer to those as LESS-I ($n = 252$), LESS-III ($n = 400$) and LESS-V ($n = 548$).

In Fig. 7, we compare the concrete time complexity of Algorithm 4 given in Theorem 2 against other known algorithms for varying q. In this comparison we fix $n = 252$ (left) and $n = 400$ (right), and the code rate to $\frac{1}{2}$, i.e., to the choices made for LESS-I and LESS-III. It can observed that for all choices of q Algorithm 4 obtains the best time complexity. Furthermore we again observe the mild dependence of CF-MITM on the value q. As known, Beullens starts improving upon Leon only for large enough q. We also provide in those graphics the lower bound for low-weight codeword based approaches as well as the semi-lowerbound assuming the list Z returned by the Compute-Monomial function is smaller than the input list V, i.e. $|Z| \leq |V|$. We find that for growing q the running time of Algorithm 4 converges to this semi-lowerbound. This is related to the used compatibility criterion, for which the probability of two non-equivalent codewords being compatible decreases in q, implying decreasing size of Z. Additionally, we provide in Table 3 the bit complexity estimates of all different algorithms for LESS parameters. We observe

[4]available at https://github.com/Crypto-TII/CryptographicEstimators

	CF-MITM	Leon	Beullens	BBPS	Algorithm 4
LESS-I	151	153	142	139	**127**
LESS-III	227	230	232	214	**196**
LESS-V	302	307	324	290	**266**

Table 3: Bitcomplexity estimates to solve LCE for suggested LESS parameters.

		(n, k, q)	CF-MITM	Leon	Beullens	BBPS	Algorithm 4
PCE	[CPS23]	$(252, 126, 127)$	151	153	131	-	**127**
		$(400, 200, 127)$	227	230	-	-	**196**
		$(548, 274, 127)$	302	307	-	-	**266**
	[BBN+22]	$(235, 108, 251)$	142	154	151	-	**123**
		$(230, 115, 127)$	140	142	119	-	**117**
LCE	[BBN+22]	$(198, 94, 251)$	123	134	113	114	**105**

Table 4: Bitcomplexity estimates for suggested LCE and PCE instances.

a bit complexity reduction by 12 (LESS-I), 18 (LESS-III) and 24 (LESS-V) bits compared to the best previous approach, which is for all parameters the BBPS algorithm.

Application to other parameters. Next, we compare the performance of Algorithm 4 to previous approaches on other suggested parameters (see Table 4). The parameters split into those for the LCE problem and those considering the special case PCE, in which the monomial is known to be a permutation. Note that Algorithm 4 applies equally to both cases. However, for the comparison, note that the BBPS algorithm only applies in case of LCE. Note that, in case of the PCE instances, the necessary requirements of Beullens are not met in 2 out of 5 instances. Overall, we observe that across all parameters Algorithm 4 obtains the lowest bit complexity estimates, with reductions ranging from 2 to 36 bits.

References

BBB+25a. Marco Baldi, Alessandro Barenghi Luke Beckwith, Jean-François Biasse, Tung Chou, Andre Esser, Kris Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani O. Saarinen, Paolo Santini, Robert Wallace, and Floyd Zweydinger. LESS (version 2.0). Tech. rep., National Institute of Standards and Technology, 2025.

BBB+25b. Huck Bennett, Drisana Bhatia, Jean-François Biasse, Medha Durisheti, Lucas LaBuff, Vincenzo Pallozzi Lavorante, and Philip Waitkevich. Asymptotic improvements to provable algorithms for the code equivalence problem. Cryptology ePrint Archive, Paper 2025/187, 2025.

BBC+25. Michele Battagliola, Giacomo Borin, Giovanni Di Crescenzo, Alessio Meneghetti, and Edoardo Persichetti. Enhancing threshold group action signature schemes: Adaptive security and scalability improvements. Cryptology ePrint Archive, Paper 2025/085, 2025.

BBMP24. Michele Battagliola, Giacomo Borin, Alessio Meneghetti, and Edoardo Persichetti. Cutting the grass: Threshold group action signature schemes. In

Elisabeth Oswald, editor, *Topics in Cryptology – CT-RSA 2024*, pages 460–489, Cham, 2024. Springer Nature Switzerland.

BBN+22. Alessandro Barenghi, Jean-François Biasse, Tran Ngo, Edoardo Persichetti, and Paolo Santini. Advanced signature functionalities from the code equivalence problem. *International Journal of Computer Mathematics: Computer Systems Theory*, 7(2):112–128, 2022.

BBPS21. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. LESS-FM: Fine-tuning signatures from the code equivalence problem. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*, pages 23–43. Springer, Heidelberg, 2021.

BBPS22. Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. On the computational hardness of the code equivalence problem in cryptography. *Cryptology ePrint Archive*, 2022.

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

BCDD+25. Alessandro Budroni, Jesús-Javier Chi-Domínguez, Giuseppe D'Alconzo, Antonio J. Di Scala, and Mukul Kulkarni. Don't use it twice! solving relaxed linear equivalence problems. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024*, pages 35–65, Singapore, 2025. Springer Nature Singapore.

BEFN. Alessandro Budroni, Andre Esser, Ermes Franch, and Andrea Natale. Accompanying repository.

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*, volume 12804 of *LNCS*, pages 387–403. Springer, Heidelberg, October 2020.

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*, volume 12174 of *LNCS*, pages 45–65. Springer, Heidelberg, July 2020.

BN24. Alessandro Budroni and Andrea Natale. On the sample complexity of linear code equivalence for all code rates. Cryptology ePrint Archive, Paper 2024/1757, 2024.

CPS23. Tung Chou, Edoardo Persichetti, and Paolo Santini. On linear equivalence, canonical forms, and digital signatures. *Cryptology ePrint Archive*, 2023.

CVE11. Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 171–186. Springer, Heidelberg, August 2011.

DST19. Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 21–51. Springer, Heidelberg, December 2019.

EVZB24. Andre Esser, Javier Verbel, Floyd Zweydinger, and Emanuele Bellini. SoK: CryptographicEstimators–a software library for cryptographic hardness esti-

mation. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, pages 560–574, 2024.

FJR22. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022.

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

Nat23. National Institute of Standards and Technology. Post-quantum cryptography: Digital signature schemes. Round 2 Additional Signatures, 2023.

Now24. Julian Nowakowski. An improved algorithm for code equivalence. Cryptology ePrint Archive, Paper 2024/1272, 2024.

Pra62. Eugene Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

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, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 351–378. Springer, Heidelberg, December 2023.

Sae17. Mohamed Ahmed Saeed. *Algebraic Approach for Code Equivalence*. PhD thesis, Normandie Université, University of Khartoum,, 2017. Available at https://theses.hal.science/tel-01678829v2.

Ste94. Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, Heidelberg, August 1994.

The22. The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.5)*, 2022.

Vér97. Pascal Véron. Improved identification schemes based on error-correcting codes. *Applicable Algebra in Engineering, Communication and Computing*, 8:57–69, 1997.

A Expected Hints from Algorithm 1

In this section, we show that the expected number of hints from Lemma 3 can be roughly approximated as ℓ^2 for sufficiently large $q > \ell$. Define

$$s(\ell, q) := \sum_{t=2}^{\ell} t^2 (q-1) \binom{\ell}{t} \left(\frac{1}{q-1}\right)^t \left(\frac{q-2}{q-1}\right)^{\ell-t}, \tag{21}$$

to be the sum in Eq. (7). Let $c := \frac{\ell}{q-1}$, we then substitute $q - 1 = \frac{\ell}{c}$ into Equation (21) obtaining

$$s(\ell, q) = \frac{\ell}{c} \sum_{t=2}^{\ell} t^2 \binom{\ell}{t} \left(\frac{c}{\ell}\right)^t \left(1 - \frac{c}{\ell}\right)^{\ell-t}.$$

Note that the value of $s(\ell, q)$ is closely approximated by restricting to the first few entries of the sum. This is because, for larger values of t, we have

$$t^2\binom{\ell}{t}\left(\frac{c}{\ell}\right)^t\left(1-\frac{c}{\ell}\right)^{\ell-t} = t^2\binom{\ell}{t}\left(\frac{1}{q-1}\right)^t\left(1-\frac{1}{q-1}\right)^{\ell-t}$$
$$\leq t^2\binom{2t}{t}\left(\frac{1}{q-1}\right)^t \approx \frac{t^{3/2}}{\sqrt{\pi}}\left(\frac{4}{q-1}\right)^t,$$

which tends to zero for $q > 5$. We therefore truncate the sum to the first few values $t \in \{1, \ldots, \bar{t}\}$. For large ℓ, the term $\left(1-\frac{c}{\ell}\right)^{\ell-\bar{t}}$ can then be approximated to

$$\left(1-\frac{c}{\ell}\right)^{\ell} \approx e^{-c}$$

and moved out of the sum. We are left with

$$s(\ell, q) = \left(\frac{\ell}{c}e^{-c}\right)\sum_{t=2}^{\bar{t}} t^2\binom{\ell}{t}\left(\frac{c}{\ell}\right)^t.$$

For large ℓ and since $t \leq \bar{t} \ll \ell$ we can approximate the term

$$\frac{\binom{\ell}{t}}{\ell^t} = \frac{\ell(\ell-1)\cdots(\ell-t+1)}{t!\ell^t}$$

as $\frac{1}{t!}$, then the sum can be rewritten as

$$\sum_{t=2}^{\bar{t}}\frac{t^2c^t}{t!} = \sum_{t=2}^{\bar{t}}\frac{tc^t}{(t-1)!} = \sum_{t=2}^{\bar{t}}\frac{c^t}{(t-2)!} + \sum_{t=2}^{\bar{t}}\frac{c^t}{(t-1)!}.$$

Recall that $e^x = \sum_{i=0}^{\infty}\frac{x^i}{i!}$ and, moreover, the sum converges to e^x quite fast, hence, also already for $\sum_{i=0}^{\bar{t}}\frac{x^i}{i!}$. We therefore approximate the first sum as c^2e^c while the second can be written as $c\sum_{t=1}^{\bar{t}}\frac{c}{t!} \approx c(e^c-1)$. Putting all together we get the following approximation

$$s(\ell, q) \approx \left(\frac{\ell}{c}e^{-c}\right)(c^2e^c + ce^c - c) = \ell(c+1-e^{-c}). \tag{22}$$

While this approximation has been done assuming large ℓ and q, Fig. 8 shows that it is accurate also for relatively small values of these parameters.

Using the approximation in (22) we have $s(\ell, q) = \lambda_{\ell,q}\ell$ where $\lambda_{\ell,q} = (c+1-e^{-c})$ and $c = \frac{\ell}{q-1}$. While it is clear that $\lambda_{\ell,q}$ is close to 0 for $q \gg \ell$ it is still a relatively small coefficient for $q \approx \ell$. As an example for $\ell = q-1$ we obtain $\lambda_{\ell,q} = 2-e^{-1} \approx 1.63$, while for $\ell = (q-1)/2$ we already get $\lambda_{\ell,q} \approx 0.89$. For large values of ℓ with respect to q the coefficient $\lambda_{\ell,q}$ can be approximated to $\frac{\ell+q-1}{q-1}$.

In conclusion the number of expected hints in Eq. (7) is asymptotically close to ℓ^2 for $q \geq \ell$ while it can be estimated as $\ell^2\frac{q}{q-1}$ for $\ell > q$.

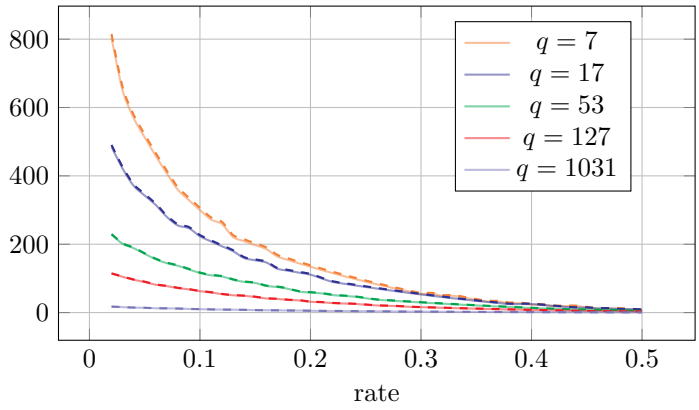


Fig. 8: The continuous lines represent the quantity $s(\ell, q)$ in (21) for various values of q while the dashed line is the approximation of the same quantity in (22). We consider $\ell = \frac{\omega_{GV}^2}{n}$, where ω_{GV} is the weight given by the Gilbert-Varshamov bound at that rate.

B On the Structure of $\mathbf{A}_{\mathsf{red}}$

In Section 3, we show how a linear code equivalence problem instance $(\mathcal{C}, \mathcal{C}' = \mathcal{C}\mathbf{Q})$ can be reduced to finding a particular solution of an underdetermined linear system when two pairs of equivalent codewords between those codes are known. In the following we provide further insights with respect to the structure of the constructed linear system, giving further evidence for the validity of Heuristic 1.

Let $\mathbf{G}$ be a generator matrix of $\mathcal{C}$ and $\mathbf{H}'$ be a parity-check matrix of $\mathcal{C}'$. Recall, that we consider the following linear system

$$\mathsf{S}: \quad \mathbf{A}\mathbf{x} = \mathbf{0}, \tag{3}$$

where $\mathbf{A} = \mathbf{G} \otimes \mathbf{H}'$ and $\mathbf{x}$ is the length-n^2 column-vector $\mathsf{vec}(\mathbf{Q})$ formed by the concatenation of the rows of $\mathbf{Q}$. We then assume knowledge of two pairs $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)$ of equivalent codewords of weight w, that is $\mathbf{w}_1 = \mathbf{v}_1\mathbf{Q}$ and $\mathbf{w}_2 = \mathbf{v}_2\mathbf{Q}$, with $I_i = \mathrm{Supp}(\mathbf{v}_i)$ and $J_i = \mathrm{Supp}(\mathbf{w}_i)$, $I_i, J_i \subset [n]$, $i = 1, 2$ being the supports of $\mathbf{v}_i$ and $\mathbf{w}_i$, respectively.

Let us divide the matrices $\mathbf{G}$ and $\mathbf{H}'$ in four parts according to the supports of $\mathbf{v}_1, \mathbf{v}_2, \mathbf{w}_1$, and $\mathbf{w}_2$, where $\bar{I} = [n] \setminus I$ indicates the complement of the set I in $[n]$.

$$\begin{aligned} &\mathbf{G}_{11} := \mathbf{G}_{I_1 \cap I_2},\ \mathbf{G}_{10} := \mathbf{G}_{I_1 \cap \bar{I}_2},\ \mathbf{G}_{01} := \mathbf{G}_{\bar{I}_1 \cap I_2},\ \mathbf{G}_{00} := \mathbf{G}_{\bar{I}_1 \cap \bar{I}_2} \\ &\mathbf{H}_{11} := \mathbf{H}_{J_1 \cap J_2},\ \mathbf{H}_{10} := \mathbf{H}_{J_1 \cap \bar{J}_2},\ \mathbf{H}_{01} := \mathbf{H}_{\bar{J}_1 \cap J_2},\ \mathbf{H}_{00} := \mathbf{H}_{\bar{J}_1 \cap \bar{J}_2} \end{aligned} \tag{23}$$

Recall that the method described in Section 3.1 (see Eq. (4)) sets to zero the variables $\mathbf{x}(i \cdot n + j)$ in Equation (3) if

- $i \in I_1 \cap I_2$ but $j \notin J_1 \cap J_2$, or

– $i \in I_1 \cap \bar{I}_2$ but $j \notin J_1 \cap \bar{J}_2$, or
– $i \in \bar{I}_1 \cap I_2$ but $j \notin \bar{J}_1 \cap J_2$, or
– $i \in \bar{I}_1 \cap \bar{I}_2$ but $j \notin \bar{J}_1 \cap \bar{J}_2$.

Note that, due to the tensor structure, the matrix describing the resulting linear system can be obtained from $\mathbf{A}$ by removing the columns corresponding to the above indices. In fact the matrix describing the reduced system then forms a column-rearrangement of the following matrix

$$\hat{\mathbf{A}} = \big[\underbrace{\mathbf{G}_{00} \otimes \mathbf{H}_{00}}_{(n-2w+\ell)^2}\ \underbrace{\mathbf{G}_{01} \otimes \mathbf{H}_{01}}_{(w-\ell)^2}\ \underbrace{\mathbf{G}_{10} \otimes \mathbf{H}_{10}}_{(w-\ell)^2}\ \underbrace{\mathbf{G}_{11} \otimes \mathbf{H}_{11}}_{\ell^2}\big].$$

The remaining unknown entries of $\mathbf{Q}$, after applying a similar rearrangement as in $\hat{\mathbf{A}}$, then form four monomial submatrices (see Figure 1b). Let us call these submatrices $\mathbf{Q}_{01}, \mathbf{Q}_{10} \in \mathsf{Mono}_{w-\ell}(\mathbb{F}_q)$, $\mathbf{Q}_{11} \in \mathsf{Mono}_{\ell}(\mathbb{F}_q)$ and $\mathbf{Q}_{00} \in \mathsf{Mono}_{n-2w+\ell}(\mathbb{F}_q)$. Note that by construction $\mathbf{Q}_{ij}$ maps indices from $I_1^{(i)} \cap I_2^{(j)}$ to $J_1^{(i)} \cap J_2^{(j)}$, where $I^{(i)} = I$ for $i = 1$ and $I^{(i)} = \bar{I}$ for $i = 0$. We then define the column vectors $\mathbf{x}_{00} = \mathsf{vec}(\mathbf{Q}_{00})$, $\mathbf{x}_{01} = \mathsf{vec}(\mathbf{Q}_{01})$, $\mathbf{x}_{10} = \mathsf{vec}(\mathbf{Q}_{10})$, and $\mathbf{x}_{11} = \mathsf{vec}(\mathbf{Q}_{11})$. Then we have that

$$\hat{\mathbf{A}}\mathbf{x} = (\mathbf{G}_{00} \otimes \mathbf{H}_{00})\mathbf{x}_{00} + (\mathbf{G}_{01} \otimes \mathbf{H}_{01})\mathbf{x}_{01} + (\mathbf{G}_{10} \otimes \mathbf{H}_{10})\mathbf{x}_{10} + (\mathbf{G}_{11} \otimes \mathbf{H}_{11})\mathbf{x}_{11}.$$

In Section 3.2, we explain how to determine some additional entries of $\mathbf{Q}$ corresponding to the submatrix $\mathbf{Q}_{11}$. We show in Appendix A that, for $c = \frac{\ell}{q-1}$, the expected number of variables in $\mathbf{x}_{11}$ that can be determined is approximated by $\ell^2 - \ell(c + 1 - e^{-c})$, which leaves us with an expected number of unknowns equal to

$$v(n, w, \ell, q) \approx 2(w-\ell)^2 + (n-2w+\ell)^2 + \ell(c+1-e^{-c}).$$

This shows that, for sufficiently large q, most of the ℓ^2 variables in $\mathbf{x}_{11}$ can be determined. We indicate by $\bar{\mathbf{x}}_{11}$ the vector of remaining variables in $\mathbf{x}_{11}$. The further reduced system is then defined via the matrix $\mathbf{A}_{\mathsf{red}}$ with solution $\bar{\mathbf{x}}^\top$ where $\bar{\mathbf{x}} = (\mathbf{x}_{00} \mid \mathbf{x}_{01} \mid \mathbf{x}_{10} \mid \bar{\mathbf{x}}_{11})$, and

$$\mathbf{A}_{\mathsf{red}} = (\mathbf{G}_{00} \otimes \mathbf{H}_{00} \mid \mathbf{G}_{01} \otimes \mathbf{H}_{01} \mid \mathbf{G}_{10} \otimes \mathbf{H}_{10} \mid \overline{\mathbf{G}_{11} \otimes \mathbf{H}_{11}}.) \tag{24}$$

Here, $\overline{\mathbf{G}_{11} \otimes \mathbf{H}_{11}}$ describes the matrix obtained by removing the columns from block $\mathbf{G}_{11} \otimes \mathbf{H}_{11}$ corresponding to the variables that got determined via Algorithm 1.

We now show the existence of an exponential amount codewords attaining an unusually low weight in the code generated by the matrix $\mathbf{A}_{\mathsf{red}}$. We then argue that this ultimately leads to the observed dimension reduction of the code if columns are removed from $\mathbf{A}_{\mathsf{red}}$, corresponding to the guesses made in Algorithm 3 to recover the secret monomial.

Proposition 1. *Suppose either* $\mathbf{H}_{01} \in \mathbb{F}_q^{(n-k)\times(w-\ell)}$ *or* $\mathbf{H}_{10} \in \mathbb{F}_q^{(n-k)\times(w-\ell)}$ *have maximal rank* $w-\ell$*. Then, the code* $\mathcal{C}_{\mathbf{A}_{\mathsf{red}}}$ *generated by the rows of* $\mathbf{A}_{\mathsf{red}}$ *in Eq. (24) contains an exponential amount of codewords of a weight smaller than or equal to* $w-\ell+t$*, where* $t \approx \ell(c+1-e^{-c})$ *is the number of columns in the block* $\overline{\mathbf{G}_{11} \otimes \mathbf{H}_{11}}$*.*

Proof. Let $\mathbf{H}_{01} \in \mathbb{F}_q^{(n-k)\times(w-\ell)}$ be of full rank $w-\ell$. The argumentation for $\mathbf{H}_{10}$ works analogously. Let $\mathbf{e}_j$ be the j-th unit vector of length $w-\ell$ in $\mathbb{F}_q$. Then we define

$$\mathcal{U}_j = \{\mathbf{u} \in \mathbb{F}_q^{n-k} \mid \mathbf{u}\mathbf{H}_{01} = \lambda\mathbf{e}_j, \lambda \in \mathbb{F}_q^*\}.$$

Let $\mathbf{u}$ be such that $\mathbf{u}\mathbf{H}_{01} = \mathbf{e}_j$, the set $\mathcal{U}_j$ can be explicitly constructed as the union of the cosets $\lambda\mathbf{u} + \mathsf{ker}(\mathbf{H}_{01})$ for $\lambda \in \mathbb{F}_q^*$ and has therefore cardinality $(q-1)q^{\dim(\mathsf{ker}(\mathbf{H}_{01}))} \approx q^{\ell+n-k-w+1}$.

Let $\mathbf{m}_2\mathbf{G} = \mathbf{v}_2$, for $\mathbf{v}_2$ being the first codeword of the second pair of equivalent codewords. Then, by the structure of $\mathbf{G}$ defined in Eq. (23), we have that $\mathbf{m}_2\mathbf{G}_{00} = \mathbf{0}$, $\mathbf{m}_2\mathbf{G}_{10} = \mathbf{0}$ and $\mathbf{m}_2\mathbf{G}_{01} = \mathbf{a}$ for some $\mathbf{a} \in \mathbb{F}_q^{w-\ell}$. Now, for any $\mathbf{u}_j \in \mathcal{U}_j$ consider the codeword

$$\mathbf{c} = (\mathbf{m}_2 \otimes \mathbf{u}_j)\mathbf{A}_{\mathsf{red}} = (\mathbf{c}_{00} \mid \mathbf{c}_{01} \mid \mathbf{c}_{10} \mid \bar{\mathbf{c}}_{11}),$$

where

$$\begin{aligned}
\mathbf{c}_{00} &= \mathbf{m}_2\mathbf{G}_{00} \otimes \mathbf{u}_j\mathbf{H}_{00} = \mathbf{0} \otimes \mathbf{u}_j\mathbf{H}_{00} = \mathbf{0} \\
\mathbf{c}_{01} &= \mathbf{m}_2\mathbf{G}_{01} \otimes \mathbf{u}_j\mathbf{H}_{01} = \mathbf{a} \otimes \lambda\mathbf{e}_j \\
\mathbf{c}_{10} &= \mathbf{m}_2\mathbf{G}_{10} \otimes \mathbf{u}_j\mathbf{H}_{10} = \mathbf{0} \otimes \mathbf{u}_j\mathbf{H}_{10} = \mathbf{0} \\
\bar{\mathbf{c}}_{11} &= (\mathbf{m}_2 \otimes \mathbf{u}_j)\overline{\mathbf{G}_{11} \otimes \mathbf{H}_{11}}.
\end{aligned}$$

The sub-vector $\mathbf{c}_{01} = \mathbf{a} \otimes \lambda\mathbf{e}_j \in \mathbb{F}_q^{(w-\ell)^2}$ has weight at most $w-\ell$ while the vector $\bar{\mathbf{c}}_{11} \in \mathbb{F}_q^t$ has weight at most t. Since we can choose $\mathbf{u}_j$ to be any element of $\mathcal{U}_j$ there are $(q-1)q^{n-k-w+\ell}$ ways to construct codewords with a support similar to $\mathbf{c}$. Note that a similar argumentation can be made in case $\mathbf{H}_{10}$ is of full rank by considering $\mathbf{v}_1$ in the consruction of $\mathbf{c}$ instead. □

We now argue that certain guesses on the remaining entries of the monomial matrix are likely to lead to a decreased dimension of the code $\mathcal{C}_{\mathbf{A}_{\mathsf{guess}}}$ in comparison to $\mathcal{C}_{\mathbf{A}_{\mathsf{red}}}$, where $\mathbf{A}_{\mathsf{guess}}$ is the matrix describing the system $\mathsf{S}_{(i,j)}$ from Eq. (13). Note that this dimension reduction is equivalent to the observed rank decrease from $\mathbf{A}_{\mathsf{red}}$ to $\mathbf{A}_{\mathsf{guess}}$ in Eq. (15).

We define the set $S_{01} := \bar{I}_1 \cap I_2 \times \bar{J}_1 \cap J_2$ indexing all coordinates of the submatrix $\mathbf{Q}_{01}$. Recall that in Algorithm 2, a guess on entry $(i,j) \in S_{01}$ corresponds to the deletion of the j-th column and the i-th row of $\mathbf{Q}_{01}$. This results in a particular puncturing of the code $\mathcal{C}_{\mathbf{A}_{\mathsf{red}}}$ over the set

$$I_{01}(i,j) := ((i,*) \cap S_{01}) \cup ((*,j) \cap S_{01}),$$

where $*$ stands for any index in $[n]$, resulting in the code $\mathcal{C}_{\mathbf{A}_{\mathsf{guess}}}$. Note that the support of the vector $\mathbf{c}_{01} = \mathbf{a} \otimes \lambda\mathbf{e}_j \in \mathbb{F}_q^{(w-\ell)^2}$ in the proof of Proposition 1 is contained in the set $I_{01}(i,j)$. This can be seen by noticing that the matrix $\mathsf{vec}^{-1}(\mathbf{c}_{01}) = \mathbf{C}_{01} \in \mathbb{F}_q^{(w-\ell)\times(w-\ell)}$ is a matrix whose columns are all zero with the exception of the j-th which is equal to $\mathbf{a}^\top$. Then, if we remove the j-th column and the i-th row from $\mathbf{C}_{01}$, we are left with a zero matrix, or analogously the zero vector when transitioning back via $\mathsf{vec}(\cdot)$

However, the support of the whole vector $\mathbf{c}$ also includes the non-null coordinates $\bar{I}_{11} := \mathrm{Supp}(\bar{\mathbf{c}}_{11})$ of $\bar{\mathbf{c}}_{11}$, which are not contained in $I_{01}(i,j)$. In Proposition 1 we prove that for any $\mathbf{u} \in \mathcal{U}_j$ we obtain codewords of the form

$$\mathbf{c} = (\mathbf{v} \otimes \mathbf{u})\mathbf{A}_{\mathsf{red}} = (\mathbf{0} \mid \mathbf{c}_{01} \mid \mathbf{0} \mid \bar{\mathbf{c}}_{11})$$

whose support is contained in $I_{01}(i,j) \cup \bar{I}_{11}$. Note that a substitution of $\mathbf{u}$ into $\mathbf{u} + \mathbf{z}$ where $\mathbf{z} \in \mathsf{ker}(H_{01})$ leads to a new codeword with same $\mathbf{c}_{01}$, but (potentially) different $\bar{\mathbf{c}}_{11}$. Now recall that we can construct a total of $(q-1)q^{n-k-w+\ell} \gg q^t$ such codewords. Therefore we expect that, at least for one of them, it holds $\bar{\mathbf{c}}_{11} = \mathbf{0} \in \mathbb{F}_q^t$, which leads to the entire support of $\mathbf{c}$ being contained in $I_{01}(i,j)$ and correspondingly leading to the mentioned dimension reduction. Note that a similar argument can be made for guesses on the entries of $\mathbf{Q}_{10}$ by instead considering the matrix $\mathbf{H}_{10}$ in the proof of Proposition 1.

This gives an intuitive explanation for why Algorithm 2 succeeds in identifying wrong guesses (i,j) for sufficiently large q with $(i,j) \in S_{01}$ or $(i,j) \in S_{10}$. However, experimentally, we observe that guesses with respect to the remaining variables in $\mathbf{Q}_{11}$ and $\mathbf{Q}_{00}$ also lead to a similar reduction in the resulting codes dimension. This implies the existence of codewords, of similarly low-weight to those constructed in Proposition 1, whose support lies entirely in the blocks corresponding to those indices. So far, we were unable to prove the existence of these codewords or to give an explicit construction. We therefore pose it as an open question to give intuitive reasoning or a rigorous proof for the success of the algorithm in those cases.