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] University of Bergen, Norway
ermes.franch@uib.no
[3] No affiliation
natalea00@gmail.com

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 algorithm that provably recovers the secret monomial from any *two* pairs of such codewords for fields of sufficiently large prime order q. Furthermore, we demonstrate that in practice this algorithm remains effective even for small prime fields. 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. Our concrete analysis reveals significant improvements as well. Most notably, in the context of the LESS signature scheme, a second-round contender in the ongoing NIST standardization effort, we achieve bit security reductions of up to 24 bits.

1 Introduction

Code-based cryptography remains a strong contender for the next post-quantum digital signature standard. Historically, most code-based signature schemes have based their security on the Syndrome Decoding Problem (SDP) [28,30,12,15,17]. 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

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

slower signing and verification. In recent years, a promising alternative emerged with the LESS signatures scheme [10], which is based on the *Linear Code Equivalence* (LCE) problem. The collective improvements established over the past years [3,21,13,7] have positioned LESS as a strong candidate [1] in the ongoing second round of the NIST standardization process for post-quantum digital signatures [19]. 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 [6,5], ring [2], and identity-based [2] 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 [18,9,4]. 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 arises 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 [9,4].[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, there usually exist exponentially many monomials $\mathbf{Q}'$ satisfying the relation $\mathbf{w} = \mathbf{v}\mathbf{Q}'$.

However, generally, the searched monomial is uniquely determined by the equivalence relationship $\mathcal{C}' = \mathcal{C}\mathbf{Q}$. By combining the code structure with the information provided by the pairs of equivalent codewords, we introduce a new algorithm that recovers the secret monomial from only *two* pairs of such codewords and prove it to run in polynomial-time whenever the field size is large enough. As a second main contribution, we then build on this result to overcome a central bottleneck in previous algorithms relying on finding low-weight codewords, resulting in the fastest known algorithm for solving LCE for most parameters. Most notably, the new algorithm reduces security levels for suggested parameter sets of the LESS signature scheme by up to 24 bits.

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 [18,9,4] and a recently introduced meet-in-the-middle technique relying on *canonical forms* [13]. The latter has attracted increased attention since its introduction, with recent results improving its success probability [20] as well as the range of applicable parameters [8]. Notably, the algorithm's

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

complexity is (almost) independent of the field size, obtaining a time complexity of roughly $2^{c(R)n}$, where c is a constant depending on the code rate $R = k/n$. For large choices of q, this algorithm becomes the fastest known algorithm to solve LCE. However, with respect to suggested parameters, algorithms relying on finding low-weight codewords remain yet superior.

An early algorithm by Leon [18] computes all low-weight codewords in both codes and recovers the secret monomial from the resulting sets provided that those sets are sufficiently large. Beullens [9] 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., codes generated by pairs of codewords with a small joint support. He then defines a criterion, called `Lex`, 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) [4] 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.

1.1 Our Contribution

In this work we combine both of the above approaches to design a faster algorithm for LCE. Therefore we first build on the concept of canonical forms to prove a fundamental result on the number of pairs of equivalent codewords required to reconstruct the secret monomial, after which we show how to leverage this new subroutine within the codeword-search framework. In the following we provide more technical details on these two contributions.

Recovering the secret monomial from two pairs of equivalent codewords We prove the following fundamental result on the recovery of the secret monomial of an LCE instance from two pairs of equivalent codewords

Theorem 1 (Informal). *From two pairs of equivalent codewords the secret monomial* $\mathbf{Q}$ *can be recovered in time*

$$\tilde{\mathcal{O}}\left(2^{\varepsilon/2} n^4\right), \quad \textit{where } \varepsilon = \mathcal{O}\left(\tfrac{n}{q} + \log q\right).$$

In order to prove Theorem 1 we build on a recent result by Chou, Persichetti and Santini [13] which recovers the secret monomial from two (information) sets $I, J \subset [n]$ with $|I| = |J| = k$ that are mapped to each other via the monomial action, i.e., $\pi(i) \in J$ for $i \in I$, where π is the permutation that defines the monomial action. In order to find two such *colliding* sets I, J, we show how to extract a subset sum problem instance from the support pattern of two equivalent codewords, whose solution corresponds to such colliding sets. From those sets we then recover $\mathbf{Q}$ in time $\tilde{\mathcal{O}}(n^4)$ via a recent variant of [13] due to Nowakowski

[20] that obtains high success probability already for small field sizes $q \geq 7$. The subset sum instance itself is defined via a (multi-) set of input integers S and target k, i.e., we are looking for a subset $S' \subseteq S$ such that $\sum_{s \in S'} s = k$. For the derived instance, it holds $|S| = \mathcal{O}(n)$ and $\sum_{s \in S} s = n$ which implies that the instance can be solved efficiently via dynamic programming (if a solution exist). In case no exact solution for target k but only for target $k - \varepsilon$ exists, we show how to extend this solution to the desired sets via an enumeration procedure that requires to test $\mathcal{O}\left(2^{\frac{\varepsilon}{2}}\right)$ candidates. We then establish an upper bound on the approximation quality ε that can be achieved, leading to Theorem 1.

Note that Theorem 1 proves the recovery of the secret monomial from two pairs of equivalent codewords to be polynomial-time whenever the field size is $q = \Omega\left(\frac{n}{\log n}\right)$. Note that for parameters suggested in the LESS NIST submission we have $q = 127$ and $n \in \{252, 400, 548\}$ and therefore those arguably already fall into the polynomial-time regime. However, we further quantify this by providing a full implementation and comprehensive experimental evidence.

Using our implementation we successfully recover the secret monomial from two pairs of equivalent codewords within minutes for all LESS parameter sets with empirical success probability one over thousands of experiments. Furthermore, we extend our experiments to larger code length up to $n = 2^{19}$ to study the asymptotic behavior of the algorithm. This confirms that the analysis around Theorem 1 is rather conservative. Namely, the required field size to enter the polynomial-time regime, i.e., the regime where an *exact* solution to the subset sum instance for target k exists ($\varepsilon = 0$) and, hence, enumeration is avoided, seems to be only $q = \Omega(\log n)$, maximizing as $q = 41$ for $n = 2^{19}$ in our experiments.

A faster algorithm solving LCE As a second contribution, we build on Theorem 1 to design a faster algorithm for solving LCE. Conceptually, the algorithm follows the standard paradigm of low-weight codeword search: we construct two lists L_1 and L_2 of fixed weight codewords from $\mathcal{C}$ and $\mathcal{C}'$, with list sizes chosen to guarantee the existence of at least two pairs of equivalent codewords between them. The algorithm then constructs all pairs of codewords from both lists, i.e., $V = L_1 \times L_1$ and $W = L_2 \times L_2$. In the last step the algorithm searches for potentially equivalent pairs between V and W and initiates the re-construction of the secret monomial for each found candidate.

Observe that the overall efficiency of this approach hinges on the fact that two pairs of equivalent codewords suffice for monomial reconstruction. Since $|V| = |W| = N^2$, where $|L_1| = |L_2| = N$, the general case where c pairs are needed requires lists V', W' of size $|V'| = |W'| = N^c$. Given that N grows exponentially, any non-constant c leads to super-exponential complexity. To avoid this, prior work bypassed the explicit construction of V' and W', instead aiming to identify c equivalent codeword pairs between V and W directly. However, this requires that all $c = \Omega(\log n)$ candidates identified as potentially equivalent are indeed equivalent with high probability. To achieve this, prior work introduced additional constraints. Namely, Beullens [9] enforces that the support of

	Previous	This work
LESS-I	139	**127**
LESS-III	214	**195**
LESS-V	290	**266**

Table 1: Bit complexity estimates for suggested LESS parameters.

the codewords largely overlaps by focusing on 2-dimensional subcodes, while BBPS generalized to smaller, but still lower bounded, support intersections [4]. In summary, both algorithms remain constraint by the high value of c, which leads to suboptimal parameter choices, which we mitigate by relying on the new algorithm achieving $c = 2$.

We then provide further practical speedups, such as the exploitation of the existence of scalar multiples of the solution to reduce the initial list size and the introduction of a new, more efficient *compatibility criterion* outperforming Beullens' Lex function. Overall, 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 (compare to Table 1). With respect to other parameters throughout the literature, we even achieve bit security reductions of up to 42 bits.

We then turn to the asymptotic analysis of the algorithm, providing the first such analysis for a codeword finding based approach dating past Leon's algorithm. In this context we also analyze a commonly known lower bound for algorithms following this paradigm. In particular, we find that this lower bound converges to the complexity of the canonical forms algorithm *from below*, highlighting the potential of codeword-finding based algorithm's even for large values of q. Motivated by our experimental results, we first analyze the case where $q = \omega(1)$ grows with n, leading to a time complexity expressed as a function of the code rate R. For example, for rate $R = \frac{1}{2}$, the complexity of our algorithm is $2^{cn+o(n)}$ with $c \in \left[\frac{2}{3}, \frac{4}{5}\right]$ depending on the size of q. In comparison, Leon's algorithm achieves $c = 1$ in this regime, while the canonical forms approach attains $c = \frac{1}{2}$, suggesting that the new algorithm falls short of the canonical forms approach.

However, this stands in clear contrast to the concrete analysis which shows that the new algorithm remains largely superior to the canonical forms approach even for cryptographic sized instances. This discrepancy goes back to the extremely slow convergence of the minimum distance of a random linear code to its asymptotic limit of $n-k$ (used in the analysis) when q grows in n, motivating a pre-asymptotic analysis, where we treat q as a constant. We show that this setting accurately reflects the concrete performance of the algorithm already for small q and, in particular, for instances of cryptographic relevance.

Artifacts. We provide all source code used to conduct the practical experiments and to compute the asymptotic and concrete complexity estimates is available at the git repository [11].

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 introduce our new algorithm for solving LCE from only two pairs. Subsequently, in Section 4, we present the new algorithm for solving LCE and its concrete analysis along with a comparison to the state of the art. Finally, in Section 5, we provide the asymptotic treatment of the algorithm's complexity.

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 $\mathbf{P} \in \mathsf{Perm}_n(\mathbb{F}_q)$. In addition, when referring to a permutation matrix $\mathbf{P}$, we use the notation $\pi(i) = j$ meaning that j is the only index such that $\mathbf{P}(j, i) \neq 0$. We say a probability p is *high* if $p = 1 - o(1)$.

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 $\mathbf{Hc} = \mathbf{0}$ if and only if $\mathbf{c} \in \mathcal{C}$. We refer to n as the *length* of $\mathcal{C}$ and to the quantity $R := k/n$ as the *code rate*. We say that $I \subset [n]$, with $|I| = k$, is an *information set* of $\mathcal{C}$ if $\mathsf{rank}(\mathbf{G}_I) = k$.

Given a codeword $\mathbf{c} \in \mathcal{C}$, we define its support $\mathrm{Supp}(\mathbf{c}) := \{i \in [n] : c_i \neq 0\}$ as the set of indices corresponding to its nonzero entries. 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 $\min(\{\mathrm{wt}(\mathbf{c}) \mid \mathbf{c} \in \mathcal{C} \setminus \{0\}\})$ of a linear code is defined as the minimum weight over all its nonzero codewords. For large n, the minimum distance of a random $[n, k]$ linear code over $\mathbb{F}_q$ is known to 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 $\omega_{\mathrm{GV}} := w_{\mathrm{GV}}/n$.

Below, we formally define the linear code equivalence problem, which we refer to as LCE problem throughout 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 (LCE) *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.*

We generally consider random instances of LCE, where $\mathbf{G}$ and $\mathbf{Q}$ are sampled uniformly at random. Those instances are known to be solvable in (heuristic) polynomial time for $q < 5$ [25], while believed to be hard for any $q \geq 5$ [26]. The special case $\mathbf{Q} \in \mathsf{Perm}_n(\mathbb{F}_q)$ is known as the *Permutation Code Equivalence* (PCE) problem, to which our results apply analogously. If a solution exists for two codes $\mathcal{C},\mathcal{C}'$, we call them *equivalent* and write $\mathcal{C}' = \mathcal{C}\mathbf{Q}$. Motivated by cryptographic constructions, we restrict to LCE instances with a unique solution $\mathbf{Q}$ (up to scalars). We focus on code rates $R \leq \frac{1}{2}$, since solving LCE at rate R is equivalent to solving it at $1-R$ via their dual codes.

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

Lemma 1 (Prange's ISD complexity [23]). *The cost of computing m of the N_w codewords of weight w in a random $[n,k]$ linear code is $\tilde{\mathcal{O}}\left(\frac{m}{N_w}\binom{n}{w}/\binom{n-k}{w}\right)$.*

Furthermore, letting $h(x) := -x\log_2 x - (1-x)\log_2(1-x)$ denote the binary entropy function, we use the following standard approximation of binomial coefficients

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

Canonical Forms Chou, Persichetti, and Santini introduced the notion of *canonical forms* in [13] to show that it suffices to find two information sets I, J for $\mathcal{C},\mathcal{C}'$ *colliding* under $\mathbf{Q}$, i.e., that are sent to each other by $\mathbf{Q}$, to recover the monomial in polynomial time, as we summarize in the following.

Theorem 2. *Let $\mathbf{G},\mathbf{G}'$ be generator matrices of two equivalent $[n,k]$ linear codes $\mathcal{C},\mathcal{C}'$ over $\mathbb{F}_q$, for $q \geq 7$, with respect to a monomial matrix $\mathbf{Q}$ and let π be the underlying permutation. Let I and J be colliding information sets of $\mathcal{C}$ and $\mathcal{C}'$, respectively, i.e., $\pi(i) \in J$, for every $i \in I$. Then there exists a polynomial-time algorithm* CF-RecoverMonomial$(\mathbf{G},\mathbf{G}',I,J)$ *that recovers $\mathbf{Q}$ with probability $1-\mathcal{O}(n^{-1})$.*

For a proof of the above, see [13, Proposition 10] and [20, Lemma 3.4]. Generally the time complexity of CF-RecoverMonomial is dominated by the computation of canonical forms which is known to lie in $\mathcal{O}(n^3)$ [7]. Nowakowski [20] introduces slight overheads to guarantee high success probability already for constant field sizes $q \geq 7$. We therefore account for this algorithm conservatively with $\tilde{\mathcal{O}}(n^4)$ in our analysis.

The Subset Sum Problem In our analysis we make use of a reduction to the subset sum problem. We refer to this problem as SSP and it is defined as follows.

Definition 2 (Subset Sum Problem). *Let $m \in \mathbb{N}$ and let $S = \{a_1, \ldots, a_m\}$ be a multiset with $a_i \in \mathbb{N}$. Given a target $b \in \mathbb{N}$, the* Subset Sum Problem *with input (S, b) asks to find a subset $S' \subseteq S$ such that $\sum_{s \in S'} s = b$. If such a subset S' exists, we say that the instance admits an* exact solution*.*

When no exact solution exists, we might consider approximate solutions, which we define as exact solutions for target $b - \varepsilon$, for some $\varepsilon > 0$. Note that while the Subset Sum Problem is NP-complete, several algorithms exist to solve it in practice, including a classical dynamic programming approach that runs in pseudo-polynomial time.

3 Solving LCE Given Two Equivalent Codeword Pairs

In this section, we introduce an algorithm that recovers the monomial transformation from only two pairs of equivalent codewords. The algorithm builds on Theorem 2, in particular, that for the recovery of the monomial it suffices to recover two *colliding information sets* $I \subset [n]$ in $\mathcal{C}$ and $J \subset [n]$ in $\mathcal{C}'$ such that $\pi(i) \in J$ for any $i \in I$, where π is the secret permutation $\mathbf{P}$ defining the monomial.

Let us start by formally defining the notion of *equivalent codewords*.

Definition 3 (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 $\mathbf{w} = \mathbf{v}\mathbf{Q}$.*

In the following, the codes $\mathcal{C}$ and $\mathcal{C}'$ are usually clear from the context, hence, we drop the term *with respect to $\mathcal{C}, \mathcal{C}'$*.

3.1 Basic Idea: Reduction to SSP

Let $\mathcal{C}, \mathcal{C}'$ be two equivalent $[n, k]$ linear codes $\mathcal{C}, \mathcal{C}'$ via a monomial $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$, and let $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2) \in \mathcal{C} \times \mathcal{C}'$ be two pairs of equivalent codewords of weight w. In the following we leverage the information on the support of the codewords, $\mathbf{v}_i, \mathbf{w}_i$ to identify two colliding information sets I, J for $\mathcal{C}$ and $\mathcal{C}'$ respectively. Recall that this implies that I gets mapped into J via the monomial $\mathbf{Q}$, which suffices to recover $\mathbf{Q}$ via Theorem 2.

Refining the SSP *instance.* Let $I_i := \mathrm{Supp}(\mathbf{v}_i) \subset [n]$ and $J_i := \mathrm{Supp}(\mathbf{w}_i) \subset [n]$, for $i = 1, 2$. We have that

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

The sizes for the sets corresponding to J_1, J_2 are analogous. Most importantly, since zero (resp. non-zero) entries of $\mathbf{v}_1, \mathbf{v}_2$ are mapped to zero (resp. non-zero) entries of $\mathbf{w}_1, \mathbf{w}_2$ by $\mathbf{Q}$, we know that the indices in $I_1 \cap I_2$ are mapped into the indices in $J_1 \cap J_2$, those of $I_1 \setminus I_2$ into $J_1 \setminus J_2$ and similarly for the remaining sets. The goal is to combine the sets from Equation (2) to obtain the desired information sets for $\mathcal{C}$ and correspondingly for $\mathcal{C}'$. This problem is a subset sum problem with input integers $\{w-\ell, w-\ell, n-2w+\ell, \ell\}$ and target k, i.e., we are searching for a sum of input integers that sums up to k. Note that this instance could be easily solved via dynamic programming as all given integers are of polynomial size. However, the small set of input integers makes it unlikely that this instance has a solution. In the following, we therefore exploit the additional information given by the support intersection of the two pairs to obtain a refined subset sum instance with a larger input set containing smaller integers.

Let $I_{\text{int}} := I_1 \cap I_2$ and $J_{\text{int}} := J_1 \cap J_2$. As already observed, a monomial $\mathbf{Q}$ with $\mathbf{w}_t = \mathbf{v}_t\mathbf{Q}$ must map the indices in I_{int} into the indices in J_{int}. This implies that for any $i \in I_{\text{int}}$ there exists a $j \in J_{\text{int}}$ such that

$$\mathbf{w}_1(j) = \alpha_{i,j} \cdot \mathbf{v}_1(i) \quad \text{and} \quad \mathbf{w}_2(j) = \alpha_{i,j} \cdot \mathbf{v}_2(i) \quad \text{for some } \alpha_{i,j} \in \mathbb{F}_q^*, \tag{3}$$

where $\alpha_{i,j} = \mathbf{Q}(i,j)$. Note that this implies that $\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)} = \frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}$. This allows us to group the indices in I_{int} and J_{int} into equivalence classes labeled by their quotient $\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)}$, respectively $\frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}$, with the permutation acting exclusively between the indices contained in equivalence classes sharing the same label. More precisely, let

$$F_c^{\mathbf{v}_1,\mathbf{v}_2} := \{i \in I_{\text{int}} \mid \ \mathbf{v}_1[i]/\mathbf{v}_2[i] = c\} \tag{4}$$

for $c \in \mathbb{F}_q^*$, and let $F_c^{\mathbf{w}_1,\mathbf{w}_2}$ be analogously defined for the set J_{int} and vectors $\mathbf{w}_i$. Then, for every $i \in F_c^{\mathbf{v}_1,\mathbf{v}_2}$ it holds $\pi(i) \in F_c^{\mathbf{w}_1,\mathbf{w}_2}$. In turn, we obtain a refined subset sum instance with input integers being the multiset

$$S = \{w-\ell, w-\ell, n-2w+\ell\} \cup D, \tag{5}$$

where D is the multiset defined as

$$D := \left\{|F_c^{\mathbf{v}_1,\mathbf{v}_2}| \colon c \in \mathbb{F}_q^*\right\} \setminus \{0\}. \tag{6}$$

Any solution to the subset sum instance with input set S from Eq. (5) and target k, can be uniquely mapped to a pair of sets $I, J \subset [n]$ of size $|I| = |J| = k$ describing indices in $\mathcal{C}, \mathcal{C}'$ satisfying $\pi(i) \in J$ for $i \in I$. Therefore, we simply translate the integers of the subset sum solution $S' \subset S$ back into the sets from which they originate. We denote this conversion by $\texttt{indices}(\cdot)$, i.e., $I = \texttt{indices}(S', \mathbf{v}_1, \mathbf{v}_2)$ and $J = \texttt{indices}(S', \mathbf{w}_1, \mathbf{w}_2)$. Now, as long as I forms an information set in $\mathcal{C}$ (or analogously J in $\mathcal{C}'$) we can recover the secret monomial via Theorem 2.

We give a pseudocode of the procedure that recovers the secret monomial from two equivalent codeword pairs in Algorithm 1. Note that, in case the subset sum instance does not admit a solution for target k but only for target $k-\varepsilon$

Algorithm 1 SSP-RecoverMonomial$(\mathbf{G}, \mathbf{G}', (\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2))$

Input: $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ generators of $\mathcal{C}, \mathcal{C}'$ respectively. Two pairs of 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 monomial $\mathbf{Q}$ such that $\mathcal{C}' = \mathcal{C}\mathbf{Q}$ or $\perp$

Construct the set S from Eq. (5) from $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)$
Compute (approximate) solution $S' \subset S$ to SSP instance (S, k), i.e., $\sum_{s \in S'} s = k - \delta$, $\delta \geq 0$
$K \leftarrow \texttt{indices}(S', \mathbf{v}_1, \mathbf{v}_2)$ and $K' \leftarrow \texttt{indices}(S', \mathbf{w}_1, \mathbf{w}_2)$
if $|K| = k$ **then**
    $\mathbf{Q} \leftarrow$ CF-RecoverMonomial$(\mathbf{G}, \mathbf{G}', K, K')$
    if $\mathbf{Q} \neq \perp$ **then**
        Return $\mathbf{Q}$
else if $|K| < k$ **then**
    Choose a non-empty set $F_c^{\mathbf{v}_1, \mathbf{v}_2}$ such that $F_c^{\mathbf{v}_1, \mathbf{v}_2} \cap K = \emptyset$, for any $c \in \mathbb{F}_q^*$
    Choose $I^* \subset F_c^{\mathbf{v}_1, \mathbf{v}_2}$ with $|I^*| = \delta$ and let $\tilde{K} \leftarrow K \cup I^*$
    Derive $F_c^{\mathbf{w}_1, \mathbf{w}_2} \subseteq J_1 \cap J_2$, the corresponding set to $F_c^{\mathbf{v}_1, \mathbf{v}_2}$ in $\mathcal{C}'$
    for $J^* \subset F_c^{\mathbf{w}_1, \mathbf{w}_2}$ with $|J^*| = \delta$ **do**
        $\tilde{K}' \leftarrow K' \cup J^*$
        $\mathbf{Q} \leftarrow$ CF-RecoverMonomial$(\mathbf{G}, \mathbf{G}', \tilde{K}, \tilde{K}')$
        if $\mathbf{Q} \neq \perp$ **then**
            Return $\mathbf{Q}$
Return $\perp$

for some $\varepsilon > 0$, we introduce an enumeration procedure to extend the sets to size k. More precisely, we extend I by elements from $F_c^{\mathbf{v}_1, \mathbf{v}_2}$ and J by elements from $F_c^{\mathbf{w}_1, \mathbf{w}_2}$ for some c such that these sets are not yet included in I, J, i.e., $F_c^{\mathbf{v}_1, \mathbf{v}_2} \cap I = F_c^{\mathbf{w}_1, \mathbf{w}_2} \cap J = \emptyset$. In the following, we first prove bounds on the approximation quality, i.e., we upper bound ε, after which we prove the correctness and complexity of the algorithm (in dependence on ε).

3.2 On the Existence of a Subset Sum Solution in Algorithm 1

In the following we lower-bound the necessary field size to ensure that the subset sum instance defined by the input set S from Eq. (5) and target k admits an approximate solution and give an upper bound on the achievable approximation error.

Therefore, consider the following, not necessarily optimal, strategy for constructing a solution. Starting from any of the obtainable sums disregarding the set D, i.e., starting with $s \in \{0, w - \ell, 2w - 2\ell, n - 2w + \ell, n - w, n - \ell\}$ we add elements from set D until we either exactly reach k or the addition of any further element from D leads to a value larger than k. With this strategy, we obtain an (approximate) solution if $s \leq k \leq s + \ell$, since $|D| = \ell$. If we restrict

our attention to $s = n - 2w + \ell$, we obtain the following condition

$$n - 2w + \ell \leq k \leq n - 2w + 2\ell.$$

In the following, we show that this strategy already leads to an approximate solution for *constant* q with an approximation error that is roughly bounded by the average size μ of elements in D.

The information set constraint. Recall that to eventually reconstruct the solution via Theorem 2 we require that the k indices corresponding to the subset sum solution form an information set in $\mathcal{C}$. Since the input integer $n - 2w + \ell$ corresponds to the set $[n] \setminus I_1 \cup I_2$, this set alone cannot represent an information set, as it does not include any elements from the support of $\mathbf{v}_1$ or $\mathbf{v}_2$. It follows that the columns of the generator matrix of $\mathcal{C}$ indexed by this set can have a rank at most $k - 2$. We therefore must ensure to include at least two more indices, one in $\mathrm{Supp}(\mathbf{v}_1)$ and one in $\mathrm{Supp}(\mathbf{v}_2)$ to obtain an information set. In the following, we enforce the stricter criterium of including at least two elements from D (originating from the support intersection) when constructing the colliding information sets. In the context of Algorithm 1 this translates into[2]

$$n - 2w + \ell + \min(D) < k \leq n - 2w + 2\ell.$$

Then, by conservatively bounding $\min(D)$ with the expected size μ of the elements in D, we obtain the condition

$$n - 2w + \ell + \mu < k \leq n - 2w + 2\ell. \tag{7}$$

We prove in Proposition 1 in Appendix A that the set of indices obtained via Equation (7) is an information set with probability analogous to a random set of k indices.

On the distribution of ℓ and D. Modeling the codewords $\mathbf{v}_i$ as uniformly distributed over the set of all weight-w words, ℓ follows a hypergeometric distribution, that is

$$\Pr[\ell = c] = \frac{\binom{w}{c}\binom{n-w}{w-c}}{\binom{n}{w}},$$

and the expected value of ℓ is $\mathbb{E}[\ell] = w^2/n$.

The elements of D are then following a binomial distribution with ℓ trials and success probability $p = \frac{1}{q-1}$ conditioned to only positive values (by definition, there are only non-zero values in D). The mean of this distribution is $\mu = \frac{\ell p}{\tau} \approx \ell p$, since $\tau := \Pr[X > 0] = 1 - (1-p)^\ell$ quickly converges to 1.

[2]For *approximate* solutions this ensures that at least one element from D is included, but then the enumeration procedure within Algorithm 1 ensures that at least one more index from the support intersection is included.

Necessary Field Size to Obtain an Approximate Solution We now show that satisfying inequality (7) leads to a lower bound on the required field size. However, for any constant rate $R \leq 1/2$, this lower bound is a (small) constant as we show in the following lemma.

Lemma 2 (SSP Approximation Exists for Constant Field Size). *Let $n \in \mathbb{N}$ be the code length, and let the rate $R \leq 1/2$ be constant. Then, there exists a constant q_0 such that for any $q \geq q_0$, and w such that $w_{GV} \leq w < \frac{(q-1)n}{q}$, the inequality in Equation (7) is satisfied with probability at least $1/2$ over the choice of the codewords $\mathbf{v}_1, \mathbf{v}_2$.*

Proof. Recall that ℓ follows a hyper-geometric distribution with $p = \frac{w}{n} = \omega$ and $w = \omega n$ trials. Let us assume $\ell \geq \mathbb{E}[\ell] = w^2/n$, which happens with probability at least $1/2$. Now, by substituting $w = \omega n$, $k = Rn$ and $\mu = \ell/(q-1)$ into Eq. (7) and dividing by n we obtain

$$1 - 2\omega + \frac{\ell}{n}\left(1 + \frac{1}{q-1}\right) < R \leq 1 - 2\omega + \frac{2\ell}{n}. \tag{8}$$

Using a standard tail bound for the hypergeometric distribution [14] of ℓ we get

$$\Pr\left[\ell \geq (\omega + \varepsilon)\omega n\right] \leq e^{-2\varepsilon^2 \omega n} \quad \text{for any } \varepsilon \geq 0.$$

Conditioning the tail bound with $\ell \geq \omega^2 n$ and letting $\epsilon := \frac{\varepsilon}{\omega}$ we have

$$\Pr\left[\ell \geq (1+\epsilon)\omega^2 n \mid \ell \geq \omega^2 n\right] = \frac{\Pr\left[\ell \geq (1+\epsilon)\omega^2 n\right]}{\Pr\left[\ell \geq \omega^2 n\right]} \leq 2e^{-2\epsilon^2 \omega^3 n}.$$

This ensure that $\omega^2 n \leq \ell < (1+\epsilon)\omega^2 n\}$ with $1/2 - o(1)$.

To prove that Ineq. 8 holds for all ℓ in this range, we substitute ℓ with $(1+\epsilon)\omega^2 n$ in the lower bound and with $\omega^2 n$ in the upper bound, which leads to

$$1 - 2\omega + \omega^2(1+\epsilon)\left(1 + \frac{1}{q-1}\right) < R \leq 1 - 2\omega + 2\omega^2. \tag{9}$$

Note that the right inequality is trivially satisfied as $-2\omega + 2\omega^2 = 2\omega(\omega - 1) \geq -\frac{1}{2}$ for $0 \leq \omega \leq 1$ and hence

$$1 - 2\omega + 2\omega^2 \geq \frac{1}{2} \geq R.$$

Consider the left side of Ineq. 9 if we define $\epsilon' = \epsilon + \frac{1+\epsilon}{q-1}$,using $1 - 2\omega + \omega^2 = (1-\omega)^2$, this inequality can be rewritten as

$$(1-\omega)^2 + \omega^2\epsilon' < R.$$

The derivative of $(1-\omega)^2 + \omega^2\epsilon'$ is $-2(1-\omega) + 2\omega\epsilon' = 2(\omega(1+\epsilon') - 1)$ which means the function is decreasing in ω for $0 \leq \omega \leq \frac{1}{1+\epsilon'}$. Therefore, if the inequality holds

for the smallest value of $\omega = \omega_{\mathrm{GV}}$ it also holds for all ω wit $\omega_{\mathrm{GV}} \leq \omega \leq \frac{1}{1+\epsilon'}$. Furthermore, as $\epsilon' \geq \frac{1}{q-1}$ we have that $\omega \leq \frac{1}{1+\epsilon'} < \frac{q-1}{q}$, which is given by the statement, it therefore suffices to concentrate on $\omega = \omega_{\mathrm{GV}}$.

The value of ω_{GV} is determined as the smallest ω for which $h_q(\omega) > 1 - R$ then $R > 1 - h_q(\omega_{\mathrm{GV}})$, where h_q is the q-ary entropy function. The left-hand side of Equation (9) can be rewritten as

$$(1 - \omega_{\mathrm{GV}})^2 + \omega_{\mathrm{GV}}^2 \epsilon' < 1 - h_q(\omega_{\mathrm{GV}}) < R,$$

from which we get the simpler expression

$$\omega_{\mathrm{GV}}(2 - \omega_{\mathrm{GV}}(1 + \epsilon')) > h_q(\omega_{\mathrm{GV}}). \tag{10}$$

For $q \to \infty$ we have that $h_q(\omega_{\mathrm{GV}}) \to \omega_{\mathrm{GV}}$ and $\epsilon' \to \epsilon$ which transforms the above inequality into

$$\omega_{\mathrm{GV}}(1 - \omega_{\mathrm{GV}}(1 + \epsilon)) > 0,$$

which is true for all w_{GV} in the considered range $0 < \omega_{\mathrm{GV}} < \frac{1}{1+\epsilon} < \frac{q-1}{q}$. This implies that there exists a q_0, not depending on n, such that the inequality is satisfied for any $q \geq q_0$. □

Lemma 2 shows that a constant field size q_0 is sufficient to ensure that we can obtain an approximate solution to the SSP instance by starting from the set $[n] \setminus I_1 \cup I_2$ and then adding at least two elements from D. While the lemma does not directly state the value of q_0, we can determine an upperbound on this constant as the smallest q satisfying (compare to Eq. (10))

$$\omega_{\mathrm{GV}} \left(2 - \omega_{\mathrm{GV}} \left(1 + \frac{1}{q-1}\right)\right) > h_q(\omega_{\mathrm{GV}}). \tag{11}$$

We illustrate the constant q_0 in Fig. 1 as a function of the rate (dashed line). It can be observed that for rates approaching zero q_0 grows, but already for rates around $R = 0.05$ a field size of $q \approx 31$ is sufficient to obtain an approximate solution.

Note that the analysis is rather conservative, as in the context of Lemma 2 it is only considered to start from the set $[n] \setminus I_1 \cup I_2$, while any other set, i.e., $I_1 \setminus I_2, I_2 \setminus I_1, \emptyset$ or any combination of the sets is also a valid starting choice. Furthermore, the analysis essentially assumes that the elements in D are all matching their expectation, while any variance leading to smaller values in D increases the probability of the subset sum instance admitting an exact solution. For these reasons, the subset sum instance for the q_0 from Lemma 2 can in many practical cases already be solved exactly. We give more details on this in Section 3.4, where we provide comprehensive experiments on the procedure. For now we proceed with the *provable* approximation quality.

On the Approximation Quality of the Solution Lemma 2 proves that already for constant field size we can expect to find an approximate solution, i.e., a

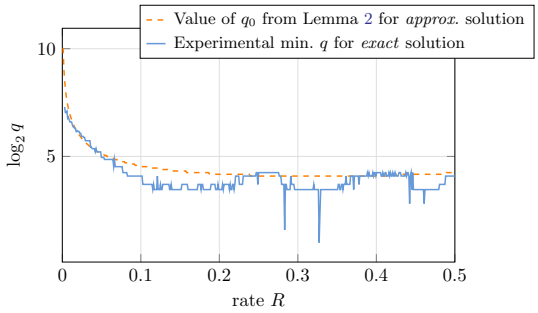


Fig. 1: Lowerbound on q_0 from Equation (11) vs. the experimentally observed minimal q allowing an exact solution to the SSP instance for 10/10 trials, $n = 1024$.

subset sum solution for target $k-\varepsilon$ for some $\varepsilon \geq 0$. The following lemma quantifies the approximation quality by providing an upper bound on ε in dependence on the weight w of the codewords and the field size q.

Lemma 3 (SSP Approximation Error). *Let $n \in \mathbb{N}$ sufficiently large, $k = Rn$ for constant $R \leq 1/2$. Let $q > q_0$ for the constant q_0 from Lemma 2, and $w < \frac{(q-1)n}{q}$. Then, with probability at least $\frac{1}{2} - o(1)$ over the choice of the codewords $\mathbf{v}_1, \mathbf{v}_2$, the subset sum instance with input set S as in Eq. (5) and target $k - \varepsilon$ admits a solution, where $\varepsilon \leq (1+\delta)\frac{2w^2}{(q-1)n} + \log q$ for any constant $\delta > 0$.*

Proof. Again consider the approximate solution constructed by starting from the set $[n] \setminus I_1 \cup I_2$ and then adding elements from the set D until the addition of any further element leads to exceeding k. According to Lemma 2, for the made choice of q, this strategy leads to an approximate solution $k-\varepsilon'$ with probability $1/2 - o(1)$. Clearly, we have that $\varepsilon' \leq \max(D)$. Now observe that the value of $\max(D)$ can be modeled as the maximum load of a bin when throwing $m = \ell$ balls (the elements in the support intersection) into $n = q-1$ bins (the possible quotients in $\mathbb{F}_q^*$). It is well known (see for example [29, Chapter 5]) that for the maximum load $\max(D)$ it holds with high probability that

$$\max(D) = \begin{cases} \mu + \mathcal{O}(\sqrt{\mu \ln n}), & \mu = \Omega(\ln n) \\ \mathcal{O}\left(\frac{\ln n}{\ln \ln n}\right), & \mu = \Theta(1), \end{cases}$$

where $\mu = \mathbb{E}[d] = \frac{m}{n}$, with $d \in D$. Now observe that $\varepsilon = (1+\delta)\mu + \log n$ is an upper bound for both cases for any constant $\delta > 0$. Furthermore, Raab and Steger [24] have shown that the intermediate regime smoothly transitions between the two behaviors resulting in ε being an upper bound for all μ, yielding $\varepsilon' \leq \max(D) \leq \varepsilon$

Now observe that Lemma 2 conditions on $\ell \geq \mathbb{E}[\ell] = \frac{w^2}{n}$ which happens with probability $\Pr[\ell \geq \mathbb{E}[\ell]] \geq 1/2$, implying

$$\mathbb{E}\big[\ell \mid \ell \geq \mathbb{E}[\ell]\big] \leq \frac{\mathbb{E}[\ell]}{\Pr[\ell \geq \mathbb{E}[\ell]]} \leq 2\mathbb{E}[\ell] = \frac{2w^2}{n}.$$

In turn, the expected bin load satisfies

$$\mu = \frac{\mathbb{E}[\ell \mid \ell \geq 1/2]}{q-1} \leq \frac{2\mathbb{E}[\ell]}{q-1} = \frac{2w^2}{n(q-1)}.$$

Substituting $\mu \leq \frac{2w^2}{n}$ and $n = q-1$ into ε yields the statement. □

Note that the approximation error given in Lemma 3 is about $\varepsilon = \mathcal{O}\left(\frac{n}{q} + \log q\right)$, since $w = \Theta(n)$.

3.3 The Complexity of Algorithm 1

We now provide an analysis of the running time of Algorithm 1. Recall that in case the subset sum instance admits an exact solution for target k the algorithm recovers the secret monomial in a straightforward fashion via Theorem 2. In cases where only approximate solutions for target $k-\varepsilon$ exist, an enumeration procedure is used to still recover the secret monomial, whose running time depends on the approximation quality ε. The following theorem states the running time of Algorithm 1 in dependence on n and ε.

Theorem 3. *Let $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ be generator matrices of two equivalent codes $\mathcal{C}, \mathcal{C}'$ via a secret monomial $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$, $q \geq q_0$ be a prime, where q_0 is the constant from Lemma 2, and $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2) \in \mathcal{C}\times\mathcal{C}'$ be two pairs of equivalent codewords of Hamming weight w. Then Algorithm 1 on input $\mathbf{G}, \mathbf{G}'$ and $(\mathbf{v}_i, \mathbf{w}_i)$, for $i = 1, 2$, recovers the secret monomial $\mathbf{Q}$ with probability at least $0.47 - o(1)$ in time $\tilde{\mathcal{O}}(2^{\varepsilon} n^4)$, where ε is defined as in Lemma 3.*

Proof. Lemma 2 guarantees that there exists an (approximate) solution to the SSP instance (S, k) such that the corresponding sets K, K' satisfy $|K| = |K'| = k - \delta$ with probability about $1/2$ for q being a sufficiently large constant. We already argued that the sets of indices K and K' for $\mathcal{C}$ and $\mathcal{C}'$ collide, i.e., they are mapped to each other via the secret monomial $\mathbf{Q}$. If the SSP instance is solved exactly, i.e., $|K| = |K'| = k$, then K and K' can be input to CF-RecoverMonomial to compute the secret monomial $\mathbf{Q}$. In case those sets form information sets in $\mathcal{C}$ and $\mathcal{C}'$, CF-RecoverMonomial has been shown to succeed in recovering the secret monomial with high probability $1 - \mathcal{O}(n^{-1})$ when $q \geq 7$ [20], while numerical analysis shows that $q_0 \geq 19$ for all code rates (see Figure 1). Furthermore, note that Equation (7) ensures that for $|K| = k$ at least two distinct subsets of indices $F_c^{\mathbf{v}_1,\mathbf{v}_2}, F_{c'}^{\mathbf{v}_1,\mathbf{v}_2}$ are included in K, and, hence, the probability of K being an information set is at least $\prod_{i=0}^{k-2}(1 - q_0^{-i})$ according to Appendix A.

The total probability of success is then at least $\frac{1}{2}\left(1-\mathcal{O}(n^{-1})\right)\prod_{i=0}^{k-2}(1-q_0^{-i}) > \frac{1}{2}\cdot 0.94 - o(1) = 0.47 - o(1)$ since $q_0 \geq 19$.

If, on the other hand, the SSP instance is not solved exactly, but instead we have $|K| = k - \delta < k$, the algorithm completes K via bruteforce to form an information set. The brute-force step chooses one of the sets $F_c^{\mathbf{v}_1,\mathbf{v}_2}$, derived from the codeword support intersection of $\mathbf{v}_1, \mathbf{v}_2$, that is not included in K, i.e., $F_c^{\mathbf{v}_1,\mathbf{v}_2} \cap K = \emptyset$. In line 10, an arbitrary subset $I^* \subset F_c^{\mathbf{v}_1,\mathbf{v}_2}$ of size δ is chosen, and the set $\tilde{K} = K \cup I^*$ is constructed, so that $|\tilde{K}| = k$. Subsequently, the algorithm constructs the set $\tilde{K}' = K' \cup J^*$ for every $J^* \subset F_c^{\mathbf{w}_1,\mathbf{w}_2}$ of size $|J^*| = \delta$, where $F_c^{\mathbf{w}_1,\mathbf{w}_2}$ is the set of indices of $\mathcal{C}'$ corresponding to $F_c^{\mathbf{v}_1,\mathbf{v}_2}$. For each trial, the algorithm attempts to recover $\mathbf{Q}$ via CF-RecoverMonomial with input sets $\tilde{K}$ and $\tilde{K}'$. Note that we already know that the indices in $F_c^{\mathbf{v}_1,\mathbf{v}_2}$ are mapped to those in $F_c^{\mathbf{w}_1,\mathbf{w}_2}$. This implies that there must be a set $J^* \subset F_c^{\mathbf{w}_1,\mathbf{w}_2}$ such that I^* is mapped into J^*. Since the algorithm exhaustively iterates over all J^* this set is found. For the correct choice of J^*, CF-RecoverMonomial returns $\mathbf{Q}$ again with probability $\left(1-\mathcal{O}(n^{-1})\right)\prod_{i=0}^{k-2}(1-q^{-i})$ as long as K (or equivalently K') form information sets in $\mathcal{C}$ (and $\mathcal{C}'$). Similar to the case $|K| = k$, Proposition 1 applies because $\tilde{K}$ contains both the set $I^* \subset F_c^{\mathbf{v}_1,\mathbf{v}_2}$ and at least one other set $F_{c'}^{\mathbf{v}_1,\mathbf{v}_2}$ (by Equation (7)) with $c \neq c'$. The proof of Proposition 1 then follows analogously, by considering I^* in place of $F_c^{\mathbf{v}_1,\mathbf{v}_2}$.

The running time of Algorithm 1 is dominated by the brute-force search over $F_c^{\mathbf{w}_1,\mathbf{w}_2}$ and the attempt to reconstruct $\mathbf{Q}$ for each guess. The number of possible sets J^* is $\binom{|F_c^{\mathbf{w}_1,\mathbf{w}_2}|}{\delta}$. This term is maximized when $|F_c^{\mathbf{w}_1,\mathbf{w}_2}| = \max(D)$ and $\delta = \max(D)/2$ as $\binom{\max(D)}{\max(D)/2} \leq 2^{\max(D)}$, while Lemma 3 shows that $\max(D) \leq \varepsilon$. Knowing that the time complexity of CF-RecoverMonomial is $\tilde{\mathcal{O}}(n^4)$ results in the stated running time. □

Note that in Appendix B we show how to further improve this running time to $\tilde{\mathcal{O}}\left(2^{\frac{\varepsilon}{2}}n^4\right)$ by leveraging a meet-in-the-middle strategy that exploits the intrinsics of canonical forms.

Recall that Lemma 3 bounds $\varepsilon = \mathcal{O}\left(\frac{n}{q} + \log q\right)$. This implies that for $q = \Omega\left(\frac{n}{\log n}\right)$ the running time in Theorem 3 is polynomial in n. On the other hand for constant q the running time might be exponential. Moreover, in the intermediate regime, i.e., q growing in n but $q = o\left(\frac{n}{\log n}\right)$, the algorithm's running time is bounded to be subexponential.

However, the analysis concentrates on a specific subset sum solution and makes further conservative assumptions on the distribution on the elements in D. For these reasons, we observe far better results in practice, achieving polynomial time complexity even for small choices of q as we demonstrate in the following section.

3.4 Practical Experiments

We provide a full implementation of the algorithm reconstructing the secret monomial from two equivalent codeword pairs. Therefore, given two pairs of

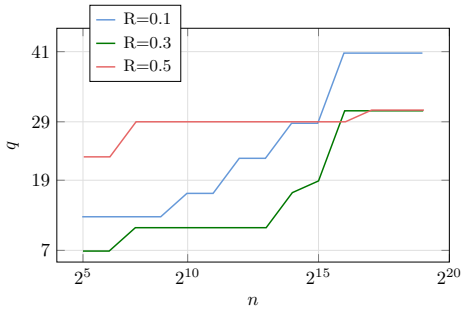


Fig. 2: Smallest prime q leading to exact solutions to the SSP instances from Algorithm 1 in 10 out of 10 trials.

equivalent codewords $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)$ we construct the input set S from Eq. (5) and then compute a solution for target k if it exists. When constructing the solution S', we ensure that $|S' \cap D| \geq 2$ such that the probability of K forming an information set is at least the one of random sets (see Proposition 1 in Appendix A).

In our experiments, if not stated otherwise, we fix $w = w_{\mathrm{GV}}$ to the smallest possible value of w, which seems to constitute the worst case. Note that for this w on average the size of the intersection $\ell := |I_1 \cap I_2|$ is minimized and, in turn, $|[n] \setminus I_1 \cup I_2| = n - 2w + \ell$ is maximized.

Necessary Field Size and Approximation Quality in Practice In Fig. 1 we additionally report for every rate the smallest value of q that led to an *exact* solution of the subset sum instance in 10 out of 10 trials for $n = 1024$ (solid blue line). As can be observed, this q is often smaller or only marginally larger than the bound given by Lemma 2 required for an *approximate* solution (dashed line in Fig. 1). We also experimentally confirm the hypothesis that w_{GV} constitutes the worst weight: A change to $w = \lfloor 1.1 \cdot w_{\mathrm{GV}} \rceil$ leads to lower required values of q.

However, as intuition suggests, the smallest field size q required for an exact solution to exist grows in n, in contrast to the value from Lemma 2, which is constant. In Fig. 2, we report the smallest value of q required to obtain an exact solution in 10 out of 10 trials as a function of n for different rates. While we observe that the value is slowly increasing in n, in the range of parameters investigated, the dependence seems only logarithmic. This would imply the polynomial regime for Algorithm 1 starts from $q = \Omega(\log n)$ rather than from $q = \Omega\left(\frac{n}{\log n}\right)$.

Distribution of Obtainable Subset Sums If the polynomial-regime indeed starts from $q = \Omega(\log n)$, this would imply that the subset sums obtained via sums of elements from D distribute densely in $[\ell]$: The total number of subset

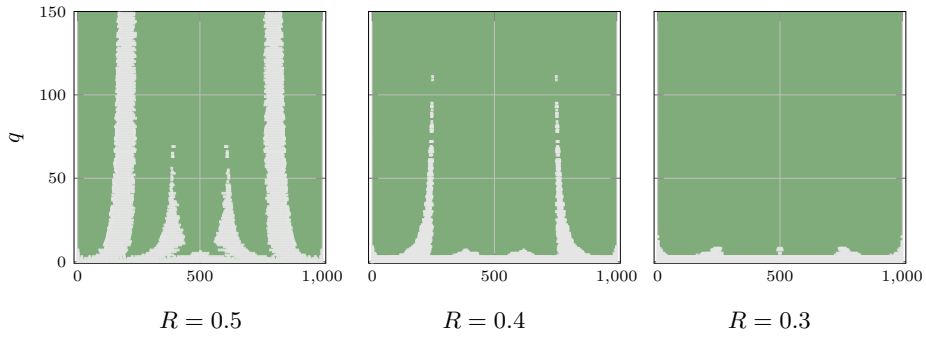


Fig. 3: Distribution of subset sums from S (Equation (5)) with $n = 1000$, $w = w_{GV}$. Green areas represent obtainable sums, gray areas not obtainable subset sums.

sums is $2^{|D|} \sim 2^{q-1}$ and for $q = \Omega(\log n)$ the average gap between obtainable sums becomes $\frac{\sum_{d \in D} d}{2^{|D|}} \sim \frac{\ell}{2^{q-1}} = \mathcal{O}(1)$, as $\ell = \Theta(n)$. Our experiments confirm that the density of the distribution of obtainable subset sums from the set S defining the subset sum instance quickly grows in q. We illustrate the reachable sums in $[n]$ in Fig. 3. Observe, that for smaller rates the sums distribute more evenly, which stems from the higher weight w_{GV} in those cases and the correspondingly larger value of ℓ.

Experiments on LESS Parameters We also provide extensive experiments on parameters suggested in the LESS signature scheme. In particular, we are able to recover the secret monomial from two pairs of equivalent codewords of weight w_{GV} on average in less than 1, 3 and 8 minutes for LESS-I, -III and -V parameters respectively with empirical success probability close to one. Note that in all trials the subset sum instances admitted *exact* solutions for target k. Therefore the only failure originated from the extracted sets K, K' not forming information sets in $\mathcal{C}, \mathcal{C}'$ which happens with probability about $1/q = \frac{1}{127} \approx 0.008$. We averaged for each parameter set over 10,000 experiments. In Table 2 we give the full details on those experiments. We also performed a similar experiment with slightly higher values of w motivated by the application given in Section 4.1, on which the algorithm, as expected, performs similarly well. Those experiments also show that the time to actually solve the subset sum instance is negligible.

4 A New Algorithm Solving Code Equivalence

In the following, we propose a new algorithm for solving LCE, following the codeword-search paradigm, that achieves significant concrete and 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 for sufficiently large q.

	(n,k,q)	w	success rate	Time SSP	Time CF
LESS-I	$(252,126,127)$	107	0.9916	0.02	36.1
		96	0.9920	0.02	35.7
LESS-III	$(400,200,127)$	167	0.9929	0.04	151
		148	0.9922	0.03	152
LESS-V	$(548,274,127)$	226	0.9928	0.05	423
		202	0.9914	0.05	423

Table 2: Empirical time (in seconds) and success probability on LESS parameters.

Algorithm 2 Solving LCE

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

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**
    $\mathbf{Q} \leftarrow \mathsf{SSP\text{-}RecoverMonomial}\big(\mathbf{G}, \mathbf{G}', (\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)\big)$
    if $\mathbf{Q} \neq \perp$ **then**
        Return $\mathbf{Q}$
Return $\perp$

High level description. The algorithm first, similarly to other algorithms based on low-weight codeword finding, 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 1. As shown in the previous section, this reconstruction is successful whenever the two codeword pairs were indeed equivalent under the searched monomial $\mathbf{Q}$ for sufficiently large fields. We summarize this procedure in Algorithm 2.

Analysis of Algorithm 2 We first establish the following theorem, stating the complexity of Algorithm 2 in dependence on the used subroutines. Subsequently, we provide a concrete instantiation of the Compatible-Pairs subroutine. In the following analyses 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. We denote by $T_{\mathsf{ISD}}^{n,k,w,q}$ the time to find a weight-w codeword in an $[n,k]$ linear code over $\mathbb{F}_q$, while by T_{Sub} and p_{Sub} we refer to the time complexity and the success probability of a subroutine Sub.

Theorem 4 (Complexity of Algorithm 2). *Let $n \in \mathbb{N}$, $k = Rn$ for constant rate R, $w \leq n-k$, and $q \geq q_0$ be a prime, where q_0 is the constant from Lemma 2. Then Algorithm 2 solves the* LCE *problem with probability at least $\frac{p_{\mathsf{SSP\text{-}RecoverMonomial}} \cdot p_{\mathsf{Compatible\text{-}Pairs}}}{2}$ in expected time and memory*

$$T = 2N \cdot T_{\mathsf{ISD}}^{n,k,w} + T_{\mathsf{Compatible\text{-}Pairs}} + |Z| \cdot T_{\mathsf{SSP\text{-}RecoverMonomial}} \quad \textit{and} \quad M = \tilde{\mathcal{O}}\left(N + |Z|\right).$$

Proof. Let us fix $N = \sqrt{2N_w}$. The time complexity of the algorithm splits into the time complexity to compute two lists of weight-w codewords of size $|L_i| = N$, the time to construct the list $|Z|$ via $\mathsf{Compatible\text{-}Pairs}$ and the time complexity to apply the $\mathsf{SSP\text{-}RecoverMonomial}$ function to each element $v \in Z$. Adding those terms, leads to the stated running time. The memory complexity is dominated by the size of the lists V, W and Z resulting in the stated memory complexity since $|V| = |W| = N$.[3]

For the correctness note that if 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 $\mathbf{w}_i = \mathbf{v}_i\mathbf{Q}$ the algorithm succeeds with probability $p_{\mathsf{Compatible\text{-}Pairs}} \cdot p_{\mathsf{SSP\text{-}RecoverMonomial}}$. Therefore it remains to show that for $|L_i| = \sqrt{2N_w}$ such pairs exist in the lists L_i with probability at least $\frac{1}{2}$. Note that for any $\mathbf{c} \in \mathcal{C}$ there is exactly one $\mathbf{w} \in \mathcal{C}'$ such that $\mathbf{c} = \mathbf{w}\mathbf{Q}$ for the searched monomial. Therefore we expect that there are $\frac{|L_1 \times L_2|}{N_w} = 2$ equivalent codewords between the two lists. Moreover this number is closely approximated by a binomial distribution with sample size $|L_1 \times L_2| = 2N_w$ and probability N_w^{-1}. For this distribution we know that $\Pr\left[X \geq \mathbb{E}[X]\right] \geq 1/2$. □

Note that in line 7 of Algorithm 2, Algorithm 1 is also called on non-equivalent pairs, not matching its input definition. This is generally not an issue, as such pairs produce non-corresponding sets of indices in line 3, leading to failure in recovering $\mathbf{Q}$ and causing the algorithm to return $\perp$.

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 an instantiation of the $\mathsf{Compatible\text{-}Pairs}$ function in Algorithm 2. More precisely, given two pairs of codewords $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2) \in \mathcal{C} \times \mathcal{C}'$, we say that those pairs are *compatible* if there exists 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, unlike equivalent pairs (see Definition 3), such a monomial $\mathbf{Q}'$ does not necessarily define a linear equivalence between $\mathcal{C}$ and $\mathcal{C}'$, i.e., it is not guaranteed to solve the code equivalence problem. However, any

[3] Here we conveniently assume that none of the subroutines consumes a higher memory, which is generally the case.

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*

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

In Section 3.1, we already observed that a suitable monomial has to map indices in I_{int} to indices in J_{int}, where $I_{\text{int}} = \text{Supp}(\mathbf{v}_1) \cap \text{Supp}(\mathbf{v}_2)$ and $J_{\text{int}} = \text{Supp}(\mathbf{w}_1) \cap \text{Supp}(\mathbf{w}_2)$ are the respective support intersections. Therefore compatibility implies that for every $i \in I_{\text{int}}$ there exists a $j \in J_{\text{int}}$ satisfying (compare to Eq. (3))

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

which translates into a simple criterion to check compatibility of pairs.

Lemma 4 (Compatibility Criterion). *Two pairs of codewords $(\mathbf{v}_1, \mathbf{w}_1)$ and $(\mathbf{v}_2, \mathbf{w}_2)$ of same Hamming weight $\text{wt}(\mathbf{v}_i) = \text{wt}(\mathbf{w}_i) = w$ are compatible if and only if the entries of $\mathbf{v}_{\text{comp}}$ and $\mathbf{w}_{\text{comp}}$ generate the same multiset, where*

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

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

Proof. Given in Appendix C. □

An algorithm for finding compatible pairs. Lemma 4 now easily turns into an algorithm to instantiate the Compatible-Pairs function in Algorithm 2: 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 3 and its complexity in Lemma 5. The time complexity is essentially linear in the sizes of V, W and Z.

Lemma 5 (Finding Compatible Pairs). *Algorithm 3 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. Given in Appendix C. □

Algorithm 3 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: $V' \leftarrow \{(\mathsf{sort}(\mathbf{v}_{\mathrm{comp}}), (\mathbf{v}_1\ , \mathbf{v}_2\ )) \in (\mathbb{F}_q^*)^\ell \times V \ \mid \ell \geq w^2/n\}$
2: $W' \leftarrow \{(\mathsf{sort}(\mathbf{w}_{\mathrm{comp}}), (\mathbf{w}_1, \mathbf{w}_2)) \in (\mathbb{F}_q^*)^\ell \times W \mid \ell \geq w^2/n\}$
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

4.1 Concrete Complexity of Algorithm 2

In this section, we begin by presenting a practical improvement of Algorithm 3, followed by a concrete analysis of Algorithm 2.

Exploiting Scalar Multiples By exploiting the existence of scalar multiples of the solution we are able to decrease the initial list size from $\sqrt{2N_w}$ to

$$N = \sqrt{\frac{2N_w}{q-1}}, \tag{12}$$

without reducing the success probability of Algorithm 2. This requires an adaptation of Algorithm 3 to still find all pairs, compatible up to a scalar, resulting in a slightly increased time complexity. We provide the full details on this improvement, including the pseudocode and proofs in Appendix D.1.

The Concrete Algorithm Consider the instantiation of Algorithm 2 using a starting list size N as specified in Eq. (12) and where the Compatible-Pairs function is instantiated via the corresponding adaptation of Algorithm 3, i.e., Algorithm 6.

Theorem 5 (Concrete Complexity of Algorithm 2). *Let $n, k, w \in \mathbb{N}$, with $w < n - k$, and let $q \geq q_0$ be a prime, where q_0 is the constant from Lemma 2. Then the* LCE *problem can be solved in expected time*

$$0.47^{-1} \cdot \left(2T_{\mathrm{ISD}}^{n,k,q,w} \cdot \sqrt{\frac{2N_w}{q-1}} + \frac{4N_w(\ell+2n)}{\sqrt{q-1}} + 2^{\varepsilon/2} n^4 \cdot \frac{16(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}\right)$$

and expected memory $M_{\mathrm{ISD}}^{n,k,q,w} + \frac{2N_w(\ell+2n)}{\sqrt{q-1}} + \frac{4(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}$, 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 ε is as defined in Lemma 3.

Proof. Given in Appendix D.2. □

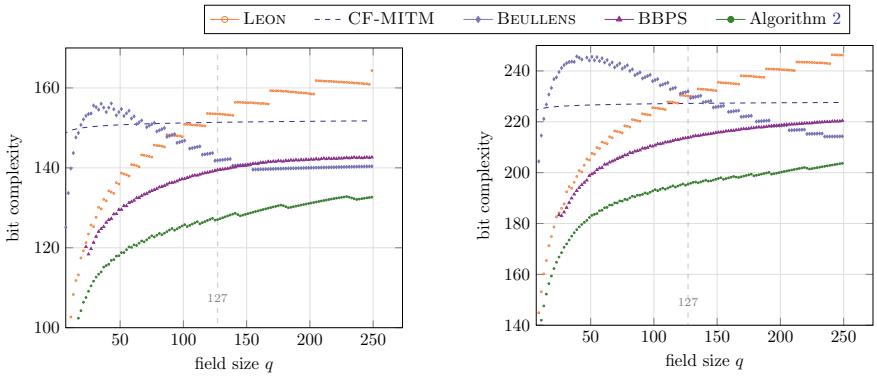


Fig. 4: Bitcomplexity of LCE for $n = 252$ (left) and $n = 400$ (right) with rate 0.5.

Concrete Comparison In the following we provide a comparison to the sate of the art. In line with our experiments from Section 3.4 for the considered parameters, we assume in the following that an exact solution to the subset sum instance exists, i.e., that $\varepsilon = 0$ in Theorem 5. For the estimation of the bitcomplexity of Leon [18], Beullens [9], BBPS [4] as well as $T_{\mathrm{ISD}}^{n,k,q,w}$ in Theorem 5 we rely on the *CryptographicEstimators* library [16][4]. Our estimations closely follow the cost model implemented in this library, ensuring a consistent and fair comparison. Additionally, for completeness, we provide a complexity formula for $T_{\mathrm{ISD}}^{n,k,q,w}$ in Appendix D.3. For the CF-MITM [13] we use the following formula, representing actually a lower bound for the algorithm's complexity

$$T_{\mathsf{CF}} = \mathcal{O}\left(k^2 \cdot n\sqrt{\binom{n}{k}}\right) = \tilde{\mathcal{O}}\left(2^{h(k/n)n/2}\right). \tag{13}$$

Application to LESS parameters. 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. 4, we compare the concrete time complexity of Algorithm 2 given in Theorem 5 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 Algorithm 2 obtains the lowest time complexity. Furthermore we again observe the mild dependence of CF-MITM on the value q. As known, Beullens improves upon Leon only for large fields.

We provide in Table 3 the bitcomplexity estimates of all different algorithms for LESS parameters, where the number in parenthesis denotes the optimal value for w. We observe a bitcomplexity reduction by 12 (LESS-I), 19 (LESS-III) and 24 (LESS-V) bits compared to the best previous algorithm, which is for all parameters the BBPS algorithm. Applied to other parameters in the literature we achieve improvements up to 42 bits (see Appendix D.4).

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

	CF-MITM	Leon	Beullens	BBPS	Algorithm 2
LESS-I	151	153	142	139	**127** (107)
LESS-III	227	230	232	214	**195** (167)
LESS-V	302	307	324	290	**266** (226)

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

5 Asymptotic Analysis of Algorithm 2

In the following, we first study the asymptotic behavior of Algorithm 2 for q growing with n. While this analysis suggests an improvement over Leon's algorithm, it still falls short of the CF-MITM algorithm. However, we observe that this asymptotic behavior is only achieved for unrealistically large $q > 2^{100}$, far beyond cryptographic relevance. This mismatch motivates the study for constant q, which yields a close approximation of the algorithm's concrete behavior already for small fields.

Let us first derive an asymptotic runtime formula from Theorem 4, which serves as a basis for the analyses. Note that from Lemma 1 it follows that $\sqrt{N_w} \cdot T_{\text{ISD}}^{n,k,q,w} = \tilde{\mathcal{O}}\left(\frac{\binom{n}{w}}{\binom{n-k}{w}\sqrt{N_w}}\right)$. Furthermore, as Algorithm 1 runs in time about $T_{\mathsf{SSP\text{-}RecoverMonomial}} = 2^{\varepsilon}$, we have $T_{\mathsf{SSP\text{-}RecoverMonomial}}|Z| = \tilde{\mathcal{O}}(2^{\varepsilon} \cdot T_{\mathsf{Compatible\text{-}Pairs}})$ as Z is the output of the Compatible-Pairs function. Eventually, the asymptotic runtime of Algorithm 2 summarizes as

$$T = \tilde{\mathcal{O}}\left(\max\left(\frac{\binom{n}{w}}{\binom{n-k}{w}\sqrt{N_w}}, 2^{\varepsilon} \cdot T_{\mathsf{Compatible\text{-}Pairs}}\right)\right). \tag{14}$$

5.1 Asymptotic for Non-Constant q

In the following, we analyze the runtime expression from Eq. (14), optimizing it over the choice of the weight w, or more precisely over the corresponding number of codewords N_w. When $q = \mathcal{O}(n)$ grows in n, the minimum distance $w_{\min}$ of a random q-ary linear code with constant rate asymptotically satisfies $w_{\min} = n - k + \delta$, where $\delta = o(n)$. At this weight, the number of codewords is constant, i.e., $N_{w_{\min}} = \Theta(1)$.

At the other extreme, for weight $w = n - k$, the number of codewords is $N_{n-k} = \Theta\left(\binom{n}{n-k}\right)$. Since N_w increases monotonically with w, varying w over the interval $[w_{\min}, n-k]$ allows us to interpolate between these two extremes — gradually increasing N_w up to $\mathcal{O}\left(\binom{n}{n-k}\right)$. In contrast, the effect of increasing w on the term $\binom{n}{w}/\binom{n-k}{w}$ is only subexponential within this range. In fact, using Stirling's approximation, we have

$$\frac{\binom{n}{w_{\min}}}{\binom{n-k}{w_{\min}}} = \binom{n}{n-k}^{1+o(1)}.$$

Furthermore Lemma 3 shows that $\varepsilon = \mathcal{O}\left(\frac{n}{q}\right)$ and, hence, $\tilde{\mathcal{O}}(2^{\varepsilon} \cdot T_{\mathsf{Compatible\text{-}Pairs}}) = T^{1+o(1)}_{\mathsf{Compatible\text{-}Pairs}}$ for any q that grows with n. Thus, for $w \in [w_{\min}, n-k]$, the total runtime from Theorem 4 further simplifies to

$$T = \max\left(\frac{\binom{n}{n-k}}{\sqrt{N_w}}, T_{\mathsf{Compatible\text{-}Pairs}}\right)^{1+o(1)}. \tag{15}$$

The following theorem now states the running time of Algorithm 2 where Compatible-Pairs is instantiated via Algorithm 3.

Theorem 6 (Asymptotic for Non-Constant Field Size). *Let $n \in \mathbb{N}$, $k = Rn$ for constant $R \in [0,1]$ and $q = c_q n$ be prime for $c_q = \mathcal{O}(1)$ and $c_q \gg \frac{1}{n}$. Then Algorithm 2 runs in expected time*

$$T = \max\left(\binom{n}{n-k}^{4/5} \cdot \beta^{\frac{(1-R)^2 n}{5}}, \binom{n}{n-k}^{2/3}\right)^{1+o(1)}, \text{ where } \beta := \min\left(1, \frac{(1-R)^2}{c_q \cdot e}\right).$$

Proof. Given in Appendix E.1. □

Note that the running time of Algorithm 2 interpolates between the two endpoints $\binom{n}{n-k}^{4/5+o(1)} = 2^{4h(1-R)/5n+o(n)}$ and $\binom{n}{n-k}^{2/3+o(1)} = 2^{2h(1-R)/3n+o(n)}$, depending on the size of q, where we approximate binomials via Eq. (1).

Lower bound for codeword-finding-based approaches. Note that a general lower bound for codeword-finding-based approaches (in the style of Algorithm 2) is obtained by making the optimistic assumptions that (1) the secret monomial can be reconstructed from a *single* pair of equivalent codewords in negligible time and (2) that this pair of codewords can be found in time linear in $|L_1| = \mathcal{O}\left(\sqrt{N_w}\right)$. Then minimizing the expression in Eq. (15) over the choice of N_w leads to $N_w = \binom{n}{n-k}^{\frac{1}{2}\pm o(1)}$ with a corresponding running time of

$$T_{\text{Lower}} = \binom{n}{n-k}^{\frac{1}{2}\pm o(1)}. \tag{16}$$

Note that this matches the complexity of the CF-MITM algorithm (compare to Eq. (13)) up to second order terms, when moving to the dual instance.

Comparison to other Approaches and Section 4.1 Recall that Leon's algorithm, requires to find all codewords of minimum weight and therefore achieves a running time (see Lemma 1) of

$$T_{\text{Leon}} = \tilde{\mathcal{O}}\left(\binom{n}{w_{\min}} / \binom{n-k}{w_{\min}}\right) = \binom{n}{n-k}^{1+o(1)} = 2^{h(1-R)n+o(n)}. \tag{17}$$

In summary, the different algorithms achieve asymptotic exponents $c = \log_{\binom{n}{n-k}} T$ (for q growing in n) of

$$c_{\text{LEON}} = 1, \quad c_{\text{new}} \in [2/3, 4/5], \quad c_{\text{Lower}} = c_{\text{CF-MITM}} = 1/2,$$

suggesting that Algorithm 2 does not improve on CF-MITM. However, this stands in clear contrast to the concrete estimates from the previous section. Moreover, the concrete runtime exponent for all LESS parameter sets, obtained as $c = \frac{\log T}{n}$ with T given in Theorem 5, satisfies $c < 0.51 < 2/3h(1-R) = 2/3$ (compare to Table 3). Therefore, even for cryptographic-sized instances, and even though c includes many polynomial factors and second order terms, the concrete runtime exponent stays far below the asymptotic result from Theorem 6.

This is explained by the extremely slow convergence of the minimum weight $w_{\min}$ to its asymptotic limit of $n-k$, used in the analysis. For example, using LESS parameters, i.e., $q = 127$ and rate $R = 1/2$, the minimum weight lies around $w_{\min} = 0.365n$ rather than the asymptotic value of $n/2 - o(n)$. Furthermore, even for q as large as 2^{20}, the minimum weight of a random linear q-ary $[n, n/2]$ code is only about $0.45n$, for $q = 2^{100}$ it reaches $0.49n$. This motivates the following study of a pre-asymptotic regime, where we treat q as a constant while still assuming the recovery of the monomial remains negligible. Note that while Theorem 3 actually yields an exponential complexity for this regime, our experiments have shown that already small values of q suffice to enter the polynomial-time regime. Furthermore, we find this analysis to closely approximates the observed concrete complexity behavior.

5.2 Asymptotic for Constant q

We again start from Eq. (14). However, note that for fixed q the running time of *Algorithm 3* becomes $T_{\mathsf{Compatible\text{-}Pairs}} = \tilde{\mathcal{O}}\left(N_w^2\right)$, since the upper bound on Z given in Lemma 5 becomes trivial implying $Z = \Theta(N_w^2)$.

Lemma 6 (Size of Z for Constant q). *Let $n \in \mathbb{N}$. For any constant q the expected size of Z in Algorithm 2 is $\mathbb{E}\big[|Z|\big] = \frac{|V \times W|}{\text{poly}(\ell)}$.*

Proof. Given in Appendix E.3. □

We now express the complexity of Algorithm 2 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 using a straightforward application of Eq. (1). For completeness we provide its proof in Appendix E.2. In those statements for correctness we include ε, while as discussed previously, we assume in the subsequent comparison $\varepsilon = o(n)$

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

$$\vartheta = \max\left(h(w) - (1-R)h\left(\frac{\omega}{1-R}\right) - \frac{\gamma}{2}, \quad 2\gamma + \log\frac{\varepsilon}{n}\right)$$

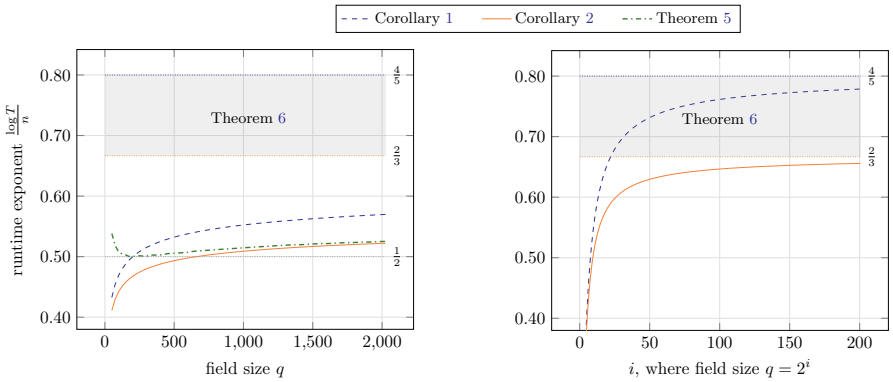


Fig. 5: Asymptotic and concrete runtime exponents where $q = k = \frac{n}{2}$.

where $\gamma = h(\omega) + \omega \log(q-1) - (1-R)\log q$ *and* ε *as defined in Lemma 3.*

For a given rate R and field size q we choose ω to minimize the expression ϑ from Corollary 1. Usually this is the case for an ω balancing both factors in the maximum, i.e., an ω satisfying $h(\omega) - (1-R)h\left(\frac{\omega}{1-R}\right) - \frac{\gamma}{2} = 2\gamma$.

Before comparing the three analyses, we state the following corollary about the running time of Corollary 1 under the assumption that the Compatible-Pairs subroutine runs in linear time $T_{\text{Compatible-Pairs}} = \tilde{\mathcal{O}}(N_w)$, changing the second term in the maximum from $2\gamma + \log \frac{\varepsilon}{n}$ to $\gamma + \log \frac{\varepsilon}{n}$.

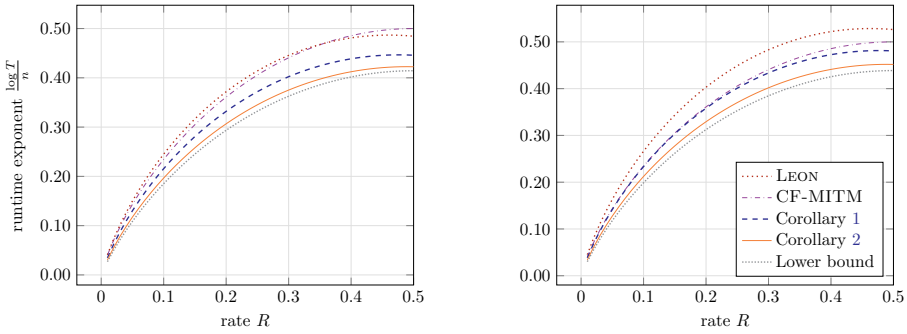
Corollary 2 (Optimistic Runtime Exponent of Algorithm 2). *Let* $n \in \mathbb{N}$, $k = Rn$, $w = \omega n$ *for constants* $\omega, R \in [0,1]$ *and* q *be a constant integer. Then assuming* $T_{\text{Compatible-Pairs}} = \tilde{\mathcal{O}}(N_w)$, *Algorithm 2 runs in expected time* $T = \tilde{\mathcal{O}}\left(2^{\vartheta n}\right)$ *with*

$$\vartheta = \max\left(h(w) - (1-R)h\left(\frac{\omega}{1-R}\right) - \frac{\gamma}{2},\quad \gamma + \log\frac{\varepsilon}{n}\right)$$

where $\gamma = h(\omega) + \omega \log(q-1) - (1-R)\log q$ *and* ε *as defined in Lemma 3.*

Comparison to previous analysis. In Fig. 5 (on the left) we illustrate the runtime exponent from Corollary 1 (blue dashed line), the optimistic exponent from Corollary 2 as well as the concrete runtime exponent obtained from Theorem 5 (green dash dotted line) for rate $R = \frac{q}{n} = \frac{1}{2}$. For comparison we also illustrate the region in which the asymptotic exponent according to Theorem 6 falls, as well as the lower bound of $1/2$.

We observe that the concrete exponent generally lies between the result of Corollary 1 and Corollary 2, with a quick convergence to the latter for increasing q. This convergence is due to the fact that, even if polynomial, the hidden factors in the output list size in Lemma 6 are of order $\Theta(n^q)$ (see the corresponding proof), and therefore quickly affect the concrete complexity even for



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

Fig. 6: Runtime exponent of different algorithms as a function of the rate R for different choices of q.

fixed q. This shows that Corollary 2 closely approximates the concrete behavior of Algorithm 2.

In the same figure on the right, we can see that Corollary 1 and Corollary 2 align well with Theorem 6. Namely, for large $q > 2^{100}$ those variants converge to the exponents $\frac{2}{3}$ and $\frac{4}{5}$ as stated in Theorem 6 when q grows with n.

Comparison to Existing Techniques Now, that we established Corollaries 1 and 2 as solid upper and lower bounds for the complexity of Algorithm 2, we compare them against existing techniques. More precisely we compare against the running time of LEON from Eq. (17) and CF-MITM from Eq. (13) as well as the theoretical lower bound for codeword-finding-based approaches from Eq. (16) for constant q.[5] For constant q the corresponding runtime exponents are

$$c_{\text{Lower}} = \max\left(h(\omega) - (1-R)h\left(\frac{\omega}{1-R}\right) - \frac{\gamma}{2},\ \ \gamma/2\right), \quad \omega \in [\omega_{\text{GV}}, 1-R]$$

similar to the results of Corollaries 1 and 2 (note the $\gamma/2$ in the max) and

$$c_{\text{LEON}} = h(\omega) - (1-R)h\left(\frac{\omega}{1-R}\right) \quad \text{and} \quad c_{\text{CF-MITM}} = \frac{h(R)}{2}.$$

Comparison for fixed q. In Fig. 6 we compare the different runtime exponents for fixed $q \in \{63, 127\}$. Here we restrict to rates $R \leq 1/2$ since solving LCE with code rate R is equivalent to solving LCE with code rate $1 - R$.

We observe that across all code rates, Algorithm 2 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 4 is a factor of $(N_w)^{3/2}$ in the second term of the maximum.

[5] For BEULLENS and BBPS no asymptotic analyses have been provided.

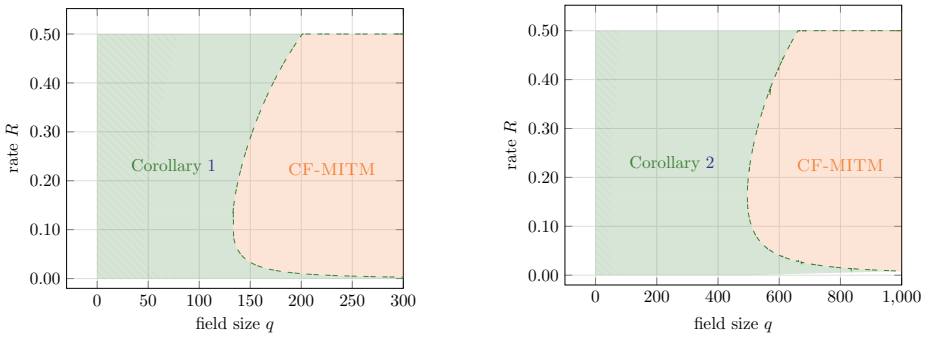


Fig. 7: Partition of the (q, R) space by optimal algorithms.

Comparison in the (q, R) space. In general, the time complexity of Algorithm 2 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 2. However, for rate $1/2$, this break-even point requires $q > 200$ with respect to the (upper) bound from Corollary 1 and $q > 650$ with respect to the (lower) bound from Corollary 2. Note that the we find the practical break even point following the analysis from Section 4.1 to require even higher $q \geq 1550$. In the case of Leon's algorithm, the break-even point is already reached at $q = 81$ (asymptotic) and $q = 98$ (concrete). Fig. 7 classifies the (q, R) space by optimal algorithms, where for Algorithm 2 we use the upper bound from Corollary 1 (left) and lower bound from Corollary 2 (right). Here, Algorithm 2 obtains the lowest runtime exponent in the entire green area, while the striped area corresponds to the regime, where Leon's algorithm is favorable to CF-MITM.

Comparison for the worst case rate. As can be observed from Fig. 6, 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. 8 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 intersections of the runtime exponents of Leon, the upper bound from Corollary 1 and the lower bound from Corollary 2 with this constant lie at $q = 81$, $q = 202$ and $q = 658$, respectively.

References

1. Baldi, M., Beckwith, A.B.L., Biasse, J.F., Chou, T., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.J.O., Santini, P., Wallace, R., Zweydinger, F.: LESS (version 2.0). Tech. rep., National Institute of Standards and Technology (2025), https://www.less-project.com/

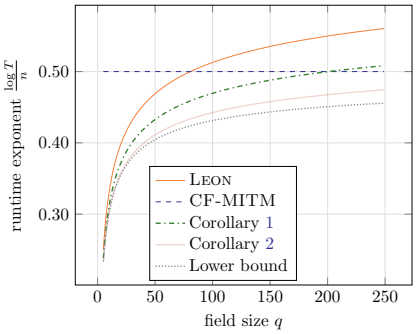


Fig. 8: Runtime exponents for fixed rate $R = \frac{1}{2}$ as a function of the field size q.

2. Barenghi, A., Biasse, J.F., Ngo, T., Persichetti, E., Santini, P.: Advanced signature functionalities from the code equivalence problem. International Journal of Computer Mathematics: Computer Systems Theory **7**(2), 112–128 (2022)
3. Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: LESS-FM: Fine-tuning signatures from the code equivalence problem. In: Cheon, J.H., Tillich, J.P. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021. pp. 23–43. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-81293-5_2
4. Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: On the computational hardness of the code equivalence problem in cryptography. Advances in Mathematics of Communications **17**(1), 23–55 (2023). https://doi.org/10.3934/amc.2022064, https://www.aimsciences.org/article/id/62fa202b4cedfd0007b8b288
5. Battagliola, M., Borin, G., Crescenzo, G.D., Meneghetti, A., Persichetti, E.: Enhancing threshold group action signature schemes: Adaptive security and scalability improvements. Cryptology ePrint Archive, Paper 2025/085 (2025), https://eprint.iacr.org/2025/085
6. Battagliola, M., Borin, G., Meneghetti, A., Persichetti, E.: Cutting the grass: Threshold group action signature schemes. In: Oswald, E. (ed.) Topics in Cryptology – CT-RSA 2024. pp. 460–489. Springer Nature Switzerland, Cham (2024), https://doi.org/10.1007/978-3-031-58868-6_18
7. Beckwith, L., Esser, A., Persichetti, E., Santini, P., Zweydinger, F.: Less is even more: Optimizing digital signatures from code equivalence. IACR Transactions on Cryptographic Hardware and Embedded Systems **2025**(4), 386–408 (2025)
8. Bennett, H., Bhatia, D., Biasse, J.F., Durisheti, M., LaBuff, L., Lavorante, V.P., Waitkevich, P.: Asymptotic improvements to provable algorithms for the code equivalence problem. Cryptology ePrint Archive, Paper 2025/187 (2025), https://eprint.iacr.org/2025/187
9. Beullens, W.: Not enough LESS: An improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In: Dunkelman, O., Jr., M.J.J., O'Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 387–403. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-81652-0_15
10. Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: LESS is more: Code-based signatures without syndromes. In: Nitaj, A., Youssef, A.M. (eds.) AFRICACRYPT 20. LNCS, vol. 12174, pp. 45–65. Springer, Heidelberg (Jul 2020). https://doi.org/10.1007/978-3-030-51938-4_3
11. Budroni, A., Esser, A., Franch, E., Natale, A.: Accompanying repository, https://github.com/AlessandroBudroni/Two-Is-All-It-Takes

12. Cayrel, P.L., Véron, P., El Yousfi Alaoui, S.M.: A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 171–186. Springer, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-19574-7_12
13. Chou, T., Persichetti, E., Santini, P.: On linear equivalence, canonical forms, and digital signatures. Designs, Codes and Cryptography (2025). https://doi.org/10.1007/s10623-025-01576-1, https://doi.org/10.1007/s10623-025-01576-1
14. Chvátal, V.: The tail of the hypergeometric distribution. Discrete Mathematics **25**(3), 285–287 (1979). https://doi.org/https://doi.org/10.1016/0012-365X(79)90084-0, https://www.sciencedirect.com/science/article/pii/0012365X79900840
15. Debris-Alazard, T., Sendrier, N., Tillich, J.P.: Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 21–51. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_2
16. Esser, A., Verbel, J., Zweydinger, F., Bellini, E.: SoK: CryptographicEstimators–a software library for cryptographic hardness estimation. In: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security. pp. 560–574 (2024)
17. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 541–572. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_19
18. Leon, J.: Computing automorphism groups of error-correcting codes. IEEE Transactions on Information Theory **28**(3), 496–511 (1982). https://doi.org/10.1109/TIT.1982.1056498
19. National Institute of Standards and Technology: Post-quantum cryptography: Digital signature schemes. Round 2 Additional Signatures (2023), https://csrc.nist.gov/projects/pqc-dig-sig/round-2-additional-signatures
20. Nowakowski, J.: An improved algorithm for code equivalence. In: Post-Quantum Cryptography: 16th International Workshop, PQCrypto 2025, Taipei, Taiwan, April 8–10, 2025, Proceedings, Part I. p. 71–103. Springer-Verlag, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-031-86599-2_3, https://doi.org/10.1007/978-3-031-86599-2_3
21. Persichetti, E., Santini, P.: A new formulation of the linear equivalence problem and shorter LESS signatures. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VII. LNCS, vol. 14444, pp. 351–378. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8739-9_12
22. Peters, C.: Information-set decoding for linear codes over F_q. In: Sendrier, N. (ed.) The Third International Workshop on Post-Quantum Cryptography, PQCRYPTO 2010. pp. 81–94. Springer, Heidelberg (May 2010). https://doi.org/10.1007/978-3-642-12929-2_7
23. Prange, E.: The use of information sets in decoding cyclic codes. IRE Transactions on Information Theory **8**(5), 5–9 (1962)
24. Raab, M., Steger, A.: "balls into bins"—a simple and tight analysis. In: International Workshop on Randomization and Approximation Techniques in Computer Science. pp. 159–170. Springer (1998)
25. Sendrier, N.: Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Transactions on Information Theory **46**(4), 1193–1203 (2000)

26. Sendrier, N., Simos, D.E.: The hardness of code equivalence over and its application to code-based cryptography. In: Gaborit, P. (ed.) Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013. pp. 203–216. Springer, Heidelberg (Jun 2013). https://doi.org/10.1007/978-3-642-38616-9_14
27. Stern, J.: A method for finding codewords of small weight. In: Cohen, G., Wolfmann, J. (eds.) Coding Theory and Applications. pp. 106–113. Springer Berlin Heidelberg, Berlin, Heidelberg (1989)
28. Stern, J.: A new identification scheme based on syndrome decoding. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_2
29. Upfal, E., Mitzenmacher, M.: Probability and computing. Cambridge University Press, Cambridge, England (2005)
30. Véron, P.: Improved identification schemes based on error-correcting codes. Applicable Algebra in Engineering, Communication and Computing **8**, 57–69 (1997)

A On the Probability of Obtaining an Information Set

In this section we derive an explicit formula for the probability that the set of indices computed in Section 3.2, when Equation (7) is satisfied, is an information set for the code $\mathcal{C}$.

Proposition 1. *Let K be the set of k indices resulting from the union of the indices in $[n] \setminus I_1 \cup I_2$ and two distinct subsets $F_c^{\mathbf{v}_1,\mathbf{v}_2}$, $F_{c'}^{\mathbf{v}_1,\mathbf{v}_2}$ of $I_1 \cap I_2$ (see Equation (4)). Then K is an information set for $\mathcal{C}$ with probability*

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

Proof. The code $\mathcal{C}$ admits a generator matrix $\mathbf{G}$ whose last two rows are the low-weight codewords $\mathbf{v}_1, \mathbf{v}_2$. Due to the randomness of the code, we can regard the matrix formed by the first $k-2$ rows of $\mathbf{G}$ as being uniformly distributed among all the matrices in $\mathbb{F}_q^{(k-2)\times n}$. Let $\mathbf{M}$ be the $k \times k$ submatrix of $\mathbf{G}$ formed by the columns indexed by K. The set K is an information set if and only if $\mathbf{M}$ is full rank. By construction, K contains $n - 2w + \ell$ indices from $[n] \setminus I_1 \cup I_2$, corresponding to the zero entries of $\mathbf{v}_1$ and $\mathbf{v}_2$, and $s := k - n + 2w - \ell \geq 2$ indices arising from the support intersection of the two codewords. It follows that there exists a combination of the rows of $\mathbf{M}$ that puts it in the following block form

$$\mathbf{M} = \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{0}_{2,k-s} & \mathbf{C} \end{array}\right) \quad \in \mathbb{F}_q^{k\times k},$$

where $\mathbf{A} \in \mathbb{F}_q^{(k-2)\times(k-s)}$, $\mathbf{B} \in \mathbb{F}_q^{(k-2)\times s}$, $\mathbf{C} \in \mathbb{F}_q^{2\times s}$, and $\mathbf{0}_{2,k-s} \in \mathbb{F}_q^{2\times(k-s)}$, with $s \geq 2$. Since K contains at least two subsets $F_c^{\mathbf{v}_1,\mathbf{v}_2}$ and $F_{c'}^{\mathbf{v}_1,\mathbf{v}_2}$ of indices coming from the support intersection of $\mathbf{v}_1$ and $\mathbf{v}_2$, the matrix $\mathbf{C}$ has rank 2. After a suitable column permutation, we can write $\mathbf{C} = (\mathbf{C}' \mid \tilde{\mathbf{C}})$ where $\tilde{\mathbf{C}} \in \mathbb{F}_q^{2\times 2}$ is

invertible and contains one column with index in $F_c^{\mathbf{v}_1,\mathbf{v}_2}$ and one with index in $F_{c'}^{\mathbf{v}_1,\mathbf{v}_2}$. Resorting its columns, we have that $\mathbf{M}$ is full rank if

$$\mathbf{M}' = \left(\begin{array}{c|c|c} \mathbf{A} & \mathbf{B}' & \tilde{\mathbf{B}} \\ \hline \mathbf{0}_{2,k-s} & \mathbf{C}' & \tilde{\mathbf{C}} \end{array}\right) \quad \in \mathbb{F}_q^{k\times k}$$

is full rank. Since $\tilde{\mathbf{C}}$ is invertible, $\mathbf{M}'$ has the same rank as:

$$\mathbf{M}'' = \left(\begin{array}{c|c|c} \mathbf{A} & \mathbf{B}'' & \mathbf{0}_{k-2,2} \\ \hline \mathbf{0}_{2,k-s} & \mathbf{0}_{2,s-2} & \mathbf{I}_2 \end{array}\right) \quad \in \mathbb{F}_q^{k\times k},$$

where $\mathbf{B}'' = \mathbf{B}' - \tilde{\mathbf{B}}\tilde{\mathbf{C}}^{-1}\mathbf{C}'$.

Thus,

$$\mathsf{rank}(\mathbf{M}) = \mathsf{rank}(\mathbf{M}'') = \mathsf{rank}(\mathbf{A} \mid \mathbf{B}'') + 2,$$

which shows that $\mathbf{M}$ is full rank k if and only if $[\mathbf{A} \mid \mathbf{B}''] \in \mathbb{F}_q^{(k-2)\times(k-2)}$ is full rank. The matrix $\mathbf{A}|\mathbf{B}'$ is a selection of some columns from the upper part of $\mathbf{G}$ which is a uniform random matrix in $\mathbb{F}_q^{(k-2)\times n}$, as $\tilde{\mathbf{B}}\tilde{\mathbf{C}}^{-1}\mathbf{C}'$ does not depend on $\mathbf{B}'$, the matrix $\mathbf{B}'' = \mathbf{B}' - \tilde{\mathbf{B}}\tilde{\mathbf{C}}^{-1}\mathbf{C}'$ is also a uniform matrix. Then we can treat $\mathbf{A}|\mathbf{B}''$ as a uniformly random matrices n $\mathbb{F}_q^{(k-2)\times(k-2)}$, which leads to

$$\Pr[\mathsf{rank}(\mathbf{M}) = k] \;=\; \Pr[\mathsf{rank}([\mathbf{A} \mid \mathbf{B}'']) = k-2] \;=\; \prod_{i=1}^{k-2}\left(1-q^{-i}\right)$$

□

Notice that, given a $k \times n$ matrix chosen uniformly at random, the probability of a random set of k indices being an information set is $\prod_{i=1}^{k}(1-q^{-i}) < \prod_{i=1}^{k-2}(1-q^{-i})$.

B Canonical Forms and Meet-in-the-Middle Attack

In this section we provide the details on how to improve the enumeration procedure from Algorithm 1 by a meet-in-the-middle technique, to halve the exponent on ε in Theorem 3. We first provide some necessary details on the concept of *canonical forms*. This concept was introduced by Chou, Persichetti and Santini to improve signature sizes in the LESS signature scheme. In general, canonical forms are built upon an equivalence relation $\sim_{\mathsf{F}}$ between linear codes. In this work, for simplicity, we consider the less-general definition from Nowakowski [20] that is given with respect to the following equivalence relation.

Definition 5 (LRL Equivalence, [20]). *Two generator matrices in systematic form*

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}_1], \qquad \mathbf{G}' = [\mathbf{I}_k \mid \mathbf{A}_2] \;\in\; \mathbb{F}_q^{k\times n}$$

are called left-right linearly equivalent *($\mathbf{G} \sim_{\mathsf{LRL}} \mathbf{G}'$) if and only if there exist* $\mathbf{Q}_r \in \mathsf{Mono}_k(\mathbb{F}_q)$ *and* $\mathbf{Q}_c \in \mathsf{Mono}_{n-k}(\mathbb{F}_q)$ *such that*

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

The equivalence class of a generator matrix in systematic form $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}]$ *is denoted by* $[\mathbf{G}]_{LRL}$.

Most importantly, LRL equivalence implies linear code equivalence among the corresponding linear codes in the sense of Definition 1.

Definition 6 (Canonical Form Function). *A* canonical form function *is a polynomial-time procedure* CF *which, given as input a generator matrix in systematic form* $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{A}] \in \mathbb{F}_q^{k \times n}$*, outputs either a triple* $(\mathbf{G}^*, \mathbf{Q}_r, \mathbf{Q}_c)$*, where* $(\mathbf{G}^*, \mathbf{Q}_r, \mathbf{Q}_c) \in [\mathbf{G}]_{LRL} \times \mathsf{Mono}_k(\mathbb{F}_q) \times \mathsf{Mono}_{n-k}(\mathbb{F}_q)$*, and* $\mathbf{G}^* = [\mathbf{I}_k \mid \mathbf{A}^*]$ *satisfies* $\mathbf{A}^* = \mathbf{Q}_r \cdot \mathbf{A} \cdot \mathbf{Q}_c$*, or the failure symbol* $\perp$.
The output $\mathbf{G}^*$ *must be* canonical*, meaning that for any two equivalent matrices* $\mathbf{G} \sim_{LRL} \mathbf{G}'$ *such that* $\mathsf{CF}(\mathbf{G}) \neq \perp$ *and* $\mathsf{CF}(\mathbf{G}') \neq \perp$*, we have*

$$\mathsf{CF}(\mathbf{G}) = \mathsf{CF}(\mathbf{G}').$$

The above definition is stated under the assumption that the input generator matrix is already given in systematic form with respect to the standard information set $\{1, \ldots, k\}$. In practice, this can be generalized to an arbitrary information set $I \subseteq \{1, \ldots, n\}$ by reducing the input generator matrix into reduced row echelon form relative to I. To this end, we denote by $\mathbf{P}^I \in \mathsf{Perm}_n(\mathbb{F}_q)$ a permutation such that, for any generator $\mathbf{G} \in \mathbb{F}_q^{k \times n}$, it produces $\mathbf{G} \cdot \mathbf{P}^I = [\mathbf{G}_I | \mathbf{G}_{n \setminus I}]$. In other words, $\mathbf{P}^I$ moves the columns of $\mathbf{G}$ indexed by I to the left, and leaves the others to the right. Hence, to compute the canonical form function with respect to a generic information set I, one can give as input $\mathsf{RREF}(\mathbf{G} \cdot \mathbf{P}^I)$.
We further assume that if I is not an information set, the canonical form function returns $\perp$. More generally, canonical form functions may fail for some input generator matrices. The success probability of a canonical form function CF is defined as

$$P_{\mathsf{CF}}(n, k, q) := \Pr_{\mathbf{A} \leftarrow \mathbb{F}_q^{k \times (n-k)}} [\, \mathsf{CF}([\mathbf{I}_k \mid \mathbf{A}]) \neq \perp \,]. \tag{18}$$

The canonical form function introduced in [20] has success probability $1 - \mathcal{O}(n^{-1})$. We report in Algorithm 4 the procedure to recover the secret monomial adapted from [20, Algorithm 1], with the modification to return $\perp$ if the input information sets are not colliding, meaning that they are not mapped to each other by the secret monomial. Overall, the time complexity of such an algorithm is ruled by the computation of the canonical form function, that we conservatively consider to be $\tilde{\mathcal{O}}(n^4)$.

CF-MITM In [13], the authors demonstrate that canonical forms can be leveraged to design a meet-in-the-middle algorithm for the code equivalence problem. Given two equivalent codes generated by $\mathbf{G}$ and $\mathbf{G}'$, the algorithm searches for

Algorithm 4 CF-RecoverMonomial$(\mathbf{G}, \mathbf{G}', I, J)$

Input: Generators $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k\times n}$ and information sets I, J.
Output: Solution $\mathbf{Q} \in \mathsf{Mono}_n(\mathbb{F}_q)$ with $\mathbf{G}' = \mathbf{U}\cdot\mathbf{G}\cdot\mathbf{Q}$, for a full-rank $\mathbf{U} \in \mathbb{F}_q^{k\times k}$.

1: Compute $\bar{\mathbf{G}} := \mathsf{RREF}\left(\mathbf{G}\cdot\mathbf{P}^I\right)$ and $\mathsf{CF}(\bar{\mathbf{G}}) = (\mathbf{G}^*,\ \mathbf{Q}_r,\ \mathbf{Q}_c)$
2: Compute $\bar{\mathbf{G}}' := \mathsf{RREF}\left(\mathbf{G}'\cdot\mathbf{P}^J\right)$ and $\mathsf{CF}(\bar{\mathbf{G}}') = (\mathbf{G}'^*,\ \mathbf{Q}'_r,\ \mathbf{Q}'_c)$
3: **if** $\mathbf{G}^* = \perp$ **or** $\mathbf{G}'^* = \perp$ **or** $\mathbf{G}^* \neq \mathbf{G}'^*$ **then**
4: **Return** $\perp$
5: Compute

$$\mathbf{Q} := \mathbf{P}^I \cdot \begin{bmatrix} \mathbf{Q}_r^{-1}\mathbf{Q}'_r & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_c\mathbf{Q}_c'^{-1} \end{bmatrix} \cdot \left(\mathbf{P}^J\right)^{-1}.$$

6: **Return Q**

information sets I and J respectively such that $\mathsf{CF}(\mathbf{G}) = \mathsf{CF}(\mathbf{G}')$. Once such a collision has been found, Algorithm 4 allows to efficiently recover the secret monomial in polynomial time. Overall, this algorithm obtains a time and memory complexity of $\tilde{\mathcal{O}}\left(2^{h(k/n)/2}\right)$, where $h(\cdot)$ denotes the binary entropy function.

B.1 Improving Algorithm 1 via meet-in-the-middle

In a nutshell, Algorithm 1 fixes one subset $I^* \subset F_c^{\mathbf{v}_1,\mathbf{v}_2}$ and searches over all subsets $J^* \subset F_c^{\mathbf{w}_1,\mathbf{w}_2}$ of appropriate size $t = k - |K'|$. Instead, Algorithm 5 constructs, for $\sqrt{\binom{|F_c^{\mathbf{v}_1,\mathbf{v}_2}|}{t}}$ subsets I^* of $F_c^{\mathbf{v}_1,\mathbf{v}_2}$, a list of canonical representatives, and then looks for a colliding subset J^* in $F_c^{\mathbf{w}_1,\mathbf{w}_2}$.

Lemma 7. *Let us consider the same setting of Theorem 3. Algorithm 5 recovers the secret monomial* $\mathbf{Q}$ *with probability at least* $0.298 - o(1)$ *and terminates in time* $\tilde{\mathcal{O}}\left(2^{\frac{\varepsilon}{2}}n^4\right)$*, where* ε *is defined as in Lemma 3*

Proof. The proof is analogous to the one of Theorem 3 except for the time complexity of the search subroutine. Here, Algorithm 5 builds a list of size $\sqrt{\binom{|F_c^{\mathbf{v}_1,\mathbf{v}_2}|}{t}}$, and then search over $\sqrt{\binom{|F_c^{\mathbf{w}_1,\mathbf{w}_2}|}{t}}$ subsets of $F_c^{\mathbf{w}_1,\mathbf{w}_2}$ for finding a collision. The complexity of this search is maximized for $|F_c^{\mathbf{v}_1,\mathbf{v}_2}| = \max(D)$ (note that $|F_c^{\mathbf{v}_1,\mathbf{v}_2}| = |F_c^{\mathbf{w}_1,\mathbf{w}_2}|$) and $t = \max(D)/2$. Lemma 3 shows that $\max(D) \leq \varepsilon$. Then, recalling that the time complexity of CF-RecoverMonomial is $\tilde{\mathcal{O}}(n^4)$, the total time complexity is given by

$$\sqrt{\binom{\varepsilon}{\varepsilon/2}} \cdot \tilde{\mathcal{O}}(n^4) = \tilde{\mathcal{O}}(2^{\frac{\varepsilon}{2}}n^4).$$

The success probability estimation is analogous to Theorem 3 times the probability of finding a collision in the meet-in-the-middle search. Using the approximation with the Poisson distribution, for any list size L, one obtains a collision

Algorithm 5 SSP-RecoverMonomialMITM$(\mathbf{G}, \mathbf{G}', (\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2))$

Input: $\mathbf{G}, \mathbf{G}' \in \mathbb{F}_q^{k \times n}$ generators of $\mathcal{C}, \mathcal{C}'$ respectively. Two pairs of 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 monomial $\mathbf{Q}$ such that $\mathcal{C}' = \mathcal{C}\mathbf{Q}$ or $\perp$

Construct the set S from Eq. (5) from $(\mathbf{v}_1, \mathbf{w}_1), (\mathbf{v}_2, \mathbf{w}_2)$
Compute (approximate) solution $S' \subset S$ to SSP instance (S, k) such that $S' \cap D| \geq 2$ and $\sum_{s \in S'} s = k - \delta$
$K \leftarrow \texttt{indices}(S', \mathbf{v}_1, \mathbf{v}_2)$ and $K' \leftarrow \texttt{indices}(S', \mathbf{w}_1, \mathbf{w}_2)$
if $|K| = k$ **then**
    $\mathbf{Q} \leftarrow$ CF-RecoverMonomial$(\mathbf{G}, \mathbf{G}', K, K')$
    if $\mathbf{Q} \neq \perp$ **then**
        Return $\mathbf{Q}$
else
    Choose a non-empty set $F_c^{\mathbf{v}_1, \mathbf{v}_2}$ such that $F_c^{\mathbf{v}_1, \mathbf{v}_2} \cap K = \emptyset$, for $c \in \mathbb{F}_q^*$
    Derive $F_c^{\mathbf{w}_1, \mathbf{w}_2}$ the corresponding set of $F_c^{\mathbf{v}_1, \mathbf{v}_2}$ in $\mathcal{C}'$ using hints
    Initialize an empty list L and let $t = k - |K|$
    for $i = 1, ..., \left\lfloor \sqrt{\binom{|F_c^{\mathbf{v}_1, \mathbf{v}_2}|}{t}} \right\rfloor$ **do**
        Choose a random $I^* \subset F_c^{\mathbf{v}_1, \mathbf{v}_2}$ of size t and set $I = K \cup I^*$
        if $\mathbf{G}_I$ has rank k **and** CF(RREF$(\mathbf{G}_I | \mathbf{G}_{[n] \setminus I})) \neq \perp$ **then**
            Parse the first component of CF(RREF$(\mathbf{G}_I | \mathbf{G}_{[n] \setminus I})$) as $\mathbf{G}^*$
            Store $(\mathbf{G}^*, I)$ in L
    for $i = 1, ..., \left\lfloor \sqrt{\binom{|F_c^{\mathbf{w}_1, \mathbf{w}_2}|}{t}} \right\rfloor$ **do**
        Choose a random $J^* \subset F_c^{\mathbf{w}_1, \mathbf{w}_2}$ of size t and set $J = K' \cup J^*$
        if $\mathbf{G}'_J$ has rank k **and** CF(RREF$(\mathbf{G}'_J | \mathbf{G}'_{[n] \setminus J})) \neq \perp$ **then**
            Parse the first component of CF(RREF$(\mathbf{G}'_J | \mathbf{G}'_{[n] \setminus J})$) as $\mathbf{G}'^*$
            if $(\mathbf{G}'^*, I) \in L$, for some I, **then**
                $\mathbf{Q} \leftarrow$ CF-RecoverMonomial$(\mathbf{G}, \mathbf{G}', I, J)$
                if $\mathbf{Q} \neq \perp$ **then**
                    Return $\mathbf{Q}$
Return $\perp$

with the following constant probability

$$1 - \frac{\binom{L-\sqrt{L}}{\sqrt{L}}}{\binom{L}{\sqrt{L}}} \approx 1 - e^{-1}.$$

Then, the total success probability is $\left(1 - e^{-1}\right) \frac{1}{2} \prod_{i=1}^{k-2} \left(1 - q_0^{-i}\right) - o(1) > 0.298 - o(1)$, for $q \geq 7$ (compare to the proof of Theorem 3). □

C Compatibility Criterion

In this section we provide omitted details on the compatibility criterion starting with the proof of Lemma 4, restating the lemma for convenience.

Lemma 4 (Compatibility Criterion). *Two pairs of codewords* $(\mathbf{v}_1, \mathbf{w}_1)$ *and* $(\mathbf{v}_2, \mathbf{w}_2)$ *of same Hamming weight* $\mathrm{wt}(\mathbf{v}_i) = \mathrm{wt}(\mathbf{w}_i) = w$ *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. (3). This yields for $\mathbf{Q}(i,j) = \alpha_{i,j} = \mathbf{w}_1(j)/\mathbf{v}_1(i)$ 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 $\mathbf{w}_1(j)/\mathbf{v}_1(i)$ (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. (3) which yields that $\mathbf{v}_{\mathrm{comp}}$ and $\mathbf{w}_{\mathrm{comp}}$ share the same multiset of entries. □

We continue with the proof of Lemma 5, again restating the lemma first.

Lemma 5 (Finding Compatible Pairs). *Algorithm 3 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 $\mathbf{v}_{\mathrm{comp}} \in (\mathbb{F}_q^*)^\ell, \mathbf{w}_{\mathrm{comp}} \in (\mathbb{F}_q^*)^\ell$ 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

$$\begin{aligned} p :&= \Pr\left[\mathsf{sort}(\mathbf{v}_{\text{comp}}) = \mathsf{sort}(\mathbf{w}_{\text{comp}}) \mid \mathbf{v}_{\text{comp}}, \mathbf{w}_{\text{comp}} \in (\mathbb{F}_q^*)^\ell\right] \\ &= \frac{|\{\pi(\mathbf{v}_{\text{comp}}) \mid \pi \in \mathsf{Perm}_\ell\}|}{(q-1)^\ell} \leq \frac{|\{\pi(\mathbf{v}_{\text{comp}}) \mid \pi \in \mathsf{Perm}_{w^2/n}\}|}{(q-1)^{w^2/n}} \\ &\leq \frac{(w^2/n)!}{(q-1)^{w^2/n}}, \end{aligned} \tag{19}$$

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 4 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., $\ell = \mathrm{Supp}(\mathbf{v}_1) \cap \mathrm{Supp}(\mathbf{v}_2)$. 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 $\mathrm{Bin}_{w,w/n}$, with mean w^2/n. For $X \sim \mathrm{Bin}_{w,w/n}$ 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 $1/2$. □

D Concrete Analysis of Algorithm 2

D.1 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 5 we chose a list size $N = |L_1| = \sqrt{2N_w}$ in Algorithm 2 that ensures that between the lists V and W there are at least two pairs of codewords equivalent under the same secret monomial $\beta\mathbf{Q}$ with constant probability. In the following we adapt this choice to still guarantee that there are two pairs of equivalent codewords between those lists with constant probability, 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 3 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.,

$$\mathbf{w}_1 = \mathbf{v}_1(\gamma\mathbf{Q}) \quad \text{and } \mathbf{w}_2 = \mathbf{v}_2(\beta\mathbf{Q}), \quad \gamma, \beta \in \mathbb{F}_q^*. \tag{20}$$

In the following, we assume, without loss of generality, that $\gamma = 1$.[6] 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

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

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 4.

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. (20) (with $\gamma = 1$) for an arbitrary monomial $\mathbf{Q}$, we can still follow the same logic used to derive Eq. (3). 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

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

where $\alpha_{i,j} = \mathbf{Q}(i,j)$, which implies $\frac{\mathbf{v}_1(i)}{\mathbf{v}_2(i)} = \beta \cdot \frac{\mathbf{w}_1(j)}{\mathbf{w}_2(j)}$. From here, following the logic of Lemma 4, 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. (20) is satisfied for $\beta = \delta$. 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

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

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

We then modify Algorithm 3 to include for every element V' a total of $|S_1|$ different labels $(\delta_1)^{-1}\cdot\mathsf{sort}(\mathbf{v}_{\text{comp}})$ and analogously $|S_2|$ different labels $\delta_2\cdot\mathsf{sort}(\mathbf{w}_{\text{comp}})$ 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 $Z = \{\big((\mathbf{v}_1, \mathbf{w}_1), (\beta\mathbf{v}_2, \mathbf{w}_2)\big) \mid \mathbf{v}_i \in V, \mathbf{w}_i \in W \wedge \beta \in \mathbb{F}_q^* \wedge (\mathbf{v}_1, \mathbf{w}_1)$ and $(\beta\mathbf{v}_2, \mathbf{w}_2)$ 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: $V' \leftarrow \{((\delta_1)^{-1}\cdot\mathsf{sort}(\mathbf{v}_{\text{comp}}), (\mathbf{v}_1, \mathbf{v}_2), \delta_1) \in (\mathbb{F}_q^*)^\ell \times V \times S_1 \mid \ell \ge w^2/n\}$
4: $W' \leftarrow \{(\delta_2 \cdot\mathsf{sort}(\mathbf{w}_{\text{comp}}), (\mathbf{w}_1, \mathbf{w}_2), \delta_2) \in (\mathbb{F}_q^*)^\ell \times W \times S_2 \mid \ell \ge w^2/n\}$
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 $\mathbf{v}_2' := \delta_1\delta_2\cdot\mathbf{v}_2$
6: **Return** Z

Lemma 8 (Improved Compatible Pairs). *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. (20) 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 5 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 $\delta_i \in S_i$. □

D.2 Concrete Time Complexity of Algorithm 2

Here we provide the proof of the concrete complexity, essentially following from the correctness and complexity statements of the involved subroutines.

Theorem 5 (Concrete Complexity of Algorithm 2). *Let $n, k, w \in \mathbb{N}$, with $w < n-k$, and let $q \ge q_0$ be a prime, where q_0 is the constant from Lemma 2. Then the* LCE *problem can be solved in expected time*

$$0.47^{-1}\cdot\left(2T_{\text{ISD}}^{n,k,q,w}\cdot\sqrt{\frac{2N_w}{q-1}} + \frac{4N_w(\ell+2n)}{\sqrt{q-1}} + 2^{\varepsilon/2}n^4\cdot\frac{16(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}\right)$$

and expected memory $M_{\mathrm{ISD}}^{n,k,q,w} + \frac{2N_w(\ell+2n)}{\sqrt{q-1}} + \frac{4(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}$*, 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* ε *is as defined in Lemma 3.*

Proof. For the correctness we have shown that the chosen list size leads on expectation to two pairs of codewords satisfying Eq. (20) 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 8, namely if their support intersection is larger than its expectation, i.e., $\ell \geq w^2/n$. Now, Theorem 3 states that from this pair the secret monomial is reconstructed via the SSP-RecoverMonomial function with probability at least $0.47 - o(1)$. However, note that from the proof of Theorem 3 it follows that once it holds $\ell \geq w^2/n$, SSP-RecoverMonomial succeeds with probability at least $0.94 - o(1)$. In summary this leads to a success probability of at least $0.47 - o(1)$ for Algorithm 2 and accordingly an expected amount of repetitions until success of at most 0.47^{-1}.

The time complexity follows from Theorem 4. Observe that for $N = \sqrt{\frac{2N_w}{q-1}}$ we have $T_{\mathrm{ISD}}^{n,k,q,w} \cdot 2N = T_{\mathrm{ISD}}^{n,k,q,w} \cdot \sqrt{\frac{8N_w}{q-1}}$. Now, Lemma 8 for $|V| = |W| = N^2$ yields a running time of

$$T_{\mathsf{Compatible\text{-}Pairs}} = \frac{4N_w(\ell+2n)}{\sqrt{q-1}} + \frac{16n(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}},$$

where the expected size of the returned list is $|Z| \leq \frac{4(N_w)^2(w^2/n)!}{(q-1)^{w^2/n+1}}$, as Algorithm 6 enforces $\ell \geq w^2/n$. Eventually, Lemma 7 states that the reconstruction of the monomial runs in time $T_{\mathsf{SSP\text{-}RecoverMonomial}} = \tilde{\mathcal{O}}(2^{\varepsilon/2}n^4)$. 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 follows equally. □

D.3 An Estimate for $T_{\mathsf{ISD}}^{n,k,w,q}$

In the estimation of the algorithm's concrete complexity we rely on the *CryptographicEstimators* for obtaining a value for $T_{\mathsf{ISD}}^{n,k,w,q}$. More precisely, the library estimates the complexity of finding one weight-w codeword in an $[n,k]$ linear code over $\mathbb{F}_q$ via Stern's ISD algorithm [27] adapted to q-ary codes [22]. However, for completeness we re-state below a formula for this algorithms complexity given in the LESS round 2 specification [1]

$$T_{\mathsf{ISD}}^{n,k,w,q} = \min_{a,u}\left\{\frac{C_{\mathrm{ITER}}(w,a,u)}{P_{\mathrm{SUCC}}(w,a,u)}\right\},$$

where

$$C_{\mathrm{ITER}}(w,a,u) = u\left(\left(\frac{k}{2}-a+1\right)+(q-1)^a\left(\binom{\lfloor \frac{k}{2}\rfloor}{a}+\binom{\lceil \frac{k}{2}\rceil}{a}\right)\right) + \frac{\frac{2aq(w-2a+1)}{q-1}\left(1+\frac{q-2}{q}\right)\binom{\lfloor \frac{k}{2}\rfloor}{a}\binom{\lceil \frac{k}{2}\rceil}{a}(q-1)^{2a}}{q^u} + \frac{(n-k)^2(n+k)}{2},$$

and

$$P_{\mathrm{SUCC}}(w,a,u) = \min\left(\frac{\binom{\lfloor \frac{k}{2}\rfloor}{a}\binom{\lceil \frac{k}{2}\rceil}{a}\binom{n-k-u}{w-2a}\cdot N_w}{\binom{n}{w}}, 1\right).$$

D.4 Application to Concrete Parameters

Details on Algorithm 2 applied to LESS. In Table 4 we provide details on the concrete complexity of the different subroutines of Algorithm 2 when applied to LESS instances for optimal parameter choices.

	w	$\lvert L_i\rvert$	$\lvert V\rvert = \lvert W\rvert$	$\lvert Z\rvert$	$T_{\mathsf{SSP\text{-}RecoverMonomial}}$	$T_{\mathsf{Criterion}}$	$T_{\mathrm{ISD}}^{n,k,w,q}$	$T_{\mathsf{Compatible\text{-}Pairs}}$	T
LESS-I	107	51.78	103.57	86.42	31.91	9.1	73.16	117.16	127.04
LESS-III	167	74.49	148.97	149.81	34.58	9.76	118.86	163.22	195.44
LESS-V	226	93.47	186.94	210.54	36.39	10.22	169.97	221.64	265.53

Table 4: Sizes and complexities (in bits) of Algorithm 2.

Here the value $T_{\mathsf{Criterion}} = \log(n+\ell)+1$ defines the time to construct the vectors $\mathbf{v}_{\mathrm{comp}}, \mathbf{w}_{\mathrm{comp}}$ per element of V, W. The value for $T_{\mathrm{ISD}}^{n,k,w,q}$ has been obtained via the *CryptographicEstimators* library using Sterns' ISD algorithm adapted to $\mathbb{F}_q$. The time complexity T is then obtained via Theorem 4. In fact we account for the cost of a field operation by a factor of $T_{\mathsf{op}} = \log^2 q$. Lastly, accounting for the number of expected repetitions $r \leq 2/0.94 < 2.13$ we obtain

$$T = r\cdot\left(2|L_i|\cdot T_{\mathrm{ISD}}^{n,k,w} + T_{\mathsf{op}}\cdot(T_{\mathsf{Compatible\text{-}Pairs}} + |Z|\cdot T_{\mathsf{SSP\text{-}RecoverMonomial}})\right).$$

Note that the value for $T_{\mathrm{ISD}}^{n,k,w}$ is already stated as bitcomplexity in Table 4. Observe that for all parameter sets the initial search for the low-weight codewords (first addend) slightly dominates.

Application to other parameters. Here, we compare the performance of Algorithm 2 to previous approaches on other suggested parameters (see Table 5). 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 2 applies equally to both cases. However, for the comparison, recall that the BBPS algorithm only applies in case of LCE. Further observe that in case of the PCE instances, the necessary requirements of Beullens are not met in 2 out of 5 instances. Overall, we find that across all parameters Algorithm 2 obtains the lowest bit complexity estimates, with reductions ranging from 2 to 42 bits.

		(n, k, q)	CF-MITM	Leon	Beullens	BBPS	Algorithm 2
PCE	[13]	$(252, 126, 127)$	151	153	131	-	**127** (107)
		$(400, 200, 127)$	227	230	-	-	**195** (167)
		$(548, 274, 127)$	302	307	-	-	**266** (226)
	[2]	$(235, 108, 251)$	142	154	151	-	**124** (112)
		$(230, 115, 127)$	140	142	119	-	**117** (98)
LCE	[2]	$(198, 94, 251)$	123	134	113	114	**107** (91)

Table 5: Bitcomplexity estimates for suggested LCE and PCE instances.

E Asymptotic Analysis

We continue with omitted details on the asymptotic analysis.

E.1 Asymptotic for Growing q

Theorem 6 (Asymptotic for Non-Constant Field Size). *Let $n \in \mathbb{N}$, $k = Rn$ for constant $R \in [0, 1]$ and $q = c_q n$ be prime for $c_q = \mathcal{O}(1)$ and $c_q \gg \frac{1}{n}$. Then Algorithm 2 runs in expected time*

$$T = \max\left(\binom{n}{n-k}^{4/5} \cdot \beta^{\frac{(1-R)^2 n}{5}}, \binom{n}{n-k}^{2/3}\right)^{1+o(1)}, \text{ where } \beta := \min\left(1, \frac{(1-R)^2}{c_q \cdot e}\right).$$

Proof. The time complexity of Algorithm 3 is $T_{\text{Compatible-Pairs}} = \tilde{\mathcal{O}}(\max(N_w, |Z|))$. In the following we analyze both cases, i.e., $|Z| = \tilde{\mathcal{O}}(N_w)$ and $N_w = \tilde{\mathcal{O}}(|Z|)$, separately.

Case $|Z| = \tilde{\mathcal{O}}(N_w)$. This implies $T_{\text{Compatible-Pairs}} = \tilde{\mathcal{O}}(N_w)$, in which case we can minimize the runtime expression from Eq. (15) by balancing the two terms in the maximum. Ignoring lower-order terms for clarity, this gives

$$\frac{\binom{n}{n-k}}{\sqrt{N_w}} = T_{\text{Compatible-Pairs}} = N_w \quad \Leftrightarrow \quad N_w = \binom{n}{n-k}^{2/3}.$$

Since we have already argued that N_w can attain any value in the range $[c,\, d\binom{n}{n-k}]$ for constants c, d, this value of N_w is achievable for some valid choice of w. Substituting this into the runtime expression yields

$$T_{\text{opt}} = \binom{n}{n-k}^{2/3+o(1)}.$$

Case $N_w = \tilde{\mathcal{O}}(|Z|)$ Lemma 5 shows that the size of the returned list $|Z|$ is bounded by

$$|Z| \leq N_w^2 \cdot \frac{(w^2/n)!}{(q-1)^{w^2/n}} = \left(N_w^2 \left(\frac{w^2/n}{q \cdot e}\right)^{w^2/n}\right)^{1+o(1)}. \tag{21}$$

Hence, this yields a running time of

$$T = \max\left(\frac{\binom{n}{n-k}}{\sqrt{N_w}},\, |Z|\right)^{1+o(1)} = \max\left(\frac{\binom{n}{n-k}}{\sqrt{N_w}},\, \min\left(N_w^2 \left(\frac{(1-R)^2}{c_q \cdot e}\right)^{(1-R)^2 n}, N_w^2\right)\right)^{1+o(1)},$$

since the upper bound from Eq. (21) remains unchanged for any $w \in [w_{\min}, n-k]$. Note that the minimum takes effect when the upper bound from Eq. (21) becomes trivial, as we know $|Z| \leq N_w^2$. Now, by again balancing both factors in the maximum over the choice of N_w we obtain

$$N_w = \left(\binom{n}{n-k}\beta^{-(1-R)^2 n}\right)^{2/5}, \text{where } \beta := \min\left(1, \frac{(1-R)^2}{c_q \cdot e}\right).$$

Combined, both cases yield the total running time of Algorithm 2 where SSP-RecoverMonomial is instantiated via Algorithm 3 as

$$T = \max\left(\binom{n}{n-k}^{4/5} \cdot \beta^{\frac{(1-R)^2 n}{5}}, \binom{n}{n-k}^{2/3}\right)^{1+o(1)}$$

□

E.2 Asymptotic Runtime Exponent

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

$$\vartheta = \max\left(h(w) - (1-R)h\left(\frac{\omega}{1-R}\right) - \frac{\gamma}{2},\quad 2\gamma + \log\frac{\varepsilon}{n}\right)$$

where $\gamma = h(\omega) + \omega\log(q-1) - (1-R)\log q$ and ε as defined in Lemma 3.

Proof. From Theorem 4 together with Lemma 1 we can conclude that for the complexity of Algorithm 2 we have

$$T = \tilde{\mathcal{O}}\left(\max\left(\frac{\binom{n}{w}}{\binom{n-k}{w}\sqrt{N_w}}, T_{\mathsf{Compatible\text{-}Pairs}}, |Z| \cdot \mathsf{SSP\text{-}RecoverMonomial}\right)\right). \tag{22}$$

From Lemma 6 and Theorem 3 it follows

$$T_{\mathsf{Compatible\text{-}Pairs}} = \tilde{\mathcal{O}}(|Z| \cdot \mathsf{SSP\text{-}RecoverMonomial}) = \tilde{\mathcal{O}}(N_w^2 \cdot 2^{\varepsilon},$$

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

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

where $\log N_w = (h(\omega) + \omega\log(q-1) - (1-R)\log q)\, n$. Now the statement follows by observing that $\gamma = \log N_w$. □

E.3 Complexity of Compatible-Pairs for Constant q

Lemma 6 (Size of Z for Constant q). *Let $n \in \mathbb{N}$. For any constant q the expected size of Z in Algorithm 2 is $\mathbb{E}\big[|Z|\big] = \frac{|V\times W|}{\mathrm{poly}(\ell)}$.*

Proof. In Eq. (19) we observed that the probability of two pairs of random weight w vectors with overlapping support size ℓ being compatible is

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

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

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

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. □