LESS-FM: Fine-tuning Signatures from a Code-based Cryptographic Group Action

Alessandro Barenghi[1], Jean-François Biasse[2], Edoardo Persichetti[3] and Paolo Santini[4]

Politecnico di Milano[1], University of South Florida[2], Florida Atlantic University[3], Universitá Politecnica delle Marche[4]

Abstract. Code-based cryptographic schemes are highly regarded among the quantum-safe alternatives to current standards. Yet, designing code-based signatures using traditional methods has always been a challenging task, and current proposals are still far from the target set by other post-quantum primitives (e.g. lattice-based). In this paper, we revisit a recent work using an innovative approach for signing, based on the hardness of the code equivalence problem. We introduce some optimizations and provide a security analysis for all variants considered. We then show that the new parameters produce instances of practical interest.

1 Introduction

Digital signature schemes are a fundamental primitive in modern times. In fact, such schemes offer a way to achieve authentication, one of the major cryptographic goals, and an all-important task in the digital world. Since their inception, signature schemes have traditionally been designed using classical number theory problems like integer factorization and computing discrete logarithms. The latter is an instance of a *cryptographic group action* (also described by Couveignes as a hard homogeneous space [10]). In the discrete logarithm setting, the action is defined by the exponentiation of elements in a group of prime order. Such an action satisfies a number of very good properties, due to which it was naturally chosen to be the base of several fundamental cryptographic protocols; besides signatures (DSA, ECDSA), it is worth mentioning at least El Gamal encryption, and, most importantly, the Diffie-Hellman key exchange. All of these schemes, however, will be obsolete once a quantum computer with sufficient computational power and stability is able to run attacks such as Shor's algorithm [19]. It is therefore a pressing issue to establish new algorithms for signature schemes, as highlighted by NIST's call for Post-Quantum Standardization [1].

Recently, cryptographic group actions came into the spotlight again, with several improvements over the original work of Couveignes [10] and Stolbunov [20]. This enabled many improvements in the field of *isogeny-based* cryptography, including primitives that were previously missing in the post-quantum scenario such as static-static key exchange protocols.

Our Contribution In this paper, we build on the work of [9], where the LESS signature scheme was proposed, based on the one-wayness of what is the first instance of a code-based cryptographic group action [14]. The scheme relies entirely on the hardness of finding isometries between linear codes, and thus represents a new direction in code-based cryptography. Our main contribution, is to leverage the cryptographic group action framework to introduce a number of significant optimizations. More specifically, we present two techniques that can be applied to the basic LESS protocol. The first is a generalization of the underlying identification scheme that makes use of multi-bit challenges, by changing the role of the selected challenge bits. This results in a tradeoff, with a reduction in signature size, at the expense of an increase in public key size. The second technique, instead, exploits the imbalance between the cost of different responses corresponding to the chosen challenge bits. Choosing the challenge string to have a fixed, low Hamming weight ends up in much shorter signatures, as well as providing constant-time verification. We show that the two techniques can be combined, providing a flexible and practical scheme. We give an explicit proof for the EUF-CMA security property of the original LESS scheme, with minor tweaks. This proof serves as a basis for the security of the variant schemes. Note that the multi-bit variants rely on a new problem which we call Multiple Codes Linear Equivalence (MCLE, Problem 2), and for which we give a tight reduction to the Code Equivalence problem. Finally, we present multiple sets of parameters for a concrete instantiation of our scheme, and make practical considerations, including a comparison with the existing code-based alternatives.

The paper is organized as follows. We begin by recalling some useful background notions in Section 2. The LESS signature scheme, and the underlying group action, are presented in Section 3. In Section 4, we describe the various optimizations for the scheme. Finally, in Section 5, we briefly summarize the different attacks techniques against the code equivalence problem, and then provide a discussion on parameter choices, including the comparison with other code-based signature schemes.

2 Background

We will use the following notation throughout the paper: a for scalars, A for sets, $\boldsymbol{a}$ for vectors, $\boldsymbol{A}$ for matrices, a for functions, $\mathcal{A}$ for algorithms. We denote with $\boldsymbol{I}_n$ the $n \times n$ identity matrix, with $[a; b]$ the set of integers $\{a, a+1, \ldots, b\}$, and with $\xleftarrow{\$} A$ the action of sampling uniformly at random from A. We denote with $\mathbb{Z}_q$ the ring of integers modulo q, and with $\mathbb{F}_q$ the finite field of order q. The multiplicative group of $\mathbb{F}_q$ is indicated as $\mathbb{F}_q^*$. We denote with $\mathsf{Aut}(\mathbb{F}_q)$ the group of automorphisms of the field $\mathbb{F}_q$. The sets of vectors and matrices with elements in $\mathbb{Z}_q$ (resp. $\mathbb{F}_q$) are denoted by $\mathbb{Z}_q^n$ and $\mathbb{Z}_q^{m\times n}$ (resp. $\mathbb{F}_q^n$ and $\mathbb{F}_q^{m\times n}$). We also write $\mathbb{Z}_{q,w}^n$ (resp. $\mathbb{F}_{q,w}^n$) to indicate the set of vectors with components in $\mathbb{Z}_q$ (resp. $\mathbb{F}_q$) with length n and Hamming weight w. We write $\mathsf{GL}_k(q)$ for the set of invertible $k \times k$ matrices with elements in $\mathbb{F}_q$, or simply GL_k when the finite

field is implicit. Let S_n be the set of permutations over n elements. For a vector $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{F}_q^n$ and a permutation $\pi \in \mathsf{S}_n$, we write the action of π on $\boldsymbol{x}$ as $\pi(\boldsymbol{x}) = (x_{\pi(1)}, \cdots, x_{\pi(n)})$. A permutation can equivalently be described as an $n \times n$ matrix with exactly one 1 per row and column. Analogously, for *linear isometries*, i.e. transformations $\mu = (\boldsymbol{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathsf{S}_n$, we write the action on a vector $\boldsymbol{x}$ as $\mu(\boldsymbol{x}) = (v_1 x_{\pi(1)}, \cdots, v_n x_{\pi(n)})$. Then, we can also describe these in matrix form as a product $\boldsymbol{Q} = \boldsymbol{DP}$ where $\boldsymbol{P}$ is an $n \times n$ permutation matrix and $\boldsymbol{D} = \{d_{ij}\}$ is an $n \times n$ diagonal matrix with entries in $\mathbb{F}_q^*$. We denote with M_n the set of such matrices, usually known as *monomial* matrices.

2.1 Cryptographic Group Actions

At a high level, a *group action* is an operator involving a group, for which an identity exists, and that satisfies the *compatibility* property, as follows.

Definition 1. *Let X be a set and $(G, \circ)$ be a group. A group action is a mapping*

$$\begin{aligned} \star : X \times G &\to X \\ (x, g) &\to x \star g \end{aligned}$$

such that, for all $x \in X$ and $g_1, g_2 \in G$, it holds that $(x \star g_1) \star g_2 = x \star (g_1 \circ g_2)$.

A group action is usually called *cryptographic* if it satisfies some additional properties that make it interesting in a cryptographic context. In the first place, besides efficient sampling, computation, and membership testing, a cryptographic group action should certainly be *one-way*, i.e. given randomly chosen $x_1, x_2 \in X$, it should be hard to find $g \in G$ such that $x_1 \star g = x_2$ (if such a g exists). Other desirable properties include, for instance, *pseudorandomness* of the output, as well as more traditional ones such as commutativity, transitivity etc. Due to space constraints, we refer the reader to [2] for an extensive treatment of cryptographic group actions and their properties.

2.2 Code Equivalence

An $[n, k]$-*linear code* $\mathfrak{C}$ of length n and dimension k over $\mathbb{F}_q$ is a k-dimensional vector subspace of $\mathbb{F}_q^n$. It can be represented by a matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$, called *generator matrix*, whose rows form a basis for the vector space, i.e., $\mathfrak{C} = \{\boldsymbol{uG},\ \ \boldsymbol{u} \in \mathbb{F}_q^k\}$. Alternatively, a linear code can be represented as the kernel of a matrix $\boldsymbol{H} \in \mathbb{F}_q^{(n-k) \times n}$, known as *parity-check matrix*, i.e. $\mathfrak{C} = \{\boldsymbol{x} \in \mathbb{F}_q^n : \boldsymbol{H}\boldsymbol{x}^T = 0\}$. For both representations, there exists a standard choice, called *systematic form*, which corresponds, respectively, to $\boldsymbol{G} = (\boldsymbol{I}_k \mid \boldsymbol{M})$ and $\boldsymbol{H} = (-\boldsymbol{M}^T \mid \boldsymbol{I}_{n-k})$. Generator (resp. parity-check) matrices in systematic form are obtained by calculating the row-reduced echelon form starting from any other generator (resp. parity-check) matrix. We denote such a procedure by sf. The parity-check matrix is important also as it is a generator for the *dual code*, defined as the set of words that are orthogonal to the code, i.e. $\mathfrak{C}^\perp = \{\boldsymbol{y} \in \mathbb{F}_q^n \ : \ \forall \boldsymbol{x} \in \mathfrak{C},\ \ \boldsymbol{x} \cdot \boldsymbol{y}^T = 0\}$. A code $\mathfrak{C}$ is called *self-orthogonal* or *weakly self-dual* if $\mathfrak{C} \subseteq \mathfrak{C}^\perp$, and simply *self-dual* if $\mathfrak{C} = \mathfrak{C}^\perp$.

The concept of *equivalence* between two codes, in its most general formulation, is defined as follows.

Definition 2 (Code Equivalence). *We say that two linear codes $\mathfrak{C}$ and $\mathfrak{C}'$ are* equivalent*, and write $\mathfrak{C} \sim \mathfrak{C}'$, if there exists a field automorphism $\alpha \in \mathsf{Aut}(\mathbb{F}_q)$ and a linear isometry $\mu = (\boldsymbol{v}; \pi) \in \mathbb{F}_q^{*n} \rtimes \mathsf{S}_n$ that maps $\mathfrak{C}$ into $\mathfrak{C}'$, i.e. such that $\mathfrak{C}' = \mu(\alpha(\mathfrak{C})) = \{\boldsymbol{y} \in \mathbb{F}_q^n \ : \boldsymbol{y} = \mu(\alpha(\boldsymbol{x})),\ \boldsymbol{x} \in \mathfrak{C}\}$.*

Clearly, if $\mathfrak{C}$ and $\mathfrak{C}'$ are two codes with generator matrices $\boldsymbol{G}$ and $\boldsymbol{G}'$, respectively, it holds that

$$\mathfrak{C} \sim \mathfrak{C}' \iff \exists (\boldsymbol{S}; (\alpha, \boldsymbol{Q})) \in \mathsf{GL}_k \rtimes (\mathsf{Aut}(\mathbb{F}_q) \times \mathsf{M}_n) \text{ s.t. } \boldsymbol{G}' = \boldsymbol{S}\alpha(\boldsymbol{G}\boldsymbol{Q}).$$

The notion we just presented is usually known as *semilinear equivalence* and it is the most generic. If the field automorphism is the trivial one (i.e. $\alpha = id$), then the notion is simply known as *linear equivalence*. If, furthermore, the monomial matrix is a permutation (i.e. $\boldsymbol{Q} = \boldsymbol{D}\boldsymbol{P}$ with $\boldsymbol{D} = \boldsymbol{I}_n$), then the notion is known as *permutation equivalence*. Note that, in this work, we always work with prime fields $\mathbb{F}_q$, and therefore the last two notions are the only ones of interest to us. Finally, we state the following computational[1] problem.

Problem 1 (Code Equivalence) *Let $\boldsymbol{G}, \boldsymbol{G}' \in \mathbb{F}_q^{k \times n}$ be two generator matrices for two linearly equivalent codes $\mathfrak{C}$ and $\mathfrak{C}'$. Find a field automorphism $\alpha \in \mathsf{Aut}(\mathbb{F}_q)$ and two matrices $\boldsymbol{S} \in \mathsf{GL}_k$ and $\boldsymbol{Q} \in \mathsf{M}_n$ such that $\boldsymbol{G}' = \boldsymbol{S}\alpha(\boldsymbol{G}\boldsymbol{Q})$.*

We normally refer, respectively, to *semilinear*, *linear* or *permutation equivalence problem*, according to what is the notion of code equivalence considered. Alternatively, we refer simply to the *code equivalence problem* where such distinction is not important.

3 Code-based Group Actions and Applications

We begin by describing the group action associated to code equivalence. To do this we consider the set $X \subseteq \mathbb{F}_q^{k \times n}$ comprised of all full-rank $k \times n$ matrices, i.e. the set of generator matrices of $[n, k]$-linear codes, and $G = \mathsf{GL}_k \rtimes (\mathsf{Aut}(\mathbb{F}_q) \times \mathsf{M}_n)$. Note that this group is isomorphic to the group $(\mathsf{GL}_k \times (\mathbb{F}_q^*)^n) \rtimes (\mathsf{Aut}(\mathbb{F}_q) \times \mathsf{S}_n)$ if we decompose each monomial matrix $\boldsymbol{Q} \in \mathsf{M}_n$ into the product $\boldsymbol{D} \cdot \boldsymbol{P} \in (\mathbb{F}_q^*)^n \rtimes \mathsf{M}_n$; then the group operation $\circ$ is defined as

$$((\boldsymbol{S}, \boldsymbol{D}); (\alpha, \boldsymbol{P})) \circ ((\boldsymbol{S}', \boldsymbol{D}'); (\alpha', \boldsymbol{P}')) = ((\boldsymbol{S}\alpha(\boldsymbol{S}'), \boldsymbol{D} \cdot \alpha(\boldsymbol{D}'\boldsymbol{P})); (\alpha\alpha', \boldsymbol{P}\boldsymbol{P}')).$$

The group action is given by

$$\begin{array}{rccc} \star : & X \times G & \to & X \\ & (\boldsymbol{G}, (\boldsymbol{S}; (\alpha, \boldsymbol{Q}))) & \to & \boldsymbol{S}\alpha(\boldsymbol{G}\boldsymbol{Q}) \end{array}$$

[1] Note that this problem is traditionally formulated as a decisional problem in literature, yet for our purposes it is more natural to present here the search version.

It is easy to see that the action is well-formed, with the identity element being $(\boldsymbol{I}_k; (id, \boldsymbol{I}_n))$. Furthermore, it possesses some essential properties that are of cryptographic interest. First of all, the action satisfies all the basic requirements such as efficient membership testing, sampling, computation etc., to which the authors in [2] assign the nomenclature of *effective*. The action is also *one-way*, based on the hardness of the code equivalence problem. In fact, given $\boldsymbol{G}$ and $\boldsymbol{S}\alpha(\boldsymbol{GQ})$, it should be infeasible to recover $\boldsymbol{S}, \alpha$ and $\boldsymbol{Q}$ in polynomial time, else this would provide a solver for the problem. Unfortunately, our group action does not satisfy some useful additional properties (as formalized in [2]). For instance, it is not *transitive*, meaning that it is not possible to connect every element of X (i.e. every generator matrix) via a group element. Most importantly, the action is not commutative, which is a considerable obstacle in the design of cryptographic protocols. Nevertheless, it is possible to employ the group action for this purpose successfully, as we will see. Note that the above formulation includes some trivial instances, e.g. those such that $\boldsymbol{Q} = \boldsymbol{D} \cdot \boldsymbol{I}_n$, in which case the action returns just a different generator matrix for the same code. Thus, in practice, it makes sense to consider a simplified version of the group action, where X contains only the (full-rank) generator matrices in systematic form, and $G = \mathsf{Aut}(\mathbb{F}_q) \times \mathsf{M}_n$.

The work of [9] introduces a 3-pass identification scheme, with soundness error $1/2$, which defines a zero-knowledge proof of knowledge of an isometry between codes, and is based precisely on code equivalence. The authors suggest that such a scheme can be turned into a signature scheme by applying the Fiat-Shamir transformation, without however providing full details. We give here an explicit description of such a scheme, with the addition of some minor tweaks[2].

Setup	Input parameters $q, n, k, \lambda \in \mathbb{N}$, then set $t = \lambda$. Choose matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ and hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$. Set $\boldsymbol{Q}_0 = \boldsymbol{I}_n$ and $\boldsymbol{G}_0 = \mathsf{sf}(\boldsymbol{G})$.
Private Key	Monomial matrix $\boldsymbol{Q}_1 \in \mathsf{M}_n$.
Public Key	Generator matrix $\boldsymbol{G}_1 = \mathsf{sf}(\boldsymbol{GQ}_1)$.

SIGNER		VERIFIER
For $i = 0 \dots t-1$, choose $\tilde{\boldsymbol{Q}}_i \xleftarrow{\$} \mathsf{M}_n$		
and set $\tilde{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}\tilde{\boldsymbol{Q}}_i)$.		
Set $h = \mathsf{H}(\tilde{\boldsymbol{G}}_0, \dots, \tilde{\boldsymbol{G}}_{t-1}, \boldsymbol{m})$.		
Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \{0,1\}$.		
For $i = 0 \dots t-1$, compute $\mu_i = \boldsymbol{Q}_{h_i}^{-1}\tilde{\boldsymbol{Q}}_i$.		
Set $\sigma = (\mu_0, \dots \mu_{t-1}, h)$.		
	$\xrightarrow{(\boldsymbol{m}, \sigma)}$	
		Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \{0,1\}$.
		For $i = 0 \dots t-1$, compute $\hat{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}_{h_i}\mu_i)$.
		Accept if $\mathsf{H}(\hat{\boldsymbol{G}}_0, \dots, \hat{\boldsymbol{G}}_{t-1}, \boldsymbol{m}) = h$.

Table 1: The LESS Signature Scheme.

[2] For example the original scheme did not use public keys in systematic form.

It is immediate to verify the correctness of the scheme, which follows from the argument given in [9, Section 4]. In particular, when $h_i = 0$, we have $\mu_i = \tilde{\boldsymbol{Q}}_i$ and so $\hat{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}_0\mu_i) = \mathsf{sf}(\boldsymbol{G}\tilde{\boldsymbol{Q}}_i) = \tilde{\boldsymbol{G}}_i$; on the other hand, when $h_i = 1$, we have $\mu_i = \boldsymbol{Q}_1^{-1}\tilde{\boldsymbol{Q}}_i$ and so again $\hat{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}_1\mu_i) = \mathsf{sf}(\boldsymbol{G}\boldsymbol{Q}_1\boldsymbol{Q}_1^{-1}\tilde{\boldsymbol{Q}}_i) = \tilde{\boldsymbol{G}}_i$.

A proof of EUF-CMA security for LESS is given in Appendix B.

4 Optimizations

In this section we discuss possible strategies for optimization. We start by noticing that, for the original LESS scheme (Table 1), we have the following features:

- the public key size is $k(n-k)\lceil\log_2(q)\rceil$ bits;
- the number of rounds t is equal to the desired security level λ;
- consequently, the average signature size in bits is given by

$$t\left(1 + \frac{l_{\mathsf{Seed}} + n\lceil\log_2(n)\rceil + n\lceil\log_2(q)\rceil}{2}\right),$$

  where l_{Seed} is the binary length of seed used as randomness. Note that, when using permutations instead of monomial transformations, the signature size gets reduced as the factor $n\lceil\log_2(q)\rceil$ is removed.

4.1 Multi-bit Challenges

A first natural observation is that signature size can be reduced by decreasing the number of rounds that are necessary to reach the desired preimage security level. This, obviously, requires the soundness error in the underlying ZK identification scheme to decrease proportionally. Such a scenario can be realized, for instance, by increasing the number of challenge bits in each round, as described in Seasign [11]. Each challenge bit becomes an ℓ-bit challenge string, which can be interpreted as an integer between 0 and $2^\ell - 1$, i.e. as an element of $\mathbb{Z}_{2^\ell}$, using the well-known correspondence $\mathbb{Z}_2^\ell = \mathbb{Z}_{2^\ell}$. Accordingly, the scheme is modified to feature $r = 2^\ell$ independent public keys; each challenge string is then used to select one of the keys, for which a response is produced (using the corresponding private key). To keep notation simple, we exploit the bijection mentioned above, and interchangeably use the same symbol to denote an ℓ-bit string (as part of a hash output) or an integer in $[0; 2^\ell - 1]$ (for example, when used as an index). A pictorial representation of this variant is given in Table 2, below, where we call the scheme LESS-M (for Multi-bit).

Note that this variant is more natural than what it may seem at a first glance. In fact, the original LESS signature scheme of Table 1 can be seen as a particular case of the LESS-M scheme, where $\ell = 1$ and $\boldsymbol{Q}_0 = \boldsymbol{I}_n$. The main difference is in the security notion underlying the scheme. The security assumption in this case becomes the following.

Setup	Input parameters $q, n, k, \ell, \lambda \in \mathbb{N}$, then set $r = 2^\ell$ and $t = \lambda/\ell$. Choose matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ and hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$.
Private Key	Monomial matrices $\boldsymbol{Q}_0 \dots \boldsymbol{Q}_{r-1} \in \mathsf{M}_n$.
Public Key	Generator matrices $\boldsymbol{G}_0 \dots \boldsymbol{G}_{r-1}$, where $\boldsymbol{G}_i = \mathsf{sf}(\boldsymbol{G}\boldsymbol{Q}_i)$ for $i = 0 \dots r-1$.

SIGNER		VERIFIER
For $i = 0 \dots t-1$, choose $\tilde{\boldsymbol{Q}}_i \xleftarrow{\$} \mathsf{M}_n$		
and set $\tilde{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}\tilde{\boldsymbol{Q}}_i)$.		
Set $h = \mathsf{H}(\tilde{\boldsymbol{G}}_0, \dots, \tilde{\boldsymbol{G}}_{t-1}, \boldsymbol{m})$.		
Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \mathbb{Z}_2^\ell$.		
For $i = 0 \dots t-1$, compute $\mu_i = \boldsymbol{Q}_{h_i}^{-1}\tilde{\boldsymbol{Q}}_i$.		
Set $\sigma = (\mu_0, \dots \mu_{t-1}, h)$.		
	$\xrightarrow{(\boldsymbol{m}, \sigma)}$	
		Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \mathbb{Z}_2^\ell$.
		For $i = 0 \dots t-1$, compute $\hat{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}_{h_i}\mu_i)$.
		Accept if $\mathsf{H}(\hat{\boldsymbol{G}}_0, \dots, \hat{\boldsymbol{G}}_{t-1}, \boldsymbol{m}) = h$.

Table 2: The LESS-M Signature Scheme.

Problem 2 (Multiple Codes Linear Equivalence) *Consider a collection of linearly equivalent $[n,k]$-linear codes $\mathfrak{C}_0 \dots \mathfrak{C}_{r-1}$, admitting generator matrices $\boldsymbol{G}_0, \dots, \boldsymbol{G}_{r-1}$ of the form $\boldsymbol{S}_0\boldsymbol{G}\boldsymbol{Q}_0, \dots, \boldsymbol{S}_{r-1}\boldsymbol{G}\boldsymbol{Q}_{r-1}$. Find matrices $\boldsymbol{S}^* \in \mathsf{GL}_k$ and $\boldsymbol{Q}^* \in \mathsf{M}_n$ such that $\boldsymbol{G}_{j'} = \boldsymbol{S}^*\boldsymbol{G}_j\boldsymbol{Q}^*$, for some $j \neq j'$.*

This problem is still hard, and directly connected to the hardness of the linear code equivalence problem. A reduction is given in Appendix C.

4.2 Fixed-weight Challenges

In this variant, the key intuition is that different responses, corresponding to different challenge bits, have a very unbalanced impact on the size of the signature. In particular, for the original LESS (Table 1), in the case $h_i = 0$ the response μ_i consists of the purely random monomial matrix $\tilde{\boldsymbol{Q}}_i$, and therefore the signer can transmit just the seed used to generate such random object. This, of course, is much more compact than the monomial matrix $\boldsymbol{Q}_1^{-1}\tilde{\boldsymbol{Q}}_i$ which needs to be transmitted, in full, when $h_i = 1$. It makes sense, therefore, to try and minimize the amount of bits h_i that are equal to 1, in the challenge string h output by H.

This idea is not entirely new in the context of identification and signature schemes. In fact, as reported in [17], the suggestion to use a fixed-weight challenge vector is already present in the original Fiat-Shamir work [15]. More recently, some signature schemes appeared that also make use of a similar approach, albeit in a different context. For instance, Picnic, which earned much praise during the NIST post-quantum standardization process [1], uses a pre-processing stage and a cut-and-choose procedure to achieve the desired security level. This technique was later revisited and generalized by Beullens [8], who presents an application to multivariate schemes, as well as PKP. In all cases, it is evident how picking the challenge vector from a carefully crafted distribution beats the simple parallel repetition of the protocol.

In our case, a natural way to implement this idea is to switch the output distribution of H, to return a vector of fixed Hamming weight. In other words, we need to pick H to be a *weight-restricted* hash function, whose range is the set $\mathbb{Z}_{2,\omega}^t$. The modified protocol is described in Table 3 below, where we call the scheme LESS-F (for Fixed-weight).

Setup	Input parameters $q, n, k, \lambda, t, \omega \in \mathbb{N}$. Choose matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ and w.r. hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_{2,\omega}^t$. Set $\boldsymbol{Q}_0 = \boldsymbol{I}_n$ and $\boldsymbol{G}_0 = \mathsf{sf}(\boldsymbol{G})$.
Private Key	Monomial matrix $\boldsymbol{Q}_1 \in \mathsf{M}_n$.
Public Key	Generator matrix $\boldsymbol{G}_1 = \mathsf{sf}(\boldsymbol{G}\boldsymbol{Q}_1)$.

SIGNER		VERIFIER
For $i = 0 \dots t-1$, choose $\tilde{\boldsymbol{Q}}_i \xleftarrow{\$} \mathsf{M}_n$ and set $\tilde{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}\tilde{\boldsymbol{Q}}_i)$. Set $h = \mathsf{H}(\tilde{\boldsymbol{G}}_0, \dots, \tilde{\boldsymbol{G}}_{t-1}, \boldsymbol{m})$. Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \{0,1\}$. For $i = 0 \dots t-1$, compute $\mu_i = \boldsymbol{Q}_{h_i}^{-1}\tilde{\boldsymbol{Q}}_i$. Set $\sigma = (\mu_0, \dots \mu_{t-1}, h)$.	$\xrightarrow{(\boldsymbol{m},\sigma)}$	
		Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \{0,1\}$. For $i = 0 \dots t-1$, compute $\hat{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}_{h_i}\mu_i)$. Accept if $\mathsf{H}(\hat{\boldsymbol{G}}_0, \dots, \hat{\boldsymbol{G}}_{t-1}, \boldsymbol{m}) = h$.

Table 3: The LESS-F Signature Scheme.

A necessary condition to avoid losing security during the process is that the final preimage security level of the protocol remain equal to the original goal of $2^{-\lambda}$. This was naturally obtained via parallel repetition. In our case, simply constraining the challenge vector to a target Hamming weight would be guaranteed to lose security bits. Indeed, this happens even in the most basic scenario, i.e. if we restrict to the expected value $\omega = t/2$; for instance, when $\lambda = 128$, sampling h among the vectors of weight 64 only leads to approximately 124 preimage security bits. To understand this, recall that *preimage security* corresponds, essentially, to the difficulty of guessing the entire challenge vector. In the case of parallel repetition, since each instance is independent from the others, this is equivalent to correctly picking the challenge in each round, which leads to a probability of ε^t, where ε is the soundness error (in our case $1/2$). However, if the challenge is sampled among vectors of fixed Hamming weight, the difficulty of guessing is the reciprocal of

$$\left|\mathbb{Z}_{2,\omega}^t\right| = \binom{t}{\omega}.$$

From this, it follows that, in order to safely switch to constrained-weight challenge vectors, it is necessary to ensure that $\log_2 \binom{t}{\omega} \geq \lambda$. This leads to an increase in the overall length of the challenge vector, yet yields consistently smaller signatures.

4.3 Combining the Approaches

In this section we explain how to combine the approaches illustrated in the previous sections. The result is depicted in Table 4, below, where we call the scheme LESS-FM (as it is a combination of the two techniques).

Setup	Input parameters $q, n, k, \ell, \lambda, t, \omega \in \mathbb{N}$, then set $r = 2^\ell$. Choose matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ and w.r. hash function $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_{2^\ell,\omega}^t$. Set $\boldsymbol{Q}_0 = \boldsymbol{I}_n$ and $\boldsymbol{G}_0 = \mathsf{sf}(\boldsymbol{G})$.
Private Key	Monomial matrices $\boldsymbol{Q}_1 \dots \boldsymbol{Q}_{r-1} \in \mathsf{M}_n$.
Public Key	Generator matrices $\boldsymbol{G}_1 \dots \boldsymbol{G}_{r-1}$, where $\boldsymbol{G}_i = \mathsf{sf}(\boldsymbol{G}\boldsymbol{Q}_i)$ for $i = 1 \dots r-1$.

SIGNER		VERIFIER
For $i = 0 \dots t-1$, choose $\tilde{\boldsymbol{Q}}_i \xleftarrow{\$} \mathsf{M}_n$ and set $\tilde{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}\tilde{\boldsymbol{Q}}_i)$.		
Set $h = \mathsf{H}(\tilde{\boldsymbol{G}}_0, \dots, \tilde{\boldsymbol{G}}_{t-1}, \boldsymbol{m})$.		
Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \{0,1\}^\ell$.		
For $i = 0 \dots t-1$, compute $\mu_i = \boldsymbol{Q}_{h_i}^{-1}\tilde{\boldsymbol{Q}}_i$.		
Set $\sigma = (\mu_0, \dots \mu_{t-1}, h)$.		
	$\xrightarrow{(\boldsymbol{m},\sigma)}$	
		Parse $h = h_0, \dots h_{t-1}$, for $h_i \in \{0,1\}^\ell$.
		For $i = 0 \dots t-1$, compute $\hat{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}_{h_i}\mu_i)$.
		Accept if $\mathsf{H}(\hat{\boldsymbol{G}}_0, \dots, \hat{\boldsymbol{G}}_{t-1}, \boldsymbol{m}) = h$.

Table 4: The LESS-FM Signature Scheme.

This formulation is the most generic, as it includes the previous ones as particular cases. The quantity to consider for preimage security is

$$\left|\mathbb{Z}_{2^\ell,\omega}^t\right| = \binom{t}{\omega}(2^\ell - 1)^\omega.$$

Optimal parameter choices for all variants will be discussed in the next section.

5 Concrete parameter sets and Implementation strategies

Before selecting parameters, we present a brief summary of the complexities of the main known techniques to solve code equivalence; the interested reader can find all the details of the security analysis in the Supplementary Material, included at the end of this manuscript.

Note that the algorithms to solve the code equivalence problem essentially do not change with the equivalence type. One could then think that the two problems are equally hard; yet, this is not true. Indeed, the permutation equivalence is actually easy in general, and only hard for some specific instances, such as for (weakly) self-dual codes. On the other hand, for the linear equivalence problem, no polynomial-time solver is currently known when q is large enough, and thus this is a much harder problem in this scenario.

Type	Algorithm	Complexity	Notes
Permutation	Leon [16]	$O\big(C_{ISD}(q,n,k,d_{GV})\cdot 2\ln(N_w)\big)$	Preferable with small finite fields and large hulls.
	Beullens [7]	$O\left(\frac{2L\cdot C_{ISD}(q,n,k,w)}{N_w\left(1-2^{L\log_2(1-L/N_w)}\right)}\right)$	Preferable with large finite fields and hulls. May fail when L is too small.
	SSA [18]	$O\left(n^3+n^2q^h\log n\right)$	Efficient with small, non-trivial hulls.
	BOS [4]	$\begin{cases} O\left(n^{2.373}C_{WGI}(n)\right) & \text{if } h=0 \\ O\left(n^{2.373+h+1}C_{WGI}(n)\right) & \text{if } h>0 \end{cases}$	Efficient with trivial hulls
Linear	Leon [16]	$O\big(C_{ISD}(q,n,k,d_{GV})\cdot 2\ln(N_w)\big)$	Preferable with small finite fields and large hulls.
	Beullens [7]	$O\left(\frac{2L\cdot C_{ISD}(q,n,k,w)}{N_w\left(1-2^{L\log_2(1-L/N_w)}\right)}\right)$	Preferable with large finite fields and hulls. May fail when L is too small.
	SSA [18]	$\begin{cases} O\left(n^3+n^2q^h\log n\right) & \text{if } q<5 \\ O\left(n^3+n^2q^k\log n\right) & \text{if } q\geq 5 \end{cases}$	Efficient if $q<5$ and the hull is trivial.

Table 5: Summary of techniques to solve the code equivalence problem

Selecting optimal parameters for LESS-FM involves a multi-target optimization where the considered figures of merit are: i) the desired security level, ii) the size of the keypair and of the transmitted signature message, and iii) the computational load required. In this work, we propose parameter sets which are targeted to a computational effort of 2^{128} classical gates, as is standard in literature. This will also facilitate a comparison with other existing signature schemes. Nevertheless, we also ensured that our parameters achieve at least 64 quantum security bits, according to the best known quantum algorithm techniques, an analysis of which is also reported in the Supplementary Material.

Version	Type	Num. of Rounds	$\lvert\mathsf{pk}\rvert$	$\lvert\sigma\rvert$
-	Perm	λ	$k(n-k)Q$	$\lambda\left(1+\frac{\lambda+nN}{2}\right)$
	Mono	λ	$k(n-k)Q$	$\lambda\left(1+\frac{\lambda+nN+nQ}{2}\right)$
M	Perm	$\lceil\frac{\lambda}{\ell}\rceil$	$(2^\ell-1)k(n-k)Q$	$\lceil\frac{\lambda}{r}\rceil\left(2\ell+\frac{nN}{2}\right)$
	Mono	$\lceil\frac{\lambda}{\ell}\rceil$	$(2^\ell-1)k(n-k)Q$	$\lceil\frac{\lambda}{r}\rceil\left(2r+\frac{nN+nQ}{2}\right)$
F	Perm	t s.t. $\binom{t}{\omega}>2^\lambda$	$k(n-k)Q$	$t+(t-\omega)\lambda+\omega nN$
	Mono	t s.t. $\binom{t}{\omega}>2^\lambda$	$k(n-k)Q$	$t+(t-\omega)\lambda+\omega n\left(N+Q\right)$
FM	Perm	t s.t. $\binom{t}{\omega}(2^\ell-1)^\omega>2^\lambda$	$(2^\ell-1)k(n-k)Q$	$\ell t+(t-\omega)\lambda+\omega nN$
	Mono	t s.t. $\binom{t}{\omega}(2^\ell-1)^\omega>2^\lambda$	$(2^\ell-1)k(n-k)Q$	$\ell t+(t-\omega)\lambda+\omega n\left(N+Q\right)$

Table 6: Overview of the number of rounds and public key/signature sizes in bits as a function of the LESS variant parameters, with $N=\lceil\log_2(n)\rceil$ and $Q=\lceil\log_2(q)\rceil$.

Table 6 gives a synthetic view of public key and signature sizes as a function of the LESS variant parameters. To achieve the reported storage complexity, the LESS keypairs are stored representing the field elements of $\mathbb{F}_q$ as $Q = \lceil \log_2(q) \rceil$ integers, and linearizing on the storage the non-trivial portion of the $\mathbf{G}_i$ matrices of the public keys performing bitpacking. This yields a public key size of $(2^\ell - 1)k(n-k)Q$ (considering one can always set $\boldsymbol{Q}_0 = \boldsymbol{I}_n$ and $\boldsymbol{G}_0 = \mathsf{sf}(\boldsymbol{G})$). Concerning the storage of the monomial matrices composing the private key, they can be represented as an length-n vector of pairs storing the index of the permuted element, and the value of the multiplicative coefficient on $\mathbb{F}_q$ of the said element. Note that, while a compact representation of the permutation alone is possible over $\lceil \log_2(n!) \rceil$ bits, this saves a relatively small amount of space (about 4.5% when minimizing both the public key and signature, where such an effect is most evident), at the cost of performing the relatively demanding computation of permutation unranking, to bring the permutation in a usable representation. Finally, as noted, we achieve a significant reduction in signature size by sending the seeds employed to generate the ephemeral random monomial matrices $\tilde{\mathbf{Q}}_i$ instead of the monomial matrices themselves. We also note that, from a computational standpoint, it is more efficient to store the inverses of the monomial matrices in the private key, moving their computation to the key generation process. This results in an overall improvement in the computation time in all the cases where long-term keys for the signatures are employed.

Optimization Criterion	LESS	Type	n	k	q	ℓ	t	ω	**pk (kB)**	**sig (kB)**	**pk + sig (kB)**
Min. pk size	F	Mono	198	94	251	1	283	28	9.77	15.2	24.97
Min. sig size	FM	Perm	235	108	251	4	66	19	205.74	5.25	210.99
Min. pk + sig size	F	Perm	230	115	127	1	233	31	11.57	10.39	21.96
Beullens [7]	-	Mono	250	125	53	1	128	-	11	28	39

Table 7: Parameter sets for LESS-FM, for a security level of $\lambda = 128$ classical bits.

Table 7 reports the result of the optimization of the LESS-FM parameters when targeting the minimization of i) the public key size, ii) the signature size or iii) the sum of the aforementioned quantities. The rationale behind these criteria is to highlight the flexibility of LESS-FM in application scenarios where i) the space for public key storage is constrained (e.g. microcontrollers with tight Flash memory limits), ii) digital certificates, which employ a concatenation of public keys and signatures, and iii) application scenarios where a large amount of signed messages are exchanged between two endpoints employing long-term keypairs. Despite our conservative quantification of the computational effort required by the most effective cryptanalytic approaches, LESS can be instantiated with parameters pushing the size of the public key below 10 kB, or keep the sum of the public key and signature below 22 kB.

Scheme	Security Level	pk (kB)	sig (kB)	pk + sig (kB)	Security Assumption
Stern [13]	80	18.48	113.5	131.98	Low-weight Hamming
Veron [13]	80	18.52	109.05	127.57	Low-weight Hamming
CVE [13]	80	5.31	66.44	71.75	Low-weight Hamming
Wave [12]	128	3205	1.04	3206.04	High-weight Hamming
cRVDC [6]	125	0.15	22.48	22.63	Low-weight Rank
Durandal - I [3]	128	15.24	4.06	19.3	Low-weight Rank
Durandal - II [3]	128	18.60	5.01	23.61	Low-weight Rank
LESS-F min. pk size	128	9.77	15.2	24.97	Linear Equivalence
LESS-FM min. sig size	128	205.74	5.25	210.99	Perm Equivalence
LESS-F min. pk + sig size	128	11.57	10.39	21.96	Perm Equivalence

Table 8: A comparison of public keys and signature sizes with alternative code-based signature schemes

Our balanced optimization criterion, combined with the use of fixed-weight challenges allows us to reduce to less than half the signature size, with respect to the parameter sets proposed by Beullens in [7], while retaining the same public key size. Finally, we are able, at the cost of a larger public key size, to achieve a minimum signature size of 5.25 kB, reducing the signature size by close to $3\times$ with respect to the other parameter sets.

Table 8 reports a comparison of the data sizes achieved by LESS variants with other code-based signature schemes. Considering the algorithms employing Fiat-Shamir, we see that LESS consistently outperforms the traditional schemes based on the Hamming metric (such as Stern, Veron and CVE), even when compared to 80-bit security versions, while its characteristics are orthogonal to those of cRVDC (which is rank-based). Indeed, despite a much larger public key, LESS achieves more compact signatures, and therefore compares favourably in scenarios ii) and iii). Finally, we consider two recent signature algorithms (namely, Wave and Durandal). We observe that LESS also provides favourable figures with respect to Wave, as it features a public key which is smaller by two orders of magnitude, albeit at the cost of an increase of an order of magnitude in signature size. This provides a practical advantage in scenarios where a public key/signature pair is transferred at each communication, such as in the TLS authentication phase, which transmits X.509 certificates. In this regard, the performance of LESS is very similar to that of Durandal, which is an adaptation of the Schnorr's paradigm to the rank metric. It is then worth noting that the closest competition for LESS, in terms of performance, is represented by rank metric schemes, an area which is somewhat further away from traditional coding theory (while strongly related to multivariate cryptography), and, in the case of Durandal, relying on younger, ad-hoc computational assumptions.

References

[1] `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`. 2017.

[2] N. Alamati et al. "Cryptographic Group Actions and Applications". In: *ASIACRYPT*. Springer. 2020, pp. 411–439.

[3] N. Aragon et al. "Durandal: A Rank Metric Based Signature Scheme". In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Y. Ishai and V. Rijmen. Cham: Springer International Publishing, 2019, pp. 728–758.

[4] M. Bardet, A. Otmani, and M. Saeed-Taha. "Permutation Code Equivalence is Not Harder Than Graph Isomorphism When Hulls Are Trivial". In: *IEEE ISIT 2019*. July 2019, pp. 2464–2468.

[5] M. Bellare and G. Neven. "Multi-signatures in the plain public-key model and a general forking lemma". In: *CCS*. 2006, pp. 390–399.

[6] E. Bellini et al. "Improved Veron Identification and Signature Schemes in the Rank Metric". In: *ISIT*. Paris, France, 2019, pp. 1872–1876.

[7] W. Beullens. *Not enough LESS: An improved algorithm for solving Code Equivalence Problems over F_q*. Cryptology ePrint Archive, Report 2020/801.

[8] W. Beullens. "Sigma protocols for MQ, PKP and SIS, and fishy signature schemes". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2020, pp. 183–211.

[9] J.-F. Biasse et al. "LESS is More: Code-Based Signatures Without Syndromes". In: *AFRICACRYPT*. Ed. by A. Nitaj and A. Youssef. Springer, 2020, pp. 45–65.

[10] J. M. Couveignes. "Hard Homogeneous Spaces." In: *IACR Cryptol. ePrint Arch.* 2006 (2006), p. 291.

[11] L. De Feo and S. D. Galbraith. "SeaSign: Compact isogeny signatures from class group actions". In: *EUROCRYPT*. Springer. 2019, pp. 759–789.

[12] T. Debris-Alazard, N. Sendrier, and J.-P. Tillich. "Wave: A new family of trapdoor one-way preimage sampleable functions based on codes". In: *ASIACRYPT*. Springer. 2019, pp. 21–51.

[13] S. M. El Yousfi Alaoui et al. "Code-Based Identification and Signature Schemes in Software". In: *Security Engineering and Intelligence Informatics*. Ed. by A. Cuzzocrea et al. Springer Berlin Heidelberg, 2013, pp. 122–136.

[14] T. Feulner. "The automorphism groups of linear codes and canonical representatives of their semilinear isometry classes." In: *Adv. Math. Commun.* 3.4 (2009), pp. 363–383.

[15] A. Fiat and A. Shamir. "How to prove yourself: Practical solutions to identification and signature problems". In: *CRYPTO*. Springer. 1986, pp. 186–194.

[16] J. Leon. "Computing automorphism groups of error-correcting codes". In: *IEEE Transactions on Information Theory* 28.3 (May 1982), pp. 496–511.

[17] R. Ransom. *Constant-time verification for cut-and-choose-based signatures*. Cryptology ePrint Archive, Report 2020/1184. 2020.

[18] N. Sendrier. "The Support Splitting Algorithm". In: *Information Theory, IEEE Transactions on* (Aug. 2000), pp. 1193–1203.

[19] P. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509.

[20] A. Stolbunov. "Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves". In: *Advances in Mathematics of Communications* 4.2 (2010), p. 215.

A Standard Definitions for Signature Schemes

Definition 3. *A Digital Signature Scheme, or simply Signature Scheme (SS), is a 6-tuple* $(\mathsf{K}, \mathsf{M}, \mathsf{S}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *defined as follows.*

- $\mathsf{K} = \mathsf{K}_{\mathsf{sign}} \times \mathsf{K}_{\mathsf{ver}}$ *is the key space, containing pairs of private/public keys* $(\mathsf{sgk}, \mathsf{vk})$*, respectively the* signing key *and the* verification key*.*
- M *and* S *are, respectively, the* message space *and the* signature space*.*
- KeyGen *is a probabilistic* key-generation algorithm *that takes as input a security parameter* λ *and outputs a keypair* $(\mathsf{sgk}, \mathsf{vk}) \in \mathsf{K}$*.*
- Sign *is a (possibly probabilistic) private* signing algorithm *that receives as input a signing key* $\mathsf{sgk} \in \mathsf{K}_{\mathsf{sign}}$ *and a message* $\boldsymbol{m} \in \mathsf{M}$ *and returns a signature* $\sigma \in \mathsf{S}$*.*
- Ver *is the (deterministic) public* verification algorithm *that receives as input a verification key* $\mathsf{vk} \in \mathsf{K}_{ver}$*, a message* $\boldsymbol{m} \in \mathsf{M}$ *and a signature* $\sigma \in \mathsf{S}$ *and outputs 1, if the signature is recognized as valid, or 0 otherwise.*

Intuitively, a signature scheme is secure if a forger has only a negligible probability of producing a valid signature without knowing the private key. Among the several models described in literature, the one which is usually considered the most desirable, is the chosen-message attack model, in which an attacker is allowed access to an arbitrary number of message/signature pairs of his choosing, via so-called *signing queries*. The resulting security notion is known as *Existential Unforgeability under Chosen-Message Attacks (EUF-CMA)*, and can be formalized as follows.

Definition 4. *An adversary* $\mathcal{A}$ *for SS in the EUF-CMA attack model is a polynomial-time algorithm playing the following attack game:*

1. *Query a key generation oracle to obtain a verification key* vk*. The corresponding signing key* sgk *is kept private and is unknown to* $\mathcal{A}$*.*
2. *Perform a polynomial number of signing queries. In each signing query,* $\mathcal{A}$ *chooses a message* m *and submits it to a signing oracle. The oracle replies with* $\sigma = \mathsf{Sign}_{\mathsf{sgk}}(\boldsymbol{m})$*.*
3. *Output a pair* (m^*, σ^*)*.*

The adversary succeeds if $\mathsf{Ver}_{\mathsf{vk}}(\boldsymbol{m}^*, \sigma^*) = 1$ *and* $\boldsymbol{m}^*$ *had not been queried before. We say that a signature scheme is* EUF-CMA secure *if the probability of success of any adversary* $\mathcal{A}$ *is negligible in the security parameter.*

B Proof of EUF-CMA Security

In here, we show that the LESS signature scheme is EUF-CMA secure. We begin with the following trivial result.

Lemma 1. *Let* M_n *be the set of monomial matrices as defined in Section 2. Then for any* $\boldsymbol{A} \in \mathsf{M}_n$ *and* $\boldsymbol{B} \xleftarrow{\$} \mathsf{M}_n$*,* $\boldsymbol{A}^{-1}\boldsymbol{B}$ *is uniformly distributed over* M_n*.*

The main result is given below.

Theorem 1. *The LESS signature scheme described in Table 1 is existentially unforgeable under adaptive chosen-message attacks, in the random oracle model, under the hardness of the linear code equivalence problem.*

Proof. Let $\mathcal{A}$ be a polynomial-time EUF-CMA adversary for the signature scheme, as defined in Definition 4. $\mathcal{A}$ takes as input a verification key vk, then performs a polynomial number of signing queries, say q_s, and a polynomial number of random oracle queries, say q_r. Eventually, $\mathcal{A}$ outputs a forgery $(\boldsymbol{m}^*, \sigma^*)$, with a certain probability of success p. We now show how to construct an adversary $\mathcal{A}'$ that is able to solve the linear code equivalence problem. $\mathcal{A}'$ will interact with $\mathcal{A}$ and use it as a subroutine, playing the role of the challenger in the EUF-CMA game and simulating correct executions of the LESS protocol, without obviously having access to the private key.

To begin with, $\mathcal{A}'$ is given an instance $(\boldsymbol{G}, \boldsymbol{G}' = \boldsymbol{SGQ})$ of Problem 1, which he sets up as public key in the simulated LESS protocol. $\mathcal{A}'$ will answer signing queries and random oracle queries as described below; to ensure consistency of the simulation, the queries will be tracked with the help of a table T, initially empty, where the calls to the random oracle will be stored as they are answered, in the form of pairs $(input, output)$.

Setup. Set $\boldsymbol{G}_0 = \boldsymbol{G}$ and $\boldsymbol{G}_1 = \boldsymbol{G}'$.

Random Oracle Queries. In a random oracle query, $\mathcal{A}$ submits an input $\boldsymbol{x}$ of the form $(\hat{\boldsymbol{G}}_0, \ldots, \hat{\boldsymbol{G}}_{t-1}, \boldsymbol{m})$ and expects to receive a λ-bit string h. $\mathcal{A}'$ proceeds as follows:

1. Look up $\boldsymbol{x}$ in T. If $(\boldsymbol{x}, h) \in \mathsf{T}$ for some h, return h and halt.
2. Generate uniformly at random a λ-bit string h.
3. Add $(\boldsymbol{x}, h)$ to T.
4. Return h.

Signing Queries. In a signing query, $\mathcal{A}$ submits a message m and expects to receive a valid signature σ for it. $\mathcal{A}'$ proceeds as follows:

1. Generate uniformly at random a λ-bit string h.
2. Generate uniformly at random matrices $\hat{\boldsymbol{Q}}_0, \ldots, \hat{\boldsymbol{Q}}_{t-1}$.
3. Set $\mu_i = \hat{\boldsymbol{Q}}_i$.
4. Return signature $\sigma = (\mu_0, \ldots \mu_{t-1}, h)$.

After that, $\mathcal{A}'$ adjusts his registry of queries by recording the query corresponding to h in table T. More specifically, $\mathcal{A}'$ will parse $h = h_0, \ldots h_{t-1}$, where $h_i \in \{0, 1\}$, then compute $\hat{\boldsymbol{G}}_i = \mathsf{sf}(\boldsymbol{G}_{h_i}\mu_i)$ and finally set h to be the response to the random oracle query with input $(\hat{\boldsymbol{G}}_0, \ldots, \hat{\boldsymbol{G}}_{t-1}, \boldsymbol{m})$. Note that, due to Lemma 1, signatures produced in this way are indistinguishable from authentic signatures, since they follow the exact same distribution.

The simulation halts if, during a signing query, the input to the random oracle had already been queried before, in which case the signing query outputs $\perp$ instead. Note that this can only happen with negligible probability; more precisely, the probability is at most q'/K^t, where $q' = q_s + q_r$ is the total number of queries performed, and K is an upper bound on the probability of finding a collision, i.e. sampling two monomial matrices that lead to linearly equivalent codes (see Proposition 1 of Appendix D)[3].

Once $\mathcal{A}$ has finished performing queries, it will output a forgery $(\boldsymbol{m}^*, \sigma^*)$, where $\sigma^* = (\mu_0^*, \dots \mu_{t-1}^*, h_0^*, \dots h_{t-1}^*)$, that successfully passes verification. At this point, $\mathcal{A}'$ rewinds his tape and plays the simulation again, in the exact same way, except that one of the random oracle queries is answered differently. By the Forking Lemma (see [5]), $\mathcal{A}$ will now output, with non-negligible probability, a forgery $(\boldsymbol{m}', \sigma')$, where $\sigma' = (\mu_0', \dots \mu_{t-1}', h_0', \dots h_{t-1}')$, for the same message $\boldsymbol{m}' = \boldsymbol{m}^*$ and the same random oracle input $(\hat{\boldsymbol{G}}_0, \dots, \hat{\boldsymbol{G}}_{t-1}, \boldsymbol{m})$, such that $\sigma' \neq \sigma^*$. Let j be the index such that $h_j' \neq h_j^*$; then $\mathsf{sf}(\boldsymbol{G}_{h_j'}\mu_j') = \mathsf{sf}(\boldsymbol{G}_{h_j^*}\mu_j^*)$, which means that the monomial matrix $\mu_j^*\mu_j^{-1}$ is a solution to the linear code equivalence problem as desired. $\square$

C Reduction for the MCLE Problem

In this section, we show that the Multiple Codes Linear Equivalence Problem (Problem 2) reduces tightly to the Linear Equivalence Problem (Problem 1).

Theorem 2. *Given an algorithm to solve Problem 2, that runs in time T and succeeds with probability ε, it is possible to solve Problem 1, in time approximately equal to $T + O(rn^3)$, with probability of success equal to $\varepsilon/2$.*

Proof. Let $\mathcal{A}$ be an adversary for Problem 2. We now show how to construct an adversary $\mathcal{A}'$ that is able to solve the linear code equivalence problem. $\mathcal{A}'$ will interact with $\mathcal{A}$ and use it as a subroutine. To begin, $\mathcal{A}'$ is given an instance $(\boldsymbol{G}, \boldsymbol{G}' = \boldsymbol{SGQ})$ of Problem 1. It will then proceed to generate $r = 2^\ell$ equivalent codes, in the following way. First, $\mathcal{A}'$ samples uniformly at random matrices $\boldsymbol{S}_0, \dots, \boldsymbol{S}_{r-1}$ and $\boldsymbol{Q}_0, \dots, \boldsymbol{Q}_{r-1}$. Then, it computes half of the codes starting from $\boldsymbol{G}$, and half starting from $\boldsymbol{G}'$; wlog, we can imagine that $\boldsymbol{G}_i$ is generated as $\boldsymbol{S}_i\boldsymbol{GQ}_i$ when $i \in [0; r/2 - 1]$, and as $\boldsymbol{S}_i\boldsymbol{GQ}_i$ when $i \in [r/2; r - 1]$ (and then reordered). It is clear that this computation can be done in polynomial time, at most $O(rn^3)$, and that there is no way to distinguish how an individual matrix was generated (i.e. from $\boldsymbol{G}$ rather than $\boldsymbol{G}'$). At this point $\mathcal{A}'$ runs $\mathcal{A}$ on input $\boldsymbol{G}_0, \dots, \boldsymbol{G}_{r-1}$, and $\mathcal{A}$ will output, with probability ε, a response $(\boldsymbol{S}^*, \boldsymbol{Q}^*)$ such that $\boldsymbol{G}_{j'} = \boldsymbol{S}^*\boldsymbol{G}_j\boldsymbol{Q}^*$. Now, if one of the two matrices was of the first type, and the other of the second type, $\mathcal{A}'$ is able to win. For instance, if $\boldsymbol{G}_j = \boldsymbol{S}_j\boldsymbol{GQ}_j$ and $\boldsymbol{G}_{j'} = \boldsymbol{S}_{j'}\boldsymbol{G}'\boldsymbol{Q}_{j'}$, then it must be $\boldsymbol{Q}^* = \boldsymbol{Q}_j^{-1}\boldsymbol{QQ}_{j'}$, which immediately reveals[4] $\boldsymbol{Q}$. Since this happens with probability $1/2$, we get the thesis. $\square$

[3] If two monomials $\boldsymbol{Q}$ and $\boldsymbol{Q}'$ are such that the codes generated by $\boldsymbol{GQ}$ and $\boldsymbol{GQ}'$ are equivalent, then there exists $\boldsymbol{S} \in \mathsf{GL}_k$ such that $\boldsymbol{G} = \boldsymbol{SGQ}'\boldsymbol{Q}^{-1}$, implying that $\boldsymbol{QQ}'^{-1}$ is an automorphism for the code generated by $\boldsymbol{G}$.

[4] If needed, $\boldsymbol{S}$ can then be found in polynomial time also.

D The Automorphism Group of a Random Code

We now derive an estimate on the size of the automorphism group of a random linear code, and use it to derive an upper bound on the probability that applying a random monomial (or permutation) returns an equivalent code. We anticipate the main result, and then proceed by proving it.

Proposition 1. *Let $\mathfrak{C} \subseteq \mathbb{F}_q^n$ be a random linear code with dimension k. Let d_{GV} denote the GV distance of $\mathfrak{C}$, and $N_{d_{GV}} = \left\lceil \binom{n}{d_{GV}}(q-1)^{d_{GV}-2}q^{k-n+1} \right\rceil$. Let $d_{GV}^{\perp}$ be the GV distance of $\mathfrak{C}^{\perp}$, and $N_{d_{GV}^{\perp}} = \left\lceil \binom{n}{d_{GV}^{\perp}}(q-1)^{d_{GV}^{\perp}-2}q^{-k+1} \right\rceil$. The probability that $\pi \xleftarrow{\$} \mathsf{S}_n$ is in the permutations automorphism group of $\mathfrak{C}$, i.e., $\pi(\mathfrak{C}) = \mathfrak{C}$, is not greater than*

$$(q-1)\min\left\{ \frac{N_{d_{GV}}!}{\binom{n}{d_{GV}!}} \, , \, \frac{N_{d_{GV}}^{\perp}!}{\binom{n}{d_{GV}^{\perp}!}} \right\},$$

while the probability that $\mu \xleftarrow{\$} \mathsf{M}_n$ is in the monomials automorphism group of $\mathfrak{C}$, i.e., $\mu(\mathfrak{C}) = \mathfrak{C}$, is not greater than

$$\min\left\{ \frac{N_{d_{GV}}!(q-1)^{-d_{GV}+1}}{\binom{n}{d_{GV}!}} \, , \, \frac{N_{d_{GV}}^{\perp}!(q-1)^{-d_{GV}^{\perp}+1}}{\binom{n}{d_{GV}^{\perp}!}} \right\}.$$

D.1 Proof for the Permutations Automorphism Group

To derive a bound on the size of the automorphism group of a code, we will consider the action of permutations on the set of minimum weight codewords. To this end, we first derive some preliminary results.

Lemma 2. *Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^n$ with the same Hamming weight d and same entries multisets. Let $\mathsf{Mor}_{\mathsf{S}_n}(\boldsymbol{a}, \boldsymbol{b}) = \{\pi \in \mathsf{S}_n \mid \pi(\boldsymbol{a}) = \boldsymbol{b}\}$. Then, the cardinality of $\mathsf{Mor}_{\mathsf{S}_n}(\boldsymbol{a}, \boldsymbol{b})$ is not greater than $w!(n-w)!$.*

Proof. Let $E = \{i \in [0; n-1] \mid a_i = 0\}$. For a permutation π, we can have $\pi(i) = j$ if and only if $a_i = b_j$. Let m_x, for $x \in \mathbb{F}_q$, be the number of entries with value equal to x in both $\boldsymbol{a}$ and $\boldsymbol{b}$; since $\boldsymbol{a}$ and $\boldsymbol{b}$ have Hamming weight w, it holds that $m_0 = n - w$ and $\sum_{x \in \mathbb{F}_q^*} m_x = w$. Then, we have

$$|\mathsf{Mor}_{\mathsf{S}_n}(\boldsymbol{a}, \boldsymbol{b})| = \prod_{x \in \mathbb{F}_q} m_x! = (n-w)! \prod_{x \in \mathbb{F}_q^*} m_x!.$$

It is immediately seen that $\prod_{x \in \mathbb{F}q^*} m_x! \leq \left(\sum_{x \in \mathbb{F}_q^*} m_x\right)! = w!$, so that as an upper bound on the size of $\mathsf{Mor}_{\mathsf{S}_n}(\boldsymbol{a}, \boldsymbol{b})$ we can use $(n-w!)w!$. □

Lemma 3. *Let $A \subseteq \mathbb{F}_q^n$, with cardinality M, such that all the contained vectors have Hamming weight w. Let $\mathsf{Aut}_{\mathsf{S}_n}(A) = \{\pi \in \mathsf{S}_n \mid \pi(\boldsymbol{a}) \in A, \ \forall \boldsymbol{a} \in A\}$; then, the size of $\mathsf{Aut}_{\mathsf{S}_n}(A)$ is not greater than $M!w!(n-w)!$.*

Proof. If $\pi \in \mathsf{Aut}_{\mathsf{S}_n}(A)$, then for each $\boldsymbol{a} \in A$, either $\pi(\boldsymbol{a}) = \boldsymbol{a}$ or there exists $\boldsymbol{a}' \in A$, $\boldsymbol{a}' \neq \boldsymbol{a}$, such that $\pi(\boldsymbol{a}) = \boldsymbol{a}'$. Let us define some order for the elements of A and write $A = \left\{\boldsymbol{a}^1, \boldsymbol{a}^2, \cdots, \boldsymbol{a}^M\right\}$. For each $\pi \in \mathsf{Aut}_{\mathsf{S}_n}(A)$, there exists one and only one bijection $f : \{1, \cdots, M\} \mapsto \{1, \cdots, M\}$ such that $f(i) = j$ if and only if $\pi(\boldsymbol{a}^i) = \boldsymbol{a}^j$. On the contrary, for a fixed bijection f, we may have more than one valid permutation, i.e., a permutation that places i in position j if and only if $f(i) = j$. It is easily seen that, for a bijection f, the set of all valid permutations is obtained as follows $\mathsf{Aut}_{\mathsf{S}_n}^{(f)}(A) = \bigcap_{i=1}^{M} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)}\right)$. Each bijection f can be seen as an element of the symmetric group on M elements (which we denote as S_M), so that the number of possible bijections is given by $M!$. Notice that, if $\pi \in \mathsf{Aut}_{\mathsf{S}_n}^{(f)}(A)$, then it is also in $\mathsf{Aut}_{\mathsf{S}_n}(A)$: hence, $\mathsf{Aut}_{\mathsf{S}_n}(A)$ corresponds to the union of all sets $\mathsf{Aut}_{\mathsf{S}_n}^{(f)}(A)$, that is

$$\mathsf{Aut}_{\mathsf{S}_n}(A) = \bigcup_{f \in \mathsf{S}_M} \mathsf{Aut}_{\mathsf{S}_n}^{(f)}(A) = \bigcup_{f \in \mathsf{S}_M} \left(\bigcap_{i=1}^{M} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)}\right) \right).$$

We are now able to derive an upper bound on the size of $\mathsf{Aut}_{\mathsf{S}_n}(A)$, as follows

$$\begin{aligned} |\mathsf{Aut}_{\mathsf{S}_n}(A)| &= \left| \bigcup_{f \in \mathsf{S}_M} \left(\bigcap_{i=1}^{M} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)}\right) \right) \right| \leq |\mathsf{S}_M| \cdot \left| \bigcap_{i=1}^{M} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)}\right) \right| \\ &= M! \cdot \left| \bigcap_{i=1}^{M} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)}\right) \right| \leq M!w!(n-w!), \end{aligned}$$

where the last inequality comes from Lemma 2. □

Using the previous results, we prove the following theorem.

Theorem 3. *Let $\mathfrak{C} \subseteq \mathbb{F}_q^n$ be a linear code with minimum distance d. Let $T_q(\boldsymbol{c}) = \left\{b\boldsymbol{c} \mid b \in \mathbb{F}_q^*\right\}$, and let $V \subset \mathbb{F}_q$ be the set of N_d codewords such that*

i) if $\boldsymbol{c} \in \mathfrak{C}$ has weight d, then $T_q(\boldsymbol{c})$ and V have only one element in common;
ii) all codewords in V have weight d.

Let $\mathsf{Aut}_{\mathsf{S}_n}(\mathfrak{C})$ be the permutations automorphism group of $\mathfrak{C}$. Then, the cardinality of $\mathsf{Aut}_{\mathsf{S}_n}(\mathfrak{C})$ is not greater than $(N_d)!(q-1)d!(n-d)!$.

Proof. Without loss of generality, we can define V such that all of its codewords have the first entry that is equal to 1. Now, let $\pi \in \mathsf{Aut}_{\mathsf{S}_n}(\mathfrak{C})$; then, π must map the set of codewords of $\mathfrak{C}$ with weight d into itself. Since this set is obtained as $V_q = \bigcup_{\boldsymbol{c} \in V} T_q(\boldsymbol{c})$, we have that the image of V_q under the permutation π is equal to itself. Hence, for each $\boldsymbol{c} \in \mathfrak{C}$ with weight d, there must be $\boldsymbol{c}' \in V$ such that

$\pi(\boldsymbol{c}) \in T_q(\boldsymbol{c}')$. Note that this also guarantees that, for each $\hat{\boldsymbol{c}} \in T_q(\boldsymbol{c})$, one also has $\pi(\hat{\boldsymbol{c}}) \in T_q(\boldsymbol{c}')$. To put it differently, for each $\boldsymbol{c} \in V$ there must exist i) another codeword $\boldsymbol{c}' \in V$, and ii) a non null element $b \in \mathbb{F}_q^*$, such that $\pi(\boldsymbol{c}) = b\boldsymbol{c}'$. Hence, we have

$$\mathsf{Aut}_{\mathsf{S}_n}(V_q) = \bigcup_{f \in \mathsf{S}_{N_d}} \bigcup_{b \in \mathbb{F}_q^*} \left(\bigcap_{i=1}^{N_d} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{c}^i, b\boldsymbol{c}^{f(i)}\right) \right).$$

This allows us to derive a bound on the size of $\mathsf{Aut}_{\mathsf{S}_n}(V_q)$, using the union bound for two times

$$\begin{aligned} |\mathsf{Aut}_{\mathsf{S}_n}(V)| &= \left| \bigcup_{f \in \mathsf{S}_{N_d}} \bigcup_{b \in \mathbb{F}_q^*} \left(\bigcap_{i=1}^{N_d} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{c}^i, b\boldsymbol{c}^{f(i)}\right) \right) \right| \\ &\leq |\mathsf{S}_{N_d}| \cdot \left|\mathbb{F}_q^*\right| \cdot \left| \bigcap_{i=1}^{N_d} \mathsf{Mor}_{\mathsf{S}_n}\left(\boldsymbol{c}^i, b\boldsymbol{c}^{f(i)}\right) \right| \leq N_d!(q-1)d!(n-d)!. \end{aligned}$$

Finally, we consider that if $\pi \in \mathsf{Aut}_{\mathsf{S}_n}(\mathfrak{C})$, then it must necessarily be $\pi \in \mathsf{Aut}_{\mathsf{S}_n}(V_q)$: hence, it must be $\mathsf{Aut}_{\mathsf{S}_n}(\mathfrak{C}) \subseteq \mathsf{Aut}_{\mathsf{S}_n}(V_q)$. So, we can use the bound on the cardinality of $\mathsf{Aut}_{\mathsf{S}_n}(V_q)$ as an upper bound for the size of $\mathsf{Aut}_{\mathsf{S}_n}(\mathfrak{C})$. □

The above results allow to prove the bound on the permutations automorphism group stated in Proposition 1. To estimate the minimum distance of a code, we use the well known Gilbert-Varshamov bound, and estimate the number of weight w codewords (without counting scalar multiples) as $\left\lceil \binom{n}{w}(q-1)^{w-2}q^{k-n+1} \right\rceil$. We then divide the upper bound on the size of the automorphism group resulting from Lemma 3 by the cardinality of S_n (that is, $n!$). Finally, we consider that the automorphism group of a code coincides with that of its dual: we repeat the reasoning for the dual code and, take the minimum between the two obtained probabilities (i.e., the one for the code and that for its dual).

D.2 Proof for the Monomials Automorphism Group

We now generalize the results in the previous section to the case of monomials.

Lemma 4. *Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^n$ with the same Hamming weight d and same entries multisets. Let $\mathsf{Mor}_{\mathsf{M}_n}(\boldsymbol{a}, \boldsymbol{b}) = \{\mu \in \mathsf{M}_n \mid \mu(\boldsymbol{a}) = \boldsymbol{b}\}$. Then, the cardinality of $\mathsf{Mor}_{\mathsf{M}_n}(\boldsymbol{a}, \boldsymbol{b})$ is equal to $w!(n-w)!(q-1)^{n-w}$.*

Proof. We reason on the characteristics that a monomial $\mu \in \mathsf{M}_n$ must have, in order to guarantee that the image of $\boldsymbol{a}$ is $\boldsymbol{b}$. To this end, we write $\mu = \pi \rtimes \boldsymbol{v}$, with $\pi \in \mathsf{S}_n$ and $\boldsymbol{v} \in \mathbb{F}_q^{*n}$. Let $E(\boldsymbol{a})$ be the set of positions pointing at null entries in $\boldsymbol{a}$, and let $\bar{E}(\boldsymbol{a})$ be that of indexes pointing an non-null entries in $\boldsymbol{a}$; the same notation is employed for $\boldsymbol{b}$. To have $\mu(\boldsymbol{a}) = \boldsymbol{b}$, the following conditions must be verified:

i) $\pi\left(E(\boldsymbol{a})\right) = E(\boldsymbol{b})$;
ii) if $i \in E$, then v_i can have whichever value;
iii) $\pi\left(\bar{E}(\boldsymbol{a})\right) = \bar{E}(\boldsymbol{b})$;
iv) if $\pi(i) = j$, then $v_i = a_i^{-1} b_j$.

The number of permutations satisfying conditions i) and iii) is given by $w!(n-w)!$, while that of vectors satisfying ii) and iv) corresponds to $(q-1)^{n-w}$. □

Lemma 5. *Let $A \subseteq \mathbb{F}_q^n$, with cardinality M, such that all the contained vectors have Hamming weight w. Let $\mathsf{Aut}_{\mathsf{S}_n}(A) = \{\pi \in \mathsf{S}_n \mid \pi(\boldsymbol{a}) \in A, \ \forall \boldsymbol{a} \in A\}$; then, the size of $\mathsf{Aut}_{\mathsf{M}_n}(A)$ is not greater than $M!w!(n-w)!(q-1)^{n-w}$.*

Proof. We reason as in the proof of Proposition 3. If $\mu \in \mathsf{Aut}_{\mathsf{M}_n}(A)$, then for each $\boldsymbol{a} \in A$, either $\mu(\boldsymbol{a}) = \boldsymbol{a}$ or there exists $\boldsymbol{a}' \in A$, $\boldsymbol{a}' \neq \boldsymbol{a}$, such that $\mu(\boldsymbol{a}) = \boldsymbol{a}'$. We write again $A = \left\{\boldsymbol{a}^1, \boldsymbol{a}^2, \cdots, \boldsymbol{a}^M\right\}$, and consider that for each $\mu \in \mathsf{Aut}_{\mathsf{M}_n}(A)$, there exists one and only one bijection $f : \{1, \cdots, M\} \mapsto \{1, \cdots, M\}$ such that $f(i) = j$ if and only if $\mu(\boldsymbol{a}^i) = \boldsymbol{a}^j$. Let $\mathsf{Aut}_{\mathsf{M}_n}^{(f)}(A) = \bigcap_{i=1}^{M} \mathsf{Mor}_{\mathsf{M}_n}\left(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)}\right)$, and consider that

$$\mathsf{Aut}_{\mathsf{M}_n}(A) = \bigcup_{f \in \mathsf{S}_M} \mathsf{Aut}_{\mathsf{M}_n}^{(f)}(A) = \bigcup_{f \in \mathsf{S}_M} \left(\bigcap_{i=1}^{M} \mathsf{Mor}_{\mathsf{M}_n}\left(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)}\right) \right).$$

Using the union bound, we find that the cardinality of $\mathsf{Aut}_{\mathsf{M}_n}(A)$ cannot be greater than $M! \left|\mathsf{Mor}_{\mathsf{M}_n}(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)})\right|$, and we finally rely on Lemma 4 to bound the cardinality of $\mathsf{Mor}_{\mathsf{M}_n}(\boldsymbol{a}^i, \boldsymbol{a}^{f(i)})$. □

Finally, we adapt Theorem 3 to the case of monomials.

Theorem 4. *Let $\mathfrak{C} \subseteq \mathbb{F}_q^n$ be a linear code with minimum distance d. Let $T_q(\boldsymbol{c}) = \left\{b\boldsymbol{c} \mid b \in \mathbb{F}_q^*\right\}$, and let $V \subset \mathbb{F}_q$ be the set of N_d codewords such that*

i) if $\boldsymbol{c} \in \mathfrak{C}$ has weight d, then $T_q(\boldsymbol{c})$ and V have only one element in common;
ii) all codewords in V have weight d.

Let $\mathsf{Aut}_{\mathsf{M}_n}(\mathfrak{C})$ be the monomials automorphism group of $\mathfrak{C}$. Then, the cardinality of $\mathsf{Aut}_{\mathsf{M}_n}(\mathfrak{C})$ is not greater than $(N_d)!(q-1)d!(n-d)!$.

Proof. As in the proof of Theorem 3, we define V such that all of its codewords have the first entry that is equal to 1, and $V_q = \bigcup_{\boldsymbol{c} \in V} T_q(\boldsymbol{c})$. If $\mu(V_q) = V_q$, then for each $\boldsymbol{c} \in V$ there must exist i) another codeword $\boldsymbol{c}' \in V$, and ii) a non null element $b \in \mathbb{F}_q^*$, such that $\pi(\boldsymbol{c}) = b\boldsymbol{c}'$. Then, we have

$$\mathsf{Aut}_{\mathsf{M}_n}\left(V_q\right) = \bigcup_{f \in \mathsf{S}_{N_d}} \bigcup_{b \in \mathbb{F}_q^*} \left(\bigcap_{i=1}^{N_d} \mathsf{Mor}_{\mathsf{M}_n}\left(\boldsymbol{c}^i, b\boldsymbol{c}^{f(i)}\right) \right).$$

Using twice the union bound, we find that an upper bound on the size of $\mathsf{Aut}_{\mathsf{M}_n}\left(V_q\right)$ is given by $N_d!(q-1)w!(n-w)!(q-1)^{n-w}$. Again, the proof is completed by noticing that $\mathsf{Aut}_{\mathsf{M}_n}\left(\mathfrak{C}\right) \subseteq \mathsf{Aut}_{\mathsf{M}_n}\left(V_q\right)$. □