Take your MEDS: Digital Signatures from Matrix Code Equivalence

Tung Chou[1], Ruben Niederhagen[1,2], Edoardo Persichetti[3], Tovohery Hajatiana Randrianarisoa[4], Krijn Reijnders[5], Simona Samardjiska[5], and Monika Trimoska[5]

[1] Academia Sinica, Taipei, Taiwan
[2] University of Southern Denmark, Odense, Denmark
[3] Florida Atlantic University, Boca Raton, USA
[4] Umea University, Umea, Sweden
[5] Radboud Universiteit, Nijmegen, The Netherlands
`blueprint@crypto.tw`, `ruben@polycephaly.org`, `epersichetti@fau.edu`, `tovo@aims.ac.za`, `{krijn,simonas,mtrimoska}@cs.ru.nl`

Abstract. In this paper, we show how to use the Matrix Code Equivalence (MCE) problem as a new basis to construct signature schemes. This extends previous work on using isomorphism problems for signature schemes, a trend that has recently emerged in post-quantum cryptography. Our new formulation leverages a more general problem and allows for smaller data sizes, achieving competitive performance and great flexibility. Using MCE, we construct a zero-knowledge protocol which we turn into a signature scheme named Matrix Equivalence Digital Signature (MEDS). We provide an initial choice of parameters for MEDS, tailored to NIST's Category 1 security level, yielding public keys as small as 2.8 kB and signatures ranging from 18 kB to just around 6.5 kB, along with a reference implementation in C.

Keywords: group action · signature scheme · code-based cryptography · post-quantum cryptography· matrix codes

1 Introduction

Post-Quantum Cryptography (PQC) comprises all the primitives that are believed to be resistant against attackers equipped with a considerable quantum computing power. Several such schemes have been around for a long time [49, 44], some being in fact almost as old as RSA [40]; however, the area itself was not formalized as a whole until the early 2000s, for instance with the first edition of the PQCrypto conference [42]. The area has seen a dramatic increase in importance and volume of research over the past few years, partially thanks to NIST's interest and the launch of the PQC Standardization process in 2017 [43]. After 4 years and 3 rounds of evaluation, the process has crystallized certain mathematical tools as standard building blocks (e.g. lattices, linear codes, multivariate equations, isogenies etc.). Some algorithms [39, 47, 36] have now been selected for standardization, with an additional one or two to be selected among a restricted set of alternates [5, 4, 1] after another round of evaluation. While having a range of candidates ready for standardization may seem satisfactory, research is still active in designing PQC primitives. In particular, NIST has expressed the desire for a greater diversity among the hardness assumptions behind signature schemes, and announced a partial re-opening of the standardization process for precisely the purpose of collecting non-lattice-based protocols.

Cryptographic group actions are a popular and powerful instrument for constructing secure and efficient cryptographic protocols. The most well-known is, without a doubt, the action of finite groups on the integers modulo a prime, or the set of points on an elliptic curve, which give rise to the *Discrete Logarithm Problem (DLP)*, i.e. the backbone of public-key cryptography. Recently, proposals for post-quantum cryptographic group actions started to emerge, based on the tools identified above: for instance, isogenies [21], linear codes [17], trilinear forms [52] and even lattices [35]. All of these group actions provide very promising solutions for cryptographic schemes, for example signatures [26, 10, 52],ring signatures [15, 11] and many others; at the same time, they are very different in nature, with unique positive and negative aspects.

Our Contribution. In this work, we formalize a new cryptographic group action based on the notion of *Matrix Code Equivalence.* This is similar in nature to the *code equivalence* notion at the basis of LESS [17, 10], and in fact belongs to a larger class of isomorphism problems that include, for example, the lattice isomorphism problem, and the well-known isomorphism of polynomials [44]. The hardness of the MCE problem was studied in [25, 50], from which it is possible to conclude that this is a suitable problem for post-quantum cryptography. Indeed, we show that it is possible to use MCE to build a zero-knowledge protocol, and hence a signature scheme, which we name *Matrix Equivalence Digital Signature*, or simply MEDS. We provide an initial parameter choice, together with several computational optimizations, resulting in a scheme with great flexibility and very competitive data sizes. Furthermore, we show that this group action allows for the construction of (linkable) ring signatures, with performance results that improve on the existing state of the art [11].

2 Preliminaries

Let $\mathbb{F}_q$ be the finite field of q elements. $\mathrm{GL}_n(q)$ and $\mathrm{AGL}_n(q)$ denote respectively the general linear group and the general affine group of degree n over $\mathbb{F}_q$. We use bold letters to denote vectors $\mathbf{a}, \mathbf{c}, \mathbf{x}, \dots$, and matrices $\mathbf{A}, \mathbf{B}, \dots$. The entries of a vector $\mathbf{a}$ are denoted by a_i, and we write $\mathbf{a} = (a_1, \dots, a_n)$ for a (row) vector of dimension n over some field. Similarly, the entries of a matrix $\mathbf{A}$ are denoted by a_{ij}. Random sampling from a set S is denoted by $a \xleftarrow{\$} S$. For two matrices $\mathbf{A}$ and $\mathbf{B}$, we denote the Kronecker product by $\mathbf{A} \otimes \mathbf{B}$. Finally, we denote the set of all $m \times n$ matrices over $\mathbb{F}_q$ by $\mathcal{M}_{m,n}(\mathbb{F}_q)$.

2.1 Cryptographic group actions

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

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

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

A group action can have a number of mathematically desirable properties. For example, we say that a group action is:

- *Commutative*: for any $g_1, g_2 \in G$, we have $g_2 \star (g_1 \star x) = g_1 \star (g_2 \star x)$.
- *Transitive*: given $x_1, x_2 \in X$, there is some $g \in G$ such that $g \star x_1 = x_2$.
- *Free*: if $g \star x = x$, then g is the identity.

In particular, a *cryptographic* group action is a group action with some additional properties that are useful for cryptographic applications. To begin with, there are some desirable properties of computational nature. Namely, the following procedures should be efficient:

- *Evaluation*: given x and g, compute $g \star x$.
- *Sampling*: sample uniformly at random from G.
- *Membership testing*: verify that $x \in X$.

Finally, cryptographic group actions should come with security guarantees; for instance, the *vectorization problem* should be hard:

Problem 1 (Group Action Vectorization).
Given: The pair $x_1, x_2 \in X$.
Goal: Find, if any, $g \in G$ such that $g \star x_1 = x_2$.

Early constructions using this paradigm are based on the action of finite groups of prime order, for which the vectorization problem is the discrete logarithm problem. Lately, multiple isogeny-based constructions have appeared: see, for instance, the work of Couveignes in [24] and later by Rostovtsev and Stolbunov [51]. A general framework based on group actions was explored in more detail by [3], allowing for the design of several primitives. The holy grail are those cryptographic group actions that possess both the mathematical and cryptographic properties listed above. Currently, CSIDH [21] is the only post-quantum commutative cryptographic group action, although there is an ongoing debate about the efficiency and quantum hardness of its vectorization problem [18]. In Section 3, we introduce the group action that is relevant to our work.

2.2 Protocols

We give here an explicit characterization of the protocols we will build. The corresponding security definitions are presented only in an informal manner; formal definitions will be included in the full version of this work.

Definition 2. A *Sigma protocol* is a three-pass interactive protocol between two parties: a prover $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ and a verifier $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$. The protocol is composed of the following procedures:

I. Keygen: on input some public data (including system parameters), output a public key pk (the instance) and the corresponding secret key sk (the witness). Give sk to the prover; pk is distributed publicly and is available to all parties. For simplicity, we assume that the public data is available as input in all the remaining procedures.

II. Commit: on input the public key pk, P_1 outputs a public commitment cmt and sends it to the verifier.

III. Challenge: on input the public key pk and the commitment cmt, V_1 samples uniformly at random a challenge ch from the challenge space C and sends it to the prover.

IV. Response: on input the secret key sk, the public key pk, the commitment cmt and the challenge ch, P_2 outputs a response rsp and sends it to the verifier.

V. Verify: on input a public key pk, the commitment cmt, the challenge ch, and the response rsp, V_2 outputs either 1 (accept) if the *transcript* $(\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp})$ is valid, or 0 (reject) otherwise.

A Sigma protocol is usually required to satisfy the following properties. First, if the statement is true, an honest prover is always able to convince an honest verifier. This property is called *Completeness*. Secondly, a dishonest prover cannot convince an honest verifier other than with a small probability. This is captured by the *Soundness* property, which also bounds such probability, usually known as *soundness error* or, informally, *cheating probability*. Finally, the protocol has to be *Zero-Knowledge*, i.e. anyone observing the transcript (including the verifier) learns nothing other than the fact that the statement is true.

Definition 3. A *Digital Signature scheme* is a protocol between 2 parties: a signer S and a verifier V. The protocol is composed of the following procedures:

I. Keygen: on input the public data (including system parameters), output a secret signing key sk for S and the corresponding public verification key pk.

II. Sign: on input a secret key sk and a message msg, output a signature σ.

III. Verify: on input a public key pk, a message msg and a signature σ, V outputs either 1 (accept) if the signature is valid, or 0 (reject) otherwise.

Correctness means that an honest signer is always able to get verified. The usual desired security notion for signature schemes is *Unforgeability*, which guarantees computationally infeasible to forge a valid signature without knowing the secret signing key.

Definition 4. A *Ring Signature scheme* is a protocol between $r + 1$ parties: potential signers $\mathsf{S}_1, \ldots, \mathsf{S}_r$ (the *ring*) and a verifier V. The protocol is composed of the following procedures:

I. Keygen: on input the public data (including system parameters), output a secret key sk_j for each signer S_j, and the corresponding public key pk_j.
II. Sign: on input a set of public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_r\}$, a secret key sk_{j^*} and a message msg, output a signature σ.
III. Verify: on input a set of public keys $\{\mathsf{pk}_1, \ldots, \mathsf{pk}_r\}$, a message msg and a signature σ, V outputs either 1 (accept) if the signature is valid, or 0 (reject) otherwise.

A crucial feature of a ring signature scheme is that anyone in the ring can produce a valid signature (on behalf on the entire ring), and the verifier can check validity, but cannot learn the identity of the signer. This is achieved by requiring that the scheme satisfies the *Anonymity* property. This captures the idea of protecting the identity of the signer, i.e. it should be impossible for a verifier to tell which secret key was used to sign. Then, *Unforgeability* corresponds to the familiar security notion for signature schemes presented above, extended to include all ring participants. In other words, it should be infeasible to forge a valid signature without knowing at least one of the secret keys in the ring. In addition, *Correctness* is also required, as before.

Linkable ring signature schemes possess additional security features. These protocols are composed of the same three procedures described in Definition 4, plus a fourth one, given below:

IV. Link: on input two signatures σ and σ', output either 1 if the signatures were produced with the same secret key, or 0 otherwise.

A linkable ring signature scheme is required to satisfy the following security properties. *Linkability* asks that it is computationally infeasible for an adversary to produce more than r unlinked valid signatures, even if some or all of the r public keys are malformed. *Linkable Anonymity* guarantees that, even if an adversary is able to obtain multiple signatures from the same signer, they are still unable to determine which secret key was used. *Non-Frameability* protects the user's identity by requiring that it is computationally infeasible for an adversary to produce a valid signature linked to one produced by an honest party.

Remark 1. Note that the linkability property is usually formulated in an alternative way, that is, if more than r valid signatures are produced, then the Link algorithm will output 1 on at least two of them. It is then easy to see, as also pointed out in [15, Remark 2.4], that unforgeability is obtained directly by satisfying linkability and non-frameability.

Finally, we recall the definition of *commitment scheme*, which is a tool necessary for our particular construction of ring signatures. This is a non-interactive function $\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{2\lambda}$ mapping a message string of arbitrary length to a commitment string of 2λ bits. The first λ bits of input, chosen uniformly at random, guarantee that the input message is hidden in a very strong sense, as captured in the next definition.

Definition 5. Given an adversary A, we define the two following quantities:

$$\mathsf{Adv}^{\mathsf{Bind}}(\mathsf{A}) = \mathsf{Pr}\Big[\mathsf{Com}(\mathsf{r}, \mathbf{x}) = \mathsf{Com}(\mathsf{r}', \mathbf{x}') \mid (\mathsf{r}, \mathbf{x}, \mathsf{r}', \mathbf{x}') \leftarrow \mathsf{A}(1^{\lambda})\Big];$$

$$\mathsf{Adv}^{\mathsf{Hide}}(\mathsf{A}, \mathbf{x}, \mathbf{x}') = \Big| \underset{\mathsf{r} \leftarrow \{0,1\}^{\lambda}}{\mathsf{Pr}}\Big[\mathsf{A}(\mathsf{Com}(\mathsf{r}, \mathbf{x})) = 1\Big] - \underset{\mathsf{r} \leftarrow \{0,1\}^{\lambda}}{\mathsf{Pr}}\Big[\mathsf{A}(\mathsf{Com}(\mathsf{r}, \mathbf{x}')) = 1\Big]\Big|.$$

We say that Com is *computationally binding* if, for all polynomial-time adversaries A, the quantity $\mathsf{Adv}^{\mathsf{Bind}}(\mathsf{A})$ is negligible in λ. We say that Com is *computationally hiding* if, for all polynomial-time adversaries A and every pair $(\mathbf{x}, \mathbf{x}')$, the quantity $\mathsf{Adv}^{\mathsf{Hide}}(\mathsf{A})$ is negligible in λ.

3 The Matrix Code Equivalence Problem

A $[m \times n, k]$ *matrix code* is a subspace $\mathcal{C}$ of $\mathcal{M}_{m,n}(\mathbb{F}_q)$. These objects are usually measured with the rank metric, where the *distance* between two matrices $\mathbf{A}, \mathbf{B} \in \mathcal{M}_{m,n}(\mathbb{F}_q)$ is defined as $d(\mathbf{A}, \mathbf{B}) = \mathrm{Rank}(\mathbf{A} - \mathbf{B})$. We denote the basis of the subspace by $\langle \mathbf{C_1}, \ldots, \mathbf{C_k} \rangle$, where the $\mathbf{C_i}$'s are linearly independent elements of $\mathcal{M}_{m,n}(\mathbb{F}_q)$. Due to symmetry, without loss of generality, in the rest of the text we will assume $m \leqslant n$.

For a matrix $\mathbf{A} \in \mathcal{M}_{m,n}(\mathbb{F}_q)$, let vec be a mapping that sends a matrix $\mathbf{A}$ to the vector $\mathsf{vec}(\mathbf{A}) \in \mathbb{F}_q^{mn}$ obtained by 'flattening' $\mathbf{A}$, i.e.:

$$\mathsf{vec}: \ \mathbf{A} = \begin{pmatrix} a_{1,1} & \ldots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \ldots & a_{m,n} \end{pmatrix} \mapsto \mathsf{vec}(\mathbf{A}) = (a_{1,1}, \ldots, a_{1,n}, \ldots, a_{m,1}, \ldots, a_{m,n}).$$

The inverse operation is denoted by mat, i.e. $\mathsf{mat}(\mathsf{vec}(\mathbf{A})) = \mathbf{A}$. Using the map vec, an $[m \times n, k]$ matrix code can be thought of as an $\mathbb{F}_q$-subspace of $\mathbb{F}_q^{mn}$, and thus we can represent it with a generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times mn}$, in a manner similar to the common representation for linear codes. Indeed, if $\mathcal{C}$ is an $[m \times n, k]$ matrix code over $\mathbb{F}_q$, we denote by $\mathsf{vec}(\mathcal{C})$ the vectorization of $\mathcal{C}$ i.e.:

$$\mathsf{vec}(\mathcal{C}) := \{\mathsf{vec}(\mathbf{A})\colon \mathbf{A} \in \mathcal{C}\}.$$

In this case, $\mathsf{vec}(\mathcal{C})$ is a k-dimensional $\mathbb{F}_q$-subspace of $\mathbb{F}_q^{mn}$.

Definition 6. Let $\mathcal{C}$ and $\mathcal{D}$ be two $[m \times n, k]$ matrix codes over $\mathbb{F}_q$. We say that $\mathcal{C}$ and $\mathcal{D}$ are *equivalent* if there exist two matrices $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$ such that $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$, i.e. for all $\mathbf{C} \in \mathcal{C}$, $\mathbf{ACB} \in \mathcal{D}$.

The equivalence between two matrix codes can be expressed using the Kronecker product of $\mathbf{A}^\top$ and $\mathbf{B}$, which we denote by $\mathbf{A}^\top \otimes \mathbf{B}$.

Lemma 1. *Let $\mathcal{C}$ and $\mathcal{D}$ be two $[m \times n, k]$ matrix codes over $\mathbb{F}_q$. Suppose that $\mathcal{C}$ and $\mathcal{D}$ are equivalent with $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$, with $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$. If $\mathbf{G}$ and $\mathbf{G}'$ are generator matrices for $\mathcal{C}$ and $\mathcal{D}$ respectively, then there exists a $\mathbf{T} \in \mathrm{GL}_k(q)$ such that $\mathbf{G}' = \mathbf{T}\mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B})$.*

It is common to write the generator matrices in systematic form (i.e., as a matrix of the shape $(I|M)$); we denote this operation by SF. Following Lemma 1, this gives us that $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$ if and only if $\mathsf{SF}(\mathbf{G}') = \mathsf{SF}(\mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B}))$.
To simplify notation, we introduce the following operator:

$$\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}) := \mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B}).$$

We are now ready to describe some hard problems connected to the objects we just introduced. The Matrix Code Equivalence (MCE) problem is formally defined as follows:

Problem 2 (Matrix Code Equivalence).
$\mathsf{MCE}(k, n, m, \mathcal{C}, \mathcal{D})$:
Given: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$.
Goal: Determine if there exist $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that $\mathcal{D} = \mathbf{A}\mathcal{C}\mathbf{B}$.

The map $(\mathbf{A}, \mathbf{B}) : \mathbf{C} \mapsto \mathbf{ACB}$ is an *isometry* between $\mathcal{C}$ and $\mathcal{D}$, in the sense that it preserves the rank i.e. $\operatorname{Rank} \mathbf{C} = \operatorname{Rank}(\mathbf{ACB})$. When $n = m$, such isometries can also be extended by transpositions of codewords, however, we choose to work with this smaller set of isometries for simplicity, at no cost to cryptographic security. Note that, although we defined MCE as a decisional problem, our signature construction relies on the computational version of it.

Remark 2. We thank Giuseppe D'Alconzo for the following sharp observation: An MCE instance of dimension k with $m \times n$ matrices over $\mathbb{F}_q$ can be viewed as a 3-tensor problem, which is symmetrical in its arguments k, m and n. This means that it is equivalent to an MCE instance of dimension m with $k \times n$ matrices and to an MCE instance of dimension n with $k \times m$ matrices. Switching to equivalent instances is a matter of changing perspective on the $k \times m \times n$ object over $\mathbb{F}_q$ defined by $A_{ijl} = A^{(l)}_{ij}$. In other words, each basis matrix $m \times n$ defines a slice of a cube, and one can take different slices for equivalent instances.

The following related problem is at the basis of the ring signature construction.

Problem 3 (Inverse Matrix Code Equivalence).
$\mathsf{IMCE}(k, n, m, \mathcal{C}, \mathcal{D}_1, \mathcal{D}_2)$:
Given: Three k-dimensional matrix codes $\mathcal{C}, \mathcal{D}_1, \mathcal{D}_2 \subset \mathcal{M}_{m,n}(q)$.
Goal: Determine if there exist $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that
$\mathcal{D}_1 = \mathbf{A}\mathcal{C}\mathbf{B}$ and $\mathcal{D}_2 = \mathbf{A}^{-1}\mathcal{C}\mathbf{B}^{-1}$.

Problem 3 is closely connected to MCE, in a manner similar to the well-known connection between discrete logarithm and related problems, i.e.:

$$\mathsf{DLOG} \geq \mathsf{DDH} \geq \mathsf{InverseDDH}.$$

Reductions in the opposite direction are not known, as explained for example in [8]. Although this does not provide any further indication about the complexity of such problems, no explicit weaknesses are known either, and the problems are generally considered hard. A more generic overview about hardness of group action-based problems (of which the discrete logarithm is only a particular instantiation) is given in [31], with similar conclusions. For our specific case, we present a discussion about the concrete hardness of IMCE in Section 6.

Finally, we present a *multiple-instance* version of MCE, which is at the base of one of the optimizations, using *multiple public keys*, which we will describe in Section 5. It is easy to see that this new problem reduces to MCE, as done for instance in [10] for the Hamming case.

Problem 4 (Multiple Matrix Code Equivalence).
$\mathsf{MMCE}(k, n, m, r, \mathcal{C}, \mathcal{D}_1, \dots, \mathcal{D}_r)$:
Given: $(r + 1)$ k-dimensional matrix codes $\mathcal{C}, \mathcal{D}_1, \dots, \mathcal{D}_r \subset \mathcal{M}_{m,n}(\mathbb{F}_q)$.
Goal: Find – if any – $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that $\mathcal{D}_i = \mathbf{A}\mathcal{C}\mathbf{B}$ for some $i \in \{1, \dots, r\}$.

The MCE problem has been shown to be at least as hard as the Code Equivalence problem in the Hamming metric [25]. Furthermore, under moderate assumptions, MCE is equivalent to the homogeneous version of the Quadratic Maps Linear Equivalence problem (QMLE) [50], which is considered the hardest among polynomial equivalence problems. An extensive security evaluation will be given in Section 6, encompassing an overview of the best attack techniques and concrete security estimates. From this, we infer a choice of parameters in Section 7.2.

To conclude, we now lay out the details of the MCE-based group action, given by the action of isometries on k-dimensional matrix codes. That is, the set X is formed by the k-dimensional matrix codes of size $m \times n$ over some base field $\mathbb{F}_q$, and the group $G = \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$ acts on this set via isometries as follows:

$$\begin{array}{rcl} \star : \quad G \times X & \rightarrow & X \\ ((\mathbf{A}, \mathbf{B}), \mathcal{C}) & \mapsto & \mathbf{A}\mathcal{C}\mathbf{B} \end{array}$$

We write $\mathcal{G}_{m,n}(q)$ to denote this group of isometries and $\mathcal{M}_{k,m,n}(q)$ for the set of k-dimensional matrix codes; to simplify notation, we drop the indices k, m, n and q when clear from context. Then, for this MCE-based group action the Vectorization Problem is precisely Problem 2. This action is not commutative and in general neither transitive nor free. We can restrict the set $\mathcal{M}$ to a single well-chosen orbit to make the group action both transitive and free. In fact, picking any orbit generated from some starting code $\mathcal{C}$ ensures transitivity, and the group action is free if the chosen code $\mathcal{C}$ has trivial automorphism group $\mathrm{Aut}_{\mathcal{G}}(\mathcal{C}) := \{\varphi \in \mathcal{G} : \varphi(\mathcal{C}) = \mathcal{C}\}$, where trivial means up to scalars in $\mathbb{F}_q$[6]. The non-commutativity is both positive and negative: although it limits the cryptographical design possibilities, e.g. key exchange becomes hard, it prevents quantum attacks to which commutative cryptographic group actions are vulnerable, such as Kuperberg's algorithm for the dihedral hidden subgroup problem [37].

With regards to efficiency, it is immediate to notice that our group action is very promising, given that the entirety of the operations in the proposed protocols is simple linear algebra; this is in contrast with code-based literature (where

[6] More accurately, as the action of an isometry (A, B) is only interesting up to scalars $\lambda, \mu \in \mathbb{F}_q$, the group that is acting *freely* is $\mathrm{PGL}_m(q) \times \mathrm{PGL}_n(q)$.

Public Data
$q, m, n, k, \lambda \in \mathbb{N}$.
$\mathsf{hash} : \{0,1\}^* \to \{0,1\}^{2\lambda}$.

II. Commit(pk)
1. $\tilde{\mathbf{A}}, \tilde{\mathbf{B}} \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
2. Compute $\tilde{\mathbf{G}} = \mathsf{SF}(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_0))$.
3. Compute $h = \mathsf{hash}(\tilde{\mathbf{G}})$.
4. Set $\mathsf{cmt} = h$.
5. Send cmt to verifier.

IV. Response(sk, pk, cmt, ch)
1. If $\mathsf{ch} = 0$ set $(\mu, \nu) = (\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$.
2. If $\mathsf{ch} = 1$ set $(\mu, \nu) = (\tilde{\mathbf{A}}\mathbf{A}^{-1}, \mathbf{B}^{-1}\tilde{\mathbf{B}})$.
3. Set $\mathsf{rsp} = (\mu, \nu)$.
4. Send rsp to verifier.

I. Keygen()
1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k \times mn}$ in standard form
2. $(\mathbf{A}, \mathbf{B}) \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
3. Compute $\mathbf{G}_1 = \mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0))$.
4. Set $\mathsf{sk} = (\mathbf{A}, \mathbf{B})$ and $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1)$.

III. Challenge()
1. $c \xleftarrow{\$} \{0,1\}$.
2. Set $\mathsf{ch} = c$.
3. Send ch to prover.

V. Verify(pk, cmt, ch, rsp)
1. If $\mathsf{ch} = 0$ compute
   $h' = \mathsf{hash}(\mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_0)))$.
2. If $\mathsf{ch} = 1$ compute
   $h' = \mathsf{hash}(\mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_1)))$.
3. Accept if $h' = \mathsf{cmt}$ or reject otherwise.

Fig. 1. MCE Sigma Protocol

complex decoding algorithms are usually required) and other group actions (e.g. isogeny-based) which are burdened by computationally heavy operations. Further details about performance are given in Section 7.

4 Protocols from Matrix Code Equivalence

The efficient non-commutative cryptographic group action provided by MCE from Section 3 yields a promising building block for post-quantum cryptographic schemes. In this section, we obtain a digital signature scheme by first designing a Sigma protocol and then applying the Fiat-Shamir transformation [32]. With a similar procedure, we are able to obtain (linkable) ring signatures as well.

4.1 Sigma Protocols and Signatures

The first building block in our work is the Sigma protocol in Figure 1, in which a Prover proves the knowledge of an isometry $(\mathbf{A}, \mathbf{B})$ between two equivalent matrix codes. The security result is given in Theorem 1.

Theorem 1. *The Sigma protocol described above is complete, 2-special sound and honest-verifier zero-knowledge assuming the hardness of the* MCE *problem.*

Proof. We prove the three properties separately.

Completeness. An honest prover will always have his response accepted by the verifier. In fact, for the case $c = 0$, we have

$$h' = \mathsf{hash}(\mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_0))) = \mathsf{hash}(\mathsf{SF}(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_0)) = \mathsf{hash}(\tilde{\mathbf{G}}) = h.$$

For the case $c = 1$, instead, observe that

$$\pi_{\tilde{\mathbf{A}}\mathbf{A}^{-1},\mathbf{B}^{-1}\tilde{\mathbf{B}}}(\mathbf{G}_1) = \pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}\left(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G}_1)\right).$$

Since $\mathbf{G}_1 = \mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0))$, then $\mathbf{G}_1 = \mathbf{T}\left(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0)\right)$ for some $\mathbf{T} \in \mathrm{GL}_k(q)$. Hence, we have that

$$\pi_{\tilde{\mathbf{A}}\mathbf{A}^{-1},\mathbf{B}^{-1}\tilde{\mathbf{B}}}(\mathbf{G}_1) = \pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}\left(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}\left(\mathbf{T}\left(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}_0)\right)\right)\right) = \mathbf{T}\left(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_0)\right).$$

Now, computing the systematic form and applying hash, we again have $h' = h$.

2-Special Soundness. To get the extractor which produces the witness, we prove that having obtained two valid transcripts with the same commitment and a different challenge, we can extract a solution for the underlying MCE problem. This demonstrates that this Sigma protocol is a Proof of Knowledge with soundness error 1/2.

Let $(h, 0, \mathsf{rsp}_0)$ and $(h, 1, \mathsf{rsp}_1)$ be two valid transcripts, where $\mathsf{rsp}_0 = (\mu_0, \nu_0)$ and $\mathsf{rsp}_1 = (\mu_1, \nu_1)$. Since both pass verification, we have that

$$\mathsf{hash}(\mathsf{SF}(\pi_{\mu_0,\nu_0}(\mathbf{G}_0))) = h = \mathsf{hash}(\mathsf{SF}(\pi_{\mu_1,\nu_1}(\mathbf{G}_1))).$$

Then, since we assume that hash is collision resistant, it must be that

$$\mathsf{SF}(\pi_{\mu_0,\nu_0}(\mathbf{G}_0)) = \mathsf{SF}(\pi_{\mu_1,\nu_1}(\mathbf{G}_1)).$$

From this, it is easy to see that $\mathsf{SF}(\pi_{\mu^*,\nu^*}(\mathbf{G}_0)) = \mathbf{G}_1$, for an isometry $(\mu^*, \nu^*) = = (\mu_1^{-1}\mu_0, \nu_0\nu_1^{-1})$, i.e. (μ^*, ν^*) is a solution to the MCE problem.

Honest-Verifier Zero-Knowledge. To show this, we provide a simulator S which, without the knowledge of the witness, is able to produce a transcript which is indistinguishable from one obtained after an interaction with an honest verifier. When the challenge is $c = 0$, the prover's response is $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$, and does not involve the witness. Hence, it is obvious that S can obtain a flawless simulation by simply performing the same steps as an honest prover would. Thus,

$$\Pr\Big[\mathsf{V}_2(\mathsf{pk}, \mathsf{cmt}, 0, \mathsf{rsp}) = 1 \mid (\mathsf{cmt}, 0, \mathsf{rsp}) \leftarrow \mathsf{S}(\mathsf{pk})\Big] = 1.$$

When $c = 1$, the simulator generates two random matrices $(\mathbf{A}^*, \mathbf{B}^*)$, then sets the commitment to $\mathsf{cmt} = \mathsf{hash}(SF(\pi_{\mathbf{A}^*,\mathbf{B}^*}(\mathbf{G}_1)))$ and the response to $\mathsf{rsp} = (\mathbf{A}^*, \mathbf{B}^*)$. When $(\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$ is uniformly generated, then the matrices $(\tilde{\mathbf{A}}\mathbf{A}^{-1}, \mathbf{B}^{-1}\tilde{\mathbf{B}})$ are uniformly generated too, and hence follow the same distribution as $(\mathbf{A}^*, \mathbf{B}^*)$. Furthermore, the triple $(\mathsf{cmt}, 1, \mathsf{rsp})$ is valid and therefore

$$\Pr\Big[\mathsf{V}_2(\mathsf{pk}, \mathsf{cmt}, 1, \mathsf{rsp}) = 1 \mid (\mathsf{cmt}, 1, \mathsf{rsp}) \leftarrow \mathsf{S}(\mathsf{pk})\Big] = 1.$$

This concludes the proof. □

Applying the Fiat-Shamir transformation gives the signature scheme in Figure 2.

```
Public Data
$q, m, n, k, t = \lambda \in \mathbb{N}$.
$\mathsf{hash} : \{0,1\}^* \to \{0,1\}^t$.

I. Keygen()
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k \times mn}$ in standard form
  2. Set $\mathbf{A}_0 = \mathbf{I}_m$, $\mathbf{B}_0 = \mathbf{I}_n$.
  3. $\mathbf{A}_1, \mathbf{B}_1 \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  4. Compute $\mathbf{G}_1 = \mathsf{SF}(\pi_{\mathbf{A}_1,\mathbf{B}_1}(\mathbf{G}_0))$.
  5. Set $\mathsf{sk} = (\mathbf{A}_1, \mathbf{B}_1)$ and $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1)$.

II. Sign(sk)
  1. For all $i = 0, \ldots, t-1$:
       i. $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
      ii. Compute $\tilde{\mathbf{G}}_i = \mathsf{SF}(\pi_{\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i}(\mathbf{G}_0))$.
  2. Compute
     $h = \mathsf{hash}(\tilde{\mathbf{G}}_0, \ldots, \tilde{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  3. Parse $h = [h_0|\ldots|h_{t-1}]$, $h_i \in \{0,1\}$.
  4. For all $i = 0, \ldots, t-1$:
       i. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i \mathbf{A}_{h_i}^{-1}, \mathbf{B}_{h_i}^{-1} \tilde{\mathbf{B}}_i)$.
  5. Set $\sigma = (h, \mu_0, \ldots, \mu_{t-1}, \nu_0, \ldots, \nu_{t-1})$.
  6. Send $\sigma$ to verifier.

III. Verify(pk, msg, $\sigma$)
  1. Parse $h = [h_0|\ldots|h_{t-1}]$, $h_i \in \{0,1\}$.
  2. For all $i = 0, \ldots, t-1$:
       i. Set $\hat{\mathbf{G}}_i = \mathsf{SF}(\pi_{\mu_i,\nu_i}(\mathbf{G}_{h_i}))$.
  3. Compute
     $h' = \mathsf{hash}(\hat{\mathbf{G}}_0, \ldots, \hat{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  4. Accept if $h' = h$ or reject otherwise.
```

Fig. 2. The basic signature scheme

Public key and signature size. We begin by calculating the communication costs for the Sigma protocol of Figure 1. Note that, for the case $c = 0$, the response (μ, ν) consists entirely of randomly-generated objects, and is efficiently represented by a single seed (that can be used to generate both matrices). This yields the following cost per round, in bits:

$$\begin{cases} 3\lambda + 1 & \text{if } c = 0 \\ 2\lambda + 1 + (m^2 + n^2)\lceil \log_2(q) \rceil & \text{if } c = 1 \end{cases}$$

remembering that seeds are λ bits and hash digests 2λ to avoid collision attacks.

For the signature scheme we calculate the sizes as follows. First, since the matrix $\mathbf{G}_0$ is random, it can also be represented via a short seed, and therefore can be included in the public key at negligible cost (see Algorithm I. of Figure 2). Keeping in mind that the number of rounds t is equal to the value of the desired security level λ, the protocol above yields the following sizes (in bits):

- Public key size: $\lambda + k(mn - k)\lceil \log_2(q) \rceil$
- Average signature size: $t\left(1 + \dfrac{\lambda + (m^2 + n^2)\lceil \log_2(q) \rceil}{2}\right)$.

4.2 Ring Signatures

The protocol of Figure 1 can be easily adapted to serve as a building block for a ring signature, by providing the ring of users with individual public keys, corresponding to equivalent matrix codes. Any user can then answer the verifier's challenge via the selected private key. The construction crucially utilizes a dedicated primitive known as *Index-Hiding Merkle Tree (IHMT)*. This primitive was first introduced in [15] as a variation on the traditional construction of Merkle trees. With this variant, in fact, the position of a leaf is not revealed,

```
Public Data
$q, m, n, k, \lambda, r \in \mathbb{N}$.
$\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{2\lambda}$.

I. Keygen($j$)
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k\times mn}$ in standard form.
  2. $(\mathbf{A}_j, \mathbf{B}_j) \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  3. Compute $\mathbf{G}_j = \mathsf{SF}(\pi_{\mathbf{A}_j,\mathbf{B}_j}(\mathbf{G}_0))$.
  4. Set $\mathsf{sk}_j = (\mathbf{A}_j, \mathbf{B}_j)$ and $\mathsf{pk}_j = \mathbf{G}_j$.
  5. Set $\mathsf{pk} = (\mathbf{G}_0, \mathsf{pk}_1, \dots, \mathsf{pk}_r)$

II. Commit(pk)
  1. $\tilde{\mathbf{A}}, \tilde{\mathbf{B}} \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  2. For all $j = 1, \dots, r$:
       i. Sample $\mathsf{r}_j \xleftarrow{\$} \{0,1\}^\lambda$.
      ii. Compute $\tilde{\mathbf{G}}_j = \mathsf{SF}(\pi_{\tilde{\mathbf{A}},\tilde{\mathbf{B}}}(\mathbf{G}_j))$.
     iii. Compute $h_j = \mathsf{Com}(\mathsf{r}_j, \tilde{\mathbf{G}}_j)$.
  3. Build IHMT from $(h_1, \dots, h_r)$ and get root.
  4. Set cmt = root.
  5. Send cmt to verifier.

III. Challenge()
  1. $c \xleftarrow{\$} \{0,1\}$.
  2. Set $\mathsf{ch} = c$.
  3. Send ch to prover.

IV. Response($\mathsf{sk}_{j^*}$, pk, cmt, ch)
  1. If $\mathsf{ch} = 0$:
       i. Set $(\mu, \nu) = (\tilde{\mathbf{A}}\mathbf{A}_{j^*}, \mathbf{B}_{j^*}\tilde{\mathbf{B}})$.
      ii. Get path corresponding to leaf $j^*$ in IHMT for $(h_1, \dots, h_r)$.
     iii. Set $\mathsf{bits} = \mathsf{r}_{j^*}$.
      iv. Set $\mathsf{rsp} = (\mu, \nu, \mathsf{path}, \mathsf{bits})$.
  2. If $\mathsf{ch} = 1$:
       i. Set $(\mu, \nu) = (\tilde{\mathbf{A}}, \tilde{\mathbf{B}})$.
      ii. Set $\mathsf{bits} = (\mathsf{r}_1, \dots, \mathsf{r}_r)$.
     iii. Set $\mathsf{rsp} = (\mu, \nu, \mathsf{bits})$.
  3. Send rsp to verifier.

V. Verify(pk, cmt, ch, rsp)
  1. If $\mathsf{ch} = 0$:
       i. Compute $\hat{\mathbf{G}} = \mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_0))$.
      ii. Compute $h' = \mathsf{Com}(\mathsf{bits}, \hat{\mathbf{G}})$.
     iii. Get root$'$ from IHMT using $h'$ and path.
  2. If $\mathsf{ch} = 1$:
       i. For all $j = 1, \dots, r$:
            a. Compute $\hat{\mathbf{G}}_j = \mathsf{SF}(\pi_{\mu,\nu}(\mathbf{G}_j))$.
            b. Compute $h'_j = \mathsf{Com}(\mathsf{r}_j, \hat{\mathbf{G}}_j)$.
      ii. Build IHMT from $(h'_1, \dots, h'_r)$ and get root$'$.
  3. Accept if root$'$ = cmt, reject otherwise.
```

Fig. 3. MCE Ring Sigma Protocol

which is necessary to ensure anonymity. This can be accomplished by specifying a different method to construct the tree, based on an alternative ordering (e.g. lexicographic). For further details, we refer the reader to [15]. A visual representation is given in Figure 3, where we remind the reader that Algorithm I. Keygen is executed *once for each prover*, Algorithm II. Commit is the same regardless of the chosen prover, and Algorithm IV. Response is instead unique for the specific prover identified by $j^* \in \{1, \dots, r\}$.

It is easy to see that the construction in Figure 3 satisfies the Completeness, 2-Special Soundness and Honest-Verifier Zero-Knowledge properties, similarly to the protocol of Figure 1. The proof is analogue to the proof of Theorem 1, with only minor differences (such as the reversal in the roles of the challenges), that have no noticeable impact; this is therefore omitted in the interest of space. The protocol can then be turned into a ring signature scheme using Fiat-Shamir as usual, i.e. in the same manner as what was done for the protocol of Figure 1. Furthermore, we can make the ring signature scheme *linkable*, by means of a few modifications which we explain next. To avoid needless repetition, we avoid presenting the ring signature scheme in its extended form (as in Figure 2) and we move on instead to discussing the linkable variant; we will then present the linkable ring signature scheme, in its entirety, to conclude this section.

To begin, recall the group action $\star : G \times X \to X$, formalized in Section 3, with $X = \mathcal{M}$ the set of k-dimensional matrix codes, $G = \mathcal{G}$ the group of isometries for such codes, and the action given by $\star : ((\mathbf{A}, \mathbf{B}), \mathcal{C}) \mapsto \mathbf{A}\mathcal{C}\mathbf{B}$. We now require another group action $* : G \times Y \to Y$, satisfying the following properties.

Definition 7. Let $\star : G \times X \to X$ and $* : G \times Y \to Y$ be two group actions. We define the following properties for the pair $(\star, *)$:

- *Linkability*: Given $(x, y) \in X \times Y$, it is hard to output $g, g' \in G$ with $g \star x = g' \star x$ and $g * y \neq g' * y$.
- *Linkable Anonymity*: Given $(x, y) \in X \times Y$, the pair $(g \star x, g * y)$ is indistinguishable from (x', y'), where g and (x', y') are sampled uniformly at random from G and $X \times Y$, respectively.
- *Non-Frameability*: Given $(x, y) \in X \times Y$, $x' = g \star x$ and $y' = g * y$, for g sampled uniformly at random from G, it is hard to output $g' \in G$ with $g' * y = y'$.

Note how these three properties recall the Linkability, Linkable Anonymity and Non-Frameability properties introduced in Section 2.2. Indeed, showing that the pair of group actions satisfies the above properties is the key to constructing a secure linkable ring signature scheme. Also, as noted before, one could define unforgeability in a manner similar to the last property (by asking to output $g' \in G$ with $g' \star x = x'$) but this is not necessary, since this is a direct consequence of linkability and non-frameability. Informally, then, one can treat elements $y \in Y$ as "tags", that can be checked to establish the link. To this end, it is convenient to introduce an efficiently computable function $\mathsf{Link} : Y \times Y \to \{0, 1\}$, defined by $\mathsf{Link}(y, y') = 1 \Leftrightarrow y = y'$. For our purposes, we require $Y = X = \mathcal{M}$ and define $*$ as $\star^{-1}$. Thus, $* : ((\mathbf{A}, \mathbf{B}), \mathcal{C}) \mapsto \mathbf{A}^{-1}\mathcal{C}\mathbf{B}^{-1}$. We will then show that the required properties hold for any given code $\mathcal{C}$, with the IMCE problem as a basis for its security.

Theorem 2. *The pair of group actions $(\star, *)$ described above is linkable, linkably anonymous and non-frameable assuming the hardness of the* IMCE *problem.*

Proof. We prove the three properties separately.

Linkability. Consider a code $\mathcal{C}$ defined by $\mathbf{G}$, together with two isometries $g = (\mathbf{A}, \mathbf{B})$ and $g' = (\mathbf{A}', \mathbf{B}')$. Suppose that $\mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G})) = \mathsf{SF}(\pi_{\mathbf{A}',\mathbf{B}'}(\mathbf{G}))$ or, equivalently, that $\mathsf{SF}(\mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B})) = \mathsf{SF}(\mathbf{G}((\mathbf{A}')^\top \otimes \mathbf{B}'))$. Then, it must be that $\mathbf{A}^\top \otimes \mathbf{B} = (\mathbf{A}')^\top \otimes \mathbf{B}'$, unless there is an automorphism of $\mathcal{C}$ hidden in the product. As noted in [50], it is plausible to assume that a random code $\mathcal{C}$ admits only the trivial automorphism, for all parameter choices we are interested in. If so, such trivial automorphism are pairs of scalar multiples of the identity for scalars $\alpha \in \mathbb{F}_q$. Hence, either we have $(\mathbf{A}^\top \otimes \mathbf{B})^{-1} = ((\mathbf{A}')^\top \otimes \mathbf{B}')^{-1}$; or, $(\mathbf{A}^\top \otimes \mathbf{B})^{-1} = \alpha^{-1}((\mathbf{A}')^\top \otimes \mathbf{B}')^{-1}$. In both cases, we have that $\mathsf{SF}(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G})) = \mathsf{SF}(\pi_{(\mathbf{A}')^{-1},(\mathbf{B}')^{-1}}(\mathbf{G}))$ i.e. $g * y = g' * y$.

Linkable Anonymity. This follows directly from the hardness of the IMCE problem. As discussed in Section 3, the problem is believed to be only marginally easier than MCE. In Section 6, we analyse dedicated attack techniques for IMCE.

Non-Frameability. For this property, an adversary A is given again a code $\mathcal{C}$ defined by $\mathbf{G}$ and codes x' and y' defined by $\mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G}))$ and $\mathsf{SF}(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G}))$ respectively, where $g = (\mathbf{A}, \mathbf{B})$ is chosen uniformly at random in $\mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$. The adversary A is asked to find $(\mathbf{A}', \mathbf{B}')$ such that

$$\mathsf{SF}(\pi_{(\mathbf{A}')^{-1},(\mathbf{B}')^{-1}}(\mathbf{G})) = y' = \mathsf{SF}(\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}(\mathbf{G})).$$

We can use such an adversary to build a distinguisher D for the IMCE problem, as follows. Suppose $(\mathcal{C}, \mathcal{D}_1, \mathcal{D}_2)$ is the given instance of the problem, and let ϵ be the success probability of A. To begin, D calls A on $(\mathcal{C}, \mathcal{D}_1, \mathcal{D}_2)$, and A will reply with $g' = (\mathbf{A}', \mathbf{B}')$ that satisfies the equation above. From what we have seen in the proof of the Linkability property above, this implies that $((\mathbf{A}')^\top \otimes \mathbf{B}')^{-1} = (\mathbf{A}^\top \otimes \mathbf{B})^{-1}$, modulo trivial automorphisms. Either way, we have that $\mathsf{SF}(\pi_{\mathbf{A},\mathbf{B}}(\mathbf{G})) = \mathsf{SF}(\pi_{\mathbf{A}',\mathbf{B}'}(\mathbf{G}))$. Thus, it is enough for D to compute $\mathsf{SF}(\pi_{\mathbf{A}',\mathbf{B}'}(\mathbf{G}))$ and $\mathsf{SF}(\pi_{(\mathbf{A}')^{-1},(\mathbf{B}')^{-1}}(\mathbf{G})))$, and check whether they define $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. D then answers 1 if this is the case, or 0 otherwise. Since the probability that a randomly-drawn pair $(\mathcal{D}_1, \mathcal{D}_2)$ satisfies this condition is negligible, the probability of success of D is essentially the same as ϵ.

Together, these three properties complete the proof. □

Having clarified the nature of the tools at hand, we are now ready to introduce the linkable version of the ring signature scheme. We first explain compactly how to modify the protocol of Figure 3, then give a full description in Figure 4.

– As part of the public data, choose a collision-resistant hash function $\overline{\mathsf{hash}}$.
– The commit procedure (Algorithm II.) now additionally uses a matrix $\mathbf{T} = \mathsf{SF}(\pi_{\mathbf{A}_{j^*}^{-1},\mathbf{B}_{j^*}^{-1}}(\mathbf{G}_0))$; this is the "tag" for prover j^*, which will be trasmitted alongside the commitment to the verifier. The prover then computes $\tilde{\mathbf{T}} = \mathsf{SF}(\pi_{\tilde{\mathbf{A}}^{-1},\tilde{\mathbf{B}}^{-1}}(\mathbf{T}))$. After obtaining the root of the IHMT, the prover obtains $\bar{h} = \overline{\mathsf{hash}}(\tilde{\mathbf{T}}, \mathsf{root})$ and the commitment is set to $\bar{h}$ instead of root.
– Algorithm IV. is identical, except that, in the case $c = 0$, the response has to include also the matrices $\bar{\mu}, \bar{\nu} = \mathbf{A}_{j^*}\tilde{\mathbf{A}}, \tilde{\mathbf{B}}\mathbf{B}_{j^*}$.
– Finally, Algorithm V. is modified to involve a check on the tag, as well. The case $c = 1$ is trivial, with the verifier essentially repeating the commitment procedure (Algorithm II.) as in the original protocol, only this time checking equality with $\bar{h}$ rather than root. For the case $c = 0$, instead, the verifier additionally computes $\hat{\mathbf{T}} = \mathsf{SF}(\pi_{\bar{\mu}^{-1},\bar{\nu}^{-1}}(\mathbf{G}_0))$, then uses this value to obtain $\overline{\mathsf{hash}}(\hat{\mathbf{T}}, \mathsf{root}')$ and verify equality with $\bar{h}$.

It is relatively easy to see that the new protocols satisfy all the necessary security properties. For instance, the ring Sigma protocol, with the above modifications, still satisfies the Completeness, 2-Special Soundness and Honest-Verifier

Public Data
$q, m, n, k, t = \lambda, r \in \mathbb{N}$. $\mathsf{Com} : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^{2\lambda}$. $\mathsf{hash}, \overline{\mathsf{hash}} : \{0,1\}^* \to \{0,1\}^t$.

```
II. Sign($\mathsf{sk}_{j^*}$, $\mathsf{pk}$)
  1. Compute the tag
     $\mathbf{T} = \mathsf{SF}(\pi_{\mathbf{A}_{j^*}^{-1}, \mathbf{B}_{j^*}^{-1}}(\mathbf{G}_0))$.
  2. For all $i = 0, \ldots, t-1$:
      i. $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
     ii. Set $\tilde{\mathbf{T}}_i = \mathsf{SF}(\pi_{\tilde{\mathbf{A}}_i^{-1}, \tilde{\mathbf{B}}_i^{-1}}(\mathbf{T}))$.
    iii. For all $j = 1, \ldots, r$:
          a. Sample $\mathsf{r}_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$.
          b. Set $\tilde{\mathbf{G}}_{i,j} = \pi_{\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i}(\mathbf{G}_j)$.
          c. Set $h_{i,j} = \mathsf{Com}(\mathsf{r}_{i,j}, \mathsf{SF}(\tilde{\mathbf{G}}_{i,j}))$.
     iv. Build IHMT from $(h_{i,1}, \ldots, h_{i,r})$
         and get $\mathsf{root}_i$.
      v. Compute $\bar{h}_i = \overline{\mathsf{hash}}(\tilde{\mathbf{T}}_i, \mathsf{root}_i)$.
  3. Conpute
     $h = \mathsf{hash}(\bar{h}_0, \ldots, \bar{h}_{t-1}, \mathbf{T}, \mathsf{msg})$.
  4. Parse $h = [h_0|\ldots|h_{t-1}]$, $h_i \in \{0,1\}$.
  5. For all $i = 0, \ldots, t-1$:
      i. If $h_i = 0$:
          a. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i \mathbf{A}_{j^*}, \mathbf{B}_{j^*} \tilde{\mathbf{B}}_i)$.
          b. Set $(\bar{\mu}_i, \bar{\nu}_i) = (\mathbf{A}_{j^*} \tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \mathbf{B}_{j^*})$.
          c. Get $\mathsf{path}_i$ corresponding to leaf
             $j^*$ in IHMT for $(h_{i,1}, \ldots, h_{i,r})$.
          d. Set $\mathsf{bits}_i = \mathsf{r}_{i,j^*}$.
          e. Set $\mathsf{rsp}_i =$
             $\{\mu_i, \nu_i, \bar{\mu}_i, \bar{\nu}_i, \mathsf{path}_i, \mathsf{bits}_i\}$.
     ii. If $h_i = 1$:
          a. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i)$.
          b. Set $\mathsf{bits}_i = (\mathsf{r}_{i,1}, \ldots, \mathsf{r}_{i,r})$.
          c. Set $\mathsf{rsp}_i = \{\mu_i, \nu_i, \mathsf{bits}_i\}$.
  6. Set $\sigma = (h, \mathsf{rsp}_0, \ldots, \mathsf{rsp}_{t-1}, \mathbf{T})$.
  7. Send $\sigma$ to verifier.

I. Keygen($j$)
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k \times mn}$ in standard form.
  2. $(\mathbf{A}_j, \mathbf{B}_j) \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
  3. Compute $\mathbf{G}_j = \mathsf{SF}(\pi_{\mathbf{A}_j, \mathbf{B}_j}(\mathbf{G}_0))$.
  4. Set $\mathsf{sk}_j = (\mathbf{A}_j, \mathbf{B}_j)$ and $\mathsf{pk}_j = \mathbf{G}_j$.
  5. Set $\mathsf{pk} = (\mathbf{G}_0, \mathsf{pk}_1, \ldots, \mathsf{pk}_r)$

III. Verify($\mathsf{pk}$, $\mathsf{msg}$, $\sigma$)
  1. Parse $h = [h_0|\ldots|h_{t-1}]$, $h_i \in \{0,1\}$.
  2. For all $i = 0, \ldots, t-1$:
      i. If $h_i = 0$:
          a. Compute $\hat{\mathbf{G}}_i = \mathsf{SF}(\pi_{\mu_i, \nu_i}(\mathbf{G}_0))$.
          b. Compute $h'_i = \mathsf{Com}(\mathsf{bits}_i, \hat{\mathbf{G}}_i)$.
          c. Get $\mathsf{root}_i$ from IHMT using $h'_i$
             and $\mathsf{path}_i$.
          d. Compute
             $\hat{\mathbf{T}}_i = \mathsf{SF}(\pi_{\bar{\mu}_i^{-1}, \bar{\nu}_i^{-1}}(\mathbf{G}_0))$.
     ii. If $h_i = 1$:
          a. For all $j = 1, \ldots, r$:
               † Compute
                 $\hat{\mathbf{G}}_{i,j} = \mathsf{SF}(\pi_{\mu_i, \nu_i}(\mathbf{G}_j))$.
               ‡ Compute
                 $h'_{i,j} = \mathsf{Com}(\mathsf{r}_{i,j}, \hat{\mathbf{G}}_{i,j})$.
          b. Build IHMT from
             $(h'_{i,1}, \ldots, h'_{i,r})$ and get $\mathsf{root}'_i$.
          c. Compute
             $\hat{\mathbf{T}}_i = \mathsf{SF}(\pi_{\mu_i^{-1}, \nu_i^{-1}}(\mathbf{T}))$.
    iii. Compute $\bar{h}'_i = \overline{\mathsf{hash}}(\hat{\mathbf{T}}_i, \mathsf{root}'_i)$.
  3. Compute
     $h' = \mathsf{hash}(\bar{h}'_0, \ldots, \bar{h}'_{t-1}, \mathbf{T}, \mathsf{msg})$.
  4. Accept if $h' = h$ or reject otherwise.
```

Fig. 4. MCE linkable ring signature scheme

Zero-Knowledge properties. The linkable ring signature scheme of Figure 4 is then just an application of the Fiat-Shamir transform. Once again, such proofs are omitted due to space limitations; the interested reader can consult for example [15], where proofs are given in all generality (see e.g. Theorems 4.3, 4.4 and 4.7). We will present computational costs for the (linkable) ring signature scheme in the next section, after discussing optimizations.

5 Matrix Equivalence Digital Signature — MEDS

We apply the following optimizations from the literature to the basic Fiat-Shamir-based signature scheme described in Section 4, to obtain our Matrix Equivalence Digital Signature (MEDS).

Multiple keys. The first optimization is a popular one in literature [26, 16, 10], and it consists of utilizing multiple public keys, i.e. multiple equivalent codes $\mathbf{G}_0, \ldots, \mathbf{G}_{s-1}$, each defined as $\mathbf{G}_i = \mathsf{SF}(\pi_{\mathbf{A}_i,\mathbf{B}_i}(\mathbf{G}_0))$ for uniformly chosen secret keys[7] $(\mathbf{A}_i, \mathbf{B}_i)$. This allows to reduce the soundness error from $1/2$ to $1/2^\ell$, where $\ell = \lceil \log_2 s \rceil$. The optimization works by grouping the challenge bits into strings of ℓ bits, which can then be interpreted as binary representations of the indices $\{0, \ldots, s-1\}$, thus dictating which public key will be used in the protocol. Security is preserved since the proof of unforgeability can be easily modified to rely on a multi-instance version of the underlying problem: in our case, MMCE (Problem 4). Note that, although in the literature s is chosen to be a power of 2, this does not have to be the case. In this work, we will instead select the value of s based on the best outcome in terms of performance and signature size.

Remark 3. This optimization comes at the cost of an s-fold increase in public-key size. As shown for instance in [26], it would be possible to reduce this impact by using Merkle trees to a hash of the tree commitment of all the public keys. This, however, would add some significant overhead to the signature size, because it would be necessary to include the paths for all openings. Considering the sizes of the objects involved, such an optimization is not advantageous in our case.

Partially seeding the public key. The previous optimization comes at significant cost to the public key, so we propose a new optimization that trades public key size for private key size. This optimization is inspired by the trade-off in the key generation of Rainbow [27] and UOV [14]. It has not been previously used in Fiat-Shamir signatures, but we expect it can be used successfully in any scheme coming from equivalence problems especially the ones using the previous optimization, such as [26, 16, 10]. With this optimization, instead of generating the secret $(\mathbf{A}_i, \mathbf{B}_i)$ from a secret seed and then deriving the public $\mathbf{G}_i$, we generate $\mathbf{G}_i$ partially from a public seed and then use it to find $(\mathbf{A}_i, \mathbf{B}_i)$ and the rest of the public key $\mathbf{G}_i$. In more detail, in order to generate the public $\mathbf{G}_i$ and the corresponding secret $(\mathbf{A}_i, \mathbf{B}_i)$ we perform the following:

- We perform a secret change of basis of $\mathbf{G}_0$ by multiplying it by a secret matrix $\mathbf{T} \in \mathrm{GL}_k(q)$ to obtain $\mathbf{G}_0'$. Assume the codewords from $\mathbf{G}_0'$ are $\mathbf{P}_1^0, \mathbf{P}_2^0, \ldots, \mathbf{P}_k^0$.
- For each $i \in \{1, \ldots, s-1\}$, we generate from a public seed a complete $m \times n$ codeword $\mathbf{P}_1^i$ and the top $m-1$ rows of codeword $\mathbf{P}_2^i$ (depending on the parameters m, n one can get slightly more rows when $m \neq n$).

[7] Again, for convenience, we choose $\mathbf{A}_0 = \mathbf{I}_m$, $\mathbf{B}_0 = \mathbf{I}_n$.

– Find $\mathbf{A}_i$ and $\mathbf{B}_i$ from the linear relations:

$$\begin{aligned}\mathbf{P}_1^i \mathbf{B}_i^{-1} &= \mathbf{A}_i \mathbf{P}_1^0 \\ \mathbf{P}_2^i \mathbf{B}_i^{-1} &= \mathbf{A}_i \mathbf{P}_2^0\end{aligned}$$

by fixing the first (top left) value of $\mathbf{A}_i$.

– Find $\mathbf{P}_j^i = \mathbf{A}_i \mathbf{P}_j^0 \mathbf{B}_i$ for all $j \in \{3, \ldots, k\}$.

– Construct the public $\mathbf{G}_i$ from $\mathbf{P}_1^i, \mathbf{P}_2^i, \ldots, \mathbf{P}_k^i$.

The public key then is the public seed together with $\mathbf{P}_3^i, \ldots, \mathbf{P}_k^i$. For verification, the complete $\mathbf{G}_i$ are reconstructed using the seed.

Fixed-weight challenges. Another common optimization is the use of fixed-weight challenges. The idea is to generate the challenge string h with a fixed number of 1s and 0s, i.e. Hamming weight, rather than uniformly at random. This is because, when $h_i = 0$, the response (μ_i, ν_i) consists entirely of randomly-generated objects, and so one can just transmit the seed used for generating them. This creates a noticeable imbalance between the two types of responses, and hence it makes sense to minimize the number of 1 values. To this end, one can utilize a so-called *weight-restricted hash function*, that outputs values in $\mathbb{Z}_{2,w}^t$, by which we denote the set of vectors with elements in $\{0, 1\}$ of length t and weight w. In this way, although the length of the challenge strings increases, the overall communication cost scales down proportionally to the value of w. In terms of security, this optimization only entails a small modification in the statement of the Forking Lemma, and it is enough to choose parameters such that $\log_2 \binom{t}{w} \geq \lambda$. Note that this optimization can easily be combined with the previous one, by mandating hash digests in $\mathbb{Z}_{s,w}^t$ and choosing parameters such that $\log_2 \left(\binom{t}{w}(s-1)^w\right) \geq \lambda$. In practice, this can be achieved with a hash function $\mathsf{hash} : \{0,1\}^* \to \{0,1\}^\lambda$, by expanding the output to a t-tuple $(h_0, \ldots, h_{t-1})$, $0 \leq h_i < s$ of weight w.

Seed tree. Finally, the signature size can be optimized again using a *seed tree*. This primitive allows to generate the many seeds used throughout the protocol in a recursive way, starting from a master seed mseed and building a binary tree, via repeated PRNG applications, having t seeds as leaves. When the required $t - w$ values need to be retrieved, it is then enough to reveal the appropriate sequence of nodes. This reduces the space required for the seeds from $\lambda(t - w)$ to λN_{seeds}, where N_{seeds} can be upper bounded by $2^{\lceil \log_2(w) \rceil} + w(\lceil \log_2(t) \rceil - \lceil \log_2(w) \rceil - 1)$, as shown in [34]. We refer the reader to Section 2.7 of [15] for more details. As suggested in [15], we are including a 256-bit salt to ward off multi-target collision attacks and the leaf address as identifier for domain separation in the inputs of the seed-tree hash functions.

To give a complete picture, we present the MEDS protocol in Figure 5, in its final form, including all applicable variants. The various parameters control different optimization: for instance s refers to the number of public keys used, whereas

```
Public Data
$q, m, n, k, \lambda, t, s, w \in \mathbb{N}$.
$\mathsf{hash} : \{0,1\}^* \to \{0,1\}^\lambda$.

I. Keygen()
  1. $\mathbf{G}_0 \xleftarrow{\$} \mathbb{F}_q^{k \times mn}$ in standard form.
  2. Set $\mathbf{A}_0 = \mathbf{I}_m$, $\mathbf{B}_0 = \mathbf{I}_n$.
  3. $\mathbf{T} \xleftarrow{\$} \mathrm{GL}_k(q)$
  4. Compute $\mathbf{G}_0' = \mathbf{T}\mathbf{G}_0$
  5. Parse the first two rows of $\mathbf{G}_0'$ into $\mathbf{P}_1^0, \mathbf{P}_2^0 \in \mathbb{F}_q^{m \times n}$
  6. For all $j = 1, \ldots, s-1$:
       i. $\mathbf{P}_1^j, \mathbf{P}_2^j \xleftarrow{\$} \mathbb{F}_q^{m \times n}$
      ii. Find $\mathbf{A}_j$ and $\mathbf{B}_j$ from:
            $\mathbf{P}_1^j \mathbf{B}_j^{-1} = \mathbf{A}_j \mathbf{P}_1^0$
            $\mathbf{P}_2^j \mathbf{B}_j^{-1} = \mathbf{A}_j \mathbf{P}_2^0$
     iii. Compute $\mathbf{G}_j = \mathsf{SF}(\pi_{\mathbf{A}_j,\mathbf{B}_j}(\mathbf{G}_0'))$.
  7. Set $\mathsf{sk} = (\mathbf{A}_1^{-1}, \mathbf{B}_1^{-1}, \ldots, \mathbf{A}_{s-1}^{-1}, \mathbf{B}_{s-1}^{-1})$.
  8. Set $\mathsf{pk} = (\mathbf{G}_0, \mathbf{G}_1, \ldots, \mathbf{G}_{s-1})$.

II. Sign(sk)
  1. For all $i = 0, \ldots, t-1$:
       i. $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \xleftarrow{\$} \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$.
      ii. Compute $\tilde{\mathbf{G}}_i = \mathsf{SF}(\pi_{\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i}(\mathbf{G}_0))$.
  2. Compute
     $h = \mathsf{hash}(\tilde{\mathbf{G}}_0, \ldots, \tilde{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  3. Expand $h$ to $(h_0, \ldots, h_{t-1})$, $0 \le h_i < s$.
  4. For all $i = 0, \ldots, t-1$:
       i. Set $(\mu_i, \nu_i) = (\tilde{\mathbf{A}}_i \mathbf{A}_{h_i}^{-1}, \mathbf{B}_{h_i}^{-1} \tilde{\mathbf{B}}_i)$.
  5. Set $\sigma = (\mu_0, \ldots, \mu_{t-1}, \nu_0, \ldots, \nu_{t-1}, h)$.
  6. Send $\sigma$ to verifier.

III. Verify(pk, msg, $\sigma$)
  1. Expand $h$ to $(h_0, \ldots, h_{t-1})$, $0 \le h_i < s$.
  2. For all $i = 0, \ldots, t-1$:
       i. Set $\hat{\mathbf{G}}_i = \mathsf{SF}(\pi_{\mu_i,\nu_i}(\mathbf{G}_{h_i}))$.
  3. Compute
     $h' = \mathsf{hash}(\hat{\mathbf{G}}_0, \ldots, \hat{\mathbf{G}}_{t-1}, \mathsf{msg})$.
  4. Accept if $h' = h$ or reject otherwise.
```

Fig. 5. The MEDS Protocol

w refers to the fixed weight of the challenge hash string. Parameter choices will be thoroughly discussed in Section 7.2. Note that some of these optimizations can be applied in an identical way to the (linkable) ring signature scheme; however, we leave an explicit description of such a scheme, as well as a full-fledged implementation of it, to a dedicated future work.

Public key and signature size. With these various optimizations, we obtain the following public key and signature size for MEDS:

- MEDS public key size: $\lambda + (s-1)((k-2)(mn-k)+n)\lceil \log_2(q) \rceil$
- MEDS signature size:

$$\underbrace{\lambda}_{h} + \underbrace{w(m^2+n^2)\lceil \log_2(q) \rceil}_{\{\mu_i,\nu_i\}_{h_i=1}} + \underbrace{\lambda N_{\text{seeds}}}_{\{\mu_i,\nu_i\}_{h_i=0}} + \underbrace{2\lambda}_{salt}$$

The optimizations described above can also be applied to the ring signature scheme, which gives us the following sizes:

- Ring signature public key size: $\lambda + rk(mn-k)\lceil \log_2(q) \rceil$
- Ring signature size:

$$\underbrace{w\lceil \log_2 t \rceil}_{h} + \underbrace{w((m^2+n^2)\lceil \log_2(q) \rceil + 2\lambda\lceil \log r \rceil + \lambda)}_{\{\mathsf{rsp}_i\}_{h_i=0}} + \underbrace{\lambda N_{\text{seeds}}}_{\{\mathsf{rsp}_i\}_{h_i=1}} + \underbrace{2\lambda}_{salt}$$

For the linkable variant, the cost of the middle term, i.e. $\{\mathsf{rsp}_i\}_{h_i=0}$, is increased to $w(2(m^2+n^2)\lceil \log_2(q) \rceil + 2\lambda\lceil \log r \rceil + \lambda)$, and the signature additionally includes $k(mn-k)\lceil \log_2(q) \rceil$ bits for the tag $\mathbf{T}$.

6 Concrete Security Analysis

In this section, we will mostly use the Big O notation $\mathcal{O}$ to express the complexity of algorithms. Where we are not interested in the polynomial factor we will use $\mathcal{O}^*$. We note that despite the notation, the estimates are quite tight and provide a good basis for choosing parameters.

Recall that the goal of an adversary against MCE is to recover the matrices $\mathbf{A}$ and $\mathbf{B}$, given a description of the matrix codes $\mathcal{C}$ and $\mathcal{D}$. The most naïve attack would be to try every $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$ until we find the correct isometry, amounting to a complexity of $\mathcal{O}(q^{n^2+m^2})$. The naïve attack can be improved by noting that once one of the matrices $\mathbf{A}$ or $\mathbf{B}$ is known, the resulting problem becomes easy [25]. Hence, we only need to brute-force one of $\mathbf{A}$ or $\mathbf{B}$, so the complexity becomes $\mathcal{O}^*(q^{\min\{m^2,n^2\}})$.

In the rest of the section, we will see that there exist several non-trivial attacks that perform much better than this upper bound.

6.1 Birthday-based graph-theoretical algorithms for solving MCE

Recent works [25, 50] investigate the hardness of MCE by connecting it to other equivalence problems, namely, the Code Equivalence problem in the Hamming metric [25] and the Quadratic Maps Linear Equivalence problem (QMLE) [50]. The latter provides complexity analysis by viewing MCE as an instance of QMLE. We recap their results here. For better understanding, we include the definition of the related QMLE problem.

Problem 5. QMLE$(k, N, \mathcal{F}, \mathcal{P})$:
Given: Two k-tuples of multivariate polynomials of degree 2

$$\mathcal{F} = (f_1, f_2, \dots, f_k),\ \mathcal{P} = (p_1, p_2, \dots, p_k) \in \mathbb{F}_q[x_1, \dots, x_N]^k.$$

Goal: Find – if any – matrices $\mathbf{S} \in \mathrm{GL}_N(q), \mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = (\mathcal{F}(\mathbf{x}\mathbf{S}))\mathbf{T}.$$

We denote by hQMLE, inhQMLE and BMLE the related problems when the polynomials are homogeneous of degree 2, inhomogeneous and bilinear, respectively. It was shown in [50] that, under the assumption that the two codes $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups (which is believed to be true with overwhelming probability for big enough parameters), MCE$(k, n, m, \mathcal{C}, \mathcal{D})$ is equivalent to hQMLE$(k, N, \mathcal{F}, \mathcal{P})$ where $N = m + n$. Concretely, an MCE instance with a solution $(\mathbf{A}, \mathbf{B})$ is transformed into an hQMLE instance with a solution $(\mathbf{S}, \mathbf{T})$ where $\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^\top \end{bmatrix}$ and $\mathbf{T}$ corresponds to a change of basis of $\mathcal{D}$. Therefore it is possible to apply algorithms for solving hQMLE to MCE instances such as the graph-theoretic algorithm of Bouillaguet et al. [20].

The algorithm is basically a collision-search algorithm comprised of two steps, as given in Algorithm 1. In the first step we build two lists L_1 and L_2 of size ℓ of

Algorithm 1 Collision-search algorithm

```
1: function BuildList($\mathcal{F}, \mathbb{P}$)
2:     $L \leftarrow \emptyset$
3:     repeat
4:         $\mathbf{x} \xleftarrow{\$} \mathbb{F}_q^{(m+n)}$
5:         if $\mathbb{P}(\mathcal{F}, \mathbf{x})$ then $L \leftarrow L \cup \{\mathbf{x}\}$
6:     until $|L| = \ell$
7:     return $L$
8: function CollisionFind($\mathcal{F}$,$\mathcal{P}$)
9:     $L_1 \leftarrow$ BuildList($\mathcal{F}, \mathbb{P}$)
10:    $L_2 \leftarrow$ BuildList($\mathcal{P}, \mathbb{P}$)
11:    for all $(\mathbf{x}, \mathbf{y}) \in \{L_1 \times L_2\}$ do
12:        $\phi \leftarrow$inhQMLE($\mathbf{x}, \mathbf{y}$)
13:        if $\phi \neq \perp$ then
14:            return solution $\phi$
15:    return $\perp$
```

elements in $\mathbb{F}_q^{(m+n)}$ that satisfy a predefined distinguishing property $\mathbb{P}$ related to the given systems of polynomials $\mathcal{F}$ and $\mathcal{P}$ and that is preserved under isometry. In the second step, we try to find a collision between the two lists that will lead us to the solution. For the property $\mathbb{P}$, the authors of [20] propose:

$$\mathbb{P}(\mathcal{F}, \mathbf{x}) = \top \Leftrightarrow Dim(Ker(D_{\mathbf{x}}(\mathcal{F}))) = \kappa$$

for a suitably chosen κ, where $D_{\mathbf{x}}(\mathcal{F}) = \mathcal{F}(\mathbf{x}+\mathbf{y}) - \mathcal{F}(\mathbf{x}) - \mathcal{F}(\mathbf{y})$ is the *differential* of $\mathcal{F}$ at a point $\mathbf{x}$. Clearly, the rank of the differential is preserved under isometry, so this is an appropriate choice of $\mathbb{P}$. Other instantiations are possible as well, as long as they are invariant under isometry, although their success depends on the distribution of elements that satisfy the property for varying κ.

Once a collision $(\mathbf{a}, \mathbf{b})$ is found, it can be used to derive an associated *inhomogeneous* QMLE instance inhQMLE $(k, (m+n), \mathcal{F}', \mathcal{P}')$ as $\mathcal{F}'(\mathbf{x}) = \mathcal{F}(\mathbf{x}+\mathbf{a})$, $\mathcal{P}'(\mathbf{x}) = \mathcal{P}(\mathbf{x}+\mathbf{b})$ on which we call an inhomogeneous solver. Since it can not be directly checked whether a pair is a collision, the solver needs to be called for each pair, similar to the guess and check approach in ISD algorithms [46].

The inhomogeneous instance can be solved much more efficiently than the homogeneous one. Heuristic evidence suggests that solving *random* instances of the inhQMLE problem using an algebraic approach takes $\mathcal{O}((m+n)^9)$ operations [30], however, the derived inhQMLE instances from the collision-search attack are not random enough. These specific instances have a solver with a complexity of $\mathcal{O}(q^\kappa)$ [19]. As κ is typically chosen to be small, this approach is still efficient in practice. Following the analysis from [50], the concrete complexity of the algorithm for $k \leqslant 2(m+n)$ follows a birthday argument and is the maximum of the complexity of the two steps, i.e.:

$$\max(\sqrt{q^{(m+n)}/d} \cdot C_{\mathbb{P}}, dq^{(m+n)} \cdot C_{\mathsf{iQ}}), \tag{1}$$

with success probability of $\approx 63\%$. Here, $C_{\mathbb{P}}$ denotes the cost of checking whether an element satisfies the property $\mathbb{P}$, d is the proportion of elements satisfying $\mathbb{P}$ and C_{iQ} denotes the cost of a single query to inhQMLE. Note that d can be calculated as $d = 1/\mathcal{O}(q^{\kappa^2+\kappa(k-(m+n))})$ and κ is chosen such that it minimizes Equation (1). Asymptotically, the complexity is $\mathcal{O}^*(q^{\frac{2}{3}(m+n)})$ by balancing the steps [50]. The memory complexity is simply the size of the lists.

It is pointed out in [50] that when $k \geq 2(m+n)$, we can no longer assume that we have distinguished elements, so we need to consider *all* elements in the collision search. Thus, we have $d = 1$ and the complexity of the algorithm is simply $\mathcal{O}(q^{m+n})$. In that case, we can consider choosing arbitrarily one element $\mathbf{x}$ and checking for a collision with all other elements $\mathbf{y} \in \mathbb{F}_q^{m+n}$. This approach yields the same complexity as the previous one, but is superior because it is deterministic and has negligible memory requirements. Note that this approach was also proposed in [20], but as an inferior (in terms of *time* complexity) deterministic variant, rather than as a solution for the lack of a distinguishing property. As this attack can be applied to any parameter set, it presents an upper-bound on the complexity of a classical collision-search algorithm.

For a quantum version of Algorithm 1, both BuildList and CollisionFind can be seen as searches of unstructured databases of a certain size, hence Grover's algorithm applies to both: we can build the list L using only $\sqrt{\ell \cdot d^{-1}}$ searches, and we can find a collision using only $\sqrt{|L_1 \times L_2|}$ queries to the solver. This requires both $\mathbb{P}$ and $\mathsf{inhQMLE}$ to be performed in superposition. The balance between both sides remains the same. In total, the complexity of the quantum version becomes $\mathcal{O}^*(q^{\frac{1}{3}(m+n)})$.

Collision-search algorithm using non-trivial roots. When viewing an MCE instance as an hQMLE instance, it is possible to use certain bilinear properties to improve Algorithm 1. When $n = m$, such instances have approximately q^{2n-k-1} non-trivial roots, which can be used to improve a subroutine of Algorithm 1, and to make it deterministic instead of probabilistic [50]. In practice, such non-trivial roots exist **i)** almost always when $k < 2n$, **ii)** with probability $1/q$ for $k = 2n$, **iii)** with probability $1/q^{k+1-2n}$ for $k > 2n$. The complexity of this approach is $\mathcal{O}^*(q^n)$, if such non-trivial roots exist. This complexity is proven under the assumption that the complexity of the inhomogenous QMLE solver is no greater than $\mathcal{O}(q^n)$, which holds trivially when $k \geq n$ [50], and heuristically when $k < n$. Finding the non-trivial roots can also be done using a bilinear XL algorithm [45]. We do not consider this approach in our analysis, as it is only interesting for a subset of parameters where the systems are (over)determined, i.e. when k is close to $m + n$.

6.2 Algebraic attacks

Direct modelling. Recently, in [50], it was shown that MCE is equivalent to BMLE. One of the natural attack avenues is thus to model the problem as an algebraic system of polynomial equations over a finite field. This approach was taken in [30], where the general Isomorphism of Polynomials (IP) problem was investigated. Here, we focus specifically on BMLE and perform a detailed complexity analysis.

First, fix arbitrary bases $(\mathbf{C}^{(1)}, \ldots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \ldots, \mathbf{D}^{(k)})$ of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. In terms of the bases, the MCE problem can be rephrased as finding $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ and $\mathbf{T} = (t_{ij}) \in \mathrm{GL}_k(q)$ such that:

$$\sum_{1 \leqslant s \leqslant k} t_{rs}\mathbf{D}^{(s)} = \mathbf{A}\mathbf{C}^{(r)}\mathbf{B}, \quad \forall r, 1 \leqslant r \leqslant k \tag{2}$$

The system (2) consists of knm equations in the $m^2+n^2+k^2$ unknown coefficients of the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{T}$. The quadratic terms of the equations are always of the form $\alpha a_{ij} b_{i'j'}$ for some coefficients a_{ij} and $b_{i'j'}$ of $\mathbf{A}$ and $\mathbf{B}$ respectively which means the system (2) is bilinear. Note that the coefficients of $\mathbf{T}$ appear only linearly. As previously, we can guess the m^2 variables from $\mathbf{A}$, which will lead us to a linear system that can be easily solved. However, we can do better by exploiting the structure of the equations.

For ease of readability of the rest of the paragraph denote by $\mathbf{M}_{i\_}$ and $\mathbf{M}_{\_i}$ the i-th row and i-th column of a matrix $\mathbf{M}$. Note that, in (2), for $i \neq j$, the unknown coefficients from two rows $\mathbf{A}_{i\_}$ and $\mathbf{A}_{j\_}$ don't appear in the same equation. Symmetrically, the same holds for $\mathbf{B}_{\_i}$ and $\mathbf{B}_{\_j}$, but we will make use of it for the matrix $\mathbf{A}$. Thus, we can consider only part of the system, and control the number of variables from $\mathbf{A}$. The goal is to reduce the number of variables that we need to guess before obtaining an overdetermined linear system, and we want to do this in an optimal way. Consider the first α rows from $\mathbf{A}$. Extracting the equations that correspond to these rows in (2) leads us to the system:

$$\sum_{1 \leqslant s \leqslant k} t_{rs}\mathbf{D}^{(s)}_{i\_} = \mathbf{A}_{i\_}\mathbf{C}^{(r)}\mathbf{B}, \quad \forall r, i, \ 1 \leqslant r \leqslant k, \ 1 \leqslant i \leqslant \alpha. \tag{3}$$

Guessing the αm coefficients from $\mathbf{A}_{i\_}$ leads to a linear system of αkn equations in $n^2 + k^2$ variables. Choosing $\alpha = \lceil \frac{n^2+k^2}{kn} \rceil$, the complexity of the approach becomes $\mathcal{O}(q^{m\lceil \frac{n^2+k^2}{kn} \rceil}(n^2 + k^2)^3)$. For the usual choice of $m = n = k$, this reduces to at least $\alpha = 2$ and a complexity of $\mathcal{O}(q^{2n}n^6)$.

Note that, one can directly solve the bilinear system (3) using for example XL [23] and the analysis for bilinear systems from [45] (similar results can be obtained from [28]). We have verified, however, that due to the large number of variables compared to the available equations, the complexity greatly surpasses the one of the simple linearization attack presented above.

Improved modelling. In order to improve upon this baseline algebraic attack, we will model the problem differently and completely avoid the t_{rs} variables. This modelling is in the spirit of the minors modellings of MinRank as in [29, 9].

As previously, let $\mathbf{G}$ and $\mathbf{G}'$ be the $k \times mn$ generator matrices of the equivalent codes $\mathcal{C}$ and $\mathcal{D}$ respectively. Then from Lemma 1, $\tilde{\mathbf{G}} = \mathbf{G}(\mathbf{A}^\top \otimes \mathbf{B})$ is a generator matrix of $\mathcal{D}$ for some invertible matrices $\mathbf{A}$ and $\mathbf{B}$. We will take the coefficients of $\mathbf{A}$ and $\mathbf{B}$ to be our unknowns. A crucial observation for this attack is that each row $\tilde{\mathbf{G}}_{i\_}$ of $\tilde{\mathbf{G}}$ is in the span of the rows of $\mathbf{G}'$, since $\mathbf{G}'$ and $\tilde{\mathbf{G}}$ define the same code. This means that adding $\tilde{\mathbf{G}}_{i\_}$ to $\mathbf{G}'$ does not change the code, i.e.,

$${}^{(i)}\mathbf{G}' = \begin{pmatrix} \mathbf{G}' \\ \tilde{\mathbf{G}}_{i\_} \end{pmatrix}$$

is not of full rank. From here, all maximal minors $|\begin{pmatrix} {}^{(i)}\mathbf{G}'_{\text{-}j_1} & {}^{(i)}\mathbf{G}'_{\text{-}j_2} \dots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}} \end{pmatrix}|$ of ${}^{(i)}\mathbf{G}'$, for every $\{j_1, j_2, \dots, j_{k+1}\} \subset \{1, 2, \dots, mn\}$, are zero.

Now, as in a minors modeling of MinRank, we can form equations in the unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$ by equating all maximal minors to zero, which amounts to a total of $\binom{mn}{k+1}$ equations. Since the unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$ appear only in the last row of the minors, and only bilinearly, the whole system is also bilinear. Thus we have reduced the problem to solving the bilinear system

$$\left\{ |\begin{pmatrix} {}^{(i)}\mathbf{G}'_{\text{-}j_1} & {}^{(i)}\mathbf{G}'_{\text{-}j_2} \dots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}} \end{pmatrix}| = 0, \quad \begin{array}{l} \text{for all } i \in \{1, 2, \dots, k\} \text{ and all} \\ \{j_1, j_2, \dots, j_{k+1}\} \subset \{1, 2, \dots, mn\} \end{array} \right. \tag{4}$$

in the $m^2 + n^2$ unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$.

At first sight, (4) seems to have more than enough equations to fully linearize the system. However, the majority of these equations are linearly dependent. In fact, there are only $(mn - k)k$ linearly independent equations. To see this, fix some i and consider a minor $|\begin{pmatrix} {}^{(i)}\mathbf{G}'_{\text{-}j_1} & {}^{(i)}\mathbf{G}'_{\text{-}j_2} \dots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}} \end{pmatrix}|$ of ${}^{(i)}\mathbf{G}'$. Since all rows except the first don't contain any variables, the equation

$$|\begin{pmatrix} {}^{(i)}\mathbf{G}'_{\text{-}j_1} & {}^{(i)}\mathbf{G}'_{\text{-}j_2} \dots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}} \end{pmatrix}| = 0$$

basically defines the linear dependence between the columns ${}^{(i)}\mathbf{G}'_{\text{-}j_1}, \dots {}^{(i)}\mathbf{G}'_{\text{-}j_{k+1}}$. But the rank of the matrix is k, so all columns can be expressed through some set of k independent columns. Thus, in total, for a fixed i we have $mn - k$ independent equations and in total $(mn - k)k$ equations for all i.

Alternatively, we can obtain the same amount of equations from $\tilde{\mathbf{G}}$ and the generator matrix $\mathbf{G}'^{\perp}$ of the dual code of $\mathcal{D}$. Since $\tilde{\mathbf{G}}$ should also be a generator matrix of $\mathcal{D}$, we construct the system:

$$\mathbf{G}'^{\perp} \cdot \tilde{\mathbf{G}}^{\top} = \mathbf{0},$$

which is again a system of $(mn - k)k$ bilinear equations in $n^2 + m^2$ variables.

The complexity of solving the obtained system using either of the modellings strongly depends on the dimension of the code – it is the smallest for $k = mn/2$, and grows as k reduces (dually, as k grows towards mn). In Section 7 we give the concrete complexity estimate for solving the system for the chosen parameters using bilinear XL and the analysis from [45].

The attack does not seem to benefit a lot from being run on a quantum computer. Since the costly part comes from solving a huge linear system for which there are no useful quantum algorithms available, the only way is to 'Groverize' an enumeration part of the algorithm. One could enumerate over one set of the variables, either of $\mathbf{A}$ or $\mathbf{B}$, typically the smaller one, and solve a biliner system of less variables. Grover's algorithm could then speed up quadratically this enumeration. However, since in the classical case the best approach is to not use enumeration, this approach only makes sense for quite small values of the field size i.e. only when $q < 4$. In this parameter regime, however, combinatorial attacks perform significantly better, so this approach becomes irrelevant.

6.3 Leon-like algorithm adapted to the rank metric

Leon [38] proposed an algorithm against the code equivalence problem in the Hamming metric that relies on the basic property that isometries preserve the weight of the codewords and that the weight distribution of two equivalent codes is the same. Thus, finding the set of codewords of smallest weight in both codes reveals enough information to find a permutation that maps one set to the other, which with high probability is the unknown isometry between the codes. This algorithm is quite unbalanced and heavy on the 'codewords finding' side, since it requires finding all codewords of minimal weight. Beullens [13] proposed to relax the procedure and instead perform a collision based algorithm, much in the spirit of Algorithm 1: Build two lists of elements of the codes of particular weight (the distinguishing property from [13] actually also includes the multiset of entries of a codeword) and find a collision between them. As in Leon's algorithm and Algorithm 1, the 'collision finding' part employs an efficient subroutine for reconstructing the isometry.

The approach from the Hamming metric can be translated to matrix codes and can be used to solve MCE, but some adjustments are necessary. First of all note that finding codewords of a given rank r is equivalent to an instance of MinRank [22, 29] for k matrices of size $m \times n$ over $\mathbb{F}_q$. Depending on the parameters, we have noticed that the kernel method or the Kipnis-Shamir modelling perform the best, so we use both in our complexity estimates.

For the collision part, notice that given two codewords $\mathbf{C}_1$ from $\mathcal{C}$ and $\mathbf{D}_1$ from $\mathcal{D}$, it is not possible to determine the isometry $(\mathbf{A}, \mathbf{B})$, as there are many isometries possible between single codewords. Thus, there is no efficient way of checking that these codewords collide nor finding the correct isometry. On the other hand, a pair of codewords is typically enough. For the pairs $(\mathbf{C}_1, \mathbf{C}_2)$ and $(\mathbf{D}_1, \mathbf{D}_2)$ we can form the system of $2mn$ linear equations

$$\begin{cases} \mathbf{A}^{-1}\mathbf{D}_1 = \mathbf{C}_1\mathbf{B} \\ \mathbf{A}^{-1}\mathbf{D}_2 = \mathbf{C}_2\mathbf{B} \end{cases} \tag{5}$$

in the $m^2 + n^2$ unknown coefficients of $\mathbf{A}$ and $\mathbf{B}$. When $m = n$, which is a typical choice, the system is expected to be overdetermined, and thus solved in $\mathcal{O}(n^6)$. In practice, and since $\mathbf{C}_1$, $\mathbf{C}_2$, $\mathbf{D}_1$ and $\mathbf{D}_2$ are low-rank codewords, there are fewer than $2n^2$ linearly independent equations, so instead of a unique solution, we can obtain a basis of the solution space. However, the dimension of the solution space is small enough so that coupling this technique with one of the algebraic modelings in Section 6.2 results in a system that can be solved through direct linearization. It is then easy to check whether the obtained isometry maps $\mathcal{C}$ to $\mathcal{D}$. We will thus assume, as a lower bound, that we find collisions between pairs of codewords.

Now, let $C(r)$ denote the number of codewords of rank r in a k-dimensional $m \times n$ matrix code. Then, using a birthday argument, two lists of size $\sqrt{2C(r)}$ of rank r codewords of $\mathcal{C}$ and $\mathcal{D}$ are enough to find two collisions. To detect the two collisions, we need to generate and solve systems as in Equation (5) for all possible pairs of elements from the respective lists, so $\left(\frac{\sqrt{2C(r)}}{2}\right)^2$ systems in total. Since $C(r) \approx q^{r(n+m-r)-nm+k}$, the total complexity amounts to

$$\mathcal{O}(q^{2(r(n+m-r)-nm+k)}(m^2+n^2)^{\omega}).$$

Note that a deterministic variant of this approach has the same asymptotic complexity. Choosing two rank r codewords of $\mathcal{C}$ and checking them for a 2-collision against all pairs of rank r codewords of $\mathcal{D}$ requires solving $\binom{C(r)}{2}$ systems.

Finally, we choose r so that both parts – the MinRank and the collision part are as close to a balance as possible. Section 7 discuses further the complexity of this approach for the chosen parameters of our scheme.

When considering the quantum version of the algorithm, we apply the same reasoning as in the case of the collision based Algorithm 1, and obtain quadratic speedup in the collision part. Because hybridization is also possible for the MinRank part, it can also benefit from using Grover, especially for larger fields.

6.4 Attacks on IMCE

The security of the linkable version of the ring signature scheme relies on the hardness of the IMCE problem. Thus, given three codes $\mathcal{C}$, $\mathcal{D}_1$ and $\mathcal{D}_2$, we need to determine whether they form a triangle of the following form, where the full lines represent the exact mappings the problem asks for, whereas the dashed ones are mappings that if found also break the problem. (The diagram uses loose notation in favor of readability.)

$$\begin{tikzcd}
& \mathcal{D}_1 \\
\mathcal{C} \arrow[ur, "\pi_{\mathbf{A},\mathbf{B}}"] \arrow[dr, "\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}"] & \\
& \mathcal{D}_2 \arrow[uu, dashed, "\pi_{\mathbf{A}^2,\mathbf{B}^2}"'] \arrow[ul, dashed, bend left, "\pi_{\mathbf{A},\mathbf{B}}"]
\end{tikzcd}$$

Extended collision-search algorithm. Our extended collision-search attack consists of finding an isometry between *any* of the three codes. Specifically, finding a collision between $\mathcal{C}$ and $\mathcal{D}_1$ allows us to derive $\pi_{\mathbf{A},\mathbf{B}}$, one between $\mathcal{C}$ and $\mathcal{D}_2$ yields $\pi_{\mathbf{A}^{-1},\mathbf{B}^{-1}}$, and one between $\mathcal{D}_1$ and $\mathcal{D}_2$ yields $\pi_{\mathbf{A}^2,\mathbf{B}^2}$. We first reduce the problem to an equivalent QMLE-based problem: solve any of the three QMLE instances $(k, m+n, \mathcal{F}, \mathcal{P}_1)$, $(k, m+n, \mathcal{F}, \mathcal{P}_2)$ or $(k, m+n, \mathcal{P}_2, \mathcal{P}_1)$, with

$$\mathcal{P}_1(\mathbf{x}) = (\mathcal{F}(\mathbf{xS}))\mathbf{T_1},\ \mathcal{P}_2(\mathbf{x}) = (\mathcal{F}(\mathbf{xS}^{-1}))\mathbf{T_2},\ \mathcal{P}_1(\mathbf{x}) = (\mathcal{P}_2(\mathbf{xS}^2))\mathbf{T_2}^{-1}\mathbf{T_1}$$

and $\mathbf{S} = \begin{bmatrix}\mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^\top\end{bmatrix}$. Then, we apply a modified version of Algorithm 1, where we now build three lists instead of two, and on Line 11, we pick pairs $(\mathbf{x}, \mathbf{y})$ from $\{L_1 \times L_2\} \cup \{L_1 \times L_3\} \cup \{L_2 \times L_3\}$, instead of just $\{L_1 \times L_2\}$.

To derive the complexity of the attack, denote by $N = dq^{m+n}$ the total number of distinguished elements. Then it can be deduced, using a birthday based analysis that the probability of not having a collision when the lists contain ℓ elements each is $P = \prod_{i=1}^{\ell}(1 - \frac{i}{N})(1 - \frac{2i}{N}) \approx e^{-\frac{3\ell^2}{2N}}$. To have a 63% chance of collision, we need to build lists of size $\ell = c_2\sqrt{N}$, with $c_2 = \sqrt{\frac{2}{3}}$. For comparison, performing a similar analysis for the collision-search algorithm for the general MCE problem (where we have only two lists), this constant is $c_1 = \sqrt{2}$.

Remark 4. For Leon's algorithm for IMCE, the arguments are similar to the extended collision search algorithm. Thus it seems that apart from possibly a constant speed-up, the algorithm does not benefit from the IMCE context.

Extended algebraic attack. Algebraically, IMCE can be treated in exactly the same manner as the minors modeling of MCE in the previous section. The types of equations are the same, and with the same structure. The only difference is that the algebraic system that we construct has twice as many equations, i.e $2(mn - k)k$, coming from the isometry between $\mathcal{C}$ and D_1 and between $\mathcal{D}_2$ and $\mathcal{C}$ but the amount of variables is the same – $n^2 + m^2$, since the isometry is the same. This means that the cost of the attack is reduced for the same choice of parameters, so adding the linkability property requires larger parameters in the regime where the algebraic attack performs the best.

7 Implementation and Evaluation

In this section we give an assessment of the performance of MEDS. We begin with important considerations about the implementation of the signature scheme. Then we provide concrete parameter choices for MEDS and a first preliminary evaluation of its performance based on a C reference implementation.

7.1 Implementation

Besides performance, the security of a cryptographic implementation is of crucial importance. By "implementation security" here we mean the resilience of an implementation against threats such as timing attacks, physical side-channel attacks, and fault injection attacks. While the requirement for side-channel and fault attacks heavily depends on whether in practice physical access is possible for an attacker, it is widely considered best practice to provide protection against timing attacks as baseline for all cryptographic implementations. In order to do this, the implementation must be *constant time*, i.e., timing variations based on secret data must be prevented. This typically means that branching and memory access based on secret data must be avoided. There are generic constructions to achieve timing security for basically any given cryptographic scheme. However, if the design of a cryptographic primitive does not take constant-time requirements into consideration, such generic constructions can be computationally expensive. Therefore, defining a cryptographic scheme such that it supports an efficient protection against timing attacks can make it easier to implement the scheme securely which in turn can make a scheme more secure and efficient in practice.

In the case of MEDS, we need to consider the implementation security of key generation and signing. Verification only operates on public data and hence does not require further protection. The basic operations of MEDS during key generation and signing are:

– field arithmetic,
– matrix multiplication,
– generating random invertible matrices, and
– computing a canonical form of a matrix.

All these operations must be implemented securely.

Field arithmetic. We are using a relatively small prime field in MEDS. Hence, for finite field addition and multiplication, we can simply perform integer arithmetic followed by a reduction modulo the prime. On most architectures, constant-time operations for adding and multiplying small operands are available. The reduction modulo the prime can be implemented using standard approaches in literature (e.g., by Granlund and Montgomery [33], Barrett [12], or Montgomery [41]). Inversion of field elements can efficiently be implemented in constant time using Fermat's little theorem and optimal addition chains.

In the C reference implementation, we simply use the modulo operation for reduction and Fermat's little theorem for inversion to get a first impression on the performance of the scheme. For using MEDS in practice, the timing side-channel security in particular of the modulo operation needs to be verified.

Matrix multiplication. The basic schoolbook algorithm for multiplying matrices is not data depended and hence constant time. More sophisticated approaches like the method by Arlazarov, Dinic, Kronrod, and Faradžev [6] may depend on potentially secret data, but most likely do not significantly improve the performance for the finite field and the matrix sizes used in MEDS. Asymptotically more efficient matrix multiplication algorithms likely are not more efficient for the given matrix dimensions neither. Hence, for the C reference implementation, we are simply using schoolbook multiplication. If a more efficient algorithm is used for optimized implementations, it must be ensured that a constant time algorithm is used.

Random invertible matrix generation. On several occasions throughout the MEDS operations, we need to generate random invertible matrices: For the partially seeded key generation (cf. Section 5) we need to compute an invertible matrix $\mathbf{T} \in \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$ and for signing (cf. step 1i. in Figure 5) we need to generate potentially secret matrices $\tilde{\mathbf{A}}_i, \tilde{\mathbf{B}}_i \in \mathrm{GL}_m(q) \times \mathrm{GL}_n(q)$. (During verification we need to recompute $\tilde{\mathbf{A}}_i$ and $\tilde{\mathbf{B}}_i$ for cases where $h_i = 0$.)

There are several approaches for generating random invertible matrices:

1. *Trial-and-error approach:* Generate a random matrix $\mathbf{M}$ and attempt to compute its inverse. If $\mathbf{M}$ does not have an inverse, try again with a new

random matrix. This approach requires constant-time Gaussian elimination if $\mathbf{M}$ needs to be kept secret and it might require several attempts before a random invertible matrix has been found.
2. *Constructive approach:* Construct a matrix $\mathbf{M}$ that is known to be inveritble. One approach for this is described in [48]: Generate a random lower-left triangular matrix $\mathbf{L}$ with the diagonal all 1 and an upper-right triangular matrix $\mathbf{U}$ with the diagonal all $\neq 0$ as well as a corresponding permutation matrix $\mathbf{P}$ akin to the result of an LUP decomposition. Then compute the random invertible matrix $\mathbf{M}$ as $\mathbf{M} = \mathbf{P}^{-1}\mathbf{LU}$.
Since generation of and multiplication with $\mathbf{P}$ are expensive to implement in constant time, one can follow the approach of [52], leave out $\mathbf{P}$, and compute $\mathbf{M}$ as $\mathbf{M} = \mathbf{LU}$ directly. This, however, covers only a subset of $((q-1)/q)^n$ matrices of all invertibe matrices in $\mathbb{F}_q^{n\times n}$. The inverse $\mathbf{M}^{-1}$ of the matrix can then be computed as $\mathbf{M}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$.

The fastest approach in our experiments was the constructive approach following [52]. Hence, we are using the constructive approach in all cases.

Canonical matrix form. MEDS requires to compute a canonical form of matrices before hashing. However, during signing, this must be computed in constant time. Computing the reduced row-echelon form of a matrix in constant time is expensive. Canonical matrix forms that can be computed in constant time more efficiently include the *systematic form* and the *semi-systematic* form of a matrix, used, e.g., in Classic McEliece [4]. Both the systematic and the semi-systematic form are special cases of the reduced row-echelon form.

For the systematic form, all pivoting elements are required to reside on the left diagonal of the matrix, i.e., a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$ in systematic form has the shape $(\mathbf{I}_k|\mathbf{G}')$ with $\mathbf{G}' \in \mathbb{F}_q^{k\times(mn-k)}$ and $\mathbf{I}_k$ denoting the $k \times k$ identity matrix. The requirements for the semi-systematic form are more relaxed: Following [4, Sect. 2.2.1], we say that a matrix $\mathbf{G}$ is in (μ, ν)-semi-systematic form if $\mathbf{G}$ has r rows (i.e., no zero rows), the pivot element in row $i \leq r - \mu$ also is in column i and the pivot element in row $i > r - \mu$ is in a column $c \leq i - \mu + \nu$.

However, not all matrices admit a systematic or semi-systematic form. In this case, we need to restart the computation with new random data. The probability that a matrix $\mathbf{G} \in \mathbb{F}_q^{k\times mn}$, $k \leq mn$ is full rank is $\prod_{i=1}^{k}(q^{mn} - q^{i-1})/q^{kmn}$. Therefore, the probability that $\mathbf{G}$ has a systematic form is $\prod_{i=1}^{k}(q^k - q^{i-1})/q^{k^2}$ and the probability that it has a semi-systematic form is $\prod_{i=1}^{k-\mu}(q^k - q^{i-1})/q^{(k-\mu)k} \cdot \prod_{i=1}^{\mu}(q^\nu - q^{i-1})/q^{\mu\nu}$. The probability and the cost of constant-time implementation for the semi-systematic form depend on μ and ν.

This gives us the following three options for avoiding to compute a reduced row-echelon form in constant time for $\tilde{\mathbf{G}}_i$ during signing:

1. *Basis change:* After computing $\tilde{\mathbf{G}}_i' = \pi_{\tilde{\mathbf{A}}_i,\tilde{\mathbf{B}}_i}(\mathbf{G}_0)$, perform a basis change $\tilde{\mathbf{G}}_i'' = \mathbf{M}_i\tilde{\mathbf{G}}_i'$ with a secret random invertible matrix $\mathbf{M}_i \in \mathbb{F}_q^{k\times k}$. Then compute a canonical form $\tilde{\mathbf{G}}_i = \mathsf{SF}(\tilde{\mathbf{G}}_i'')$ on a public $\tilde{\mathbf{G}}_i''$ without the need for a constant time implementation. This removes the requirement of a constant

$\lceil \log_2 q \rceil$	$n = k$	Birthday	Algebraic	Leon	SIG
9	16	235.29	181.55	131.20	13 296
9	17	249.04	194.55	149.65	16 237
10	15	244.62	174.75	130.50	12 428
11	14	250.79	160.24	131.21	12 519
12	14	272.40	160.24	141.17	13 548
13	**13**	**274.10**	**146.76**	**130.41**	**11 586**
14	13	294.10	146.76	134.41	13 632
20	12	383.75	138.46	135.40	16 320

Table 1. Cost of the investigated attacks in terms of finite field operations in log scale, and 'SIG' for 'signature size in bytes. Preferred choice in bold.

time computation of the canonical form but introduces extra cost for the generation of and multiplication with a random invertible matrix $\mathbf{M}_i$. Instead of an invertible matrix $\mathbf{M}_i$, just a random matrix can be used. With low probability (see above), a random matrix does not have full rank and the computation of the canonical form fails. In that case, the process needs to be restarted with a different $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$.

2. *Semi-systematic form:* Compute the semi-systematic form. This requires a less expensive constant-time implementation than for the reduced-row-echelon form. However, computing the canonical form might fail if no semi-systematic form exists, in which case the computation needs to be restarted.
3. *Systematic form:* Compute the systematic form. This can be implemented even more easily and cheaper in constant time than computing the semi-systematic form. However, systemization failure is more frequent and it is more likely that computations need to be restarted.

We performed generic experiments to investigate the performance impact of these variants. Our experiments indicate that the implementation cost for variant 1 is higher than that of the other two. For the specific parameter sets we propose in Section 7.2, the probability that $\tilde{\mathbf{G}}_i$ does not have a systematic form is about 0.0015%. Therefore, even though the failure probability can be reduced by computing the semi-systematic form compared to the systematic form with well-chosen μ and ν, the overall overhead (including cost for constant-time implementation) of computing the semi-systematic form (variant 2) is likely higher than the overall overhead for of computing the systematic form (variant 3). Hence, we decided to use the systematic form throughout as canonical form.

7.2 Parameter choice and evaluation

A summary of the cost of the three different attacks described in Section 6 is given in Table 1. First, we decide to set $n = k$, as this seems to be the Goldilocks zone for our scheme. For k larger, the algebraic attack becomes significantly better, and the same is true for Leon's attack when k is smaller. Then, for finite fields of different sizes, we find the smallest value of n that achieves the required

Parameter Set	q	n	m	k	s	t	w	ST	PK	SIG	FS
MEDS-2826-st	8191	13	13	13	2	256	30	✓	2826	18 020	−129.74
MEDS-8445-st-f	8191	13	13	13	4	160	23	✓	8445	13 946	−128.01
MEDS-8445-st	8191	13	13	13	4	464	17	✓	8445	10 726	−128.76
MEDS-8445-st-s	8191	13	13	13	4	1760	13	✓	8445	8702	−128.16
MEDS-11255-st	8191	13	13	13	5	224	19	✓	11 255	11 618	−128.45
MEDS-11255	8191	13	13	13	5	224	19	–	11 255	13 778	−128.45
MEDS-42161-st	8191	13	13	13	16	128	16	✓	42 161	9616	−128.85
MEDS-356839-st	8191	13	13	13	128	80	12	✓	356 839	7288	−129.64
MEDS-716471-st	8191	13	13	13	256	64	11	✓	716 471	6530	−127.37

Table 2. Parameters for MEDS, for $\lambda = 128$ bits of classical security. 'ST' for seed tree. 'PK' for 'public key size' and 'SIG' for 'signature size in bytes, 'FS' for 'Fiat-Shamir' probability logarithmic to base 2.

Parameter Set	Key Generation		Signing		Verification	
	(ms)	(mcyc.)	(ms)	(mcyc.)	(ms)	(mcyc.)
MEDS-2826-st	71.13	135.14	102.79	195.30	98.00	186.21
MEDS-8445-st-f	211.45	401.75	63.21	120.09	60.14	114.27
MEDS-8445-st	211.35	401.57	185.68	352.79	178.42	339.01
MEDS-8445-st-s	211.77	402.36	697.00	1324.31	673.19	1279.05
MEDS-11255-st	258.18	490.54	88.12	167.44	84.47	160.48
MEDS-11255	258.99	492.08	88.19	167.56	84.50	160.56
MEDS-42161-st	969.97	1842.95	50.54	96.03	48.42	92.00
MEDS-356839-st	8200.83	15 581.58	31.63	60.10	32.38	61.52
MEDS-716471-st	18 003.07	34 205.83	25.57	48.58	28.94	54.98

Table 3. Performance of MEDS in time (ms) and mega cycles (mcyc.) at 1900 MHz on an AMD Ryzen 7 PRO 5850U CPU following the SUPERCOP setup[9] computed as median of 16 randomly seeded runs each.

security level of 128 bits. We see that Leon's algorithm performs the best in most cases, although the algebraic approach is almost as good. Finally, to determine the optimal value for q, we choose the optimization parameters (s, t, and w) such that the sizes of the public key and the signature are comparable, and we report the signature size in the last column of Table 1. We conclude that the sweet spot for 128-bit security is given for the 13-bit prime $q = 8191$ and $n = k = 13$.

Remark 5. Given these parameters, we heuristically assume that the automorphism group of the codes is trivial with overwhelming probability. It is computationally infeasible to compute the automorphism group of codes of this size; however, data on smaller-sized codes shows that the probability of a random code having a trivial automorphism group grows rapidly as q, n, and m increase.

In this setting, we can vary s, t, and w for different trade-offs of public key and signature sizes as well as performance. We also checked the impact of q if we

[9] `https://bench.cr.yp.to/supercop.html`

aim for small public keys or small signatures (instead if balancing these two as in Table 1). In such cases, both 11-bit and 13-bit primes for q seem to perform similarly well. Hence, we stick to the 13-bit prime $q = 8191$ in our discussion.

Table 2 provides an overview of 128-bit security parameters for MEDS, highlighting different performance and key/signature size trade-offs. The best attack for all parameter set based on $q = 8191$, $n = 13$, and $k = 13$ is the Leon-like attack as shown in Table 1 with an expected cost of slightly over 2^{130} operations. The best quantum attack is obtained by Groverizing Leon's algorithm and has a cost of around 2^{88} operations. We select s, t, and w such that the probability of an attack on the Fiat-Shamir construction is around 2^{-128}. To improve the efficiency of vectorized implementations using SIMD instructions in the future, we select t as multiple of 16. In general, we are using all optimizations discussed in Section 5. However, we provide one parameter set without using the seed tree (without '-st' in the name of the parameter set).

Table 3 shows the resulting performance of these parameter sets from our constant-time C reference implementation on an AMD Ryzen 7 PRO 5850U CPU. The C reference implementation follows the implementation discussion above but does not apply any further algorithmic or platform-specific optimizations. We expect that optimized and vectorized implementations can significantly increase the performance.

The parameter set MEDS-2826-st with $s = 2$ provides the smallest public key with about 2.8 kB and a signature of about 18 kB. MEDS-8445-st increases the public key size with $s = 4$ to slightly over 8 kB while reducing the signature size to about 10.5 kB. MEDS-8445-st-f is a 'fast' variant of this parameter set with a smaller $t = 160$ but a larger $w = 23$, resulting in a larger signature size of about 14 kB. MEDS-8445-st-s is 'small' and goes the opposite direction, providing a smaller signature size of about 8.5 kB due to a smaller $w = 13$ at a larger computational cost due to $t = 1760$. These three sibling parameter sets illustrate the impact of t and w on performance and signature size.

MEDS-11255-st provides balanced public key and signature sizes, with both around 11 kB, and a small sum of signature and public key size at moderate computational cost for signing and verification due to $t = 224$. Removing the seed tree optimization comes with an increase in signature size of about 2 kB, which illustrates the impact of the seed tree.

Finally, sets MEDS-42161-st, MEDS-356839-st, and MEDS-716471-st push the public key size to an extreme at the expense of key generation time in the pursue of reducing signature size and computational cost for signing and verification. However, we expect that at least the key generation time can significantly be improved by optimizing the computation of solving the medium-size sparse linear system used for partially seeding the public key.

Overall, Table 2 and Table 3 highlight the large degree of flexibility offered by the MEDS scheme. All parameter sets are competitive with respect to existing literature, such as the LESS-FM scheme.

Finally, we discuss performance for the ring signature scheme. With the MEDS-2696-st parameter set, the size of a signature is in fact given by approxi-

Scheme	pk size (byte)	sig size (byte)	key gen (mcyc.)	sign (mcyc.)	verify (mcyc.)
ed25519 (scop)	32	64	0.0484	0.0513	0.182
[39] dilithium2 (scop)	1 312	2 420	0.151	0.363	0.163
[47] falcon512dyn (scop)	897	666	19.5	0.880	0.0856
[36] sphincsf128shake256 (scop)	32	16 976	6.86	220	9.91
[36] sphincss128shake256 (scop)	32	8 080	218	3 500	4.04
[14] UOV ov-Ip	278 432	128	2.90	0.105	0.0903
[10] LESS-I	8 748	12 728	—	—	—
[7] Wavelet	3 236 327	930	7 400	1 640	1.09
[2] SDitH Var3f	144	12 115	—	4.03	3.04
[2] SDitH Var3sss	144	5 689	—	994	969
MEDS-8445-st-f	8 445	13 914	402	120	114
MEDS-11255-st	11 255	11 586	491	168	160
MEDS-42161-st	42 161	9 584	1 840	96.0	92.0
MEDS-716471-st	716 471	6 498	34 200	48.6	55.0

Table 4. Performance comparison to other relevant schemes (mcyc. rounded to three significant figures). Data marked with '(scop)' is from the SUPERCOP website. For SPHINCS+ we list results for the 'simple' variant.

mately $(\log_2 r + 19.26)$ kB; in the linkable case, this increases to $(\log_2 r + 39.22)$ kB. These numbers are close to the lattice instantiation (Falafl) given in [15]; judging by the timings in Table 3, we also expect it to be much faster than the isogeny instantiation (Calamari). On the other hand, the work of [11] obtains smaller sizes, but does not propose a linkable variant. Note that both sizes and timings can be improved by choosing dedicated parameters and designing an ad-hoc implementation, which we leave as part of a future work.

7.3 Comparison to related signature schemes

Table 4 shows a comparison of public key and signature sizes as well as computational performance of our new MEDS scheme with some established schemes and related recent proposals. While the comparison of public key and signature sizes is accurate, the comparison of the performance needs to be taken with a large grain of salt: While we provide numbers in the same performance metric (mega cycles – mcyc.), a direct comparison is still quite hard since not all schemes have received the same degree of optimization and since not all performance data has been obtained on the same CPU architecture.

The performance data from the 'classical' scheme ed25519 as well as from the NIST PQC schemes CRYSTALS-Dilithium [39], Falcon [47], and SPHNICS+[36] has been obtained from the SUPERCOP website[10]. We selected the performance data from the AMD64 Zen CPU, which is an AMD Ryzen 7 1700 from 2017,

[10] `https://bench.cr.yp.to/results-sign.html` – amd64; Zen (800f11); 2017 AMD Ryzen 7 1700; 8 x 3000MHz; rumba7, supercop-20220506

i.e., the same microarchitecture (but a different CPU) as we used for our measurements of MEDS. We are reporting median cycles directly from the website.

For UOV [14], LESS [10] and Wavelet [7] we list the performance data as reported in the respective papers unless such data was unavailable. In the case of SDitH [2], only reports of performance data in milliseconds on a 3.1 GHz Intel Core i9-9990K are available. We computed the corresponding number of cycles from this to enable a rough comparison to the other schemes, but note that this data is therefore not entirely accurate.

Table 4 shows that, although code-based schemes do not compete well with pre-quantum or lattice-based PQC schemes, MEDS fills a gap that was not previously available for multivariate or code-based schemes, with a relatively small combined size of public key and signature. Furthermore, its versatility in parameter selection allows for great flexibility for specific applications. In terms of performance, the current implementation of MEDS is still unoptimized. We expect speed-ups of at least one order of magnitude from SIMD parallelization on AVX256 and AVX512 CPUs, since both the data-independent loop of the Fiat-Shamir construction and the matrix arithmetic lend themselves to efficient parallelization. Providing optimized implementations of MEDS for modern SIMD architectures as well as embedded systems is an open task for future work.

References

[1] C. Aguilar Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, E. Persichetti, G. Zémor, and J. Bos. HQC. NIST PQC Submission, 2020.

[2] C. Aguilar-Melchor, N. Gama, J. Howe, A. Hülsing, D. Joseph, and D. Yue. The return of the SDitH. Cryptology ePrint Archive, Paper 2022/1645, 2022. To appear at Eurocrypt 2023.

[3] N. Alamati, L. De Feo, H. Montgomery, and S. Patranabis. Cryptographic group actions and applications. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020*, volume 12492 of *LNCS*, pages 411–439. Springer, 2020.

[4] M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. von Maurich, R. Misoczki, R. Niederhagen, K. G. Paterson, E. Persichetti, C. Peters, P. Schwabe, N. Sendrier, J. Szefer, C. J. Tjhai, M. Tomlinson, and W. Wang. Classic McEliece. NIST PQC Submission, 2020.

[5] N. Aragon, P. Barreto, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Guneysu, C. Aguilar Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, G. Zémor, V. Vasseur, and S. Ghosh. BIKE. NIST PQC Submission, 2020.

[6] V. Arlazarov, E. Dinic, M. Kronrod, and I. Faradžev. On Economical Construction of the Transitive Closure of an Oriented Graph. In *Doklady Akademii Nauk*, volume 194, pages 487–488. Russian Academy of Sciences, 1970.

[7] G. Banegas, T. Debris-Alazard, M. Nedeljković, and B. Smith. Wavelet: Code-based postquantum signatures with fast verification on microcontrollers. Cryptology ePrint Archive, Paper 2021/1432, 2021.

[8] F. Bao, R. H. Deng, and H. Zhu. Variations of diffie-hellman problem. In S. Qing, D. Gollmann, and J. Zhou, editors, *ICICS 2003*, volume 2836 of *LNCS*, pages 301–312. Springer, 2003.

[9] M. Bardet, M. Bros, D. Cabarcas, P. Gaborit, R. A. Perlner, D. Smith-Tone, J. Tillich, and J. A. Verbel. Improvements of Algebraic Attacks for Solving the Rank Decoding and MinRank Problems. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 507–536. Springer, 2020.

[10] A. Barenghi, J. Biasse, E. Persichetti, and P. Santini. LESS-FM: fine-tuning signatures from the code equivalence problem. In J. H. Cheon and J. Tillich, editors, *PQCrypto 2021*, volume 12841 of *LNCS*, pages 23–43. Springer, 2021.

[11] A. Barenghi, J.-F. Biasse, T. Ngo, E. Persichetti, and P. Santini. Advanced signature functionalities from the code equivalence problem. *Int. J. Comput. Math. Comput. Syst. Theory*, 7(2):112–128, 2022.

[12] P. Barrett. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. In A. M. Odlyzko, editor, *CRYPTO' 86*, pages 311–323. Springer Berlin Heidelberg, 1987.

[13] W. Beullens. Not enough LESS: an improved algorithm for solving code equivalence problems over $\mathbb{F}_q$. In O. Dunkelman, M. J. Jacobson, and C. O'Flynn, editors, *SAC 2020*, volume 12804 of *LNCS*, pages 387–403. Springer, 2020.

[14] W. Beullens, M.-S. Chen, S.-H. Hung, M. J. Kannwischer, B.-Y. Peng, C.-J. Shih, and B.-Y. Yang. Oil and vinegar: Modern parameters and implementations. Cryptology ePrint Archive, Paper 2023/059, 2023.

[15] W. Beullens, S. Katsumata, and F. Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020*, volume 12492 of *LNCS*, pages 464–492. Springer, 2020.

[16] W. Beullens, T. Kleinjung, and F. Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In S. D. Galbraith and S. Moriai, editors, *ASIACRYPT 2019*, volume 11921 of *LNCS*, pages 227–247. Springer, 2019.

[17] J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini. LESS is More: Code-Based Signatures Without Syndromes. In A. Nitaj and A. Youssef, editors, *AFRICACRYPT 2020*, volume 12174 of *LNCS*, pages 45–65. Springer, 2020.

[18] X. Bonnetain and A. Schrottenloher. Quantum security analysis of CSIDH. In A. Canteaut and Y. Ishai, editors, *EUROCRYPT 2020*, volume 12106 of *LNCS*, pages 493–522. Springer, 2020.

[19] C. Bouillaguet. *Algorithms for some hard problems and cryptographic attacks against specific cryptographic primitives.* PhD thesis, Université Paris Diderot, 2011.

[20] C. Bouillaguet, P. Fouque, and A. Véber. Graph-theoretic algorithms for the "isomorphism of polynomials" problem. In T. Johansson and P. Q.

Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 211–227. Springer, 2013.

[21] W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: an efficient post-quantum commutative group action. In T. Peyrin and S. D. Galbraith, editors, *ASIACRYPT 2018*, volume 11274 of *LNCS*, pages 395–427. Springer, 2018.

[22] N. T. Courtois. Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 402–421. Springer, 2001.

[23] N. T. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In B. Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 392–407. Springer, 2000.

[24] J.-M. Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Paper 2006/291, 2006.

[25] A. Couvreur, T. Debris-Alazard, and P. Gaborit. On the hardness of code equivalence problems in rank metric. *CoRR*, abs/2011.04611, 2020.

[26] L. De Feo and S. D. Galbraith. SeaSign: Compact isogeny signatures from class group actions. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019*, volume 11478 of *LNCS*, pages 759–789. Springer, 2019.

[27] J. Ding, M.-S. Chen, A. Petzoldt, D. Schmidt, B.-Y. Yang, M. Kannwischer, and J. Patarin. Rainbow. Technical report, National Institute of Standards and Technology, 2020.

[28] J.-C. Faugère, M. S. E. Din, and P.-J. Spaenlehauer. Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1, 1): Algorithms and complexity. *J. Symb. Comput.*, 46(4):406–437, 2011.

[29] J.-C. Faugère, F. L. dit Vehel, and L. Perret. Cryptanalysis of MinRank. In D. A. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 280–296. Springer, 2008.

[30] J.-C. Faugère and L. Perret. Polynomial equivalence problems: Algorithmic and theoretical aspects. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 30–47. Springer, 2006.

[31] J. Felderhoff. Hard Homogenous Spaces and Commutative Supersingular Isogeny based Diffie-Hellman. Internship report, LIX, Ecole polytechnique and ENS de Lyon, Aug. 2019.

[32] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[33] T. Granlund and P. L. Montgomery. Division by invariant integers using multiplication. In *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, PLDI '94, page 61–72, New York, NY, USA, 1994. Association for Computing Machinery.

[34] S. Gueron, E. Persichetti, and P. Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptogr.*, 6(1):5, 2022.

[35] I. Haviv and O. Regev. On the lattice isomorphism problem. In C. Chekuri, editor, *SODA 2014*, pages 391–404. ACM SIAM, 2014.

[36] A. Hulsing, D. J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, P. Kampanakis, S. Kolbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, J.-P. Aumasson, B. Westerbaan, and W. Beullens. SPHINCS+. NIST PQC Submission, 2020.

[37] G. Kuperberg. Another subexponential-time quantum algorithm for the dihedral hidden subgroup problem. In S. Severini and F. G. S. L. Brandão, editors, *TQC 2013*, volume 22 of *LIPIcs*, pages 20–34. Schloss Dagstuhl, 2013.

[38] J. S. Leon. Computing automorphism groups of error-correcting codes. *IEEE Trans. Inf. Theory*, 28(3):496–510, 1982.

[39] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, and S. Bai. CRYSTALS. NIST PQC Submission, 2020.

[40] R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. DSN PR 42-44, California Institute of Technology, Jan./Feb. 1978.

[41] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44:519–521, 1985.

[42] P. Nguyen and C. Wolf. International workshop on post-quantum cryptography, 2006.

[43] NIST. Post-Quantum Cryptography Standardization, 2017. URL: `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`.

[44] J. Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of asymmetric algorithms. In U. M. Maurer, editor, *EUROCRYPT '96*, volume 1070 of *LNCS*, pages 33–48. Springer, 1996.

[45] R. Perlner and D. Smith-Tone. Rainbow band separation is better than we thought. Cryptology ePrint Archive, Paper 2020/702, 2020.

[46] E. Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Information Theory*, 8(5):5–9, 1962.

[47] T. Prest, P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang. FALCON. NIST PQC Submission, 2020.

[48] D. Randall. Efficient Generation of Random Nonsingular Matrices. Technical Report UCB/CSD-91-658, EECS Department, UC Berkeley, 1991.

[49] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In H. N. Gabow and R. Fagin, editors, *Theory of computing*, pages 84–93. ACM, 2005.

[50] K. Reijnders, S. Samardjiska, and M. Trimoska. Hardness estimates of the code equivalence problem in the rank metric. Cryptology ePrint Archive, Paper 2022/276, 2022.

[51] A. Rostovtsev and A. Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Paper 2006/145, 2006.

[52] G. Tang, D. H. Duong, A. Joux, T. Plantard, Y. Qiao, and W. Susilo. Practical post-quantum signature schemes from isomorphism problems of trilinear

forms. In O. Dunkelman and S. Dziembowski, editors, *EUROCRYPT 2022*, volume 13277 of *LNCS*, pages 582–612. Springer, 2022.

A Formal security definitions

Definition 8. For two sets X and W, let R be a relation on $X \times W$. If $(x, w) \in R$, we say that w is the *witness* for the *instance* x. We define a Sigma protocol for the relation R as in Definition 2, with $\mathsf{pk} = x$ and $\mathsf{sk} = w$. We then define the following properties for such a Sigma protocol:

– *Completeness*: when $(x, w) \in R$, a honest prover is accepted with probability 1, i.e.

$$\Pr\left[\mathsf{V}_2(x, \mathsf{cmt}, \mathsf{ch}, \mathsf{rsp}) = 1 \;\middle|\; \begin{array}{c} \mathsf{cmt} \leftarrow \mathsf{P}_1(x) \\ \mathsf{ch} \xleftarrow{\$} C \\ \mathsf{rsp} \leftarrow \mathsf{P}_2(x, w, \mathsf{cmt}, \mathsf{ch}) \end{array}\right] = 1.$$

– 2-*Special Soundness*: there exists an extractor algorithm E such that, for an instance x, any two valid transcripts, $(\mathsf{cmt}, \mathsf{ch}_1, \mathsf{rsp}_1)$ and $(\mathsf{cmt}, \mathsf{ch}_2, \mathsf{rsp}_2)$, the algorithm output $\mathsf{E}(x, \mathsf{cmt}, \mathsf{ch}_1, \mathsf{rsp}_1, c_2, \mathsf{rsp}_2)$ is a witness for $\mathcal{R}$ with high probability. More formally, the probability

$$\Pr\left[(x, w) \in \mathcal{R} \,|\, w \leftarrow \mathsf{E}(x, \mathsf{cmt}, \mathsf{ch}_1, \mathsf{rsp}_1, \mathsf{ch}_2, \mathsf{rsp}_2)\right] = 1.$$

– *Honest-Verifier Zero-Knowledge*: for any $(x, w) \in R$, there exists a simulator S, with input only the public key x, which outputs a valid transcript $(\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp})$ in polynomial time, such that

$$\Pr\left[\mathsf{V}_2(x, \mathsf{cmt}, \mathsf{ch}, \mathsf{rsp}) = 1 \,|\, (\mathsf{cmt}, \mathsf{ch}, \mathsf{rsp}) \leftarrow \mathsf{S}(x)\right] = 1.$$

Moreover, the output distribution of S on input (x, ch) is equal to the distribution of those outputs generated via an honest execution, conditioned on the verifier using ch as the challenge.

Definition 9. A digital signature is expected to satisfy the following properties:

1. *Correctness*: A honest prover is accepted with probability 1, i.e.

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \sigma) = 1 \,|\, \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{msg})\right] = 1.$$

2. *Unforgeability*: The signature scheme is said to be unforgeable if any adversary $\mathcal{A}$, with access to any number of signature Σ, is not able to sign a new message except with a negligible probability. In other words, the probability

$$\Pr\left[\mathsf{Verify}(\mathsf{pk}, \mathsf{msg}, \sigma) = 1 \,|\, (\mathsf{msg}, \sigma) \notin \Sigma, \sigma \leftarrow \mathsf{AdvSign}(\mathsf{pk}, \mathsf{msg})\right]$$

is small.

Definition 10. A ring signature scheme is expected to satisfy the following properties:

1. *Correctness*: A ring signature is correct if for every ring of r signers, for every index $I \in \{1, \ldots, r\}$ and for every message msg, we have

$$\mathsf{Pr}\left[\mathsf{Verify}(K, \mathsf{msg}, \sigma) \middle| \begin{array}{c} (\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \mathsf{Keygen}, \forall j \in \{1, \ldots, r\}, \\ K := \{\mathsf{pk}_1, \ldots, \mathsf{pk}_r\}, \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{j^*}, \mathsf{msg}, K) \end{array}\right] = 1.$$

2. *Anonymity*: A ring signature is anonymous if for every ring of r signers, any adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:
   (A) The challenger first runs the algorithm Keygen to obtain r secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$, $i \in \{1, \ldots, r\}$. He samples a bit $b \xleftarrow{\$} \{0, 1\}$.
   (B) The challenger gives all the pk_j, $i = 1, \ldots, r$ to $\mathcal{A}$.
   (C) $\mathcal{A}$ sends to the challenger a challenge $(K, \mathsf{msg}, j_0, j_1)$. The set of public keys K must contain the public keys pk_{j_0} and pk_{j_1}. The challenger computes the signature $\sigma^* \leftarrow \mathsf{Sign}(\mathsf{sk}_{j_b}, \mathsf{msg}, K)$ and sends it to $\mathcal{A}$.
   (D) $\mathcal{A}$ outputs a bit b^* and wins if $b = b^*$.
3. *Unforgeability*: A ring signature is anonymous if for every ring of r signers, any adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:
   (A) The challenger first runs the algorithm Keygen to obtain r secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i)$, $i \in \{1, \ldots, r\}$. He calls $K' = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_r\}$ the set with the public keys. He finally initialises two empty sets S_1 and S_2.
   (B) The challenger gives the set K' to $\mathcal{A}$.
   (C) $\mathcal{A}$ can create signing and corruption queries a polynomial number of times:
      – $(\mathsf{AdvSign}, j, \mathsf{msg}, K)$: the challenger checks if $\mathsf{pk}_j \in K \subseteq K'$. If that is true, he computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{pk}_j, \mathsf{msg}, K)$. The challenger gives σ to $\mathcal{A}$ and adds (j, msg, K) to S_1.
      – $(\mathsf{AdvCorrupt}, j)$: the challenger adds pk_j to S_2 and returns sk_j to $\mathcal{A}$.
   (D) $\mathcal{A}$ outputs $(K^*, \mathsf{msg}^*, \sigma^*)$. If $K^* \subset K' \backslash S_2$, $(\cdot, \mathsf{msg}^*, K^*) \notin S_1$ and $\mathsf{Verify}(K^*, \mathsf{msg}^*, \sigma^*) = 1$, then the adversary wins.

Definition 11. A linkable ring signature scheme is expected to satisfy the following properties

1. *Linkability*: A linkable ring signature is called *linkable* if for every ring of r signers, any adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:
   (A) $\mathcal{A}$ runs the algorithm Keygen and outputs the set $K' = \{\mathsf{pk}_1, \ldots, \mathsf{pk}_r\}$ and the set of tuples $\{(\sigma_1, \mathsf{msg}_1, K_1), \ldots, (\sigma_{r+1}, \mathsf{msg}_{r+1}, K_{r+1})\}$.
   (B) $\mathcal{A}$ wins if the following three conditions hold:
      – $\forall j \in \{1, \ldots, r+1\}$, $K_i \subseteq K'$.
      – $\forall j \in \{1, \ldots, r+1\}$, the algorithm $\mathsf{Verify}(K_j, \mathsf{msg}_j, \sigma_i)$ outputs 1.

   – $\forall i, j \in \{1, \dots, r+1\}$ such that $i \neq j$, $\mathsf{Link}(\sigma_i, \sigma_j) = 0$.

2. *Linkable Anonymity*: A linkable ring signature is linkable anonymous if for every ring of r signers, any adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:
   (A) The challenger first runs the algorithm Keygen to obtain r key pairs $(\mathsf{sk}_j, \mathsf{pk}_j), j \in \{1, \dots, r\}$. He calls $K' = \{\mathsf{pk}_1, ..., \mathsf{pk}_r\}$ the set with the public keys. He samples a bit $b \xleftarrow{\$} \{0, 1\}$.
   (B) The challenger gives K' to $\mathcal{A}$.
   (C) The adversary chooses and outputs two public keys $(\mathsf{pk}_{i_0}, \mathsf{pk}_{i_1}) \in K'$. The corresponding secret keys are denoted by $(\mathsf{sk}_{i_0}, \mathsf{sk}_{i_1})$.
   (D) The challenger gives to $\mathcal{A}$ the set $\{(\mathsf{pk}_i, \mathsf{sk}_i) \colon 1 \leq i \leq r, i \notin \{i_0, i_1\}\}$.
   (E) A queries for signatures, giving as inputs to the challenger a public key $\mathsf{pk} \in \{\mathsf{pk}_{i_0}, \mathsf{pk}_{i_1}\}$, a message msg and a ring K that contains $\{\mathsf{pk}_{i_0}, \mathsf{pk}_{i_1}\}$:
      – If $\mathsf{pk} = \mathsf{pk}_{i_0}$, the challenger outputs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{i_b}, \mathsf{msg}, K)$.
      – If $\mathsf{pk} = \mathsf{pk}_{i_1}$, the challenger outputs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{1-b}, \mathsf{msg}, \mathbb{F}_q)$.
   (F) $\mathcal{A}$ outputs a bit b^*, and he wins the game if $b = b^*$.

3. *Non-Frameability*: A linkable ring signature is non-frameable if for every ring of r signers, any adversary $\mathcal{A}$ has at most a negligible advantage when playing the following game against a challenger:
   (A) The challenger first runs the algorithm Keygen to obtain r key pairs $(\mathsf{sk}_j, \mathsf{pk}_j), j \in \{1, \dots, r\}$. He calls $K' = \{\mathsf{pk}_1, ..., \mathsf{pk}_r\}$ the set with the public keys. He also intialises two empty sets S_1 and S_2.
   (B) The challenger gives the set K' to the adversary $\mathcal{A}$.
   (C) A can create signing and corruption queries a polynomial number of times:
      – $(\mathsf{AdvSign}, j, \mathsf{msg}, K)$: the challenger checks if $\mathsf{pk}_j \in K \subseteq K'$. If that is true, he computes $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_j, \mathsf{msg}, K)$. The challenger gives σ to $\mathcal{A}$ and adds (j, msg, K) to S_1.
      – $(\mathsf{AdvCorrupt}, j)$: the challenger adds pk_j to S_2 and returns sk_j to $\mathcal{A}$.
   (D) $\mathcal{A}$ outputs $(K^*, \mathsf{msg}^*, \sigma^*)$; he wins if the following conditions hold:
      – $\mathsf{Verify}(K^*, \mathsf{msg}^*, \sigma^*) = 1$ and $(\cdot, \mathsf{msg}^*, K^*) \notin S_1$;
      – $\mathsf{Link}(\sigma^*, \sigma) = 1$ for some signature σ given by the challenger starting from a query of the form $(i, \mathsf{msg}, K) \in S_1$ with $\mathsf{pk}_i \in K' \backslash S_2$.

   the adversary wins.