Hardness estimates of the Code Equivalence Problem in the Rank Metric

Krijn Reijnders, Simona Samardjiska and Monika Trimoska

Digital Security, Radboud University, Nijmegen, Netherlands.

Contributing authors: {krijn,simonas,mtrimoska}@cs.ru.nl;

Abstract

In this paper, we analyze the hardness of the Matrix Code Equivalence (**MCE**) problem for matrix codes endowed with the rank metric, and provide the first algorithms for solving it. We do this by making a connection to another well-known equivalence problem from multivariate cryptography - the Isomorphism of Polynomials (**IP**). Under mild assumptions, we give tight reductions from **MCE** to the homogenous version of the Quadratic Maps Linear Equivalence (**QMLE**) problem, and vice versa. Furthermore, we present reductions to and from similar problems in the sum-rank metric, showing that **MCE** is at the core of code equivalence problems. On the practical side, using birthday techniques known for **IP**, we present two algorithms: a probabilistic algorithm for **MCE** running in time $\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$, and a deterministic algorithm for **MCE** with roots, running in time $\mathcal{O}^*(q^m)$. Lastly, to confirm these findings, we solve randomly-generated instances of **MCE** using these two algorithms.

1 Introduction

Given two mathematical objects of the same type, an equivalence problem asks the question whether there exists an equivalence map between these objects – and how to find it – that preserves some important property of the objects. These kind of problems come in different flavors depending on the objects – groups, graphs, curves, codes, quadratic forms, etc. and quite often the interesting maps are isomorphisms or isometries. Interestingly, equivalence problems are one of the core hard problems underlying the security of many public-key cryptosystems, especially post-quantum ones. Many multivariate and code-based systems employ an equivalence transformation as a hiding technique, and thus intrinsically rely on the assumption that a particular equivalence problem is intractable, for example [1–6]. In addition, quite remarkably, a hard equivalence problem gives rise to a Sigma protocol and, through the Fiat-Shamir transform, a provably secure digital signature scheme [7]. This idea has been revisited many times, being the basis of several signature schemes [1, 8–12]. Two such schemes actually appeared during the writing of this manuscript [13, 14] as a result of NIST's announcement for an additional fourth round on signatures in the post quantum standardization process [15]. Understanding the hardness of these equivalence problems is an essential task in choosing appropriate parameters that attain a certain security level of these cryptographic schemes.

One of these problems is the Code Equivalence problem, which given two codes (with the Hamming metric), asks for an isometry (equivalence transformation that preserves the metric) that maps one code to the other. It was first studied by Leon [16] who proposed an algorithm that takes advantage of the Hamming weight being invariant under monomial permutations. It was improved very recently by Beullens [17] using collision-based techniques. Sendrier [18] proposed another type of algorithm, the Support Splitting Algorithm (SSA), that is exponential in the dimension of the hull (the intersection of a code and its dual). Interestingly, in low characteristic, random codes have very small hull, rendering the problem easy.

In this work, we focus on the code equivalence problem, but for matrix codes (an $\mathbb{F}_q$-linear subspace of the space of $m \times n$ matrices over $\mathbb{F}_q$) endowed with the rank metric - *Matrix Code Equivalence* (MCE). Evaluating the hardness of this problem is only natural – rank-based cryptography has become serious competition for its Hamming-based counterpart, showing superiority in key sizes for the same security level [19–22]. This problem, and variations of it, has been introduced by Berger in [23], but it was only recently that the first concrete statements about its hardness were shown in two concurrent independent works publicly available as preprints[1]. Couvreur et al. [25] showed that MCE is at least as hard as the (Monomial) Code Equivalence problem in the Hamming metric, while for only right equivalence, or when the codes are $\mathbb{F}_{q^m}$-linear, the problem becomes easy. Grochow and Qiao [24] show the same reduction from (Monomial) Code Equivalence to MCE but using a completely different technique of linear algebra coloring gadgets which makes the reduction looser than the one in [25].

[1] The two works use different techniques and terminology, and seem to be mutually unaware of the line of work preceding the other. In [24] the MCE problem is referred to as Matrix Space Equivalence problem and 3-Tensor Isomorphism problem.

1.1 Our contributions

In this paper, we investigate the theoretical and practical hardness of the Matrix Code Equivalence (MCE) problem. Our contributions can be summarized as follows:

First, we link in a straightforward manner the MCE problem to hard problems on systems of polynomials by showing that MCE is polynomial-time equivalent to the Bilinear Maps Linear Equivalence (BMLE) problem. We then extend this result by proving that MCE is polynomial-time equivalent to the Quadratic Maps Linear Equivalence (QMLE) problem, under a mild assumption of trivial automorphism groups of the codes in question. While our technique fails to give a proof without this assumption, we consider it to be reasonable for randomly generated codes and for cryptographic purposes. As the QMLE problem is considered to be the hardest equivalence problem for systems of multivariate polynomials, it is essential to understand under which conditions MCE and QMLE reduce to one another. Note that previous work[2] requires much stronger assumptions for related results [24, 26, 27], such as algebraically closed fields or existence of square or third roots. Our reduction to QMLE is tight and gives a tight upper bound on the hardness of MCE. Furthermore, it is very simple, thus establishing connection between code equivalence problems and polynomial equivalence problems that is usable in practice. This is the basis of our contributions on the practical hardness of MCE.

Second, using similar techniques, and under the same assumptions, we show that MCE is polynomial-time equivalent to other code equivalence problems, such as Matrix Sum-Rank Code Equivalence Problem, and at least as hard as the Vector Sum-Rank Code Equivalence Problem. All these connections and our results are visualized in Figure 1.

On the practical side, we provide the first two non-trivial algorithms for solving MCE using the connection to QMLE. The first algorithm is a generalization of a known birthday-based algorithm for QMLE [28, 29] for systems of polynomials with the same number of variables as equations. We show that this algorithm extends to different invariance properties and code dimensions, which helps us to prove a complexity for MCE of $\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$ for $m \times n$ matrix codes. The algorithm is probabilistic with 63% success probability, and can be used for code dimensions up to $2(m+n)$. For larger dimensions, the complexity becomes $\mathcal{O}^*(q^{(n+m)})$, but the algorithm is deterministic. The birthday-based algorithm for QMLE [28] assumed to existence of a polynomial-time solver for the inhomogeneous variant of QMLE to achieve these complexities. Interestingly, due to the specific instances of the inhomogeneous QMLE arising from the collision search, the problem seems to be much harder than for random instances – a fact previously overlooked in [28]. In contrast, [29] uses a non-polynomial estimate for this solver. We analyse the most recent results

[2] We were made aware of this line of work by one of the authors after our results were first presented at WCC 2022.

regarding such solvers, and show that for parameter sets of cryptographical interest the above complexities hold, even if such solvers do not achieve polynomial time.

Our second algorithm uses the bilinear structure of the polynomials arising from MCE and runs in time $\mathcal{O}^*(q^m)$. This algorithm is deterministic and does not require a polynomial-time solver for the inhomogeneous QMLE instance, but the weaker assumption that the solver has a complexity of $\mathcal{O}(q^m)$ at most. This algorithm works only for code dimensions up to $m+n$. We analyse the performance of solvers for these parameter sets, which shows that the specific assumption on the solver holds for codes of dimension m to $m+n$.

Lastly, to verify the results and performance of these algorithms in practice, we have implemented both and solved randomly generated instances of MCE for different parameter sets. The results of these experiments show that our assumptions are reasonable and the above complexities hold. Our implementations are open source and available at: https://github.com/mtrimoska/matrix-code-equivalence.

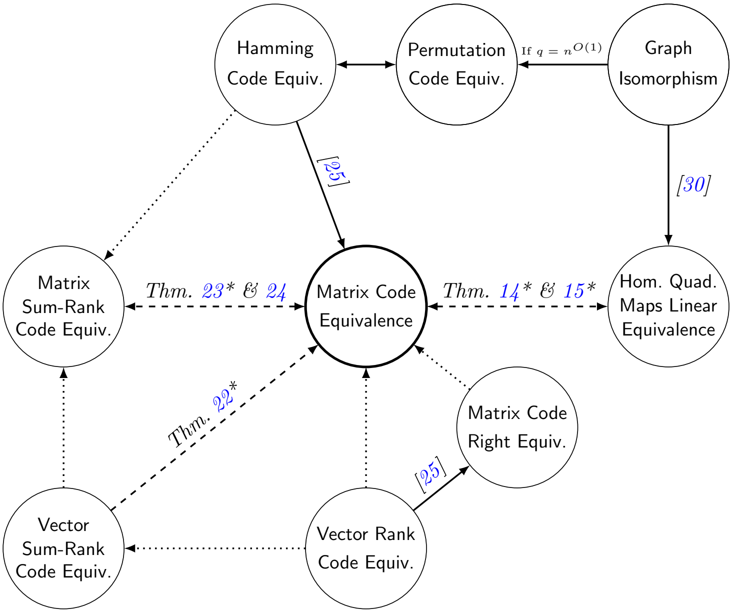

Figure 1: Reductions around Matrix Code Equivalence. Dashed arrows are contributions from this work, dotted arrows are trivial reductions. “A ⟶ B” means that “Problem A reduces to Problem B in polynomial time”. Results with * assume trivial automorphism groups.

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 $\mathbf{a}, \mathbf{c}, \mathbf{x}, \ldots$, and matrices $\mathbf{A}, \mathbf{B}, \ldots$. The entries of a vector $\mathbf{a}$ are denoted by a_i, and we write $\mathbf{a} = (a_1, \ldots, a_n)$ for a (row) vector of dimension n over some field and $\mathbf{a}^\top = (a_1, \ldots, a_n)^\top$ for the respective column vector. Similarly, the entries of a matrix $\mathbf{A}$ are denoted by A_{ij}. A matrix $\mathbf{A}$ is called symmetric if $\mathbf{A}^\top = \mathbf{A}$ and skew-symmetric if $\mathbf{A}^\top = -\mathbf{A}$. The space of matrices over $\mathbb{F}_q$ of size $m \times n$ is denoted $\mathcal{M}_{m,n}(q)$.

Random sampling from a set S is denoted by $a \xleftarrow{\$} S$. We use the notation $\mathcal{O}^*(\cdot)$ when polynomial factors of the complexity are omitted.

2.1 The Matrix Code Equivalence problem.

This section introduces basic notions on matrix codes and their equivalences. A more thorough introduction on matrix codes can be found in [31]. The usual choice for measuring distance between matrices over a finite field is the so called *rank metric*, defined as follows.

Definition 1 Let $\mathrm{Rank}(\mathbf{M})$ denote the rank of a matrix $\mathbf{M} \in \mathcal{M}_{m,n}(q)$. The *rank distance* between two $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$ over $\mathbb{F}_q$ is defined as

$$d(\mathbf{A}, \mathbf{B}) = \mathrm{Rank}(\mathbf{A} - \mathbf{B}).$$

An *isometry* is a map $\mu : \mathcal{M}_{m,n}(q) \to \mathcal{M}_{m,n}(q)$ that preserves the rank, i.e. $\mathrm{Rank}(\mu(\mathbf{M})) = \mathrm{Rank}(\mathbf{M})$ for all $\mathbf{M} \in \mathcal{M}_{m,n}(q)$.

By symmetry, without loss of generality, in the rest of the text we assume $n \geqslant m$.

Definition 2 A *matrix code* is a subspace $\mathcal{C}$ of $m \times n$ matrices over $\mathbb{F}_q$ endowed with the rank metric. Let k denote the dimension of $\mathcal{C}$ as a subspace of $\mathbb{F}_q^{m\times n}$ and its basis by $\langle \mathbf{C_1}, \ldots, \mathbf{C_k} \rangle$, with $\mathbf{C_i} \in \mathbb{F}_q^{m\times n}$ linearly independent. Two matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$ are said to be *equivalent* if there exists an isometry μ with $\mu(\mathcal{C}) = \mathcal{D}$.

An isometry from $\mathcal{C}$ to $\mathcal{D}$ is always of the form $\mathbf{M} \mapsto \mathbf{AMB}$, $\mathbf{M} \mapsto \mathbf{M}^\top$ or a composition of these two, where $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$ [32, 33]. We restrict our attention to the isometries of the first form and we will say that two matrix codes are equivalent if there exists a map $\mathbf{C} \mapsto \mathbf{ACB}$ from $\mathcal{C}$ to $\mathcal{D}$ where $\mathbf{A} \in \mathrm{GL}_m(q)$ and $\mathbf{B} \in \mathrm{GL}_n(q)$. We will denote this map as a pair $(\mathbf{A}, \mathbf{B})$. When $n = m$, If there exists a map $(\mathbf{A}, \mathbf{A}^\top) : \mathbf{C} \mapsto \mathbf{ACA}^\top$ from $\mathcal{C}$ to $\mathcal{D}$, where $\mathbf{A} \in \mathrm{GL}_m(q)$, we will say that the codes $\mathcal{C}$ and $\mathcal{D}$ are *congruent*. This is a direct generalization of the notion of congruent matrices. An *automorphism* of a code is a map $(\mathbf{A}, \mathbf{B}) : \mathcal{C} \to \mathcal{C}$, i.e. for each $\mathbf{C} \in \mathcal{C}$, we get $\mathbf{ACB} \in \mathcal{C}$. The *automorphism group* of $\mathcal{C}$ contains all the automorphisms of $\mathcal{C}$. If the automorphism group contains only the maps $(\lambda\mathbf{I}, \nu\mathbf{I})$ for constants $\lambda, \nu \in \mathbb{F}_q$, we say the automorphism group is trivial.

The main focus of this article will be the *Matrix Code Equivalence*(MCE) problem which is formally defined as follows:

Problem 3 $\mathsf{MCE}(n, m, k, \mathcal{C}, \mathcal{D})$:
Input: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$
Question: Find – if any – a map $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{ACB} \in \mathcal{D}$.

This is the computational version of MCE which, similarly to its counterpart in the Hamming metric [11, 12, 34], seems to be more interesting for cryptographic applications than its decisional variant. We will thus be interested in evaluating the practical hardness only of MCE, and present algorithms only for MCE and not its decisional variant. It is also interesting to consider the following variant of MCE:

Problem 4 $\mathsf{MCEbase}(n, m, k, \mathcal{C}, \mathcal{D})$:
Input: The bases $(\mathbf{C}^{(1)}, \dots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(k)})$ of two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$
Question: Find – if any – a map $(\mathbf{A}, \mathbf{B})$, where $\mathbf{A} \in \mathrm{GL}_m(q), \mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C}^{(i)}$, it holds that $\mathbf{AC^{(i)}B} = \mathbf{D}^{(i)}$.

Intuitively, MCEbase seems easier than MCE, and as a matter of fact, we will show later that it can be solved in polynomial time. Another variant of the MCE problem is the *Matrix Codes Right Equivalence* problem (MCRE) (left equivalence could be defined similarly):

Problem 5 $\mathsf{MCRE}(n, m, k, \mathcal{C}, \mathcal{D})$:
Input: Two k-dimensional matrix codes $\mathcal{C}, \mathcal{D} \subset \mathcal{M}_{m,n}(q)$
Question: Find – if any – $\mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mathbf{CB} \in \mathcal{D}$.

It has been shown in [25] that MCE is at least as hard as code equivalence in the Hamming metric, *Hamming Code Equivalence* (HCE), also known as Linear or Monomial Equivalence. Interestingly, the same paper shows that MCRE is actually easy and can always be solved in probabilistic-polynomial time.

For *vector rank codes* $\mathcal{C} \subset \mathbb{F}_{q^m}^n$, isometries are similar to the case of matrix codes. We get the *Vector Rank Code Equivalence* (VRCE) problem.

Problem 6 $\mathsf{VRCE}(n, m, k, \mathcal{C}, \mathcal{D})$:
Input: Two k-dimensional vector rank codes $\mathcal{C}, \mathcal{D} \subset \mathbb{F}_{q^m}^n$
Question: Find – if any – a pair $(\alpha, \mathbf{B})$ where $\alpha \in \mathbb{F}_{q^m}^*, \mathbf{B} \in \mathrm{GL}_n(q)$ such that for all $\mathbf{c} \in \mathcal{C}$, it holds that $\alpha \mathbf{cB} \in \mathcal{D}$.

Given a vector rank code $\mathcal{C} \subset \mathbb{F}_{q^m}^n$ and a basis Γ for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$, each vector $\mathbf{c} \in \mathcal{C}$ can be expanded to a matrix $\Gamma(\mathbf{c}) \in \mathcal{M}_{m,n}(q)$, giving rise to a matrix code $\Gamma(\mathcal{C})$. For any two bases Γ and Γ', an equivalence between two vector rank codes $\mathcal{C}$ and $\mathcal{D}$ implies an equivalence between the matrix codes $\Gamma(\mathcal{C})$ and $\Gamma'(\mathcal{D})$ [31], so VRCE is trivially a subproblem of MCE. However, using the $\mathbb{F}_{q^m}$-linearity of vector rank codes, VRCE reduces *non-trivially* to MCRE [25].

2.2 Systems of quadratic polynomials.

Let $\mathcal{P} = (p_1, p_2, \ldots, p_k) : \mathbb{F}_q^N \to \mathbb{F}_q^k$ be a vectorial function of k quadratic polynomials in N variables $x_1, \ldots, x_N$, where

$$p_s(x_1, \ldots, x_N) = \sum_{1 \leqslant i \leqslant j \leqslant N} \gamma_{ij}^{(s)} x_i x_j + \sum_{i=1}^{N} \beta_i^{(s)} x_i + \alpha^{(s)},$$

with $\gamma_{ij}^{(s)}, \beta_i^{(s)}, \alpha^{(s)} \in \mathbb{F}_q$ for $1 \leqslant s \leqslant k$.

It is common to represent the quadratic homogeneous part of the components of $\mathcal{P}$ using symmetric matrices, but unfortunately, a natural correspondence only exists for finite fields of odd characteristic. For the case of even characteristic, we will adopt a technical representation that is a common workaround in the literature of multivariate cryptography and will still be good for our purposes.

Let $p(x_1, \ldots, x_N) = \sum\limits_{1 \leqslant i \leqslant j \leqslant N} \gamma_{ij} x_i x_j$ be a quadratic form over $\mathbb{F}_q$. Then, for fields of odd characteristic, we can associate to p a symmetric matrix $\mathbf{P} = \overline{\mathbf{P}} + \overline{\mathbf{P}}^\top$, where $\overline{\mathbf{P}}$ is an upper triangular matrix with coefficients $\overline{\mathbf{P}}_{ij} = \gamma_{ij}/2$ for $i \leqslant j$. Clearly, there is a one-to-one correspondence between quadratic forms and symmetric matrices, since for $\mathbf{x} = (x_1, \ldots, x_N)$ it holds that

$$p(x_1, \ldots, x_N) = \mathbf{x}\mathbf{P}\mathbf{x}^\top. \tag{1}$$

Now, all operations on quadratic forms naturally transform into operations on matrices since the one-to-one correspondence between quadratic forms and symmetric matrices is actually an isomorphism. Note that, in matrix form, change of variables (basis) works as:

$$p(\mathbf{x}\mathbf{S}) = \mathbf{x}\mathbf{S}\mathbf{P}\mathbf{S}^\top\mathbf{x}^\top. \tag{2}$$

In what follows, we will interchangeably work with both the quadratic form p and its matrix representation $\mathbf{P}$.

Over fields $\mathbb{F}_q$ of even characteristic, the relation (1) does not hold, since for a symmetric matrix $\mathbf{P}$ we have $(\mathbf{P}_{ij} + \mathbf{P}_{ji})x_i x_j = 2\mathbf{P}_{ij} x_i x_j = 0$. The nice correspondence between quadratic forms and symmetric matrices is broken, but we would still like to be able to use some sort of matrix representation for quadratic forms. Thus, in even characteristic we associate to p a symmetric matrix $\mathbf{P} = \overline{\mathbf{P}} + \overline{\mathbf{P}}^\top$, where $\overline{\mathbf{P}}$ is an upper triangular matrix with coefficients $\overline{\mathbf{P}}_{ij} = \gamma_{ij}$ for $i \leqslant j$.

This representation can also be used in odd characteristic when it comes to linear operations and changes of basis, as the correspondence $p \mapsto \mathbf{P}$ is a homomorphism. However, it is not a bijection, since all the quadratic forms in the set $\{\sum\limits_{1 \leqslant i < j \leqslant N} \gamma_{ij} x_i x_j + \sum\limits_{1 \leqslant i \leqslant N} \gamma_{ii} x_i^2 \mid \gamma_{ii} \in \mathbb{F}_q\}$ map to the same symmetric matrix (note that it has zeros on the diagonal). In practical, cryptographic applications, this typically does not pose a problem, and can be overcome. The same holds for our purpose of solving equivalence problems for systems of quadratic polynomials.

2.2.1 Differential of quadratic functions.

Given a non-zero $\mathbf{a} \in \mathbb{F}_q^N$, a well studied object in multivariate cryptology is the *differential* of $\mathcal{P}$ at $\mathbf{a}$ (see [35, 36]):

$$D_{\mathbf{a}}\mathcal{P} : \mathbb{F}_q^N \to \mathbb{F}_q^k, \quad \mathbf{x} \mapsto \mathcal{P}(\mathbf{x} + \mathbf{a}) - \mathcal{P}(\mathbf{x}) - \mathcal{P}(\mathbf{a}).$$

Note that the differential of a quadratic function is closely related to the bilinear form $\beta(\mathbf{x},\mathbf{y}) = q(\mathbf{x}+\mathbf{y}) - q(\mathbf{x}) - q(\mathbf{y})$ associated to a quadratic form q. In this work we are especially interested in the kernel of $D_{\mathbf{a}}\mathcal{P}$, as $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$ implies $\mathcal{P}(\mathbf{x}+\mathbf{a}) = \mathcal{P}(\mathbf{x}) + \mathcal{P}(\mathbf{a})$. So $\mathcal{P}$ behaves linear on the kernel of $D_{\mathbf{a}}\mathcal{P}$.

2.3 Isomorphism of polynomials.

The Isomorphism of Polynomials (IP) problem (or Polynomial Equivalence (PE) [37]) was first defined by Patarin in [1] for the purpose of designing a "graph isomorphism"-like identification scheme and a digital signature using the Fiat-Shamir transform [7]. It is defined as follows.

Problem 7 $\mathsf{IP}(N, k, \mathcal{F}, \mathcal{P})$:
Input: Two k-tuples of multivariate polynomials $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[x_1, \ldots, x_N]^k$.
Question: Find – if any – $(\mathbf{S}, \mathbf{s}) \in \mathrm{AGL}_N(q)$, $(\mathbf{T}, \mathbf{t}) \in \mathrm{AGL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = \mathcal{F}(\mathbf{x}\mathbf{S} + \mathbf{s})\mathbf{T} + \mathbf{t}. \tag{3}$$

The variant of the problem where $(\mathbf{T}, \mathbf{t})$ is trivial is known as the Isomorphism of Polynomials with one secret ($\mathsf{IP1}\mathcal{S}$), whereas if $\mathcal{P}$ and $\mathcal{F}$ are quadratic and both $\mathbf{s}$ and $\mathbf{t}$ are the null vector, the problem is known as Quadratic Maps Linear Equivalence (QMLE) problem.

The decisional version of IP is not $\mathcal{NP}$-complete [30], but it is known that even $\mathsf{IP1}\mathcal{S}$ is at least as difficult as the Graph Isomorphism problem [30]. The IP problem has been investigated by several authors, initially for the security of the C^* scheme [30]. In [38] it was shown that the $\mathsf{IP1}\mathcal{S}$ is polynomially solvable for most of the instances with $k \geq N$, and Bouillaguet et al. [39] gave an algorithm with running time of $\mathcal{O}(N^6)$ for random instances of the $\mathsf{IP1}\mathcal{S}$ problem, thus fully breaking Patarin's identification scheme [1]. The authors of [30] gave an algorithm for solving the general IP, called To-and-Fro, that runs in time $\mathcal{O}(q^{2N})$ for $q > 2$ and $\mathcal{O}(q^{3N})$ for $q = 2$. It was noted in [28] that the algorithm is only suited for bijective mappings $\mathcal{F}$ and $\mathcal{P}$. Getting rid of the bijectivity constraint has been explored in [29] with the conclusion that the proposed workarounds either have a non-negligible probability of failure or it is unclear how greatly they affect the complexity of the algorithm.

Regarding QMLE, the linear variant of IP, an empirical argument was given in [37] that random inhomogeneous instances are solvable in $\mathcal{O}(N^9)$ time, but a rigorous proof for this case still remains an open problem. Under this assumption, the same paper provides an algorithm of complexity $\mathcal{O}(N^9 q^N)$ for the homogeneous case which is considered the hardest, that was subsequently improved to $\mathcal{O}(N^9 q^{2N/3})$ in [28]. Both works reduce a homogenous instance to an inhomogenous instance and assume the obtained inhomogeneous instance behaves as a random instance. This, however, is a wrong assumption which questions the claimed complexity of the algorithm.

In this work, we will be interested in the homogeneous variant of QMLE, that we denote hQMLE, as the hardest and most interesting instance of QMLE. Formally, the hQMLE problem is defined as follows.

Problem 8 $\mathsf{hQMLE}(N, k, \mathcal{F}, \mathcal{P})$:
Input: Two k-tuples of homogeneous multivariate polynomials of degree 2

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

Question: Find – if any – a map $(\mathbf{S}, \mathbf{T})$ where $\mathbf{S} \in \mathrm{GL}_N(q), \mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}) = (\mathcal{F}(\mathbf{xS}))\mathbf{T}. \tag{4}$$

Interestingly, the case of $k = 1$, which we will call Quadratic Form Equivalence (QFE) has been completely solved for more than 80 years already in the works of Witt [40] and Arf [41]. It is known that every quadratic form is equivalent to a unique canonical diagonal (for odd characteristic) or block diagonal (for even characteristic) form which can be obtained in time $\mathcal{O}(N^3)$. Thus, QFE can also be solved in time $\mathcal{O}(N^3)$ by first calculating the transformations to the canonical forms of the two quadratic forms. If the canonical forms are the same, by composition, one can find the equivalence. If the canonical forms are not the same, the two quadratic forms are not equivalent.

In this work we also consider a variant of QMLE where $\mathcal{F}$ and $\mathcal{P}$ are bilinear forms. We call this problem Bilinear Maps Linear Equivalence (BMLE). In this variant, $\mathcal{F}$ and $\mathcal{P}$ are k-tuples of homogeneous polynomials of degree 2 in two sets of variables $[x_1, \dots, x_n]$ and $[y_1, \dots, y_m]$, where each monomial is of the form $x_i y_j$. Formally, the BMLE problem is defined as follows.

Problem 9 $\mathsf{BMLE}(n, m, k, \mathcal{F}, \mathcal{P})$:
Input: Two k-tuples of bilinear forms

$$\mathcal{F} = (f_1, f_2, \dots, f_k),\ \mathcal{P} = (p_1, p_2, \dots, p_k) \in \mathbb{F}_q[x_1, \dots, x_n, y_1, \dots, y_m]^k$$

Question: Find – if any – a triplet $(\mathbf{S}_1, \mathbf{S}_2, \mathbf{T})$ where $\mathbf{S}_1 \in \mathrm{GL}_n(q), \mathbf{S}_2 \in \mathrm{GL}_m(q)$, $\mathbf{T} \in \mathrm{GL}_k(q)$ such that

$$\mathcal{P}(\mathbf{x}, \mathbf{y}) = (\mathcal{F}(\mathbf{xS}_1, \mathbf{yS}_2))\mathbf{T}. \tag{5}$$

The inhomogenous versions of QMLE and BMLE will be referred to as inhQMLE and inhBMLE respectively. We write inh(Q/B)MLE when it does not matter if we are referring to the quadratic or the bilinear version.

3 How hard is MCE?

In this section we investigate the relation of the MCE problem to other known problems that we notably split in two groups – equivalence problems for systems of multivariate quadratic polynomials and equivalence problems for codes.

3.1 Relations to equivalence problems for qaudratic polynomials

We start with establishing a straightforward link between MCE and polynomial equivalence problems by proving that the MCE and BMLE problems are equivalent.

Theorem 10 *The* MCE *problem is at least as hard as the* BMLE *problem.*

Proof In order to prove our claim, we need to show that an algorithm $\mathcal{A}$ solving any instance of the MCE problem can be transformed in polynomial time to an algorithm $\mathcal{B}$ solving any instance of the BMLE problem.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{BMLE}}(n,m,k,\mathcal{F},\mathcal{P})$ of BMLE, where $\mathcal{F} = (f_1, f_2, \dots, f_k)$, $\mathcal{P} = (p_1, p_2, \dots, p_k) \in \mathbb{F}_q[\mathbf{x},\mathbf{y}]^k$ are k-tuples of bilinear forms. Without loss of generality, we assume $f_1, f_2, \dots, f_k$ (respectively $p_1, p_2, \dots, p_k$) are linearly independent. $\mathcal{B}$ can efficiently construct an instance of the MCE problem as follows.

$\mathcal{B}$ represents the components f_s and p_s, $s \in \{1, \dots, k\}$ of the mappings $\mathcal{F}$ and $\mathcal{P}$ as $n \times m$ matrices $\mathbf{F}^{(s)}$ and $\mathbf{P}^{(s)}$, where $\mathbf{F}^{(s)}_{i,j}$ equals the coefficient of $x_i x_j$ in f_s and $\mathbf{P}^{(s)}_{i,j}$ equals the coefficient of $x_i x_j$ in p_s. Taking $(\mathbf{F}^{(1)}, \dots, \mathbf{F}^{(k)})$ to be a basis of a matrix code $\mathcal{C}$ and $(\mathbf{P}^{(1)}, \dots, \mathbf{P}^{(k)})$ a basis of a matrix code $\mathcal{D}$, $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{MCE}}(n,m,k,\mathcal{C},\mathcal{D})$ of MCE.

$\mathcal{B}$ gives the instance $\mathcal{I}_{\mathsf{MCE}}(n,m,k,\mathcal{C},\mathcal{D})$ as an input to $\mathcal{A}$. After time t and advantage ϵ, $\mathcal{A}$ outputs a solution $(\mathbf{A},\mathbf{B})$ to the MCE instance. From here, $\mathcal{B}$ constructs the matrices $\mathbf{R}^{(s)} = \mathbf{A}\mathbf{F}^{(s)}\mathbf{B} \in \mathcal{D}$ and solves the following system of equations in the variables $t_{i,j}$:

$$\sum_{j=1}^{k} t_{j,i} \cdot \mathbf{R}^{(j)} = \mathbf{P}^{(i)}, \forall i \in \{1, \dots, k\} \tag{6}$$

The system has always a solution, since $(\mathbf{R}^{(1)}, \dots, \mathbf{R}^{(k)})$ is a basis of the code $\mathcal{D}$.

$\mathcal{B}$ sets $\mathbf{T} = \big(t_{i,j}\big)$, and outputs $(\mathbf{A}, \mathbf{B}^\top, \mathbf{T})$ as the solution to $\mathcal{I}_{\mathsf{BMLE}}(n,m,k,\mathcal{F},\mathcal{P})$. $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$ and runs in time $t + \mathcal{O}(k^6)$.

□

Theorem 11 BMLE *is at least as hard as* MCE*.*

Proof We proceed similarly as in the other direction – Given an algorithm $\mathcal{A}$ solving any instance of BMLE, we can construct in polynomial time an algorithm $\mathcal{B}$ that can solve any instance of MCE.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{MCE}}(n,m,k,\mathcal{C},\mathcal{D})$ of MCE. $\mathcal{B}$ takes arbitrary bases $(\mathbf{C}^{(1)}, \dots, \mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(k)})$ of the codes $\mathcal{C}$ and $\mathcal{D}$ respectively. For each of the matrices $\mathbf{C}^{(s)}$ $\mathcal{B}$ constructs the bilinear forms $c_s(\mathbf{x},\mathbf{y}) = \sum\limits_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n} \mathbf{C}^{(s)}_{ij} x_i y_j$ and for the matrices $\mathbf{D}^{(s)}$ the bilinear forms $d_s(\mathbf{x},\mathbf{y}) = \sum\limits_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n} \mathbf{D}^{(s)}_{ij} x_i y_j, \forall s, 1 \leqslant s \leqslant k$. Taking $\mathcal{F} = (c_1, c_2, \dots, c_k)$ and $\mathcal{P} = (d_1, d_2, \dots, d_k)$ we obtain an instance $\mathcal{I}_{\mathsf{BMLE}}(n,m,k,\mathcal{F},\mathcal{P})$ of BMLE.

$\mathcal{B}$ gives the instance $\mathcal{I}_{\mathsf{BMLE}}(n,m,k,\mathcal{F},\mathcal{P})$ to $\mathcal{A}$ which in time t and with probability ϵ outputs a solution $(\mathbf{S}_1, \mathbf{S}_2, \mathbf{T})$ to the BMLE instance. This immediately gives a solution $(\mathbf{S}_1, \mathbf{S}_2^\top)$ to the MCE instance. □

In order to prove the connection of MCE to the more general problem hQMLE we first need to establish some properties of matrix codes.

Lemma 12 *Let $\mathcal{C}$ and $\mathcal{D}$ be matrix codes generated by the bases $= (\mathbf{C}_1, \ldots, \mathbf{C}_k)$ and $(\mathbf{D}_1, \ldots, \mathbf{D}_k)$ of (skew-)symmetric matrices, and assume that $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups. Then $\mathcal{C}$ is equivalent to $\mathcal{D}$ if and only if $\mathcal{C}$ is congruent to $\mathcal{D}$.*

Proof Clearly, by definition if $\mathcal{C}$ is congruent to $\mathcal{D}$, then $\mathcal{C}$ is equivalent to $\mathcal{D}$.

For the opposite direction, let $\mathcal{C}$ be equivalent to $\mathcal{D}$. Then there exist nonsingular matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{T}$ such that

$$\sum_{i=1}^{k} t_{j,i}\mathbf{D}_i = \mathbf{A}\mathbf{C}_j\mathbf{B}$$

Since $\mathbf{C}_i$ and $\mathbf{D}_i$ are (skew-)symmetric the last rewrites as

$$\sum_{i=1}^{k} t_{j,i}\mathbf{D}_i = \mathbf{B}^\top\mathbf{C}_j\mathbf{A}^\top$$

Combining the two, and since $\mathbf{A}$ and $\mathbf{B}$ are non-singular, we obtain

$$\mathbf{C}_j = \mathbf{A}^{-1}\mathbf{B}^\top\mathbf{C}_j\mathbf{A}^\top\mathbf{B}^{-1}$$

The automorphism group being trivial implies $\mathbf{A} = \lambda\mathbf{B}^\top$ for some $\lambda \in \mathbb{F}_q$ which in turn implies that $\mathcal{C}$ is congruent to $\mathcal{D}$. □

Remark 13 The result of Lemma 12 has already been known for algebraically closed fields of non-even characteristic [27, 42]. Since finite fields are not algebraically closed, this result is not useful in our context. On the other hand, requiring a trivial automorphism group for the codes is not a huge restriction, and we typically expect the automorphism group to be trivial for randomly chosen matrix codes. Specifically for cryptographic purposes with regards to MCE, one wants the orbit of $\mathcal{C}$ to be maximal under the action of suitable isometries, which happens when the automorphism group of $\mathcal{C}$ is trivial. Similar requirements for trivial or small automorphism groups occur in the Hamming metric, where it is known that without this requirement there might exist weak keys [43, 44].

Now, we are ready to show the following.

Theorem 14 *The* MCE *problem is at least as hard as the* hQMLE *problem for systems of polynomials whose symmetric matrix representations have trivial automorphism groups.*

Proof We perform the reduction in a similar manner as previously.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{hQMLE}}(N, k, \mathcal{F}, \mathcal{P})$ of hQMLE, where $\mathcal{F} = (f_1, f_2, \ldots, f_k)$, $\mathcal{P} = (p_1, p_2, \ldots, p_k) \in \mathbb{F}_q[\mathbf{x}, \mathbf{y}]^k$ are k-tuples of linearly independent quadratic forms. $\mathcal{B}$ can efficiently construct an instance of the MCE problem as follows.

$\mathcal{B}$ forms the $N \times N$ symmetric matrices $\mathbf{F}^{(s)}$ and $\mathbf{P}^{(s)}$ associated to the components f_s and p_s, $s \in \{1, \ldots, k\}$ of the mappings $\mathcal{F}$ and $\mathcal{P}$. Taking $(\mathbf{P}^{(1)}, \ldots, \mathbf{P}^{(k)})$

to be a basis of a matrix code $\mathcal{D}$ and $(\mathbf{F}^{(1)},\ldots,\mathbf{F}^{(k)})$ a basis of a matrix code $\mathcal{C}$, $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{MCE}}(N,N,k,\mathcal{C},\mathcal{D})$ of MCE. Per assumption, the matrix codes $\mathcal{C}$ and $\mathcal{D}$ have trivial automorphism groups.

$\mathcal{B}$ gives the instance $\mathcal{I}_{\mathsf{MCE}}(N,N,k,\mathcal{C},\mathcal{D})$ as an input to $\mathcal{A}$. After time t and advantage ϵ, $\mathcal{A}$ outputs a solution $(\mathbf{A},\mathbf{B})$ to the MCE instance. From Lemma 12, since the matrices are symmetric, $\mathbf{A}=\mathbf{B}^\top$. Now, $\mathcal{B}$ applies the change of variables $\mathbf{xA}$ to $\mathcal{F}$ and obtains $\mathcal{R}(\mathbf{x})=\mathcal{F}(\mathbf{xA})$. It then solves the system

$$\sum_{j=1}^{k} t_{j,s}\cdot r_j = p_s, \forall s\in\{1,\ldots,k\} \tag{7}$$

The system has a solution if $\mathcal{I}_{\mathsf{hQMLE}}(N,k,\mathcal{F},\mathcal{P})$ is a positive instance. This is always the case in odd characteristic, because there is a one-to-one correspondence between polynomials and their symmetric matrix representation. Over characteristic 2, it may happen that the $\mathcal{I}_{\mathsf{hQMLE}}(N,k,\mathcal{F},\mathcal{P})$ is not a positive instance while its symmetric matrix representation $\mathcal{I}_{\mathsf{MCE}}(N,N,k,\mathcal{C},\mathcal{D})$ is. In this case, the system (7) does not have a solution.

If the system has a solution, $\mathcal{B}$ sets $\mathbf{T}=\left(t_{i,j}\right)$, and outputs $(\mathbf{A},\mathbf{T})$ as the solution to $\mathcal{I}_{\mathsf{hQMLE}}(N,k,\mathcal{F},\mathcal{P})$. Oterwise, it outputs $\perp$. $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$ and runs in time $t+\mathcal{O}(k^6)$.

□

For the following theorem, we define the symmetric matrix representation of a matrix code $\mathcal{C}$ as the code $\{ \begin{bmatrix}\mathbf{0} & \mathbf{C}^\top\\ \mathbf{C} & \mathbf{0}\end{bmatrix} \mid \mathbf{C}\in\mathcal{C}\}$.

Theorem 15 *The* hQMLE *problem is at least as hard as the* MCE *problem for codes whose symmetric matrix representations have trivial automorphism groups.*

Proof We show that an algorithm $\mathcal{A}$ that solves the hQMLE problem can be transformed in polynomial time to an algorithm $\mathcal{B}$ that solves the MCE problem if we assume that codes of symmetric matrices have trivial automorphism groups.

Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{MCE}}(n,m,k,\mathcal{C},\mathcal{D})$ of MCE. $\mathcal{B}$ can efficiently construct an instance of the hQMLE problem as follows.

$\mathcal{B}$ fixes bases $(\mathbf{D}^{(1)},\ldots,\mathbf{D}^{(k)})$ of the code $\mathcal{D}$ and $(\mathbf{C}^{(1)},\ldots,\mathbf{C}^{(k)})$ of the code $\mathcal{C}$. For each of the matrices $\mathbf{C}^{(s)}$, $\mathcal{B}$ constructs the quadratic forms $c_s(\mathbf{x}) = \sum\limits_{1\leqslant i\leqslant m, m+1\leqslant j\leqslant m+n} \mathbf{C}^{(s)}_{ij} x_i x_j$ and for the matrices $\mathbf{D}^{(s)}$ the quadratic forms $d_s(\mathbf{x}) = \sum\limits_{1\leqslant i\leqslant m, m+1\leqslant j\leqslant m+n} \mathbf{D}^{(s)}_{ij} x_i x_j, \forall s, 1\leqslant s\leqslant k$, where $\mathbf{x}=(x_1,\ldots,x_{m+n})$. Taking $\mathcal{F} = (c_1,c_2,\ldots,c_k)$ and $\mathcal{P}=(d_1,d_2,\ldots,d_k)$ $\mathcal{B}$ obtains an instance $\mathcal{I}_{\mathsf{hQMLE}}(n+m,k,\mathcal{F},\mathcal{P})$ of hQMLE.

$\mathcal{B}$ gives the instance $\mathcal{I}_{\mathsf{hQMLE}}(n+m,k,\mathcal{F},\mathcal{P})$ to $\mathcal{A}$ which in time t and with probability ϵ outputs a solution $(\mathbf{S},\mathbf{T})$ to the hQMLE instance.

We argue that this solution can be transformed to a solution to the MCE instance, if it is a positive instance. The symmetric matrix representation of the codes $\mathcal{C}$ and

$\mathcal{D}$ is given by

$$\begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix}, i \in \{1, \dots, k\}. \tag{8}$$

The solution $(\mathbf{S}, \mathbf{T})$ means

$$\sum \tilde{t}_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \mathbf{S} \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix} \mathbf{S}^\top, i \in \{1, \dots, k\}. \tag{9}$$

If the given MCE instance is positive, then there exist matrices $\mathbf{A}, \mathbf{B}, \mathbf{L}$ such that $\mathbf{A}\mathbf{C}_i\mathbf{B} = \sum_j l_{i,j}\mathbf{D}_j$. This implies

$$\sum l_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{0} & (\mathbf{C}^{(i)})^\top \\ \mathbf{C}^{(i)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^\top \end{bmatrix}, i \in \{1, \dots, k\}. \tag{10}$$

The last two imply

$$\sum \lambda_{i,j} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(j)})^\top \\ \mathbf{D}^{(j)} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \mathbf{S}^{-1} \begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix} \mathbf{S}^{-\top} \begin{bmatrix} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^\top \end{bmatrix}, i \in \{1, \dots, k\}. \tag{11}$$

By assumption, the automorphism group of the $\begin{bmatrix} \mathbf{0} & (\mathbf{D}^{(i)})^\top \\ \mathbf{D}^{(i)} & \mathbf{0} \end{bmatrix}$ matrices is trivial, which means $\mathbf{S}$ necessarily equals $\begin{bmatrix} \mathbf{B}^\top & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix}$ up to scalar multiplication. For such an $\mathbf{S}$, the MCE solution can immediately be extracted. $\mathcal{B}$ then outputs the extracted solution.

If on the other hand, $\mathbf{S}$ is not of such block-diagonal form, $\mathcal{B}$ outputs $\perp$. $\mathcal{B}$ has the same advantage ϵ as $\mathcal{A}$ and runs in the same time t.

□

Remark 16 Using the above reduction between MCE and hQMLE, we can reduce the MCEbase problem to and from a special case of IP known as IP1$\mathcal{S}$. Interestingly, Perret [38] shows IP1$\mathcal{S}$ is polynomially solvable for most instances $k \geq N$, and later work [39] gives an algorithm with running time of $\mathcal{O}(N^6)$ for random instances. This implies that the MCEbase problem can be solved in polynomial time for most cryptographically interesting parameters.

3.2 Relations to equivalence problems for linear codes

In this section, we show that MCE is at the heart of code equivalence problems. Equivalence problems for different metrics, such as the Hamming metric or the sum-rank metric, reduce to MCE, making the hardness analysis of MCE the more exciting.

3.2.1 Hamming code equivalence.

Codes $\mathcal{C} \subset \mathbb{F}_q^n$ equipped with the *Hamming metric* have isometries of the form

$$\tau : (c_1, \dots, c_n) \mapsto (\alpha_1 c_{\pi^{-1}(1)}, \dots, \alpha_n c_{\pi^{-1}(n)}), \quad \alpha_i \in \mathbb{F}_q^*,\ \pi \in S_n. \tag{12}$$

From this, we define *Hamming code equivalence* (HCE) as the existence of an isometry between two Hamming codes $\mathcal{C}$ and $\mathcal{D}$.

Problem 17 $\mathsf{HCE}(k, n, \mathcal{C}, \mathcal{D})$:
Input: Two k-dimensional Hamming codes $\mathcal{C}, \mathcal{D} \subset \mathbb{F}_q^n$
Question: Find – if any – $\alpha \in \mathbb{F}_q^{*n}, \pi \in S_n$ such that $\alpha\pi(\mathbf{c}) \in \mathcal{D}$ holds for all $\mathbf{c} \in \mathcal{C}$.

The subproblem where α is trivial is called the *monomial equivalence problem.* It is easy to turn an HCE-instance into a MCE-instance [25], given the description of isometries in Equation (12). First, define $\Phi : \mathbb{F}_q^n \to \mathcal{M}_n(\mathbb{F}_q)$ by

$$\mathbf{x} = (x_1, \ldots, x_n) \mapsto \begin{pmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{pmatrix}.$$

The map Φ is an isometry from the Hamming metric to the rank metric: codewords with weigh t are mapped to matrices of rank t. From this, we quickly get the reduction: Writing π as a matrix $\mathbf{P} \in \mathrm{GL}_n(q)$, Φ translates a Hamming isometry τ to a rank-metric isometry by

$$\Phi(\tau) : \Phi(\mathbf{x}) \mapsto \mathbf{P}^{-1}\Phi(\mathbf{x})\mathbf{AP}, \quad \text{where } \mathbf{A} = \begin{pmatrix} \alpha_1 & & \\ & \ddots & \\ & & \alpha_n \end{pmatrix} \in \mathrm{GL}_n(q).$$

A second reduction from HCE to MCE is given later in [25], which concerns the search variant of the problem, and is more explicit. Both reductions however do not help with solving HCE in practice: both the permutational ($\mathbf{A}$ is trivial) and the linear variant of code equivalence in the Hamming metric have algorithms [12, 45] that perform much better for an HCE-instance τ than the algorithms we propose for solving $\Phi(\tau)$ as an MCE-instance.

3.2.2 Sum-rank code equivalence.

The *sum-rank metric* [46] is a metric that is gaining in popularity in coding theory. It is commonly given as a generalization of the vector-rank metric, but one can also define a variant that generalizes matrix-rank metric. We will reduce both vector and matrix sum-rank equivalence problems to MCE. The idea is the same as for HCE, we find the right isometry from sum-rank metric to rank metric to get the reduction.

Definition 18 Let n be partitioned as $n = n_1 + \ldots + n_\ell$. Let $\mathbf{v}^{(i)} = (v_1^{(i)}, \ldots, v_{n_i}^{(i)}) \in \mathbb{F}_{q^m}^{n_i}$. and $\mathbf{v} = (\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(\ell)}) \in \mathbb{F}_{q^m}^n$. Let Γ be a basis for $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Then the *vector sum-rank* of $\mathbf{v}$ is defined as

$$\mathrm{SumRank}(\mathbf{v}) := \sum_{i=1}^{\ell} \mathrm{Rank}\,\Gamma(\mathbf{v}^{(i)}).$$

Let m be partitioned as $m = m_1 + \ldots + m_\ell$. Let $\mathbf{V}^{(i)} \in \mathcal{M}_{m_i \times n_i}(\mathbb{F}_q)$ and $\mathbf{V} = (\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(\ell)})$. Then the *matrix sum-rank* of $\mathbf{V}$ is defined as

$$\mathrm{SumRank}(\mathbf{V}) = \sum_{i=1}^{\ell} \mathrm{Rank}\,\mathbf{V}^{(i)}.$$

The sum-rank generalizes both the Hamming metric and the rank metric: taking $\ell = n$ gives the Hamming metric, whereas $\ell = 1$ gives the rank metric. We define isometries again as maps that preserve the sum-rank. Sum-rank isometries are simple generalisations of rank isometries (see Problem 6).

Proposition 19 ([47, Thm. 3.7]) *Isometries with respect to the vector sum-rank metric are given by vector rank isometries $\mu^{(i)} : \mathbf{x}^{(i)} \mapsto \alpha^{(i)}\mathbf{x}^{(i)}\mathbf{B}^{(i)}$ per 'block' with $\alpha^{(i)} \in \mathbb{F}_{q^m}^*$ and $\mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$, and suitable permutations π of such blocks if $n_i = n_j$ for $i \neq j$, so*

$$\mu : (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}) \mapsto (\alpha^{(1)}\mathbf{x}^{\pi^{-1}(1)}\mathbf{B}^{(1)}, \ldots, \alpha^{(\ell)}\mathbf{x}^{\pi^{-1}(\ell)}\mathbf{B}^{(\ell)})$$

is a general description of a vector sum-rank isometry.

Generalizing to matrix sum-rank codes is achieved by simply replacing $\alpha^{(i)} \in \mathbb{F}_{q^m}^*$ with $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q)$ [48, Prop. 4.25]. This gives us the *Vector Sum-Rank Code Equivalence* (VSRCE) and *Matrix Sum-Rank Code Equivalence* (MSRCE) problems.

Problem 20 VSRCE$(n, m, k, \mathcal{C}, \mathcal{D})$:
Input: Two k-dimensional vector sum-rank codes $\mathcal{C}, \mathcal{D} \subset \mathbb{F}_{q^m}^n$
Question: Find – if any – $\alpha^{(i)} \in \mathbb{F}_{q^m}^*, \mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$ and a permuation π such that for all $\mathbf{c} \in \mathcal{C}$, it holds that $\mu(\mathbf{c}) \in \mathcal{D}$.

Problem 21 MSRCE$(n, m, k, \mathcal{C}, \mathcal{D})$:
Input: Two k-dimensional matrix sum-rank codes $\mathcal{C}, \mathcal{D} \subset (\mathcal{M}_{m_i \times n_i}(\mathbb{F}_q))_i$
Question: Find – if any – $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q), \mathbf{B}^{(i)} \in \mathrm{GL}_{n_i}(q)$ and a permuation π such that for all $\mathbf{C} \in \mathcal{C}$, it holds that $\mu(\mathbf{C}) \in \mathcal{D}$.

This gives a reduction to matrix isometries, by the same trick as for HCE. First, we define a 'nice' map $\Psi : \mathbb{F}_q^n \to \mathcal{M}_{\ell \cdot m \times n}(\mathbb{F}_q)$ by

$$\mathbf{x} = (\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)}) \mapsto \begin{pmatrix} \mathrm{Mat}(\mathbf{x}^{(1)}) & & \\ & \ddots & \\ & & \mathrm{Mat}(\mathbf{x}^{(\ell)}) \end{pmatrix}.$$

It is clear that Ψ is an isometry from the vector sum-rank metric to the rank metric, as it preserves the weight. We get the following reduction.

Theorem 22 *Assuming trivial automorphism groups,* MCE *is at least as hard as* VSRCE*.*

Proof Suppose $\mathcal{B}$ is given an instance $\mathcal{I}_{\mathsf{VSRCE}}(n, m, k, \mathcal{C}, \mathcal{D})$ of VSRCE, where $\mathcal{C}$ and $\mathcal{D}$ are k-dimensional vector sum-rank codes. $\mathcal{B}$ can efficiently construct an instance of the MCE problem as follows. By writing the permutation π of the 'blocks' by a matrix representation $\mathbf{P}$, $\mathcal{B}$ can translate a vector sum-rank isometry μ into a matrix code isometry $\Psi(\mu)$ by

$$\Psi(\mu) : \Psi(\mathbf{x}) \mapsto \mathbf{P}^{-1}\mathbf{A}\Psi(\mathbf{x})\mathbf{B}\mathbf{P} \quad \text{where } \mathbf{A} = \begin{pmatrix} \alpha^{(1)} & & \\ & \ddots & \\ & & \alpha^{(\ell)} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{B}^{(1)} & & \\ & \ddots & \\ & & \mathbf{B}^{(\ell)} \end{pmatrix}$$

with $\mathbf{A} \in \mathrm{GL}_\ell(q^m)$, $\mathbf{B} \in \mathrm{GL}_n(q)$. Hence, $\Psi(\mu)$ is an $(n, m, k, \mathcal{C}, \mathcal{D})$-instance of MCE which $\mathcal{B}$ gives to $\mathcal{A}$. In time t and with probability ε, $\mathcal{A}$ outputs a solution $(\mathbf{A}', \mathbf{B}')$ to this MCE instance. As the automorphism group is trivial, $\mathcal{B}$ computes $\lambda\mathbf{A}' = \mathbf{P}^{-1}\mathbf{A}$ and $\lambda\mathbf{B}' = \mathbf{B}\mathbf{P}$ for $\lambda \in \mathbb{F}_q$, and therefore solves $\mathcal{I}_{\mathsf{VSRCE}}$ with the same advantage ε as $\mathcal{A}$ and in the same time t. □

From vector sum-rank code equivalence to matrix sum-rank code equivalence is only a small step. Given a partition $m = m_1 + \ldots + m_\ell$, the map we need is only slightly different from Ψ, namely $\tilde{\Psi} : (\mathcal{M}_{m_i \times n_i}(\mathbb{F}_q))_i \to \mathcal{M}_{m \times n}(\mathbb{F}_q)$ by

$$\mathbf{X} = (\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(\ell)}) \mapsto \begin{pmatrix} \mathbf{X}^{(1)} & & \\ & \ddots & \\ & & \mathbf{X}^{(\ell)} \end{pmatrix}.$$

Theorem 23 *Assuming trivial automorphism groups,* MCE *is at least as hard as* MSRCE*.*

Proof This is a simple generalization of Theorem 22: Replace $\alpha^{(i)}$ by $\mathbf{A}^{(i)} \in \mathrm{GL}_{m_i}(q)$ so that $\mathbf{A} \in \mathrm{GL}_m(q)$. Then again, for a matrix sum-rank μ we get $\tilde{\Psi}(\mu)$ by $\Psi(\mathbf{x}) \mapsto \mathbf{P}^{-1}\mathbf{A}\Psi(\mathbf{x})\mathbf{B}\mathbf{P}$. □

The link between such instances $\Psi(\mu)$ and $\tilde{\Psi}(\mu)$ is given by a representation $\rho : \mathbb{F}_{q^m}^* \to \mathrm{GL}_m(q)$. We map a vector sum-rank instance to a matrix sum-rank instance by $\mathbf{A}^{(i)} = \rho(\alpha^{(i)})$, so that $\mathbf{A} \in \mathrm{GL}_{\ell \cdot m}(q)$.

To show the equivalences between the rank and sum-rank instances, we need to show that an MCE-instances is also a MSRCE-instance. But this is trivial: the sum-rank metric generalizes the rank metric, thus an MCE-instance is an MSRCE-instance with $\ell = 1$. So we get the following theorem for free.

Theorem 24 MSRCE *is at least as hard as* MCE*.*

4 Solving Matrix Code Equivalence

In this section, we analyze the complexity of solving an instance of MCE. As a baseline we have a straightforward algorithm that uses a result from [25] that MCRE can be solved in polynomial time. By enumerating either $\mathbf{A}$ or $\mathbf{B}$, we obtain an instance of MCRE. This means the dominating complexity is the enumeration resulting in an overall complexity of $\mathcal{O}^*(q^{m^2})$ for MCE.

The approach we oultine in the section makes use of the reduction of MCE to hQMLE (see Theorem 15). This means that we use techniques already applied for solving hQMLE, but generalize and improve them by making use of the specific structure that MCE instances show when viewed as hQMLE instances.

4.1 Solving MCE as QMLE

At Eurocrypt 2013, Bouillaguet et al. [28] proposed an algorithm for solving hQMLE using techniques from graph theory. Their main idea was to reduce the homogeneous case to the inhomogeneous case, which they assume is efficiently solvable (e.g. using the heuristic algebraic approach of [37]). Starting from an instance of hQMLE, they build two exponentially-large graphs that correspond to the given maps $\mathcal{F}$ and $\mathcal{P}$ such that, finding an isomorphism between the two graphs is equivalent to finding an isomorphism between the two quadratic maps. Since the graphs are exponentially large, a technique is provided to *walk* through the graphs without constructing them. Walking through the graphs consists of finding adjacent vertices and computing the degree of a vertex, both in polynomial time. The algorithm consists in finding pairs of vertices from the first and the second graph that have the same degree and making queries to an inhomogenous QMLE solver. If the solver finds an isomorphism by which two vertices are related, then the isomorphism between the two graphs, and thus the isomorphism between the two quadratic maps, is found.

4.2 First algorithm for solving MCE

The algorithm for solving hQMLE from [28] considers a graph arising from the differential of a given polynomial map – a vertex $\mathbf{a}$ is connected to all the vertices that vanish at the differential at $\mathbf{a}$. It is, however, not entirely clear how the property we choose to construct such graphs impacts the complexity of the algorithm. We revisit the algorithm, and show how it can be generalized, i.e. abstracted from the property used in [28], under certain conditions. In this section we present this generalization – a birthday-based algorithm for finding an isomorphism between two objects when a specific solver exists. In this form, it can be applied to a broader type of equivalence problems, using more general invariants, here implemented as a predicate $\mathbb{P}$.

Let S_1 and S_2 be subsets of a universe U of equal size N. Algorithm 1 finds an equivalence function $\phi : S_1 \to S_2$. We assume there exists a predicate $\mathbb{P} : U \to \{\top, \bot\}$ that can be computed in polynomial time, and we denote the cost $C_{\mathbb{P}}$. We assume $\mathbb{P}$ is invariant under the equivalence ϕ, i.e. $\mathbb{P}(x) = \top \leftrightarrow \mathbb{P}(\phi(x)) = \top$. Let $U_\top = \{x \in U \mid \mathbb{P}(x) = \top\}$, and $d = |U_\top|/|U|$. We will call d the *density* of the predicate $\mathbb{P}$ and we assume the density on S_1 and S_2 is approximately equal to d. We further assume the existence of an algorithm FindFunction, that given $x \in S_1, y \in S_2$ returns ϕ if $y = \phi(x)$ and $\bot$ otherwise. We denote the cost of a query to FindFunction by C_{FF}.

Lemma 25 *Algorithm 1 performs on average $\mathcal{O}(\sqrt{N/d})$ operations in* SampleSet*, queries* FindFunction *at most $d \cdot N$ times, and succeeds with probability $1 - 1/e$.*

The optimal value for d, up to a polynomial factor, is $d = N^{-1/3} \cdot C_{\mathrm{FF}}^{-2/3}$, for which the total time complexity of the algorithm is $\mathcal{O}(N^{\frac{2}{3}} \cdot C_{\mathrm{FF}}^{\frac{1}{3}})$ and the memory complexity is $\mathcal{O}(N^{\frac{1}{3}} C_{\mathrm{FF}}^{-\frac{1}{3}})$. If FindFunction *runs in polynomial time, this reduces to time complexity of $\mathcal{O}^*(N^{\frac{2}{3}})$ and memory complexity of $\mathcal{O}(N^{\frac{1}{3}})$.*

Proof First note that the expected number of elements in S_1 and S_2 such that $\mathbb{P}(x)$ holds is equal to dN. Taking the lists of size $\ell = \sqrt{d \cdot N}$, by the birthday paradox, we get a probability of $1 - \frac{1}{e}$ that FindFunction returns a solution. Here, the number of queries to FindFunction is dN. On the other hand, the number of samples needed

Algorithm 1 General Birthday-based Equivalence Finder

```
function SampleSet($S, \mathbb{P}$)
    $L \leftarrow \emptyset$
    repeat
        $a \xleftarrow{\$} S$
        if $\mathbb{P}(a)$ then $L \leftarrow L \cup \{a\}$
        end if
    until $\mathsf{L} = \ell$
    return $L$
end function

function CollisionFind($S_1, S_2$)
    $L_1 \leftarrow$ SampleSet($S_1, \mathbb{P}$)
    $L_2 \leftarrow$ SampleSet($S_2, \mathbb{P}$)
    for all $(a, b) \in L_1 \times L_2$ do
        $\phi \leftarrow$FindFunction($a, b$)
        if $\phi \neq \perp$ then
            return solution $\phi$
        end if
    end for
    return $\perp$
end function
```

to build the list L_1 (resp. L_2) of elements $a \in S_1$ (resp. $b \in S_2$) such that $\mathbb{P}(a)$ (resp. $\mathbb{P}(b)$) holds is ℓ/d, which gives a complexity of $\mathcal{O}(\sqrt{N/d})$ to build these lists L_i.

The total running time is optimal when these two quantities $\sqrt{N/d}$ and $d \cdot N \cdot C_{\mathrm{FF}}$ are equal, which holds when $d = N^{-1/3} \cdot C_{\mathrm{FF}}^{-2/3}$. Such a density gives complexity of $\mathcal{O}(N^{\frac{2}{3}} \cdot C_{\mathrm{FF}}^{\frac{1}{3}})$ for SampleSet and at most $N^{\frac{2}{3}}$ queries to FindFunction. If C_{FF} is polynomial, this gives a total time complexity of $\mathcal{O}^*(N^{\frac{2}{3}})$. The memory requirements of the algorithm correspond to the size of the lists L_i. This results in a memory complexity of $\mathcal{O}(N^{\frac{1}{3}} C_{\mathrm{FF}}^{-\frac{1}{3}})$, or $\mathcal{O}(N^{\frac{1}{3}})$ if C_{FF} is polynomial. □

As said earlier, the algorithm presented in [28] is a special case of Algorithm 1. Their algorithm can be seen as an instantiation of Algorithm 1 by defining $G_{\mathcal{F}}$ (resp. $G_{\mathcal{P}}$) to be the linearity graph of $\mathcal{F}$ (resp. $\mathcal{P}$), where a vertex $\mathbf{a}$ is connected to all vertices $\mathbf{x}$ such that $D_{\mathbf{a}}\mathcal{F}(\mathbf{x}) = 0$ (resp. $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$), taking the predicate $\mathbb{P}_\kappa(\mathbf{a}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$ on the universe $\mathcal{M}_{k,N}(q)$, and taking for FindFunction the assumed polynomial-time solver from [37] for $\mathsf{inhQMLE}$. Finding a collision (α, β) such that $\beta = \alpha S$ makes the instance $\mathcal{P}(\mathbf{x}+\alpha) = \mathcal{F}(\mathbf{x}S+\beta)\mathbf{T}$ an inhomogeneous instance by defining $\mathcal{P}'(\mathbf{x}) = \mathcal{P}(\mathbf{x}+\alpha)$ and $\mathcal{F}'(\mathbf{x}) = \mathcal{F}(\mathbf{x}+\beta)$. Running FindFunction on $\mathcal{P}'$ and $\mathcal{F}'$ then returns $\mathbf{S}$ and $\mathbf{T}$. In this case, Lemma 25 gives the precise result from [28, Thm. 1], which we present as a corollary to our Lemma 25, for completeness.

Corollary 26 *Assuming a polynomial-time solver for the inhomogenous case of* QMLE*, an* hQMLE *instance* $(N, k, \mathcal{F}, \mathcal{P})$ *with* $N = k$ *over* $\mathbb{F}_q$ *can be solved with complexity and number of queries equal to* $\mathcal{O}^*(q^{\frac{2}{3}N})$ *with success probability of* $\approx 63\%$ *and a memory complexity of* $\mathcal{O}(q^{\frac{1}{3}N})$*.*

Proof Let $G_{\mathcal{F}}$ (i.e. $G_{\mathcal{P}}$) be the linearity graph of $\mathcal{F}$ (i.e. $\mathcal{P}$), where a vertex $\mathbf{a}$ is connected to all $\mathbf{x}$ such that $D_{\mathbf{a}}\mathcal{F}(\mathbf{x}) = 0$ (i.e. $D_{\mathbf{x}}\mathcal{P}(\mathbf{a}) = 0$). We use the predicate $\mathbb{P}_\kappa(\mathbf{a}) : \dim\ker D_{\mathbf{a}}\mathcal{F} = \kappa$ we have that $\deg(\mathbf{a}) = q^\kappa$. The density of the predicate d_κ in the universe of $N \times N$ matrices is independent of $\mathcal{F}$ and $\mathcal{P}$, and is therefore the same as the density of linear maps with kernel of dimension κ. Thus, d_κ is approximately a monotonic decreasing function in κ, going from 1 to 0. Hence, by Lemma 25, there exists some optimal κ for which we get that $d_\kappa \approx |G_\mathcal{P}|^{-1/3} = q^{-N/3}$, which gives a total time complexity of $q^{\frac{2}{3}N}$ and a memory complexity of $q^{\frac{1}{3}N}$. □

Remark 27 The assumption on a polynomial-time solver in [28] turns out to be too strong: such a solver exists for random instances, however, for inhQMLE instances as obtained in Corollary 26 the running time is probably not polynomial [29]. Nevertheless, the algorithm and result are valid, but require a different rebalancing depending on C_FF. Section 5 analyzes C_FF in detail for different instances.

To apply this approach to MCE instances, we need to generalize to the case of N not necessarily equal to k. For an MCE instance $\mathcal{I}_\mathsf{MCE}(n, m, k, \mathcal{C}, \mathcal{D})$, we get an hQMLE instance $\mathcal{I}_\mathsf{QMLE}(n+m, k, \mathcal{F}, \mathcal{P})$ by Theorem 15. We take again the predicate $\mathbb{P}_\kappa(\mathbf{a}) : \dim\ker D_\mathbf{a}\mathcal{F} = \kappa$, but this time on the universe $\mathcal{M}_{k,n+m}(q)$, where $D_\mathbf{a}\mathcal{F}$ lives. To get a similar result to Corollary 26, we need to show two things. **a)**, that this predicate satisfies the assumptions required for Algorithm 1. **b)**, that there is a κ such that the density d_κ of $\mathbb{P}_\kappa$ is optimal as described in Lemma 25. If both are satisfied, we get a complexity of $\mathcal{O}(q^{\frac{2}{3}(n+m)} C_\mathrm{FF}^{\frac{1}{3}})$, hence $\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$ when the solver is polynomial, with a probability of $1 - 1/e \approx 63\%$ of success for an MCE instance $\mathcal{I}_\mathsf{MCE}(n, m, k, \mathcal{C}, \mathcal{D})$. We start with **a)**.

Lemma 28 *The predicate* $\mathbb{P}_\kappa(D_\mathbf{a}\mathcal{F}) : \dim\ker D_\mathbf{a}\mathcal{F} = \kappa$ *is a suitable predicate for Algorithm 1, as **i)*** $\mathbb{P}_\kappa$ *can be computed in polynomial time, **ii)** is invariant under equivalence, **iii)** and* d_κ *does not depend on* $\mathcal{F}$.

Proof

1. The cost $C_{\mathbb{P}_\kappa}$ is the cost of computing $\dim\ker D_\mathbf{a}\mathcal{F}$, i.e. computing the kernel of a $k \times (n+m)$ matrix over $\mathbb{F}_q$. This can be done in polynomial time.
2. Let $\mathcal{P}(\mathbf{x}) = \mathcal{F}(\mathbf{xS})\mathbf{T}$ be the equivalence. If $\mathbf{x} \in \ker D_\mathbf{a}\mathcal{P}$ then $\mathbf{xS} \in \ker \mathcal{F}_{\mathbf{aS}}$ and vice versa, as $\mathbf{T}$ does not affect the kernel. As $\mathbf{S}$ is invertible, we get a one-to-one correspondence $\mathbf{x} \mapsto \mathbf{xS}$ between the kernels, so $\mathbb{P}_\kappa(D_{\mathbf{aS}}\mathcal{F}) = \mathbb{P}_\kappa(D_\mathbf{a}\mathcal{P})$.
3. For $\mathcal{F}$ coming from an MCE-instance, we always have $-\mathbf{a} \in \ker D_\mathbf{a}\mathcal{F}$. We want to show that the distribution of the rank of $D_\mathbf{a}\mathcal{F}$ follows the ranks of linear maps vanishing at $-\mathbf{a}$. This is given by [35, Thm. 2] for even characteristic and easily adapted to odd characteristic, which shows d_κ is independent of $\mathcal{F}$.

□

We now continue with **b)**: we show that there is a κ such that d_κ is optimal. For now, existence of κ is enough to derive a complexity on MCE. We will explicitely compute κ later, in Section 5, when we have a detailed view of C_FF for specific parameter sets (k, n, m).

The general density d_κ for the predicate $\mathbb{P}_\kappa$ is given by the following lemma, taking $a = k$ and $b = n+m$ to avoid confusion with regards to n, m and $n+m$.

Lemma 29 *Define the predicate* $\mathbb{P}_\kappa : \dim\ker \mathbf{M} = \kappa$ *for* $\mathbf{M} \in U = \mathcal{M}_{a,b}(q)$ *with* $a \geqslant b$. *Then the density of the predicate* $\mathbb{P}_\kappa$ *is* $d_\kappa \approx q^{-(\kappa^2+\kappa\cdot(a-b))}$.

Proof There are $|U| = q^{ab}$ matrices in $\mathcal{M}_{a,b}(q)$, out of which

$$\prod_{i=0}^{r-1} \frac{(q^a - q^i)(q^b - q^i)}{q^r - q^i} = \mathcal{O}\left(q^{(a+b-r)r}\right)$$

have rank r [49]. We have $\kappa = b - r$ and so $d_\kappa^{-1} = \frac{|U|}{|U_\top|} \approx \frac{q^{ab}}{q^{-(a+b-r)r}} = q^{\kappa^2+\kappa(a-b)}$. Specifically when the matrix is square, $d_\kappa^{-1} \approx q^{\kappa^2}$. □

From Lemma 29 we can conclude that for some κ, the density d_κ is optimal. This means we satisfy both **a)** and **b)** and we can apply Lemma 25.

In conclusion, we get our first result on the hardness of MCE, which significantly impoves straightforward enumeration. This requires that such a κ exists, which happens when $k \leqslant 2(n+m)$, by Lemma 29. Note that, in contrast to [28, Thm. 1], we do not assume a polynomial-time solver for the inhomogeneous case of QMLE. Instead, we write this cost as C_{FF} and explore the precise cost in Section 5.

Theorem 30 *An* MCE *instance* $(n, m, k, \mathcal{F}, \mathcal{P})$ *over* $\mathbb{F}_q$ *with* $k \leqslant 2(n+m)$ *can be solved using Algorithm 1 with time complexity equal to* $\mathcal{O}(q^{\frac{2}{3}(n+m)} \cdot C_{\mathrm{FF}}^{\frac{1}{3}} \cdot (C_{\mathbb{P}_\kappa} + 1))$, *memory complexity equal to* $\mathcal{O}(q^{\frac{1}{3}(m+n)} C_{\mathrm{FF}}^{-\frac{1}{3}})$ *and with success probability of* $\approx 63\%$, *where* C_{FF} *denotes the cost of a single query to* FindFunction.

We will show in Section 5 that, even though C_{FF} is *not* polynomial-time, the complexity of Algorithm 1 is still $\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$ for some optimal κ.

Remark 31 When $k \geq 2(n+m)$ we can no longer assume elements with $\dim \ker D_{\mathbf{a}}\mathcal{F} > 1$ exist, as practically all differentials $D_{\mathbf{a}}\mathcal{F}$ will have only the trivial kernel spanned by $-\mathbf{a}$. In such a scenario, the best we can do is to take a single element $\mathbf{a}$ and run FindFunction on $(\mathbf{a}, \mathbf{b})$ for all $\mathbf{b} \in \mathbb{F}_q^{n+m}$ until we find the isometry. This deterministic process has a time complexity of $\mathcal{O}(q^{(n+m)} \cdot C_{\mathrm{FF}})$. The memory requirements of this algorithm are negligible, since we do not build lists of elements.

4.3 Second algorithm

The algorithm that we presented in the previous section does not take advantage of the bilinear structure of an instance of MCE when viewed as hQMLE. In such a case, the differential $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}$ of a k-dimensional bilinear form admits a special structure.

Lemma 32 *Let* $\mathcal{F}(\mathbf{x}, \mathbf{y})$ *be a* k*-dimensional bilinear form with* $\mathbf{x} \in \mathbb{F}_q^m$ *and* $\mathbf{y} \in \mathbb{F}_q^n$. *Let* $\mathbf{F}_\mathbf{a}$ *denote the* $k \times n$ *matrix of the linear map* $\mathcal{F}(\mathbf{a}, -) : \mathbb{F}_q^n \to \mathbb{F}_q^k$ *for a fixed* $\mathbf{a} \in \mathbb{F}_q^m$. *Similarly let* $\mathbf{F}_\mathbf{b}$ *denote the* $k \times m$ *matrix of the linear map* $\mathcal{F}(-, \mathbf{b}) : \mathbb{F}_q^m \to \mathbb{F}_q^k$ *for a fixed* $\mathbf{b} \in \mathbb{F}_q^n$. *Then*

$$D_{(\mathbf{a},\mathbf{b})}\mathcal{F}(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \mathbf{F}_\mathbf{b} & \mathbf{F}_\mathbf{a} \end{pmatrix} \begin{pmatrix} \mathbf{x}^\top \\ \mathbf{y}^\top \end{pmatrix}.$$

Proof By bilinearity, $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}(\mathbf{x},\mathbf{y}) := \mathcal{F}(\mathbf{x}+\mathbf{a},\mathbf{y}+\mathbf{b}) - \mathcal{F}(\mathbf{x},\mathbf{y}) - \mathcal{F}(\mathbf{a},\mathbf{b})$ equals $\mathcal{F}(\mathbf{a},\mathbf{y}) + \mathcal{F}(\mathbf{x},\mathbf{b}) = \mathbf{F_a}\mathbf{y}^\top + \mathbf{F_b}\mathbf{x}^\top$. □

Similarly for $\mathcal{P}$, we use the notation $\mathbf{P_a}$ and $\mathbf{P_b}$. The equivalence in such a case becomes $\mathcal{P}(\mathbf{x},\mathbf{y}) = \mathcal{F}(\mathbf{xA},\mathbf{yB}^\top)\mathbf{T}$, with $\mathbf{A},\mathbf{B}$ precisely the matrices from the MCE instance. Then, as $\mathcal{F}$ and $\mathcal{P}$ are bilinear, one can see SampleSet in Algorithm 1 as sampling both $\mathbf{a} \in \mathbb{F}_q^n$ and $\mathbf{b} \in \mathbb{F}_q^m$ at the same time as one $(\mathbf{a},\mathbf{b}) \in \mathbb{F}_q^{n+m}$, until $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}$ has a kernel of dimension κ. However in the bilinear case, $\mathbf{a}$ influences only the matrix $\mathbf{F_a}$, and $\mathbf{b}$ influences only $\mathbf{F_b}$. Hence, we can sample $\mathbf{a} \in \mathbb{F}_q^m$ and $\mathbf{b} \in \mathbb{F}_q^n$ separately. This hints that we can apply ideas from Algorithm 1 to the smaller universes $U_\mathbf{a} = \mathcal{M}_{k,n}(q)$ and $U_\mathbf{b} = \mathcal{M}_{k,m}(q)$, where $\mathbf{F_a}$ and $\mathbf{F_b}$ live. By finding well-chosen predicates in these smaller universes, we hope to find collisions faster.

We first analyse the preoperties of $\mathbf{F_a}$ and $\mathbf{F_b}$ a bit more. Let $\mathfrak{F}_a$ be the set of elements $\mathbf{a}$ for which $\dim\ker\mathbf{F_a}$ is non-trivial, and $\mathfrak{F}_b$ similarly, i.e.

$$\mathfrak{F}_a = \{\mathbf{a} \in \mathbb{F}_q^m \mid \dim\ker\mathcal{F}(\mathbf{a},-) > 0\}, \quad \mathfrak{F}_b = \{\mathbf{b} \in \mathbb{F}_q^n \mid \dim\ker\mathcal{F}(-,\mathbf{b}) > 0\}.$$

For $\mathcal{P}$, we define $\mathfrak{P}_a$ and $\mathfrak{P}_b$ similarly. For isomorphic bilinear forms $\mathcal{F}$ and $\mathcal{P}$, these sets have special properties.

Lemma 33 *Let $(\mathbf{A},\mathbf{B},\mathbf{T}) : \mathcal{F} \to \mathcal{P}$ be an isomorphism between two k-tuples of bilinear homogenous quadratic polynomials $\mathcal{F}$ and $\mathcal{P}$, such that $\mathcal{P}(\mathbf{x},\mathbf{y}) = \mathcal{F}(\mathbf{xA},\mathbf{yB}^\top)\mathbf{T}$. We have the following properties:*

1. *Given $\mathbf{a} \in \mathfrak{F}_a$ and any $\mathbf{b} \in \ker\mathbf{F_a}$, we get $\mathcal{F}(\mathbf{a},\mathbf{b}) = 0$.*
2. *$\mathfrak{F}_b$ is completely determined by $\mathfrak{F}_a$, as $\mathfrak{F}_b = \bigcup_{\mathbf{a}\in\mathfrak{F}_a}\ker\mathbf{F_a}$.*
3. *For $\mathbf{a} \in \mathbb{F}_q^n$ and $\mathbf{y} \in \mathbb{F}_q^m$, we have $\mathbf{P_a}(\mathbf{y}) = \mathbf{F_{aA}}(\mathbf{yB}^\top)\mathbf{T}$.*
4. *For $\mathbf{a} \in \mathbb{F}_q^n$, we get $\ker\mathbf{P_a} = \ker\mathcal{F}_{\mathbf{aA}} \cdot \mathbf{B}^\top$.*
5. *The isomorphism $(\mathbf{A},\mathbf{B},\mathbf{T})$ induces the bijections*

$$\mathfrak{P}_a \to \mathfrak{F}_a : \mathbf{a} \mapsto \mathbf{aA}, \quad \mathfrak{P}_b \to \mathfrak{F}_b : \mathbf{b} \mapsto \mathbf{bB}^\top.$$

Proof

1. $\mathbf{b} \in \ker\mathbf{F_a}$ is equivalent by definition to $\mathbf{F_a}\mathbf{b}^\top = \mathcal{F}(\mathbf{a},\mathbf{b}) = \mathbf{0}$.
2. This follows directly from 1.: $\mathbf{b} \in \mathfrak{F}_b$ only if there exists an $\mathbf{a} \in \mathfrak{F}_a$ such that $\mathcal{F}(\mathbf{a},\mathbf{b}) = \mathbf{0}$. But then $\mathbf{b} \in \ker\mathbf{F_a}$ for this specific $\mathbf{a}$.
3. Per definition $\mathbf{P_a}(\mathbf{y}) = \mathcal{P}(\mathbf{a},\mathbf{y}) = \mathcal{F}(\mathbf{aA},\mathbf{yB}^\top)\mathbf{T} = \mathbf{F_{aA}}(\mathbf{yB}^\top)\mathbf{T}$.
4. This follows directly from 3.: as $\mathbf{T}$ is invertible, it does not affect the kernels, so $\mathbf{y} \in \ker\mathbf{P_a}$ if and only if $\mathbf{yB}^\top \in \ker\mathbf{F_{aA}}$
5. This follows directly from 4.: Given $\mathbf{a} \in \mathfrak{P}_a$ we get $\mathbf{aA} \in \mathfrak{F}_a$ and vice versa as $\mathbf{A} \in \mathrm{GL}_m(q)$. A similar argument gives $\mathfrak{F}_b \to \mathfrak{P}_b$.

□

Lemma 33 shows that $\mathbf{a} \in \mathfrak{F}_a$ and $\mathbf{b} \in \mathfrak{F}_b$ describe all non-trivial roots $(\mathbf{a},\mathbf{b})$ of a given $\mathcal{F}$. For an instance $(\mathbf{A},\mathbf{B},\mathbf{T}) : \mathcal{F} \to \mathcal{P}$, Item 5 shows that non-trivial roots are mapped bijectively by $(\mathbf{A},\mathbf{B},\mathbf{T})$. Such non-trivial roots can be used to find collisions more easily between $\mathcal{F}$ and $\mathcal{P}$. However, this requires that instances $\mathcal{F} \to \mathcal{P}$ *have* non-trivial roots. We can get an estimate on the sizes of $\mathfrak{F}_a$, $\mathfrak{F}_b$, $\mathfrak{P}_a$, and $\mathfrak{P}_b$ for given parameters n, m, and k, in the following way.

Lemma 34 *When $k \geqslant n$, $|\mathfrak{F}_a| = |\mathfrak{P}_a| \approx q^{2n-k-1}$ and $|\mathfrak{F}_b| = |\mathfrak{P}_b| \approx q^{2m-k-1}$.*

Proof By Lemma 33 we get $|\mathfrak{F}_a| = |\mathfrak{P}_a|$. Then, using Lemma 29 we see that the size of these sets is dominated by elements $\mathbf{a}$ with $\kappa = \dim\ker \mathbf{F_a} = 1$ (a one-dimensional kernel). From the same lemma, the density of $\kappa = \dim\ker \mathbf{F_a} = 1$ elements is $d_1 = q^{-(1+1\cdot(k-n))}$. Hence we expect $d_1 \cdot q^n = q^{2n-k-1}$ such elements. A similar argument gives $|\mathfrak{F}_b| = |\mathfrak{P}_b| \approx q^{2m-k-1}$. □

Summarizing specifically for parameters of cryptographical interest, this means

Corollary 35 *Assuming $n = m$ as the hardest case, an* MCE *instance $\mathcal{I}_{\mathsf{MCE}}(n, m, k, \mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ has non-trivial roots*

- *with* very high *probability, when $k < 2n$,*
- *with probability $\frac{1}{q}$, when $k = 2n$,*
- *with* very low *probability, when $k > 2n$.*

From these results, we can expect non-trivial roots for an MCE instance $(k, n, m, \mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$ with $k \leq n + m$. These non-trivial roots can be seen as a suitable predicate on the smaller universes $U_\mathbf{a}$ and $U_\mathbf{b}$: we search for collisions $(\mathbf{a}, \mathbf{b}) \times (\mathbf{aA}, \mathbf{bB}^\top)$, where $(\mathbf{a}, \mathbf{b})$ is a non-trivial root of $\mathcal{P}$, and $(\mathbf{aA}, \mathbf{bB}^\top)$ of $\mathcal{F}$. Given such a collision, we proceed as in Section 4.2.

The following result shows that we always find such a collision if $\mathcal{F}$ and $\mathcal{P}$ have non-zero roots.

Lemma 36 *Let $\mathfrak{F}_a$, $\mathfrak{F}_b$ and $\mathfrak{P}_a$, $\mathfrak{P}_b$ describe the non-trivial roots of an* MCE *instance $\mathcal{I}_{\mathsf{MCE}}(n, m, k, \mathcal{F}, \mathcal{P})$ over $\mathbb{F}_q$. Let $\mathbf{x} = (\mathbf{a}, \mathbf{b}) \in \mathfrak{F}_a \times \mathfrak{F}_b$, then looping over $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$ gives a collision $(\mathbf{x}, \mathbf{y})$ with certainty.*

Proof This follows quickly from Lemma 33. We have $\mathbf{x} = (\mathbf{a}, \mathbf{b})$ and two bijections $\mathfrak{F}_a \to \mathfrak{P}_a$ and $\mathfrak{F}_b \to \mathfrak{P}_b$, so $\mathbf{x}$ is mapped to some $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$. As this set is finite, we can loop over it in a finite number of steps until we find the collision. □

Therefore, as soon as we have non-trivial roots, we can use a single one of them to find a collision. This leads to the following pseudo-algorithm:

1. compute $\mathfrak{F}_b$ by computing $\ker \mathbf{F_b}$ for all $\mathbf{b} \in \mathbb{F}_q^m$,
2. if $\mathfrak{F}_b$ is non-empty, compute $\mathfrak{F}_a$ using Lemma 33-2. Same for $\mathfrak{P}_a$ and $\mathfrak{P}_a$.
3. sample a single $\mathbf{x} \in \mathfrak{F}_a \times \mathfrak{F}_b$
4. loop over $\mathbf{y} \in \mathfrak{P}_a \times \mathfrak{P}_b$ with FindFunction$(\mathbf{x}, \mathbf{y})$ until the solver finds μ.

Corollary 37 *Assuming $m \leqslant n$, the above algorithm terminates successfully and has a total complexity of $\mathcal{O}(q^m \cdot C_{\mathbb{P}_\kappa} + q^{2(n+m-k-1)} \cdot C_{\mathrm{FF}})$, where $C_\mathbb{P}$ denotes the cost of computing* $\ker \mathbf{F_b}$ *and C_{FF} denotes the cost of a single query to* FindFunction*.*

Proof Building $\mathfrak{F}_b$ and $\mathfrak{P}_b$ has a complexity of $\mathcal{O}(q^m \cdot C_{\mathbb{P}_\kappa})$, and these give us $\mathfrak{F}_a$ and $\mathfrak{P}_a$ by Lemma 33. Then for every step in the loop we get a query to FindFunction. By Lemma 34, the size of $\mathfrak{P}_a \times \mathfrak{P}_b$ is at most $\mathcal{O}(q^{2(n+m-k-1)})$. □

We will see later in Section 5 that the dominating complexity is $q^m \cdot C_{\mathbb{P}_\kappa}$ as for specific parameters (k, n, m) the number of queries z can be reduced so that $z \cdot C_{\mathrm{FF}} < q^m$. As $C_{\mathbb{P}_\kappa}$ is polynomial, we get a complexity of $\mathcal{O}^*(q^m)$ for such instances.

For efficiency, one can decrease further the number of queries to FindFunction by applying other, secondary predicates. For example, the sets $\mathfrak{F}_a \times \mathfrak{F}_b$ and $\mathfrak{P}_a \times \mathfrak{P}_b$ can be split into zeros $\mathfrak{F}^{\mathbf{0}} = \{\mathbf{x} \in \mathbb{F}_q^{n+m} | \mathcal{F}(\mathbf{x}) = \mathbf{0}\}$ and non-zeros $\mathfrak{F} = \mathfrak{F}_a \times \mathfrak{F}_b \setminus \mathfrak{F}^{\mathbf{0}}$, which reduces the collision search to each of these sets. Another secondary predicate is to only use elements $\mathbf{a}$ with $\dim \ker \mathbf{F_a} = \kappa$ for some specific value $\kappa > 0$.

We summarize the MCE solver for instances with roots in Algorithm 2. Practically, since the algorithm is deterministic, we do not need to build and store the list $\mathfrak{F}$. We only need to find one element from it. However, for iterating through the list $\mathfrak{P}$, S_a and S_b need to be stored. The estimated size of these lists is $q^{n+m-k-1}$.

Algorithm 2 Bilinear MCE-Solver, assuming $n \geqslant m$.

```
function SampleZeroes($\mathcal{F}$)
    $S, S_a, S_b \leftarrow \emptyset$
    for all $\mathbf{b} \in \mathbb{F}_q^m$ do
        if $\dim \ker \mathbf{F_b} > 0$ then
            $S_b \leftarrow S_b \cup \{\mathbf{b}\}$
            $S_a \leftarrow S_a \cup \ker \mathbf{F_b} \setminus \{0\}$
        end if
    end for
    $S \leftarrow S_a \times S_b$
    return $S$
end function
function CollisionFind($\mathcal{F}, \mathcal{P}$)
    $\mathfrak{F} \leftarrow$ SampleZeroes($\mathcal{F}$)
    $\mathfrak{P} \leftarrow$ SampleZeroes($\mathcal{P}$)
    $\mathbf{x} \xleftarrow{\$} \mathfrak{F}$
    for all $\mathbf{y} \in \mathfrak{P}$ do
        $\mu \leftarrow$FindFunction($\mathbf{x}, \mathbf{y}$)
        if $\mu \neq \perp$ then
            return solution $\mu$
        end if
    end for
    return $\perp$
end function
```

The next theorem summarises this subsection on using bilinearity to solve MCE.

Theorem 38 *Assuming a solver for the inhomogenous case of* QMLE *with cost* C_{FF}*, an* MCE *instance* $(n, m, k, \mathcal{F}, \mathcal{P})$ *over* $\mathbb{F}_q$ *with* $n \geqslant m$ *and roots existing for* $\mathcal{F}$ *and* $\mathcal{P}$ *can be solved using Algorithm 2 with* $\mathcal{O}(q^m \cdot C_{\mathbb{P}_\kappa})$ *operations in* SampleZeroes *and* z *queries to the solver. This amounts to a total time complexity of* $\mathcal{O}(q^m \cdot C_{\mathbb{P}_\kappa} + z \cdot C_{\mathrm{FF}})$*. The memory complexity of the algorithm is* $\mathcal{O}(q^{n+m-k-1})$*.*

We will show in Section 5 that, even though C_{FF} is *not* polynomial-time, the dominating factor in this complexity is still $q^m \cdot C_{\mathbb{P}_\kappa}$, where $C_{\mathbb{P}_\kappa}$ is the cost to compute the kernel of an $m \times k$ matrix.

5 Filling the gaps in the complexity analysis

The cost $C_{\mathbb{P}}$ is polynomial in all of the cases because it either requires computing the rank of a linear map or sampling a random element from a set. The FindFunction in Algorithms 1 and 2 checks whether a given pair of vectors is a collision, and if so, it returns the solution to the MCE instance. This is done by solving an instance of the inhBMLE that has the same solutions as the input MCE instance. Thus, to estimate the value of C_{FF}, we analyse the complexity of inhBMLE on these instances, by relying on algorithms that have been developed for the inhQMLE case with $N = k$.

5.1 Algorithms for inhQMLE

The two algorithms described in this section have been used for tackling the inhQMLE problem within the birthday-based algorithm for hQMLE [28, 29]. Their analysis is thus important to estimate C_{FF}. In Section 5.2 we adapt this analysis for the inhBMLE case with arbitrary k and N and we see how this affects Algorithms 1 and 2 for different parameter sets.

5.1.1 The Gröbner attack

The algebraic attack on the inhQMLE problem starts by reducing $\mathcal{P}(\mathbf{x})\mathbf{T}^{-1} = \mathcal{F}(\mathbf{x}\mathbf{S})$, with $\mathbf{S}$ and $\mathbf{T}$ unknown, to a system of polynomial equations. By rewriting the problem in matrix form we obtain the following constraints

$$\begin{aligned} \sum_{1\leqslant r\leqslant k} \widetilde{T}_{rs}\mathbf{P}^{(r)} &= \mathbf{S}\mathbf{F}^{(s)}\mathbf{S}^{\top}, \ \forall s, 1\leqslant s\leqslant k, \\ \mathbf{P}^{[1]}\mathbf{T}^{-1} &= \mathbf{S}\mathbf{F}^{[1]}, \\ \mathbf{P}^{[0]}\mathbf{T}^{-1} &= \mathbf{F}^{[0]}, \end{aligned} \tag{13}$$

where $\mathbf{F}^{[1]} \in \mathbb{F}_q^{N\times k}$ and $\mathbf{P}^{[1]} \in \mathbb{F}_q^{N\times k}$ describe the degree-1 homogeneous part of an inh(Q/B)MLE instance and $\mathbf{F}^{[0]} \in \mathbb{F}_q^{k}$ and $\mathbf{P}^{[0]} \in \mathbb{F}_q^{k}$ describe the degree-0 part. We will denote the subsystem of equations derived from the degree-d homogeneous part as $\mathcal{S}_d$. The resulting system can be solved using Gröbner basis algorithms and this is referred to as the Gröbner attack [37]. The observation that $\mathbf{S}$ and $\mathbf{T}$ are common solutions to homogeneous parts of separate degrees of an inhQMLE instance (also proven in [39, Lemma 1]) and the idea that moving $\mathbf{T}$ to the other side of the equality results in a lower degree system where we solve for $\mathbf{T}^{-1}$ originate from this work.

The complexity of Gröbner basis algorithms depends foremost on the *degree of regularity*, which is usually hard to estimate, but it can sometimes be observed through experimental work. Such experiments applied to inhQMLE instances imply that the system is solved at degree three. A degree-three linearized system in n variables is represented by a matrix of size roughly n^3 and thus, Gaussian Elimination on such a system is performed in $\mathcal{O}(n^{3\omega})$ operations, where ω is the linear algebra constant. This reasoning leads to the assumption that there exists a polynomial-time solver for the inhomogeneous case of QMLE. Another empirical observation made in [37] is that the time to construct the system exceeds the time of the Gröbner basis computation. Since the generation of the system is known to be polynomial, this suggests that the Gröbner basis computation is performed in polynomial time as well. However, these experiments are performed on random inhomogeneous instances of the QMLE problem.

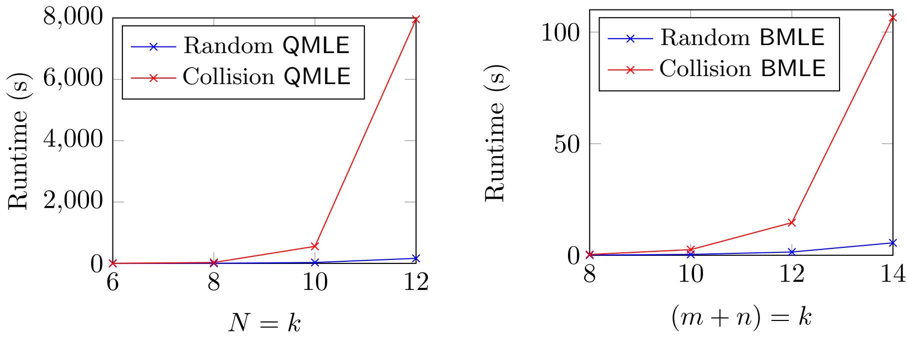

Figure 2: Comparison of runtime for solving random and collision-derived $\mathsf{inh(Q/B)MLE}$ instances using the Gröbner attack. Results are averaged over 50 runs.

In the birthday-based approach for solving QMLE, $\mathbf{F}^{[1]}$, $\mathbf{P}^{[1]}$, $\mathbf{F}^{[0]}$ and $\mathbf{P}^{[0]}$ are obtained from a collision [28]. Specifically, if we have a collision on $\mathbf{x} \in \mathbb{F}_q^N$ and $\mathbf{y} \in \mathbb{F}_q^N$ such that $\mathbf{y} = \mathbf{x}\mathbf{S}$, they are obtained as

$$\mathbf{F}^{[1]} = D_{\mathbf{y}}\mathcal{F}, \qquad \mathbf{P}^{[1]} = D_{\mathbf{x}}\mathcal{P},$$
$$\mathbf{F}^{[0]} = \mathcal{F}(\mathbf{y}), \qquad \mathbf{P}^{[0]} = \mathcal{P}(\mathbf{x}).$$

Instances of $\mathsf{inhQMLE}$ derived from a collision are, on average, harder to solve than random $\mathsf{inhQMLE}$ instances. Recall that in Algorithm 1 the instances of $\mathsf{inhQMLE}$ are chosen such that $\dim\ker D_{\mathbf{y}}\mathcal{F} = \dim\ker D_{\mathbf{x}}\mathcal{P} = \kappa$. Hence, the number of linearly independent equations in $\mathcal{S}_1$ is exactly $k(N - \kappa)$, instead of the expected kN on average. The size of $\mathcal{S}_0$ can also depend on the predicate that we choose for the birthday-based algorithm. For instance, when we use the predicate of searching for a collision between the non-trivial roots of $\mathcal{P}$ and $\mathcal{F}$, we obtain no equations in $\mathcal{S}_0$. Additionally, since $\mathbf{F}^{[1]}$ (i.e. $\mathbf{P}^{[1]}$) and $\mathbf{F}^{[0]}$ (i.e. $\mathbf{P}^{[0]}$) are obtained respectively from computing the differential of and evaluating $\mathcal{F}$ (i.e $\mathcal{P}$) at a given point, $\mathcal{S}_1$ and $\mathcal{S}_0$ are not as independent from $\mathcal{S}_2$ as they would be in the random case. It is difficult to estimate the complexity of solving these instances compared to solving random instances with the same structure. Figure 2 shows experiments confirming our intuition that the complexity of collision-derived instances is worse than that of random ones. This implies that we can not rely on the experimental observations in [37] to estimate the complexity of these specific instances. We conclude that, in contrast with the literature, we can not assume that C_{FF} is polynomial when the Gröbner attack is used.

5.1.2 The matrix-pencil attack

The matrix-pencil attack was proposed in Bouillaguet's thesis [29] and used for the implementation of the birthday-based attack [28]. This algorithm has a complexity of $\mathcal{O}(N^6)$ with non-negligible probability for random $\mathsf{inhQMLE}$ instances where $N = k$. Its complexity for $\mathsf{inhQMLE}$ instances derived from a collision attack depends strongly on the parameter κ. We give a general description of the approach. For details on how it relates to the matrix pencil equivalence problem, we refer to [29, Ch. 14].

The first step is to retrieve a basis of the solution space V of the subsystem of linear equations $\mathcal{S}_1$. Let $\ell = \dim V$ and let $(\mathbf{S}^{[1]}, \mathbf{T}^{[1]}), \ldots, (\mathbf{S}^{[\ell]}, \mathbf{T}^{[\ell]})$ be a basis of V. Once the solution space of $\mathcal{S}_1$ is known, in order to find the solution space of the overall system one rewrites $\mathcal{S}_2$ as a system in ℓ variables. Concretely, this is done by replacing $\mathbf{S}$ and $\mathbf{T}$ by $\sum_{i=1}^{\ell} x_i \mathbf{S}^{[i]}$ and $\sum_{i=1}^{\ell} x_i \mathbf{T}^{[i]}$ in Equation (13) and then looking for solutions in variables $x_1, \ldots, x_\ell$. This standard approach is also described in [39]. A key idea in the matrix-pencil attack is to use the knowledge of $\mathbf{F}^{[1]}/\mathbf{P}^{[1]}$ and $\mathbf{F}^{[0]}/\mathbf{P}^{[0]}$ to find a (second) collision and double the number of linear equations in $\mathcal{S}_1$. Supposing that there exists $\mathbf{x}'$ such that $\mathbf{x}'\mathbf{P}^{[1]} = \mathbf{P}^{[0]}$, we infer that there also exists $\mathbf{y}'$ such that $\mathbf{y}'\mathbf{F}^{[1]} = \mathbf{F}^{[0]}$ and that $\mathbf{y}' = \mathbf{x}'\mathbf{S}$. We can thus append the equations obtained from $(D_{\mathbf{x}'}\mathcal{P})\mathbf{T}^{-1} = \mathbf{S}(D_{\mathbf{y}'}\mathcal{F})$ to $\mathcal{S}_1$. After applying this technique, the resulting system is usually highly overdetermined and can be solved through direct linearization. The most favorable case is when $\mathbf{x}'$ and $\mathbf{y}'$ are uniquely identified. However, if $\dim\ker \mathbf{F}^{[1]} = \kappa > 1$, then $\mathbf{x}'$ is chosen arbitrarily and we loop through the q^κ possible values for $\mathbf{y}'$. The complexity of the algorithm is $\mathcal{O}(q^\kappa \ell^2 N^4)$, under the condition that $\ell(\ell+1)/2 \leq |\mathcal{S}_2|$. Another condition for the success of this approach is that $\mathcal{P}(\mathbf{x}) \neq 0$ and $\exists \mathbf{x}, \mathbf{x} D_{\mathbf{x}}\mathcal{P} = \mathcal{P}(\mathbf{x})$, because this assumption is used to find the second collision. As per the analysis in [29], the probability that the condition for success is met is $1 - 1/q + 1/q^3 + \mathcal{O}(1/q^6)$.

5.2 The complexity of inhBMLE for three parameter sets

In the following analysis, we use the matrix-pencil algorithm as the inhBMLE solver, as it seems to outperform the Gröbner attack and we have a better understanding of its complexity for these specific instances.

5.2.1 The case $k \leq n + m$

The case when $k \leq n + m$ uses Algorithm 2, where the complexity is dominated by the SampleZeroes function, as long as the complexity of the inhBMLE solver does not surpass $\mathcal{O}(q^m)$. In the matrix-pencil algorithm, we can not use the zero subsets $\mathfrak{F}^{\mathbf{0}}$ and $\mathfrak{P}^{\mathbf{0}}$, as this contradicts its condition for success $\mathcal{P}(\mathbf{x}) \neq 0$. The non-zeros subsets $\mathfrak{F}$ and $\mathfrak{P}$ can be used with a small adjustment to the algorithm: after finding a basis of the solution space of $\mathcal{S}_1$, we rewrite and solve through linearization the system comprised of both $\mathcal{S}_2$ and $\mathcal{S}_0$. Note that $\mathfrak{F}$ and $\mathfrak{P}$ are non-empty only when the instance has at least two roots. Since in Algorithm 2 we do not restrict the value of κ, we will approximate to the one that has the highest probability, which for the case of $k \leq n + m$ is $\kappa = (m + n) - k$. Hence, C_{FF} is approximated to

$$\mathcal{O}(q^{m+n-k} \cdot (m+n)^6).$$

When $k \geq m$, this is always smaller than $\mathcal{O}(q^m)$.

5.2.2 The case $n + m < k < 2(n + m)$

In this case we use Algorithm 1. Since the complexity of the inhBMLE solver contains a non-negligible factor of q^κ, the choice of κ needs to be adapted, so that the running times of SampleSet and CollisionFind are equal. Let $N = n+m$ and let $r = N-k$. The optimal κ is chosen such that

$$q^{\frac{N-(\kappa^2+\kappa r)}{2}} \cdot q^{\kappa^2+\kappa r} \approx q^{N-(\kappa^2+\kappa r)} \cdot q^\kappa.$$

This gives us $\kappa = \frac{k-(n+m+\sqrt{\delta})}{2} + \frac{1}{3}$, with $\delta = (k-(n+m))^2 + \frac{4}{3}(k+\frac{1}{3})$. The complexity of the overall algorithm with this optimal choice for κ is then

$$q^{\frac{n+m}{2}+\frac{k-\sqrt{\delta}}{6}+\frac{1}{9}}.$$

We get that $\sqrt{\delta} \geqslant |k-(n+m)|$ and so for all values of k between $n+m$ and $2(n+m)$, the term $k-\sqrt{\delta}$ is bounded by $n+m$, and hence this gives a bound on the complexity by $\mathcal{O}(q^{\frac{2}{3}(n+m)+\frac{1}{9}})$. The term $\frac{1}{9}$ adds a few bits at most to this complexity, but is negligible for most cryptographic purposes.

5.2.3 The case $k \geq 2(n+m)$

When $k \geq 2(n+m)$, as per Lemma 29, the probability that there exist elements with $\dim\ker D_{(\mathbf{a},\mathbf{b})}\mathcal{F} > 1$ is extremely small, which is why we can not define a distinguishing predicate for Algorithm 1 and $\kappa = 1$ with overwhelming probability. In this case, the complexity of the matrix-pencil algorithm is

$$\mathcal{O}(q \cdot (m+n)^6),$$

as with random $\mathsf{inhBMLE}$ instances.

5.3 Summary of results

After determining the value of $C_{\mathbb{P}}$ and C_{FF} for all parameter sets, we can summarize the results on the hardness of MCE as follows.

Table 1: Summary of best algorithms and performance for MCE for different values of k, assuming $m \leqslant n$.

Case	Result	Dominating factor	Probability
$m \leqslant k \leqslant n+m$	Theorem 38	$\mathcal{O}^*(q^m)$	100% (deterministic)
$(n+m) \leqslant k \leqslant 2(n+m)$	Theorem 30	$\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$	63% (probabilistic)
$k > 2(n+m)$	Remark 31	$\mathcal{O}^*(q^{(n+m)})$	100% (deterministic)

6 Experimental results

To confirm our theoretical findings, we solved randomly generated positive instances of the MCE problem, using the two approaches presented in this paper. First, we implement the birthday-based Algorithm 1 in three steps. (1) We randomly generate a positive instance $(k, n, m, \mathcal{C}, \mathcal{D})$ of MCE and reduce it to an instance $(m+n, k, \mathcal{F}, \mathcal{P})$ of hBMLE. (2) We build the two sample sets for a predefined predicate $\mathbb{P}$ and we combine them to create pairs of potential collisions. (3) For each pair we create an $\mathsf{inhBMLE}$ instance and we query an $\mathsf{inhBMLE}$ solver until it outputs a solution for the maps $\mathbf{S}$ and $\mathbf{T}$. Our implementation is built on top of the open source birthday-based hQMLE solver from [29], which is implemented in MAGMA [50].

Table 2 shows running times for solving the MCE problem using Algorithm 1. The goal of this first experiments was to confirm that there is a parameter choice where the probability of success of the algorithm surpasses $1 - 1/e$ and that our running times are comparable to the ones given in [28]. These experiments are done with the parameter $q = 2$ and all results are an average of 50 runs.

Table 2: Experimental results on solving the MCE problem using Algorithm 1.

$m = n$	k	κ	Sample set size	Runtime (s) SampleSet	Runtime (s) inhQMLE solver	Success probability
10	20	5	2	21	3154	0.70
11	22	5	3	31	2004	0.63
12	24	5	6	76	13873	0.73

The second approach, described in Section 4.3 uses the bilinear structure of hQMLE instances derived from MCE instances to have an improved algorithm for building the sample sets and a more precise predicate that results in fewer queries to the inhQMLE solver. The consequence of these two improvements to the runtime can be observed in Table 3 where we show experimental results of Algorithm 2 using the non-zeros subsets. Recall that, this approach can be used only when there exist at least two roots of $\mathcal{F}$ and $\mathcal{P}$. Otherwise, the sampled sets contain only the trivial root and the instance is solved using Algorithm 1. Table 3 shows results of the case when the sets are non-trivial and the probability of this case for the given parameters is shown in the last column. For efficiency, we take the minimal subset with a common dimension of the kernel of $\mathbf{F}_\mathbf{b}$, and when looking for collisions, we are careful to skip pairs $(\mathbf{ab}, \mathbf{a}'\mathbf{b}')$ where $\dim\ker \mathbf{F}_\mathbf{b} = \dim\ker \mathbf{P}_{\mathbf{b}'}$ but $\dim\ker D_{(\mathbf{a},\mathbf{b})}\mathcal{F} \neq \dim\ker D_{(\mathbf{a}',\mathbf{b}')}\mathcal{P}$. In these experiments, $q = 3$ and all results are an average of 50 runs.

Table 3: Experimental results on solving the MCE problem using the non-zeros-subsets variant of Algorithm 2.

$m = n$	k	Sample set size	Runtime (s) SampleZeros	Runtime (s) inhQMLE solver	% instances with two roots
8	15	10.4	0.56	175.34	24
	14	35.56	0.60	236.12	68
9	17	12.00	1.74	396.04	22
	16	37.97	1.72	1020.25	70
10	19	25.6	5.13	2822.32	14
	18	36.72	5.05	1809.09	82

Our experiments confirm that Algorithm 2 outperforms Algorithm 1 for solving MCE instances with non-trivial roots.

Acknowledgements.

The authors thank Charles Bouillaguet for providing the implementation resulting from [29].

References

[1] Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 1070, pp. 33–48. Springer, Berlin, Heidelberg (1996)

[2] Ding, J., Schmidt, D.: Rainbow, a New Multivariable Polynomial Signature Scheme. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS. Lecture Notes in Computer Science, vol. 3531, pp. 164–175 (2005)

[3] McEliece, R.J.: A Public-Key System Based on Algebraic Coding Theory. Jet Propulsion Laboratory, California Institute of Technology, 114–116 (1978). DSN Progress Report 44

[4] Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Probl. Control Inf. Theory **15**, 159–166 (1986)

[5] Beullens, W., Preneel, B.: Field Lifting for Smaller UOV Public Keys. In: Patra, A., Smart, N.P. (eds.) Progress in Cryptology – INDOCRYPT 2017, pp. 227–246. Springer, Cham (2017)

[6] Casanova, A., Faugère, J.-C., Macario-Rat, G., Patarin, J., Perret, L., Ryckeghem, J.: GeMSS: A Great Multivariate Short Signature. (2017)

[7] Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Proceedings on Advances in cryptology—CRYPTO '86, pp. 186–194. Springer, London, UK (1987)

[8] Girault, M.: A (Non-Practical) Three-Pass Identification Protocol Using Coding Theory. In: Proceedings of the International Conference on Cryptology on Advances in Cryptology. AUSCRYPT '90, pp. 265–272. Springer, Berlin, Heidelberg (1990)

[9] De Feo, L., Galbraith, S.D.: SeaSign: Compact Isogeny Signatures from Class Group Actions. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, pp. 759–789. Springer, Cham (2019)

[10] De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In: Moriai, S., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2020, pp. 64–93. Springer, Cham (2020)

[11] Biasse, J.-F., Micheli, G., Persichetti, E., Santini, P.: LESS is More: Code-Based Signatures Without Syndromes. In: Nitaj, A., Youssef, A. (eds.) Progress in Cryptology - AFRICACRYPT 2020, pp. 45–65. Springer, Cham (2020)

[12] Barenghi, A., Biasse, J.-F., Persichetti, E., Santini, P.: LESS-FM: Fine-tuning Signatures from the Code Equivalence Problem. Cryptology ePrint Archive, Report 2021/396. https://ia.cr/2021/396 (2021)

[13] Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, pp. 582–612. Springer, Cham (2022)

[14] Ducas, L., van Woerden, W.: On the lattice isomorphism problem, quadratic forms, remarkable lattices, and cryptography. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, pp. 643–673. Springer, Cham (2022)

[15] National Institute for Standards and Technology: NIST Workshop on Cybersecurity in a Post-Quantum World. http://www.nist.gov/itl/csd/ct/post-quantum-crypto-workshop-2015.cfm Accessed 01.10.2014

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

[17] Beullens, W.: Not enough LESS: An improved algorithm for solving Code Equivalence Problems over $\mathbb{F}_q$. Cryptology ePrint Archive, Report 2020/801. https://ia.cr/2020/801 (2020)

[18] Sendrier, N.: Finding the permutation between equivalent linear codes: The support splitting algorithm. IEEE Trans. Inf. Theory **46**, 1193–1203 (2000)

[19] Aragon, N., Blazy, O., Deneuville, J.-C., Gaborit, P., Hauteville, A., Ruatta, O., Tillich, J.-P., Zemor, G., Melchor, C.A., Bettaieb, S., Bidoux, L., Bardet, M., Otmani, A.: ROLLO (Rank-Ouroboros, LAKE and LOCKER) (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions

[20] Melchor, C.A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Zemor, G., Couvreur, A., Hauteville, A.: RQC (2019). https://csrc.nist.gov/projects/post-quantum-cryptography/round-2-submissions

[21] Aragon, N., Gaborit, P., Hauteville, A., Ruatta, O., Zémor, G.: Low Rank Parity Check Codes: New Decoding Algorithms and Applications to Cryptography. IEEE Transactions on Information Theory **65**, 7697–7717 (2019)

[22] Bellini, E., Caullery, F., Gaborit, P., Manzano, M., Mateu, V.: Improved Veron Identification and Signature Schemes in the Rank Metric. 2019

IEEE International Symposium on Information Theory (ISIT), 1872–1876 (2019)

[23] Berger, T.P.: Isometries for rank distance and permutation group of Gabidulin codes. IEEE Trans. Inf. Theory **49**, 3016–3019 (2003)

[24] Grochow, J.A., Qiao, Y.: Isomorphism problems for tensors, groups, and cubic forms: completeness and reductions. arXiv (2019). https://doi.org/10.48550/ARXIV.1907.00309. https://arxiv.org/abs/1907.00309

[25] Couvreur, A., Debris-Alazard, T., Gaborit, P.: On the hardness of code equivalence problems in rank metric (2021)

[26] Futorny, V., Grochow, J.A., Sergeichuk, V.V.: Wildness for tensors. Linear Algebra and its Applications **566**, 212–244 (2019). https://doi.org/10.1016/j.laa.2018.12.022

[27] Belitskii, G.R., Futorny, V., Muzychuk, M., Sergeichuk, V.V.: Congruence of matrix spaces, matrix tuples, and multilinear maps. Linear Algebra and its Applications **609**, 317–331 (2021). https://doi.org/10.1016/j.laa.2020.09.018

[28] Bouillaguet, C., Fouque, P., Véber, A.: Graph-theoretic algorithms for the "isomorphism of polynomials" problem. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7881, pp. 211–227. Springer, ??? (2013). https://doi.org/10.1007/978-3-642-38348-9_13. https://doi.org/10.1007/978-3-642-38348-9_13

[29] Bouillaguet, C.: Algorithms for some hard problems and cryptographic attacks against specific cryptographic primitives. (études d'hypothèses algorithmiques et attaques de primitives cryptographiques). PhD thesis, Paris Diderot University, France (2011). https://tel.archives-ouvertes.fr/tel-03630843

[30] Patarin, J., Goubin, L., Courtois, N.: Improved Algorithms for Isomorphisms of Polynomials. In: EUROCRYPT '98. Lecture Notes in Computer Science, vol. 1403, pp. 184–200. Springer, Berlin, Heidelberg (1998)

[31] Gorla, E.: Rank-metric codes. CoRR **abs/1902.02650** (2019) https://arxiv.org/abs/1902.02650

[32] Hua, L.-K.: A theorem on matrices over a sfield and its applications. In: Bulletin of the American Mathematical Society, vol. 55, pp. 1046–1046 (1949)

[33] Wan, Z.-X.: A proof of the automorphisms of linear groups over a sfield of characteristic 2. Sci. Sinica **11**, 1183–1194 (1962)

[34] Barenghi, A., Biasse, J.-F., Persichetti, E., Santini, P.: On the Computational Hardness of the Code Equivalence Problem in Cryptography. Cryptology ePrint Archive, Paper 2022/967. https://eprint.iacr.org/2022/967 (2022). https://eprint.iacr.org/2022/967

[35] Dubois, V., Granboulan, L., Stern, J.: An efficient provable distinguisher for HFE. In: International Colloquium on Automata, Languages, and Programming, pp. 156–167 (2006). Springer

[36] Fouque, P.-A., Granboulan, L., Stern, J.: Differential Cryptanalysis for Multivariate Schemes. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 341–353. Springer, Berlin, Heidelberg (2005)

[37] Faugère, J.-C., Perret, L.: Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In: Vaudenay, S. (ed.) EUROCRYPT '06. Lecture Notes in Computer Science, vol. 4004, pp. 30–47. Springer, ??? (2006)

[38] Perret, L.: A Fast Cryptanalysis of the Isomorphism of Polynomials with One Secret Problem. In: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3494, pp. 354–370. Springer, Berlin, Heidelberg (2005)

[39] Bouillaguet, C., Faugère, J.-C., Fouque, P.A., Perret, L.: Practical Cryptanalysis of the Identification Scheme Based on the Isomorphism of Polynomial With One Secret Problem. In: Public Key Cryptography – PKC 2011. Lecture Notes in Computer Science, vol. 6571, pp. 441–458. Springer, Berlin, Heidelberg (2011)

[40] Theorie der quadratischen formen in beliebigen korpern. J. Reine Angew. Math. **176**, 31–44 (1937)

[41] Untersuchungen über quadratische formen in korpern der charakteristik 2, i. J. Reine Angew. Math. **183**, 148–167 (1941)

[42] Sergeĭchuk, V.V.: CLASSIFICATION PROBLEMS FOR SYSTEMS OF FORMS AND LINEAR MAPPINGS. Mathematics of the USSR-Izvestiya **31**(3), 481–501 (1988). https://doi.org/10.1070/im1988v031n03abeh001086

[43] Faugère, J.-C., Otmani, A., Perret, L., Portzamparc, F., Tillich, J.-P.: Structural cryptanalysis of mceliece schemes with compact keys. Des. Codes Cryptography **79**(1), 87–112 (2016). https://doi.org/10.1007/s10623-015-0036-z

[44] Faugère, J., Otmani, A., Perret, L., de Portzamparc, F., Tillich, J.: Folding alternant and goppa codes with non-trivial automorphism groups. IEEE Trans. Inf. Theory **62**(1), 184–198 (2016). https://doi.org/10.1109/TIT.2015.2493539

[45] Peters, C.: Information-set decoding for linear codes over $\mathbb{F}_q$. In: International Workshop on Post-Quantum Cryptography, pp. 81–94 (2010). Springer

[46] Nóbrega, R.W., Uchôa-Filho, B.F.: Multishot codes for network coding using rank-metric codes. In: 2010 Third IEEE International Workshop on Wireless Network Coding, pp. 1–6 (2010). IEEE

[47] Alfarano, G.N., Lobillo, F.J., Neri, A., Wachter-Zeh, A.: Sum-rank product codes and bounds on the minimum distance. Finite Fields and Their Applications **80**, 102013 (2022)

[48] Neri, A.: Twisted linearized Reed-Solomon codes: A skew polynomial framework. arXiv preprint arXiv:2105.10451 (2021)

[49] Landsberg, G.: Ueber eine Anzahlbestimmung und eine damit zusammenhängende Reihe. (1893)

[50] Bosma, W., Cannon, J., Playoust, C.: The Magma Algebra System. I. The User Language. J. Symbolic Comput. **24**(3-4), 235–265 (1997). Computational algebra and number theory (London, 1993)