Hardness estimates of the Code Equivalence Problem in the Rank Metric

Krijn Reijnders, Simona Samardjiska, and Monika Trimoska

Radboud Universiteit, Nijmegen, The Netherlands
{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). We show that MCE is equivalent to the homogenous version of the Quadratic Maps Linear Equivalence (QMLE) problem. Using birthday techniques known for IP, we present an algorithm for MCE running in time $\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$, and an algorithm for MCE with roots, running in time $\mathcal{O}^*(q^m)$. We verify these algorithms in practice.

1 Introduction

Equivalence problems are one of the core 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 [26, 15, 23, 25, 6, 11]. 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 [19]. This idea has been revisited many times, being the basis of several signature schemes [26, 21, 13, 14, 7]. Understanding the hardness of these equivalence problems is an essential task in choosing appropriate parameters.

One of these problems is the Code Equivalence problem, which given two codes, asks for the equivalence transformation mapping one to the other. It was first studied by Leon [22] who proposed an algorithm that takes advantage of the Hamming weight being invariant under monomial permutations. It was improved very recently by Beullens [5] using collision-based techniques. Sendrier [30] 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 paper, 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 - MCE (Matrix Code Equivalence). 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 [1, 24, 2, 3]. This problem, and variations of it, has been introduced by Berger in [4], but it was only recently that Couvreur et al. [12] showed first concrete statements about its hardness. Namely, they showed that MCE is at least as hard as the 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.

1.1 Our contributions

In this paper, we investigate the theoretical and practical hardness of the Matrix Code Equivalence (MCE) problem. Our contributions are twofold:

First, we prove that MCE is equivalent to the Quadratic Maps Linear Equivalence (QMLE) problem, by providing a polynomial-time reduction from MCE to QMLE and vice versa. Our work is the first to provide a reduction *to* a known problem, thus giving an upper bound on the hardness of MCE. Furthermore, we establish a so far unknown connection between code equivalence problems and polynomial equivalence problems. This is visualized in Figure 1.

Second, we provide algorithms for solving MCE. The first algorithm is a generalization of a known birthday-based algorithm for QMLE [10]. We show that the algorithm extends to different invariance properties, which helps us prove a complexity for MCE of $\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$ for $m \times n$ matrix codes. We adapt the implementation of [10] and provide benchmarks showing success probability higher than 63%, which is consistent with birthday-based algorithms. The second algorithm uses the bilinear structure of the polynomials arising from MCE to improve the performance of the first algorithm. In particular, when the polynomials have common roots, we provide an algorithm running in time $\mathcal{O}^*(q^m)$ deterministically.

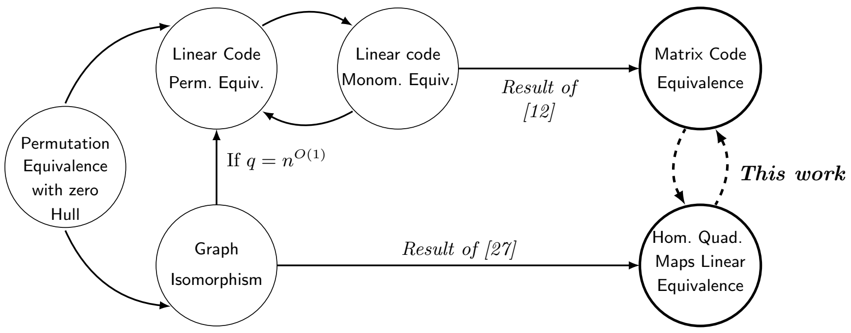

Figure1. Reductions around Matrix Code Equivalence, with our contribution dashed. "A $\longrightarrow$ B" means that "Problem A reduces to Problem B in polynomial time".

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. Similarly, the entries of a matrix $\mathbf{A}$ are denoted by A_{ij}. Random sampling from a set S is denoted by $a \xleftarrow{\$} S$.

The Matrix Code Equivalence problem. A *matrix code* is a subspace $\mathcal{C}$ of $m \times n$ matrices over $\mathbb{F}_q$ endowed with the rank metric defined as $d(\mathbf{A}, \mathbf{B}) = \mathrm{Rank}(\mathbf{A} - \mathbf{B})$. We denote by k the dimension of $\mathcal{C}$ as a subspace of $\mathbb{F}_q^{m \times n}$ and its basis by $\langle \mathbf{C_1}, \ldots, \mathbf{C_k} \rangle$, where $\mathbf{C_i} \in \mathbb{F}_q^{m \times n}$ are linearly independent. Due to symmetry, without loss of generality, in the rest of the text we will assume $n \geqslant m$.

The matrix code equivalence (MCE) problem is formally defined as follows:

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

One can also consider an easier variant of MCE, with $\mu : \mathbf{C} \mapsto \mathbf{ACB}$ and $\mathbf{A}$ (or $\mathbf{B}$) trivial. This is known as the *Matrix Codes Right (Left) Equivalence* problem (MCRE), and contains *Vector Rank Code Equivalence* as a sub-problem. The authors of [12] analyse both MCE and MCRE and show that MCE is at least as hard as the Monomial Equivalence problem for linear codes over $\mathbb{F}_q$, while MCRE can always be solved in probabilistic-polynomial time.

Although the authors of [12] don't give any algorithms or complexity estimates for MCE, it is easy to see that by brute-forcing either $\mathbf{A}$ or $\mathbf{B}$ and solving the other as an instance of MCRE, we get a complexity of $\mathcal{O}(q^{m^2})$ for MCE.

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

$$p_s(x_1, \ldots, x_N) = \sum_{1 \leqslant i \leqslant j \leqslant N} \gamma_{ij}^{(s)} x_i x_j + \sum_{i=1}^{N} \beta_i^{(s)} x_i + \alpha^{(s)}, \quad \gamma_{ij}^{(s)}, \beta_i^{(s)}, \alpha^{(s)} \in \mathbb{F}_q,\ 1 \leqslant s \leqslant k.$$

It is common to represent the homogeneous components of $\mathcal{P}$ as $\mathbf{x}\mathbf{P}^{(s)}\mathbf{x}^\top$, where $\mathbf{x} = (x_1, \ldots, x_N)$ and $\mathbf{P}^{(s)}$ is a $(N \times N)$ symmetric matrix describing the degree-2 homogeneous component of p_s. In odd characteristic, $\mathbf{P}_{ij}^{(s)} = \mathbf{P}_{ji}^{(s)} = \gamma_{ij}^{(s)}/2$ for $i \leqslant j$ and $\mathbf{P}_{ii}^{(s)} = \gamma_{ii}^{(s)}$, whereas in even characteristic, $\mathbf{P}_{ij}^{(s)} = \mathbf{P}_{ji}^{(s)} = \gamma_{ij}^{(s)}$ for $i \leqslant j$ and $\mathbf{P}_{ii}^{(s)} = 0$.

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 [16, 20]):

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

In this work we are mostly interested in the kernel of this linear map, as $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$ implies that $\mathcal{P}$ behaves linear at $(\mathbf{x} + \mathbf{a})$, as $\mathcal{P}(\mathbf{x} + \mathbf{a}) = \mathcal{P}(\mathbf{x}) + \mathcal{P}(\mathbf{a})$.

Isomorphism of polynomials. The Isomorphism of Polynomials (IP) problem (or Polynomial Equivalence (PE) [18]) was first defined by Patarin in [26] for the purpose of designing a "graph isomorphism"-like identification scheme and a digital signature using the Fiat-Shamir transform. It is defined as follows.
$\mathsf{IP}(N, k, \mathcal{F}, \mathcal{P})$:
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

$$\mathcal{P}(\mathbf{x}) = \mathcal{F}(\mathbf{x}\mathbf{S} + \mathbf{s})\mathbf{T} + \mathbf{t}. \tag{1}$$

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

The decisional version of IP is not $\mathcal{NP}$-complete [27], but it is known that even $\mathsf{IP1}\mathcal{S}$ is at least as difficult as the Graph isomorphism problem [27]. The IP problem has been investigated by several authors, initially for the security of the C^* scheme [27], but later also in [28, 29, 18, 9, 10]. In [18] an empirical argument was given 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, that was subsequently improved to $\mathcal{O}(N^9 q^{2N/3})$ in [10].

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.

Graph-Theoretic Algorithm for hQMLE. At Eurocrypt 2013 Bouillaguet et al. [10] 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 [18]). We briefly explain their method.

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.

3 How hard is MCE

In this section we show that the hQMLE and the MCE problem are equivalent.

Theorem 1. *The* MCE *problem is at least as hard as the* hQMLE *problem.*

Proof (Sketch). Let $(N,k,\mathcal{F},\mathcal{P})$, where $\mathcal{F}=(f_1,f_2,\dots,f_k)$, $\mathcal{P}=(p_1,p_2,\dots,p_k)\in\mathbb{F}_q[x_1,\dots,x_N]^k$ are k-tuples of quadratic homogenous polynomials, be an instance of hQMLE. We can efficiently represent it as an instance of the MCE problem as follows. Since the components p_s and f_s for every $s\in\{1,\dots,k\}$ of the mappings $\mathcal{P}$ and $\mathcal{F}$ are quadratic and homogenous, they can be represented as $N\times N$ symmetric matrices $\mathbf{P}^{(s)}$ and $\mathbf{F}^{(s)}$. Taking $(\mathbf{P}^{(1)},\dots,\mathbf{P}^{(k)})$ to be a basis of a matrix code $\mathcal{D}$ and $(\mathbf{F}^{(1)},\dots,\mathbf{F}^{(k)})$ a basis of a matrix code $\mathcal{C}$ we obtain an instance $(k,N,N,\mathcal{C},\mathcal{D})$ of MCE.

Now, let $(N,k,\mathcal{F},\mathcal{P})$ be a positive instance of hQMLE. This means that there exist $\mathbf{S}\in\mathrm{GL}_N(q)$, $\mathbf{T}\in\mathrm{GL}_k(q)$ such that $\mathcal{P}(\mathbf{x})=\mathcal{F}(\mathbf{xS})\mathbf{T}$. Since $\mathbf{T}$ is nonsingular, we can write this as $\mathcal{P}(\mathbf{x})\mathbf{T}^{-1}=\mathcal{F}(\mathbf{xS})$. This can be rewritten in a matrix form as $\sum\limits_{1\leqslant j\leqslant k}\widetilde{t}_{sj}\mathbf{P}^{(j)}=\mathbf{SF}^{(s)}\mathbf{S}^{\top},\ \ \forall s,1\leqslant s\leqslant k$ where $\widetilde{t}_{sj}$ is the (j,s) entry of the matrix $\mathbf{T}^{-1}$. Since $(\mathbf{P}^{(1)},\dots,\mathbf{P}^{(k)})$ is a basis of the code $\mathcal{D}$, any $\sum\limits_{1\leqslant j\leqslant k}\widetilde{t}_{sj}\mathbf{P}^{(j)}$ belongs to $\mathcal{D}$. Hence, $\mathbf{SF}^{(s)}\mathbf{S}^{\top}\in\mathcal{D},\ \ \forall s,1\leqslant s\leqslant k$. Since $(\mathbf{F}^{(1)},\dots,\mathbf{F}^{(k)})$ is a basis of the code $\mathcal{C}$ this means that $\mathbf{S}\mathcal{C}\mathbf{S}^{\top}=\mathcal{D}$ i.e. $(k,N,N,\mathcal{C},\mathcal{D})$ is a positive instance of MCE.

A similar argument in the opposite direction shows that a positive instance $(k,N,N,\mathcal{C},\mathcal{D})$ of MCE results in a positive instance $(N,k,\mathcal{F},\mathcal{P})$ of hQMLE. $\square$

Theorem 2. hQMLE *is at least as hard as* MCE*.*

Proof (Sketch). We proceed similarly as the other direction. Let $(k,n,m,\mathcal{C},\mathcal{D})$ be instance of MCE, where $\mathcal{C}$ and $\mathcal{D}$ are $m\times n$ matrix codes (of matrices from $M_{m,n}(q)$) of dimension k over $\mathbb{F}_q$. We can efficiently represent it as an instance of the hQMLE problem as follows. Let $(\mathbf{D}^{(1)},\dots,\mathbf{D}^{(k)})$ be a basis of the code $\mathcal{D}$ and $(\mathbf{C}^{(1)},\dots,\mathbf{C}^{(k)})$ a basis of the code $\mathcal{C}$. We make an important observation: There is a one-to-one correspondence between matrices from $M_{m,n}(q)$ and bilinear forms in variables $x_1,\dots,x_m,y_1,\dots,y_n$ over $\mathbb{F}_q$ given by: $\mathbf{M}\mapsto\sum\limits_{1\leqslant i\leqslant m,1\leqslant j\leqslant n}\mathbf{M}_{ij}x_iy_j$. This means that each of the matrices in the bases $(\mathbf{C}^{(1)},\dots,\mathbf{C}^{(k)})$ and $(\mathbf{D}^{(1)},\dots,\mathbf{D}^{(k)})$, can be seen as unique representations of bilinear forms. In particular, for the matrix $\mathbf{C}^{(s)}$ we construct the bilinear form $c_s(\mathbf{x},\mathbf{y})=\sum\limits_{1\leqslant i\leqslant m,1\leqslant j\leqslant n}\mathbf{C}^{(s)}_{ij}x_iy_j$ and for the matrix $\mathbf{D}^{(s)}$ the bilinear form $d_s(\mathbf{x},\mathbf{y})=\sum\limits_{1\leqslant i\leqslant m,1\leqslant j\leqslant n}\mathbf{D}^{(s)}_{ij}x_iy_j,\forall s,1\leqslant s\leqslant k$. Taking $\mathcal{F}=(c_1,c_2,\dots,c_k)$ and $\mathcal{P}=(d_1,d_2,\dots,d_k)$, and renaming variables as $\mathbf{z}=(\mathbf{x},\mathbf{y})=(x_1,\dots,x_m,y_1,\dots,y_n)$ we obtain an instance $(m+n,k,\mathcal{F},\mathcal{P})$ of hQMLE.

Now, let $(k,n,m,\mathcal{C},\mathcal{D})$ be a positive instance of MCE. This means, there exist matrices $\mathbf{A}\in\mathrm{GL}_m(q)$ and $\mathbf{B}\in\mathrm{GL}_n(q)$ such that $\mathcal{D}=\mathbf{A}\mathcal{C}\mathbf{B}$. In terms of the bases of $\mathcal{C}$ and $\mathcal{D}$, this means that there exist linearly independent vectors $(\widetilde{t}_{1j},\widetilde{t}_{2j},\dots,\widetilde{t}_{kj})$, for $1\leqslant j\leqslant k$ such that

$$\sum_{1\leqslant j\leqslant k}\widetilde{t}_{sj}\mathbf{D}^{(j)}=\mathbf{AC}^{(s)}\mathbf{B},\ \ \forall s,1\leqslant s\leqslant k. \tag{2}$$

Using the bilinear forms, this equation can be rewritten as

$$\sum_{1\leqslant j\leqslant k} \widetilde{t}_{sj} d_j(\mathbf{x},\mathbf{y}) = c_s(\mathbf{x}\mathbf{A}, \mathbf{y}\mathbf{B}^\top), \quad \forall s, 1 \leqslant s \leqslant k. \tag{3}$$

Taking $\mathbf{S} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^\top \end{bmatrix}$ and $\mathbf{z} = (\mathbf{x},\mathbf{y}) = (x_1,\ldots,x_m,y_1,\ldots,y_n)$, we get

$$\sum_{1\leqslant j\leqslant k} \widetilde{t}_{sj} d_j(\mathbf{z}) = c_s(\mathbf{z}\mathbf{S}), \quad \forall s, 1 \leqslant s \leqslant k.$$

This further rewrites to $\mathcal{P}(\mathbf{z}) = \mathcal{F}(\mathbf{z}\mathbf{S})\mathbf{T}$, where $\widetilde{t}_{sj}$ is the (j,s) entry of the matrix $\mathbf{T}^{-1}$, which means $(m+n,k,\mathcal{F},\mathcal{P})$ is a positive instance of hQMLE.

We can similarly show that a positive instance $(m+n,k,\mathcal{F},\mathcal{P})$ of hQMLE results in a positive instance $(k,n,m,\mathcal{C},\mathcal{D})$ of MCE. □

4 Solving MCE as QMLE

4.1 First algorithm

We start out with a description of a birthday-based algorithm for finding an isomorphism between two objects when a certain polynomial-time solver exists. This algorithm is just a generalization of the graph-based algorithm in [10] for solving hQMLE. 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 that 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 $\mathbb{P}$. We further assume the existance of an efficient algorithm FindFunction, that given $x \in S_1, y \in S_2$ returns ϕ if $y = \phi(x)$ and $\bot$ otherwise.

Algorithm 1 General Birthday-based Equivalence Finder

```
1: function SampleSet($S, \mathbb{P}$)
2:     $L \leftarrow \emptyset$
3:     repeat
4:         $a \xleftarrow{\$} S$
5:         if $\mathbb{P}(a)$ then $L \leftarrow L \cup \{a\}$
6:     until $|L| = \ell$
7:     return $L$

8: function CollisionFind($S_1, S_2$)
9:     $L_1 \leftarrow$ SampleSet($S_1, \mathbb{P}$)
10:    $L_2 \leftarrow$ SampleSet($S_2, \mathbb{P}$)
11:    for all $(a,b) \in L_1 \times L_2$ do
12:        $\phi \leftarrow$FindFunction($a, b$)
13:        if $\phi \neq \bot$ then
14:            return solution $\phi$
15:    return $\bot$
```

Lemma 1. *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 $d = N^{-1/3}$, for which the total complexity of the algorithm is $\mathcal{O}^*(N^{\frac{2}{3}})$.*

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 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 to build the required lists L_i of $\mathcal{O}(\sqrt{N/d})$.

The total running time is optimal when these two quantities $\sqrt{N/d}$ and $d \cdot N$ are equal, which holds when $d = N^{-1/3}$. Such a density gives complexity of $\mathcal{O}(N^{\frac{2}{3}})$ for SAMPLESET and at most $N^{\frac{2}{3}}$ queries to FINDFUNCTION. $\square$

In [10], the authors use a graph-theoretic approach to solve an instance of hQMLE. They define $G_{\mathcal{F}}$ (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. $D_{\mathbf{a}}\mathcal{P}(\mathbf{x}) = 0$). Their algorithm can be instantiated from Algorithm 1 by 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 polynomial-time solver from [18] for inhomogenous QMLE. Applying Lemma 1 gives exactly the result from [10, Thm. 2]:

Corollary 2 *Assuming a polynomial-time solver for the inhomogenous case of* QMLE*, an* hQMLE *instance* $(N, k, \mathcal{F}, \mathcal{P})$ *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* $\approx 63\%$.

This approach enables us to solve an MCE instance by reducing it to an instance of hQMLE. For an MCE instance $(k, n, m, \mathcal{C}, \mathcal{D})$, we get an hQMLE instance $(n+m, k, \mathcal{F}, \mathcal{P})$. By Corollary 2, this leads to a complexity of $\mathcal{O}^*(q^{\frac{2}{3}(n+m)})$, with a probability of $1 - 1/e \approx 63\%$ of success. To be more precise, 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. Theorem 2 from [16] can be used to show that the density d_κ does not depend on $\mathcal{F}$, but only on the universe $\mathcal{M}_{k,n+m}(q)$. The optimal κ given in Lemma 1, i.e. such that $d_\kappa = q^{-(n+m)/3}$, is harder to compute than previously, but note that, interestingly enough, this does not affect the complexity of Algorithm 1. To compute the optimal κ, we can use the following lemma, taking $a = k$ and $b = n + m$.

Lemma 3. *Define the predicate* $\mathbb{P}_\kappa : \dim\ker \mathbf{M} = \kappa$ *for* $\mathbf{M} \in U = \mathcal{M}_{a,b}(q)$. *Then the density of the predicate* $\mathbb{P}_\kappa$ *is* $d_\kappa \approx q^{-(\kappa^2 + \kappa\cdot(a-b))}$. $\square$

As we want $d_\kappa \approx q^{-(n+m)/3}$, we need $q^{\kappa^2+\kappa\cdot(k-(n+m))} = q^{(n+m)/3}$, i.e.

$$\kappa = \delta + \sqrt{\delta^2 + \frac{n+m}{3}}, \quad \delta = \frac{(n+m)-k}{2}. \tag{4}$$

which resolves to $\kappa = \sqrt{\frac{n+m}{3}}$ when $k = n + m$.

In conclusion, given κ as above, we get our first result on the hardness of MCE, compared to a straightforward enumeration.

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

4.2 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, as it satisfies $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}(\mathbf{x},\mathbf{y}) = \mathcal{F}(\mathbf{a},\mathbf{y}) + \mathcal{F}(\mathbf{x},\mathbf{b})$. We denote by $\mathbf{F_a}$ the $k \times n$ matrix of the linear map $\mathcal{F}(\mathbf{a},-) : \mathbb{F}_q^n \to \mathbb{F}_q^k$ and by $\mathbf{F_b}$ the $k \times m$ matrix of $\mathcal{F}(-,\mathbf{b}) : \mathbb{F}_q^m \to \mathbb{F}_q^k$. Similarly for $\mathcal{P}$, we have $\mathbf{P_a}$ and $\mathbf{P_b}$. We get

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

When the polynomials are bilinear, one can see SampleSet in Algorithm 1 as sampling both $\mathbf{a} \in \mathbb{F}_q^n, \mathbf{b} \in \mathbb{F}_q^m$ at the same time, until $D_{(\mathbf{a},\mathbf{b})}\mathcal{F}$ has a kernel of size 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 and use predicates in these much smaller universes to find collisions much faster.

We denote by $\mathfrak{F}_a$ the set of elements $\mathbf{a}$ for which $\dim\ker\mathbf{F_a}$ is non-trivial, i.e. $\mathfrak{F}_a = \{\mathbf{a} \in \mathbb{F}_q^n \mid \dim\ker\mathbf{F_a} > 0\}$, and we define $\mathfrak{F}_b$, $\mathfrak{P}_a$ and $\mathfrak{P}_b$ similarly.

Lemma 4. *Let $\mu : \mathcal{F} \to \mathcal{P}$ be an isomorphism between two k-tuples of bilinear homogenous quadratic polynomials $\mathcal{F}$ and $\mathcal{P}$. We have the following properties:*

1. *For $\mathbf{a} \in \mathfrak{F}_a$, $(\mathbf{a},\mathbf{b})$ is a root of $\mathcal{F}$ for all $\mathbf{b} \in \ker\mathbf{F_a}$.*
2. *$\mathfrak{F}_b = \bigcup_{\mathbf{a}\in\mathfrak{F}_a} \ker\mathbf{F_a}$.*
3. *The isomorphism μ bijectively maps $\mathfrak{F}_a \to \mathfrak{P}_a$ and $\mathfrak{F}_b \to \mathfrak{P}_b$.*
4. *When $n = m$ and $k \geqslant n$, $|\mathfrak{F}_a| = |\mathfrak{P}_a| \approx |\mathfrak{F}_b| = |\mathfrak{P}_b| \approx q^{2n-k-1}$.*

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.: $\mathbf{b} \in \mathfrak{F}_b$ means that there exists an $\mathbf{a} \in \mathfrak{F}_a$ such that $\mathcal{F}(\mathbf{a},\mathbf{b}) = \mathbf{0}$, which is the same as $\mathbf{b} \in \ker\mathbf{F_a}$ for this specific $\mathbf{a}$.
3. Recall first that the isomorphism μ implies existence of matrices $\mathbf{A},\mathbf{B},\mathbf{T}$ such that $\mathcal{P}(\mathbf{x},\mathbf{y}) = \mathcal{F}(\mathbf{x}\mathbf{A},\mathbf{y}\mathbf{B}^\top)\mathbf{T}$.
   Let $\mathbf{a} \in \mathfrak{F}_a$. Then $(\mathbf{a},\mathbf{b})$ is a root of $\mathcal{F}$ for all $\mathbf{b} \in \ker\mathbf{F_a}$. From the above, $(\mathbf{a}',\mathbf{b}') = (\mathbf{a}\mathbf{A}^{-1}, \mathbf{b}(\mathbf{B}^\top)^{-1})$ is a root of $\mathcal{P}$ for all $\mathbf{b} \in \ker\mathbf{F_a}$, i.e. for all $\mathbf{b}' \in \ker\mathbf{F}_{\mathbf{a}'\mathbf{A}}\mathbf{B} = \ker\mathbf{P}_{\mathbf{a}'}$. Hence, $\mathbf{a}' \in \mathfrak{P}_a$. The isomorphism maps $\mathbf{a}$ to $\mathbf{a}'$, hence $\mathfrak{F}_a$ to $\mathfrak{P}_a$. A similar argument gives $\mathfrak{F}_b \to \mathfrak{P}_b$.
4. Note that by 3 we get $|\mathfrak{F}_a| = |\mathfrak{P}_a|$. Now, from Lemma 3 we see that the size of these sets is dominated by elements with $\kappa = 1$ (a one-dimensional kernel). From the same lemma, the density of $\kappa = 1$ elements is $d_1 = q^{-(1+1\cdot(k-n))}$. Hence we expect $d_1 \cdot q^n = q^{2n-k-1}$ such elements. □

Using Lemma 4, we can find collisions with full certainty whenever $\mathcal{F}$ and $\mathcal{P}$ have non-trivial roots. By item 4, assuming $n = m$ as the hardest case, this happens when $k < n+m$ with high probability, when $k = n+m$ with probability $\frac{1}{q}$, and when $k > n+m$ almost never. From this, the resulting algorithm is simple: we compute the roots of $\mathcal{F}$ and $\mathcal{P}$ by computing $\ker \mathbf{F_b}$ for all $\mathbf{b} \in \mathbb{F}_q^m$ to find non-trivial $\ker \mathbf{F_b}$. This gives us sets $\mathfrak{F}_a \times \mathfrak{F}_b$ and $\mathfrak{P}_a \times \mathfrak{P}_b$. If they are non-empty, we feed them to the solver for a guaranteed collision. Otherwise, we get no speed-up to solve MCE compared to Algorithm 1. This happens when $k > n + m$. Furthermore, 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. In summary, we have Algorithm 2.

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

```
 1: function SampleZeroes($\mathcal{F}$)
 2:     $S, S^{\mathbf{0}}, S_a, S_b \leftarrow \emptyset$
 3:     for all $\mathbf{b} \in \mathbb{F}_q^m$ do
 4:         if $\dim \ker \mathbf{F_b} > 0$ then
 5:             $S_b \leftarrow S_b \cup \{\mathbf{b}\}$
 6:             $S_a \leftarrow S_a \cup \ker \mathbf{F_b} \setminus \{0\}$
 7:     $S \leftarrow S_a \times S_b$
 8:     for all $\mathbf{x} \in S$ do
 9:         if $\mathcal{F}(\mathbf{x}) = \mathbf{0}$ then
10:             $S^{\mathbf{0}} \leftarrow S^{\mathbf{0}} \cup \{\mathbf{x}\}$
11:     $S \leftarrow S \setminus S^{\mathbf{0}}$
12:     return $S, S^{\mathbf{0}}$
13: function CollisionFind($\mathcal{F}, \mathcal{P}$)
14:     $\mathfrak{F}, \mathfrak{F}^{\mathbf{0}} \leftarrow$ SampleZeroes($\mathcal{F}$)
15:     $\mathfrak{P}, \mathfrak{P}^{\mathbf{0}} \leftarrow$ SampleZeroes($\mathcal{P}$)
16:     for all $(\mathbf{x}, \mathbf{y}) \in \mathfrak{F} \times \mathfrak{P}$ do
17:         $\mu \leftarrow$FindFunction($\mathbf{x}, \mathbf{y}$)
18:         if $\mu \neq \perp$ then
19:             return solution $\mu$
20:     for all $(\mathbf{x}, \mathbf{y}) \in \mathfrak{F}^{\mathbf{0}} \times \mathfrak{P}^{\mathbf{0}}$ do
21:         $\mu \leftarrow$FindFunction($\mathbf{x}, \mathbf{y}$)
22:         if $\mu \neq \perp$ then
23:             return solution $\mu$
24:     return $\perp$
```

Note that we have certainty of collision, hence we can take a single $\mathbf{x} \in \mathfrak{F}$ (or equivalently $\mathfrak{F}^{\mathbf{0}}$) and compare it against every $\mathbf{y} \in \mathfrak{P}$ (or $\mathfrak{P}^{\mathbf{0}}$) to decrease the number of queries to FindFunction. For technical reasons, our inhomogeneous solver slightly prefers $(\mathbf{x}, \mathbf{y}) \in \mathfrak{F} \times \mathfrak{P}$ to $(\mathbf{x}, \mathbf{y}) \in \mathfrak{F}^{\mathbf{0}} \times \mathfrak{P}^{\mathbf{0}}$. For parameters where k is close to $n + m$, both sets are small, so the number of queries is limited. For parameters where $\mathfrak{F}$ and $\mathfrak{P}$ become too large, one can decrease the number of queries by switching to $\mathfrak{F}^{\mathbf{0}} \times \mathfrak{P}^{\mathbf{0}}$ or by applying other (secondary) predicates. This ensures the number of queries never exceeds q^m, and there is reason to assume the number of queries can be made much lower. The next theorem summarises this subsection on using bilinearity to solve MCE.

Theorem 4. *Assuming a polynomial-time solver for the inhomogenous case of* QMLE*, an* MCE *instance* $(k, n, m, \mathcal{F}, \mathcal{P})$ *over* $\mathbb{F}_q$ *with* $n \geqslant m$ *and roots existing for* $\mathcal{F}$ *and* $\mathcal{P}$*, can be solved using Algorithm 2 with* $\mathcal{O}(q^m)$ *operations in* SampleZeroes *and at most* q^m *queries to the inhomogenous solver. For* $m \leqslant k \leqslant n + m$ *this amounts to a total complexity of* $\mathcal{O}^*(q^m)$ *for solving the* MCE *instance* $(k, n, m, \mathcal{F}, \mathcal{P})$.

4.3 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 implemented 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 hQMLE. (2) We build the two sample sets for a predefined predicate $\mathbb{P}_\kappa$ and we combine them to create pairs of potential collisions. This step is performed using the open source implementation from [10]. (3) For each pair we create an inhQMLE instance and we query an inhQMLE solver until it outputs a solution for the maps $\mathbf{S}$ and $\mathbf{T}$. Our inhomogenous solver consists of reducing $\mathcal{P}(\mathbf{x})\mathbf{T}^{-1} = \mathcal{F}(\mathbf{xS})$, with $\mathbf{S}$ and $\mathbf{T}$ unknown, to a system of polynomial equations and solving the resulting system using the F4 [17] implementation in MAGMA [8]. The results of this experiment using $\kappa = 5$ are shown in Table 1.

Table1. Experimental results on solving the MCE problem using Algorithm 1.

$m=n$	k	Sample set size	Runtime (s) SampleSet	Runtime (s) inhQMLE solver	Success probability
10	20	2	22	53	0.69
11	22	4	47	180	0.79
12	24	7	76	903	0.71

The second approach, described in Section 4.2, differs from the first one only in step (2) above. The bilinear structure of hQMLE instances derived from MCE instances can be exploited 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 2. Recall that, this approach can be used only when there exist roots of $\mathcal{F}$ and $\mathcal{P}$. Otherwise, the sampled sets are empty and the instance is solved using Algorithm 1. Table 2 shows results of the case when the sets are not empty and the probability of this case for the given parameters is shown in the last column.

Table2. Experimental results on solving the MCE problem using Algorithm 2.

$m=n$	k	Sample set size	Runtime (s) SampleZeros	Runtime (s) inhQMLE solver	% instances with roots
11	22	2.06	0.15	104	59
	21	4.64	0.16	87	86
	20	13.06	0.16	93	99
	19	64.02	0.15	171	100
12	24	1.95	0.32	230	63
	23	5.19	0.27	211	85
	22	17.00	0.29	235	99
	21	54.89	0.30	349	100

All experiments are for the case of $q = 2$ and all results are an average of 100 runs.

Acknowledgements. The authors thank Charles Bouillaguet for providing the implementation resulting from [10].

Bibliography

[1] N. Aragon, O. Blazy, J.-C. Deneuville, P. Gaborit, A. Hauteville, O. Ruatta, J.-P. Tillich, G. Zemor, C. A. Melchor, S. Bettaieb, L. Bidoux, M. Bardet, and A. Otmani. ROLLO (Rank-Ouroboros, LAKE and LOCKER), 2019.

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

[3] E. Bellini, F. Caullery, P. Gaborit, M. Manzano, and V. Mateu. Improved Veron Identification and Signature Schemes in the Rank Metric. *2019 IEEE International Symposium on Information Theory (ISIT)*, pages 1872–1876, 2019.

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

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

[6] W. Beullens and B. Preneel. Field Lifting for Smaller UOV Public Keys. In A. Patra and N. P. Smart, editors, *Progress in Cryptology – INDOCRYPT 2017*, pages 227–246, Cham, 2017. Springer International Publishing.

[7] J.-F. Biasse, G. Micheli, E. Persichetti, and P. Santini. LESS is More: Code-Based Signatures Without Syndromes. In A. Nitaj and A. Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020*, pages 45–65, Cham, 2020. Springer International Publishing.

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

[9] C. Bouillaguet, J.-C. Faugère, P. Fouque, and L. Perret. Practical Cryptanalysis of the Identification Scheme Based on the Isomorphism of Polynomial With One Secret Problem. In *Public Key Cryptography – PKC 2011*, volume 6571 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2011.

[10] C. Bouillaguet, P.-A. Fouque, and A. Véber. Graph-theoretic algorithms for the "Isomorphism of Polynomials" problem. pages 211–227, 2013.

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

[12] A. Couvreur, T. Debris-Alazard, and P. Gaborit. On the hardness of code equivalence problems in rank metric, 2021.

[13] L. De Feo and S. D. Galbraith. SeaSign: Compact Isogeny Signatures from Class Group Actions. In Y. Ishai and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 759–789, Cham, 2019. Springer International Publishing.

[14] L. De Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski. SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In S. Moriai and H. Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 64–93, Cham, 2020. Springer International Publishing.

[15] J. Ding and D. Schmidt. Rainbow, a New Multivariable Polynomial Signature Scheme. In J. Ioannidis, A. D. Keromytis, and M. Yung, editors, *ACNS*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175, 2005.

[16] V. Dubois, L. Granboulan, and J. Stern. An efficient provable distinguisher for HFE. In *International Colloquium on Automata, Languages, and Programming*, pages 156–167. Springer, 2006.

[17] J.-C. Faugère. A New Efficient Algorithm for Computing Gröbner basis (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.

[18] J.-C. Faugère and L. Perret. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In S. Vaudenay, editor, *EUROCRYPT '06*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer, 2006.

[19] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[20] P.-A. Fouque, L. Granboulan, and J. Stern. Differential Cryptanalysis for Multivariate Schemes. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 341–353. Springer Berlin Heidelberg, 2005.

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

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

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

[24] C. A. Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, G. Zemor, A. Couvreur, and A. Hauteville. RQC, 2019.

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

[26] J. Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996.

[27] J. Patarin, L. Goubin, and N. Courtois. Improved Algorithms for Isomorphisms of Polynomials. In *EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, page 184–200. Springer, 1998.

[28] L. Perret. On the computational complexity of some equivalence problems of polynomial systems of equations over finite fields. *Electronic Colloquium on Computational Complexity (ECCC)*, (116), 2004.

[29] L. Perret. 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*, volume 3494 of *Lecture Notes in Computer Science*, pages 354–370. Springer, 2005.

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