Graph-Theoretic Algorithms for the "Isomorphism of Polynomials" Problem

Charles Bouillaguet@Univ-Lille1[1], Pierre-Alain Fouque[2] and Amandine Véber[3]

[1] University of Lille-1
charles.bouillaguet@univ-lille1.fr
[2] University of Rennes-1
pierre-alain.fouque@univ-rennes1.fr
[3] CMAP Lab, CNRS and Ecole Polytechnique
amandine.veber@cmap.polytechnique.fr

Abstract. We give three new algorithms to solve the "isomorphism of polynomial" problem, which was underlying the hardness of recovering the secret-key in some multivariate trapdoor one-way functions. In this problem, the adversary is given two quadratic functions, with the promise that they are equal up to linear changes of coordinates. Her objective is to compute these changes of coordinates, a task which is known to be harder than Graph-Isomorphism. Our new algorithm build on previous work in a novel way. Exploiting the birthday paradox, we break instances of the problem in time $q^{2n/3}$ (rigorously) and $q^{n/2}$ (heuristically), where q^n is the time needed to invert the quadratic trapdoor function by exhaustive search. These results are obtained by turning the algebraic problem into a combinatorial one, namely that of recovering partial information on an isomorphism between two exponentially large graphs. These graphs, derived from the quadratic functions, are new tools in multivariate cryptanalysis.

1 Introduction

The notion of *equivalent linear maps* is a basic concept in linear algebra; two linear functions f and g over vector spaces are equivalent if and only if there exist two other linear bijective functions S and T such that $f = T \circ g \circ S$. Geometrically speaking, this means that f and g are essentially the same function, but with coordinates expressed in different bases. The computational problem consisting in testing the equivalence of two linear functions (given by matrices) is easy, because it is well-known that two linear maps are equivalent if and only if they have the same rank.

This notion of equivalent linear maps lends itself to an obvious generalization, by dropping the requirement that the functions shall be linear. Then, given two vector spaces U and V, of respective dimension n and m, two functions $f, g : U \to V$ are said to be equivalent if there exist an invertible $n \times n$ matrix S and an invertible $m \times m$ matrix T such that $g = T \circ f \circ S$. Again, the geometric interpretation of this notion is that g and f are "the same function", up to linear

changes of coordinates. However, deciding the equivalence of two such functions is no longer easy in general.

The case where f and g are polynomial maps is particularly relevant, not only because it is a natural generalization of the linear case, but also because f and g admit a compact representation. It is understood that a polynomial map f is such that each coordinate of the vector $f(x)$ is a polynomial expression of the coordinates of the vector x. Testing the equivalence of two polynomial maps has been called the "Isomorphism of Polynomials" (IP) problem by Patarin in 1996 [28], and later the "Polynomial Linear Equivalence" (PLE) problem by Faugère et al. in 2006 [19].

One aspect of PLE that makes it a bit difficult to study is that depending on the parameters (dimensions and base field of the vector spaces, degree of the polynomials, special restrictions, etc.), the problem can take very different forms. We will thus focus on the case where the base field of the vector space is finite (of size q), where polynomials are quadratic, and where their domain and codomain are the same, *i.e.*, where $f, g : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^n$ are quadratic maps. This is the setting that appears in most cryptographic constructions. In the sequel we will call this particular restriction the Quadratic Maps Linear Equivalence (QMLE) problem. In order to make our exposition simpler, we will furthermore assume that q, the size of the finite field, is a power of two. The theory of quadratic forms presents itself very differently for odd characteristic and for characteristic two, and in order not to expose two variants of each of our results, we chose the most computer-oriented setting.

The first "multivariate" cryptographic schemes relied on a somewhat heuristic construction to build Trapdoor One-Way Functions, whose security was based on the hardness of QMLE. Starting with an easy-to-invert quadratic map f, one builds an apparently random-looking one by setting $g = T \circ f \circ S$. The idea is that the changes of coordinate would hide the structure of f that makes it easy to invert, so that g would look random. Inverting random quadratic maps is extremely hard, and the best options in general are exhaustive search (if q is small), or the computation of a Groebner basis (when q is large), both techniques being exponential in n. This construction backed one of the advertized goals of multivariate cryptography, namely the ability to encrypt or sign n-bit blocks while offering n bits of security, as opposed to, *e.g.* RSA.

In this setting, g (and eventually f) is the public key, while S and T are the secret key. When f is public, then recovering the secret-key precisely means solving an instance of QMLE. Several cryptosystems have been built on this idea [6, 29, 41, 13], but they have *all* been broken [22, 18, 5, 26, 21]. The main reason behind this fiasco is that the specific instances of QMLE exposed by these schemes were weak because f was too special, so that polynomial-time and/or efficient algorithms to crack them have eventually been designed.

In a different direction, Patarin also proposed to use the hardness of *arbitrarily chosen* instances of the PLE problem to design a public-key identification scheme, thus potentially avoiding the aforementioned disaster. A *prover*, who has generated a pair of private/public keys (PK, SK), wants to prove her iden-

tity to a *verifier* who knows PK. In fact the prover aims to convince that she knows SK, but without revealing any information about SK to the verifier, or to anybody else. In 1986, Goldreich, Micali and Wigderson [24] built an elegant zero-knowledge proof system for Graph Isomorphism (GI) and used it to build an identification scheme. There, PK is a pair of isomorphic graphs, and SK is the isomorphism (a permutation of the vertices). In order for this system to be secure, it must be hard to solve the instance of GI formed by the public-key. Despite a large research effort, until now no algorithm has been able to solve instances of GI in worst-case polynomial, which is certainly encouraging. However, most instances of GI, and in particular random instances, are *extremely easy* to solve. Thus, the identification scheme of [24] relied on a presumably hard problem for which we do not know how to generate non-trivial instances...

Patarin's suggestion was that Graph Isomorphism could be replaced by QMLE, with the hope that *random instances* of the problem would then be hard, and that key-generation would then be straightforward. There was apparently nothing to lose with the new problem, because it was shown to be harder than GI [30]. Using random instances would in principle avoid the weak instances that had been broken. The resulting QMLE-based identification scheme is not particularly efficient, and does not enjoy very attractive key-sizes, but it is quite simple. It also has a few interesting features compared to other identification schemes based on NP-hard combinatorial problems such as [33–38]: most notably, it does not require hash functions nor commitment schemes, and it does not require the parties to share a (usually large) public common string describing an instance of the NP-complete problem.

1.1 Related Work

The QMLE problem is reminiscent of the Even-Mansour cipher [17], which turns a fixed n-bit permutation P into an n-bit block-cipher with $2n$-bit key by setting $E_{k_1,k_2}(x) = P(x + k_1) + k_2$. Attacks against this construction aim to recover the keys while only having black-box access to E and P. One of its distinctive features is that the performance of a successful adversary running in time t and sending q queries is limited by $t \cdot q \geq 2^n$, under the assumption that P is a random permutation. The known attacks match this bound [12, 16]. As mentioned above, the hardness of QMLE would allow a similar construction where a fixed and public quadratic permutation P is turned into a public-key encryption primitive $E_{S,T} = T \circ P \circ S$. In this context, adversaries not only have oracle to E and P, but know their full description.

Essentially two non-trivial algorithms have been proposed so far for QMLE: the "To-and-Fro" approach [30] on the one hand, and the "Groebner Basis" approach [19] on the other hand. There are also several, more efficient algorithms for the special case where the secret T matrix is known to be the identity matrix [23, 32, 9, 27]. This sub-problem is also GI-hard, even in very restricted settings [1]. The article [2] considers the particular case of testing whether two boolean functions are equal modulo a permutation of their inputs. It shows that $2^{n/2}$ queries are necessary if one only has black-box access to the boolean functions.

Back to the full QMLE problem, the "To-and-Fro" algorithm, while being simple, was exposed on a toy example, without pseudo-code nor detailed analyzis. We are convinced that the algorithm works when the polynomial maps f and g are *bijective*, but it cannot work as-is when they are not (the authors of [19] made the same observation). Note that a random polynomial map is not bijective with overwhelming probability. As is it given in [30], the "to-and-fro" algorithm is thus not applicable to random instances of QMLE. We found out that it *is* nevertheless possible to adapt the algorithm to work in the non-bijective case, but there are several ways to do so, and some are more efficient than others. Figuring that out required some work, and exposing it requires some space, so we will not go deeper into this issue in this paper. In any case, the authors of [30] claim that the complexity of their algorithm is of order $\mathcal{O}\left(q^{2n}\right)$ when $q > 2$ and $\mathcal{O}\left(2^{3n}\right)$ when $q = 2$, and we agree with them. The algorithm was later independently rediscovered under the form of a procedure to test the linear equivalence of S-boxes [7].

The "Groebner basis" algorithm, on the other hand is not heuristic, and is well-specified. It consists in identifying coefficient-wise the equation $T^{-1} \circ g = f \circ S$, which relates two vectors of n quadratic forms. It is therefore equivalent to about n^3 quadratic equations in the $2n^2$ coefficients of the unknown changes of coordinates. These equations are then solved through the computation of a Groebner basis. The complexity of Groebner basis algorithms is notoriously tricky to study, and the authors of [19] did not give any definitive results. However, they empirically observed an important fact, namely that when f and g are *inhomogeneous* quadratic maps, *i.e.*, when f and g contains non-zero linear and constant terms, then their algorithm terminated in polynomial time $\mathcal{O}\left(n^9\right)$. In the homogeneous case, the authors of [19] conjectured that their algorithm is subexponential, without providing any argument nor any evidence that it is the case. This assertion is impossible to verify in practice because the complexities are too high, but our own reasoning makes us more inclined to believe that the algorithm is plainly exponential. Assuming that the equations form a semi-regular sequence would allow to estimate the complexity of the Groebner basis computation [4]; doing so results in a total complexity of $\mathcal{O}\left(2^{18n}\right)$, yet assuming that the equations are semi-regular is probably a bit of a stretch. Establishing the complexity of this algorithm is thus essentially an open problem.

In the sequel, we will nevertheless take for granted that inhomogeneous instances of QMLE are tractable and can be solved in polynomial time, using the "Groebner-based" algorithm for instance.

It must be noted that in [30], the existence of an algorithm based on the birthday paradox and running in time $\mathcal{O}\left(q^{n/2}\right)$ is asserted, and that this algorithm is itself partially described in [31], where it is called the "combined powers attack". This algorithm is sometimes acknowledged for in the literature (*e.g.* in [19]). However, it is underspecified to the point that it is impossible to implement it, and some of the bits that are specified have major problems. Some of them deterministically fail to meet their goal, and the whole construction relies

on heuristic assumptions that are empirically false (sometimes provably). This "algorithm" should thus be disregarded.

1.2 Our Results

We give three algorithms to solve QMLE in the homogeneous case. All these algorithms work by reducing the solution of a homogeneous (hard) instance into that of one or several inhomogeneous (easy) instances after some preprocessing. We will thus assume that we are given a (black-box) *Inhomogeneous solver* that presumably works in polynomial time, and we will count the number of *inhomogeneous queries* sent to this oracle. We are well-aware that this assumption is quite strong. The empirical success of the algorithm of [19] convinced us that it works in polynomial-time on average, yet moving from there to "worst-case polynomial time" seems like a leap of faith. However, this assumption eases our exposition considerably, and in practice there does not seem to be any problem (probably because the queries sent to the inhomogeneous oracle are random enough).

Our three algorithms differ by the number of queries they send to the oracle, by the amount of computation they perform themselves, and by their success probability.

Algo.	Preprocessing	Inhom. queries	success prob.	
1		q^n	1	
2	$\mathcal{O}\left(n^3 \cdot q^{2n/3}\right)$	$q^{2n/3}$	62%	
3	$\mathcal{O}\left(n^5 \cdot q^{n/2}\right)$	1	62 %	only when $q = 2$

Algorithm 1 is deterministic, and essentially performs an exhaustive search in $(\mathbb{F}_q)^n$, sending one inhomogeneous query per vector. Using the algorithm of [19] to deal with the inhomogeneous instances, the resulting complexity is $\mathcal{O}\left(n^9 \cdot q^n\right)$, which already improves on the "to-and-fro" algorithm of [30].

Algorithms 2 and 3 rely on the birthday paradox to improve on exhaustive search and break the q^n barrier. To this end, two exponentially large isomorphic graphs are derived from the two quadratic maps. Recovering a bit of information on an isomorphism allows to make the problem inhomogeneous, and thus easy to solve. The trick is that this partial information must be extracted without knowing the full graphs, because they are too large. The construction of these graphs borrows from the differential techniques that have broken SFLASH, amongst others.

Algorithm 2 is relatively easy to analyze and we rigorously establish its complexity and success probability when dealing with random instances of the problem. Algorithm 3 is more efficient but more sophisticated and harder to analyze (as well as somewhat heuristic). We provide an as-rigorous-as-possible complexity analysis under a conjecture on random quadratic maps, and we verify experimentally that we are not off by too much.

Because our algorithms are exponential in n, we do not fully break Patarin's identification scheme (it is of no practical value anyway), even though its key-sizes should in principle be doubled. The construction of a Trapdoor One-Way

Function from QMLE outlined above has already been bludgeoned to death by cryptanalysts, and it now lies on the autopsy table. We take the role of the medical examiner that appears in every good police drama, only to discover that the corpse had a fatal disease even before being brutally assaulted. We indeed believe that our algorithms condemn this generic construction of a Trapdoor One-Way Function *post-mortem*, and give a theoretical reason not to try again, besides the obvious "they have all been broken" argument. Our algorithms indeed break the QMLE instance and retrieve the secret-key (asymptotically) much faster than inverting the quadratic map by exhaustive search. This shows in passing that this construction can only offer $n/2$ bits of security, instead of the n that was its original objective.

2 A First Algorithm Based on Dehomogenization

Confronted with a *homogeneous* instance of QMLE, our strategy throughout this paper is to build an *inhomogeneous instance* admitting the exact same solutions. This inhomogeneous instance can in turn be solved in polynomial time, and reveals the solution(s) of the original problem. The downside of this approach is that the image of S must be known at one arbitrary point of the vector space. Indeed, if $\beta = S \cdot \alpha$, then:

$$\forall x.\ g(x) = T \cdot f(S \cdot x) \qquad \iff \qquad \forall x.\ g(x+\alpha) = T \cdot f(S \cdot x + \beta).$$

Thus defining $g'(x) = g(x+\alpha)$ and $f' = f(x+\beta)$ yields an equivalent problem, *i.e.*, an instance that has the same solutions as the original one. In addition, the new instance is inhomogeneous. This follows from the simple observation that although x^2 is a homogeneous polynomial, $(x+\alpha)^2 = x^2 + 2\alpha x + \alpha^2$ is not since it has a non-trivial linear term αx and a non-trivial constant term α^2.

It follows that solving (homogeneous) instances of QMLE essentially boils down to finding $S\alpha$, for some known and non-zero vector α. Exhaustive search is the first option that comes to mind, leading to Algorithm 1. This algorithm sends q^n queries to the inhomogeneous solver in the worst case, and finds the solutions when they exist. This algorithm terminates with probability one in time $\mathcal{O}\left(n^9 \cdot q^n\right)$ if the Groebner-based algorithm of [19] is used to solve the inhomogeneous instances. Despite being extremely simple, Algorithm 1 is asymptotically q^n times faster than to the "to-and-fro" algorithm of [30].

This *dehomogenization* technique exposes a crucial asymmetry in the problem: it is apparently much more critical to obtain knowledge on S than on T. This is not new: the "To-and-Fro" algorithm relies on the ability to transfer knowledge of a relation $\beta = S \cdot \alpha$ to a relation $g(\alpha) = T \cdot f(\beta)$.

3 Moving the Problem Into a Graphic World

Using the birthday paradox is a natural idea to improve on exhaustive search algorithms in many scenarii, with the hope to halve the exponent in the complexity. Here, we wish to use the birthday paradox to obtain the image of S at

Algorithm 1 Simple algorithm based on dehomogenization.

function EXHAUSTIVE-DEHOMOGENIZATION(f, g)
  $x \leftarrow$ random non-zero vector in $(\mathbb{F}_q)^n$
  for all $0 \neq y \in (\mathbb{F}_q)^n$ **do**
    $f'(z) \leftarrow f(z + y)$
    $g'(z) \leftarrow g(z + x)$
    query IQMLE-SOLVER with (f', g'')
    if solution (S, T) found **then return** (S, T)
  return "Not Equivalent"

one point, and build a dehomogenized instance, just as we did in the previous section. One difficulty is that we want to focus only on S, and leave T alone. To this end, we introduce a tool which is, to the best of our knowledge, new. We associate a *graph* G_h to any quadratic map $h : (\mathbb{F}_q)^n \mapsto (\mathbb{F}_q)^n$. Its vertices are the elements of $(\mathbb{F}_q)^n$, and there is an edge between $x, y \in (\mathbb{F}_q)^n$ if and only if $h(x + y) = h(x) + h(y)$. To some extent, G_h expresses the "linear behavior" of h (even though h is not linear) and thus we call these graphs the "linearity graphs" of the associated quadratic maps.

These graphs are natural objects associated to quadratic maps. For instance, the distinguisher of [15] to determine whether a given quadratic map f is an HFE public key can be rephrased as follows: pick a random node in G_f, and count its neighbors. If their number exceeds a given bound (which depends on the degree of the internal HFE polynomial), then return "random", else return "HFE". With the right bound on the number of neighbors, this algorithm achieves subexponential advantage.

The essential interest of linearity graphs for our purposes is that the two graphs G_f and G_g are connected by the secret matrix S.

Lemma 1. *If $T \circ g = f \circ S$ then S is a graph isomorphism that sends G_f to G_g.*

Proof. Indeed, if $x \leftrightarrow y$ in G_g, then by definition $g(x + y) = g(x) + g(y)$, and it follows that $T \circ g(x + y) = T \circ g(x) + T \circ g(y)$, and thus that $f(S \cdot x + S \cdot y) = f(S \cdot x) + f(S \cdot y)$. This in turn means that $S \cdot x \leftrightarrow S \cdot y$ in G_f. It follows that S is a graph isomorphism between G_f and G_g.

Linearity graphs thus allows a formulation of the problem where the other secret matrix T is no longer present. We have two (exponentially large) isomorphic graphs G_f and G_g, and we ultimately need to recover the whole isomorphism S. However, thanks to the dehomogenization technique of the previous section, and thanks to the ease with which inhomogeneous instances can be solved, it turns out that recovering just a little bit of information on the isomorphism is enough to find it completely. More precisely, we just need to know how the isomorphism S transforms one arbitrary vertex.

Of course, completely building these graphs is prohibitively expensive (they have q^n vertices). It turns out that this is never necessary, because it is possible to walk in these graphs without fully knowing them.

Walking in Linearity Graphs. The function $\psi(x,y) = f(x+y)+f(x)+f(y)$ is a generalization of the *polar form* of a quadratic form to vectors thereof, in characteristic two. It is easy to check that ψ is bilinear. Given a (non-zero) vertex $x \in (\mathbb{F}_q)^n$ in the graph, the function:

$$\mathrm{D}_x f : y \mapsto \psi(x,y) = f(x+y) + f(x) + f(y)$$

is a familiar object in multivariate cryptology, called the *Differential of f at x* [20, 15, 14, 22]. It is a linear function from $(\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^n$, which is then conveniently represented by a matrix. The set of nodes adjacent to x in G_f is in fact the kernel of $\mathrm{D}_x f$. Note that x always belong to $\ker \mathrm{D}_x f$, because $x + x = 0$. The main reason we chose to focus on the case where $q = 2^e$ is that this fact is not true when q is not a power of two.

The matrix $\mathrm{D}_x f$ is easy to compute given f and x. If f is a (homogeneous) quadratic map, then it is in fact a vector of n quadratic forms, which can conveniently be described by a collection of n matrices $F_1, \ldots, F_n$, that are interpreted as follows: $F_k[i,j]$ is the coefficient of $x_i x_j$ in the k-th component of f. If ${}^tM$ denotes the transpose of M, then the matrix representation of the differential of f at x is given by:

$$\mathrm{D}_x f = \left(x \cdot (F_1 + {}^tF_1) \middle| \ldots \middle| x \cdot (F_n + {}^tF_n) \right).$$

Thus, given a vector x, finding the neighbors of x in G_f can be done in time $\mathcal{O}\left(n^3\right)$: computing the matrix $\mathrm{D}_x f$ requires n matrix-vector products, and determining its kernel classically takes $\mathcal{O}\left(n^3\right)$ operations. It is thus possible to crawl the linearity graphs by spending a polynomial number of elementary operations on each traversed vertex.

Structure in Linearity Graphs. Linearity graphs possess a rich structure, thanks to their algebraic origin. Recall that in G_f, two nodes x and y are adjacent if $\psi(x,y) = 0$, where ψ is the symmetric bilinear map defined above. The bilinearity of ψ induces a lot of structure in G_f. For instance, we always have $\psi(x,x) = 0$, and by bilinearity $\psi(\lambda x, \mu x) = \lambda\mu\psi(x,x) = 0$, so that the q multiples of a vector x form a clique in G_f. The set of all multiples of x are thus topologically indifferentiable (they all have the exact same neighborhood).

Furthermore, the same reasoning shows that if two vectors x and y are adjacent in G_f, then the set of q^2 linear combinations $\lambda x + \mu y$ form a clique in G_f of size q^2.

Degree Distribution. If a quadratic map f is randomly chosen (amongst the finite number of possibilities), then the resulting linearity graph G_f follows a certain —mostly unknown— probability distribution, and any property of G_f can be seen as a random variable. One of the most interesting properties of G_f is the distribution of the *degree* (*i.e.*, of the number of neighbors) of vertices in

G_f. This result is stated in terms of the probability that a random $n \times n$ matrix over $\mathbb{F}_q$ is invertible. We denote it by $\lambda(n)$:

$$\lambda(n) = \prod_{i=1}^{n} \left(1 - \frac{1}{q^i}\right)$$

Lemma 2 (theorem 2 in [15]). *Let $x \in (\mathbb{F}_q)^n$ be a non-zero vector, and $f : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^n$ be a uniformly random quadratic map. Then $\mathrm{D}_x f$ is a uniformly random matrix vanishing over x. As a consequence, the probability that $\mathrm{D}_x f$ has a kernel of dimension $k \geq 1$ is:*

$$\frac{\lambda(n)\lambda(n-1)}{\lambda(k)\lambda(k-1)\lambda(n-k)} q^{-k(k-1)}$$

Because $\lambda(n)$ is a decreasing function of n that converges to a finite limit bounded away from zero, then the ratio of the λ-expressions lives in a small interval, independently of q, n and k, so that the probability is in fact of order $q^{-k(k-1)}$. Of course, over $\mathbb{F}_q$, a k-dimensional vector space contains q^k elements, so that if $\dim \ker \mathrm{D}_x f = k$, then the vertex x has q^k neighbors.

Sparsity. Computing the expectation and the variance of the degree is technical, but feasible:

$$\mathbb{E}\left[\text{degree}\right] = q - \frac{1}{q^{n-2}} \qquad \sigma^2 = q^2(q-1)\left(1 - \frac{q^2+1}{q^n} + \frac{q^2}{q^{2n}}\right)$$

Establishing these two expressions is somewhat technical, yet because both are sums of q-hypergeometric terms, they can be computed by "creative telescoping" thanks to the q-analog of Zeilberger's algorithm [42]. It follows that the expected number of edges of G_f is essentially $q^{n+1}/2$. In other terms, G_f is a very sparse graph that has barely more edges than it has vertices.

Disconnecting Linearity Graphs. A linearity graph G_f is fully connected, because all vertices are adjacent to the "zero" vertex. This "zero" vertex is not very interesting (since it is adjacent to *every* other vertex), and, as a matter of fact, it even turns out to be a bit annoying. Thus, it seems that there is nothing to lose by removing it. In addition, we could also get rif of the self-loops ; they are useless since *every* vertex has one.

We thus denote by G_f^* the simple graph G_f in which the zero vertex has been removed, and where self-edges are removed. It is interesting to note that the resulting graph is no longer connected, and that there are in fact very many connected components. Indeed, if $\dim \ker \mathrm{D}_x f = 1$, then the only neighbors of x are its multiples, and x belong to a connected component of size $q-1$. Lemma 2 tells us that this happens with probability $\lambda(n)/\lambda(1)$, and this converges to a finite limit bounded away from zero when n goes to infinity. Thus, a constant fraction of the vertices belong to "small" connected components of size $q-1$. Working a bit on the λ functions reveals that this proportion grows like $1 - 1/q^2$.

4 Just Count Your Neighbors

It is well-know that if two graphs (V_1, E_1) and (V_2, E_2) are isomorphic, and if ρ is an isomorphism between them, then $u \in V_1$ and $\rho(u) \in V_2$ have the same *degree*, *i.e.*, the same number of neighbors. It follows that if $u \in V_1$ and $v \in V_2$ do not have the same degree, then they cannot be related by ρ.

We adapt this simple idea in the context of QMLE, under the form of Algorithm 2. The main idea in this algorithm is to target vertices in the linearity graphs of f and g that have a specific degree: we only look for a "right pair" $y = S \cdot x$ amongst vertices x, y that have a prescribed degree (chosen to optimise the complexity of the algorithm). The remaining of this section is devoted to establishing the properties of this algorithm, which are summarized in the following theorem.

Algorithm 2 First Birthday Based Algorithm

function SampleSet(h)
  $L \leftarrow \emptyset$
  repeat
    repeat
      $x \leftarrow$ random vertex of G_h
    until x has $q^{\sqrt{n/3}}$ neighbors
    $L \leftarrow L \cup \{x\}$
  until $|L| = \sqrt{2}q^{n/3}$
  return L

function Neighbor-Counting-QMLE(f, g)
  $U \leftarrow$ SampleSet (f)
  $V \leftarrow$ SampleSet (g)
  for all $(x, y) \in U \times V$ **do**
    $f'(z) \leftarrow f(z + y)$
    $g'(z) \leftarrow g(z + x)$
    query IQMLE-Solver with (f', g')
    if solution (S, T) found **then return** (S, T)
  return "Probably not equivalent"

Theorem 1. *Algorithm 2 performs* $\mathcal{O}\left(q^{2n/3}\right)$ *units of computations on average, sends at most* $q^{2n/3}$ *queries to the inhomogeneous solver, and succeeds with probability* $1 - 1/e$.

The helper function SampleSet returns a set of $\mathcal{O}\left(q^{n/3}\right)$ vertices of G_f (resp. G_g), each having $q^{\sqrt{n/3}}$ neighbors in the graph. It follows that there are $q^{2n/3}$ queries to the inhomogeneous solver, because this is the size of the cartesian product $U \times V$.

It remains to establish the complexity of SampleSet, and the success probability of the algorithm. As explained above, since we are looking for a "right

pair" $y = S \cdot x$, it is safe to restrict our attention to vertices x, y that have a specific degree (as long as vertices with such a degree exist in the graphs).

Lemma 2 gives us the expected number iterations of the innermost loop of SampleSet that are required to find a random vertex with the required degree. Up to a constant factor, finding a vertex with degree q^k requires $q^{k(k-1)}$ trials, so that finding each new random vertex requires $\mathcal{O}\left(q^{n/3}\right)$ rank computations on $n \times n$ matrices, hence $\mathcal{O}\left(n^3 \cdot q^{n/3}\right)$ operations.

Lemma 2 also tells us that there are on average $q^{n-k(k-1)}$ vertices in G_f each having degree q^k. In Algorithm 2 we look specifically at vertices of degree $q^{\sqrt{n/3}}$, and we thus expect G_f to contain $q^{2n/3}$ of them. Since the number of iterations of the outermost **repeat**...**until** loop is roughly the square root of this number, we do not expect more than a constant number of "extra" iterations finding an already-known vector x. Putting everything together, we conclude that SampleSet terminates after $\mathcal{O}\left(n^3 q^{2n/3}\right)$ operations.

Now, the birthday bound tells us that $U \times V$ contains a "right pair" $y = Sx$ with probability greater than 63%, because both U and V contain about the square root of the total number of vertices with degree $q^{\sqrt{n/3}}$ (see [39] for a precise statement of this specific version of the birthday paradox).

Practical Results. We have implemented Algorithm 2 inside the MAGMA computer algebra system[8], running on one core of a 2.8 Ghz Xeon machine. As shown in Table 1, we found out that in practice it is difficult to balance the cost of building U and V on the one hand, and going through the candidate pair on the other hand, because the target degree can only take $\sqrt{n}$ integer values. We could nevertheless verify in practice that the complexity of building the lists and the expected number of right pairs in them is consistent with our expectations. The source code is in the public domain, and is available on the webpage of the first author. It uses an unpublished algorithm to solve the inhomogeneous instances.

n	q	generating U and V	total time	$\log_q$ (target degree)	$\lvert U \rvert$	# pairs
16	2	0s	68s	3	1	4
22	2	28s	9h45m	4	13	400
28	2	4913s	2h15m	5	8	64

Table 1. Experimental results on Algorithm 2.

5 Map Your Neighborhood

We have seen in section 3 that the linearity graphs, once deprived from the "zero" vertex, contain many small connected components. Of course, if $y = Sx$, then

the connected component of x is isomorphic to the connected component of y. In this section, we describe an algorithm that builds upon this idea—instead of just looking at immediate neighbors, as we did in algorithm 2, we now try to look at the whole connected component, in order to distinguish between vertices of the same degree.

Canonical Graph Labeling. Given a graph G, a *Canonical Labeling algorithm* relabels the vertices of G, thus producing a graph $Canon(G)$, which is by definition isomorphic to G. The result is canonical in the sense that if G and H are isomorphic graphs, then $Canon(G) = Canon(H)$. The canonical labels are therefore complete invariants of the isomorphism class, and as such, computing a canonical labeling is necessarily harder than checking if two graphs are isomorphic. However, computing a canonical labeling can be done in average linear time [3], because except for an exponentially small fraction of all graphs, it can be done with a very simple linear algorithm.

Back to our more specific problem, let us denote by C_x (resp C_y) the connected component of x in G_f^* (resp. of y in G_g^*). The key idea of the algorithm presented in this section is that $y = Sx$ implies $Canon(C_x) = Canon(C_y)$. Thus, it seems that the function $H : u \mapsto Canon(C_u)$ could be used as a "hash function". In fact, in algorithm 2, we used the degree as such a "hash function", but it was not very discriminating, because the degree does not contain enough entropy. We hope that H behaves as a good hash function, and that *false positives*, *i.e.*, pairs (x, y) such that $H(x) = H(y)$ but $y \neq Sx$, should be very rare.

One problem is that H does not distinguish between vertices of the same connected component. To improve it, we would need a way to single out a specific vertex in the connected component. Fortunately, most canonical labeling algorithm return the isomorphism (say ρ) between their argument G and $Canon(G)$. To single a vertex x out in G, it is sufficient to send $\rho(x)$ along with the canonical labeling of G.

A Canonical-Labeling-Based Algorithm As discussed in section 3, G_f^* contains many small connected components that are all isomorphic to each others, since they are all cliques of size $q - 1$. Therefore, if we want our "hash function" to be discriminating, we must avoid small connected components. Our "hash function" will thus reject the vector x if there is no simple path starting from x and of length at least r. In the other direction, we cannot exclude the existence of a giant connected component of exponential size. Therefore, we only consider the radius-r neighborhood of the vertex x we are interested in, *i.e.*, the set of all vertices that can be reached from x by crossing at most r edges. This is the basis of algorithm 3.

Remarks on Algorithm 3. Establishing the complexity and success probability of algorithm 3 is surprisingly difficult, probably because is relies on topological properties of G_f^*, which is a somewhat random but very structured graph.

Algorithm 3 Canonical Labeling/Birthday Based Algorithm

function $\textsc{Hashable}^{[r]}(G, x)$
    Perform a Breadth-First Search in G starting from x
    return $\textsc{True}$ if the BFS hits a vertex r edges away from x

function $\mathrm{H}^{[r]}(G, x)$
    $C_x \leftarrow$ subgraph of G formed by all vertices at most r edges away from x.
    $\rho, \mathcal{G} \leftarrow \textsc{CanonicalLabeling}(C_x)$
    return $\big(\mathcal{G}, \rho(x)\big)$

function $\textsc{SampleHashTable}(h)$
    $L \leftarrow \emptyset$
    repeat
        repeat
            $x \leftarrow$ random vertex of G_h^*
        until $\textsc{Hashable}^{[r]}(G_h^*, x)$
        $L\left[\mathrm{H}^{[r]}\left(G_h^*, x\right)\right] \leftarrow x$
    until $|L| = \sqrt{2}q^{n/2}$
    return L

function $\textsc{Canonical-Labeling-QMLE}(f, g)$
    $U \leftarrow \textsc{SampleHashTable}\,(f)$
    $V \leftarrow \textsc{SampleHashTable}\,(g)$
    for all $(h_1 \mapsto x) \in U, (h_2 \mapsto y) \in V$ **such that** $h_1 = h_2$ **do**
        $f'(z) \leftarrow f(z + y)$
        $g'(z) \leftarrow g(z + x)$
        query IQMLE-$\textsc{Solver}$ with (f', g')
        if solution (S, T) found **then return** (S, T)
    return "Probably not equivalent"

Algorithm 3 has been written in a generic way, independently of the actual value of q. However, we have only been able to discuss its properties when $q = 2$. We have verified that the algorithm works as we expected in this case, but the situation when $q \neq 2$ is not so clear. We tend to believe that the complexity and/or success probability degrade exponentially fast when q grows, but we fall short of definitive conclusion.

When $q = 2$, the structure of G_f^* seems to be richer. For instance, we already alluded to the fact that the fraction of nodes whose connected component is of size only $q - 1$, grows like $1 - 1/q^2$. In addition, as we will see in the next section, setting $q = 2$ allows us to turn most more-or-less-random graphs into trees, which are much easier to deal with.

Preliminary Analysis of Algorithm 3. When $q = 2$, the correctness of the algorithm is implied by the following three heuristic statements.

Claim. *i)* $\textsc{Hashable}^{[r]}\left(G_f^*, x\right)$ is true with probability $\approx 1/r$ over the random choice of f (assuming $x \neq 0$).

ii) Both $\textsc{Hashable}^{[r]}$ and $\mathrm{H}^{[r]}$ can be evaluated in expected time $\mathcal{O}\left(rn^3\right)$.

iii) When restricted to elements that are $\textsc{Hashable}^{[r]}$, then $\mathrm{H}^{[r]}\left(G_f^*, \cdot\right)$ is an *ε^r-almost universal hash function family* (indexed by f) for some $\varepsilon < 1$.

The notion of almost universal hash function is usually useful when the hash function is "less injective" than a random function. In this paper though, $\mathrm{H}^{[r]}$ can become *more injective* than a random function, as soon as r becomes sufficiently large.

It follows from claim *i* that the expected number of iterations of the loop of lines 11–13 is $\mathcal{O}(r)$, and it follows from claim *ii* that finding one admissible vector x requires $\mathcal{O}\left(r^2n^3\right)$ operations on average. Claim *iii* then guarantees that if we choose r to be a bit larger than n, then the probability to find hash collisions can be made smaller than 2^{-n}, and standard birthday-type results guarantee that the number of expected hash collisions in the execution of SampleHashTable is constant. From this, we conclude that SampleHashTable runs in expected time $\mathcal{O}\left(r^2n^3q^{n/2}\right)$.

It follows from the birthday paradox [39] that there is a "right pair" in $U \times V$, *i.e.*, a pair (x, y) with $y = Sx$, with probability greater than $1 - 1/e$. This is because $(\mathbb{F}_q)^n$ has q^n elements and that the sizes of both U and V are essentially $q^{n/2}$. This guarantees the success probability of the algorithm.

Let us denote by $\mathcal{N}$ the number of bogus inhomogeneous queries, *i.e.*, the number of pairs $x \neq y \in U \times V$ with the same hash. It follows from Markov's inequality and claim *iii* that $\mathbb{P}\left[\mathcal{N} \geq 1\right] \leq 2q^n \cdot \varepsilon^r$. Thus, as soon as r is asymptotically larger than n, *e.g.* $r = n \log\log n$, then the probability that $\mathcal{N} \geq 1$ gets exponentially small. This concludes our preliminary analysis: algorithm 3 runs in time $\mathcal{O}\left(n^5q^{n/2}\right)$, and sends a constant number of inhomogeneous queries. It now remains to show that our claims are valid, but we first find it reassuring to show that the practical behavior of the algorithm is very consistent with our expectations.

Discussion of the Claims. Because of space limitations, we discuss these claims in the extended version of the paper [10].

experimental results. We have implemented Algorithm 3 using the MAGMA computer algebra system [8], and we found out that it works well in practice, as Table 2 shows. The experiment clearly shows that $\mathcal{N}$ is constant, as expected. This justify our heuristic analysis *a posteriori*. The implementation is in the public domain and is available on the webpage of the first author.

n	q	generating U and V	finding collisions	$\lvert U\rvert$	$\mathcal{N}$
16	2	3.6 s	1s	64	6
24	2	123 s	13s	836	5
32	2	61 min	200s	11585	2
40	2	31 h	2h	165794	7

Table 2. Experimental results on Algorithm 3

References

1. Agrawal, M., Saxena, N.: Equivalence of f-algebras and cubic forms. In Durand, B., Thomas, W., eds.: STACS. Volume 3884 of Lecture Notes in Computer Science., Springer (2006) 115–126
2. Alon, N., Blais, E.: Testing boolean function isomorphism. In Serna, M.J., Shaltiel, R., Jansen, K., Rolim, J.D.P., eds.: APPROX-RANDOM. Volume 6302 of Lecture Notes in Computer Science., Springer (2010) 394–405
3. Babai, L., Kucera, L.: Canonical labelling of graphs in linear average time. In: FOCS, IEEE Computer Society (1979) 39–46
4. Bardet, M., Faugère, J.C., Salvy, B.: On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In: Proc. International Conference on Polynomial System Solving (ICPSS). (2004) 71–75
5. Bettale, L., Faugère, J.C., Perret, L.: Cryptanalysis of the trms signature scheme of pkc'05. In Vaudenay, S., ed.: AFRICACRYPT. Volume 5023 of Lecture Notes in Computer Science., Springer (2008) 143–155
6. Billet, O., Gilbert, H.: A traceable block cipher. In Laih, C.S., ed.: ASIACRYPT. Volume 2894 of Lecture Notes in Computer Science., Springer (2003) 331–346
7. Biryukov, A., Cannière, C.D., Braeken, A., Preneel, B.: A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In: EUROCRYPT. (2003) 33–50
8. Bosma, W., Cannon, J.J., Playoust, C.: The Magma Algebra System I: The User Language. J. Symb. Comput. **24**(3/4) (1997) 235–265
9. 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 Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A., eds.: Public Key Cryptography. Volume 6571 of Lecture Notes in Computer Science., Springer (2011) 473–493

10. Bouillaguet, C., Fouque, P.A., Véber, A.: Graph-theoretic algorithms for the "isomorphism of polynomials" problem. Cryptology ePrint Archive, Report 2012/607 (2012) `http://eprint.iacr.org/`.
11. Cramer, R., ed.: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. In Cramer, R., ed.: EUROCRYPT'05. Volume 3494 of Lecture Notes in Computer Science., Springer (2005)
12. Daemen, J.: Limitations of the even-mansour construction. [25] 495–498
13. Ding, J., Wolf, C., Yang, B.Y.: -invertible cycles for multivariate quadratic public key cryptographyℓ. In Okamoto, T., Wang, X., eds.: Public Key Cryptography. Volume 4450 of Lecture Notes in Computer Science., Springer (2007) 266–281
14. Dubois, V., Fouque, P.A., Shamir, A., Stern, J.: Practical Cryptanalysis of SFLASH. In: CRYPTO. Volume 4622., Springer (2007) 1–12
15. Dubois, V., Granboulan, L., Stern, J.: An efficient provable distinguisher for hfe. In Bugliesi, M., Preneel, B., Sassone, V., Wegener, I., eds.: ICALP (2). Volume 4052 of Lecture Notes in Computer Science., Springer (2006) 156–167
16. Dunkelman, O., Keller, N., Shamir, A.: Minimalism in cryptography: The even-mansour scheme revisited. In Pointcheval, D., Johansson, T., eds.: EUROCRYPT. Volume 7237 of Lecture Notes in Computer Science., Springer (2012) 336–354
17. Even, S., Mansour, Y.: A construction of a cioher from a single pseudorandom permutation. [25] 210–224
18. Faugère, J.C., Joux, A., Perret, L., Treger, J.: Cryptanalysis of the hidden matrix cryptosystem. In Abdalla, M., Barreto, P.S.L.M., eds.: LATINCRYPT. Volume 6212 of Lecture Notes in Computer Science., Springer (2010) 241–254
19. Faugère, J.C., Perret, L.: Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In Vaudenay, S., ed.: EUROCRYPT. Volume 4004 of Lecture Notes in Computer Science., Springer (2006) 30–47
20. Fouque, P.A., Granboulan, L., Stern, J.: Differential cryptanalysis for multivariate schemes. [11] 341–353
21. Fouque, P.A., Macario-Rat, G., Perret, L., Stern, J.: Total break of the ℓ-ic signature scheme. In Cramer, R., ed.: Public Key Cryptography. Volume 4939 of Lecture Notes in Computer Science., Springer (2008) 1–17
22. Fouque, P.A., Macario-Rat, G., Stern, J.: Key Recovery on Hidden Monomial Multivariate Schemes. In Smart, N.P., ed.: EUROCRYPT. Volume 4965 of Lecture Notes in Computer Science., Springer (2008) 19–30
23. Geiselmann, W., Meier, W., Steinwandt, R.: An Attack on the Isomorphisms of Polynomials Problem with One Secret. Int. J. Inf. Sec. **2**(1) (2003) 59–64
24. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: FOCS, IEEE (1986) 174–187
25. Imai, H., Rivest, R.L., Matsumoto, T., eds.: Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings. In Imai, H., Rivest, R.L., Matsumoto, T., eds.: ASIACRYPT. Volume 739 of Lecture Notes in Computer Science., Springer (1993)
26. Joux, A., Kunz-Jacques, S., Muller, F., Ricordel, P.M.: Cryptanalysis of the tractable rational map cryptosystem. [40] 258–274
27. Kayal, N.: Efficient algorithms for some special cases of the polynomial equivalence problem. In Randall, D., ed.: SODA, SIAM (2011) 1409–1421
28. Patarin, J.: Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In: EUROCRYPT. (1996) 33–48

29. Patarin, J., Goubin, L., Courtois, N.: c^* _-+ and hm: Variations around two schemes of t. matsumoto and h. imai. In Ohta, K., Pei, D., eds.: ASIACRYPT. Volume 1514 of Lecture Notes in Computer Science., Springer (1998) 35–49
30. Patarin, J., Goubin, L., Courtois, N.: Improved Algorithms for Isomorphisms of Polynomials. In: EUROCRYPT. (1998) 184–200
31. Patarin, J., Goubin, L., Courtois, N.: Improved Algorithms for Isomorphisms of Polynomials – Extended Version. available at `http://minrank.org/ip6long.pdf` (1998)
32. Perret, L.: A Fast Cryptanalysis of the Isomorphism of Polynomials with One Secret Problem. [11] 354–370
33. Pointcheval, D.: A new identification scheme based on the perceptrons problem. In: EUROCRYPT. (1995) 319–328
34. Sakumoto, K.: Public-key identification schemes based on multivariate cubic polynomials. In Fischlin, M., Buchmann, J., Manulis, M., eds.: Public Key Cryptography. Volume 7293 of Lecture Notes in Computer Science., Springer (2012) 172–189
35. Sakumoto, K., Shirai, T., Hiwatari, H.: Public-key identification schemes based on multivariate quadratic polynomials. In Rogaway, P., ed.: CRYPTO. Volume 6841 of Lecture Notes in Computer Science., Springer (2011) 706–723
36. Shamir, A.: An efficient identification scheme based on permuted kernels (extended abstract). In Brassard, G., ed.: CRYPTO. Volume 435 of Lecture Notes in Computer Science., Springer (1989) 606–609
37. Stern, J.: A new identification scheme based on syndrome decoding. In Stinson, D.R., ed.: CRYPTO. Volume 773 of Lecture Notes in Computer Science., Springer (1993) 13–21
38. Stern, J.: Designing identification schemes with keys of short size. In Desmedt, Y., ed.: CRYPTO. Volume 839 of Lecture Notes in Computer Science., Springer (1994) 164–173
39. Vaudenay, S.: A Classical Introduction to Cryptography: Applications for Communications Security. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2005)
40. Vaudenay, S., ed.: Public Key Cryptography - PKC 2005, 8th International Workshop on Theory and Practice in Public Key Cryptography, Les Diablerets, Switzerland, January 23-26, 2005, Proceedings. In Vaudenay, S., ed.: Public Key Cryptography. Volume 3386 of Lecture Notes in Computer Science., Springer (2005)
41. Wang, L.C., Hu, Y.H., Lai, F., yen Chou, C., Yang, B.Y.: Tractable rational map signature. [40] 244–257
42. Wilf, H., Zeilberger, D.: An algorithmic proof theory for hypergeometric (ordinary and "q") multisum/integral identities. Inventiones Mathematicae **108** (1992) 575–633 10.1007/BF02100618.