Differential-Algebraic Algorithms for the Isomorphism of Polynomials Problem

Charles Bouillaguet[1], Jean-Charles Faugère[2,3],
Pierre-Alain Fouque[1] and Ludovic Perret[3,2]

[1] Ecole Normale Supérieure
{charles.bouillaguet, pierre-alain.fouque}@ens.fr
[2] SALSA Project - INRIA (Centre Paris-Rocquencourt)
UPMC, Univ Paris 06 - CNRS, UMR 7606, LIP6
104, avenue du Président Kennedy 75016 Paris, France
jean-charles.faugere@inria.fr, ludovic.perret@lip6.fr

Abstract. In this paper, we investigate the difficulty of the Isomorphism of Polynomials (IP) Problem as well as one of its variant IP1S. The Isomorphism of Polynomials is a well-known problem studied more particularly in multivariate cryptography as it is related to the hardness of the key recovery of such cryptosystems. The problem is the following: given two families of multivariate polynomials $\mathbf{a}$ and $\mathbf{b}$, find two invertible linear (or affine) mappings S and T such that $\mathbf{b} = T \circ \mathbf{a} \circ S$. For IP1S, we suppose that T is the identity. It is known that the difficulty of such problems depends on the structure of the polynomials (*i.e.*, homogeneous, or not) and the nature of the transformations (affine, or linear). Here, we analyze the different cases and propose improved algorithms. We precisely describe the situation in terms of complexity and sufficient conditions to make the algorithms work. The algorithms presented here combine linear algebra techniques, including the use of differentials, together with Gröbner bases and statistical tools such as the birthday paradox and a totally new object in the IP-context, the so-called *Galton-Watson trees*. We show that random instances of IP1S with quadratic polynomials can be broken in time $\mathcal{O}\left(n^6\right)$, where n is the number of variables, independently of the number of polynomials. For IP1S with cubic polynomials, as well as for IP, we propose new algorithms of complexity $\mathcal{O}\left(n^6\right)$ if the polynomials of $\mathbf{a}$ are inhomogeneous and S, T linear. In all the other cases, we propose an algorithm that requires $\mathcal{O}\left(n^6 \cdot q^{n/2}\right)$ computation. Finally, we propose several algorithms for different subcases of the full IP problem, the best of which has complexity $\mathcal{O}\left(q^{n/2}\right)$. These new tools allow to break the challenges proposed by Patarin in practice, and also raise some fundamental questions about the general security of multivariate public-key schemes.

1 Introduction

Multivariate cryptography is concerned with the use of multivariate polynomials over finite fields in cryptographic schemes. The use of polynomial systems in cryptography dates back to the mid eighties with the design of C* [40], and many others proposals appeared afterwards [45,46,47,36,56]. The security of multivariate schemes is in general related to the difficulty of solving random or structured systems of multivariate polynomials. This problem has been proved to be NP-complete [30], and it is conjectured [3] that systems of random polynomials are hard to solve in practice. As usual when a trapdoor must be embedded in a hard problem, easy instances are transformed into random-looking instances using secret transformations. In multivariate cryptography, it is common to turn an easily-invertible collection of polynomials $\mathbf{a}$ into an apparently random one $\mathbf{b}$. It is then assumed that, being supposedly indistinguishable from random, $\mathbf{b}$ should be hard to solve. The structure-hiding transformation is very often the composition with linear (or affine) invertible mappings S and T: $\mathbf{b} = T \circ \mathbf{a} \circ S$. The matrices S and T are generally part of the secret-key.

The Isomorphism of Polynomials (IP) is the problem of recovering the secret transformations S and T given $\mathbf{a}$ and $\mathbf{b}$. It is a fundamental problem of multivariate cryptography, since its hardness implies the difficulty of the key-recovery for various multivariate cryptosystems. With the notable exceptions of the HFE encryption and signature scheme [43] and of the UOV signature scheme [35], most of the multivariate schemes that were broken fell victim of a key-recovery attack, *i.e.*, of a *customized* IP algorithm. Notorious examples include C* [40], the traitor tracing scheme proposed by Billet and Gilbert [8], the SFLASH signature scheme [46], the ℓ-IC signature scheme [17], the square-vinegar signature scheme [2] and the Square encryption scheme [15]. The hardness of the key-recovery problem for HFE or UOV do not rely on the hardness of IP, since the easily-invertible mapping $\mathbf{a}$ is kept secret. In this paper, we do not target a specific cryptosystem nor use the fact that the system $\mathbf{a}$ is built in a special way so that it is efficient to invert. We in fact consider both $\mathbf{a}$ and $\mathbf{b}$ to be random collections of polynomials, which is to some extent a worst-case assumption.

An important special case of IP is the *IP problem with one secret* (IP1S for short). Patarin suggested in 1996 [43] to construct a zero-knowledge public-key authentication scheme relying on the hardness of IP1S, inspired by the Zero-Knowledge proof system for Graph Isomorphism of [33]. The proposed parameters lead to relatively small key sizes (for instance to secret and public keys of 256 bits each and no additional information), as the complexity of the problem was believed to be exponential. These parameters have not been broken so far. The IP1S problem is also interesting from a complexity-theoretic point of view: it has been proved [49] that IP1S is *Graph Isomorphism-hard* (GI-hard for short). This lead Patarin *et al.* to claim that IP1S (and *a fortiori* IP) is unlikely to be solvable in polynomial time, because no polynomial algorithm is known for GI in spite of more than forty years of research. On the other hand, GI is not known to be NP-complete. Forging hard instances of the GI problem is pretty non-trivial, and there are powerful heuristics as well as expected linear time algorithms for random graphs [26]. This compromises the use of GI as an authentication mechanism, and was part of the motivation for introducing IP1S as an alternative.

Related Work. As already explained, IP has been introduced by Patarin in [44,43]. In this section, we summarize existing results on the IP and IP1S problems. The first algorithm for IP, known as the "To and Fro" technique, is due to Courtois *et al.* [49]. In its primitive form, this algorithm assumes the ability to invert the polynomial systems, and has therefore an exponential complexity. Moreover, the image of S has to be known on at least one point, and is found via exhaustive search if no other solution applies. An improved, vaguely described and highly heuristic version of this algorithm is claimed to find a few relations on S in time and space $\mathcal{O}\left(q^{n/2}\right)$. It can be mentionned that the "To and Fro" technique has been adapted in [9] to solve the Linear/Affine Equivalence problem, a parent of IP for SBoxes.

In [25], Perret and Faugère present a new technique for solving IP when S and T are linear (as opposed to affine) invertible mappings. The idea is to model the problem as an algebraic system of equations and solve it by means of Gröbner bases [12,16]. In practice, the technique turns out to be efficient for instances of IP where the coefficients of all the monomials of all degree are randomly chosen in $\mathbb{F}_q$. For random instances of IP, the practical complexity is estimated to be $\mathcal{O}\left(n^9\right)$. This complexity corresponds to the empirical observation that the maximum degree reached during the Gröbner basis computation [22,23] seems to be bounded from above by 3. Note this result no longer holds as soon as the IP instances are "structured". Typically, [25] observed that *homogeneous* instances of IP (in which the polynomials of $\mathbf{a}$ and $\mathbf{b}$ are homogeneous) are much harder to solve in practice, and could be then used for an IP based authentication [44,43].

IP1S is a subcase of IP, thus most of the algorithms presented above could be applied to the one secret variant almost directly. However, several algorithms were developed to exploit the specificities of IP1S. It was shown in [49] that linear equations on the coefficients of the secret matrix can be obtained in some cases. Our quadratic IP1S technique is based on an extension of this observation. To our knowledge, the first algorithm dedicated to IP1S can be found in Geiselmann *et al.* [31]. We briefly explain the idea in the linear case. The authors of [31] remarked that each row of a matrix solution of IP1S verifies an algebraic system of equations. They then used an exhaustive search to find the solutions of such system. Soon after, this technique has been improved by Levy-dit-Vehel and Perret [18] who replaced this exhaustive search by a Gröbner basis computation. This still yields exponential algorithms, and the improvement induced by this replacement is mostly significant for instances defined over a "large" field.

Finally, Perret [50] shows that the affine and linear variants of IP1S are equivalent, *i.e.*, one can without loss of generality restrict our attention to the linear case. In addition, a new approach for solving IP1S using the Jacobian matrix was proposed. The algorithm is polynomial when the number u of polynomials in $\mathbf{a}$ and $\mathbf{b}$ is equal to the number of variables n. However, when $u < n$, the complexity of this approach is not well understood. Moreover, when the number of polynomials is very small, for instance $u = 2$, this algorithm is totally inefficient. Whilst $u = n$ is the classical setting for IP, it is less interesting for IP1S, where u is typically much smaller than n.

All in all, the existing literature on the IP problem discussed above can be split in two categories: *heuristic* algorithms with (more or less vaguely) "known" complexity and unknown success probability [49], and *rigourous* algorithms that always succeeds but with unknown complexity [25,50,18,31]. This situation makes it very difficult, if not plainly impossible to compare these algorithms based on their theoretical features. The class of instances that can be solved by a given algorithm of the first type is in general not known. Conversely, the class of instances over which an algorithm of the second type terminates quickly are often not known as well. This lead the authors of IP algorithms to measure the efficiency of their techniques in practice, or even not to measure it at all (such as Courtois *et al.* [49]). Several IP and IP1S challenges were proposed by Patarin in [43], and can be used to measure the progress accomplished since their introduction. Also, the lack of theoretical understanding of the previous techniques makes it quite difficult to get a clear picture of the secure and broken ranges of parameters.

Techniques. The algorithms presented here are mostly deterministic, and rely on the two weapons that have dealt a severe blow to multivariate cryptography: simple linear algebra and Gröbner bases. However, two algorithms also rely on the birthday paradox, and thus are probablistic by nature. One of them looks for collisions between unranked, unlabelled trees generated using a recursive process. Our ideas borrow to the recent differential cryptanalysis of multivariate schemes mentionned above. The analysis of all our algorithms makes heavy use of known linear algerba results, for instance about the dimension of the commutant of a matrix, and also on known results about random matrices, most notably the distribution of the rank and the probability of being cyclic. The two most delicate steps involve lower-bounding the dimension of the kernel of a homogeneous system of matrix equations, and upper-bounding the degree of polynomials manipulated by a Gröbner-basis algorithm. Analyzing the birthday-based algorithms additionally makes use of known results on the properties of Galton-Watson trees, which is to our knowledge a completely new tool to address IP/IP1S problems.

Our Results. We present six new algorithms, which are summarized in Figure 1. On the practical side, these algorithms are efficient: random quadratic IP1S instances and random inhomogeneous instances of IP or IP1S can all be broken in practical time for any size of the parameters. In particular, all the quadratic IP1S challenges are now broken in a few seconds. The biggest cubic IP1S challenge is broken in less than 2 CPU-month. The IP1S authentication scheme is thus broken beyond repair in the quadratic case. In the case of cubic IP1S, the security parameter have to be seriously reconsidered, which makes the scheme much less attractive in terms of key size.

Problem	Degree	Subcase	Previous state of the art	Our contribution
IP1S	2 or 3	$u = n$, homogeneous	Polynomial [50]	$\mathcal{O}\left(n^6\right)$, section 3.1
	2	$u \ll n$	exhaustive search : $\mathcal{O}\left(q^{n^2}\right)$	
	3	$u \ll n$, inhomogeneous		$\mathcal{O}\left(n^6\right)$, section 3.2
		$u \ll n$, homogeneous		$\mathcal{O}\left(n^6 \cdot q^{n/2}\right)$, section 3.2
IP	2	$u = n$, inhomogeneous	$\mathcal{O}\left(n^9\right)$ [25]	Heuristic : $\mathcal{O}\left(n^3\right)$, section 4.1
				Rigorous : $\mathcal{O}\left(n^6\right)$, section 4.2
		$u = n$, homogeneous/affine	$\mathcal{O}\left(n \cdot q^n\right)$ or allegedly $\mathcal{O}\left(q^{n/2}\right)$ [49]	$\mathcal{O}\left(n^6 \cdot q^{2n/3}\right)$, section 5.1
				$\mathcal{O}\left(n^{3.5} \cdot q^{n/2}\right)$, section 5.2

Fig. 1: Our results, compared with previous work.

We present two algorithms to deal with the "easy cases" of the full IP problem, namely when S and T are linear (as opposed to affine), and when a and b are random and inhomogeneous. These algorithms deal with the same case where the Gröbner-based algorithm of [25] was already polynomial $\left(\text{in } \mathcal{O}\left(n^9\right)\right)$, but they are much faster, with respective complexity of $\mathcal{O}\left(n^3\right)$ and $\mathcal{O}\left(n^6\right)$.

Lastly, we present two algorithms that deal with the hardest case of the IP problem, when $\mathbf{a}$ and $\mathbf{b}$ are homogeneous. These algorithms rely on the birthday paradox to find the image of a vector by S, then fall back on the inhomogeneous case. The first one, which has complexity $\mathcal{O}\left(n^6 \cdot q^{\frac{2}{3}n}\right)$ and is simple to analyze, use the fact that the rank is an invariant of the differential (and essentially reduces to solve "easy" instances of the MinRank problem [13]). The second one uses a new invariant of the differential, which allows to associate a branching process (*i.e.*, a Galton-Watson tree) to each point, and to find collisions between such random trees. Its complexity is $\mathcal{O}\left(n^{3.5} \cdot q^{n/2}\right)$ under a plausible conjecture about Galton-Watson trees.

A rigorous analysis of our algorithms is both necessary and tricky. When generating linear equations, special care has to be taken to count how many of them are independent. The recent history of algebraic cryptanalysis taught us that failure to do so may have drastic consequences. Additionally, the complexity of Gröbner bases computation, even though a bit more well-understood now in the generic case, is still often a delicate matter for structured systems. The algorithms presented in this paper mostly belongs to "rigorous" type. One exception is the heuristic for linear inhomogeneous IP presented in section 4.1. By showing a case where this heuristic fails, we highlight even more the need for a rigorous analysis. The probabilistic algorithms of section 5 cannot, by definition, be "rigorous". However, we analyzed their success probability on random instances, and give their running time so that this probability is close to one.

A distinctive feature of our algorithms compared to the previous state of affairs, and one of our main theoretical contribution, is that we characterize the class of instances that can be solved by each algorithm in polynomial time. We then estimate as rigorously as possible the probability that a random instance falls in this class. The rigorous analysis of the algorithm of section 4.2 shows that it solves random linear inhomogeneous instances of IP in time $\mathcal{O}\left(n^6\right)$ with non-negligible probability, and gives *en route* a theoretical justification to the empirical observation of a polynomial behavior for the same subproblem in [25].

Trying to establish the complexity of our algorithms lead us to revisit the techniques of Courtois *et al.* [49] with new theoretical tools. Unfortunately, we found out that some of their claims are unfounded if not plainly wrong. While the idea presented in section 9 ("Some Easy Cases of IP1S") of [49] is powerful and constitutes a building block of our algorithm, we show that it cannot break IP1S by itself, contrary to what is implied in [49]. Additionally, in section 10.3 ("Combined Power Attack") of the same paper an algorithm is vaguely described and claimed to break the full IP problem in time $\mathcal{O}\left(q^{n/2}\right)$. It introduced the idea of using the birthday paradox to find relations on the secret matrix S faster than by exhaustive search. We show that several heuristic assumptions used to establish the complexity and success probability of the algorithm of [49] are in fact not true. As a consequence, its complexity is higher than expected, and is in fact unknown.

On the theoretical side again, we clarify the question of the relative complexity of various classes of IP subproblems. We identify the parameters that make an IP instance easy or hard. Such parameters include the homogeneousness of the polynomials, their degree, the parity of the characteristic of the field, and the presence of an affine part in the linear masks S and T. We extend the affine/linear equivalence result of Perret [50] to the full IP problem. A consequence is that the hardest IP instances are not the affine ones, but the linear homogeneous ones. All the previous algorithms that were dealing with the affine case [49,31] first guessed the affine component of S or T in order to reduce the affine instance to a linear one; these algorithms were exponential by nature. Our algorithm for random quadratic IP1S instances is polynomial even in the affine case.

Cryptographic Implications. The consequences of our results are threefold. Our algorithms directly break the authentication scheme proposed by Patarin in [44], as well as the Hidden Matrix scheme (HM) [48].

Secondly, these improved IP algorithms can be used as intermediate steps in further cryptanalysis: for instance it has recently been shown [11] that the "subfield" variant of HFE (where all the secrets coefficients are drawn from a subfield to reduce the size of the public key) was susceptible to an attack in which an instance of the IP problem has to be solved.

Lastly, we hope our algorithms should affect the way the community thinks about multivariate cryptography. We do not break any new schemes, because all the schemes whose security relied on the hardness of some instances of the IP problem were *already* broken by *ad hoc* IP algorithms. This includes C* [40], but also HM [48], SFLASH [46], or more recently the ℓ-IC scheme [17], the "Medium Field Encryption" (MFE) [52], the "Tractable Rational Map Signatures" [51], all the numerous variants of the "Tame Transformation Signatures" (TTS) and the 2009 proposal Square [15]. Our algorithms justify the weakness of all these schemes *a posteriori* and show that this weakness is fundamental and unavoidable. To some extent, this raises some questions about the general security of multivariate public-key schemes (MPKS). Thanks to our results, it is clear that designs of multivariate schemes relying on the hardness of IP are unsound by nature. We must avoid multivariate schemes having a fixed secret internal polynomial. Instead, the easily-invertible quadratic mapping should be part of the secret key, but doing so is very likely to lead to the existence of many equivalent keys [55,54], and the resulting situation should be considered with a particular attention.

Organisation of the paper. In section 2, we present some basic ingredients that explain the Faugère and Perret algorithm and our basic strategy to solve the IP problem. Then, in section 3, we present our results concerning the IP1S problem with quadratic and cubic polynomials. In section 4, we present and analyze our deterministic algorithms to solve the IP problem with two secrets. Finally, in section 5, we present improved probabilistic algorithms for solving IP which use a birthday paradox and take advantage of Galton-Watson trees. All the proofs have been moved to appendix A.1.

2 Preliminaries

In this section, we give the definition of the differential of a function and we present a basic fact which is the core of the Faugère-Perret algorithm [25]. Then, we present our meta-algorithm for the IP problem.

Differentials. We denote by $\mathrm{D}f : Dom^2 \rightarrow Rng$ the *differential* of a function $f : Dom \rightarrow Rng$. $\mathrm{D}f$ is defined by:

$$\mathrm{D}f(\mathbf{x}, \mathbf{y}) = f(\mathbf{x} + \mathbf{y}) - f(\mathbf{x}) - f(\mathbf{y}) + f(0)$$

It is easy to see that $\mathrm{D}f(\mathbf{x}, \mathbf{y}) = \mathrm{D}f(\mathbf{y}, \mathbf{x})$. If f is a polynomial of total degree d, then $\mathrm{D}f$ is a polynomial of total degree d, but of degree $d-1$ in $\mathbf{x}$ and $\mathbf{y}$. For a given point $c \in (\mathbb{F}_q)^n$, we define:

$$\frac{\partial f}{\partial c} = \mathrm{D}f(c, \mathbf{x})$$

When f is quadratic, then $\mathrm{D}f$ is a symmetric bilinear function, and therefore $\frac{\partial f}{\partial c}$ is a linear mapping. By an abuse of notation, we also denote by $\frac{\partial f}{\partial c}$ the corresponding matrix.

A Structural Observation on IP Problems. Amongst the various parameters that influence the hardness of breaking instances of the IP problem, two play a special, correlated role: whether S and T are *linear* or *affine* transforms on the one hand, and whether the polynomials in $\mathbf{a}$ and $\mathbf{b}$ are *homogeneous* or not. Let us assume that $T(x) = T_c + T_\ell \cdot x$, and $S(x) = S_c + S_\ell \cdot x$, with $S_\ell, T_\ell \in \mathrm{GL}_n(\mathbb{F}_q)$. The following lemma is the fundamental result over which the Gröbner-based algorithm of [25] is built:

Lemma 1. *i) For all $k \geq 1$, we have:*

$$\mathbf{b}^{(k)} = T_\ell \circ \left(\mathbf{a} + \frac{\partial \mathbf{a}}{\partial S_c}\right)^{(k)} \circ S_\ell.$$

ii) In particular, if S and T are linear*, then for all k we have: $\mathbf{b}^{(k)} = T \circ \mathbf{a}^{(k)} \circ S$.*

iii) In all cases, if d is the degree of $\mathbf{a}$ and $\mathbf{b}$, then $\mathbf{b}^{(d)} = T \circ \mathbf{a}^{(d)} \circ S$.

iv) S transforms the set of common zeroes of $\mathbf{a}$ into the set of common zeroes of $\mathbf{b}$.

v) If $\mathbf{a}$ and $\mathbf{b}$ are quadratic, then for any $\mathbf{x}$, we have: $S\left(\ker \frac{\partial \mathbf{b}}{\partial \mathbf{x}}\right) = \ker \frac{\partial \mathbf{a}}{\partial S \cdot \mathbf{x}}$.

Point $iii)$ of lemma 1 shows that the solution of an affine instance is also the solution of the linear homogeneous instance formed by taking only the homogeneous component of higher degree of $\mathbf{a}$ and $\mathbf{b}$. Conversely, if the linear component of the solution of an affine instance is known, retrieving the affine part of the solution is often easy, and it can be accomplished using the technique shown in [32], which we recall. Lemma 1, point $i)$ shows that

$$T_\ell^{-1} \circ \mathbf{b}^{(1)} \circ S_\ell^{-1} - \mathbf{a}^{(1)} = \left(\frac{\partial \mathbf{a}}{\partial S_c} \right)^{(1)}. \tag{1}$$

Once S_ℓ and T_ℓ is known, then the left-hand side of (1) may be computed, and the right-hand side is a function of degree $d-1$ of S_c. In the particular case where $d = 2$, equation (1) in fact describes a system of linear equation that admits S_c as a solution.

A conclusion is that when the polynomials are quadratic, the linear homogeneous case is *complete*, and thus contains the hardest instances. This is in accordance with the experimental results of [25]: they found that their algorithm was polynomial on linear inhomogeneous instances, and exponential on linear homogeneous instances. This is also in accordance with the results presented in this paper: polynomial time algorithms are applicable to the inhomogeneous cases (section 3.2 and 4.1), but only (sub) exponential algorithms deal with the homogeneous cases (in sections 4.3 and 5).

A Meta-Algorithm For IP Problems. In this section, we present a meta-algorithm that can be applied to a wide variety of "IP-like" situations. To recover the isomorphism between two vectors of polynomials, we go through the following steps:

1. Find linear equations between the coefficients of S and those of T^{-1}. Denote them by $\mathcal{S}$.
2. If $\mathcal{S}$ has rank $2n^2 - 1$, then both S and T^{-1} can be retrieved, and we are done.
3. Otherwise, (S, T^{-1}) lives in a vector space of dimension $k = \dim \ker \mathcal{S}$. Applying the algorithm of [25] yields a system $\mathcal{S}_{\text{quad}}$ of $un^2/2$ quadratic equations in k unknown.
4. Solve the system of quadratic equations by computing a Gröbner basis of $\mathcal{S}_{\text{quad}}$. This gives the two secrets S and T.

This method yields all the possible solutions of a given instance. Moreover, it succeeds with probability one, and is deterministic. At the beginning of stage 3, S and T^{-1} can be expressed as a linear combinaison of k elements. It is expected that the cost of the whole procedure is dominated by the last step, the complexity of which is difficult to predict in general. However, if all the equations of $\mathcal{S}_{\text{quad}}$ were linearly independent, and $k(k-1)/2 \leq u \cdot n^2/2$, then the equations of $\mathcal{S}_{\text{quad}}$ could be solved by linearization in time $\mathcal{O}\left(n^6\right)$ (and actually $\mathcal{O}\left(n^5\right)$ using more sophisticated sparse algebra subroutines). It most be noted that bigger values of u actually make the problem *easier*. In practice, if $\mathcal{S}_{\text{quad}}$ is that overdetermined, the computation of the Gröbner basis will also be polynomial (with roughly the same complexity) as reported in [25].

The heart of the meta-algorithm therefore lies in collecting independent linear equations, and the main difficulty in analyzing its complexity is estimating the rank of $\mathcal{S}$. This general strategy will be instantiated twice in this paper: for the quadratic IP1S problem in section 3.1 and for a particular case of the full IP problem in section 4.2.

3 Isomorphism of Polynomials with One Secret

In this section, we investigate more particularly the IP1S problem; *i.e.*, given two sets of polynomials $\mathbf{a}$ and $\mathbf{b}$ the task is to find S such that:

$$\mathbf{b} = \mathbf{a} \circ S. \tag{2}$$

A consequence of the structural observation of section 2 is that solving the linear homogeneous case is sufficient to solve all the other ones (this observation was also present in [50]). It was pointed out in [49] that if there is only one quadratic polynomial, then the problem is easily solved in polynomial time (as quadratic forms admit a canonical representation, see [38]). We will therefore focus on the case of $u \geq 2$ quadratic polynomials, and on the case of one cubic polynomial. These problems being qualitatively pretty different, we study them separately.

3.1 Quadratic IP1S

The quadratic IP1S problem is a good candidate for the application of the general strategy described in section 2. The important idea that we will use throughout this paper is that by *differentiating* equation (2), it will be possible to collect linear equations between the coefficients of S and those of S^{-1}. Indeed, for all vectors $\mathbf{x}, \mathbf{y} \in (\mathbb{F}_q)^n$, we have:

$$\forall \mathbf{x}, \mathbf{y} \in (\mathbb{F}_q)^n, \quad \mathrm{D}\mathbf{b}(\mathbf{x}, \mathbf{y}) = \mathrm{D}\mathbf{a}\left(S \cdot \mathbf{x}, S \cdot \mathbf{y}\right).$$

Using the change of variable $\mathbf{y}' = S \cdot \mathbf{y}$, this equation becomes:

$$\forall \mathbf{x}, \mathbf{y}' \in (\mathbb{F}_q)^n, \quad \mathrm{D}\mathbf{b}(\mathbf{x}, S^{-1} \cdot \mathbf{y}') = \mathrm{D}\mathbf{a}(S \cdot \mathbf{x}, \mathbf{y}'). \tag{3}$$

When $\mathbf{a}$ and $\mathbf{b}$ are of total degree 2, D$\mathbf{a}$ and D$\mathbf{b}$ are *bilinear* (symmetric) mappings. In this case, since equation (3) is valid for all $\mathbf{x}$ and $\mathbf{y}$, then in particular it is valid on a basis of $(\mathbb{F}_q)^n \times (\mathbb{F}_q)^n$, and substituting fixed basis vectors for $\mathbf{x}$ and $\mathbf{y}$ yields *linear equations* between the coefficients of S and those of S^{-1}. This idea for obtaining linear equations can also be described relatively simply using the usual theory of quadratic forms.[3] If $\mathbb{F}_q$ is a field of odd (resp. even) characteristic, then the set of homogeneous quadratic polynomials is in one-to-one correspondance with the set of symmetric matrices (resp. with zero diagonal). Let us assume odd characteristic for now, and let $\mathcal{P}(\mathbf{a}_k)$ $\big($resp. $\mathcal{P}(\mathbf{b}_k)\big)$ denote the matrix of the bilinear form associated with $\mathbf{a}_k$ (resp $\mathbf{b}_k$), which is sometimes called the *polar form* of $\mathbf{a}_k$. Recall that the coefficient of index (i, j) of $\mathcal{P}(\mathbf{a}_k)$ is $\mathrm{D}\mathbf{a}_k(e_i, e_j)/2$, where $(e_i)_{1\leq i\leq n}$ is a basis of $(\mathbb{F}_q)^n$. We then have:

$$\mathcal{S}: \quad \begin{cases} S^{-1}\cdot\mathcal{P}(\mathbf{b}_1) = \mathcal{P}(\mathbf{a}_1)\cdot {}^tS \\ \qquad\vdots \\ S^{-1}\cdot\mathcal{P}(\mathbf{b}_u) = \mathcal{P}(\mathbf{a}_u)\cdot {}^tS \end{cases} \tag{4}$$

It is worth noticing that these u matrix equations also hold in characteristic two, if we remove the division by two in the definition of the polar form. Each such matrix equation yields n^2 linear homogeneous equations between the $2n^2$ coefficients of S and those of S^{-1}. These last $u\cdot n^2$ linear equations cannot be linearly independent as they admit a non-trivial solution $\big(S^{-1}, S\big)$. The kernel of $\mathcal{S}$ is thus non-trivial, and our hope would be that it describes only one solution. When u is strictly greater than two, this is empirically always the case. When $u = 2$, however, the situation is not as nice; Theorem 1 below shows that the kernel of $\mathcal{S}$ is of dimension higher that n. This means that solving the linear equations cannot by itself solve the problem, because they admit at least q^n solutions, out of which only very few are actual solutions of the IP1S instance. This contradicts the hope expressed in section 9 of [49]. However, the linear equations collected this way can be used in the context of our meta-algorithm described in section 2. We now report on the efficiency of doing so.

Practical Behavior of the Algorithm. We implemented an instanciation of the meta-algorithm of section 2 that collects the linear equations described by (4) using the computer algebra system MAGMA [10]. Solving the equations of $\mathcal{S}_{\text{quad}}$ is achieved by first computing a Gröbner basis of these equations for the Graded-Reverse Lexicographic order using the F$_4$ algorithm [21], and then converting it to the Lexicographic order using the FGLM algorithm [20]. This implementation breaks all the proposed quadratic IP1S challenges in negligible time, less than 30 seconds for the biggest one. To illustrate its effectiveness, we built and broke a random IP1S instance with $q = 2$, $u = 2$ and $n = 32$ in a matter of seconds (recall that the biggest proposed quadratic IP1S challenge, and never broken before was $q = u = 2$ and $n = 16$). The dominating part in the execution of the algorithm is in fact the symbolic manipulation of polynomials required to write down the equations of $\mathcal{S}_{\text{quad}}$. Actually solving the resulting quadratic equations turns out to be easier than generating them.

Complexity Analysis. As a foreword, we would like to point out that the algorithm empirically works better than what the analysis suggests. The reason is that we were only able to get a precise bound on the rank of $\mathcal{S}$ under an hypothesis on the instance: we require one of the two quadratic forms (say, $\mathbf{a}_1$) to be non-degenerate, and we discuss below that this happens with a non-negligible probability in the random case. However, if both quadratic forms are of rank, say, $n-1$, then the algorithm still works pretty well, even though our analysis says nothing about it. Our complexity argument boils down to showing that the kernel of $\mathcal{S}$ is of sufficiently small dimension so that computing a solution of $\mathcal{S}_{\text{quad}}$ is essentially a linearization. Our result is expressed in terms of the *similarity invariants*$P_1, \ldots, P_s$ of a matrix M. Their product is the characteristic polynomial of M, p_s is the minimal polynomial of M, and P_i divides P_{i+1}.

Theorem 1. *Let A_1, A_2, B_1, B_2 be four given matrices of size $n\times n$ with coefficients in $\mathbb{F}_q$. Let us consider the set of all pairs (X, Y) of $n\times n$ matrices satisfying the following linear equations:*

$$\mathcal{S}: \quad \begin{cases} B_1 = X\cdot A_1\cdot Y \\ B_2 = X\cdot A_2\cdot Y \end{cases}$$

Let us assume that $\mathcal{S}$ admits at least one solution (S_0, Y_0), and that A_1 is invertible.

i) There is a vector-space isomorphism between the kernel of $\mathcal{S}$ and the commutant of $\mathcal{C} = A_2\cdot A_1^{-1}$.
ii) $n \leq \dim\ker\mathcal{S}$.
iii) Let $P_1, \ldots, P_s$ be the similarity invariants of $\mathcal{C}$. Then:

$$\dim\ker\mathcal{S} = \sum_{j=1}^{s}(2s-2j+1)\cdot\deg P_j$$

This theorem (proven in annex A.2) directly apply to our study of the quadratic IP1S problem with $A_i = \mathcal{P}(\mathbf{a}_i)$ and $B_i = \mathcal{P}(\mathbf{b}_i)$.

[3] This highlight the fact that the IP1S problem is a bit different in odd and even characteristic.

Application of Theorem 1 to the Random Case. Theorem 1 holds only if $\mathcal{P}(\mathbf{a}_1)$ or $\mathcal{P}(\mathbf{a}_2)$ is invertible (we may swap them if we wish). If q is odd, then $\mathcal{P}(\mathbf{a}_1)$ is a random symmetric matrix, and the probability that it has a given rank is given by Lemma 9 (found in annex B). The probability that $\mathcal{P}(\mathbf{a}_1)$ is invertible is about 0.639 for $q = 3$, and is a (rapidly) increasing function of q. The probability that either $\mathcal{P}(\mathbf{a}_1)$ or $\mathcal{P}(\mathbf{a}_2)$ is invertible is about 0.870 for $q = 3$. If q is even, then $\mathcal{P}(\mathbf{a}_1)$ is a random symmetric matrix with zeros on the diagonal, and the probability that it has a given rank is provided by Lemma 10 (also found in annex B). The probability that $\mathcal{P}(\mathbf{a}_1)$ is invertible if $q = 2$ is about 0.419 (again, this probability increase exponentially with q). The probability that either $\mathcal{P}(\mathbf{a}_1)$ or $\mathcal{P}(\mathbf{a}_2)$ is invertible is about 0.662 for $q = 2$.

Theorem 1 is then applicable in more than half of the cases. When it is applicable, what guarantee does it exactly offer? We would need to know something about the invariant factors of $\mathcal{C}$. An easy case would be when the minimal and characteristic polynomials are the same (then there is only one invariant factor, and it is $\chi_{\mathcal{C}}$). Then Theorem 1 tells us that the dimension of $\ker \mathcal{S}$ is n. The probability of this event is given by lemma 11 (found in annex B): for random matrices over $\mathbb{F}_2$, and for n big enough, the proportion of cyclic matrices approaches 0.746. Unfortunately, in even characteristic, $\mathcal{C}$ is *never* cyclic:

Lemma 2. *Let q be even. If n is even, then the characteristic polynomial of $\mathcal{C}$, $\chi_{\mathcal{C}}$, is a square. If n is odd, then the square-free part of $\chi_{\mathcal{C}}$ is X.*

In characteristic two, the minimal polynomial of $\mathcal{C}$ is therefore of degree at most $n/2$, which means that there are *at least* two similarity invariants. Therefore the kernel of $\mathcal{S}$ is in fact of dimension *at least* $2n$. What we would need to know is the probability that it is precisely of dimension $2n$.

We could not compute precisely this probability, but we measured experimentally that it is about 0.74 for $q = 2$ (this value is strikingly close to the proportion of cyclic random matrices...). In this setting, Theorem 1 guarantees that the dimension of $\ker \mathcal{S}$ is exactly $2n$. This is not completely sufficient to guarantee that we will be able to solve the equations of $\mathcal{S}_{\text{quad}}$ by linearization (because we would have $2n^2$ linear equations in $4n^2$ variables). However, it creates a system of quadratic equations so overdefined that the maximal degree of polynomials reached when computing a Gröbner basis appears to be always upper-bounded by 2 in practice. The computation of the Gröbner basis therefore terminates in time $\mathcal{O}\left(n^6\right)$ in most cases.

3.2 Cubic IP1S

In this section, we focus on the case where $\mathbf{a}$ and $\mathbf{b}$ are composed of a single cubic polynomial. We assume that $\mathbf{a}$ and $\mathbf{b}$ are given explicitly, i.e.:

$$\mathbf{a} = \sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j}^{n} A_{i,j,k} \cdot x_i x_j x_k, \qquad \mathbf{b} = \sum_{i=1}^{n}\sum_{j=i}^{n}\sum_{k=j}^{n} B_{i,j,k} \cdot x_i x_j x_k.$$

As already explained, we can restrict our attention to the homogenous case. The techniques developed previously for the quadratic case cannot directly applied in this setting. Indeed, the differential is no longer a bilinear mapping, and then there is no obvious linear equations between the coefficients of a solution and those of its inverse. However, we can combine the use of the differential together with the Gröbner basis approach proposed in [25]. We denote by $S_0 = \{s^0_{i,j}\}_{1 \le i,j \le n}$ a particular solution of IP1S between $\mathbf{a}$ and $\mathbf{b}$, *i.e.*, it holds that $\mathbf{b} = \mathbf{a} \circ S_0$. For all vectors $\mathbf{x}, \mathbf{y} \in (\mathbb{F}_q)^n$, we have:

$$\mathrm{D}\mathbf{a}(S_0 \cdot \mathbf{x}, \mathbf{y}) = \mathrm{D}\mathbf{b}(\mathbf{x}, S_0^{-1} \cdot \mathbf{y}).$$

$\mathbf{a}$ and $\mathbf{b}$ being of total degree 3, the coefficients of S_0 and S_0^{-1} appear with degree two in the expression of $\mathrm{D}\mathbf{a}$ and $\mathrm{D}\mathbf{b}$ above. Let R be the ring $\mathbb{K}[s_{1,1}, \ldots, s_{n,n}, u_{1,1}, \ldots, u_{n,n}]$. We consider the algebra $\mathcal{A}^s$ of all $n \times n$ matrices over R. Let $S = \{s_{i,j}\}$ and $U = \{u_{i,j}\}$ in $\mathcal{A}^s$ be symbolic matrices. We denote by $\mathcal{I}_{\mathbf{a},\mathbf{b}}$ the ideal generated by all the coefficients in R of the equations:

$$\mathrm{D}\mathbf{a}(S \cdot \mathbf{x}, \mathbf{y}) - \mathrm{D}\mathbf{b}(\mathbf{x}, U \cdot \mathbf{y}) = 0, \qquad U \cdot S - 1_n = 0_n, \qquad S \cdot U - 1_n = 0_n.$$

It is easy to see that $U = S_0^{-1}$ and $S = S_0$ is particular solution of this system, and also a solution of IP1S between $\mathbf{b}$ and $\mathbf{a}$. Our goal is to provide an upper bound on the maximum degree reached during a Gröbner basis computation of $\mathcal{I}_{\mathbf{a},\mathbf{b}}$. This degree, called *degree of regularity*, is the key parameter in establishing the complexity of the Gröbner basis computation. Indeed, the cost of computing a Gröbner basis is polynomial in the degree of regularity D_{reg} of the system[4] considered: $\mathcal{O}(N^{\omega \cdot D_{\text{reg}}})$, with $2 < \omega \le 3$ the linear algebra constant, and N the number of variables of system considered. In our case, $N = n^2$. The behavior of the degree of regularity is well understood for "random" systems of equations [4,5,6] (*i.e.*, regular or semi-regular systems). On the other hand, as soon as the system has some kind of structure, which is always the case in cryptography, this degree is much more difficult to predict. In some particular cases, it is however possible to bound the degree of regularity (see for HFE [24,34]). We prove here that $D_{\text{reg}} = 2$ for $\mathcal{I}_{\mathbf{a},\mathbf{b}}$ under the hypothesis that we know one row of a particular solution S_0, *i.e.*, we assume then that we know the following ideal $\mathcal{J} = \left\langle s_{1,j} - s^{(0)}_{1,j} \mid j = 1, \ldots, n \right\rangle$.

[4] This is true in the zero-dimensional case, *i.e.*, when the number of solutions is finite.

Theorem 2. *The degree of regularity of $\mathcal{I}_{\mathbf{a},\mathbf{b}} + \mathcal{J}$ is 2. Therefore, computing a Gröbner basis of this ideal takes time $\mathcal{O}\left(n^6\right)$.*

Application to the Linear Inhomogeneous Case. If S can be assumed to be a linear bijective mapping, and if $\mathbf{a}$ has a non-trivial homogeneous component of degree 1, then we are in a situation where theorem 2 is applicable, and S can be determined though a Gröbner basis computation which terminates in time $\mathcal{O}\left(n^6\right)$.

Application to the Other Cases. All the other cases reduce to the linear homogeneous case, as mentionned in section 2. In this setting, the problem is that we do not have enough knowledge on S to make the Gröbner basis computation efficient. A simple idea would be to guess a colum of S then compute the Gröbner basis. This approach has complexity $\mathcal{O}\left(n^6 \cdot q^n\right)$ as explained before. The biggest proposed cubic IP1S challenge has $u = 1$, $n = 16$ and $q = 2$. Given one relation on S, the computation of the Gröbner basis takes 90 seconds on a 2.8Ghz Xeon computer. Since this has to be repeated 2^{16} times, the whole process takes about two months.

The worst case complexity can be further improved using an idea of [50]. Indeed, according to the author:

$$\mathbf{b} = \mathbf{a} \circ S \Longrightarrow J_{\mathbf{b}}(\mathbf{x}) = S^T \circ J_{\mathbf{a}}(S \cdot \mathbf{x}),$$

$J_{\mathbf{b}}(\mathbf{x})$ denoting the Jacobian matrix of $\mathbf{b}$ in $\mathbf{x}$ (*i.e.*, whose components are partial derivatives of the polynomials of $\mathbf{b}$), and $J_{\mathbf{a}}(S \cdot \mathbf{x})$ the Jacobian matrix of $\mathbf{a}$ in $S \cdot \mathbf{x}$. We have then a particular instance of the general IP-problem addressed in the next sections. In particular, a probabilistic technique is described in Section 5.2 allowing to find one right pair $\mathbf{y} = S \cdot \mathbf{x}$ in time $\mathcal{O}\left(q^{n/2}\right)$, leading to a complexity of $\mathcal{O}\left(n^6 \cdot q^{n/2}\right)$ for the cubic IP1S algorithm.

Miscellaneous Remarks. We conclude this section with a simple idea that could have lead to an improvement, by efficiently giving a relation on $S\ell$, but which fails surprisingly. Let us denote by $Z_{\mathbf{a}}$ (resp. $Z_{\mathbf{b}}$) the set of zeroes of $\mathbf{a}$ (resp. $\mathbf{b}$). Because of lemma 1, and since S is linear, we have:

$$S_\ell \left(\sum_{\mathbf{x} \in Z_{\mathbf{a}}} \mathbf{x} \right) = \sum_{\mathbf{y} \in Z_{\mathbf{b}}} \mathbf{y}$$

This yields a relation on S_ℓ, which is enough to use theorem 2. $\mathbf{a}$ and $\mathbf{b}$ may be assumed to have about q^{n-1} zeroes. Finding them requires time $\mathcal{O}\left(q^n\right)$. The complexity of the attack could thus be improved to $\mathcal{O}\left(n^6 + q^n\right)$. Surprisingly, this trick fails systematically, and this happen to be consequence of the Chevalley-Warning theorem [14,53].

Lemma 3. *The sum of the zeroes of a cubic form on 5 variables or more over $\mathbb{F}_q$ is always zero.*

4 Isomorphisms of Polynomials with Two Secrets

In this section we focus on the full IP problem (with two secret affine or linear mappings S and T). Recall that finding a polynomial isomorphism between two vectors of multivariate polynomials $\mathbf{a}$ and $\mathbf{b}$ means finding two matrices S_ℓ and T_ℓ, as well as two vectors S_c and T_c such that:

$$\mathbf{b}\left(\mathbf{x}\right) = T_c + T_\ell \left(\mathbf{a}\left(S_\ell \cdot x + S_c\right)\right). \tag{5}$$

Similarly to the IP1S problem, the hardness of the unrestricted IP problem depends on the parity of the characteristic of the field. As opposed to the IP1S problem though, it also depends pretty much on whether $u = n$, whether the polynomials are homogeneous and whether S and T are linear or affine. We first present two polynomial algorithms that solve the easiest case, namely the linear homogeneous IP problem with as many polynomial as variables, for which the Gröbner-based algorithm of [25] is also polynomial. While the latter runs in time $\mathcal{O}\left(n^9\right)$, a new heuristic algorithm runs in time $\mathcal{O}\left(n^3\right)$, and a more rigorous variation runs in time $\mathcal{O}\left(n^6\right)$.

4.1 The Linear Inhomogeneous Case

We first present a simple heuristic which combine the structural observation of section 2 and the to-and-fro approach [49]. As before, we will denote by $\mathbf{a}^{(1)}$ the homogeneous component of degree one of $\mathbf{a}$ (*i.e.*, the linear terms of $\mathbf{a}$). The application of lemma 1 immediately yields:

$$\mathbf{b}^{(1)} = T_\ell \cdot \mathbf{a}^{(1)} \cdot S_\ell \tag{6}$$

An important observation is the following: in the linear inhomogeneous case we have *a priori* knowledge on T_ℓ, since $T_\ell \cdot \mathbf{a}(0) = \mathbf{b}(0)$. In some cases this knowledge can be "transferred" to S_ℓ using an "obvious trick": assume that $\mathbf{b}(0) = \mathbf{b}^{(1)} \cdot \mathbf{x}$ and $\mathbf{a}(0) = \mathbf{a}^{(1)} \cdot \mathbf{y}$. Then $\mathbf{y} = S_\ell \cdot \mathbf{x}$. Note that because of equation (6) the two necessary condition for the obvious trick to work are in fact the same. When these conditions are satisfied, we say that the obvious trick succeeds.

If $\mathbf{a}^{(1)}$ and $\mathbf{b}^{(1)}$ are invertible and if $\mathbf{a}(0) \neq 0$, then the obvious trick always succeeds, and it is possible to use it iteratively, as shown in figure 2. The complexity of this algorithm is $\mathcal{O}\left(n^3\right)$. Inverting both $\mathbf{a}^{(1)}$ and $\mathbf{b}^{(1)}$ can be done once for all. The

Fig. 2 A variant of the “To-and-Fro” IP algorithm using linear terms to go back

```
$x_1 \leftarrow \mathbf{a}(0)$
$y_1 \leftarrow \mathbf{b}(0)$
for $i = 1$ to $n$ do // At this point one has $y_i = T_\ell \cdot x_i$
    $y'_i \leftarrow \left(\mathbf{a}^{(1)}\right)^{-1} \cdot x_i$
    $x'_i \leftarrow \left(\mathbf{b}^{(1)}\right)^{-1} \cdot y_i$
    // And we obtain $y'_i = S_\ell \cdot x'_i$
    $y_{i+1} \leftarrow \mathbf{b}\left(x'_i\right)$
    $x_{i+1} \leftarrow \mathbf{a}\left(y'_i\right)$
end for
Reconstruct $S_\ell$ from the pairs $(x'_i, y'_i)$ and $T_\ell$ from the pairs $(x_i, y_i)$.
```

matrix-vector products take $\mathcal{O}\left(n^2\right)$ and there are n of them. Lastly, reconstructing S_ℓ and T_ℓ takes only $\mathcal{O}\left(n^3\right)$, because in the basis $(x_i)_{i \leq n}$, S is made of the y_i's. Changing the basis amounts to performing one matrix inversion and two matrix-matrix products.

This heuristic works well for random inhomogeneous instances (*i.e.*, instances where all the coefficients of all degrees are randomly chosen). It is straightforward that the number of $n \times n$ invertible matrices over $\mathbb{F}_q$ is $\prod_{i=0}^{n-1}\left(q^n - q^i\right)$. This tells us that the probability that $\mathbf{a}^{(1)}$ is invertible is about 0.288 when $q = 2$ (higher for bigger q) and the probability that $\mathbf{a}(0) \neq 0$ is $1 - 1/q^n$. Again, $q = 2$ looks like a worst case. For all realistic sets of parameters, the heuristic either fails or terminates in less than a second, even for parameters that were taking several minutes to the Gröbner-based algorithm of [25].

To highlight the danger of relying on heuristic assumptions when reasoning about IP algorithms, we point out that it is quite easy to cook up an instance of the IP problem *satisfying the two conditions stated above* but on which this algorithms fails completely, since these conditions are necessary but not sufficient.

Suppose that $\mathbf{a} = T_\ell \circ \mathbf{a} \circ S_\ell$. Such an example in fact came up naturally when studying a variation of HFE [11]. Our initial “bootstrapping” relation $T_\ell \cdot \mathbf{a}(0) = \mathbf{a}(0)$ in fact describes an eigenvector $\mathbf{x}$ satisfying the equation $T_\ell \cdot \mathbf{x} = \mathbf{x}$. Thanks to equation (6), this relation is transferred to an eigenvector of S_ℓ satisfying the equation $S_\ell \cdot \mathbf{y} = \mathbf{y}$ (with $\mathbf{y} = {\mathbf{a}^{(1)}}^{-1} \cdot \mathbf{x}$). Now, using the easy and natural way to produce new relations on T_ℓ, we obtain other eigenvectors of T_ℓ satisfying the same eigenvalue equation: $T_\ell \cdot \mathbf{a}(\mathbf{y}) = \mathbf{a}(\mathbf{y})$. The number of independent linear relations that we may accumulate this way is upper-bounded by the dimension of the eigenspace of T_ℓ for the eigenvalue 1. This eigenspace cannot span the whole $\left(\mathbb{F}_q\right)^n$ (otherwise T_ℓ would be the identity matrix). The heuristic thus cannot fully determine T_ℓ, as after a given point, all the new linear relations found at each iteration will be linearly dependent from the previous ones.

4.2 A More Robust Algorithm.

We propose a new, “rigorous” algorithm, which albeit less efficiently, is easy to analyze. It is another instanciation of the meta-strategy of section 2, and it also collects linear equations by differentiating equation (5). The starting point is the fact that equation (6) provides n^2 linear relations between the coefficients of S_ℓ and those of T_ℓ^{-1}. This is not enough to recover the two matrices by linear algebra, since there are $2n^2$ unknowns. However, if a pair $(\mathbf{x}, \mathbf{y})$ such that $\mathbf{y} = S_\ell \cdot \mathbf{x}$ were to be known, it would allow us to differentiate equation (5), and this would yield:

$$\frac{\partial \mathbf{b}}{\partial \mathbf{x}} = T_\ell \cdot \frac{\partial \mathbf{a}}{\partial \mathbf{y}} \cdot S_\ell. \tag{7}$$

This would provide the n^2 new linear equations between the coefficients of S_ℓ and T_ℓ^{-1} that were required. The problem thus comes down to finding such a relation $(\mathbf{x}, \mathbf{y})$ on S_ℓ. This relation can in fact be readily acquired from the public key if the obvious trick mentionned above succeeds. Again, analyzing this algorithm reduces to estimate the rank of the following system of linear equations:

$$\mathcal{S}: \quad \begin{cases} \mathbf{b}^{(1)} = T_\ell \cdot \mathbf{a}^{(1)} \cdot S \\ \frac{\partial \mathbf{b}}{\partial \mathbf{x}} \ = \ T_\ell \cdot \frac{\partial \mathbf{a}}{\partial \mathbf{y}} \cdot S \end{cases} \tag{8}$$

Our main tool is again theorem 1. The matrix $\mathbf{b}^{(1)}$ is completely random, thus it is invertible with probability $\prod_{i=0}^{n-1}\left(1 - q^{i-n}\right)$ (greater than 0.288 for $q = 2$). The following lemma summarizes the result.

Lemma 4. *Let p be the probability that $\dim \ker \mathcal{S} \leq n + 2$ conditionned on $\mathbf{b}^{(1)}$ being invertible. Then:*

$$p \geq 1 - \frac{1}{\left(q^2 - 1\right)\left(q - 1\right)}, \qquad \text{and} \qquad \lim_{n \to \infty} p = \frac{q^5 - 1}{q^2(q-1)(q^2-1)} \cdot \prod_{i=1}^{\infty}\left(1 - \frac{1}{q^i}\right)$$

4.3 The Affine and/or Homogeneous Case

As pointed out in section 2, the affine case reduces to linear homogeneous case. Unfortunately, the two methods discussed in the previous section cannot be applied in the homogeneous case. It is therefore natural to try to "de-homogenize" the problem. This requires *one* known relation on S_ℓ: if we know $y_0 = S_\ell \cdot x_0$, then we define $\mathbf{a}'(\mathbf{x}) = \mathbf{a}(\mathbf{x} + y_0)$ and $\mathbf{b}'(\mathbf{x}) = \mathbf{b}(\mathbf{x} + x_0)$, and we have $\mathbf{b}' = T \circ \mathbf{a}' \circ S_\ell$. It follows that:

$$\mathbf{a}'^{(2)} = \mathbf{a}^{(2)} \qquad\qquad \mathbf{a}'^{(1)} = \frac{\partial \mathbf{a}}{\partial y_0}$$

A first consequence is that the linear component of the de-homogenized version of the instance is never invertible in even characteristic. This makes the use of the algorithms of section 4.1 tricky, and no complexity analysis is readily available. There is also a further difficulty. The rigorous algorithm of section 4.2 requires a relation on $\mathcal{S}_\ell$ to differentiate equation (5). Using the de-homogenization relation for this purpose is not possible, as it would yield $y_0 = S_\ell \cdot x_0$, $\mathbf{b}^{(1)} = \frac{\partial \mathbf{b}}{\partial x_0}$, and the two equations of (8) would in fact be the same.

The obvious trick could still be used, however $\mathbf{a}'^{(1)}$ will never be invertible in characteristic two. There is however a non-zero probability p_{OT} that the obvious trick can still be applied. If $\mathbf{b}^{(1)}$ has rank $n - k$, then a random vector has probability q^{-k} to be in the range of $\mathbf{b}^{(1)}$. Using lemma 12 we see that in characteristic two, $p_{OT} = \sum_{i=1}^{n} \alpha_{n,i} \cdot q^{-i^2}$. If $q = 2$, we find that $p_{OT} = 0.390$.[5] Therefore, in even characteristic, there is a pretty non-negligible chance that *two* independent relations on S_ℓ are in fact needed for a complete resolution. The problem is then to acquire the initial relation(s) on S_ℓ. Below, and in the next section, we discuss three techniques for decreasing asymptotic complexity. For the sake of simplicity, our discussion focuses on the case $q = 2$, but it is nearly always a worst case.

Finding Common Zeroes. Lemma 1 tells us that S_ℓ transforms the zeroes of $\mathbf{a}$ into those of $\mathbf{b}$. If there are k of them, then there are $\binom{k}{2}$ ways to match them to obtain two relations on S. The algorithm of section 4.2 has to be run for each combination. It is known [29] that a random system of n polynomials in n variables has exactly $s \geq 0$ common zeroes with probability $1/(e \cdot s!)$. Finding the zeroes of $\mathbf{a}$ and $\mathbf{b}$ thus permits to find at least one relation on S_ℓ with probability 0.632, and at least two with probability 0.264. If only one relation has been found, the obvious trick might still apply, and all in all the success probability of this method is:

$$p_{CZ} = 1 - \frac{2}{e} + \frac{p_{OT}}{e}$$

Thus, if $q = 2$, then about 40% of the random instances can be solved in polynomial time *after the zeroes have been found*. Solving quadratic equations is typically exponential, but the interest of the method lies in the fact that the constant factors in the complexity are small, and that in practice, this may be a realistic method. Depending on the value of q, it will be faster using Gröbner bases (big q), or via exhaustive search (small q). For $q = 2$, a GPU-based implementation of exhaustive search solves 48 equations in 48 variables in 30 minutes on a \$500 GPU (an Nvidia GT295). Thus the challenge with $q = 2$, $u = n = 64$ has a 40% chance to be breakable in less than 4 GPU-year with inexpensive hardware.

5 Probabilistic Algorithm for IP with Two Secrets

We consider here a probabilistic approach to solve the homogeneous quadratic IP problem. The main idea of these algorithms is to find two "right pairs" $(\mathbf{x}, \mathbf{y})$ such that $\mathbf{y} = S_\ell \cdot \mathbf{x}$, and then use the previous deterministic algorithms to find the complete solution. To break the $\mathcal{O}(q^n)$ bound, a natural idea is to use the birthday paradox, and indeed the two algoithms described below find these right pairs by finding collisions between two subsets of $(\mathbb{F}_q)^n$.

5.1 A Rank-Based Approach

To find the right pairs, we use the fact that if $\mathbf{y} = S_\ell \cdot \mathbf{x}$, then the rank of $\frac{\partial \mathbf{b}}{\partial x}$ is equal to the rank of $\frac{\partial \mathbf{a}}{\partial y}$ (this is a consequence of lemma 1). The idea is to restrict our attentions to the subset of $(\mathbb{F}_q)^n$ over which the differential has a sufficienlty small rank. If this subset is small enough, the set of pairs $(\mathbf{x}, \mathbf{y})$ can also become sufficiently small to be exhaustively testable.
If f is a quadratic map from $(\mathbb{F}_q)^n$ to $(\mathbb{F}_q)^n$, then let $R_k(f)$ denote the subset of $(\mathbb{F}_q)^n$ formed of points over which the kernel of the differential has dimension k. If the pair $(\mathbf{a}, \mathbf{b})$ has been chosen randomly, then lemma 12 grants us that when q is even:

$$\mathbb{E}\big[|R_k(f)|\big] = \alpha_{n,k} \cdot q^{n-k(k-1)}$$

where $\alpha_{n,k} \in [0.16; 3.58]$ if $q = 2$. We may expect $|R_k|$ to be non-empty if $k \leq \frac{1}{2} + \sqrt{n+3}$. Our algorithm depends on two parameters ℓ and k, and is shown in figure 3.

[5] We are aware that there are other means to get a new relation on S from a first one. Let us mention for instance the fact that if ker $\frac{\partial \mathbf{a}}{\partial \mathbf{y}}$ is of dimension greater than one, other relations might be found easily.

Fig. 3 Rank/Birthday Based Algorithm

```
$L_{\mathbf{a}} \leftarrow \ell$ random vectors from $R_k(\mathbf{a})$.
$L_{\mathbf{b}} \leftarrow \ell$ random vectors from $R_k(\mathbf{b})$.
for all $(x, y) \in L_{\mathbf{a}} \times L_{\mathbf{b}}$ do
    $(\mathbf{a}', \mathbf{b}') \leftarrow$ DEHOMOGENIZE($\mathbf{b} = T \circ \mathbf{a} \circ S,$ using the assumption that $y = S \cdot x$).
    if obvious trick succeeds on $(\mathbf{a}', \mathbf{b}')$ then
        SOLVE($\mathbf{b} = T \circ \mathbf{a} \circ S$) // using the algorithm of section 4.2
        if solution is found then
            Abort the loop, report success.
        end if
    end if
end for
Report failure.
```

We say that a *right pair* (x, y) in $L_{\mathbf{a}} \times L_{\mathbf{b}}$ is such that $y = S_\ell \cdot x$. The algorithm succeeds if there is a right pair, and if the obvious trick succeds in the corresponding iteration. The birthday paradox tells us that there is a right pair with probability higher than $1/2$ if $\ell = \sqrt{|R_k|}$. This follows from the fact that if (x, y) is a right pair, then $\frac{\partial \mathbf{a}}{\partial y}$ and $\frac{\partial \mathbf{b}}{\partial x}$ have the same rank (this itself follows from lemma 1). We must make sure to have $1/p_{OT}$ right pairs, so we choose: $\ell = 1/p_{OT} \cdot q^{\frac{n-k(k-1)}{2}}$.

The two costly steps in the algorithm are the generation of the lists, and the $\ell^2 \cdot P_{OT}$ calls to SOLVE. To generate the lists, we randomly sample from $(\mathbb{F}_q)^n$ until the kernel of the differential is k, and we repeat the process until ℓ vectors have been accumulated. Lemma 12 tells us that $\alpha_{n,k} \cdot q^{-k(k-1)}$ random trials are necessary in average to find one element of the lists. Building the lists thus cost $\ell \cdot q^{k(k-1)}$ operations on $n \times n$ matrices (neglecting the small $\alpha_{n,k}$ constant). The optimal value of k is therefore: $k = 1/2 + \sqrt{n/3 + 1/4}$. The complexity of the algorithm is then essentially the cost of $1/p_{OT} \cdot q^{2n/3}$ calls to SOLVE, which makes $\mathcal{O}\left(n^6 \cdot q^{2n/3}\right)$.

5.2 A Branching Process/Birthday Approach.

To improve the previous approach, we try to define a more precise criterium to detect right pairs. So far, we used the fact that if (x, y) is a right pair, then $\frac{\partial \mathbf{a}}{\partial y}$ and $\frac{\partial \mathbf{b}}{\partial x}$ have the same rank. It is possible to go one step further, by noticing that the following *multiset* equality follows from lemma 1, item v:

$$\left\{\!\!\left\{ \operatorname{rank} \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \,\middle|\, \mathbf{z} \in \ker \frac{\partial \mathbf{a}}{\partial \mathbf{y}} \right\}\!\!\right\} = \left\{\!\!\left\{ \operatorname{rank} \frac{\partial \mathbf{b}}{\partial \mathbf{z}} \,\middle|\, \mathbf{z} \in \ker \frac{\partial \mathbf{b}}{\partial \mathbf{x}} \right\}\!\!\right\}$$

It is even possible to go down one level with nested multisets:

$$\left\{\!\!\left\{ \left\{\!\!\left\{ \operatorname{rank} \frac{\partial \mathbf{a}}{\partial \mathbf{t}} \,\middle|\, \mathbf{t} \in \ker \frac{\partial \mathbf{a}}{\partial \mathbf{z}} \right\}\!\!\right\} \,\middle|\, \mathbf{z} \in \ker \frac{\partial \mathbf{a}}{\partial \mathbf{y}} \right\}\!\!\right\} = \left\{\!\!\left\{ \left\{\!\!\left\{ \operatorname{rank} \frac{\partial \mathbf{b}}{\partial \mathbf{t}} \,\middle|\, \mathbf{t} \in \ker \frac{\partial \mathbf{b}}{\partial \mathbf{z}} \right\}\!\!\right\} \,\middle|\, \mathbf{z} \in \ker \frac{\partial \mathbf{b}}{\partial \mathbf{x}} \right\}\!\!\right\}$$

It should be clear that is would be possible to go as "deep" as we want by nesting more levels of multisets. In fact, these multisets contain some redundant information: if $\mathbf{z} \in \ker \frac{\partial \mathbf{a}}{\partial \mathbf{y}}$, then we know that $\mathbf{y} \in \frac{\partial \mathbf{a}}{\partial \mathbf{z}}$. To avoid this redundancy, we define:

$$\textsc{Root}(f, \mathbf{x}) = \left\{\!\!\left\{ \textsc{Leaf}(f, \mathbf{x}, \mathbf{t}) \,\middle|\, \mathbf{t} \in \ker \frac{\partial f}{\partial \mathbf{x}} - \mathit{Span}(\mathbf{x}) \right\}\!\!\right\}$$
$$\textsc{Leaf}(f, \mathbf{x}, \mathbf{z}) = \left\{\!\!\left\{ \textsc{Leaf}(f, \mathbf{z}, \mathbf{t}) \,\middle|\, \mathbf{t} \in \ker \frac{\partial f}{\partial \mathbf{z}} - \mathit{Span}(\mathbf{z}, \mathbf{x}) \right\}\!\!\right\}$$

This recursive definition allows us to associate to each point of $(\mathbb{F}_q)^n$ an unranked, unordered, potentially infinite tree such that if $(\mathbf{x}, \mathbf{y})$ is a right pair, then $\textsc{Root}(\mathbf{b}, \mathbf{x}) = \textsc{Root}(\mathbf{a}, \mathbf{y})$. Let $H_k(f)$ be the set of points $\mathbf{x}$ of $(\mathbb{F}_q)^n$ on which $\textsc{Root}(f, \mathbf{x})$ had depth at least k. Our last algorithm is shown in figure 4. Its analysis requires that we investigate a bit the properties of the "trees" associated to each point of $(\mathbb{F}_q)^n$. For this purpose, we need an extension of lemma 12.

Lemma 5. *Let $\mathbb{F}_q$ be a field of characteristic two. Let $f : (\mathbb{F}_q)^n \rightarrow (\mathbb{F}_q)^n$ be a random quadratic map, and $\mathbf{x}$ be a non-zero element of $(\mathbb{F}_q)^n$ such that there exist $\mathbf{y} \in \left(\ker \frac{\partial f}{\partial \mathbf{x}}\right) - \mathit{Span}(\mathbf{x})$. The rank of $\frac{\partial f}{\partial \mathbf{y}}$ follows the distribution of ranks of endomorphisms of $(\mathbb{F}_q)^n$ vanishing at $\mathbf{x}$ and $\mathbf{y}$. Therefore, for any t in $[2, n]$ the probability that $\frac{\partial f}{\partial \mathbf{y}}$ has rank $n - t$ is:*

$$p_{n,t} = \frac{\lambda(n-2)\lambda(n)}{\lambda(n-t)\lambda(t-2)\lambda(t)} \cdot q^{(-t)(t-2)}$$

Fig. 4 Branching Process/Birthday Based Algorithm.

```
$L_{\mathbf{a}} \leftarrow \ell$ random vectors from $H_k(\mathbf{a})$.
$L_{\mathbf{b}} \leftarrow \ell$ random vectors from $H_k(\mathbf{b})$.
$\mathcal{P} \leftarrow \big\{(x, y) \in L_{\mathbf{a}} \times L_{\mathbf{b}} \mid \textsc{Root}(\mathbf{a}, y) = \textsc{Root}(\mathbf{b}, x)\big\}$
for all $(x, y) \in \mathcal{P}$ do
    $(\mathbf{a}', \mathbf{b}') \leftarrow$ DeHomogenize($\mathbf{b} = T \circ \mathbf{a} \circ S$, using the assumption that $y = S \cdot x$).
    if obvious trick succeeds on $(\mathbf{a}', \mathbf{b}')$ then
        Solve($\mathbf{b} = T \circ \mathbf{a} \circ S$) // using the algorithm of section 4.2
        if solution is found then
            Abort the loop, report success.
        end if
    end if
end for
Report failure.
```

If the instance $(\mathbf{a}, \mathbf{b})$ is randomly chosen, then thanks to lemma 5, $\textsc{Root}(\mathbf{a}, x)$ will sample random Galton-Watson trees, in which the number of descendants of each node (except the root) is $q^i - q^2$ with probability $p_{n,i}$, for $2 \leq i \leq n$. The expected number of children μ of each node (except the root) and its variance σ^2 are therefore:

$$\mu = \sum_{k=2}^{n} \left(q^k - q^2\right) p_{n,k} = 1 + \mathcal{O}\left(q^{-n}\right), \qquad \sigma^2 = \sum_{k=2}^{n} \left(q^k - q^2 - \mu\right)^2 p_{n,k} = q^3 - q^2 + \mathcal{O}\left(q^{-n}\right)$$

Considering the expression of μ, we will consider that our Galton-Watson process is critical.[6] It then follows from [1, chapter I, part A, section 5, theorem 1] that these trees are finite with probability one. It is worthwhile to note that the probability that such a random tree has depth greater than k is equivalent to $2/(k\sigma^2)$ [1, chapter I, part B, section 9, theorem 1], and this shows that $\mathbb{E}\big[|H_k|\big] = (2/k) \cdot q^{n-2}/(q-1)$. Also, the expected total number of nodes after k generation is $k+1$ [42], so that computing $\textsc{Root}$ takes time about $\mathcal{O}\left(n^4\right)$. Note that it is always possible to truncate the trees to a certain depth by replacing the nodes that are too deep with empty multisets, in order to avoid an explosion in the cost of the computation of the tree.

The complexity of the algorithm depends of course on the number of pairs in $\mathcal{P}$, but this quantity is difficult to control. If all the trees sampled by $\textsc{Root}$ have the same shape, then $\mathcal{P}$ will be big. On the contrary, if all the trees sampled by $\textsc{Root}$ have different shapes, then $\mathcal{P}$ will be very small. The most common tree returned by $\textsc{Root}$ is…the tree with only one node, and lemma 12 tells us that it occurs with probability about 0.578 when $q = 2$. This sounds like bad news, but if we restrict our attention to trees of a certain height, then we get more "entropy".

We observed empirically that by randomly sampling $\sqrt{|H_n|}$ vectors from H_n we get nearly all-different trees, and this is sufficient for our purpose (it suggests to set $k = n$). However, rigorous results about the number of distinct trees obtained when sampling N times in H_k, for various N and k, were not readily available in the existing literature, to the best of our knowledge. This is an interesting subject to explore.

Under the conjecture that we get a "small" (say, polynomial in n) number of pairs in $\mathcal{P}$ by choosing $k = n$, then it seems reasonable to choose $\ell = 1/p_{OT} \cdot \sqrt{q^n/n}$. The complexity of the algorithms mainly lies in the cost of generating the lists, and is $\mathcal{O}\left(n^{3.5} \cdot q^{n/2}\right)$.

We implemented this algorithm inside the MAGMA computer algebra system, runnning on one core of a 2.8Ghz Xeon machine. The implementation was not particularly efficient nor optimized. We tried to break a challenge with $q = 2$, $u = n = 32$ (resp. 40). We chose $k = n$, and $\ell = \sqrt{q^n/n} = 11585$ (resp. 165794). Generating the two lists took 61 minutes (resp. 31h). Computing the trees and matching the two lists took 200s (resp. 2h) and resulted in $\mathcal{P}$ containing two pairs (resp. seven), out of which one was right. The whole procedure required 85 Mbytes of RAM (resp 1256Mb). This allows us to claim that our unoptimized implementation should outperform in practice the "finding the common zeroes" approach described in section 4.3 for $n \geq 75$ if $q = 2$. Our implementation of this technique was not optimized, and was written in a very high-level language, and on the contrary our GPU implementation of exhaustive search to find common zeroes is a pretty optimized piece of software. There is therefore clearly much room for lowering this thershold with a better implementation.

To conclude this part, it is interesting to compare the techniques presented in this section with previously known approach approach for IP, namely [25]. Due to the lack of space, we postpone this in appendix D.

Some Notes About an Earlier Algorithm. The algorithm we give for the hardest case has a complexity of order $q^{n/2}$. In the extended version of [49], an algorithm is given, and its complexity is claimed to be the same. This algorithm has never been implemented nor tested, and its practical impact is therefore unknown. It is highly heuristic, and its complexity and success probability are in fact unknown. In this section, we point out several shortcomings in its description and analysis which convinced us that further analysis is required to fully understand its complexity and its behavior. More specifically, the following facts are proved in annex C. $i)$ This algorithm fails systematically over $\mathbb{F}_q$ with $q = 2^m$ if $m > 1$. $ii)$ One of the

[6] In fact it is subcritical, because $\mu < 1$, but the difference is so small that it is not perceptible

suggested “boosting functions” fails systematically, and iii) The non-failing one succeeds with probability 0.144 (instead of $1/2$ as claimed originally). We believe that this is enough to justify that this algorithm deserves a further analysis.

6 Conclusion

In this paper, we present algorithms for the IP problem that solve the IP problem with one secret for random quadratic equations and one cubic equation and with two secrets. Moreover, we explain the complexity, success probability and give sufficient conditions so that the algorithms work. The basic idea of our algorithm is simple and some of them have been proposed for example on the IP with one secret but never formally analyzed. We try to find linear relations on the secrets and then apply Gröbner basis algorithm to linearize the system. In order to find linear relations we use the idea of Faugère and Perret and the differential of the systems. To deal with the most difficult cases we combine algebraic and statistical tools. All the proposed IP challenge can be broken in practice by the technique we describe.

References

1. Athreya, K.B., Ney, P.: Branching processes. Springer-Verlag, Berlin, New York, (1972)
2. Baena, J., Clough, C., Ding, J.: Square-vinegar signature scheme. In: PQCrypto ’08: Proceedings of the 2nd International Workshop on Post-Quantum Cryptography, Berlin, Heidelberg, Springer-Verlag (2008) 17–30
3. Bardet, M., Faugère, J.C., Salvy, B., Yang, B.Y.: Asymptotic Behaviour of the Degree of Regularity of Semi-Regular Polynomial Systems. In: MEGA’05. (2005) Eighth International Symposium on Effective Methods in Algebraic Geometry, Porto Conte, Alghero, Sardinia (Italy), May 27th – June 1st.
4. Bardet, M.: Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie. PhD thesis, Université de Paris VI (2004)
5. 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
6. Bardet, M., Faugère, J.C., Salvy, B., Yang, B.Y.: Asymptotic behaviour of the degree of regularity of semi-regular polynomial systems. In: Proc. of MEGA 2005, Eighth International Symposium on Effective Methods in Algebraic Geometry. (2005)
7. Bernstein, D.S.: Matrix mathematics. Theory, facts, and formulas. 2nd expanded ed. Princeton, NJ: Princeton University Press. xxxix, 1139 p. (2009)
8. 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
9. Biryukov, A., Cannière, C.D., Braeken, A., Preneel, B.: A toolbox for cryptanalysis: Linear and affine equivalence algorithms. In: EUROCRYPT. (2003) 33–50
10. Bosma, W., Cannon, J.J., Playoust, C.: The Magma Algebra System I: The User Language. J. Symb. Comput. **24**(3/4) (1997) 235–265
11. Bouillaguet, C., Fouque, P.A., Joux, A., Treger, J.: A family of weak keys in hfe (and the corresponding practical key-recovery). Cryptology ePrint Archive, Report 2009/619 (2009) `http://eprint.iacr.org/`.
12. Buchberger, B.: Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. PhD thesis, University of Innsbruck (1965)
13. Buss, J.F., Frandsen, G.S., Shallit, J.: The Computational Complexity of Some Problems of Linear Algebra. J. Comput. Syst. Sci. **58**(3) (1999) 572–596
14. Chevalley, C.: Démonstration d’une hypothèse de M. Artin. Abh. Math. Semin. Hamb. Univ. **11** (1935) 73–75
15. Clough, C., Baena, J., Ding, J., Yang, B.Y., Chen, M.S.: Square, a new multivariate encryption scheme. In Fischlin, M., ed.: CT-RSA. Volume 5473 of Lecture Notes in Computer Science., Springer (2009) 252–264
16. Cox, D.A., Little, J.B., O’Shea, D.: Ideals, Varieties and Algorithms. Springer (2005)
17. Ding, J., Wolf, C., Yang, B.Y.: ℓ-invertible cycles for multivariate quadratic public key cryptography. In Okamoto, T., Wang, X., eds.: Public Key Cryptography. Volume 4450 of Lecture Notes in Computer Science., Springer (2007) 266–281
18. dit Vehel, F.L., Perret, L.: Polynomial Equivalence Problems and Applications to Multivariate Cryptosystems. In Johansson, T., Maitra, S., eds.: INDOCRYPT. Volume 2904 of Lecture Notes in Computer Science., Springer (2003) 235–251
19. 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
20. Faugère, J.C., Gianni, P., Lazard, D., Mora, T.: Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. Journal of Symbolic Computation **16**(4) (1993) 329–344
21. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra **139**(1-3) (June 1999) 61–88
22. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra **139** (June 1999) 61–88
23. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F5). In Mora, T., ed.: Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation ISSAC, ACM Press (July 2002) 75–83 isbn: 1-58113-484-3.
24. Faugère, J.C., Joux, A.: Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases. In Boneh, D., ed.: Advances in Cryptology - CRYPTO 2003. Volume 2729 of LNCS., Springer (2003) 44–60
25. 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

26. Fortin, S.: The graph isomorphism problem. Technical report, University of Alberta (1996)
27. Fuhrmann, P.A.: A polynomial approach to linear algebra. Springer-Verlag New York, Inc., New York, NY, USA (1996)
28. Fulman, J.: Random matrix theory over finite fields. Bull. Amer. Math. Soc. (N.S **39** 51–85
29. Fusco, G., Bach, E.: Phase transition of multivariate polynomial systems. Mathematical. Structures in Comp. Sci. **19**(1) (2009) 9–23
30. Garey, M.R., Johnson, D.S.: Computers and Intractability, A Guide to the Theory of *NP*-Completeness. Freeman, New-York (1979)
31. 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
32. Geiselmann, W., Steinwandt, R., Beth, T.: Attacking the Affine Parts of SFLASH. In Honary, B., ed.: IMA Int. Conf. Volume 2260 of Lecture Notes in Computer Science., Springer (2001) 355–359
33. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity for all languages in np have zero-knowledge proof systems. J. ACM **38**(3) (1991) 691–729
34. Granboulan, L., Joux, A., Stern, J.: Inverting HFE Is Quasipolynomial. In Dwork, C., ed.: CRYPTO. Volume 4117 of Lecture Notes in Computer Science., Springer (2006) 345–356
35. Kipnis, A., Patarin, J., Goubin, L.: Unbalanced Oil and Vinegar Signature Schemes. In: EUROCRYPT. (1999) 206–222
36. Koblitz, N.: Algebraic Aspects of Cryptography. Volume 3 of Algorithms and Computation in Mathematics. Springer-Verlag (1998)
37. Lazard, D.: Gröbner-bases, gaussian elimination and resolution of systems of algebraic equations. In van Hulzen, J.A., ed.: EUROCAL. Volume 162 of Lecture Notes in Computer Science., Springer (1983) 146–156
38. Lidl, R., Niederreiter, H.: Finite fields. Cambridge University Press, New York, NY, USA (1997)
39. MacWilliams, J.: Orthogonal matrices over finite fields. The American Mathematical Monthly **76**(2) (1969) 152–164
40. Matsumoto, T., Imai, H.: Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In: Advances in Cryptology – EUROCRYPT 1988. Volume 330 of LNCS., Springer–Verlag (1988) 419–453
41. Naccache, D., ed.: Topics in Cryptology - CT-RSA 2001, The Cryptographer's Track at RSA Conference 2001, San Francisco, CA, USA, April 8-12, 2001, Proceedings. In Naccache, D., ed.: CT-RSA. Volume 2020 of Lecture Notes in Computer Science., Springer (2001)
42. Pakes, A.G.: Some limit theorems for the total progeny of a branching process. Advances in Applied Probability **3**(1) (1971) 176–192
43. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: EUROCRYPT. (1996) 33–48 Etended version available on `http://www.minrank.org/hfe.pdf`.
44. Patarin, J.: Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In: EUROCRYPT. (1996) 33–48
45. Patarin, J.: The Oil and Vinegar signature scheme. presented at the Dagstuhl Workshop on Cryptography (1997)
46. Patarin, J., Courtois, N., Goubin, L.: Flash, a fast multivariate signature algorithm. [41] 298–307
47. Patarin, J., Courtois, N., Goubin, L.: QUARTZ, 128-Bit Long Digital Signatures. [41] 282–297
48. 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
49. Patarin, J., Goubin, L., Courtois, N.: Improved Algorithms for Isomorphisms of Polynomials. In: EUROCRYPT. (1998) 184–200
50. Perret, L.: A Fast Cryptanalysis of the Isomorphism of Polynomials with One Secret Problem. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 354–370
51. Wang, L.C., Hu, Y.H., Lai, F., yen Chou, C., Yang, B.Y.: Tractable rational map signature. In Vaudenay, S., ed.: Public Key Cryptography. Volume 3386 of Lecture Notes in Computer Science., Springer (2005) 244–257
52. Wang, L.C., Yang, B.Y., Hu, Y.H., Lai, F.: A "medium-field" multivariate public-key encryption scheme. In Pointcheval, D., ed.: CT-RSA. Volume 3860 of Lecture Notes in Computer Science., Springer (2006) 132–149
53. Warning, E.: Bemerkung zur vorstehenden Arbeit von Herrn Chevalley. Abh. Math. Semin. Hamb. Univ. **11** (1935) 76–83
54. Wolf, C., Preneel, B.: Equivalent keys in hfe, c^*, and variations. In: Mycrypt. (2005) 33–49
55. Wolf, C., Preneel, B.: Large superfluous keys in multivariate quadratic asymmetric systems. In: Public Key Cryptography. (2005) 275–287
56. Wolf, C., Preneel, B.: Taxonomy of Public Key Schemes Based on the Problem of Multivariate Quadratic Equations. Cryptology ePrint Archive, Report 2005/077 (2005)

A Some Proofs

A.1 Proof of lemma 1

Proof.

$$\begin{aligned}
\mathbf{b}(x) &= T_c + T_\ell \cdot \mathbf{a}(S_\ell \cdot x + S_c) \\
&= T_c + T_\ell \cdot \Big(\mathrm{D}\mathbf{a}(S_\ell \cdot x, S_c) + \mathbf{a}(S_\ell \cdot x) + \mathbf{a}(S_c) - \mathbf{a}(0)\Big) \\
&= \Big[T_c + T_\ell \cdot \big(\mathbf{a}\left(S_c\right) - \mathbf{a}(0)\big)\Big] + \left(T_\ell \circ \frac{\partial \mathbf{a}}{\partial S_c} \circ S_\ell\right)(x) + \left(T_\ell \circ \mathbf{a} \circ S_\ell\right)(x)
\end{aligned}$$

The first statement follows from the application of [25, lemma 4] to the last equality. The second and third statements are direct consequences of the first one. □

A.2 Proof of theorem 1

Proof. Because a solution of $\mathcal{S}$ exists, then B_1 is invertible. Thanks to this, we can write:

$$\mathcal{S}: \quad \begin{cases} Y = A_1^{-1} \cdot X^{-1} \cdot B_1 \\ B_2 \cdot B_1^{-1} \cdot X = X \cdot A_2 \cdot A_1^{-1} \end{cases}$$

Using the particular solution X_0 then gives:

$$\mathcal{S}: \quad \begin{cases} Y = A_1^{-1} \cdot X^{-1} \cdot B_1 \\ \mathcal{C} \cdot \left(X_0^{-1} \cdot X\right) = \left(X_0^{-1} \cdot X\right) \cdot \mathcal{C} \end{cases}$$

From there, it is not difficult to see that the kernel of $\mathcal{S}$ is in one-to-one correspondance with the commutant of $\mathcal{C}$, the isomorphism being $(X,Y) \mapsto X_0^{-1} \cdot X$. The second point of the theorem follows from the well-known fact that n lower-bounds the dimension of the commutant of any endomorphism on a vector space of dimension n (see for instance [7, Fact 2.18.9]). The third point follows from a general result on the dimension of the commutant [27, chapter 6, exercise 32]. □

A.3 Proof of Lemma 2

Proof. First, if q is even then $\mathcal{P}(\mathbf{a}_1)$ and $\mathcal{P}(\mathbf{a}_2)$ are symmetric matrices with null-diagonal. Since the inverse of a symmetric matrix with null-diagonal is also a symmetric matrix with null-diagonal, it follows that $\mathcal{C}$ is the product of two symmetric matrices with null-diagonal.

If I and J denote subsets of $\{1, \dots, n\}$ of the same cardinality, then $[M]_{I,J}$ denotes the minor of M that correspond to the rows in I and the columns in J. If $I = J$, the minor is said to be principal.

Let us write $\chi_{\mathcal{C}} = \sum_{i=0}^{n} a_i X^{n-i}$. We have:

$$a_k = \sum_{|I|=k} [\mathcal{C}]_{I,I}$$

And by the well-known extension of the Binet-Cauchy Formula, we have:

$$a_k = \sum_{|I|=k} \sum_{|J|=k} [\mathcal{P}(\mathbf{a}_1)]_{I,J} \cdot [\mathcal{P}(\mathbf{a}_2)]_{J,I}$$

Let us assume that I and J are of size k in the sequel, to lighten the notational burden. We can write:

$$a_k = \left(\sum_{I} [\mathcal{P}(\mathbf{a}_1)]_{I,I} \cdot [\mathcal{P}(\mathbf{a}_2)]_{I,I} \right) + \left(\sum_{I \neq J} [\mathcal{P}(\mathbf{a}_1)]_{I,J} \cdot [\mathcal{P}(\mathbf{a}_2)]_{J,I} \right)$$

The second term of this sum is in fact zero. To see why it is the case, let us consider all subsets of $\{1, \dots, n\}$ of cardinality k, and let us denote then by $U_1, \dots, U_m$. We may reorganize the terms of the sum:

$$\sum_{I \neq J} [\mathcal{P}(\mathbf{a}_1)]_{I,J} \cdot [\mathcal{P}(\mathbf{a}_2)]_{J,I} = \sum_{i=1}^{m} \sum_{j<i} \Big([\mathcal{P}(\mathbf{a}_1)]_{U_i,U_j} \cdot [\mathcal{P}(\mathbf{a}_2)]_{U_j,U_i} + [\mathcal{P}(\mathbf{a}_1)]_{U_j,U_i} \cdot [\mathcal{P}(\mathbf{a}_2)]_{U_i,U_j} \Big)$$

Then, because $\mathcal{P}(\mathbf{a}_1)$ is symmetric, we have $[\mathcal{P}(\mathbf{a}_1)]_{I,J} = [\mathcal{P}(\mathbf{a}_1)]_{J,I}$, and thus all the terms of the sum cancel out. So we are left with:

$$a_k = \sum_{I} [\mathcal{P}(\mathbf{a}_1)]_{I,I} \cdot [\mathcal{P}(\mathbf{a}_2)]_{I,I}$$

We now prove that the determinant of a $n \times n$ symmetric matrix with null-diagonal is zero if n is odd, and is a square if n is even. This will show that a_k is zero if k is odd, and is a square if k is even.

This property of a_k shows that all the coefficients of $\chi_{\mathcal{C}}$ such that $k + n \equiv 1 \bmod 2$ are zero, and the others are squares. This is sufficient to establish the result. □

A.4 Proof of theorem 2.

Proof. We use the fact that the degree of regularity of an ideal is generically left invariant by any linear change of the variables or generators [37]. In particular, we consider the ideal $\mathcal{I}'_{\mathbf{a},\mathbf{b}}$ generated by all the coefficients in $\mathbb{K}[x_1,\ldots,x_n,y_1,\ldots,y_n]$ of the equations:

$$\mathrm{D}\mathbf{a}(S_0(S+I_n)\mathbf{x},\mathbf{y}) - \mathrm{D}\mathbf{b}(\mathbf{x},(U+I_n)S_0^{-1}\mathbf{y}) = 0, \qquad U\cdot S = 0_n, \qquad S\cdot U = 0_n.$$

It is clear that $\mathcal{I}'_{\mathbf{a},\mathbf{b}}$ is obtained from $\mathcal{I}_{\mathbf{a},\mathbf{b}}$ by replacing S (resp. U) by $S_0(I_n+S)$ (resp. $(U+I_n)S_0^{-1}$). Thus, the degree of regularity of $\mathcal{I}'_{\mathbf{a},\mathbf{b}}$ and $\mathcal{I}_{\mathbf{a},\mathbf{b}}$ are equal. Using the same transformation, the ideal $\mathcal{J}$ becomes

$$\mathcal{J}' = \langle s_{1,j} \mid j = 1,\ldots,n\rangle\,.$$

We now estimate the degree of regularity of the ideal $\mathcal{I}'_{\mathbf{a},\mathbf{b}} + \mathcal{J}'$. For a reason which will become clear in the sequel, it is more convenient to work with $\mathcal{I}'_{\mathbf{a},\mathbf{b}} + \mathcal{J}'$. In what follows, F will denote the generators of $\mathcal{I}'_{\mathbf{a},\mathbf{b}} + \mathcal{J}'$. We will show that many new linear equations appear when considering equations of degree 2. To formalize this, we introduce some definitions related to the F_4 algorithm [22]. In particular, we will denote by $I_{d,k}$ the linear space generated during the k-th step of F_4 when considering polynomials of degree d.

Definition 1. *We have the following recursive definition of $I_{d,k}$:*

$$\begin{aligned}
I_{d,0}(F) &= \mathit{Vect}_{\mathbb{K}}\,(F)\\
I_{d,1}(F) &= \mathit{Vect}_{\mathbb{K}}\,(s_{i,j}f \mid 1 \leqslant i,j \leqslant n \text{ and } f \in I_{d,0}(F))\\
&\quad + \mathit{Vect}_{\mathbb{K}}\,(u_{i,j}f \mid 1 \leqslant i,j \leqslant n \text{ and } f \in I_{d,0}(F))\\
I_{d,k}(F) &= \mathit{Vect}_{\mathbb{K}}\,(s_{i,j}f \mid 1 \leqslant i,j \leqslant n \text{ and } f \in I_{d,k-1}(F) \text{ and } \deg(f) \le d-1)\\
&\quad + \mathit{Vect}_{\mathbb{K}}\,(u_{i,j}f \mid 1 \leqslant i,j \leqslant n \text{ and } f \in I_{d,k-1}(F) \text{ and } \deg(f) \le d-1)\,.
\end{aligned}$$

Roughly speaking, the index k is the number of steps in the F_4/F_5 [23] algorithm to compute an element $f \in I_{d,k}(F)$. We show that $I_{2,1}(F)$ contains exactly $n^2 + 2n$ linear equations. This means that we have already many linear equations generated during the first step of a Gröbner basis computation of F.

Lemma 6. *$I_{2,1}(F)$ contains the following linear equations:*

$$\{u_{1,j} \mid j = 1,\ldots,n\}. \tag{9}$$

Proof. From the first row of the following zero matrix $S \cdot U$ we obtain the following equations:

$$\begin{cases}
s_{1,1}\,u_{1,1} + s_{1,2}\,u_{2,1} + s_{1,3}\,u_{3,1} + \cdots + s_{1,n}\,u_{n,1} = 0,\\
s_{1,1}\,u_{1,2} + s_{1,2}\,u_{2,2} + s_{1,3}\,u_{3,2} + \cdots + s_{1,n}\,u_{n,2} = 0,\\
s_{1,1}\,u_{1,3} + s_{1,2}\,u_{2,3} + s_{1,3}\,u_{3,3} + \cdots + s_{1,n}\,u_{n,3} = 0,\\
\quad\cdots\\
s_{1,1}\,u_{1,n} + s_{1,2}\,u_{2,n} + s_{1,3}\,u_{3,n} + \cdots + s_{1,n}\,u_{n,n} = 0
\end{cases}$$

Using the equations $s_{1,j} = 0$ from the ideal $\mathcal{J}'$, we obtain then $u_{1,1} = 0, u_{1,2} = 0, \ldots, u_{1,n} = 0$. □

We can also predict the existence of other linear equations in $I_{2,1}(F)$.

Lemma 7. *For all $(i,j) \in \{1,\ldots,n\}^2$ the coefficient of $y_1y_ix_j$ in $\mathrm{D}\mathbf{a}(S_0(S+I_n)\mathbf{x},\mathbf{y}) - \mathrm{D}\mathbf{b}(\mathbf{x},(U+I_n)S_0^{-1}\mathbf{y})$ is a non zero*[7] *linear equation modulo the equations of the ideal $\mathcal{J}'$ and* (9)*. Among these equations, there are n which depend only of the variables $\{s_{k,\ell} \mid 1 \le k,\ell \le n\}$.*

Proof. We consider the coefficient of the monomial $m = y_1y_ix_j$ in the expression

$$\Delta = \Delta_a - \Delta_b = \mathrm{D}\mathbf{a}(S_0(S+I_n)\mathbf{x},\mathbf{y}) - \mathrm{D}\mathbf{b}(\mathbf{x},(U+I_n)S_0^{-1}\mathbf{y}).$$

Since the monomial m is linear in x_j it is clear that the corresponding coefficient in $\Delta_a = \mathrm{D}\mathbf{a}(S_0(S+I_n)\mathbf{x},\mathbf{y})$ is also linear in the variables $s_{i,j}$; moreover this coefficient is non zero. We have now to consider the coefficient of m in Δ_b. Since $\mathrm{D}\mathbf{b}(\mathbf{x},\mathbf{y})$ is the differential of an homogenous polynomial of degree 3 we can always write:

$$\mathrm{D}\mathbf{b}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{n}\sum_{j=i}^{n} \ell_{i,j}(y_1,\ldots,y_n)\,x_ix_j + \sum_{i=1}^{n} q_i(y_1,\ldots,y_n)x_i \tag{10}$$

[7] more precisely, generically non zero.

where $\ell_{i,j}$ (resp. q_i) is a polynomial of degree 1 (resp. 2). Consequently, the coefficient of m in $\mathrm{D}\mathbf{b}$ is also the coefficient of $y_1 y_i$ in $q_j((U+I_n)S_0^{-1}\mathbf{y})$. That is to say, in $q_j(\mathbf{y})$ we have now to replace $\mathbf{y} = (y_1, \ldots, y_n)$ by $((U+I_n)S_0^{-1}\mathbf{y})$. Thus, modulo the equations of the ideal $\mathcal{J}'$ and (9), we can write the product $((U+I_n)S_0^{-1}\mathbf{y})$ as

$$= \begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ u_{2,1} & \cdots & \cdots & u_{2,n} \\ \vdots & \cdots & \cdots & \vdots \\ u_{n,1} & \cdots & \cdots & u_{n,n} \end{pmatrix} \begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}$$

$$= \begin{pmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{pmatrix} \begin{pmatrix} * & * \quad * & * \\ (*u_{2,1} + \cdots + *u_{2,n}) & \cdots\cdots & (*u_{2,1} + \cdots + *u_{2,n}) \\ \vdots & \cdots\cdots & \vdots \\ (*u_{n,1} + \cdots + *u_{n,n}) & \cdots\cdots & (*u_{2,1} + \cdots + *u_{n,n}) \end{pmatrix}$$

$$= \begin{pmatrix} *y_1 + (*u_{2,1} + \cdots + *u_{2,n})y_2 + \cdots + (*u_{n,1} + \cdots + *u_{n,n})y_n \\ *y_1 + (*u_{2,1} + \cdots + *u_{2,n})y_2 + \cdots + (*u_{n,1} + \cdots + *u_{n,n})y_n \\ \vdots \\ *y_1 + (*u_{2,1} + \cdots + *u_{2,n})y_2 + \cdots + (*u_{n,1} + \cdots + *u_{n,n})y_n \end{pmatrix}$$

Hence the coefficient of $y_1 y_i$ in $q_j((U+I_n)S_0^{-1}\mathbf{y})$ is linear in the variables $u_{k,l}$ when $i \neq 1$ and the coefficient of y_1^2 is a constant. □

To summarize:

Lemma 8. *$I_{2,1}(F)$ contains exactly $n^2 + 2n$ linear equations.*

Proof. In $I_{2,1}(F)$, we have n linear equations from lemma 7, n linear equations from the very definition of $\mathcal{J}'$, and n^2 linear equations from lemma 7 □

As explained before, we obtain $n^2 + 2n$ linear equations for $I_{2,1}(F)$. However, we have $2n^2$ variables. So, we have to consider $I_{2,2}(F)$, *i.e.*, the equations generated at degree 2 during the second step. Thanks to lemma 8, we can reduce the original system to a quadratic system in $2n^2 - (2n + n^2) = (n-1)^2$ variables. W.l.o.g we can assume that we keep only the variable $u_{i,j}$ where $2 \leq i, j \leq n$. Let F' be the system obtained from F after substituting the $2n + n^2$ linear equations of lemma 8. All the monomials in $\mathbb{K}[x_1, \ldots, x_n, y_1, \ldots, y_n]$ of $\mathrm{D}\mathbf{a}(S_0(S+I_n)\mathbf{x}, \mathbf{y}) - \mathrm{D}\mathbf{b}(\mathbf{x}, (U+I_n)S_0^{-1}\mathbf{y})$ have the following shape:

$$x_i y_j y_k \text{ or } y_i x_j x_k \text{ with } 1 \leq i, j, k \leq n.$$

Hence the number of such monomials is $2n\frac{n(n+1)}{2} = n^2(n+1) \approx n^3$, which implies that the number of equations in F' is also n^3. Thanks to this remark, we will now prove that we can linearize F'. Let $T(F')$ be the set of all monomials occurring in F'. We can assume that $T(G') = [t_1 < t_2 < \cdots < t_N]$. It is important to remark that $t_1 = u_{2,2}$ up to $t_{(n-1)^2} = u_{n,n}$ are in fact variables. Now, let M be the matrix representation of G' w.r.t. $T(G')$. Since we know precisely the shape of the equations from the proof of lemma 7, it is possible to establish that:

1. most of the equations are very sparse, namely each equation contains about n^2 non-zero terms.
2. all the variables $t_1, \ldots, t_{(n-1)^2}$ occur in *all* the equations

After a Gaussian elimination of the matrix M, we obtain the following shape:

$$\widetilde{M} = \begin{bmatrix} 1_{(n-1)^2} & 0 & 0 & 0 \\ 0 & \times & \cdots & \cdots \\ 0 & \times & \ddots & \vdots \\ 0 & \times & \cdots & \ddots \end{bmatrix}$$

Hence, we obtain after a second step of computation in degree 2 the equations $u_{2,2} = \cdots = u_{n,n} = 0$. This means that after 2 steps of computation at degree 2, we obtain $(n-1)^2 + 2n + n^2 = 2n^2$ linear equations in $2n^2$. This explains why the maximum degree reached during the Gröbner basis computation of $\mathcal{I}'_{\mathbf{a},\mathbf{b}} + \mathcal{J}'$ is bounded by 2, and concludes the proof of theorem 2. □

A.5 Proof of Lemma 3

Proof. Let us consider the elements of $Z_{\mathbf{a}}$ having α as their first coordinate, and let us denote by n_α their number. These are in fact the common zeroes of $(\mathbf{a}, x_1 - \alpha)$. By the Chevalley-Warning theorem [14,53], if $\mathbf{a}$ has at least 5 variables, then the characteristic of the field divides n_α. Therefore, their sum has zero on the first coordinate. Applying this result for all values of α shows that the sum of zeroes of $\mathbf{a}$ has a null first coordinate. We then just consider all coordinates successively. □

A.6 Proof of Lemma 4

Proof. Fortunately, we can use theorem 1, using the hypothesis that $\mathbf{b}^{(1)}$ is invertible. The dimension of the kernel of $\mathcal{S}$ will be exactly n if and only if the minimal and characteristic polynomial of $\mathcal{C} = \frac{\partial \mathbf{b}}{\partial \mathbf{x}} \cdot \left(\mathbf{b}^{(1)}\right)^{-1}$ are the same. We already mentioned that this happens with a noticeable probability for random matrices (see lemma 11). Unfortunately, in characteristic two, $\frac{\partial \mathbf{b}}{\partial \mathbf{x}}$ always vanish on $\mathbf{x}$, and $\mathcal{C}$ always has a non-trivial kernel. As a consequence, $\mathcal{C}$ is hardly random. However, lemma 12 tells us that $\mathcal{C}$ is in fact randomly distributed amongst the set of linear mappings that vanish at $\mathbf{x}$.

Let $e_n = \mathbf{x}$, and let us extend this vector to a basis $e_1, \ldots, e_n$ of $\left(\mathbb{F}_q\right)^n$. In this new basis, the last column of $\mathcal{C}$ is zero. Expanding the determinant of $\mathcal{C} - X \cdot 1_n$ along the last column yields $\chi_{\mathcal{C}} = X \cdot \chi_{\mathcal{C}'}$, where $\mathcal{C}'$ is the matrix obtained by removing the last row and the last column of the matrix representing the same endomorphism as $\mathcal{C}$ in the new basis. This shows that the characteristic polynomial of $\mathcal{C}$ is the product of X and of the characteristic polynomial of a random matrix of dimension $(n-1) \times (n-1)$. This time, since $\mathcal{C}'$ is random, lemma 11 is applicable and tells us about the probability that the minimal polynomial of $\mathcal{C}'$ is of degree $n-1$. If this event happens, there are therefore two possible situations:

1. Either $\mathcal{C}$ has only one invariant factor, and the kernel of $\mathcal{S}$ has dimension n.
2. Or $\mathcal{C}$ has two invariant factors, X and $\chi_{\mathcal{C}'}$, and the kernel of $\mathcal{S}$ has dimension $n+2$.

□

B Probabilities For Random Matrices

If q is odd, then the probability that a random symmetric matrix has a given rank is given by the following result.

Lemma 9 ([39], theorem 2). *Let $N(n,r)$ denote the number of symmetric matrices of size $n \times n$ over $\mathbb{F}_q$ and of rank r.*

$$N(n, 2s) = \prod_{i=1}^{s} \frac{q^{2i}}{q^{2i}-1} \cdot \prod_{i=1}^{2s-1} \left(q^{n-i} - 1\right)$$
$$N(n, 2s+1) = \prod_{i=1}^{s} \frac{q^{2i}}{q^{2i}-1} \cdot \prod_{i=1}^{2s} \left(q^{n-i} - 1\right)$$

If q is even, then the probability that a random symmetric matrix with zeros on the diagonal, has a given rank is given by this other result.

Lemma 10 ([39], theorem 3). *Let $N_0(n,r)$ denote the number of symmetric matrices of size $n \times n$ over $\mathbb{F}_q$ with zeros on the diagonal and of rank r.*

$$N_0(n, 2s) = \prod_{i=1}^{s} \frac{q^{2i-2}}{q^{2i}-1} \cdot \prod_{i=1}^{2s-1} \left(q^{n-i} - 1\right)$$
$$N_0(n, 2s+1) = 0$$

Lemma 11 ([28], theorem 1). *Let $c(n,q)$ be the proportion of cyclic $n \times n$ matrices (i.e., matrices for which the minimal polynomial is of degree n). We have:*

$$\frac{1}{q^2(q+1)} < 1 - c(n,q) < \frac{1}{(q^2-1)(q-1)}$$

And asymptotically, we have:

$$\lim_{n \to \infty} c(n,q) = \frac{q^5 - 1}{q^2(q-1)(q^2-1)} \cdot \prod_{i=1}^{\infty} \left(1 - \frac{1}{q^i}\right)$$

Lemma 12 ([19], theorem 2). *Let $\mathbb{F}_q$ be a field of characteristic two. Given a random quadratic map $f : (\mathbb{F}_q)^n \to (\mathbb{F}_q)^n$, then and a non-zero element $\mathbf{x}$ of $(\mathbb{F}_q)^n$:*

i) The derivative $\frac{\partial f}{\partial \mathbf{x}}$ follows the distribution of endomorphisms of $(\mathbb{F}_q)^n$ vanishing at $\mathbf{x}$;
ii) For any t in $[1, n]$ the probability that $\frac{\partial f}{\partial \mathbf{x}}$ has rank $n - t$ is:

$$\alpha_{n,t} \cdot q^{(-t)(t-1)} \qquad \textit{with } \alpha_{n,t} = \frac{\lambda(n-1)\lambda(n)}{\lambda(n-t)\lambda(t-1)\lambda(t)} \textit{ and } \lambda(k) = \prod_{i=1}^{k} \left(1 - \frac{1}{q^i}\right)$$

C A Birthday Paradox Algorithm?

Complete Failure over $\mathbb{F}_{2^m}$, $m > 1$. This algorithm makes a critical use of so-called “boosting functions”. A boosting function is in fact a pair of functions $F_\mathbf{a}$ and $F_\mathbf{b}$ such that $S_\ell \cdot \mathbf{x} = \mathbf{y}$ implies $S_\ell \cdot F_\mathbf{b}(\mathbf{x}) = F_\mathbf{a}(\mathbf{y})$ with non-negligible probability. The evaluation of $F_\mathbf{a}$ and $F_\mathbf{b}$ must be efficient (typically, polynomial in the parameters), and the functions are not assumed to be deterministic. It is expected that $F_\mathbf{b}(\mathbf{x})$ is different from both zero and $\mathbf{x}$ with a noticeable probability (otherwise, they do not help much).

The starting point over which boosting functions are constructed in [49] is the observation that for a given vector $\mathbf{z}$, the sets $E_\mathbf{a} = \{\mathbf{y} | \mathbf{a}(S_\ell \cdot \mathbf{z} + \mathbf{y}) = \mathbf{a}(\mathbf{y})\}$ and $E_\mathbf{b} = \{\mathbf{x} | \mathbf{b}(\mathbf{z} + \mathbf{x}) = \mathbf{b}(\mathbf{x})\}$ are in correspondence via S_ℓ: $E_\mathbf{a} = S_\ell(E_\mathbf{b})$. This equality between sets can be collapsed to a single vector equality by summing over all the elements:

$$S_\ell \cdot \left(\sum_{\mathbf{y} \in E_\mathbf{b}} \mathbf{y} \right) = \sum_{\mathbf{x} \in E_\mathbf{a}} \mathbf{x}$$

.

This would naturally lead to the definition of a boosting function:

$$F_\mathbf{a}(\mathbf{z}) = \sum_{\mathbf{a}(\mathbf{y}+\mathbf{z})=\mathbf{a}(\mathbf{y})} \mathbf{y} \qquad F_\mathbf{b}(\mathbf{z}) = \sum_{\mathbf{b}(\mathbf{x}+\mathbf{z})=\mathbf{b}(\mathbf{x})} \mathbf{x}$$

Unfortunately, as the authors of [49] pointed out, over fields of characteristic two, we have that if $\mathbf{x} \in E_\mathbf{b}$, then so does $\mathbf{x} + \mathbf{z}$, and the elements of $E_\mathbf{b}$ would always sum to a vector collinear to $\mathbf{z}$ (a similar result applies to $E_\mathbf{a}$). In any case, no new information on S_ℓ can be obtained this way. To overcome this issue, they suggested to sum over the elements of $E_\mathbf{b}$, but excluding $\mathbf{x} + \mathbf{z}$ if $\mathbf{x}$ enters the sum. More precisely, they defined:

$$F_\mathbf{a}(\mathbf{z}) = \sum_{\substack{\mathbf{a}(\mathbf{y}+\mathbf{z})=\mathbf{a}(\mathbf{y}) \\ \text{one only} \in \{\mathbf{y}, \mathbf{y}+\mathbf{z}\}}} \mathbf{y} \qquad F_\mathbf{b}(\mathbf{z}) = \sum_{\substack{\mathbf{b}(\mathbf{x}+\mathbf{z})=\mathbf{b}(\mathbf{x}) \\ \text{one only} \in \{\mathbf{x}, \mathbf{x}+\mathbf{z}\}}} \mathbf{x}$$

In other terms, they suggested using the algorithm shown in figure 5 to compute the sum. In other terms, $F_\mathbf{a}(\mathbf{z}) = \mathbf{PartialSum}(E_\mathbf{a}, \mathbf{z})$ and $F_\mathbf{b}(\mathbf{z}) = \mathbf{PartialSum}(E_\mathbf{b}, \mathbf{z})$. Unfortunately again, this way of doing things does not solve the problem, as we now establish. For this purpose, we first prove a general result on $\mathbf{PartialSum}$.

Fig. 5 : $\mathbf{PartialSum}(V, \mathbf{z})$

```
$Ban \leftarrow \emptyset$
$sum \leftarrow 0$
while $V - Ban \neq \emptyset$ do
    let $\mathbf{x}$ be a random element of $(V - Ban)$
    $Ban \leftarrow Ban \cup \{\mathbf{x}, \mathbf{x} + \mathbf{z}\}$
    $sum \leftarrow sum + \mathbf{x}$
end whilereturn $sum$
```

Lemma 13. *Let V be a vector space such that $\mathbf{z} \in V$, and c be a vector in $(\mathbb{F}_q)^n$. We denote by $c + V$ the affine space formed by the vectors $c + x$, for all $x \in V$.*

i) If $q > 2$ or $\dim V > 1$, then $\mathbf{PartialSum}(c + V, \mathbf{z}) \in \mathrm{Span}(\mathbf{z})$.
ii) If $q = 2$ and $\dim V \leq 1$, then $\mathbf{PartialSum}(c + V, \mathbf{z})$ returns either c or $c + \mathbf{z}$

Proof. Let us denote by k the dimension of V. As a consequence, we have $\left|V\right| = q^k$. The vectors that are added to sum on line 6 are of the form : $c + v$, with $v \in V$, and the "while" loop on line 3 is iterated $q^k/2$ times.

If $q = 2$ and $k = 1$, the loop is iterated only once, and since $V = \{0, \mathbf{z}\}$, either c or $c + \mathbf{z}$ will be selected, which proves the second part of our statement.

The number of iterations of the loop is even as soon as $q > 2$ or $\dim V > 1$. If either one of these conditions is true, then c being added an even number of times to sum cancels out, and we have that:

$$\mathbf{PartialSum}(c + V, \mathbf{z}) = \mathbf{PartialSum}(V, \mathbf{z})$$

Next, it is known that $\mathbf{z} \neq 0$ belongs to V. Therefore, we can extend $\mathbf{z}$ to a basis of V. More precisely, let us denote by $U = \operatorname{Span}(b_1, b_2, \ldots, b_{k-1})$ a $(k-1)$-dimensional vector space such that $V = \operatorname{Span}(\mathbf{z}) \bigoplus U$. Let us denote by ρ the projection from V to U (*i.e.*,, the mapping that keeps only the last $k-1$ coordinates). To establish the first part of our statement, we need to show that $\rho(sum) = 0$ at the end of the loop.

Now, there are exactly q vectors in V that are sent to the same element α of U by ρ. During the loop, $q/2$ of these vectors will be selected (the other half being discarded).

- If $q = 2$, exactly one vector the projection of which is α will be selected, for all $\alpha \in U$. Therefore,

$$\rho(sum) = \sum_{\alpha \in U} \alpha = 0$$

- If $q > 2$, because $k > 1$, then an even number of vectors with the same projection will be selected in the loop, and therefore their projetions sum up to zero. □

Corollary 1. *Unless $q = 2$ and the rank of $\frac{\partial \mathbf{b}}{\partial \mathbf{z}}$ is $n - 1$, $F_{\mathbf{a}}(\mathbf{z})$ is collinear to $\mathbf{z}$.*

Proof. The set $E_{\mathbf{b}}$ is in fact an affine space: it is easy to see that $\mathbf{x} \in E_{\mathbf{b}}$ is equivalent to $\mathbf{b}(\mathbf{z}) + \mathrm{D}_{\mathbf{z}} b \cdot \mathbf{x} = 0$. This equation yields u affine relations, the solution of which can be written:

$$E_{\mathbf{b}} = c + \ker \mathrm{D}_{\mathbf{z}} b$$

where c denotes a particular solution. It must be noted that $\mathbf{z} \neq 0$ belongs to $\ker \mathrm{D}_{\mathbf{z}} b$. Then, lemma 13 concludes the proof. □

We have shown that as soon as $q > 2$, this particular boosting function cannot, by any means, provide additional knowledge on S_ℓ, and when using it, the while algorithm is bound to fail. On the other hand, it was also bound to fail when $\mathbf{a}$ and $\mathbf{b}$ are injective mappings (since in this case, $E_{\mathbf{a}} = E_{\mathbf{b}} = \emptyset$). Note that the same reasoning about the size of the field can be extended to any boosting function making use of $\mathbf{PartialSum}$.

Systematic Failure of a Boosting Function. An other boosting function is defined in [49], with the aim of overcoming this last problem:

$$G_{\mathbf{a}}(\mathbf{z}) = \sum_{\substack{\mathbf{a}(\mathbf{y}+\mathbf{z}) = \mathbf{a}(\mathbf{y}) + \mathbf{a}(\mathbf{z}) \\ \text{one only} \in \{\mathbf{y}, \mathbf{y}+\mathbf{z}\}}} \mathbf{y} \qquad G_{\mathbf{b}}(\mathbf{z}) = \sum_{\substack{\mathbf{b}(\mathbf{x}+\mathbf{z}) = \mathbf{b}(\mathbf{x}) + \mathbf{b}(\mathbf{z}) \\ \text{one only} \in \{\mathbf{x}, \mathbf{x}+\mathbf{z}\}}} \mathbf{x}$$

Or, using our terminology:

$$G_{\mathbf{a}}(\mathbf{z}) = \mathbf{PartialSum}\left(\ker \frac{\partial \mathbf{a}}{\partial \mathbf{z}}, \mathbf{z}\right) \qquad G_{\mathbf{b}}(\mathbf{z}) = \mathbf{PartialSum}\left(\ker \frac{\partial \mathbf{b}}{\partial \mathbf{z}}, \mathbf{z}\right)$$

Unfortunately, G cannot be used as a boosting function at all, even if $q = 2$.

Lemma 14. *$G_{\mathbf{b}}(\mathbf{z})$ is collinear to $\mathbf{z}$.*

Proof. Here, we have $E_{\mathbf{b}} = \ker \mathrm{D}_{\mathbf{z}} \mathbf{b}$. By lemma 13 (and in either of the considered cases), we obtain the result. □

Success Probability of the Boosting Function. So far, we have shown that the algorithm proposed in [49] does not work at all over $\mathbb{F}_q$ with $q = 2^k > 2$. Therefore, it cannot be used to break some of the challenges proposed by Patarin. On the positive side, it was shown that boosting functions do exist. It is even possible to study them rigorously.

Lemma 15. *When $q = 2$, and assuming that $\mathbf{b}$ is chosen randomly amongst all quadratic maps, and that $\mathbf{z}$ is randomly chosen, the success probability of the boosting function is about 0.144 (and not always $1/2$ as incorrectly stated in [49]).*

Proof. If $q = 2$, the probability that $\dim \ker \frac{\partial \mathbf{b}}{\partial \mathbf{z}} = 1$, for a random $\mathbf{z}$, is $2\lambda(n)$ according to lemma 12.

If $q = 2$ and $\dim \ker \frac{\partial \mathbf{b}}{\partial \mathbf{z}} = 1$, then $E_{\mathbf{b}}$ will be empty if the affine equation $\mathbf{b}(\mathbf{z}) + \frac{\partial \mathbf{b}}{\partial \mathbf{z}} \cdot \mathbf{x} = 0$ has no solutions. It has solutions if and only if $\mathbf{b}(\mathbf{z})$ belongs to the range of $\frac{\partial \mathbf{b}}{\partial \mathbf{z}}$. Because $\frac{\partial \mathbf{b}}{\partial \mathbf{z}}$ is of rank $n - 1$, a random vector belongs to its range with probability $1/2$. Now, $\frac{\partial \mathbf{b}}{\partial \mathbf{z}}$ and $\mathbf{b}(\mathbf{z})$ are not statistically independent, but they are nevertheless empirically sufficiently independent for the observed probability to be $1/2$.

If it is not empty, then $E_{\mathbf{b}} = \{c, c + \mathbf{z}\}$, and sum will be chosen randomly amongst these two vectors. Since a random choice also takes place in $F_{\mathbf{a}}$, the probability that the boosting function provides a right relation for S is $1/2$, under these hypotheses.

So all in all, the success probability is exactly $\lambda(n)/2$. This quantity is a decreasing function of n that quickly converges to its limit (0.144). This is confirmed by experiments. □

It is not very hard to improve on those presented in [49]. More specifically, it is possible, based on the same ideas, to build a boosting function that succeeds 3 times more often that $F_{\mathbf{a}}$ (by simply returning a random vector from $\ker \frac{\partial \mathbf{a}}{\partial \mathbf{z}} - \{\mathbf{z}\}$).

Complexity of $\boldsymbol{\Omega\left(n^4 \cdot q^{n/2}\right)}$. Evaluating the boosting function $F_{\mathbf{a}}$ takes $\mathcal{O}\left(n^3\right)$ operations, as it requires solving a system of n linear equations in n unknowns. It is clear from the (vague) description of the algorithm in [49] that the boosting function $H_{\mathbf{a}}$ is evaluated at least $n \cdot q^{n/2}$ times.

D Comparaison

To conclude this part, it is interesting to compare the techniques presented in Section 5 with previously known approach approach for IP, namely [25]. To simplify the analysis, we denote simply by $T \approx \mathcal{O}\left(q^{n/2}\right)$ the dominant par of the complexity of the algorithm 4. As already pointed out, this known approach is efficient (polynomial) in the affine case, but fails in the homogeneous case. We showed here that it is possible to reduce the homogeneous case to the linear one with the knowledge of one relation $y = S_\ell \cdot x$. The cost of [25] in the homogenous case is then $\approx \mathcal{O}\left(T^4\right)$, i.e. in this paper we have divided by 4 the complexity of [25]. Note that we can a bit improved the complexity of finding a pair $(x, S_\ell \cdot x)$. To do so, we can assume with good probability ($\approx 1/2$) that 1 is an eigenvalue of S_ℓ. In such case, it it sufficient to find an eigenvector x of S_ℓ to have a right pair as $S_\ell \cdot x = 1 \cdot x$. So, we have a probabilistic version of [25] working in $\approx \mathcal{O}\left(T^2\right)$. For the sack of comparison, we provide below experimental results obtained with [25], and its probabilistic version. We have reported the total number of operations required for several values of n. This is the exact cost of computing a solution in the affine case with [25] multiplied by 2^n or $2^{n/2}$ (i.e. we performed the experiments over $\mathbb{F}_2$). We also added the cost of our new algorithm presented in Section 5.

	[25]	probabilistic [25]	This paper
n	$\mathcal{O}\left(q^{2n}\right)$	$\mathcal{O}\left(q^{n}\right)$	$\mathcal{O}\left(q^{n/2}\right)$
12	2^{40}	2^{28}	$2^{18.5}$
16	$2^{50.5}$	$2^{34.5}$	2^{22}
20	$2^{60.4}$	$2^{40.4}$	2^{25}
30	$2^{83.9}$	$2^{53.9}$	2^{32}
40	$> 2^{80}$		$2^{38.6}$

The difference will of course increase as n get bigger.