# Improved Algorithms for Isomorphisms of Polynomials

– Extended Version –


Jacques Patarin, Louis Goubin
Bull Smart Cards and Terminals
68 route de Versailles - BP 45
78431 Louveciennes Cedex - France
e-mail : {J.Patarin,L.Goubin}@frlv.bull.fr

Nicolas Courtois
Systèmes Information Signal (SIS)
Université de Toulon et du Var - BP 132
83957 La Garde Cedex - France
e-mail : courtois@univ-tln.fr



**Abstract**

This paper is about the design of improved algorithms to solve Isomorphisms of Polynomials (IP) problems. These problems were first explicitly related to the problem of finding the secret key of some asymmetric cryptographic algorithms (such as Matsumoto and Imai's $C^*$ scheme of [13], or some variations of Patarin's HFE scheme of [15]). Moreover, in [15], it was shown that IP can be used in order to design an asymmetric authentication or signature scheme in a straightforward way. We also introduce the more general Morphisms of Polynomials problem (MP). As we see in this paper, these problems IP and MP have deep links with famous problems such as the Isomorphism of Graphs problem or the problem of fast multiplication of $n \times n$ matrices.

The complexities of our algorithms for IP are still not polynomial, but they are much more efficient than the previously known algorithms. For example, for the IP problem of finding the two secret matrices of a Matsumoto-Imai $C^*$ scheme over $K = \mathbf{F}_q$, the complexity of our algorithms is $\mathcal{O}(q^{n/2})$ instead of $\mathcal{O}(q^{(n^2)})$ for previous algorithms. (In [14], the $C^*$ scheme was broken, but the secret key was not found). Moreover, we have algorithms to achieve a complexity $\mathcal{O}(q^{\frac{3}{2}n})$ on any system of $n$ quadratic equations with $n$ variables over $K = \mathbf{F}_q$ (not only equations from $C^*$). We also show that the problem of deciding whether a polynomial isomorphism exists between two sets of equations is <u>not</u> NP-complete (assuming the classical hypothesis about Arthur-Merlin games), but solving IP is at least as difficult as the Graph Isomorphism problem (GI) (and perhaps much more difficult), so that IP ! is unl ikely to be solvable in polynomial time. Moreover, the more general Morphisms of Polynomials problem (MP) is NP-hard. Finally, we suggest some variations of the IP problem that may be particularly convenient for cryptographic use.



**Key Words:** Morphisms of Polynomials, Isomorphisms of Polynomials, Graph Isomorphisms, Zero-Knowledge Authentications, Asymmetric Signatures.

**Note:** This paper is the extended version of the paper with the same title published at Eurocrypt'98.


## 1 Introduction

This paper presents new algorithms for the Isomorphism of Polynomials (IP) problem. IP was explicitly described in [15], where it was shown that if some efficient algorithm exists for IP, then the secret key of some asymmetric cryptosystems (such as the $C^*$ scheme of [13] or some variations of the HFE scheme of [15]) would be found. (The cryptanalysis of $C^*$ given in [14] breaks the scheme without finding the secret key). No polynomial algorithm is known for IP. On the other side, in [15], it was also shown that if no efficient algorithm exists for IP, then this IP problem can be used to design some zero-knowledge authentication schemes. These schemes can also be transformed in order to have an asymmetric signature scheme.

This paper is divided into two parts. In part I, we present different variations of IP and we study a closely related problem, called MP for "Morphisms of Polynomials". We then show that these problems are closely related to other and more famous problems such as the Graph Isomorphism problem, and Fast Matrix Multiplication. In part II, we present our improved algorithms for IP. These algorithms are not polynomial (so the schemes based on IP are not broken), but they are much more efficient that the previously known schemes.

# Part I: Presentation and general properties of the IP and MP problems

## 2 The IP and MP problems

IP was presented in [15]. Let us recall what this problem is, in the particular case of quadratic forms (the problem can be generalized without difficulties to cubic forms, as well as forms of higher degree). Let $u$ and $n$ be two integers. Let $\mathbf{F}_q$ be a finite field. Let $(\mathcal{A})$ be a public set of $u$ quadratic equations with $n$ variables $a_1$, ..., $a_n$ over the field $\mathbf{F}_q$. We can write these equations as follows:

$$b_k = \sum_i \sum_j \gamma_{ijk} a_i a_j + \sum_i \mu_{ik} a_i + \delta_k \qquad (1 \leq k \leq u). \qquad (\mathcal{A})$$

Now let $s$ be a bijective and affine transformation of the variables $a_i$, $1 \leq i \leq n$, and let $t$ be a bijective and affine transformation of the variables $b_k$, $1 \leq k \leq u$.
We denote $s(a_1, ..., a_n) = (x_1, ..., x_n)$ and $t(b_1, ..., b_n) = (y_1, ..., y_n)$.
¿From $(\mathcal{A})$ we obtain another set $(\mathcal{B})$ of $k$ equations that give the $y_k$ values from the $x_i$ values:

$$y_k = \sum_i \sum_j \gamma'_{ijk} x_i x_j + \sum_i \mu'_{ik} x_i + \delta'_k \qquad (1 \leq k \leq u). \qquad (\mathcal{B})$$

We say that $(s, t)$ is an *isomorphism* from $(\mathcal{A})$ to $(\mathcal{B})$, and we say that $(\mathcal{A})$ and $(\mathcal{B})$ are *isomorphic*.
The IP problem is the following: if $(\mathcal{A})$ and $(\mathcal{B})$ are two public sets of $u$ quadratic equations, and if $(\mathcal{A})$ and $(\mathcal{B})$ are isomorphic, find an isomorphism $(s, t)$ from $(\mathcal{A})$ to $(\mathcal{B})$.
When $s$ and $t$ are not necessary bijective, we call the corresponding problem the "Morphism of Polynomials" problem (MP).

## 3 Examples

We now present three "toy examples" in order to become more familiar with the IP and MP problems.

**Example of IP with two secrets**

Let $K = \mathbf{F}_2$ be the field in which all the variables $x_0$, ..., $x_4$, $y_0$, ..., $y_4$, $a_0$, ..., $a_4$, $b_0$, ..., $b_4$ lie.
Let:

$$(\mathcal{A}) \begin{cases} b_1 = a_1 + a_1a_5 + a_2a_3 + a_2a_4 + a_3a_4 \\ b_2 = a_3 + a_4 + a_5 + a_1a_2 + a_1a_4 + a_4a_5 \\ b_3 = a_5 + a_1a_2 + a_1a_3 + a_1a_5 + a_2a_3 + a_3a_5 + a_4a_5 \\ b_4 = a_2 + a_3 + a_4 + a_5 + a_1a_5 + a_3a_4 + a_3a_5 \\ b_5 = a_4 + a_1a_3 + a_1a_5 + a_2a_3 + a_2a_4 + a_2a_5 + a_3a_4 + a_3a_5 + a_4a_5 \end{cases}$$

and let:

$$(\mathcal{B}) \begin{cases} y_1 = x_1 + x_3 + x_4 + x_5 + x_1x_2 + x_1x_3 + x_1x_5 + x_2x_4 + x_3x_5 + x_4x_5 \\ y_2 = x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_1x_5 + x_4x_5 \\ y_3 = x_1 + x_3 + x_4 + x_1x_2 + x_1x_5 + x_2x_3 + x_3x_4 \\ y_4 = x_3 + x_4 + x_1x_2 + x_1x_5 + x_3x_4 + x_3x_5 + x_4x_5 \\ y_5 = x_2 + x_4 + x_5 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_2x_5 \end{cases}$$

The problem is to find two bijective and linear transformations $s$ and $t$ such that $(x_1, ..., x_5) = s(a_1, ..., a_5)$, $(y_1, ..., y_5) = t(b_1, ..., b_5)$ and such that $(\mathcal{A})$ becomes $(\mathcal{B})$ with these transformations. (This is sometimes called an "IP problem with two secrets" since here there are <u>two</u> secret affine transformations to find: $s$ and $t$).

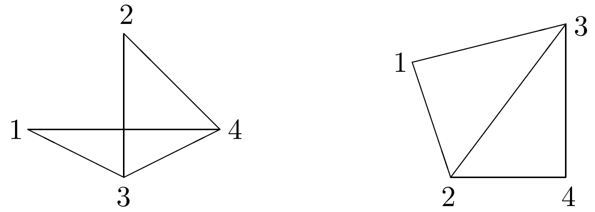


Figure 1: Graph $(I)$ Graph $(II)$

**Note 1:** This "toy example" comes from the "toy example" given in [13] for $C^*$, when the 8 variables are separated in $3+5$ variables, as explained in [14]. The equation $(\mathcal{A})$ come from the equation $b = a^3$ in $\mathbf{F}_{2^5}$.

**Note 2:** It is possible to show that, when $s$ is found, then $t$ is easy to find. However, an exhaustive search on $s$ would require $2^{n^2} = 2^{25}$ computations in this toy example. Our aim is to improve this complexity.

**Example of IP with one secret**

Let us consider the problem of finding an isomorphism between graphs $(I)$ and $(II)$ of figure 1.
Let $K$ be the finite field $\mathbf{F}_2$.
Let $(\mathcal{A})$ and $(\mathcal{B})$ be the two following sets of equations:

$$(\mathcal{A}) \begin{cases} y_1 = a_1a_3 + a_1a_4 + a_2a_3 + a_2a_4 + a_3a_4 \\ y_2 = a_1^2 + a_2^2 + a_3^2 + a_4^2 \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_1 = x_1x_2 + x_1x_3 + x_2x_3 + x_2x_4 + x_3x_4 \\ y_2 = x_1^2 + x_2^2 + x_3^2 + x_4^2 \end{cases}$$

The problem is to find a bijective and linear transformation $s$ such that $(x_1, ..., x_4) = s(a_1, ..., a_4)$, and such that – with this change of variables – the system $(\mathcal{A})$ becomes the system $(\mathcal{B})$ (*i.e.* such that $y_1$ becomes as written in $(\mathcal{B})$, and $y_2$ also becomes as written in $(\mathcal{B})$).
If we are able to find all the solutions of this IP problem (this is sometimes called an "IP problem with one secret" since here there is <u>one</u> affine transformation $s$ to find), then some of these solutions will be Graph Isomorphisms from Graph $(I)$ to Graph $(II)$. This comes from the fact that – in $(\mathcal{A})$ – $y_1$ was written such that the monomial $a_ia_j$ is in $y_1$ if and only if the point $i$ is linked with the point $j$ in graph $(I)$. Similarly – in $(\mathcal{B})$ – $y_1$ was written such that the monomial $x_ix_j$ is in $y_1$ if and only if the point $i$ is linked with the point $j$ in graph $(II)$. Moreover, $y_2$ was chosen to be a symmetrical polynomial of the variables, so that any graph isomorphism between graph $(I)$ and graph $(II)$ will be a solution to this particular IP problem with one secret. In section 4, we will generalize this construction to study more precisely the links between Graph Isomorphism and IP with one secret.

**Example of MP**

The variables belong to a ring or to a field $K$. Let $(\mathcal{A})$ and $(\mathcal{B})$ be the two following sets of equations:

$$(\mathcal{A}) \begin{cases} b_1 = a_1a_1' \\ b_2 = a_2a_2' \\ b_3 = a_3a_3' \\ b_4 = a_4a_4' \\ b_5 = a_5a_5' \\ b_6 = a_6a_6' \\ b_7 = a_7a_7' \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_1 = x_1x_1' + x_3x_2' \\ y_2 = x_2x_1' + x_4x_2' \\ y_3 = x_1x_3' + x_3x_4' \\ y_4 = x_2x_3' + x_4x_4' \end{cases}$$

The problem is to find two (non bijective) linear transformations $s$ and $t$ such that $(a_1, ..., a_7, a_1', ..., a_7') = s(x_1, ..., x_4, x_1', ..., x_4')$, $(y_1, ..., y_4) = t(b_1, ..., b_7)$, $s$ of rank 8, $t$ of rank 4, and such that when $(\mathcal{A})$ is satisfied and when these two transformations $s$ and $t$ are done, then $(\mathcal{B})$ is satisfied.
We can notice that the system $(\mathcal{B})$ is the system of equations of a product of two $2 \times 2$ matrices:

$$\begin{pmatrix} y_1 & y_3 \\ y_2 & y_4 \end{pmatrix} = \begin{pmatrix} x_1 & x_3 \\ x_2 & x_4 \end{pmatrix} \cdot \begin{pmatrix} x_1' & x_3' \\ x_2' & x_4' \end{pmatrix}$$

In the system $(\mathcal{A})$, we have exactly seven multiplications, so that solving the MP problem is solving the problem: "How to multiply $2 \times 2$ matrices with 7 (instead of 8) multiplications". (The number of additions, subtractions and multiplications with constants of $K$ can be high, but the number of multiplications of terms of the matrices is at most 7).
V. Strassen found in 1969 how to multiply $2 \times 2$ matrices with 7 multiplications (cf [17]). His solution is the following:

$$\begin{cases} a_1 = x_3 - x_4 \\ a_2 = x_1 + x_4 \\ a_3 = x_1 - x_2 \\ a_4 = x_1 + x_3 \\ a_5 = x_1 \\ a_6 = x_4 \\ a_7 = x_2 + x_4 \end{cases} \qquad \begin{cases} a'_1 = x'_2 + x'_4 \\ a'_2 = x'_1 + x'_4 \\ a'_3 = x'_1 + x'_3 \\ a'_4 = x'_4 \\ a'_5 = x'_3 - x'_4 \\ a'_6 = x'_2 - x'_1 \\ a'_7 = x'_1 \end{cases} \qquad \text{and} \qquad \begin{cases} y_1 = b_1 + b_2 - b_4 + b_6 \\ y_2 = b_4 + b_5 \\ y_3 = b_6 + b_7 \\ y_4 = b_2 - b_3 + b_5 - b_7 \end{cases}$$

# 4 IP with one secret is at least as difficult as Graph Isomorphism

### First links between the two problems

We first generalize the construction of section 3 about IP with one secret. Let $(I)$ and $(II)$ be two $n$-vertices graphs, and let $(\mathcal{A})$ and $(\mathcal{B})$ the following systems:

$$(\mathcal{A}) \begin{cases} y_1 = \sum\limits_{i,j} \gamma_{ij} x_i x_j \\ y_2 = \sum\limits_{i=1}^{n} x_i^2 \end{cases} \qquad \text{and} \qquad (\mathcal{B}) \begin{cases} y_1 = \sum\limits_{i,j} \mu_{ij} a_i a_j \\ y_2 = \sum\limits_{i=1}^{n} a_i^2, \end{cases}$$

where all the $\gamma_{ij}$ and $\mu_{ij}$ coefficients lie in $\{0, 1\}$ and satisfy:

$$\gamma_{ij} = 1 \Leftrightarrow \textit{The } i \textit{ and } j \textit{ vertices of graph } (I) \textit{ are linked together}$$

$$\mu_{ij} = 1 \Leftrightarrow \textit{The } i \textit{ and } j \textit{ vertices of graph } (II) \textit{ are linked together.}$$

It is easy to see that each graph isomorphism between $(I)$ and $(II)$ corresponds to a morphism of polynomials between $(\mathcal{A})$ and $(\mathcal{B})$. This morphism of polynomials is a very special one, since it can be defined by $a_i = x_{\varphi(i)}$, where $\varphi$ is some permutation of $\{1, ..., n\}$.
Reciprocally, can *any* isomorphism of polynomials between $(\mathcal{A})$ and $(\mathcal{B})$ be characterized by $a_i = x_{\varphi(i)}$ for some permutation $\varphi$ of $\{1, ..., n\}$ ? Of course not, and explicit examples can be built to be convinced. Nevertheless, we can consider the following argument: there exist $q^{n(n+1)}$ possible affine transformations between the $x_i$ and the $a_i$, among which $q^n(q^n - 1)(q^n - q)(q^n - q^2)...(q^n - q^{n-1})$ are bijective. The probability that such a transformation leaves $y_1$ invariant is *a priori* $\leq \frac{1}{q^{\frac{n(n+1)}{2}+1}}$ (because there are $\frac{n(n+1)}{2}$ monomials $x_i x_j$ $(i \leq j)$, and a constant term). The same property is *a priori* true for $y_2$. Moreover, since

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

we can think that – "most of the time" – there are very few solutions for the IP problem between $(\mathcal{A})$ and $(\mathcal{B})$ that do not correspond to a solution of Graph Isomorphism. Of course, it is a rough evaluation. Nevertheless, it suggests that if a method is found to solve the IP problem with one secret, then we should be able to use this method to also solve the Graph Isomorphism problem. We now study how to obtain a real proof of this property.

### Possible improvements

Notice that several practical solutions can be thought of to better eliminate the "unwanted" solutions of the IP problem between $(\mathcal{A})$ and $(\mathcal{B})$ (*i.e.* those which do not correspond to a graph isomorphism). For instance, we can add the equation $y_3 = x_1 + x_2 + ... + x_n$ to $(\mathcal{A})$, and similarly $y_3 = a_1 + a_2 + ... + a_n$ to $(\mathcal{B})$. More generally, any symmetrical polynomial can be added, for example $y_4 = x_1^3 + x_2^3 + ... + x_n^3$

to $(\mathcal{A})$ and $y_4 = a_1^3 + a_2^3 + ... + a_n^3$ to $(\mathcal{B})$. (In that case, we must suppose that we can solve IP not only for quadratic systems, but also for systems of higher degree).
Another *ad hoc* solution consists in adding the equation $y_3 = x_1^2$ to $(\mathcal{A})$ and $y_3 = a_k^2$ to $(\mathcal{B})$, where $k$ is randomly chosen between 1 and $n$. This way, we can expect – for some of these $k$ values – to eliminate the "unwanted" solutions, while keeping at least one solution of the Graph Isomorphism problem (but this is not a proof).

A better idea, in order to obtain real proofs, is to introduce two new variables, say $X$ and $X'$, and to consider the two following systems:

$$(\mathcal{A}) \begin{cases} y_1 = \sum\limits_{i,j} \gamma_{ij} x_i x_j \\ y_2 = (X - x_1)(X - x_2)...(X - x_n) \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_1 = \sum\limits_{i,j} \mu_{ij} a_i a_j \\ y_2 = (X' - a_1)(X' - a_2)...(X' - a_n) \end{cases},$$

where the $\gamma_{ij}$ and $\mu_{ij}$ values are defined as before.
Now, since $K[X, X', a_1, ..., a_n, x_1, ..., x_n]$ is a factorial ring, any solution of the IP problem between $(\mathcal{A})$ and $(\mathcal{B})$ is indeed a solution of the Graph Isomorphism problem (note that the two equations in $(\mathcal{A})$ and $(\mathcal{B})$ can also be combined in one equation like this: $y_1 = (X - x_1)...(X - x_n)(X - \sum_{i,j} \gamma_{ij} x_i x_j)$).
However, there is still a problem with this construction: the equations $y_2$ are of degree $n$, and if an equation of degree $n$, with $n+1$ variables, is given as a sum of monomials (expanded form), it has a non polynomial (in $n$) number of terms. We now present the real proof of the link between IP and GI, where this problem is solved, and moreover, we consider again IP on quadratic forms only.

### The real proof

We first prove a general property about permutations:

**Theorem 4.1** *Any permutation $\sigma$ of $\{1, ..., n\}$ can be written in a unique way as follows:*

$$\sigma = \tau_{i_n,n} \circ \tau_{i_{n-1},n-1} \circ ... \circ \tau_{i_1,1},$$

*where $i_k \in \{1, ..., k\}$ for all $k$, $1 \le k \le n$, and where $\tau_{i,j}$ is the permutation that exchanges $i$ and $j$ (by convention, $\tau_{i,i}$ =Id).*

**Proof:** We proceed by induction on $n$.
It is easy to check the result for $n = 1$ and $n = 2$. Let us suppose that the result is true for a given integer $n$, and let us consider a permutation $\sigma$ of $\{1, ..., n+1\}$.
Let $i_{n+1} = \sigma(n+1)$, and let $\sigma' = \tau_{i_{n+1},n+1} \circ \sigma$. The fact that $\sigma'(n+1) = n+1$ shows that $\sigma'$ induces a permutation of $\{1, ..., n\}$ (that we also denote by $\sigma'$).
¿From the induction hypothesis, there exist indices $i_1$, ..., $i_n$ satisfying $i_k \in \{1, ..., k\}$ for all $k$, $1 \le k \le n$, and such that:

$$\sigma' = \tau_{i_n,n} \circ ... \circ \tau_{i_1,1}.$$

As a result, we have:

$$\sigma = \tau_{i_{n+1},n+1} \circ ... \circ \tau_{i_1,1},$$

with $i_k \in \{1, ..., k\}$ for all $k$, $1 \le k \le n+1$.
We have thus proved the existence of the above decomposition of $\sigma$.
The unicity is a consequence of the following combinatoric argument. Let us denote by $S_n$ the set of all permutations of $\{1, ..., n\}$ and by $T$ the set of all permutations that can be written as $\tau_{i_n,n} \circ ... \circ \tau_{i_1,1}$. There are at most $n$ possibilities for $i_n$, $n-1$ possibilities for $i_{n-1}$, ..., and 1 possibility for $i_1$. Therefore, $T$ contains $\le n!$ elements. Moreover, we have just proven that the function

$$F : \begin{cases} T \to S_n \\ (\tau_{i_n,n}, ..., \tau_{i_1,1}) \mapsto \tau_{i_n,n} \circ ... \circ \tau_{i_1,1} \end{cases}$$

is surjective. Since $|T| \le |S_n|$, we can conclude that $F$ is bijective. The unicity is thus proven.

We now see how to use this theorem to "translate" the fact that the transformation $s$ involved in the NP-problem corresponding to a Graph Isomorphism instance is characterized by $a_i = x_{\varphi(i)}$ for some

permutation $\varphi \in S_n$. According to theorem 4.1, such a permutation can be written as the product of (at most) $n$ permutations $\tau_{i,j}$.
To simplify, let us first give a "translation" of the fact that an affine transformation $s : x \mapsto a$ is characterized either by $s =$Id, or by $a_i = x_{\varphi(i)}$ (for all $i$), with $\varphi = \tau_{ij}$ for two given indices $i$ and $j$. It is equivalent to saying that $s$ is an isomorphism of polynomials between the two following systems:

$$(\mathcal{A}) \begin{cases} y_0 = (X - x_i)(X - x_j) \\ y_k = x_k \qquad (1 \le k \le n,\ k \ne i, j) \\ y_{n+1} = X \end{cases} \quad \text{and} \quad (\mathcal{B}) \begin{cases} y_0 = (A - a_i)(A - a_j) \\ y_k = a_k \qquad (1 \le k \le n,\ k \ne i, j) \\ y_{n+1} = A \end{cases}$$

By using this argument several times, we obtain the following "translation" of a Graph Isomorphism problem into an IP problem: $s$ corresponds to an isomorphism of the graphs (*i.e.* in particular is of the form $a_i = x_{\varphi(i)}$ for some $\varphi \in S_n$) iff it is an isomorphism of polynomials between the two following systems:

$$(\mathcal{A}) \begin{cases} y_0 = \sum\limits_{i,j} \gamma_{ij} x_i x_j \\ \forall i, j,\ 1 \le i < j \le n,\ (i,j) \ne (n-1, n), \begin{cases} y_{(2n-1)\nu_{ij}+1} = \left(X - x_i^{(\nu_{ij})}\right)\left(X - x_j^{(\nu_{ij})}\right) \\ y_{(2n-1)\nu_{ij}+k+1} = x_k^{(\nu_{ij})} \qquad (1 \le k < i) \\ y_{(2n-1)\nu_{ij}+k} = x_k^{(\nu_{ij})} \qquad (i < k < j) \\ y_{(2n-1)\nu_{ij}+k-1} = x_k^{(\nu_{ij})} \qquad (j < k \le n) \\ y_{(2n-1)\nu_{ij}+(n-1)+k} = x_k^{(\nu_{ij}+1)} \qquad (1 \le k \le n) \end{cases} \\ \begin{cases} y_{(2n-1)(n-2)(n+1)/2+1} = \left(X - x_i^{((n-2)(n+1)/2)}\right)\left(X - x_j^{((n-2)(n+1)/2)}\right) \\ y_{(2n-1)(n-2)(n+1)/2+k+1} = x_k^{((n-2)(n+1)/2)} \qquad (1 \le k < i) \\ y_{(2n-1)(n-2)(n+1)/2+k} = x_k^{((n-2)(n+1)/2)} \qquad (i < k < j) \\ y_{(2n-1)(n-2)(n+1)/2+k-1} = x_k^{((n-2)(n+1)/2)} \qquad (j < k \le n) \end{cases} \\ y_{(2n-1)(n-2)(n+1)/2+n} = X \end{cases}$$

and

$$(\mathcal{B}) \begin{cases} y_0 = \sum\limits_{i,j} \mu_{ij} x_i x_j \\ \forall i, j,\ 1 \le i < j \le n,\ (i,j) \ne (n-1, n), \begin{cases} y_{(2n-1)\nu_{ij}+1} = \left(A - a_i^{(\nu_{ij}+1)}\right)\left(A - a_j^{(\nu_{ij}+1)}\right) \\ y_{(2n-1)\nu_{ij}+k+1} = a_k^{(\nu_{ij}+1)} \qquad (1 \le k < i) \\ y_{(2n-1)\nu_{ij}+k} = a_k^{(\nu_{ij}+1)} \qquad (i < k < j) \\ y_{(2n-1)\nu_{ij}+k-1} = a_k^{(\nu_{ij}+1)} \qquad (j < k \le n) \\ y_{(2n-1)\nu_{ij}+(n-1)+k} = a_k^{(\nu_{ij}+1)} \qquad (1 \le k \le n) \end{cases} \\ \begin{cases} y_{(2n-1)(n-2)(n+1)/2+1} = (A - a_i)(A - a_j) \\ y_{(2n-1)(n-2)(n+1)/2+k+1} = a_k \qquad (1 \le k < i) \\ y_{(2n-1)(n-2)(n+1)/2+k} = a_k \qquad (i < k < j) \\ y_{(2n-1)(n-2)(n+1)/2+k-1} = a_k \qquad (j < k \le n) \end{cases} \\ y_{(2n-1)(n-2)(n+1)/2+n} = A \end{cases}$$

where $\nu_{ij} = i - 1 + \frac{(j-1)(j-2)}{2}$ and $X$, $A$, as well as the $x_k^{(\nu)}$ and $a_k^{(\nu)}$, are intermediate variables. Each of these two systems contains $\frac{2n^3-3n^2-n+4}{2}$ equations over $\frac{(n+1)(n^2-2n+2)}{2}$ variables.

**Conclusion:** By solving IP with one secret on a set of $\mathcal{O}(n^3)$ quadratic equations we can solve any Graph Isomorphism problem with $n$ vertices. Therefore, IP is at least as hard as GI.

**Remark:**

1. In the particular case where $(\mathcal{A})$ and $(\mathcal{B})$ contain only *one* quadratic equation with $n$ variables, there exist a polynomial algorithm to find the isomorphism (see [15]).

2. We do not pretend that this construction gives a new and more efficient way to solve the Graph Isomorphism problem. In fact, although no polynomial algorithm is known for the Graph Isomorphism problem, in practice very efficient algorithms are known: for example it is feasible to find an isomorphism for graphs even for "hard" instances of 1000 vertices graphs in less than 10 minutes on a personal computer (see [6] p. 22). So the main interest of this construction is to show that the IP problem with one secret is probably not solvable with a probabilistic algorithm of polynomial complexity. (Because GI was carefully studied and many people think that GI is not solvable in polynomial complexity).

# 5 MP is NP-hard

In this section, we prove that the Morphism of Polynomials (MP) problem, defined in section 2, is NP-hard for any finite field, and for the rational numbers.
The proof uses some properties of three-dimensional tensors. Let us first recall some basic definitions:

**Definitions:**

1. A *three-dimensional tensor* is a three-dimensional array $T = (t_{ijk})$ of numbers.
2. It has *rank 1* iff it can be written as the outer product of three vectors (*i.e.* iff there exist three vectors $x$, $y$ and $z$ such that $m_{ijk} = x_i y_j z_k$ for all indices $i$, $j$, $k$).
3. The rank of a general tensor $T$ is the minimal number of rank 1 tensors $T_\nu$ such that $T = \sum_\nu T_\nu$.

The following result about the complexity of finding the rank of a three-dimensional tensor was proved by Johan Håstad in [11]:

**Theorem 5.1 (Håstad)** *Tensor rank is NP-complete for any finite field and NP-hard for the rational numbers.*

Let us now see how this fundamental property can be applied to our MP problem.
Let us suppose we have an algorithm $\Phi$ that solves the MP-problem in polynomial time (in particular, this algorithm can be used to know whether there exist a solution or not for some given instance of the MP-problem).
Let $T = (t_{ijk})$ be a $m \times n \times \ell$ (three-dimensional) tensor. It is well known (see for instance [18]) that the rank of $T$ is exactly equal to the minimal number of multiplications needed to compute the following corresponding set of bilinear forms by a bilinear non-commutative algorithm:

$$(\mathcal{B}) \begin{cases} y_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij1} x_i x'_j \\ \quad \vdots \\ y_\ell = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij\ell} x_i x'_j. \end{cases}$$

Let $r$ $(= rank(T))$ be this minimal value. By definition, $r$ can be thought as the smallest integer $u$ such that two linear transformations $s : \ (x_1, ..., x_m, x'_1, ..., x'_n) \mapsto (a_1, ..., a_u, a'_1, ..., a'_u)$ and $t : \ (b_1, ..., b_u) \mapsto (y_1, ..., y_\ell)$ exist, that transform

$$(\mathcal{A}) \begin{cases} b_1 = a_1 \cdot a'_1 \\ \quad \vdots \\ b_u = a_u \cdot a'_u. \end{cases}$$

into $(\mathcal{B})$.
This is a particular instance of the MP problem. For $u = mn$, finding a solution $(s, t)$ is quite easy. Namely, we can define $s$ and $t$ by the following formulas:

$$\forall i, \ 1 \le i \le m, \ \forall j, \ 1 \le j \le n, \ \begin{cases} a_{(i-1)n+j} = x_i \\ a'_{(i-1)n+j} = x'_j. \end{cases}$$

$$\forall k, \ 1 \le k \le \ell, \ y_k = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ijk} b_{(i-1)n+j}.$$

**Note:** Moreover, any symmetry of a given three-dimensional tensor leaves its rank unchanged, so that we have in fact:

$$r \le \min(mn, nk, km),$$

which is the analogue of the classical inequality $rank(M) \le \min(m, n)$ for a (two-dimensional) $m \times n$ matrix $M$.
The $\Phi$ algorithm can now be used to build a polynomial algorithm that computes the exact value of $r$:

1. Let $u = mn$.
2. With the help of $\Phi$, try to find two linear transformations $s:\ (x_1, ..., x_m, x'_1, ..., x'_n) \mapsto (a_1, ..., a_u, a'_1, ..., a'_u)$ and $t:\ (b_1, ..., b_u) \mapsto (y_1, ..., y_\ell)$ that transform

$$(\mathcal{A}) \begin{cases} b_1 = a_1 \cdot a'_1 \\ \quad \vdots \\ b_u = a_u \cdot a'_u. \end{cases}$$

   into $(\mathcal{B})$.
3. If $\Phi$ gives a solution, then replace $u$ by $u - 1$ and go back to step 2, else output "$r = u + 1$".

It is easy to see that this algorithm outputs the correct value of $r$ after at most $mn$ calls to the (polynomial) $\Phi$ algorithm, so that we have built an algorithm that computes the rank of a three-dimensional tensor in polynomial time. According to the result of Johan Håstad mentioned above, we can thus conclude that the MP-problem is NP-hard for any finite field, and for the rational numbers.

**Remarks:**

1. More precisely, we have just proven that the "Deciding MP" problem (*i.e.* the problem of finding whether there exist a morphism between to given sets of multivariate polynomial equations) is NP-complete. As we will see in section 6, this property becomes false if we replace "MP" by "IP".
2. MP is clearly a very important problem in mathematics: an efficient algorithm would give the computation of the minimum number of standard multiplications to compute the product of two $3 \times 3$, $4 \times 4$ or $k \times k$ matrices, for small $k$, and – from this – improved algorithms for Gaussian reductions and related problems may be found. (The best known asymptotic algorithms are at the present in $\mathcal{O}(n^c)$, where $c \simeq 2.3755$, see [4]. It is also interesting to see what kind of brute-force computations have been done so far to solve such problems, see [10].)
3. The fact that MP is NP-hard, and moreover the fact that MP is a very important problem that seems to be very difficult even with very small parameters, are strong motivations to design cryptographic algorithms based on this problem. From [2] and [7], it is known that any problem of NP can be used to design an asymmetric authentication scheme (with the help of "good" hash functions). (The proof is extendable to design asymmetric signature schemes. again with the help of "good" hash functions). Since MP is in NP, we can apply these general results. However, these very general constructions are not very practical, and it may be more difficult to design efficient schemes from MP than from IP.

## 6 Deciding IP is not NP-complete

We call "Deciding IP" the problem of finding whether there exist an isomorphism between two sets of multivariate polynomials equations. In this section, we prove that the Deciding IP problem is not NP-complete, under the classical hypothesis that the so-called "polynomial-time hierarchy" does not collapse.
The proof is based on the following general results (see [9] and [3] for proofs):

**Theorem 6.1 (Goldwasser, Sipser)** *If a problem has a constant-round interactive proof, then it also has a constant-round Arthur-Merlin protocol.*

**Theorem 6.2 (Boppana, Håstad, Zachos)** *If the complement of a problem $\Pi$ has an Arthur-Merlin protocol with a constant number of rounds, and if $\Pi$ is NP-complete, then the polynomial-time hierarchy collapses.*

**Note:** Arthur-Merlin protocols have been studied by Babai and Moran in [1], but we do not need to go into further details for our proof.

What remains to do is building a constant-round interactive proof for the "Non Isomorphism of Polynomials" (Non-IP) problem. For that purpose, we use techniques discovered by Goldwasser, Micali and Rackoff ([8]) for the analogous "Quadratic Non-Residuosity" problem, and also used by Goldreich, Micali and Wigderson ([7]) for "Graph Non-Isomorphism", and by Petrank and Roth ([16]) for "Code Non-Equivalence".
As usual, we suppose that two sets $(U_0)$ and $(U_1)$ of polynomial equations are public, and an infinitely-powerful prover (called Merlin) wants to convince a polynomial-time verifier (called Arthur) that $(U_0)$ and $(U_1)$ are not isomorphic. They can proceed as follows:

**First round:** Arthur takes $K$ numbers $\alpha_1, ..., \alpha_K$ randomly chosen in $\{0, 1\}$. For each $k$, $1 \le k \le K$, Arthur builds a set $(V_k)$ which is isomorphic to $(U_{\alpha_k})$, by choosing two random $s_k$ and $t_k$ bijective affine permutations and computing $(V_k) = t_k \circ (U_{\alpha_k}) \circ s_k$. He sends $(V_1), ..., (V_K)$ to Merlin.

**Second round:** For each $k$, $1 \le k \le K$, Merlin tries to guess $\alpha_k$, *i.e.* he tries to find whether $(V_k)$ is isomorphic to $(U_0)$ or $(U_1)$. He sends his guesses $(\beta_1, ..., \beta_K)$ to Arthur.

**Final verification:** Arthur accepts the proof iff $\beta_k = \alpha_k$ for all $k$, $1 \le k \le K$.

It is easy to see that:

1. If $(U_0)$ and $(U_1)$ are non isomorphic, Merlin never fails in convincing Arthur.
2. If $(U_0)$ and $(U_1)$ are isomorphic, the probability that Arthur is convinced that $(U_0)$ and $(U_1)$ are non-isomorphic is at most $2^{-K}$.

# Part II: New algorithms for IP

We use the same notations as in section 2 (for $u$, $n$, $q$, $(\mathcal{A})$ and $(\mathcal{B})$). In sections 7 and 8, we will design some improved algorithms for the IP problem on two sets of $u$ quadratic equations with $n$ variables. Then, in section 10, we will concentrate on the special case $u = n$.

## 7 A first algorithm for IP

Let $\alpha_1, \ldots, \alpha_u$ be $u$ values randomly chosen in $K = \mathbf{F}_q$. There is a probability $1/q^u$ that $\sum_{i=1}^{u} \alpha_i b_i = y_1 \circ s \quad (*)$.
Moreover when $(*)$ occurs it is very easy to find an affine bijection $S$ transforming $a$ into $x$ such that $(*)$ does occur (when $(*)$ is written in $a$ and $x$ instead of $b$ and $y$).
The reason for this is that there is a "canonical" representation of each equation of degree two over a finite field (cf. [12], chapter 6 for example), and from the canonical representations of $\sum_{i=1}^{u} \alpha_i b_i$ and $y_1$, when these equations are written in $a$ and $x$, it will be easy to find such a bijection $S$. What is the probability that this $S$ is indeed the right $s$ ?
If $u \neq 1$, in the matrix of the secret affine function $s$ we have $n(n+1)$ coefficients and $(*)$ gives only $1+n(n+1)/2$ quadratic equations on these coefficients if $K = \mathbf{F}_2$ and $n(n+1)/2+n+1$ is $K \neq \mathbf{F}_2$. So the probability that we will recover the right $s$ this way is expected to be approximately $1/q^{u-1}.q^{\frac{n^2+n}{2}}$ if $K = \mathbf{F}_2$ and approximately $1/q^{u-1} \cdot q^{\frac{n^2-n}{2}}$ if $K \neq \mathbf{F}_2$. Then, when $s$ is found it is then easy to recover $t$ by gaussian reductions on the coefficients of $t$.

The complexity of this algorithm to recover the secrets is expected to be $\mathcal{O}\left(q^{\frac{n^2+n+2u}{2}}\right)$ if $K = \mathbf{F}_2$ and $\mathcal{O}\left(q^{\frac{n^2-n+2u}{2}}\right)$ if $K \neq \mathbf{F}_2$ (because we try again another $S$ until we find a valid solution). This is better than the complexity $\mathcal{O}(q^{n^2+n})$ of exhaustive search on $s$.

**Note 1.** If $u = 1$ then the first part of the algorithm will easily find an $s$ solution (not necessary the right $s$) so $u$ must be $\neq 1$.

**Note 2.** It is logical to try to improve this algorithm, for example by using two equations $\sum_{i=1}^{u} \alpha_i b_i = y_1 \circ s$ and $\sum_{i=1}^{u} \alpha'_i b_i = y_2 \circ s$. However, it is not clear how to combine these equations in order to improve the algorithm. More precisely, when $u \geq 2$, we do not know any polynomial algorithm for the IP problem. Even for $u = 2$, finding such an algorithm appears to be as difficult as finding a probabilistic polynomial algorithm for the Graph Isomorphism problem, as we saw in section 4.

## 8 A second algorithm for IP (found by H. Gilbert and F. Cherbonnier)

As before, let $\mathcal{A}$ be the quadratic transformation that gives $b$ from $a$, and $\mathcal{B}$ be the quadratic transformation that gives $y$ from $x$:

$$b = \mathcal{A}(a) \ ; \quad y = \mathcal{B}(x).$$

Let $s$ and $t$ be the two secret affine transformations such that $a = s(x)$ and $y = t(b)$. Therefore:

$$b = t^{-1} \circ \mathcal{B}(x) = \mathcal{A} \circ s(x).$$

The idea is to use the fact that the components of $\mathcal{A} \circ s$ are always linear combinations of the components of $\mathcal{B}$. Moreover, this is of course also true if we consider $\mathcal{A} \circ s$ only on a small number of variables.

**Example** Let $(\mathcal{A})$ and $(\mathcal{B})$ be the following two sets of equations:

$$(\mathcal{A}) \ : \ \begin{cases} b_1 = a_1 + a_1a_5 + a_2a_3 + a_2a_4 + a_3a_4 \\ b_2 = a_3 + a_4 + a_5 + a_1a_2 + a_1a_4 + a_4a_5 \\ b_3 = a_5 + a_1a_2 + a_1a_3 + a_1a_5 + a_2a_3 + a_3a_5 + a_4a_5 \\ b_4 = a_2 + a_3 + a_4 + a_5 + a_1a_5 + a_3a_4 + a_3a_5 \\ b_5 = a_4 + a_1a_3 + a_1a_5 + a_2a_3 + a_2a_4 + a_2a_5 + a_3a_4 + a_3a_5 + a_4a_5 \end{cases}$$

($(\mathcal{A})$ comes from $b = a^3$).

$$(\mathcal{B}) \ : \ \begin{cases} y_1 = x_1 + x_3 + x_4 + x_5 + x_1x_2 + x_1x_3 + x_1x_5 + x_2x_4 + x_3x_5 + x_4x_5 \\ y_2 = x_1x_4 + x_1x_5 + x_2x_3 + x_2x_4 + x_2x_5 + x_4x_5 \\ y_3 = x_1 + x_3 + x_4 + x_1x_2 + x_1x_5 + x_2x_3 + x_3x_4 \\ y_4 = x_3 + x_4 + x_1x_2 + x_1x_5 + x_3x_4 + x_3x_5 + x_4x_5 \\ y_5 = x_2 + x_4 + x_5 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_2x_5 \end{cases}$$

The problem is to find two linear transformations $s$ and $t$ such that $a = s(x)$ and $y = t(b)$.

**Remark** This example comes from a transformation made in [14] of the "toy example" given in [13].

We will choose $x_4 = 0$ and $x_5 = 0$ (so that we obtain a vector space of dimension $6 > 5$ generated by the monomials $x_1$, $x_2$, $x_3$, $x_1x_2$, $x_1x_3$, $x_2x_3$). $(\mathcal{B})$ becomes:

$$(\mathcal{B}) \ : \ \begin{cases} y_1 = x_1 + x_3 + x_1x_2 + x_1x_3 \\ y_2 = x_2x_3 \\ y_3 = x_1 + x_3 + x_1x_2 + x_2x_3 \\ y_4 = x_3 + x_1x_2 \\ y_5 = x_2 + x_1x_3 + x_2x_3 \end{cases}$$

So the vector space generated by the equations of $(\mathcal{B})$ is the vector space generated by the 5 vectors $\{x_1,\ x_2,\ x_1x_3,\ x_2x_3,\ x_3+x_1x_2\}$.
Let $s$ be:

$$(s)\ :\ \begin{cases} a_1=\alpha_1x_1+\alpha_2x_2+\alpha_3x_3\ (+\alpha_4x_4+\alpha_5x_5)\\ a_2=\beta_1x_1+\beta_2x_2+\beta_3x_3\ (+\beta_4x_4+\beta_5x_5)\\ a_3=\gamma_1x_1+\gamma_2x_2+\gamma_3x_3\ (+\gamma_4x_4+\gamma_5x_5)\\ a_4=\delta_1x_1+\delta_2x_2+\delta_3x_3\ (+\delta_4x_4+\delta_5x_5)\\ a_5=\varepsilon_1x_1+\varepsilon_2x_2+\varepsilon_3x_3\ (+\varepsilon_4x_4+\varepsilon_5x_5)\end{cases}$$

We then perform an exhaustive search on the $\alpha_i$, $\beta_i$, $\gamma_i$, $\delta_i$, $\varepsilon_i$ $(1\le i\le 3)$ values, *i.e.* there are $2^{15}$ possibilities for these values.

**Example of value:** We try

$$\begin{cases} a_1=x_1\ (+\alpha_4x_4+\alpha_5x_5)\\ a_2=x_2\ (+\beta_4x_4+\beta_5x_5)\\ a_3=x_3\ (+\gamma_4x_4+\gamma_5x_5)\\ a_4=0\ (+\delta_4x_4+\delta_5x_5)\\ a_5=x_1+x_2\ (+\varepsilon_4x_4+\varepsilon_5x_5)\end{cases}$$

Then:

$$b_1=x_1x_2+x_2x_3.$$

However, this $b_1$ is **not** in the vector space generated by $\{x_1,x_2,x_1x_3,x_2x_3,x_3+x_1x_2\}$. As a result, this choice for the values $\alpha_i$, $\beta_i$, $\gamma_i$, $\delta_i$, $\varepsilon_i$ $(1\le i\le 3)$ can be eliminated.

**Complexity:** For about one try among 32, the five values $b_i$ $(1\le i\le 5)$ will be in the vector space generated by $\{x_1,x_2,x_1x_3,x_2x_3,x_3+x_1x_2\}$ (because each value $b_i$ has a probability about $\frac{1}{2}$ to be in this vector space). Then for these cases, we will consider $x_4\neq 0$ and we will try all the values for $\alpha_4$, $\beta_4$, $\gamma_4$, $\delta_4$, $\varepsilon_4$.
This way, we may thus hope to find the good $s$ (and $t$) after about $2^{15}$ tried values.

**General complexity of the algorithm** In general, $(\mathcal{A})$ and $(\mathcal{B})$ are two sets of the following type:

$$(\mathcal{A})\ :\ \begin{cases} b_1=P_1(a_1,...,a_n)\\ \quad\vdots\\ b_u=P_u(a_1,...,a_n)\end{cases}\quad\text{and}\quad(\mathcal{B})\ :\ \begin{cases} y_1=P'_1(x_1,...,x_n)\\ \quad\vdots\\ y_u=P'_u(x_1,...,x_n)\end{cases}$$

We will choose all the values $x_i=0$, $i>\lambda$, where $\lambda$ is such that:

$$\lambda+\frac{\lambda(\lambda-1)}{2}>u\qquad\qquad(\textit{i.e. }\lambda\simeq\sqrt{2u})$$

Therefore, the vector space generated by $x_1^2$, $x_2^2$, ..., $x_\lambda^2$, $x_1x_2$, ..., $x_ix_j$ $(1\le i<j\le\lambda)$, ..., $x_{\lambda-1}x_\lambda$ will have a dimension $>u$.
We will then perform an exhaustive search on the part of $s$ that gives $a_1$, ..., $a_n$ from $x_1$, ..., $x_\lambda$, *i.e.* on $\lambda n$ values. The complexity of this part 1 is thus expected to be about $q^{n\lambda}=q^{n\sqrt{2}\sqrt{u}}$.
Then for about one try among $\left(q^{\frac{\lambda^2+\lambda}{2}-u}\right)^u$, the $u$ values $b_i$ $(1\le i\le u)$ will be in the right vector space. Then for these cases we will continue by introducing the value $x_{\lambda+1}$. This introduces $q^n$ new possibilities, so that the typical value $\lambda$ for the complexity of this part 2 of the algorithm is such that $\frac{\lambda^2+\lambda}{2}-u\ge\frac{n}{u}$ (*i.e.* $\lambda\simeq\sqrt{2}\sqrt{\frac{n}{u}+u}$).

**Conclusion:** The complexity of this algorithm is thus expected to be at most $q^{n\lambda}=q^{n\sqrt{2}\sqrt{\frac{n}{u}+u}}$.

**Remarks**

- When $u=n$, this is $q^{\mathcal{O}(n^{\frac{3}{2}})}$, which is better than the bound $q^{\mathcal{O}(n^2)}$ of the previous algorithms.
- When $u=2$, this is also $q^{\mathcal{O}(n^{\frac{3}{2}})}$.

## 9 Some easy cases of IP with one secret

As seen in section 4, IP with one secret is at least as difficult as Graph Isomorphism. However, some instances of IP with one secret are easy to solve, as we will see now.
Let us consider for example the case of IP on two quadratic equations over a field of characteristic $p \neq 2$. We can write these equations as:

$$\begin{cases} q_1 = {}^t a A_1 a \\ q_2 = {}^t a A_2 a \end{cases} \quad \text{and} \quad \begin{cases} q_1 = {}^t x B_1 x \\ q_2 = {}^t x B_2 x \end{cases}$$

These two systems of two equations are public. The secret invertible matrix is $S$ such that $x = Sa$. The problem is to find $S$. Let us assume that $A_1$ or $A_2$ is invertible (say $A_1$ for example). Since

$$\begin{cases} A_1 = {}^t S B_1 S \\ A_2 = {}^t S B_2 S \end{cases}$$

we have (since $A_1$ is invertible):

$$B_2 S = B_1 S A_1^{-1} A_2. \qquad (*)$$

This equation $(*)$ is of <u>degree one</u> in the values of the secret matrix $S$ ! So, from $(*)$ we will obtain some informations on the matrix $S$.
However, it is not clear how to obtain similar informations on $S$ when $A_1$ and $A_2$ are not invertible (the case $p = 2$ may also create some technical difficulties, but it may be also feasible to find $S$ when $p = 2$ and $A_1$ or $A_2$ is invertible).

**Toy example 1:** Let $K = \mathbf{F}_3$ and let the public equations be:

$$\begin{cases} q_1 = 2a_1^2 + a_1a_2 + 2a_1a_3 + a_2^2 + 2a_2a_3 \\ q_2 = a_1^2 + a_1a_3 + a_2^2 + 2a_3^2 \end{cases} \quad \text{and} \quad \begin{cases} q_1 = 2x_1^2 + x_1x_2 + 2x_1x_3 + x_2^2 + 2x_2x_3 \\ q_2 = 2x_1^2 + x_1x_2 + x_1x_3 + x_2^2 + 2x_2x_3 + 2x_3^2. \end{cases}$$

So $A_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, $A_2 = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 2 \end{pmatrix}$, $B_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ and $B_2 = \begin{pmatrix} 2 & 2 & 2 \\ 2 & 1 & 1 \\ 2 & 1 & 2 \end{pmatrix}$.
Here $A_1$ and $A_2$ are invertible.
Let $S = \begin{pmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{pmatrix}$.
By writing that $B_2 S = B_1 S A_1^{-1} A_2$, we find (after some Gaussian reductions):

$$\begin{array}{lll} \alpha_1 = 0, & \alpha_2 = \ ?, & \alpha_3 = 2\alpha_2, \\ \beta_1 = 2\alpha_2, & \beta_2 = 2\alpha_2, & \beta_3 = 0, \\ \gamma_1 = 0, & \gamma_2 = \ ?, & \gamma_3 = \alpha_2. \end{array}$$

So after a few tries on $\alpha_2$ and $\gamma_2$, a solution can easily be found for $S$:

$$S = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}.$$

**Toy example 2:** Let $K = \mathbf{F}_3$ and let the public equations be:

$$\begin{cases} q_1 = 2a_1^2 + a_1a_2 + 2a_1a_3 + a_2^2 + 2a_2a_3 \\ q_2 = 2a_1^2 + a_1a_3 + a_2^2 + 2a_3^2 \end{cases} \quad \text{and} \quad \begin{cases} q_1 = 2x_1^2 + x_1x_2 + 2x_1x_3 + x_2^2 + 2x_2x_3 \\ q_2 = 2x_1x_2 + x_1x_3 + 2x_2^2 + 2x_2x_3 + 2x_3^2. \end{cases}$$

So $A_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, $A_2 = \begin{pmatrix} 2 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 2 \end{pmatrix}$, $B_1 = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ and $B_2 = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 2 \end{pmatrix}$.
Here $A_1$ is invertible, and $A_2$ is not invertible.

Let $S = \begin{pmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \end{pmatrix}$.
By writing that $B_2 S = B_1 S A_1^{-1} A_2$, we find (after some Gaussian reductions):

$$\begin{array}{lll} \alpha_1 = \ ?, & \alpha_2 = \ ?, & \alpha_3 = \ ?, \\ \beta_1 = \alpha_1 + \alpha_2 + 2\alpha_3, & \beta_2 = \alpha_1 + \alpha_3 & \beta_3 = \alpha_1 + 2\alpha_2 + 2\alpha_3, \\ \gamma_1 = \alpha_1, & \gamma_2 = 2\alpha_2 + 2\alpha_3, & \gamma_3 = \alpha_2. \end{array}$$

So after a few tries on $\alpha_1$, $\alpha_2$, $\alpha_3$, a solution can easily be found for $S$:

$$S = \begin{pmatrix} 0 & 1 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}.$$

## 10 Algorithms for IP with two secrets in $\mathcal{O}(q^{\mathcal{O}(n)})$

Further investigation of the IP problem requires to divide it into smaller subproblems. Thus, we separate the one and two secret cases, and more precisely we limit ourselves to the central case $u = n$. We call this case central because the case $u = 1$ is easily solved as most cases $u > (n+1)(n+2)/2$. As a matter of fact it becomes easy as soon as some $(n+1)(n+2)/2$ equations are independent (as a formal sum of $x_i x_j$ coefficients), so that the $u$ quadratic equations form a generating set. The probability of being so rapidly tends to 1 as $u$ grows. In that case **any** invertible $s$ allow to find at least one $t$ by Gaussian reduction and **every** two generating sets of equations are isomorphic.
We also found interesting to separate the affine part of applications $s$ and $t$, as the linear case seems a bit easier. However, since it is always possible to find the constant terms by exhaustive search in $\mathcal{O}(q^n)$, any algorithm for the IP problem with linear $s$ and $t$ with complexity $\mathcal{O}(q^{\alpha n})$ can obviously be transformed into a general algorithm for affine $s$ and $t$ with complexity $\mathcal{O}(q^{(\alpha+1)n})$.

So from now on and throughout the present section we exclusively consider the IP problem with two secret linear $s$ and $t$ transformations.
As a consequence of separation between affine and linear parts, we should suppose that $\mathcal{A}(0) = 0$ and $\mathcal{B}(0) = 0$. Otherwise we would have a completely artificial weakness, as we know that $t(\mathcal{A}(0)) = \mathcal{B}(0)$ and it gives an *a priori* knowledge on $t$.

The present section is divided into three parts (in these three parts, $s$ and $t$ are assumed to be linear). First, we will see a simple algorithm solving most cases of the IP in $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and using $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ memory. Then we will explain a very different approach which uses only polynomial memory and runs in between $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and $n^{\mathcal{O}(1)}\mathcal{O}(q^{2n})$ for all IP cases. Finally we will try to combine the ideas of these two attacks in a very powerful, 'birthday paradox'-based attack which to the best of our knowledge runs for every IP in $n^{\mathcal{O}(1)}\mathcal{O}(q^{n/2})$ with $n^{\mathcal{O}(1)}\mathcal{O}(q^{n/2})$ memory. Only the last attack uses the fact that equations given in IP are quadratic forms. The first two can operate on any function of indifferent degree.

### 10.1 A simple attack in $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ based on inversion

This attack operates in $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and uses $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ of memory space used essentially to form a complete table of $\mathcal{A}$ and $\mathcal{B}$ functions.
Let $\mathcal{A}$ be a randomly chosen quadratic equations set. We studied a probability $p_i$ of a random value $b$ of the $\mathcal{A}$ form to have $i$ possible corresponding entries $a$. We made a lot of computer simulations and we have found that the probability distribution of $p_i$ for a randomly chosen set of quadratic equations is the same as for any randomly chosen function, which can be easily shown to be $p_i = \frac{1}{ei!}$.
It allows to divide entries into classes, following their image's inversion degree. The elements of each class can be summed up and the corresponding values for $\mathcal{A}$ and $\mathcal{B}$ must correspond through the linear transformation $s$. They do not necessarily sum up to 0, since as $p_i \to 0$ we have very small classes and for a random $\mathcal{A}$ they should not sum to 0.

It is easy to show that if $n \leq q$ we would get more than $n$ equations and we could recover $s$ by Gaussian reduction. With less than $n$ equations we iterate the process as follows:

If we have found one equation $s(x^{(0)}) = a^{(0)}$ we can increase the number of classes in our partition of initial values space. We use the fact that if $s(x) = a$, we also have $s(x + x^{(0)}) = a + a^{(0)}$. Therefore we classify $a$ not only by the inversion degree of its image $\mathcal{A}(a)$ as before, but we will also consider the class of $a + a^{(0)}$. The number of classes will systematically increase.

This attack will work for any sensible non-bijective (> 2 classes) quadratic equations set. Yet we cannot solve a toy example given at the beginning of the article, since it is bijective. In order to solve these cases we hit upon another approach.

## 10.2 The "to and fro" attack

We will describe the attack on the toy example from the chapter 2, showing directly how it works. Let $q = 2$ and $n = 5$. We consider

$$\mathcal{A} : \begin{cases} b_0 = a_0 + a_0a_4 + a_1a_2 + a_1a_3 + a_2a_3 \\ b_1 = a_2 + a_3 + a_4 + a_0a_1 + a_0a_3 + a_3a_4 \\ b_2 = a_4 + a_0a_1 + a_0a_2 + a_0a_4 + a_1a_2 + a_2a_4 + a_3a_4 \\ b_3 = a_1 + a_2 + a_3 + a_4 + a_0a_4 + a_2a_3 + a_2a_4 \\ b_4 = a_3 + a_0a_2 + a_0a_4 + a_1a_2 + a_1a_3 + a_1a_4 + a_2a_3 + a_2a_4 + a_3a_4 \end{cases}$$

and

$$\mathcal{B} : \begin{cases} y_0 = x_0 + x_2 + x_3 + x_4 + x_0x_1 + x_0x_2 + x_0x_4 + x_1x_3 + x_2x_4 + x_3x_4 \\ y_1 = x_0x_3 + x_0x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_3x_4 \\ y_2 = x_0 + x_2 + x_3 + x_0x_1 + x_0x_4 + x_1x_2 + x_2x_3 \\ y_3 = x_2 + x_3 + x_0x_1 + x_0x_4 + x_2x_3 + x_2x_4 + x_3x_4 \\ y_4 = x_1 + x_3 + x_4 + x_0x_2 + x_0x_3 + x_1x_2 + x_1x_3 + x_1x_4 \end{cases}$$

as functions $\mathbf{F}_{2^5} \to \mathbf{F}_{2^5}$.
We have made a table of functions $\mathcal{A}$ and $\mathcal{B}$, where for simplification purposes we denote $\mathbf{F}_{2^5}$ elements by integers, so that to $x = (x_4, x_3, x_2, x_1, x_0)$ corresponds $x = \sum_{i=0}^{n} x_i 2^i$. Therefore if we read in the table that $\mathcal{A}(5) = 31$ it means that if $a = (a_4, a_3, a_2, a_1, a_0) = (0, 0, 1, 0, 1)$ then $x = s(a)$ equals to $(1, 1, 1, 1, 1)$.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 2 8 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\mathcal{A}$ | 0 | 1 | 8 | 15 | 10 | 31 | 23 | 4 | 26 | 25 | 3 | 6 | 9 | 30 | 5 | 20 | 14 | 18 | 22 | 12 | 24 | 16 | 21 | 27 | 2 | 28 | 11 | 19 | 13 | 7 | 17 | 29 |
| $\mathcal{B}$ | 0 | 5 | 16 | 24 | 13 | 25 | 11 | 18 | 29 | 10 | 30 | 4 | 28 | 26 | 9 | 2 | 17 | 27 | 19 | 20 | 21 | 14 | 1 | 23 | 7 | 31 | 22 | 3 | 15 | 6 | 8 | 12 |

In the real application we will not construct any table, and we will proceed with exhaustive search at every time we need to compute $\mathcal{A}^{-1}$ or $\mathcal{B}^{-1}$ which happens $\mathcal{O}(n)$ times in this attack.

The main idea is to say that if we have $k$ equations on $s$:

$$\begin{cases} s(x^{(1)}) & = & a^{(1)} \\ & \vdots & \\ s(x^{(k)}) & = & a^{(k)} \end{cases}$$

that are linearly independent, we have $q^k - 1$ dependent equations

$$\begin{cases} s(\sum \sim x^{(i)}) & = & \sum \sim a^{(i)} \\ & \vdots & \end{cases}$$

– where $\sim$ denotes some coefficients. But since $\mathcal{A}$ is generally non-linear we can get from that as much as (or almost) $q^k - 1$ independent equations 'on the other side':

$$\begin{cases} \mathcal{A}\left(s(\sum \sim x^{(i)})\right) & = & \mathcal{A}\left(\sum \sim a^{(i)}\right) \\ & \vdots & \end{cases}$$

Then we apply $t$ (formally), which changes nothing as to linear independence of these equations:

$$\left\{\begin{array}{rcl} t\left(\mathcal{A}(s(\sum \sim x^{(i)}))\right) & = & t\left(\mathcal{A}(\sum \sim a^{(i)})\right) \\ & \vdots & \end{array}\right.$$

And finally, since $\mathcal{B} = t \circ \mathcal{A} \circ s$ we have about $q^k - 1$ implicit equations on $t$:

$$\left\{\begin{array}{rcl} \mathcal{B}(\sum \sim x^{(i)}) & = & t\left(\mathcal{A}(\sum \sim a^{(i)})\right) \\ & \vdots & \end{array}\right.$$

We can do that the other direction, start from equations on $t$ and get $s$ in a very similar way, but it is less convenient since we need to compute $\mathcal{A}^{-1}$ or $\mathcal{B}^{-1}$ and sum up all possible solutions. In our example it was easy, since it is bijective.

The whole method is called "to and fro" as it proceeds toing and froing from one side to another and ends when we get $n$ independent equations on $s$ and another $n$ on $t$.

If $q \neq 2$ we have $q^k - 1 >> k$ and the attack complexity is about $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$ and $n^{\mathcal{O}(1)}\mathcal{O}(q^{2n})$ at most to find initial 1 or 2 equations.
If $q = 2$ it is $n^{\mathcal{O}(1)}\mathcal{O}(q^{2n})$ in most cases (no exceptions are known) since we can start with 2 equations.

Let us see how it works on our example:
We start with the two following equations:

$$\left\{\begin{array}{l} s(1) = 1 \\ s(2) = 7 \end{array}\right.$$

Our equation set $\mathcal{A}$ comes from $b = a^3$ in the $\mathbf{F}_{2^n}$, $n = 5$ field, and it has $n \cdot 2^n$ automorphisms of the form:

$$\left\{\begin{array}{l} s : x \mapsto \alpha x^{1+2^\beta} \\ t : x \mapsto \alpha^{-1-2^\theta} x^{2^{-\beta}} \end{array}\right. \quad \text{with } \alpha \neq 0,\ \beta \in \{0, 1, \ldots, n-1\}.$$

Therefore there are at least $n2^n$ solutions $s$ and $t$ (in fact we have found there are no more). Thus the probability of our 2 equations being right is $\frac{n2^n}{2^{2n}} = \frac{n}{2^n}$ and the whole attack complexity is only $n^{\mathcal{O}(1)}\mathcal{O}(q^n)$. Moreover, for $n = 5$ one gets $\frac{n}{2^n} \sim 0.15$ and we admit to have been lucky choosing two starting equations right. Then we get 3 dependent equations:

$$\left\{\begin{array}{l} s(1) = 1 \\ s(2) = 7 \\ s(3) = 6 \end{array}\right.$$

as – in $\mathbf{F}_{32}$ – $1 + 2 = 3$ and $1 + 7 = 6$.

Then we use the table to 'transfer' equations on the other side:

$$\left\{\begin{array}{l} t(1) = 5 \\ t(4) = 16 \\ t(23) = 24 \end{array}\right.$$

For example with $s(2) = 7$, we get $\mathcal{A}(7) = 4$ and $\mathcal{B}(2) = 16$ and all that implies that $t(4) = 16$.

Now we combine them again to get $2^3 - 1 = 7$ dependent equations:

$$\left\{\begin{array}{l} t(1) = 5 \\ t(4) = 16 \\ t(5) = 21 \\ t(18) = 13 \\ t(19) = 8 \\ t(22) = 29 \\ t(23) = 24 \end{array}\right.$$

And using $\mathcal{A}^{-1}$ and $\mathcal{B}^{-1}$ we get 7 equations on $s$:

$$\begin{cases} s(1) = 1 \\ s(2) = 7 \\ s(3) = 6 \\ s(4) = 17 \\ s(8) = 18 \\ s(20) = 14 \\ s(30) = 27 \end{cases}$$

6 from those 7 equation are actually independent, and yet it is enough to recover $s$ by Gaussian reduction. Similarly we get $t$ and verify the correctness of our solution.

$$(a_0, a_1, a_2, a_3, a_4) = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} (x_0, x_1, x_2, x_3, x_4)$$

$$(y_0, y_1, y_2, y_3, y_4) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix} (b_0, b_1, b_2, b_3, b_4)$$

If we look closely at the "to and fro" attack we can see that it can be transformed into an 'birthday paradox' or 'meet in the middle' attack with all the related speed-memory tradeoffs. This is possible because in our attack we can operate as follows:

1. First, we start with given 2 or 1 values on entries of $\mathcal{A}$ we write on the one side, and the corresponding entries for $\mathcal{B}$ we write on the other side.
2. Then on each side we compute new values as before by successive linear transformations, applying $\mathcal{A}$ or $\mathcal{A}^{-1}$ and $\mathcal{B}$ or $\mathcal{B}^{-1}$, again linear transformations etc. Thus we obtain new values which still match the respective values on the other side. We call this phenomenon "boosting" as the initial information is amplified.
3. At the end we look for a linear $s$ that maps these two (ordered) sets.

Now in order to transform the attack into the 'birthday paradox'-based attack, we need only to modify part 3. Let us assume we have $n+\varepsilon$ related entries $a^{(i)}$ for $\mathcal{A}$ and on the other side $n+\varepsilon$ related entries $x^{(i)}$ for $\mathcal{B}$. We do not need to compute $s$ to know they are really related by $s$. There is a different way: we find separately how the $n+\varepsilon$ related entries $a^{(i)}$ are linearly dependent between **themselves** and we do the same for the $n+\varepsilon$ entries $x^{(i)}$ of $\mathcal{B}$. Two sets which are related by $s$ must have the same linear dependencies. Now we can easily detect a collision in two huge lists of sets looking for the same dependencies.

The probability of error can be made as small as we want since $n+\varepsilon$ have at least $\varepsilon$ linear dependencies, and the probability of them being all true for a linearly non-related set of data grows very small with $\varepsilon$: $1/q^{\varepsilon n}$.

Our next attack explores this 'birthday' approach with several improvements, as we want to insure the possibility of starting all cases of "to and fro" with one equation and we do not want to do any exhaustive search for inversion of $\mathcal{A}$ or $\mathcal{B}$.

### 10.3 Combined power attack.

In order to further improve our attacks we found out that it would be nice to have a function which we call "boosting function" with the following properties: on an entry $x$ we compute in polynomial time $x' = F(x)$ such that $F$ is preserved by the isomorphism of polynomials:

$$\textit{If } s(x) = a,\ F_{\mathcal{B}}(x) = x' \textit{ and } F_{\mathcal{A}}(a) = a',\ \textit{then } s(x') = a'.$$

However:

**Theorem 10.1** *There is no non-trivial boosting function which works for all the quadratic equations sets.*

**Proof:** The "to and fro" attacks will almost always, apart from improbable cases, allow to recover in a unique, deterministic way, the entire $s$ and $t$ starting from 2 equations on $s$. Yet, a non-trivial boosting function would allow, with some non-zero probability, to do the same starting with only one equation.
Now, in the very precise equations set derived from $b = a^3$ it is not possible. From the structure of $b = a^3$ automorphisms shown in the last chapter we can easily see that there are at least $n$ possible solutions $s$ and $t$ satisfying any given equation on $s$.

The situation is not that bad though, since in most $q \neq 2$ cases the very simple function:

$$F(z) = \mathcal{B}(rz),\ \text{with } r \in \mathbf{F}_p$$

can give few equations on $t$ with one equation on $s$.

Therefore we must look for less powerful functions and in some way relax its axioms. We have found few ways of doing so:

1. We may consider more general functions, dealing with different sets of information on entries and results of a function.
2. Moreover we may use different boosting functions depending on the particularities $\mathcal{A}$.
3. We ask the condition of $x' = F(x)$ being preserved by the isomorphism to be true only with a certain noticeable probability.
4. We can have a function $F$ probabilistic.
5. It is enough to suppose that $x' = F(x)$ is preserved no more by **all**, but by at least by **one** possible isomorphism that satisfies the initial data.

The last two additions are important, since they are those which do demolish our proof of non-existence of boosting function.
We call a boosting function or "weak boosting function" a function which satisfies 1.-5., and the previous definition is referred to as a "strong boosting function".
In the following attack description we used 3 different boosting functions $F$, $G$ and $H$ to cover different cases of quadratic forms set $\mathcal{A}$. All these function are based on the same principle of 'differential solving' described as follows:

Since we have quadratic equations in $\mathcal{A}$, it is possible to perform a 'differential solving' of these equations, *i.e.* given a pair $c$ and $d$ one can find two entries $a$ and $a'$ such that $a - a' = c$ and $\mathcal{A}(a) - \mathcal{A}(a') = d$. Here is how we can proceed: we write the polar version of the $\mathcal{A}$ form set:

$$\mathcal{Q}(a, c) = \mathcal{A}(a + c) - \mathcal{A}(a) - \mathcal{A}(c)$$

which is bilinear in the $x_i$ and the $y_i$. Then the equation we need to solve:

$$\mathcal{A}(a + c) - \mathcal{A}(a) = d$$

is the same as

$$\mathcal{Q}(a,c) = \mathcal{A}(c) + d$$

which – once $c$ and $d$ are fixed – contains only linear equations on the $a_i$, and allows us to compute by Gaussian reduction all its solutions.

The differential solving, as described above, leads to the following boosting function:

$$Function(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x)}} x.$$

It works fine for $p \neq 2$, but it fails for $p = 2$. The problem is that if $a$ is a solution, $a+c$ is a solution as well, and we cannot either easily tell them apart, either sum them up, since $a + (a + c)$ gives $a$, which is not a new data. We will take one of these values, at random, in every pair $(x, x + z)$. The sum will be known with possible $(+z)$, and exactly with probability $1/2$ which is not bad. We put:

$$F_{\mathcal{B}}(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x) \\ \text{one only } \in\{x,x+z\}}} x.$$

The function is defined the same way for $\mathcal{A}$:

$$F_{\mathcal{A}}(c) = \sum_{\substack{a \text{ such that} \\ \mathcal{A}(a+c)=\mathcal{A}(a) \\ \text{one only } \in\{a,a+c\}}} a.$$

It is clear that the boosting function $F$ works only for non-bijective forms. For bijective forms we use another boosting function based on the same principle:

$$G_{\mathcal{B}}(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x)+\mathcal{B}(z) \\ \text{one only } \in\{x,x+z\}}} x.$$

Finally, because of the theorem we showed, neither of $F$ and $G$ can work in the particular case of $C^*$ cryptosystem quadratic forms. We use then:

$$H_{\mathcal{B}}(z) = \sum_{\substack{x \text{ such that} \\ \mathcal{B}(x+z)=\mathcal{B}(x)+\alpha_{\mathcal{B}} \\ \text{one only } \in\{x,x+z\}}} x.$$

with $\alpha_{\mathcal{B}}$ chosen at random, but as a function of $\mathcal{B}$, so it is always the same for the same $\mathcal{B}$. We do the same for $\mathcal{A}$. The $H$ works because it restricts the number of solutions $s$ we are going to find to those for which $t(\alpha_{\mathcal{A}}) = \alpha_{\mathcal{B}}$ and there are still enough of them.
In [5] we describe another boosting function $K$ which could be useful in case the present should fail on some particular quadratic form set.
We now describe the whole attack:

**The combined power attack**
This attack is designed to solve **all** the IP cases, $u = n$ and with linear $s$ and $t$ transformations, with the complexity $n^{\mathcal{O}(1)}q^{n/2}$ and using $n^{\mathcal{O}(1)}q^{n/2}$ of space. We do not know any particular function which would resist it.

1. *Initialisation.*
   We pick at random $(\log n)^{\mathcal{O}(1)}q^{n/2}$ entries $z^{(i)}$ and $(\log n)^{\mathcal{O}(1)}q^{n/2}$ entries $c^{(i)}$ (the same can be used). Thus by the well known 'birthday principle' there are at least $(\log n)^{\mathcal{O}(1)}$ collisions such that $s(z^{(i)}) = c^{(j)}$.

2. Starting (1 value $\mapsto \log\log n + \mathcal{O}(1)$ values):
   We put $z^{(i,0)} = z^{(i)}$ et $c^{(i,0)} = c^{(i)}$. Then we apply $\log\log n + \mathcal{O}(1)$ times an appropriate boosting function:
   $$z^{(i,j+1)} = F_{\mathcal{B}}(z^{(i,j)})$$

and

$$c^{(i,j+1)} = F_{\mathcal{A}}(c^{(i,j)})$$

(eventually replace $F$ by $G$ or $H$.)

On this stage, if any of the $F_{\mathcal{B}}(z^{(i,j)})$ or $F_{\mathcal{A}}(c^{(i,j)})$ is zero, we throw away the initial value $z^{(i)}$ or $c^{(i)}$. Thus we keep 1 out of about $\mathcal{O}(1)^{\log\log n+\mathcal{O}(1)} = (\log n)^{\mathcal{O}(1)}$ initial entries which is still enough.

If $p = 2$ our boosting functions values are randomly chosen from 2 possibilities and there are few possible lists $(z^{(i,j)}, j = 1..)$. Still, the number of possible lists is small: $2^{\log\log n+\mathcal{O}(1)} = \mathcal{O}(\log n)$.

Having thrown away some values, we have still about $\mathcal{O}(\log n)$ collisions $s(z^{(i)}) = c^{(j)}$. And even if $p = 2$, we can expect about $\mathcal{O}(1)$ to have made the same choices in probabilistic functions $F$, $G$, $H$. Finally, there are at least $\mathcal{O}(1)$ pairs $(z^{(i)}, c^{(j)})$ such that

$$\forall k = 1..(\log\log n + \mathcal{O}(1)), \quad S(z^{(i,k)}) = c^{(j,k)}$$

3. *Pre-Expansion (*$\log\log n + \mathcal{O}(1)$ *input values* $\mapsto$ $\log n + \mathcal{O}(1)$ *output values):*
   We apply simply $\mathcal{B}$ to all possible linear combinations of the $z^{(i)}$and we get about $q^{\log\log n+\mathcal{O}(1)} > \log n + \mathcal{O}(1)$ new values we note $t^{(i,j)}$, $j = 1..(\log n + \mathcal{O}(1))$.

   We can suppose that at least $\log n + \mathcal{O}(1)$ are independent, and we take care to have them always written in the same fixed order.

   We do the same for $\mathcal{A}$ and we get the $d^{(i,j)}$ list.

4. *Expansion (*$\log n + \mathcal{O}(1)$ *output values* $\mapsto$ $n + \mathcal{O}(1)$ *input values):*
   For every candidate $i$ we apply again the 'differential solving' principle for $\mathcal{B}$. The input difference will always be the same $z^{(i,0)}$. For the output differences we will take all the possible linear combinations of the $t^{(i,j)}$, $j = 1..(\log n + \mathcal{O}(1))$.

   Thus we get about $q^{\log n+\mathcal{O}(1)} > n + \mathcal{O}(1)$ values defined as follows:

   $$\sum_{\substack{\text{the } x \text{ such that} \\ \mathcal{B}(x+z)-\mathcal{B}(x)=\sum_j \sim t^{(i,j)} \\ \text{one only } \in\{x,x+z\}}} x.$$

   with $\sim$ being some coefficients $\in \mathbf{F}_q$.

   ¿From these values we keep $n + \varepsilon$, $\varepsilon \in \mathcal{O}(1)$ which again need to be taken always in the same ordering. We call them $z^{(i,j)}$, $j = 1..(n + \varepsilon)$.

   We do the same for $\mathcal{A}$ and it gives $c^{(i,j)}$, $j = 1..(n + \varepsilon)$.

5. *Label distribution.*
   For each list $z^{(i,j)}$, $j = 1..(n + \varepsilon)$ we are going to detect by Gaussian reduction all the possible linear dependencies. In order to easily manage the set of dependencies we will proceed in a way to get only independent linear equations, as follows: for $j = 1, 2, \dots$ we seek all the linear dependencies between $z^{(i,j)}$ and those of the $z^{(i,k)}$, $k < j$ which are non-zero and have not been found linearly dependent before.

   This list of linear dependencies, having between 2 and $n + \varepsilon$ members is a label, a signature we give to the list $z^{(i,j)}$, $j = 1..$, and actually to the sole value $z^{(i,0)}$.

As we explained before (5.2), if two lists, one for $\mathcal{A}$ and another for $\mathcal{B}$, are related by the linear application $s$, they will have the same label. If they are not, the error probability is as small as $1/q^{\varepsilon n}$. Furthermore $\varepsilon = 2$ is enough.

6. *Collision detection.*
If $p \neq 2$ we simply look for two same labels.

If $p = 2$, two lists $z^{(i,k)}$, $k = 1..$ and $c^{(j,k)}$, $k = 1..$ can be still presumed to correspond through $s$ for some $i$, $j$ to be found, but imperfectly known: with possible $+z^{(i,0)}$ for the first list and $+c^{(i,0)}$ for the second.

The probability of a $n + \varepsilon + 1$ elements list to have the same label as if some randomly chosen elements would have been added the same value $z^{(i,0)}$ is quite big: about $\frac{1}{2^\varepsilon}$ as we expect to have $\varepsilon$ linear relations in a list and it is $1/2$ for one relation. Therefore we expect that with the probability about $\frac{1}{2^{2\varepsilon}}$ we can detect if two $z^{(i,0)}$ and $c^{(j,0)}$ are related by $s$. Again $\varepsilon = 2$ is enough and we have $\frac{1}{2^{2\varepsilon}} = \frac{1}{16}$. It means that for $p = 2$ one in 16 $s$ functions we find in the present attack is correct.

Finally, in all cases we simply search for $i$ and $j$ such that $z^{(i,k)}$, $k = 0..(n + \varepsilon)$ and $c^{(j,k)}$, $k = 0..(n + \varepsilon)$ have the same label.

7. *Computing of s.*
We can then compute $s$ by Gaussian reduction on $> n$ equations on $s$: $s(z^{(i,k)}) = c^{(j,k)}$, $k = 0..(n + \varepsilon)$.

8. *Ending step.*
¿From $s$ we easily get $t$ and we verify the solution. Eventually, if $p = 2$, we need to try about 16 collisions.

We do not know any example of quadratic equations for which all the three variations of the attack would fail altogether. We did not program the whole attack so far, but we did testing on particular points that seemed to be uncertain.

## 11 Suggestions of IP variations

### IP with one secret

As we have seen above, some improved algorithms exist for IP with two secrets. Therefore, when IP is used for authentication or signature as explained in [15], it might be suggested to use IP with one secret instead of IP with two secrets in order to have a more efficient scheme. It may look surprising that IP with one secret might be a more difficult problem (with practical values of the parameters) than IP with two secrets. However, this is not so surprising: in IP with two secrets, we have about $2n^2$ unknown coefficients of $K$ (the secret values of the $s$ and $t$ matrices) and about $\frac{n^3}{2}$ quadratic equations on these unknown values (when we formally identify the two sets of equations $(\mathcal{A})$ and $(\mathcal{B})$), *i.e.* much more equations than unknowns. However, in IP with one secret, the parameters can be chosen in order that the number of equations (given by a formal identification) will be about approximately equal to the number of unknown! s. Thi s occurs for instance when $(\mathcal{A})$ and $(\mathcal{B})$ are two sets of two quadratic equations. (However, in section 9, we have seen that some instances of this problem are easy to solve.) As a result, despite the fact that there is only one affine change of variables, such a problem might be more difficult than the IP with two secrets. (IP with one secret on a single cubic equation might also be of interest.)

### Subgroups of $GL_n(K)$

Another possible idea would be to choose the secret transformations $s$ and $t$ of IP with two secrets in a subgroup $G$ of $GL_n(K)$. ($GL_n(K)$ is the set of all linear bijective transformations from $K^n$ to $K^n$). This way, $n$ may be chosen rather large (in order to make the problem difficult) and the length of the asymmetric signature (when the problem is used as explained in [15] for asymmetric signatures) might still be of reasonable size. For instance, $G$ might be the orthogonal group of some quadratic form $q$ (*i.e.* the set of all $g \in GL_n(K)$ such that $\forall x \in K^n$, $q(g(x)) = q(x)$), or $G$ might be the group of all matrices of the form $\begin{pmatrix} A & -B \\ B & A \end{pmatrix}$, where $A$ and $B$ are two $\frac{n}{2} \times \frac{n}{2}$ matrices. It is not clear whether choosing $s$ and $t$ in such a subgroup $G$ makes it easier to solve the IP problem or not.

**Remark:** A similar idea is used in the DSS or in Schnorr's algorithm, where – in order to have shorter length for the signatures – a subgroup is chosen for the exponents used in these schemes. However, here, the schemes are very different.

## 12 Conclusion

In this paper, we have shown that the MP and IP problems are not only related to the problem of finding the secret key of esoteric cryptographic algorithms, but also to very general problems such as Graph Isomorphism and Fast Matrix Multiplication. From their definitions, MP and IP look very similar but, as we have seen in this paper, Deciding MP is NP-complete whereas Deciding IP is not NP-complete (assuming the usual hypothesis about Arthur-Merlin games).
If $\mathcal{O}(n^2)$ is the length of the secrets to be found, then our new algorithms for IP ("with two secrets") have a complexity in $\mathcal{O}(q^n)$ or $\mathcal{O}(q^{n/2})$ when $s$ and $t$ are linear (so at most $\mathcal{O}(q^{\frac{3}{2}n})$ when $s$ and $t$ are affine). This is not polynomial, but this is clearly much better than the complexity $\mathcal{O}(q^{(n^2)})$ of the previous algorithms. For example, for the toy example given in [13], we have found the secret key with only about 7 computations (!), instead of $2^{25}$ for previously known algorithms (or $2^{50}$ for exhaustive search on the two secret matrices $s$ and $t$). This means that the schemes based on IP problems with two secrets (such as some the schemes described in [15]) should have larger values of the parameters than expected for security. However, there are a lot of possible variations for IP and the choice of the variations and the choice of the parameters can still be done in order to ! have both efficient asymmetric schemes and no practical attacks.

**Acknowledgements:** We thank Henri Gilbert and Frédéric Cherbonnier for giving their algorithm of section 8. We also want to thank Jean-Pierre Seifert for pointing out references [11] and [16] to us, and Don Coppersmith for pointing out a possible link between morphisms of polynomials and fast matrix multiplication.

## References

[1] László Babai, Shlomo Moran, *Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity classes*, JCSS, vol. 36, 1988, pp. 254-276.

[2] Manuel Blum, *How to prove a theorem so no one else can claim it*, Proceeedings of the International Congress of Mathematics, Berkeley CA, 1986, pp. 1444-1451.

[3] Ravi B. Boppana, Johan Håstad, Stathis Zachos, *Does co-NP have short interactive proofs*, Information Proc. Letters, vol. 25, 1987, pp. 127-132.

[4] Don Coppersmith, Shmuel Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Computation (1990), **9**, pp. 251-280.

[5] Nicolas Courtois, *Les cryptosystèmes asymétriques à représentation obscure*, Rapport de DEA, Paris 6 University, July 1997.

[6] Scott Fortin, *The Graph Isomorphism Problem*, Technical Report 93-20, University of Alberta, Edmonton, Alberta, Canada, July 1996. This paper is available at ftp://ftp.cs.ualberta.ca/pub/TechReports/1996/TR96-20/TR96-20.ps.gz

[7] Oded Goldreich, Silvio Micali, Avi Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*, Journal of the ACM, v. 38, n. 1, Jul. 1991, pp. 691-729.

[8] Shafi Goldwasser, Silvio Micali, Charles Rackoff, *The knowledge complexity of interactive proofs*, SIAM J. Comput., vol. 18, 1989, pp. 186-208.

[9] Shafi Goldwasser, Michael Sipser, *Private coins vs. public coins in interactive proof systems*, Advances in Computing Research, S. Micali (Ed.), vol. 5, 1989, pp. 73-90.

[10] John Gustafson, Srinivas Aluru, *Massively Parallel Searching for Better Algorithms or, How to Do a Cross Product with Five Multiplications*, Ames Laboratory, Department of Energy, ISU, Ames, Iowa. This paper is available at http://www.scl.ameslab.gov/Publications/FiveMultiplications/Five.html

[11] Johan Håstad, *Tensor Rank is NP-Complete*, Journal of Algorithms, vol. 11, pp. 644-654, 1990.

[12] Rudolf Lidl, Harald Niederreiter, *"Finite Fields"*, *Encyclopedia of Mathematics and its applications*, Volume 20, Cambridge University Press.

[13] Tsutomu Matsumoto, Hideki Imai, *Public quadratic polynomial-Tuples for efficient Signature-Verification and Message-Encryption*, EUROCRYPT'88, Springer-Verlag, pp. 419-453.

[14] Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public Key Scheme of Eurocrypt'88*, CRYPTO'95, Springer-Verlag, pp. 248-261.

[15] Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new Families of asymmetric Algorithms*, EUROCRYPT'96, Springer-Verlag, pp. 33-48.

[16] Erev Petrank, Ron M. Roth, *Is Code Equivalence Easy to Decide ?*, IEEE Transactions on Information Theory, 1997.

[17] Volker Strassen, *Gaussian elimination is not optimal*, Numerische Mathematik 13, 1969, pp. 354-356.

[18] Volker Strassen, *The asymptotic spectrum of tensors*, J. Reine Angew. Math., vol. 384, pp. 102-152, 1988.