An Attack on the Isomorphisms of Polynomials Problem with One Secret

Willi Geiselmann[1], Willi Meier[2], and Rainer Steinwandt[1]

[1] IAKS, Arbeitsgruppe Systemsicherheit, Prof. Dr. Th. Beth,
Fakultät für Informatik, Universität Karlsruhe,
Am Fasanengarten 5, 76 131 Karlsruhe, Germany
[2] Fachhochschule Aargau,
Klosterzelgstrasse, 5210 Windisch, Switzerland

Abstract. At EUROCRYPT '96 J. Patarin introduced the *Isomorphisms of Polynomials (IP)* problem as a basis of authentication and signature schemes [4, 5]. We describe an attack on the secret key of *IP with one secret* and demonstrate its efficiency through examples with realistic parameter sizes. To prevent our attack, additional restrictions on the suggested parameters should be imposed.

Keywords: cryptanalysis, multivariate polynomials

1 Introduction

At EUROCRYPT '96 J. Patarin introduced the problem of *Isomorphisms of Polynomials (IP)* and demonstrated how to use this problem for constructing an authentication scheme and an asymmetric signature scheme. Besides being of interest in their own, IP-based schemes are also of interest with regard to other cryptographic schemes using multivariate polynomials: as explained in [4, 5], an efficient algorithm for IP would yield a new, more powerful, attack on the Matsumoto-Imai algorithm [3], and it would have strong implications on the design and security of HFE [4, 5].

There are different variants of the IP problem, and in [6, 7] several techniques for solving the so-called *IP with two secrets* are explained. From these results the authors conclude that "when IP is used for authentication or signature ... it might be suggested to use IP with one secret instead of IP with two secrets ..., 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." As evidence for the hardness of the IP problem with one secret, in [6, 7] it is shown that this problem is as least as hard as the graph isomorphism problem. Namely, it is shown that for solving a graph isomorphism problem with n vertices, it is sufficient to solve IP with one secret on a set of $\mathcal{O}(n^3)$ quadratic equations.

In this contribution we focus on the one secret variant of IP and demonstrate that the parameter choices considered in [4, 5] do not guarantee cryptographic security: after recalling the specification of the IP problem we introduce a "column-wise" attack on the secret key of IP with one secret. We demonstrate that for several suggested parameter choices of the IP problem it is possible to reveal the secret key by means of a computer algebra system and rather modest hardware.

2 The IP problem

First we recall the specification of the IP problem introduced in [4, 5]; for sake of completeness we give the definition of both IP with two secrets and IP with one secret. By $\mathbb{F}_q$ we denote a finite field with q elements and by $\mathbb{F}_q[x] := \mathbb{F}_q[x_1, \ldots, x_n]$ the polynomial ring in the indeterminates $x = x_1, \ldots, x_n$ over $\mathbb{F}_q$.

2.1 IP with two secrets

Let $u > 1$ be an integer and $A, B \in \mathbb{F}_q[x]^u$ such that all polynomials in $A = (a_1(x), \ldots, a_u(x))$ and $B = (b_1(x), \ldots, b_u(x))$ are of total degree 2. Then we call A and B *isomorphic* if there are matrices $S_L \in \mathrm{GL}_n(\mathbb{F}_q)$, $T_L \in \mathrm{GL}_u(\mathbb{F}_q)$ and column vectors $S_C \in \mathbb{F}_q^n$, $T_C \in \mathbb{F}_q^u$ satisfying

$$\begin{pmatrix} b_1(x) \\ \vdots \\ b_u(x) \end{pmatrix} = T_L \cdot \begin{pmatrix} a_1(S_L \cdot (x_1, \ldots, x_n)^{\mathrm{t}} + S_C) \\ \vdots \\ a_u(S_L \cdot (x_1, \ldots, x_n)^{\mathrm{t}} + S_C) \end{pmatrix} + T_C \quad . \qquad (1)$$

Writing S, T for the map $z \mapsto S_L \cdot z + S_C$ and $z \mapsto T_L \cdot z + T_C$ respectively, and $\circ$ for functional composition we can summarize Equation (1) as

$$B = T \circ A \circ S.$$

The *IP problem with two secrets* can be stated as follows: given isomorphic $A, B \in \mathbb{F}_q[x]^u$ as above, find an isomorphism (S, T) from A to B.

Note: for $q = 2$ it makes a difference whether Equation (1) holds over $\mathbb{F}_q[x]$ or $\mathbb{F}_q[x]/(x_1^q - x_1, \ldots, x_n^q - x_n)$: in the latter case q^{th} powers of the variables can be reduced. From the description in [4, 5] it is not clear which definition is intended. Below we will see that an analogous distinction in IP with one secret is quite relevant for the attack considered in the sequel, and we will handle the two cases separately.

2.2 IP with one secret

Let $u > 0$ be an integer and $A, B \in \mathbb{F}_q[x]^u$ such that one of the following conditions holds:

- $u = 1$ and A, B consist of a polynomial of total degree 3 each
- $u \geq 2$ and A, B consist of polynomials of total degree 2 each

Then we call A and B *isomorphic* if there is a matrix $S_L \in \mathrm{GL}_n(\mathbb{F}_q)$ and a column vector $S_C \in \mathbb{F}_q^n$ satisfying

$$\begin{pmatrix} b_1(x) \\ \vdots \\ b_u(x) \end{pmatrix} = \begin{pmatrix} a_1(S_L \cdot (x_1, \dots, x_n)^{\mathrm{t}} + S_C) \\ \vdots \\ a_u(S_L \cdot (x_1, \dots, x_n)^{\mathrm{t}} + S_C) \end{pmatrix} . \tag{2}$$

Writing S for the map $z \mapsto S_L \cdot z + S_C$ and $\circ$ for functional composition we can summarize Equation (2) as

$$B = A \circ S.$$

The *IP problem with one secret* can be stated as follows: given isomorphic $A, B \in \mathbb{F}_q[x]^u$ as above, find an isomorphism S from A to B.

Note: similarly as in the case of two secrets, for $q \leq 3$ it makes a difference whether Equation (2) holds over $\mathbb{F}_q[x]$ or $\mathbb{F}_q[x]/(x_1^q - x_1, \dots, x_n^q - x_n)$: in the latter case q^{th} powers of the variables can be reduced. From the description in [4, 5] it is not clear which definition is intended. For the attack described in the next section this distinction will be of importance.

2.3 Cryptographic schemes using IP

In [4, 5] it has been shown how the IP problem can be used for deriving an authentication scheme and an asymmetric signature scheme. We do not recall these constructions here, as our attack aims at the underlying IP problem itself which in an IP-based scheme is assumed to be difficult. Of course, in the experimental examples we focus on parameter choices for IP which have been considered for cryptographic applications. Namely, we focus on the following parameter choices—taken from [4, 5]—for IP with one secret:

- $q = 2$ and $n \geq 16$ (with the remark "$n \geq 12$ might be sufficient...")
- $q = 16$ and $n = 6$
- $q^{\sqrt{2} \cdot n^{3/2}} > 2^{64}$

3 Attacking IP "column-wise"

The above parameter sizes do not prevent the possibility of an iteration over the q^n possible choices of one column of S_L. To see how we can exploit this, we assume for the moment that the affine part S_C of S is known already. The question of how to find S_C will be discussed in Section 4.

For our attack we distinguish two cases. First, we consider the case where q is greater than the degrees of the polynomials in A, i. e., if A consists of a cubic polynomial, then we have $q \geq 4$; for A containing several quadratic polynomials we require $q \geq 3$.

3.1 Finding a column for $q \geq 4$

The assumption $q \geq 4$ resp. $q \geq 3$ for quadratic polynomials ensures that Equation (2) holds over $\mathbb{F}_q[x]$. We want to exploit this fact by evaluating the two sides of Equation (2) at points with coordinates lying in a suitable $\mathbb{F}_q$-algebra: if our guess for (parts of) the secret key (S_L, S_C) was correct, then we must obtain the same values for the left- and for the right-hand side of Equation (2). On the other hand, if the obtained values are different then our guess was incorrect.

Why using evaluations over $\mathbb{F}_q$-algebras and not simply over $\mathbb{F}_q$? The ground field $\mathbb{F}_q$ can be quite small, and so the number of possible results (a subset of $\mathbb{F}_q^u$) when evaluating the polynomials in A or B can be rather small. In order to to decrease the chance for "random hits", we pass to evaluation points over a suitable $\mathbb{F}_q$-algebra. For avoiding potentially harmful algebraic dependencies among the coordinates, it is tempting to choose the non-zero coordinates of the evaluation points as new transcendental (over $\mathbb{F}_q$) elements. Experimentally this means that for evaluating the polynomials in A and B one makes use of (slow) arithmetics in a polynomial ring over $\mathbb{F}_q$. For sake of efficiency we decided to use a different approach: in our experiments we used the computer algebra system Magma [2] where arithmetics in finite extension fields $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$ is pretty efficient, as long as q^m is of moderate size (say $\leq 2^{20}$). So in our experiments we used evaluation points over an extension field $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$ which is large enough for making the chance of "random hits" small, but where the available arithmetics in Magma is still fast.

Now assume that some vector $v \in \mathbb{F}_{q^m}^n$ with coordinates in an extension field $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$ has been fixed. Then we know that evaluating a polynomial $b_l \in B$ at v and evaluating the corresponding polynomial $a_l \in A$ at $S(v) = S_L \cdot v^{\mathrm{t}} + S_C$ must yield the same value. In particular this holds for all vectors $v_{\alpha,i} := (0,\dots,0,\alpha,0,\dots,0)_{1\le j\le n} \in \mathbb{F}_{q^m}^n$ with a single non-zero entry $\alpha \in \mathbb{F}_{q^m}^\times := \mathbb{F}_{q^m} \setminus \{0\}$ at the i^{th} position and zero entries everywhere else.

For computing $S(v_{\alpha,i})$ it is sufficient to know the constant part S_C and the i^{th} column of S_L. Moreover, if S' is an affine map $z \mapsto S'_L \cdot z + S_C$ where the i^{th} columns of S_L and S'_L differ, then for $\mathbb{F}_{q^m}$ being "not too small", one can expect that evaluating a polynomial $a_l(x)$ in A at $S'(v_{\alpha,i})$ "typically" yields a value different from $b_l(v_{\alpha,i})$. In other words, we have a criterion for excluding some $c \in \mathbb{F}_q^n$ from the list of candidates for the i^{th} column of S_L: if for any $\alpha \in \mathbb{F}_{q^m}^\times$, $l \in \{1,\dots,u\}$ the values $b_l(v_{\alpha,i})$ and $a_l(\alpha \cdot c + S_C)$ are different, then the i^{th} column of S_L cannot equal c.

To get an idea of the practical use of this simple criterion, let us consider an example with realistic parameter sizes (as indicated above, for the computations we used the computer algebra system Magma):

Example 1. Let $q = 7$ and $n = 7$, i.e., $q^{\sqrt{2}\cdot n^{3/2}} \approx 2^{73.5} > 2^{64}$ as required. For $A := (a_1(x))$ we choose at random a polynomial of total degree 3 from $\mathbb{F}_7[x_1,\dots,x_7]$ and fix—also at random—the linear part $S_L \in \mathrm{GL}_7(\mathbb{F}_7)$ and the affine part $S_C \in \mathbb{F}_7^7$ of the secret affine map S.

Assuming S_C to be known and choosing for the non-zero coordinate α of the evaluation points $v_{\alpha,i}$ a primitive element of $\mathbb{F}_{7^7}$, experimentally we typically obtain $\approx 2^{11.2}$ possible candidates (instead of originally $\approx 2^{19.7}$) per column.

By applying the above criterion to the n columns of S_L, we can compute n subsets $C_1,\dots,C_n \subseteq \mathbb{F}_q^n$ where C_i contains the candidates for the i^{th} column of S_L. Now assume that there are two indices $1 \le i_1 < i_2 \le n$ such that the cardinality of $C_{i_1} \times C_{i_2}$ is "not too large".

Then using a brute-force search over $C_{i_1} \times C_{i_2}$ we can proceed similarly as above to reduce the number of possible candidates $(c_1, c_2) \in C_{i_1} \times C_{i_2}$ for the i_1^{st} and i_2^{nd} column of S_L. Namely, we use evaluation points of the form $v_{\alpha_1,i_1} + v_{\alpha_2,i_2} \in \mathbb{F}_{q^m}^n$ containing precisely two non-zero entries and exploit the equality

$$a_l(S_L \cdot (v_{\alpha_1,i_1} + v_{\alpha_2,i_2}) + S_C) = b_l(v_{\alpha_1,i_1} + v_{\alpha_2,i_2}) \quad (1 \le l \le u) \quad . \quad (3)$$

Evaluating the right-hand side of Equation (3) is trivial, and for evaluating the left-hand side it is clearly sufficient to know S_C and the columns no. i_1, i_2 of S_L. As above, it seems sensible to assume that if we make an incorrect guess $(c_1, c_2) \in C_{i_1} \times C_{i_2}$ for the columns no. i_1, i_2 of S_L—i.e., c_1 does not coincide with the i_1^{st} column of S_L and/or c_2 does not coincide with the i_2^{nd} column of S_L—then evaluating the two sides of Equation (3) often yields different values.

If the resulting set of candidates for the columns i_1, i_2 of S_L is "not too big" we can of course iterate the above procedure and attack a third column, say column no. i_3: for this we consider evaluation points of the form $v_{\alpha_1,i_1} + v_{\alpha_2,i_2} + v_{\alpha_3,i_3}$, i. e., we allow three non-zero coordinates. For each candidate for the columns no. i_1, i_2 we check which candidates $c_3 \in C_{i_3}$ for the i_3^{rd} column of S_L are in accordance with the condition

$$a_l(S_L \cdot (v_{\alpha_1,i_1} + v_{\alpha_2,i_2} + v_{\alpha_3,i_3}) + S_C) = b_l(v_{\alpha_1,i_1} + v_{\alpha_2,i_2} + v_{\alpha_3,i_3}) \quad (1 \leq l \leq u).$$

Of course, one now hopes that after this step there are "not too many" candidates left for the columns no. i_1, i_2, i_3 of S_L, because then one can continue in the same way. If the number of candidates for the already considered columns never becomes "too large", then in this manner finally all columns of S_L (possibly along with other isomorphisms from A to B) can be found.

Intuitively, one would expect that the most critical part of the above attack is the beginning, when only very few columns of the secret matrix are involved. Once the evaluation points contain comparatively many non-zero coordinates, "random hits" seem not to be very likely. In the practical experiment this conjecture turns out to be true. In fact, for the suggested parameters the number of candidates obtained in the first steps often remains quite modest, and the above attack turns out to work quite nicely:

Example 2. Let $q = 7$, $n = 7$ as above, and let $A = (a_1(x))$ consist of a (randomly chosen) polynomial from $\mathbb{F}_q$ of total degree 3. Assuming the affine part S_C to be known and using evaluations over $\mathbb{F}_{7^7}$, in such an example we obtained $\approx 2^{11.3}$ candidates for the first column of S_L and $\approx 2^{14}$ candidates for the first two columns. Then for the first three columns we obtained $\approx 2^{8.3}$ and for the first four columns 2 candidates. In each of the remaining rounds only one candidate was left.

On an 800 MHz Linux PC with the computer algebra system Magma, revealing the complete matrix S_L took less than two hours.

Example 3. Let $q = 16$, $n = 6$ and $A = (a_1(x), a_2(x))$ consist of two (randomly chosen) quadratic polynomials from $\mathbb{F}_q$. Again, we assume the affine part S_C to be known and for the evaluations we use elements from $\mathbb{F}_{16^5}$. Using these parameters, we could recover the (randomly chosen) matrix S_L in less than 1.5 hours with the computer algebra system Magma on a 1.3 GHz Linux PC.

So far in our attack we did not really exploit the condition $q \geq 4$ (resp. $q \geq 3$ for quadratic equations). What we actually made use of is the fact that Equation (2) holds over $\mathbb{F}_q[x]$. This equality allowed us to compare the values of the public polynomials at evaluation points with coordinates in an extension field of $\mathbb{F}_q$. But if $q \leq 3$ and Equation (2) is only valid over $\mathbb{F}_q[x]/(x_1^q - x_1, \ldots, x_n^q - x_n)$—i. e., possibly q^{th} powers of the variables have been reduced—then the above approach is no longer valid: for $\mathbb{F}_q \subsetneq \mathbb{F}_{q^m}$ a proper field extension and $\alpha \in \mathbb{F}_{q^m} \setminus \mathbb{F}_q$ we have $\alpha^q \neq \alpha$. In the next section we look at the question of how to deal with this situation.

3.2 Finding a column for $q \leq 3$

As mentioned above already, for the attack described in the previous section it is not vital that the evaluations took place over an extension field of $\mathbb{F}_q$: let $v_{\alpha_1,i_1} + \cdots + v_{\alpha_r,i_r}$ $(1 \leq r \leq n)$ be an evaluation point used throughout the attack. Then we could as well have used an evaluation point $v_{\theta_1,i_1} + \cdots + v_{\theta_r,i_r}$ where $\theta_1, \ldots, \theta_r$ are new indeterminates. Besides requiring (potentially slow) computations in $\mathbb{F}_q[\theta_1, \ldots, \theta_r]$, evaluation points of the latter form are fine for our purposes.

Now assume that Equation (2) holds over $\mathbb{F}_q[x]/(x_1^q - x_1, \ldots, x_n^q - x_n)$, and denote by X_i a representative of the residue class of x_i modulo the ideal $(x_1^q - x_1, \ldots, x_n^q - x_n) \subseteq \mathbb{F}_q[x]$ $(1 \leq i \leq n)$. Then evaluating a polynomial $b_l(x)$ in B at

$$v_{X_1,i_1} + \cdots + v_{X_r,i_r} \in (\mathbb{F}_q[x]/(x_1^q - x_1, \ldots, x_n^q - x_n))^n,$$

i. e., a point with X_j at the coordinate no. i_j $(1 \leq j \leq r)$ and zeros everywhere else, must yield the same value as evaluating the corresponding $a_l(x)$ at $S(v_{X_1,i_1} + \cdots + v_{X_r,i_r})$.

For computing effectively in $\mathbb{F}_q[x]/(x_1^q - x_1, \ldots, x_n^q - x_n)$ we can, e. g., compute in the polynomial ring $\mathbb{F}_q[x]$ and reduce all occurring polynomials modulo the Gröbner basis $\{x_1^q - x_1, \ldots, x_n^q - x_n\}$ (cf. [1, Lemma 5.66], for instance). In summary, we can try to mount the same attack as in the previous section by using evaluation points whose non-zero coordinates are equal to X_i for some $1 \leq i \leq n$.

When the public tuples A, B consist of $u \geq 2$ polynomials, this approach experimentally turns out to be quite promising:

Example 4. Let $q = 2$, $n = 16$, and let $A = (a_1(x), \ldots, a_6(x))$ consist of six polynomials. Moreover, assume that the affine part S_C is known.

In a (random) example with these parameters, using Magma on a 1.3 GHz Linux PC, the complete matrix S_L could be revealed in 2.5 hours.

Example 5. Let $q = 2$, $n = 16$, and let $A = (a_1(x), \ldots, a_7(x))$ consist of seven polynomials. Moreover, assume that the affine part S_C is known.

In a (random) example with these parameters, using Magma on a 1.3 GHz Linux PC, the complete matrix S_L could be revealed in 17 minutes.

So far in our attacks we assumed the affine part S_C to be known already. The straightforward approach for extending the above attack to an attack on the complete secret key is to iterate over all q^n possible affine parts and to apply to each candidate for S_C the above attack. Having in mind that different affine parts can be checked in parallel and that all of the above timings were obtained by means of a computer algebra system on "rather moderate" hardware platforms, for some of the proposed parameters this approach can indeed be feasible. Nevertheless, in the next section we demonstrate that the additional effort for revealing S_C is often significantly smaller than one might expect.

4 Attacking the affine part

In a first step, possible affine parts have to be checked for consistency with q^n candidates for one column, which means a search through q^{2n} pairs. In examples with small parameters we made the following observations:

- There are only very few affine parts to match any first column.
- For small q nearly all q^n candidates for the first column had a matching affine part.
- For larger q only very few matching pairs for a first column and an affine part were found.

Therefore we tried two different heuristics to avoid the search through q^{2n} candidates in a first step.

4.1 Finding the affine part for $q \geq 4$

If we choose for the evaluation vector v (as described in Section 3.1) vectors $v_{1,i} := (0, \ldots, 0, 1, 0, \ldots, 0)_{1 \leq j \leq n} \in \mathbb{F}_q^n$ with the single non-zero entry equal to $1 \in \mathbb{F}_q$ in the ground field, our test for consistency ("$b_l(v_{1,i}) = a_l(S_L \cdot v_{1,i}^{\mathrm{t}} + S_C) =: a_l(w)$") eliminates all q^n pairs $(S_L \cdot v_{1,i}^{\mathrm{t}}, S_C)$ with $(S_L \cdot v_{1,i}^{\mathrm{t}} + S_C) = w$ in case of failure. For the sums w remaining after the test, all q^n possible separations of w into $S_L \cdot v_{1,i}^{\mathrm{t}} + S_C$ have to be checked as described in Section 3.

Example 6. For $q = 16$, various choices for n and the number u of polynomials in A and B have been tested. In the following table "#sums" gives the number of sums $w = S_L \cdot v_{1,1}^{\mathrm{t}} + S_C$ remaining after the test from the originally q^n candidates for w, and "#pairs" is the number of pairs $(S_L \cdot v_{1,1}^{\mathrm{t}}, S_C)$ left after testing all separations of w. (In the case $n = 6$ and $u = 3$ the number of pairs is estimated after separating $\approx 8\%$ of the sums w.)

n	u	#sums	#pairs
3	2	18	3
4	2	251	167
5	3	16	16
6	4	279	3
6	3	4 181	$\approx$ 3 756

In a (random) example with $q = 16$, $n = 6$, and four polynomials the complete matrix S_L and the affine part S_C have been found in about 58 hours, using Magma on an 800 MHz Linux PC.

Example 7. In another (more lucky) random example with $q = 16$, $n = 6$, and $u = 3$, the complete secret key has been found in less than 7.2 hours on a 400 MHz Linux PC.

4.2 Finding the affine part for $q \leq 3$

In this case there remain too many candidates for the pairs $(S_L \cdot v_{1,1}^{\mathrm{t}}, S_C)$ after the consistency test, so that most sums occur, and the previous approach results in testing nearly q^{2n} candidates. In all our examples (where $n \leq 16$ and $u \leq 8$) the number of possible affine parts $c_0 \in \mathbb{F}_q^n$ turned out to be quite small, but the number of matching first columns $c_1 \in \mathbb{F}_q^n$ of S_L to be quite large. Thus the following probabilistic algorithm is quite promising:

1) Choose a candidate $c_1 \in \mathbb{F}_q^n$ for the first column at random, and test all affine parts $c_0 \in \mathbb{F}_q^n$ for consistency.
2) If a (new) candidate c_0 for the affine part is found, test it according to Section 3.
3) If no matching matrix S_L is found, then return to Step 1).

The efficiency of this algorithm is illustrated in the next examples:

Example 8. Let $q = 2$, $n = 16$, and let $A = (a_1(x), \ldots, a_8(x))$ consist of eight polynomials.

With these parameters we performed four runs of the algorithm above and found S_C after testing 24, 129, 153, and 234 candidates for the affine part. Completely revealing the matrix S_L and the affine part S_C took about 11 hours (for the worst case), using Magma on a 1.3 GHz Linux PC.

Example 9. Let $q = 3$, $n = 10$, i.e., $q^{\sqrt{2}n^{3/2}} \approx 2^{70.9} > 2^{64}$, and let A consist of three polynomials. In a random example with these parameters the complete secret key could be found in less than 8.1 hours on an 800 MHz Linux PC.

The following table, that is based on the experimental data from Example 8, justifies our assumption made at the beginning of this section:

#cand. first col.	#hits affine part	#affine parts
100	97	90
200	204	157
300	314	195
400	439	228
500	561	251

This table is to be interpreted as follows: after testing "#cand. first col." candidates for the first column, we found "#hits affine part" affine parts matching any of the first columns examined. After eliminating duplicates from the list we ended up with "#affine parts" candidates.

There is an obvious slow-down in finding new candidates for the affine part in the end; probably we are close to having found all of them. In other words, from the 2^{16} potential affine parts, probably only ≈ 300 have to be checked in more detail (as described in Section 3). The total number of first columns matching the 251 affine parts (mentioned in the above table) was 41 621. Thus, our probabilistic algorithm worked perfectly well.

With a decrease of the number u of polynomials the number of affine parts to test increased, e.g. to 290 for seven polynomials and to 618 for six polynomials. Testing one of these affine parts took 17 minutes and 1.5 hours respectively (on a 1.3 GHz Linux PC).

This growth in the computing time might suggest a level of security for a smaller number of polynomials that does not necessarily exist. In our algorithms there is still a lot of optimization possible: using additional conditions for the consistency test (other columns, combinations of columns, ...) might reduce the number of candidates to be checked significantly.

Furthermore, for a small number of polynomials not only the number of candidates to be checked to find S_L and S_C increases, but there might be more valid solutions of Equation (2) ("fake" keys). In smaller examples, with two public polynomials, where a complete search was possible with our equipment, we found 16 solutions for S_L and S_C with $q = 2$, $n = 7$, and 128 solutions for $q = 2$, $n = 8$.

5 Limits of the attack

We have demonstrated that for several suggested parameter choices of the IP problem our attack works quite fine. However, there are clearly limits of the method presented here: if a brute-force search over a single column of S_L is not feasible, then the attack fails. Moreover, in our experiments examples with several public polynomials and/or $q \geq 4$ turned out to be more vulnerable than examples where both $u = 1$ and $q = 2$ (with applied relations $x_1^2 - x_1, \ldots, x_n^2 - x_n$).

Consequently, a parameter choice with $u = 1$, $q = 2$ (with the relations $x_i^q - x_i$ applied), and n sufficiently large seems to be quite resistant against our attack. Of course, a larger column size q^n also results in larger keys, which may reduce the practicality of cryptographic schemes based on the difficulty of IP.

6 Conclusion

We have described a "column-wise" attack on IP with one secret. The attack has been applied successfully against several instances of IP whose parameters have been considered as suitable for cryptographic applications. The attack can be defeated by using larger keys where a brute-force search over one column of the secret matrix is infeasible, but such parameter choices also reduce the attractivity of IP-based schemes for practical applications.

References

1. T. Becker and V. Weispfenning, *Gröbner Bases: A Computational Approach to Commutative Algebra*, vol. 141 of Graduate Texts in Mathematics, Springer, New York, 1993. In cooperation with Heinz Kredel.
2. W. Bosma, J. Cannon, and C. Playoust, *The Magma Algebra System I: The User Language*, Journal of Symbolic Computation, 24 (1997), pp. 235–265.

3. T. MATSUMOTO AND H. IMAI, *Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption*, in Advances in Cryptology—EUROCRYPT '88; Workshop on the Theory and Application of Cryptographic Techniques, C. G. Günther, ed., vol. 330 of Lecture Notes in Computer Science, Springer, 1988, pp. 419–453.
4. J. PATARIN, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, in Advances in Cryptology — EUROCRYPT '96, U. Maurer, ed., vol. 1070 of Lecture Notes in Computer Science, Springer, 1996, pp. 33–48.
5. ———, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*. Extended version of [4]. At the time of writing available electronically at the URL `http://.8ngm.9.ut.rba8..ollibee.oo..rest/sct/uk/partners/page/publi/eurocrypt96b.ps`.
6. J. PATARIN, L. GOUBIN, AND N. COURTOIS, *Improved Algorithms for Isomorphisms of Polynomials*, in Advances in Cryptology — EUROCRYPT '98, K. Nyberg, ed., vol. 1403 of Lecture Notes in Computer Science, Springer, 1998, pp. 184–200.
7. ———, *Improved Algorithms for Isomorphisms of Polynomials*. Extended version of [6]. At the time of writing available electronically at the URL `http://www.minrank.org/ip6long.ps`.