Cryptanalysis of the Tractable Rational Map Cryptosystem

Antoine Joux[1], Sébastien Kunz-Jacques[2], Frédéric Muller[2], and Pierre-Michel Ricordel[2]

[1] SPOTI `Antoine.Joux@m4x.org`
[2] DCSSI Crypto Lab 51, Boulevard de La Tour-Maubourg
75700 Paris 07 SP France
{`Sebastien.Kunz-Jacques, Frederic.Muller, Pierre-Michel.Ricordel`}`@sgdn.pm.gouv.fr`

Abstract. In this paper, we present the cryptanalysis of a public key scheme based on a system of multivariate polynomial equations, the "tractable rational map" cryptosystem. We show combinatorial weaknesses of the cryptosystem, and introduce a variant of the XL resolution algorithm, the Linear Method, which is able to leverage these weaknesses to invert in short time the trapdoor one-way function defined by the cipher using only the public key, and even rebuild a private key. We also interpret the behavior of the Linear Method on random instances of the scheme, and show that various generalizations of the cipher, as well as an increase of the security parameter, cannot lead to a secure scheme.

Key words: Public Key Cryptography, Polynomial Systems, Tractable Rational Map Cryptosystem, XL, Gröbner Bases, Isomorphism of Polynomials

1 Introduction

Several recent public key cryptosystems use multivariate polynomial systems of equations instead of number-theoretic constructions. The "public" operation in such a system is to evaluate the system output on a given input value: this is a very simple operation even for devices with limited resources such as smart cards, although the system needs to be stored. The "private" operation is to find a preimage of a given value.

Determining whether a random system of n polynomial equations with n variables over any finite field has a solution is known to be a NP-complete problem, and thus seems to be a good starting point to build a cryptosystem. But the polynomial systems used in cryptographic applications must have a special form to make the solving operation possible given the knowledge of a secret backdoor. Thus the cryptanalyst does not have to solve random polynomial systems, but rather random instances of a special subfamily of polynomial systems.

The first cryptosystems based on polynomial equations were defeated. For example the Matsumoto-Imai scheme introduced in 1988 [6] was cryptanalyzed by J. Patarin (Crypto 95, [14]). More recently, some attacks against stronger schemes, such as HFE (Eurocrypt 96, [15]) or SFLASH (RSA Conference 2001, [13]), have emerged. In addition, a 80-bit HFE challenge was broken by J.-C. Faugere in 2002 [4]. It was later described by Faugère and Joux how to attack HFE using an optimized Gröbner basis algorithm and a linear algebra approach (see [7]).

All these cryptosystems share some common properties:

- They use only quadratic equations on the ground field. We can however notice that, in general, an equation of degree more than 2 is equivalent to a quadratic system with more variables.
- Public and private keys are systems of equations related by a linear or affine masking: a composition with a linear or affine transformation on the left, and a linear or affine substitution of variables on the right is performed on the private key to hide its structure, and the result constitutes the public key.

  Finding whether two random systems are equal under such a transformation is a difficult problem (it is referred to as the "Isomorphisms of Polynomials" (IP) problem, and is studied in [12]), thus the linear/affine masking seems a strong enough barrier between the public and the private key.

These cryptosystems also generally use the relation between n-variable polynomials over a field F, and univariate polynomials over an extension G of degree n of F: any system composed of n equations in n variables over F can be transformed into a unique 1-variable polynomial over G. For example, the HFE private key is a sparse polynomial over an extension of $GF(2^7)$; the function it defines can be inverted using the Berlekamp algorithm. But the system can also be expressed using several polynomials over the ground field $GF(2^7)$.

The Tractable Rational Map Cryptosystem (TRMC, [2]) also follows this framework: its private key comprises equations on various extensions of $GF(2^8)$. It is in a block triangular form: a subset of the equations can be solved, and then the result injected into other equations to further solve the system. We will show that this structure is not well hidden by a linear masking. In fact, an attacker can solve the public system using essentially the same resolution technique as the owner of the private key (of course, the resolution time will be higher, albeit still feasible in a reasonable time).

The outline of this paper is as follows: first, we present techniques used to solve systems of polynomial equations, and in particular the technique we implemented to break TRMC. Then, we introduce the cryptosystem and compute the complexity of finding preimages of some fixed values with the resolution method we have chosen. We then present our experimental results, and discuss the security of variants of the cryptostem that would use more unknowns and/or more equations. Finally, me discuss a method that can rebuild a "pseudo-private key", a system almost as easy to invert as the private key itself.

2 Algorithms for Solving Polynomial Systems

In this section, we review some known algorithms that allow to solve a system of multivariate polynomial equations. These algorithms fall into two categories: special-purpose solving algorithms that only apply to systems having a unique solution (at least without further work), and Gröbner basis algorithms.

Since the method we used in the case of the TRMC is inspired by linear algebra solving techniques and not by Gröbner basis techniques, we will not discuss this second category of algorithms extensively. We will however make a quick review of the Buchberger algorithm, which is the historical Gröbner basis algorithm, and on more recent algorithms like F_4 and F_5. Reference material on Gröbner basis can be found in [1].

From now on, we will deal only with systems having a unique solution or "zero-dimensional ideals", and deal with Gröbner basis computation algorithms only in the case of such systems. Moreover, since systems of interest for us are quadratic, we will freely assume in the description of the algorithms that we deal with sets of quadratic polynomials.

2.1 Linearization, Relinearization

Linearization is the most simple and natural resolution technique. The idea behind linearization is to consider each quadratic monomial of the system as a new unknown. If the system has n variables, this introduces at most $n(n+1)/2$ unknowns. Each equation is then viewed as a line vector of a matrix, with higher degree monomials leftmost. Then Gaussian elimination is applied to the system. If there are enough linearly independent equations, this will hopefully yield new polynomials without quadratic terms. Since the size of the matrix is $O\left(n^2\right)$, the simplest reduction algorithm has a cost of $O\left(n^6\right)$ additions and multiplications in the finite field. Note that as soon as the number of linearly independent equations exceeds the total number of quadratic monomials present in the system, the gaussian elimination will yield at least one linear polynomial in the ideal, which will allow to eliminate one unknown in the original system and to iterate the method to finish the resolution.

Unfortunately, linearization requires approximately $n^2/2$ equations, which is not suitable for most practical situations (there are systems with n equations or more that have a unique solution, therefore linearization leaves many systems with a unique solution unsolved).

Relinearization is a method introduced by A. Kipnis and A. Shamir to cryptanalyze HFE in [8], and further analyzed (and compared to XL) in [11]. It is a generalization of the linearization method that works with less equations. In fact, there are several variants of the relinearization method, that are able to cope with various lower bounds for the ratio m/n^2, where n is the number of unknowns and m the number of equations.

The simplest relinearization technique, the *fourth degree relinearization*, goes as follows. Build the linearized matrix as in the linearization method. This time, we have less linearly independent polynomials than quadratic monomials in the system, thus the matrix has a non-trivial kernel. Parameter the kernel space by new unknowns $z_1, \ldots, z_k$ ($k = \frac{n(n+1)}{2} + n + 1 - m$). Now, each quadratic monomial of the original system $x_i x_j = y_{ij}$ is viewed as a linear combination of the z_i. We can write quadratic equations on the z_i by writing compatibility equations on the y_{ij}:

$$x_i x_j x_k x_\ell = y_{ij} y_{k\ell} = y_{ik} y_{j\ell} = y_{i\ell} y_{jk}$$

We can write $2\binom{n}{4} = \dfrac{n(n-1)(n-2)(n-3)}{12}$ such equations. Thus we have about $n^4/12$ quadratic equations for the z_i. These equations can be proven to be linearly independent. If we linearize the new quadratic system, this gives us a new (relinearized) system with $n^4/12$ equations and $\left(n^2/2 - m\right)^2/2$ unknowns. This new system will have more equations than unknowns if

$$m > \left(\frac{1}{2} - \frac{1}{\sqrt{6}}\right) n^2 \approx 0.09 n^2$$

This degree 4 relinearization solves the original system if the above condition is met.

Higher degree relinearizations are able to cope with systems with less equations. They consist in writing higher degree consistency equations on the z_i, like for example for a degree 6 relinearization, $y_{ij} y_{k\ell} y_{pq} = y_{ik} y_{jp} y_{\ell q}$. Even for degree 6 relinearization, it is difficult to perfom a precise computation of the threshold m/n^2 above which systems become solvable. This is related to the fact that many consistency equations are linearly dependent, and we cannot precisely estimate the number of equations needed.

2.2 XL

XL was introduced by N. Courtois, A. Klimov, J. Patarin and A. Shamir in [11]. It relates to some works peformed by formal calculus researchers like D. Lazard (see, for example, [9]), aimed at improving the efficiency of Gröbner basis computation by using linear algebra and Gaussian reduction.

XL is partly inspired from an idea introduced to use the Buchberger algorithm to explicitly solve systems of equations having a unique solution. The Bucherger algorithm allows to eliminate monomials in the polynomials of an ideal, that is to find new polynomials of the ideal that are written using only a specific set of monomials. Thus to solve a system, one can try to eliminate all the monomials but the powers of a selected unknown of the system, say x_1. If this succeeds, this leads to at least one univariate polynomial in x_1 that is in the ideal. One can then use the Berlekamp algorithm to solve such a univariate polynomial equation, replace x_1 by its value in the original system, and run the algorithm again with the new system that has one unknown less than the original one.

Let S be a system of multivariate polynomial equations having a unique solution, I the ideal generated by the polynomials in S and $p \in I$. Since p is a sum of elements of S with polynomial coefficients, p is also a sum with scalar coefficients of all the multiples of elements of S by all monomials of degree $\leq d$, for some degree d. This applies in particular to the univariate polynomials of the ideal (we know there are such polynomials in I since S has a unique solution).

Following the preceding observations, XL looks for univariate polynomials built from the elements of S as follows. First, a monomial order is chosen where all the powers of some unkown, say x_1, come last. Then the matrix of all the polynomials that are multiple of some element in S by some monomial of degree d is built. The polynomials are mapped to lines in the matrix and each column gives the coefficient of the polynomials with respect to some particular monomial. Monomials that come first in the order are leftmost in the matrix. Then a Gaussian reduction is performed. If d is high enough, this step yields at least one non-zero univariate polynomial in x_1. The algorithm then loops as described above.

Note that at this point, there is no need for a combinatorial argument about the number of polynomials built and the number of monomials of a given degree to ensure that, for some d, we will find univariate polynomials. It suffices to see that such polynomials are in the ideal and that they can be written as polynomial combinations of elements of S.

In [11], it was proven that XL is more powerful than the relinearization algorithm, in the following sense: if a d-degree relinearization succeeds in solving a system S, then XL will also succeed by building the matrix of (total) degree d from S. Moreover, the system size of the matrix in XL will be lower than the relinearization matrix. Estimates of the complexity of XL are also given.

2.3 Gröbner bases, Buchberger, F_4, F_5

In general, a system of polynomial equations does not have a unique solution thus "solving" it does not necessarily make sense. The relevant concept is the *Gröbner basis* of a polynomial system. A Gröbner basis of an ideal is a family of polynomials of the ideal that plays the same role in the multivariate case, than the polynomial generating an ideal in the univariate case. Indeed, with a Gröbner basis of an ideal I, it can be quickly decided whether a polynomial p belongs to I or not. This is done with an euclidian division algorithm generalized to the multivariate case that reduces p on the basis. The special property of Gröbner bases is that a polynomial reduces to 0 iff it belongs to the ideal. This is not true in general for a family $\mathcal{F}$ generating an ideal I: if a polynomial reduces to 0 on $\mathcal{F}$, it belongs to the ideal (since it is a sum of elements of $\mathcal{F}$), but the converse needs not to be true.

In the case of a system having a unique solution $x_1 = a_1, \ldots, x_n = a_n$, the family $X_1 - a_1, \ldots, X_n - a_n$ generates the ideal of the system and is a Gröbner basis. More generally, any (minimal) Gröbner basis of such a system will contain only degree 1 polynomials, and there will be sufficiently many of them to recover the solution of the system. Thus Gröbner basis algorithms are of interest for us.

In the univariate case, the euclidian division crucially uses the properties of the degree. The degree enables to totally order the monomials of a polynomial and then, by only considering the leading terms of two polynomials (p_1, p_2), one can decide whether p_1 can be reduced by p_2 or not. In a similar fashion, in the multivariate case, we use monomial orderings (total, well-funded, compatible with multiplication). These monomial orderings are at the heart of reduction algorithms because they associate to each polynomial a leading monomial in a consistent way, and reduction decisions are made only by considering leading monomials.

- **The Buchberger Algorithm**
  The central notion in the Buchberger algorithm is the *S-polynomial* S formed from a pair of polynomials (p_1, p_2). S is the simplest polynomial combination of p_1 and p_2 that has a leading term strictly smaller than the least common multiple of the leading terms of p_1 and p_2. It is formed by mutiplying p_1 and p_2 by appropriate monomials so that in the sum of the results, the two leading terms cancel each other.

  A Gröbner basis has the characteristic property that all S-polynomials built upon it reduce to zero on the base; this results from the special property of Gröbner bases since S-polynomials belong to the ideal. Based on this observation, the Buchberger algorithm works as follows: starting from a polynomial family F, one builds all the S-polynomials that can be formed from F, then reduces them on F. If all polynomials reduce to zero, F is a Gröbner basis. If not, non-zero polynomials that have been found after reduction are added to F. This yields new pairs to examine. This algorithm always terminates, but the execution time and the size of the resulting basis are hard to predict; in particular, the resulting family is not in general a minimal Gröbner basis. It usually contains many redundant polynomials and can be "cleaned up".

  One of the problems of the Buchberger algorithm is that once it has built a Gröbner basis of an ideal, there are usually many pairs left to examine and the algorithm will terminate only when all these pairs have been reduced to zero. This termination phase usually represents a significant part of the computation. It is possible to avoid reducing some pairs, but we will see that in F_5 or in linear algebraic approaches, an efficient criterion can be found to avoid considering polynomials trivially reducing to zero.
- **$\mathbf{F}_4$ and $\mathbf{F}_5$**
  Both of these algorithms were engineered by J.C. Faugère and his team. F_4 was introduced in [5] and F_5 in [3]. F_4 uses some ideas from the Buchberger algorithm combined with linear algebra. Its performance is roughly equivalent to XL for a system that has a unique solution.

  F_5 is built upon F_4 but has the additional property to avoid trivial reductions to zero. This is performed by maintaining a set of known generators G of the ideal, and avoiding to form polynomial relations $gh - hg = 0$ $(g, h \in G)$. Other trivial relations may also arise from the Frobenius map of the finite field, but F_5 avoids considering them too.

3 A Variant of XL: the Linear Method

In this section, we describe the variant of XL that we implemented. We call it the Linear Method. Just like XL tries to build univariate polynomials, our method looks for linear polynomials in the ideal. Once sufficiently many (linearly independent) linear polynomials have been built, the solution of the system can be found. This purely linear approach has provable properties that will be very useful to break TRMC, even if XL might be more efficient.

3.1 Principles of the Linear Method

Let S be a set of polynomials and $I =< S >$ the ideal it generates. The basic operation of the algorithm, for a target degree d, unfolds as follows. Consider $p \in S$, of degree $d' \leq d$. Every multiple of p by a monomial m of degree $d' - d$ is in I, and of degree d. The algorithm builds a matrix description of all the polynomials of degree d obtained this way, for all $p \in S$ of degree $\leq d$ and all suitable m. Each line in the matrix describes a polynomial, and each column gives the coefficient of a particular monomial in the polynomials. Monomials of lower degree correspond to rightmost columns in the matrix. Starting with m quadratic polynomials with n variables in S, the degree d matrix has $m\binom{n-1+d-2}{d-2}$ rows and $\displaystyle\sum_{d'=0}^{d}\binom{n-1+d'}{d'}$ columns.

The matrix can then be row reduced by the Gauss algorithm. Since this reduction cancels the coefficients of the higher degree monomials in the polynomials described by the matrix, it may yield new polynomials of degree $< d$. They are in I, since they are expressed as linear combinations of polynomials of I.

The aim of the algorithm is to build linear polynomials in the ideal by building and reducing degree d matrices for various values of d. Having built and reduced the degree d matrix, what degree should we analyze next? Since reducing degrees smaller than d is far less costly than reducing degree d, one could choose to always reduce degree d' when new polynomials of degree $d' < d$ have been found during the reduction of degree d, and reduce degree $d + 1$ otherwise. Another variant would be to go into degree d' as soon as one polynomial of degree $d' < d$ is found when reducing degree d. In general, it is difficult to find an optimal strategy. Moreover, the behavior of the algorithm is heavily dependent on the structure of the system solved. For random systems, the choice of strategy is usually not so important, because no fall of degree will happen before the *critical degree* for which the corresponding matrix has mores lines than columns.

We specialized our algorithm to solve TRMC, and since we wanted to explore in detail the combinatorial behavior of the system, we did not implement any particular stategy and rather opted for a manual sequencing.

– **Numerical data for TRMC**
  With 40 variables such as in the case of the TRMC, and 48 polynomials in S, the degree 4 matrix is 39360×123410. It is only feasible to go to degree 4 or 5 on a typical 32-bit machine, e.g. a PC with 2 GB of RAM.
– **An alternative to the Gauss reduction: sparse matrix algebra**
  Let A be the the degree d matrix before Gauss reduction. The columns of A represent monomials of degree $\leq d$: split A horizontally in A_1, corresponding to monomials of degree d, and A_2, corresponding to monomials of degree $< d$. If v is a nontrivial kernel vector of ${}^tA_1$, then vA_2 represents a polynomial of I of degree $< d$. Any such v can be found using sparse algebra resolution techniques like Lanczos or Wiedemann. Since lines in A are sparse, this technique saves memory for high values of d.
– **Room for improvement**
  • Starting with degree 4, if the polynomials of the system have degree 2, the matrices that are built yield many polynomials trivially reducing to zero arising from the relations $fg - gf = 0$ (see section 5.1 for an example). This can be avoided by selectively removing some polynomial multiples when building the matrix.
  • When a linear polynomial ℓ has been found, it can be used to reduce the number of unknowns in the system by a direct susbstitution, instead of adding multiples of ℓ to the known polynomials. The same polynomials are found in both cases, but the first approach is faster and saves memory.

  In the case of TRMC however, these optimizations are not relevant since they would save very little computation time.

3.2 Properties of the Linear Method

Here, we present two key properties of the Linear Method which enable it to break TRMC. The proofs are given in annex A.

In the course of the resolution of a system, we are interested in the number of linearly independent polynomials of degree $d' < d$ that appear when reducing the degree d matrix. These falls of degree are strongly related to the ability to solve a system. The number of falls of degree that appear at all degrees d and for all sequencing choices of the algorithm when solving a system S are what we call the **combinatorial properties** of S.

Independence from Linear-Affine Masking Two systems equal up to left linear and right affine invertible transformations have the same combinatorial properties w.r.t. the linear method.

Independence from Subfield Projection If the Linear Method is able to solve a system S expressed on a finite extension G of a field F by reducing degrees less than d, it will also be able to solve S expressed on F by reducing degrees less than d.

In the case of TRMC, these properties mean that the public key and private key systems have the same combinatorial properties. In particular, the Linear Method will give the same results on both systems.

4 The Tractable Rational Map Cryptosystem

The Tractable Rational Map Cryptosystem was introduced in [2] by F. Chang and L. Wang. Its private key is a system of 48 quadratic equations with 40 unknowns over $F = GF(2^8)$. In the public key, these equations are masked by affine transformations on the left (before the polynomial system) and on the right (after the polynomial system). Some equations of the private key are derived from extensions of F, $GF(2^{16})$, $GF(2^{32})$ and $GF(2^{128})$. As in [2], we will use the notation $x_{i,\ldots,i+k-1}$ for a k-uple of elements $x_i, \ldots, x_{i+k-1}$ of $GF(2^8)$ viewed as an element of the extension $GF(2^{8k})$ of $GF(2^8)$.

The input of the private system is $x_1, \ldots, x_{40}$, and its output is $y_1, \ldots, y_{48}$. The system can be written as follows :

$$
\begin{aligned}
y_{1,2} &= q_1(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (1)\\
y_{3,4} &= q_2(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (2)\\
y_{5,6} &= q_3(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (3)\\
y_{7,8} &= q_4(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (4)\\
y_{9,10} &= q_5(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (5)\\
y_{11,12} &= q_6(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (6)\\
y_{13,14} &= q_7(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (7)\\
y_{15,16} &= q_8(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (8)\\
y_{17,18} &= q_9(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (9)\\
y_{19,20} &= q_{10}(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (10)\\
y_{21,22} &= q_{11}(x_{1,2}, x_{3,4}, x_{5,6}, x_{7,8}, x_{9,10}, x_{11,12}, x_{13,14}) && (11)\\
y_{23,\ldots,26} &= x_{15,\ldots,18}(x_{2,\ldots,5}^{256} + x_{2,\ldots,5} + a) + b\, x_{6,\ldots,9} x_{10,\ldots,13} && (12)\\
y_{27} &= x_{19} + f_1(x_1, \ldots, x_{18}) && (13)\\
y_{28} &= x_{20} + f_2(x_1, \ldots, x_{19}) && (14)\\
y_{29} &= x_{21} + f_3(x_1, \ldots, x_{20}) && (15)\\
y_{30} &= x_{22} + f_4(x_1, \ldots, x_{21}) && (16)\\
y_{31} &= x_{23} + f_5(x_1, \ldots, x_{22}) && (17)\\
y_{32} &= x_{24} + f_6(x_1, \ldots, x_{23}) && (18)\\
y_{33,48} &= x_{25,\ldots,40}(x_{7,\ldots,22}^{256} + x_{7,\ldots,22} + c) + f_{7,\ldots,22}(x_1, \ldots, x_{24}) && (19)
\end{aligned}
$$

where:

- a, b and c are random values in $GF(2^8)$,

- $f_1, \ldots, f_6$ are random quadratic polynomials over $GF(2^8)$ (the number of variables of each polynomial ranges from 18 to 23),
- $f_{7,\ldots,22}$ is a random system of 16 quadratic polynomials over $GF(2^8)$ with 24 variables,
- $q_1, \ldots, q_{11}$ are random quadratic polynomials with 7 variables over $GF(2^{16})$.

Note that in any extension of $F = GF(2^8)$, viewed as a F-vector space, $x \mapsto x^{256}$ is linear, and each multiplication coordinate is a quadratic form. Therefore all equations, including equations 12 and 19, yield quadratic equations when expressed over F.

As far as our attack is concerned, we will only retain the following aspects of the structure of the system: it has a block triangular structure; it contains a random susbsystem of 11 equations with 7 variables over $GF(2^{16})$, that must be solved first. The next equations allow to retrieve one or several variables at a time (depending on the field on which they are written).

5 Combinatorial Properties of the Public and Private Key of TRMC

By section 3.2, we know that the combinatorial properties of the public and private key of TRMC are the same. That means that the Linear Method is able to break TRMC without exploring a higher degree than the one needed to solve the private key system expressed over $GF(2^8)$. In this section, we show that the private key system can be solved by analyzing degrees ≤ 4.

We first review the behavior of the Linear Method on the subsystem of 11 equations with 7 variables over $GF(2^{16})$.

We do not have to consider the role of field equations ($x^{|k|} - x = 0$) since the maximum degree of the polynomials we will consider, 4, is less than the size of the smallest field considered, $GF(2^8)$.

5.1 Resolution of the Subsystem Over $GF(2^{16})$

Let us compute the number of monomials of a given degree with 7 variables. There are $\binom{n-1+d}{d}$ monomials of degree d with n variables, thus we have

degree	1	2	3	4
monomials	7	28	84	210

We also need the number of polynomials of a given degree d that can be formed from the 11 original polynomials:

degree	2	3	4
polynomials	11	77	308

Suppose now we try to build linear polynomials by multiplying the original polynomials by quadratic polynomials. Will we find some? This is equivalent to saying that we are looking for linear combinations of the 11+77+308=396 polynomials of the preceding table, that are linear. We thus need to cancel 210+84+28=322 terms in these polynomials. Unfortunately, our 396 polynomials are not linearly independent, because there are $\binom{11}{2} = 55$ relations in degree 4 of the form $fg - gh = 0$ with $g \neq h$ belonging to the set of the original polynomials. This leaves us with 396-55-322=19 linear polynomials. They cannot be linearly independent. Indeed, since we only have 7 variables and since our system has a unique solution, the dimension of the linear polynomials in the system is 7. Thus we have 12 more cancellations, which are in fact caused by the redundancy of the original equations (with very high probability, not all of the equations of the original system are needed for the system to have a unique solution).

We know by section 3.2 that when translated over $GF(2^8)$, the subsystem, which becomes a system of 22 equations with 14 variables, is still solvable by exploring degrees 4, 3, 2 then 1.

These theoretical observations are confirmed when running our algorithm on such a system. Building and reducing the matrices of degrees 4, then 3, 2 and 1 of the system expressed over $GF(2^{16})$ yields the solution as expected. The total number of cancellations occuring during the computation is 67, which corresponds to the 55 "$fg - gh$" cancellations and the 12 redundancy cancellations. Over $GF(2^8)$, the system can be solved as well, but the number of cancellations observed (255) is higher than 2×67, because many parasistic redundant equations are induced by the projections.

5.2 Behavior of the Linear Method over the Full Private Key

Because of the results of section 5.1, when reducing degrees 4, 3, 2, then 1, we expect to find 14 linearly independent linear polynomials in the ideal. Then, by adding multiples of these polynomials to the degree 2 matrix (which amounts to using these relations to reduce the number of variables in the original system), and reducing degree 2 again, we expect to find 4 more linear polynomials (because once $x_1, \ldots, x_{14}$ have been found and their value substituted into the system, equation 12 becomes linear in $x_{15}, \ldots, x_{18}$). Substituting into the degree 2 polynomials and reducing will yield x_{19}, and so on. In our experiments, this phenomenon was observed as predicted. The only surprise is that we do not have to know $x_1, \ldots, x_{24}$ to get a partial information on $x_{25}, \ldots, x_{40}$. Indeed, once $x_1, \ldots, x_{22}$ are known, equation 19 becomes linear in $x_{25}, \ldots, x_{40}$, and thus equations 17, 18 and 19 form a set of 18 equations over $GF(2^8)$ with 18 linear terms and only 3 quadratic terms (x_{23}^2, $x_{23}x_{24}$, and x_{24}^2): their reduction gives 15 linear polynomials, among which there is x_{23}, one linear combination of $x_{24}, \ldots, x_{40}$ and 13 linear combinations of $x_{25}, \ldots, x_{40}$. The next step is identical except that x_{23} and x_{23}^2 are now constants; there is now only one quadratic term left, instead of 3, and thus reduction gives x_{24} and another linear combination of $x_{25}, \ldots, x_{40}$. The last step yields the last linear relation needed to compute the values of $x_{25}, \ldots, x_{40}$.

6 Experimental Results and Complexity Estimates

6.1 Linear Method used Over Instances of TRMC Public Keys

To be in a realistic cryptanalysis situation, we built a random public key with Magma [10], computed the image of a random vector by the public key, and built the system composed of the value obtained substracted to the key. We then tried to solve the resulting system using the Linear Method[3].

The resolution process follows exactly the steps described in subsection 5.2: reduction of degrees 4 downto 1 yields 14 linear polynomials, and then we only have to loop between degrees 1 and 2 to get 4, then 1, 1, 1, 1, 15, 2 and 1 linear equations. The longest step is the degree 4 reduction, which we have performed using a lanczos algorithm. The computation time of the lanczos algorithm is proportional to the number of vectors computed. Thus, instead of looking for degree 3 polynomials in the degree 4 matrix M, we tried to build directly quadratic polynomials: this gave us 23 polynomials instead of the 273 cubic polynomials that can be buit from M (these are experimental figures obtained from experiments on a system of 11 equations with 7 variables on $GF(2^{16})$ and expressed over $GF(2^8)$). Overall, the lanczos resolution took 5 hours on a cluster of 6 bi-pentium IV PCs and used 400MB of RAM on each machine (data was duplicated on every machine). In that case, a Gauss reduction would have probably been faster but broke the 2GB per process limit, and could not be implemented simply on a 32-bit PC.

The other steps are performed in a few minutes on an average PC.

6.2 Asymptotic Security of TRMC

Here, we estimate the computation time ratio between the legitimate user of the system and the cryptanalyst who tries to decrypt a message, first for the "plain" TRMC algorithm, and then in the asymtptotic limit of a generalized TRMC with more variables and equations.

The preimage computation method suggested by the authors of TRMC is to solve first the random subsystem using XL, and then to substitute the result into the other equations. Using the Linear Method instead of XL, solving the random subsystem S requires to build the degree 4 matrix from the 11 equations with 7 variables of S. The complexity of a legitimate inversion is thus roughly equal to the computation of the kernel of a $\left[11 * \binom{7-1+2}{2} = 308\right] \times \left[\binom{7-1+4}{4} = 210\right]$ matrix. On the other hand, the cryptanalyst must deal with 48 equations with 40 unknowns, and thus compute the kernel of a matrix of size 39360×123410. Suppose this computation is performed using a Gauss reduction, and that the cost of a reduction of a $a \times b$ matrix is $a^2 b$, then the complexity ratio between the cryptanalyst and the legitimate user is about 2^{23}.

[3] The resulting system is guaranteed to have at least one solution (the random vector) but combinatorial arguments show that this solution is very likely to be unique. Thus we can apply the Linear Method

Now, put TRMC in the following more general setting: suppose we have a random quadratic system of m equations with n variables that can be solved by building and reducing matrices of degree less than or equal to d with the Linear Method, and that this subsystem is embedded in a block triangular system of $m' > m$ equations with $n' > n$ variables. Then the Linear Method is able to solve the big system by iterative explorations of polynomials of degree $\leq d$ built from the m' equations with n' variables.

For the legitimate user, the biggest matrix that must be built is

$$m\binom{n-1+d-2}{d-2} \times \binom{n-1+d}{d} \approx mn^{d-2} \times n^d$$

For the cryptanalyst, it is

$$m'\binom{n'-1+d-2}{d-2} \times \binom{n'-1+d}{d} \approx m'n'^{d-2} \times n'^d$$

.

At degree d since the systems can be solved, we have $mn^{d-2} \geq n^d$ and $m'n'^{d-2} \geq n'^d$. Thus with

Note that since the system has more equations than unknowns, not every value has a preimage by the system; this is why we had to compute first an image value.a Gaussian elimination algorithm as before, the ratios of the running times is $\leq \left(\frac{m'n'^{d-2}}{n^d}\right)^3$. This rough estimate is sufficient to show that an increase in the number of variables of the big system, n', increases the overall security of the scheme at most polynomially in n'.

In this analysis, we did not consider the degrees of the field extensions involved as a security parameter. The idea to use extensions of variable degree is used, for instance, in HFE, and is analyzed in [7]. Although the authors of [2] do not explicitly state what the security parameter of TRMC is, the algorithm does not seem to be designed with extensions of variable degree in mind.

7 Computing a Pseudo-Private Key

Here, we show that the knowledge of the combinatorial properties of the system of equations of the public key allows the attacker to build a system equivalent to the public or the private key and that has the block triangular form of the private key. Although this pseudo-private key is not necessarily equal to the private key, it enables the attacker to speed up further attacks.

As we saw in section 5.2 and 6.1, linear equations are computed in several passes during the course of the resolution. For example, the first group of linear equations obtained corresponds to the innermost subsystem hidden in the public key. This subsystem can be extracted from the public key in the following way. Let S bet the first group of 14 linear equations obtained during resolution, with their constant part removed. Complete S with other linear equations to obtain an invertible linear system with 40 variables. Apply the inverse of this change of variables to the public key. Let us call the new variables $z_1, \ldots, z_{40}$, with z_1 to z_{14} corresponding to elements of S. In the resulting system, there are 22 linear combinations of the equations that only depend on the $z_1, \ldots, z_{14}$. These equations can be computed by a Gaussian elimination on this system, by putting linear and quadratic monomials depending only on $z_1, \ldots, z_{14}$ leftmost.

Since $z_1, \ldots, z_{14}$ are only equal to $x_1, \ldots, x_{14}$ up to an invertible linear transformation, the resulting subsystem is not necessarily equal to the subsystem of the private key.

We can iterate this method to further mimic the structure of the original system, but the main interest of this technique is to recover the random subsystem up to a linear transform.

Each preimage computation now requires from the cryptanalyst to find the kernel of the degree 4 matrix built from 22 polynomials with 14 variables, a matrix that is 2310×2380. This computation is roughly 2^9 times slower than a legitimate preimage computation.

8 Conclusion

In this article, we performed a practical and full cryptanalysis of a public key scheme using sets of polynomial equations over finite field, the Tractable Rational Map Cryptosystem. To do so, we used a variant of the XL algorithm which we call the Linear Method. Our cryptanalysis is two-staged. A first resolution step is performed using the Linear Method to find a preimage of some value; depending on the usage of TRMC, this might correspond for example to a signature forgery or to a decryption of some message. This operation has a cost of 2^{23} legitimate preimage computations. Using its result and additional information about the process of the computation, we can then build a pseudo-private key that reduces the cost of finding a new preimage to only 2^9 legitimate preimage computations.

We also showed that the very principle of TRMC is flawed in that its security parameter cannot be reasonably increased to make it secure.

The Linear Method behaves identically on a system whether it is masked by linear or affine transformations or not. These masking techniques are used to separate the public key from the private key not only in TRMC but also in well-known schemes such as HFE or sFLASH. As with other cryptanalysis techniques like relinearization ([8]), the difficulty in breaking HFE with the linear method comes from the combination of a projection on a subfield and a linear masking, and not from the linear masking alone.

References

1. W. Adams and P. Loustaunau. *An introduction to Gröbner Bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society, 1994.
2. L. Wang F. Chang. Tractable Rational Map Cryptosystem. Cryptology ePrint archive, Report 2004/046, available at `http://eprint.iacr.org`.

3. J.-C. Faugère. A New Efficient Algorithm for Computing Gröbner Bases without reduction to zero (F_5). In T. Mora, editor, *ISSAC 2002*, pages 75–83, 2002.
4. J.-C. Faugère. Report on a Successful Attack of HFE Challenge 1 with Gröbner Basis Algorithm F5/2. Announcement on sci.crypt newsgroup, in April 19th 2002.
5. J.-C. Faugère. A New Efficient Algorithm for Computing Gröbner Bases (F_4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
6. T. Matsumoto H. Imai. Public Quadratic Polynomial-tuples for Efficient Signature Verification and Message Encryption. In C. G. Günther, editor, *Advances in Cryptology - Eurocrypt'88*, volume 330 of *LNCS*, pages 419–453. Springer Verlag, 1988.
7. A. Joux J.-C. Faugère. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In D. Boneh, editor, *Advances in Cryptology - Crypto'2003*, volume 2729 of *LNCS*, pages 44–60. Springer Verlag, 2003.
8. A. Kipnis and A. Shamir. Cryptanalysis of the HFE Public-key Cryptosystem. In M. Wiener, editor, *Advances in Cryptology - Crypto'99*, volume 1666 of *LNCS*, pages 19–30. Springer Verlag, 1999.
9. D. Lazard. Gröbner Basis, Gaussian Elimination and Resolution of Systems of Algebraic Equations. In J. A. van Hulzen, editor, *EUROCAL '83, European Computer Algebra Conference*, volume 162 of *LNCS*, pages 146–156. Springer Verlag, 1983.
10. The magma home page. `http://www.maths.usyd.edu.au/u/magma`.
11. J. Patarin N. Courtois, A. Klimov and A. Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In B. Preneel, editor, *Advances in Cryptology - Eurocrypt'2000*, volume 180 of *LNCS*, pages 392–407. Springer Verlag, 2000.
12. J. Patarin N. Courtois, L. Goubin. Improved Algorithms for Isomorphisms of Polynomials. In K. Nyberg, editor, *Advances in Cryptology - Eurocrypt'98*, volume 1403 of *LNCS*, pages 184–200. Springer-Verlag, 1998.
13. J. Patarin N. Courtois, L. Goubin. Flash, a Fast Multivariate Signature Algorithm. In D. Naccache, editor, *The Cryptographers' Track at RSA Conference 2001*, volume 2020 of *LNCS*, pages 298–307. Springer-Verlag, 2001.
14. J. Patarin. Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88. In D. Coppersmith, editor, *Advances in Cryptology - Crypto'95*, volume 963 of *LNCS*, pages 248–261. Springer Verlag, 1995.
15. J. Patarin. Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In *Advances in Cryptology - Eurocrypt'96*, volume 1070 of *LNCS*, pages 33–48. Springer Verlag, 1996.

A Proofs of the Properties of the Linear Method

In this section, we prove the two results stated in section 3.2.

A.1 Notions of *d*-relations and depth

Let $S = \{p_i\}$ be a finite set of polynomials. We are looking for the existence of *d-relations* of the form

$$\sum m_j e_j = p \quad (1)$$

where $\forall\, j, \quad \deg(m_j e_j) \leq d$. p is the *result* of the relation. The polynomials e_j are either elements of S or results of other d-relations, thus relation results always belong to the ideal generated by S, denoted $< S >$. Note that substituting one d-relation into another yields a e-relation for $e > d$, that is not in general a d-relation. This means that such a relation can be found in two passes by exploring degrees at most d, but in one pass by exploring degree e. We introduce the notion of depth that captures the number of degree d explorations needed to compute a polynomial as element of $< S >$.

Let $p \in < S >$. The *d-depth* of p is defined recursively as follows: elements of S have d-depth 0. If p is obtained by a d-relation $\sum\limits_j m_j e_j$,

$$\mathrm{depth}_d(p) = 1 + \max_j \mathrm{depth}_d(e_j)$$

If p is the multiple of a depth k polynomial by a monomial, then $\mathrm{depth}(p) = k$.

The depth of a polynomial p might not be uniquely defined. This is because there may be several sequences of reductions (relations) and multiplications that lead to p. In this case, we define the depth of p as the minimum of all depths of p.

The d-depth of a polynomial p is thus the minimal number of d-relations required to construct p as an element of $< S >$. Some polynomials p in $< S >$ might never be reached through d-relations, and for these p we set $\mathrm{depth}_d(p) = \infty$.

The depth is an useful tool to perform recursions on relations.

A.2 Behavior of the Linear Method With Respect to Linear of Affine Masking

Here we prove that two systems equal up to left linear and right affine invertible transformations, have the same combinatorial properties w.r.t. the Linear Method.

First, let us show that a change of variables has no influence on relations. Intuitively, this is clear because multiples of polynomials in the transformed system are just transformed of multiples of the original system. To prove the result formally, we show that there is a one-to-one depth-preserving correspondence between the d-relations of the two systems for any d.

We fix some value of d and perform a recursion on the relation depth.

Let $S = \{p_j\}$ and $T = \{q_j\}$ be two families of polynomials satisfying

$$\forall\, j, \quad q_j = p_j \circ A$$

where A is an invertible affine transformation of the variables. This exactly means that $\varphi : p \mapsto p \circ A$ is a one-to-one correspondence between depth 0 polynomials of $< S >$ and $< T >$.

Suppose that φ establishes a one-to-one correspondence between polynomials of d-depth k in $< S >$ and $< T >$, and that p is a polynomial of d-depth $k+1$ w.r.t. S:

$$\sum m_j e_j = p$$

with $\forall\, j, \quad \deg(m_j ej) \leq d$ and $\mathrm{depth}_d(e_j) \leq k$.
Then for T, $\mathrm{depth}_d(e_j \circ A) \leq k$, hence

$$\sum (m_j \circ A)(e_j \circ A) = p \circ A$$

is a d-relation. Since A is affine, $\forall\, j, \quad \deg((m_j \circ A)(e_j \circ A)) \leq d$, thus

$$\mathrm{depth}_d(p \circ A) \leq k+1$$

Therefore we have shown that the right affine invertible transformations do not change the relations results. Similarily, left invertible linear transformations do not change at all relations results since they do not change the vector space spanned by a polynomial family.

Left affine transformations *do* change relations in general. Indeed, such an operation can even transform a system having a unique solution into a system that does not have this property. In TRMC or other cryptosystems, left affine transformations are used, but the systems to which we apply the linear resolution method are not masked systems, but *masked systems minus a masked image value.* Thus the constant of the affine transformation is cancelled, and we only have to consider left *linear* transformations.

A.3 Behavior of the Linear Method w.r.t Projection on a Smaller Field

Here, we prove that if the Linear Method is able to solve a system $S = \{p_j\}$ expressed on a finite extension G of a field F by reducing degrees less than d, it is also able to solve S expressed over F by reducing degrees no more than d. Roughly said, this is because the projected system contains all the projections of the relations of the original system.

Let $[G : F] = \ell$, and $p_k, 1 \leq k \leq \ell$, be the projections from G to F associated to some basis $\{b_1, \ldots, b_\ell\}$ of G over F. If q is a polynomial over G with u unknowns, it defines a function $f : F^{ul} \to G$ that can be composed with any p_k.

$p_k(q)$ is the polynomial over F corresponding to the k-th coordinate of f (this polynomial is unique with some extra conditions on its degree). Each equation $e \in S$ is translated into ℓ equations $p_1(e), \ldots, p_\ell(e)$ over F. Thus the starting point of the Linear Method over F is the set $S' = \{p_k(e) | 1 \le k \le \ell, e \in S\}$.

We only have to prove that, for any d and $q \in < S >$, if the d-depth of q is n, then for $1 \le k \le \ell$, the d-depth of $p_k(q)$ is at most n.

Indeed, if that result holds, then by applying it to the case where q is linear, we get that as soon as the algorithm in G computes enough linear relations to solve the system, the algorithm running over F solves the system too.

For depth 0, the result is true because S' contains the $p_k(S)$, $1 \le k \le \ell$. Suppose it is true at depth n. Let $q \in < S >$ of depth $n+1$ output by the relation

$$\sum_j m_j e_j = q$$

with $\deg(m_j e_j) \le d$ and $\mathrm{depth}(e_j) \le n$.

$$p_k(q) = p_k\left(\sum_j m_j e_j\right) = \sum_j p_k\left(m_j e_j\right)$$

Since $\forall\, i, j,\ \mathrm{depth}(p_i(e_j)) \le n$, we only have to show that for all j, the $p_k(m_j e_j)$ can be written $\sum_i m_{ij} p_i(e_j)$, with $\forall\, i, j, \quad \deg(m_{ij} p_i(e_j)) \le d$.

This is true because for any polynomial r over G, a projection of a multiple of r $p_k(mr)$ can be expressed as a sum of multiples of projections of r

$$\sum_i m_i p_i(r)$$

with m_i polynomials over F and $\forall\, i, \quad \deg m_i \le \deg m$.

Let $\alpha_{ijk} \in F$ such that $\forall\, i, j, \quad b_i b_j = \sum_k \alpha_{ijk}\, b_k$. Then since $r = \sum_i p_i(r) b_i$ and $m = \sum_j p_j(m) b_j$,

$$mr = \sum_{i,j} p_i(r) p_j(m) b_i b_j = \sum_k \left(\sum_{i,j} \alpha_{ijk}\, p_j(m) p_i(r)\right) b_k$$

thus

$$p_k(mr) = \sum_i \left(\sum_j \alpha_{ijk}\, p_j(m)\right) p_i(r)$$