Improve the Behavior of XL Family by Reducing the Excrescent Multiply Monomials

Xijin Tang, Yong Feng *

Chengdu Institute of Computer Applications
Chinese Academy of Sciences
e-mail : tangxij@mails.gscas.ac.cn

April 20, 2005

Abstract

The XL (EXTENDED LINEARIZATION) is a equation-solving algorithm ,which is used as direct attacks against multivariate Public-Key Cryptosystems and as final stages for many algebraic cryptanalysis used today. XL produce lots excrescent multiply monomials in the solving process. The original equation multiplied by excrescent monomials are worthless for solving equation, what's more they increase the complexity of computation. In this paper we analyze the behavior of XL, and give a new version of XL called RXL to handle this problem. In RXL, part of XL's excrescent multiply monomials are removed, so, RXL is more efficient than XL. Based on our stimulation, to solve same problem, RXL need about 75% computation of XL. We also show that it is easy to extend our method to the existing XL's variants, and all algorithms of XL family are improved.

Key words: XL,RXL, multivariate cryptography, system of quadratic equations, algebraic attack.

1 Introduction

Public Key Cryptography depends on the intractability of "hard problems". One of such problems is the problem MQ, which is to solve a system of multivariate quadratic equations over a small finite field. Several public

*The work is supported in part by NKBRPC-2004CB318003.

key cryptosystems based on MQ have been proposed, for example the HFE family [13].

At Crypto'99, Shamir and Kipnis present a surprising method called relinearization [7] for solving overdefined systems of multivariate quadratic equations. They point out that, if such a system of equations is overdefined (much more equations than needed), it can be solved much faster than expected. Subsequently, at Eurocrypt 2000 [8], Courtois, Klimov, Patarin and Shamir, present a new algorithm called XL, (and also FXL) that can be seen as an improved version of relinearization.

From [8] , several XL's variants have been proposed, such as XL2, XL', XLF, XFL. Those variants improved XL's efficiency over GF(2) or GF(2^k), but no one have paid attention to the excrescent monomials in XL, which make XL inefficient. The original equation multiplied by excrescent monomials are worthless for solving the equation system, because they will reduce to zero after the Gaussian elimination. These worthless equations are more that 50 percents of the equations XL generate on average.

In this paper, we analyze the behavior of XL and give a new version of XL called RXL to handle excrescent monomials in XL. We do following:

- Analyzing the behavior of XL and show the two ways of excrescent monomials from;
- Giving RXL algorithm to remove part of the excrescent monomials in XL;
- Comparing the behaviors of XL and RXL;
- Discussing the relationship between RXL and other variants

The rest of this paper is organized as follows: In section 2, we give the natation and convention used in this paper. In section 3, we recall the base principle of XL and its variants. In section 4, we give the RXL algorithm. In section 5, we give a toy example of RXL. in section 6, we compare the behavior of XL and RXL. In section 7, we discuss the relationship between RXL and other variants. In section 8, we conclude this paper.

2 Notations and Conventions

In this paper we consider the problem of solving a system of m multivariate quadratic equations with n variables over a finite field $GF(q)$. The input variables are denoted by x_i and belong to $GF(q)$ with $q = 2^k$. The equations

are denoted by l_i and are quadratic, which may include linear and constant terms. The system to solve will be:

$$A = \begin{cases} l_1(x_1, x_2, ...x_n) = 0 \\ \quad . \\ \quad . \\ \quad . \\ l_m(x_1, x_2, ...x_n) = 0 \end{cases}$$

Given m, n, q, we call MQ the problem of finding one (not necessarily all) solution to such a system chosen at random.

Because the right hand of all our equations is always 0, it is very useful to identify a multivariate polynomial and an equation that says it is equal to 0. Thus the equation $l_i(x_1, ..., x_n) = 0$ can be simply called equation l_i, and the equation $x_1 l_2(x_1, ..., x_n) = 0$ can be called simply $x_1 l_2$.

The set of monomials $x^k = (x_1^{b_1} x_2^{b_2} ... x_n^{b_n})$, stands for all monomials whose degree is $k(\sum_i b_i = k)$. The equations of the form $x^k l_j$, are of type $x^k l$, and $x^k l$ is denoted by the set of all these equations. For example the initial equations A are of type l. We observe that each solution x that satisfies all the equations l_i, also does satisfy all the equations of type $x^k l$, for any $k \geq 0$. By extension we define $x^0 = 1$, the constant monomial.

Let $D \in IN$. We consider all the polynomials $x^b l_i$ of total degree $\leq D$. Let I_D be the set of equations they span. I_D is the linear space generated by all the $x^k l, 0 \leq k \leq D-2$. We have $I_D \subset I$, where I is the ideal spanned by the l_i. We denoted by τ the set of monomials, including the constant monomials, which appears in all the equations of $I_D = \bigcup_{i=0}^{D} x^i$.

3 The XL Family of Algorithms

3.1 The Basic Principle of XL

Let D be the parameter of XL algorithm. The XL algorithm is as follows [8, 10]:

The XL algorithm. Execute the following steps:

1. Multiply: Generate all the products $x^k l_j \in I_D$ with $k \leq D-2$.

2. Linearize: Consider each monomial in the x^i of degree $\leq D$ as a new variable and perform Gaussian elimination on the equations obtained in step 1. The ordering on the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.

3. Find x_1: Assume that step 2 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite fields (e.g., with Berlekamp's algorithm). There may be several roots.

4. Recover the other variables: For each root $x1$ substitute it to the expanded equations. From the Gaussian reduction done in step 3, find the values of all the other monomials, in particular for all the other variables x_i.

Note: we denote by $Free$ the exact dimension of I^D.

3.2 The Family of XL Variants

There are serval XL's variants, we will give the base idea of every variant below. First, we will look at the FXL.

3.2.1 FXL: Guessing as Aid to Equation-Solving

The"F"in FXL stands for"fix" [8]. The attacker assigns random values to f variables, in effect guessing at them, hoping to decrease the degree D needed for XL. After guessing, we run XL and test at the end if any solution found is valid. We note the fixing concept was applied almost verbatim to the F4 and F5. I.e., we may also guess at a few variables before applying a Gröbner Bases method.

3.2.2 XL2: Gaining Extra Equations via the T' method

It was first proposed [9] as an addendum to XL over GF(2), to add useful equations. Let T' count the monomials that when multiplied by a given variable will still be in $\tau = \tau^{(D)}$. I.e. $T' = |T'_i|$, where $T'_i = x^b : x_i x^b \in \tau$ for each i. Suppose $Free$ is not as large as $T - D$, but $C \equiv T'$+Free$-T > 0$ (i.e. we have enough equations to eliminate all monomials not in T'_i), then:

1. Eliminate from the system $R = I_D$ the monomials not in T'_1 first. We are then left with relations R_1 that gives each monomial in T
T'_1 as a linear combination of those monomials in T'_1 , plus C equations R'_1 with only monomials in T'_1.

2. Repeat for T'_2 to get the equations R_2 and R'_2 (we should also have $|R'_2| = C$).

3. For each $l \in R'_1$, monomial in the equation $x_1 l = 0$ are either in T'_2 or can be reduced (using R_2) into T'_2 . Ditto each $x_2 l (l \in R'_2)$ and we get $2C$ new equations.

3.2.3 XL': Searching as the Final Step

XL' [10] is XL except that we come down to a system in r variables and at least r equations, then end by brute-force search. The new terminations conditions are: instead of requiring $T - Free \leq D$, we only require $T - Free \leq \binom{r+D}{D} - r$.

3.2.4 XLF: Using the Field Relations

[11] proposes to use the field relations $x^q = x$ to improve XL's behavior when $q = 2k$.

—Consider (x_i^2), (x_i^4),..., $(x_i^{2^k-1})$ independent variables in K in addition to x_i.

—Equations are generated as in every other XL method, then each generated equation is raised to the second, fourth,. . . powers easily (since this is a linear operation) as equations in (x_i^2), (x_i^4),..., $(x_i^{2^k-1})$, for k times as many variables and equations.

—That all equivalent monomials are ipso facto equal become new equations, which may let the algorithm execute with a lower D.

3.2.5 XFL: Guessing with a Twist

Another variant proposed with the name improved FXL and later XFL [11, 12]:

1. Choose f ("to fix") variables. Multiply the equations by all monomials up to degree $D - 2$ in the other $n - f$ variables only.

2. Order the monomials so that all monomials of exactly degree D with no "to-fix" factor comes first. Eliminate all such monomials from the top-degree block.

3. Substitute actual values for "to-fix" variables, then collate the terms and try to continue XL, re-ordering the monomials if needed, until we find at least one solution.

4 RXL – A New Version of XL to Reduce Excrescent Multiply Monomials

4.1 The Source of Excrescent Multiply Monomials in XL

Table 1 is from [11]. From the table it is found that XL produce a lot of excrescent multiply monomials. For example, when $n = 3, m = 3, D =$

$8, R = 252, Free = 165$, $87(R - Free)$ equations are worthless. When $n = 4, m = 4, D = 16, R = 12240, Free = 4829$, $7411(R - Free)$ equations are worthless. These worthless equations are more than 50 percents of the equations XL generate on average, which make the matrix size quite large, and it is costly to eliminate these equations to zero. If we could find a method to avoid generating those worthless equations, the efficiency of XL will be improved.

Table 1: XL over GF(2^7) for m = n

n	2	2	2	3	3	3	3	3	4	4	4	4	4
m	2	2	2	3	3	3	3	3	4	4	4	4	4
D	2	3	4	2	4	6	7	8	4	5	10	15	16
R	2	6	12	3	30	105	168	252	60	140	1980	2860	12240
T	6	10	15	10	35	84	120	165	70	126	1001	1365	4845
$Free$	2	6	11	3	27	76	112	157	54	110	985	1349	4829
$\frac{Free}{T-D}$	0.5	0.86	1.00	0.38	0.87	0.97	0.99	1.00	0.82	0.91	0.99	0.99	1.00
Success			OK					OK					OK

Legend:
n number of variables.
m number of equations.
D we generate equations of total degree $\leq D$ in the x.
R number of equations generated (independent or not).
T number of monomials of degree $\leq D$.
$Free$ number of linearly independent equations among the R equations.

these excrescent multiply monomials are from two ways:

1.only one or two the original equation are involved in the equation : $x^u \times (l_i \times l_j - l_j \times l_i) = 0$,for some u, i, j;(i could be equal to j)

2.more than two original equations are involved in the equation: $x^{u_1} \times (l_i \times l_j - l_j \times l_i) + x^{u_2} \times (l_j \times l_k - l_k \times l_j) + ... = 0$,for some $u_1, u_2, ...; i, j, k....$

From [5] it is known that, to remove the excrescent multiply monomials from **way 2** , we need to compute the Gröbner Bases of $(l_i, l_j, l_k, ...)$, which will be costly sometimes, what's worse is that XL will turn to be a pure Gröbner Bases algorithm. But when XL turn to be a pure Gröbner Bases algorithm, it will lose its own characteristic. Therefore we don't intend to do so.

4.2 RXL algorithm

To remove the excrescent multiply monomials from **way 1**, we will introduce a new version of XL called RXL, in which "R" means reduce the excrescent multiply monomials. The basic idea is from the simple fact that $l_1 \times l_2 = l_2 \times$

l_1. For example $l_1 = x_1+x_3+x_2x_3, l_2 = x_2+x_3+x_1x_3$, expand the left part of this equation's two sides, we get $(x_1+x_3+x_2x_3)\times l_2 = (x_2+x_3+x_1x_3)\times l_1$. Suppose matrix M is constructed from original equation l_1 multiplied by x_2, x_3, x_1x_3 and original equation l_2 multiplied by x_1, x_3, x_2x_3. There will be one zero row after Gaussian elimination. By removing the equation $x1\times l2$ or any other equation from these 6 equations, there will be no zero row after Gaussian elimination.

Following the basic idea, we first remove the excrescent monomials in x^2. Then we generate $x^{i=3..D}$ by x^{i-2} and x^2, to reduce excrescent monomials.

RXL algorithm:
step 1. initialize step:
$x^0 = 1$;
x^1 ={variable occurred in the original MQ system};
step 2. Generate x^2:
x^2={all the monomials whose degree is 2};
for i from 1 to m
In l_i, choose one monomial a of degree 2, which is different from the monomials chosen before;
$x^2 = x^2 - a$;
step 3. Generate $x^{i=3..D}$ by x^{i-2}:
$x^i = \{\}$;
for each monomial b in x^{i-2}
for each monomial c in x^2
$x^i = x^i + b \times c$;
step 4. Multiply: Generate all the products $x^k l_j \in I_D$ with $k \leq D-2$.
step 5. Linearize: Consider each monomial in the x^i of degree $\leq D$ as a new variable and perform Gaussian elimination on the equations obtained in step 4. The ordering on the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.
step 6. Find x_1: Assume that step 5 yields at least one univariate equation in the powers of x_1. Solve this equation over the finite fields (e.g., with Berlekamp's algorithm). There may be several roots.
step 7. Recover the other variables: For each root x_1 substitute it to the expanded equations and, directly from the Gaussian reduction done in step 3, find the values of all the other monomials, in particular for all the other variables x_i

In order to simplify algorithm, we remove more equation in RXL than that should be removed follow the basic idea. As in the equation RXL

generated there still exist dependent equations, RXL will always succeed when D is large(based on our simulation, usually ≥ 5). When we solve a MQ problem, D that XL need to terminate is usually ≥ 5, so RXL will succeed in most cases.

5 A Toy Example of RXL

Consider the following system A , $n = 3, m = 3, q = 5$, and $D = 8$.

$$A = \begin{cases} l_1 : x_1 + x_3 + x_2x_3 + x_3^2 - 4 = 0 \\ l_2 : x_2 + x_3 + x_1x_3 + x_2^2 - 4 = 0 \\ l_3 : x_3 + x_1 + x_1x_2 + x_1^2 - 4 = 0 \end{cases}$$

In step 1, we get $x^0 = 1, x^1 = \{x_1, x_2, x_3\}$.

In step 2, first we get $x2 = \{x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3\}$, then we choose x_2x_3 from l_1, choose x_1x_3 from l_2, choose x_1x_2 from l_3, remove the monomials we have chosen from x^2, we get $x^2 = \{x_1^2, x_2^2, x_3^2\}$.

In step 3, we get

$x^3 = \{x_1^2x_2, x_1^2x_3, x_1x_2^2, x_1x_3^2, x_1^3, x_2^3, x_3^3, x_2^2x_3, x_2x_3^2\}$,

$x^4 = \{x_1^2x_2^2, x_1^2x_3^2, x_1^4, x_2^4, x_3^4, x_2^2x_3^2\}$,

$x^5 = \{x_1^4x_2, x1^4x_3, x_1^3x_2^2, x_1^3x_3^2, x_1^2x_2^3, x_1^2x_3^3, x_1x_2^4, x_1x_3^4, x_1^5, x_2^5, x_3^5,$
$x_1^2x_2^2x_3, x_1^2x_2x_3^2, x_1x_2^2x_3^2, x_2^4x_3, x_2^3x_3^2, x_2^2x_3^3, x_2x_3^4\}$,

The following steps are same as XL, in step 4 we multiply every original equation by the monomials generated in step1, step2 and step3, then we get 150 equations. In step 5 we get a 120×120 matrix, after Gaussian elimination, it has 142 rows are not equal to zero. Rewiring the last nonzero row in to equation form, we get $x_3^7 + x_3^6 + 2x_3^5 + x_3^4 + 2x_3^3 + 3x_3^2 + 3x_3 + 2$, it is only in one variable. Finally we solve the equation got in step 5, and recover other variants.

6 Comparing XL and RXL

From the toy example, it is obvious that RXL generate smaller matrix than XL. If we use XL to solve the toy example,we will get a 168×120 matrix, as same as in the Table 1. Suppose a matrix has m rows and n columns, we usually believe Gaussian elimination's complexity is $O(m \times n \times n)$. So we estimate RXL only do about $\frac{120 \times 120 \times 120}{168 \times 120 \times 120} \approx 71\%$ computation of XL's in that example.

Next two tables are our simulations on RXL.

Table 2: RXL over GF(5) for m = n

n	2	2	2	3	3	3	3
m	2	2	2	3	3	3	3
D	2	3	4	2	4	6	7
R	2	6	8	3	21	66	120
T	6	10	13	10	35	84	120
$Free$	2	6	8	3	21	66	112
$\frac{Free}{T-D}$	0.5	0.86	0.88	0.38	0.68	0.84	0.99
Success			OK				OK
%	1	1	0.5	1	0.7	0.63	0.71

Legend:
% $\frac{R \times T \times T}{corresponding\ R'\ in\ XL \times corresponding\ T'\ in\ XL \times T'}$

Table 3: comparing RXL and XL over GF(5) for m = n =4

D	4	5	6	7	8	9	10	12	14	16
R	60	140	280	504	840	1320	1980	2860	7280	12240
T	70	126	210	330	495	715	1001	1820	3060	4845
R'	44	120	196	408	584	1040	1380	2804	5124	8656
%	0.73	0.85	0.70	0.80	0.69	0.79	0.69	0.70	0.70	0.71

Legend: R' number of equations generated by RXL (independent or not).

Because we have not implemented RXL over $GF(2^k)$, we run RXL over $GF(5)$ instead of $GF(2^k)$. RXL and XL have the similar behavior over $GF(5)$ and $GF(2^k)$, so it will not influence our conclusion.

In our simulation, the RXL's behavior is consistent with our anticipation. When XL terminate on a special D, RXL also terminate on that value, both XL and RXL give us a equation only in one variable. RXL could not decrease the D needed for XL, but RXL generate smaller matrix than XL. Based on our simulation, RXL's matrix size is usually 75% of XL's. When solving MQ problem, if we use RXL instead of XL, we could save about 25% computation on average.

7 RXL and The Family of XL Variants

RXL distinguish from other variants, for it improve the core process of XL. So it is easily to apply "R" to the family of XL variants. Take FXL for example, after fix f serval variables, and assigns random values to f variables, we could run RXL instead of XL, the result will be the same as XL, but with smaller matrix and less computation.

By applying "R" to the family of XL variants,we could get RFXL, RXL', RXLF, and RXFL. All the family of XL variants' efficiency could be improved.

8 Conclusion and Perspectives

In this paper, we present a new version of XL called "RXL", which make XL more efficient by choosing the multiply monomials more precisely. RXL could save about 25% computation time.

RXL have not solved the XL's excrescent multiply monomials problem completely, there still exist many excrescent multiply monomials in RXL. XL is a simple algorithm, from [3] it is known that XL is slower than Gröbner algorithm F4,F5, even slower than the classical Buchberger algorithm on HFE, We believe XL has the potential to be a good and quick algorithm. To improve XL's efficiency, the choice of multiply monomials is a key factor. In the future we will concentrate on the choice of XL's multiply monomials.

References

[1] Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations, Eurocrypt2000, LNCS 1807, Springer, pp. 392-407.

[2] Adi Shamir, Aviad Kipnis: Cryptanalysis of the HFE Public Key Cryptosystem; In Advances in Cryptology, Proceedings of Crypto'99, Springer, LNCS.

[3] G. Ars and J.-C. Faugre, Comparison Between XL and Gröbner Basis Algorithms . ASIACRYPT'04 ,LNCS 3329,pp.338-353,2004.

[4] B. Barkee et al, Why You Cannot Even Hope to Use Gröbner Bases in Public-Key Cryptography, J. Symbolic Computations, 18 (1994), pp. 497-501.

[5] J.-C. Faugeere, A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5), Proceedings of ISSAC 2002, pp. 75-83, ACM Press 2002

[6] J.-C. Faugere, A New Efficient Algorithm for Computing Gröbner Bases (F4), Journal of Pure and Applied Algebra, 139 (1999), pp. 61.88.

[7] A. Kipnis and A. Shamir, Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization, Crypto'99, lncs v. 1666, pp. 19-30.

[8] N. Courtois,A. Klimov, J. Patarin, andA. Shamir, Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations, Eurocrypt 2000, lncs v. 1807, pp. 392-407

[9] N. Courtois and J. Pieprzyk,Cryptanalysis of Block Ciphers with Overdefined Systems of Equations, ASIACRYPT 2002, LNCS 2501, pp. 267.287.

[10] . Courtois and J. Patarin, About the XL Algorithm over GF(2), CT-RSA 2003, LNCS 2612, pp. 141.157

[11] N. Courtois, Algebraic Attacks over GF(2^k), Cryptanalysis of HFE Challenge 2 and SFLASHv2, PKC'04, LNCS 2947, pp. 201-217.

[12] B.-Y. Yang, J.-M. Chen, and N. Courtois, On Asymptotic Security Estimates in XL and Gröbner Bases-Related Algebraic Cryptanalysis, ICICS '04, LNCS 3269, pp. 401-413. Formerly titled Exact and Asymptotic Behavior of XL-Related Methods.

[13] Jacques Patarin: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials(IP): two new families of Asymm. Algorithms, Eurocrypt'96, Springer, pp. 33-48.