MutantXL: Solving Multivariate Polynomial Equations for Cryptanalysis

Johannes Buchmann[1], Jintai Ding[2], Mohamed Saied Emam Mohamed[1], and Wael Said Abd Elmageed Mohamed[1]

[1] TU Darmstadt, FB Informatik
Hochschulstrasse 10, 64289 Darmstadt, Germany
{buchmann,mohamed,wael}@cdc.informatik.tu-darmstadt.de
[2] Department of Mathematical Sciences, University of Cincinnati
Cincinnati OH 45220, USA
jintai.ding@uc.edu

Abstract. MutantXL is an algorithm for solving systems of polynomial equations that was proposed at SCC 2008 and improved in PQC 2008. This article gives an overview over the MutantXL algorithm. It also presents experimental results comparing the behavior of the MutantXL algorithm to the F_4 algorithm on HFE and randomly generated multivariate systems. In both cases MutantXL is faster and uses less memory than the Magma's implementation of F_4.

1 Introduction

Solving systems of multivariate quadratic equations is an important problem in cryptology. The problem of solving such systems over finite fields is called the Multivariate Quadratic (MQ) problem. In the last two decades, several cryptosystems based on the MQ problem have been proposed as in [1–4]. The general MQ problem is NP-complete! However for some cryptographic schemes the problem of solving the corresponding MQ system has been demonstrated to be easier, allowing these schemes to be broken. Therefore it is very important to study the MQ problem.

XL was introduced in [5] as an efficient algorithm for solving polynomial equations in case only a single solution exists. The MutantXL algorithm was proposed as a variant of XL. It is based on the mutant strategy [6, 7]. The MXL_2 algorithm [8] is an improvement to MutantXL. It uses the partial enlargement technique [8] and a minimum number of mutants.

In this article we give an experimental comparison between our implementation of the MutantXL algorithm and Magma's implementation of the F_4 algorithm [9] on some HFE cryptosystems of univariate degree 288 and some randomly generated instances of the MQ problem. We show that for the HFE systems MutantXL can solve systems that have number of variables up to 48 while F_4 can not solve system with more than 39 variables under the same memory restriction. Moreover, we show that MutantXL solves all systems faster

Dagstuhl Seminar Proceedings 09031
Symmetric Cryptography
http://drops.dagstuhl.de/opus/volltexte/2009/1945

and uses less memory than F_4. Another indicator for the fact that a variant of MutantXL outperforms F_4 can be found in [10].

This paper is organized as follows. In Section 2 we review the XL and mutant strategies. In Section 3 we describe the MutantXL algorithm. In Section 4 we give our experimental results on random and HFE systems and finally Section 5 contains the conclusion and the future work.

2 XL algorithm and Mutant Strategy

In this section we present a brief overview of the XL algorithm [5] and the mutant strategy [7, 6].

Throughout the paper we let F be a finite field and we let q be its cardinality. We consider the ring

$$R = F[x_1, \ldots, x_n]/(x_1^q - x_1, ..., x_n^q - x_n)$$

of functions over R in the n variables $x_1, \ldots, x_n$. Here $x_i^q - x_i = 0$, $1 \leq i \leq n$ are the so-called field equations. In R, each element is uniquely expressed as a polynomial where the degree in each x_i is less than q. The degree of this polynomial is called the degree of the corresponding function. For the sake of convenience, we will identify functions in R with their representing polynomials.

Let P be a finite set of polynomials in $F[\underline{x}]$, $\underline{x} = (x_1 \ldots, x_n)$. Given a degree bound D, the XL strategy is simply based on extending the set of polynomials P by multiplying each polynomial in P by all the possible monomials such that the resulting polynomials have degree less than or equal to D. Then, by using linear algebra, XL computes $\widetilde{P}$, a row echelon form of the extended set P. XL uses univariate polynomials in $\widetilde{P}$ to solve $P(\underline{x}) = 0$ at least partially. If the system can not be solved, D is increased.

In [7, 6], it was pointed out that during the linear algebra step, certain polynomials of degrees lower than expected appear. These polynomials are called mutants. The Mutant strategy is to add mutants to P which allows solving system with a small D. The precise definition of mutants is as follows.

Let I be the ideal generated by the finite set of polynomials P. An element f in I can be written as

$$f = \sum_{p \in P} f_p p \tag{1}$$

where $f_p \in F[\underline{x}]$. The maximum degree of $f_p p$, $p \in P$, is the *level* of this representation. The level of f is the minimum level of all of its representations. The polynomial f is called *mutant* with respect to P if $\deg(p)$ is less than its level.

3 Description of the MutantXL Algorithm

We briefly describe the MutantXL algorithm including our improvements that enable us to beat the F_4 implementation of Magma. Those improvements consist

in further reducing the number of mutants used to solve the system and the partial enlargement technique that solves the too few mutants problem.

We let $F = \mathbb{F}_2$. The algorithm can be easily generalized. Let $X := \{x_1, \ldots, x_n\}$ be a set of variables, upon which we impose the following order: $x_1 > x_2 > \ldots > x_n$. Let

$$R = \mathbb{F}_2[x_1, \ldots, x_n]/(x_1^2 - x_1, ..., x_n^2 - x_n)$$

be the ring of polynomial functions over $\mathbb{F}_2$ in X with the monomials of R ordered by the graded lexicographical order $<_{glex}$. Let $P = (p_1, \ldots, p_m) \in R^m$ be an m-tuple of polynomials in R. In the algorithm a degree bound D will be used. It the maximum degree of the polynomials contained in P. P will be changed in the algorithm. The algorithm performs the following steps:

- *Initialize*: Set $P = \{p_1, \ldots, p_m\}$, $D = \text{Max}\{\deg(p) : p \in P\}$, the elimination degree $ED = \min\{\deg(p) : p \in P\}$, the set of mutants $M = \emptyset$.
  compute the row echelon form $\widetilde{P}$ of P. Set $P = \widetilde{P}$.
  Here polynomials are identified with their coefficient vectors as explained in [9].

- *Solve*: If there are univariate polynomials in P, then determine the values of the corresponding variables and substitute in P. If this solves the system return the solution and terminate. Otherwise, set $D = \text{Max}\{\deg(p) : p \in P\}$, set the elimination degree $ED = \min\{\deg(p) : p \in P\}$, and go back to *Echelonize*.
- *ExtractMutants*: Add all new elements of degree less than D in P to M.
- *MultiplyMutants*: If $M \neq \emptyset$, then select the necessary number of mutants that have degree k = $\min\{\deg(p)$: $p \in M\}$ and multiply them by all terms of degree one, remove the selected mutants from M, add the new polynomials obtained from the multiplication to P, set ED to $k + 1$ and go back to *Echelonize*.
  The necessary number of mutants are numerically computed as in [8].
- *Extend*: Extend P by adding all polynomials that are obtained by multiplying a subset of the degree D elements in P by all monomials of degree one. Increment D by one, set $ED = D$ and go back to *Echelonize*.
  The strategy for selecting the subset is taken from [8].

4 Experimental Results

We compare the efficiency of MutantXL to the efficiency of F_4 by solving some random systems generated by Courtois [11] as well as some HFE systems generated by the code of John Baena. We ran all the experiments in a Sun X4440 server, with four "Quad-Core AMD Opteron^{TM} Processor 8356" CPUs and 128 GB of main memory. Each CPU is running at 2.3 GHz. We used only one out of the 16 cores.

Tables 1 and 2 show the results of dense random systems and the results of HFE systems of univariate degree 288, respectively. In both tables we denote the

	MutantXL				F_4			
n	D	max. matrix	Memory	Time	D	max. matrix	Memory	Time
25	6	71159×76414	698	704	6	248495×108746	5128	1341
26	6	96937×102246	1207	1429	6	298592×148804	8431	3325
27	6	134518×140344	2315	2853	6	354189×197902	13312	6431
28	6	201636×197051	4836	7982	6	420773×261160	20433	13810
29	6	279288×281192	9375	18796	6	499222×340254	30044	25631
30	6	347263×351537	15062	42501	6	1283869×374081	72258	92033
31	6	427164×436598	23078	99597	6	868614×489702	108738	162118

Table 1. Performance of MutantXL versus F_4 for dense random system

number of variables and equations by n and the maximum degree used in the algorithm by D. The tables also show the maximum matrix size, the memory used in Megabytes, and the execution time in seconds. It is evident from Tables 1 and 2 that MutantXL solves the random generated systems and HFE systems faster and consumes less memory than F_4.

	MutantXL				F_4			
n	D	max. matrix	Memory	Time	D	max. matrix	Memory	Time
30	5	124767×130211	1997	3260	5	149532×136004	7105	3806
35	5	172524×296872	6349	10198	5	199839×332645	39710	11282
36	5	191939×344968	8090	14555	5	219438×382252	50846	15220
37	5	210416×399151	10137	20375	5	246121×44360	65205	22291
38	5	233219×459985	12472	29123	5	274970×512296	82103	32530
39	5	254218×528068	16365	36833	5	305745×588617	102258	36965
40	5	280911×604033	22732	63460≃17.6 hours	no solution obtained			
45	5	432554×1126819	58982	299355≃3.46 days	no solution obtained			
47	5	508064×1417468	86548	371088≃4.3 days	no solution obtained			
48	5	538602×1583807	102919	689235≃7.9 days	no solution obtained			

Table 2. Performance of MutantXL versus F_4 for HFE(288,n) systems

Table 2 shows that all the HFE systems of univariate degree 288 up to 48 variables are solved by using MutantXL, whereas F_4 could only solve HFE systems up to 39 variables with the same memory resources.

In Table 3 we compare the performance of the MutantXL algorithm with the F_4 algorithm for solving the random system $n = 31$. For MutantXL, we give the elimination degree (D), the matrix size, the rank of the matrix (Rank), the number of mutants found (NM), the number of used mutants (UM), and the lowest degree of mutants found (MD). For F_4, we give the degree D in each step, the matrix size, and the step memory in MB.

Table 3 shows that by using the mutant strategy, MutantXL can easily solve the 31 variables random system with a small matrix size compared to F_4. MutantXL starts to generate mutants at step 5. In this step 13950 mutants

	MutantXL						F_4		
Step	D	Matrix Size	Rank	NM	UM	MD	D	Matrix Size	Memory
1	2	31×497	31	0	0	-	2	31×497	14.9
2	3	990×4992	990	0	0	-	3	999×4992	14.9
3	4	15375×36457	14911	0	0	-	4	14838×35123	254.9
4	5	140114×206368	138880	0	0	-	5	122365×172445	5841
5	6	415654×436598	382912	13950	1702	5	6	692242×557944	59836
6	6	**427164×436598**	427164	12260	330	4	6	**868614×489702**	108738
7	5	205662×206368	205662	3294	26	3	2	777×497	108738
8	4	35657×36457	35657	434, 2	0, 2	2, 1	3	5412×4195	108738
9	2	517×497	496	29	0	1	4	27173×22950	108738
10							5	105805×77982	108738
11							6	100519×77029	108738

Table 3. MutantXL: Results for the system Random-31

of degree 5 are generated, out of which only 1702 are multiplied. Due the degree of the generated mutants, the elimination degree remains the same in the next step, i.e. , $D = 6$. Starting from step 7, D starts to decrease. In step 8, the system generates 434 quadratic mutants and 2 linear mutants. By using only the two linear mutants, MutantXL generates additional 29 linear mutants in the next step, which in turn leads to solving the system.

Another important comparison is to study the performance of MutantXL and F_4 in solving HFE systems of different univariate degrees (d). Table 4 shows the results of solving HFE systems with 25 variables and with different d. It is clear that MutantXL is faster and uses less memory since it constructs smaller matrices than F_4. It is important to point out that for F_4, we did not set the HFE parameter to true since we examine the general case of HFE systems whose univariate degrees can go above 128.

	MutantXL				F_4			
d	D	max. matrix	Memory	Time	D	max. matrix	Memory	Time
$d < 17$	3	2620×2626	1.5	0.16	3	3669×2500	15	0.35
$16 < d < 64$	4	7100×13252	12	1.56	4	8745×8101	68	2.25
$63 < d < 96$	4	9200×13252	15	2.34	4	10837×13913	137	6.9
$95 < d < 128$	4	10615×13252	17.6	3.28	4	12497×13648	153.5	15.1
$d = 128$	4	15050×15276	28.7	5.9	4	14344×15245	227	22.7
$128 < d < 257$	5	31400×41610	217	70.6	5	52333×52665	750	148
$256 < d < 513$	5	40458×41610	231	110	5	53388×52835	1135	273
$d > 512$	6	72456×76414	735	811	6	257316×108618	5716	1420

Table 4. Performance of MutantXL versus F_4 for HFE(d,25) systems

For F_4, we used Magma version V2.13-10 implementation of Faugère's F_4 algorithm which is considered the best available tool for solving multivariate

systems. For MutantXL, we used our C++ implementation. For the *Echelonize* step, we used an adapted version of M4RI [12], the dense matrix linear algebra over $\mathbb{F}_2$ library. Our adaptation was in changing the strategy of selecting the pivot during Gaussian elimination to keep the old elements in the system intact.

5 Conclusion and Future Work

The MutantXL algorithm is a new and very efficient alternative to solve multivariate polynomial equations on the function ring over $\mathbb{F}_2$. The experiments showed that both for classical cryptographic challenges and random systems, MutantXL algorithm performs substantially better than the F_4 algorithm implemented in Magma, currently the best publicly available implementation of F_4.

As a next step we will use the Wiedemann algorithm for solving sparse linear systems to generate mutants and do the linear algebra step in MutantXL.

This article further demonstrates the great potential of the mutant strategy and much more is still needed to be done to realize its full potential.

6 Acknowledgments

We would like to thank Ralf-Philipp Weinmann for several helpful discussions and comments. Also, we would like to thank John Baena for supporting us with his code of generating HFE systems.

References

1. Matsumoto, T., Imai, H.: Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In: Workshop on the Theory and Application of of Cryptographic Techniques Advances in Cryptology- EUROCRYPT. Volume 330 of Lecture Notes in Computer Science., Davos, Switzerland, Springer (1988) 419–453
2. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms. In: Proceeding of International Conference on the Theory and Application of Cryptographic Techniques Advances in Cryptology- Eurocrypt. Volume 1070 of Lecture Notes in Computer Science., Saragossa, Spain, Springer (1996) 33–48
3. Patarin, J., Goubin, L., Courtois, N.: C^*_{-+} and HM : Variations Around Two Schemes of T. Matsumoto and H. Imai. In: Proceeding of International Conference on the Theory and Application of Cryptology and Information Security Advances in Cryptology ASIACRYPT. Volume 1514 of Lecture Notes in Computer Science., Beijing, China, Springer (1998) 35 – 50
4. Moh, T.: A Public Key System With Signature And Master Key Functions. In: Communications in Algebra. (1999) 2207 – 2222
5. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Proceedings of International Conference on the Theory and Application of Cryptographic

Techniques(EUROCRYPT). Volume 1807 of Lecture Notes in Computer Science., Bruges, Belgium, Springer (2000) 392–407
6. Ding, J., Buchmann, J., Mohamed, M.S.E., Moahmed, W.S.A., Weinmann, R.P.: MutantXL. In: Proceedings of the 1st international conference on Symbolic Computation and Cryptography (SCC08), Beijing, China, LMIB (2008) 16 – 22
7. Ding, J.: Mutants and its impact on polynomial solving strategies and algorithms. (Privately distributed research note, University of Cincinnati and Technical University of Darmstadt, 2006)
8. Mohamed, M.S.E., Mohamed, W.S.A.E., Ding, J., Buchmann, J.: MXL2: Solving Polynomial Equations over GF(2) using an Improved Mutant Strategy. In: Proceedings of The Second international Workshop on Post-Quantum Cryptography, (PQCrypto08). Lecture Notes in Computer Science, Cincinnati, USA, Springer-Verlag, Berlin (2008) 203–215
9. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Pure and Applied Algebra **139** (1999) 61–88
10. Mohamed, M.S.E., Ding, J., Buchmann, J.: Algebraic Cryptanalysis of MQQ Public Key Cryptosystem by MutantXL. Technical Report 2008/451, Cryptology ePrint Archive (2008)
11. Courtois, N.T.: Experimental Algebraic Cryptanalysis of Block Ciphers. http://www.cryptosystem.net/aes/toyciphers.html (2007)
12. Albrecht, M., Bard, G.: M4RI – Linear Algebra over GF(2). http://m4ri.sagemath.org/index.html (2008)
13. chung Wang, L., hwang Chang, F.: Tractable rational map cryptosystem (2005)