An Improvement of Linearization-based Algebraic Attacks

Satrajit Ghosh
Abhijit Das

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, India
satrajit,abhij@cse.iitkgp.ernet.in

Abstract. In an algebraic attack on a cipher, one expresses the encryption function as a system (usually overdefined) of multivariate polynomial equations in the bits of the plaintext, the ciphertext and the key, and subsequently solves the system for the unknown key bits from the knowledge of one or more plaintext/ciphertext pairs. The standard eXtended Linearization algorithm (XL) expands the initial system of equations by monomial multiplications. The expanded system is treated as a linear system in the monomials. For most block ciphers (like the Advanced Encryption Standard (AES)), the size of the linearized system turns out to be very large, and consequently, the complexity to solve the system often exceeds the complexity of brute-force search. In this paper, we propose a heuristic strategy XL_SGE to reduce the number of linearized equations. This reduction is achieved by applying structured Gaussian elimination before each stage of monomial multiplication. Experimentation on small random systems indicates that XL_SGE has the potential to improve the performance of the XL algorithm in terms of the size of the final solvable system. This performance gain is exhibited by our heuristic also in the case of a toy version of AES.

Keywords: Block cipher, AES, multivariate polynomial equation, algebraic attack, linearization, XL, sparse linear system, structured Gaussian elimination

1 Introduction

The security of many cryptosystems is based on the difficulty of solving large systems of nonlinear multivariate polynomial equations [1]. The main idea of algebraic cryptanalysis is to express the encryption transform of a cipher as an overdefined system of multivariate polynomial equations in the bits of the plaintext, the ciphertext and the key. Algebraic cryptanalysis deals with different ways of solving the multivariate system from known plaintext/ciphertext pairs. Techniques based upon Gröbner-basis computation (like Faugère's F_4 and F_5 algorithms [2, 3]) usually take exponential (or more) time in the size of the system, and so are practically infeasible. Kipnis and Shamir [4] introduce an

alternative elimination technique called *relinearization* which is expected to run in subexponential time. Several variants of this relinearization technique have been proposed in the literature (like XL [5], XSL [6] and MutantXL [7]). A third elimination technique proposed by Bard et al. [8] makes use of SAT solvers.

There are practical examples of algebraic attacks on stream ciphers, block ciphers and public-key cryptosystems. In 1999, Kipnis and Shamir cryptanalyze the HFE public-key cryptosystem by their relinearization technique [4]. In 2000, Courtois et al. [5] propose the XL algorithm (eXtended Linearization). Its modification named XSL (eXtended Sparse Linearization) is proposed by Courtois and Pieprzyk [6] in 2002. In 2007, Courtois and Bard [9] cryptanalyze six rounds of DES from only one known plaintext/ciphertext pair using SAT solvers. In 2008, Courtois et al. [10] use slide-algebraic attack to cryptanalyze the KeeLoq block cipher using SAT solvers, with a complexity equivalent to about 2^{53} KeeLoq encryptions (with 2^{16} known pairs). In 2009, Courtois et al. [11] describe a full-key recovery attack on the Hitag2 stream cipher. The Master's thesis of Vörös [12] lists practical algebraic attacks on some other stream ciphers.

Although algebraic attacks have a few success stories, the general time complexity of these attacks is prohibitively high. The main problem of applying algebraic attacks to the case of block ciphers is that the size of the final solvable system becomes unmanageably huge. As a result, the attack complexity exceeds the complexity of brute-force search. For example, in the case of 128-bit AES, a direct application of XL produces a solvable system for $D = 18$ [6], but the size of the solvable system is very large (about 2^{110}). As a result, the complexity to solve that system (more than 2^{220} with sparse system solvers) exceeds the complexity of brute-force search (at most 2^{128} encryptions). To bring down the complexity of algebraic attacks on block ciphers, one possibility is to reduce the size of the final system so that the system can be generated and solved efficiently. Since the linearized equations generated by XL are usually sparse, special sparse system-solving algorithms may be exploited in the context of XL.

Our Contribution: In this paper, we propose a new heuristic to improve the XL method by reducing the size of the final linearized system. The heuristic uses the structured Gaussian elimination (SGE) algorithm [13] to reduce the growth of the number of variables during the expansion stage of XL. It also helps by decreasing the number of linearly dependent equations. SGE sometimes exhibits excessive reduction in the system size (a phenomenon called *avalanche effect*) which adversely affects the application of SGE in tandem with XL. We control the avalanche effect by tuning a heuristic parameter. Experiments carried out on small systems and toy ciphers indicate that our heuristic holds the promise of bringing down the complexity of XL.

In short, the basic novelty of our work is the application of sparse system-solving techniques in the *expansion phase* of the standard XL algorithm. Two main improvements of the XL algorithm, already available in the literature, are XSL [6] and MutantXL [7]. Both of these are capable of generating smaller linearized systems compared to XL. However, neither of these seems to be practical for solving real-life ciphers like 128-bit AES. Our heuristic too does not imme-

diately lead to a practical cryptanalytic method for AES (or, for that matter, for any other real-life cipher). It is instead proposed as another improvement of XL with the hope that it may throw some insight in research pertaining to algebraic attacks. Indeed, most of the current algebraic attack techniques are essentially heuristic in nature, and many of them lack solid analytic foundations. Our method too seems promising only from the positive results we obtained from our experimental experience with it. As a final remark, we mention that XL_SGE is, by design, not competing with XSL or MutantXL. On the contrary, it can be used to boost the performance of these XL variants in the same way as it aids XL. Currently, we have experimented with XL only.

The rest of the paper is organized as follows. Section 2 provides some basic background on algebraic attacks over AES-like block ciphers. In particular, it describes the XL algorithm. Moreover, we briefly discuss the structured Gaussian elimination procedure in this section. In Sections 3, we propose our algorithm XL_SGE. In Section 4, we supply our experimental results, and compare the performance of XL_SGE with that of XL. We conclude the paper in Section 5 after highlighting scopes for further research in this direction.

2 Background

In this section, we briefly describe algebraic attacks and the structured Gaussian elimination procedure.

2.1 Algebraic Attack on AES-like Ciphers

In August 2000, the block cipher Rijndael [14] was selected as the Advanced Encryption Standard (AES). Rijndael is a key-iterated block cipher with a strong algebraic structure. AES can be represented as algebraically closed equations over GF(2^8) [6]. It can also be represented as a system of multivariate *quadratic* equations over GF(2) with plaintext, ciphertext and key bits as variables.

The *MQ problem* is the problem of solving systems of multivariate quadratic equations. The MQ problem is NP-Hard for a general field [15]. Solving a system of quadratic equations over any *finite* field is NP-Complete [12] (since over a finite field, one can verify a correct solution in polynomial time). In general, no polynomial-time algorithm is known to solve the MQ problem. However, for overdefined systems of multivariate quadratic equations (Number of equations $\gg$ Number of variables), there exist algorithms which can run in polynomial time under certain conditions [4, 5].

An algebraic attack consists of two basic steps: (1) Equation generation, and (2) Solving the system of equations. These steps are briefly described below.

Equation Generation
Usually, a block cipher consists of a linear part and a nonlinear part. The nonlinear part is due to the presence of S-Boxes in the cipher. Constructing equations for the linear part is trivial. To construct the equations for the nonlinear part

of the cipher, one follows two different approaches. First, the structure of the S-Boxes is exploited to generate equations. Second, one uses the null-space equations for the S-Boxes. For our experiments, we have used a scaled-down version of AES (Baby Rijndael) as described in [15]. Baby Rijndael has the same algebraic structure as AES. The block size and the key size of baby Rijndael are 16 bits. The linear layer of baby Rijndael yields linear equations, whereas the S-Boxes produce quadratic equations. Using the inverse function used in the S-Box of baby Rijndael, one obtains 11 quadratic equations per S-Box. Computing the null space for each S-Box yields 21 linearly independent equations. For details on how the equations are generated, we refer the reader to [15].

Solving the System of Equations
The usual method to solve overdefined multivariate systems of equations is to use Gröbner-basis algorithms. The fastest of such algorithms are F_4 and F_5 proposed by Faugère [2, 3]. The XL (eXtended Linearization) algorithm was proposed as an efficient alternative [5]. For a system of m quadratic equations with n variables, the algorithm is expected to run in polynomial time with an exponent $O(1/\sqrt{\epsilon})$, if $m \geq \epsilon n^2$, $0 < \epsilon \leq 1/2$. In general, the XL algorithm is expected to run in subexponential time. A third approach based upon SAT solvers is also proposed in the literature [8] to solve systems of multivariate algebraic equations.

2.2 eXtended Linearization (XL)

The XL algorithm is effective when the number of equations exceeds the number of variables. The main idea is to increase the number of initial equations by adding new algebraically dependent equations which are linearly independent of the initial system. This system expansion is carried out using multiplications by monomials of limited degrees.

The XL algorithm accepts as input the initial system of equations $\mathbb{A}$ (which has at least one solution), and a degree bound $D \in \mathbb{N}$. The steps of the algorithm are described now.

1. **Multiply:** Generate the new system $\mathbb{A}'$:
$$\mathbb{A}' = \bigcup_{0 \leq k \leq D - d_{max}} X^k \mathbb{A},$$
where X^k stands for the set of all monomials of degree k, and d_{max} is the maximum degree of the initial system of equations.
2. **Linearize:** Consider each monomial in the variables x_i of degree $\leq D$ as a new variable, and perform Gaussian elimination on the system $\mathbb{A}'$. The ordering of the monomials must be such that all the terms containing single variables (like x_1) are eliminated last.
3. **Solve:** Assume that Step 2 yields at least one univariate polynomial equation in some variable x_1. Solve this equation over the underlying finite field using a standard root-finding algorithm.
4. **Repeat:** Simplify the equations, and repeat the process to find the values of the other variables.

2.3 Structured Gaussian Elimination

Structured Gaussian Elimination (SGE) is an algorithm used to reduce the dimension of a sparse matrix by eliminating some of its rows and columns [13]. SGE exploits the special structure of the matrices arising from integer-factorization and discrete-logarithm algorithms. It is applicable for sparse matrices where the columns can be divided into two types: heavy-weight and light-weight. It is a heuristic procedure that tends to preserve the sparsity of the light columns.

SGE repeats the following steps until no further reduction is possible.

1. Delete columns of weight 0 and 1.
2. Delete rows of weight 0 and 1.
3. Delete rows of weight 1 in the light part. After Step 2 and Step 3, update column weights.
4. Delete redundant rows.

3 eXtended Linearization with Structured Gaussian Elimination (XL_SGE)

3.1 Motivation

The problem with the XL algorithm is that the size of the system increases drastically with the increase in the degree bound D used in the algorithm. Many linearly dependent equations are generated during the expansion process (Step 1) in XL. The equations generated by the XL algorithm are generally very sparse. Moreover, we have observed, from the statistics of the system obtained in XL (for $D = 2$), that the columns of the generated system can be distinguished as heavy-weight and light-weight. Depending on these observations, we propose a new heuristic (XL_SGE) to reduce the number of linearized equations in XL. According to the heuristic, the generated intermediate systems are reduced using structured Gaussian elimination (SGE). The reduced systems are multiplied with monomials to get systems of higher algebraic degrees.

The XL_SGE algorithm reduces the sizes of the intermediate systems of equations in XL using the first three steps of structured Gaussian elimination. It does not use the apparently irrelevant fourth step of SGE. The main motivation behind proposing XL_SGE is size reduction. Besides this, XL_SGE is expected to exhibit some side effects, some of which can be exploited to our advantage. For example, partial elimination of variables before each stage of monomial multiplication may result in the generation of fewer linearized variables (higher-degree monomials). This, in turn, is capable of reducing the rank deficit. As a result, we may even expect a smaller degree bound D than XL for arriving at a solvable system. One should, however, avoid the avalanche effect of SGE, which results in a slow growth of the linearized system, demanding larger values of D than needed in XL.

3.2 XL_SGE Algorithm

The XL_SGE algorithm accepts as input the initial system of equations (consisting of linear equations and quadratic equations) $\mathbb{A}$ (which has at least one solution), and a degree bound $D \in \mathbb{N}$. The basic steps of the XL_SGE expansion procedure are as follows.

1. Expand the initial system of equations $\mathbb{A}$ up to degree $d = 2$ using XL to obtain a linearized system $\mathbb{A}'$.
2. Apply structured Gaussian elimination (SGE) on $\mathbb{A}'$ to obtain a reduced system of equations $\mathbb{A}''$ of degree d.
3. Multiply the reduced system $\mathbb{A}''$ with monomials of degree 1, append the generated equations to $\mathbb{A}''$, and rename this appended system as $\mathbb{A}'$. $\mathbb{A}'$ now contains equations of degrees up to $d + 1$.
4. If the degree of the system of equations $\mathbb{A}'$ is D, end the process. Otherwise, go to Step 2.

If we get a full-rank system (or a close-to-full-rank system) for a particular D, we solve that system. Otherwise, we increase the degree bound D, and run XL_SGE again to obtain a system of smaller rank deficit. This process is repeated until the rank deficit becomes zero or goes below a tolerable limit.

We have observed that sometimes due to avalanche effect, most of the equations are removed in the SGE stage. Consequently, XL_SGE suffers from a slow growth in the size of the linearized system with the increase in the degree bound D, and the rank deficit in XL_SGE decreases much more slowly with D than in XL. To reduce this avalanche effect, we use a parameter K in Step 2 of the XL_SGE algorithm. Suppose that the j-th column has weight 1 with the non-zero entry appearing in the i-th row. Only if this row contains at least K non-zero entries, the i-th row and the j-th column are removed. The value of K is heuristically chosen depending upon the weight distribution of the rows.

An optional preprocessing of $\mathbb{A}$ offers a possibility of initial reduction in the system size. As mentioned during the description of baby Rijndael, we get both linear and quadratic equations from the encryption rounds. If we substitute the linear equations in appropriate quadratic equations, we can eliminate some of the variables, and remove all the linear equations from the initial system $\mathbb{A}$. The reduced system consisting only of quadratic equations is expanded. Although the number of non-zero terms in each quadratic equation increases because of these substitutions, the effects of this increase can be appropriately handled. However, whether this initial reduction helps at all is not clear from our experiments.

4 Experimental Results

We have tried the heuristic (XL_SGE) on small random sparse quadratic systems, and have found that the heuristic significantly improves the performance of the XL algorithm in most cases, in terms of the size of the final system. The results obtained for some small random systems are shown in Table 1. This table

corresponds to $K = 0$, that is, the avalanche effect for SGE is not handled in these experiments. The initial system size $x \times y$ indicates x quadratic equations in y variables. On the other hand, the final system size $x \times y$ indicates x linearized equations in y monomials. For both XL and XL_SGE, we report the final system size. Notice that we apply SGE *before* each stage of monomial multiplication. This, in turn, implies that the final systems in XL_SGE, reported in all the tables below, are again expected to reduce in size if another round of SGE is applied to them. Indeed, it is a standard practice to apply SGE to any large sparse system before solving it. The final systems available from XL would also experience size reduction upon application of a round of SGE. For both XL and XL_SGE, the sizes reported in the tables correspond to those systems before that external application of SGE which may be used to solve the systems.

Size of Initial System	D in XL_SGE	System Size after XL_SGE	Rank Deficit	D in XL	System Size after XL	Rank Deficit
10×6	3	67×27	0	3	149×42	0
15×8	3	231×87	0	3	276×93	0
20×10	3	427×156	0	3	500×172	0
20×10	6	3959×655	0	5	7445×638	0
20×12	7	2809×917	11	7	98611×3302	0
20×12	7	5006×1547	10	7	114863×3302	0
20×12	3	714×271	0	3	795×299	0
22×12	3	708×209	0	4	5464×794	0
22×12	3	897×263	0	4	6478×794	0
24×13	3	1029×375	0	3	1137×378	0
24×14	3	1085×449	0	5	44476×3473	0

Table 1. Comparison of XL with XL_SGE (with $K = 0$) for random systems

From the experimental results, it is clear that for the same degree bound (D), the size of the final system obtained from XL_SGE is in most cases much smaller than the size of the final system obtained from XL. There are instances where larger degree bounds D are needed by XL_SGE (than XL) for obtaining a full-rank system, but the size reduction is always a positive feature of XL_SGE. The performance of the XL_SGE algorithm hugely depends on the structure of the initial system of equations. We have observed that if the initial system of equations enjoys the following two properties, XL_SGE performs significantly better than XL.

1. Number of equations $\gg$ Number of variables
2. Number of equations $\ll$ Number of one-degree terms + Number of two-degree terms

There are cases where XL performs better than XL_SGE. In some cases, XL generates a full-rank system, whereas XL_SGE fails to generate a full-rank

system. Consider the example of Row 5 of Table 1. In this case, we get a full-rank system for $D = 7$ using XL. For the same D, the rank deficit in case of XL_SGE is 11. However, the size of the final system in case of XL is much larger, that is, a slightly increased rank deficit for XL_SGE is more than compensated by a dramatic reduction in the system size. Interestingly, for the same initial system, we obtain a system of size 502×253 and rank deficit 2 using XL_SGE for $D = 3$ (using XL with $D = 3$, the system size is 639×296 with rank deficit 10).

The performance of XL_SGE also depends on the proportion of linear equations and quadratic equations present in the initial system. For some systems, we get a full-rank system after few iterations of XL_SGE (say, for $D = 3$). So the size of the final system is small in those cases. For some other systems of the same initial size, we get full-rank systems after more number of iterations of XL_SGE (say, for $D = 6$). In those cases, the size of the final system is large. Consider the examples of Row 3 and Row 4 of Table 1. In both cases, the sizes of the initial systems are the same. The initial system of the third row contains 4 linear equations and 16 quadratic equations. The initial system of the fourth row contains 2 linear equations and 18 quadratic equations. In the case of Row 3, XL_SGE gives a full-rank system of size 427×156 for $D = 3$, whereas for Row 4, we get a full-rank system of size 3959×655 for $D = 6$.

The main problem with XL_SGE is the avalanche effect suffered by the SGE stage. If any intermediate generated system of XL_SGE experiences avalanche effect, no further increment in the size of the system is possible. In that case, XL_SGE fails to generate a full-rank system, no matter how large the degree bound D is. In some cases, little reduction takes place (depends on the structure of the initial system) with XL_SGE. In those cases, the performances of XL_SGE and XL are similar.

Size of Initial System	D in XL_SGE	K in XL_SGE	System Size after XL_SGE	Rank Deficit	D in XL	System Size after XL	Rank Deficit
22×12	3	4	513×292	0	3	534×298	0
23×13	4	4	2863×1073	0	5	11219×2379	0
24×13	3	0	726×377	0	3	726×377	0
24×15	4	0	6400×1940	0	4	6451×1940	0
24×16	4	7	6311×2516	0	4	6587×2516	0
24×16	4	5	6513×2516	0	4	6527×2516	0
25×17	4	6	8609×3213	0	4	8609×3213	0
25×18	5	6	34027×12615	0	5	36825×12615	0

Table 2. Comparison of XL with XL_SGE (with $K \geq 0$) for random systems

Table 2 lists results on some small random systems with the avalanche effect taken into account. For a given D, we have tuned the parameter K in the sequence $0, 1, 2, \ldots$ until we obtain a value of K for which the rank deficit of the expanded system is zero. In all our experiments, we could locate suitable values

for K (although there is no theoretical guarantee that such a K must exist). These results once again illustrate the superiority of XL_SGE over XL in terms of the size of the final solvable system.

Table 3 describes the variation of the performance of the XL_SGE expansion procedure with the parameter K for a random initial system of size 25×18. This is the same system reported in the last row of Table 2. In general, for small values of K, the size reduction in SGE may be too high, that is, the avalanche effect may set in. This may lead XL_SGE to obtain higher rank deficits compared to XL for the same degree bound D. On the other hand, if K is too large, SGE fails to reduce the intermediate system sizes, and consequently, the performance of XL_SGE becomes identical to that of XL. A good value of K can be experimentally chosen for a given input system.

K	$D = 3$		$D = 4$		$D = 5$	
	System Size	Rank Deficit	System Size	Rank Deficit	System Size	Rank Deficit
0	922×975	271	6015×4047	294	28070×12615	131
5	958×976	244	6357×4047	132	30043×12615	38
6	1032×982	192	7043×4047	19	34027×12615	0
8	1050×983	179	7214×4047	10	35014×12615	0
10	1086×987	154	7556×4047	4	36988×12615	0

Table 3. Dependence of the performance of XL_SGE on the parameter K

Depending on the initial structure of the system, some modifications of the XL_SGE algorithm may improve the performance of the algorithm. The exact nature of this dependence is not clear yet. To see whether XL_SGE works well on the systems generated by AES-like block ciphers, we have generated systems of equations for the toy version of AES (Baby Rijndael) as described in Section 2.1. On this system, XL_SGE exhibits slightly better performance than XL. The results are shown in Table 4.

Number of Rounds	Size of Initial System	K in XL_SGE	System Size after XL_SGE	Rank Deficit	System Size after XL	Rank Deficit
1	232×64	0	142945×43745	0	178892×43745	0
2	448×112	3	634810×233633	24	642423×234225	48
3	664×160	7	1755432×682273	576	1768628×682401	576

Table 4. Comparison of XL with XL_SGE for baby Rijndael for $D = 3$

We have also reduced the initial system (according to the last paragraph of Section 3) of baby Rijndael for one round, and get a system of 192 quadratic equations in 24 variables. After expanding that system using XL_SGE, we get

a final system of size 97447×12919 for $D = 4$ with rank deficit 36. On the other hand, XL gives a final system of size 97943×12919 with rank deficit 36 for the same D. It, therefore, remains uncertain whether the preprocessing of the initial system (that is, absorbing the linear equations in the quadratic equations) produces any noticeable benefits at all.

The programs for generating equations and expanding equations using XL and XL_SGE were written in the C programming language. The PARI/GP package was used to carry out some intermediate calculations needed to generate equations. The mathematical package Sage (Version 4.4.2) was used to calculate the rank of sparse matrices available from XL and XL_SGE.

5 Conclusion

The main problem with algebraic attacks on block ciphers is that the solvable system size becomes large, and so the complexity to solve the system often exceeds the complexity of brute-force search. XL generates too many linearly dependent equations while expanding the initial system of equations. The number of variables also grows rapidly during the expansion stage of XL. Our proposed heuristic XL_SGE uses structured Gaussian elimination in order to improve the performance of XL by reducing the growth of variables and of linearly dependent equations in the expansion stage of the XL algorithm. Experiments reveal that XL_SGE performs better than XL in many cases for random systems and also for a toy version of AES.

We end this paper after highlighting some directions for future research.

- It is not yet clear on which factors the performance of XL_SGE depends. A theoretical analysis of XL_SGE is required, and accordingly modifications of our present algorithm are called for to make it more versatile and effective. As an example, the nature of dependency of the performance of XL_SGE on the choice of the heuristic parameter K needs to be analytically investigated.
- Columns of weight two can be eliminated in the SGE phase without increasing the number of non-zero entries in the matrix. However, elimination of columns of weight three or more cannot be so gracefully handled.
- Partial monomial multiplication during the expansion phase can effectively reduce the size (both the number of variables and the number of equations) of the final solvable system. Moreover, after each application of SGE, all columns have weight at least two. Complete monomial multiplication on this system can never generate columns of weight zero or one. Partial monomial multiplication can potentially solve this problem, but possibly at the cost of degradation in the rank profile with increasing D.
- Another important area of investigation is to use SGE in conjunction with the variants of XL (like XSL and MutantXL) already proposed in the literature. Comparisons with other algebraic-attack algorithms (like F_4, F_5, SAT-solver techniques) are also worth studying.

References

1. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal **28** (1949) 657–715
2. Faugère, J.C.: A new efficient algorithm for computing Gröbner basis (F_4). Journal of Pure and Applied Algebra **139**(1) (1999) 61–88
3. Faugère, J.C.: A new efficient algorithm for computing Gröbner basis without reduction to zero (F_5). ISSAC '02 (2002) 75–83
4. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: CRYPTO. (1999) 19–30
5. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: EUROCRYPT. (2000) 392–407
6. Courtois, N., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: ASIACRYPT. (2002) 267–287
7. Ding, J., Buchmann, J., Mohamed, M., Moahmed, W., Weinmann, R.: Mutantxl. In: SCC. (2008) 16–22
8. Bard, G., Courtois, N., Jefferson, C.: Solution of sparse polynomial systems over GF(2) via sat-solvers. In: ECRYPT workshop Tools for Cryptanalysis. (2007)
9. Courtois, N., Bard, G.V.: Algebraic cryptanalysis of the data encryption standard. In: IMA Int. Conf. (2007) 152–169
10. Courtois, N., Bard, G.V., Wagner, D.: Algebraic and slide attacks on Keeloq. In: FSE. (2008) 97–115
11. Courtois, N., O'Neil, S., Quisquater, J.J.: Practical algebraic attacks on the Hitag2 stream cipher. In: ISC. (2009) 167–176
12. Vörös, M.: Algebraic attack on stream ciphers. Master's thesis, Comenius University, Faculty of Mathematics, Physics and Informatics, Department of Computer Science (2007)
13. LaMacchia, B., Odlyzko, A.: Solving large sparse linear systems over finite fields. In: Advances in Cryptology-CRYPT0 90. Volume 537 of LNCS. (1991) 109–133
14. Daemen, J., Rijmen, V.: Rijndael for AES. In: AES Candidate Conference. (2000) 343–348
15. Kleiman, E.: The XL and XSL attacks on Baby Rijndael. Master's thesis, Iowa State University, Department of Mathematics (2005)