Improvements of Algebraic Attacks Based on Structured Gaussian Elimination

Satrajit Ghosh
Abhijit Das

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

Abstract. Algebraic attacks are studied as a potential cryptanalytic procedure for various types of ciphers. The XL_SGE algorithm has been recently proposed to improve the complexity of the XL attack. XL_SGE uses structured Gaussian elimination (SGE) during the expansion phase of XL. In this paper, we establish that XL_SGE suffers from some serious drawbacks that impair the effectiveness of SGE-based reduction at all multiplication stages except the first. In order to avoid this problem, we propose several improvements of XL_SGE. Our modifications are based upon partial monomial multiplication and handling of columns of weight two. Our modified algorithms have been experimentally verified to be substantially superior to XL_SGE.

Keywords: Multivariate polynomial equation, algebraic attack, linearization, XL, sparse linear system, structured Gaussian elimination

1 Introduction

Algebraic attacks pertain to cryptanalyzing some types of cipher that can be rephrased in terms of multivariate quadratic equations in the bits of the plaintext, the ciphertext and the key. In general, solving such systems over finite fields is an NP-Complete problem. However, when the multivariate system is overdefined, some algorithms faster than exponential-time are known. These algorithms can be broadly classified into three categories.

- Algorithms based on Gröbner basis computations: These include F_4 and F_5 algorithms [1, 2].
- Algorithms based on linearization: Kipnis and Shamir's relinearization [3], Courtois et al.'s eXtended Linearization (XL) [4], Courtois and Pieprzyk's eXtended Sparse Linearization [5] and Ding et al.'s MutantXL [6].
- Algorithms based on SAT solvers: Bard et al.'s algorithm [7]

Although superficially different, these approaches are occasionally proved to be computationally equivalent. For example, see [8].

Some cryptosystems have been successfully cryptanalyzed by algebraic attacks [3, 9–12]. However, these attacks are usually not practical for most ciphers.

For example, Courtois and Pieprzyk [5] estimate an effort of nearly 2^{230} bit operations to cryptanalyze 128-bit AES [13] using XSL. Consequently, it is of important research concern to propose practical improvements of known algorithms on algebraic attacks.

Our recent proposal XL_SGE [14] uses a heuristic to improve the performance of the XL method by reducing the size of the final linearized system. It uses structured Gaussian elimination (SGE) [15] 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. There is an apparent resemblance of this proposal with WXL or its parallel version PWXL discussed in [16], but we argue that this resemblance is only superficial. Sparse system solvers are used in [16] as a faster alternative to Gaussian elimination, and may be used in conjunction with SGE as indicated in the conclusion of that paper. On the contrary, XL_SGE uses SGE at every stage of monomial multiplication in order to curb the growth of the linearized system by setting aside some equations and preventing them to take further role in the expansion process. At each level, we can use, as WXL does, any sparse system solver to identify the solvability of the system.

XL_SGE, however, suffers from a design flaw, and it is this flaw that we address in this paper. In many cases, the reduction in the system size obtained by XL_SGE is not significant, as is evident from the experimental results presented in [14]. We identify some sources of ineffectiveness of XL_SGE, and propose improvements to overcome these shortcomings of XL_SGE. These improvements make use of two novel techniques. First, random monomial multiplications are used during the multiplication stage of XL_SGE. Second, variables with column weight two are considered in the SGE stage of the algorithm. Experiments carried out on small systems and toy ciphers indicate that our new heuristic ideas bring down the complexity of XL_SGE substantially. Indeed, the new proposals exploit more prudently the fact that an overdetermined system is expected to have the same solution space even if we throw away a sizeable fraction of the equations. This constrains the growth in the multiplication phase further (compared to XL_SGE), hopefully without having significant detrimental effects on the rank profile.

Like XL_SGE, these new proposals, although heuristically stronger and experimentally validated on random systems and toy ciphers, lack ironclad theoretical guarantees of effectiveness. Nevertheless, we believe that these proposals merit attention of the algebraic cryptanalysis community.

The rest of this paper is organized as follows. Section 2 describes the background material needed for understanding our modifications. In particular, the algorithms for XL, SGE and XL_SGE are reproduced in the form presented in [14]. In Section 3, we identify the weaknesses of XL_SGE. In Section 4, we propose two different ways of repairing these drawbacks of XL_SGE. Our experimental results and associated remarks follow in Section 5. The concluding Section 6 highlights some scopes for further research in this direction.

2 A Description of XL_SGE

In what follows, we assume that we are given a system $\mathbb{A}$ over $GF(2)$ of multivariate equations, some of which are quadratic and the rest of which are linear. Such systems are available from block ciphers like AES. We assume that the initial system is sparse and has (at least) one solution. We concentrate upon linearizing the system followed by solving the generated linear system using well-known linear-algebra tools.

2.1 eXtended Linearization (XL)

The XL algorithm [4] is presented below as Algorithm 1. In addition to the initial system $\mathbb{A}$, a degree bound D is also supplied as an input to XL.

Algorithm 1: Extended Linearization (XL) of multivariate systems

---

1. **Multiply:** Generate the new system $\mathbb{A}'$:
$$\mathbb{A}' = \bigcup_{0 \le k \le D - d_{max}} X^k \mathbb{A},$$
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 $\le D$ 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.

2.2 Structured Gaussian Elimination

Structured Gaussian Elimination (SGE) [15] is an algorithm that can substantially reduce the sizes of sparse linear systems. Its working is based upon classifying columns as heavy-weight and light-weight. Algorithm 2 describes one iteration of SGE.

Algorithm 2: Structured Gaussian Elimination (SGE)

---

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.

2.3 eXtended Linearization with Structured Gaussian Elimination (XL_SGE)

Step 1 of XL multiplies the initial system by all available monomials in one shot so that the degrees of the freshly generated equations is no larger than D. XL_SGE replaces this by $D - 2$ stages of multiplications by the initial set of variables. After every multiplication stage, the system is reduced by applying SGE. This reduction controls the growth of the system in the linearization process by eliminating some rows and columns and also by reducing the number of equations generated in each subsequent multiplication stage. XL_SGE uses only the first three steps of SGE. Algorithm 3 describes the stages of XL_SGE. The degree bound D is so adjusted that the final linearized system is of full (or nearly full) rank.

Algorithm 3: Extended Linearization with Structured Gaussian Elimination (XL_SGE)

---

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.

XL_SGE [14] controls excessive reduction of intermediate systems due to avalanche effects by using a heuristic parameter K during the application of SGE. More specifically, the i-th row and the j-th column are eliminated if and only if the following three conditions are satisfied.

- The j-th column has weight 1.
- The (i, j)-th entry is non-zero (1, to be precise).
- The weight of the i-th row is at least K.

In this paper, we use the parameter K exactly in the same way.

3 Analysis of XL_SGE

XL_SGE is designed to reduce the size of the final solvable system in comparison with XL. However, there are many instances where this size reduction is not substantial. There are even situations where an application of SGE increases (though only slightly) the number of equations compared to XL. Thus, the basic goal of arriving at reduced systems is often not achieved by XL_SGE.

XL_SGE adopts a *layer-wise multiplication* strategy. At the d-th layer, we have a system $\mathbb{A}'$ of maximum (algebraic) degree d. First, SGE is applied on

$\mathbb{A}'$ to get a reduced system $\mathbb{A}''$ of the same degree d. Then, $\mathbb{A}''$ is multiplied by monomials of degree 1 (variables) to get a system (renamed again as $\mathbb{A}'$) of degree $d+1$. This is repeated until the degree of the equations in $\mathbb{A}'$ reaches a predetermined bound D.

Our experiments reveal that SGE on $\mathbb{A}'$ for $d = 2$ yields sizable reduction in the system size. Subsequently, for $d \geq 3$, SGE progressively loses effectiveness in bringing down the system size. From the column-weight distribution, we observe that for $d = 2$, $\mathbb{A}'$ contains many columns of weight 1. For $d \geq 3$, such columns are rare in $\mathbb{A}'$, so Step 1 in Algorithm 2 is executed for only a few number of times.

These experimental observations can be justified intuitively. After the initial expansion for $d = 2$ using XL, $\mathbb{A}'$ is expected to contain many variables with column weight one. After SGE reduces this system, all columns that remain are of weights at least two. The subsequent monomial-multiplication stage generates new variables each expected to be of column weight at least two. Consider a variable x in the system of equations after the last application of SGE. The column weight of x is at least two at this point, that is, x appears in at least two equations. Let y be a monomial of degree 1. If the system of equations is multiplied by the monomial y, then x is multiplied by y at least twice, so the column weight of the new variable xy will be at least two. There may, however, be some cancellation of terms (after algebraic simplification using $a^2 = a$ for any initial variable a and $z + z = 0$ for any linearized variable z). If there are many initial variables, this phenomenon does not occur frequently.

While applying SGE for $d \geq 3$, only Step 3 of Algorithm 2 can create new columns of weight 1 (or 0) by deleting certain rows. Since monomial multiplication increases the size of the system, some previously heavy columns may turn light after monomial multiplication, potentially creating new avenues for row deletion in Step 3. Step 2 of Algorithm 2 is quite ineffective, since the previous round of SGE leaves no rows of weight 0 or 1, and monomial multiplication does not reduce row weights except in rather infrequent situations (like multiplication of $x_1x_2 + x_2$ by x_1 in an equation).

As an example, let a, b, c, d be $GF(2)$-valued variables. Suppose that an intermediate application of SGE leaves us with the following linearized system of equations of algebraic degree three. The linearized variables in the system are a, b, d, abc, bc, and ad each with column weight at least two.

$$\begin{aligned}
abc + bc \quad\quad + a \quad\quad\quad &= 0 && (1)\\
abc + bc \quad\quad\quad\quad + b + d &= 1 && (2)\\
abc \quad\quad + ad \quad\quad + b \quad\quad &= 1 && (3)\\
ad + a \quad\quad + d &= 0 && (4)
\end{aligned}$$

To generate a system for the next degree four, XL_SGE multiplies the system with the variables a, b, c, d. This yields the following system with 15 equations and 14 variables. Eqn (8) is generated twice (Eqn (4) $\times$ a and Eqn(7) $\times$ a), but is shown only once.

$$a = 0 \tag{5}$$
$$ab + ad + a = 0 \tag{6}$$
$$abc + ab + ad + a = 0 \tag{7}$$
$$abc + ab + bc = 0 \tag{8}$$
$$abc + bc + bd = 0 \tag{9}$$
$$abc + abd = 0 \tag{10}$$
$$abd + ab + bd = 0 \tag{11}$$
$$abc + ac + bc = 0 \tag{12}$$
$$abc + cd + c = 0 \tag{13}$$
$$abc + acd + bc + c = 0 \tag{14}$$
$$acd + ac + cd = 0 \tag{15}$$
$$abcd + bcd + ad = 0 \tag{16}$$
$$abcd + bcd + bd = 0 \tag{17}$$
$$abcd + ad + bd + d = 0 \tag{18}$$
$$d = 0 \tag{19}$$

The monomial abc appears thrice in the system (4)–(7). Multiplication of these equations by d generates three occurrences of $abcd$ in the expanded system (8)–(22). The monomial bd appears in the expanded system four times, twice from multiplying the term b in Eqns (5) and (6) by d, and twice from multiplying the term d in Eqns (5) and (7) by b. There is also a case of cancellation arising out of algebraic simplification. For example, ad appears twice in the system (4)–(7). Multiplying Eqn (6) by d leaves the term ad, but multiplying Eqn (7) by d removes this term. Unfortunately, however, the term ad appears from other sources too. For example, Eqn (4) $\times$ d and Eqn (5) $\times$ a give Eqns (19) and (9) respectively, each containing the non-zero term ad.

To sum up, all the monomials appearing in the expanded system (8)–(22) happen to have column weights two or more. When SGE is applied to this expanded system, no variable and equation can be removed by Step 1 of SGE (Algorithm 2).

4 Improvements of XL_SGE

To ensure reduction of system sizes by SGE for all degrees of $\mathbb{A}'$, two possibilities can be explored. First, we investigate how variables of column weight one may reappear in the system. Second, we look into modifying SGE to work even when all variables have column weights two or more.

– **Partial monomial multiplication:** Suppose that a variable x appears in two or more equations. When both these equations are multiplied by the same monomial y of degree one, the common variable xy appears in both the new equations. If one of these multiplications is skipped, the number of occurrences of xy (that is, its column weight) reduces by one. For example, consider the expansion of the system (4)–(7) to generate the system (8)–(22). The term abd appears in both Eqns (13) and (14) upon multiplication of ad in Eqns (6) and (7) by b. If we skip the second multiplication, we no longer generate Eqn (14), so abd occurs with column weight one. Similarly, if we avoid the multiplication of Eqn (7) by c (so that Eqn (18) is not generated), we reduce the column weights of both ac and cd. Note that ac occurs in Eqn (15) and (18) from the common term a in Eqns (4) and (7), whereas cd occurs in Eqns (16) and (18) from the common term d in Eqns (5) and (7). Therefore, skipping a single multiplication leaves both ac and cd with column weight one. Finally, the variable ad occurs in four equations of the expanded system, so skipping only one of the four multiplications yielding this variable cannot bring down the column weight of ad to one.

The above discussion highlights that carefully skipping certain monomial multiplications has some benefits. First, fewer equations are generated, and second, SGE may again discover variables of column weight one. On the darker side, generation of fewer equations may adversely affect the rank profile of the expanded system. If, however, too many monomial multiplications are not skipped, we hope not to encounter a big trouble with the rank profile. Therefore, two important issues are of relevance in this context: which monomial multiplications would be skipped, and how many.

– **Deletion of variables with weight more than one:** Suppose that a variable z appears in $t \geq 2$ equations in an expanded system. If we add one of these equations to the remaining $t - 1$ equations, the column weight of z reduces to one, so SGE (Algorithm 2) can remove this variable in Step 1. This, however, increases the weight of these $t - 1$ equations. This increase in row weights may increase weights of certain columns. That is, an effort to forcibly eliminate z may stand in the way of the elimination of other variables. However, if $t = 2$, this processing of z followed by the removal of the only equation containing z does not increase the total weight of the system. Still, the density (average weight per row or column) of the system increases (since one equation and one variable are now removed), but the expanded systems, particularly if large, are expected to absorb this problem without sufficient degradation of the performance of XL_SGE.

Our modifications of XL_SGE following these two ideas are now elaborated.

4.1 XL_SGE with Random Monomial Multiplication (XL_SGE-2)

For the time being, we skip monomial multiplications randomly, and the amount of skipping is governed by a probability $p \in (0, 1]$. More precisely, each equation is multiplied by each monomial of degree one with probability p (and skipped with probability $1 - p$). If $p = 1$, we have the original XL_SGE algorithm. For $p < 1$, we expect more size reduction compared to XL_SGE. Several choices of p are experimentally studied.

The modified algorithm XL_SGE-2 accepts as input the initial system of equations (both linear and quadratic) $\mathbb{A}$ (which has at least one solution), a degree bound $D \in \mathbb{N}$, the avalanche-control parameter $K \in \mathbb{N}$, and a multiplication probability $p \in (0, 1]$. The steps of XL_SGE-2 follow.

Algorithm 4: XL_SGE with Random Monomial Multiplication (XL_SGE-2)

1. Expand the initial system $\mathbb{A}$ 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. Here, K is used to control the avalanche effect in SGE (see Section 2.3).
3. Multiply each equation in $\mathbb{A}''$ by each monomial of degree 1 with probability p (that is, with probability $1 - p$, a multiplication is skipped). Append the generated equations to $\mathbb{A}''$, and rename this appended system as $\mathbb{A}'$ which 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-2 again to obtain a system with smaller rank deficit. This process is repeated until the rank deficit becomes zero or goes below a tolerable limit.

The multiplication probability p has been heuristically chosen in our experiments. We have worked with several fixed values of p in different layers (degrees d of $\mathbb{A}'$). From our experimental experiences, we recommend values of $p \geq 0.5$. A slight modification in the above algorithm for XL_SGE-2 is also studied. In this variant, monomial multiplications are randomly skipped even in Step 1 (that is, since the very beginning of the expansion process).

Another possibility is to use different probabilities in different layers of multiplication. We study two models for varying p with the degree d of $\mathbb{A}'$. In the first model, we take $p_1 = 1 - \frac{1}{d+1}$, that is, the probability of monomial multiplication gradually increases with the degree d of the expanded system. The motivation behind this choice is that we initially restrict the expansion of the system. If this initial restriction allows us to arrive at a solvable system quickly, we are done. If, on the other hand, the initial restriction leads to large rank deficits, we progressively remove the restriction on the growth of the system. Note that this model can be applied even to Step 1 of XL_SGE-2 (that is, for $d = 1$).

In the second model, we take the gradually decreasing sequence of probabilities $p_2 = \frac{D-d}{D-d+1}$. Initially, the system size is small, so we can afford the system to grow at this stage. As d increases, $\mathbb{A}'$ becomes increasingly large, and restricting the growth of the system gradually controls the eventual growth of the system. Note also that higher-degree monomials appear in the linearized system from a larger number of sources. For example, the lower-degree monomial $y_1y_2y_3$ may appear by multiplying y_2y_3 with y_1 or by multiplying y_1y_3 with y_2 or by multiplying y_1y_2 with y_3. On the contrary, a higher-degree monomial like $y_1y_2 \cdots y_{10}$

may appear in the system in ten different ways: by multiplying $y_1 y_2 \cdots y_{10}/y_i$ by y_i for $i = 1, 2, \ldots, 10$. So $y_1 y_2 \cdots y_{10}$ having column weight one requires more restriction at the expansion stage for $d = 9$ than $y_1 y_2 y_3$ requires at $d = 2$.

XL_SGE fails to exploit Step 4 of the SGE algorithm. Our partial monomial strategy is a possible way to address this issue in the sense that deleting some rows is in effect equivalent to not generating the rows at all.

4.2 Column-weight Two Reduction

As discussed earlier, the original SGE procedure (Algorithm 2) can be modified so as to remove columns of weights two or more. In order that the rank profile of the expanded system does not deteriorate too much, we have experimented with deletion of columns of weight two only. The modified SGE algorithm is described below. The algorithm repeats Steps 1–4 until no further reduction is possible. Notice that this strategy is independent of the partial monomial multiplication strategy described above, and is applicable equally well to both XL_SGE and XL_SGE-2. Moreover, this can be viewed as another approach to effectively exploit Step 4 of the SGE algorithm.

Algorithm 5: Structured Gaussian Elimination with Column-weight Two Reduction (SGE′)

---

1. Delete columns of weight 0 and 1.
2. Delete columns of weight 2: If a column has weight 2, delete one equation corresponding to that variable. Substitute that equation in the other equation. Delete the corresponding column.
3. Delete rows of weight 0 and 1.
4. Delete rows of weight 1 in the light part. After Steps 2–4, update column weights.

Although this heuristic modification of SGE seems to be effective, in the current form it does not work very well. One must not use Algorithm 5 to reduce the initial quadratic system (available after Step 1 of XL_SGE or XL_SGE-2), since random systems at this stage exhibit the tendency of losing all quadratic variables. Using the modified SGE for all $d \geq 3$ sometimes show good performance. But the general observation is that the system suffers from drastic reduction in size (a form of avalanche effect) resulting in degraded rank profile and demanding a large number of iterations (that is, large values of D). It appears that the modified SGE procedure of Algorithm 5 should be skipped for certain small values of d (in addition to $d = 2$). However, the exact range of applicability of Algorithm 5 (that is, the minimum d from which it is safe to use this algorithm) has not yet been experimentally or theoretically determined. Such a study would require initial systems larger than what we have experimented with.

Table 1. Performance of XL_SGE-2
(Random monomial multiplication done in Step 1 of Algorithm 4)

Initial	XL			XL_SGE-2 *					XL_SGE-2 †			
system size	D	System Size	δ	D	K	p	System Size	δ	D	K	System Size	δ
12×7	4	317×98	0	3	3	0.75	39×35	0	3	3	35×36	2
13×8	3	275×92	0	3	4	0.67	158×91	0	3	5	132×92	0
13×9	3	273×129	0	3	4	0.67	183×122	0	3	5	142×124	3
15×10	3	388×175	0	3	3	0.67	210×169	0	3	6	182×173	3
16×10	3	309×175	0	3	6	0.67	222×174	0	3	7	177×171	6
16×10	4	1229×385	0	4	3	0.67	686×362	0	4	7	619×385	1
16×11	5	4450×1023	0	5	6	0.80	3763×869	0	5	7	2963×1018	2
16×12	5	5232×1585	0	5	7	0.67	4163×1584	0	5	8	2693×1557	4
17×12	3	548×298	0	3	8	0.67	394×297	0	3	8	273×289	26
17×12	4	1758×793	0	4	8	0.67	1217×788	1	4	7	941×781	12
17×12	4	1984×792	0	4	7	0.67	1517×785	0	4	8	1028×788	12
17×12	4	2074×793	0	4	7	0.75	1506×758	0	4	8	1086×786	7
17×13	5	8889×2379	0	5	6	0.67	7543×2378	0	5	7	5383×2379	0
18×12	3	628×298	0	3	5	0.75	481×296	0	3	7	309×295	2
18×13	4	2796×1092	0	4	6	0.67	1954×1091	0	4	8	1521×1092	1
18×14	4	3333×1470	0	4	8	0.67	2489×1470	0	4	8	1750×1469	6
19×14	4	4154×1470	0	4	4	0.80	3520×1457	0	4	8	2267×1466	2
19×14	4	4473×1470	0	4	5	0.67	3375×1470	0	4	8	2183×1468	2
19×14	4	4500×1470	0	4	4	0.75	3707×1460	0	4	4	2484×1467	2
19×15	3	1247×575	1	3	4	0.67	841×574	1	3	8	583×572	3
20×14	4	3212×1470	0	4	7	0.67	2255×1470	0	4	8	1562×1460	7
20×15	4	4640×1940	0	4	8	0.67	3437×1938	0	4	8	2463×1928	16
20×16	4	7092×2516	0	4	7	0.67	4909×2514	0	4	8	3526×2511	0

* Smallest systems obtained among the choices $p = 0.67, 0.75, 0.80$ are reported.
† Fixed probability $p = 0.50$ is used.

5 Experimental Results and Discussion

We have experimented with XL_SGE-2, the modified version of XL_SGE, on small random sparse multivariate quadratic systems, and have found that the heuristic substantially improves the performance of the XL algorithm in terms of the size of the final linearized system. Indeed, XL_SGE-2 performs consistently much better than XL_SGE too. Even in those cases where XL_SGE fails to improve upon XL, our modified algorithm XL_SGE-2 produces positive results. In fact, XL_SGE-2 has been found to significantly improve XL in almost all the experiments we have conducted. However, the column-weight two reduction strategy works well only in a limited set of experiments. In general, this strategy results in massive reductions in the system size, leading to failure in generating the final solvable system.

The results obtained for some small random systems with XL_SGE-2 are shown in Table 1. Here, random monomial multiplication is used since Step 1 of Algorithm 4. Each row in the central columns of the table shows the smallest system size obtained, where the minimum is taken over the three choices $0.67, 0.75$ and 0.80 of the monomial-multiplication probability p. In the table, δ represents the rank deficit of the final system. In most of the cases, we get the smallest system for $p = 0.67$. However, the dependence of the final output on p or on the structure of the initial system is not yet fully understood.

The results obtained for $p = 0.50$ are shown in the last four columns of Table 1. In the case of $p = 0.50$, we usually get much smaller final systems than produced by larger probabilities (like 0.67). But at the same time, $p = 0.50$ yields final systems with some positive (albeit small) rank deficits, whereas the larger values of p leave no rank deficits in identical settings (like same values of D). These experiments demonstrate the expected tradeoff between system reduction and rank profile. If we use partial monomial multiplication since Step 1 of Algorithm 4, $p = 0.67$ appears to be the experimentally recommended choice.

The size of the final system also depends on the random seed chosen during the execution of the algorithm. Different seeds correspond to different (random) choices of monomial multiplication. It has been observed that for the same initial system and the same p, different final systems can be obtained using different seeds. The variations of the final system size on different seeds are shown in Table 2 . Both the parts in the table correspond to the same initial system of size 16×10. For this system, XL gives a final solvable system of size 1229×385 for $D = 4$. Table 2(a) shows the results obtained by XL_SGE-2 for $p = 0.67$, and Table 2(b) shows the results obtained by XL_SGE-2 for $p = 0.50$. For $p = 0.67$, the variation in the number of equations in the final system is observed to be within 50% of one another, whereas for $p = 0.50$, this variation is within 20% of one another. In all cases, however, we obtain noticeably smaller systems compared to XL. The rank deficit exhibits the same essential behavior as in Table 1, but the choice of monomial multiplications has some effect on this rank deficit for $p = 0.50$. It, however, appears assuring that the variation of the performance of XL_SGE-2 on the seed is not annoyingly large, that is, our strategy of random monomial multiplication is experimentally validated to exhibit good performance in general.

Table 3 shows the results obtained by using random monomial multiplication only in Step 2 of Algorithm 4. In Step 1, all the multiplications are carried out. Here, we have experimented with four different values $0.50, 0.67, 0.75$ and 0.80 of p. The Table shows the smallest system size obtained among these four choices of p. XL_SGE-2 works very well in most (more than 90%) of the cases. Even in cases where XL works better than XL_SGE, the modified XL_SGE-2 outperforms XL. In a few (less than 10%) experiments, however, XL_SGE-2 exhibits poorer behavior than XL in terms of the rank profile of the expanded systems.

The performance of XL_SGE-2 is compared with the performance of XL and XL_SGE in Table 4. In the columns under XL_SGE-2 in this table, we have reported the best results obtained by XL_SGE-2. Here, the best is obtained among several choices of p, including the fixed values 0.50, 0.67, 0.75 and 0.80, and also including the variable probability sequences $p_1 = 1 - \frac{1}{d+1}$ and $p_2 = \frac{D-d}{D-d+1}$. If partial monomial multiplication is used after the $d = 1$ layer (that is, all multi-

Table 2. Variation of the final system size in XL_SGE-2 for some seed values

Final system size	D	K	Seed	δ
674×355	4	0	11056	0
747×361	4	0	5356	0
964×383	4	5	9065	0
966×385	4	6	8517	0
968×385	4	5	3438	0
983×384	4	6	5120	0

(a) $p = 0.67$

Final system size	D	K	Seed	δ
518×373	4	6	8252	4
518×378	4	6	8637	9
536×375	4	7	11395	5
543×383	4	6	7581	3
583×384	4	7	4067	3
619×385	4	7	2596	1

(b) $p = 0.50$

Table 3. Performance of XL_SGE-2
(Random monomial multiplication not done in Step 1 of Algorithm 4)

Initial system size	XL			XL_SGE-2				
	D	System Size	δ	D	K	p	System Size	δ
12×7	4	317×98	0	3	3	0.50	29×29	1
13×8	3	275×92	0	3	0	0.50	205×92	0
13×9	3	273×129	0	3	4	0.50	178×124	0
14×10	3	322×175	1	3	6	0.67	266×172	1
15×10	3	388×175	0	3	0	0.50	270×175	0
16×10	3	309×175	0	3	5	0.67	238×173	0
16×10	4	1229×385	0	4	0	0.67	674×355	0
16×11	5	4450×1023	0	5	6	0.67	3324×870	0
16×12	5	5232×1585	0	5	7	0.67	4221×1584	0
17×12	3	548×298	0	3	7	0.67	466×298	0
17×12	4	1758×793	0	4	7	0.67	1391×792	0
17×12	4	1984×792	0	4	7	0.67	1573×790	0
17×12	4	2074×793	0	4	8	0.50	1410×791	0
17×13	5	8889×2379	0	5	6	0.50	6069×2378	0
18×12	3	628×298	0	3	0	0.50	452×295	0
18×13	4	2796×1092	0	4	6	0.50	1714×1092	0
18×14	4	3333×1470	0	4	8	0.50	2074×1467	0
19×13	4	2280×1090	0	4	8	0.67	1816×1086	0
19×14	4	4154×1470	0	4	7	0.50	2775×1469	0
19×14	4	4473×1470	0	4	5	0.50	3099×1469	0
19×14	4	4500×1470	0	4	5	0.50	3155×1464	0
19×15	3	1247×575	1	3	0	0.67	1068×575	1
20×14	4	3212×1470	0	4	8	0.50	1845×1470	0
20×15	4	4640×1940	0	4	8	0.50	3006×1936	0
20×16	4	7092×2516	0	4	0	0.50	5002×2514	0
21×16	4	5513×2516	0	4	8	0.80	5224×2509	11
21×17	3	1737×833	0	3	0	0.50	1274×833	0

Table 4. Comparison of performances of XL, XL_SGE and XL_SGE-2

System size	XL			XL_SGE				XL_SGE-2				
	D	System Size	δ	D	K	System Size	δ	D	K	p	System Size	δ
12×7	4	317×98	0	3	0	44×38	1	3	3	$0.50_{d \geq 2}$	29×29	1
13×8	3	275×92	0	3	0	295×92	0	3	5	p_1	115×88	0
13×9	3	273×129	0	3	4	264×129	0	3	4	$0.50_{d \geq 2}$	178×124	0
14×10	3	322×175	1	3	5	338×173	1	3	6	$0.67_{d \geq 2}$	266×172	1
15×10	3	388×175	0	3	0	378×175	0	3	4	p_2	187×171	0
16×10	3	309×175	0	3	3	276×173	0	3	5	$0.67_{d \geq 2}$	238×173	0
16×10	4	1229×385	0	4	5	1207×385	0	4	3	0.67	686×362	0
16×11	5	4450×1023	0	5	6	4127×866	0	5	7	p_2	2960×867	1
16×12	5	5232×1585	0	5	6	4565×1583	2	5	7	p_2	3947×1581	0
17×12	3	548×298	0	3	4	548×298	0	3	8	p_2	394×297	0
17×12	4	1758×793	0	4	7	1703×793	0	4	7	p_2	1160×792	0
17×12	4	1984×792	0	4	7	2262×792	0	4	7	$(p_2)_{d \geq 2}$	1474×786	0
17×12	4	2074×793	0	4	0	2596×793	0	4	7	0.75	1506×758	0
17×13	5	8889×2379	0	5	4	12152×2342	4	5	7	0.50	5383×2379	0
18×12	3	628×298	0	3	0	614×296	0	3	7	p_1	330×291	1
18×13	4	2796×1092	0	4	5	2429×1092	1	4	8	0.50	1521×1092	1
18×14	4	3333×1470	0	4	0	3310×1470	0	4	8	$0.50_{d \geq 2}$	2074×1467	0
19×13	4	2280×1090	0	4	7	2277×1089	0	4	7	p_1	1630×1075	1
19×14	4	4154×1470	0	4	4	4202×1470	0	4	7	$0.50_{d \geq 2}$	2775×1469	0
19×14	4	4473×1470	0	4	0	4149×1470	0	4	0	$(p_1)_{d \geq 2}$	2765×1469	0
19×14	4	4500×1470	0	4	0	4750×1470	0	4	5	$0.50_{d \geq 2}$	3155×1464	0
19×15	3	1247×575	1	3	0	1247×575	1	3	0	$(p_1)_{d \geq 2}$	717×575	1
20×14	4	3212×1470	0	4	5	2781×1470	0	4	8	$0.50_{d \geq 2}$	1845×1470	0
20×15	4	4640×1940	0	4	8	4779×1940	0	4	8	$0.50_{d \geq 2}$	3006×1936	0
20×16	4	7092×2516	0	4	0	7085×2516	0	4	8	0.50	3526×2511	0
21×17	3	1737×833	0	3	0	1730×833	0	3	0	p_2	875×828	0

Table 5. Performance of XL_SGE′ (XL_SGE with column-weight 2 reduction)

Initial system size	XL			XL_SGE with col-wt 2 reduction			
	D	System Size	δ	D	K	System Size	δ
15×10	4	947×385	0	4	5	854×369	0
15×11	5	3906×1022	0	4	6	1572×502	0
15×11	4	1155×559	2	4	6	1073×491	4
17×12	5	5549×1505	0	5	7	6851×1577	0
17×13	6	19349×4095	8	5	5	9630×2278	1
18×14	4	4088×1470	0	4	5	3948×1470	0
21×17	5	29702×9401	0	5	7	44234×9380	4
22×17	4	7388×3213	0	4	6	6878×3213	0

plications are done in Step 1 of Algorithm 4), a suffix $d \geq 2$ is written against the probability. The general observation is that XL_SGE performs slightly better than XL in most cases, whereas XL_SGE-2 performs considerably better than the other two algorithms consistently in almost all cases.

Some positive results obtained by XL_SGE with column-weight two reduction (henceforth referred to as XL_SGE′) are shown in Table 5. In some cases, XL_SGE′ works very well. In Row 2 of Table 5, XL_SGE′ shows 60% reduction in the number of equations and 51% reduction in the number of variables, in comparison with XL. For the same initial system, XL produces the final solvable system for $D = 5$, whereas XL_SGE′ gives the solvable system for $D = 4$. In Row 5 of Table 5, XL_SGE′ shows 50% reduction in the number of equations and 44% reduction in the number of variables, in comparison with XL. More interestingly, XL produces a system for $D = 6$ with rank deficit 8, whereas XL_SGE′ produces the final system for $D = 5$ with rank deficit 1 only. However, there are many instances (not shown in Table 5) where XL_SGE′ does not work better than XL. In fact, in some of these cases, XL_SGE′ is even poorer than XL_SGE.

We have not studied XL_SGE-2 with column-weight two reduction (XL_SGE-2′). It is perhaps not the case that XL_SGE-2 is incompatible with the column-weight two reduction strategy. However, the effectiveness of column-weight two reduction is expected to show up for relatively large values of d. On the contrary, XL_SGE-2 demonstrates superior performance compared to XL and XL_SGE even for small values of d. Since our experiments are typically restricted to the upper bound $D \leq 5$ of the degree of $\mathbb{A}'$, our experiments miss the opportunity to study XL_SGE-2′ in a proper setting.

Table 6. Performances of XL and variants of XL_SGE for four-round baby-Rijndael ($D = 3$). The initial system size is 890×208.

Algorithm	K	p	Final System Size	Rank Deficit
XL	0	1	2594060×1498713	96936
XL_SGE	3	1	2571848×1476481	93172
XL_SGE-2	0	$0.75_{d\geq 2}$	2276971×1442363	89387
XL_SGE′	0	1	2556116×1449153	81576

As an example of more cryptographic flavor than random systems, we have experimented with several versions of XL_SGE on the initial system obtained from four-round baby-Rijndael [17]. We have found that XL_SGE-2 significantly improves the performance of XL and XL_SGE, both in terms of the size and the rank deficit of the final system. The results obtained for four-round baby-Rijndael are shown in Table 6. In the table, XL_SGE′ stands for XL_SGE with column-weight two reduction strategy. XL_SGE′ too has been found to show better performance than XL and XL_SGE.

6 Conclusion

Controlling the sizes of the linearized systems in the XL algorithm has been studied by various researchers. The chief technical contribution of this paper is our efforts to improve upon the XL family of algebraic attacks. More precisely, we suggest variants of a recent proposal called XL_SGE. Our experiments establish the effectiveness of using our modifications in tandem with XL_SGE. Like all other variants of the XL family, our proposals too are not feasible in cryptanalyzing most practical ciphers (like AES). We, however, believe that the improvements we have already achieved are worth reporting to the research community.

While our proposals address some of the open problems associated with XL_SGE, some other issues continue to remain unattended. Moreover, our study opens up some new avenues for research, some of which are stated below.

- The domains of applicability of XL_SGE′ need to be experimentally and/or theoretically determined. Modifications of XL_SGE′ may also be called for to make the column-weight two reduction strategy useful in a variety of situations.
- We have demonstrated how partial monomial multiplication may improve the performance of XL and XL_SGE. So far, XL_SGE-2 uses only random monomial multiplication. A more intelligent partial-multiplication strategy may exploit the structures of the intermediate linearized systems better than a random strategy can. Moreover, the dependence of the system size and rank profile on the seed (multiplication decisions)—a property inevitably associated with a randomized algorithm like XL_SGE-2—should better be minimized, if not eliminated altogether.
- An optimal choice for p (fixed or varying with degree d) requires more experimentation and theoretical analysis.

References

1. Faugère, J.C.: A new efficient algorithm for computing Gröbner basis (F_4) (2000)
2. Faugère, J.C.: A new efficient algorithm for computing Gröbner basis without reduction to zero (F_5). ISSAC '02 (2002) 75–83
3. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE public key cryptosystem by relinearization. In: CRYPTO. (1999) 19–30
4. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In: EUROCRYPT. (2000) 392–407
5. Courtois, N., Pieprzyk, J.: Cryptanalysis of block ciphers with overdefined systems of equations. In: ASIACRYPT. (2002) 267–287
6. Ding, J., Buchmann, J., Mohamed, M., Mohamed, W., Weinmann, R.: MutantXL. In: SCC. (2008) 16–22
7. Bard, G., Courtois, N., Jefferson, C.: Solution of sparse polynomial systems over GF(2) via sat-solvers. In: ECRYPT workshop Tools for Cryptanalysis. (2007)

8. Ars, G., Faugère, J.C., Imai, H., Kawazoe, M., Sugita, M.: Comparison between XL and Gröbner basis algorithms. In: ASIACRYPT. (2004) 338–353
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. Courtois, N.T.: Algebraic complexity reduction and cryptanalysis of GOST. Cryptology ePrint Archive, Report 2011/626 (2011) `http://eprint.iacr.org/`.
13. Daemen, J., Rijmen, V.: Rijndael for AES. In: AES Candidate Conference. (2000) 343–348
14. Ghosh, S., Das, A.: An improvement of linearization-based algebraic attacks. In: InfoSecHiComNet. (2011) 157–167
15. LaMacchia, B., Odlyzko, A.: Solving large sparse linear systems over finite fields. In: CRYPTO. (1991) 109–133
16. Mohamed, W.S.A., Ding, J., Kleinjung, T., Bulygin, S., Buchmann, J.: PWXL: A parallel Wiedemann-XL algorithm for solving polynomial equations over GF(2). In: SCC. (2010) 89–100
17. Kleiman, E.: The XL and XSL attacks on Baby Rijndael. Master's thesis, Iowa State University, Department of Mathematics (2005)