IMPROVEMENTS OF LINEARIZATION-BASED ALGEBRAIC ATTACKS ON BLOCK CIPHERS

Satrajit Ghosh

IMPROVEMENTS OF LINEARIZATION-BASED ALGEBRAIC ATTACKS ON BLOCK CIPHERS

Thesis submitted in partial fulfillment
of the requirements for the award of the degree

of

Master of Science

by

Satrajit Ghosh

Under the supervision of

Dr. Abhijit Das

Department of Computer Science and Engineering

Indian Institute of Technology, Kharagpur

August 2012

APPROVAL OF THE VIVA-VOCE BOARD

Certified that the thesis entitled **"Improvements of Linearization-based Algebraic Attacks on Block Ciphers"** submitted by **Satrajit Ghosh** to the Indian Institute of Technology, Kharagpur, for the award of the degree of Master of Science has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Member of the DAC)   (Member of the DAC)   (Member of the DAC)

(Supervisor)

(External Examiner)   (Chairman)

Date:

CERTIFICATE

This is to certify that the thesis entitled **"Improvements of Linearization-based Algebraic Attacks on Block Ciphers"**, submitted by **Satrajit Ghosh** to the Indian Institute of Technology, Kharagpur, is a record of bona fide research work carried out by him under my supervision and guidance. The thesis in my opinion, is worthy of consideration for the award of the degree of Master of Science of the Institute. To the best of my knowledge, the results embodied in this thesis have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Dr. Abhijit Das
Associate Professor
CSE, IIT Kharagpur

Date:

DECLARATION

I certify that

(a) The work contained in the thesis is original and has been done by myself under the general supervision of my supervisors.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have followed the guidelines provided by the Institute in writing the thesis.

(d) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(e) Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

(f) Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Satrajit Ghosh

ACKNOWLEDGEMENTS

If I start to thank all the numerous people who have been there for help and support whenever I needed it in the course of this project, the acknowledgement section might surpass all the remaining sections of this thesis. But anyway, let me take this opportunity to mention some of them who have really been instrumental in making this thesis possible.

To start with, I would like to express my appreciation to my advisor, Dr. Abhijit Das, for his guidance and encouragement. I offer my sincere gratitude to my supervisor for giving me such a challenging and interesting field to work on. His intuitive and innovative ideas, guidance, support and inspiration always encouraged me, and will continue to drive me, to explore deeper. I am grateful to the members of my Departmental Academic Committee (DAC) for their useful suggestions and insightful comments about my work at various occasions. I would like to convey my special thanks to Dr. Dipanwita Roy Chowdhury and Dr. Debdeep Mukhopadhyay, for their inspiration and encouragement. I would also like to thank all the staff and other faculty members of the Computer Science and Engineering Department who have been helpful in many ways.

Next, I would like to thank all my friends and colleagues to make this journey memorable. Specifically, I want to express my sincere thanks to Anup, Binu, Souvik, Karati, Dhiman, Soumik, Bhombol, Somnath, Pratyay, Sagnik, Joy Da, Sudakshina, Suprobhat, Jyotirmoy Da, Sudip Da, Utsab, Sourav, Angshuman, Athar, Projit and all the mess members in VSRC, to make my KGP life a story worth remembering. I want to thank my very special friend Gunja to be there by my side all the time.

Finally, I wish to express my indebtedness to my family for their support, sacrifice and understanding.

Satrajit Ghosh

ABSTRACT

Algebraic attacks are studied as a potential cryptanalytic procedure for several cryptographic primitives. 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 systems of equations arising from ciphers are typically multivariate polynomial equations over finite fields (usually, $GF(2)$). Algebraic techniques have been practically applied for solving systems available from some block ciphers, stream ciphers and public-key cryptosystems. However, the general complexity of algebraic attacks is poor—indeed poorer than exhaustive key search.

In 2000, eXtended Linearization (XL) was introduced as a tool for solving systems of multivariate polynomial equations. The standard XL algorithm expands the initial system of equations by monomial multiplications. The expanded system is treated as a linear system in the monomials. Linear-algebra techniques are then used to solve for the monomials, and in particular, the unknown variables standing for the key. However, for most block ciphers (including the Advanced Encryption Standard (AES)), the monomial-multiplication phase yields linearized systems, solving which demands more effort than exhaustive key search.

In this thesis, we first propose a heuristic strategy XL_SGE to reduce the count of linearized equations in the expanded system. This reduction is achieved by decomposing the expansion stage of XL into a sequence of variable-multiplication stages, and applying structured Gaussian elimination (SGE) before each stage of variable multiplication. This first proposal XL_SGE suffers from some drawbacks that impair the effectiveness of SGE-based reduction at all multiplication stages except the first. In order to avoid this problem, we propose three improved variants of XL_SGE. XL_SGE-2 uses a partial monomial-multiplication strategy to curb the generation of linearized equations in a random (but controlled) fashion. We also handle a variant of SGE in which columns of weight two are eliminated without increasing the weight of the coefficient matrix (we call this variant XL_SGE′). In our third modification XL_SGE-3, we use intelligent strategies to identify and remove many redundant equations before each variable-multiplication stage. In short, the key contribution of this thesis is the proposal of using SGE during the expansion phase of XL. (The subsequent linear-algebra phase is naturally expected to exploit SGE, anyway.)

All of our modified algorithms have been experimentally verified to be substantially superior to XL_SGE. Experimentation on small random systems indicates that the proposed XL_SGE family has the potential to significantly improve upon the performance of XL in terms of the size of the final solvable system. We also experimented with a toy version of AES, and noticed significant performance gains achieved by XL_SGE variants over XL. A theoretical analysis of the superiority of XL_SGE over XL continues to remain an open area of research.

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

Contents

List of Tables

Chapter 1

Introduction

The security of many cryptosystems is based on the difficulty of solving large systems of nonlinear multivariate polynomial equations. In algebraic cryptanalysis, we 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, and then solve that system for the key bits from some known plaintext/ciphertext pairs. In this work we study linearization based algebraic attacks on block ciphers and we have proposed some improved variants of linearization based attack techniques.

In Section 1.1, the research problems along with their motivations are briefly described. The main contributions of our work are summarized in Section 1.2. The organization of the thesis is provided in Section 1.3.

1.1 Overview and Motivation

In principle, any cryptosystem can be modeled as a set of algebraic equations over a finite field [1]. In fact the same cryptographic primitive can be expressed as several algebraic systems. According to Claude Shannon,“breaking a good cipher should require as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type” [2]. But in general, solving such systems over finite fields is an NP-Complete problem. However, when the multivariate system is overdefined, some reasonable algorithms are known, like Gröbner-basis computation [3, 4], linearization [5, 6, 7, 8], and algorithms based on SAT-solvers [9].

Many cryptosystems have been successfully cryptanalyzed by algebraic attacks [10, 11, 12, 13, 14]. But in general, time and memory complexity of these attacks are prohibitively high in case of block ciphers. Although algebraic attacks against block cipher have received much attention since the proposal in [7, 15] against Advanced Encryption Standard (AES) and Serpent, so far it has limited success against modern block ciphers. 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, Courtois and Pieprzyk [7] estimate an effort of nearly 2^{230} bit operations to cryptanalyze 128-bit AES using XSL, which exceeds the complexity of brute-force search (at most 2^{128} encryptions). Consequently, it is of important research concern to propose practical improvements of known algorithms on algebraic attacks. 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.

1.2 Contributions

To start with, we propose a new heuristic XL_SGE to improve the performance of the XL method [6] by reducing the size of the final linearized system. XL_SGE uses structured Gaussian elimination (SGE) [16] 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 (*avalanche effect*). XL_SGE gets rid of this problem by tuning a heuristic parameter. 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.

In many cases, the reduction in the system size obtained by XL_SGE is not significant. We identify sources of inefficiency of XL_SGE, and propose some improvements to repair this problem. These improvements employ three 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. Finally, redundant equations are deleted in a controlled manner. Experiments carried out on small systems and toy ciphers indicate that our new heuristic ideas bring down the complexity of XL substantially.

1.3 Thesis Organization

The rest of the dissertation is organized as follows.

- **Chapter** 2 provides the necessary background materials along with related research works on algebraic attack.
- **Chapter** 3 describes our first proposal extended linearization with structured Gaussian elimination (XL_SGE).
- **Chapter** 4 identifies some problems in XL_SGE and proposes improved variants of XL_SGE, namely XL_SGE-2 and XL_SGE$'$.
- **Chapter** 5 discusses another improved variant of XL_SGE (XL_SGE-3), which uses intelligent row deletion strategy along with XL_SGE.
- **Chapter** 6 summarizes the work done, and concludes the thesis after mentioning some possible extensions of our work.

Chapter 2

Literature Survey

Algebraic attacks are very important techniques in the analysis of symmetric primitives. These are very powerful methods that apply to both block ciphers and stream ciphers (also in case of hash functions). The basic principle of algebraic cryptanalysis is to express the whole cryptographic algorithm as a large system of multivariate algebraic equations (typically over $GF(2)$), which can be solved to recover the secret key. In this chapter, we are going to give an overall idea of algebraic attack. In Section 2.1, we give a description about algebraic attack on AES like block ciphers. That includes a description of equation generation for a toy version of AES and a brief description of extended linearization (XL) algorithm. It also includes an overview of structured Gaussian elimination algorithm, which is important in order to understand our proposals in the next chapters. In Section 2.2, we provide a brief survey on algebraic attacks.

2.1 Algebraic Attack on AES-like Ciphers

Block ciphers are an important building block of modern cryptography. In August 2000, the block cipher Rijndael was selected as the Advanced Encryption Standard (AES) [17]. Rijndael is a key-iterated block cipher with a very strong algebraic structure. AES can be represented as algebraically closed equations over $GF(2^8)$ [7]. It can also be represented as a system of multivariate quadratic equations over $GF(2)$ with plain-text, cipher-text and key bits as variables. The system of equations can be solved for the key-bit values, given a few known plain-text/cipher-text pairs under a specific key.

The MQ problem is the problem of solving multivariate quadratic equations. The MQ problem is NP-Hard in general [18]. Solving a system of quadratic equations over any *finite* field is NP-Complete [19] (since over a finite field, one can verify a correct solution in polynomial time). In general, no polynomial-time algorithm is known to solve this problem. However, for overdefined systems of multivariate quadratic equations (Number of equations $\gg$ Number of variables), some reasonable algorithms are known [5, 6, 8, 20, 3, 4].

An algebraic attack consists of two basic steps:

- Equation Generation
- Solve the system of equations

2.1.1 Equation Generation

Every cryptosystem can be represented as a system of polynomial equations over $GF(2)$ with plain-text, cipher-text and key bits as variables. In general, a block cipher consists of a linear part and a non-linear part. The non-linear 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 non-linear part of the cipher, we have used two different techniques. First, we use the structure of the S-Boxes to generate equations. Second, we use 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 [18].

Baby Rijndael Structure

Baby Rijndael is a scaled down version of AES with the same algebraic structure as AES. The block size and the key size of baby Rijndael are 16 bits. Each block and key can be represented by 4 hexadecimal digits as $h_0h_1h_2h_3$ and $k_0k_1k_2k_3$. Baby Rijndael consists of several rounds with identical structure.

The state is considered to be a 2×2 array of hex digits. For the mix-column operation, the state is considered to be an 8×2 array of bits. For converting between the two, each hex digit is considered to be a column of 4 bits with the high-order bit at the top. Initially, the input block $h_0h_1h_2h_3$ is loaded into the state $\begin{pmatrix} h_0 & h_2 \\ h_1 & h_3 \end{pmatrix}$, and the round keys are computed according to the key-schedule algorithm. The encryption function of the 4-round baby Rijndael can be described as:

$$E(\mathbf{a}) = r_4 \circ r_3 \circ r_2 \circ r_1(\mathbf{a} \oplus \mathbf{k_0}).$$

Here, $\mathbf{a}$ denotes the state, $\mathbf{k_0}, \mathbf{k_1}, \mathbf{k_2}, \mathbf{k_3}, \mathbf{k_4}$ are the round keys, and

$$r_i(\mathbf{a}) = \mathbf{t}(\hat{\sigma}(S(\mathbf{a}))) \oplus \mathbf{k_i},$$

In r_4, the mix-column operation $\mathbf{t}$ is omitted. Individual functions of the cipher are described below.

- SubBytes: The S operation is a table lookup applied to each hex digit of the state:
$$\begin{pmatrix} h_0 & h_2 \\ h_1 & h_3 \end{pmatrix} \xmapsto{\text{S}} \begin{pmatrix} s(h_0) & s(h_2) \\ s(h_1) & s(h_3) \end{pmatrix},$$
where the s function is given in Table 2.1.

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$s(x)$	a	4	3	b	8	e	2	c	5	7	6	f	0	1	9	d

Table 2.1: S-Box table lookup

- ShiftRows: The $\hat{\sigma}$ operation swaps the entries in the second row of the state.
$$\begin{pmatrix} h_0 & h_2 \\ h_1 & h_3 \end{pmatrix} \xmapsto{\hat{\sigma}} \begin{pmatrix} h_0 & h_2 \\ h_3 & h_1 \end{pmatrix}$$

- MixColumns: The matrix $\mathbf{T}$ is an 8×8 matrix of bits shown below. For the mix-column transformation, the state is considered to be an 8×2 matrix of bits. The state is multiplied by $\mathbf{T}$ on the left using matrix multiplication modulo 2.
$$\mathbf{a} \xmapsto{\mathbf{t}} \mathbf{T} \cdot \mathbf{a}$$

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

- KeySchedule: At the beginning of the cipher and at the end of each round, the state is bit-wise added (mod 2) to the round key. The round keys are 2×2 arrays of hex digits similar to the state. The columns of the round keys are defined recursively as follows:

$$\omega_0 = \begin{pmatrix} k_0 \\ k_1 \end{pmatrix}, \ \omega_1 = \begin{pmatrix} k_2 \\ k_3 \end{pmatrix},$$
$$\omega_{2i} = \omega_{2i-2} \oplus S(reverse(\omega_{2i-1})) \oplus y_i,$$
$$\omega_{2i+1} = \omega_{2i-1} \oplus \omega_{2i},$$

for $i = 1, 2, 3, 4$ and $y_i = \begin{pmatrix} 2^{i-1} \\ 0 \end{pmatrix}$. The *reverse* function interchanges the two entries of a column. For $i = 0, 1, 2, 3, 4$, the round key k_i is the matrix whose columns are ω_{2i} and ω_{2i+1}.

Equation Generation from the Structure of Baby Rijndael

We have generated linear equations from the linear layer of baby Rijndael and quadratic equations from the structure of the baby Rijndael S-Box.

- The S-Boxes: Baby Rijndael S-Box (like AES S-Box) consists of two main layers.

  1. Multiplicative inverse
     In case of baby Rijndael, the input of the S-Box over $GF(2^4)$ is inverted modulo $m(x)$, where $m(x)$ is an irreducible polynomial

defining $GF(2^4)$. In our case, $m(x) = x^4 + x + 1$. In case of AES, the input is over $GF(2^8)$, and $m(x) = x^8 + x^4 + x^3 + x + 1$.

2. Affine Transformation

   The inverted input passes through an affine transformation. In case of baby Rijndael, the affine transformation is defined as $s(x) = b(x)g(x) + c(x)$, where $g(x)$ is the inverse of the input of the S-Box, $b(x) = x^3 + x^2 + x$, and $c(x) = x^3 + x$. The result is computed modulo $x^4 + 1$. The affine transformation can be represented in the form

$$\begin{pmatrix} s_3 \\ s_2 \\ s_1 \\ s_0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} g_3 \\ g_2 \\ g_1 \\ g_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix},$$

   where g_i (for $i = 0, 1, 2, 3$) are the output of the multiplicative-inverse layer, and s_i are the output of the S-Box.

Assume that the input bit variables of an S-Box of baby Rijndael are a_0, a_1, a_2, a_3, and the output bit variables after the inverse layer of the S-Box are b_0, b_1, b_2, b_3. Let f_1 and f_2 be the polynomial representations of the input and the output of the inverse layer of the S-Box. From the property of multiplicative inverse,

$$f_1 f_2 - 1 = 0, \tag{2.1}$$

where

$$f_1 = a_0 + a_1 x + a_2 x^2 + a_3 x^3, a_i \in GF(2), 0 \leq i \leq 3,$$
$$f_2 = b_0 + b_1 x + b_2 x^2 + b_3 x^3, b_i \in GF(2), 0 \leq i \leq 3.$$

Each multiplication is modulo $m(x) = x^4+x+1$, and additions are bit-wise over $GF(2)$. Comparing coefficients of both sides of equation (2.1), we get 4 quadratic equations over $GF(2)$:

$$b_3a_0 + b_2a_1 + b_1a_2 + b_0a_3 + b_3a_3 = 0,$$
$$b_2a_0 + b_1a_1 + b_0a_2 + b_3a_2 + b_2a_3 + b_3a_3 = 0,$$
$$b_1a_0 + b_0a_1 + b_3a_1 + b_2a_2 + b_3a_2 + b_1a_3 + b_2a_3 = 0,$$
$$b_0a_0 + b_3a_1 + b_2a_2 + b_1a_3 + 1 = 0.$$

There is a problem with the last equation, as 0^{-1} is defined as 0. In that case, $f_1 = 0$ and $f_2 = 0$, so $f_1 f_2 = 0$. Even then, we can use the first three equations. So by comparing the coefficient of equation (2.1), we can have three consistent quadratic equations for each S-Box.

More quadratic equations per S-Box can be extracted from the following relations [21]:

$$f_1^2 f_2 - f_1 = 0, \tag{2.2}$$

$$f_1 f_2^2 - f_2 = 0. \tag{2.3}$$

Equations (2.2) and (2.3) hold even in the case of 0^{-1}. So 8 quadratic equations can be found using these equations.

- The linear layer
  Assume that the output bit variables of the non-linear layer (S-Boxes) are $z_0, z_1, \ldots, z_{15}$, the round key variables are $k_0, k_1, \ldots, k_{15}$, and the input bit variables of the next round are $y_0, y_1, \ldots, y_{15}$. The output of the non-linear layer is going through a linear layer consisting of shift-row and mix-column operations. Let that layer be L. So we can extract the linear equations of the form

  $$y_i = L_i(z_0, z_1, \ldots, z_{15}) \oplus k_i, 0 \leq i \leq 15,$$

  where L_i denotes the i-th output bit line after the linear layer of the cipher.

- The Key Schedule
  Equations for the round keys can be generated from the key-schedule algorithm. From each round of baby Rijndael, we can construct 22 quadratic equations from the structure of the S-Box.

Null-space Equations

Assume that $X = (x_3, x_2, x_1, x_0)$ is the input to an S-Box, and $Y = (y_3, y_2, y_1, y_0)$ is the output of that S-Box. To find the null-space equations, we build a 16×37 matrix. Each row contains the values of the 37 monomials

$$1, x_3, \ldots, x_0, y_3, \ldots, y_0, x_3x_2, x_3x_1, \ldots, x_1x_0, x_3y_3, x_3y_2, \ldots, x_0y_0, y_3y_2, y_3y_1, \ldots, y_1y_0$$

for each of the 16 possible input values of x_3, x_2, x_1, x_0. We find the reduced row echelon form of the matrix by Gaussian elimination, and then write all the variables in terms of the free variables. From the null space, one gets 21 linearly independent quadratic equations for each S-Box of baby Rijndael.

2.1.2 Solve the System of Equations

Solving the overdefined multivariate quadratic system of equations consists of two parts. First, one expands the initial system of equations to generate more linearly independent equations in order to get a system with full column rank or close to full column rank. Second, the expanded system is solved by linear system solvers.

The usual method to solve systems of algebraic equations is to use algorithms based on Gröbner-basis computation. The fastest of such algorithms (like, Faugère's F_4 and F_5 [3, 4]), usually take exponential (or more) time in the size of the system, and so are practically infeasible. The XL (eXtended Linearization) algorithm have been proposed as an efficient alternative [6]. 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 $m \geq \epsilon n^2$, $0 < \epsilon \leq 1/2$.

eXtended Linearization (XL)

Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir [6] introduce a new algorithm called XL (eXtended Linearization) for solving a system of multivariate polynomial equations. The XL algorithm tries to benefit from the fact that 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. In [6] the authors provide some evidence, that XL can solve randomly generated systems of polynomial equations in subexponential time when m (number of equations) exceeds n (number of unknowns) by a number that increases slowly with n. However, now it is known that XL has an exponential complexity [22, 23] in general.

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.

Algorithm 1: Extended Linearization (XL) of multivariate systems

1. **Multiply:** Generate the new system $\mathbb{B} = \bigcup_{0 \leq k \leq D-d_{max}} X^k \mathbb{A}$, where X^k stands for the set of all monomials of degree k, and d_{max} is the maximum degree of the initial system.
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 $\mathbb{B}$. 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. Find the roots of this equation in the underlying finite field.
4. **Repeat:** Simplify the equations, and repeat the process to solve for the other variables.

Structured Gaussian Elimination (SGE)

Structured Gaussian Elimination (SGE) is an algorithm used to reduce the dimension of a sparse matrix by eliminating some of its rows and columns [24, 16]. SGE exploits the special structure of the matrices arising from integer-factorization and discrete-logarithm algorithms. The basic idea of SGE is to declare some columns as heavy-weight, and to work only on preserving the sparsity of the remaining light-weight columns. It is a heuristic procedure that tends to preserve the sparsity of the light columns. The weight of a row or column of a matrix is the number of non-zero entries in that row or column. Initially we can identify a column as heavy-weight, if the weight of that column is more than αm, for a predetermined small positive fraction α and total number of rows m. According to [16] the best performance of SGE can be achieved when the linear system is sparse and most importantly there should be considerably more equations than unknowns. The algorithm consists of sequence of steps as follow.

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.

2.2 A Survey on Algebraic Attacks

In general, algebraic cryptanalysis of symmetric primitives is equivalent to MQ problem. Although MQ problem is NP-Hard in nature, for overdefined MQ some reasonable methods are known. They can be broadly classified into the following categories.

- Algorithms based on Gröbner basis computations: These include F_4 and F_5 algorithms [3, 4].
- Algorithms based on linearization: Kipnis and Shamir's relinearization [5], Courtois et al.'s eXtended Linearization (XL) [6], Courtois and Pieprzyk's eXtended Sparse Linearization [7] and Ding et al.'s MutantXL [8].
- Algorithms based on SAT solvers: Bard et al.'s algorithm [9]

Although superficially different, these approaches are occasionally proved to be computationally equivalent. In the following part we give a brief survey of some algorithms, that are used in algebraic cryptanalysis and also discuss some practical attacks.

- Kipnis and Shamir [5] introduce a technique called *relinearization* to solve system of multivariate polynomial equations. In the paper, the authors cryptanalyze HFE public key cryptosystem by relinearization. Relinearization is expected to run in polynomial time for a system of ϵm^2 quadratic equations in m variables, for any constant $\epsilon > 0$.

- In Eurocrypt 2000, Courtois et al. [6] show that many of the equations generated by relinearization are linearly dependent. They propose a new variant of linearization technique, namely eXtended Linearization (XL). We have already discussed XL algorithm in 2.1.2. In the same paper the authors propose a variant of XL called FXL (stands for Fixing and XL). They observe that over a large field the smallest working degree D (parameter of the XL algorithm) decreases dramatically when $m - n$ increases. So, in FXL they first fix μ variables, and then solve the resulting system of m equations with $n - \mu$ variables using XL. The authors expect the FXL algorithm to be sub-exponential, even when $m = n$. All the simulations of XL have been done over $GF(127)$ and with $D < 127$. The authors give an approximate estimation of D, assuming most of the equations are linearly independent in XL, as $D \geq n/\sqrt{m}$.

- Courtois and Patarin [15] study the behavior of XL for system of quadratic equations over $GF(2)$. They have showed that use of the field equations of $GF(2)$ improves the performance of XL algorithm. In the same paper, the authors introduce two variants of XL, namely XL$'$ and XL2. They give an explanation for the linear dependencies that appear in the XL algorithm, and derive a formula for the number of linearly independent equations in XL or XL2. According to the paper, it is not clear whether XL is asymptotically sub-exponential.

- Gröbner basis algorithms are one of the most popular and efficient methods for solving polynomial system of equations. In 1965, Buchberger introduces the notion of a Gröbner basis and also a criterion to test whether a set of polynomials is a Gröbner basis. This criterion naturally leads to Buchbergers algorithm for computing a Gröbner basis from a given ideal basis. Worst case time complexity of this algorithm is doubly exponential to the input size. Jean Charles Faugère proposes F_4, F_5 [3, 4] algorithms as efficient alternatives of Buchberger algorithm. The worst case time complexity is exponential in these alternatives. Many cryptosystems have been successfully cryptanalyzed using these Gröbner basis algorithms [12, 13, 14].

- Courtois and Pieprzyk [7] introduce a variant of XL technique, eXtended Sparse Linearization (XSL), for solving sparse system of multivariate equations. The XSL method is based on the XL algorithm, but attempts to use the sparsity and specific structure of the equations. Instead of multiplying the equations by all monomials of degree $\leq D-2$ (supposing that the original equations were quadratic), in the XSL algorithm the equations are multiplied only by "carefully selected monomials" [7]. This has the intention to create less new terms when generating the new equations. According to [7], XSL seems to break Rijndael (AES) 256 bits and Serpent for key lengths 192 and 256 bits. The analysis of XSL is not straight forward, as the description of the algorithm leaves some room for interpretation. However according to [25, 26], in the form presented in [7], XSL cannot solve the system arising from the AES.

- Research has shown that XL and Gröbner basis algorithms are closely related. In fact in [27], it is shown that the XL algorithm can be viewed as a redundant version of F_4 algorithm. For a system of quadratic polynomials, when the number of equations is $m = n + c$ (where, n is the number of unknowns and c is a constant), then the minimum degree for XL to succeed for a generic system is

  $$D \geq n/(\sqrt{c-1}+1).$$

  Since the number of monomials is $\binom{n}{D}$ in the binary case, and $\binom{n+D}{D}$ in the general case, this theorem implies that XL has an exponential complexity [22, 23].

- Bard et al. [28] demonstrate the method of conversion of solving system of equations over $GF(2)$ into a CNF-SAT problem, and show the technique to solve the problem using SAT-solver. They further claim that solving such a CNF-SAT problem on a SAT-solver is faster than brute force for sparse cases.

- Courtois and Bard [29] cryptanalyze 6 rounds of DES using only one single known plaintext. They use two different techniques for solving the system of equations. First they use an elimination algorithm ElimLin, which can be seen as a very simplified version of known Gröbner basis algorithms. In the second case, they simply convert the equations from ANF to CNF, and solve the corresponding SAT problem using SAT-solver MiniSat 2.0. Using the ElimLin algorithm the authors successfully cryptanalyze DES upto 5 round, using 3 known plaintexts and 23 variables fixed, in 173 seconds. Brute force attack for the same would require 540 seconds. Using ANF to CNF converter and SAT-solver MiniSat 2.0, the authors mount a successful attack on 6 rounds of DES by fixing 20 key variables in 68 seconds, while the exhaustive search would take about 4000 seconds. The authors try to solve the same system of equations using MAGMA and Singular. Both the tools crash with "out of memory" message after allocating nearly 2 Gbytes of memory. Whereas, the memory usage reported by MiniSat is 9 Mbytes.

- In [30], Courtois et al. present a slide-algebraic attack (on KeeLoq block cipher), that use a SAT-solver, with the complexity equivalent to about 2^{53} KeeLoq encryptions (with 2^{16} known plaintexts). Using SAT-solver MiniSat, without guessing any key variables, the key is computed in about 2 seconds.

- Courtois et al. [31] describe a full key recovery attack on Hitag2 stream cipher. Hitag2 is a stream cipher widely used in RFID car locks in the automobile industry. The algebraic immunity of the Hitag2 stream cipher is very high (at least 4), which suggests that Hitag2 should be secure under algebraic attack. The authors express the cipher as system of multivariate low degree equations, convert that into a SAT problem, and use SAT-solver MiniSat 2.0 to solve the system in feasible time.

- Ding et al. [8] use the concept of mutants to speed up the XL algorithm for solving system of multivariate equations over finite fields. In this paper, the authors propose the MutantXL algorithm. The authors compare the performance of MutantXL with XL for HFE systems. They conclude that, the MutantXL algorithm can indeed outperform the XL algorithm and can solve multivariate systems at a lower degree than the usual XL algorithm.

The authors suggest that the use of sparse linear algebra algorithms, such as Wiedemann algorithm, can further improve the performance of MutantXL.

- Mohamed et al. [20] propose MXL2 algorithm as an improved variant of MutantXL over $GF(2)$ with significantly reduced memory usage. The paper shows some experimental results comparing MXL2 with XL, MutantXL and MAGMA's implementation of F_4. For that comparison the authors have chosen small randomly generated instances of the MQ problem and quadratic systems derived from HFE instances. The largest matrices produced by MXL2 are substantially smaller than the ones produced by MutantXL and XL. For significant number of cases, the authors observe a reduction of size of the largest matrix when they compare MXL2 against MAGMA's F_4 implementation.

- Mohamed et al. [32] describe an efficient attack on Multivariate Quadratic Quasigroups (MQQ) public key cryptosystem by solving system of multivariate quadratic polynomial equations, using a modified version of the MutantXL algorithm. The authors successfully cryptanalyze MQQ system of 160 bits using their MutantXL implementation. They compare the results with MAGMA F_4 attack to MQQ cryptosystems, which turn out to be inefficient in terms of memory.

- Erickson et al. [33] explore algebraic attacks on SMS4 and SSMS4 (a toy version of SMS4) using Gröbner basis attacks on equation systems over $GF(2)$ and $GF(2^8)$, as well as attacks using a SAT-solver derived from the $GF(2)$ model. The results indicate that the MAGMA (used for Gröbner basis attacks) is more effective than MiniSat for the full cipher, while MiniSat outperform MAGMA in the simplified version of the cipher. The authors were unable to break 6 round SMS4 in their setup.

- In [34], Albrecht et al. identify the relation between the MutantXL family of algorithms and Gröbner basis algorithms. They claim that their results map all novel concepts from the MutantXL family of algorithms to their well-known Gröbner basis equivalents. In fact they conclude that MutantXL family of algorithms can be fundamentally reduced to redundant variants of F_4.

The algorithms used for algebraic attacks are of exponential complexity. As a result, they have very limited use in the cryptanalysis of real world ciphers, given the large sizes of the system involved. The current algorithms can not handle the equation systems obtained from full scale encryption algorithms within reasonable time and resources. As a result, simplified toy versions are considered and results for those reduced variants serve as a measure for the performance of pure algebraic attack. In fact, no result is available in literature, where a modern block cipher was broken using algebraic attacks faster than with other techniques. Thus, improving algebraic attacks on block ciphers is currently a very active area of research. In the next chapters, we will discuss some techniques to improve existing linearization based algebraic attack techniques.

Chapter 3

A Heuristic Improvement of XL

In this chapter, 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 [16] 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 substantially.

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 [7] and MutantXL [8]. 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 block ciphers like 128-bit AES. Our heuristic too does not immediately 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.

The rest of the chapter is organized as follows. Section 3.1 provides a basic idea of algebraic attacks over AES-like block ciphers. In particular, the XL algorithm and structured Gaussian elimination procedure are discussed in brief. In Section 3.2, we propose our algorithm XL_SGE. Section 3.3 provides an estimation of the degree bound in case of XL_SGE. In Section 3.4, we supply our experimental results, and compare the performance of XL_SGE with that of XL. We conclude the chapter in Section 3.5.

3.1 Background

Algebraic attack on AES-like block ciphers is equivalent to solving large systems of multivariate quadratic equations over finite field (MQ problem). In general this problem is NP-Complete. However, for overdefined system of equations few reasonable methods are known. In order to mount algebraic attack on a block cipher, one has to express the cipher as a system of multivariate quadratic equations and then has to solve the system. Thus an algebraic attack consists of two basic steps: (1) Equation generation, and (2) Solving the system of equations.

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, several methods are used. For our experiments, we have used a scaled-down version of AES (baby Rijndael). Section 2.1.1, provides a detailed description of equation generation from the structure of baby Rijndael. To solve multivariate systems of equations, Gröbner-basis algorithms are usually used. F_4 and F_5, proposed by Faugère [3, 4] are fastest algorithms to compute Gröbner-basis. The XL (eXtended Linearization) algorithm is proposed as an efficient alternative [6]. We have proposed some improvements over XL algorithm using structured Gaussian elimination (SGE) algorithm. Detailed description of XL and SGE can be found in 2.1.2 of Chapter 2.

3.2 eXtended Linearization with Structured Gaussian Elimination (XL_SGE)

3.2.1 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.

3.2.2 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), a degree bound $D \in \mathbb{N}$ and a avalanche-control parameter $K \in \mathbb{N}$. The basic steps of the XL_SGE expansion procedure are as follows.

Algorithm 3: XL with Structured Gaussian Elimination (XL_SGE)

```
1. Expand the initial system $\mathbb{A}$ up to degree $d = 2$ using XL to obtain a
   linearized system $\mathbb{A}'$. Make a copy of the linearized system $\mathbb{A}'$ as $\mathbb{B}$.
2. Apply structured Gaussian elimination (SGE) on $\mathbb{A}'$ with avalanche-control
   parameter $K$ to obtain a reduced system of equations $\mathbb{A}''$ of degree $d$.
3. Multiply each equation in $\mathbb{A}''$ by each monomial of degree 1 to get a system
   $\mathbb{A}'''$ of degree $d+1$. Append the equations of $\mathbb{A}'''$ to $\mathbb{B}$. $\mathbb{B}$ now has equations
   of degrees $\leq d+1$. Rename $\mathbb{A}'''$ as $\mathbb{A}'$.
4. If the degree of the system of equations $\mathbb{B}$ is $D$, end the process. Otherwise,
   go to Step 2 with $d$ incremented by 1.
```

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.

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. Whereas 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.

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 the 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. 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.

One important thing to note is that, we do not delete equations using SGE in XL_SGE. We deselect those equations from the future monomial multiplication. In fact, the final system $\mathbb{B}$ already contains the equations removed by SGE at any intermediate stage of the algorithm. However, those equations are not used during the expansion stage. Let, $x+y+z=0$ be an equation in an intermediate system before applying SGE. The variable x has column-weight one and the other two variables have column-weight greater than one. Using Step 1 of SGE, XL_SGE deselects the equation for future monomial multiplication assuming that the solution of y and z can be found by expanding the reduced system only. Once we get the solution of y and z, we can easily plug those solutions in the equation $x+y+z=0$ (which is already in $\mathbb{B}$) to find the solution for x.

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.

3.3 An Evaluation of XL_SGE over $GF(2)$

We have experimented XL_SGE for the system of multivariate quadratic equations over $GF(2)$. While applying XL_SGE over $GF(2)$ we always use the fact ${x_i}^2 = x_i$. When we multiply the equations with monomials, we use the equation ${x_i}^2 = x_i$ to eliminate all powers of x_i. Therefore over $GF(2)$ the number of monomials of degree k is exactly $\binom{n}{k}$ [15], for a initial system with m equations and n variables.

Now, let T be the total number of monomials upto degree D, R be the total number of equations generated by XL. Then we have, $T = \sum_{\lambda=0}^{D} \binom{n}{\lambda}$ and $R = m(\sum_{\lambda=0}^{D-2} \binom{n}{\lambda})$. Let r_e number of equations and r_v number of variables are removed by XL_SGE. If μ is the proportion of linearly independent equations in XL_SGE, then $\mu = Free/R - r_e$, where $Free$ is the number of linearly independent equations. Then XL_SGE algorithm will succeed when $[R - r_e] \times \mu \geq [T - r_v]$, i.e. when,

$$[m(\sum_{\lambda=0}^{D-2} \binom{n}{\lambda}) - r_e] \times \mu \geq [\sum_{\lambda=0}^{D} \binom{n}{\lambda} - r_v].$$

Considering the main two terms of the summation:

$$\begin{aligned} &[m\binom{n+1}{D-2} - r_e] \times \mu \geq \binom{n+1}{D} - r_v \\ &m\frac{(n+1)!}{(D-2)!(n-D+3)!} \times \mu \geq \frac{(n+1)!}{D!(n-D+1)!} + (\mu r_e - r_v) \end{aligned}$$

Therefore we get:

$$m\mu \geq \frac{(n-D+3)(n-D+2)}{D(D-1)} + \frac{(\mu r_e - r_v)(D-2)!(n-D+3)!}{(n+1)!}$$

Now when $n \gg D$, then $\frac{(D-2)!(n-D+3)!}{(n+1)!} \ll 1$. We have observed that $r_e >> r_v$ in all the cases, and if $\mu \approx 1$, then $(\mu r_e - r_v)$ is a small positive quantity. With this assumption, we get an approximate evaluation of D in XL_SGE as:

$$D \gtrsim \frac{n}{\sqrt{\mu}\sqrt{m}}.$$

Which is same as D in case of XL over $GF(2)$. So, for large n value of D is expected to be same as in XL. Thus we get a speedup in case of XL_SGE, in compare to XL, mainly due to the size reduction of the final system, not due to reduction in degree of the final system. However, from this evaluation, it is impossible to say anything about the amount of size reduction in XL_SGE, which actually governs the speed up of the algorithm. Empirically we have observed that the performance of XL_SGE depends on the structure of the initial system. However, the exact nature of dependency is not very much clear yet. Also the nature of dependency of the performance of XL_SGE on the choice of the heuristic parameter K remains as an open question.

3.4 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 3.1. 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 $m \times n$ indicates m linearized equations in n monomials. For both XL and XL_SGE, we report the final system size. In most of the cases, XL_SGE produces full-rank systems. In case of full-rank systems the complexity of the attack will be $O(n^2)$. In case of a small rank deficit r, we can write the $(n - r)$ variables in terms of r variables. By guessing r variables over $GF(2)$, we can find 2^r solutions. In that case the complexity of this attack will be $2^r \times O(n^2)$.

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.

Table 3.1: Comparison of XL with XL_SGE (with $K = 0$) for random systems

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

δ: rank deficit of the final linearized system $\mathbb{B}$.

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 3.1. 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 $D = 3$ (for 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 3.1. 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.

Table 3.2 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 3.3 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.

Table 3.2: Comparison of XL with XL_SGE (with $K \geq 0$) for random systems

Size	XL			XL_SGE			
of $\mathbb{A}$	D	Size of $\mathbb{B}$	δ	D	K	Size of $\mathbb{B}$	δ
22×12	3	534×298	0	3	4	513×292	0
23×13	5	11219×2379	0	4	4	2863×1073	0
24×13	3	726×377	0	3	0	726×377	0
24×15	4	6451×1940	0	4	0	6400×1940	0
24×16	4	6587×2516	0	4	7	6311×2516	0
24×16	3	2966×696	0	3	0	1800×696	0
25×17	3	3916×833	0	3	0	2246×833	0
25×18	4	55127×4047	0	4	0	12496×4045	0
25×18	5	36825×12615	0	5	6	34027×12615	0

This is the same system reported in the last row of Table 3.2. 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.

Table 3.3: Dependence of the performance of XL_SGE on the parameter K

K	$D = 3$		$D = 4$		$D = 5$	
	System Size	δ	System Size	δ	System Size	δ
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

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 better performance than XL. The results are shown in Table 3.4.

Table 3.4: Comparison of XL with XL_SGE for baby Rijndael for $D = 3$

Number of rounds	Size of $\mathbb{A}$	XL		XL_SGE		
		Size of $\mathbb{B}$	δ	K	Size of $\mathbb{B}$	δ
1	232×64	178892×43745	0	0	142945×43745	0
2	448×112	853358×234225	48	3	634810×233633	24
3	664×160	2359598×682401	576	7	1755432×682273	576
4	890×208	2594060×1498713	96936	3	2571848×1476481	93172

We have also reduced the initial system (according to the last paragraph of Section 3.2) 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 are written in the C programming language. The PARI/GP package is used to carry out some intermediate calculations needed to generate equations. The mathematical package Sage (Version 4.4.2) is used to calculate the rank of sparse matrices available from XL and XL_SGE.

3.5 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. In order to reduce the system size in XL we propose XL_SGE. XL_SGE uses structured Gaussian elimination 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. Experimentation on small random systems and a toy version of AES indicates that XL_SGE has the potential to improve the performance of the XL in terms of the size of the final solvable system.

Chapter 4

Improvements of XL_SGE

In the previous chapter, we have proposed the XL_SGE algorithm to improve the complexity of XL attack by using structured Gaussian elimination (SGE) during the expansion phase of XL. In this chapter, 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 two improvements of XL_SGE. In the rest of the chapter, Section 4.2 identifies the weaknesses of XL_SGE. In Section 4.3, we propose two different ways of repairing these drawbacks of XL_SGE. Our experimental results and associated remarks follow in Section 4.4. The concluding Section 4.5 highlights some scopes for further research in this direction.

4.1 Introduction

The XL_SGE algorithm 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) [16] 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. But in many cases, the reduction in the system size obtained by XL_SGE is not significant. 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 modifications bring down the complexity of XL_SGE substantially.

4.2 Problem 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 SGE 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.

$$abc + bc + a = 0 \tag{4.1}$$
$$abc + bc + b + d = 1 \tag{4.2}$$
$$abc + ad + b = 1 \tag{4.3}$$
$$ad + a + d = 0 \tag{4.4}$$

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 (4.5) is generated twice (Eqn (4.1) $\times$ a and Eqn(4.4) $\times$ a), but is shown only once.

$$
\begin{aligned}
a &= 0 && (4.5)\\
ab + ad + a &= 0 && (4.6)\\
abc + ab + ad + a &= 0 && (4.7)\\
abc + ab + bc &= 0 && (4.8)\\
abc + bc + bd &= 0 && (4.9)\\
abc + abd &= 0 && (4.10)\\
abd + ab + bd &= 0 && (4.11)\\
abc + ac + bc &= 0 && (4.12)\\
abc + cd + c &= 0 && (4.13)\\
abc + acd + bc + c &= 0 && (4.14)\\
acd + ac + cd &= 0 && (4.15)\\
abcd + bcd + ad &= 0 && (4.16)\\
abcd + bcd + bd &= 0 && (4.17)\\
abcd + ad + bd + d &= 0 && (4.18)\\
d &= 0 && (4.19)
\end{aligned}
$$

The monomial abc appears thrice in the system (4.1)–(4.4). Multiplication of these equations by d generates three occurrences of $abcd$ in the expanded system (4.5)–(4.19). The monomial bd appears in the expanded system four times, twice from multiplying the term b in Eqns (4.2) and (4.3) by d, and twice from multiplying the term d in Eqns (4.2) and (4.4) by b. There is also a case of cancellation arising out of algebraic simplification. For example, ad appears twice in the system (4.1)–(4.4). Multiplying Eqn (4.3) by d leaves the term ad, but multiplying Eqn (4.4) by d removes this term. Unfortunately, however, the term ad appears from other sources too. For example, Eqn (4.1) $\times$ d and Eqn (4.2) $\times$ a give Eqns (4.16) and (4.6) respectively, each containing the non-zero term ad.

To sum up, all the monomials appearing in the expanded system (4.5)–(4.19) 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.

4.3 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 (4.1)–(4.4) to generate the system (4.5)–(4.19). The term abd appears in both Eqns (4.10) and (4.11) upon multiplication of ad in Eqns (4.3) and (4.4) by b. If we skip the second multiplication, we no longer generate Eqn (4.11), so abd occurs with column weight one. Similarly, if we avoid the multiplication of Eqn (4.4) by c (so that Eqn (4.15) is not generated), we reduce the column weights of both ac and cd. Note that ac occurs in Eqn (4.12) and (4.15) from the common term a in Eqns (4.1) and (4.4), whereas cd occurs in Eqns (4.13) and (4.15) from the common term d in Eqns (4.2) and (4.4). 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 (In terms of rank deficit) 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.

4.3.1 XL_SGE with Random Monomial Multiplication (XL_SGE-2)

As a first attempt, 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.

The modified algorithm XL_SGE-2 accepts as input the initial system of equations $\mathbb{A}$, 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 $\mathbb{A}'$. Make a copy of the linearized system $\mathbb{A}'$ as $\mathbb{B}$.
2. Apply structured Gaussian elimination (SGE) on $\mathbb{A}'$ with avalanche-control parameter K to obtain a reduced system of equations $\mathbb{A}''$ of degree d.

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) to obtain a system $\mathbb{A}'''$ of degree $d + 1$. Append the equations of $\mathbb{A}'''$ to $\mathbb{B}$. $\mathbb{B}$ now contains equations of degrees up to $d + 1$. Rename the system $\mathbb{A}'''$ as $\mathbb{A}'$.
4. If the degree of the system of equations $\mathbb{B}$ is D, end the process. Otherwise, go to Step 2 with d incremented by 1.

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 (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 $y_1y_2\cdots y_{10}/y_i$ by y_i for $i = 1, 2, \ldots, 10$. So $y_1y_2\cdots y_{10}$ having column weight one requires more restriction at the expansion stage for $d = 9$ than $y_1y_2y_3$ 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.

4.3.2 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: SGE 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, and delete the 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 4.1: Performance of XL_SGE-2
(Random monomial multiplication done in Step 1 of Algorithm 4)

Size	XL			XL_SGE-2 *					XL_SGE-2 †			
of $\mathbb{A}$	D	Size of $\mathbb{B}$	δ	D	K	p	Size of $\mathbb{B}$	δ	D	K	Size of $\mathbb{B}$	δ
12×7	4	317×98	0	3	3	0.75	39×35	0	3	3	35×36	2
15×10	3	388×175	0	3	3	0.67	210×169	0	3	6	182×173	3
16×10	4	1229×385	0	4	3	0.67	686×362	0	4	7	619×385	1
16×12	5	5232×1585	0	5	7	0.67	4163×1584	0	5	8	2693×1557	4
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
18×13	4	2796×1092	0	4	6	0.67	1954×1091	0	4	8	1521×1092	1
19×14	4	4473×1470	0	4	5	0.67	3375×1470	0	4	8	2183×1468	2
19×15	3	1247×575	1	3	4	0.67	841×574	1	3	8	583×572	3
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.

δ: rank deficit of the final linearized system $\mathbb{B}$.

4.4 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 (XL_SGE$'$) 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 4.1. 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 4.1. 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 4.2 . 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 4.2(a) shows the results obtained by XL_SGE-2 for $p = 0.67$,

Table 4.2: Variation of the final system size in XL_SGE-2 for some seed values

Size of $\mathbb{B}$	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) $p = 0.67$

Size of $\mathbb{B}$	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 4.3: Performance of XL_SGE-2
(Random monomial multiplication not done in Step 1 of Algorithm 4)

Size of	XL			XL_SGE-2				
$\mathbb{A}$	D	Size of $\mathbb{B}$	δ	D	K	p	Size of $\mathbb{B}$	δ
12×7	4	317×98	0	3	3	0.50	29×29	1
15×10	3	388×175	0	3	0	0.50	270×175	0
16×10	4	1229×385	0	4	0	0.67	674×355	0
16×12	5	5232×1585	0	5	7	0.67	4221×1584	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
18×13	4	2796×1092	0	4	6	0.50	1714×1092	0
19×14	4	4473×1470	0	4	5	0.50	3099×1469	0
19×15	3	1247×575	1	3	0	0.67	1068×575	1
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

and Table 4.2(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 4.1, 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 4.3 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 4.4. 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 multiplications 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 4.5. In some cases, XL_SGE$'$ works very well. In Row 2 of Table 4.5, 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

Table 4.4: Comparison of performances of XL, XL_SGE and XL_SGE-2

Size of	XL			XL_SGE				XL_SGE-2				
$\mathbb{A}$	D	Size of $\mathbb{B}$	δ	D	K	Size of $\mathbb{B}$	δ	D	K	p	Size of $\mathbb{B}$	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 4.5: 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

system for $D = 5$, whereas XL_SGE′ gives the solvable system for $D = 4$. In Row 5 of Table 4.5, 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 4.5), where XL_SGE′ does not work better than XL. In fact, in some of those cases, the performance of XL_SGE′ is even poorer than XL_SGE.

We have not studied XL_SGE-2 with column-weight two reduction. 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.

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 [18]. 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

Table 4.6: Performances of XL and variants of XL_SGE for four-round baby-Rijndael ($D = 3$)

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

baby-Rijndael are shown in Table 4.6. 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.

4.5 Conclusion

In this chapter, we suggest improved variants of our proposal XL_SGE. The variant XL_SGE$'$ works very well in some cases. However, in the current form, it fails to perform in most of the cases. More investigation is needed in order to make XL_SGE$'$ more versatile and effective. We have used random monomial multiplication strategy in XL_SGE-2, which significantly improve the performance of XL algorithm in almost all the cases. A more intelligent partial monomial multiplication strategy may exploit the structures of the intermediate systems in a positive way. In the next chapter, we will discuss an intelligent variant of XL_SGE, which exploits the structures of the intermediate systems to reduce the system size in a controlled way.

Chapter 5

XL_SGE with Row Deletion

In the previous chapter, we have proposed some improved variants of XL_SGE based upon random monomial multiplication and handling of columns of weight two. In this chapter, we are going to discuss an intelligent strategy, by deleting redundant equations in a controlled manner, to improve the performance of XL_SGE. The rest of the chapter is organized as follows. Section 5.2 discusses XL_SGE with redundant row deletion strategy. Section 5.3 shows our experimental results and compare different variants of XL_SGE algorithm with XL. Section 5.4 concludes the chapter with the future scopes of research.

5.1 Introduction

In the previous chapter, we demonstrate the benefits of using partial monomial multiplication with XL_SGE. In XL_SGE-2 we randomly skip some monomial multiplications with some probability in order to facilitate application of SGE in the next layer of the algorithm. Fewer multiplication also generates fewer number of variables and reduce the system size even before the application of SGE. Instead of blindly skipping certain multiplications, we can adopt a more intelligent strategy. We first carry out all monomial multiplications. Subsequently, by analyzing the column statistics of the expanded system, we mark some equations as less important than the others. We delete the less important equations from the system and then perform SGE before the next stage of multiplication. This variant, henceforth referred to as XL_SGE-3, has one potential advantage over XL_SGE-2. Now, we have a better control over the initial reduction in the system size in the sense that the degradation of the rank profile can be carefully handled, if not eliminated altogether. Moreover, the subsequent application of SGE is expected to be more effective in XL_SGE-3 than in XL_SGE-2.

5.2 XL_SGE with Row Deletion (XL_SGE-3)

The algorithm XL_SGE-3 in general accepts as input the initial system of equations $\mathbb{A}$, a degree bound $D \in \mathbb{N}$, the avalanche-control parameter $K \in \mathbb{N}$. The steps of XL_SGE-2 follow.

Algorithm 6: XL_SGE with Row Deletion (XL_SGE-3)

1. Expand the initial system $\mathbb{A}$ up to degree $d = 2$ using XL to obtain a linearized system $\mathbb{A}'$. Make a copy of the linearized system $\mathbb{A}'$ as $\mathbb{B}$.
2. Apply structured Gaussian elimination (SGE) with avalanche-control parameter K on $\mathbb{A}'$ to obtain a reduced system $\mathbb{A}''$ of degree d.
3. Multiply the reduced system $\mathbb{A}''$ with monomials of degree 1 and linearize the system to obtain a system $\mathbb{A}'''$ of degree $d+1$.
4. Identify and delete some rows of $\mathbb{A}'''$. Append the equations of $\mathbb{A}'''$ to $\mathbb{B}$. $\mathbb{B}$ now contains equations of degrees up to $d+1$. Rename $\mathbb{A}'''$ as $\mathbb{A}'$.
5. If the degree of the system of equations $\mathbb{B}$ is D, end the process. Otherwise, go to step 2 after incrementing d by 1

Depending upon how we identify the redundant rows for deletion in Step 4, we have different variants of XL_SGE-3, some of which are elaborated below. The deletion of redundant equations can also be employed after Step 1 of Algorithm 6.

5.2.1 XL_SGE-3 with Deterministic Deletion (XL_SGE-3d)

We have experimented with some deterministic deletion strategies in Step 4 of Algorithm 6. We have considered only the variables of column weight two. Among the two equations containing a variable with column weight two, we delete (at most) one equation depending upon one of the following strategies.

Strategy 1

- If any of these two equations contains a variable with column weight one, then skip the deletion of both the equations. (In this case, the equation with the variable with column weight one is anyway deleted by SGE, thereby reducing the weight of the variable with column weight two.)
- Otherwise, delete the equation with the larger row weight. If both the equations have the same row weight, delete any one of these arbitrarily.

Strategy 2

- If any of these two equations contains a variable with column weight one, then skip the deletion of both the equations.
- If both the equations have the same right side (0 or 1), delete the equation with the larger row weight. Make arbitrary choices to break ties.
- If exactly one of the two equations has right side 1, then keep that equation, and delete the other.

Strategy 3

- If any one of the equations contains a variable with column weight one, then determine whether that variable can reappear in the system in a future monomial-multiplication stage. If not, none of the equations is deleted. Otherwise, delete the equation containing the variable with column weight one.
- If both the equations contain variables of column weight one that can reappear from a future monomial-multiplication stage, then delete one of them depending on their row weights (as in Strategy 1).

- If both the equations contain no variables of column weight one, then take decision as in Strategy 1.

Let $z = x_1x_2x_3$ be a monomial with column weight one, and let the equation containing z also contain a variable with column weight two. In Strategy 3, we check whether z can reappear in the next multiplication layer (like multiplication of x_1x_3 by x_2). If that is the case, the current rank degradation incurred by the deletion of the equation containing z will be repaired later.

5.2.2 XL_SGE-3 with Random Deletion (XL_SGE-3r)

Let z be a variable (monomial) with weight t. We delete m of the t equations in which z appears. If the system is overdefined, this deletion is, in general, not expected to have a bad effect on the rank profile. The details of this strategy are given below. In our experiments, we have worked with $t = 2$ and 3, and $m = 1$.

- Find an equation with a variable of column weight t.
- If the equation contains a variable of column weight one, skip the deletion.
- Otherwise, delete the equation with probability p_d.
- Repeat this process until there are no removable equations with variables of column weight t.

5.3 Experimental Results

We have experimented with small random multivariate quadratic systems and also with the initial system of size 890×208 obtained from four-round baby-Rijndael.

The best performance of XL_SGE-3 with different deletion strategies is compared with the performance of XL and XL_SGE in Tables 5.1 and 5.2. For XL_SGE-3r, we have used the fixed deletion probabilities $p_d = 0.50$, 0.33, 0.25

Table 5.1: Comparison of performances of XL, XL_SGE and XL_SGE-3d

Size	XL			XL_SGE				XL_SGE-3d			
of $\mathbb{A}$	D	Size of $\mathbb{B}$	δ	D	K	Size of $\mathbb{B}$	δ	D	K	Size of $\mathbb{B}$	δ
13×9	3	233×129	1	3	5	231×129	1	3	5	158×129	1
14×11	4	2797×561	0	4	0	2080×561	0	4	4	1748×561	0
15×12	6	12831×2509	0	6	7	6903×1641	21	6	8	8002×2509	3
17×14	4	3242×1470	0	4	7	3230×1470	0	4	8	2798×1470	1
18×14	4	4541×1470	0	3	0	900×458	1	3	0	882×456	1
19×16	3	1282×696	0	3	5	1282×696	0	3	5	1172×696	0
19×16	3	1401×696	0	3	0	1401×696	0	3	0	1348×696	1
20×16	3	1420×696	0	3	0	1400×696	0	3	0	1365×696	0
20×17	4	7820×3213	0	4	3	7514×3213	0	4	7	7158×3213	1
22×19	5	53404×16663	0	5	6	52944×16663	0	5	6	48797×16661	2

and 0.20. If deletion is used after Step 1 of Algorithm 6, we use the suffix $d \geq 2$. XL_SGE-3 is evidently superior than XL and XL_SGE. Moreover, XL_SGE-3 has been experimentally found (see Table 5.4) to have performance comparable with XL_SGE-2.

The performances of several variants of XL_SGE on small random systems are shown in Table 5.3. Table 5.4 shows that XL_SGE-2 and XL_SGE-3 significantly improve the performance of XL and XL_SGE, both in terms of the size and the rank deficit of the final system in case of four round baby-Rijndael. In case of small random multivariate quadratic systems XL_SGE-3r performs significantly better than that of XL_SGE-3d. However, in case of four round baby-Rijndael XL_SGE-3d (for $D = 3$) produces smallest system with lowest rank deficit.

5.4 Conclusion

Redundant row deletion improves the performance of XL_SGE. Specifically random row deletion strategy performs significantly better than deterministic row deletion strategies in case of small random multivariate quadratic systems. We have observed that the performance of XL_SGE-2 and XL_SGE-3r are equivalent almost in all the cases. However, an optimal identification of redundant equations

Table 5.2: Comparison of performances of XL, XL_SGE and XL_SGE-3r

Size	XL			XL_SGE				XL_SGE-3r				
of $\mathbb{A}$	D	Size of $\mathbb{B}$	δ	D	K	Size of $\mathbb{B}$	δ	D	K	p_d	Size of $\mathbb{B}$	δ
15×11	4	1463×561	0	4	5	1326×561	0	4	6	$0.50_{d\geq 2}$	743×560	0
15×11	4	1580×561	0	4	3	2152×560	0	4	6	$0.50_{d\geq 2}$	784×560	0
16×12	5	5242×1585	0	5	7	5383×1585	0	5	8	$0.50_{d\geq 2}$	2387×1585	1
17×13	5	7653×2379	0	5	7	8267×2379	0	5	8	$0.50_{d\geq 2}$	4182×2379	0
19×15	4	5324×1940	0	4	0	5052×1940	0	4	8	$0.50_{d\geq 2}$	2673×1940	0
20×16	5	22658×6884	0	5	0	21485×6884	0	5	7	0.50	14746×6884	0
22×18	4	10277×4047	0	4	3	9881×4047	0	4	6	0.50	6992×4047	0
22×18	5	42721×12615	0	5	3	44090×12615	0	5	8	$0.50_{d\geq 2}$	24432×12615	0
23×19	6	149801×43795	0	6	7	127571×43788	7	6	7	0.33	118274×43764	3
23×19	6	161823×43795	0	6	8	144223×43795	1	6	8	0.50	113284×43795	1

Table 5.3: Performances of XL and variants of XL_SGE for random systems

	Size of $\mathbb{B}$				
Size of $\mathbb{A}$	XL	XL_SGE	XL_SGE-2	XL_SGE-3d	XL_SGE-3r
15×10	2712×637	2528×619	1447×631	1939×637	1360×637
16×11	2846×561	2119×561	943×561	1322×560	934×561
17×12	749×298	748×298	460×298	714×298	394×298
18×14	5347×1470	4796×1469	2199×1461	4356×1469	2462×1469
19×14	4831×1470	3620×1470	2333×1468	3447×1470	2414×1470
20×15	3783×1940	3963×1940	2907×1940	3149×1940	3073×1940
20×16	6402×2516	6094×2516	3700×2514	5407×2516	3994×2516
23×18	117996×31179	122701×31175	86200×31175	112307×31172	85227×31179

Table 5.4: Performances of XL and variants of XL_SGE for four-round baby-Rijndael ($D = 3$)

Algorithm	K	p	p_d	Size of $\mathbb{B}$	δ
XL	0	1	0	2594060×1498713	96936
XL_SGE	3	1	0	2571848×1476481	93172
XL_SGE-2	0	$0.75_{d\geq 2}$	0	2276971×1442363	89387
XL_SGE′	0	1	0	2556116×1449153	81576
XL_SGE-3d	0	1	0	1934149×1163740	79630
XL_SGE-3r	0	1	0.20	2355165×1449152	85470
XL_SGE-3r	0	1	0.25	2283125×1449152	89640

remain as an open problem. A better selection criterion of redundant equations may enhance the performance of XL_SGE-3 furthermore.

Chapter 6

Conclusion

Algebraic cryptanalysis has received much attention recently from the community of cryptographers. This is a totally different cryptanalytic approach compared to the other statistical cryptanalysis techniques. Algebraic attacks exploit the intrinsic algebraic structure of the cipher. In these attacks, one describes the encryption operation as a large set of multivariate polynomial equations, which once solved can be used to recover the secret key. Thus the difficulty of solving large multivariate polynomial system of equations arising from a cipher is directly related to the security of the cipher. In principle, algebraic attacks are applicable to both stream ciphers and block ciphers. However, they have been much more effective in the analysis of stream ciphers than that of block ciphers. The problem in case of block ciphers is that the system size becomes unmanageably large, and consequently complexity of solving such system often exceeds the complexity of brute-force search. In this thesis, we have proposed several improvements upon the XL family of algebraic attacks.

6.1 Summary of Work Done

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.

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. We identify the problems in XL_SGE and propose three variants of XL_SGE, based upon partial monomial multiplication (XL_SGE-2), handling of columns of weight two (XL_SGE$'$) and deletion of redundant equations (XL_SGE-3). Depending on the deletion criterion (whether random or deterministic), we propose XL_SGE-3r and XL_SGE-3d as two variants of XL_SGE-3. Our modified algorithms have been experimentally verified to be superior to XL_SGE and also these variants of XL_SGE (specifically XL_SGE-2 and XL_SGE-3r) significantly improve the performance of XL algorithm.

6.2 Directions of Future Research

We end this thesis 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 family 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.
- 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.
- The dependence of the system size and rank profile on the seed (multiplication or deletion decisions)—a property inevitably associated

with randomized algorithms like XL_SGE-2 and XL_SGE-3r—should be studied.

- An optimal choice for multiplication probability p (in XL_SGE-2) and deletion probability p_d (in XL_SGE-3r) requires more experimentation and theoretical analysis.

- We have used some deletion criteria in XL_SGE-3d and XL_SGE-3r. A different choice of selection criteria for redundant rows may enhance the performance of XL_SGE-3 furthermore. Finding an optimal (or close to optimal) deletion criteria (without affecting the rank profile adversely) can be an interesting problem to investigate.

- The domains of applicability of XL_SGE$'$ need to be experimentally and 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.

- Another important area of investigation is to use SGE in conjunction with the variants of linearization techniques (like XSL and MutantXL). Comparisons with other algebraic-attack algorithms (like F4, F5, SAT-solver techniques) are also worth studying.

Dissemination of Work

[1] Satrajit Ghosh and Abhijit Das, “An improvement of linearization-based algebraic attacks,” *Security Aspects in Information Technology*, volumn 7011 of Lecture Notes in Computer Science, pages 157 − 167, Springer, 2011.

[2] Satrajit Ghosh and Abhijit Das, “New variants of algebraic attacks based on structured Gaussian elimination,” in *Third international conference on Symbolic Computation and Cryptography (SCC* 2012*)*, pages 119 − 125, Castro Urdiales, Spain, July 2012.

Bibliography

[1] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[2] Claude E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28:657–715, 1949.

[3] Jean Charles Faugère. A new efficient algorithm for computing Gröbner basis (F_4), 2000.

[4] Jean Charles Faugère. A new efficient algorithm for computing Gröbner basis without reduction to zero (F_5). ISSAC '02, pages 75–83, 2002.

[5] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *CRYPTO*, pages 19–30, 1999.

[6] Nicolas T. Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *EUROCRYPT*, pages 392–407, 2000.

[7] Nicolas T. Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *ASIACRYPT*, pages 267–287, 2002.

[8] J. Ding, J. Buchmann, M.S.E. Mohamed, W.S.A. Mohamed, and R.P. Weinmann. MutantXL. In *SCC*, pages 16–22, 2008.

[9] Nicolas T. Courtois Gregory V. Bard and Chris Jefferson. Solution of sparse polynomial systems over GF(2) via sat-solvers. In *ECRYPT workshop Tools for Cryptanalysis*, 2007.

[10] Nicolas T. Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *Proceedings of the 22nd international conference*

on Theory and applications of cryptographic techniques, EUROCRYPT'03, pages 345–359. Springer-Verlag, 2003.

[11] Nicolas T. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer, 2003.

[12] Jean Charles Faugère, Françoise Levy dit Vehel, and Ludovic Perret. Cryptanalysis of MinRank. In David Wagner, editor, *Advances in Cryptology CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 280–296. Springer, 2008.

[13] Jean Charles Faugère and Antoine Joux. Algebraic cryptanalysis of hidden field equation (HFE) cryptosystems using Gröbner bases. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003.

[14] Jean Charles Faugère and Ludovic Perret. Cryptanalysis of $2R^-$ schemes. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 357–372. Springer Berlin, 2006.

[15] Nicolas T. Courtois and Jacques Patarin. About the XL algorithm over GF(2). In *Proceedings of the 2003 RSA conference on The cryptographers' track*, CT-RSA'03, pages 141–157. Springer-Verlag, 2003.

[16] B. LaMacchia and A. Odlyzko. Solving large sparse linear systems over finite fields. In *CRYPTO*, pages 109–133. 1991.

[17] Joan Daemen and Vincent Rijmen. Rijndael for AES. In *AES Candidate Conference*, pages 343–348, 2000.

[18] Elizabeth Kleiman. The XL and XSL attacks on baby Rijndael. Master's thesis, Iowa State University, Department of Mathematics, 2005.

[19] Martin Vörös. Algebraic attack on stream ciphers. Master's thesis, Comenius University, Faculty of Mathematics, Physics and Informatics, Department of Computer Science, 2007.

[20] Mohamed Mohamed, Wael Mohamed, Jintai Ding, and Johannes Buchmann. MXL2: Solving polynomial equations over GF(2) using an improved mutant strategy. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 203–215. Springer, 2008.

[21] Ralf-Philipp Weinmann. Evaluating algebraic attacks on the AES. Master's thesis, Fachbereich Informatik, Fachgebiet Kryptographie und Computeralgebra, Technische Universität Darmstadt, 2003.

[22] Bo yin Yang and Jiun ming Chen. Theoretical analysis of XL over small fields. In *In Proceedings of the 9th Australasian Conference on Information Security and Privacy*, pages 277–288. Springer, 2004.

[23] Claus Diem. The XL-algorithm and a conjecture from commutative algebra. In *Proceedings of Asiacrypt 2004, LNCS, volume 3329*, pages 323–337. Springer-Verlag, 2004.

[24] Abhijit Das. Computational Number Theory. manuscript of a book.

[25] Carlos Cid and Gaëtan Leurent. An analysis of the XSL algorithm. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2005.

[26] Chu-Wee Lim and Khoongming Khoo. An analysis of XSL applied to BES. In Alex Biryukov, editor, *Fast Software Encryption*, volume 4593 of *Lecture Notes in Computer Science*, pages 242–253. Springer, 2007.

[27] Gwénolé Ars, Jean Charles Faugère, Hideki Imai, Mitsuru Kawazoe, and Makoto Sugita. Comparison between XL and Gröbner basis algorithms. In *ASIACRYPT 2004, Lecture*, pages 338–353. Springer-Verlag, 2004.

[28] Nicolas T. Courtois Gregory V. Bard and Chris Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over GF(2) via SAT-solvers. Cryptology ePrint Archive, Report 2007/024, 2007.

[29] Nicolas T. Courtois and Gregory V. Bard. Algebraic cryptanalysis of the Data Encryption Standard. In *IMA Int. Conf.*, pages 152–169, 2007.

[30] Nicolas T. Courtois, Gregory V. Bard, and David Wagner. Algebraic and slide attacks on Keeloq. In *FSE*, pages 97–115, 2008.

[31] Nicolas T. Courtois, Sean O'Neil, and Jean-Jacques Quisquater. Practical algebraic attacks on the Hitag2 stream cipher. In *ISC*, pages 167–176, 2009.

[32] Mohamed Saied Mohamed, Jintai Ding, Johannes Buchmann, and Fabian Werner. Algebraic attack on the MQQ public key cryptosystem. In *Proceedings of the 8th International Conference on Cryptology and Network Security*, CANS '09, pages 392–401. Springer-Verlag, 2009.

[33] Jeremy Erickson, Jintai Ding, and Chris Christensen. Algebraic cryptanalysis of SMS4: gröbner basis attack and SAT attack compared. In *Proceedings of the 12th international conference on Information security and cryptology*, ICISC'09, pages 73–86. Springer-Verlag, 2010.

[34] Jean Charles Faugère Martin R. Albrecht, Carlos Cid and Ludovic Perret. On the relation between the MXL family of algorithms and Gröbner basis algorithms. *J. Symb. Comput.*, 47(8):926–941, August 2012.