Algebraic Attack Efficiency versus S-box Representation

Hossein Arabnezhad-Khanoki[1], Babak Sadeghiyan[1], and Josef Pieprzyk[2,3]

[1] Department of Computer Engineering & Information Technology, Amirkabir University of Technology, Tehran, Iran
{arabnezhad,basadegh}@aut.ac.ir

[2] School of Electrical Engineering and Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
josef.pieprzyk@qut.edu.au

[3] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Abstract. Algebraic analysis of block ciphers aims at finding the secret key by solving a collection of polynomial equations that describe the internal structure of a cipher for chosen observations of plaintext/ciphertext pairs. Although algebraic attacks are addressed for cryptanalysis of block and stream ciphers, there is a lack of understanding of the impact of algebraic representation of the cipher on efficiency of solving the resulting collection of equations. The work investigates different S-box representations and their effect on complexity of algebraic attacks. In particular, we observe that a S-box representation defined in the work as *Forward-Backward* (FWBW) leads to a collection of equations that can be solved efficiently. We show that the $SR(10, 2, 1, 4)$ cipher can be broken using standard algebra software Singular and FGb. This is the best result achieved so far. The effect of description of S-boxes for some light-weight block ciphers is investigated. A by-product of this result is that we have achieved some improvements on the algebraic cryptanalysis of LBlock, PRESENT and MIBS light-weight block ciphers. Our study and experiments confirms a counter-intuitive conclusion that algebraic attacks work best for the FWBW S-box representation. This contradicts a common belief that algebraic attacks are more efficient for quadratic S-box representation.

1 Introduction

The winner of the AES competition was the Rijndael block cipher [19]. Soon after the announcement, NIST approved it as a new advanced encryption standard (AES). An elegant and simple algebraic structure of AES gives a strong motivation towards development of new cryptanalysis techniques. These techniques first describe a block cipher as a system of relations and then the system of relations is solved. One of such approaches was given by Courtois and Pieprzyk [16], where an extended sparse linearization (XSL) is used to solve the system of relations. The AES block cipher is described by a system of polynomial equations over $\mathbb{F}_2$ and the XSL algorithm is used to recover the secret key. It was estimated that the XSL attack would be able to break AES slightly faster than the exhaustive search.

The XSL attack turns out to be ineffective against AES block cipher [11,12]. Note that the system of relations described the AES block cipher with 128-bit keys consists of 8000 nonlinear relations in 1600 variables. It is impossible to verify experimentally the claimed complexity of the XSL attack. Cid et al. [10] propose a family of scalable versions of AES to study and experiment with cryptanalysis of AES. The authors of [10] report results of experiments, where cryptanalysis is performed for 4-bit and 8-bit versions of an round-reduced AES using the Gröbner basis algorithm F4 [24]. To our knowledge, best algebraic cryptanalysis results for this family of ciphers is reported in [9].

Another interesting approach in solving systems of polynomial equations over $\mathbb{F}_2$ is application of SAT-solvers. Courtois et all. [14] used this approach to analyse DES. The DES relations written in *algebraic normal form* (ANF) have to be translated into *conjunctive normal form* (CNF) and then given to a SAT-solver.

In general algebraic attacks progress in two following steps:

1. finding a system of relations that describe the block cipher, where an adversary can observe plaintext and ciphertext pairs. The unknowns are bits/bytes of secret key,
2. solving the obtained system of relations using appropriate algorithm (XL, F4, SAT solvers, ...).

There exists virtually infinitely many ways to algebraically describe a block cipher. An adversary (cryptanalyzer) would like to form these algebraic relations in such a way that they can be solved as fast as possible. In [16], along with the introduction of XSL cryptanalysis, the effect of quadratic and cubic description of AES S-box on the method is studied. Courtois et al. in [15], discussed different types of equations that relate input and output of DES S-boxes. Courtois and Bard in [14], break 6 and 4 rounds of the block cipher DES using quadratic and cubic description of S-boxes, with SAT solver and ElimLin algorithm, respectively. The quadratic description of S-boxes in [14] are derived from an efficient hardware implementation of the DES S-boxes. In [1,2] different descriptions for various S-boxes are presented.

Efficiency of description of S-boxes for hardware or software implementation of the cipher might result in efficient algebraic cryptanalysis using SAT solver [14]. In [25], a method is proposed to generate a shortest linear program for linear transformation of AES S-box. In [13], a method based on SAT solver for generating low gate complexity or low multiplicative complexity implementation for S-boxes is proposed. The method obtained a low gate circuit for PRESENT light-weight cipher. In this paper, our aim is to propose an S-box description, considering efficient algebraic cryptanalysis based on Gröbner basis methods.

AES-like block ciphers apply the Shannon's concept of SP networks. Each iteration applies nonlinear S-box operations and linear diffusion. S-box transformations are directly responsible for algebraic degree of relations and a suspected rapid growth of algebraic degree of concatenations of consecutive iterations.

In this work, we investigate two algorithms for solving algebraic relations, namely, Gröbner basis algorithms and SAT-solvers. Our study and theoretical results are verified experimentally on (small-scale) variants of the AES family (SR) presented by Cid at al. in [10]. In particular, we investigate the impact of algebraic descriptions of S-boxes on efficiency of algebraic attacks. We consider relation describing S-boxes as a generating set for the ideal determined by the variety of the S-box. We show that a representation of the SR S-box by a system of relations (polynomials) for both the S-box and its inverse gives the best results. In particular, using this representation, we are able to solve system of equations for $SR(10, 2, 1, 4)$ with computer algebra softwares Singular [20] and FGb [22], where $SR(n, r, c, e)$ is a AES variant with n rounds, r is the number of rows, c - the number of columns and e is the size of the word in bits. The analysis reported in [10,9] uses the F4 algorithm [24] and Singular. However, it fails for the number of rounds $n \geq 5$. We also give an algorithm for finding very sparse system of relations representing the SR S-box. Interestingly enough, application of the sparse quadratic system in our cryptanalysis gives worse results. We are able to solve system of polynomial relations for $SR(10, 2, 2, 4)$ in 2.69 seconds on the average using CryptoMiniSat [30]. This result is better than the one reported in [6]. We also study lightweight block ciphers LBlock [33], PRESENT [4] and MIBS [27] and analyse their strength against algebraic attacks with proposed description of S-boxes.

Contributions: The main contributions of the work are as follows:

(a) developing a framework that allows us to find different representations of S-boxes. This includes a new algorithm for generating sparse polynomial systems,
(b) finding a counter-intuitive example of algebraic analysis, where sparse quadratic relations for 4-bit S-boxes give worse results when either the Gröbner basis algorithm or the SAT-solver is used to solve them,
(c) showing that using both *forward-backward* polynomial relations for S-boxes allows us to break 11-round version of the LBlock cipher using the Gröbner basis tools,

(d) showing that using both *forward-backward* polynomial relations for S-boxes allows us to break 6-round version of the PRESENT cipher using the Gröbner basis tools,
(e) presenting the first algebraic cryptanalysis of 6 rounds of the MIBS cipher.

The paper is organized as follows. In Section 2, we present algebraic background for computation of Gröbner basis over finite fields. Then, in Section 4, we present basic concepts related to the description of S-boxes by a system of polynomials. We also give an algorithm that allows us to find a sparse system of polynomial relations describing S-boxes. In Section 5, we review the small-scale variants of AES and published results. Our experiments are presented in Section 6. We give conclusions and future research directions in Section 7.

2 Preliminaries

We adopt notations and definitions from [18]. A polynomial is defined over a field $\mathbf{K}$. The set of all polynomials in variables $x_1, x_2, \ldots, x_n$ with coefficients in $\mathbf{K}$ is denoted by $\mathbf{K}[x_1, x_2, \ldots, x_n]$. In algebraic cryptanalysis, a block cipher is described by polynomial relations over $\mathbb{F}_2$. The polynomials describing the cipher reflect the behaviour of the cipher as long as the secret key is fixed. Their solution should pinpoint the secret key. Solutions form an algebraic object called *variety*, which is defined as follows.

Definition 1 ([18]). *Given a field* $\mathbf{K}$ *and a positive integer* n*, we define the* n*-dimensional* ***Affine Space*** *over* $\mathbf{K}$ *to be the set*

$$\mathbf{K}^n = \{(a_1, \ldots, a_n) : a_1, \ldots, a_n \in \mathbf{K}\}$$

Definition 2 ([18]). *Let* $\mathbf{K}$ *be a field, and let* $f_1, \ldots, f_s$ *be polynomials in* $\mathbf{K}[x_1, \ldots, x_n]$*. Then we set*

$$\mathbf{V}(f_1, \ldots, f_s) = \{(a_1, \ldots, a_n) \in \mathbf{K}^n : f_i(a_1, \ldots, a_n) = 0 \text{ for all } 1 \leq i \leq s\}$$

where $\mathbf{V}(f_1, \ldots, f_s)$ *is called the* ***Affine Variety*** *defined by* $f_1, \ldots, f_s$*.*

Note that $\mathbf{V}(f_1, \ldots, f_s)$, is the set of all common solutions to the polynomials $f_1, \ldots, f_s$ as well. Another related mathematical object is *ideal*. An ideal is defined as follows.

Definition 3 ([18]). *A subset* $I \subset \mathbf{K}[x_1, ..., x_n]$ *is an ideal if it satisfies:*

1. $0 \in I$.
2. *If* $f, g \in I$*, then* $f + g \in I$.
3. *If* $f \in I$ *and* $h \in \mathbf{K}[x_1, ..., x_n]$*, then* $h \cdot f \in I$.

We can form an ideal for any set of polynomials in $\mathbf{K}[x_1, ..., x_n]$.

Definition 4 ([18]). *Let* $f_1, \ldots, f_s$ *be polynomials in* $\mathbf{K}[x_1, \ldots, x_n]$*. Then we set*

$$\langle f_1, \ldots, f_s \rangle = \left\{ \sum_{i=1}^{s} h_i \cdot f_i : \forall h_1, \ldots, h_s \in \mathbf{K}[x_1, \ldots, x_n] \right\}$$

Lemma 1 ([18]). *If* $f_1, \ldots, f_s \in \mathbf{K}[x_1, \ldots, x_n]$*, then* $\langle f_1, \ldots, f_s \rangle$ *is an ideal of* $\mathbf{K}[x_1, \ldots, x_n]$*. We call* $\langle f_1, \ldots, f_s \rangle$ *the* ***ideal generated by*** $f_1, \ldots, f_s$*.*

Hilbert Basis Theorem states that any ideal in $\mathbf{K}[x_1, ..., x_n]$ can be generated by a finite set of polynomials. The following two definitions explain the relation of ideal and variety.

Definition 5 **([18]).** *Let $I \subset \mathbf{K}[x_1, ..., x_n]$ be an ideal. We denote by $\mathbf{V}(I)$ the set*

$$\mathbf{V}(I) = \{(a_1, \ldots, a_n) \in \mathbf{K}^n : f(a_1, \ldots, a_n) = 0 \text{ for all } f \in I\}$$

Please note that $\mathbf{V}(I)$ is the variety defined by ideal I.

Definition 6 **([18]).** *Let $V \subset \mathbf{K}^n$ be an affine variety. Then **ideal of** V will be denoted by $\mathbf{I}(V)$ the set*

$$\mathbf{I}(V) = \{f \in \mathbf{K}[x_1, ..., x_n] : f(a_1, \ldots, a_n) = 0 \text{ for all } (a_1, \ldots, a_n) \in V\}$$

There are some decision problems related to ideal and variety that are answered by computation of *Gröbner Basis* of an ideal. They are:

- ***Ideal Membership:*** whether a polynomial f belongs to an ideal I?
- ***Ideal Equality:*** whether two ideals I_1 and I_2 are equal?

In 1965, Buchberger introduced the concept of Gröbner basis and proposed an algorithm for computation of Gröbner basis for a set of polynomials [7]. To compute Gröbner basis, we set an ordering over monomials that appear in polynomials. Three normal ordering are: *Lexicographic* (**lex**), *Degree Lexicographic* (**deglex**)and *Degree Reverse Lexicographic* (**degrevlex**). In **lex**, monomials are ordered based on lexicographic order of variables. In **deglex** and **degrevlex**, which are degree based orders, at first monomials are ordered based on their degree and then, for each degree, monomials are ordered based on **lex** or its reverse order.

Definition 7 **([18]).** *Fix a monomial order. A finite subset $G = \{g_1, \ldots, g_t\}$ of an ideal I is said to be a Gröbner basis (or standard basis) if*

$$\langle LT(g_1), \ldots, LT(g_t)\rangle = \langle LT(I)\rangle$$

In the above definition, $LT(g_i)$ denotes the *leading term* of g_i, and $LT(I)$ denotes the set of the leading terms of elements of I, and $\langle LT(I)\rangle$ denotes the ideal generated by the leading terms of elements of I.

Gröbner basis allows us to answer ideal membership problem by reducing (dividing) polynomial f modulo (by) G, where G is the Gröbner basis of I. If it reduces to zero or does not have a remainder, then $f \in I$.

Definition 8 **([18]).** *A **reduced Gröbner basis** for a polynomial ideal I is a Gröbner basis G for I such that:*

1. *$LC(p) = 1$ for all $p \in G$, where LC is the coefficient of the leading term of p.*
2. *For all $p \in G$, no monomial of p lies in $\langle LT(G - \{p\})\rangle$*

The main property of reduced Gröbner basis is stated in the following proposition of [18].

Proposition 1 **([18]).** *Let $I \neq \{0\}$ be a polynomial ideal. Then, for a given monomial ordering, I has a unique reduced Gröbner basis.*

The uniqueness of a reduced Gröbner basis allows us to answer the ideal equality problem. To check whether ideals generated by two set of polynomials F and G are equal, we first compute the reduced Gröbner basis for each one, i.e. for F and G. Then if the two bases are equal, we conclude that F and G generate the same ideal.

Computation of Gröbner basis can be used to solve a system of polynomial relations. The following theorem explains how this can be done when relations are defined over $\mathbb{F}_2$.

Proposition 2 **([26]).** *A Gröbner basis G of ideal I in $\mathbb{F}_2[x_1, \ldots, x_n]$, where $I = \langle f_1, \ldots, f_m, x_1^2 - x_1, \ldots, x_n^2 - x_n\rangle$, describes all the solutions of $\mathbf{V}(I)$ in $\mathbb{F}_2$. Particular useful cases are:*

– $\mathbf{V}(I) = \emptyset$ *iff* $G = \langle 1 \rangle$.
– $\mathbf{V}(I)$ *has exactly one solution iff* $G = \langle x_1 - a_1, \ldots, x_n - a_n \rangle$ *where* $a_i \in \mathbb{F}_2$. *Then* $(a_1, \ldots, a_n)$ *is the solution in* $\mathbb{F}_2$ *of the algebaric system.*

The first algorithm for computation of Gröbner basis was proposed by Buchberger [8], which is easy to implement but not efficient. Faugere [24] proposed F4 algorithm, which uses linear algebra to do the reduction. A highly optimized version of algorithm F4 is implemented in FGb [22]. SlimGB [5] is a variant of Buchberger algorithm. Polynomials are prioritized according to the numbers of their monomials. The computation of Gröbner basis is done recursively where an intermediate basis is replaced by polynomials with smaller numbers of monomials. SlimGB is implemented in the Singular computer algebra software[20]. There is also an implementation of SlimGB with highly optimized data structures for representing Boolean polynomials. It is available as PolyBoRi library in C++ [6].

3 SAT-Solvers

Polynomial relations can be solved using SAT-Solvers. However, as ciphers are usually described using polynomials written in ANF, the ANF polynomials need to be converted to a single CNF formula. Having the cipher description written in CNF, we can apply a SAT-Solver. A solution (if exists) is an assignment to the variables, for which the CNF formula is true. Courtois et al. [3] propose a method to convert ANF polynomials into a single CNF. The method is used by the authors of [14] to translate *Data Encryption Standard* (DES) relations into a CNF formula. Next a SAT-Solver is applied to find a solution. This analysis allows to break DES if the number of rounds is no bigger than 6. Our investigations show that the relations describing a block cipher have an impact on the time needed to solve an instance. Hence, there is an interesting research question: how to algebraically describe a cipher as being converted to a single CNF formula, so finding solution is as fast as possible?

4 S-box Description

Modern block ciphers are designed to be implemented efficiently in both hardware and software. This is done using the well-known design principle based on Shannon's SP networks. The crux of the design is a single round, which consists of nonlinear layer of S-boxes and a linear layer that provides a fast diffusion. A round key is normally XORed at the beginning of the round. The single round is called *Round Function* and the round keys are produced by *Key Schedule Algorithm*. There is a fine balance between efficiency and security of a block cipher. A small number of rounds gives a fast cipher but its security may not achieve the expected level (normally determined by the length of the secret key). A large number of rounds is likely to give a very secure but slow cipher.

S-boxes are the only nonlinear components of block ciphers. Normally S-boxes translate input bits into output bits. A notable exception is DES S-boxes that translate 6-bit inputs into 4-bit outputs. Designing of cryptographically strong S-boxes is an important part of Cryptography. Again because of efficiency aspect, small S-boxes look more attractive (especially if they are implemented as lookup tables). Note however that any permutation 2×2 S-box is linear/affine. In practice, permutation $n \times n$ S-boxes are used, if $2 < n \leq 8$.

Forward (FW) Equations

Permutation $n \times n$ S-boxes translate n binary input variables $x_0, \ldots, x_{n-1}$ into n binary outputs $y_0, \ldots, y_{n-1}$. For example, the permutation 4×4 S-box from $SR(n, 2, 1, 4)$ [10] can be described with 4 equations of degree 3 as follows:

$$\begin{array}{l} f_0 : y_0 + x_0x_1x_2 + x_0x_1x_3 + x_0x_2x_3 + x_0x_2 + x_0x_3 + x_0 + x_1x_2x_3 + x_1x_3 + x_1 + x_3 \\ f_1 : y_1 + x_0x_1x_3 + x_0x_2x_3 + x_1x_2x_3 + x_1x_2 + x_1 + x_2x_3 + x_3 + 1 \\ f_2 : y_2 + x_0x_1x_2 + x_0x_1 + x_0x_2x_3 + x_0 + x_1x_2 + x_1x_3 + x_2x_3 + x_2 + x_3 + 1 \\ f_3 : y_3 + x_0x_1x_2 + x_0x_1x_3 + x_0x_1 + x_0x_3 + x_0 + x_2x_3 + x_3 \end{array} \tag{1}$$

We call polynomials in Equation (1) *FW* equations.

Backward (BW) Equations

Permutation S-boxes have their inverse permutations. We can derive a collection of equations that expresses the relation between output and input bits. We call them *Backward equations* or *BW equations* for short. For the permutation 4×4 S-box from $SR(n, 2, 1, 4)$ [10], we get the following relations:

$$\begin{array}{l} w_0 : x_0 + y_0y_1 + y_0y_2y_3 + y_0 + y_1y_2y_3 + y_1y_2 + y_2y_3 + y_2 + y_3 \\ w_1 : x_1 + y_0y_1y_3 + y_0y_1 + y_0y_2y_3 + y_0y_2 + y_0 + y_1y_2y_3 + y_1y_3 + y_1 + 1 \\ w_2 : x_2 + y_0y_1y_2 + y_0y_2y_3 + y_0y_3 + y_1y_2y_3 + y_2 + 1 \\ w_3 : x_3 + y_0y_1y_2 + y_0y_1y_3 + y_0y_1 + y_1y_2y_3 + y_1y_2 + y_1 + y_2y_3 + y_2 + 1 \end{array} \tag{2}$$

We call polynomials in Equation (2) *BW* equations.

Multivariate Quadratic (MQ) Equations

It is expected that the degree and sparsity of S-box equations have an effect on time needed to solve the system of equations for the whole cipher. Courtois et al. [16] observes that the AES S-box has a very compact and low degree representation for *Multivariate Quadratic* equations. For the permutation 4×4 S-box from $SR(n, 2, 1, 4)$ [10], we can get 21 polynomial equations of degree 2. They are:

$$\begin{array}{l} g_0 : x_2x_3 + y_1y_2 + y_0x_1 + y_0x_0 + y_2 + y_1 + y_0 + 1 \\ g_1 : x_1x_3 + y_0x_2 + y_0x_1 + y_0y_3 + y_0y_2 + x_0 + y_3 + y_1 + y_0 + 1 \\ g_2 : x_1x_2 + y_1y_2 + y_0x_2 + y_0x_0 + y_0y_1 + x_1 + y_2 \\ g_3 : x_0x_3 + y_0x_1 + y_0x_0 + y_0y_1 + x_3 + x_2 + x_1 + x_0 + y_2 + 1 \\ g_4 : x_0x_2 + y_1y_2 + y_0y_2 + x_3 + y_3 + y_2 + y_1 + y_0 + 1 \\ g_5 : x_0x_1 + y_0x_2 + y_0x_0 + y_0y_2 + x_2 + x_0 + y_2 + y_0 + 1 \\ g_6 : y_3x_3 + y_0x_1 + y_0y_3 + y_0y_1 + x_3 + x_2 + x_1 + x_0 + y_2 + 1 \\ g_7 : y_3x_2 + y_1y_2 + y_0x_2 + y_0x_1 + y_0x_0 + y_0y_3 + y_0y_2 + x_0 + y_2 + y_0 \\ g_8 : y_3x_1 + y_0x_1 + y_0x_0 + y_0y_3 + y_0y_1 + x_2 + y_3 + y_2 + y_0 + 1 \\ g_9 : y_3x_0 + y_0x_2 + y_0x_1 + x_3 + x_0 + y_1 + 1 \\ g_{10} : y_2x_3 + y_1y_2 + y_0x_1 + y_0y_2 + y_0y_1 + x_0 + y_3 + y_2 + y_0 \\ g_{11} : y_2x_2 + y_0x_2 + y_0x_0 + y_0y_3 + y_0y_1 + x_2 + y_2 + 1 \\ g_{12} : y_2x_1 + y_1y_2 + y_0x_2 + y_0y_3 + y_0y_2 + y_2 + y_0 \\ g_{13} : y_2x_0 + y_1y_2 + y_0y_3 + y_0y_2 + x_2 + 1 \\ g_{14} : y_2y_3 + y_1y_2 + y_0x_2 + y_0x_1 + y_0y_1 + x_2 + x_0 + y_3 + 1 \\ g_{15} : y_1x_3 + y_0x_2 + y_0x_0 + x_3 + x_0 + y_3 + y_1 + y_0 + 1 \\ g_{16} : y_1x_2 + y_1y_2 + y_0x_0 + y_0y_2 + y_0y_1 + x_3 + x_2 + x_0 + y_3 + y_2 \\ g_{17} : y_1x_1 + y_0x_2 + y_0y_3 + y_0y_1 + x_1 + y_1 + 1 \\ g_{18} : y_1x_0 + y_0x_1 + y_0y_3 + y_0y_1 + x_3 + x_2 + x_1 + x_0 + y_3 + y_2 + 1 \\ g_{19} : y_1y_3 + y_0x_2 + y_0x_0 + y_0y_3 + y_0y_2 + x_3 + x_2 + x_1 + x_0 + y_3 + y_2 + 1 \\ g_{20} : y_0x_3 + y_0x_0 + y_0y_3 + y_0y_1 + x_3 + x_0 + y_3 + y_1 + y_0 + 1 \end{array} \tag{3}$$

4.1 New Algorithm for Sparse Polynomial Equations

Consider Equation (1). The polynomials defined there form a variety $V = \mathbf{V}(f_0, f_1, f_2, f_3)$, which is

$$V = \{(x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3) : f_i(x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3) = 0 \text{ for } 0 \leq i \leq 3\}$$

If the ideal $I_{\mathcal{S}} = \mathbf{I}(V)$ is *radical*, then $I_{\mathcal{S}}$ would be the set of all polynomials that describe a relation between input bits and output bits of an S-box. A radical ideal is defined as follows.

Definition 9 ([18]). *An ideal I is **radical** if $f^m \in I$ for some integer $m \geq 1$ implies that $f \in I$.*

Naturally, radical of an ideal can be defined as follows:

Definition 10 ([18]). *Let $I \subset \mathbf{K}[x_1, \ldots, x_n]$ be an ideal.The **radical** of I is denoted by $\sqrt{I}$, is the set:*

$$\sqrt{I} = \{f : f^m \in I \textit{ for some integer } m \geq 1\}$$

Main property of a radical ideal generated by $\langle f_1, \ldots, f_s \rangle$, is [18]:

$$\mathbf{I}(\mathbf{V}(f_1, \ldots, f_s)) = \langle f_1, \ldots, f_s \rangle$$

We need further theorems that are given below.

Theorem 1 ([18]). *Let $\mathbf{K}$ be an algebraically closed field. If I is an ideal in $\mathbf{K}[x_1, \ldots, x_n]$, then*

$$\mathbf{I}(\mathbf{V}(I)) = \sqrt{I}$$

Based on Hilbert's Strong Nullstellensatz, if the ring $\mathbf{K}$ is an algebraic closed field, set of all polynomials that vanishes on a variety is defined by radical of ideal generated by set of polynomials. There is also a finite field version of Nullstellensatz stated by Gao in [26]:

Theorem 2 ([26]). *For an arbitrary finite field $\mathbb{F}_q$, let $J \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ be an ideal, then*

$$\mathbf{I}(\mathbf{V}(J)) = J + \langle x_1^q - x_1, \ldots x_n^q - x_n \rangle$$

Polynomials of form $x_i^q - x_i$ are called *field polynomials*. Let Q denotes the ideal generated by field equations. Based on Theorem 2, the set of polynomials describing S-box in $\mathbb{F}_2[x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3]$, i.e. $I_{\mathcal{S}}$, can be generated as follows:

$$I_{\mathcal{S}} = \langle f_0, f_1, f_2, f_3 \rangle + Q = \langle g_0, g_1, \ldots, g_{20} \rangle + Q \tag{4}$$

Therefore any set of polynomials that can generate ideal $I_{\mathcal{S}}$, can be considered as a set of polynomials describing the S-box.

Sparse Multivariate Quadratic Equations (SMQ)

Polynomials $h_0, h_1, h_2, h_3, h_4, h_5$ given below plus field polynomials describe the S-box of $SR(n, 2, 1, 4)$.

$$\begin{aligned}
h_0 &: y_1y_2 + y_1y_3 + x_1x_3 + x_1 + x_2x_3 + x_2 + x_3 + 1 \\
h_1 &: y_0y_3 + y_2 + x_0x_2 + x_0x_3 + x_1x_2 + x_1x_3 + x_2 + 1 \\
h_2 &: y_0y_3 + y_0 + y_1y_3 + y_3 + x_0x_1 + x_1 + x_3 \\
h_3 &: y_0y_1 + y_2y_3 + y_3 + x_0x_3 + x_1x_2 + x_3 \\
h_4 &: y_0y_1 + y_0 + y_1y_3 + y_1 + x_0x_3 + x_0 + x_1x_3 + 1 \\
h_5 &: y_0y_1 + y_0y_2 + y_1 + y_2y_3 + y_3 + x_0x_1 + x_2x_3 + 1
\end{aligned} \tag{5}$$

Consider the polynomials from Equation (5). Note that all polynomials are of degree 2 and interestingly enough all monomials of degree 2 are of the form x_ix_j, y_iy_j. This collection is very sparse

Algorithm 1 Algorithm for generating S-box equations

Require: M: set of allowed monomials
Require: G: reduced Gröbner basis of $I_{\mathcal{S}}$
Ensure: F: collection of equations representing S-box
  $F \leftarrow \emptyset$
  $g \leftarrow \{0\}$
  while $g \neq G$ **do**
    $p \leftarrow NextPoly(M)$
    if $LM(p) \notin LM(F)$ and $p \xrightarrow{G} 0$ **then**
      if $p \stackrel{g}{\not\to} 0$ **then**
        $F \leftarrow F \cup \{p\}$
        $g \leftarrow$ reduced Gröbner basis of F
      end if
    end if
  end while
  $F \leftarrow Sort(F)$
  for $p \in F$ **do**
    $g \leftarrow$ *Gröbner basis of* $F - \{p\}$
    if $p \to 0$ **then**
      $F \leftarrow F - \{p\}$
    end if
  end for
  return F

if you compare it to the collection given by Equation (3), We call this collection *sparse multivariate quadratic*. The collection defined by Equation (5) is generated by Algorithm 1 when the form of permitted monomials is restricted to quadratic monomials (x_ix_j and y_iy_j) and linear ones (x_i and y_i).

Algorithm 1 runs through several iterations. The intermediate collection F and its reduced Gröbner basis g is repeatedly updated. The algorithm iterates over all polynomials in $\mathbb{F}_2[x_0, \ldots, x_3, y_0, \ldots, y_3]$ that are linear combinations of monomials from the set M. At each step, we check whether p belongs to ideal $I_{\mathcal{S}}$. If $p \in I_{\mathcal{S}}$, we further check whether if $p \in g$. If $p \in g$, it means that this polynomial can be generated by previously found polynomials in g and it is discarded. Otherwise p is added to F and g is updated. At the end, some polynomials remain in the set F that can be generated by other polynomials in F. These polynomials can be removed from F. For example, we can sort polynomials by the number of monomials in order to remove the longest ones first. So far, we introduced four types of description of S-boxes. Our experiments show that a combination of FW and BW descriptions of S-boxes allow to solve polynomial equations for a cipher quicker. However, we have not been able to develop a mathematical argument that this is true for all S-boxes and ciphers.

In the following two Sections, we give the results of experiments on SR block cipher to verify our investigation.

5 SR Block Ciphers

The family of SR ciphers has been proposed by Cid et al. in [10]. It is a very useful tool to study and experiment with different algebraic attacks. Because of scalability, attacks can be tried for relatively weak versions of a AES-like block cipher and then try to identify the limits after which the attack becomes infeasible. An instance $SR(n, r, c, s)$ of the family is defined by specific parameters (n, r, c, s), where n is the number of rounds, r, c denotes the number of columns and rows of the matrix that represent an intermediate state and s denotes S-box size in bits. Additionally, two S-boxes are defined for 4-bit and 8-bit permutations. The 4-bit S-box is structurally similar to AES S-box. The 8-bit S-box is the same AES S-box. For more details, we refer the readers to [10].

A summary of cryptanalysis results for $SR(n, r, c, s)$ instances is given in Table 1. The first column shows an instance of the SR cipher, the second column gives the time complexity of best attack, the third points the algorithm used to solve a collection of equations, the forth displays monomial ordering used and the fifth refers to the work in which the results are published.

Table 1. Cryptanalysis of SR ciphers

Instance	Time	Tool	Note	Ref.
SR(10,1,1,4)	74.06	F4	**degrevlex**	[10]
SR(10,2,2,4)	1193.19	PolyBoRi	**lex**	[6]
SR(10,2,2,4)	190.58	MiniSat		[6]
SR(4,2,1,4)	0.63	PolyBoRi	**lex**	[6]
SR(10,2,1,4)	0.225	PolyBoRi	**lex**	[9]
SR(10,2,1,4)	10327.3	F4		[29]
SR(10,2,2,4)	1850.01	PolyBoRi	**lex**	[9]

We use a similar approach to the one used by Bulygin et al. [9]. However, instead of the **lex** ordering, we apply the **degrevlex** ordering, where secret key variables have lower order than other variables. The $SR(n, 2, 1, 4)$ cipher is described by the following equations:

$$\begin{cases} sbox(p_0 + k_{0,0}, x_{0,0}) & \\ sbox(p_1 + k_{0,1}, x_{0,1}) & \\ sbox(L_0(x_{i-1,0}, x_{i-1,1}) + k_{i,0}, x_{i,0}) & \text{for } i = 1, \ldots, n \\ sbox(L_1(x_{i-1,0}, x_{i-1,1}) + k_{i,1}, x_{i,1}) & \text{for } i = 1, \ldots, n \\ c_0 + L_0(x_{n,0}, x_{n,1}) + k_{n,0} & \\ c_1 + L_1(x_{n,0}, x_{n,1}) + k_{n,1} & \\ sbox(k_{i,0}, k_{i+1,0} + rc_i) & \text{for } i = 0, \ldots, n-1 \\ sbox(k_{i,1}, k_{i+1,1} + rc_i) & \text{for } i = 0, \ldots, n-1 \end{cases} \tag{6}$$

In Equation (6), $sbox()$ gives a collection of equations that defines input/output relation of the S-box. Arguments of $sbox()$ are vectors in the polynomial ring $\mathbb{F}_2[]$. The constant rc_i is added to the ith round of the cipher. Parameters p_0 and p_1 denote first and second nibble of the plaintext, x_i denotes an intermediate vector of variables for the ith round and $x_{i,j}$ denotes the j-th nibble of x_i. $L_0()$ and $L_1()$ denote multiplication of first and second rows of the MixColumn matrix.

6 Experiments

6.1 Gröbner Basis Algorithms

We study an impact the S-box representation (*FW, BW, MQ* or *SMQ* - see Section 4) has on the time needed to solve a collection of equations for the whole cipher. We apply three software packages for computation of Gröbner basis. We also experiment with CryptoMiniSat SAT-solver [30]. The tools are employed through the SAGE computer algebra system, version 6.7-x86_64 [32]. Experiments are conducted on a desktop computer with 16 GB of RAM, clocked by a Core i7 4770 processor and running a single core.

For the computation of Gröbner basis, we experiment with the $SR(n, 2, 1, 4)$ cipher. For each experiment run, we generate a collection of polynomial equations for 50 instances of cipher with randomly chosen plaintexts and keys. The time needed to solve the 50 instances is used to calculate

Table 2. Average running time, in seconds, for computation of Gröbner basis

N_r	PolyBoRi			Singular			FGb		
	FW	MQ	FWBW	FW	MQ	FWBW	FW	MQ	FWBW
5	24.67	71.89	12.91	27.24	59.66	7.85	215.18	212.84	28.39
6	58.48	169.73	26.06	68.83	139.19	19.42	422.36	456.27	58.51
7	145.44	405.5	40.49	131.89	275.31	24.77	717.76	791.79	86.95
8	179.19	927.3	57.98	274.92	$\perp$	55.67	$\perp$	$\perp$	142.75
9	256.14(45)	1961.4	83.98	404.98(36)	$\perp$	78.8	$\perp$	$\perp$	202.24
10	259.65(46)	3314.03(49)	116.95	205.5(23)	$\perp$	123.8	$\perp$	$\perp$	296.83

the average. Gröber basis computation is done using the **degrevlex** monomial ordering, in which key variables are the lowest in the order. Table 2 presents the average running time in seconds for computing the Gröbner bases using software packages PolyBoRi, Singular and FGb. *FW, MQ, SMQ* and *FWBW* denote the S-box representations defined by Equations (1), (3), (5) and combination Equations (1) and (2), respectively. N_r stands for the number of cipher rounds. The numbers in parentheses show the number of solved instances (out of 50). The entries with $\perp$ indicate cases where computation of Gröbner basis has failed due to memory limits. With the FGb, we set a time limit of 1000 seconds for experiment runs and the symbol $\perp$ may indicate a failure due to time limit. In case of PolyBoRi tool, we set the `selection_size` argument to 10000, which means at each reduction step, at most 10000 polynomials are selected for F4 algorithm reduction step.

When using the FGb tool, the collection of polynomials is inter-reduced before feeding into the tool. Our observations indicate that this leads to faster computation of the Gröbner basis. We do not, however, notice any significant impact on processing time when other tools are used.

We note that the S-box representation has a significant impact on the time needed to solve a cipher instance. In particular, we make the following observations. If the S-box is represented as

- MQ polynomials, then collections of polynomials derived for $SR(n, 2, 1, 4)$ with $n > 8$ could not be solved withSingular or FGb. With PolyBoRi, the running time is much bigger than the running time for FWBW representation. Our experiments suggest that for 4-bit S-box based block ciphers, it would be better to apply an representation of S-boxes that is not quadratic. This finding is quite counter-intuitive,
- FW polynomials, then best results for computation of Gröbner basis are obtained using the PolyBoRi library. An advanced version of the SlimGB algorithm that is optimized for manipulating Boolean polynomials, is used in PolyBoRi,
- FWBW polynomials, then computation of Gröbner basis by PolyBoRi is greatly improved. However, when we used Singular, we observed an improvement of both running time and memory requirements. This enables us to solve instances of $SR(10, 2, 1, 4)$ with minimal memory requirements. When using FGb, the results are not improved (when comparing to PolyBoRi and Singular). Nevertheless, we are able to solve instances of $SR(10, 2, 1, 4)$. Note that Cid et al. in [10] managed to compute Gröber basis of $SR(3, 2, 1, 4)$ over $\mathbb{F}_{2^4}$ in 519.92 seconds using the F4 algorithm,
- SMQ polynomials, then computation of Gröbner basis (even for $SR(5, 2, 1, 4)$) consumes a lot of memory. In fact, we run out of memory during our experiments and have failed to solve even a single instance. This fact suggests that a description of S-box plays a significant role in solving instances of the SR cipher. The choice of an algebraic tool seems to be less important.

The most time consuming part of Gröbner basis computation algorithms is the reduction of a polynomial or multiple polynomials modulo an intermediate basis. All of the tools that is being used in experiments, implement a variant of F4 algorithm for computation of Gröbner basis. The F4 algorithm, reduce the polynomials modulo an intermediate basis by Gaussian elimination (or by reducing to row echelon form) of the corresponding matrices. Hence, the larger portion of running time is spent on the Gaussian elimination of matrices. To have more insight in how the algebraic representation of S-boxes affect the running time, Table 3, indicate dimensions of the largest matrix that generated by PolyBoRi for computation of Gröbner basis. The results are related to a sample of $SR(n, 2, 1, 4)$ for different number of rounds, where plaintext and key both are set to 0x55.

Table 3. Dimensions of largest matrices generated by PolyBoRi

N_r	MQ	FWBW
5	48476×41218	19330×13293
6	66529×63074	22776×14071
7	90212×92552	18891×22100
8	142596×142467	28121×32174
9	175211×181603	31326×36192
10	214416×228497	46249×51847

From the Table 3, MQ representation of S-boxes yields larger matrices in comparison with the FWBW representation. In Figures 1 and 2, dimensions of matrices generated during computation Gröbner basis for a sample of $SR(10, 2, 1, 4)$ is plotted, for MQ and FWBW representations, respectively. For MQ representation, most matrices have dimensions of order 10^5 and for FWBW representation, the dimensions are of order 10^4. Therefore FWBW representation improves efficiency of algebraic attacks in two dimensions: 1- the running time for computation of Gröbner basis and 2- the memory requirement for storing matrices.

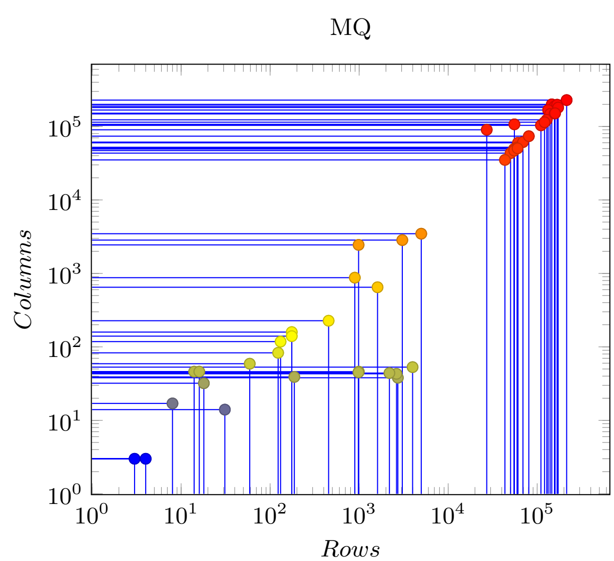

Fig. 1. Dimension of matrices generated by PolyBoRi, for a single instance of $SR(10, 2, 1, 4)$ with MQ representation of S-boxes

Experiments seem to confirm the fact that an "appropriate" representation of S-boxes has much bigger impact than the algebraic tool used to solve an instance.

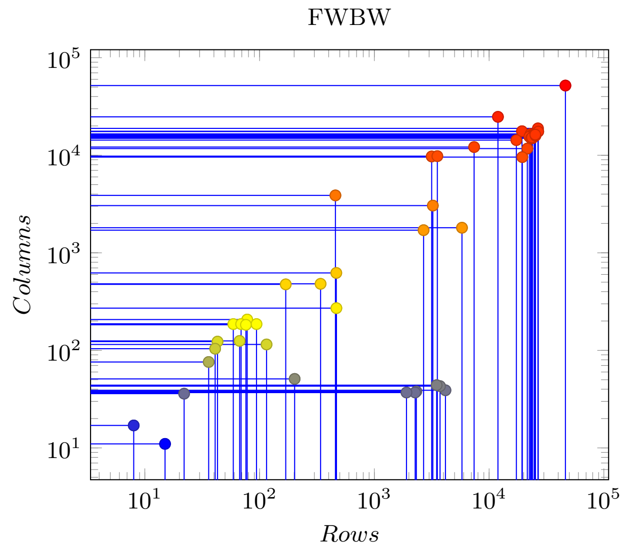

Fig. 2. Dimensions of matrices generated by PolyBoRi, for a single instance of $SR(10, 2, 1, 4)$ with FWBW representation of S-boxes

6.2 SAT Solvers

We also consider an effect of S-box representation on efficiency of solving a collection of polynomial equations using SAT-solvers. Cipher instances of $SR(10, 2, 1, 4)$ can be solved in less than a second. Next we consider instances of $SR(n, 2, 2, 4)$, where the number of rounds is not fixed. The collection of polynomials equations describing the cipher is presented below:

$$\begin{cases} sbox(p_0 + k_{0,0}, x_{0,0}) & \\ sbox(p_1 + k_{0,1}, x_{0,1}) & \\ sbox(p_2 + k_{0,2}, x_{0,2}) & \\ sbox(p_3 + k_{0,3}, x_{0,3}) & \\ w_{i,0} + L_0(x_{i-1,0}, x_{i-1,3}) + k_{i,0} & \text{for } i = 1, \ldots, n \\ sbox(w_{i,0}, x_{i,0}) & \text{for } i = 1, \ldots, n \\ w_{i,1} + L_1(x_{i-1,0}, x_{i-1,3}) + k_{i,1} & \text{for } i = 1, \ldots, n \\ sbox(w_{i,1}, x_{i,1}) & \text{for } i = 1, \ldots, n \\ w_{i,2} + L_0(x_{i-1,2}, x_{i-1,1}) + k_{i,2} & \text{for } i = 1, \ldots, n \\ sbox(w_{i,2}, x_{i,2}) & \text{for } i = 1, \ldots, n \\ w_{i,3} + L_1(x_{i-1,2}, x_{i-1,1}) + k_{i,3} & \text{for } i = 1, \ldots, n \\ sbox(w_{i,3}, x_{i,3}) & \text{for } i = 1, \ldots, n \\ c_0 + L_0(x_{n,0}, x_{n,3}) + k_{n,0} & \\ c_1 + L_1(x_{n,0}, x_{n,3}) + k_{n,1} & \\ c_2 + L_0(x_{n,2}, x_{n,1}) + k_{n,0} & \\ c_3 + L_1(x_{n,2}, x_{n,1}) + k_{n,1} & \\ k_{i,3} + k_{i+1,1} + k_{i+1,3} & \\ k_{i,2} + k_{i+1,0} + k_{i+1,2} & \\ sbox(k_{i,3}, k_{i,0} + k_{i+1,0} + rc_i) & \text{for } i = 0, \ldots, n-1 \\ sbox(k_{i,2}, k_{i,1} + k_{i+1,1}) & \text{for } i = 0, \ldots, n-1 \end{cases} \tag{7}$$

This collection is converted to CNF using the method proposed by Bard et al. [3]. Resulting SAT instances are solved with the CryptoMiniSAT solver. The obtained results are reported in Tables

4 and 5. In Table 5, #var denotes the number of variables and #cls points the average number of clauses after conversion to CNF. For the FW representation, instances of $SR(10, 2, 2, 4)$ can

Table 4. Average running time of SAT-Solver, in seconds

N_r	CryptoMiniSAT			
	FW	FWBW	MQ	SMQ
5	1.59	3.63	10.07	6.45
6	1.96	3.49	12.48	7.87
7	2.14	3.98	17.36	8.85
8	2.19	4.6	19.24	10.53
9	2.63	3.61	25.28	13.32
10	2.69	4.37	25.68	12.49

Table 5. Number of variable and average number of clauses for different descriptions conversion

N_r	#var	#cls			
		FW	FWBW	MQ	SMQ
5	256	1509	4434	39602	9577
6	304	1810	5316	46594	10656
7	352	2109	6199	44132	12332
8	400	2406	7074	60884	13567
9	448	2704	7984	51304	14780
10	496	3007	8846	58886	16343

be solved in 2.69 seconds on average. This is a significant improvement compared to the results reported by Bulygin et al. [9]. They were able to solve $SR(10, 2, 2, 4)$ instances in 1850.01 seconds using PolyBoRi and in 190 seconds using MiniSat solver. The FWBW representation has improved efficiency of Gröbner basis computations. Unfortunately, SAT-solvers are working less efficient than for the FW representation. One can guess that this may be attributed to an increase of the number of variables and clauses. For the MQ representation, solving $SR(10, 2, 2, 4)$ takes on average 25.68 seconds. This is a much higher time complexity than for both FW and FWBW representations. This suggests that the MQ representation of S-boxes does not improve time complexity of algorithms to solve SR ciphers for 4-bit S-boxes. For the SMQ representation, solving instances of $SR(10, 2, 2, 4)$ takes 12.49 seconds. In Table 5, number of clauses generated for each system is reported. From the Table, the MQ representation generates the largest number of clauses, and hence results in more running time, while the FW representation yields least number of clauses and hence the lowest running time.

6.3 Lightweight Ciphers

We further study the impact of the S-box representation on efficiency of solving round-reduced lightweight ciphers such as LBlock [33], PRESENT [4] and MIBS [27]. The summary of best algebraic attack results published so far is given in Table 6, where g denotes the number of guessed key bits and *Data* denotes the number of plaintexts needed in the attack.

Courtois et al. [17] and Susil et al. [31] mount their attacks on the LBlock cipher using the ElimLin algorithm [14]. Courtois et al. [17] choose plaintexts at random. In contrast, Susil et al.

Table 6. Algebraic attacks on LBlock, PRESENT and MIBS, w.r.t rounds

N_r	g	*RunTime*	*Data*	*note*	*work*
			LBlock		
8	32	907 s	6 KP	ElimLin	[17]
8	32	$\perp$	6 KP	PolyBoRi	[17]
8	0	Not Reported	8 CP	ElimLin	[31]
10	0	Not Reported	16 CP	ElimLin	[31]
11	0	10106 s	128 CP	PolyBoRi-FWBW	this
			PRESENT		
5	35	1.845 h	16 KP	ElimLin	[28]
5	35	$\perp$	16 KP	PolyBoRi	[28]
6	0	2009.03 s	32 CP	PolyBoRi-FWBW	this
			MIBS		
4	20	0.137 h	32 KP	ElimLin	[17]
4	20	$\perp$	32 KP	PolyBoRi	[17]
5	16	0.137 h	6 KP	MiniSAT 2.0	[17]
5	16	$\perp$	6 KP	PolyBoRi	[17]
6	0	68.46 s	12 CP	PolyBoRi-FWBW	this

[31] select plaintexts via a successful cube attack [21]. This approach allows the authors to solve the cipher instances by using the ElimLin algorithm and without guessing key bits. Analysis of MIBS and PRESENT ciphers is given in [17] and [28].

In our experiments, plaintexts are chosen by a strategy similar to the one applied by both Faugere et al. in [23] and by Susil et al. in [31], i.e correlated messages. Unlike [31] in our strategy, the i-th plaintext is the 64 bit representation of number i. The results obtained for the LBlock, PRESENT and MIBS ciphers are presented in Table 7 where the following notations is used. The number of solved instances, is denoted by N_S. The average running time for computation of Gröbner basis with PolyBoRi is denoted by T_G. The column *Data* denotes the number chosen plaintexts used to mount the attack.

We note that we are getting better results when we apply SlimGB with FWBW. However, there is a downside – we are running out of memory quicker. This is the reason why we have chosen PolyBoRi as it provides an optimized data structure for memory efficient manipulation of Boolean polynomials. In contrast with our earlier experiments with the SR cipher, we set the `faguere` argument to **False** in order to disable the use of linear algebra for reduction of polynomials. This due the fact that large dimensions of matrices that are generated for the reduction step, causes to get quickly out of memory.

We applied FWBW representation for the LBlock S-boxes. We are able to break the cipher with 8 rounds, knowing 3 chosen plaintext/ciphertext observations. This is a better result than described in [31], where the attack needs 8 plaintext/ciphertext pairs. The 9-round LBlock is broken with 4 observations. Using such a simple startegy for selection of plaintext/ciphertext pairs, we are able to break the cipher with 10 rounds with 12 pairs, which is better than the one published by Susil et al in [31]. With 128 plaintexts, it is possible to break 11 rounds LBlock cipher. According to our best knowledge, this is the first algebraic attack on 11-round LBlock.

Considering Table 7 for PRESENT, the FWBW representation leads to a successful attack on the 6-round cipher with 32 chosen plaintext/ciphertext pairs and no key guessing. This is a better result than the one reported by Jorge et al in [28], which needs 16 plaintext/ciphertext pairs and a guess of 35 key bits on five rounds of PRESENT.

Table 7. Algebraic attacks on LBlock, PRESENT and MIBS using FWBW description of S-boxes

N_r	T_G	*Data*	N_S
	LBlock		
8	1.18 s	3 CP	50/50
9	8.7 s	4 CP	50/50
10	224.2	12 CP	50/50
11	10106 s	128 CP	10/10
	PRESENT		
4	3.25 s	4 CP	50/50
5	23.57 s	6 CP	50/50
6	12048.28 s	32 CP	14/15
	MIBS		
4	10.17 s	5 CP	50/50

Considering 4-round MIBS cipher, the FWBW yields an attack that needs 5 chosen plaintext/ciphertext pairs only. This outperforms the previous result by Courtois et al. [17], which needs 32 plaintext/ciphertext pairs and a guess of 20 key bits. The run time of attack is reduced to 10.17 seconds.

6.4 Effect of Pre-processing Polynomials

During our experiments on MIBS, we noticed that if we pre-process equations, the efficiency of computation of Gröbner basis increases significantly. We found experimentally that the inter-reduction of polynomials related to the first half of the cipher, would enable us to break up to 6 rounds of MIBS on our desktop computer. While without this pre-processing, we quickly get out of memory during the computation of Gröbner basis for 5 rounds of MIBS cipher. We call this pre-processing, the *partial inter-reduction*. The results for PRESENT and MIBS are presented in Table 8. In the Table, the average running time for inter-reduction is denoted by T_I. For the LBlock cipher, the improvement is moderate, therefore we did not report the results.

Table 8. Algebraic attacks on PRESENT and MIBS using FWBW description of S-boxes and preprocessing

N_r	T_I	T_G	*Data*	N_S
		PRESENT		
4	0.68	3.6 s	4 CP	50/50
5	4.66	23.7 s	6 CP	50/50
6	82.21	1926.82 s	32 CP	21/21
		MIBS		
4	0.94	1.07 s	4 CP	50/50
5	6.01	7.73 s	5 CP	50/50
6	35.68	32.78 s	12 CP	50/50

Considering the PRESENT cipher, partial inter-reduction reduces the running time for computation of Gröbner basis of polynomials related to 6 rounds of PRESENT, to 1926.82 s in average. For the MIBS cipher, partial inter-reduction not only reduces the running time for attacking the

4-round version to 1.07 in average, but allowed us to break two more rounds. For 5 and 6 rounds of MIBS, the cipher breaks using 5 and 12 chosen plaintext/ciphertext pairs, respectively. According to our best knowledge, this is the first algebraic attack on 6-round MIBS.

7 Conclusion

We study the effect of S-box representation on the efficiency of algebraic analysis of block ciphers. In general, algebraic analysis takes two stages. In the first stage, a cipher is described by a collection of polynomials. In the second stage, the collection is solved using an "appropriate" algorithm. For AES-like block ciphers, the strength of encryption relies on cryptographic properties of S-boxes (such as non-linearity, avalanche criterium, algebraic degree to name a few).

Our study shows that S-box representation has a significant impact on efficiency of algebraic attacks. In particular, the FWBW representation seems to be most effective as confirmed by our numerous experiments. We have managed to break the $SR(10,2,1,4)$ cipher using the computer algebra softwares Singular and FGb. We also presented first algebraic attacks on 11 rounds LBlock and 6 rounds of PRESENT and MIBS using FWBW representation with PolyBoRi library. The following list gives suggestions as to future/open research directions:

- Generalization of our study for 8-bit S-boxes. A related problem is the choice of the field $GF(2^n)$, which can selected to represent S-box polynomials.
- Investigation of S-box polynomial representations. One would hope to discover "ideal" representations (or perhaps the ideal one), which maximise the efficiency of algebraic attacks. This would give a better understanding of algebraic properties of ideal representations and perhaps make a connection between S-box representations and their cryptographic properties.
- Extension of experiments done in the work. It would be very beneficial to cover a wide range of ciphers to generalise our findings. Another possible extension is to use more computational resources to determine limits of algebraic analysis.

References

1. Some algebraic description for various s-boxes. http://www.nicolascourtois.com/equations/block/sboxes/misc_sboxes.ZIP, accessed: 2017-01-31
2. Some algebraic description for various s-boxes. http://http://www.nicolascourtois.com/equations/block/gost/gost_boxes.ZIP, accessed: 2017-01-31
3. Bard, G.V., Courtois, N.T., Jefferson, C.: 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), http://eprint.iacr.org/2007/024
4. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Cryptographic Hardware and Embedded Systems-CHES 2007, pp. 450–466. Springer (2007)
5. Brickenstein, M.: Slimgb: Gröbner bases with slim polynomials. Revista Matemática Complutense 23(2), 453–466 (Dec 2009)
6. Brickenstein, M., Dreyer, A.: PolyBoRi: A framework for Gröbner-basis computations with Boolean polynomials. Journal of Symbolic Computation 44(9), 1326–1345 (Sep 2009)
7. Buchberger, B.: Bruno Buchberger's PhD thesis 1965: An algorithm for finding the basis elements of the residue class ring of a zero dimensional polynomial ideal. Journal of Symbolic Computation 41(3–4), 475–511
8. Buchberger, B.: A Criterion for Detecting Unnecessary Reductions in the Construction of Gröbner Bases. In: Proceedings of the International Symposiumon on Symbolic and Algebraic Computation. pp. 3–21. EUROSAM '79, Springer-Verlag
9. Bulygin, S., Brickenstein, M.: Obtaining and Solving Systems of Equations in Key Variables Only for the Small Variants of AES. Mathematics in Computer Science 3(2), 185–200 (Apr 2010)

10. Cid, C., Murphy, S., Robshaw, M.J.B.: Small Scale Variants of the AES. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption, pp. 145–162. No. 3557 in Lecture Notes in Computer Science, Springer Berlin Heidelberg
11. Cid, C.: Some Algebraic Aspects of the Advanced Encryption Standard. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) Advanced Encryption Standard – AES, pp. 58–66. No. 3373 in Lecture Notes in Computer Science, Springer Berlin Heidelberg
12. Cid, C., Leurent, G.: An Analysis of the XSL Algorithm. In: Roy, B. (ed.) Advances in Cryptology - ASIACRYPT 2005, pp. 333–352. No. 3788 in Lecture Notes in Computer Science, Springer Berlin Heidelberg
13. Courtois, N., Mourouzis, T., Hulme, D.: Exact logic minimization and multiplicative complexity of concrete algebraic and cryptographic circuits. International Journal On Advances in Intelligent Systems 6(3), 165–176 (2013)
14. Courtois, N.T., Bard, G.V.: Algebraic Cryptanalysis of the Data Encryption Standard. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007, pp. 152–169. No. 4887 in Lecture Notes in Computer Science, Springer Berlin Heidelberg
15. Courtois, N.T., Castagnos, G., Goubin, L.: What do des s-boxes say to each other ? Cryptology ePrint Archive, Report 2003/184 (2003), `http://eprint.iacr.org/2003/184`
16. Courtois, N.T., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) Advances in Cryptology — ASIACRYPT 2002, pp. 267–287. No. 2501 in Lecture Notes in Computer Science, Springer Berlin Heidelberg
17. Courtois, N.T., Sepehrdad, P., Sušil, P., Vaudenay, S.: ElimLin Algorithm Revisited. In: Canteaut, A. (ed.) Fast Software Encryption, pp. 306–325. No. 7549 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2012)
18. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms. Undergraduate Texts in Mathematics, Springer New York (2007)
19. Daemen, J., Rijmen, V.: AES proposal: Rijndael, in NIST AES Proposal (1998)
20. Decker, W., Greuel, G.M., Pfister, G., Schönemann, H.: Singular 3-1-7 — A computer algebra system for polynomial computations. `http://www.singular.uni-kl.de` (2015)
21. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, pp. 278–299. No. 5479 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2009)
22. Faugère, J.C.: FGb: A Library for Computing Gröbner Bases. In: Fukuda, K., van der Hoeven, J., Joswig, M., Takayama, N. (eds.) Mathematical Software – ICMS 2010, pp. 84–87. No. 6327 in Lecture Notes in Computer Science, Springer Berlin Heidelberg
23. Faugère, J.C., Perret, L.: Algebraic Cryptanalysis of Curry and Flurry Using Correlated Messages. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Information Security and Cryptology, pp. 266–277. No. 6151 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2010)
24. Faugére, J.C.: A new efficient algorithm for computing Gröbner bases (F4). Journal of Pure and Applied Algebra 139(1–3), 61–88 (Jun 1999)
25. Fuhs, C., Schneider-Kamp, P.: Synthesizing Shortest Linear Straight-Line Programs over GF(2) Using SAT. In: Strichman, O., Szeider, S. (eds.) Theory and Applications of Satisfiability Testing – SAT 2010. pp. 71–84. Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jul 2010)
26. Gao, S.: Counting Zeros over Finite Fields with Gröbner Bases. Master's thesis, Carnegie Mellon (2009), `http://www.andrew.cmu.edu/user/avigad/Students/gao_ms_thesis.pdf`
27. Izadi, M., Sadeghiyan, B., Sadeghian, S.S., Arabnezhad Khanooki, H.: MIBS: A New Lightweight Block Cipher. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) Cryptology and Network Security, pp. 334–348. No. 5888 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2009)
28. Jr, J.N., Sepehrdad, P., Zhang, B., Wang, M.: Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) Cryptology and Network Security, pp. 58–75. No. 5888 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Dec 2009)
29. Miolane, C.: Block Cipher Analysis. Ph.D. thesis, Technical University of Denmark (DTU) (2009), `http://orbit.dtu.dk/services/downloadRegister/5009704/thesis_cvm_v1.pdf`
30. Soos, M.: SAT-solver cryptominisat, Version 2.9.0, January 20 2011
31. Sušil, P., Sepehrdad, P., Vaudenay, S.: On Selection of Samples in Algebraic Attacks and a New Technique to Find Hidden Low Degree Equations. In: Susilo, W., Mu, Y. (eds.) Information Security and Privacy, pp. 50–65. No. 8544 in Lecture Notes in Computer Science, Springer International Publishing (Jan 2014)

32. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 6.7). `http://www.sagemath.org`
33. Wu, W., Zhang, L.: LBlock: A Lightweight Block Cipher. In: Lopez, J., Tsudik, G. (eds.) Applied Cryptography and Network Security, pp. 327–344. No. 6715 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Jan 2011)