IET Information Security

Research Article

IET Journals
The Institution of Engineering and Technology

S-boxes representation and efficiency of algebraic attack

ISSN 1751-8709
Received on 7th May 2018
Revised 20th January 2019
Accepted on 4th March 2019
E-First on 3rd April 2019
doi: 10.1049/iet-ifs.2018.5201
www.ietdl.org

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

[1]*Department of Computer Engineering & Information Technology, Amirkabir University of Technology, Tehran, Iran*
[2]*Data61, CSIRO, Sydney, Australia*
[3]*Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland*
✉ *E-mail: basadegh@aut.ac.ir*

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 study investigates some different S-box representations and their effect on complexity of algebraic attacks. In particular, the authors 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. They show that the *SR*(10,2,1,4) cipher can be broken with algebraic cryptanalysis 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 some improvements have been achieved on the algebraic cryptanalysis of LBlock, PRESENT and MIBS light-weight block ciphers. The authors' study and experiments confirm 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 with quadratic S-box representation.

1 Introduction

The winner of the advanced encryption standard (AES) competition was the Rijndael block cipher [1]. Soon after the announcement, NIST approved it as a new AES. An elegant and simple algebraic structure of AES gives a strong motivation towards the 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 [2], where an extended sparse linearisation (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 [3, 4]. Note that the system of relations described the AES block cipher with 128-bit keys that consists of 8000 non-linear relations in 1600 variables. It is impossible to verify experimentally the claimed complexity of the XSL attack. Cid *et al.* [5] propose a family of scalable versions of AES to study and experiment with cryptanalysis of AES. Cid *et al.* [5] 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 [6]. To our knowledge, the best algebraic cryptanalysis results for this family of ciphers are reported in [7].

Another interesting approach in solving systems of polynomial equations over $\mathbb{F}_2$ is application of SAT-solvers. Courtois and Bard [8] used this approach to analyse data encryption standard (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 an appropriate algorithm (XL, F4, SAT solvers etc.).

There exists virtually infinitely many ways to algebraically describe a block cipher. An adversary (cryptanalyser) would like to form these algebraic relations in such a way that they can be solved as fast as possible. In [2], 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 [9] discuss different types of equations that relate input and output of DES S-boxes. Courtois and Bard break six and four rounds of the block cipher DES using quadratic and cubic description of S-boxes, with SAT solver and ElimLin algorithm, respectively (see [8]). The quadratic relations of S-boxes in [8] are derived from an efficient hardware implementation of the DES S-boxes. Alternative relations for various S-boxes are presented in [10, 11].

1.1 Problem

A form of S-box description may have an impact on the efficiency of algebraic cryptanalysis as noted by Courtois and Bard [8]. A method proposed in [12] generates the 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. The purpose of this work is twofold. First, we present an alternative S-box representation. Second, we investigate the impact of S-box descriptions on the efficiency of algebraic cryptanalysis with Gröbner basis.

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

In this work, we investigate two algorithms for solving the obtained 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) defined by Cid *et al.* in [5]. In particular, we investigate the impact of some algebraic descriptions of S-boxes on the efficiency of algebraic attacks. We consider the relations 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 packages SINGULAR [14] and FGb [15], where $SR(n, r, c, e)$ is a AES variant with n rounds, r is the number of rows, c is the number of columns and e is the size of the word in bits. The analysis reported in [5, 7] uses the F4 algorithm [6] and SINGULAR. However, for the number of rounds $n \geq 5$, the analysis fails. We also give an algorithm for finding very sparse relations representing the SR S-box. Interestingly enough, application of the sparse quadratic relations in our cryptanalysis gives worse results. We are able to solve system of polynomial relations for $SR(10, 2, 2, 4)$ in 2.69 s on the average using CryptoMiniSat [16]. This result is better than the one reported in [17]. We also study lightweight block ciphers LBlock [18], PRESENT [19] and MIBS [20] and analyse their strength against algebraic attacks with the proposed description of S-boxes.

1.2 Results

The main results of the work are as follows:

1. Developing a framework that allows us to find different representations of S-boxes. This includes a new algorithm for generating sparse polynomial systems.
2. 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.
3. Showing that using both *forward–backward (FWBW)* polynomial relations for S-boxes allows us to break 11-round version of the LBlock cipher using the Gröbner basis tools.
4. Demonstrating that using both *FWBW* polynomial relations for S-boxes allows us to break six-round version of the PRESENT cipher using the Gröbner basis tools.
5. Presenting the first algebraic cryptanalysis of six rounds of the MIBS cipher.

1.3 Organisation of the paper

The paper is organised as follows. In Section 2, we present an algebraic background for computation of Gröbner basis over finite fields. Then, in Section 3, 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 4, we review the small-scale variants of AES and published results. We analyse small-scale variants of AES and present experimental results in Section 5. In that section, we also report some attacks with our approach on three light-weight block ciphers, and compare the attacks with some published results so far. We give conclusions and future research directions in Section 6. In Appendix 1, we give more investigation on SMQ representation. In Appendix 2, we discuss algebraic attacks based on SAT-solvers.

2 Preliminaries

We adopt notations and definitions from [21]. In this paper, polynomial ring in variables $x_1, x_2, \ldots, x_n$ with coefficients in $\boldsymbol{K}$ is denoted by $\boldsymbol{K}[x_1, x_2, \ldots, x_n]$. 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 [21].

Definition 1: Given a field $\boldsymbol{K}$ and a positive integer *n, we define the n*-dimensional *Affine Space* over $\boldsymbol{K}$ to be the set

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

Definition 2: Let $\boldsymbol{K}$ be a field, and let $f_1, \ldots, f_s$ be polynomials in $\boldsymbol{K}[x_1, \ldots, x_n]$. Then we set

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

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

Note that $\boldsymbol{V}(f_1, \ldots, f_s)$ 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 [22].

Definition 3: A subset $I \subseteq \boldsymbol{K}[x_1, \ldots, x_n]$ is an ideal if it satisfies:

1. I is an additive subgroup of $\boldsymbol{K}[x_1, \ldots, x_n]$.
2. $\forall f \in I$ and $\forall h \in \boldsymbol{K}[x_1, \ldots, x_n]$, then $h \cdot f \in I$.

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

Definition 4: Let $f_1, \ldots, f_s$ be polynomials in $\boldsymbol{K}[x_1, \ldots, x_n]$ [21]. Then we set

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

Lemma 1: If $f_1, \ldots, f_s \in \boldsymbol{K}[x_1, \ldots, x_n]$, then $\langle f_1, \ldots, f_s \rangle$ is an ideal of $\boldsymbol{K}[x_1, \ldots, x_n]$ [21]. We call $\langle f_1, \ldots, f_s \rangle$ the *ideal generated by* $f_1, \ldots, f_s$.

Hilbert basis theorem [21] states that any *ideal* in $\boldsymbol{K}[x_1, \ldots, x_n]$ can be generated by a finite set of polynomials. The following two [21] definitions explain the relation of *ideal* and *variety*.

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

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

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

Definition 6: Let $V \subset \boldsymbol{K}^n$ be an affine variety [21]. Then *ideal of* V will be denoted by $\boldsymbol{I}(V)$ the set

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

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 [23]. To compute Gröbner basis, an ordering is set 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. Gröbner basis of an ideal I is defined as follows [21].

Definition 7: 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 $f \in I$. The following defines a reduced Gröbner basis for an ideal I [21].

Definition 8: 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 [21].

Proposition 1: 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 sets 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 proposition explains how this can be done when relations are defined over $\mathbb{F}_2$ [24].

Proposition 2: 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 $\boldsymbol{V}(I)$ in $\mathbb{F}_2$. Particular useful cases are:

- $\boldsymbol{V}(I) = \varnothing$ iff $G = \langle 1\rangle$.
- $\boldsymbol{V}(I)$ 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 [25], which is easy to implement but not efficient. Faugere [6] proposed F4 algorithm, which uses linear algebra to do the reduction. A highly optimised version of algorithm F4 is implemented in FGb [15]. SlimGB [26] is a variant of Buchberger algorithm. Polynomials are ordered 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 [14]. There is also an implementation of SlimGB with highly optimised data structures for representing Boolean polynomials. It is available as POLYBORI library in C++ [17].

3 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 non-linear 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.

In many block cipher designs, S-boxes are the only non-linear component. S-boxes are functions from $\boldsymbol{F}_2^n$ to $\boldsymbol{F}_2^m$, where in permutation S-boxes $m = n$. Design of cryptographically strong S-boxes is an important part of cryptography. Again because of efficiency aspect, small S-boxes look more attractive. Note however that any permutation 2×2 S-box is linear/affine. In practice, permutation $n \times n$ S-boxes are used when $2 < n \leq 8$.

3.1 Forward (FW) equations

Permutation $n \times n$ S-boxes are permutation functions that 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 $SR(n, 2, 1, 4)$ [5] can be represented by four equations of degree 3 as follows:

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

We call polynomials such as (1) *Forward equations* or *FW equations* for short.

3.2 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 $SR(n, 2, 1, 4)$ [5], we get the following relations:

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

3.3 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 and Pieprzyk [2] observe that the AES S-box has a very compact and low degree representation for *MQ* equations. For the permutation 4×4 S-box from $SR(n, 2, 1, 4)$ [5], we can get 21 polynomial equations of degree 2. They are:

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

3.4 New algorithm for sparse polynomial equations

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

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

If the *ideal* $I_{\mathcal{S}} = \boldsymbol{I}(V)$ 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 [21].

Definition 9: An ideal I is **radical** if $f^m \in I$ for some integer $m \ge 1$ implies that $f \in I$.

Naturally, radical of an *ideal* can be defined as follows [21]:

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

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

Main property of a radical *ideal* generated by $\langle f_1, \ldots, f_s\rangle$ is [21]

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

We need further theorems that are given below.

Theorem 1: Let $\boldsymbol{K}$ be an algebraically closed field. If I is an ideal in $\boldsymbol{K}[x_1, \ldots, x_n]$, then

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

Based on Hilbert's Strong Nullstellensatz, i.e. theorem 1 [21], if the ring $\boldsymbol{K}$ 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 [24].

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

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

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.

3.5 Sparse multivariate quadratic (SMQ)

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

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

Consider the polynomials from (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 if you compare it to the collection given by (3). We call this collection *sparse multivariate quadratic*. The collection defined by (5) is generated by Algorithm 1 (see Fig. 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 (Fig. 1) runs through several iterations. The intermediate collection F and its reduced Gröbner basis g is repeatedly updated. In Algorithm 1 (Fig. 1), $\xrightarrow{G}$ and $\xrightarrow{g}$ denote the reduction of p modulo G or g, respectively. 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 the 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 allows to solve polynomial equations for many ciphers 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.

Require: M: set of allowed monomials
Require: G: reduced Gröbner basis of I_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 $p \not\xrightarrow{g} 0$ **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 $p \rightarrow 0$ **then**
      $F \leftarrow F - \{p\}$
    end if
  end for
  return F

Fig. 1 *Algorithm 1: Algorithm for generating S-box equations*

Table 1 Cryptanalysis of SR ciphers

Instance	Time	Tool	Note	Ref.
SR(10,1,1,4)	74.06	F4	**degrevlex**	[5]
SR(10,2,2,4)	1193.19	PolyBoRi	**lex**	[17]
SR(10,2,2,4)	190.58	MiniSat	—	[17]
SR(4,2,1,4)	0.63	PolyBoRi	**lex**	[17]
SR(10,2,1,4)	0.225	PolyBoRi	**lex**	[7]
SR(10,2,1,4)	10,327.3	F4	—	[27]
SR(10,2,2,4)	1850.01	PolyBoRi	**lex**	[7]

4 SR block ciphers

The family of SR ciphers is defined by Cid *et al.* in [5]. It provides a useful workbench to study and experiment with different algebraic attacks. Due to 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 [5].

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 fourth displays monomial ordering used and the fifth refers to the work in which the results are published.

We apply the **degrevlex** ordering, where secret key variables have lower order than other variables. We use a similar notation to the one used by Bulygin and Brickenstein [7]. The $SR(n, 2, 1, 4)$ cipher is described by the following system of equations:

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

In (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 jth nibble of x_i. L_0() and L_1() denote multiplication of first and second rows of the MixColumn matrix, respectively.

5 Experiments

5.1 Gröbner basis algorithms

In this section, we experiment on the impact of S-box representation (*FW, BW, MQ* or *SMQ* – see Section 3) on the time needed to solve a collection of equations for the whole cipher and determine the secret key. We use three software packages for computation of Gröbner basis. We also experiment with the CryptoMiniSat SAT-solver [16]. The tools are employed through the SAGE computer algebra system, version 6.7-x86_64 [28]. 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 the average. Gröbner 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. The S-box representations of the type *FW, MQ, SMQ* and *FWBW* for $SR(n, 2, 1, 4)$ are defined with (1), (3), (5) and combination (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 instances). The entries with $\perp$ indicate cases where computation of Gröbner basis has failed due to memory limits of the system and/or the employed software tool. With the FGb, we set a time limit of 1000 s 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 10,000, which means at each reduction step, at most 10,000 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 $n > 8$ cannot be solved with Singular 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 a 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 optimised for manipulating Boolean polynomials, is used in PolyBoRi.
- FWBW polynomials, then computation of Gröbner basis by PolyBoRi is greatly improved. However, when we use Singular, we observe 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 [5] managed

to compute Gröbner basis of $SR(3,2,1,4)$ over $\mathbb{F}_{2^4}$ in 519.92 s using 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 [With a small modification to the PolyBoRi package, it is possible to solve some instances of $SR(n,2,1,4)$ with SMQ representation of S-boxes. This is reported in Appendix 1.]. Our results suggest 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, for this representation.

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 reduces 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 indicates dimensions of the largest matrix that are 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 0×55.

From Table 3, MQ representation of S-boxes yields larger matrices in comparison with the FWBW representation. In Figs. 2 and 3, dimensions of matrices generated during computation of 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 the efficiency of algebraic attacks in two dimensions: (i) the running time for computation of Gröbner basis and (ii) the memory requirement for storing matrices.

The results of experiments suggest that an 'appropriate' representation of S-boxes has much bigger impact than the algebraic tool used to solve an instance.

5.2 Experiments on $SR(n,4,4,4)$

Considering the similarity of $SR(n,4,4,4)$ to AES, we also investigate the effect of S-box description for this cipher. The results for FWBW description are presented in Table 4. Considering $SR(2,4,4,4)$ with eight known plaintexts, computation of Gröbner basis takes just 3.54 s on average. Taking $SR(3,4,4,4)$ with three chosen plaintexts, the Gröbner basis is computed in 36.41 s, averagely.

For the MQ representation, we were not able to solve any instances of the cipher within a reasonable time. With $SR(n,4,4,4)$, we noticed that the order of variables greatly influences the running time for computation of Gröbner basis. In contrast with previous experiments, we might use **degerevlex** where the variables denoting key bits are highest in the order. With opposite order, where key variables are the lowest in the order, we noticed that during the computation of Gröbner basis, intermediate polynomials get larger, in such a way that we stop with 'out of memory' error. The swelling of intermediate polynomials might be due to the structure of key schedule algorithm of $SR(n,4,4,4)$ cipher, as intermediate variables in the key schedule are being rewritten with long polynomials in other key variables, and then polynomials in the encryption part, that contains these variables are rewritten with larger polynomials.

5.3 Lightweight ciphers

We further study the impact of the S-box representation on the efficiency of solving round-reduced lightweight ciphers such as LBlock [18], PRESENT [19] and MIBS [20]. The summary of best algebraic attack results published so far is given in Table 5, where g denotes the number of guessed key bits and *Data* denotes the number of plaintexts needed in the attack.

Courtois *et al.* [29] and Susil *et al.* [30] mount their attacks on the LBlock cipher using the ElimLin algorithm [8]. Courtois *et al.* [29] choose plaintexts at random. In contrast, Susil *et al.* [30] select plaintexts via a successful cube attack [32]. 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 [29, 31].

In our experiments, plaintexts are chosen to be correlated messages – the same strategy as in [30, 33]. Unlike [30] where the messages are selected based on a successful cube attack, in our strategy the ith plaintext is the 64-bit representation of number i. The results obtained for the LBlock, PRESENT and MIBS ciphers are presented in Table 6 where the following notations are 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 optimised data structure for memory efficient manipulation of Boolean polynomials. In contrast with our earlier experiments with the SR cipher, we set the

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	⊥	55.67	⊥	⊥	142.75
9	256.14(45)	1961.4	83.98	404.98(36)	⊥	78.8	⊥	⊥	202.24
10	259.65(46)	3314.03(49)	116.95	205.5(23)	⊥	123.8	⊥	⊥	296.83

Table 3 Dimensions of largest matrices generated by PolyBoRi

N_r	MQ	FWBW
5	$48,476 \times 41,218$	$19,330 \times 13,293$
6	$66,529 \times 63,074$	22776×14071
7	$90,212 \times 92,552$	$18,891 \times 22,100$
8	$142,596 \times 142,467$	$28,121 \times 32,174$
9	$175,211 \times 181,603$	$31,326 \times 36,192$
10	$214,416 \times 228,497$	$46,249 \times 51,847$

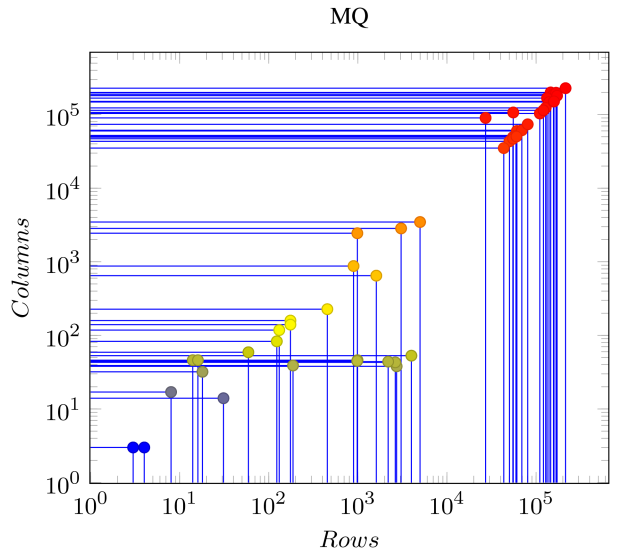

Fig. 2 *Dimension of intermediate matrices generated by* POLYBORI *during computation of Gröner basis, for a single instance of* $SR(10, 2, 1, 4)$ *with MQ representation of S-boxes*

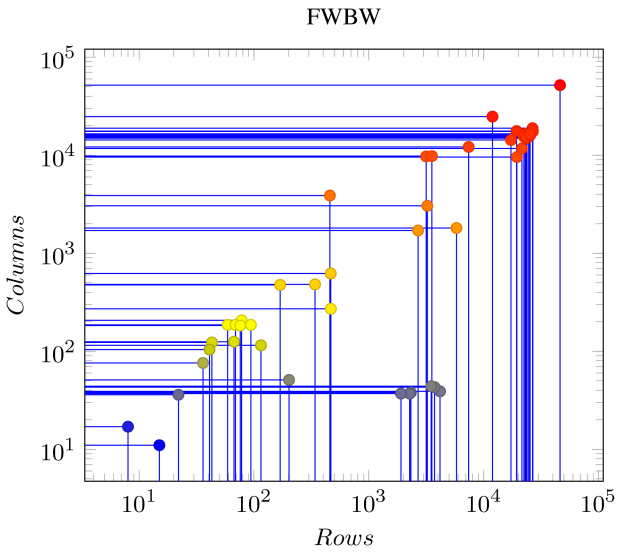

Fig. 3 *Dimensions of matrices generated by* POLYBORI *during computation of Gröner basis, for a single instance of* $SR(10, 2, 1, 4)$ *with FWBW representation of S-boxes*

Table 4 Algebraic attacks on $SR(n, 4, 4, 4)$ using FWBW description of S-boxes

N_r	g	T_G, s	Data	N_S
2	0	3.54	8KP	50/50
3	0	36.41	3CP	50/50

`faguere` argument to **False** in order to disable the use of linear algebra for reduction of polynomials. This is due to 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 eight rounds, knowing three chosen plaintext/ciphertext observations. This is a better result than described in [30], where the attack needs eight plaintext/ciphertext pairs. The nine-round LBlock is broken with four observations. Using such a simple strategy 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 [30]. 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 6 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 Nakahara *et al.* in [31], which needs 16 plaintext/ciphertext pairs and a guess of 35 key bits on five rounds of PRESENT.

Considering four-round MIBS cipher, the FWBW yields an attack that needs five chosen plaintext/ciphertext pairs only. This outperforms the previous result by Courtois *et al.* [29], which needs 32 plaintext/ciphertext pairs and a guess of 20 key bits. The run time of attack is reduced to 10.17 s.

5.4 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 six 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 five rounds of MIBS cipher. We call this pre-processing, the *partial inter-reduction*. The results for PRESENT and MIBS are presented in Table 7. 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.

Considering the PRESENT cipher, partial inter-reduction reduces the running time for computation of Gröbner basis of polynomials related to six rounds of PRESENT, to 1926.82 s on average. For the MIBS cipher, partial inter-reduction not only reduces the running time for attacking the four-round version to 1.07 on 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 six-round MIBS.

5.5 Attacking with known-plain text scenario

In [29, 31], the attacks presented on the ciphers LBlock, PRESENT and MIBS consider a known plaintext scenario. These attacks use MQ for algebraic description of S-boxes. For a better comparison, we present some results in this scenario with FWBW description. The results are presented in Table 5. In the work [29], an 8-round LBlock is attacked with six known plaintexts and with guessing of 32 bits of the key. With same parameters with FWBW and the tool POLYBORI, the instances were solved on average in just 27.49 s. In case of MIBS cipher, we attack 5 rounds of MIBS with 18-bit guess key and with 64 known plaintexts in just 94.92 s on average, which is faster than [29], with MiniSAT.

Considering PRESENT, we are able to attack four rounds of PRESENT without guessing any key bit and with 32 known plaintexts.

As it is indicated in Table 5, we managed to achieve better performance when we use FWBW description of S-boxes.

5.6 Comparison

In this paper, we have reported algebraic attacks on the lightweight block ciphers LBlock, MIBS and PRESENT. We have shown the effectiveness of FWBW representation of S-boxes when applying the Gröbner basis approach for algebraic cryptanalysis. Table 8 summarises other attacks on the above-mentioned ciphers. However, it is difficult to compare them with our algebraic cryptanalysis. This is mainly related to different nature of the attacks under consideration. For example, differential attacks are probabilistic and their efficiency can be relatively easily extrapolated, while algebraic attacks are deterministic and their success depends on solving a system of (non-linear) relations. So far we do not know a generic algorithm that would solve (efficiently) a system of non-linear relations, particularly non-linear relations arising from block ciphers for large number of rounds. Nevertheless, our research points out that Gröbner basis computation algorithms are very useful tools in algebraic cryptanalysis. Due to the algorithms behaviour, it is difficult to

Table 5 Algebraic attacks on LBlock, PRESENT and MIBS, w.r.t. rounds

N_r	g	Runtime, s	Data	Note	Work
LBlock					
8	32/80	907	6 KP	ElimLin	[29]
8	32/80	⊥	6 KP	POLYBORI	[29]
8	0/80	not reported	8 CP	ElimLin	[30]
8	32/80	27.49	6 KP	POLYBORI-FWBW	this
10	0/80	not reported	16 CP	ElimLin	[30]
11	0/80	10,106	128 CP	POLYBORI-FWBW	this
PRESENT					
4	0/80	155.06	32 KP	POLYBORI-FWBW	this
5	35/80	16,092	16 KP	ElimLin	[31]
5	35/80	⊥	16 KP	POLYBORI	[31]
6	0/80	2009.03	32 CP	POLYBORI-FWBW	this
MIBS					
4	20/80	493.2	32 KP	ElimLin	[29]
4	20/80	⊥	32 KP	POLYBORI	[29]
4	18/80	5.5	6 KP	POLYBORI-FWBW	this
5	16/64	1422	6 KP	MiniSAT 2.0	[29]
5	16/64	⊥	6 KP	POLYBORI	[29]
5	18/80	94.92	64 KP	POLYBORI-FWBW	this
6	0/80	68.46	12 CP	POLYBORI-FWBW	this

Table 6 Algebraic attacks on LBlock, PRESENT and MIBS using FWBW description of S-boxes

N_r	T_G, s	Data	N_S
LBlock			
8	1.18	3 CP	50/50
9	8.7	4 CP	50/50
10	224.2	12 CP	50/50
11	10,106	128 CP	10/10
PRESENT			
4	3.25	4 CP	50/50
5	23.57	6 CP	50/50
6	12,048.28	32 CP	14/15
MIBS			
4	10.17	5 CP	50/50

Table 7 Algebraic attacks on PRESENT and MIBS using FWBW description of S-boxes and preprocessing

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

derive efficiency measurement as we do not know tight bounds on data, time and memory requirements of the algorithm. It is worth noting that algebraic cryptanalysis is still evolving and many of its aspects are yet to be discovered. Many results in this area are reported based on experiments.

Table 8 details differential, integral and cube attacks for the three ciphers with 80 bits key versions.

In [34], a Cube attack on 9 rounds of LBlock cipher with 1184 chosen plaintexts and time complexity of 2^{47} encryption is reported. In contrast, our algebraic attack that uses the FWBW S-box representation needs 128 chosen plaintext only. Consider differential cryptanalysis of LBlock. The best differential characteristics for 10-round LBlock imply at least 18 active S-boxes [18]. If this characteristic is used in an 11-round key recovery attack, the data complexity of the attack would be of order $O(2^{38})$. Our attack requires a much smaller number of plaintexts, i.e. 128. The work [35] reports an integral attack on 22-round LBlock with data and time complexity of 2^{61} and $O(2^{70})$, respectively. The attack is based on a 15-round Integral distinguisher.

Table 8 Some integral and differential cryptanalysis of LBlock, MIBS and PRESENT

N_r	Runtime	Data	Note	Work
LBlock				
9	2^{47}	1184 CP	cube attack	[34]
10	—	$O(2^{36})$ CP	differential characteristics	[18]
11	—	$O(2^{44})$ CP	differential characteristics	[18]
22	$O(2^{70})$	2^{61} CP	integral cryptanalysis	[35]
PRESENT				
6	$2^{41.7}$	$2^{22.4}$ CP	integral cryptanalysis	[36]
9	—	2^{60} CP	integral distinguisher	[37]
9	2^{60}	$2^{20.3}$ CP	integral cryptanalysis	[38]
16	2^{64}	2^{64} CP	differential cryptanalysis	[39]
MIBS				
4	—	$O(2^{15})$ CP	differential characteristics	[20]
13	2^{56}	2^{61} CP	differential cryptanalysis	[40]

Z'aba *et al.* [36] present a bit-pattern-based integral attack on six rounds of PRESENT-80. The attack takes advantage of a 4.5 round integral distinguisher. The data and time complexity of the attack is $2^{22.4}$ and $2^{41.7}$, respectively. Our algebraic cryptanalysis attack requires 32 chosen plaintext only and a running time of about 2000 s on average. In [37], an integral distinguisher for nine rounds of PRESENT is also reported.

For MIBS cipher, the best four-round differential characteristics has the probability of $O(2^{-15})$ [20]. If this characteristic is used in a hypothetical key recovery attack on six-round MIBS, it would require a data complexity of at least $O(2^{15})$. In this paper, however, the 6-round cipher is broken with 12 chosen plaintexts only in an algebraic cryptanalysis attack. In [40], a differential cryptanalysis attack is proposed on 13 rounds of MIBS based on a 12-round differential characteristic, with data and time complexity of 2^{61} and 2^{56}, respectively.

Our results indicate that algebraic attack works better than other cryptanalysis techniques for a small number of rounds. However, for larger number of rounds, the data, memory and time complexity of algebraic attacks is not known and is still an open problem. A breakthrough in cryptanalysis would be a discovery of efficient algorithm for solving a generic system of non-linear relations. As this is a very unlikely event, one would hope that such algorithm may exist for an attacked cipher, which is an open problem for most cryptographically strong ciphers.

6 Conclusion

In this paper, we have studied 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 has shown that S-box representation has a significant impact on the 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 software 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:

- Generalisation of our study for 8-bit S-boxes. A related problem is the choice of the field $GF(2^n)$, which can be 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.

7 References

[1] Daemen, J., Rijmen, V.: '*AES proposal*' (NIST AES Proposal, Rijndael, 1998)

[2] Courtois, N.T., Pieprzyk, J.: 'Cryptanalysis of block ciphers with overdefined systems of equations'. Advances in Cryptology – (ASIACRYPT 2002), Queenstown, New Zealand, 2002, pp. 267–287

[3] Cid, C.: 'Some algebraic aspects of the advanced encryption standard', in Dobbertin, H., Rijmen, V., Sowa, A. (eds.): '*Advanced encryption standard – AES*' (Springer, Heidelberg, 2005), pp. 58–66

[4] Cid, C., Leurent, G.: 'An analysis of the XSL algorithm'. Advances in Cryptology (ASIACRYPT 2005), Paris, France, 2005, pp. 333–352

[5] Cid, C., Murphy, S., Robshaw, M.J.B.: 'Small scale variants of the AES', in Gilbert, H., Handschuh, H. (eds.): '*Fast software encryption*' (Springer, Heidelberg, 2005), pp. 145–162

[6] Faugére, J.C.: 'A new efficient algorithm for computing gröbner bases (F4)', *J. Pure Appl. Algebra*, 1999, **139**, (1-3), pp. 61–88

[7] Bulygin, S., Brickenstein, M.: 'Obtaining and solving systems of equations in key variables only for the small variants of AES', *Math. Comput. Sci.*, 2010, **3**, (2), pp. 185–200

[8] Courtois, N.T., Bard, G.V.: 'Algebraic cryptanalysis of the data encryption standard', in Galbraith, S.D. (ed.): '*Cryptography and coding*' (Springer, Heidelberg, 2007), pp. 152–169

[9] Courtois, N.T., Castagnos, G., Goubin, L.: 'What do DES S-boxes Say to Each Other?'. Cryptology ePrint Archive Report 2003/184. Available at http://eprint.iacr.org/2003/184

[10] Courtois, N.T.: 'Some algebraic description for various S-boxes'. Accessed 2017-01-31. Available at http://www.nicolascourtois.com/equations/block/sboxes/misc_sboxes.ZIP

[11] Courtois, N.T.: 'Some algebraic description for various S-boxes'. Accessed 2017-01-31. Available at http://www.nicolascourtois.com/equations/block/gost/gost_boxes.ZIP

[12] Fuhs, C., Schneider-Kamp, P.: 'Synthesizing shortest linear straight-line programs over GF(2) using SAT'. Theory and Applications of Satisfiability Testing (SAT 2010), Edinburgh, UK, 2010, pp. 71–84

[13] Courtois, N., Mourouzis, T., Hulme, D.: 'Exact logic minimization and multiplicative complexity of concrete algebraic and cryptographic circuits', *Int. J. Adv. Intell. Syst.*, 2013, **6**, (3), pp. 165–176

[14] Decker, W., Greuel, G.M., Pfister, G., *et al.*: 'Singular 3-1-7 – a computer algebra system for polynomial computations', 2015. Available at http://www.singular.uni-kl.de

[15] Faugère, J.C.: 'FGb: a library for computing gröbner bases'. Mathematical software – (ICMS 2010), Kobe, Japan, 2010, pp. 84–87

[16] Soos, M.: 'SAT-solver cryptominisat, Version 2.9.0, January 20 2011 '

[17] Brickenstein, M., Dreyer, A.: 'Polybori: a framework for Gröbner-basis computations with Boolean polynomials', *J. Symb. Comput.*, 2009, **44**, (9), pp. 1326–1345

[18] Wu, W., Zhang, L.: 'LBlock: a lightweight block cipher', in Lopez, J., Tsudik, G. (eds.): '*Applied cryptography and network security*' (Springer, Heidelberg, 2011), pp. 327–344

[19] Bogdanov, A., Knudsen, L.R., Leander, G., *et al.*: 'PRESENT: an ultra-lightweight block cipher'. Cryptographic Hardware and Embedded Systems (CHES 2007), Vienna, Austria, 2007, pp. 450–466

[20] Izadi, M., Sadeghiyan, B., Sadeghian, S.S., *et al.*: 'MIBS: a new lightweight block cipher', in Garay, J.A., Miyaji, A., Otsuka, A. (eds.): '*Cryptology and network security*' (Springer, Heidelberg, 2009), pp. 334–348

[21] Cox, D., Little, J., O'Shea, D.: '*Ideals, varieties, and algorithms*', Undergraduate Texts in Mathematics (Springer, New York, 2007)

[22] Bosch, S.: '*Algebraic geometry and commutative algebra*', Universitext (Springer-Verlag, London, 2013). Available at http://www.springer.com/us/book/9781447148289

[23] 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', *J. Symb. Comput.*, 2006, **41**, (3-4), pp. 475–511

[24] Gao, S.: '*Counting zeros over finite fields with Gröbner bases*' (Carnegie Mellon, Pittsburgh, Pennsylvania, USA, 2009). Available at http://www.andrew.cmu.edu/user/avigad/Students/gao_ms_thesis.pdf

[25] Buchberger, B.: 'A criterion for detecting unnecessary reductions in the construction of Gröbner-bases', in Ng, E.W. (ed.): '*Symbolic and algebraic computation*' (Springer, Heidelberg, 1979), pp. 3–21

[26] Brickenstein, M.: 'Slimgb: gröbner bases with slim polynomials', *Rev. Mat. Complutense*, 2009, **23**, (2), pp. 453–466

[27] Miolane, C.: '*Block cipher Analysis*' (Technical University of Denmark (DTU), Copenhagen, Denmark, 2009). Available at http://orbit.dtu.dk/services/downloadRegister/5009704/thesis_cvm_v1.pdf

[28] The Sage Developers: 'Sagemath, the sage mathematics software system (version 6.7)', 2015, Available at http://www.sagemath.org

[29] Courtois, N.T., Sepehrdad, P., Sušil, P., *et al.*: 'Elimlin algorithm revisited', in Canteaut, A. (ed.): '*Fast software encryption*' (Springer, Heidelberg, 2012), pp. 306–325

[30] 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*' (Springer, Cham, 2014), pp. 50–65

[31] Nakahara, J., Sepehrdad, P., Zhang, B., *et al.*: 'Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT', in Garay, J.A., Miyaji, A., Otsuka, A. (eds.): '*Cryptology and network security*' (Springer, Heidelberg, 2009), pp. 58–75

[32] Dinur, I., Shamir, A.: 'Cube attacks on tweakable black box polynomials'. Advances in Cryptology (EUROCRYPT 2009), Cologne, Germany, 2009, pp. 278–299

[33] 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*' (Springer, Heidelberg, 2010), pp. 266–277

[34] Islam, S., Afzal, M., Rashdi, A.: 'On the security of LBlock against the cube attack and side channel cube attack', in Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.): '*Security engineering and intelligence informatics*' (Springer, Heidelberg, 2013), pp. 105–121

[35] Sasaki, Y., Wang, L.: 'Comprehensive study of integral analysis on 22-round LBlock'. Information Security and Cryptology (ICISC 2012), Seoul, South Korea, 2013, pp. 156–169

[36] Z'aba, M.R., Raddum, H., Henricksen, M., *et al.*: 'Bit-pattern based integral attack', in Nyberg, K. (ed.): '*Fast software encryption*' (Springer, Heidelberg, 2008), pp. 363–381

[37] Eskandari, Z., Kidmose, A.B., Kölbl, S., *et al.*: 'Finding integral distinguishers with ease'. Cryptology ePrint Archive, Report 2018/688. Available at https://eprint.iacr.org/2018/688

[38] Wu, S., Wang, M.: 'Integral attacks on reduced-round PRESENT', in Qing, S., Zhou, J., Liu, D. (eds.): '*Information and communications security*' (Springer, Cham, 2013), pp. 331–345

[39] Wang, M.: 'Differential cryptanalysis of reduced-round PRESENT'. Progress in Cryptology (AFRICACRYPT 2008), Casablanca, Morocco, 2008, pp. 40–49

[40] Bay, A., Nakahara, J., Vaudenay, S.: 'Cryptanalysis of reduced-round MIBS block cipher', in Heng, S.H., Wright, R.N., Goi, B.M. (eds.): '*Cryptology and network security*' (Springer, Heidelberg, 2010), pp. 1–19

[41] 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. Available at http://eprint.iacr.org/2007/024

Table 9 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 10 Number of variable and average number of clauses for different descriptions conversion

N_r		#cls			
	#var	FW	FWBW	MQ	SMQ
5	256	1509	4434	39,602	9577
6	304	1810	5316	46,594	10,656
7	352	2109	6199	44,132	12,332
8	400	2406	7074	60,884	13,567
9	448	2704	7984	51,304	14,780
10	496	3007	8846	58,886	16,343

8 Appendix 1

During the computation of Gröbner basis of instances with SMQ representation of S-boxes for $SR(n, 2, 1, 4)$, with PolyBoRi package, size of intermediate matrices quickly exceeds the built-in limit for matrix dimensions, and then the tool fails. Considering source code of python interface for PolyBoRi, we noticed that for the degree-based order of variables such as **deglex**, after selection of polynomials for reduction, the tool find minimum degree of polynomials. Then, the reduction of polynomials with the degree higher than the minimum, is being delayed. We noticed that if this check is skipped and take all of selected polynomials for reduction, then we are able to solve some of instances for five and six rounds of cipher $SR(n, 2, 1, 4)$ with PolyBoRi. For $SR(5, 2, 1, 4)$, 31 of 50 instances solved in 169.80 s on average, and for $SR(6, 2, 1, 4)$, 17 of 50 instances solved in 410.11 s averagely.

9 Appendix 2

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. Bard *et al.* [41] propose a method to convert ANF polynomials into a single CNF. The method is used by Courtois and Bard [8] to translate 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?

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.* [41]. Resulting SAT instances are solved with the CryptoMiniSAT solver. The obtained results are reported in Tables 9 and 10. In Table 10, #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 be solved in 2.69 s on average. This is a significant improvement compared to the results reported by Bulygin and Brickenstein [7]. They were able to solve $SR(10, 2, 2, 4)$ instances in 1850.01 s using PolyBoRi and in 190 s 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 s. 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 s. In Table 10, 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.