computation

Article

Algebraic Analysis of a Simplified Encryption Algorithm GOST R 34.12-2015

Evgenia Ishchukova [1], Ekaterina Maro [1,*] and Pavel Pristalov [2]

[1] Department of Information Security, Southern Federal University, Rostov Oblast 347922, Russia; uaishukova@sfedu.ru
[2] LLC TopSoft (Altarix), Moscow 440000, Russia; pristalov@gmail.com
* Correspondence: eamaro@sfedu.ru

Received: 20 April 2020; Accepted: 18 May 2020; Published: 28 May 2020

Abstract: In January 2016, a new standard for symmetric block encryption was established in the Russian Federation. The standard contains two encryption algorithms: Magma and Kuznyechik. In this paper we propose to consider the possibility of applying the algebraic analysis method to these ciphers. To do this, we use the simplified algorithms Magma $\oplus$ and S-KN2. To solve sets of nonlinear Boolean equations, we choose two different approaches: a reduction and solving of the Boolean satisfiability problem (by using the CryptoMiniSat solver) and an extended linearization method (XL). In our research, we suggest using a security assessment approach that identifies the resistance of block ciphers to algebraic cryptanalysis. The algebraic analysis of an eight-round Magma (68 key bits were fixed) with the CryptoMiniSat solver demanded four known text pairs and took 3029.56 s to complete (the search took 416.31 s). The algebraic analysis of a five-round Magma cipher with weakened S-boxes required seven known text pairs and took 1135.61 s (the search took 3.36 s). The algebraic analysis of a five-round Magma cipher with disabled S-blocks (equivalent value substitution) led to getting only one solution for five known text pairs in 501.18 s (the search took 4.92 s). The complexity of the XL algebraic analysis of a four-round S-KN2 cipher with three text pairs was 236.33 s (took 1.191 Gb RAM).

Keywords: S-box; systems of multivariate quadratic equations; algebraic analysis; SAT solving method; extended linearization method; GOST R 34.12-2015 encryption algorithms; Magma; Kuznyechik

1. Introduction

The discussion of the potential vulnerability of algebraic analysis has attracted the attention of scientists all over the world to the algebraic methods of attack on information protection systems. The advantage of this method, compared to those described above, is in obtaining the correct encryption key in the presence of a small number of known pairs of plaintext/ciphertext. Mostly algebraic analysis is focused on nonlinear primitives of encryption algorithms and is based on describing the encryption algorithm in the form of systems of nonlinear equations connecting the secret key and the known data. In the work [1], it is stated that the author performed the analysis of the full-round algorithm GOST 28147-89. At the same time, the work did not disclose the used approaches to the analysis, but gives only a general description of the work done and its approximate complexity.

Referring to the work of C. Shannon [2], we can say that the assessment of reliability information protection with encryption algorithms is equal to doing "as much work as to solve a system of equations with a large number of unknowns". For a long time, in the process of analyzing block encryption algorithms, most attention was paid to statistical methods of analysis, and algebraic methods of analysis that describe general approaches to the problem of analyzing the reliability of encryption algorithms were not sufficiently taken into consideration.

There are various ways to find solutions of nonlinear Boolean equations systems. During the analysis of science research bases, three main areas were identified in the field of solving such systems used in information security assessment (reliability of encryption algorithms) [3]:

- SAT solvers: MiniSat, CryptoMiniSat, Glucose, Riss, Slime, Lingeling, Plingeling, CaDiCaL, etc.
- Methods based on a Gröbner basis: Buchberger's algorithm, F4, F5, Matrix-F5, Tropical F5, Method of Four Russians, etc.
- Methods based on the linearization principle: relinearization, extended linearization (XL), extended sparse linearization (XSL), MutantXL, FXL, ElimLin, etc.

The above algebraic analysis methods are being actively developed and improved as applied to cryptographic algorithms (especially lightweight, because of their simplified mathematical structure). There are some recent results published on the topic of algebraic analysis (namely linearization methods and SAT solving). Taking into account the importance of substitution S-boxes for algebraic analysis, we note research [4], which discussed the effects of S-box representation on the efficiency of the algebraic analysis of block ciphers. Additionally, algebraic analysis is based on representation addition modulo 2^n operations and their influence on efficiency is discussed in [5].

We should note that paper [6] aimed at analyzing the practical effectiveness of algebraic attacks using experiments for reduced block ciphers (an algebraic attack was implemented on a reduced LowMC cipher). As the basis of the attack, a dynamical elimination algorithm was developed. The research [7] is devoted to the development and analysis of quantum algorithms for solving a system of quadratic equations. The complexity calculations of the algorithms XL, FXL, ReversibleXL and GroverXL for random systems are performed. Research [8] is focused on the application of the XL algorithm to generic systems with 32 variables and 64 equations over GF16. Experiments have been carried out on two computer systems: a 64-core NUMA system and an8-node InfiniBand cluster, and a comparison of the investigated implementation of an XL algorithm with PWXL (a parallel implementation of XL) and Faugère's F4 algorithms was also made. Algebraic attack (by a linearization method) on two/three/four rounds of Keccak-384 and two/three rounds of Keccak-512 were investigated in paper [9]. Paper [10] describes the application of algebraic cryptanalyses to 12-round LBlock, six-round MIBS, seven-round PRESENT and nine-round SKINNY lightweight block ciphers. A new approach to simplify the equation system was presented (by using additional polynomial relations—linear relation between intermediate state bits). Research [11] demonstrated that the block ciphers Jarvis and Friday (members of the MARVELlous family of cryptographic primitives) are vulnerable to Gröbner basis attacks. Algebraic attacks on reduced- and full-round DESL (a lightweight version of DES) are presented in [12].

Paper [13] describes the SAT-based algorithm to determine the multiplicative complexity of a Boolean function. A SAT-based cryptanalysis for the Grain v1 stream cipher is presented in [14]. Practical SAT-based guess-and-determine attacks for several stream ciphers are developed in [15]. The dissertation [16] describes an opportunity of synergy between Gröbner-like and DPLL-like solving. The author presented new types of solving algorithms (SRES) and made some experiments of algebraic fault attacks on the symmetric ciphers LED and derivatives of the block cipher AES.

In this paper, we consider the possibility of using linearization methods and SAT solvers to analyze information security properties for Russian symmetric block encryption standards and their modified versions.

Also we propose considering approaches to algebraic cryptanalysis of the simplified algorithms Magma and Kuznyechik (S-KN2, which is presented in [17]). The paper is organized as follows: Section 1 contains a brief description of the Russian symmetric block encryption standard GOST R 34.12-2015 and how its simplified versions are applied. Section 2 is devoted to the investigation of the basic algebraic analysis methods extended linearization and SAT solving. Section 3 includes proposed security assessment approaches and their input and output parameters. In Section 4 we consider algorithms for describing encryption transformations as systems of linearly independent equations (we fixed two basic nonlinear elements: S-box and addition modulo 2^n). Section 5 presents the experimental results of applying algebraic analysis methods to the Magma cipher (some versions) and S-KN2 cipher.

2. GOST R 34.12-2015

GOST R 34.12-2015 was introduced as a new symmetric block cipher standard in Russia in 2016 [18]. The standard contains the descriptions of two encryption algorithms: the Magma cipher (GOST R 34.12-2015 $n = 64$) and Kuznyechik cipher (GOST R 34.12-2015 $n = 128$). We describe both of them below, as well as the simplified versions used (Magma $\oplus$ and S-KN2).

2.1. Magma Cipher (GOST R 34.12-2015 $n = 64$) and Magma $\oplus$

The Magma encryption algorithm is part of the symmetric encryption standard in the Russian Federation [18]. Previously, this encryption algorithm was called GOST 28147-89 and was slightly different from the current version. In the earlier version of the cipher, unfixed S-boxes were used. The Magma cipher is a symmetric block cipher designed according to the Feistel scheme. In Electronic Codebook (ECB) mode, 64 bits of a data block (T) are converted to 64 bits of ciphertext (C) under the influence of a 256-bit secret key. According to Feistel's scheme, a data block is divided into two parts, each part containing 32 bits. The right part of the data is processed by the F-function in each round. The F-function consists of three operations:

- Mixing data with secret key bits using module 2^{32} addition;
- S-box bit substitution;
- 11 position cyclic shift to the left.

The F-function output is mixed with the left part of the data block by addition module two. After that, the left and right parts of the text are swapped. The scheme of the Magma encryption algorithm is shown in Figure 1. The S-boxes recommended by the GOST R 34.12-2015 standard for use in the Magma cipher are presented in Table 1.

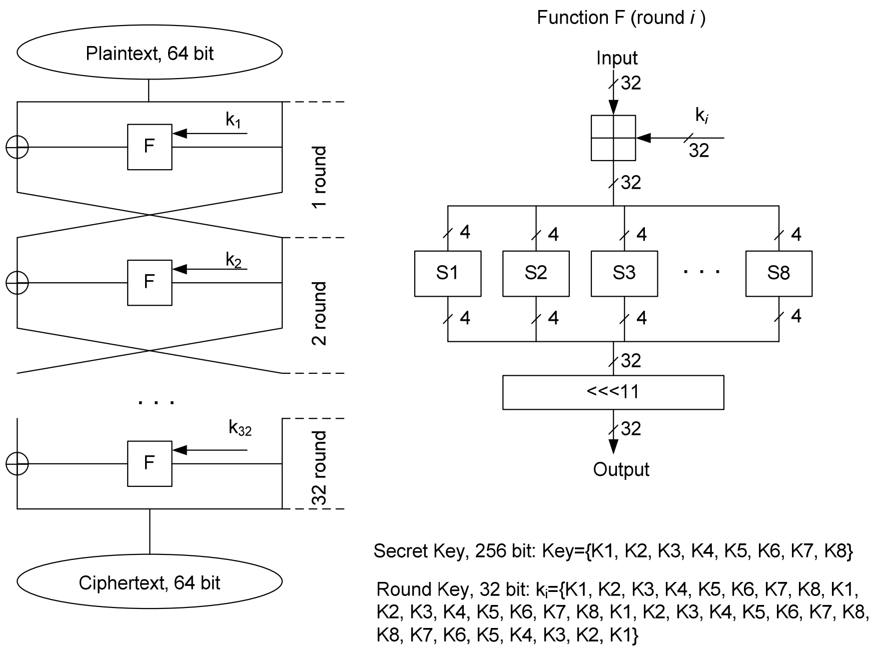
Plaintext, 64 bit
F
k_1
1 round
F
k_2
2 round
. . .
F
k_{32}
32 round
Ciphertext, 64 bit
Function F (round i)
Input
32
k_i
32
32
4
S1
S2
S3
. . .
S8
4
32
<<<11
32
Output
Secret Key, 256 bit: Key={K1, K2, K3, K4, K5, K6, K7, K8}
Round Key, 32 bit: k_i={K1, K2, K3, K4, K5, K6, K7, K8, K1, K2, K3, K4, K5, K6, K7, K8, K1, K2, K3, K4, K5, K6, K7, K8, K8, K7, K6, K5, K4, K3, K2, K1}

Figure 1. Magma encryption algorithm.

Table 1. Magma S-boxes.

S1	12	4	6	2	10	5	11	9	14	8	13	7	0	3	15	1
S2	6	8	2	3	9	10	5	12	1	14	4	7	11	13	0	15
S3	11	3	5	8	2	15	10	13	14	1	7	4	12	9	6	0
S4	12	8	2	1	13	4	15	6	7	0	10	5	3	14	9	11
S5	7	15	5	10	8	1	6	13	0	9	3	14	11	4	2	12
S6	5	13	15	6	9	2	12	10	11	7	8	1	4	3	14	0
S7	8	14	2	5	6	9	1	12	15	4	11	0	13	10	3	7
S8	1	7	14	13	0	5	8	3	4	15	10	6	9	12	11	2

The round encryption keys for use in each round are retrieved from the original 256-bit key. The original secret key is divided into eight 32-bit parts: K1, K2, K3, K4, K5, K6, K7, and K8. The round keys must be used in direct order from K1 to K8 in rounds one to 24 (three times) and in reverse order from K8 to K1 in rounds 25 to 32.

2.2. Kuznyechik Cipher (GOST R 34.12-2015 n = 128) and S-KN2 Cipher

The Kuznyechik cipher has been part of the government standard for symmetric data encryption in the Russian Federation since 2016. A full description of the encryption can be found in [18]. The cipher is based on the substitution and permutation network principle. The cipher input contains a data block of 128 bits, and the cipher text of 128 bits is generated at the output. For conversion, the secret key of 256 bits is used. The cipher starts by mixing the data with the first round key (addition module 2). After that, nine rounds of encryption are performed. Each round of encryption consists of three operations:

- Byte exchange with S-block;
- linear mixing bits L;
- mixing data with secret key bits using module 2 addition.

The output of the ninth round of encryption forms the ciphertext. In order to decrypt the data, the reverse order of operations must be used, the operations must be inversed, respectively, and the round keys must also be used in reverse order.

All operations in Kuznyechik are performed in the finite field GF (2)[x]/p(x), where p(x) = x8 + x7 + x6 + x + 1 ∈ GF (2)[x]. Kuznyechik uses the following steps:

- Byte exchange with S-block. The data block is divided into 16 bytes. Each byte is replaced by a new value according to the table defined in the standard;
- linear mixing bits L. The operation is performed by using the multiplication of polynomials in the given field. Multiplication is performed 16 times until all bytes are changed;
- mixing data with secret key bits using module 2 addition.

The secret key is divided into two parts, K1 and K2, which form the first two round key connections. These keys are used as inputs into a special Feistel scheme to form the remaining round keys.

The Kuznyechik cipher is a new algorithm and has not been sufficiently researched. The investigation of its properties is important. Applying simplified models for finding cryptographic properties is a common approach in cryptography.

For example, the S-DES algorithm, proposed by E. Schaefer and W. Stollings, is widely used for educational purposes [19]. A few simplified versions of AES were proposed by various authors: Rafael Chun-Wei Fan [20], Mohammed Musa, Edward Schaefer, and Stephen Vedig [21], Henri Gilbert [22], and others. These ciphers are used not only in education, but also to model various types of cryptanalytic attacks. Linear cryptanalysis of S-DES was proposed in [23]. The possibility of using differential cryptanalysis in addition to linear cryptanalysis is presented in [24]. The authors of [25] investigate the use of heuristic cryptanalysis for S-DES analysis. The authors of [26] present an approach to the cryptanalysis of S-DES using a genetic algorithm. The attack is carried out only on ciphertext, and on the basis of fitness function, many optimal keys are created. The authors of [27] present a new cryptanalysis attack aimed at the ciphertext generated by S-DES. The attack was carried out using a modified version of the BPSO (binary particle swarm optimization) algorithm. It is clear from the publication date that simplified versions of DES are still of interest to cryptographers. The number of publications on S-AES is no less. The authors of [28] present an approach to S-AES analysis using an impossible differential method. B. Hitapuru and S. Indarjani consider the possibility of applying the square attack to mini-AES [29]. The linear cryptanalysis of S-AES is presented by S. Indarjani, D. Mansouri, and H. K. Bizaki [30]. This investigation was continued by S. Campbell et al. [31]. The authors characterize a class of strongly nonlinear S-boxes for which their algorithm is always successful. They also show how to construct S-boxes to make the algorithm more resistant to linear cryptanalysis. S. Simmonds reviewed different approaches to S-AES analysis [32].

Two simplified versions for the Kuznyechik cipher were presented in [17]. The SKN-2 cipher was developed to simulate various cryptographic attack scenarios. The SKN-2 cipher is built in the image and likeness of the original Kuznyechik cipher. It converts a 16-bit data block using an SP network for three rounds. The secret key contains 32 bits. Round keys are generated from the original secret key using four rounds of the Feistel scheme. Each round consists of substitution S, linear transformation L, and the addition with the round subkey modulo two.

The original description of the S-KN2 cipher uses four rounds [17]. However, this amount can be increased easily. If the experiment allows you to use more rounds, then it is necessary to develop additional round keys according to the scheme. The S-KN2 cipher is shown in Figure 2. The first transformation is the mixing of data with the first round key. Hereinafter, data mixing with a round key is performed using modulo 2 addition (XOR operation). Further, each round contains three operations: replacing S, shuffling L, and addition with a round key. Consider each transformation in more detail.

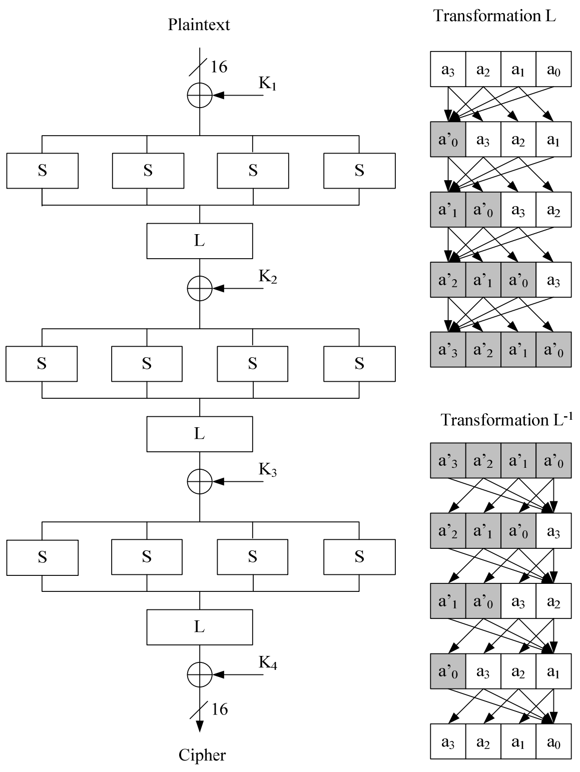

Figure 2. General data flow diagram of S-KN2.

In operation S, the data block is divided into four nibbles. For each nibble, a replacement is performed using Table 2. Table 2 has to be interpreted as follows: the upper row indicates the S-box input, while the lower one indicates the corresponding output. In this case, the inverse table will take the form shown in Table 3. The data in Tables 2 and 3 are presented in hexadecimal form.

Table 2. S-box setup.

Input	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
Output	3	6	a	7	f	0	5	b	2	c	1	e	4	9	d	8

Table 3. Inverse S-box setup.

Input	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
Output	5	a	8	0	c	6	1	3	f	d	2	7	9	e	b	4

The L transform contains four iterations, each changing one nibble and right shifting another one. The nibbles are modified during encryption, as described by the following formulas:

$$a'_0 = 4*a_3 \oplus a_2 \oplus 3*a_1 \oplus a_0.$$

$$a'_1 = 4*a'_0 \oplus a_3 \oplus 3*a_2 \oplus a_1.$$

$$a'_2 = 4*a'_1 \oplus a'_0 \oplus 3*a_3 \oplus a_2.$$

$$a'_3 = 4*a'_2 \oplus a'_1 \oplus 3*a'_0 \oplus a_3.$$

The information is presented in the form of polynomials for the transformations. Multiplication is performed modulo $\Psi(x) = x^4 \oplus x \oplus 1$. In [17], we propose an input–output matches table for polynomial multiplication by three. This table is easy to build and easy to use. A similar table can be constructed for multiplication by four.

For decryption, it is necessary to use the inverse operation L^{-1} (Figure 2). The following set of equations is used for that:

$$a_3 = 4*a'_2 \oplus a'_1 \oplus 3*a'_0 \oplus a'_3.$$

$$a_2 = 4*a'_1 \oplus a'_0 \oplus 3*a_3 \oplus a'_2.$$

$$a_1 = 4*a'_0 \oplus a_3 \oplus 3*a_2 \oplus a'_1.$$

$$a_0 = 4*a_3 \oplus a_2 \oplus 3*a_1 \oplus a'_0.$$

To generate round keys, the Feistel scheme, shown in Figure 3, is used. The first two round keys (K1 and K2, which are obtained from the original secret key) are considered as input. The right part (key K1) is input into the function F. The function F is also shown in Figure 3 and consists of three operations: addition modulo two for the data block and constant Ci, replacement with the S-box, and linear mixing L. C_i constants are obtained by transforming i with the linear transform L.

Data decryption is implemented in the reverse direction from bottom to top. The inverse operations are applied instead of their counterparts. A sample encryption and decryption by S-KN2 is given in [17].

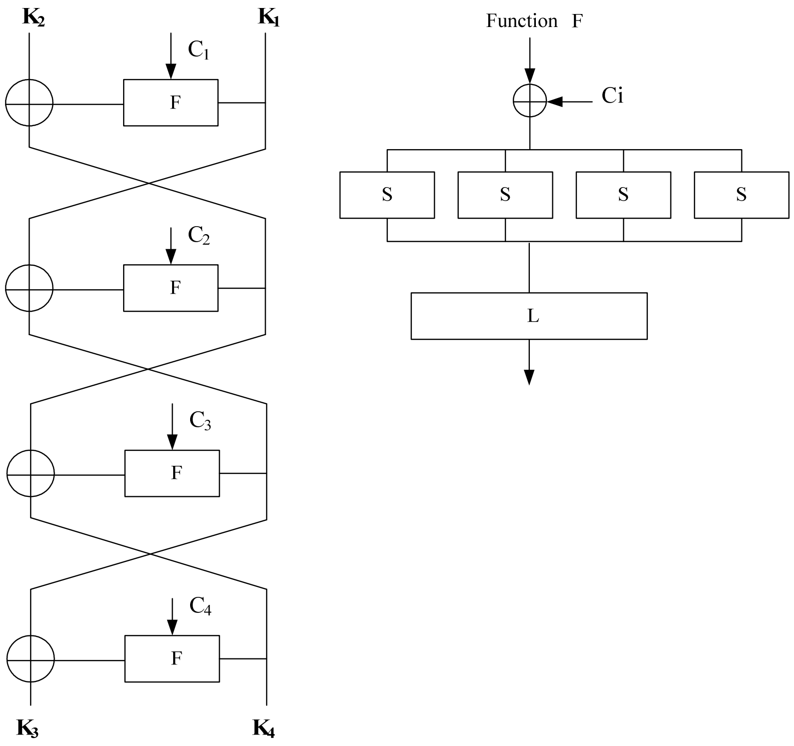

Figure 3. Round subkeys generation scheme.

3. Methods of Algebraic Analysis

3.1. Extended Linearization Method

The extended linearization method (XL) was proposed by N. Courtois, A. Klimov, J. Patarin, and A. Shamir in [33].

Let a field K and a system of quadratic equations ℓ_m be given, where m is the number of equations of the system. Each equation ℓ_i of the system is a polynomial of the form $f_i(x_1, \ldots, x_n) - b_i$., where i—is the number of the equation of the system, $x_1, \ldots, x_n$—are unknowns, b—is the free term of the equation. The goal of the extended linearization method is to obtain at least one solution $x = (x_1, \ldots, x_n) \in K^n$ for a given value of $b = (b_1, \ldots, b_m) \in K^m$. All possible multiplications of unknowns x of degree k are denoted as the set $W = \left\{\prod_{j=1}^{k} x_{i_j}, \ldots, \prod_{j=1}^{k} x_{e_j}\right\}$, where $x_{i_j}, x_{e_j} \in \{x_1, \ldots, x_n\}$, $i_1, \ldots, i_k \in \{1, 2, \ldots, n\}$, $e_1, \ldots, e_k \in \{1, 2, \ldots, n\}$. Let $D \in N$, N—the set of natural numbers, then I_D—is an ideal, generated by equations of the form $w_j\ell_i$, where $w_j \in W$, $j \in \{1, 2, \ldots, |W|\}$, $i \in \{1, 2, \ldots, m\}$, $k \in \{1, 2, \ldots, D-2\}$.

Stages of the extended linearization method:

a. Formation of the set W of all possible multiplications of variables x of degree k, where $k \in \{1, 2, \ldots, D-2\}$.
b. Composition of all multiplications of the form $w_j\ell_i \in I_D$, where $w_j \in W$, $j \in \{1, 2, \ldots, |W|\}$, $i \in \{1, 2, \ldots, m\}$, $k \in \{1, 2, \ldots, D-2\}$.
c. Consideration of each monomial of degree greater than D as a new variable and applying the Gauss elimination method to the equations obtained in (b).
d. Repeat step (c) until the result is at least one equation with a single unknown x_i.
e. Simplification of equations and repetition of the process to find the values of other unknowns.

Consider the calculation of the parameter of the XL analysis algorithm. Let $D \in N$ —parameter of the XL algorithm. The algorithm is based on multiplying all the equations of the system by the products of variables in degree $D-2$. As a result of multiplication, we obtain approximately $R \approx \binom{n}{D-2} m$

new equations. The total number of monomials found in these equations is $T = \binom{n}{D}$. Most of the resulting equations are linearly independent. In this case, you need to choose a large enough D such that:

$$R = \binom{n}{D-2} m \geq \binom{n}{D} = T.$$

It is obvious that the total number of linearly independent equations cannot exceed the number of monomials T. If the system has a unique solution, then there is a value D, for which the inequality $R \geq T$. holds. Moreover, the number of linearly independent equations from R will be close enough to the value of T. If the difference between the number of monomials and linearly independent equations is not large, then the system will be solvable. The system will be solved most easily with a very small value of the difference between the number of monomials and linearly independent equations.

It is expected that the value of D, at which the extended linearization method is applicable, will be equal to or close to the theoretical value of the parameter D. In this case, the algorithm of the extended linearization method will be effective, provided that:

$$R \geq T \Rightarrow m \geq \binom{n}{D} / \binom{n}{D-2} \approx n^2/D^2. \tag{1}$$

From the Formula (1) we obtain that:

$$D \approx \frac{n}{\sqrt{m}}.$$

3.2. *SAT Solvers for Solving a System of Boolean Algebraic Equations*

Any SAT problem is based on two key stages—checking the feasibility of an arbitrary Boolean function represented in conjunctive normal form (CNF) and finding a set of values at which such CNF is executed. Many SAT solvers are based on the DPLL (Devis, Putnam, Logemann, Loveland) algorithm, which was developed in 1962 precisely to determine the feasibility of Boolean formulas in CNF. For more than half a century, the DPLL algorithm has been the basis for most effective solvers for SAT problems. The main idea of the DPLL algorithm is to use methods to bypass the search tree in depth and apply the single clause rule [34]. The DPLL algorithm splits the set of sought CNF variables into two subsets, A and B, where variables with the value "true" are included in the set A, and variables with the value "false" in the set B. At each step, an arbitrary variable of the CNF is selected and the value "true" is assigned to it (adding a variable to subset A). Then the initial formula is simplified, and the simplified problem is solved. If the CNF obtained after simplification is feasible, then the value of the variable is chosen correctly, otherwise the selected variable is assigned the value "false" and it is transferred to the subset B. The task is solved again for the selected value of the "false" variable. Thus, it will either find the correct value of the variable ("true" or "false"), or it will be proved that the original formula is not feasible.

Each time a variable is checked, the original CNF is simplified according to the following two rules:

1. Variable propagation. If there is only one variable left in the sentence, assign it such a value that the sentence becomes true (put the variable in the subset A if there is no negation in the sentence, or put it in the set B if there is negation).
2. The elimination of "pure variables". If a variable is found in the formula with only negation or only without negation, then it is called "pure" and it can be assigned such a value that it is always "true" (in this way we reduce the number of free variables).

If, after simplification, an empty clause is received (i.e., all simple conjuncts are false), the formula is not feasible and we return to the previous step. If no free variables remain, then the formula is considered feasible, and the operation of the algorithm can be stopped. If there is no disjoint left (uncommitted free variables can be set arbitrarily), then the check of the feasibility of the CNF also ends.

Consider the representation of the addition operation modulo two in CNF, and let the formula be given

$$x \oplus y \oplus z,$$

then it should be presented in the form of clauses:

$$\begin{array}{ll} x \vee \overline{y} \vee \overline{z}, & \overline{x} \vee \overline{y} \vee z, \\ x \vee y \vee z, & \overline{x} \vee y \vee \overline{z}. \end{array}$$

You will need to create 2^{n-1} clauses to describe addition modulo two lengths n.

In order to reduce the number of clauses in the SAT representation, the fragmentation of the modulo two addition operation is used.

The formula of the form

$$x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7,$$

can be represented in the SAT solver, as:

$$\begin{array}{l} \overline{y}_0 \oplus x_0 \oplus x_1 \oplus x_2 \oplus x_3, \\ \overline{y}_1 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7, \\ y_0 \oplus y_1, \end{array}$$

where $y_0 = x_0 \oplus x_1 \oplus x_2 \oplus x_3$, $y_1 = x_4 \oplus x_5 \oplus x_6 \oplus x_7$.

To simplify the search for solution sets using SAT solvers, a search on variables is also used instead of a search on literals.

For example, to represent the formula:

$$x_1 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_3,$$

each product of the unknowns should be represented as a separate clause, i.e., replaced with additional $i_1, \dots, i_3$ variables

$$\begin{array}{l} x_1 \oplus i_1 \oplus i_2 \oplus i_3, \\ i_1 = x_1x_2, \\ i_2 = x_2x_3, \\ i_3 = x_1x_3. \end{array}$$

We use the CryptoMiniSat package in the SageMath environment to solve a system of Boolean equations as an SAT problem. CryptoMiniSat is a DPLL-based SAT solver based on the MiniSat. The fundamental differences between CryptoMiniSat and MiniSat solver are as follows:

1. Clauses of addition modulo two are distinguished at the beginning of the search for solutions. They have their own separate search list, a separate extension mechanism, and a categorization algorithm. The use of such opportunities leads to a speed increase in searching for solutions of Boolean equations systems.
2. Clauses of addition modulo two (binary) are processed by special methods. First, the search is usually performed using special heuristics. Secondly, a tree structure is constructed of them, reflecting which of the variables is equivalent or anti-valent. The upper level of the constructed trees is usually replaced by lower values in the tree, thereby reducing the number of classes and variables in the analyzed task. This usually leads to the necessity of reassigning variables.

3. Technical and cryptographic SAT problems are very different, so CryptoMiniSat allows you to change the restart settings and change the type of learning heuristics using the Glucose or MiniSat training methods.
4. Clauses are removed from CNF as soon as at least one of the literals included in this clause takes a value equal to true. Unlike the MiniSat, the literals equal to false are also deleted in the clauses, thereby allowing the clause to be reduced.
5. The removal of dependent variables is carried out among the associated clauses of addition modulo two. Dependent variables are variables that are found only in one clause of addition modulo two. This simplification allows you to remove the variable from the task. It should be remembered that such a variable cannot be removed by using the exclusion of "pure" literals.
6. Variables take values "false" and "true" at fixed intervals. If one of the search branches leads to an error (returning an impracticable formula), then the second branch is checked. Moreover, the results of checking both branches of the search are saved for subsequent comparison.

The proposed algorithm for representing the transformations of substitutions allows us to form a system of Boolean equations describing the transformations in an arbitrary S-box. Using SAT solvers for algebraic analysis can be described in the following steps:

- Representation of cryptographic transformation as a system of Boolean equations in the algebraic normal form (ANF).
- Convert the equation system from ANF to CNF.
- Solve a SAT problem to find a set of solutions by SAT solvers.

After the generation of the system of Boolean nonlinear equations, we substitute the input and output vectors of the S-block through known text pairs (plaintexts and ciphertexts) using a knowledge of encryption algorithm structure. At this stage, a system of Boolean equations presented in algebraic normal form (ANF) is obtained.

To use existed SAT solvers, we should convert the formed system of Boolean equations (in the ANF) to CNF. First, we should simplify the presentation of the equations generated for block ciphers in ANF.

In cryptographic tasks, the application of the following algorithm for converting from ANF to CNF turned out to be effective [35]:

1. Replacing constant 1 by a new unknown, since CNF should not contain constants.
2. Replacing all products of unknowns by new variables (apply the linearization method to the original nonlinear system).
3. The splitting of long chains formed as the addition modulo two unknowns into substrings of shorter length (for example, only four unknowns).
4. Representation of the transformed system in CNF.

In general terms, it can be said that a 2^{p-1} clause is required to represent the sum of unknowns with a length of p. Defragmentation of long sum chains (length l) for the equation $x_0 \oplus x_1 \oplus x_2 \oplus \ldots \oplus x_{l-1} = 0$ takes the form:

$$\begin{aligned}
&x_0 \oplus x_1 \oplus x_2 \oplus y_0 = 0,\\
&y_0 \oplus x_5 \oplus x_6 \oplus y_1 = 0,\\
&\ldots\\
&y_{i-1} \oplus x_{4i+1} \oplus x_{4i+2} \oplus y_i = 0,\\
&\ldots\\
&y_h \oplus x_{l-3} \oplus x_{l-2} \oplus x_{l-1} = 0,
\end{aligned}$$

where $x_0, \ldots, x_{l-1}$—are the original unknowns, $y_0, \ldots, y_h$—are the new variables used to reduce the number of terms in the sum, l–is the number of the original unknowns, and $h + 1$—is the number of new variables.

To represent the resulting system in CNF, you can use the anf2cnf conversion library [36–38]. Then the system of equations in CNF is transferred to the SAT solver algorithm. We chose CryptoMiniSat 2.5 as one of the most efficient SAT solvers for cryptotasks. We made an experiment in the cloud environment SageMath Cloud [38].

4. Generation of a System of Boolean Equations Describing an Encryption Algorithm

The first step of algebraic analysis is the generation of a system of equations linking known data (plaintexts and ciphertexts) and an encryption key. For most encryption algorithms, a system of equations is constructed for substitution boxes because they are often the only non-linear encryption transformation.

Denote by x_i, y_i bits of the input and output vector of the substitution box over the field GF(2^s), where $i \in N, 0 \leq i \leq s-1$. We need to present the substitution operation in S-boxes in the form of a subsystem of equations valid with probability 1 for all possible input and corresponding output values of the observed S-Box. The common form of the equation describing the transformations in the S-Box can be given by the formula:

$$\sum_{i,j=0}^{s-1} a_{i,j}x_ix_j \oplus \sum_{i,j=0}^{s-1} \beta_{i,j}y_iy_j \oplus \sum_{i,j=0}^{s-1} \gamma_{i,j}x_iy_j \oplus \sum_{i=0}^{s-1} \delta_i x_i \oplus \sum_{i=0}^{s-1} \varepsilon_i y_i \oplus \eta = 0$$

where x_ix_j is the multiplication of the input bits of the S-box, y_iy_j is the multiplication of the output bits of the S-box, x_iy_j is the multiplication of the input and output bits, x_i and y_i are the input and output bits of the S-box, respectively, and $\alpha, \beta, \gamma, \delta, \varepsilon, \eta$ are coefficients taking values of 0 or 1.

As part of the research, it is enough to consider the multiplication of two variables, however, for some algorithms (for which the algebraic immunity of substitution box is three), it may be necessary to increase the number of monomials used by including the product of three variables ($x_ix_jy_k, x_iy_jy_k$) into the equations.

In this case, equations will be given by the formula:

$$\begin{aligned} &\sum_{i,j=0}^{s-1} a_{i,j,k}x_iy_jy_k \oplus \sum_{i,j=0}^{s-1} \beta_{i,j,k}x_ix_jy_k \oplus \sum_{i,j=0}^{s-1} \gamma_{i,j}x_iy_j \oplus \\ &\oplus \sum_{i=0}^{s-1} \delta_i x_i \oplus \sum_{i=0}^{s-1} \varepsilon_i y_i \oplus \eta = 0. \end{aligned}$$

For a substitution box with an input size of s bits, we will get 2^t possible equations, where t is the number of monomials in the system of equations. The parameter, t, is calculated by the formula:

$$t = \binom{2s}{2} + 2s + 1.$$

When the block size is four bits, the number of monomials in the system is $t = 37$. Therefore, it is possible to make no more than 2^{37} = 137,438,953,472 quadratic equations. Then, to select from the total number of possible equations, a truth table is formed using only transformations valid to the used substitution box. A general view of the truth table is shown in Table 4.

Some of the found equations, which were valid to the S-box substitution table, turned out to be linearly dependent and were not suitable for further use for algebraic analysis. It was necessary to choose only linearly independent equations for inclusion in the resultant system describing the transformations in the substitution box. When choosing linearly independent equations, we use the following condition [39]:

For any substitution box $S(x_1, \ldots, x_s) \to (y_1, \ldots, y_h)$, if the condition $t > 2^s$ is satisfied, then there are at least $t - 2^s$ linearly independent equations valid for all input values of the substitution box.

Table 4. Truth table for an s-bit S-block.

				S-Block Input			**S-Block Output**			**All Compositions of S-Block Inputs and Outputs**					
	x_s	…	x_1	y_s	…	y_1	$x_s x_{s-1}$	…	$x_2 \times 1$	$y_s y_{s-1}$	…	$y_2 y_1$	$x_s y_s$	…	$x_1 y_1$
All possible S-block inputs (from 0 to 2^s)	0	…	0	1	…	1									
								…							
	1	…	1	0	…	1									

For the algebraic analysis of a GOST R 34.12-2015 ($n = 64$) cipher, it will be necessary to expand the system of equations by including the bitwise dependence between the input substitution bits, plaintext bits (or the input round vector), and the secret round key, i.e., to include the equations connecting addition modulo 2^{32}. Consider three vectors of n-bit size $X = (x_0 \ldots, x_{n-1})$, $Y = (y_0 \ldots, y_{n-1})$, $Z = (z_0 \ldots, z_{n-1})$: for which addition modulo 2^n is performed:

$$Z = X + Y \mathrm{mod} 2^n.$$

In the modulo 2^n addition operation, each result bit z_i depends on the previous bits $x_{n-1}, \ldots, x_i, y_{n-1}, \ldots, y_i$.

Such transformations can be described as follows through two subsystems [40]:

$$\begin{cases} \mathrm{z_{n-1}} = \mathrm{x_{n-1}} \oplus \mathrm{y_{n-1}}, \\ \mathrm{z_{n-2}} = \mathrm{x_{n-2}} \oplus \mathrm{y_{n-2}} \oplus \mathrm{c_{n-2}}, \\ \mathrm{z_{n-3}} = \mathrm{x_{n-3}} \oplus \mathrm{y_{n-3}} \oplus \mathrm{c_{n-3}}, \\ \ldots \\ \mathrm{z_i} = \mathrm{x_i} \oplus \mathrm{y_i} \oplus \mathrm{c_i}, \\ \ldots \\ \mathrm{z_0} = \mathrm{x_0} \oplus \mathrm{y_0} \oplus \mathrm{c_0}. \end{cases} \Rightarrow \begin{cases} \mathrm{c_{n-2}} = \mathrm{x_{n-1}} \mathrm{y_{n-1}}, \\ \mathrm{c_{n-3}} = \mathrm{x_{n-2}} \mathrm{y_{n-2}} \oplus (\mathrm{x_{n-2}} \oplus \mathrm{y_{n-2}}) \mathrm{c_{n-2}}, \\ \ldots \\ \mathrm{c_i} = \mathrm{x_{i-1}}\, \mathrm{y_{i-1}} \oplus (\mathrm{x_{i-1}} \oplus \mathrm{y_{i-1}}) \mathrm{c_{i-1}}, \\ \ldots \\ \mathrm{c_0} = \mathrm{x_1} \mathrm{y_1} \oplus (\mathrm{x_1} \oplus \mathrm{y_1}) \mathrm{c_1}, \end{cases}$$

where $c_{n-2}, \ldots, c_0$ is the transfer coefficients between digits.

In virtue of the considered above systems, it can be noted that the value c_i can be immediately expressed through the remaining unknowns and simplified. In this case, we can create the following system describing the transformation of addition modulo 2^n:

$$\begin{cases} z_{n-1} = x_{n-1} \oplus y_{n-1}, \\ z_{n-2} = x_{n-2} \oplus y_{n-2} \oplus x_{n-1} y_{n-1}, \\ z_{n-3} = x_{n-3} \oplus y_{n-3} \oplus x_{n-1} y_{n-1} \oplus (x_{n-2} \oplus y_{n-2})(x_{n-2} \oplus y_{n-2} \oplus z_{n-2}), \\ \vdots \\ z_i = x_i \oplus y_i \oplus x_{i+1} \oplus y_{i+1} \oplus (x_{i+1} \oplus y_{i+1})(x_{i+1} \oplus y_{i+1} \oplus z_{i+1}), \\ \vdots \\ z_0 = x_0 \oplus y_0 \oplus x_1 y_1 \oplus (x_1 \oplus y_1)(x_1 \oplus y_1 \oplus z_1). \end{cases}$$

For any n in system of equations, one linear and $n - 1$ quadratic equations are additionally obtained. Thus, to use bitwise dependencies for 32-bit vectors of round keys, the system of equations will include an extra 32 equations for each round key. In this case, for each round key a new variable in the system will be used (an additional 32 unknowns).

5. Assessment Approaches by Algebraic Analysis Methods

In the course of this research, a methodology was proposed for conducting algebraic analysis based on the application of the XL method and SAT solvers [41]. Two main encryption operations (substitution primitives and addition modulo 2^n) were considered.

The initial data for the approach are:

- The mathematical structure of the encryption algorithm;
- the structure of the substitution operations (as they are defined in the algorithm);
- available known data (the number of plaintext-ciphertext pairs).

The resulting characteristics of the approach are:

- the encryption key value (or some sets of possible values);
- parameters of the nonlinear Boolean equation system: the numbers of equations, unknowns, and monomials;
- the computational complexity of solving the Boolean equation system;
- the time complexity of solving the Boolean equation system;
- the minimal number of known data (text pairs) that are needed to find the key in a reasonable time;
- the required RAM for analysis.

General diagram of proposed methodology is shown in Figure 4.

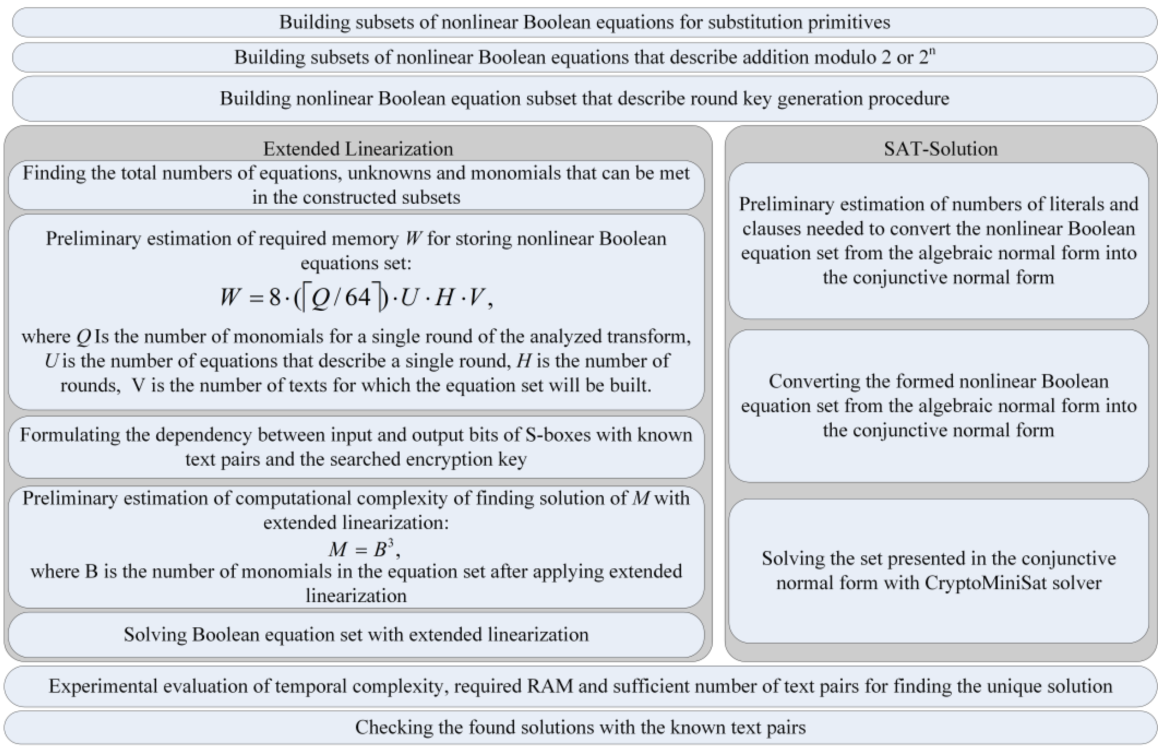
Building subsets of nonlinear Boolean equations for substitution primitives
Building subsets of nonlinear Boolean equations that describe addition modulo 2 or 2^n
Building nonlinear Boolean equation subset that describe round key generation procedure
Extended Linearization
Finding the total numbers of equations, unknowns and monomials that can be met in the constructed subsets
Preliminary estimation of required memory W for storing nonlinear Boolean equations set:
$W = 8 \cdot (\lceil Q/64 \rceil) \cdot U \cdot H \cdot V,$
where Q Is the number of monomials for a single round of the analyzed transform, U is the number of equations that describe a single round, H is the number of rounds, V is the number of texts for which the equation set will be built.
Formulating the dependency between input and output bits of S-boxes with known text pairs and the searched encryption key
Preliminary estimation of computational complexity of finding solution of M with extended linearization:
$M = B^3,$
where B is the number of monomials in the equation set after applying extended linearization
Solving Boolean equation set with extended linearization
SAT-Solution
Preliminary estimation of numbers of literals and clauses needed to convert the nonlinear Boolean equation set from the algebraic normal form into the conjunctive normal form
Converting the formed nonlinear Boolean equation set from the algebraic normal form into the conjunctive normal form
Solving the set presented in the conjunctive normal form with CryptoMiniSat solver
Experimental evaluation of temporal complexity, required RAM and sufficient number of text pairs for finding the unique solution
Checking the found solutions with the known text pairs

Figure 4. Security assessment methodology approach.

SageMath [42] was chosen as the software development environment. The algebraic cryptanalysis was implemented by using the functions of sage.sat.converters.polybori for transforming the equation set from ANF into CNF (CNFEncoder). We applied sage.sat.boolean_polynomials library [43,44] for access to the functionality of the SAT solver CryptoMiniSat. An IntelCore i5 2.8 GHz 8 GByte PC was used as a test bench, on which we received some numerical experimental results of the algebraic cryptanalysis application to Magma cipher (Table 5).

Table 5. Algebraic analysis of Magma.

	SAT Method								XL Method	
Number of Known Text Pairs	**Number of Equations**	**Number of Unknowns**	**Number of Literals**	**Number of Clauses**	**Number of Solutions**	**Total Time, s**	**Search Time, s**	**RAM, GByte**	**Complexity by Number of Additions**	**Search Time, s**
Three rounds of Magma ⊕										
2	1008	160	2997	52,438	32	20	0.26	0.11	$2^{36,33}$	0.86
3	1512	192	4306	78,516	2	32.7	0.52	0.14	$2^{38,08}$	2.91
4	2016	224	5512	104,654	1	38.51	0.48	0.16	$2^{39,32}$	6.86
Three rounds of Magma (with weakened S-boxes)										
2	1200	352	1470	9400	8	28.51	0.35	0.13	$2^{36,67}$	1.09
3	1800	480	1965	14,084	1	39.91	0.47	0.14	$2^{38,43}$	3.71
Three rounds of Magma (with S-box S(X) = X)										
2	1200	352	1292	7393	2048	21.11	2.10	0.28	236.67	1.09
3	1800	480	1889	11,014	1	37.19	1.07	0.16	238.43	3.71
Three rounds of Magma										
2	1200	352	2819	38,811	1	36.55	0.35	0.13	$2^{36,67}$	1.09
Four rounds of Magma ⊕										
2	1344	256	6492	114,472	256	52.59	1.93	0.32	$2^{37,57}$	2.04
3	2016	320	9435	171,324	2	106.33	0.71	0.39	$2^{39,33}$	6.91
4	2688	384	12,384	228,358	1	135.42	0.69	0.48	$2^{40,57}$	16.32
Four rounds of Magma (with weakened S-boxes)										
2	1600	512	2940	16,642	128	96.10	0.77	0.31	$2^{37,92}$	2.6
3	2400	704	3869	24,982	2	142.36	0.88	0.35	$2^{39,67}$	8.75
4	3200	896	4798	33,316	2	195.61	1.23	0.36	$2^{40,92}$	20.79
5	4000	1088	5725	41,646	2	294.72	1.51	0.38	$2^{41,88}$	40.46
6	4800	1280	6839	49,922	2	612.03	4.11	0.39	$2^{42,67}$	69.98
7	5600	1472	7951	58,174	1	700.96	2.63	0.41	$2^{43,34}$	111.43
Four rounds of Magma (with S-box S(X) = X)										
3	2400	704	3726	19,280	4	106.76	1.33	0.28	$2^{39,67}$	8.75
4	3200	896	4606	25,830	2	144.66	1.10	0.30	$2^{40,92}$	20.79
5	4000	1088	5484	32,294	1	242.21	1.24	0.35	$2^{41,88}$	40.46

Table 5. *Cont.*

				SAT Method					XL Method	
Number of Known Text Pairs	**Number of Equations**	**Number of Unknowns**	**Number of Literals**	**Number of Clauses**	**Number of Solutions**	**Total Time, s**	**Search Time, s**	**RAM, GByte**	**Complexity by Number of Additions**	**Search Time, s**
				Four rounds of Magma						
2	1600	512	4839	67,886	4	93.44	10.27	0.37	$2^{37,91}$	2.58
3	2400	704	7202	101,968	1	202.56	1.32	0.48	$2^{39,67}$	8.75
				Five rounds of Magma $\oplus$						
3	2520	448	14,696	266,200	4	452.74	84.41	1.38	$2^{40,29}$	13.44
4	3360	554	19,448	354,354	2	729.15	124.47	1.44	$2^{41,54}$	31.97
5	4200	640	24,259	443,932	1	797.21	24.66	1.52	$2^{42,50}$	62.20
				Five rounds of Magma (with weakened S-boxes)						
3	3000	928	5932	39,549	4	320.17	4.15	1.62	$2^{40,64}$	17.14
4	4000	1184	7436	52,872	2	488.65	4.47	1.76	$2^{41,88}$	40.46
5	5000	1440	9194	66,113	2	478.31	2.51	1.87	$2^{42,85}$	79.25
6	6000	1696	10,705	79,506	2	875.24	6.91	2.13	$2^{43,64}$	137.09
7	7000	1952	12,459	92,707	1	1135.61	3.36	2.26	$2^{44,30}$	216.27
				Five rounds of Magma (with S-box S(X) = X)						
3	3000	928	5338	32,873	40	678.80	520.89	1.73	$2^{40,64}$	17.14
4	4000	1184	7068	43,926	2	415.31	28.65	1.82	$2^{41,88}$	40.46
5	5000	1440	8787	54,647	1	501.18	4.92	1.85	$2^{42,85}$	79.25
				Five rounds of Magma						
3	3000	928	10,596	152,045	1	394.68	24.96	1.42	$2^{40,64}$	17.14
				Eight rounds of Magma (with weakened S-boxes)						
4	5376	2048	15,395	110,844	4096	5972.41	1843.67	4.59	$2^{43,92}$	166.3
				Eight rounds of Magma (with S-box S(X) = X)						
4	5376	2048	13,764	92,370	1024	4842.31	1374.12	3.86	$2^{43,92}$	166.3
				Eight rounds of Magma						
4	5376	2048	30,062	431,267	1	3029.56	416.31	3.6	$2^{43,92}$	166.3

The algebraic analysis of an eight-round Magma (68 key bits were fixed) with a CryptoMiniSat solver demanded four known text pairs and took 3029.56 s to complete (the search took 416.31 s). During the analysis of an eight round Magma, 68 key bits were fixed: 0–15, 51–55, 64–66, 128–130, 179–183, 192–207, 224–231, and 244–255. The algebraic analysis of a five-round Magma cipher with weakened S-boxes required seven known text pairs and took 1135.61 s (the search took 3.36 s). The algebraic analysis of a five-round Magma cipher with disabled S-blocks (equivalent value substitution) led to getting only one solution for five known text pairs in 501.18 s (the search took 4.92 s).

As seen from the experimental results, to find the only one existing solution for disabled and weakened S-boxes, we need to add more known data (text pairs) to the SAT solvers.

The results of the experiments of time and memory complexity of the Magma cipher algebraic analysis are presented in Figure 5.

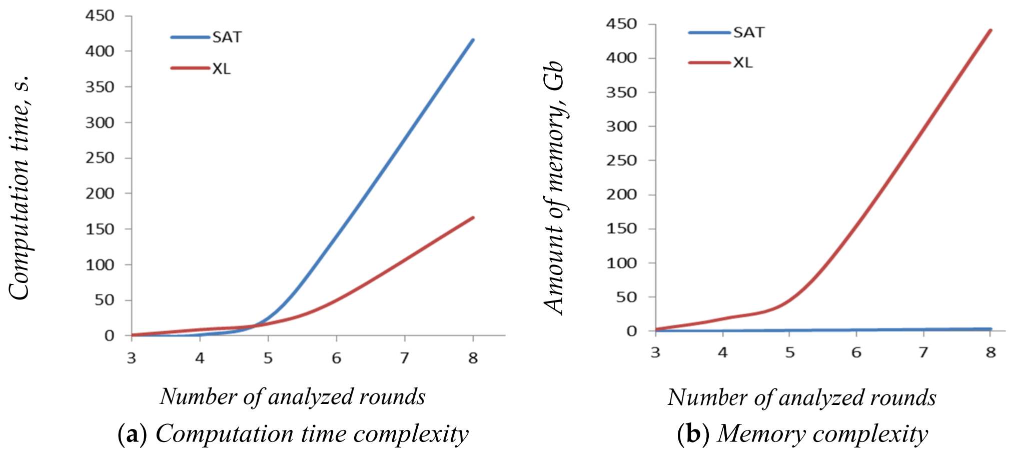

Figure 5. Search complexity with different numbers of Magma rounds.

The algebraic analysis of Magma ⊕ encryption has the following obtained time dependence for the search for all sets of solutions (Figure 6).

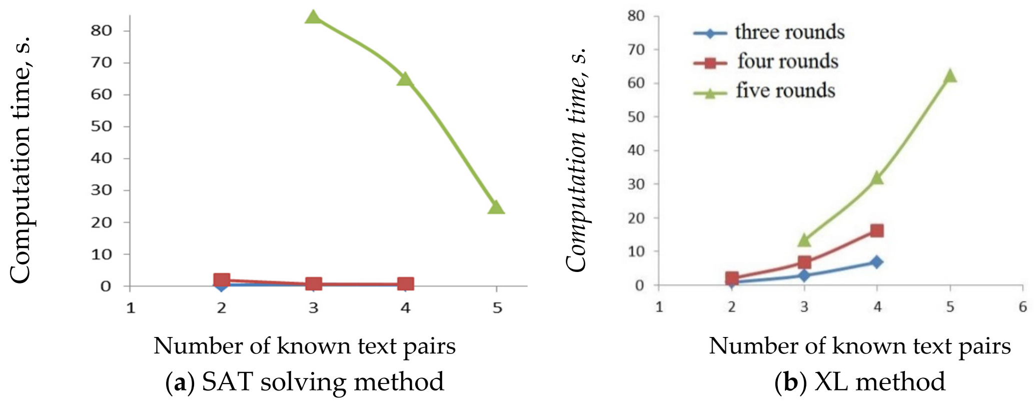

Figure 6. Search complexity with different numbers of Magma rounds.

The experimental dependences between the algebraic analysis total time complexity and the number of known plaintexts for the Magma ⊕ algorithm are presented in Figure 7.

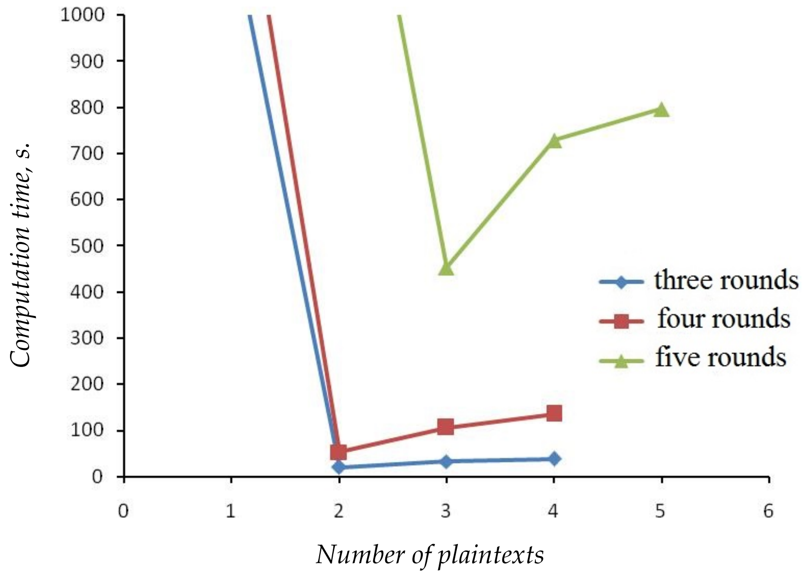

Figure 7. Dependences between the total computation time of the algebraic analysis and the numbers of known plaintexts.

We evaluated the complexity of the algebraic analysis (by the XL method) for a simplified version of the Kuznyechik algorithm, namely S-KN2, without a round subkey generation scheme. For the substitution box (Table 2) we generated the following quadratic system of equations (including 21 linear independent equations):

$$
\begin{aligned}
&x1 \oplus x3 \oplus y1 = 0,\\
&x0*x2 \oplus x1*x2 \oplus x1*x3 \oplus x0*y1 \oplus y2 \oplus 1 = 0,\\
&x0*x1 \oplus x0*x3 \oplus x0*y1 = 0,\\
&x0*x2 \oplus x1*x2 \oplus x0*y1 \oplus x1*y1 \oplus x1 \oplus y2 \oplus 1 = 0,\\
&x0*x3 \oplus x0*y1 \oplus x0*y2 \oplus x1*y2 \oplus x0 \oplus x1 \oplus y2 \oplus 1 = 0,\\
&x0*y3 \oplus x1*y3 \oplus x0 \oplus x1 \oplus x2 \oplus x3 \oplus y2 \oplus y3 = 0,\\
&x0*x3 \oplus x2*x3 \oplus x0 \oplus x1 \oplus x3 \oplus y0 \oplus y2 \oplus y3 = 0,\\
&x1*x2 \oplus x1*y0 \oplus x2*y0 \oplus x0 \oplus x2 \oplus x3 \oplus y0 \oplus y3 \oplus 1 = 0,\\
&x1*x2 \oplus x0*x3 \oplus x2*y1 \oplus x0 \oplus x1 \oplus x3 \oplus y0 \oplus y2 \oplus y3 = 0,\\
&x0*x2 \oplus x0*x3 \oplus x0*y0 \oplus x1*y0 \oplus x0*y1 \oplus x0*y2 \oplus x2*y2 \oplus x1 \oplus x3 \oplus y2 \oplus y3 = 0,\\
&x0*y3 \oplus x2*y3 \oplus x1 \oplus x2 \oplus y0 \oplus y2 \oplus 1 = 0,\\
&x1*x2 \oplus x0*y0 \oplus x1*y0 \oplus x3*y0 \oplus x0*y3 \oplus x0 \oplus x1 \oplus x2 \oplus x3 \oplus y2 \oplus y3 = 0,\\
&x0*x2 \oplus x1*x2 \oplus x0*y1 \oplus x3*y1 \oplus x3 \oplus y2 \oplus 1 = 0,\\
&x0*y0 \oplus x1*y0 \oplus x0*y1 \oplus x3*y2 \oplus x0 \oplus x1 \oplus x2 \oplus y3 \oplus 1 = 0,\\
&x0*x2 \oplus x0*x3 \oplus x0*y3 \oplus x3*y3 \oplus x0 \oplus x1 \oplus x3 \oplus y0 \oplus y2 \oplus y3 = 0,\\
&x1*x2 \oplus x0*y0 \oplus y0*y1 \oplus x0*y3 \oplus x0 \oplus x1 \oplus x2 \oplus x3 \oplus y2 \oplus y3 = 0,\\
&x0*x2 \oplus x1*y0 \oplus x0*y1 \oplus x0*y2 \oplus y0*y2 \oplus x0*y3 \oplus x0 \oplus x1 \oplus y0 \oplus y2 \oplus 1 = 0,\\
&x0*x2 \oplus x0*x3 \oplus y0*y3 \oplus x1 \oplus y2 \oplus 1 = 0,\\
&x0*x3 \oplus x0*y0 \oplus x1*y0 \oplus x0*y2 \oplus y1*y2 \oplus x2 \oplus y2 \oplus y3 = 0,\\
&x0*x2 \oplus x0*x3 \oplus y1*y3 \oplus x2 \oplus y0 = 0,\\
&x0*x2 \oplus x0*x3 \oplus y2*y3 \oplus x0 \oplus x2 \oplus x3 \oplus 1 = 0,
\end{aligned}
$$

For one encryption round at a substitution layer, we get 84 linear independent equations with 32 variables. At the next layer L-transformation, we increase the number of equations by adding four equations with 16 new bit variables:

$$a'_0 = 4{*}a_3 \oplus a_2 \oplus 3{*}a_1 \oplus a_0.$$

$$a'_1 = 4{*}a'_0 \oplus a_3 \oplus 3{*}a_2 \oplus a_1.$$

$$a'_2 = 4{*}a'_1 \oplus a'_0 \oplus 3{*}a_3 \oplus a_2.$$

$$a'_3 = 4{*}a'_2 \oplus a'_1 \oplus 3{*}a'_0 \oplus a_3.$$

The estimations of the complexity of solving the system by the XL method and maximum required memory are given in Table 6.

Table 6. Evaluated complexity of the algebraic analysis of the S-KN2 cipher by the XL method.

Number of Known Text Pairs	Number of Equations	Number of Unknowns	Complexity by Number of Additions	RAM, GByte
		One-round S-KN2 cipher		
1	88	32	$2^{25,57}$	0.001
2	176	48	$2^{28,32}$	0.008
3	264	64	$2^{30,12}$	0.028
		Two-round S-KN2 cipher		
1	176	64	$2^{28,57}$	0.008
2	352	96	$2^{31,57}$	0.066
3	528	128	$2^{33,33}$	0.224
		Three-round S-KN2 cipher		
1	264	80	$2^{30,33}$	0.028
2	528	112	$2^{33,16}$	0.187
3	792	144	$2^{35,08}$	0.756
		Four-round S-KN2 cipher		
1	352	112	$2^{31,57}$	0.066
2	704	160	$2^{34,57}$	0.531
3	1056	208	$2^{36,33}$	1.191

6. Conclusions

In this article, we described the main steps of the algebraic analysis of cipher reliability and observed the Russian block encryption standard GOST R 34.12-2015 (n = 64, Magma and n = 128, Kuznyechik). We presented algorithms that can be used to teach the principles of the Kuznyechik block cipher (GOST R 34.12-2015). The S-KN2 algorithm can be used to illustrate popular cryptanalysis attacks against Kuznyechik and other similar block ciphers.

We observed approaches to implementing algebraic analysis to symmetric block ciphers: linearization, extended linearization, extended sparse linearization, and SAT solving. We proposed the experimental results of finding encryption keys with SAT solvers and extended linearization using Magma block ciphers. As examples for the experiment, we chose three fillings of Magma cipher substitution boxes (one from the standard, substitution with equivalent values S(X) = X, and a weak one) and simplified version Magma ⊕. We described the reduction of encryption transformation to a SAT problem. The number of literals and clauses, which are encountered with different numbers of known text pairs, was found. We also computed the evaluated complexity of the algebraic analysis of the S-KN2 cipher by the XL method for some known plaintext number.

The proposed approaches and algorithms can be further used for the security assessment of arbitrary ciphers based on substitutions and addition modulo 2^n in terms of their resistance to algebraic cryptanalysis.

Author Contributions: Methodology, E.M.; software, P.P.; validation, E.I., E.M. and P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Southern Federal University.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Courtois, N. Algebraic Complexity Reduction and Cryptanalysis of GOST. Available online: http://www.nicolascourtois.com/papers/gostac11.pdf (accessed on 27 May 2020).
2. Shannon, C.E. Communication Theory of Secrecy Systems. *Bell Syst. Tech. J.* **1949**, *28*, 656–715. [CrossRef]
3. Pasalic, E. On Cryptographically Significant Mappings over GF(2). Available online: Arithmetic of Finite Fields. In Proceedings of the Second International Workshop, WAIFI, Siena, Italy, 6–9 July 2008; pp. 189–204.
4. Arabnezhad-Khanoki, H.; Sadeghiyan, B.; Pieprzyk, J. Algebraic Attack Efficiency Versus S-Box Representation. Available online: https://eprint.iacr.org/2017/007.pdf (accessed on 27 May 2020).
5. Dehnavi, S.M.; Rishakani, A.M.; Mirzaeeshamsabad, M.R.; Maimani, H.; Pasha, E. Cryptographic Properties of Addition Modulo 2n. Available online: https://eprint.iacr.org/2016/181.pdf (accessed on 27 May 2020).
6. Greve, B.; Ytrehus, Ø.; Raddum, H. Variable Elimination-a Tool for Algebraic Cryptanalysis. Available online: https://eprint.iacr.org/2019/112.pdf (accessed on 27 May 2020).
7. Bernstein, D.J.; Yang, B.-Y. Asymptotically Faster Quantum Algorithmsto Solve Multivariate Quadratic Equations. Available online: https://eprint.iacr.org/2017/1206.pdf (accessed on 27 May 2020).
8. Cheng, C.-M.; Chou, T.; Niederhagen, R.; Yang, B.-Y. Solving Quadratic Equations with XLon Parallel Architectures. Extended Version. Available online: https://eprint.iacr.org/2016/412.pdf (accessed on 27 May 2020).
9. Liu, F.; Isobe, T.; Meier, W.; Yang, Z. Algebraic Attacks on Round-ReducedKeccak/Xoodoo. Available online: https://eprint.iacr.org/2020/346.pdf (accessed on 27 May 2020).
10. Arabnezhad-Khanoki, H.; Sadeghiyan, B. Toward a More Efficient Gröbner-Based Algebraic Cryptanalysis. Available online: https://eprint.iacr.org/2019/1415.pdf (accessed on 27 May 2020).
11. Albrecht, M.R.; Cid, C.; Grassi, L.; Khovratovich, D.; Lüftenegger, R.; Rechberger, C.; Schofnegger, M. Algebraic Cryptanalysis of STARK-Friendly Designs: Application toMARVELlous and MiMC. Available online: https://eprint.iacr.org/2019/419.pdf (accessed on 27 May 2020).
12. Matheis, K.; Steinwandt, R.; Suárez Corona, A. Algebraic Properties of the Block Cipher DESL. *Symmetry* **2019**, *11*, 1411. [CrossRef]
13. Soeken, M. Determining the Multiplicative Complexity of Boolean Functions Using SAT. Available online: https://eprint.iacr.org/2020/530.pdf (accessed on 27 May 2020).
14. Schaffhauser, A. SAT Solvers and their Limits with NFSR-based Stream Ciphers: An Example with Grain v1. In Proceedings of the Third Central European Cybersecurity Conference (CECC 2019), Munich, Germany, 15–16 November 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 1–5. [CrossRef]
15. Pavlenko, A.; Semenov, A.; Ulyantsev, V. Evolutionary Computation Techniques for Constructing SAT-Based Attacks in Algebraic Cryptanalysis. In *Applications of Evolutionary Computation*; EvoApplications 2019; Lecture Notes in Computer Science; Kaufmann, P., Castillo, P., Eds.; Springer: Cham, Switzerland, 2019; Volume 11454.
16. Horáček, J. Algebraic and Logic Solving Methods for Cryptanalysis. Ph.D. Thesis, University of Passau, Passau, Germany, 2020. Available online: https://opus4.kobv.de/opus4-uni-passau/files/773/horacek_dissertation.pdf (accessed on 27 May 2020).
17. Ishchukova, E.; Babenko, L.; Anikeev, M. Two Simplified Versions of Kuznyechik Cipher (GOST R 34.12-2015). In Proceedings of the 10th International Conference on Security of Information and Networks, Jaipur, India, 13–15 October 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 287–290. [CrossRef]
18. Information Technology Cryptographic Data Security Block Ciphers English Version. Available online: https://tc26.ru/upload/iblock/fc9/GOST_R_34_12_2015_ENG.pdf (accessed on 27 May 2020).
19. Stallings, W. *Cryptography and Network Security: Principles and Practice*; Prentice Hall Pub: Upper Saddle River, NJ, USA, 1998; p. 562.

20. Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students, by Raphael Chung-Wei Phan. *Cryptologia* **2002**, *26*, 283–306. [CrossRef]
21. Musa, M.A.; Schaefer, E.F.; Wedig, S. A simplified AES algorithm and its linear and differential cryptanalyses. *Cryptologia* **2003**, *27*, 148–177. [CrossRef]
22. Gilbert, H. A Simplified Representation of AES. In *Advances in Cryptology–ASIACRYPT 2014*; Lecture Notes in Computer Science; Sarkar, P., Iwata, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8873.
23. Sunjiv Soyjaudah, K.M.; Sumithra Devi, K.A. Overview of Linear Cryptanalysis on S-DES and Block Ciphers using Hill Cipher Method. *Int. J. Comput. Appl.* **2013**, *63*.
24. Ooi, K.S.; Vito, B.C. Cryptanalysis of S-DES. *IACR Cryptol. ePrint Arch.* **2002**, *2002*, 45. Available online: https://eprint.iacr.org/2002/045.pdf (accessed on 27 May 2020).
25. Nalini, N.; Rao, G.R. Cryptanalysis of Simplified Data Encryption Standard via Optimisation Heuristics. *IJCSNS Int. J. Comput. Sci. Netw. Secur.* **2006**, *6*, 240–246.
26. Vimalathithan, R.; Valarmathi, M.L. Cryptanalysis of S-DES using Genetic Algorithm. *Int. J. Recent Trends Eng.* **2009**, *2*, 76–79.
27. Dworak, K.; Boryczka, U. Cryptanalysis of SDES Using Modified Version of Binary Particle Swarm Optimization. In *Computational Collective Intelligence*; Lecture Notes in Computer Science; Núñez, M., Nguyen, N., Camacho, D., Trawiński, B., Eds.; Springer: Cham, Switzerland, 2015; Volume 9330.
28. Phan, R.C.-W. Impossible Differential Cryptanalysis of Mini-AES. *Cryptologia* **2003**, *27*, 361–374. [CrossRef]
29. Hitapuru, B.; Indarjani, S. Square attack on Mini-AES and Simplified AES using all variants of active nibble position. In *AIP Conference Proceedings*; AIP Publishing LLC: Melville, NY, USA, 2016; Volume 1729.
30. Mansoori, S.D.; Bizaki, H.K. On the vulnerability of Simplified AES Algorithm Against Linear Cryptanalysis. *IJCSNS Int. J. Comput. Sci. Netw. Secur.* **2007**, *7*, 257–263.
31. Campbell, S.; Grinchenko, M.; Smith, W. Linear Cryptanalysis of Simplified AES Under Change of S-Box. *Cryptologia* **2013**, *37*, 120–138. [CrossRef]
32. Simmons, S. Algebraic Cryptanalysis of Simplified AES. *Cryptologia* **2009**, *33*, 305–314. Available online: http://www.nku.edu/~{}christensen/Alg%20cryptanalysis%20SAES.pdf (accessed on 27 May 2020). [CrossRef]
33. Courtois, N.; Klimov, A.; Patarin, J.; Shamir, A. Efficient algorithms for solving over defined systems of multivariate polynomial equations. In *EUROCRYPT 2000*; Lecture Notes in Computer Science; Springer: Berlin, Heidelberg, Germany, 2000; Volume 1807, pp. 392–407. [CrossRef]
34. Ripatti, A.V. Algorithms of Splitting and van der Waerden Numbers. Available online: https://habrahabr.ru/post/224069/ (accessed on 27 May 2020). (In Russian).
35. Bard, G.; Courtois, N.; 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. 2007, Volume 24. Available online: https://eprint.iacr.org/2007/024.pdf (accessed on 27 May 2020).
36. Albrecht, M. Tools for Algebraic Cryptanalysis. In Proceedings of the ECRYPT Workshop on Tools for Cryptanalysis, Egham, UK, 22–23 June 2010; pp. 13–14.
37. Soos, M. ANF2CNF Script Released. Available online: http://www.msoos.org/2010/09/anf2cnf-script-released/ (accessed on 27 May 2020).
38. Soos, M. ANF to CNF Conversion. Available online: http://www.msoos.org/2010/07/anf-to-cnf-conversion/ (accessed on 27 May 2020).
39. Courtois, N.; Bard, G. Algebraic Cryptanalysis of the Data Encryption Standard. In Proceedings of the 11-th IMA Conference, Cirencester, UK, 18–20 December 2007; pp. 152–169.
40. Courtois, N.; Debraize, B. Specific S-Box Criteria in Algebraic Attacks on Block Ciphers with Several Known Plaintexts. In Proceedings of the WEWoRC 2007, Bochum, Germany, 4–6 July 2007; Volume 4945, pp. 100–113.
41. Maro, E.A. Modeling of algebraic analysis of PRESENT cipher by SAT solvers. In Proceedings of the VIII International Workshop on Mathematical Models and their Applications (IWMMA-2019), Krasnoyarsk, Russia, 18–21 November 2019; IOP Conference Series: Materials Science and Engineering. Volume 734.
42. The Sage Developers. *SageMath, the Sage Mathematics Software System*; Version 9.1; The Sage Developers: Newcastle upon Tyne, UK, 2020; Available online: http://www.sagemath.org (accessed on 27 May 2020).

43. Sage Reference Manual: Sat Release 7.5. Available online: http://doc.sagemath.org/pdf/en/reference/sat/sat.pdf (accessed on 27 May 2020).
44. Sage Reference Manual: Cryptography Release 7.5. Available online: http://doc.sagemath.org/pdf/en/reference/cryptography/cryptography.pdf (accessed on 27 May 2020).