Advances in Computer Science Research, volume 89
2019 International Conference on Wireless Communication, Network and Multimedia Engineering (WCNME 2019)

Algebraic Cryptanalysis of Block Ciphers

Rustem Biyashev[1], Dilmuhanbet Dyusenbayev[1], Kunbolat Algazy[1] and Nursulu Kapalova[1,*]
[1]Information Security laboratory Institute of Information and Computational Technologies CS MES RK Almaty, Kazakhstan
[*]Corresponding author

Abstract— **Algebraic methods of cryptanalysis are applicable to present-day ciphers. These methods are based on generation of an equation system, where elements of a ciphertext and a key are chosen as variables of the system. When implementing a linearization method to solve the equation system, we consider a possibility to find partial elements of a key. Generally, cryptosystems use S-blocks, which are the only element contributing to nonlinearity of a ciphering transformation and the level of its strength against cryptanalytic attacks. In this paper we present, by the example of encryption algorithm Kuznechik, an algebraic analysis applicable to some block ciphers.**

Keywords—algebraic cryptanalysis; linearization; S-block; residual variables; equation systems; truth tables; Boolean function; Zhegalkin polynomial

I. Introduction

While developing cryptosystems, engineers in many applications use S-blocks, which are the only element contributing to nonlinearity of a ciphering transformation and the level of its strength against cryptanalytic attacks.

Algebraic methods are based on formation of a nonlinear equation system representing transformations in S-blocks, and then solving the nonlinear system. In the equations composed for a linear transformation, the number of variables is not increasing when passing to next rounds, i.e. the number of variables after the first round is equal to the number of variables after the *n*th round. In contrast, in nonlinear transformations the number of variables increases after each round owing to multiplications of variables from a previous round. An algorithm complexity can be evaluated on the basis of the systems of linear equations resulting from linearization.

The size of variables chosen for a study depends on the type of the algorithm under consideration, as well as preferences of a cryptanalyst. For example, variables can have a size of a bit or byte.

When implementing a linearization method to solve a set of equations, it is possible to find elements of a key in parts. Our study was conducted with the developed software enabling generation of an equation system for linear and nonlinear transformations, namely for substitution S-blocks. The essence of the method is generation of equations representing nonlinear substitution transformations of S-blocks with a subsequent attempt to solve the generated equation systems and obtain parts of a cipher key. When performing the attack, a cryptanalyst can see in a stepwise manner changes in outputs after each round.

The attack was conducted against encryption algorithm Kuznechik. Our study has shown that all processes of the algorithm, apart from S-block, can be presented in a linear form. If a ciphertext has been known then the respective plaintext and key are considered as variables [1,2]. Each output of S-block can be converted with an input and represented by nonlinear expressions. We then obtain a set of nonlinear equations and through linearization pass to a linear equation system. Here, the number of variables is increasing exponentially with each round. So, in our studies we used an abridged linearization with residual variables. By residual variables we mean a grouped sum of terms, which are products of two or more variables.

II. Block cipher «Kuznechik»

This is a symmetric block encryption algorithm with a block size of 128 bits and a 256-bit key, generated with a Feistel network. The cipher is built on the basis of an SP-network, which is a transformation involving several identical rounds, where each round comprises of linear and nonlinear conversions followed by applying a key. An SP-network transforms the whole input block as against a half-block in a Feistel network. The length of input block of the algorithm is 128 bits, and the length of master key is 256 bits [3, 4].

Kuznechik peculiarities:

– Round keys are generated from a master key based on a Feistel network, where a round transformation of the original algorithm serves as a function;
– A linear transformation can be performed with a shift register.

When implementing encryption and decryption algorithms, the following transformations are used:

$$X[k]: V_{128} \to V_{128}, X[k](a) = k \oplus a, \text{ where } k, a \in V_{128}; \quad (1)$$

$$S: V_{128} \to V_{128}, S(a) = S(a_{15}\|\ldots\|a_0) = \pi(a_{15})\|\ldots\|\pi(a_0), \quad (2)$$

where $a = a_{15}\|\ldots\|a_0 \in V_{128}$, $a_i \in V_8$, $i = 0, 1, \ldots, 15$;

$$L: V_{128} \to V_{128} - L(a) = R^{16}(a), \text{ where } a \in V_{128}; \quad (3)$$

The linear transformation is given by mapping ℓ: V816 → V8, which is defined as follows:

$$\ell(a_{15}, \ldots, a_0) = \nabla(148 \cdot \Delta(a_{15}) + 32 \cdot \Delta(a_{14}) + 133 \cdot \Delta(a_{13}) + 16 \cdot\Delta(a_{12}) + 194 \cdot \Delta(a_{11}) + 192 \cdot \Delta(a_{10}) + 1 \cdot \Delta(a_9) + 251 \cdot \Delta(a_8) + 1\ \Delta(a_7) + 192 \cdot \Delta(a_6) + 194 \cdot \Delta(a_5) + 16 \cdot \Delta(a_4) + 133 \cdot \Delta(a_3) + 32\ \Delta(a_2) + 148 \cdot \Delta(a_1) + 1 \cdot \Delta(a_0)) \quad (4)$$

III. Algebraic cryptanalysis of Kuznechic algorithm

Let $(x_1^0, x_2^0, \dots, x_n^0)$ and $(x_1^1, x_2^1, \dots, x_n^1)$ be input and output bit blocks of the linear transformation at the initial step. Output bits are expressible by:

$$x_i^1 = q_{i,1}x_1^0 \oplus q_{i,2}x_2^0 \oplus \dots \oplus q_{i,n}x_n^0, \qquad (5)$$

where coefficients $q_{i,j} \in \{0,1\}$, $i,j = \overline{1,n}$ (n – block length) reflect the contribution of bit x_j^0 to the equation. In Kuznechik algorithm, $n = 384$, $m = 128$.

Suppose $(x_1^r, x_2^r, \dots, x_n^r)$ is an output vector of round r. The number of variables in linear equations remains unchanged. The equations governing bits of the output vector of round r are given by formula:

$$\mathrm{x}_\mathrm{i}^\mathrm{r} = \mathrm{q}_{\mathrm{i},1}^\mathrm{r}\mathrm{x}_1^0 \oplus \mathrm{q}_{\mathrm{i},2}^\mathrm{r}\mathrm{x}_2^0 \oplus \dots \oplus \mathrm{q}_{\mathrm{i},\mathrm{n}}^\mathrm{r}\mathrm{x}_\mathrm{n}^0 \qquad (6)$$

The regularity above cannot be attributed merely to S-block. If we represent input bits as $(x_1, x_2, \dots, x_8)$ and output bits of S-block as $(y_1, y_2, \dots, y_8)$, then the output bits can be transformed by input bits by the following expression:

$$\mathrm{y_i} = \mathrm{q_i} \oplus \sum_{\mathrm{j}=1}^{8} (\mathrm{q_{ij}x_j}) \oplus$$

$$\oplus \textstyle\sum_{1\le j_1,j_2\le 8} (q_{ij_1j_2}x_{j_1}x_{j_2}) \oplus \dots \oplus q_{i1\dots 8}x_1 \dots x_8 \qquad (7)$$

If we linearize the S-block equation represented by output bits, then the number of variables will be less than 256. In the event the number of variables is equal 256, S-block cannot provide a random permutation. When the equation expresses a complete combination, then:

$$\sum_{\mathrm{j}=1}^{8} \mathrm{x_j} \oplus \sum_{1\le \mathrm{j_1,j_2}\le 8} \mathrm{x_{j_1}x_{j_2}} \bigoplus \dots \oplus \mathrm{x_1} \dots \mathrm{x_8} =$$

$$= (\mathrm{x_1} \oplus 1) \dots (\mathrm{x_8} \oplus 1) \oplus 1 = \mathrm{x_1} \vee \dots \vee \mathrm{x_8}$$

Providing that the block length is 128, we will get not more than 4096 output variables for 16 input bytes. Here, inputs of S-blocks involve 8 variables as against 16 ones. If the number of variables is k, then the number of combinations will not exceed $K = \sum_{i=0}^{k} C_i^k$. In the general case, the S-block output variables for k input variables can be expressed as y_i:

$$y_i = a_{i,0} \oplus \sum_{j=1}^{k} (a_{i,j}x_j) \oplus \sum_{1\le j_1,j_2\le k} (a_{i,j_1,j_2}x_{j_1}x_{j_2}) \oplus$$

$$\oplus \dots \oplus a_{i,1,\dots,k}x_1 \times \dots \times x_k \qquad (8)$$

where $i = \overline{1,8}$.

A. Analysis of Linear Operations

When using formula (6) for transformation (3), a linear transformation can be represented as a transition matrix of size [128×128]. Matrix $M_{128\times 128}^r$ expresses 128 output bits of transformation (3) and describes 128 expressions, where each expression linearly depends on 128 input bits of (3) and r is the number of transformations performed. To get the matrix, we need to represent each value of the algorithm operations as a mathematical expression in the form of polynomials with elements L(a)=R16(a). The linear transformation involves two operations, i.e. a byte shift and a conversion. In turn, the conversion is composed of several multiplication operators.

1) Multiplication operator

In what follows, each multiplication operator from (4) is considered separately for the cases where output bits are represented by input bits. Here, a multiplication operators are performed over finite augmented field

$$GF(2)[z]\backslash(z^8 + z^7 + z^6 + z + 1).$$

The first term of linear transformation (4) (148 · X) can be expressed as follows:

$$(z^7 + z^4 + z^2) \cdot (x_7 \cdot z^7 + x_6 \cdot z^6 + x_5 \cdot z^5 + x_4 \cdot z^4 + x_3 \cdot z^3 +$$
$$+ x_2 \cdot z^2 + x_1 \cdot z^1 + x_0 \cdot z^0 = x_7 \cdot z^{14} + x_6 \cdot z^{13} + x_5 \cdot z^{12} +$$
$$+ (x_7 \oplus x_4) \cdot z^{11} + (x_6 \oplus x_3) \cdot z^{10} + (x_7 \oplus x_5 \oplus x_2) \cdot z^9 +$$
$$+(x_6 \oplus x_4 \oplus x_1) \cdot z^8 + ((x_5 \oplus x_3 \oplus x_0) \cdot z^7 + (x_4 \oplus x_2) \cdot z^6 +$$
$$+ (x_3 \oplus x_1) \cdot z^5 + (x_2 \oplus x_0) \cdot z^4 + x_1 \cdot z^3 + x_0 \cdot z^2;$$

Now we perform a modulo operation, which result in the following:

$$(148 \cdot X) \mid \mathrm{mod}\ (z^8 + z^7 + z^6 + z + 1) = (x_7 \cdot z^{14} + x_6 \cdot z^{13} +$$
$$+ x_5 \cdot z^{12} + (x_7 \oplus x_4) \cdot z^{11} + (x_6 \oplus x_3) \cdot z^{10} +$$
$$+ (x_7 \oplus x_5 \oplus x_2) \cdot z^9 + (x_6 \oplus x_4 \oplus x_1) \cdot z^8 +$$
$$+ ((x_5 \oplus x_3 \oplus x_0) \cdot z^7 + (x_4 \oplus x_2) \cdot z^6 + (x_3 \oplus x_1) \cdot z^5 +$$
$$+ (x_2 \oplus x_0) \cdot z^4 + x_1 \cdot z^3 + x_0 \cdot z^2) \mid \mathrm{mod}\ (z^8 + z^7 + z^6 + z + 1) =$$
$$= (x_7 \oplus x_6 \oplus x_5 \oplus x_1 \oplus x_0) \cdot z^7 + (x_7 \oplus x_4 \oplus x_2) \cdot z^6 +$$
$$+ (x_7 \oplus x_5 \oplus x_3 \oplus x_1) \cdot z^5 + (x_6 \oplus x_4 \oplus x_2 \oplus x_0) \cdot z^4 +$$
$$+ (x_5 \oplus x_3 \oplus x_1) \cdot z^3 + (x_7 \oplus x_4 \oplus x_2 \oplus x_0) \cdot z^2 +$$
$$+ (x_6 \oplus x_3 \oplus x_1) \cdot z^1 + (x_7 \oplus x_6 \oplus x_2 \oplus x_0);$$

We repeat the steps for the other terms.

2) Byte-shift operations

The algorithm in question works with bytes. Since we have a bit representation, we consider ai in the bitwise manner as follows:

$$a_{16} = (x_{127}, x_{126}, \dots, x_{120}),\ a_{15} =$$
$$=(x_{119}, x_{118}, \dots, x_{112}), \dots,\ a_1 =$$
$$= (x_{15}, x_{14}, \dots, x_8),\ a_0 = (x_7, x_6, \dots, x_0).$$

After each cycle, all bytes rotate right shift of one position, i.e. ai = ai+1, except for the 16th (a16) byte, which is determined from formula a16 = ℓ(a15, …, a0). If we represent bytes as bits, then after r-th cycle the output bits will follow the formula:

$$\mathrm{a}^{\mathrm{r}}_{16} = (\mathrm{x}^{\mathrm{r}}_{127}, \dots, \mathrm{x}^{\mathrm{r}}_{120}), \tag{9}$$

where

$$x^{r+1}_{127} = x^r_{127}\oplus x^r_{126}\oplus x^r_{125}\oplus x^r_{121}\oplus x^r_{120}\oplus x^r_{118}\oplus x^r_{117}\oplus$$
$$\oplus x^r_{115}\oplus x^r_{114}\oplus x^r_{111}\oplus x^r_{109}\oplus x^r_{108}\oplus x^r_{107}\oplus x^r_{105}\oplus$$
$$\oplus x^r_{104}\oplus x^r_{103}\oplus x^r_{102}\oplus x^r_{100}\oplus x^r_{99}\oplus x^r_{93}\oplus x^r_{91}\oplus x^r_{90}\oplus$$
$$\oplus x^r_{88}\oplus x^r_{87}\oplus x^r_{86}\oplus x^r_{85}\oplus x^r_{83}\oplus x^r_{82}\oplus x^r_{80}\oplus x^r_{79}\oplus x^r_{71}\oplus$$
$$\oplus x^r_{69}\oplus x^r_{67}\oplus x^r_{64}\oplus x^r_{63}\oplus x^r_{55}\oplus x^r_{54}\oplus x^r_{53}\oplus x^r_{51}\oplus x^r_{50}\oplus$$
$$\oplus x^r_{48}\oplus x^r_{45}\oplus x^r_{43}\oplus x^r_{42}\oplus x^r_{40}\oplus x^r_{39}\oplus x^r_{38}\oplus x^r_{36}\oplus x^r_{35}\oplus$$
$$\oplus x^r_{31}\oplus x^r_{29}\oplus x^r_{28}\oplus x^r_{27}\oplus x^r_{25}\oplus x^r_{24}\oplus x^r_{22}\oplus x^r_{21}\oplus x^r_{19}\oplus$$
$$\oplus x^r_{18}\oplus x^r_{15}\oplus x^r_{14}\oplus x^r_{13}\oplus x^r_{9}\oplus x^r_{8}\oplus x^r_{7} =$$
$$= q^{r+1}_{127,1}x^0_1\oplus q^{r+1}_{127,2}x^0_2\oplus \dots \oplus q^{r+1}_{127,n}x^0_n$$

By formula (4) we can also represent the following bits:

$$x^{r+1}_{126}, x^{r+1}_{125}, x^{r+1}_{124}, x^{r+1}_{123}, x^{r+1}_{122}, x^{r+1}_{121}, x^{r+1}_{120}.$$

A byte shift represented in bits is as follows:

$$x^{r+1}_i = x^r_{i-8} = q^r_{i,1}x^0_1\oplus q^r_{i,2}x^0_2\oplus \dots \oplus q^r_{i,n}x^0_n =$$
$$= q^{r+1}_{i,1}x^0_1\oplus q^{r+1}_{i,2}x^0_2\oplus \dots \oplus q^{r+1}_{i,n}x^0_n \tag{10}$$

Formula (10) is true for all i, r, i.e. an output bit of every cycle can be expressed by elements of previous cycles. For any i, r equation $q^r_i = q^{r+1}_1$ holds true.

Formula (10) involves n variables. The equation system under consideration uses 256 variables for key generation, and 384 variables for encryption.

B. Analysis of S-blocks

We look at S-block in its bit representation:

$$(x_1, x_2, \dots, x_8) \overset{S}{\rightarrow} (y_1, y_2, \dots, y_8), \tag{11}$$

where $x_i, y_i \in \{0,1\}, i = \overline{1,8}$

The bit representations are set as eight truth tables. For each truth table we construct a Boolean function of 8 variables involving in a Zhegalkin polynomial [5,6], i.e. you can describe each output bit of the S-box. In the equations above our concern was only with monomials involving one variable. The sum of the rest terms we denote as residual variables.

When linearizing the equation system we obtain as follows

$$y_1(x_1, x_2, \dots, x_8) = \delta_1,$$
$$y_2(x_1, x_2, \dots, x_8) = \delta_2, y_3(x_1, x_2, \dots, x_8) = x_8\oplus x_3\oplus x_2\oplus\delta_3$$
$$y_4(x_1, x_2, \dots, x_8) = x_7\oplus x_5\oplus x_3\oplus x_1\oplus\delta_4,$$
$$y_5(x_1, x_2, \dots, x_8) = \delta_5, \tag{12}$$
$$y_6(x_1, x_2, \dots, x_8) = x_7\oplus x_6\oplus x_5\oplus x_4\oplus\delta_6,$$
$$\mathrm{y}_7(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_8) = \mathrm{x}_8\oplus\mathrm{x}_7\oplus\mathrm{x}_4\oplus\mathrm{x}_3\oplus\mathrm{x}_1\oplus\delta_7,$$
$$\mathrm{y}_8(\mathrm{x}_1, \mathrm{x}_2, \dots, \mathrm{x}_8) = \mathrm{x}_8\oplus\mathrm{x}_7\oplus\mathrm{x}_6\oplus\mathrm{x}_5\oplus\mathrm{x}_4\oplus\mathrm{x}_3\oplus\mathrm{x}_2\oplus\delta_8$$

A similar abridgement can be performed using residual variables, which are products of three different variables. As an example:

$$y_1(x_1, x_2, \dots, x_8) = x_8x_7\oplus x_7x_6\oplus x_6x_5\oplus x_5x_4\oplus x_8x_3\oplus$$
$$\oplus x_7x_2\oplus x_6x_2\oplus x_3x_2\oplus x_7x_1\oplus x_6x_1\oplus x_5x_1\oplus x_3x_1\oplus x_2x_1\oplus\delta_1.$$
$$y_2(x_1, x_2, \dots, x_8) = x_8x_7\oplus x_8x_6\oplus x_7x_6\oplus x_7x_5\oplus x_6x_5\oplus x_6x_4\oplus$$
$$\oplus x_5x_4\oplus x_5x_3\oplus x_8x_2\oplus x_7x_2\oplus x_3x_2\oplus x_7x_1\oplus x_6x_1\oplus x_2x_1\oplus\delta_2.$$
$$y_3(x_1, x_2, \dots, x_8) = x_8\oplus x_8x_6\oplus x_7x_6\oplus x_8x_5\oplus x_6x_5\oplus x_6x_4\oplus$$
$$\oplus x_5x_4\oplus x_3\oplus x_7x_3\oplus x_7x_2\oplus x_6x_3\oplus x_4x_3\oplus x_2\oplus x_7x_2\oplus x_6x_2\oplus$$
$$\oplus x_5x_2\oplus x_4x_2\oplus x_8x_1\oplus x_6x_1\oplus x_4x_1\oplus\delta_3.$$
$$y_4(x_1, x_2, \dots, x_8) = x_7\oplus x_8x_6\oplus x_7x_6\oplus x_5\oplus x_7x_4\oplus x_6x_4\oplus$$
$$\oplus x_5x_4x_3\oplus x_8x_3\oplus x_7x_3\oplus x_8x_2\oplus x_6x_2\oplus x_5x_2\oplus x_3x_2\oplus x_1\oplus$$
$$\oplus x_8x_1\oplus x_6x_1\oplus x_3x_1\oplus x_2x_1\oplus\delta_4.$$
$$y_5(x_1, x_2, \dots, x_8) = x_8x_7\oplus x_8x_6\oplus x_7x_6\oplus x_8x_5\oplus x_7x_5\oplus x_6x_5\oplus$$
$$\oplus x_3\oplus x_8x_4\oplus x_7x_4\oplus x_6x_4x_5x_4\oplus x_6x_3\oplus x_5x_3\oplus x_4x_3\oplus x_8x_2\oplus$$
$$\oplus x_6x_2\oplus x_5x_2\oplus x_3x_2\oplus x_8x_1\oplus x_7x_1\oplus x_5x_1\oplus x_4x_1\oplus x_2x_1\oplus\delta_{5.}$$
$$y_6(x_1, x_2, \dots, x_8) = x_7\oplus x_8x_7\oplus x_6\oplus x_8x_6\oplus x_5\oplus x_4\oplus x_8x_4\oplus$$
$$\oplus x_7x_4\oplus x_6x_4\oplus x_7x_2\oplus x_6x_2\oplus x_3x_2\oplus x_7x_1\oplus x_5x_1\oplus$$
$$\oplus x_4x_1\oplus x_2x_1\oplus\delta_6. \tag{13}$$
$$y_7(x_1, x_2, \dots, x_8) = x_8\oplus x_7\oplus x_8x_7\oplus x_7x_6\oplus x_8x_5\oplus x_7x_5\oplus x_4\oplus$$
$$\oplus x_8x_4\oplus x_7x_4\oplus x_6x_4\oplus x_3\oplus x_8x_3\oplus x_7x_3\oplus x_4x_3\oplus$$
$$\oplus x_8x_2\oplus x_7x_2\oplus x_3x_2\oplus x_1\oplus x_6x_1\oplus x_3x_1\oplus x_2x_1\oplus\delta_7.$$
$$y_8(x_1, x_2, \dots, x_8) = x_8\oplus x_7\oplus x_8x_7\oplus x_6\oplus x_8x_6\oplus x_7x_6\oplus x_5\oplus$$
$$\oplus x_7x_5\oplus x_6x_5\oplus x_4\oplus x_8x_4\oplus x_6x_4\oplus x_5x_4\oplus x_3\oplus x_5x_3\oplus$$
$$\oplus x_4x_3\oplus x_2\oplus x_6x_2\oplus x_3x_2\oplus x_7x_1\oplus x_6x_1\oplus x_4x_1\oplus x_3x_1\oplus\delta_8.$$

C. Linearization Abridgement

Let $\sum_{1\le j_1,j_2\le n}\left(q_{ij_1j_2}x_{j_1}x_{j_2}\right)\oplus \dots \oplus q_{i1\dots n}x_1 \dots x_n$ be a part of expression (8). We denote the remaining part as a new variable $\delta_i \in \{0,1\}$, then formula (8) becomes as follows:

$$x^r_i = q^r_{i,0}\oplus q^r_{i,1}x^0_1\oplus q^r_{i,2}x^0_2\oplus \dots \oplus q^r_{i,n}x^0_n\oplus\delta_i \tag{14}$$

where $x^r_i, i = \overline{1,128}$ are bits of a ciphertext.

We divide linear equation system (14) into two parts:

$F(K, X) = q^r_{i,0}\oplus q^r_{i,1}x^0_1\oplus q^r_{i,2}x^0_2\oplus \dots \oplus q^r_{i,n}x^0_n$, and residual variables

$$\mathrm{Q} = (\delta_1, \delta_2, \dots, \delta_{128}), \mathrm{F(K, X)}\oplus\mathrm{Q} = \mathrm{Y} \tag{15}$$

Where $K = (x_1, x_2, \dots, x_{256})$ are key bits, $X = (x_{257}, x_{258}, \dots, x_{384})$ are plaintext bits, and $Y = (y_1, y_2, \dots, y_{128})$ are ciphertext bits.

Let there be given two blocks of the cipher $F_1(K, X_1)\oplus Q_1 = Y_1$ and $F_2(K, X_2)\oplus Q_2 = Y_2$.

If we add the plaintext variables to residuals, we will obtain:

$$F^*(K) \oplus Q_1^* = Y_1^*, F^*(K) \oplus Q_2^* = Y_2^* \quad (16)$$

When adding together equation systems (16), we get as follows:

$$Q_1^* \oplus Q_2^* = Y_1^* \oplus Y_2^* \quad (17)$$

The resulting equation system has 256 variables. In a general way, it needs an exhaustive search of 2^{256} operations to retrieve the key. By using equation system (17) the complexity of exhaustive search will be 2^{128}.

IV. Summary

Based on the research findings it can be proposed as follows:

- If equation system (15) provides for a possibility to calculate variables of the system in parts, then the complexity of the exhaustive search will be less than 2^{128};
- If equation system (15) does not provide for a possibility to calculate variables of the system in parts, then the complexity of the exhaustive search will be more than 2^{128};

With the developed software implementing the approach described above, it is possible to trace involvement of each key bit in the process of transformation. Our study of Kuznechik algorithm has shown that the transformation results lead to the second proposition, i.e. the complexity of the exhaustive search will be more than 2^{128}. The algebraic method discussed in the paper can also be implemented to other block encryption algorithms.

Acknowledgment

This research was financially supported by the Ministry of Education and Science of the Republic of Kazakhstan.

References

[1] Voronin R. I., Algebraic cryptoanalysis of one-round S-AES, PDM, 2011, 4, 29–31.

[2] Kapalova N., Dyusenbayev D., Security analysis of an encryption scheme based on nonpositional polynomial notations // Journal Open Engineering. - 2016. – 6: p. 250-258.

[3] National Standard of the Russian Federation, GOST R 34.12-2015, Information technology, Cryptographic data security, Block ciphers. Moscow, Москва, Standartinform, 2015, p. 21.

[4] E.A. Ischukova, R.A. Koshutskiy, L.K. Babenko. Development and implementation of high speed data encryption algorithm Kuznechik, Electronic scientific journal of the Kursk State University. 2015. No. 4 (08).

[5] S.D. Shaporev. Mathematical logic. Course of lectures and tutorial. – SPb.: BHV-Peterburg, 2007. – p. 416: il.

[6] A.G. Rostovtsev, E.B. Makhovenko. Theoretical cryptography. NPO Professional, Sankt-Petersburg, 2004.