INFORMATION AND CONTROLLING SYSTEM

Lightweight encryption algorithms are considered a relatively new direction in the development of private key cryptography. This need arose as a result of the emergence of a large number of devices with little computing power and memory. Therefore, it became necessary to develop algorithms that can provide a sufficient level of security, with minimal use of resources. The paper presents a new lightweight LBC encryption algorithm. LBC is a 64–bit symmetric block algorithm. It supports 80 bit secret key. The number of rounds is 20. The algorithm has a Feistel network structure. The developed lightweight algorithm has a simple implementation scheme, and the transformations used in this algorithm have good cryptographic properties. This was verified by studying the cryptographic properties of the algorithm using the "avalanche effect" and statistical tests. The avalanche property was checked for each round when each bit of the source text was changed. Based on the work carried out, it was found that the proposed encryption algorithm is effective to ensure a good avalanche effect and the binary sequence obtained after encryption is close to random. Its security against linear and differential cryptanalysis is also evaluated. The results of the research revealed good cryptographic properties of this algorithm. The algorithm will be used for devices with small hardware resources, in information and communication systems where confidential information circulates, and it is also extremely necessary to exchange information in a protected form in an operationally acceptable time

Keywords: encryption algorithm, lightweight algorithm, cryptographic transformations, avalanche effect, cryptographic stability

UDC 004.056.5
DOI: 10.15587/1729-4061.2023.280055

DEVELOPMENT OF A NEW LIGHTWEIGHT ENCRYPTION ALGORITHM

Nursulu Kapalova
Leading Researcher, Candidate of Technical Sciences, Associate Professor*
Kunbolat Algazy
Senior Researcher, PhD*
Armanbek Haumen
Corresponding author
Research Associate*
E-mail: haumen.armanbek@gmail.com
*Information Security Laboratory
Institute of Information and Computational Technologies
Shevchenko str., 28, Almaty, Republic of Kazakhstan, 050010

Received date 17.03.2023
Accepted date 23.05.2023
Published date 30.06.2023

How to Cite: *Kapalova, N., Algazy, K., Haumen, A. (2023). Development of a new lightweight encryption algorithm. Eastern-European Journal of Enterprise Technologies, 3 (9 (123)), 6–19. doi: https://doi.org/10.15587/1729-4061.2023.280055*

1. Introduction

There is a trend of mass transition from the Internet of personal computers to the Internet of things. Electronic devices aimed at improving life and interacting with the Internet in one way or another are increasingly entering the life of society. Mobile phones, electronic keys, household appliances are all equipped with a processor and operate as part of a large Internet network. With the growing demand, lightweight ciphers are gaining more and more attention. In 2018, NIST issued a call for lightweight ciphers, accepting applications for simplified authenticated encryption and hashing for environments with software or hardware limitations. In this regard, the development and analysis of lightweight cryptographic algorithms have gained popularity among researchers. The studies have focused on the development of a lightweight block cipher, security analysis, and performance evaluation, etc. Such lightweight algorithms are designed to reduce power and memory consumption to meet the requirements of resource-constrained applications such as RFID and Internet of Things (IoT) devices.

There are many encryption algorithms that are focused on solving a variety of tasks. In the case of limited resources, each of them has different advantages in software and hardware efficiency. The trade-off between security, cost, and efficiency is a key issue that needs to be addressed when developing a lightweight block cipher. In particular, the key length of the lightweight block cipher, the total number of rounds, and the structure algorithm affect security, cost, and efficiency, but the security of the cryptographic algorithm itself plays the most significant role. However, this process takes up significant resources in the field of service data and leads to a high cost of hardware implementation and a significant decrease in efficiency and performance.

The block encryption algorithm is structured in two groups: the SP network and the Feistel network. The advantage of using an SP network is the diffusion level. Its circular function can modify all block messages in an iterative loop; hence, its safety is relatively high. To solve these problems, ciphers in traditional Feistel structures usually require many rounds, thus this increases energy consumption.

The intensive development of information technology capabilities, including computing power, contributes to the emergence of new modifications of existing attacks, which requires constant development and updating of protection systems.

Thus, the area of research under consideration is relevant. A comprehensive study of the block encryption components used in the development of lightweight algorithms, as well as their compliance with modern technologies, requires continuous and breakthrough scientific investigation.

2. Literature review and problem statement

Article [1] discusses a new lightweight encryption algorithm SIT (Secure Internet of Things), proposed by a group of Pakistani cryptographers. The structure of the algorithm is a combination of Feistel and SP (substitution-permutation) networks, which uses cryptographic transformations such as modulo 2 (xor), negation of modulo 2 (nxor), substitution (S-box), and permutation. According to the results of the study, the algorithm found that it provides security even after five rounds of encryption. According to the authors, differential and linear cryptanalysis will not bring success in a crypto attack. However, in each round, when using a 4-bit S-block, it would be better to demonstrate the complexity of the attack, taking into account the difference characteristic with sufficient probability.

Paper [2] proposes a new permutation-based streaming algorithm for encrypting video data. The developed stream algorithm becomes resistant to attacks with known plaintext, which is a common problem for modern video data encryption algorithms. As the authors emphasize, encryption before compression is used to prevent the loss of video data in the event of packet loss during transmission. Experiments were carried out to evaluate performance, taking into account the probability of data loss in case of packet loss and encryption speed. Still, the authors of the algorithm propose to conduct further research in order to minimize the increased encryption time to the level of encryption algorithms after compression, which is an urgent task, as well as any in-line and block algorithm aimed at IoT.

LT10 is a symmetric block encryption algorithm that uses a 128-bit key and plaintext. The key generation algorithm consists of rounds of encryption that use the Kasumi encryption algorithm. The authors argue that, as you know, increasing the number of rounds provides high security, therefore, directly affects the performance of encryption. But the cited article describes only the performance of the proposed algorithm and the study, and evaluation of the analysis of cryptographic strength and ensuring the protection of information was not considered [3].

Speck and Simon ciphers are two families of lightweight block ciphers that were introduced by the US National Security Agency in 2013, and in 2014 were included in the international standard ISO/IEC 29192. Both families support different block sizes for encryption and key: the input block sizes are 32, 48, 64, 96, 128 bits, and the secret key sizes are 64, 72, 96, 128, 144, 256 bits. The number of rounds for encryption depends on the selected block sizes and key. Article [4] gives a complete description of the algorithms themselves, a schedule of keys, and a comparative analysis of their performance, as well as their software and hardware implementations with different encryption parameters. The performance assessment is a practical benchmark in the development of new encryption algorithms for lightweight cryptography.

Paper [5] reports attacks on the Present encryption algorithm, including error analysis with statistical cryptanalysis methods. The analysis uses statistical cryptanalysis techniques in practice and exploits the weakness of bit permutation adopted by many lightweight block ciphers in an error attack. The results of the work show that it takes about a fifth of iterative rounds to protect these lightweight ciphers with a permutation of bits. However, this often requires erroneous encryption of the penultimate or last round, as the errors of the middle round are suitable for conducting an attack.

In recent years, the technology of cyber-physical systems (CPS) and the Internet of Things (IoT) have grown exponentially. This has led to the development of low-resource encryption algorithms. Paper [6] presents a lightweight BRISI cipher using modulo addition, shift and XOR, as well as the Fiestel structure. The encryption algorithm uses a 32-bit block of plaintext and a 64-bit key. These resource-constrained devices are designed to meet the requirements of heterogeneous IoT and CPS applications. In such devices, privacy and security have become the most difficult issues. Resource-constrained devices were not designed to provide security features as the 32-bit block used is not suitable for today's technological capabilities.

A lightweight encryption scheme based on Attribute-BasedEncryption is reported in [7], where it used elliptic curves to solve the security problem. A central authority was used to generate keys, attributes, and users, but this can be a problem when it comes to a multi-author application. However, their approach seems fair in terms of the cost of computing and communication for IoT devices.

In [1] a lightweight cipher called Secure IoT (SIT) is presented, this is a 64-bit block cipher, the key length is 64-bit. The architecture of the algorithm is a mixture of Feistel and a single substitution-permutation network. The results of the work show that the algorithm provides significant security in just five rounds of encryption and that the encrypted image is completely covered, i.e., no traces remain. Although known algorithms cannot achieve such a result without the use of encryption modes, which leads to doubts about what is claimed. Therefore, a detailed understanding of the algorithm's encryption process is required.

3. The aim and objectives of the study

The aim of this study is to develop a lightweight data encryption algorithm.

To accomplish the aim, the following tasks have been set:

– to develop a block scheme of the algorithm, which meets the basic requirements for cryptographic transformations, and provides high performance in software and hardware implementation;

– to investigate the reliability of the proposed encryption algorithm by methods of cryptographic analysis and avalanche effect.

4. The study materials and methods

4. 1. The object and hypothesis of the study

The object of our study is a lightweight block data encryption algorithm. As part of this study, the main hypothesis was put forward that by using simple cryptographic transformations, it is possible to create a lightweight algorithm that meets the basic requirements for cryptographic ciphers and is not inferior in cryptographic strength to known lightweight algorithms. When developing a lightweight algorithm, it was assumed that the structure of the created algorithm could improve the cryptographic properties of previously created lightweight algorithms.

4. 2. Nonlinear transformation in encryption algorithms

Every modern encryption algorithm uses a nonlinear transformation in the form of a lookup table. It is proved that it is a strong cryptographic primitive against linear and differential cryptanalysis [8]. In lightweight block algorithms, table substitutions (*S*-blocks) are also used for nonlinear transformations. But according to the requirement for lightweight algorithms, S-blocks should not occupy a large amount of memory (ROM and RAM). To store a table that replaces 8-bit pieces of data, you need 256 bytes, and to store a 4-bit S-block, one requires only 16 bytes [9].

To increase the resistance of a block cipher to linear and differential cryptanalysis, two main approaches are used:

1) an increase in the number of active S-blocks;

2) the use of S-blocks with strong cryptographic properties [10].

The procedure of *S*-block transformation in the cipher provides scattering and mixing of plaintext bytes, which in turn significantly increases the cryptographic strength of the cipher test to the implementation of various cryptanalytic attacks [11–13]. The most important cryptographic properties that an *S*-block should have are balance, high nonlinearity, high algebraic degree, low differential homogeneity, and low autocorrelation.

The following are the basic definitions that are used in evaluating the cryptographic properties of nonlinear replacement nodes.

A Boolean function of n variables is an arbitrary mapping of the form $f: F_2^n \to F_2$, by a binary vector of length n set $x=(x_1, x_2, ..., x_n)$.

A function f over F_2^n is balanced if its output values are equally probable:

$$\left|\{x \mid f(x)=0\}\right| = \left|\{x \mid f(x)=1\}\right| = 2^n - 1. \quad (1)$$

The Hamming weight $w(f)$ of the Boolean function f is the Hamming weight of its vector of values, i. e., the number of unities in the vector:

$$hw(f) = \sum_{x=0}^{2^n-1} f(x). \quad (2)$$

The Hamming distance $hd(f,g)$ between two Boolean functions f, g is defined as the number of vectors of space F_2^n on which these functions take different values, i. e.:

$$hd(f,g) = set\left\{x \in F_2^n / f(x) \neq g(x)\right\}. \quad (3)$$

The nonlinearity of the Boolean function f from n variables is the minimum Hamming distance between f and all affine functions: $N_f=\min\{d(f,\varphi)\}$ where φ is the set of affine functions.

The Walsh transform $F(\omega)$ of the function f over a field F_2^n is defined as:

$$F(\omega) = \sum_{x=0}^{2^n-1} (-1)^{f(x)\oplus\langle\omega,x\rangle}, \quad (4)$$

where $\langle\omega,x\rangle$ is the scalar product $\omega_1 x_1 \oplus \omega_2 x_2 \oplus ... \oplus \omega_n x_n$.

The function f is called the correlation-immune of order m, $0<m\leq n$ if for any vector $u \in F_2^n$, such that $1\leq w(u)\leq n$, the equality $w_f(u)=0$ is satisfied.

The autocorrelation function and *SSI* (sum-of-square indicators) are defined as follows:

$$\Delta_f(u) = \sum_{x=0}^{2^n-1} (-1)^{f(x)\oplus f(x\oplus u)}, \quad SSI = \sum_{u=0}^{2^n-1} \Delta_f^2(u). \quad (5)$$

The function f satisfies the Strict Avalanche Criterion (SAC) if $\Delta_f(u)=0$ for all u, such that $w(u)=1$.

The function f has correlation immunity of order k ($CI(k)$) if $1\leq hw(u)\leq m$ and $W(u)=0$ are fulfilled for all u.

A balanced correlation-immune of order t function is called a t-stable function. The analysis of systems of equations is an important and complex scientific task. Therefore, the function f cannot be randomly selected.

4. 3. "Avalanche effect" in encryption algorithms

The avalanche effect is a manifestation of the dependence of all output bits of ciphertext on each input bit of plaintext. It manifests itself in the dependence of all output bits on each input bit.

When assessing the avalanche effect, the following statistical safety indicators are considered:

– the average number of output bits that change when one input bit changes (avalanche effect);

– degree of completeness (d_c);

– degree of avalanche effect (d_a);

– the degree of strict avalanche criterion (D_{sa}).

They will be considered for a different number of loops and randomly taken encryption keys.

A cryptographic algorithm satisfies the avalanche criterion if an average of half of the output bits changes when one bit of the input sequence changes.

To characterize the degree of avalanche effect in the transformation, an avalanche parameter is determined and used – the numerical value of the deviation of the probability of a change in bits in the output sequence when the bits in the input sequence change from the required probability value equal to 0.5.

For the avalanche criterion, the value of the avalanche parameter is determined by the following formula:

$$\varepsilon_i = \left|2k_i - 1\right|, \quad (6)$$

where i is the number of the bit to be changed at the input, k_i is the probability of changing half of the bits in the output sequence when the i-th bit in the input sequence changes.

The values of the specified avalanche parameter are in the range from 0 to 1 inclusive. At the same time, the lower the value of the avalanche parameter, the stronger the avalanche effect in the transformation [14].

Verification of the statistical security indicators of the cipher begins with the construction of the dependence matrix A and the distance matrix B of the cipher, described by the function $f:(GF(2))^n \to (GF(2))^m$ for the set of X inputs, where X is the set itself $(GF(2))^n$ or a randomly selected subset of the set $(GF(2))^n$ [15].

A dependence matrix and a distance matrix are also defined as:

$$\begin{aligned} a_{ij} &= \#\left\{x \in X \mid \left(f\left(x^{(i)}\right)\right)_j \neq \left(f(x)\right)_j\right\}, \\ a_{ij} &= \#\left\{x \in (GF(2))^n \mid \left(f\left(x^{(i)}\right)\right)_j \neq \left(f(x)\right)_j\right\}, \\ b_{ij} &= \#\left\{x \in X \mid w\left(f\left(x^{(i)}\right) - f(x) = j\right)\right\}, \end{aligned} \quad (7)$$

here $i=1,...,n$ and $j=1,...,m$, where X is a "suitable" randomly selected subset $(GF(2))^n$.

If such matrices are constructed, then the degree of completeness is defined as:

$$d_c = 1 - \frac{\#\{(i,j) \mid a_{ij} = 0\}}{nm}. \quad (8)$$

The degree of avalanche effect is determined as follows:

$$d_a = 1 - \frac{\sum_{i=1}^{n} \left| \frac{1}{\#X} \sum_{j=1}^{m} 2jb_{ij} - m \right|}{nm}. \quad (9)$$

The degree of strict avalanche criterion is defined as:

$$d_{sa} = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \left| \frac{2a_{ij}}{\#X} - 1 \right|}{nm}. \quad (10)$$

Ciphers that have a good degree of completeness, a good avalanche effect, and satisfy a strict avalanche criterion must have values d_c, d_a, that satisfy the conditions: $d_c \approx 1$, $d_a \approx 1$, $d_{sa} \approx 1$ [15].

4. 4. Differential cryptanalysis of symmetric block encryption algorithms

A block cipher is considered secure enough for practical use after it has undergone extensive cryptanalysis. One of the main methods used for cryptanalysis of block ciphers is differential cryptanalysis. A successful differential attack is based on the discovery of a highly probable differential characteristic, which will be used as a statistical characteristic for key recovery [16].

The differential attack introduced in [17] assumes the existence of ordered pairs (α, β) of binary strings, such that m is a block of plaintext, c and c' are ciphertexts. The same texts are associated with m and $m+\alpha$. The bitwise difference $c \oplus c'$ is more likely to be β than if c and c' were randomly selected binary strings. Such an ordered pair (α, β) is called a differential. The greater the probability of a differential, the more effective the attack. A related criterion for (n, m) – function F used as an S-block in round cipher functions is that the output for its derivatives $D_a(x)=F(x)+F(x+a)$; $x, a \in F_n^2$, should be distributed as evenly as possible [18].

As you know, at the design stage of modern block ciphers, protection against differential and linear cryptanalysis was laid [19].

When conducting differential cryptanalysis, there are four stages:

– analysis of the differential properties of the transformations used in the algorithm;

– finding the most probable value of the difference;

– search for the right pairs of texts;

– analysis of the correct pairs of texts to determine the bits of the key.

4. 5. Linear cryptanalysis of symmetric block encryption algorithms

Linear cryptanalysis has been used to analyze many ciphers [20]. It focuses on a linear approximation between plaintext, ciphertext, and key.

A linear approximation shows what kind of linear relationship exists between some bits of plaintext, bits of ciphertext, and bits of an unknown key:

$$A[a_1, a_2, ..., a_n] \oplus C[c_1, c_2, ..., c_m] = K[k_1, k_2, ..., k_l], \quad (11)$$

where $a_1, a_2, ..., a_n$, $c_1, c_2, ..., c_m$ and $k_1, k_2, ..., k_i$ denote fixed bit positions, and equation (11) is satisfied with probability $p \neq 1/2$ for arbitrarily given plaintext A, corresponding ciphertext C, and key K [21, 22].

First, an approximation is searched for individual operations within the cipher, then they are combined into approximations that are valid for one round of the cipher. By appropriately concatenation of single-round approximations, the attacker eventually obtains an approximation for the entire cipher [23].

To determine the complexity of the attack, the probability of a linear characteristic is estimated. The linear approximation of a single round can be thought of as a random variable of the form $\alpha_1 X_1 \oplus \alpha_2 X_2 \oplus ... \oplus \alpha_n X_n \oplus \beta_1 Y_1 \oplus \beta_2 Y_2 \oplus ... \oplus \beta_m Y_m$, which takes either a value of zero or one (depending on the bits of the key). Then the linear characteristic *xor* of these random variables and the probability of the linear characteristic can be calculated using the sign collision lemma (lemmas 1, 2 [21]).

Consider two independent random variables X_1 and X_2. Hence, $P(X_i=0)=p_i$ and $P(X_i=1)=1-p_i$ for $i \in \{1,2\}$. Then, from the independence of X_1 and X_2 it follows that $P(X_1=0, X_2=0)=p_1 p_2$ and that $P(X_1=1, X_2=1)=(1-p_1)(1-p_2)$.

Thus, $P(X_1 \oplus X_2=0)=p_1 p_2+(1-p_1)(1-p_2)$ [24].

Lemma 1. Let $X_i (1 \leq i \leq n)$ be independent random variables that take values from Z_2 whose values are equal to zero with probability $\frac{1}{2}+\varepsilon$. Then the probability that:

$$X_1 \otimes X_2 \otimes ... \otimes X_n = 0 \text{ is equal to } \frac{1}{2}+\varepsilon, \quad (12)$$

where $\varepsilon = 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2} \right)$

Lemma 2. Let N be the number of random plaintexts given and p be the probability that equation (12) holds and let $|p-1/2|$ be small enough. Then the probability of success of the algorithm is:

$$\int_{-2\sqrt{N}|p-1/2|}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx. \quad (13)$$

4. 6. Calculation of conditional logical elements

The area of the chip is usually measured in μm^2, but this parameter is highly dependent on the technologies used and the standard cell libraries. In order to compare microcircuits manufactured using different technologies, there is a unification that allows you to measure the dimensions in conditional logic elements (Gate Equivalent – GE).

Conventionally, a logic element refers to the number of equivalent gates that can perform a specific logical operation. To count equivalent gates, you must first determine the type of valves used (e.g., AND, OR, NOT, NAND, etc.). Then, you can use a standard set of equivalent logic gates.

To count the equivalent gates of a particular logic function, it is necessary to determine the number of gates of each type required to implement the function, and then sum the equivalent gates for each type of valve. From the number of summations of equivalent implementation valves, they are classified as:

– requiring less than 1000 GE – ultra-lightweight;

– requiring no more than 2000 GE – low-cost;

– requiring no more than 3000 GE – lightweight.

The most compact hardware implementation of the PRESENT algorithm requires 1570 GE.

5. Results of the development of a new lightweight encryption algorithm and the investigation of its security

5. 1. Development of a principal scheme of an encryption algorithm

5. 1. 1. Overview of the encryption algorithm scheme

The LBC algorithm is designed to encrypt block-type data with a length of 64 bits with an 80-bit key. LBC performs 20 rounds when encrypted. Each round includes 4 types of transformation:

– transformation *S*;

– transformation *RL*;

– transformation *L*;

– transformation *K*.

The encryption process. In encryption, the original block of plaintext is divided into 4 subblocks of 16 bits. Encryption begins by adding the first 64 bits of the master key to the source text (Fig. 1). Next, round transformations are performed.

If we designate the round encryption process as *E*, we get:

$$E(A)=\left[K\left(L\begin{pmatrix}S(A_0)\|RL(S(A_0))\oplus\\ \oplus S(A_1)\|S(A_2)\|S(A_3)\end{pmatrix}\right)\right]_{rc}=B, \quad (14)$$

where $A=\{A_0, A_1, A_2, A_3\}$ is the 64-bit source text, $\{A_i\}$ is the 16-bit subblock, B is the 64-bit ciphertext, rc is the number of rounds, $\oplus$ is the addition modulo 2.

The *RL* function is executed only on the subblock $\{A_0\}$. That is, $\{A_1\}=RL(A_0)\oplus A_1$.

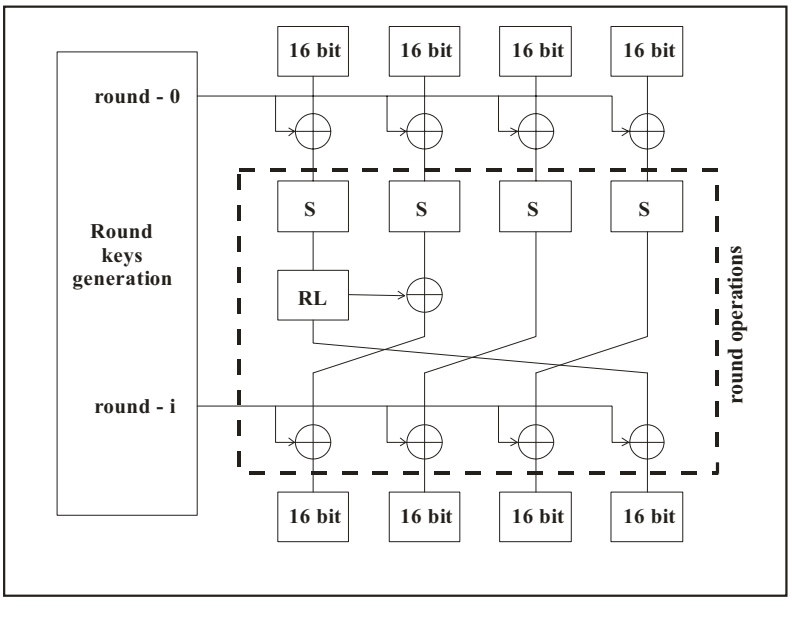

Fig. 1. The general scheme of the LBC algorithm

Transformation S. The *S* transformation was designed to replace block bits with other bits using tables (*S-block*). This nonlinear transformation of the algorithm uses nibble operations (4 bits). A nonlinear bijective substitution is applied to each nibble, specified by a one-dimensional array consisting of 16 elements (Table 1).

A 16-bit subblock is fed into the input of the *S* transform and is divided into 4 groups of 4 bits. Each nibble of the input block is an index of the value in the lookup table (Table 1). Thus, the output of the transformation *S* will be a set of nibbles located at the corresponding indexes in the given lookup table:

$$S(A)=S(a_0)\|S(a_1)\|S(a_2)\|S(a_3), S: V_4\to V_4, \quad (15)$$

where A is the input 16 bits, a_i is the nibbles of the subblock.

Table 1

Lookup table

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(X)$	9	2	A	4	0	6	7	D	5	1	8	3	E	F	B	C

The S-block is generated by obtaining the inverse element modulo x^4+x^3+1. After that, the following affine transformation was applied to each element of the table:

$$\begin{Vmatrix}b_3'\\ b_2'\\ b_1'\\ b_0'\end{Vmatrix}=\begin{Vmatrix}1&1&0&1\\ 1&1&1&0\\ 0&1&1&1\\ 1&0&1&1\end{Vmatrix}*\begin{Vmatrix}b_3\\ b_2\\ b_1\\ b_0\end{Vmatrix}+\begin{Vmatrix}1\\ 0\\ 0\\ 1\end{Vmatrix}, \quad (16)$$

where b_i are bits of a half-byte, b_i' – new bits of a half-byte.

Transformation RL. A linear transformation of the LBC algorithm is a cyclic shift of bits and subblocks to a certain position. The 64 bits of the input block are divided into 4 sub-blocks of 16 bits each. Linear transformation at the subblock level is carried out using the *RL* function, which applies only to the first subblock (Fig. 1). The result of the *RL* function is summed with the second subblock modulo 2 (*xor* operation). The *RL* function has the following form:

$$RL(a)=a\oplus(a\lll 7)\oplus(a\lll 10),\ a\in F_2^{16}, \quad (17)$$

where « $\lll$ » – operator of cyclic shift of bits of a subblock to the left.

Transformation L. This linear transformation is carried out over the whole block. In this case, the subblocks are cyclically shifted to the left by one position, i. e., the second subblock will be in the first position, the third subblock will be in place of the second, the fourth subblock will be in place of the third, and the first subblock will move to the fourth position (Fig. 1).

Transformation K. After cyclic shifts of subblocks, round keys are added to the data block modulo 2. The round key, consisting of 64 bits, is divided into 4 connections of 16 bits each and summed up with the corresponding data subblocks.

The decryption process. The decryption process is performed in reverse order. In this case, instead of the *S* and *L* transformations, the inverse transformations S^{-1} and L^{-1} are used, and the *RL* and *K* transformations remain the same as in the encryption process:

$$D(B)=\left[S^{-1}\left(L^{-1}\begin{pmatrix}K(B_0)\|RL(K(B_0))\oplus\\ \oplus K(B_1)\|K(B_2)\|K(B_3)\end{pmatrix}\right)\right]_{rc}=A, \quad (18)$$

where $\{B_i\}$ are the subblocks of the ciphertext. A is the decrypted source text. The RL function is executed only on the $\{B_0\}$subblock. That is, $B_1\}=RL(B_0)\oplus B_1$.

Transformation S^{-1}. This is the inverse of the conversion to the tabular replacement of nibble. It is carried out using Table 2.

Table 2

Reverse lookup table

X	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(X)$	4	9	1	B	3	8	5	6	A	0	2	E	F	7	C	D

At the input of the S^{-1} transformation, half bytes of the input block are supplied. The same nibbles will serve as the index of the value located in the reverse lookup table (Table 2). Thus, the output is sets of nibbles.

5. 1. 2. Generation of round keys

The LBC algorithm has a secret key that is 80 bits long. From this sequence of bits, the round keys of the algorithm are generated. The key generation process consists of several transformations. The scheme for generating round keys is shown in Fig. 2.

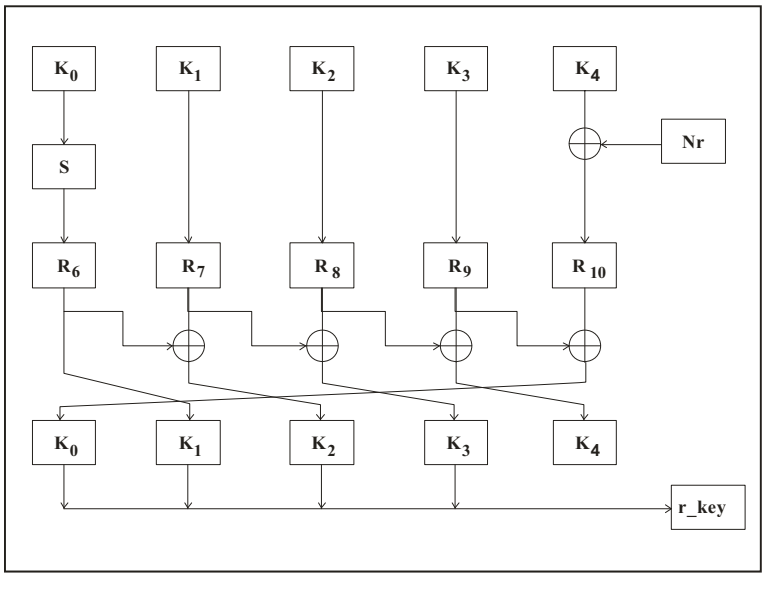

Fig. 2. Scheme for generating round keys of the LBC algorithm

The master key of 80 bits is divided into 5 subblocks of 16 bits. At the first stage, the first subblock of keys is replaced with other bits using the S-block lookup table (Table 1). Modulo 2 constant Nr is added to the fourth subblock. Here, Nr is the number of the round, which has values from 1 to 20.

In the second step, the bits of each subblock are cyclically shifted to the left by a certain position. This procedure is performed by the function R_i, here i is the number of positions by which the bits of the subblock will be shifted. As shown in Fig. 2, the bits of the first subblock are shifted cyclically to the left by position 6, the second subblock by 7, the third subblock by 8, and the remaining subblocks by 9 and 10, respectively.

At the third stage, the subblocks are summed up modulo 2 with the adjacent right subblock: the first with the second, the second with the third, the third with the fourth subblock.

At the last stage of the transformation, the subblocks are cyclically shifted to the left by one position, that is, the subblocks are swapped entirely.

After the above transformations of 5 subblocks (80 bits), the initial 4 subblocks (64 bits) are taken from the left side of the sequence and used as round encryption (decryption) keys. The 80 bits obtained from the previous transformations will be used as the basis for the next round keys. This process continues until all round encryption keys are obtained.

5. 2. Investigation of the security of the proposed lightweight encryption algorithm

5. 2. 1. Investigation of the properties of *S*-block

The LBC algorithm uses a 4-bit S-block. The 4-bit S-block contains 16 unique and distinct elements, which range from 0 to F in hexadecimal format (Table 1). In the study of the characteristics of the S-block, such properties as balance, weight, and Hamming distance, nonlinearity, correlation value, etc. were studied. The results are given in Table 3.

Table 3

Characteristics of the *S*-block used

Properties		Value
Balance		yes
Hamming weight		8
Hamming distance		8
Non-linearity	maximum	12
	minimum	4
Correlation value	maximum	8
	minimum	–8
$\|AC\|_{max}$		8
SSI		640
SAC		no
CI		no
t-stability		no
Cyclic structure		1: – (0; 4; 16)

These indicators of the effectiveness of the S-block affect the level of resistance of the developed cipher to various crypto attacks, therefore, conclusions about its strength are based on their values.

5. 2. 2. Estimation of the "avalanche effect" of the lightweight LBC encryption algorithm

The LBC encryption algorithm splits text into blocks that are 64 bits long. For the practical assessment of the avalanche effect, the avalanche criterion was used from text [25, 26], the results of which are given in Tables 4, 5.

The lower the value of the avalanche parameter, the stronger the avalanche effect in the transformation. From Tables 4, 5 it follows that the LBC encryption algorithm satisfies the requirements of the avalanche criterion after the seventh round.

Let's look at the example of the spread of the avalanche effect. To do this, select two plain texts in hexadecimal form. Plaintext1: «AA AA AA AA AA AA AA» consists only of «AA» characters. Plaintext2: «AA AA AA AA AA BA AA AA AA» differs from the first plaintext by only one bit of the character «B». The following sequence of characters is selected as the key: "30 B5 13 CE BB 69 0B F3 95 33". Table 6 shows the avalanche effect of the LBC algorithm from one to seven rounds.

Table 4

Analysis of the avalanche effect of the LBC algorithm after the first round

i	ε_i
1	0.906
2	0.906
3	0.938
4	0.938
5	0.906
6	0.969
7	0.969
8	0.906
9	0.875
10	0.969
11	0.906
12	0.906
13	0.875
14	0.969
15	0.906
16	0.969
17	0.875
18	0.938
19	0.906
20	0.906
21	0.938
22	0.938
23	0.906
24	0.938
25	0.938
26	0.938
27	0.906
28	0.906
29	0.875
30	0.969
31	0.906
32	0.625
33	0.563
34	0.875
35	0.688
36	0.813
37	0.875
38	0.688
39	0.750
40	0.625
41	0.813
42	0.750
43	0.625
44	0.750
45	0.813
46	0.750
47	0.688
48	0.906
49	0.906
50	0.938
51	0.969
52	0.938
53	0.938
54	0.938
55	0.906
56	0.938
57	0.969
58	0.938
59	0.906
60	0.906
61	0.938
62	0.938
63	0.906
64	0.938

Table 5

Analysis of the avalanche effect of the LBC algorithm after the seventh round

i	ε_i
1	0.031
2	0.063
3	0.031
4	0.125
5	0.094
6	0.094
7	0.125
8	0.250
9	0.031
10	0.000
11	0.094
12	0.188
13	0.063
14	0.031
15	0.031
16	0.219
17	0.219
18	0.219
19	0.031
20	0.125
21	0.094
22	0.094
23	0.063
24	0.063
25	0.063
26	0.125
27	0.000
28	0.031
29	0.000
30	0.000
31	0.094
32	0.031
33	0.000
34	0.063
35	0.063
36	0.188
37	0.000
38	0.125
39	0.094
40	0.031
41	0.063
42	0.188
43	0.094
44	0.094
45	0.031
46	0.250
47	0.344
48	0.125
49	0.156
50	0.188
51	0.188
52	0.031
53	0.125
54	0.156
55	0.125
56	0.063
57	0.156
58	0.156
59	0.063
60	0.063
61	0.094
62	0.156
63	0.063
64	0.094

The LBC algorithm uses the following conversions: the *S*-block of substitution (*S*), the *RL* function, and the bitwise addition of the key (*xor*) operation. The *xor* operation does not affect the propagation of aggregate changes.

Table 6

An example of the avalanche effect of the LBC algorithm

Round Number	F	Ciphertext 1	Ciphertext 1	A	B
1	S	18 2c 31 70 22 e4 82 61	18 2c 31 70 92 e4 82 61	1	6
	R	a3 73 6f 76 28 fa 96 9e	a3 73 df 76 28 fa 96 9e	1	6
2	S	84 d4 c0 52 a5 c8 17 1b	84 d4 40 52 a5 c8 17 1b	1	6
	R	1f 71 24 e3 5f 16 d6 58	9f 71 24 e3 5f 16 d6 58	1	6
3	S	2c d2 ad c3 6c 27 f7 65	1c d2 9d 1b 6c 27 f7 65	3	19
	R	66 9c 45 18 4b 9a b4 5a	56 44 45 18 4b 9a 84 5a	4	25
4	S	77 1e 87 dc 03 18 30 68	67 00 e0 8a 03 18 50 68	7	44
	R	32 ba d8 20 74 c4 17 36	55 ec d8 20 14 c4 07 28	8	50
5	S	4a 38 42 9c d0 e0 2d 47	66 be 35 be 20 e0 9d a5	11	68
	R	31 e6 4d 2d 2c e7 7a 11	46 c4 bd 2d 9c 05 56 97	12	75
6	S	42 b7 cb b3 ae bd d8 22	07 e0 48 53 1e 96 67 1d	14	87
	R	92 21 98 88 08 d9 04 78	11 c1 28 a3 b7 e6 41 2f	14	87
7	S	1a a2 d6 90 95 f1 90 d5	22 e2 7e fc 3d b7 02 ac	15	94
	R	d3 d4 d3 69 cc d6 fb e9	7b b8 7b 2f 5e af c3 a7	16	100

Column A indicates the number of changed bits, column B indicates the approximate percentage; gray are the changed positions of the ciphertexts. The avalanche effect reaches 100 % after the seventh round.

A similarly stronger criterion, called the strict avalanche criterion (SAC), requires that for any i and j, when the input bit i of the A-matrix is inverted, any output bit j changes with a probability of 1/2.

Table 7 shows statistical safety indicators, where d_c is the degree of completeness, d_{sa} is the degree of strict avalanche criterion, d_a is the degree of avalanche effect.

Table 7

Indicators of statistical stability of the LBC algorithm

Round	d_c	d_{sa}	d_a
1	0.105468	0.085441	0.118147
2	0.229003	0.196785	0.227644
3	0.417724	0.371504	0.408414
4	0.652343	0.600590	0.632497
5	0.84375	0.796862	0.819022
6	0.953125	0.921488	0.937189
7	1	0.980191	0.990348
8	1	0.991667	0.998851
9	1	0.992272	0.999134
10	1	0.991969	0.999070
11	1	0.992095	0.998978
12	1	0.992066	0.999068
13	1	0.992104	0.999139
14	1	0.991972	0.999089
15	1	0.991820	0.998888
16	1	0.992118	0.998876
17	1	0.991917	0.999034
18	1	0.992010	0.999148
19	1	0.991977	0.999034
20	1	0.992092	0.999010

The data given in Table 7 show that the depth of the avalanche effect of the LBC algorithm is reached in the seventh round.

5. 2. 3. Differential cryptanalysis of the LBC algorithm

For differential cryptanalysis, two pairs of pre-selected ciphertexts A_1 and A_2 are taken, where the attacker calculates the differential: $\Delta A=A_1\oplus A_2$, and with the help of a calculated differential tries to determine what the differential of ciphertexts should be $\Delta B=B_1\oplus B_2$.

In most cases, the probability of an attacker guessing the exact value of ΔB is extremely low. An attacker is able to determine the frequency of returning ΔB for a given ΔA, which in turn gives him the opportunity to obtain part of the key or the whole key.

The operation of bitwise cyclic shift and *xor* do not have any effect on the difference, it is easy to check. However, in order to construct multi-round characteristics, it is important to determine exactly how the values of the desired bytes will be converted, which will be obtained after other transformations. When the round key is added modulo 2, its bits will be mutually destroyed. Therefore, the value of the key also does not affect the differentials. In this regard, the change in difference depends mainly on the differential properties of *S*-blocks.

The LBC algorithm uses a 4-bit *S*-block. For the *S*-block, it is necessary to build a table of the distribution of the difference. The methodology for constructing such a table is given in [27, 28]. When conducting differential cryptanalysis, it is necessary to track all combinations of binary vectors, add all possible input and output elements of *S*-block, count its zeros, and enter the results into the difference distribution table (Table 8).

Table 8

Table of the distribution of the difference for the *S*-block

$\Delta X/\Delta Y$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	0	0	2	0	2	2	0	0	2	4	0	0	2	0
2	0	2	4	0	2	2	2	0	0	0	2	0	2	0	0
3	2	2	0	2	0	2	0	2	2	0	0	0	4	0	0
4	0	0	2	2	0	0	0	0	4	0	2	0	2	2	2
5	0	2	2	2	0	0	2	2	0	2	0	0	0	0	4
6	0	2	0	0	0	2	0	0	0	2	0	2	2	4	2
7	0	0	0	4	2	0	2	0	2	2	0	2	2	0	0
8	2	2	2	0	0	0	2	0	2	0	0	4	0	2	0
9	0	0	0	0	0	2	2	4	2	0	2	2	0	0	2
10	4	2	0	0	2	0	0	0	2	2	2	0	0	0	2
11	2	0	2	0	0	0	0	2	0	4	2	2	2	0	0
12	2	0	0	0	2	0	4	2	0	0	0	0	2	2	2
13	2	0	2	2	2	4	0	0	0	0	0	2	0	0	2
14	0	4	0	2	2	0	0	2	0	0	2	2	0	2	0
15	0	0	2	0	4	2	0	2	2	2	0	0	0	2	0

In Table 8 of the distribution of differences, the maximum value of the probability is the values 4/16=1/4. For

convenience, Table 9 shows the input and corresponding output differences with maximum values (in hexadecimal form).

Table 9

Result of difference conversion for *S*-block

ΔX	ΔY	ΔX	ΔY	ΔX	ΔY	ΔX	ΔY	ΔX	ΔY
0x01	0x0b	0x02	0x03	0x03	0x0d	0x04	0x09	0x05	0x0f
0x07	0x04	0x08	0x0c	0x09	0x08	0x0a	0x01	0x0b	0x0a
0x0d	0x06	0x0e	0x02	0x0f	0x05	0x06	0x0a	0x0c	0x07

In Table 9, ΔX is the input to the substitution block, and ΔY is the value obtained at the output of the substitution block, corresponding to ΔX.

The next encryption step (*RL* transform) performs linear transformations using a cyclic shift and an *xor* operation. They do not have any effect on the change in the probability of the difference transformation since these shifts are performed only within one subblock.

Based on the differential properties of the transformations used in the encryption algorithm, we construct several round characteristics and obtain their probability. The main task is to find the segment of the ciphertext where the least number of active *S*-blocks was affected. The probability of obtaining the correct pair of texts for a given characteristic depends on this. Then, you need to determine the number of rounds for the algorithm, which will make it possible to analyze the ciphertext faster than the brute force method. The brute force complexity for a 64-bit block of data and an 80-bit key length is 2^{80}.

The 64-bit input block is divided into 4 sub-blocks of 16 bits each and designated as X_1, X_2, X_3, X_4. During the analysis, it was established that the smallest mixing of bits occurs in the second sub-block (X_2). As it was noted earlier, all outgoing bits depend on all incoming bits after the seventh round. Therefore, the description of the output equation for each X_i, $i=\overline{1,4}$ after the seventh round will be needed to find the correct pairs. Two transformations are used in the structure of the algorithm: *S*-block and $R(X)=X\oplus(X\lll 7)\oplus(X\lll 10)$. Denote the corresponding output expressions for each X_i via Y_i, $i=\overline{1,4}$:

$$Y_0=RS\left(\begin{array}{l} RS\left[\begin{array}{l} RS\left(\begin{array}{l} RS\left[\begin{array}{l} RS\left(\begin{array}{l} RS(X_1)\oplus \\ \oplus S(X_2) \end{array}\right)\oplus \\ \oplus\oplus SS(X_3) \end{array}\right]\oplus \\ \oplus SSS(X_4) \end{array}\right)\oplus \\ \oplus SSSS(X_1) \end{array}\right]\oplus \\ \oplus SSSS\left[RS\left(RS(X_1)\oplus S(X_2)\right)\oplus SS(X_3)\right] \end{array}\right)\oplus$$
$$\oplus SSSS\left\{RS\left[RS\left(RS(X_1)\oplus S(X_2)\right)\oplus SS(X_3)\right]\oplus SSS(X_4)\right\},$$

$$Y_1=SSS\left(\begin{array}{l} RS\left\{\begin{array}{l} RS\left[\begin{array}{l} RS\left(RS(X_1)\oplus S(X_2)\right)\oplus \\ \oplus SS(X_3) \end{array}\right]\oplus \\ \oplus SSS(X_4) \end{array}\right\}\oplus \\ \oplus SSSS(X_1) \end{array}\right),$$

$$Y_3=S\left(\begin{array}{l} RS\left[\begin{array}{l} RS\left(\begin{array}{l} RS\left\{\begin{array}{l} RS\left[\begin{array}{l} RS\left(\begin{array}{l} RS(X_1)\oplus \\ \oplus S(X_2) \end{array}\right)\oplus \\ \oplus SS(X_3) \end{array}\right]\oplus \\ \oplus SSS(X_4) \end{array}\right\}\oplus \\ \oplus SSSS(X_1) \end{array}\right)\oplus \\ \oplus SSSS\left(RS(X_1)\oplus S(X_2)\right) \end{array}\right]\oplus \\ \oplus SSSS\left[RS\left(RS(X_1)\oplus S(X_2)\right)\oplus SS(X_3)\right] \end{array}\right).$$

From the obtained equations, it can be seen that the second subblock (Y_1) uses the smallest number of transformations. Therefore, after going through the rounds of encryption step by step, we will show one pair with the highest probabilities.

Any pairs of input and output data with values in cells 4 can be selected from the distribution table (table 8) of differences since the number four is the largest number in the table. Let's choose a pair with input 1 and output 11 and write them in the hexadecimal number system for one subblock: $0x1111 \overset{P=1/4}{\rightarrow} 0xBBBB$. At the next stage of encryption, the output difference is transformed through the *R* function and since it is a linear function, the regularity and probability of differences are preserved: $0xBBBB \overset{P=1/4}{\rightarrow} 0x888A$. If what we have found so far belongs to the first subblock ($RS(X_1)$), then we can choose these differences as the correct pair for the second subblock. In general, you can choose any pair from Table 9 as they are considered correct. Each row of the table contains one cell with the number 4, that is, each input has a corresponding output element with equal probability. So, $0x888A \rightarrow 0x3331$ are the correct pairs for the expression $S(X_2)\oplus RS(X_1)$ and their probability is equal to $\frac{1}{4}\cdot\frac{1}{4}=\frac{1}{16}=\left(\frac{1}{2}\right)^4$. Since the *S*-block of replacement is used in the next step of encryption, according to Table 9, for $0x3331$ the difference will most likely be $0xDDDB$.

According to the scheme of the encryption algorithm, if we choose pairs with a high probability, as indicated above, we will end up with the following input and output sequences for seven rounds:

$$0x1111 \rightarrow 0xBBBB \rightarrow 0x888A \rightarrow 0x3331 \rightarrow$$
$$\rightarrow 0xDDDB \rightarrow 0x5F40 \rightarrow 0xF5E8 \rightarrow$$
$$0x5F2C \rightarrow 0x787F \rightarrow 0x696E \rightarrow 0xE8E2 \rightarrow$$
$$\rightarrow 0x1235 \rightarrow 0xA98E \rightarrow 0x18C5 \rightarrow$$
$$0xBC7F \rightarrow 0xA745.$$

Therefore, for the input difference $0x1111$, the most likely output difference will be $0xA745$. It is not difficult to calculate that the S-block occurs in the output expression for the second sub-block 13 times, i. e., the probability of the desired difference will be equal to $\left(\frac{1}{4}\right)^{13}=\left(\frac{1}{2}\right)^{26}$. Then, by analyzing the movement of each element of the open text and monitoring the execution of the substitution at each step, it was determined that the probability of finding the key for twelve rounds of the cipher will be equal to $\frac{1}{2^{98}}$.

5. 2. 4. Linear cryptanalysis of the LBC algorithm

To start the analysis, we fill in the linear approximation table (LAT) for the S-block.

During the construction of the table, all possible combinations of input and output binary vectors are traced. Each pair of vectors is used as a mask, which is applied to all possible input-output pairs of the replacement block and is determined by the following ratio:

$$LAT(\alpha,\beta) \overset{def}{=} \left\{ X \mid X \in Z_2^8, \bigoplus_{i=1}^{8} X[i]\cdot\alpha[i] = \right\} \left\{ = \bigoplus_{i=1}^{8} S\left(X[i]\right)\cdot\beta[i] \right\}, \quad (19)$$

where $\alpha, \beta \in Z_{256}$ and the multiplication sign denotes the scalar product operation [13, 29, 30]. The obtained values are shown in Table 10.

Table 10

L.1—Linear approximation table for *S*-block

Input Mask/Output Mask	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	8	8	8	10	10	10	10	6	6	10	10	4	12	8	8
2	8	8	4	8	8	12	8	10	10	10	6	10	10	6	10
3	8	12	8	6	6	6	10	8	8	8	12	10	10	6	10
4	8	10	6	12	8	6	6	10	6	8	8	6	6	4	8
5	8	10	6	6	10	8	8	4	8	6	6	6	6	8	12
6	4	10	6	8	8	6	6	8	8	6	6	8	12	10	6
7	4	6	10	6	6	8	8	10	10	8	8	4	8	6	10
8	10	8	6	8	10	8	6	10	12	6	12	6	8	10	8
9	6	8	10	10	8	10	4	8	6	8	10	10	8	10	12
10	10	8	10	12	6	8	10	8	10	4	6	8	10	8	10
11	6	12	10	10	8	10	8	6	12	10	8	8	6	8	6
12	10	6	8	8	6	6	4	4	10	10	8	8	10	6	8
13	6	6	4	10	4	8	10	6	8	8	10	8	6	10	8
14	6	6	8	8	10	10	8	6	8	4	10	10	8	4	6
15	10	10	8	6	4	12	6	8	6	6	8	6	8	8	6

In the linear approximation table (Table 10), the first column contains the input masks, and the first row contains the output masks. If the 4-bit linear equation is satisfied 0 times, then it can be concluded that this 4-bit linear relation is absent for this particular *S*-block. If the 4-bit linear equation is satisfied 16 times, then it is also possible to conclude that this 4-bit linear relation is present for this particular 4-bit *S*-block. In both cases, full information is given to cryptanalysts. The result is better for cryptanalysts if the probability of the presence or absence of unique 4-bit linear equations is far from 1/2 or close to 0 or 1. If the probabilities of the presence or absence of all unique 4-bit linear relationships are equal to 1/2 or close to 1/2, then they say that it is difficult to perform linear cryptanalysis on a 4-bit *S*-block. Therefore, the result for the cryptanalyst will be better if the number of eights in the table is smaller. If the number of eights is much greater than the second numbers in the table, then it is said that the 4-bit cryptographic *S*-block is more resistant to linear cryptanalysis [12, 31, 32].

In LAT, cells with numbers farthest from the number 8 consist of 12 or 4. Therefore, according to the table, the effective linear equations necessary for further analysis and the probability of each of them is 3/4, i. e., with the largest deviations from 1/2:

$$x[3]\oplus s[0]\oplus s[1]=1,$$

$$x[3]\oplus s[0]\oplus s[1]\oplus s[3]=0,$$

$$x[2]\oplus s[2]\oplus s[3]=1,$$

$$x[2]\oplus s[1]\oplus s[2]=0,$$

$$x[2]\oplus x[3]\oplus s[2]=0,$$

$$x[2]\oplus x[3]\oplus s[0]\oplus s[2]\oplus s[3]=0,$$

$$x[1]\oplus s[1]=0,$$

$$x[1]\oplus s[0]\oplus s[1]\oplus s[2]=1,$$

$$x[1]\oplus x[3]\oplus s[0]=1,$$

$$x[1]\oplus x[3]\oplus s[0]\oplus s[1]\oplus s[2]\oplus s[3]=0,$$

$$x[1]\oplus x[2]\oplus s[3]=1,$$

$$x[1]\oplus x[2]\oplus s[0]\oplus s[1]\oplus s[3]=0,$$

$$x[1]\oplus x[2]\oplus x[3]\oplus s[3]=1,$$

$$x[1]\oplus x[2]\oplus x[3]\oplus s[0]\oplus s[1]=1,$$

$$x[0]\oplus s[0]\oplus s[3]=0,$$

$$x[0]\oplus s[0]\oplus s[2]\oplus s[3]=0,$$

$$x[0]\oplus x[3]\oplus s[1]\oplus s[2]\oplus s[3]=1,$$

$$x[0]\oplus x[3]\oplus s[0]\oplus s[1]\oplus s[2]\oplus s[3]=0,$$

$$x[0]\oplus x[2]\oplus s[1]=0,$$

$$x[0]\oplus x[2]\oplus s[0]\oplus s[2]=1,$$

$$x[0]\oplus x[2]\oplus x[3]\oplus s[2]=0,$$

$$x[0]\oplus x[2]\oplus x[3]\oplus s[0]\oplus s[3]=0,$$

$$x[0]\oplus x[1]\oplus s[1]\oplus s[2]\oplus s[3]=1,$$

$$x[0]\oplus x[1]\oplus s[0]=1,$$

$$x[0]\oplus x[1]\oplus x[3]\oplus s[2]\oplus s[3]=1,$$

$$x[0]\oplus x[1]\oplus x[3]\oplus s[1]\oplus s[3]=1,$$

$$x[0]\oplus x[1]\oplus x[2]\oplus s[0]\oplus s[2]=1,$$

$$x[0]\oplus x[1]\oplus x[2]\oplus s[0]\oplus s[1]\oplus s[2]=1,$$

$$x[0]\oplus x[1]\oplus x[2]\oplus x[3]\oplus s[1]\oplus s[3]=1,$$

$$x[0]\oplus x[1]\oplus x[2]\oplus x[3]\oplus s[1]\oplus s[2]=0.$$

First, let's introduce the notation, $k[i,j]$ are the elements of the round keys, $x[i,j]$ are the input values of the block $s[i,j]$, and $y[i,j]$ are the output values of the *S*-block of substitution and the RL-function, where i is the round number $\left(i=\overline{1,20}\right)$ and j is the position number in the block, $\left(j=\overline{0,64}\right)$.

As in differential cryptanalysis, we will analyze the encryption algorithm on seven rounds since all output bits depend on all input bits after the seventh round. Also, we

choose the output equation for the second sub-block since the output equation has the smallest number of variables:

$$Y_1 = \\ = SSS\left(\begin{array}{l} RS\left\{\begin{array}{l} RS\left[RS\left(\begin{array}{l} RS(X_1)\oplus \\ \oplus S(X_2)\end{array}\right)\oplus SS(X_3)\right]\oplus \\ \oplus SSS(X_4)\end{array}\right\}\oplus \\ \oplus SSSS(X_1)\end{array}\right). \quad (20)$$

From among the linear effective equations obtained from *S*-blocks, we choose $x[1]\oplus s[1]=0$. Since the input and output variables are involved only once, this in turn is convenient for further analysis. Let's write this equation for the first subblock in the first round (taking into account that the key is added): $x[0,1]\oplus k[0,1]\oplus s[0,1]\oplus 1=0$. The probability of this equation is 3/4. Further, according to the encryption scheme, the function *R* is performed by determining the sums corresponding to the output element *y*[0, 1]. Fig. 3 shows the *R*-function for the first sub-block.

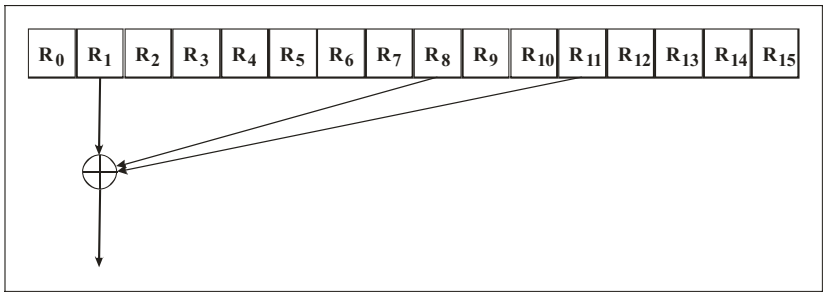

Fig. 3. R function for the first subblock

According to the scheme of the R-function shown in Fig. 3, the output element *y*[0, 1] consists of the sum of the first, eighth, and eleventh elements, $y[0,1]=s[0,1]\oplus s[0,8]\oplus s[0,11]$. *s*[0, 8] and *s*[0, 11] are the 0th and 2nd output bits of the *S*-block of replacement. Therefore, for these variables, it is better to choose the equations $x[1]\oplus x[3]\oplus y[0]=0$ and $x[2]\oplus x[3]\oplus y[2]=1$.

According to the scheme of the encryption algorithm, the variable *y*[0, 1] is combined with the variable *y*[0, 17] to obtain the output expression y[1, 1] for the first round. The most linearly approximate equation for *y*[0,17], taking into account the first selected equation, has the form:

$$y[1,1]=x[0,1]\oplus x[0,8]\oplus x[0,11]\oplus x[0,17]\oplus \\ \oplus k[0,1]\oplus k[0,8]\oplus k[0,11]\oplus x[0,17]. \quad (21)$$

Then, according to lemma 1, the probability of a given effective equation will be equal to $\frac{1}{2}+2^2\cdot\left(\frac{1}{2}-\frac{1}{4}\right)^3=\frac{1}{2}+2^{-4}$.

If we use equation (21) for the second round, we get the following equation:

$$y[2,1]=y[1,1]\oplus y[1,8]\oplus y[1,11]\oplus y[1,33]\oplus k[1,1]\oplus \\ \oplus k[1,8]\oplus k[1,11]\oplus k[2,33].$$

Given that each *y*[1, i], *i*=1,8,11,33 consists of input variables in the resulting equation and under the influence of the R-function, the number of variables in the first subblock doubles at each step. Also, as a result of the analysis, it was found that 31 variables are involved in the equation obtained after the seventh round. Let's write the equation of linear approximation of the output element for the second subblock in the seventh round, using equation (20):

$$y[7,1]\oplus y[6,1]\oplus y[6,8]\oplus y[6,11]\oplus y[6,49]\oplus \\ \oplus k[6,1]\oplus k[6,8]\oplus k[6,11]\oplus k[7,49]=0.$$

We calculate the deviation of the probability of the resulting equation from 0.5, according to lemma 1:

$$\varepsilon=2^{31-1}\cdot\left(\frac{1}{2}-\frac{1}{4}\right)^{31}=2^{-32}.$$

The next stage consists in finding the number of pairs of open texts and their probabilities, which will allow solving these equations. For these purposes, we use Lemma 2. To facilitate the calculation, we select from the table of normal distributed values that have one of the highest probabilities, for example 0.977. Then:

$$-2\sqrt{N}\left|p-\frac{1}{2}\right|\approx -2\Rightarrow N= \\ =\left(\frac{1}{0.5+2^{-32}-0.5}\right)^2=2^{64}.$$

To perform an effective attack using linear cryptanalysis, 2^{64} open/closed text pairs are needed.

5. 2. 5. Investigation of the hardware performance of the LBC algorithm

The main characteristics of the hardware implementation of the algorithm are given in Table 11. The calculation of the number of conditional logical elements was carried out according to the methodology presented in article [33].

Table 11

Required area for the LBC cipher

Algorithm LBC			Schedule of round keys		
module	GE	%	module	GE	%
data state	384.39	19.55	KS: key state	480.49	24.44
S-box	448.45	22.81	KS: S-box	112.11	5.70
R-functuion	128.13	6.52	KS: Shift, Rotation	0	0
Permutation	0	0	KS: counter-XOR	213.55	10.86
counter: state	28.36	1.44	key-XOR	170.84	8.69
Total				1966.32	100

From Table 11, it follows that the main part of the area is occupied by the place for storing the key, the state of the data, and the *S*-block. They are followed by the addition of a key. The permutation can be implemented with simple wiring and therefore does not require GE. It follows from the above that the total GE in the developed algorithm is 1966.32, and thus the LBC cipher can be attributed to the low-cost group. Therefore, this architecture is suitable for low-cost devices, in addition, it can be used to create a cryptographic coprocessor with low area consumption. A comparison of the obtained results with other ciphers is given in Table 12.

The developed LBC algorithm, like Present-80, can be effectively used as a hardware implementation for devices with low hardware resources.

The results presented in Table 13 indicate that the encryption time is much less than Present.

Table 12

Comparative characteristics of ciphers

Ciphers	Block length	Key length	GE
LBC	64	80	1966
Present-80[G3]	64	80	1570

Table 13

The results of research on the performance of ciphers

No.	Device	Algorithm name	File(data) size	File (data) encryption time	Key establishment time
1	Arduino Uno R3 with ATmega328 controller	LBC	8 bytes	100.7 μs	965.59 μs
2		Present		2111.56 μs	1541.31 μs

6. Discussion of results of the study of the LBC algorithm

In the development and research of the cryptographic properties of the LBC algorithm, the main reference was to take the lightweight PRESENT algorithm. In 2012, the PRESENT algorithm was included in the international standard for lightweight encryption ISO/IEC 29192-2:2012.

The developed lightweight LBC algorithm is structurally different from the lightweight algorithms that exist today.

The LBC algorithm has its own 4-bit lookup table (Table 1). For the linear transformation, the *RL* function was used. This function is added to achieve a good quality of the "avalanche effect". And the cyclic shift of subblocks provides good bit shuffling along with the *RL* function.

The number of rounds of the LBC algorithm is 20. According to the results of the study, a good avalanche effect is achieved starting from the 7^{th} round (Table 7). Therefore, 20-round conversions provide sufficient and good crypto strength.

Also, the most distinctive feature of the LBC algorithm from Present is the generation of round keys. It uses the shift functions R_i, similar to *RL*, which is used in transformations of the main encryption algorithm. While in the Present algorithm only the leftmost 4 bits of the master key are replaced via the *S*-block, then in the LBC algorithm, 16 bits of the leftmost subblock of the key are replaced using the *S*-block used in the main encryption process. Here, the R_i function and the cyclic shift of the subblocks of the master key provide a good avalanche effect, thereby improving the properties of round keys. The algorithm has a simple but flexible structure. The flexibility relates to the fact that, if necessary, you can increase the block size and the size of the master key by adding an additional subblock with a size of 16 bits. At the same time, the basic structure of the algorithm is preserved.

Theoretical and experimental tests have shown that the algorithm fully complies with the basic cryptographic requirements.

The first study was to test the indicators of the avalanche effect. For this, a special program was created. The program read the avalanche criterion and the values of the avalanche parameter as each bit of the input data block changed. Table 4 shows the results of the study of the avalanche effect in the first round with changes in the first bit of the input block in each round. Table 4 shows that the average value of the avalanche effect is 0.8766, the maximum value is 0.969. After the seventh round, the average value is 0.0992. Here you can see that, starting from the 7th round, the average value approaches 0.09, in turn, this property shows a good quality of the avalanche effect. The smaller the value of the avalanche parameter, the stronger the avalanche effect in the transformation. From Tables 4, 5 it follows that the LBC encryption algorithm satisfies the requirements of the avalanche criterion after the seventh round. Therefore, we believe that 20-round transformations sufficiently provide good cryptographic strength. And in the study of the avalanche effect, various options for the input block were taken and the results are shown in Table 6.

During the research, the per-cycle values of the statistical safety indicators of the LBC algorithm were also calculated. For this, another computer program has been created. To compare the values of statistical safety indicators, the per-cycle values of the Present algorithm were additionally calculated. As can be seen from the table, the value of the degree of completeness (d_c) for the Present algorithm acquires a good indicator starting from the second round (Table 5). And for the LBC algorithm, the value of the degree of completeness (d_c) from the 8^{th} round acquires the values 1 (Table 7). This indicator also proves the good qualities of the avalanche effect.

The second stage of the study of the properties of the LBC algorithm was two types of cryptanalysis: differential and linear.

When conducting differential cryptanalysis, it was found that for an input difference of $0x1111$, the most probable output difference would be $0xA745$. It is easy to calculate that the *S*-block occurs 13 times in the output expression for the second sub-block, i. e., the probability of the desired difference is equal to $\frac{1}{4}\cdot\frac{1}{4}=\frac{1}{16}=\left(\frac{1}{2}\right)^4$ and, as a result, a suffi cient level of security is not provided. When analyzing the movement of each element and tracking the substitution at each step, it was found that after the twelfth round, the *S*-block will be used 127, 49, 67, and 92 times on each subblock. Repeatedly choosing differentials with maximum probabilities, it turns out that the probability of finding the key for twelve rounds of the cipher will be equal to $\frac{1}{2^{98}}$.

The results of linear cryptanalysis showed that after the seventh round, the ciphertexts repeat the properties of random substitutions, for an effective attack using linear cryptanalysis, 2^{64} open/closed text pairs are needed. Due to the fact that 64-bit blocks are fed into the encryption algorithm, the maximum possible number of open/closed text pairs is 2^{64}. Thus, there is no need to perform linear cryptanalysis for more rounds.

In practical use, the developed encryption algorithm does not require significant restrictions, but when implementing this algorithm in devices, it is necessary to know the main characteristics of this device. The results of the study, obtained during the assessment of reliability and speed, showed that the developed lightweight encryption algorithm fully complies with the basic requirements. It was noted that, taking into account modern technological capabilities, the length of the block and the master key can be increased.

Research work in this direction can be continued in terms of a deeper analysis of the security of the developed lightweight encryption algorithm. In particular, it is possible to carry out a number of other cryptographic attacks to improve the algorithm.

7. Conclusions

1. During the implementation of our task, a block diagram of the algorithm was developed, which meets the basic requirements for cryptographic transformations. The scheme of the algorithm is built in such a way as to increase performance due to parallel computing, by dividing the block into subblocks. If necessary, the structure of the algorithm make it possible to increase the size of the block and the key master by adding an additional subblock, which in turn enhances the cryptographic strength.

Also, counting the number of conditional logic elements and comparing the performance of ciphers with the Present algorithm showed the effectiveness of its hardware implementation. That is, the number of GE counts confirms that the algorithm falls into the "low-cost" implementation, which requires no more than 2000 GE. The time spent by the LBC encryption algorithm is 20 times faster than Present, and by the LBC key generation algorithm it is 1.5 times faster than Present.

2. Based on the tests and studies carried out, it was concluded that the LBC encryption algorithm is effective in providing an avalanche effect since it meets the requirements of the avalanche criterion after the seventh round.

Differential linear cryptanalysis of the algorithm showed that the obtained probability value by differential cryptanalysis exceeds the complexity of exhaustive search, i.e., after the eleventh round, the encryption algorithm is resistant to differential cryptanalysis. Due to the extremely low probability of obtaining correct pairs of texts, the algorithm for finding such texts cannot be implemented by the means that are currently available.

The obtained results of linear cryptanalysis indicate that in the LBC encryption algorithm, linear indicators after the seventh round repeat the properties of random substitutions. The probability of finding correct pairs is very small and tends to the complexity of the exhaustive search of the algorithm, which allows us to conclude that it is resistant to linear cryptanalysis.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study and the results reported in this paper.

Funding

The research work was carried out within the framework of the project AP09259570 "Development and research of a domestic lightweight encryption algorithm with limited resources" at the Institute of Information and Computing Technologies of CS MES RK.

Data availability

Manuscript has no associated data.

References

1. Usman, M., Ahmed, I., Imran, M., Khan, S., Ali, U. (2017). SIT: A Lightweight Encryption Algorithm for Secure Internet of Things. International Journal of Advanced Computer Science and Applications, 8 (1). doi: https://doi.org/10.14569/ijacsa.2017.080151
2. Yun, J., Kim, M. (2020). JLVEA: Lightweight Real-Time Video Stream Encryption Algorithm for Internet of Things. Sensors, 20 (13), 3627. doi: https://doi.org/10.3390/s20133627
3. Taresh, H. (2018). LT10 a lightweight proposed encryption algorithm for IOT. Iraqi Journal for Computers and Informatics, 44 (1), 1–5. doi: https://doi.org/10.25195/ijci.v44i1.64
4. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L. (2013). Paper 2013/404. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2013/404
5. Gu, D., Li, J., Li, S., Ma, Z., Guo, Z., Liu, J. (2012). Differential Fault Analysis on Lightweight Blockciphers with Statistical Cryptanalysis Techniques. 2012 Workshop on Fault Diagnosis and Tolerance in Cryptography. doi: https://doi.org/10.1109/fdtc.2012.16
6. Kumar V G, K., Rai C, S. (2021). Design and Implementation of Novel BRISI Lightweight Cipher for Resource Constrained Devices. Microprocessors and Microsystems, 84, 104267. doi: https://doi.org/10.1016/j.micpro.2021.104267
7. Yang, W., Wang, R., Guan, Z., Wu, L., Du, X., Guizani, M. (2020). A Lightweight Attribute Based Encryption Scheme with Constant Size Ciphertext for Internet of Things. ICC 2020 - 2020 IEEE International Conference on Communications (ICC). doi: https://doi.org/10.1109/icc40277.2020.9149294
8. Kazlauskas, K., Kazlauskas, J. (2009). Key-Dependent S-Box Generation in AES Block Cipher System. Informatica, 20 (1), 23–34. doi: https://doi.org/10.15388/informatica.2009.235
9. Preneel, B. (2010). Perspectives on Lightweight Cryptography. Shanghai. Available at: https://homes.esat.kuleuven.be/~preneel/preneel_lightweight_shanghaiv1.pdf
10. Ivanov, G., Nikolov, N., Nikova, S. (2016). Cryptographically Strong S-Boxes Generated by Modified Immune Algorithm. Lecture Notes in Computer Science, 31–42. doi: https://doi.org/10.1007/978-3-319-29172-7_3
11. Horbenko, I. D., Horbenko, Yu. I. (2012). Prykladna kryptolohiya. Teoriya. Praktyka. Zastosuvannia. Kharkiv: Vydavnytstvo «Fort», 870.
12. Dey, S., Ghosh, R. (2018). A Review of Existing 4-Bit Crypto S-Box Cryptanalysis Techniques and Two New Techniques with 4-Bit Boolean Functions for Cryptanalysis of 4-Bit Crypto S-Boxes*. Advances in Pure Mathematics, 08 (03), 272–306. doi: https://doi.org/10.4236/apm.2018.83015

13. Khompysh, A., Kapalova, N., Algazy, K., Dyusenbayev, D., Sakan, K. (2022). Design of substitution nodes (S-Boxes) of a block cipher intended for preliminary encryption of confidential information. Cogent Engineering, 9 (1). doi: https://doi.org/10.1080/23311916.2022.2080623
14. Kapalova, N. A., Khaumen, A., Sa an, K. (2020). Rasseivayuschie svoystva lineynykh preobrazovaniy. Mater. nauch. konf. IIVT MON RK «Sovremennye problemy informatiki i vychislitel'nykh tekhnologiy». Almaty, 191–196. Available at: https://conf.iict.kz/wp-content/uploads/2020/10/mpcsct-collection-08.07.2020-final.pdf
15. Lisitskaya, I. V., Nastenko, A. A. (2011). Great ciphers - casual substitution. Radiotekhnika, 166, 50–55. Available at: https://openarchive.nure.ua/bitstream/document/15255/1/Radiotehnika_V166_2011_rus.pdf
16. Teh, J. S., Tham, L. J., Jamil, N., Yap, W.-S. (2022). New differential cryptanalysis results for the lightweight block cipher BORON. Journal of Information Security and Applications, 66, 103129. doi: https://doi.org/10.1016/j.jisa.2022.103129
17. Biham, E., Shamir, A. (1991). Differential Cryptanalysis of DES-like Cryptosystems. Lecture Notes in Computer Science, 2–21. doi: https://doi.org/10.1007/3-540-38424-3_1
18. Carlet, C. (2010). Vectorial Boolean Functions for Cryptography. Boolean Models and Methods in Mathematics, Computer Science, and Engineering, 398–470. doi: https://doi.org/10.1017/cbo9780511780448.012
19. Kim, J., Hong, S., Lim, J. (2010). Impossible differential cryptanalysis using matrix method. Discrete Mathematics, 310 (5), 988–1002. doi: https://doi.org/10.1016/j.disc.2009.10.019
20. Liu, Y., Liang, H., Wang, W., Wang, M. (2017). New Linear Cryptanalysis of Chinese Commercial Block Cipher Standard SM4. Security and Communication Networks, 2017, 1–10. doi: https://doi.org/10.1155/2017/1461520
21. Matsui, M. (1994). Linear Cryptanalysis Method for DES Cipher. Lecture Notes in Computer Science, 386–397. doi: https://doi.org/10.1007/3-540-48285-7_33
22. Liu, Z. (2021). Differential-linear cryptanalysis of PRINCE cipher. Chinese Journal of Network and Information Security, 7 (4), 131–140. doi: https://doi.org/10.11959/j.issn.2096-109x.2021072
23. Biryukov, A., De Cannière, C. (2011). Linear Cryptanalysis for Block Ciphers. Encyclopedia of Cryptography and Security, 722–725. doi: https://doi.org/10.1007/978-1-4419-5906-5_589
24. Borghoff, J. (2011). 4.6 Linear Cryptanalysis. Cryptanalysis of Lightweight Ciphers. Technical University of Denmark, 60–65. Available at: https://backend.orbit.dtu.dk/ws/portalfiles/portal/5456432/phd-thesis_Julia_Borghoff.pdf
25. Vergili, I., Yücel, M. D. (2001). Avalanche and Bit Independence Properties for the Ensembles of Randomly Cho-sen S-Boxes. Turkish Journal of Electrical Engineering and Computer Sciences, 9 (2), 137–146. Available at: https://journals.tubitak.gov.tr/elektrik/vol9/iss2/3
26. Shnayer, B. (2002). Prikladnaya kriptografiya. Moscow:Triumf, 816.
27. Babenko, L. K., Ischukova, E. A. (2006). Sovremennye algoritmy blochnogo shifrovaniya i metody ikh analiza. Moscow: «Gelios ARV», 376.
28. Algazy, K. T., Babenko, L. K., Biyashev, R. G., Ishchukova, E. A., Kapalova, N. A., Nysynbaeva, S. E., Smolarz, A. (2020). Differential Cryptanalysis of New Qamal Encryption Algorithm. International journal of electronics and telecommunications, 66 (4), 647–653. doi: https://doi.org/10.24425/ijet.2020.134023
29. O'Connor, L. (1995). Properties of linear approximation tables. Lecture Notes in Computer Science, 131–136. doi: https://doi.org/10.1007/3-540-60590-8_10
30. Kuznetsov, A. A., Lisitskaya, I. V., Isaev, S. A. (2011). Lineynye svoystva blochnykh simmetrichnykh shifrov, predstavlennykh na ukrainskiy konkurs. Prikladnaya radioelektronika, 10 (2), 135–140.
31. Heys, H. M. (2002). A tutorial on linear and differential cryptanalysis. Cryptologia, 26 (3), 189–221. doi: https://doi.org/10.1080/0161-110291890885
32. Kapalova, N., Sakan, K., Algazy, K., Dyusenbayev, D. (2022). Development and Study of an Encryption Algorithm. Computation, 10 (11), 198. doi: https://doi.org/10.3390/computation10110198
33. Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B. et al. (2007). PRESENT: An Ultra-Lightweight Block Cipher. Lecture Notes in Computer Science, 450–466. doi: https://doi.org/10.1007/978-3-540-74735-2_31