computation

MDPI

Article

Development and Study of an Encryption Algorithm

Nursulu Kapalova [1,2], Kairat Sakan [1,2], Kunbolat Algazy [1,2,*] and Dilmukhanbet Dyusenbayev [1]

[1] Institute of Information and Computational Technologies, Almaty 050010, Kazakhstan
[2] Faculty of Information Technology, Al-Farabi Kazakh National University, Almaty 050040, Kazakhstan
* Correspondence: kunbolat@mail.ru

Abstract: A new symmetric block cipher algorithm called AL02 has been developed. The algorithm scheme provides five-round encryption of 128-bit blocks, while the data size at the input and output of the *S*-box is 8 bits. The main transformation is the F transformation. The difference between the proposed algorithm and the classical scheme is that the F transformation provides the maximum possible dependence of the output vector bits on the input bits and is based on "modulo 2 addition" and a substitution *S*-box. To assess the strength of the AL02 algorithm, it was programmatically implemented in the C programming language. During the analysis, the cryptographic properties of the developed encryption algorithm were tested. The algorithm was tested for statistical security. For an experimental assessment, in order to ensure that the ciphertext is not inferior to a random sequence in its properties, the well-known sets of statistical tests by NIST (National Institute of Standards and Technology) and Donald Knuth were used. The property of the avalanche effect was also checked. The strength was evaluated using the methods of differential and linear cryptanalysis.

Keywords: cryptanalysis; cryptography; encryption algorithm; information security; statistical tests

Citation: Kapalova, N.; Sakan, K.; Algazy, K.; Dyusenbayev, D. Development and Study of an Encryption Algorithm. *Computation* **2022**, *10*, 198. https://doi.org/10.3390/computation10110198

Academic Editor: Xinwei Cao

Received: 22 September 2022
Accepted: 1 November 2022
Published: 4 November 2022

1. Introduction

The development of new information technologies is accompanied by negative phenomena such as industrial espionage, computer crimes, and unauthorized access to confidential information.

Due to humans' growing dependence on information technology and the need to ensure information security, the protection of information, including the use of cryptographic methods, has become important for society as a whole.

Modern cryptography is a specific research area at the intersection of mathematics and computer science, which forms one of the main subsystems of any information security system. Of the means of cryptographic protection, symmetric block ciphers have gained widespread popularity due to their practicality.

A strong block cipher must satisfy certain conditions. These conditions are formulated in [1], which is fundamental in the theory of encryption. A strong cipher must have confusion and diffusion properties. Confusion hides the relationship between the ciphertext and the key, while diffusion hides the relationship between the ciphertext and the plaintext.

To meet the requirements for block ciphers, modern encryption algorithms use various cryptographic transformations. This also applies to linear transformations, including MDS matrices over finite fields.

As a result of scientific developments, new-generation algorithms with the SP-network structure have appeared. The general design of these algorithms is a variant of the substitution–permutation network (SP network). This network uses an iterative transformation consisting of a substitution layer (non-linear elements), a linear (mixing) layer, and a key adding layer. By transforming the entire data block at each iteration, this design provides much faster mixing of the input vector compared to the Feistel network. This effect is enhanced by an MDS matrix (Maximal Distance Separable matrix) in the linear layer. This matrix provides the maximum possible dependence of the bits of the output

vector on the bits of the input, and it is used in linear transformations in many well-known encryption algorithms [2].

In [3], a new method for estimating the integral property, truncated, and impossible differentials for substitution–permutation network (SPN) block ciphers is proposed. This method was tested only for truncated algorithms.

The paper [4] considers the structure and properties of a cryptographic information protection algorithm model based on non-positional polynomial notations (NPNs) and constructed on the basis of a SP network. Taking into account the properties of the NPN, the speed of the algorithm in the case of its implementation was not considered.

In symmetric block algorithms, a well-tested MDS transform is used as a linear transformation block. It gives the best diffusion properties. MDS codes (Maximum Distance Separable codes) are mainly used to multiply an original text vector by some matrix of constants, which is usually called an MDS matrix [5]. The scheme for generating subkeys (round keys) of known algorithms has the following disadvantages:

– The ability to restore the master key using one subkey;
– Fairly simple dependencies between subkeys (vulnerability to attacks on related keys);
– The first of the subkeys is the master key;
– The weak influence of changes in the master key bits on the bits of the first subkeys;
– Use in the unfolding scheme of another design, different from the cyclic function;
– Different complexity of generating a sequence of subkeys for encryption and decryption.

When designing block algorithms, their structural components, such as *S*-boxes, Boolean functions, and polynomials, play an important role. Depending on their characteristics and properties, block algorithms provide a certain level of security. In [6], the influence of the properties of *S*-boxes on the reliability of algorithms is considered.

The article [7] presents differential cryptanalysis of the Boron block cipher. The authors use differentials consisting of several differential characteristics with the same input and output differences. Each characteristic that corresponds to a given differential increases its overall probability.

By evaluating the strength of round key algorithms, potential loopholes in the overall encryption process can be anticipated. The paper [8] proposes a criterion for estimating the cryptographic strength of round key algorithms. This criterion includes various methods for generating data from subsections and a suitable set of statistical tests.

Quantum cryptography or quantum key distribution is considered one of the information technologies capable of shaping the image of the telecommunications networks of the future. It allows two parties connected over an open communication channel to create a common random key known only to them and use it to encrypt and decrypt messages. The authors of [9] were devoted to ensuring the security of quantum key distribution. The article, which consists of ten chapters, describes almost all practical aspects of security in creating and breaking quantum code. The conclusion includes a systematic and staged presentation of the research results. Technical aspects related to phase synchronization, which provides a high key transfer rate, are studied in [10]. When using single-photon sources in the Z-basis, their protocol provides security up to a higher error rate by establishing non-trivial mutual information between the bit-flip and phase error patterns, thereby achieving a higher key rate. In the history of the development of quantum cryptography, the quantum digital signature also plays an important role. Existing quantum digital signature schemes use non-orthogonal quantum states distributed as “quantum signatures”. Modern schemes allow a message to be securely signed so that one or more recipients can be sure of its authenticity. The use of quantum digital signatures and their exchange protocols is described in more detail in [11,12].

Thus, when developing a symmetric block encryption algorithm, it is necessary to take into account the features of the cryptographic procedures mentioned above. As a rule, when creating a new algorithm or modifying an existing encryption algorithm, its properties are first analyzed.

2. Materials and Methods

When constructing ciphers, the main issue is to ensure their strength. So far, there is no complete theory of block ciphers, which would allow the development of a full-fledged method for evaluating strength. Only some general requirements for the quality of encryption transformations are proposed. If the cipher satisfies such requirements, then we speak of provable security (against known methods of cryptanalysis) or the achievement of guaranteed properties of ciphering transformations.

The main methods for evaluating the strength of cryptographic algorithms include the assessment of their statistical security. Statistical tests are used to experimentally evaluate whether a ciphertext is as good as a random sequence in its properties.

The following test suites are known: NIST, CRYPT-X, DIEHARD, TESTU01, and test sets by Donald Knuth and Ali Doganaksoy. They are used to study the statistical properties of cryptographic primitives and make it possible to obtain a preliminary estimate of cryptographic strength.

To analyze the reliability of block encryption algorithms, powerful attacks such as linear and differential cryptanalysis have been developed [13,14]. Knowledge of these methods of analysis makes it possible, even at the design stage of ciphers, to provide for their excessive security and exclude the possibility of their breaking via these techniques.

The goal of linear cryptanalysis is to find the next "efficient" linear expression for a given encryption algorithm:

$$A_{i_1} \oplus A_{i_2} \oplus \ldots \oplus A_{i_a} \oplus C_{j_1} \oplus C_{j_2} \oplus \ldots \oplus C_{j_b} = K_{k_1} \oplus K_{k_2} \oplus \ldots \oplus K_{k_c}, \quad (1)$$

where $i_1, i_2, \ldots, i_a$, $j_1, j_2, \ldots, j_b$ and $k_1, k_2, \ldots, k_c$ denote fixed bit positions, and Equation (1) is satisfied with probability $p \neq \frac{1}{2}$ for an arbitrarily given plaintext A corresponding to ciphertext C and key K.

Linear cryptanalysis is implemented in two steps. The first is to build relationships between the plaintext, the ciphertext, and the key that are true, with high probability. The second is to use these relationships, along with known plaintext–ciphertext pairs, to derive the key bits. To estimate linear analysis, we use the following two lemmas [15].

Lemma 1. *(Piling-up lemma). Let $X_i (1 \leq i \leq n)$ be independent random variables whose values are 0 with probability p_i or 1 with probability $1 - p_i$. Then, the probability that $X_1 \oplus X_2 \oplus \ldots \oplus X_n = 0$ is $\frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \left(p_i - \frac{1}{2}\right)$.*

Lemma 2. *Let N be the number of given random plaintexts and p be the probability that Equation (1) is satisfied, and let $\left|p - \frac{1}{2}\right|$ be sufficiently small. Then, the probability of success of Algorithm 1 is $\int_{-2\sqrt{N}|p-\frac{1}{2}|}^{\infty} e^{-\frac{x^2}{2}} dx$.*

If it is possible to obtain an effective linear expression, then based on the maximum likelihood method, the following algorithm can determine one bit of the key $K[k_1, k_2, \ldots, k_c]$:

Algorithm 1: Steps to determine one bit of the key $K[k_1, k_2, \ldots, k_c]$

Step 1. Let T be the number of plaintexts, such that the left side of Equation (1) is equal to zero.
Step 2. If T > N/2 (N stands for the number of plaintexts),
then we set $K[k_1, k_2, \ldots, k_c] = 0$ (for $p > 1/2$) or 1 (for $p < 1/2$).
Otherwise, $K[k_1, k_2, \ldots, k_c] = 1$ (for $p > 1/2$) or 0 (for $p < 1/2$).

The probability of success of this method obviously increases with increasing N or $|p - \frac{1}{2}|$. Now, consider the most efficient linear expression (i.e., $|p - \frac{1}{2}|$ is the maximum) as the best expression and the probability p as the best probability. Then, our main task is the following:

1. Methods for finding effective linear expressions.
2. An explicit description of the probability of success in terms of N and p.
3. Finding the best expression and calculating the best probability.

When designing an encryption algorithm, it is also necessary that the cipher satisfy the avalanche criterion. The avalanche effect is an important cryptographic property for encryption. It means that changing the value of a small number of bits in the input text or the key leads to an avalanche change in the values of the output bits of the ciphertext. To characterize the degree of the avalanche effect in a cryptographic transformation, an avalanche parameter is determined and used: the numerical value of the deviation of the probability of a bit change in the output sequence when a bit in the input sequence changes from the required probability value equal to 0.5.

In this work, we will follow the methodology for analyzing statistical security indicators, described in [16]. Here, we consider such indicators of statistical security as:

- Degree of completeness (*dc*);
- Degree of avalanche effect (*da*);
- Degree of strict avalanche criterion (*dsa*).

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

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

$$d_c = 1 - \frac{\#\{(i,j)|a_{ij} = 0\}}{nm}. \quad (2)$$

The degree of the avalanche effect is defined as:

$$d_a = 1 - \frac{\sum_{i=1}^{n} \left| \frac{1}{\#X} \sum_{j=1}^{m} 2jb_{ij} - m \right|}{nm}. \quad (3)$$

The degree of the strict avalanche criterion is defined as:

$$d_{sa} = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \left| \frac{2a_{ij}}{\#X} - 1 \right|}{nm}. \quad (4)$$

For the avalanche criterion, the value of the avalanche parameter is determined by the formula $\mathcal{E} = \max\{|2k_i - 1|, 1 \le i \le n\}$, where i is the number of the changed bit in the input sequence, k_i is the probability of changing half of the bits in the output sequence when the i-th bit at the input changes compared to the original (unchanged) input sequence.

3. Building the AL02 Encryption Algorithm

We have developed the AL02 encryption algorithm, which is new in its architecture and meets modern requirements.

Algorithm parameters: the block length is 128 bits. The number of rounds is five. The data size at the input and output of the *S*-box, which is used in the algorithm as a non-linear function, is 8 bits. The main transformations used are "modulo 2 addition" and *S*-box substitution, as well as the F transformation based on these two transformations.

The scheme of this algorithm is shown in Figure 1.

After the fifth round, the encryption result is passed through the *S*-box, and bitwise addition with the key is performed.

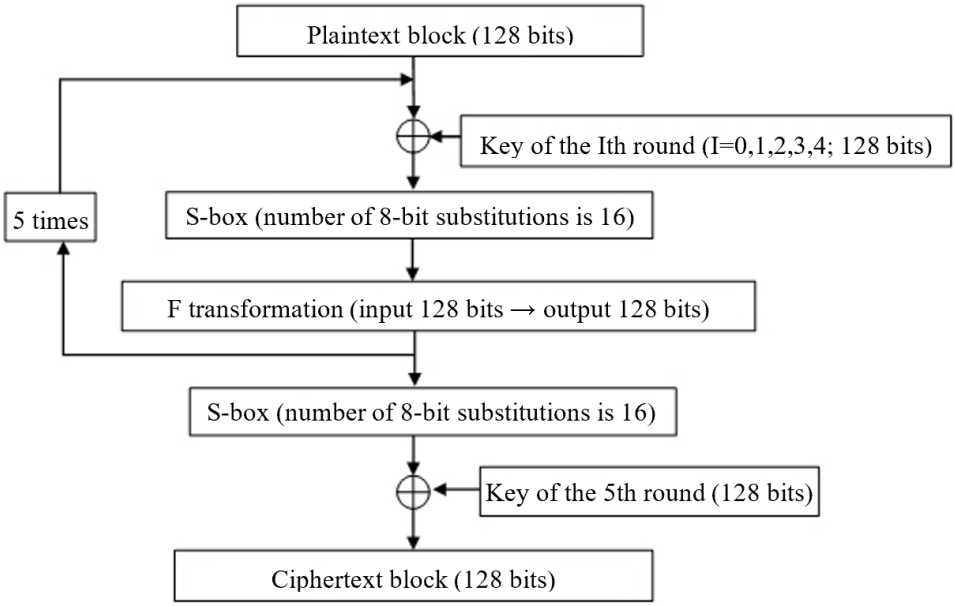

Figure 1. Scheme of the AL02 algorithm.

3.1. Encryption

A 128-bit block is an input as a sequence of 16 bytes:

$$A = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}).$$

This sequence can be represented as a 4×4 square matrix:

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix}. \quad (5)$$

The transformation $F(S(X(K_i,\ I)))$, $i = 0,\ 1,\ 2,\ 3,\ 4$ is repeated five times. In the sixth round, the transformation $X(K_5,\ S(I))$ of the obtained values is performed to get an encrypted block.

Thus, each block is encrypted according to the following scheme:

$$CX(K_5, S(F(S(X(K_4, F(S(X(K_3, F(S(X(K_2, F(S(X(K_1, F(S(X(K_0, A)\ldots))). \quad (6)$$

Here, A is the plain text, C is the ciphertext, X is the XOR operation, S is the S-box transformation, F is a non-linear transformation, and I is an intermediate value.

F transformation of a block. Take four elements with intermediate values corresponding to each row of the matrix and perform a bitwise addition using the following formula:

$$b_i = a_{i,0} \oplus S(a_{i,1}) \oplus S(a_{i,2} \oplus a_{i,3}) = a_{4i} \oplus S(a_{4i+1}) \oplus S(a_{4i+2} \oplus a_{4i+3}), \quad (7)$$

that is, the value of the first element is summed modulo 2 with the value obtained by applying the S-box to the second element and with the value obtained by applying the S-box to the result of bitwise addition of the third and fourth elements.

The resulting four values are added modulo 2, and the sum is converted using the S-box, which is taken as the first element of the first row of the matrix:

$$c_{0,0} = S(b_0 \oplus b_1 \oplus b_2 \oplus b_3). \quad (8)$$

The remaining elements of the new matrix are calculated as follows.

The second, third, and fourth elements of each row are obtained by applying an *S*-box to the result of the bitwise addition of the processed element of the transformed matrix with the previous element of the same row of the transformed matrix:

$$c_{i,j+1} = S\left(c_{i,j} \oplus a_{i,j+1}\right), i = 0,\ 1,\ 2,\ 3; j = 0, 1, 2. \tag{9}$$

To get the first element of a row, the *S*-box is applied to the result of the bitwise addition of the processed element of the transformed matrix and the last element of the previous row of the transformed matrix:

$$c_{i,0} = S(c_{i-1,3} \oplus a_{i,0}), i = 1,\ 2,\ 3. \tag{10}$$

All elements of the transformed matrix are calculated in the same way.

3.2. Decryption

Text decryption is performed by the sequential processing of 16-byte blocks using inverse transformation. Each block is represented as a 16-byte vector or 4×4 square matrix: $C = (c_0, c_1, c_2, c_3, c_4,\ c_5, c_6, c_7, c_8, c_9, c_{10}, c_{11}, c_{12}, c_{13}, c_{14}, c_{15})$, or

$$C = \begin{pmatrix} c_{0,0} & c_{0,1} & c_{0,2} & a_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{pmatrix} = \begin{pmatrix} c_0 & c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 & c_7 \\ c_8 & c_9 & c_{10} & c_{11} \\ c_{12} & c_{13} & c_{14} & c_{15} \end{pmatrix}.$$

To obtain a plaintext block, it is necessary to repeat the inverse transformation $F^{-1}\left(S^{-1}(X(K_i,\ C))\right),\ i = 1,\ 2,\ 3,\ 4,\ 5$ five times and then perform the transformation $X\left(K_0,\ S^{-1}(I)\right)$ using the compositional method.

Each block is decrypted as per the following scheme:

$$A = X\Big(K_0, S^{-1}\Big(F^{-1}\Big(S^{-1}(X(K_1, \ldots, F^{-1}(S^{-1}(X(K_5, C)\ldots\Big)\Big)\Big), \tag{11}$$

where A is the plain text, C is the ciphertext, X is the XOR operation, S^{-1} is the inverse *S*-box, F^{-1} is the inverse non-linear transformation, and I is an intermediate value.

F^{-1} (reverse) block transformation: As a result of applying the inverse transformation to the matrix of ciphertext elements, a new matrix is obtained. In reverse transformation, calculations start from the second element of the first row. The inverse *S*-box is applied to the second, third, and fourth elements of each row of the matrix to be transformed. After that, bitwise addition is performed with the previous element of the same row, resulting in an element of the transformed matrix located in the same position:

$$a_{i,j} = S^{-1}\left(c_{i,j}\right) \oplus c_{i,j-1}, i = 0,\ 1,\ 2,\ 3; j = 1,\ 2,\ 3. \tag{12}$$

The first element of the second, third, and fourth row is calculated by applying the inverse *S*-box to the corresponding element of the matrix to be transformed, and then adding modulo 2 to the last element of the previous row:

$$a_{i,0} = S^{-1}(c_{i,0}) \oplus c_{i-1,3}, i = 1,\ 2,\ 3. \tag{13}$$

B as a result of this transformation, we get all the elements of the resulting matrix, except for the first element in the first row. To get the first element of the first row, the following steps are performed. Starting from the second line, three intermediate values, b_1, b_2, and b_3, are calculated by "modulo 2 addition" of the value of the first element, the value obtained by applying the *S*-box to the second element, and the value obtained by applying the *S*-box to the bit sum of the third and fourth elements:

$$b_i = a_{i,0} \oplus S(a_{i,1}) \oplus S(a_{i,2} \oplus a_{i,3}), i = 1,\ 2,\ 3. \tag{14}$$

The intermediate value b_0 is obtained from the matrix element located at the intersection of the first row and the first column by applying the inverse S-box to it, and bitwise addition with the other three intermediate values:

$$b_0 = S^{-1}(c_{0,0}) \oplus b_1 \oplus b_2 \oplus b_3. \quad (15)$$

To find the first element of the first row of the inverted matrix, the resulting intermediate value b_0 is summed modulo 2 with the two results of applying the S-box, i.e., with the result of applying the S-box to the second element of the first row, and the value obtained as a result of applying the S-box to the bit sum of the second and third elements:

$$a_{0,0} = b_0 \oplus S(a_{0,1}) \oplus S(a_{0,2} \oplus a_{0,3}). \quad (16)$$

To obtain a plaintext block, it is necessary to repeat the inverse transformation five times using the compositional method.

The method of obtaining the S-box is the same as in the Rijndael encryption algorithm, except for the chosen irreducible polynomials and the matrix for multiplications. As a result of the study, we made sure that when choosing any irreducible polynomial, the cryptographic strength of the S-box does not decrease. However, the row elements of the selected matrix must be linearly independent. The irreducible polynomial $z(x)$, the vector v, and the matrix used to construct the S-box are: $z(x) = x^8 + x^7 + x^5 + x + 1$, $v = (110100011)$.

Let us show how to get the output element corresponding to each input element of the S-box:

In the field $GF(2^8)$, we get the inverse value for each element.

Next, the following transformation is performed:

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \\ s'_4 \\ s'_5 \\ s'_6 \\ s'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} + \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix}$$

where s_i, $i = 0, \ldots, 7$ are the coefficients of the obtained inverse elements in the field $GF(2^8)$.

The resulting S-boxes are shown in Tables 1 and 2.

Table 1. Substitution S-box of the AL02 encryption algorithm.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	A5	04	A6	A7	F7	C6	A4	12	5F	C8	C7	D1	F6	D4	7E	7B
1	0B	EF	13	AD	94	5B	4C	8A	0C	FC	CE	1C	9B	76	19	F3
2	21	68	53	96	2D	D0	A1	89	3D	9C	DA	6D	51	AF	E1	E9
3	A2	E3	09	FE	C3	3F	AA	1E	BA	DD	9F	1D	28	54	8E	92
4	E7	D5	43	33	DE	81	3C	97	32	EC	1F	72	74	CD	B3	60
5	3A	95	39	FA	1A	0E	C1	05	DF	CC	A0	8D	87	58	83	D3
6	26	FD	86	7C	20	4B	08	36	45	DC	3B	79	22	BE	AB	14
7	2A	03	99	2C	6B	E5	F9	5C	B0	85	5D	B2	30	80	ED	DB
8	57	8F	9D	A9	D6	B8	EE	24	CB	84	B7	D8	69	A8	6F	50
9	BD	F1	01	38	F8	40	4E	BF	9E	0D	91	C9	7D	F4	47	07
A	B9	63	6E	0F	EB	70	D9	6A	7A	2B	A3	CF	44	65	F5	00
B	98	35	C2	41	27	1B	62	AC	67	23	88	10	B6	8C	4D	C0

Table 1. *Cont.*

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	64	3E	5A	E8	34	D7	9A	16	B4	29	D2	37	73	F2	6C	46
D	06	E6	CA	C4	EA	7F	18	E0	B5	31	FB	FF	71	17	AE	02
E	B1	15	25	78	BB	F0	61	93	11	4F	56	82	8B	42	59	48
F	2F	E2	66	4A	0A	90	2E	75	BC	C5	E4	55	52	77	49	5E

Table 2. Reverse *S*-box of the AL02 encryption algorithm.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	AF	92	DF	71	01	57	D0	9F	66	32	F4	10	18	99	55	AF
1	BB	E8	07	12	6F	E1	C7	DD	D6	1E	54	B5	1B	3B	37	BB
2	64	20	6C	B9	87	E2	60	B4	3C	C9	70	A9	73	24	F6	64
3	7C	D9	48	43	C4	B1	67	CB	93	52	50	6A	46	28	C1	7C
4	95	B3	ED	42	AC	68	CF	9E	EF	FE	F3	65	16	BE	96	95
5	8F	2C	FC	22	3D	FB	EA	80	5D	EE	C2	15	77	7A	FF	8F
6	4F	E6	B6	A1	C0	AD	F2	B8	21	8C	A7	74	CE	2B	A2	4F
7	A5	DC	4B	CC	4C	F7	1D	FD	E3	6B	A8	0F	63	9C	0E	A5
8	7D	45	EB	5E	89	79	62	5C	BA	27	17	EC	BD	5B	3E	7D
9	F5	9A	3F	E7	14	51	23	47	B0	72	C6	1C	29	82	98	F5
A	5A	26	30	AA	06	00	02	03	8D	83	36	6E	B7	13	DE	5A
B	78	E0	7B	4E	C8	D8	BC	8A	85	A0	38	E4	F8	90	6D	78
C	BF	56	B2	34	D3	F9	05	0A	09	9B	D2	88	59	4D	1A	BF
D	25	0B	CA	5F	0D	41	84	C5	8B	A6	2A	7F	69	39	44	25
E	D7	2E	F1	31	FA	75	D1	40	C3	2F	D4	A4	49	7E	86	D7
F	E5	91	CD	1F	9D	AE	0C	04	94	76	53	DA	19	61	33	E5

For convenience, *S*-boxes are presented in tabular form and have the following characteristics: Balance—yes; Nonlinearity—112; Hamming weight—128; t-stability—0, Bijectivity—yes, Correlation value—32.

3.3. Round Key Algorithm

The 128-bit seed key is set randomly. Based on this key, round keys are formed for each iteration of this transformation. Figure 2 shows the scheme for generating round keys, where G is a non-linear transformation.

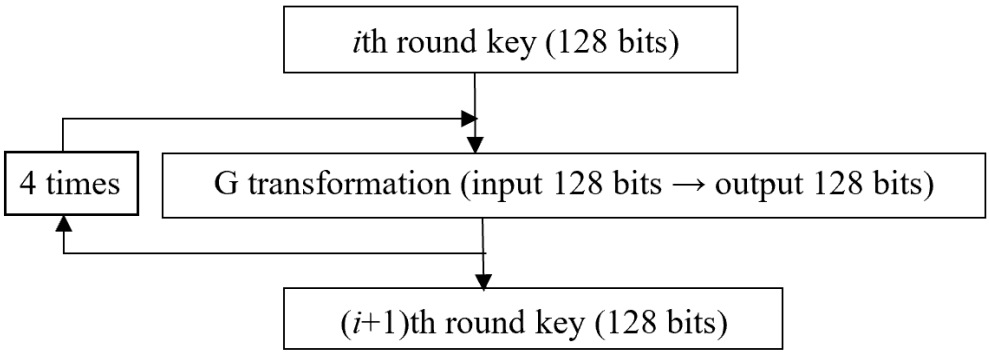

Figure 2. Scheme of the round key algorithm.

G transformation: The value of the initial key is chosen as input for the key transformation. After the input data are defined, we can proceed to the transformation itself.

First, 16 bytes of input data are represented as a 16-byte vector $A = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15})$ or a 4×4 square matrix

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{pmatrix}.$$

The first element of the first row of the matrix is summed as modulo 2 with the value obtained by applying the *S*-box to the second element, and with the result of applying the *S*-box to the sum of the third and fourth elements. The resulting sum is then converted using the *S*-box.

The resulting value b_0 is added to the first element of the second row, then to the value obtained by applying the *S*-box to the second element, and then to the value obtained by applying the *S*-box to the bit sum of the third and fourth elements. The resulting bit sum is also converted using the *S*-box:

$$b_1 = S(b_0 \oplus a_{0,0} \oplus S(a_{0,1}) \oplus S(a_{0,2} \oplus a_{0,3})) = S(b_0 \oplus a_4 \oplus S(a_5) \oplus S(a_6 \oplus a_7)). \quad (17)$$

The same steps are repeated for the third and fourth lines:

$$b_i = S(b_{i-1} \oplus a_{i,0} \oplus S(a_{i,1}) \oplus S(a_{i,2} \oplus a_{i,3})) = S(b_{i-1} \oplus a_{4i} \oplus S(a_{4i+1}) \oplus S(a_{4i+2} \oplus a_{4i+3})), i = 2,\ 3. \quad (18)$$

Next, the byte matrix is represented as a 128-bit sequence and rotated left by 3 bits, and then the 8-bit elements of the resulting sequence are converted using the *S*-box.

This byte matrix transformation can be represented as follows:

$$\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} \text{ rotate left by three bits and}$$

$$\text{apply } S - \text{box} \begin{pmatrix} a'_{0,0} & a'_{0,1} & a'_{0,2} & a'_{0,3} \\ a'_{1,0} & a'_{1,1} & a'_{1,2} & a'_{1,3} \\ a'_{2,0} & a'_{2,1} & a'_{2,2} & a'_{2,3} \\ a'_{3,0} & a'_{3,1} & a'_{3,2} & a'_{3,3} \end{pmatrix}.$$

where the elements $a'_{i,j}$ of the transformed matrix are obtained as follows:

$$a'_{i,j} = S\Big(\Big(\Big(a_{i,j}(x) \times x^3\Big) mod x^8\Big) \oplus \Big(\Big(a_{i,(j+1)}(x) \times x^3\Big) mod x^8\Big) \oplus \Big(\Big(a_{i,(j+1)}(x) \times x^3\Big) mod (x^8 + 1)\Big)\Big), \quad (19)$$

where, $i = 0,\ 1,\ 2,\ 3;\ j = 0,\ 1,\ 2$.

$$a'_{i,3} = S\Big(\big((a_{i,3}(x) \times x^3) mod x^8\big) \oplus \Big(\Big(a_{(i+1) mod 4,0}(x) \times x^3\Big) mod x^8\Big) \oplus \Big(\Big(a_{(i+1) mod 4,0}(x) \times x^3\Big) mod (x^8 + 1)\Big)\Big),\ i = 0,\ 1,\ 2,\ 3.$$

As the first element of the first row of the matrix after the key transformation, the value obtained as a result of applying the *S*-box to the bit sum of $b3$ and $a'_{0,0}$ is taken:

$$c_{0,0} = S(b_3 \oplus a). \quad (20)$$

The values of the remaining elements of the transformed matrix are obtained by applying the *S*-box to the sum of the previous element of the transformed matrix with the corresponding element of the transformed matrix:

$$c_{i,j+1} = S\big(c_{i,j} \oplus a_{i,j+1}\big); i = 0,\ 1,\ 2,\ 3; j = 0,\ 1,\ 2. \quad (21)$$

To obtain the first elements of the remaining rows of the transformed matrix, it is necessary to calculate the value obtained as a result of applying the *S*-box to the bit sum

of the last element of the previous row of the transformed matrix with the corresponding element of the transformed matrix:

$$c_{i,0} = S(c_{i-1,3} \oplus a_{i,0}), i = 0,\ 1,\ 2,\ 3. \quad (22)$$

In this way, all elements of the transformed matrix are calculated. In the case of the key generation algorithm, the key transformation is repeated five times using the compositional method.

When generating round keys, the matrix obtained as a result of a subsequent step of the key transformation is taken as the next round key.

To determine the number of rounds providing non-linearity, each round of the algorithm was considered to evaluate the algebraic normal forms (ANF) of the used permutation functions in the initial 128-bit key space. Since the round keys depend on the initial keys, in each round, the output bit of the ANF is evaluated by the monomial (single term) and the degree of the ANF. To ensure good statistical performance, each ANF should consist of approximately 2^{127} monomials and have a degree of 127. With 128 variables in the ANF, the maximum number of monomials is 2^{128} and the exponent is 128. It was found that in the considered algorithm, after the third round, the output bit in the ANF equation has 2^{127} monomials and a degree of 127. As the number of rounds increases, the number of monomials increases correspondingly. It turned out that five rounds are sufficient for the algorithm.

The sufficiency of five rounds was assessed using statistical tests and an avalanche test (Table 3 and Figure 3) for one file.

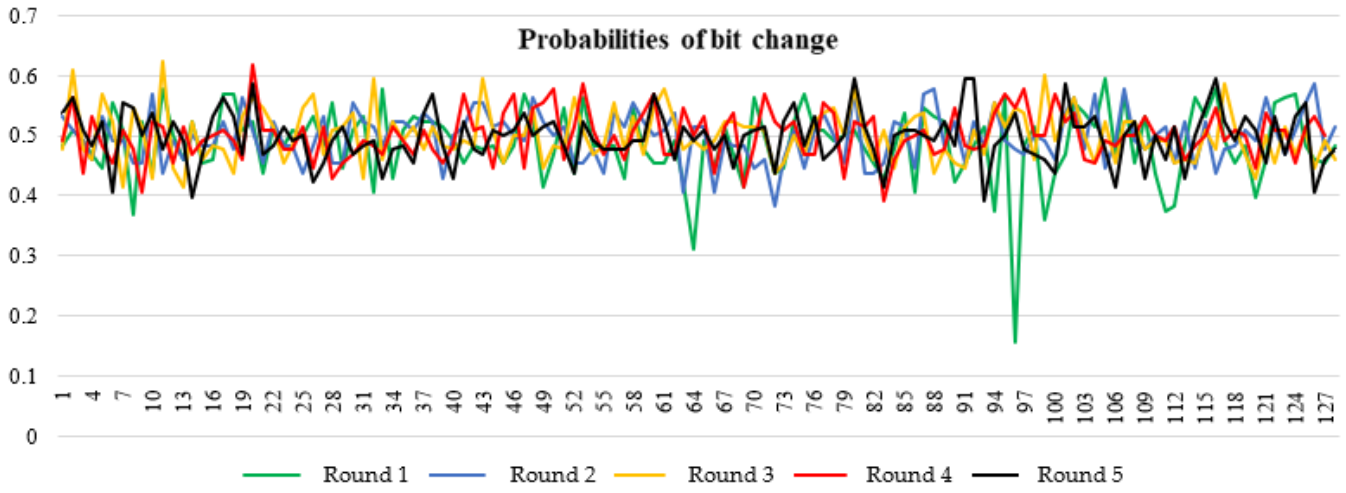

Figure 3. Results of the avalanche effect by rounds.

Table 3. Results of the conducted statistical tests by rounds.

Tests	Round 1	Round 2	Round 3	Round 4	Round 5
Frequency Test	+	-	+	+	+
Frequency Test within a Block	+	+	+	+	+
Run Test	+	+	+	+	+
Longest Run of Ones in a Block	+	+	+	+	+
Binary Matrix Rank Test	+	+	+	+	+
Discrete Fourier Transform	-	+	+	+	+
Non-Overlapping Template Matching Test	+	+	+	+	+
Overlapping Template Matching Test	+	+	+	+	+
Maurer's Universal Statistical test	+	-	+	+	+
Linear Complexity Test	+	+	+	+	+
Serial Test	-	+	+	+	+
Approximate Entropy Test	-	+	+	+	+

Table 3. *Cont.*

Tests	Round 1	Round 2	Round 3	Round 4	Round 5
Cumulative Sums (Forward) Test	+	-	+	+	+
Cumulative Sums (Reverse) Test	+	-	+	+	+
Random Excursions Test	-	-	+	+	+

The analysis results shown in Table 3 show that the AL02 algorithm passes all tests after the third round. Figure 3 shows data on the "avalanche effect" property with positive results after the second round of encryption. This confirms the adequacy of the established number of rounds of the algorithm.

Table 4 shows an example of encryption for the AL02 algorithm.

Table 4. Encryption example for the AL02 algorithm.

	Values in Hex Format
Key	a3 13 ab e8 1d 5f b4 77 1d 59 72 0b a8 41 a5 ce
1 round key	f2 ca bb d6 fd 87 c4 2b e9 fe 25 6f 11 fb 29 62
2 round key	30 16 22 d3 48 93 63 55 9e d6 87 14 d8 e0 60 41
3 round key	3d 55 1a e0 c4 52 2c b4 a1 ee c8 fa c7 1f 55 44
4 round key	cc 5d 7d 24 9b c1 e5 51 bb 17 cc c1 a6 37 0 2d
5 round key	35 a1 7 ba 15 ae 5a f3 e4 a6 b6 8a 25 d9 74 8f
Plain text	53 6a 23 34 87 e2 42 d0 ff 31 69 e3 b8 be d3 aa
Results of the 1st round encryption	2f a3 45 c3 39 1b 3f e 6d 3d 68 85 6f e3 ce 59
Results of the 2nd round encryption	a7 b2 55 7e 1f 86 c4 94 43 cf a6 2f 5a c1 39 2d
Results of the 3rd round encryption	fa ef 41 2d 3d 8 86 67 2a 18 2a 1e a9 3e 66 de
Results of the 4th round of encryption	b4 28 70 9d df db b0 6b 41 14 e3 df 6b a7 cc 58
Results of the 5th round of encryption	93 f9 af 69 ac 86 cb 1a 49 59 98 8f 80 54 89 22
Ciphertext	0d 64 7 66 51 40 6d 3d 8 6a 28 da 72 c3 f0 dc

Table 5 shows the experimental results of the performance characteristics of different encryption algorithms. The experiment was conducted on a computer with the following parameters: Processor: Intel(R) Core(TM) i7-10700 CPU @ 2.90 GHz 2.90 GHz, RAM: 16.0 GB (available: 15.7 GB), System type: 64-bit operating system Windows 10 Pro.

Table 5. Comparative performance characteristics of encryption algorithms.

No	Name of the Encryption Algorithm	Data Processing Speed	Software Developer
1.	BLOWFISH-128	51.68 MB/s	GNU Project (Libgcrypt Library)
2.	AL02	51.64 MB/s	Institute of Information and Computational Technologies CS MES RK
3.	CAST-128	49.04 MB/s	GNU Project (Libgcrypt Library)
4.	IDEA	46.28 MB/s	GNU Project (Libgcrypt Library)
5.	3DES	26.85 MB/s	GNU Project (Libgcrypt Library)
6.	GOST 34.12-2015	6.89 MB/s	TK-26 (reference implementation)

4. Cryptographic Security Analysis of the Developed Encryption Algorithm

4.1. Statistical Analysis

For a computer study of the developed algorithm, 20 files with different extensions and 5 keys were used. The file characteristics were as follows: 1.docx (108 KB), 2.xls (20.0 KB), 3.pptx (453 KB), 4.pdf (561 KB), 5.rar (1059 KB), 6.zip (1063 KB), 7.jpg (205 KB), 8.png (889 KB), 9.txt (45 KB), 10.html (281 KB), 11.html (17 KB), 12.cat (928 KB), 13.mp4 (39 KB), 14.wmz (70 KB), 15.dll (210 KB), 16.log (508 KB), 17.lex (396 KB), 18.djvu (354 KB), 19. xml (689 KB), and 20 mp3 (193 KB).

Using the selected keys and plaintexts for the developed AL02 symmetric block encryption algorithm, 100 ciphertexts were obtained, which were tested for statistical

security. For testing, a developed software package was used, which implements a system for assessing the quality of ciphertexts according to tests from the sets by D. Knuth and NIST-822 STS.

The results of statistical tests for different keys with a probability value of 0.5 and with a probability value of α are shown in Tables 6 and 7.

Table 6. The results of the statistical tests for different keys with a probability value of 0.5.

Statistical Tests	Number of Files	Key 1	Key 2	Key 3	Key 4	Key 5
Frequency test	20	18	19	19	19	19
Serial test	20	19	20	18	19	17
Character test	20	19	18	20	19	20
Gap test	20	18	17	17	17	16
Poker test	20	20	20	20	20	19
Coupon collector's test	20	17	20	19	18	20
Permutation test	20	18	18	18	17	20
Monotonicity test	20	18; 19	19; 19	19; 20	19; 19	18; 18
Correlation test	20	20	20	20	20	20
Linear complexity test	20	20	20	20	20	20

Table 7. The results of the statistical tests with a probability value of α.

Statistical Tests	Number of Files	Probability α		
		0.05	0.01	0.001
Frequency test	100	94	98	100
Serial test	100	93	99	100
Character test	100	96	100	100
Gap test	100	85	95	97
Poker test	100	99	100	100
Coupon collector's test	100	94	99	100
Permutation test	100	93	97	100
Monotonicity test	100	95	97	100
Correlation test	100	100	100	100
Linear complexity test	100	100	100	100

The number of sequences that successfully passed each of the statistical tests was determined. The significance level α was set as 0.001, 0.01, and 0.05.

The statistical parameters of the ciphertexts were evaluated using the NIST-822 STS package (Table 8).

Table 8. Results of statistical tests from the NIST-822 STS set.

Tests	Key1	Key2	Key3	Key4	Key5	Results
Frequency Test	20	20	19	20	20	99%
Frequency Test within a Block	20	20	20	20	19	99%
Runs Test	20	20	20	20	20	100%
Tests for the Longest-Run-of-Ones in a Block	20	20	20	20	20	100%
Binary Matrix Rank Test	20	20	20	20	20	100%
Discrete Fourier Transform	20	20	19	20	20	99%
Non-Overlapping Template Match	20	19	20	20	19	98%
Overlapping Template Matching Test	20	20	20	20	20	100%
Maurer's "Universal Statistical" Test	15	15	16	16	16	91%
Linear Complexity Test	20	20	20	19	20	99%
Serial Test	20	20	20	20	20	100%
Approximate Entropy Test	18	20	20	20	20	98%
Cumulative Sums (Forward) Test	20	20	19	20	19	98%

Table 8. *Cont.*

Tests	Key1	Key2	Key3	Key4	Key5	Results
Cumulative Sums (Reverse) Test	20	20	18	20	20	98%
Random Excursions Test	16	19	17	19	16	93%
Random Excursions Variant Test	18	17	18	19	20	92%

For convenience, the experimental verification of the files under study in the NIST-822 CTC environment is presented in the form of a table, the last column of which contains the percentages of the tests passed.

Ciphers with good statistical security indicators should have d_c, d_a, and d_{sa} values satisfying the following conditions: $d_c \approx 1$, $d_a \approx 1$, $d_{sa} \approx 1$ [16]. Table 9 shows the comparative indicators of the statistical security of the algorithms AL02 and AES128.

Table 9. Single-round values of indicators of statistical security of the ciphers AES128 and AL02.

Round No	AES128			AL02		
	d_c	d_a	d_{sa}	d_c	d_a	d_{sa}
1	0.0625	0.061535	0.059118	1	0.995010	0.991544
2	0.25	0.251176	0.247659	1	0.999642	0.992065
3	1	0.995746	0.991457	1	0.999212	0.992005
4	1	0.999514	0.992006	1	0.999304	0.992048
5	1	0.999773	0.992022	1	0.999257	0.992001
6	1	0.999712	0.992015			
7	1	0.998668	0.992003			
8	1	0.999176	0.992065			
9	1	0.998481	0.992030			
10	1	0.999595	0.991962			

4.2. Avalanche Effect

The avalanche effect was tested in two versions. In the first version, 129 blocks of plain texts were used, which differed from each other by one bit. These files were encrypted with the AL02 algorithm using the same key. Then, the encrypted files were checked for a match by the location of the bits. In the second version, the check was carried out for 129 different keys and with one plaintext, where the keys were selected in such a way that they differed from each other by only one bit. The resulting ciphertexts were also checked for matches in terms of bit arrangement. The test results are shown in Figure 4, where the blue line shows the results of the first option, and the orange line shows the results of the second option.

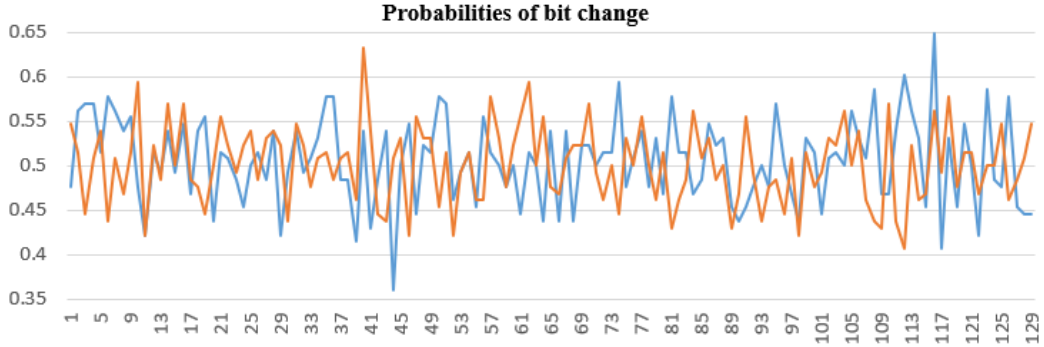

Figure 4. Results of testing the avalanche effect.

In the first test, the chi-square value for a degree of freedom of 127 is 62.203, and in the second test, the chi-square value is 77.313.

4.3. Differential Analysis of the AL02 Algorithm

For the differential analysis of the AL02 algorithm, the difference values for each function of a single round are first estimated. The AL02 algorithm consists of an XOR operation, a non-linear transformation using an *S*-box, and the F-function.

The XOR operation is linear, so the difference has the same value for both the input and the output, so the probability is one. The maximum substitution difference using the *S*-box is 4, so in this case, the probability will be 2–6.

For the F function of the AL02 algorithm, the finding of the round keys with respect to pairs of ciphertexts and the correspondence with the chosen pair of plaintexts are evaluated.

$$A = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}, \; A' = \begin{pmatrix} a_{0,0}+1 & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0}+1 & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0}+1 & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0}+1 & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

Differences between selected plaintexts A and A' are presented as a matrix Δ_A:

$$\Delta_A = A \oplus A' = \begin{pmatrix} 1, & 0, & 0, & 0 \\ 1, & 0, & 0, & 0 \\ 1, & 0, & 0, & 0 \\ 1, & 0, & 0, & 0 \end{pmatrix}.$$

Intermediate values bi and b'_i, $i = 0,1,2,3$ for plaintexts are $b_i = a_{i,0} \oplus S(a_{i,1}) \oplus S(a_{i,2} \oplus a_{i,3})$ and $b'_i = a_{i,0} \oplus 1 \oplus S(a_{i,1}) \oplus S(a_{i,2} \oplus a_{i,3})$. The first element of the first row of the new matrix is: $c_{0,0} = S(b_0 \oplus b_1 \oplus b_2 \oplus b_3) = c'_{0,0} = S\left(\oplus_{i=0}^{3} b'_i\right) = S\left(\oplus_{i=0}^{3} b_i\right)$.

This implies $c_{0,0} \oplus c'_{0,0} = 0$, $c_{0,1} = S(c_{0,0} \oplus a_{0,1}) = c'_{0,1} = S(c_{0,0} \oplus a_{0,1})$, $c_{0,2} = S(c_{0,1} \oplus a_{0,2}) = c'_{0,2} = S(c_{0,1} \oplus a_{0,2})$, $c_{0,3} = c'_{0,3}$ since the first row of the difference of the new matrices is completely equal to zero. The second row changes depending on the key or plaintext. If one bit of the key or plaintext has been changed, then completely different values are obtained in the second row.

As a result of the differential analysis of the F-function, the maximum probability is 2–12 and the minimum probability is 2–60, so the input of the function is selected in such a way as to minimize the use of the *S*-box. In this case, the minimum *S*-box usage is 2 (the maximum *S*-box usage is 10).

Differential analysis was carried out to obtain the upper bound for each round for the selected pairs of open and closed texts (Table 10).

Table 10. Differential analysis of the AL02 algorithm.

Number of Rounds	Number of *S*-Boxes Used	Probability of Finding Round Keys	Key Finding Probability
1	3	$2^{-6} \times 2^{-6} \times 2^{-6} = 2^{-18}$	2^{-18}
2	10	$2^{-6} \times 2^{-12} \times 2^{-12} \times 2^{-6} \times 2^{-6} = 2^{-42}$	$2^{-18} \times 2^{-42} = 2^{-60}$
3	17	$2^{-6} \times 2^{-12} \times 2^{-12} \times 2^{-6}\, 2^{-6} = 2^{-42}$	$2^{-60} \times 2^{-42} = 2^{-102}$
4	24	$2^{-6} \times 2^{-12} \times 2^{-12} \times 2^{-6}\, 2^{-6} = 2^{-42}$	$2^{-102} \times 2^{-42} = 2^{-144}$
5	31	$2^{-6} \times 2^{-12} \times 2^{-12} \times 2^{-6}\, 2^{-6} = 2^{-42}$	$2^{-144} \times 2^{-42} = 2^{-186}$

This algorithm does not allow us to keep the minimum difference at the input of any of the rounds [17,18].

4.4. Linear Analysis

Before performing a linear analysis, the non-linear functions of the algorithm are first examined separately. In the linear analysis table, the considered *S*-box has significant deviations from the linear approximation, which is equal to 16. The maximum probability of deviation is 0.5625. The overall linear analysis bound is determined by the upper value. Consider each transformation applied in the encryption algorithm. The cipher uses bitwise

addition (XOR), substitution (*S*-box), and cyclic shifts. The bitwise addition operation and cyclic shifts are linear functions.

Lemmas 1 and 2 show that the probability of finding linear equations in one round is $0.5 + 2^9(0.5 - 0.5625)^{10} = 0.5 + 2^9\left(2^{-4}\right)^{10} = 0.5 + 2^{-31}$, since the corresponding obtained equations of one round of input and output variables of the linear approximation pass through the *S*-box 10 times. Hence, to find a key with a probability of 0.97, 2^{62} pairs of plaintexts and ciphertexts are required (Table 11). For the full five rounds, the probability of finding linear equations given 40 *S*-boxes is $0.5 + 2^{49}(0.5 - 0.5625)^{50} = 0.5 + 2^{49}\left(2^{-4}\right)^{50} = 0.5 + 2^{-151}$.

Table 11. Linear analysis of the AL02 algorithm.

Number of Rounds	Number of *S*-Boxes Used	Truth Probability of Equations for Pairs of Plaintexts and Ciphertexts	Number of Pairs of Plaintexts and Ciphertexts
1	10	2^{-31}	2^{62}
2	20	2^{-61}	2^{122}
3	30	2^{-91}	2^{182}
4	40	2^{-121}	2^{242}
5	50	2^{-151}	2^{302}

Therefore, to determine the key with a probability of 0.97, 2302 pairs of plaintexts and ciphertexts are required.

5. Conclusions

Symmetric cryptographic primitives are widely used due to their high performance and low implementation complexity. Such algorithms use non-linear operations for confusion and linear transformations for diffusion. Consistent repeated application of confusion and diffusion makes it possible to achieve a high level of cryptographic strength. We have developed a new symmetric block cipher algorithm AL02, the architecture of which includes non-linear substitution nodes in the form of substitution tables, the F function, and the round key scheduling algorithm. The advantage of the developed algorithm is the possibility of its effective implementation on specialized equipment designed to perform encryption and decryption operations. Theoretical and experimental tests have shown that the algorithm fully complies with the basic cryptographic requirements. The study of the cryptographic strength of the encryption algorithm will be continued in subsequent works.

The developed algorithm was investigated for reliability using well-known cryptanalysis methods. It was shown that the parameters correspond to the requirements imposed on them.

The algorithm was tested for randomness using statistical tests from the NIST and D. Knuth sets. Based on the study results, it was found that the binary sequence obtained after encryption by the proposed algorithm is close to random. Additionally, the *S*-box and the AL02 encryption algorithm were tested for the presence of an avalanche effect. Based on the tests and studies carried out, it has been established that the proposed *S*-box and the AL02 encryption algorithm itself effectively provide a good avalanche effect.

It was also shown that the necessary level of resistance of the developed encryption algorithm to linear and differential cryptanalysis is provided. Research is ongoing to test the reliability of the algorithm against other existing cryptographic attacks.

Author Contributions: Conceptualization, N.K. and D.D.; data curation, N.K.; formal analysis, K.S. and K.A.; funding acquisition, N.K.; investigation, D.D.; methodology, K.A.; project administration N.K.; resources, K.A.; software, K.S.; supervision, K.S.; validation, N.K., K.S. and K.A.; visualization, N.K.; writing—original draft preparation, N.K.; writing—review and editing, D.D. All authors have read and agreed to the published version of the manuscript.

Funding: The research work was funded by the Ministry of Science and Higher Education of Kazakhstan and carried out within the framework of the project AP14870719 "Development and study of post-quantum cryptography algorithms based on hash functions" at the Institute of Information and Computational Technologies.

Data Availability Statement: Not applicable.

Acknowledgments: The authors are grateful to all lab members of "Information security laboratory" (Institute of Information and Computational Technologies) for their useful suggestions and support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Haitner, I.; Vadhan, S. The Many Entropies in One-Way Functions. In *Tutorials on the Foundations of Cryptography. Information Security and Cryptography*; Lindell, Y., Ed.; Springer: Cham, Switzerland, 2017; pp. 159–217. [CrossRef]
2. Panasenko, S.P. *Encryption Algorithms*; Special Reference-Book; S-P, BHV-Petersburg: Saint Petersburg, Russia, 2009; p. 576.
3. Zhang, W.; Cao, M.; Guo, J.; Pasalic, E. Improved Security Evaluation of SPN Block Ciphers and its Applications in the Single-key Attack on SKINNY. *IACR Trans. Symmetric Cryptol.* **2019**, *4*, 171–191. [CrossRef]
4. Kapalova, N.; Haumen, A. The model of encryption algorithm based on non-positional polynomial notations and constructed on an SP network. *Open Eng.* **2018**, *1*, 140–146. [CrossRef]
5. Bondarenko, A.; Marshalko, G.; Shishkin, V. GOST R 34.12–2015: What to expect from the new standard? *Inf. Secur.* **2015**, *4*, 48–50.
6. Dragomir, I.; Măluţan, S.; Lazar, M. An analysis of cryptographic algorithm strength based on *S*-box properties. In Proceedings of the 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 27–29 June 2019; pp. 1–4.
7. Teh, J.S.; Tham, L.J.; Norziana Jamil, N.; Yap, W. New differential cryptanalysis results for the lightweight block cipher BORON. *J. Inf. Secur. Appl.* **2022**, *66*, 103129. [CrossRef]
8. Afzal, S.; Yousaf, M.; Afzal, H.; Alharbe, N.; Rafiq, M. Mufti Cryptographic Strength Evaluation of Key Schedule Algorithms. *Secur. Commun. Netw.* **2020**, *5*, 3189601.
9. Xu, F.; Ma, X.; Zhang, Q.; Lo, H.-K.; Pan, J.-W. Secure quantum key distribution with realistic devices. *Rev. Mod. Phys.* **2020**, *92*, 025002. [CrossRef]
10. Xie, Y.; Lu, Y.; Weng, C.; Cao, X.; Jia, Z.; Bao, Y.; Wang, Y.; Fu, Y.; Yin, H.; Chen, Z. Breaking the Rate-Loss Relationship of Quantum Key Distribution with Asynchronous Two-Photon Interference. *arXiv* **2021**, arXiv:2112.11635. [CrossRef]
11. Yin, H.; Fu, Y.; Chen, Z. Practical quantum digital signature. *Phys. Rev. A* **2016**, *93*, 032316. [CrossRef]
12. Lu, Y.S.; Cao, X.Y.; Weng, C.X.; Gu, J.; Xie, Y.M.; Zhou, M.G.; Yin, H.L.; Chen, Z.B. Efficient quantum digital signatures without symmetrization step. *Opt. Express* **2021**, *29*, 10162–10171. [CrossRef] [PubMed]
13. Babenko, L.K.; Ischukova, E.A. *Modern Block Cipher Algorithms and Methods for Their Analysis*; Helios ARV: Moscow, Russia, 2006; p. 376. Available online: http://www.tnu.in.ua/study/books/entry-1782833.html (accessed on 5 June 2022).
14. Vergili, I.; Yücel, M.D. Avalanche and Bit Independence Properties for the Ensembles of Randomly Cho-sen *S*-Boxes. *Turk. J. Electr. Eng.* **2001**, *2*, 137–145.
15. Matsui, M. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology*; EUROCRYPT '93, LNCS; Helleseth, T., Ed.; Springer: Berlin/Heidelberg, Germany, 23–27 May 1993; pp. 386–397. Available online: https://link.springer.com/content/pdf/10.1007/3-540-48285-7_33.pdf (accessed on 10 September 2022).
16. Pascale, S. *The Degrees of Completeness, of Avalanche Effect, and of Strict Avalanche Criterion for MARS, RC6, Rijndael, Serpent, and Twofish with Reduced Number of Rounds*; Siemens AG, ZT IK 3; KU Leuven, ESAT/COSIC: Leuven, Belgium, 3 April 2000. Available online: https://www.cosic.esat.kuleuven.be/nessie/reports/phase1/sagwp3-003.pdf (accessed on 3 September 2022).
17. Dyusenbaev, D.S.; Algazy, K.T.; Sakan, K.S. Study of nonlinear knots used in symmetric ciphers. In Proceedings of the Actual Problems of Information Security in Kazakhstan APIBK-2020, Almaty, Kazakhstan, 11 June 2021; pp. 34–39. Available online: https://conf.iict.kz/wp-content/uploads/2021/06/collection-apibk2021.pdf (accessed on 3 September 2022).
18. Algazy, K.T.; Babenko, L.K.; Biyashev, R.G.; Ishchukova, E.A.; Kapalova, N.A.; Nyssanbaeva, S.E.; Smolarz, A. Differential Cryptanalysis of New Qamal Encryption Algorithm. *Int. J. Electron. Telecommun.* **2019**, *66*, 647–653. [CrossRef]