Using component-wise functions in cryptographical transformation algorithm from Russian National Standard GOST R 34.12-2015

I I Vasilishin[1] and S Y Korabelshchikova[1]

[1]Northern (Arctic) Federal University named after M.V. Lomonosov, Severnaya Dvina Emb. 17, Arkhangelsk, Russia, 163007

Abstract. The paper presents the general approach to selecting functions, keeping the first argument field, in the process of symmetric encryption of a plaintext. Quantitative estimation and general characteristics of ordinate vector for such functions are given. Ten component-wise functions of binary logic algebra of three arguments, replacing one function of digit-wise addition modulo two in the cryptographictransformation algorithm from Russian National Standard GOST R 34.12-2015, are presented in the paper. Using component-wise functions widens the range of intermediate options of round transformations in block encryption, which complicates the decryption (cracking) algorithm for the cipher.

1. Introduction

The Russian National Standard GOST R 34.12-2015[1]is a symmetric cipher in which plaintext is converted by blocks of fixed length of 128 or 64 digits and a 256-digitlong key, and the encryption/decryption algorithms are reverse procedures using multi-round operations of substitutions and transformations the sequence of which isdeveloped by bitwise operation "addition modulo two" (AMT) for the source text and the first iterative key. Consequently, the multi-variant substitution of the AMT operation proposed by the authors leads to a greater variety of intermediate variants of the bit addition, which changes the results of the final operations, and in general complicates the algorithm of decryption (hacking) of the ciphertext.

2. General analysis of the existing algorithm

Cryptographic transformation of information used in GOST R 34.12-2015 is based on the principles of block data encryption [2, 3, 4] and contains a combination of operators that ensures the implementation of the properties of symmetric encryption in the standard [1]:

- *bitwise addition is formed by the AMT operation on the current transformation (a) and the round key (k), which corresponds to the bit transformation* $X[k]$: $V_{128} \rightarrow V_{128}$, where the result determines the equality of blocks before and after the AMT operation and is defined by the formula

$$X[k](a) = k \oplus a, \quad (1)$$

where: $k, a \in V_{128}$;

- *mixing information* is formed with a nonlinear bijective transformation over replacement block S performing both byte-substitution operation ($a_{15}, \ldots, a_0$) of128-bit value S: $V_{128} \rightarrow V_{128}$, where the value a_i

determines the index of the replacement array π, and the result $V_{128} \rightarrow V_{128}$ determines the equality of blocks before and after the replacement, and the formation of analytical complexity of dependencies between the key and the encrypted text, and is provided by the conversion

$$S(a) = S(a_{15}||...||a_0) = \pi(a_{15})||...||\pi(a_0), \quad (2)$$

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

- *dispersion of information* is achieved both by making nine-round consecutive embedding *F(a)* for sixteen times byte-wise conversion in each round *L(a)* over a 128-bit value of the replacement block *S(π(a))*, and by spreading of influence of each symbol in the plaintext to all the characters of the ciphertext, which is provided with transformations

$$F[k](a_1, a_0) = LSX[k](a_1) \oplus a_0, a_1, \quad (3)$$

where: $L(a) = R^{16}(a)$; $R(a) = R(a_{15}||...||a_0) = \ell(a_{15}, \ldots, a_0)||\, a_{15}||\ldots||\, a_1$; $k, a_i \in V_{128}$.

Generalizing the transformations given in equations (1), (2) and (3), let us form a complete encoding algorithm $E_{Ki}(a)$ that performs transformations of 128-bit source block of information, where the direct numbering substitution of iterative keys is used

$$E_{K1, \ldots, K10}(a) = (k_1 \oplus a)F(a)(k_2 \oplus a)F(a)\ldots(k_9 \oplus a)F(a)(k_{10} \oplus a). \quad (4)$$

It should be noted that the $D_{Ki}(a)$ full information decryption algorithm uses inverse transformations $S^{-1}(a)$, $R^{-1}(a)$ and $L^{-1}(a)$, and the numbering of iterative key substitution is done in the reverse order

$$D_{K10, \ldots, K1}(a) = (k_{10} \oplus a)F^{-1}(a)(k_9 \oplus a)F^{-1}(a)\ldots(k_2 \oplus a)F^{-1}(a)(k_1 \oplus a). \quad (5)$$

In accordance with the transformations given in equations (1) – (5), the complete algorithmic sequence of information encryption/decryption contains both repetitive AMT operation over the iterative key (k_i) and the current transformation *(a)* and round wise mixing and dispersion of information for direct substitutions $S(a)$, $R(a)$, $L(a)$ and inverse substitutions $S^{-1}(a)$, $R^{-1}(a)$, $L^{-1}(a)$. Consequently, the entire periodic process of encryption / decryption of information should be divided, for the current presentation, into two groups of operations: AMT and substitutions.

3. Introduction of component-wise functions

The bitwise operation AMT used in cryptographic information conversion algorithms D(a) belongs to the functions of Boolean algebra [3], is generated by a combination of logical values of two variables and is oneof the operations that has the property of "restoring the input value of one of the variables" in sequential application of the operation in the process of encryption, and then in the process of decryption

$$D(a) = (k \oplus a) \oplus k, \quad (6)$$

where k is an iterative key and a is information for encryption.

Taking into account that the AMT operation belongs to the functions of Boolean algebra of two variables, we would like to point out that other functions generated by combinations of logical values of three [5], four or more variables possess the same property, but the generated functions do not perform the "classical" AMT operation. Therefore, let us present a new function for restoration of the input values, as an operation equivalent to the "classical" AMT operation and call it by a new termcomponent-wise function *M(a)*.

Let us present (see figure) the location of the component functions$M_j(a)$ in the process of encrypting 128-bit block of information in the form of an algorithm based on the transformation (4), for the standard transformation and the oneproposed by the authors.

As shown in the figure, the differences in the authors' encryption algorithm component-wise functions $M_j(a)$are used, replacing the only AMT operation used in the transformation (4), as well as the multiple use of iterative keys, while the production and the number of iterative keys remains the same as the standard and equals ten. It should be noted that the decryption process remains similar to the encryption process, except for changing the order numbers of the iterative keys to the opposite, similar to the transformation (5).

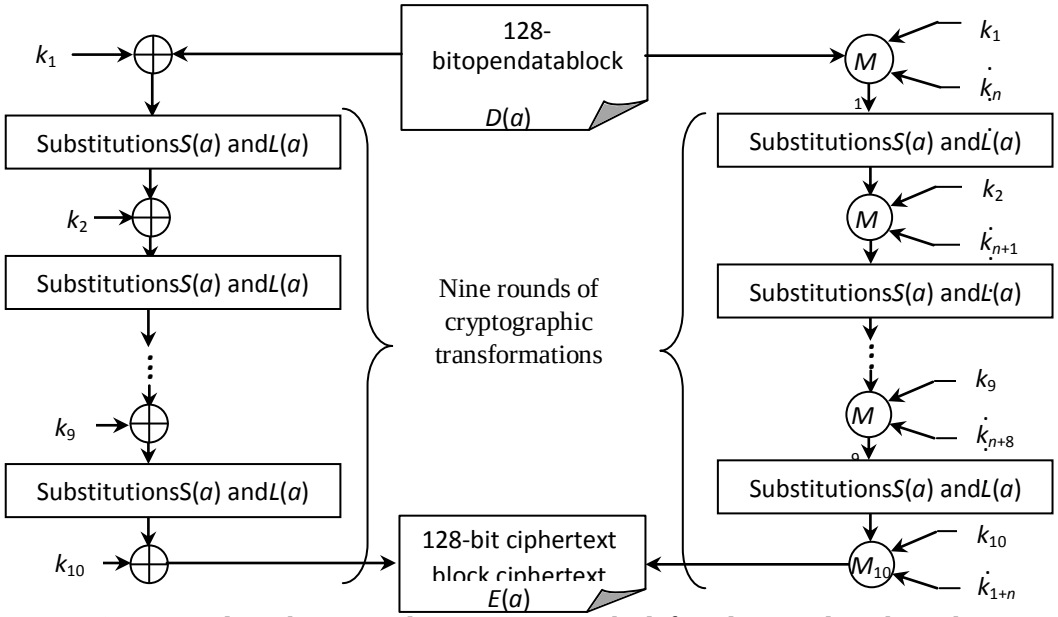

Figure 1. Algorithms encoding: existing on the left and original on the right.

4. The formation of the component-wise functions

The essence of the forming component functions consists in theoretical determination of the characteristics of functions that have the property of restoring the input value for one of the arguments when performing only the logical operation of the function over the operations of forward and inverse transformation, similar to the use of the AMT operation in equation (1). Further, the symbol "◈" defines a logical operation of the component function.

In the theory of abstract algebra, the proofs of existence of Boolean algebras for any number of variables are presented, indexing of Boolean functions is introduced, and belonging of the set of indexed functions of Boolean algebras to systems of normal forms is established [6]. Using the terminology of [5], we will present the principles of formation of the component functions in the form of statements, which are valid for any number of variables. For certainty we use component-wise functions of three variables.

Statement 1. On the distribution of meaning of variables of potentially suitable component-wise functions.

Direct conversion $M(a)[k]$ the argument field A contains the data to be converted, and the arguments B and C contain the iterative key values in direct numbering

$$M_j(a)[k]= M_j(A, B, C) = a◈k_i◈k_{i+1}≡X[k](a), \quad (7)$$

where j is the sequence number of component-wise functions; i= 1, 2, …, 10 isthe sequence number of the iteration key if $i> 10 \Rightarrow i := i \bmod 10$; k is the logical values of bits of the iteration key.

Inverse transformation $M^{-1}_j(a)[k]$:the argument A field contains the data to restore, and the arguments B and C contain the iterative key values in reverse numbering

$$M^{-1}_j(a)[k]= M^{-1}_j(A, B, C) = a◈k_{i+1}◈k_i≡X^{-1}[k](a), \quad (8)$$

where: i = 10, 9, …, 1 is the sequence number of the iteration key if $i< 1 \Rightarrow i := i \bmod 10$.

Let's apply the direct transformation specified in equation (7) in the process of encryption and the inverse equation (8) in the process of decryption

$$D(a) = M_j^{-1}M_j(a)[k], \quad (9)$$

consequently, such a double transformation results in restoration of the input data, similar to the transformation specified in equation (6).

Statement 2. On the distribution of the field of logical zeros and ones of the truth table of potentially suitable component functions.

The forward and reverse conversion functions, potentially suitable for use, contain an equal number of zeros and ones, as confirmed by the condition

$$M^{-1}{}_j M_j(a)[k] = M(M(A, B, C), B, C) = A, \quad (10)$$

which gives us:

$$\begin{cases} M(M(0,B,C),B,C) = 0 \\ M(M(1,B,C),B,C) = 1 \end{cases}.$$

When performing substitution for all combinations of arguments, we choose from potentially suitable functions, which restore the field of argument A. Let's not consider here a function identical to the first argument, and its negation as not depending significantly on other arguments and, therefore, unsuitable for encryption. In accordance with the statements above and using the theory of abstract algebra and combinatorial techniques, let us perform quantitative calculations for the functions of two, three, and four arguments on the condition of equation (10), the results of which are given in table 1.

Table 1. Quantitative characteristics of component-wise functions.

№	Caracteristics	Numberofvariables		
		2	3	4
1	TotalBooleanfunctions	16	256	65536
2	Potentiallysuitablefunctions	6	70	17920
3	All functions that restore the variable A	2	14	254
4	Usefulfunctions	2	10	218

As follows from condition (10), the functions, keeping the argument A field, contain equal amount of zeros and ones in the ordinate vector. We consider such functions potentially applicable in the encryption algorithm. The number of such functions of n variables is described with the formula:

$$C(2^n, 2^{n-1}), \quad (11)$$

where C is the number of combinations.

In particular, for n=4 we get C(16, 8)=17920

But not all the potentially suitable Boolean functions restore the argument A field. Another condition can be derived from equality (10): for inverse sets of argument values with A variable the function takes on inverse values. This condition can be easily confirmed with the following algorithm.

Algorithm 1.

Step 1. Let us divide the field of logical values of Boolean function F in half. Wegettwovectors: F_1andF_2.

Step 2. If $F_2 = \neg F_1$, then function F restores the first argument. Otherwise it doesn't.

Performing the final selection of component functions, the number of which is specified inTable 1, let us form their perfect and then the minimal disjunctive normal form (MDNF). The results of the MDNF function formation are given in Table 2.

It should be clarified that the symmetric arrangement of the same transition probabilities in Table 2, with a natural increase in weight coefficients of the field of zeros and ones, characterizes the completeness and correctness of the representations of component-wise functions.

Unfortunately, some of the presented functions are interdependent. It is obvious, that if function F keeps the argument A field, function ¬F obtains the same property. Such dependence between cryptographic functions is unacceptable, that is why only the 5 functions, listed above, are to be used in the encryption/decryption algorithm, while F or ¬F can be picked randomly.

There are other functions to restore arguments B and C, MDNF for which are not given in Table 2. Functions, keeping the argument B field, meet the following condition: for sets of argument values, inverse on variable B, i.e. for sets (*A*,*B*,*C*) and (*A*,¬*B*,*C*), they take on inverse values. This condition can be easily confirmed with the following algorithm.

Table 2. Component-wise functions used to transform data.

№ (*j*)	Component-wise functions		Transitionprobabilities, %	
	FunctionBooleanvalues	function MDNF	0→0 1→1	0→1 1→0
1.	00011110	¬ABC∨A¬B∨A¬C	75	25
2.	00101101	¬AB¬C∨A¬B∨AC	75	25
3.	01001011	AB∨A¬C∨¬A¬BC	75	25
4.	01101001	AB¬C∨A¬BC∨A¬B¬C∨ABC	50	50
5.	01111000	¬AB∨¬AC∨A¬B¬C	25	75
6.	10000111	AB∨AC∨¬A¬B¬C	75	25
7.	10010110	A¬BC∨AB¬C∨¬A¬B¬C∨¬ABC	50	50
8.	10110100	¬AB∨¬A¬C∨A¬BC	25	75
9.	11010010	¬A¬B∨¬AC∨AB¬C	25	75
10.	11100001	¬A¬B∨¬A¬C∨ABC	25	75

Algorithm 2.

Step 1. Let us divide the field of logical values of Boolean function F in 4 equal parts: F_1, F_2, F_3, and F_4.

Step 2. If $F_2 = \neg F_1$ and $F_4 = \neg F_3$, then function F restores the second argument. Otherwiseitdoesn't.

Similar algorithm can be used to find functions, restoring the third argument. The only difference is that the logical values field of Boolean function F is divided into 8 equal parts (for n=3 we get one component in each part), and set them equal pairwise. To define a component-wise function by an ordinate vector, we can randomly pick half of the values, and the other half is constructed inversely to the first one. So the number of Boolean functions of n variables that restore second, third, …, n^{th} argument is described with formula:

$$2^{2\wedge(n-1)}. \quad (12)$$

In the table 3 we present component-wise functions of three variables, restoring the second argument and significantly dependent on all the variables.

Table 3. Component-wise functions used to restore the second argument.

№ (*j*)	Component-wise functions		Transitionprobabilities, %	
	FunctionBooleanvalues	function MDNF	0→0 1→1	0→1 1→0
1.	00110110	¬AB∨A¬BC∨AB¬C	75	25
2.	00111001	¬AB∨A¬B¬C∨BC	75	25
3.	01100011	AB∨¬AB¬C∨¬A¬BC	75	25
4.	01101001	¬A¬BC∨¬AB¬C∨A¬B¬C∨ABC	50	50
5.	01101100	A¬B∨¬BC∨¬AB¬C	25	75
6.	10010011	AB∨BC∨¬A¬B¬C	75	25
7.	10010110	A¬BC∨AB¬C∨¬A¬B¬C∨¬ABC	50	50
8.	10011100	¬AB∨¬B¬C∨¬ABC	25	75
9.	11000110	¬A¬B∨¬B¬C∨ABC	25	75
10.	11001001	¬A¬B∨A¬B¬C∨ABC	25	75

In addition, there are other functions for restoring variables C that do not exist in Table 2 and 3.Show them in the Table 4.

Table 4. Component-wise functions used to restore the third argument.

1.	01010110	6.	10010101
2.	01011001	7.	10010110
3.	01100101	8.	10011010
4.	01101001	9.	10100110
5.	01101010	10.	10101001

We have eliminated 6 functions with fiction variables. When we calculated the total number of Boolean functions from the four variables used to transform the data, we excluded from the total number 38 functions with fiction variables.

Statement 3. On the use of component-wise functions in cryptographic transformation.

The existing algorithm of cryptographic transformation GOST R 34.12-2015 uses only one function of two variables, the AMT function, to be exact, (see the left algorithm in the Figure). According to Table 2, there are ten different functions of the three variables that satisfy the condition of restoring the input value of the field of argument A, therefore, in cryptographic information conversion can be used one of the functions listed in the table as well as any combination of these functions. When you use the required number of three-argument functions, the encryption algorithm takes the form shown on the right side of the Figure.

5. Analysis of transformations by component-wise functions

One way to demonstrate the practical capabilities of component functions is to perform transformations of functions from Table 2 and compare the results with the results given in GOST R 34.12-2015 [1].

To use component-wise functions, we extend the general appearance of the bitwise addition property, as shown in transformation (1), to apply the binary logic functions of three arguments [4]

$$X[k](a) = M_j\{(a)[k_1][k_2]\}, \quad (13)$$

where: M_j is the sequence number of the functions in Table 2; k_1 and k_2 are the iterative keys.

Extending properties of bitwise addition in the conversion (1) to the form of transformation (13) makes other combinations of operators, which gives the whole cryptographic transformation a new property, i.e. *component-wise transformation*.

We are going to present the results of a component-wise transformation for direct transformation $M_j(a)[k]$, corresponding to the application of equation (7), taking into account transformation (13), for functions the j = 1 and 2 from Table 2:

$$M_1(a)[k] = a \boxplus k_1 \boxplus k_2 = 99ba99dc51325510ffefddefbbabddef; \quad (13, a)$$

$$M_2(a)[k] = a \boxplus k_1 \boxplus k_2 = 6766232267666700fecc988832221000, \quad (13, б)$$

where: a = 1122334455667700ffeeddccbbaa9988 [1, p. 14]; k_1 = 8899aabbccddeeff0011223344556677 [1,p. 13]; k_2 = fedcba98765432100123456789abcdef [1, p. 13].

It should be noted that the result of the bitwise addition (1) given in [1, p. 14] has the form

$$X[k](a) = k_1 \oplus a = 99bb99ff99bb99ffffffffffffffff, \quad (14)$$

expectedly different from transforming equation (13), the results of which are given in transformation (13, a and b) proposed by the authors method of component conversion.

6. Conclusion

The three-argument binary logic algebra functions extending the cryptographic transformation modes of Russian National Standard GOST R 34.12-2015 are the beginning of a series of similar functions for four, five and more arguments, and their use will undoubtedly increase the possibilities of a bit-wise algorithm transformation. It is also possible to extend the approach to the considered question using the method of work [7], provided that the values of the arguments of the component-wise functions are read from the

file. Further research will be aimed at the development of a new type of functional transformation, followed by its implementation in the software and hardware complex, which will enable introduction of the next version of GOST R 34.12-2015.

7. References

[1] *GOSTR 34.12-2015 2015 Information Technology. Cryptosecurity. Block Ciphers* (Moscow: Standartinform) p 25
[2] Shannon C E 1963 *Works on Communication Theory and Cybernetics* (Moscow: Publishing house of foreign literature) p 830
[3] Shannon C E 1948 A Mathematical Theory of Communication *Bell System Technical Journal* **27** 379-423, 623-656
[4] Fomichev M I 2003 *Discrete Mathematics and Cryptology* (Moscow: Dialogue-MEPhI) p 400
[5] Sultanov D M, Vasilischin I I and Pugin M S 2016 Architecture Concept of the Encryption Unit, Performing the Russian National Standard GOST R 34.12-2015 Cryptographic Transformation Based on PAC. Expanding the Modes of the GOST R 34.12-2015 Cryptographic Transformation Based on a Specific Microcontroller *Proceedings of the international scientific conf. Parallel Computing Technology* 797
[6] Lidl R and Pilz G 1984 *Applied Abstract Algebra* (Springer-Verlag New York Inc.) p 743
[7] Yumaganov A S and Myasnikov V V 2017 A method of searching for similar code sequences in executable binary files using a featureless approach *Computer Optics* **41(5)** 756-764 DOI: 10.18287/2412-6179-2017-41-5-756-764