http://doi.org/10.35784/iapgos.919

BLOCK CIPHERS ON THE BASIS OF REVERSIBLE CELLULAR AUTOMATA

Yuliya Tanasyuk, Petro Burdeinyi
Yuriy Fedkovych Chernivtsi National University, Department of Computer Systems and Networks, Chernivtsi, Ukraine

Abstract. *The given paper is devoted to the software development of block cipher based on reversible one-dimensional cellular automata and the study of their statistical properties. The software implementation of the proposed encryption algorithm is performed in C# programming language in Visual Studio 2017. The paper presents specially designed approach for key generation. To ensure a desired cryptographic stability, the shared secret parameters can be adjusted in order to contain information needed for creating substitution tables, defining reversible rules, and hiding the final data. For the first time, it is suggested to create substitution tables based on iterations of a cellular automaton that is initialized by the key data.*

Keywords: block cipher, symmetric encryption algorithm, reversible cellular automata

SZYFRY BLOKOWE NA PODSTAWIE ODWRACALNYCH AUTOMATÓW KOMÓRKOWYCH

Streszczenie. *Niniejszy artykuł poświęcony jest rozwojowi oprogramowania szyfrów blokowych opartych na odwracalnych jednowymiarowych automatach komórkowych oraz badaniu ich właściwości statystycznych. Zastosowanie oprogramowania w proponowanym algorytmie kodowania wykonywane jest w języku programowania C# w Visual Studio 2017. Artykuł przedstawia specjalnie zaprojektowane podejście do generowania klucza. Aby zapewnić pożądaną stabilność kryptograficzną, dostosowane mogą zostać wspólne tajne parametry w taki sposób, aby zawierały informacje wymagane dla stworzenia tabel substytucyjnych, określające zasady odwracalne oraz ukrywające dane końcowe. Po raz pierwszy, proponowane jest tworzenie tabeli substytucyjnych w oparciu o iteracje automatów komórkowych, które zostają zainicjowane poprzez dane klucza.*

Słowa kluczowe: szyfr bloku, algorytm szyfrowania symetrycznego, odwracalny automat komórkowy

Introduction

The increased use of computers, converged networks with high-speed Internet access and IoT deployment resulted in an urgent need for means to protect information and to provide various security services. Encryption is known to be a primary method of protecting valuable electronic information. A cryptographic algorithm, or cipher, is a set of well-defined but complex mathematical instructions used to encrypt or decrypt data. The encryption and decryption processes depend on a cryptographic key selected by the parties participating in the communication process. Typically, details of the algorithm are publicly open. However, operation of the algorithm and security of the encrypted message relies on the cryptographic key used in the encryption and decryption process.

The transformation of a message from plaintext to cipher-text occurs through a substitution or a transposition process, or a combination of both. A substitution cipher replaces a digit or a data block in a message with another arbitrarily chosen digit or data portion. A transposition cipher implies different permutations of a data block. Based on how cryptographic algorithms are applied on the plaintext, they are categorized as block ciphers and stream ciphers.

As the name implies, the block ciphers work on a fixed-length segment of plaintext, typically a 64- or 128-bit block as input, and produces a fixed length cipher text, usually of the same size as the input. The message is broken into blocks, and each block is processed in the same manner. Where there is insufficient data to fill a block, the blank space will be padded prior to the encryption. Block ciphers are mostly used in the symmetric key encryption. DES, Triple DES and AES are some of the well-recognized examples of block ciphers [4, 7].

Cellular automata (CA) are typically considered as a regular grid of cells, with each presenting a finite number of possible states. These automata cells are modified independently by the transition function on a discrete time step. The application of the function to each cell in the grid leads to the next generation for the grid. The outcome of the transition function depends on states of the cell itself and of their neighbors. Every cell follows the same rule for determining these transitions. Types of their interaction are simple and diverse, while their implementation imposes low demands for computational complexity. Some of the CAs are reversible, enabling one to restore the information processed through direct transformations [7].

1. Reversible cellular automata

A number of papers are dedicated to the application of CA in cryptography [1–3, 5–7]. Namely, they are considered as promising candidates for symmetric and asymmetric enciphering. Security of the latter was based on the complexity to solve non-linear polynomial equations. Stream enciphering with the use of CA was first proposed by Wolfram [11]. The idea consists in usage of CA as the generator of pseudorandom numbers. The considerations were further embodied in the algorithms, developed by Seredynski [7] and Tomassini [8]. The block cipher using both reversible and irreversible rules was reported in [1, 3].

As a dynamic system CA can be represented as follows [7]:

$$A = \{S, Z^d, f, V\}, \quad (1)$$

where S is a finite number of states; Z is the set of integers; d is the size of automation; Z^d is the space of CA (the number of cells), f is a rule (transition function), V is the set of neighbours (including the current cell and the neighbours involved in interaction).

The simplest CA can be represented as one-dimensional array of 0 and 1, as the states of cells. Each cell has its network environment of three cells: left, right and a current cell itself. Normally, finite CA are used with cyclic edge conditions, when the first and the last cells are treated as neighbours. The CA consists of a number of steps. When calculating the next state of a cell, the step changes. In order to execute the next function of the state, three states of the interacting cells are applied as an input, producing the next state of the cell on the output.

A CA space denotes the number of the cells, which are updated according to some rule f [11]. In total, the 256 rules of CA interaction are defined. For example, the rule 30 in terms of Boolean functions is given as follows:

$$C[i]' = C[i\text{-}1] \oplus (C[i] \vee C[i+1]) \quad (2)$$

where C[i] is a current cell, C[i]′is the value of the current cell after the rule application, C[i-1], C[i+1] are previous and next neighbor cells, and $\oplus$, $\vee$ denote the bitwise XOR and OR operations, respectively. As shown in Fig. 1, the rule 30 is called so since all possible combinations of cell states at step t produce a sequence of 00011110 which when converted to the decimal system gives a value of 30.

t	1 1 1	1 1 0	1 0 1	1 0 0	0 1 1	0 1 0	0 0 1	0 0 0
t + 1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**

Fig. 1. CA cell states resulted from application of rule 30

Some CA rules possess an interesting property of being reversible, providing not only a direct but also a reverse iteration. When applying reversible rule, this enables the CA getting back to the initial state. To be applicable for cryptographic purpose the reverse rules must comply with the following criteria: they must be numerous and exhibit complex behavior. When analyzing elementary CA, it turns out that only a small number of rules are known to be reversible. For example, of all the 256 elementary radius rules, only six are stated to be reversible. In addition, their behavior is very simple [3, 7]. For this reason, standalone elementary reversible rules cannot be used for encryption.

In order to accomplish this task, it is proposed to use a class of reversible rules, first described by Wolfram in [9, 11]. Each rule belonging to this class can be described by two elementary CA transition rules. The first one determines the state of transition in the case when at step *t*-1 the cell is in the state of 0, and the second rule applies for the cell state of 1. These two rules depend on each other. By knowing one rule, we can derive another one using the following formula:

$$R_2 = 2^n - R_1 - 1, \quad (3)$$

where R is the elementary rule; n is the neighborhood of the cell. Fig. 2. shows the reversible rule consisting of rule 57 $(00111001)_2$ and 198 $(11000110)_2$, inverse to it.

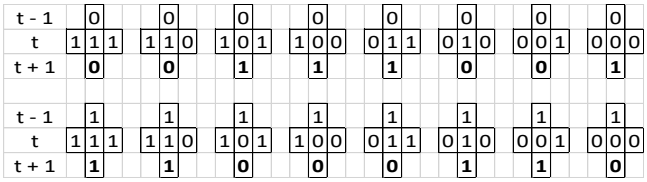


Fig. 2. Reversible rule, including CA transformation rules 57 and 198

2. Cryptographic application of the reversible CA

Since the reversible rule depends on a previous step, the initial state of the CA must consist of two consecutive configurations. The text to be encrypted comes as a second configuration, while the first configuration is populated with random data (Fig. 3). Traditionally, the encryption is performed by direct iteration of the CA. However, the final outcome consists of two configurations and both of them must be used in the decryption. The first is encrypted text and the second is called the final data. During decryption, these operations are executed in reverse order [10].

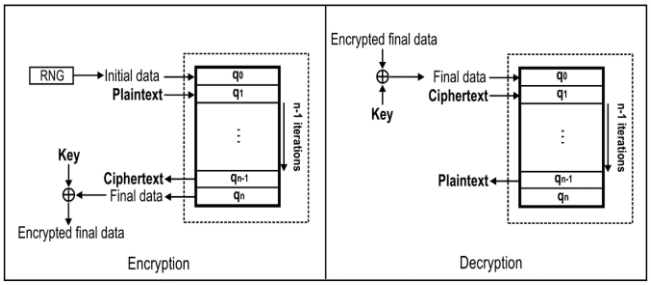


Fig. 3. Basic scheme of encryption and decryption process [9]

A rule used for both encryption and decryption is considered as a secret key. The end data must be kept in secret, since knowing two consecutive configurations (end data and encrypted message) one can easily define the rule, applied for the encryption.

There are two approaches for processing the final data generated in the encryption process. The most secure one assumes that this information is kept private, and therefore it becomes a part of a key. Now, the key consists of the rule and final configuration. The drawback of this option is that after each encryption the key is to be changed and shared with the message recipient. According to the second option the end data is encrypted with the use of Vernam algorithm [5] applying logical bitwise operation of XOR to the final data and the key portion as follows:

$$efd_i = k_i \oplus fd_i, \quad (4)$$

where k_i is the *i*-th bit of the key, fd_i is the *i*-th bit of the end data, efd_i is the *i*-th bit of the encrypted end data, $\oplus$ denotes XOR operation.

Now, the encrypted final data should be no longer kept in secret and can be added to the cipher text.

3. Software implementation of the block cipher on the basis of reversible one-dimensional CA

The paper aimed at development of the block cipher on one-dimensional CA, processed by reversible CA transformation rules. Software implementation of the proposed encryption algorithm has been performed in the C# programming language in the integrated application development environment of Visual Studio 2017.

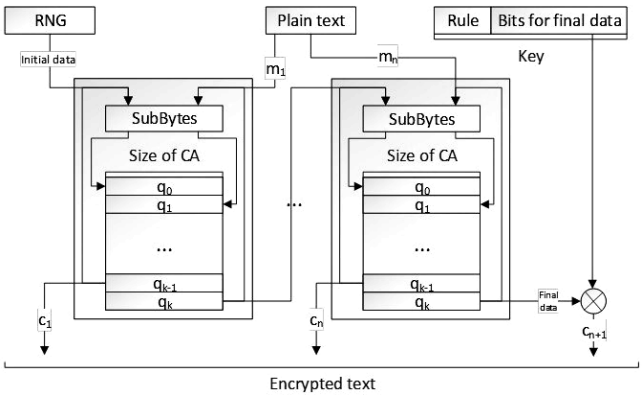


Fig. 4. Generalized representation of the chosen encryption approach

The designed algorithm uses the single reversible one-dimensional CA and the corresponding rules. To achieve basic cryptographic strength, we have proposed a novel approach to key formation, when the rule to use for CA transformation and specific bits for concealing the final data are contained inside the key. Depending on the needs the rule may be implemented with different radii (1, 2, 3). The larger the radius of the rule, the more time the calculations take, yet producing more tangled output.

The block size may acquire values of 128, 256 and 512 bits. The key may be 384, 512 or 640 bits long. The standard algorithm leans upon one reversible rule, however, its modification implies utilization of several transition functions. The number of iterations and rounds may vary. The size of CA equals the doubled size of the block, since the reversible rule depends not only on the neighbours on the right or left, but on the state of the cell at previous iteration. Schematically, the implemented encryption algorithm is shown in Fig. 4.

To utilize the algorithm, input messages are read and padded to the size multiple of the block *S* size. Then some random value (Initial data) of *S* bits in size is generated. The CA is initialized with this initial data and a block of information to be encrypted (m_1).

Before being supplied to the CA, the data undergo procedure of byte substitution, denoted as SubBytes, with the use of the AES substitution tables. The algorithm implies utilization of the alternative substitution tables which should be generated and transmitted together with the secret key. After that, the CA is processed with the reversible rule of radius 3 obtained from the key. The *h* rounds produce a portion of encrypted information (c_1) and data that can be used to initialize the cellular automaton when encrypting the next block of information. In this way all blocks of information are encrypted. The last piece of the encrypted message (c_n), i.e. final data, should be hidden because their discovery may provide a clue for deciphering all the information. For this reason, XOR operation is additionally applied to the final data and specific bits of the key, producing the outcome which supplements the encrypted message (c_{n+1}) [7].

The decryption algorithm includes the same steps of the encryption algorithm in the reverse order.

4. The function creating a key and substitution tables

The cryptographic key consists of the following components:

- bits for initialization of the CA, generated by the substitution tables (128 bits);
- CA rules with the radius of 3 (128 bits);
- special bits for hiding the CA final data (the size is equal to the block size).

The size of the key is calculated using a formula:

$$L = 128 + 128 + S, \tag{5}$$

where S is the size of the block.

Thus, the key may be 384, 512 or 640 bits long.

For key generation the System.Security.Cryptography module of NET Framework has been used. The data generated, experience 1000 iterations on the CA with rule 30 and radius 1. As a result, the encryption key of the algorithm is obtained.

The function of substitution tables formation creates two S-Box tables and one Inverse S-Box, providing protection against attacks based on simple algebraic properties. In fact, this is an example of common cipher of the simple substitution.

The substitution tables are generated on the basis of the CA operating with the application of rule 30 with the radius 1 and the key bits used for the CA initialization (Fig. 5).

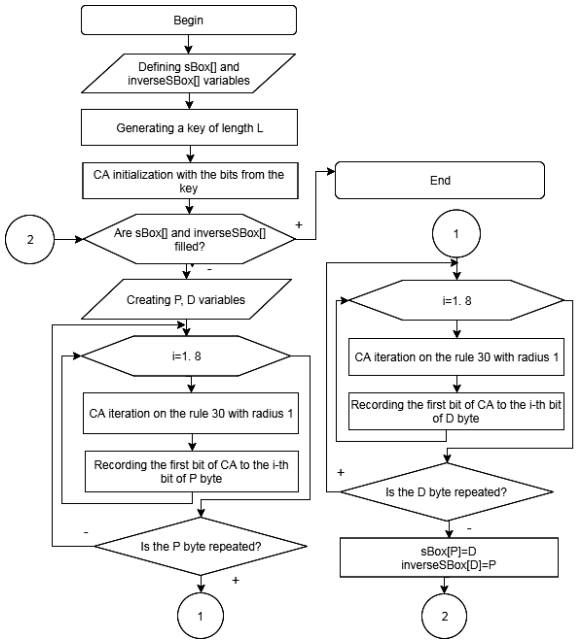


Fig. 5. The flow chart of generating substitution tables using the reversible CA

The procedure forming the substitution tables is as follows. First, the CA is initialized with the specific part of the key to generate the tables. The CA are processed until direct and inverse tables are completed.

With each iteration the first bit of CA is written to form a location byte P, pointing to a cell in the substitution table. If this cell appears to be already used, new location byte will be generated. After the location byte is formed, another byte D is derived. If its value is already in use, a new byte value will be formed. In parallel, the inverse substitution table is created. In this table the byte-value (D) becomes a location byte (P) and vice versa. As a result, two inverted substitution tables are formed. Repeating this function forms the identical substitution tables.

Depending on demands, the designed algorithm can be easily modified. The key comprises additional reversible CA rule with radius 3 to be applied. Eq. 5 used to calculate the key size is altered as follows:

$$L = 128 + 128N + S, \tag{6}$$

where N is the number of the additional reversible CA rules.

5. Scattering properties of the designed block cipher

Pseudorandom behavior is generally considered as a good indicator of a secure block cipher. We have used a technique of NIST STS statistical testing in order to check randomness properties of the developed encryption algorithm. Good encryption algorithm should also satisfy the Strict Avalanche Criterion [6]. This means that each output bit should change with a probability of one half when-ever a single input bit is complemented.

Investigation of the scattering properties of the block cipher based on reversible one-dimensional cellular automata has been performed on the binary file of 12.3 MB resulted from the programmed encryption procedure through the designed algorithm applied to cellular automata of the corresponding length. The statistical suit of NIST STS v.2.1.2 divided generated binary sequences into 100 equal parts of 10^6 bits each. The bit strings were tested against 15 statistical tests with different parameters. The randomness properties were assessed in terms of probability of the tests being passed. As a result, a vector of 189 values of probability was formed. Ideally, only one sequence out of a hundred can be rejected, providing a coefficient of the test passing of 99%. However, this requirement is rather strict. In most cases the evaluation is conducted within a confidence interval, the lower limit of which is assumed to be at the level of 96% [10].

The following initial parameters were used during the testing:

- binary file of 12.3 MB;
- sequences of 103.4 Mbits.

The performance of the selected set of transformations was evaluated on a following hardware platform: AMD Athlon X4 740 Quad Core Processor 3.2 GHz, AMD Radeon HD 7700 (1050 MHz), 8 GB RAM.

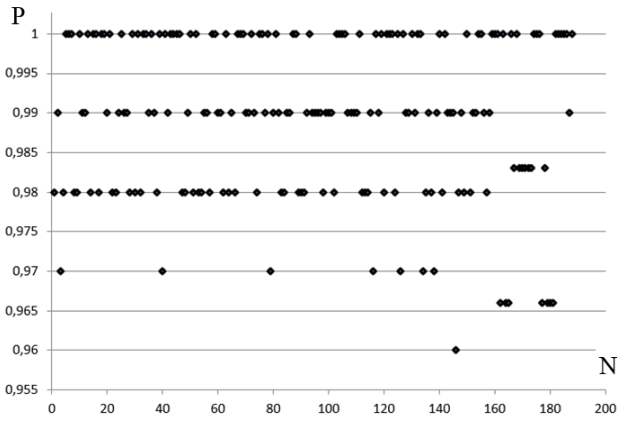
When studying the block cipher encryption algorithm based on reversible one-dimensional cellular automata, we used a combination of one (RCA1), two (RCA2) and three (RCA3) reversible rules with a radius of 3. The results presented in Table 1 consider the block size of 256 bits, 5 processing rounds and 5 iterations.

Table 1. Statistical and performance parameters of the designed block ciphers

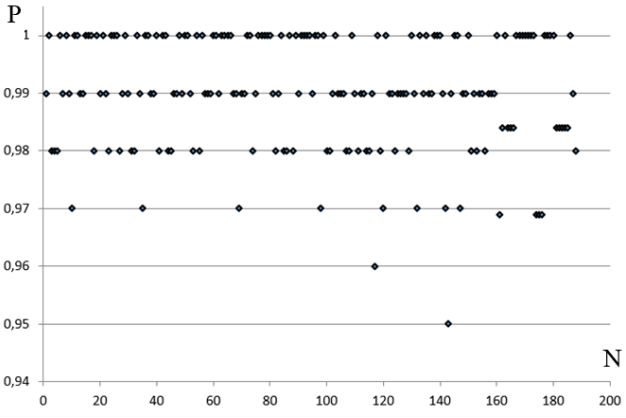
Parameters	RCA1	RCA2	RCA3
The number of tests passed by at least 99% of the sequences	68.6%	72.3%	73.9%
The number of tests passed by at least 96% of the sequences	100%	99.5%	99.5%
Minimal proportion of the tests passed	96%	95%	95.5%
12.3 MB file encryption time	7 min 55 sec	15 min 22 sec	23 min 26 sec

Fig. 6. shows the results of the conducted statistical testing. The obtained data prove that the least ratio of bit sequences that successfully passed the tests, is at the level of 96% – 97%, pointing out satisfactory scattering properties of the developed encryption algorithm.

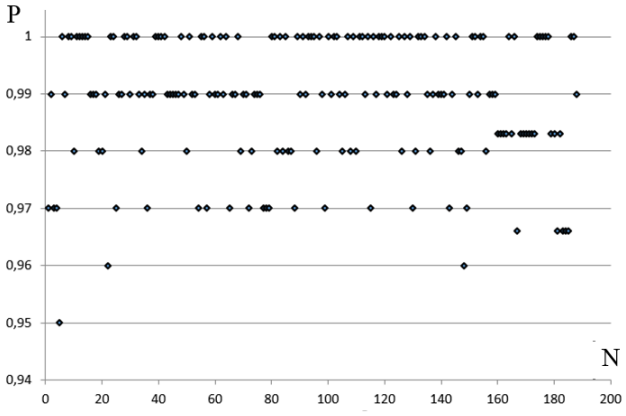
Investigating scattering properties of the proposed block cipher built on the basis of reversible one-dimensional cellular automata with NIST STS revealed the applying three transformation rules to be most effective. Inclusion of additional processing rules to the algorithm ensures better scattering properties. However, the most optimal in terms of performance and statistical properties is a one-way design with a radius of 3, using 5 rounds and 5 iterations. Avalanche effect investigations should be further performed.



a



b



c

Fig. 6. Statistical portraits of the block cipher on the reversible one-dimensional CA, processed by one (a), two (b) and three (c) transformation rules with a radius of 3. The block is 256 bits long. The chosen reversible rules were applied for 5 round including 5 iterations. N is a number of a test, P is the portion of sequences under study that passed the test

6. Conclusions

Thus, summarizing the investigations carried out, the following conclusions can be made:

1) By means of the C# programming language, a software implementing the block cipher on reversible one-dimensional cellular automata has been developed, which allows to process files of arbitrary types.
2) In order to ensure the cryptographic stability, a key generation approach has been developed. The key is designed to contain information about processing rules, data to create substitution tables, and information to hide the final data.
3) For the first time we have proposed to form substitution tables based on iterations of the cellular automaton, initialized by the key data. This may allow for additional protection against attacks in case when the applied reversible rules are revealed.
4) To enhance cryptographic strength, the basic encryption algorithm with the use of single one-dimensional CA and one reversible rule with radius 3 can be complemented with two or three reversible rules.
5) The created block cipher design uses blocks of 128, 256, 512 bits, and allows one to generate the keys of 384, 512 and 640 bits.
6) Investigations of the scattering properties of the block cipher based on one-dimensional CA using NIST STS statistical tests revealed that the minimum portion of studied bit sequences, meeting the requirements of the tests, fell within 96% – 97%, indicating the qualitative statistical characteristics of the developed cryptosystem.
7) According to the research conducted, for the same number of processing rounds, the use of three reversible rules with a radius of 3 appeared to be most effective, ensuring better scattering characteristics. However, in terms of performance a one-rule design with a radius of 3, turned out to be more appropriate.

References

[1] Bouchkaren S., Lazaar S.: A fast cryptosystem using reversible cellular automata. International Journal of Advanced Computer Science and Applications 5(5)/2014, 207–210.
[2] Debasis D., Abhishek R.: A parallel encryption algorithm for block ciphers based on programmable reversible cellular automata. J. Computer Science and Engineering 1(1)/2010, 82–90.
[3] Gutowitz H.A.: Cryptography with Dynamical Systems: Cellular Automata and Cooperative Phenomena. Kluwer Academic Press, Dordrecht 1993.
[4] Paar C., Peltz J.: Understanding cryptography. Springer-Verlag, Berlin Heidelberg 2010.
[5] Leporati A., Mariot L.: Cryptographic properties of bipermutive cellular automata rules. J. Cellular Automata 9/2014, 437–475.
[6] Seredynski M., Bouvry P.: Block cipher based on cellular automata. New Generation computing 23(3)/2005, 245–258.
[7] Seredynski F., Bouvry P., Zomaya A. Y.: Cellular automata and secret key cryptography. Parallel Computing 30(5-6)/2004, 753–766, [http://doi.org/10.1016/j.parco.2003.12.014].
[8] Tomassini M., Perrenoud M.: Stream Cyphers with One- and Two-Dimensional Cellular Automata. Parallel Problem Solving from Nature PPSN VI. PPSN. Lecture Notes in Computer Science 1917. Springer, Berlin, Heidelberg, 2000, 722–731.
[9] Wolfram S.: Cryptography with Cellular Automata.: Advances in Cryptology: Crypto'85, Springer-Verlag LNCS 218, 1985, 429–432.
[10] NIST SP 800-22: Documentation and Software. Random bit generation. Guide to the statistical tests, https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software/Guide-to-the-Statistical-Tests
[11] Wolfram S.: A New Kind of Science.: Wolfram Media, Inc, 2002, 1197, http://www.wolframscience.com/nksonline/toc.html

Ph.D. Yuliya Tanasyuk
e-mail: y.tanasyuk@chnu.edu.ua

Associate professor at Department of Computer Systems and Networks, Physical, Technical and Computer Sciences Institute, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine. Research interests and academic activities: programming, network information technologies, cryptography.

http://orcid.org/0000-0001-8650-0521

M.Sc. Petro Burdeinyi
e-mail: pburdeyniy@gmail.com

Master in Computer Engineering, Department of Computer Systems and Networks, Physical, Technical and Computer Sciences Institute, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine. Research interests and academic activities: cryptography, information technologies, software engineering.

http://orcid.org/0000-0002-3859-7522

otrzymano/received: 15.11.2019 *przyjęto do druku/accepted: 15.02.2020*