DOI: 10.5604/01.3001.0010.8638

DEVELOPMENT AND RESEARCH OF CRYPTOGRAPHIC HASH FUNCTIONS BASED ON TWO-DIMENSIONAL CELLULAR AUTOMATA

Yuliya Tanasyuk[1], Sergey Ostapov[2]
[1]Yuriy Fedkovych Chernivtsi National University, Department of Computer Systems and Networks, [2]Yuriy Fedkovych Chernivtsi National University, Computer Systems Software Department

Abstract. *Software solution for cryptographic hash functions based on sponge construction with inner state implemented as two-dimensional cellular automata (CA) has been developed. To perform pseudorandom permutation in round transformation function several combinations of CA rules 30, 54, 86, 150 and 158 have been proposed. The developed hashing mechanism provides effective parallel processing, ensures good statistical and scattering properties, enables one to obtain hash of a varying length and reveals strong avalanche effect.*

Keywords: cryptographic hash functions, cellular automata, cryptographic sponge, pseudo-random permutations

OPRACOWANIE I BADANIA KRYPTOGRAFICZNYCH FUNKCJI SKRÓTU (HASH) NA PODSTAWIE DWUWYMIAROWYCH AUTOMATÓW KOMÓRKOWYCH

Streszczenie. *Za pomocą oprogramowania zostały opracowane kryptograficzne funkcje skrótu (hash) na podstawie gąbki kryptograficznej, której stan wewnętrzny został zrealizowany w postaci dwuwymiarowych automatów komórkowych (KA). W celu implementacji permutacji pseudolosowych zaproponowano kombinację zasad obróbki CA 30, 54, 86, 150 i 158 w celu realizacji funkcji transformacji rundy. Opracowany mechanizm haszowania pozwala na skuteczne przetwarzanie równoległe, zapewnia jakościowe charakterystyki statystyczne i rozproszenia, pozwala na otrzymanie skrótu o zmiennej długości i ujawnia stabilny efekt lawinowy.*

Słowa kluczowe: kryptograficzne funkcje skrótu, automaty komórkowe, gąbka kryptograficzna, przekształcenia pseudolosowe

Introduction

Hash functions have many applications in modern cryptography, including Internet Security Protocol (IP Sec), digital signature schemes, password storage and key derivation. Among all, these cryptographic primitives are probably best known for the important role they play in the practical use for data integrity and message authentication. Hash functions are considered to be efficient with respect to energy consumption and computations needed to produce a concise fixed-size signature of the message of arbitrary length. A computed fingerprint should be highly sensitive to all input bits. Finally, the hash functions must be secure, since they are normally used without keys and deal with a plaintext. A robust hash function possesses one-wayness, i.e. given a fingerprint it's infeasible to derive a matching message. Another security property of the hash functions, desirable for digital signature, is their collision resistance. It's essential that two different messages do not hash to the same value and it should be computationally infeasible to find such messages [6].

In order to evaluate the cryptographic level of security, provided by a hash function, it's a good practice to analyze its statistical properties. A hash function that behaves like a pseudorandom function and satisfies the avalanche effect is generally considered to be secure and can be used safely for cryptographic purposes. The avalanche effect reflects the sensitivity of the hash function to elementary changes in the hashed message: a little change in the input message (flipping one single bit) produces a significant change of the output (the final hash) [2].

A large number of hash functions have been proposed over the last three decades. The most popular algorithms of MD5 and SHA-1 are based on compression mechanism that is reported to cause the collision occurrence. Moreover, they appear to be incompatible with the algorithms of AES and 3DES, due to insufficient level of security. This drawback was eliminated in four existing versions of the SHA-2 algorithm, each providing the message digest of the fixed length (224, 256, 384 and 512 bits), yet falling under a threat of being compromised [7]. However, in 2012 through public competition, initiated by NIST, a new hashing algorithm named Keccak was adopted as a novel standard of SHA-3 to provide a backup for the existing hashing mechanisms in case of any further vulnerabilities revealed. The selected algorithm possesses many admirable properties. In particular, it is praised for the ability to run well on different platforms of computing devices, including embedded and smart ones, and for higher performance in hardware, comparing to SHA-2.

It should be stressed that Keccak has a quite different internal structure than the hash functions that belong to the MD4 family, including SHA-1 and SHA-2. It doesn't rely on a compression approach of its predecessors, but is based on a sponge construction. The sponge is an interactive framework, which utilizes pseudorandom permutation functions and provides the output close to that of random oracle [1, 6].

In order to achieve such an effect in programmable way, the use of cellular automata and their transformation rules seem to be appealing.

Cellular automata (CA) have several properties that favor their use as a basis for the design of hash functions. Their chaotic, complex and unpredictable behavior for some types of the applicable rules promotes their effective use to design safe and reliable hash functions. In addition, CA are very appropriate to design hash functions with low hardware and software complexity because of the involved logical operations, parallelism and extreme sensitivity to initial conditions alterations [2].

According to the results of investigations, reported in [5] linear CA, evolving with the use of various transformation rules, provide efficient means of creating high-performance key stream generators, block ciphers and hash functions with reasonable statistical properties. It is suggested, that deployment of multi-dimensional structures may bring about some additional opportunities for their parallel processing and performance enhancement.

The purpose of the given paper is to create and study crypto-graphic hash functions based on two-dimensional CA, with elaborated combinations of the processing rules, underlying permutation function of the sponge construction.

1. Cryptographic sponge: parameters and details of operation

In terms of SHA-3 standard Keccak the hash function supports the message digest lengths of 224, 256, 384 and 512 bits. Last three hash values provide an attack complexity of approximately 2^{128}, 2^{192} and 2^{256}, respectively, corresponding to the cryptographic strength that the three key lengths of AES provide against brute-force attacks. Similarly, 224-bit output shows the same collision resistance as 3DES with a cryptographic strength of 2^{212} [6].

The construction of sponge, used in Keccak algorithm, has its inner state − the binary array with a width of $b = 1600$ bits, divided into two parts: r and c, $b = r + c$. A parameter of r is called a bit rate. This very part is combined with the blocks of input message, for which the hash value is to be calculated. A second parameter of c, known as a capacity, doesn't interact directly with the portions of the input message, being involved only in the permutation process. Capacity is said to be responsible for the security level and with proven mathematical stability needs to be twice as large as the desired hash value. The security level denotes the number of computations an attacker has to perform to break the hash function. Table 1 gives recommended values of the sponge parameters for inners state of 1600 bits with respect to the hash length.

Table 1. The parameters of Keccak hash function [6]

Sponge state *b*, bits	Hash length, *Z*, bits	Rate *r*, bits	Capacity *c = b-r*, bits	Security level, *Z/2*, bits
1600	224	1152	448	112
1600	256	1088	512	128
1600	384	832	768	192
1600	512	576	1024	256

It's noteworthy, that Keccak function is not confined to the hash lengths, listed in Table 1. One may generate a message digest of any arbitrary number of bits, holding corresponding correlation of the parameters and security level [6].

In Keccak algorithm the sponge state is arranged as three-dimensional array of 5×5 64-bit words. The operation of sponge construction consists of two stages: absorbing and squeezing (Fig. 1). Prior to the actual processing by the hash function an input message undergoes preprocessing, which implies its padding to the length of a multiple of r bits [1]. Further the complimented string is divided into the equal portions of r length.

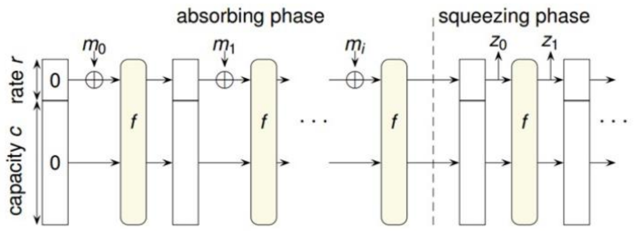

Fig. 1. Operating phases of the cryptographic sponge [1]

At each step of absorbing sponge intakes a message block, combines it with the r portion of the sponge state through XOR operation and further processes the whole state with round permutation function.

The squeezing stage begins after loading of all message blocks. It also includes permutation, followed by extracting a predetermined number of bits from r portion and their appending to the output hash string, until a message digest of the desired length is obtained.

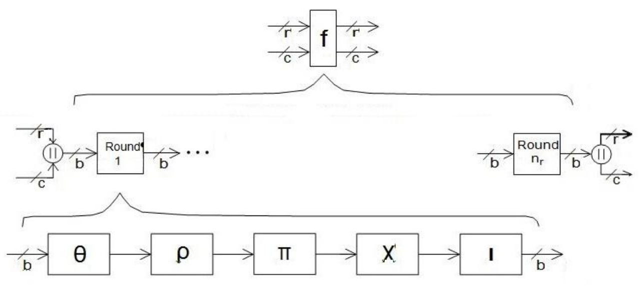

Fig. 2. The steps of the round function of Keccak-f permutation [9]

The Keccak-*f* permutation function is at the heart of the hash algorithm and is used in both phases of the sponge construction. The function includes n rounds. Each round has an input of $b = r+c$ bits. The number of rounds is defined as $n = 12+2\log_2 k$, where $k = b/25$, and for $b = 1600$ bits, $n = 24$. As shown in Fig. 2, each round consists of a sequence of five steps denoted by Greek letters: θ (theta), ρ (rho), π (pi), χ (chi) and ι (iota). The named functions include various combinations of bitwise operations, and are claimed to be relatively hardware friendly resulting in high performance of the Keccak algorithm [1, 6].

2. Hash functions with the use of processing rules of two-dimensional CA

Our experiments were aimed at creating a software model of Keccak-like hash function based on two-dimensional CA. Cellular automata can be viewed as a collection of cells organized in a grid, when each cell has a corresponding current state. The states of the cells evolve over time, depending on their current states and the states of the neighboring cells, according to a local and identical interaction rule in the case of uniform CA, or different interaction rules in the case of non-uniform or hybrid CA [3, 9].

The paper presents the results on the hash functions implemented as a sponge construction with the two-dimensional CA inner state, processed by the following transformation rules:

$$\text{rule 30: } C[i]' = C[i-1] \oplus (C[i] \vee C[i+1]) \tag{1}$$

$$\text{rule 54: } C[i]' = (C[i-1] \vee C[i+1]) \oplus C[i] \tag{2}$$

$$\text{rule 86: } C[i]' = (C[i-1] \vee C[i]) \oplus C[i+1] \tag{3}$$

$$\text{rule 150: } C[i]' = C[i-1] \oplus C[i] \oplus C[i+1] \tag{4}$$

$$\text{rule 158: } C[i]' = C[i-1] \oplus C[i] \oplus C[i+1] \vee C[i] \wedge C[i+1] \tag{5}$$

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 $\oplus$, $\wedge$, $\vee$ denote the bitwise XOR, AND, and OR operations, respectively.

The permutation functions under investigation assumed utilization of several CA processing rules, both linear and nonlinear, to provide collision resistance and nonlinearity to the hashing mechanism [4].

2.1. Interaction scheme of two-dimensional CA

In two-dimensional representation the sponge state is arranged as an array of 25 vectors of 64 bits, making 1600 bits in total. The cells are localized according to the Moore neighborhood [9], when two cells are considered adjacent if they have either a common edge or a vertex. Therefore, each cell interacts with its eight direct neighbors, denoted as parts of the world (Fig. 3).

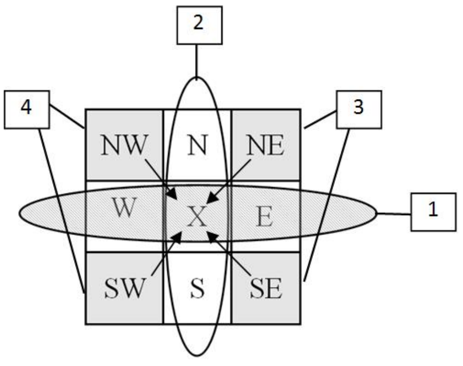

Fig. 3. The notation of the cells in two-dimensional 8-neighbor CA, used in the permutation function, where 1, 2, 3 and 4 are index numbers of possible interactions of cells

Cells, represented by N, W, NW, NE are considered as previos, while S, E, SW and SE participate as the next neighbours. Extreme cells are connected in tor with their counterparts on the opposite edge (row/column) of the array. In order to enhance performance, selected rules were applied to the entire rows concurrently, rather than to single bits. In particular, to implement transformation within one row two copies of it were created: one-bit cyclically shifted to the right instance represented all previous cells (W), while one-bit cyclically left-shifted one contained all next cells (E). Similarly, cyclically shifted to the right previous and next rows correspond to NW and SW vectors, respectively while their one-step cyclic shift to the left produces NE and SE bit strings, correspondingly.

Table 2. The developed permutation functions based on the CA processing rules, corresponding to the index numbers of interactions, given in Fig. 3

id	Notation of the permutation function	Interaction number	CA rule	Cell's interaction
1.	Rule_30_150_86	1	rule 30	X' = W⊕(X∨E)
		2	rule 150	X' = N⊕X⊕S
		3	rule 86	X' = (NE∨X)⊕SE
		4	rule 86	X' = (NW∨X)⊕SW
2.	Rule_54_150_86	1	rule 54	X' = (W∨E) ⊕X
		2	rule 150	X' = N⊕X⊕S
		3	rule 86	X' = (NE∨X)⊕SE
		4	rule 150	X' = NW⊕X⊕SW
3.	Rule_54_158_150_86	1	rule 54	X' = (W∨E)⊕X
		2	rule 158	X' = N⊕X⊕S∨X∧S
		3	rule 86	X' = (NE∨X)⊕SE
		4	rule 150	X' = NW⊕X⊕SW

To ensure effective permutation, combinations of adjacent cells were processed with different CA transformation rules for a number of times successively. In general, we'll consider three types of permutation functions, the notations and related CA rules of which are given in Table 2.

2.2. Implementation of sponge construction on two-dimensional CA

The software solution for the cryptographic sponge with the use of processing rules of CA has been developed in C++ programming language. The sponge state is implemented as a two-dimensional binary array of 25×64 bits, which consists of two parts of *r* and *c*. The values of the parameters are aligned with those proposed in the original Keccak algorithm. A number of vectors that belong to *r* portion of the state can be defined as *r*/64. E.g. for hash length of 512, *r* equals 576 (Table 1) and spans first nine 64-bit rows of two-dimensional array. These vectors are initialized with binary 1s. For better diffusion, a bit corresponding to a vector's index is inverted. The rows of capacity portion are set to 0 s. The whole sponge state is preliminary processed by application of one of the mentioned permutation function for 25 times.

At the absorbing phase *r* vectors of the sponge are loaded with equally divided blocks of the padded input message. Then, the whole state undergoes consecutive permutation for a number of times according to the deterministic rule combination.

In course of squeezing, on completion of permutation rounds, the content of the first vector is appended to the output message digest. When the hash string of the initially defined length is obtained, it is output to a user in hexadecimal representation.

3. Statistical properties and avalanche effect

Pseudorandom behavior and avalanche effect are generally considered as good indicators of a secure hash function, the output of which should resemble that of the random oracle. We've used a technique of NIST STS statistical testing in order to check randomness properties of the developed permutation functions.

In the performed experiment the implemented hash functions with the parameters (bits): $b = 1600$, hash length $Z = 512$, $r = 576$, $c = 1024$, have been utilized in the mode of a generator of pseudorandom numbers to create a binary file of 100 MB. 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 16 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 of a hundred can be rejected. This means that a coefficient of test passing equals 99%. However, this requirement is rather strict. In most cases the evaluation is carried out based on the confidence interval, the lower limit of which is assumed to be at the level of 96%. Fig. 4 shows the results of the conducted statistical testing.

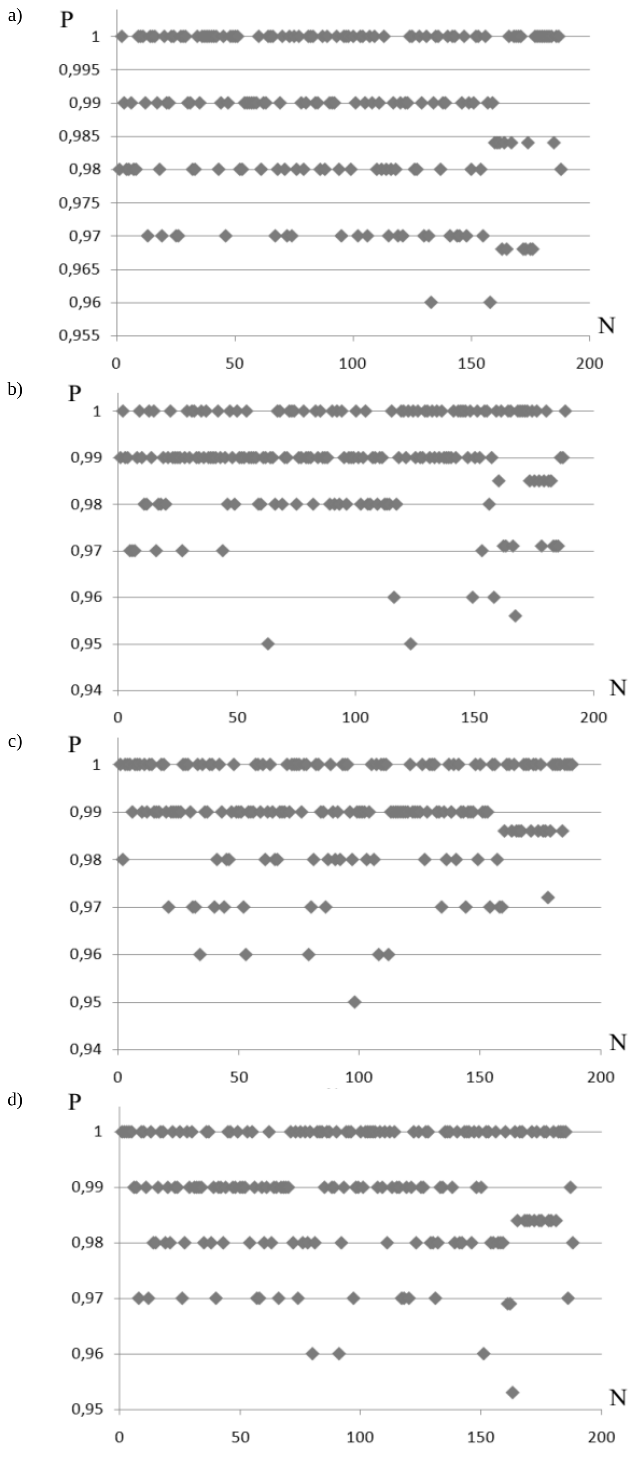

Fig. 4. Statistical portraits of the cryptographic hash functions, based on two-dimensional CA processing rules (Table 2), applied for a number of permutation rounds: a) RULE_30_150_86, 5 rounds; b) RULE_54_150_86, 5 rounds; c) RULE_54_150_86, 10 rounds; d) RULE_54_158_150_86, 5 rounds, where N is a number of a test, P is he portion of tests sequence that passed the test

According to the obtained statistical data, on average, up to 99% of all tested bit sequences have successfully passed the tests, with only a few of them revealing values at the level of 95%. The overall results of NIST STS testing for the proposed permutation functions at different number of processing rounds are listed in Table 3. The investigation revealed that with inclusion of rules 54 and 158 and at greater number of permutations the body of sequences, which passed the tests is slightly increased.

Table 3. Results of NIST STS testing of the hash functions based on the proposed permutation mechanisms of two-dimensional CA (see Table 2) at different number of processing rounds

Permutation function ID	Number of rounds	NIST STS testing results				
		0.95	0.96	0.98-0.97	1-0.99	Average
1	5	0	2	66	121	0.9889
2	5	2	4	48	135	0.9897
2	10	1	5	34	149	0.9907
3	5	1	3	57	128	0.9893
3	10	1	4	52	132	0.9896

The application of the developed permutation functions for obtaining a hash image revealed occurrence of strong avalanche effect. Namely, for the same input message a distinct hash string was obtained when changing hash length (224, 256, 384, 512 bits), applying permutation functions of various types, and modifying the number of processing rounds. Moreover, very small alterations of the input string produced significant changes in the output. The hexadecimal hash strings of 512 bits, resulting from 5-round application of RULE_30_150_86 permutation function, for the test vectors, are given below.

1) Empty file
baf53bbea29e4bb458aac3df56c023c55bf8ac117cfdabbcbc2e1
e2a069363cd453cba9c2468dd0389aef8630ae3e14b9461236e8
430388f9435fb529c3a9dbf
2) The quick brown fox jumps over the lazy dog
116312e16c3e9cf29c58687ed40f18e06ca478b64ae4c52e7ce4
e3a5dfacdc111f8cc6d97cfb5f1cb24d1ccb30edb8dd9a9ec4f1e
6deb2acb4517aa93d52709c
3) The quick brown fox jumps over the lazy **dog.**
48a4f612dfae9c2583923f0c5a03528f86c50386fff5958021635
d5a678be53c9f0d9e552468cb99c49e820da3376295928f37ae5
5e9d22516447f5379681126
4) The quick brown fox **jump** over the lazy dog.
e87efd05456a290eacb516f2c17f8a24c32ca5920f3f866b213d3
2d4ae42e0f67ee931537a5a09d420ffc485aae41e4f9760bb3c1e
a70bc7c7b3b99c8c7751bc

It should be outlined that the use of the two-dimensional CA and the designed combinations of the processing rules enabled us to considerably reduce the number of processing rounds. As reported in [8], for hash functions on the basis of one-dimensional CA, processed with combination of rules 30, 86, 150 and cyclic shift operations, the avalanche effect was achieved after 50 rounds of permutation. While a full change of hash image occurs for the designed two-dimensional hash functions starting from 5 processing rounds.

4. Conclusions

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

1) For the first time two-dimensional cellular automata, which evolve according to a series of processing rules, have been studied with respect to their utilization as a permutation function of the cryptographic sponge.
2) The developed permutation approach, which includes concurrent processing of two-dimensional CA and application of various interaction rules, provides good statistical properties and can be considered as a promising candidate for cryptographic purposes.
3) The designed hash functions on the basis of two-dimensional CA reveal strong avalanche effect, pointing out non-linearity and pseudorandom properties of the developed permutation schemes.

References

[1] Bertoni G. et al.: The Keccak sponge function family. http://keccak.noekeon.org/ [1.01.2018].
[2] Eddine A., Belfedhal K., Faraoun M.: Building secure and fast cryptographic hash functions using programmable cellular automata. J. of Computer and Information Technology CIT 4/2015, 317–328.
[3] Jamil N., Mahmood R., Z'aba R., Udzir N.I. Zukarnaen N.A.: A new cryptographic hash function based on crllular automata rule 30, 134 and omega-flip network. ICICN 27/2012, 163–169.
[4] Jeon J.Ch.: Analysis of hash functions and cellular automata based schemes. International Journal of Security and Applications 7(3)/2013, 303–316.
[5] Ostapov S., Val O., Yanushevsky S., Chyzhevsky D.: Cryptography on the Base of Cellular Automata. Internet in the Information Society – Chapter 6, Scientific Publishing University of Dabrowa Gornicza 2015.
[6] Paar Ch., Peltz J.: Understanding cryptography. Springer-Verlag, Berlin Heidelberg 2010.
[7] Robshaw M.J.B.: MD2, MD4, MD5, SHA, and Other Hash Functions. Technical Report TR-101/1994.
[8] Tanasyuk Yu., Melnychuk Kh., Ostapov S.: Development and research of cryptographic hash functions on the basis of cellular automata. Information Processing Systems 4(150)/2017, 122–127.
[9] Wolfram S.: A New Kind of Science Wolfram Media Inc., 2002, 1197 http://www.wolframscience.com/ nksonline/toc.html [1.01.2018].

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.

Prof. Sergey Ostapov
e-mail: s.ostapov@chnu.edu.ua

Head of Computer Systems Software Department, Physical, Technical and Computer Sciences Institute, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine.
Research interests and academic activities: cryptography, information security, big data analysis
The author of more than 150 publications in the given research field.

otrzymano/received: 10.10.2017 *przyjęto do druku/accepted: 15.02.2018*