sciendo
ISSN 1335-8871
MEASUREMENT SCIENCE REVIEW
Journal homepage: https://content.sciendo.com

Development of Modified Blum-Blum-Shub Pseudorandom Sequence Generator and its Use in Education

Shanshan Yu [1], Przystupa Krzysztof[2], Lingyu Yan[1], Volodymyr Maksymovych [3], Roman Stakhiv [3], Andrii Malohlovets [3], Orest Kochan[1,3]

[1]*Hubei University of Technology, Wuhan, China; yuss@hbut.edu.cn, ORCID: 0000-0002-5164-3132 (S.Y.), yanlingyu@hbut.edu.cn ORCID 0000-0003-2468-3881 (L.Y.).*
[2]*Department of Automation, Lublin University of Technology, Nadbystrzycka 36, Lublin 20-618, Poland; k.przystupa@pollub.pl, ORCID: 0000-0003-4361-2763*
[3] *Lviv Polytechnic National University, 12 Bandera Str., Lviv, Ukraine, volodymyr.maksymovych@gmail.com, roman.i.stakhiv@lpnu/ua, andrii.s.malohlovets@lpnu.ua, orest.v.kochan@lpnu.ua*

In information security systems, the algorithm of the Blum-Blum-Shub (BBS) generator, which is based on the use of a one-way function and is a cryptographically secure pseudorandom number generator, became widespread. In this paper, the problem of the analysis of modified algorithms of the BBS generator operation is considered to improve their statistical characteristics, namely, the sequence repetition period. It has been established that in order to improve the characteristics of the classic BBS algorithm, it is necessary to systematize approaches to change the recurrent equation itself, the relationship between the current and the previous members of the sequence. For this purpose, a generalized unified model of the modification of the classical BBS algorithm is derived. The repetition period with computational complexity were analyzed for classical algorithm and 80 proposed modifications. A gain in statistical characteristics is improved with slight increase in the required computing power of the system. The proposed modified BBS pseudorandom sequence generator can be used in training of students when teaching cryptographic stability of information security systems. The study of this generator combines the knowledge of students acquired in both digital electronics and mathematics.

Keywords: Pseudorandom sequence, pseudorandom sequence generators, one-way functions, Blum-Blum-Shub generators, computational complexity.

1. INTRODUCTION

Modern industry and science need more and more accurate sensors, equipment and systems [1]-[4]. Recent improvements often use the means of artificial intelligence [3], [5]. The new concept of the Internet of Things [6], [7] opens new possibilities. For instance, it makes the local systems [8] global [9] and their update does not need much extra resources. However, the issues of quality of service [7] as well as safety and security of data are now of primary importance [10]. That is why recently the methods and the devices that improve safety and security got considerable attention in scientific studies.

The pseudorandom sequence generators (PRSG), that include the pseudorandom number generators (PRNG) and the pseudorandom bit generators (PRBG), are used in various fields of technology in the information security systems. The PRNG can be classified according to different features: by the method of implementation (software, hardware), for resistance to disclosure (cryptographically secure, cryptographically insecure), or according to the algorithms on which they operate (based on the elementary functions, based on the shift registers, based on the one-way functions, etc.) [11]-[19].

In the information security systems, the algorithm of the Blum-Blum-Shub (BBS) generator (by the names of authors Lenore Blum, Manuel Blum, and Michael Shub), which is cryptographically secure PRNG (CSPRNG) [20]-[24], was proposed by the authors in 1986 [25] and became widespread [24], [26]-[33]. The principle of BBS generators is based on the use of the one-way function, which is crypto secured, and aimed, first of all, for the program implementation.

There are several modifications of the BBS generator algorithms [22], [34], [35], each of which is aimed at improving some of their characteristics, the main of which are: crypto security, statistical characteristics (including the repetition period of the output sequence), speed, and volume of key information (length of the key). In [23], an algorithm is proposed in which the number-module of the recurrence equation is modified. In this case, special attention is drawn to the possibility of increasing the speed by forming at each

DOI: 10.2478/msr-2022-0018

stroke not one, but a few bits of the original sequence. Changing the algorithm of the BBS generator may also be in the modification of the recurrent equation itself. This possibility is little investigated, which confirms the relevance of this work.

The purpose of the work is to study modified BBS generator algorithms. At the same time, the main attention is paid to the study of repetition periods of the initial sequence and the definition of the computational complexity.

In this paper, the problem of the analysis of modified algorithms of the BBS generator operation is considered to improve their statistical characteristics, namely, the sequence repetition period.

2. CLASSIC BBS ALGORITHM

The model of the classic Blum-Blum-Shub algorithm is based on the following equation [25]:

$$x_{n+1} = x_n^2 \, mod \, M, \tag{1}$$

where M is the key that results from the product of two numbers of Blum, p and q:

$$M = p \cdot q. \tag{2}$$

The Blum numbers are odd prime numbers for which the following condition is fulfilled [21]:

$$p \equiv 3(mod\ 4), q \equiv 3(mod\ 4), p \neq q. \tag{3}$$

The lower is the output from expression (4), the higher is the maximum value of the repetition period.

$$GCD(\varphi(p-1), \varphi(q-1)), \tag{4}$$

where *GCD(a, b)* – greatest common divisor for a and b, $\varphi(a)$ - Euler function for a.

At each step of the algorithm, the output data is obtained by taking either a parity bit or one or more least significant bits.

To use this model, it is necessary to set limits on which its input and output data are located. Since the equation (1) uses the result remaining from an integer division with the key M, at each step the input parameter and the result of iteration lie in the following ranges:

$$x_n \in [0: M), x_{n+1} \in [0: M). \tag{5}$$

Since the value of the key M does not change during iterations, equation (1) has one input parameter:

$$x_{n+1} = f(x_n), \tag{6}$$

then the repetition period of the classical algorithm lies in the next range:

$$P \in [1: M-1). \tag{7}$$

Given the expression (7), maximum value of the classic algorithm repetition period is the following:

$$P_{max} = (M-1) - 1 = M - 2. \tag{8}$$

The special Blum numbers are odd prime numbers for which the following condition is fulfilled [27], [28]:

$$\begin{aligned} p &= 2 \cdot p_1 + 1 = 4 \cdot p_2 + 3, \\ q &= 2 \cdot q_1 + 1 = 4 \cdot q_2 + 3, \end{aligned} \tag{9}$$

where p_1, p_2, q_1, q_2 are odd prime numbers too.

For the special key M, which uses the special Blum numbers, p and q, maximum value of the classic algorithm repetition period could be calculated from the following expression:

$$P_{max} = 2 \cdot p_2 \cdot q_2. \tag{10}$$

To evaluate the quality of modifications, we calculate the one iteration computational complexity of the classic BBS algorithm.

For the following form of expression –

$$x^y mod \, M, \tag{11}$$

the computational complexity is calculated as follows [28]:

$$O(M(n) \cdot \lceil log_2 y \rceil), \tag{12}$$

where n – the number of bits of the number x, *M(n)* – computational complexity of the selected multiplication algorithm.

Let the computational complexity of the chosen multiplication algorithm have the following value:

$$M(n) = n^2. \tag{13}$$

Since the maximum number of bits x is equal to the number of bits of the M key:

$$n = \lceil log_2 M \rceil. \tag{14}$$

The variable y from expression (12) takes the following value for equation (1):

$$y = 2. \tag{15}$$

Considering equations (13), (14), (15), the expression (12) takes the following form:

$$O(\lceil log_2 M \rceil^2). \tag{16}$$

Having carried out the analysis of the classical model (1), we can distinguish the following disadvantages:

- According to equation (16), the operations of squaring and remainder of integer division require a lot of system resources [38]. In addition, this dependence is quadratic.
- For each value in a sequence, only a limited number of bits can be used, which is calculated using the following equation [22]:

$$log_2 \, log_2 M. \tag{17}$$

- To ensure a minimum level of cryptographic keys necessary to use a length greater than 1024 bits. [39].

3. MODIFICATIONS OF THE BBS ALGORITHM

The classic algorithm could be improved through modification of its model (1).

Summarizing possible approaches for improving characteristics, they could be grouped by the location of the changes into the following groups:

Changing the dependency of the next member of the sequence from the previous one. The effectiveness of this method has been partially considered in the work [34].

Changing of the *M* key. Methods and effectiveness of this approach are considered in [35].

Improving the classic algorithm speed could be achieved through transformation of operations with their equivalents, for example, using the Montgomery algorithm, which was considered in [21], [35], [40].

Approaches that change the relationship between current and previous members of the sequence were systematized to improve the characteristics of the classic BBS algorithm. To do this, we present a generalized unified modification model based on equation (6) with the addition of parameters *a* and *b*:

$$x_{n+1} = f(x_n, a, b). \tag{18}$$

The parameter *a* has significant impact on the iteration result, so this parameter will be called "major". The value of this parameter is in the following range:

$$a \in [0: (M-1)^2). \tag{19}$$

The parameter *b* has less significant impact on the iteration result, so this parameter will be called "minor". The value of this parameter is in the following range:

$$b \in [0: log_2 M). \tag{20}$$

A complete unified modification model is obtained by extending equation (1) to equation (18). This model is presented below:

$$x_{n+1} = (x_n^2 + a + b)\, mod\, M. \tag{21}$$

From the expression (16), the computational complexity of one iteration of the modified algorithm, (21), is as follows:

$$O(\lceil log_2 M \rceil^2 + A(a) + B(b)), \tag{22}$$

where *A(a)* - computational complexity of the selected parameter *a*, *B(b)* - computational complexity of the selected parameter *b*. Computational complexities *A(a)* and *B(b)* include the operation of adding to in equation (21).

Consider the effect of the major parameter considering that the minor parameter *b* is equal to b_0. For the given mathematical model, the parameter has the following value b_0:

$$b_0 = 0. \tag{23}$$

Given the condition (23), equation (21) takes the following form:

$$x_{n+1} = (x_n^2 + a)\, mod\, M. \tag{24}$$

Let the major parameter a_1 be determined by the following equation:

$$a_1 = x_n. \tag{25}$$

Based on the principles (24) and (25), we can determine the range of possible values of the input parameters used in the formation of this model. Since the principles (24) and (25) are generalized to the expression (6), the range of possible values of the input parameters is determined by the expression (5). According to this, repetition period corresponds to the repetition period of the classic algorithm (7).

Since (25) does not contain additional operations and the value of x_n is available, the computational complexity of one iteration is equal to the computational complexity of the operation of adding parameter *A*, which is in the range (19):

$$A(a_1) = \lceil log_2(2 \cdot M) \rceil. \tag{26}$$

Let the major parameter a_2 be determined by the following equation:

$$a_2 = x_{n-1}, \tag{27}$$

where x_{n-1} – result value from the previous iteration of the algorithm. The range of possible values of parameter x_{n-1} lies within (5). Considering the principles (24) and (27), current iteration result depends on the current and previous iteration values (28).

$$x_{n+1} = f(x_n, x_{n-1}). \tag{28}$$

As the result of the operation depends on a set of input parameters, the repetition period of the modified algorithm is in the following range:

$$P \in [1:(M-1)^2). \tag{29}$$

Given the expression (29), the maximum value of the classic algorithm repetition period is as follows:

$$P_{max} = (M-1)^2 - 1. \tag{30}$$

Since receiving parameter a_2 similar to parameter a_1 on the side of the computational complexity, as well as the ranges of their values lying within (5), the complexity of operation a_2 corresponds to the expression (26).

Based on equation (21), we introduce the concept of an intermediate value x_{temp}, which is obtained by dividing the expression (21) into static and variable input data (31) (32). The intermediate value of x_{temp} is used for the modifications a_3, a_4, a_5, a_6.

$$x_{temp} = x_n^2 + a + b, \tag{31}$$

$$x_{n+1} = x_{temp}\, mod\, M. \tag{32}$$

Given equations (5), (19) and (20), the intermediate value lies in the following range:

$$x_{temp} \in [0: 2 \cdot (M-1)^2 + 2 \cdot log_2 M). \quad (33)$$

Let the major parameter a_3 be determined by the following equation:

$$a_3 = x_{temp}, \quad (34)$$

where x_{temp} – intermediate value from the previous iteration of the algorithm.

Given the previous sentence and equations (31) and (32), x_{n-1} and x_{temp} are separated by integer division by static key M, so the following statement is true:

$$x_{n+1} = f(x_n, x_{temp}) \approx f(x_n, x_{n-1}). \quad (35)$$

Since equation (35) is equivalent to equation (28), the repetition period of the a_3, a_4, a_5, a_6 modification of the algorithm lies in the range (29).

Let the major parameter a_4 be determined by the following equation:

$$a_4 = (x_{temp} << log_2 M) >> log_2 M, \quad (36)$$

where $a<<b$ – the left logical shift number a of b bits, $a>>b$ – the right logical shift number a of b bits.

Let the major parameter a_5 be determined by the following equation:

$$a_5 = x_{temp} >> log_2 M. \quad (37)$$

Let the major parameter a_6 be determined by the following equation:

$$a_6 = (x_{temp} >> log_2 M) << log_2 M. \quad (38)$$

The parameters a_3, a_4, a_5, a_6 only select the resulting bits, then the parameters a_3, a_4, a_5, a_6 are similar to the parameter on the side of the computational complexity. Since the ranges of the values of the parameters a_4, a_5 lie within (5), the computational complexity of a_4, a_5 corresponds to the expression (26). Since the ranges of the values of the parameters a_3, a_6 lie within (33), the computational complexity of a_3, a_6 corresponds to the expression (26).

The modifications a_7, a_8 use combinations x_n and x_{n-1}, and therefore, they can be generalized to equation (28), hence, it follows that the repetition period of the algorithm modifications lies in (29).

Let the major parameter a_7 be determined by the following equation:

$$a_7 = x_n + x_{n-1}. \quad (39)$$

Since (39) contains an additional operation and values x_n and x_{n-1} are available, the computational complexity of one iteration is equal to the computational complexity of the addition operations x_n and x_{n-1}, as well as the parameter a_7, (41), the value of which, considering the limits of the input parameters (5), lies within (40).

$$a_7 \in [0: 2 \cdot (M-1)). \quad (40)$$

$$A(a_7) = 2 \cdot \lceil log_2 M \rceil. \quad (41)$$

Let the major parameter a_8 be determined by the following equation:

$$a_8 = x_n \& x_{n-1}, \quad (42)$$

where "&" – the bitwise multiplication operation, "*AND*".

Since obtaining the parameter a_8 is analogous to parameter a_7 on the side of the computational complexity, but the result of the calculation of a_8 lies in (5), then the computational complexity of one iteration is equal to the complexity:

$$A(a_8) = \lceil log_2 2 \cdot M \rceil + \lceil log_2 2 \cdot M \rceil = 2 \cdot \lceil log_2 2 \cdot M \rceil. \quad (43)$$

Consider the effect of the minor parameter considering that the major parameter a is equal to a_0, which has the following value:

$$a_0 = 0. \quad (44)$$

Given the condition (44), equation (21) takes the following form:

$$x_{n+1} = (x_n^2 + b) \, mod \, M. \quad (45)$$

Let the minor parameter b_1 be determined by the following equation:

$$b_1 = \sum_{i=0}^{log_2 M} x_n[i], \quad (46)$$

where $x_n[i]$ – the i-th bit of x_n number.

Considering the above, equation (45), and also equation (46), the range of possible values lies in (5). Since equation (46) is generalized to equation (6), the repetition period corresponds to the repetition period of the classic algorithm, equation (7).

Since the result (46) lies in (20) and the value of x_n is available, the computational complexity of one iteration is equal to the computational complexity (14) - the operation of adding a bit of the number x_n to an amount, the value of which lies within (5):

$$B(b_1) = \lceil log_2 2 \cdot M \rceil + (\lceil log_2 M \rceil - 1) \cdot \lceil log_2 M \rceil. \quad (47)$$

Let the minor parameter b_2 be determined by the following equation:

$$b_2 = \sum_{i=0}^{log_2 M} x_{n-1}[i]. \quad (48)$$

The range of possible values lies in (20). Given equations (45) and (48), equation (18) takes the form (28).

Given the ranges of input values: equations (5) and (20), - the repetition period for this algorithm modification lies in the

following range:

$$P \in [1{:}(M-1)(log_2 M - 1)). \quad (49)$$

Since receiving the parameter b_2 similar to the parameter b_1 on the side of the computational complexity, the complexity of operation b_2 corresponds to the expression (47).

The minor modifications b_3 and b_4 use the remainder of the integer division of the module 2, hence the range of possible values lies in the next range:

$$b_3, b_4 \in [0{:}\,2). \quad (50)$$

Let the minor parameter b_3 be determined by the following equation:

$$b_3 = \left(\sum_{i=0}^{log_2 M} x_n[i]\right) mod\ 2. \quad (51)$$

Considering the above, equation (45), and also equation (51), the range of possible values lies in (5). Since equation (51) is generalized to equation (6), the repetition period corresponds to the repetition period of the classic algorithm, equation (7).

Let the minor parameter b_4 be determined by the following equation:

$$b_4 = \left(\sum_{i=0}^{log_2 M} x_{n-1}[i]\right) mod\ 2. \quad (52)$$

Given equations (45) and (52), equation (18) takes the form (28).

Given the ranges of input values: equations (5) and (50), - the repetition period for this algorithm modification lies in the following range:

$$P \in [1{:}\,2 \cdot (M-1)). \quad (53)$$

Since the reception of the parameters b_3 and b_4 is similar to the parameter b_1 on the side of the complexity of the calculation, but the result of computing b_3 and b_4 lies in (50), the computational complexity of one iteration is as follows:

$$B(b_3) = B(b_4) = \lceil log_2 2 \cdot M \rceil + (\lceil log_2 M \rceil - 1). \quad (54)$$

The minor modifications b_5, b_6, b_7, b_8 use the sum of bits on even or odd positions, and therefore the range of possible values lies in the next range:

$$b_5, b_6, b_7, b_8 \in [0{:}\,\lceil (log_2 M)/2 \rceil). \quad (55)$$

Let the minor parameter b_5 be determined by the following equation:

$$b_5 = \sum_{i=0}^{\lceil (log_2 M)/2 \rceil} x_n[2 \cdot i]. \quad (56)$$

Considering the above, equation (45), and also equation (56), the range of possible values lies in (5). Since equation (56) is generalized to equation (6), the repetition period corresponds to the repetition period of the classic algorithm, equation (7).

Let the minor parameter b_6 be determined by the following equation:

$$b_6 = \sum_{i=0}^{\lceil (log_2 M)/2 \rceil} x_{n-1}[2 \cdot i]. \quad (57)$$

Given equations (45) and (57), equation (18) takes the form (28).

Given the ranges of input values: equations (5) and (55), - the repetition period for this algorithm modification lies in the following range:

$$P \in [1{:}(M-1) \cdot \lceil (log_2 M)/2 \rceil). \quad (58)$$

Let the minor parameter b_7 be determined by the following equation:

$$b_7 = \sum_{i=0}^{\lceil (log_2 M)/2 \rceil} x_n[2 \cdot i + 1]. \quad (59)$$

Considering the above, equation (45), and also equation (59), the range of possible values lies in (5). Since equation (59) is generalized to equation (6), the repetition period corresponds to the repetition period of the classic algorithm, equation (7).

Let the minor parameter b_8 be determined by the following equation:

$$b_8 = \sum_{i=0}^{\lceil (log_2 M)/2 \rceil} x_{n-1}[2 \cdot i + 1]. \quad (60)$$

Given equations (45) and (60), equation (18) takes the form (28).

Given the ranges of input values: equations (5) and (20), - the repetition period for this algorithm modification lies in the range (58).

Since the acquisition of parameters b_5, b_6, b_7, b_8 is similar to the parameter b_1 on the side of the complexity of the calculation, but the result of computing b_5, b_6, b_7, b_8 lies in (55), the computational complexity of one iteration is as follows:

$$B(b_5) = B(b_6) = B(b_7) = B(b_8) = \lceil log_2 2 \cdot M \rceil + (\lceil log_2 M\,/2 \rceil - 1) \cdot \lceil log_2 M\,/2 \rceil. \quad (61)$$

4. SIMULATION RESULTS

For the experiment, all the keys in length from 5 to 9 bits were chosen, that are satisfying the condition (3), and are presented in Table 1.

The initial values, seeds s, for each key were taken according to the following equation [21]:

$$s \in [2{:}\,M-1]. \quad (62)$$

Since the models depend on a pair of values, (28), for the calculation of the repetition period, it was considered that the repetition period is a sequence in which a pair of values, current and previous values of the iterations, were repeated.

Due to (62), and the classic algorithm, and the number of modifications - 80, 739 206 periods were found.

Table 1. M keys for the BBS generator.

#	Blum number		Key length, bits	Key value	Key GCD value
	P	*Q*			
1	3	7	5	21	1
2	3	11	6	33	1
3	3	19	6	57	1
4	3	23	7	69	1
5	7	11	7	77	2
6	3	31	7	93	1
7	3	43	8	129	1
8	7	19	8	133	2
9	3	47	8	141	1
10	7	23	8	161	2
11	3	59	8	177	1
12	3	67	8	201	1
13	11	19	8	209	2
14	3	71	8	213	1
15	11	23	8	253	2
16	7	43	9	301	2
17	3	103	9	309	1
18	3	107	9	321	1
19	7	47	9	329	2
20	11	31	9	341	4
21	3	127	9	381	1
22	3	131	9	393	1
23	7	59	9	413	2
24	3	139	9	417	1
25	19	23	9	437	2
26	3	151	9	453	1
27	7	67	9	469	2
28	11	43	9	473	4

Since there are 81 combinations for 28 keys, we need to calculate relative average value of repetition period. Given equations (8) and (62), the relative average value of the repetition period for one key will be calculated as follows:

$$P_{avg}(M) = \frac{\sum_0^{M-3} P(M,s_i)}{(M-3)\cdot(M-2)}, \tag{63}$$

where P(M, s) – the repetition period for key M and initial value s. Relative average value of repetition period for one modification will be calculated as follows:

$$P_{avgMod} = \frac{\sum_0^{n-1} P_{avg}(M)}{n}, \tag{64}$$

where *n* - number of keys.

Table 2. contains PavgMod that were calculated for each modification.

Regarding condition (9), there is only one special key in Table 1. - row 15. Its maximal theoretic repetition period per equation (10) is 20. Maximal theoretic value according to equation (63) is 7.9861 %. But its empirical value in input range defined by (62) is 6.9919 %.

From Table 2., 76 modifications are better than the classic algorithm, but 4 modifications are worse. Also, the results for the 8 modifications are significantly higher. These modifications relate to the expression (28), whose repetition period is in the range (29). Given the previous sentence and equation (30), relative average value of the repetition period for one key will be calculated as follows:

$$P_{avg}(M) = \frac{\sum_0^{M-3} P(M,s_i)}{(M-3)\cdot((M-1)^2-1)}. \tag{65}$$

PavgMod for equations (63) and (65) are listed in Table 3.

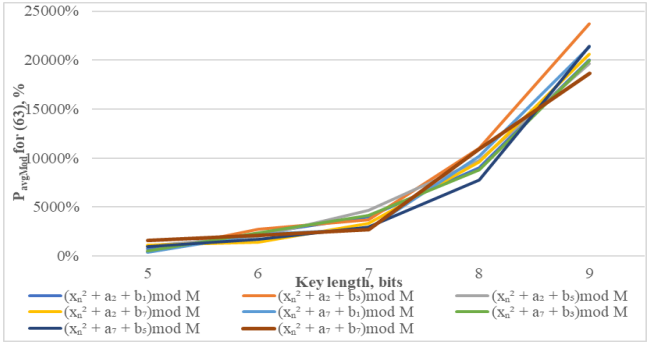

Fig.1. The average relative value of the period to (7) for the length of the keys.

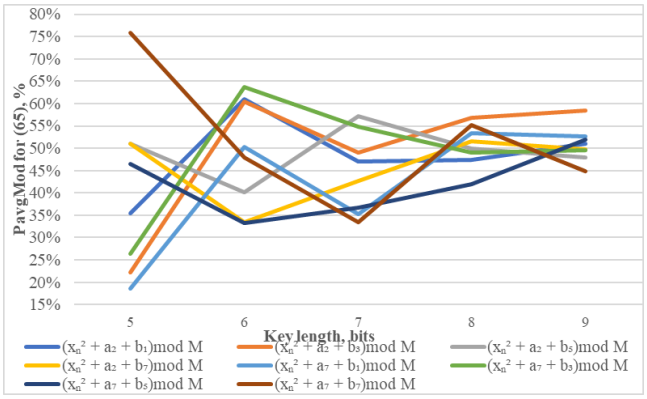

Fig.2. The average relative value of the period to (29) for the length of the keys.

Since output for equation (30) contains quadratic dependency on key, PavgMod for (63) will have it too, this could be seen with split PavgMod per key bits length. The result of previous sentence is available in Fig.1.

PavgMod for (65) per key bits length is available in Fig.2.

Table 2. and Table 3. show that the most successful are the modifications that use combinations of the parameters that consist of the a_2 and a_7 major parameters together with the b_1, b_3, b_5, b_7 minor parameters.

From the foregoing and equations (27) and (39) for the a_2 and a_7 major parameters, it follows that the major parameter must directly depend on the previous iteration value, xn-1.

From the foregoing and equations (46), (51), (56), (59) for the b_1, b_3, b_5, b_7 minor parameters, it follows that the minor parameter must be a combination of the current value, x_n.

Table 2. Average relative repetition period of the period for the classic algorithm and modifications of the BBS generator.

#	Index of		PavgMod, %	#	Index of		PavgMod, %	#	Index of		PavgMod, %
	a	*b*			*a*	*b*			*a*	*b*	
1	0	0	7.2952	28	3	0	6.4960	55	6	0	46.9421
2	0	1	10.1275	29	3	1	9.9349	56	6	1	88.6475
3	0	2	20.6734	30	3	2	20.9172	57	6	2	98.3819
4	0	3	7.1232	31	3	3	8.8297	58	6	3	64.8248
5	0	4	9.6952	32	3	4	12.2593	59	6	4	82.6788
6	0	5	8.8592	33	3	5	9.6094	60	6	5	78.5479
7	0	6	16.3713	34	3	6	18.1217	61	6	6	90.8849
8	0	7	7.9332	35	3	7	9.8660	62	6	7	78.4829
9	0	8	14.7875	36	3	8	17.4116	63	6	8	91.5243
10	1	0	5.7875	37	4	0	6.4960	64	7	0	374.6296
11	1	1	9.0671	38	4	1	9.9349	65	7	1	14022.2353
12	1	2	21.0732	39	4	2	20.9172	66	7	2	182.1908
13	1	3	8.1752	40	4	3	8.8297	67	7	3	13037.6050
14	1	4	11.3888	41	4	4	12.2593	68	7	4	142.0743
15	1	5	9.1021	42	4	5	9.6094	69	7	5	13206.1112
16	1	6	17.6384	43	4	6	18.1217	70	7	6	154.7777
17	1	7	9.2927	44	4	7	9.8660	71	7	7	13018.1933
18	1	8	17.7612	45	4	8	17.4116	72	7	8	190.6706
19	2	0	214.4532	46	5	0	77.2653	73	8	0	263.3034
20	2	1	13103.4792	47	5	1	72.2712	74	8	1	307.3031
21	2	2	158.6466	48	5	2	101.3708	75	8	2	116.1185
22	2	3	15546.2871	49	5	3	74.9338	76	8	3	344.3523
23	2	4	165.7766	50	5	4	86.6606	77	8	4	152.7034
24	2	5	13252.9975	51	5	5	90.9435	78	8	5	326.2900
25	2	6	144.2575	52	5	6	103.4514	79	8	6	112.3834
26	2	7	13503.6419	53	5	7	82.6686	80	8	7	307.8142
27	2	8	176.3226	54	5	8	96.2355	81	8	8	136.0933

Table 3. Average relative values of BBS modification period with quadratic character.

#	Index of		PavgMod for (63), %	PavgMod for (65), %
	a	b		
71	7	7	13018.1933	48.6041
67	7	3	13037.6050	49.9731
20	2	1	13103.4792	49.4746
69	7	5	13206.1112	45.8159
24	2	5	13252.9975	49.0195
26	2	7	13503.6419	48.7998
65	7	1	14022.2353	50.2448
22	2	3	15546.2871	56.1337

5. CONCLUSIONS

In the given work, the analysis of modified algorithms work of BBS generators is carried out. As a result, it has been established that in order to improve the characteristics of the classic BBS algorithm, it is necessary to systematize approaches to change the relationship between the current and the previous members of the sequence. For this purpose, the generalized unified model of the modification of the classic BBS algorithm is derived.

The proposed model made it possible to improve the statistical characteristics of the classic BBS algorithm, in particular, the sequence repetition period.

For the analysis of the sequence repetition period based on the classic algorithm and the 80 proposed modifications, 739 206 tests were performed. As a result of this research, it was established:

1. For the keys of 5 to 9 bits the repetition period is on average 7.2952 % of the key value.
2. 95 % of the proposed modifications have a more efficient repetition period compared to the classic algorithm.
3. 10 % of the modifications of the classic BBS algorithm have a much longer period of repetition, which has quadratic dependency.
4. A gain in statistical characteristics is improved with slight increase in the required computing power of the system.

Implementation and study of the proposed modified BBS pseudorandom sequence generator by students of the direction of training of cybersecurity allows them to solve information security problems, in particular, generating cryptographic keys, hashing passwords, generating network protocol keys, and user authentication.

REFERENCES

[1] Krolczyk, G., Gajek, M., Legutko, S. (2013). Predicting the tool life in the dry machining of duplex stainless steel. *Eksploatcja i Niezawodnosc-Maintenance and Reliability*, 15, 62-65.

[2] Jun, S., Kochan, O. (2015). Common mode noise rejection in measuring channels. *Instruments and Experimental Techniques*, 58 (1), 86-89.

[3] Glowacz, A. (2021). Thermographic fault diagnosis of ventilation in BLDC motors. *Sensors*, 21 (21), 7245. https://doi.org/10.3390/s21217245

[4] Jun, S., Kochan, O., Kochan, R. (2016). Thermocouples with built-in self-testing. *International Journal of Thermophysics*, 37 (4), 1-9. https://doi.org/10.1007/s10765-016-2044-2

[5] Wang, J., Przystupa, K., Maksymovych, V., Stakhiv, R., Kochan, O. (2020). Computer modelling of two-level digital frequency synthesizer with Poisson probability distribution of output pulses. *Measurement Science Review*, 20 (2), 65-72. https://doi.org/10.2478/msr-2020-0009

[6] Greengard, S. (2015). *The Internet of Things*. MIT Press, ISBN 9780262527736.

[7] Jun, S., Przystupa, K., Beshley, M., Kochan, O., Beshley, H., Klymash, M., Pieniak, D.A. (2020). Cost-efficient software based router and traffic generator for simulation and testing of IP network. *Electronics*, 9 (1), 40. https://doi.org/10.3390/electronics9010040

[8] Su, J., Kochan, O., Wang, C., Kochan, R. (2015). Theoretical and experimental research of error of method of thermocouple with controlled profile of temperature field. *Measurement Science Review*, 15 (6), 304-312. https://doi.org/10.1515/msr-2015-0041

[9] Fraczyk, A., Jaworski, T., Urbanek, P., Kucharski, J. (2014). The design for a smart high frequency generator for induction heating of loads. *Przegląd Elektrotechniczny [Electrical Review]*, 2, 20-23. DOI 10.12915/pe.2014.02.6.

[10] Song, W., Beshley, M., Przystupa, K., Beshley, H., Kochan, O., Pryslupskyi, A., Su, J. (2020). A software deep packet inspection system for network traffic analysis and anomaly detection. *Sensors*, 20 (6), 1637. https://doi.org/10.3390/s20061637

[11] Maksymovych, V., Shabatura, M., Harasymchuk, O., Karpinski, M., Jancarczyk, D., Sawicki, P. (2022). Development of additive Fibonacci generators with improved characteristics for cybersecurity needs. *Applied Sciences*, 12 (3), 1519. https://doi.org/10.3390/app12031519

[12] Mandrona, M., Maksymovych, V., Harasymchuk, O., Kostiv, Y. (2014). Generator of pseudorandom bit sequence with increased cryptographic security. *Metallurgical and Mining Industry,* 5, 25-29.

[13] Maksymovych, V., Harasymchuk, O., Karpinski, M., Shabatura, M., Jancarczyk, D., Kajstura, K. (2021). A new approach to the development of additive Fibonacci generators based on prime numbers. *Electronics*, 10, 2912. https://doi.org/10.3390/electronics10232912

[14] Mandrona, M., Maksymovych, V. (2017). Comparative analysis of pseudorandom bit sequence generators. *Journal of Automation and Information Sciences*, 49 (3), 78-86. https://doi.org/10.1615/JAutomatInfScien.v49.i3.90

[15] Maksymovych, V., Harasymchuk, O., Mandrona, M. (2017). Designing generators of Poisson pulse sequences based on the additive Fibonacci generators. *Journal of Automation and Information Sciences,* 49 (12), 1-12.

[16] Maksymovych, V., Mandrona, M., Garasimchuk, O., Kostiv, Y. (2016). A study of the characteristics of the fibonacci modified additive generator with a delay. *Journal of Automation and Information Sciences*, 48 (11), 76-82.

[17] Maksymovych, V., Harasymchuk, O., Opirskyy, I. (2018). The designing and research of generators of Poisson pulse sequences on base of Fibonacci modified additive generator. In *Advances in Computer Science for Engineering and Education*. Springer, 43-53. https://doi.org/10.1007/978-3-319-91008-6_5

[18] Maksymovych, V., Mandrona, M., Harasymchuk, O. (2020). Dosimetric detector hardware simulation model based on modified additive Fibonacci generator. In *Advances in Computer Science for Engineering and Education II.* Springer, Vol. 938, 162-171. https://doi.org/10.1007/978-3-030-16621-2_15

[19] Maksymovych, V., Mandrona, M., Kostiv, Y., Harasymchuk, O. (2017). Investigating the statistical characteristics of Poisson pulse sequences generators constructed in different ways. *Journal of Automation and Information Sciences,* 49 (10), 11-19.

[20] Agerblad, J., Andersen, M. (2013). *Provably secure pseudo-random generators*. Thesis, School of Computer Science and Communication, The Royal Institute of Technology, Stockhol, Sweden. http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-134830.

[21] Junod, P. (1999). *Cryptographic secure pseudo-random bits generation: The Blum-Blum-Shub generator*. http://crypto.junod.info/bbs.pdf

[22] Shrestha, B. (2016). *Multiprime Blum-Blum-Shub pseudorandom number generator*. Thesis, Naval Postgraduate School, Monterey, CA. https://apps.dtic.mil/dtic/tr/fulltext/u2/1030047.pdf

[23] Divyanjali, Ankur, Pareek, V. (2014). An overview of cryptographically secure pseudorandom number generators and BBS. In *IJCA Proceedings of the International Conference on Advances in Computer Engineering and Applications ICACEA*, 19-28.

[24] Sodhi, G.K., Gaba, G.S. (2017). DNA and Blum Blum Shub random number generator based security key generation algorithm. *International Journal of Security and its Applications*, 11 (4), 1-10. http://dx.doi.org/10.14257/ijsia.2017.11.4.01

[25] Blum, L., Blum, M., Shub, M. (1983). Comparison of two pseudo-random number generators. In *Advances in Cryptology: Proceedings of Crypto 82*. Springer, 61-78. http://dx.doi.org/10.1007/978-1-4757-0602-4_6

[26] Kapur, V., Paladi, S.T., Dubbakula, N. (2015). Two level image encryption using pseudo random number generators. *International Journal of Computer Applications*, 115 (12), 1-4. http://dx.doi.org/10.5120/20200-2446

[27] Aissa, B., Khaled, M., Lakhdar, G. (2014). Implementation of Blum Blum Shub generator for message encryption. In *Proceedings of the International Conference on Control, Engineering and Information Technology (CEIT'14)*. IPCO, 118-123.

[28] Lopez, P., Millan, E., van der Lubbe, J., Entrena, L. (2010). Cryptographically secure pseudorandom bit generator for RFID tags. In *2010 International Conference for Internet Technology and Secured Transactions*. IEEE, 1-6.

[29] Panda, A., Ray, K. (2018). Design and FPGA prototype of 1024-bit Blum-Blum-Shub PRBG architecture. In *2018 IEEE International Conference on Information Communication and Signal Processing (ICICSP)*. IEEE, 38-43, DOI 10.1109/ICICSP.2018.8549715.

[30] Rock, A. (2005). *Pseudorandom number generators for cryptographic applications*. Thesis, Universität Salzburg, Salzburg, Austria. https://cutt.ly/sPSuTVt

[31] Hassan, N. (2017). Color images encryption using cipher system with different types of random number generator. *International Journal of Innovative Research in Computer and Communication Engineering,* 5 (5).

[32] Omorog, C.D., Gerardo, B.D., Medina, R.P. (2018). Enhanced pseudorandom number generator based on Blum-Blum-Shub and elliptic curves. In *2018 IEEE Symposium on Computer Applications and Industrial Electronics (ISCAIE)*. IEEE, 269-274, DOI 10.1109/ISCAIE.2018.8405483.

[33] Siahaan, A.P.U. (2016). Blum Blum Shub in generating key in RC4. *The International Journal of Science & Technoledge,* 4 (10), 1-5.

[34] Malohlovets, A., Maksymovych, V. (2017). Research of methods for improving statistical characteristics for cryptographically strong BBS pseudorandom number and bit generators. In *Proceedings of the 6th International Academic Technical Conference "Information and Information Systems Security"*, Lviv, Ukraine, 73-74.

[35] Gawande, K., Mundle, M. (1999). *Various implementations of Blum Blum Shub pseudo-random sequence generator*. http://koclab.cs.ucsb.edu/teaching/cren/project/2005past/gawande-mundle.pdf

[36] Blum, L., Blum, M., Shub, M. (1986). A simple unpredictable pseudorandom number generator. *SIAM Journal on Computing,* 15 (2), 364-383. https://doi.org/10.1137/0215025

[37] Markov, I., Saeedi, M. (2012). Constant-optimized quantum circuits for modular multiplication and exponentiation. *Quantum Information & Computation*, 12 (5-6), 1-28.

[38] Sewak, K., Rajput, P., Panda, A.K. (2012). FPGA implementation of 16 bit BBS and LFSR PN sequence generator: A comparative study. In *2012 IEEE Students' Conference on Electrical, Electronics and Computer* Science. IEEE, 769-773. DOI 10.1109/SCEECS.2012.6184758.

[39] Sidorenko, A., Schoenmakers, B. (2005). Concrete security of the Blum-Blum-Shub pseudorandom generator. In *Cryptography and Coding: 10th IMA International Conference*. Springer, Vol. 3796, 355-375. https://doi.org/10.1007/11586821_24

[40] Malohlovets, A., Maksymovych, V. (2016). Research of the methods for improving performance for cryptographically strong BBS pseudorandom bit sequences generators. In *Proceedings of the 6th International Youth Science Forum "Litteris et Artibus"*, Lviv, Ukraine, 54-55.

Received January 01, 2022
Accepted February 28, 2022