Iraqi Journal of Science, 2022, Vol. 63, No. 5, pp: 2230-2242
DOI: 10.24996/ijs.2022.63.5.35

ISSN: 0067-2904

A Pseudo-Random Number Generator Based on New Hybrid LFSR and LCG Algorithm

Balsam Abdulkadhim Hameedi*[1], Dr Anwar Abbas Hattab [1], Dr Muna M. Laftah [2]
[1] *Department of Computer Science, College of Education, University of Mustansiriyah*
[2] *Department of Computer Science, College of Education for Women, University of Baghdad*

Received: 14/4/2021 Accepted:19/7/2021 Published: 30/5/2022

Abstract

In many areas, such as simulation, numerical analysis, computer programming, decision-making, entertainment, and coding, a random number input is required. The pseudo-random number uses its seed value. In this paper, a hybrid method for pseudo number generation is proposed using Linear Feedback Shift Registers (LFSR) and Linear Congruential Generator (LCG). The hybrid method for generating keys is proposed by merging technologies. In each method, a new large in key-space group of numbers were generated separately. Also, a higher level of secrecy is gained such that the internal numbers generated from LFSR are combined with LCG (The adoption of roots in non-linear iteration loops). LCG and LFSR are linear structures and outputs of these Random Number Generators (RNGs) are predictable, while the proposal avoids this predictable nature. The results were tested in terms of randomness, in terms of the correlation between the keys and the effect of changing the initial state on the generated keys and the results of the tests showed that they had successfully passed the tests and resist brute force and differential attack.

Keywords: LFSR, LCG, pseudo number generator, NIST, Hamming distance Correlation test

مولد ارقام عشوائي يعتمد على خوارزمية هجينة من دمج (LFSR) مع (LCG)

بلسم عبد الكاظم حميدي [1*] ، انوار عباس حطاب[1]، منى مجيد لفتة[2]
[1]قسم علوم الحاسبات، كلية التربيه ، الجامعة المستنصريه، بغداد، العراق
[2]قسم علوم الحاسبات، كلية التربيه بنات ، جامعة بغداد، بغداد، العراق

الخلاصه

في العديد من التطبيقات نحتاج توليد ارقام عشوائية مثل المحاكاة والتحليل العددي وبرامج الحاسوب واتخاذ القرار وبرامج الالعاب والترفيه وتشفير البيانات وعادة هذه الارقام العشوائية تحتاج قيم ابتدائية. في هذا البحث ، تم اقتراح طريقة هجينة لتوليد سلسلة من الأرقام باستخدام دمج مسجل التغذية الراجعة الخطي (LFSR) والمولد المطابق الخطي (LCG). هذه الطريقة المقترحة تتم من خلال دمج اكثر من تقنية والحصول على نتائج افضل عن طريق انشاء مجموعة جديدة من الأرقام كبيرة في في كل خطوة من (LFSR) ,وتطبيق عليها معادلة الطريقة الثانية (LCG) مع بعض التعديل لكي يتم الحصول على مستوى أعلى من السرية بحيث يتم دمج الأرقام الداخلية الناتجة من LFSR مع LCG (اعتماد الجذور في حلقات التكرار غير الخطية). تم

*Email:balsamKadom278@uomustansiryah.edu.iq

اختبار النتائج من حيث العشوائية ، من حيث الارتباط بين المفاتيح وتأثير تغيير الحالة الأولية على المفاتيح المتولدة وأظهرت نتائج الاختبارات أنها نجحت في اجتياز الاختبارات ومقاومة المهاجمة التي تجرب كل الاحتمالات (Brute Force) والهجوم التفاضلي (Differential Attack).

1. Introduction

In communication and digital computing applications, pseudo-random number sequences are usually used. Bits pattern must never be replicated in a completely random sequence. However, such a sequence of practical systems is extremely difficult to generate. Many apps need random yet user-predictable disappearance [1]. A pseudo-random sequence complies with the random requirements; moreover, the whole sequence is continuously repeated. The Linear Congruential Generator (LCG) which can be used to produce such sequences, minimum memory (usually 32 or 64 bits) is required to retain the state. For multiple independent streams, this makes it valuable [2]. Linear Feedback Shift Register (LFSR) is a shift register with a feedback path linearly related to the nodes using XOR gates. LFSRs are more popular because of their compactness and simple design [3]. Many previous works related to the topic of generating random numbers based on LFSR and LCG. The proposed generator in paper [4] is based on LFSR and extracts entropy from sound sources. PRNG is impervious to large-scale attacks on pseudo-random number generators. The research presented in [5] is a comparison between pseudo-random number generators (PRNGs) such as LFSR, LCGs, and combined LFSR and Cellular Automata Shift Register (CASR). The last one used to avoid the predictable nature of the PRNGs, it considered a non-linear combination generator which is combined the LFSR and CASR. It generates random numbers better than LFSR and LCG separately. A simple method is proposed in [6] for designing the Chaotic Linear Feedback Register (CLFSR). The key concept of the proposed method is the LFSR output modified by a stream bit with exclusively-or (XOR) to reduce the linearity and the repetition of the LFSR output using a chaotic map system. Paper [7] propose a method of the coupling of two newly formed variable input LCGs that generates pseudo-random bit at every uniform clock rate, which attains maximum length sequence, and reduces one comparator area as compared to the dual coupled-linear congruential generator Dual CLCG architecture. It is the replacing of constant parameters with variable inputs in the LCG equation and further used the concept of coupling as post-processing between two variable-input LCG (Vi-LCGs) for generating pseudorandom bit at every iteration.

In this paper, a hybrid algorithm is proposed to generate random numbers based on combining the technique LFSR with LCG to generate a new random number. The main contribution of the proposal is mixing interior numbers of LFSR with LCG (variables a and b be not constants), root value of LCG is used for increasing complexity (the security is increasing). It also does not require large material resources in terms of storage and processing so it is suitable for smart device applications.

2. Generators of Linear Congruential:

The LCG proposed by Lehmer referred to as the linear congruential technique [8], is widely used method for generating pseudorandom numbers. The method is parametrized with the following four numbers:

The modulus m where $m > 0$, the multiplier $0 < a < m$, c the increment $0 <= c < m$, X_0 the starting value, or seed $0 <= X_0 < m$. The arrangement of arbitrary numbers $\{X_n\}$ is obtained through the following iterative equation:

$$X_{n+1} = (aX_n + c)\ mod\ m \ldots (1)$$

This technique generates a string of integers with each integer falling within the range $0 <= X_n < m$ if m, a, c, and X_0 are all integers. To build a good random number generator, the values for a, c, and m are critical.

3. Linear Feedback Shift Registers:

The input bit of an LFSR shift register is the output of a linear function of two or more of

its previous states. An LFSR of length m is made up of m stages numbered 0, 1, ..., m, each of which may store one bit, and a clock that controls data interchange. The shift register would be initialized with a vector with the elements s_0, ..., s_m. the steps are as follow: the output includes s_i (the content of stage 0), the content of stage I is transferred to stage i1, for 1im1, by XORing a subset of the content of m stages, the new content (the feedback bit) of stage m1 would be produced. There are many conceivable setups; in Figure 1 shows a simple one that starts with an input of all 1s and is very simple to implement in software and hardware. An LFSR of this type will never contain only 0s and will stop if a binary string containing only 0s is input into it.

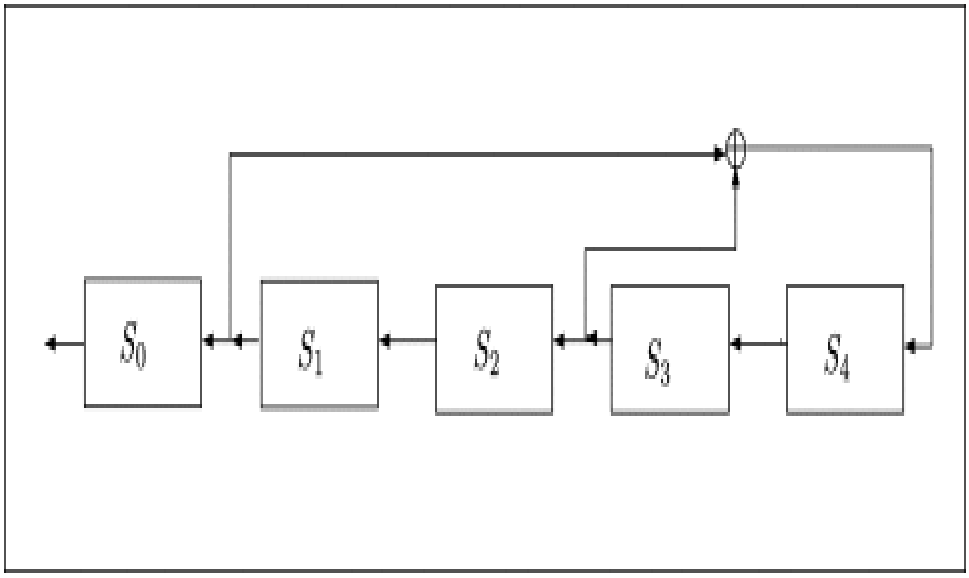

Figure 1- Linear feedback shift register simple example

4. Methodology:

In this paper, a hybrid algorithm was proposed to generate random numbers based on combining the technique LFSR with LCG and generating a new series whose randomness is satisfying the randomness requirement and gives high flexibility through controlling the elementary parameters. To generate a complex random number generator, a mixture of LFSR and updated LCG is proposed in this work and denoted as the LFSR-LCG method. For generating numbers, LFSR provides a sequence with a cycle length of 2^N-1 bits and LCG will have the same maximum or less than the LFSR cycle length. LFSR will be run twice in parallel to generate two Key vectors that have been used as the multiplier factor (a) and the increment (c) in the general recurrence formula to generate the final key vector by LCG. The fundamental equation of LCG has been updated in the proposed form by applying the square root for the starting value (seed) as shown in equation (2):

$$X_{n+1} = \left(an\sqrt{X_n} + cn\right) mod\ m \qquad n \geq 0 \qquad \ldots\ldots\ldots (2)$$

Where: a is the multiplier; $0 \leq a_n < m$, c, the increment; $0 \leq c_n < m$, m, the modulus; $m > 0$, X_n, the starting value; $0 \leq X_n < m$, the mod m notation implies that the right-hand expression of the equation is divided by m, and the remainder is substituted. The characteristics of the random number generator are determined by parameters a, c and m, a and b, are two LFSRs required two seeds of bits denoted by X_0. (This seed obtained the same generator if running with the same parameter values and with the same seed). LFSR are periodic so the proposed method tries to produced pseudo-random numbers. The equation of LCG applied on each number result in each stage of LFSR and the number is square rooted break the linearity. Whereas, by taken random and different values for the parameters a and c will improve the statistical performance of the generated sequence. Usually, LCG provides sequences "get into

a loop"; i.e., there is ultimately a cycle of numbers that are repeated endlessly. The repeating cycle is called the period. The Proposed generated sequence will have a relatively long period.

The range of the random numbers that will be generated using the proposed relation is from 0 to m-1. Applying the square root for the starting value (seed) will terminate the (linear) lattice construction and give a nonlinear performance to the proposed generator. Figure 2 illustrates the Flowchart of the LFSR-CG method.

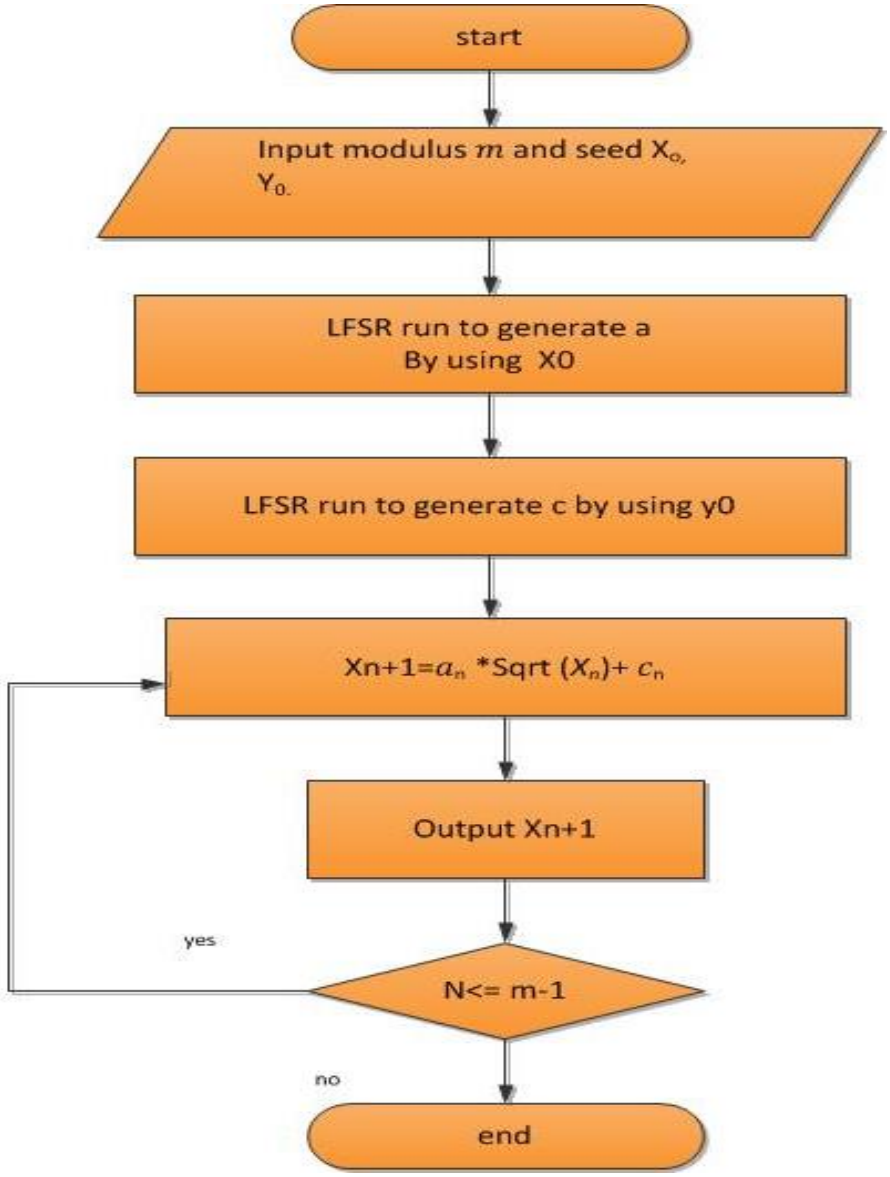

Figure 2- Flowchart of LFSR-CG method.

The total steps of the proposed method are mention in the algorithm (1).

Algorithm (1): Proposed Hybrid LFSR-CG:
Input: Modulus m and seed X_o, Y_0.
Output: Key vector X.
Step1: Run LFSR for a predetermined number of bits to generate the multiplier (a) based on X_0.
Step2: Rerun LFSR to generate increment (c) based on Y_0.
Step3: Convert a and c to floating numbers
Step4: For $n \leftarrow 1$ to $m-1$
Step 5: $X_{n+1} \leftarrow (a_n \text{ Sqrt } (X_n) + c_n) \bmod m$
Step 6: Output Xn_{+1}
Step5: END For n
Step6: END Algorithm

An example of generation number is explained in Table 1 where no. of bits=11 and Modulus=10000.

Table 1-Samples of the generated number using LFSR-CG

#	a	c	Key Vector	#	a	c	Key Vector
1	1745	1557	7075	7	1883	1496	4771
2	872	778	4125	8	941	748	5742
3	1460	1413	5188	9	470	1398	7013
4	730	1730	4312	10	235	699	378
5	1389	1889	3102	11	1141	349	2539
6	1718	944	6626	12	1594	1198	1513

4. Experimental Results:

The quality of the sequences produced by LFSR-CG is should be tested. The sequences should have a high degree of randomness and be decorrelated to one another.

4.1. Statistical analysis: This type contains many tests that show the statistical characteristics of the results, including:

4.1.1 Randomness evaluation

NIST (National Institute of Standards and Technology of the U.S. Government) is used tests that can be applied on binary sequences. Here, the sequences are evaluated through statistical tests suite NIST [9]. Such a suite consists of a statistical package of fifteen tests developed to quantify and assess the randomness of binary sequences, produced by pseudo-random number generators. The result of the frequency test explains in Table 2 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the frequency test.

Table 2-The p-value of Frequency Test

Sequences	512	1024	1536	2048	2560
1	0.79088	0.31731	0.35833	0.65853	0.65853
2	0.11161	0.03359	0.05248	0.08203	0.03359
3	0.21592	0.08012	0.05248	0.06836	0.08012
4	0.15730	0.34850	0.79860	0.56561	0.56561
5	0.53610	0.13361	0.04123	0.03779	0.03779

The result of frequency within the block test explains in Table 3 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the frequency within the block test.

Table 3-The p-value of frequency within a block test

Sequences	512	1024	1536	2048	2560
1	0.57363	0.17182	0.22157	0.29200	0.52793
2	0.01326	0.01526	0.02396	0.13261	0.31542
3	0.39761	0.59141	0.71357	0.19338	0.21742
4	0.58968	0.70650	0.45324	0.60570	0.30918
5	0.86455	0.66123	0.32529	0.39989	0.16893

The result of the cumulative sum forward test explains in Table 4 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the run test.

Table 4-The p-value of Cumulative sum forward test

Sequences	512	1024	1536	2048	2560
1	0.59075	0.59094	0.69817	0.73635	0.69809
2	0.55060	0.53975	0.65807	0.73328	0.78025
3	0.54129	0.56578	0.66041	0.67949	0.67194
4	0.55471	0.50115	0.69560	0.76426	0.69842
5	0.56266	0.58830	0.69556	0.69419	0.65815

The result of the cumulative sum reverse test explains in Table 5 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the run test.

Table 5-The p-value of Cumulative sum reverse test

Sequences	512	1024	1536	2048	2560
1	0.58124	0.61188	0.75933	0.76670	0.77180
2	0.57554	0.63441	0.68805	0.76023	0.73136
3	0.57742	0.58870	0.73047	0.71731	0.73523
4	0.57044	0.66980	0.67556	0.78717	0.77541
5	0.64476	0.59542	0.68226	0.84543	0.80324

The result of the Run test explains in Table 6 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the run test.

Table 6-The p-value of Run Test

Sequences	512	1024	1536	2048	2560
1	0.79325	0.92523	0.85504	0.96819	0.57519
2	0.23866	0.05750	0.05982	0.05635	0.06167
3	0.36424	0.97656	0.84266	0.63202	0.91414
4	0.28696	0.73366	0.30663	0.10992	0.05591
5	0.9160	0.52595	0.91510	0.85392	0.86466

The result of the longest run of one test explains in Table 7 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 7-The p-value of the longest run of ones in a block test

Sequences	512	1024	1536	2048	2560
1	0.14209	0.27188	0.19857	0.62350	0.21497
2	0.50523	0.41485	0.19601	0.41282	0.09245
3	0.48421	0.18896	0.43826	0.27813	0.21003
4	0.64462	0.67440	0.61710	0.77006	0.53682
5	0.12468	0.06469	0.07087	0.12535	0.07317

The result of the binary matrix rank test explains in Table 8 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test except the sequence of 512-length. It may be a shorter length than required the test the p-values are less than the threshold.

Table 8-The p-value of binary matrix rank

Sequences	512	1024	1536	2048	2560
1	Fail	0.69372	0.69372	0.48125	0.40125
2	Fail	0.29189	0.29189	0.74191	0.71191
3	Fail	0.69372	0.69372	0.48125	0.48125
4	Fail	0.03910	0.03910	0.52812	0.44763
5	Fail	0.69372	0.69372	0.74191	0.70804

The result of the discrete Fourier Transform test explains in Table 9 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 9-The p-value of discrete Fourier Transform test

Sequences	512	1024	1536	2048	2560
1	0.74560	0.01365	0.30190	0.96765	0.46816
2	0.01866	0.32955	0.39927	0.33039	0.14679
3	0.93535	0.10829	0.70793	0.51641	0.14679
4	0.37228	0.86339	0.92538	0.20868	0.85608
5	0.46539	0.30190	0.70793	0.57019	0.85608

The result of the non-overlapping template matching test explains in Table 10 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 10-The p-value of non-overlapping template matching

Sequences	512	1024	1536	2048	2560
1	0.99925	0.06988	0.27084	0.41284	0.52158
2	0.99925	0.98609	0.27084	0.85603	0.83937
3	0.08933	0.62326	0.81922	0.41901	0.25890
4	0.99925	0.98609	0.27084	0.41901	0.05429
5	0.99925	0.98609	0.00545	0.03589	0.08824

The result of the overlapping template matching test explains in Table 11 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test except the lengths 512 and 1024. The p-value in the lengths 512 and 1024 are less than the required threshold.

Table 11-The p-value of overlapping template matching test

Sequences	512	1024	1536	2048	2560
1	Fail	Fail	0.28276	0.28276	0.50121
2	Fail	Fail	0.88659	0.88659	0.70253
3	Fail	Fail	0.28276	0.28276	0.02857
4	Fail	Fail	0.28276	0.28276	0.02857
5	Fail	Fail	0.88659	0.88659	0.49644

The result of the universal test explains in Table 12 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 12-The p-value of Universal test

Sequences	512	1024	1536	2048	2560
1	0.64507	0.73854	0.79694	0.87150	0.78471
2	0.63120	0.65463	0.81593	0.78180	0.84350
3	0.66677	0.74329	0.73042	0.90105	0.86948
4	0.70742	0.62776	0.75118	0.86013	0.89396
5	0.70915	0.71770	0.77851	0.90991	0.82961

The result of the approximate entropy test explains in Table 13 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 13-The p-value of approximate entropy test

Sequences	512	1024	1536	2048	2560
1	1.00000	0.99990	0.29371	0.00106	0.00011
2	1.00000	0.99956	0.51138	0.01239	0.00026
3	1.00000	0.99964	0.08497	0.00059	0.00000
4	1.00000	0.99999	0.46355	0.00598	0.00039
5	1.00000	0.99383	0.09368	0.00016	0.00001

The result of the random excursion test explains in Table 14 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 14- The p-value of random excursion test

Sequences	512	1024	1536	2048	2560
1	0.67702	0.76726	0.78783	0.77603	0.86433
2	0.67822	0.66563	0.74793	0.73808	0.88619
3	0.75925	0.72391	0.79800	0.78257	0.83818
4	0.71561	0.70934	0.77836	0.74594	0.92550
5	0.67126	0.66337	0.79458	0.80957	0.82258

The result of the linear complexity test explains in Table 15 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 15-The p-value of Linear complexity test

Sequences	512	1024	1536	2048	2560
1	0.31550	0.34496	0.32881	0.42352	0.50332
2	0.32905	0.22125	0.36025	0.34021	0.39968
3	0.26491	0.31807	0.32263	0.37845	0.48389
4	0.24368	0.29621	0.36166	0.36795	0.34147
5	0.27012	0.26703	0.37015	0.30047	0.39101

The result of the serial test explains in Table 16 using the length of sequences 512, 1024, 1536, 2048, and 2560 respectively. All resulted p-values are passed the test.

Table 16-The p-value of serial test

Sequences	512	1024	1536	2048	2560
1	0.54002	0.46654	0.58387	0.61780	0.62344
2	0.44214	0.44161	0.59624	0.53340	0.60028
3	0.45242	0.56010	0.51750	0.52321	0.69178
4	0.45975	0.51647	0.55081	0.56788	0.59137
5	0.47734	0.45870	0.54219	0.62917	0.61233

4.1.2 Hamming distance

Hamming distance is part of the correlation assessment that is made by using it for each pair of sequences generated. It is analyzed based on the bits of pseudo-alterations made. The Hamming distance is the number of positions where two binary sequences are with same length. If the binary sequences are truly random, the value should be around half the overall length of the sequence, corresponding to 0.50 [10]. This distance is calculated from each pair of sequences generated as addressed in Table 17.

Table 17-Hamming Distance

Sequences	Length of 512	Length 1024	Length 1536	Length 2048	Length 2560
1 and 2	241	494	756	1008	1267
1 and 3	261	502	764	1022	1270
1 and 4	243	493	751	1003	1254
1 and 5	262	514	772	1031	1276
2 and 3	246	482	740	1008	1273
2 and 4	244	507	769	1017	1263
2 and 5	263	512	766	1017	1277
3 and 4	248	491	735	993	1248
3 and 5	243	512	752	1003	1258
4 and 5	247	509	741	1028	1262

4.1.3 Pearson's Correlation Coefficient

Pearson's correlation coefficient is the second method used to analyze the correlation between the pseudo-random sequences [10]. The analyses consist to compute Pearson's correlation coefficient between each pair of sequences and presenting the distribution of the values through a histogram. Consider a pair of sequences. The corresponding correlation coefficient is shown in Figure 3. The mean values of S_1 and S_2, respectively. Two uncorrelated sequences are approximated to zero. The closer the value of correlation to one represents the stronger correlation between the two sequences. In the case of two independent sequences, the value of correlation is equal to zero. The coefficients have an absolute value in [-0.05, 0.05] then only a small correlation is detected.

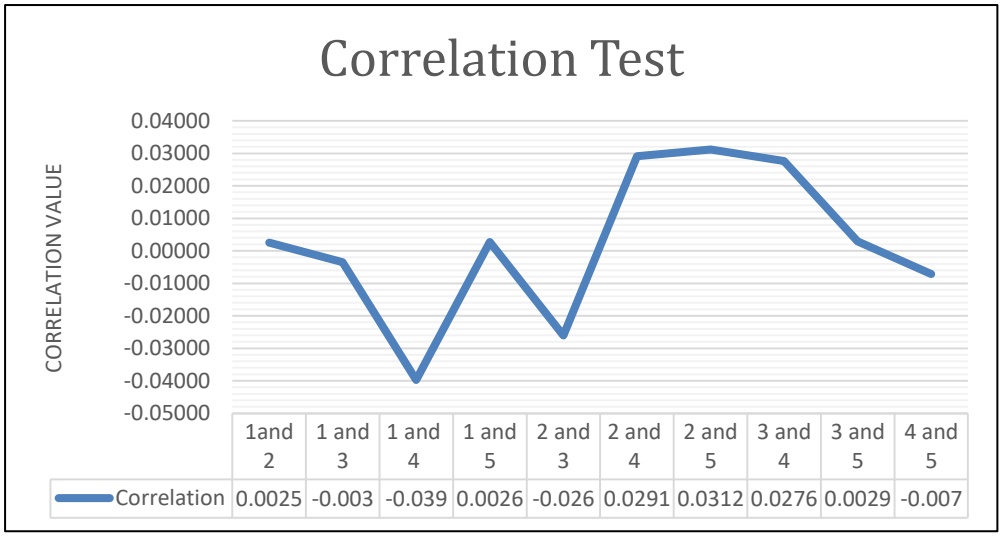

Figure 3-The correlation test of each sequence's pear

4.2. Security Analysis

There are several points concerning the safety of the proposed LFSR-CG, including the size of the seed space, the length of the period of the process, and some fundamental attacks (differential attack and brute-force attack). The seed space is associated with computer resources.

4.2.1 Seed Space

A seed space of size smaller than 2^{128} is not secure enough. A broad key space is essential for a robust PRBG to have a wide range of pseudo-random number generation options. The LFSR-CG has a large keyspace depend on the size of the initial number generator and the LCG generation. Each number is initialized with a seed corresponding to a binary 64 floating-point number. Thus, the total number of choices for the three initial seeds is 128 plus the initial value of the LFSR which is about {11, 13, 17, 19, 23, etc.}.

4.2.2 Attack of Brute Force

In theory, because it is not laid back to detect any flaw in the proposal which will make the job easier, a brute-force attack that can be used against is usually used. The attack strategy is straightforward: search all possible keys systematically before the original key is discovered. To find the initial seeds, just half of the keyspace must be tested on average. Such an attack can be thwarted by a large keyspace. A keyspace with a size greater than 2^{128} is secure computationally stable to resist such an attack. The range of LFSR is (high size of seed could be used) and the LCG could also apply to the output of LFSR.

4.2. Differential attack

The theory of such a cryptanalysis strategy, like a chosen-plaintext attack, is to examine the influence of a minor variation in input pairs (i.e., seeds) on the difference of corresponding output pairs (i.e., sequences) [12]. The most likely key that was used to produce the pseudo-random sequence can be obtained using this technique. The initial variance may be an XOR difference, with the diffusion aspect being determined by a differential probability. The proposed algorithm is intended to protect against such attacks. Those were the initial seeds. The statistical study also revealed that, even though the seeds are slightly different, the pseudo-random sequences are strongly decorrelated from one another, as seen in Fig. (4). As a result, we believe that the proposed approach would be resistant to differential cryptanalysis. There are week correlations between the sequences generated from the proposed method.

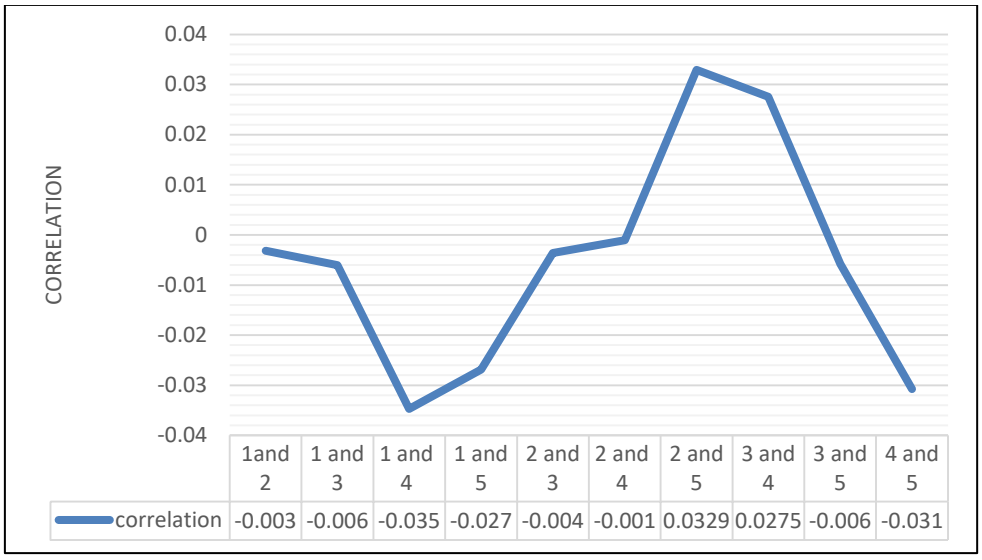

Figure 4-The correlation test of differential-attack each sequence's pear

5. Conclusions:

An effective pseudo-random number generator is resistant to major PRNG attacks, uses less memory and CPU power, and simpler to implement. The suggested generator has passed most of the checks in the NIST SP 800-22 statistical evaluation suite. In the presented algorithm, we have modulated the random sequences generated from combining the operation of the LFSR and LCG equation to obtain efficient keystream sequences for encryption. All the simulation and experimental analyses show that the proposed method has a high sensitivity to initial conditions and has a large keyspace, which is by far very safe for encryption applications. The method is also resistant to a differential analysis by generating a correlated sequence when no significant change in initial values.

References:

[1] Bird, Jordan J.; Ekárt, Anikó; Faria, Diego R., "On the effects of pseudorandom and quantum-random number generators in soft computing," *Soft Computing. Springer Science and Business Media LLC*. 24 (12): 9243–9256. doi:10.1007/s00500-019-04450-0. ISSN 1432-7643, (2019-10-28).

[2] Bi, Yu, "A Reconfigurable Supercomputing Library for Accelerated Parallel Lagged-Fibonacci Pseudorandom Number Generation," Master's Thesis, University of Tennessee, 2006.

[3] Patra S., Sinhamahapatra S., Mishra S., "Critique on Signature Analysis Using Cellular Automata and Linear Feedback Shift Register," In Behera H., Mohapatra D. (eds) Computational Intelligence in Data Mining. *Advances in Intelligent Systems and Computing*, vol 556. Springer, Singapore, (2017), https://doi.org/10.1007/978-981-10-3874-7_36

[4] M. Auqib Hamid Lone, A. Samad AL-khatib, "Acoustic Lightweight Pseudo-Random Number Generator based on Cryptographically Secure LFSR," *I. J. Computer Network and Information Security*, 2018, 2, 38-45.

[5] Shruthi K., Prasanna Kumar C., Akshatha, " Comparative Analysis of Highly Efficient Random Number Generator, " *IOSR Journal of Electronics and Communication Engineering* (IOSR-JECE), Volume 12, Issue 4, Ver. II (Jul.-Aug. 2017).

[6] S. Muhi Falih, "A Pseudorandom Binary Generator Based on Chaotic Linear Feedback Shift Register," *Iraq J. Electrical and Electronic Engineering*, Vol. 12, No. 2, 2016.

[7] A. Kumar Panda and K. Chandra Ray, "A Coupled Variable Input LCG Method and Its VLSI Architecture for Pseudorandom Bit Generation", *IEEE transactions on instrumentation and measurement*, vol. 69, no. 4, April 2020.

[8] Stallings, William. "*Entitled Cryptography and Network Security: Principles and Practice," 7th Edition, London EC1N 8TS: Pearson Education*, 2017.

[9] Cao, Y., Guo, H., & Gong, W., "An algorithm to generate the pseudorandom sequence based on knight-tour," In 2016 IEEE International Conference on Signal and Image Processing (ICSIP) (pp. 352-357). IEEE, (2016, August).

[10] LI Cai-hong, LI Yi-bin, ZHAO Lei, "Research on statistical characteristics of chaotic pseudorandom sequence for one-dimensional Logistic map," *Application Research of Computers*, vol. 31, no. 5, pp. 1403-1406, 2014.

[11] I. Dogaru, R. Dogaru, "FPGA implementation and evaluation of two cryptographically secure hybrid cellular automata," Proceedings of the IEEE Conference COMM 2014, Bucharest, 29-31 May 2014.

[12] Xiao Xu-Tao, Zhang Xue-Feng, "Pseudo-random sequence generation method based on LFSR and combination cat map," *Application Research of Computers*, vol. 30, no. 1, pp. 161-164, 2013.

[13] Mahmood, A.S., Mohd, M. S., "Generating and Expanding of an Encryption Key Based on Knight Tour Problem," *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 7, pp. 1485-1495, 2017.

[14] Singh, M., Kakkar, A., & Singh, M., "Image Encryption Scheme Based on Knight's Tour Problem," *Procedia Computer Science*, vol. 70, pp. 245-250, 2015.

[15] Elkies, N. D., Stanley, R. P., Kleber, M., & Vakil, R., "*The mathematical knight. The Mathematical Intelligencer*," vol. 25, no. 1, pp. 22-34, 2003. http://dx.doi. org/10.1007 /BF02985635

[16] Golenia, B., Golenia, S., & Erde, J., "The closed knight tour problem in higher dimensions," *arXiv preprint arXiv*:1202.5291, (2012).