Journal of Theoretical and Applied Information Technology
31st October 2020. Vol.98. No 20
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

PSEUDO RANDOM NUMBER GENERATOR BASED ON LOOK-UP TABLE AND CHAOTIC MAPS

MOUSA FARAJALLAH, MOHAMMAD ABUTAHA, MOHAMMAD ABU JOODEH, OMAR SALHAB, NOOR JWEIHAN

College Of Information Technology And Computer Engineering - Palestine Polytechnic University - Hebron-Palestine
E-mail: mousa_math@ppu.edu, m_abutaha@ppu.edu, 131089@ppu.edu.ps, 131082@ppu.edu.ps, 131035@ppu.edu.ps

ABSTRACT

Pseudo-Random Number Generators (PRNGs) play an important role in many cryptographic applications. Many network security algorithms and protocols that are based on cryptography also require random or pseudorandom numbers at certain points. PRNG is an algorithm that is used to generate a random sequence that has the same behavior of truly random numbers. Each generated number should not depend on the previously generated numbers, and thus it is not possible to predict such numbers. However, not all PRNGs are suitable for cryptographic applications. Therefore, various statistical tests can be applied to the obtained sequence to evaluate and compare it with truly random sequences. A PRNG that passes all statistical tests can be considered as a statistically secure PRNG. Based on some properties of chaos, such as randomness, unpredictability, and high sensitivity to the secret key, we proposed a new PRNG. The proposed PRNG is based on a modified chaotic map proposed in [20], and a new lookup table, which is created in such way that the stored numbers will not be duplicated in the same row, column, or diameter. The proposed map and lookup table are used to produce non-linear and non-invertible functions, which are the main targets of any secure PRNG. As a result, the behavior of the proposed PRNG simulates the behavior of TRNGs. The obtained results from the cryptographic analysis and the standard statistical National Institute of Standards and Technology (NIST) tests indicate the robustness of the proposed PRNG. Furthermore, it is robust against known cryptographic attacks, and it has a strong non-linearity compared to the other systems. A comparison study of efficiency in terms of both speed performance and robustness against cryptanalysis was done, and the results demonstrate the superiority of the proposed algorithm.

Keywords: *PRNG, Cryptographic, Randomness, Statistical Tests, Chaos.*

1. INTRODUCTION

In the ever-changing world of huge data communications, low cost of internet connections, and very fast emerging in software development, security is becoming a big challenge. Nowadays, security is considered one of the most important requirements due to the insecurities found in global computing. When a sender sends data to a receiver, the data might pass through several points before reaching the receiver, giving a chance to others the chance to modify it.

Technological advances in digital content processing, production, and delivery have risen up into new signal processing applications in which security threats can no longer be handled in a traditional model. These applications range from multimedia content (image, video, audio, etc.), production, and distribution to advanced biometric signal processing for access control, identity verification, and authentication. However, due to the potential risks related to security and privacy, it is becoming hard to adopt new image and video processing services.

Encryption is important because it allows you to securely preserve data that you don't want anyone else to have access to. Governments, organizations, individuals use it not only to protect classified, personal information to guard against attacks, but also, to securely protect folder contents, which could contain email archives, chat histories, credentials information, credit card numbers, or any other sensitive information [2].

Security in real-time and embedded systems is a subject that has received an increasing amount of attention from industry and academic point of view in recent years. These systems are being deployed in a wide range of application areas [3, 4]. Embedded and real-time systems are facing more and more security problems. Malicious and harmful attacks targeting the system are one of the main causes that could lead to security degradation and system vulnerability

Nowadays, cryptographic methods and techniques are widely applied to image and video processing applications. The cryptographic techniques used in these applications must be able to protect the multimedia data against attacks. The confidentiality of images and video contents is a hot topic and should be considered with particular attention to both compression and encryption requirements. Chaos-based systems are more suitable to protect these huge data (images and videos). Indeed, there exists a good interesting relationship between chaos and cryptographic systems. The main properties of chaotic systems are highlighted Shannon's paper on cryptography [5], for example, the sensitivity to control parameters or initial condition, deterministic dynamics, and structure complexity.

The performance of chaos-based crypto and crypto-compression systems consists of a trade-off between robustness against cryptanalysis and computational cost [1].

One Time Pad is considered one of the fastest and most powerful stream cipher algorithms out there. It guarantees the perfect secrecy, but because of that, it requires a random key with length greater or equal to plain text length. Therefore, it is not practical to use it because we can't satisfy this requirement. Thus, we are looking for a PRNG that takes an available and small secret key and expands it to a long dynamic key to be used in encryption and decryption algorithms. However, this property shouldn't come at the cost of security. We must make sure that this PRNG is secure and guarantee its performance [6].

1.1. Related Works

Sequences of random numbers are generated by using any non-linear dynamical systems. This generation is governed by a set of differential equations, iterative equations, or simply chaotic maps [7, 8]. Many chaotic maps and generators based on these maps have been proposed in the literature.

El Assad et al. [9] surveyed and analyzed some digital chaotic generators infinite precision: the Logistic map, the piecewise linear chaotic map (PWLCM), and the Frey map. They showed the weaknesses of these chaotic maps, such as having a limited cycle length and not exhibiting very good statistical properties when used exclusively. Thus, to improve the properties of these maps, they integrate these maps with a recursive structure and implement a technique of disturbance of the pseudo-chaotic orbit [10].

Li et al. [11–13] revealed the statistical properties of digital PWLC maps. The roles of these maps in cryptography were highlighted in their research. They proposed a pseudo-random number generator (PRNG) with very good cryptographic properties based on a system of two chaotic maps.

Thomas et.al [15], proposed a hardware random number generator based on a Linear Feedback Shift Register (LFSR) and a Cellular Automata Shift Register (CASR). The output of the generator has resulted from the xor operation between the LFSR and the CASR.

Other well-known bi-dimensional chaotic maps such as 2D-Standard map, 2D-Cat map, and 2D-Baker map were discussed and used in symmetric ciphers [1, 16–19].

Lian et al. [8] elaborated on the performance of the 2D Standard, Cat, and Baker maps. They demonstrated that the Cat map has the smallest keyspace. In this work, both the keyspace the key sensitivity has been increased. The key spaces for the Standard map and Baker map are both larger than that of the Cat map. However, to keep high key sensitivity, the number of iterations should be larger than 4 for the Standard map and bigger than 12 for the Baker map. These maps are suitable for systems where the same key is used in different iterations. However, to achieve a good confusion property, the average distance change in the whole image must be greater than 40%.

Faraoun [26] proposed a chaos-based generator using an n-ary keystream generator, based on a hierarchical combination of three chaotic maps, based on the Faraoun generator a cryptosystem to encrypt image is presented and analyzed which show a good statistical and analysis regarding attacks.

Most of the previous chaos-based generators operate with floating-point data operations. This raises a problem when the computer's resolution of the sender and receiver are slightly different. Both the chaotic sequences generated by the sender in a cryptographic system and the one generated by the receiver will be different. To overcome this problem a fixed finite precision of N bits is used. However, the chaotic dynamics are degraded with a finite precision N and short cycles may occur [7, 10].

Shannon clarifies [5], that the fundamental techniques to encrypt a block of bytes are substitution and permutation. A chaos-based encryption algorithm relies on chaotic maps. Indeed, the substitution and permutation operations are performed according to a chaotic sequence. In the following section, we report some of them.

2. PROPOSED CHAOTIC GENERATOR

The architecture of the proposed Pseudo-Random Number Generator (PRNG) relies on a previous work proposed in [20]. The system is formed by two recursive filters of order one. The first recursive cell contains a discrete Skew Tent map (with modifications of [20]), and the second recursive cell contains a discrete piecewise linear chaotic (PWLC) map. These maps are used as non-linear functions [7]. To produce the final Xg random sequence, map outputs are combined as follows (see Equation 1).

Where N is the input sequence, F1 is a modified version of the discrete Skew Tent map[20], F2 is the discrete PWLC map, and F is the result of combining the two map outputs.

To further increase the complexity of the chaotic signal, we have implemented a lookup table. The look-up table is generated using the standard PRNG function in C++ libraries under Linux with some conditions to improve the results. The main idea of the table is to guarantee that the numbers will not be duplicated in the same row or column, which indicates that there is no relation between the number and its position. Moreover, each number has the same probability distribution which confirms the random behavior. Thus, it stimulates the behavior of TRNGs. This table is generated using a pre-processing process, and it is stored in a file for later use. To achieve a high-performance mixture between the Look-up table and the Chaotic Maps, we are using an 8-bit look-up table which includes randomly distributed numbers (0 to 255) in 256 rows and 256 columns. The results of the chaotic maps are combined with the results of the look-up table to generate a new sequence.

2.1. Look-Up Table

Figure 1 shows an example of a 4-bit look-up. As shown in the figure, the numbers from 0 to 15 are not duplicated in the same row or column. In other words, there is no mathematical relationship between the number and its position. Therefore, the PRNG will simulate the behavior of TRNGs. To select one number from the table we need two 4-bit numbers, i.e., 4 bits to select the row and 4 bits to select the column. In our real proposed we have an 8-bit look-up table which has numbers from 0 to 255, and to select one number from the table we need two 8-bit numbers.

Depending on the output of the maps we will select four numbers randomly from the table to generate the table sequence to be XOR-ed with the sequence of the map. Whereas depending on the output of the chaotic generator, we will have a number of 32-bit as output (Map) in each round of generation, this number will be used as follows:

1. Enter the map value to Fn1, this function will generate four numbers of 8-bit value from Map value.

   · Fn1: take each 8 bits as a single number.

2. The four numbers that were generated using Fn1 will be entered to Fn2. This function will find all permutation that can be generated from these numbers, concatenate these numbers and store them in an array (a). As a result, array (a) will contain 24 numbers, each of 32 bits.

$$array\ size = number\ of\ Permutation = n!,\quad where\ n = number\ of\ inputs \quad (2)$$

3. Using the array (a) that we obtained in step 2, we will do multi simple XOR operations using equation 3.

$$array2[i] = array1[i] \oplus array1[array\ size - 1 - i]\ ,\quad 0 \le i < arraysize \quad (3)$$

Where *array1* is the input array with *n* numbers, and *array2* is the output result with *n/2* numbers.

4. After applying step 3 multi times, we will get a final array with 3 elements, each one of 32 bits.

5. After these calculations, we will obtain a 32-bit number from a final array (z). Using Fn1, we can generate four 8-bit numbers from (z).

   - The index of the chosen number done without any calculation, we choose the first element, but it is up to the user to choose another one, but he should specify it before the start.

6. Enter the four numbers to the function Fn4 to find the indices of the look-up table. This function will use the four numbers to generate four pairs of (x, y) to specify the indices of four numbers from the look-up table. After testing various pairs of numbers, we decided to use the following pairs as this selection of pairs achieved the best results in terms of security and performance:

$$\begin{aligned} position1 &= Table[z[0]][z[3]] \\ position2 &= Table[z[3]][z[0]] \\ position3 &= Table[z[1]][z[2]] \\ position4 &= Table[z[2]][z[1]] \quad (4) \end{aligned}$$

7. Now that we have the positions of four 8-bit numbers from the look-up table, we can apply the steps from 2 to 7 on these four numbers that were chosen from the table.

8. Finally, we have a new unique 32-bit number to be used in the PRNG.

The mixture of the look-up table and chaotic generator will guarantee high performance and improve the security. Equation 5 shows the integration between Chaotic Maps and look-up Table.

$$Final\ Sequence(n) = Maps(n) \oplus TableValue(n) \quad (5)$$

TableValue(n) is generated using Fn concatenate. This value uses the last four values generated in step 6 to generate one 32-bit value. This value will be XOR-ed with chaotic output (Map) to evaluate the last output sequence of the proposed PRNG.

The results obtained after implementation of our proposed PRNG are presented in the next section.

The entropy source of the proposed RNG comes from Linux RNG [1,21]. The process of entropy extraction can be achieved using the following steps: 1) updating the contents of the pool, 2) extracting the output which consists of random bits, and 3) decrementing the counter of the pool. This process involves hashing the pool contents using SHA-1 and adding the results to the pool [1,21].

The equations for the two maps: Discrete Skew Tent Map and Discrete PWLCM are given in equations 6 & 7:

Discrete Skew Tent Map:

$$X_s[n] = \begin{cases} \left\lceil 2^N \times \dfrac{X_s[n-1]}{P1} \right\rceil & if\ \ 0 < X_s[n-1] < P1 \\ 2^N - 1 & if\ \ X_s[n-1] = P1 \\ \left\lfloor 2^N \times \dfrac{2^N - X_s[n-1]}{2^N - P1} \right\rfloor & if\ \ P1 < X_p[n-1] < 2^N \end{cases} \quad (6)$$

where P is the control parameter and P ranges from 1 to 2^N- 1, and the precision N is the number of bits.

Discrete PWLCM map:

$$X_p[n] = \begin{cases} \left\lceil 2^N \times \dfrac{X_p[n-1]}{P2} \right\rceil . & if\ \ 0 < X_p[n-1] \le P2 \\ \left\lceil 2^N \times \dfrac{X_p[n-1] - P2}{2^{N-1} - P2} \right\rceil . & if\ \ P2 < X_p[n-1] \le 2^{N-1} \\ \left\lceil 2^N \times \dfrac{2^N - 1 - P2 - X_p[n-1]}{2^{N-1} - P2} \right\rceil & if\ \ 2^{N-1} < X_p[n-1] \le 2^N - P2 \\ \left\lceil 2^N \times \dfrac{2^N - X_p[n-1]}{P2} \right\rceil . & if\ \ 2^{N-1} - P2 < X[n-1] \le 2^N - 1 \\ 2^N - 1 - P2 \ . & other\ wise \end{cases} \quad (7)$$

Here, the control parameter P ranges from 1 to $2^{(N-1)}$-1.

We give below the outputs of the recursive cell containing the STmap and of the recursive cell containing the PWLC map respectively. Hence, the output equation of the recursive cell STmap is:

$$X_s = STmap\{F1[n-1]\,.P1\} \oplus \text{Q1} \quad (8)$$

With

$$F1[n-1] = mod[U_s + X_s(0)] \sum_{i=1}^{3} [K(i)_s \times X(n-i)_s], 2^N] \quad (9)$$

And the output equation of the recursive cell PWLC is:

$$X_p = PWLCmap\{F2[n-1].P2\} \oplus Q2 \quad (10)$$

With

$$F2[n-1] = mod[U_p + X_p(0)] \sum_{i=1}^{3} [K(i)_p \times X(n-i)_p].2^N] \quad (11)$$

In the equations above, P1 and P2 are control parameters in the range $[1,\ 2^N - 1]$ and $[1,\ 2^{N-1} - 1]$ respectively. Q1 and Q2 are perturbing signals produced by the linear feedback shift registers

(LFSRs). $K1_s$, $K2_s$, $K3_s$, $K1_p$, $K2_p$, $K3_p$ are the coefficients of the recursive cells in the interval $[1, 2^N - 1]$. U_s and U_p, each of 32 bits are deduced from IVg of 64 bits.

In parallel implementation we fix the number of threads to four, and each created thread needs to have its own secret key to generate samples.

3. STATISTICAL TESTS OF PROPOSED CHAOS-BASED GENERATOR

In this section, we report the results of several statistical tests that were carried out to quantify the properties of the proposed generator related to statistics. They concern Mapping, Histogram, Chi-square, Approximated invariant measures, Auto and Cross-Correlation, and the NIST test [1].

3.1. Mapping

The phase space trajectory is one of the characteristics of the generated sequence that reflects the dynamic behavior of the system. The resulting mapping in Figure 2a for delay 1 seem to be random compared to the known mapping (signature) of a Skewtent and a PWLC map. This is due to the recursion of the structure, the perturbation, and the chaotic multiplexing technique or XORing operation. In this case, it is impossible from the generated sequences to know which type of map is used. Therefore, the security of the system is improved [1].

3.2. Histogram

Another key property of any hearty PRNG is to give a uniform circulation in the entire stage space. We present in Figure 2b the histograms for three generated sequences corresponding to delay 1. We can visually observe that the histograms have uniform distribution [1].

3.3. Chi-Square Test And Approximated Invariant Measures

We apply the Chi-Square test to affirm the consistency of produced successions. The statistical Chi-Square test χ2 is calculated by the following formula:

$$X^2_{exp} = \sum_{i=0}^{K-1} \frac{(O_i - E_i)^2}{E_i} \quad (12)$$

With K being the number of classes (sub-intervals) equal to 1000.Oi: the number of observed (calculated) samples in the i-th class.

Ei: the expected number of samples of a uniform distribution, $E_i = 10^7/k$.

We compare the experimental value calculated above with a theoretical value obtained for a threshold α=0.05 and a degree of freedom K-1=999. To demonstrate the consistency of a created succession, the actual value of χ2 must be lower than the expected one $X^2_{exp} < X^2_{th}$. The More the experimental value of χ2 is smaller than the theoretical one; the better the uniformity is of the generated sequence. The experimental and theoretical values of the Chi-Square test for sequences are presented in Table 1.

Table 1: Experimental and theoretical values of the Chi-Square test for the proposed generator

Test	***Value***
Theoretical	1073.642651
Experimental	1017.450800

3.4. Auto And Cross Correlation

The properties of a random sequence are that the values in the sequences are not repeated or correlated, and the cross-correlation of two sequences x and y (generated with slightly different keys) is close to zero (see Figures 2c). The correlation coefficient ρ_{xy} of the two sequences x and y is calculated by the following mathematical equations [1, 22]:

$$\rho_{xy} = \frac{cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \quad (13)$$

Where

$$cov(x, y) = \frac{1}{N}\sum_{i=1}^{N}([x_i - E(x)][y_i - E(y)]) \quad (14)$$

$$D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2 \quad (15)$$

$$E(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i) \quad (16)$$

For Proposed PRNG $\rho = 0.00096$

In the previous equations, x_i and y_i are the values of x and y respectively.

3.2.5. Hamming Distance

Hamming distance is a measurement used to quantify the Avalanche E. The Avalanche Effect is achieved when a small change in input plaintext or key results in a huge change in the output. The HD is given by [1, 23, 24].

$$HD(S_1, S_2) = \frac{1}{|Ib|}\sum_{K=1}^{|Ib|}(S_1(K) \oplus S_2(K)) \quad (17)$$

where $S1, S2$ are samples of output with small change in input

The indicator in the proposed PRNG of the uniformity distribution is lower than Salsa20. To verify those indicators, the three generators under the test (Salsa20, RC4, and Chaos-based cipher) are reevaluated and the number of sequences having local HD more than 75 % or less than 25% are calculated and presented in the same table. Our indicator regarding Salsa20 proves the robustness and uniformity distribution of the generated bits since the number of sequences is only one sequence. In RC4, also our indicator is true since the Percentage of local HD less than 25% is 0.034536 and almost the same for those more than 75% which are not negligible percentages. Regarding the Chaos-based cipher, the percentage can be negligible in some applications and not in other [1].

Table 2. *Experimental and theoretical values of the Chi-Square test for the proposed generator.*

Generator	Salsa20	RC4	Proposed PRNG
Global HD	50.0364 %	50.0474 %	49.8969 %
Mini local HD	25 %	0 %	9.38 %
Max local HD	76.6 %	100 %	81.25 %
Percentage of Local HD less than 25 %	0	0.034536	0.001056
Percentage of Local HD more than 75 %	0.000064	0.35192	0.001024

3.6. Security Analysis

The key size of proposed PRNG is robustness against the Brute Force attack since it has a keyspace with 299 bits and an 8-bit look-up table. The output keystream in very sensitive to input based on the result of HD which is about 0.49 and it is very close to the optimal value. This indicates a small change in input will affect about 50% of output. The produced keystream is multiple of 32 bit which can be much more than millions of bits. That makes the proposed PRNG more complex to attack.

3.7. Entropy Analysis

The Information entropy is the probability of occurrence for each symbol in the video frame. The value of the entropy should be 8 for the truly random frame. The probability of the occurrence of each encrypted block in the encrypted frame number by the proposed chaos-based SE scheme is near to the theoretical value 8. This indicates that the proposed scheme is secure and robust against the entropy attack.

3.8 Key security

From the generated sequences, it is impossible to find the secret key; this is because of the structure of the chaotic generator which also includes chaotic switching. The knowledge of part of the secret key is not very useful for an attacker because of the intrinsic property of the chaotic signal, which is extremely sensitive to the secret key. The size of the secret key, formed by all the initial conditions and by all the parameters of the system, varies from 299 bits, with delay = 1, to 555 bits, with delay = 3. This means that the brute force attack is impracticable.

3.8. NIST Tests

To assess the factual exhibitions of the keystream delivered, we additionally utilize a standout amongst the most well-known test for examining the arbitrariness of information, specifically the NIST measurable test [25]. This test is a statistical package

Average Generation Time (ts)	0.0722789 ms
BR	1450.7478 Mbps
NCpB	14.4174les/Byte

that consists of 188 tests and sub-tests that were proposed to assess the randomness of arbitrarily long binary sequences.

Table 3. *NIST Test values with delay 1*

Test	*P value*	*proportion*
Frequency test	0.437	100.000
Block-frequency test	0.883	99.000
Cumulative-sums test	0.507	100.000
Runs test	0.679	99.000
Longest-run test	0.401	99.000
Rank test	0.494	98.000
FFT test	0.319	99.000
Nonperiodic-templates	0.480	99.054
Universal	0.401	100.000
Approximate Entropy	0.720	99.000
Random-excursions	0.363	99.008
Random-excursions-variant	0.544	99.471
Serial test	0.449	99.500
Linear-complexity	0.936	100.00
Overlapping-templates	0.304	100.000

These tests center around various non-arbitrariness practices that may exist in a sequence. We used the NIST test on all of these entities. All the 188 tests and sub-tests pass the NIST since all results up to the optimal value which about 95 (see figure 2d). In Table 3 we give the P-value and the proportion for the 15 NIST tests.

4. TIME PERFORMANCE TEST RESULTS

In this section, the complexity of the proposed generator is evaluated based on two terms: throughput and the required Cycles to generate one Byte (NCpB). (See (17) and (18)) [1]. The results in Table 4 confirm that the proposed generator has a high speed and suitable for Realtime secure applications.

$$BR = \frac{DataSize(Mbit)}{Time(\mu s)} \qquad (18)$$

$$NCpB = \frac{CPUSpeed(Hertz)}{Bit\ Rate\left(\frac{Mbit}{s}\right)} \qquad (19)$$

Table 4. Time Performance Results for Proposed PRNG

5. CONCLUSION

We designed and implemented, in an efficient and secure way a PRNG based on look-up Table and Chaotic Maps. Its structure is modular, generic, and allows the production of highly secure sequences. The proposed PRNG system is robust against known cryptographic attacks. Furthermore, it has strong non-linearity and non-invertibility compared to the other systems. In fact, the results obtained from the proposed generator indicate the good performance regarding the execution time and the security level, as future work, the proposed work should be implemented based on threads to speed up the encryption speed.

REFERENCES

[1] SALHAB, OMAR, Et Al. "SURVEY PAPER: PSEUDO RANDOM NUMBER GENERATORS AND SECURITY TESTS." Journal Of Theoretical & Applied Information Technology 96.7 (2018).

[2] D. SHARMA, A. KULSHRESHTHA, S. RAM, Network Security Challenges And Cryptography In Network Security, Journal Of Computer And Mathematical Sciences Vol 2 (1) (2011) 1–169.

[3] H. Kopetz, Real-Time Systems: Design Principles For Distributed Embedded Applications, Springer Science & Business Media, 2011.

[4] P. Koopman, Embedded System Security, Computer 37 (7) (2004) 95–97.

[5] C. E. Shannon, Communication Theory Of Secrecy Systems*, Bell System Technical Journal 28 (4) (1949) 656–715.

[6] J. Wiley, I. Sons, Applied Cryptography: Protocols, Algorithms, And Source Code In C., Elsevier, 1996.

[7] K. Desnos, S. El Assad, A. Arlicot, M. Pelcat, D. Menard, E Cient Multicore Implementation Of An Advanced Generator Of Discrete Chaotic Sequences, In: Chaos-Information Hiding And Security (CIHS), International Workshop On, 2014.

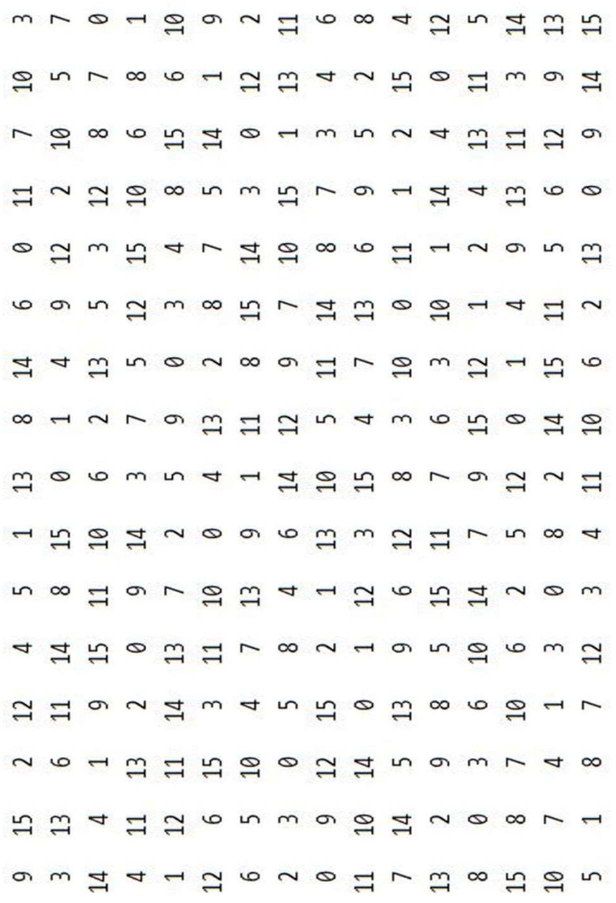
[8] S. Lian, J. Sun, Z. Wang, Security Analysis Of A Chaos-Based Image Encryption Algorithm, Physica A: Statistical Mechanics And Its Applications 351 (2) (2005) 645–661.

[9] S. El Assad, H. Noura, I. Taralova, Design And Analyses Of E Cient Chaotic Generators For Crypto-Systems, In: World Congress On Engineering And Computer Science 2008, WCECS'08. Advances In Electrical And Electronics Engineering-IAENG Special Edition Of The, IEEE, 2008, Pp. 3–12.
[10] S. El Assad, H. Noura, Generator Of Chaotic Sequences And Corresponding Generating System, WO2011121218 A1 Extension To : Europe EP-2553567 A1, February 2013 ; China : CN-103124955 A, May 2013 ; Japan : JP-2013524271 A, June 2013 ; United States : US-20130170641, July 2013. (10 2011).
[11] S. Li, G. Chen, X. Mou, On The Dynamical Degradation Of Digital Piecewise Linear Chaotic Maps, International Journal Of Bifurcation And Chaos 15 (10) (2005) 3119–3151.
[12] S. Li, Q. Li, W. Li, X. Mou, Y. Cai, Statistical Properties Of Digital Piecewise Linear Chaotic Maps And Their Roles In Cryptography And Pseudo-Random Coding, In: Cryptography And Coding, Springer, 2001, Pp. 205–221.
[13] S. Li, X. Mou, Y.-L. Cai, Pseudo-Random Bit Generator Based On Couple Chaotic Systems And Its Application In Stream-Ciphers Cryptography, In: Progress In Cryptology–INDOCRYPT 2001: Second International Conference On Cryptology In India Chennai, India, December 16 C20, 2001 Proceedings, 2001, Pp. 316–329.
[15] T. E. Tkacik, A Hardware Random Number Generator, In: Cryptographic Hardware And Embedded Systems-CHES 2002, Springer, 2003, Pp. 450–453.
[16] M. Hasler, Y. L. Maistrenko, An Introduction To The Synchronization Of Chaotic Systems: Coupled Skew Tent Maps, Circuits And Systems I: Fundamental Theory And Applications, IEEE Transactions On 44 (10) (1997) 856–866.
[17] J. Fridrich, Symmetric Ciphers Based On Two-Dimensional Chaotic Maps, International Journal Of Bifurcation And Chaos 8 (06) (1998) 1259– 1284.
[18] R. Lozi, Giga-Periodic Orbits For Weakly Coupled Tent And Logistic Discretized Maps, Proc. Conf. Intern. On Industrial And Appl. Math., New Delhi, India, Invited Conference (2007) 1–45.
[19] A. Senouci, I. Benkhaddra, A. Boukabou, A. Bouridane, A. Ouslimani, Implementation And Evaluation Of A New Unified Hyperchaos-Based Prng, In: Microelectronics (ICM), 2014 26th International Conference On, IEEE, 2014, Pp. 1–4.
[20] Farajallah, Mousa. Chaos-Based Crypto And Joint Crypto-Compression Systems For Images And Videos. Diss. UNIVERSITE DE NANTES, 2015.
[21] Z. Gutterman, B. Pinkas, T. Reinman, Analysis Of The Linux Random Number Generator, In: Security And Privacy, 2006 IEEE Symposium On, IEEE, 2006, Pp. 2 – 16.
[21] C.-Y. Song, Y.-L. Qiao, X.-Z. Zhang, An Image Encryption Scheme Based On New Spatiotemporal Chaos, Optik-International Journal For Light And Electron Optics 124 (18) (2013) 3329–3334.
[23] S. CE, A Mathematical Theory Of Communication. Acm Sigmobile Mobile Computing And Communications Review., Chaos, Solitons, Fractals 34 (3) (2001) 3–55.
[24] M. F. Zaher Amro, Survey:Derive A Standard Mathematical Tools To Evaluate Image Encryption Algorithms, Journal Of Theoretical And Applied Information Technology Of Computer Science Issues 7 (5) (2018) 1–22.
[25] B. Elaine, K. John, Recommendation for random number generation using deterministic random bit generators, Tech. rep., NIST SP 800-90 Rev A (2012).

[26] Faraoun, Kamel. "Chaos-Based Key Stream Generator Based on Multiple Maps Combinations and its Application to Images Encryption." Int. Arab J. Inf. Technol. 7.3 (2010): 231-240.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Appendices



9	15	2	12	4	5	1	13	8	14	6	0	11	7	10	3
3	13	6	11	14	8	15	0	1	4	9	12	2	10	5	7
14	4	1	9	15	11	10	6	2	13	5	3	12	8	7	0
4	11	13	2	0	9	14	3	7	5	12	15	10	6	8	1
1	12	11	14	13	7	2	5	9	0	3	4	8	15	6	10
12	6	15	3	11	10	0	4	13	2	8	7	5	14	1	9
6	5	10	4	7	13	9	1	11	8	15	14	3	0	12	2
2	3	0	5	8	4	6	14	12	9	7	10	15	1	13	11
0	9	12	15	2	1	13	10	5	11	14	8	7	3	4	6
11	10	14	0	1	12	3	15	4	7	13	6	9	5	2	8
7	14	5	13	9	6	12	8	3	10	0	11	1	2	15	4
13	2	9	8	5	15	11	7	6	3	10	1	14	4	0	12
8	0	3	6	10	14	7	9	15	12	1	2	4	13	11	5
15	8	7	10	6	2	5	12	0	1	4	9	13	11	3	14
10	7	4	1	3	0	8	2	14	15	11	5	6	12	9	13
5	1	8	7	12	3	4	11	10	6	2	13	0	9	14	15

Figure. 1.: *4-bit Look-up Table*

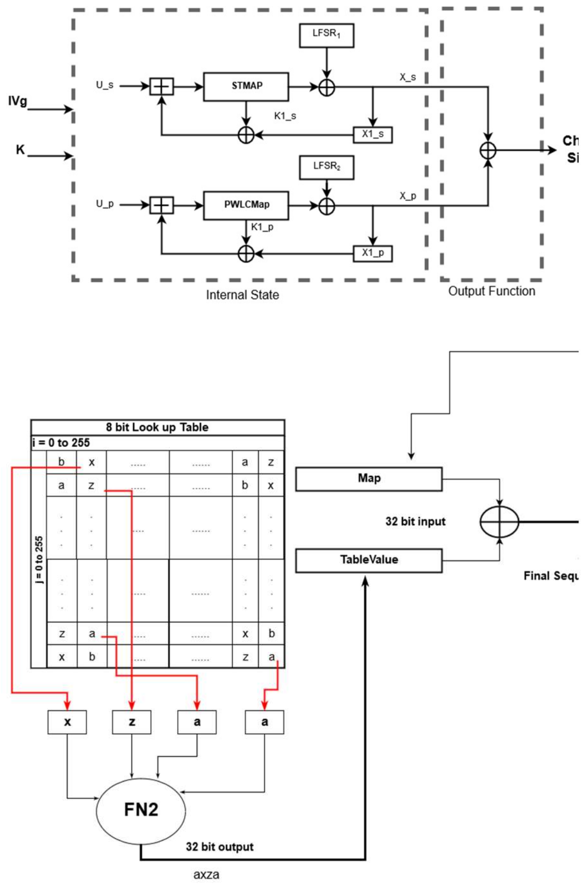


Figure 3: Architecture of the proposed generator with internal feedback mode

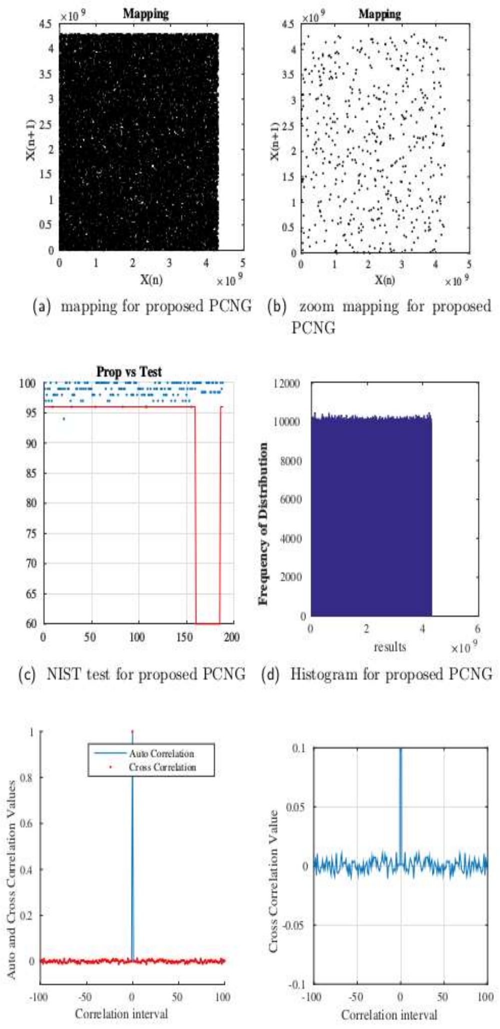


Figure 2: Test results of PRNG