Journal of Engineering and Applied Sciences 14 (18): 6714-6722, 2019
ISSN: 1816-949X

PRNG Impementation Based on Chaotic Neural Network (CNN)

Mohamed Ibrahim Shujaa
Electrical Engineering Technical College, Middle Technical University, Baghdad, Iraq
dralnedawy@yahoo.co.uk

Abstract: In this research, a neural network with chaos activation function has been applied as a Pseudo-Random Number Generator (PRNG). Chaotic Neural Network (CNN) is used because of its noise like behaviour which is important for cryptanalyst to know about the hidden information as it is hard to predict. A suitable adaptive architecture was adopted to generate a binary number and the result was examined for randomness using National Institute of Standard Technology (NIST) randomness tests. Although, the applications of CNN in cryptography have less effective than traditional implementations, this is because these systems need large numbers of digital logic or even a computer system. This research will focus on applications that can use the proposed system in an efficient way that minimize the system complexity.

Key words: Chaotic neural network, pseudo-random number, prediction, system complexity, NIST, traditional implementations

INTRODUCTION

Random number generation algorithms are very important in many practical applications of the cryptographic. Although, all of these algorithms are deterministic where sequences numbers are produced which are not statistically random but the algorithm produces sequences pass many reasonable tests of randomness, such numbers are referred to as pseudorandom numbers (Stallings, 2011).

One of the most important applications used the PRN is the stream cipher where cipher text output is created one bit-by-one bit or one byte-by-one byte from a stream of plaintext input where the PRNG used instead of True Random Number Generator (TRNG) because:

- The sender need only to deliver the key (or the seed) which is exemplary 54 or 128 bit, to receiver in the secure fashion
- It able to generate much faster than the true random number generator

The requirement compatible needs for a sequence of random number (Stallings, 2011; Guler and Ergun, 2010) is: next random bit must be forward and backward unpredictable where in both cases we cannot determine the next or previous bits from knowledge for any generated values.

Random bit stream appear random even though it is deterministic and must pass the statistical tests of randomness (e.g., NIST 800-22 test suite Rukhin *et al.* (2001)). The same random bit stream must not be able to be reproduced chaotic systems can provide those requirements where the main characteristics of chaotic systems are (Chatzidakis *et al*., 2014; Singla *et al*., 2014):

- Dynamical systems that highly sensitive to initial condition, i.e., a small differences in initial conditions cause unpredicted output
- Noise-like behavior, a small differences in initial conditions cause unpredicted output
- Unstable intermittent circles with extensive stretches

Because of these features, chaotic frameworks systems are broadly consolidated into encryption frameworks system as a random generator (Guler and Ergun, 2010; Hsiao *et al*., 2014; Behnia *et al*., 2011; Akhshani *et al*., 2014; Mansingka *et al*., 2012) or block cipher application (Lian, 2009).

On the other hand, Artificial Neural Network (ANN) represents highly nonlinear systems able to handle noisy data and fault tolerance and difficult decrypting by brute-force attack (Ertugrul, 2014), make it more suitable choice in cryptosystem. So, we can find several application of neural network in cryptosystem like:

- PRNG (Yayyk and Kutlu, 2014)
- Image and data encryption (Joshi *et al*., 2012)
- Public key generation (Jhajharia *et al*., 2013)
- Block cipher (Kotlarz and Kotulski, 2007)

And many other applications can be reviewed by Zoghabi *et al.* (2013) and Bafghi *et al.* (2008). The aim of this research is to implement the proposed PRNG based chaotic neural network using MATLAB and test the executive of the proposed generator using NIST 800-22 test suite.

Literature review: The major weakness of the most present random number generators is linearity. In other words, if we obtained portion of a random sequence, the successive numbers may be calculated using the associated linear function (Bafghi *et al*., 2008). We can find different applications of the neural network in cryptography by Yee and De Silva (2002), this review gives some examples of highly nonlinear PRNGs and some applications of different neural networks architecture in cryptography.

Singla *et al*. (2014) merged the features and qualities of chaos and neural system are consolidated to outline a pseudo-irregular binary succession generator. The statistical performance was examined against the NIST SP800-22 randomness tests. The consequences and results of examinations are promising and portray its pertinence for cryptographic applications.

The structure of artificial neural network was used as a key as a solution of synchronization in cryptography (Ertugrul, 2014). The proposed method was employed for text, audio and image data. The results were compared with k nearest neighbor and wavelet transforms and showed that his algorithm faster than the others with 100% decryption accuracy.

Yayyk and Kutlu (2014), proposed a neural network-based pseudo-random numbers. The performance of this generator was tested for randomness using National Institute of Standard Technology (NIST) irregularity tests. After they built two identical ANNs, one for non-direct encryption was demonstrated utilizing connection building usefulness. The encoded information was decoded with the second neural system utilizing basic leadership usefulness recurrent neural network was used to design a symmetric cipher able to resisting different attacks (Arvandi *et al*., 2008). The weight distribution of the hidden layers was totally depends on the original key. The proposed system supports variable key and block length.

In this research a PRNG using the CNN as described by Singla *et al*. (2014) was implemented using MATLAB, at same time several programs was built to test the generator performance based on the NIST SP800-22 (Rukhin *et al*., 2001).

MATERIALS AND METHODS

Basic of artificial network: The basic element of the neural network is the neuron or node which gets input from some different units or from an outer source. Each input has a related weight w which can be adjusted or modified based on a learning algorithm. The unit computes some function f of the weighted sum of its inputs and gives an output y to other nodes as shown in Fig. 1. The output of the neuron can calculate by:

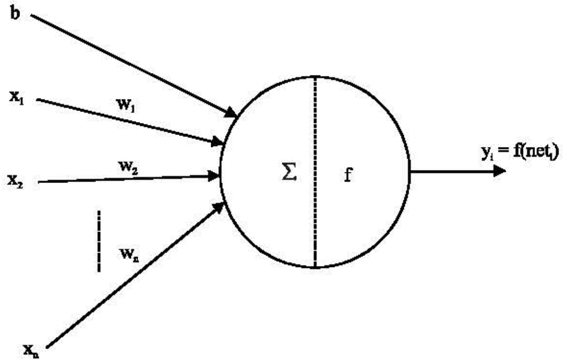

Fig. 1: An artificial neuron

$$y_i = f(net_i, b) = f\left(\sum w_i x_i + b\right) = f\,(WX+b) \quad (1)$$

Where:

X = Input vector $[x_1\ x_2, ..., x_n]$
W = Weigh vector $[w_1\ w_2, ..., w_n]$
b = Bias
y = Neuron output
f = Activation function (different types of activation function)
net = WX

Combination of several nodes together constitutes the Artificial Neural Networks (ANN) or simply neural networks. Neural networks are typically organized in layers. Layers are made up of a number of interconnected neurons. Neural networks have many topologies, the most useful one is the feed Forward Neural Network (FNN).

In the FNN, pattern are introduced to the system through the input layer which conveys to at least one hidden layers where the genuine process is done by a system of weighted connection.

The hidden layers at that point connect to a output layer where the appropriate response is output. As shown in Fig. 2 where in this configuration we have one hidden layers with different number of neurons in each layer. To calculate the output:

$$O_1 = X \quad (2)$$

$$O_2 = f\,(WX+B_1) \quad (3)$$

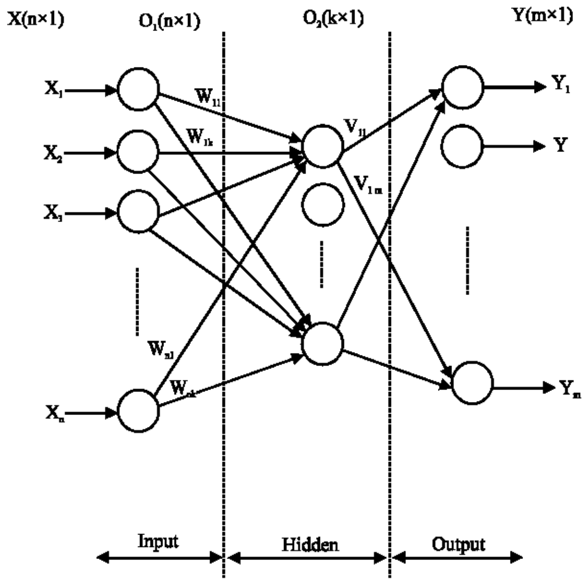

Fig. 2: Feed forward ANN

$$Y = f(VO_1 + B_2) \tag{4}$$

Where:
X = Input vector $[x_1, x_2, ..., x_n]$
O_1 = Output of the first layers, it's the same input for this example
O_2 = Output of the hidden layer $[o_1, o_2, ..., o_k]$
Y = Output vector $[y_1, y_2, ..., y_m]$
W = n×k weight matrix
V = k×m weight matrix
f = Activation function

The main characteristic of the ANNs is ability to learn. The learning is done by modifying the weights of the connection as indicated by the input pattern that it is given. There are a wide range of sorts of learning rules utilized by neural systems network but the most famous one is an abbreviation for the backward propagation of error as shown in Fig. 3.

So, the connection weights between network layers will modified according to this error by:

$$V(i+1) = V(i) + f_2(Y) \tag{5}$$

$$W(i+1) = W(i) + f_1(O) \tag{6}$$

where, f_1 and f_2 represent a specific learning algorithm 1. The general performance of ANNs can be described by the following algorithm:

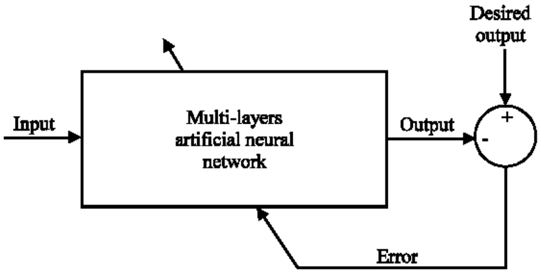

Fig. 3: ANN training scheme

Algorithm 1; The general performance of ANNs:

```
1) Initialize weight matrices
      W(n×k) and V(k×m)
Choose the error function, the iteration and the desired output
2) On given input, compute the actual output (forward computation)
      $O_2 = f(W×X+B_1)$        (3)
      $Y = f(V×O_1+B_2)$        (4)
3) Compare actual output to desired output
4) Update the weights (backward adaptation)
      $V(i+1) = V(i)+f_2(Y)$     (5)
      $W(i+1) = W(i)+f_1(O)$     (6)
5) Re-compute the output according to new weights and go to step 2
```

Chaotic system: According to business dictionary (Anonymous, 2019) it is complex framework that shows affectability to beginning conditions. In such frameworks any vulnerability (regardless of how little) at the outset will deliver quickly raising and aggravating error in the predication of the framework's future behavior. In other words, it is impossible to predict the future behavior of any complex (chaotic) system". The most important feature for such system which is very useful in cryptosystem is the unpredictability and uncertainty. The next subsections describe the behavior of selected chaotic equations and its implementation using MATLAB.

Mackey Glass chaotic: This system can be described in a first order differential equation as (Kostenko *et al*., 2009):

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x(t-\tau)^{10}} - bx(t) \tag{7}$$

Where:
x(t) = The state
a, b and τ = Control parameters

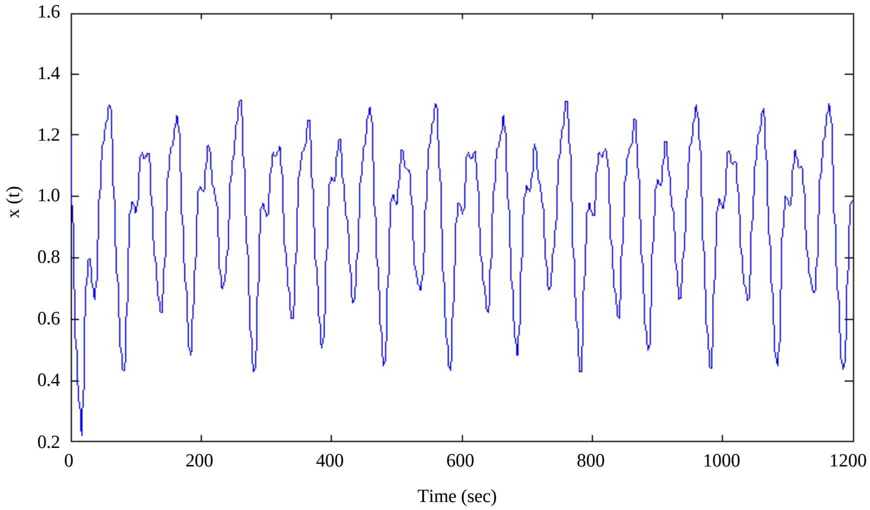

Fig. 4: Mackey Glass output; Mackey-Glass chaotic time series

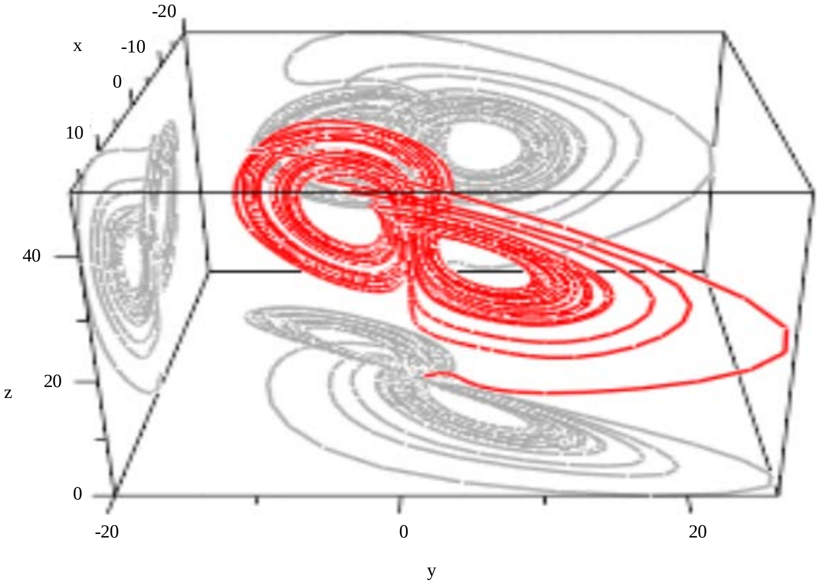

Fig. 5: Lorenz attractor output

As an example, let a = 0.2, b = 0.1, τ = 17 and x(0) = 0 the MATLAB implementation of the system gives an output shown in Fig. 4.

And we can gets a totally different outputs in small changes either in initial value x (0) or in one of the control parameters.

Lorenz attractor: It is three first order equations given by Viswanath (2004) and McEvoy (2009):

$$\frac{d}{dt}\begin{pmatrix}x\\y\\z\end{pmatrix}=\begin{pmatrix}p(y-x)\\-xz+rx-y\\xy-bz\end{pmatrix} \quad (8)$$

Where:
Y = [x y z] = The states
p, r and b = Control parameters

Figure 5 shows an example for p = 10, r = 28, b = 1 and the initial values Y0 = [1 1 1]. This behavior will totally different with small variation in one of the control parameters as shown in Fig. 6.

Piecewise Linear Chaotic Map (PWLCM): The one dimensional chaotic asymmetric tent map in the algorithm is represented as Singla *et al*. (2014) and Li *et al*. (2011):

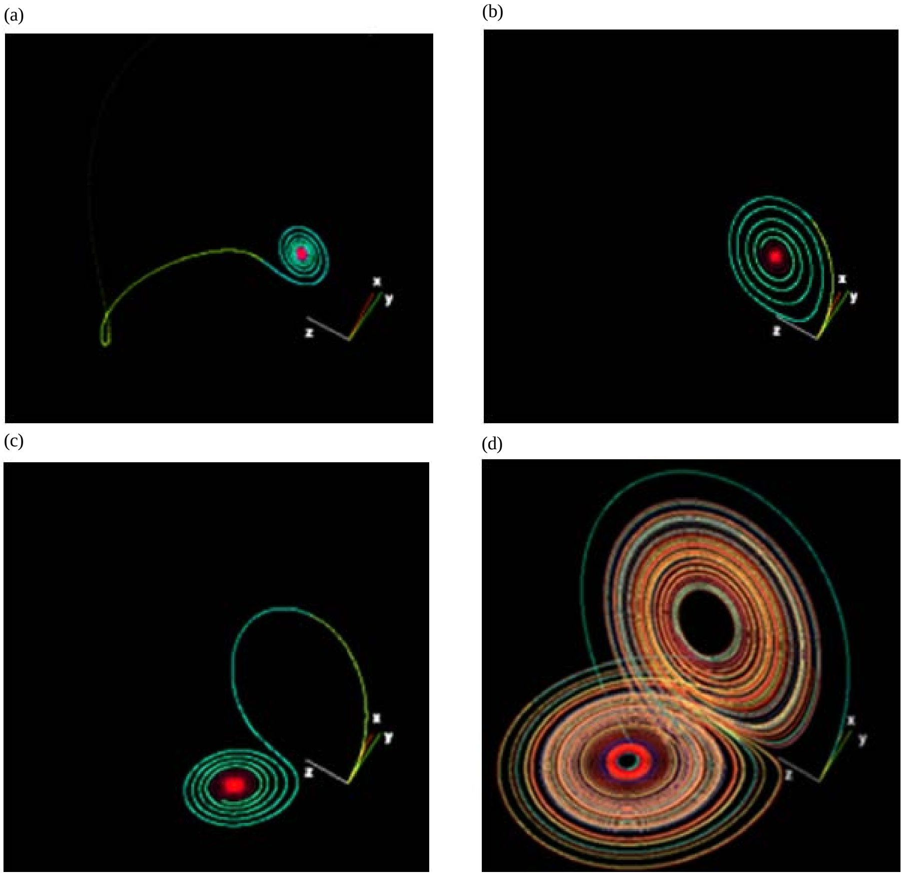

Fig. 6: Lorenz attractor behavior with different values of p (Anonymous, 2014): a) P 14; b) P13; c) P15 and d) P27

$$x(k+1)=f(x(k),q)=\begin{cases}\dfrac{x(k)}{q} & 0<x(k)\leq q\\ \dfrac{1-x(k)}{1-q} & q<x(k)<1\end{cases} \quad (9)$$

Where:

x(k) = State of the map $0<x(k)<1$

q = Control parameter $0<q<1$

Li *et al* (2011) shows the PWLCM behavior varies and give an expected behavior with different values of q and/or initial value. Figure 7 shows the MATLAB implementation for $q = 0.2$ and $x(0) = 0.6$ The PWLCM has most astounding Lyapunov example at $q = 0.5$.

PRNG based on chaotic neural network: In this study the proposed PRNG architecture will discuss. Figure 8 shows the general structure of the proposed system. The network consists of 4 layers: input layer, the first, hidden layer, the second, hidden layer and the output layer. The function of each layer (or so called forward computation) given by:

Input layer: The input for this network is 64 bits represent the seed (P = 64 bit) of the PRNG and the output of this layer given by:

$$net_0 = W_0P+B_0 \quad (10)$$

$$Y_0(0)=f(net_0,Q_0) \quad (11)$$

$$Y_0(k+1)=F(Y_0(k),\ Q_0) \quad k=1:n_0 \quad (12)$$

Hidden layer 1:

$$net_1 = W_1Y_0+B_1 \quad (13)$$

$$Y_1(0)=f(net_1,Q_1) \quad (14)$$

$$Y_1(k+1)=F(Y_1(k),\ Q_1) \quad k=1:n_1 \quad (15)$$

Hidden layer 2:

$$net_2 = W_2Y_1+B_2 \quad (16)$$

$$Y_2(0)=f(net_2,Q_2) \quad (17)$$

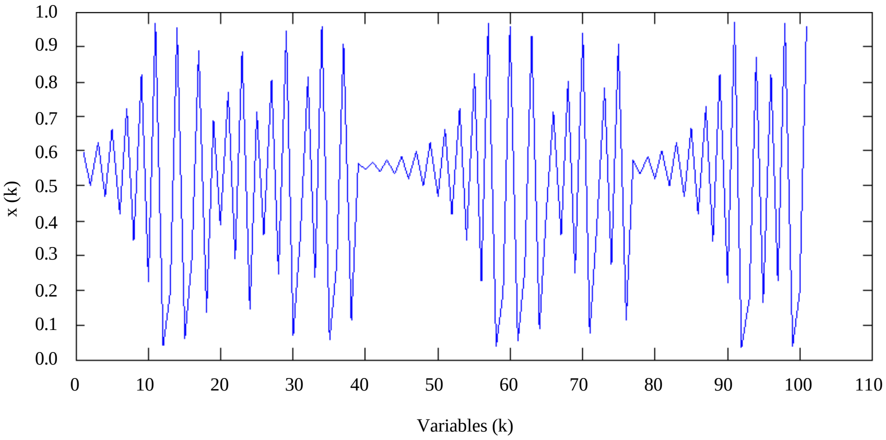

Fig. 7: PWLCM output

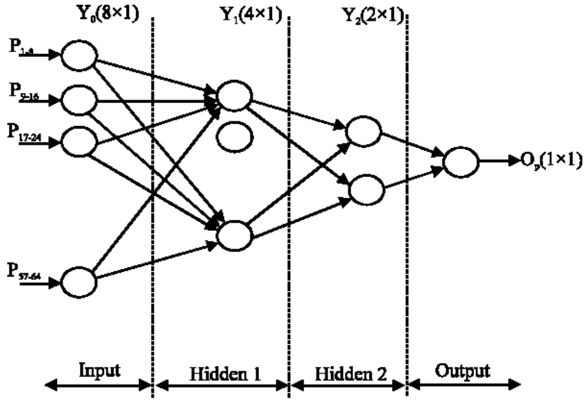

Fig. 8: Proposed PRNG architecture

$$Y_2(k+1) = F\left(Y_2(k), Q_2\right) \quad k = 1:n_2 \tag{18}$$

Output layer:

$$net_3 = W_3Y_2 + B_3 \tag{19}$$

$$O_p(0) = f\left(net_3, Q_3\right) \tag{20}$$

$$O_p(k+1) = F\left(O_p(k), Q_3\right) \quad k = 1:n_3 \tag{21}$$

Normalize the output:

$$O_p = \left(O_p \times 10^{10}\right)\text{mod}(256) = \begin{cases} 0 & \text{if} \quad <127 \\ 1 & \text{if} \quad \geq 127 \end{cases} \tag{22}$$

Where:

- $W_0(8\times8)$, $W_1(4\times8)$, $W_2(2\times4)$, $W_3(1\times2)$: Weight matrices <1
- $B_0(8\times1)$, $B_1(4\times1)$, $B_2(2\times1)$, $B_3(1\times1)$: bias vectors <1
- $Q_0(8\times1)$, $Q_1(4\times1)$, $Q_2(2\times1)$, $Q_3(1\times1)$: Control parameters $0.4<q<0.6$
- $0<n_0, n_1, n_2, n_3\leq 10$: number of iteration $1\leq n\leq 10$

All these values are initialized using 64 bit key as describe in the next study.

Key generator and initial values (Singla *et al.*, 2014): A 64 bit key with a 1-D chaotic cubic map used to generate the initial values of the CNN. As described in the following algorithm.

Algorithm 2; Generate the initial values of th CNN:

1) $K = K_1 K_2 K_3 K_4$

Where Ki is a 16- bit component of the key K, (64 bit)

2) Calculation of initial condition:

$$x(1) = \sum \frac{K_i}{2^{16}} \text{mod}(1) \tag{23}$$

3) Calculate the state of the cubic map:

$$x(k+1) = \lambda.x(k).(1 - x(k)^2) \tag{24}$$

Where:

λ = Control parameter ($\lambda = 2.59$)
$x(k)$ = The state ($0\leq x(k)\leq 1$)

Backward adaptation: The only adapted values are the control parameter matrices Q_i by Singla *et al*. (2014):

$$\begin{aligned} Q_0 &= 0.2\times Y_1 + 0.4 \\ Q_1 &= 0.2\times Y_2 + 0.4 \\ Q_2 &= 0.2\times Y_3 + 0.4 \\ Q_3 &= 0.2\times O_p + 0.4 \end{aligned} \tag{25}$$

RESULTS AND DISCUSSION

PRNG based on CNN implementation: The proposed generator was implemented and evaluated using MATLAB programming where the general steps given by:

Algorithm 3; The proposed generator was implemented and evaluated using MATLAB programming:

```
1.  Input K and calculate x form (Eq 23 and 24):
    Where:
            $x(k+1) = 2.59.x(k).(1-x(k)^2)$
Where:
            $x(1) = \sum \frac{K_i}{2^{16}} \bmod (1)$

2.  Initialize matrices based on value of x

    Weight matrices:
    W0 (8*8), W1(4*8), W2(2*4), W3(1*2)
    Bias vectors:
        B0 (8*1), B1(4*1), B2(2*1), B3(1*1)
    Control parameters:
    Q0 (8*1), Q1(4*1), Q2(2*1), Q3(1*1)
    Layer iteration:    n0, n1, n2, n3

    Where:

    $0<Wi, Bi, Qi<1$ and $1 \leq ni \leq 10$
3.  Input seed (P = 64 bit)
4.  Operate the neural network to calculate the Op (Forward Computation)
5.  Update the values of ($Q_0$-$Q_4$)
6.  Repeat steps 4-6 to gain the PRN succession of wanted length
```

Performance evaluation: In this study, the execution and performance of the system was measured which include: the 0/1 balance test and the NIST randomness tests.

0/1 balance test: The function named (balance test.m) based on MATLAB used to count number of ones and compute the average as shown in Table 1. The equality distribution measures are discovered close to 50% shown in that the proposed generator fulfill the correspondence dissemination property.

Table 1: Equality distribution of the PRNG

Sequence length	Count of 1s	Percentage
1000	510	51.00
10000	5175	51.75
20000	10226	51.13
50000	25420	50.84
100000	50525	50.52
200000	101153	50.58
500000	252176	50.44

Table 2: Some of NIST randomness tests

Randomness test	p-values (1000)	p-values (10000)
Frequency test	0.5271	0.2327
Block frequency test	0.3857	0.6882
Run test	0.4786	0.2135
Longest run of ones in a block	0.7532	0.0210
Discrete fourier transform	0.5617	0.1989

NIST randomness test: Many of the statistical test suite proposed by NIST (Arvandi *et al.*, 2008) implemented using MATLAB programming language (NIST test.m). The randomness results of the proposed generator for first 1000 and 10000 bits are listed in Table 2. According to Singla *et al.* (2014), this generator passes all the NIST tests for 100,000 samples. But for my simulation results the sequence of generated bits didn't pass all these tests for 1000 and even 10000 bits, it failed in at least one p-value. But most of my tests output the p-values obtained were >0.01 which guarantees the high arbitrariness and random of the created sequences.

CONCLUSION

After implementation of the PRNG base on CNN using MATLAB and perform several statistical tests on the generated binary sequence, I can summarize the main pros of this algorithm into: this generator uses the high sensitivity and randomness property of chaotic functions (Piece-wise linear chaotic map). The four-layer neural network increase the nonlinear complexity of the generator. The key space proposed in this simulator is (128 bit) where:

- The 64 bit key used to initialize the network components
- The 64 bit seed used as an input to the network

According to Singla *et. al.* (2014) and my implementation of some of the NIST randomness tests, this generator passes most of the NIST tests. The generated sequence pass equality distribution (numbers equals to 0's and 1's), i.e., uniform distributed. It satisfy the two necessary suitable requirements for a sequence of random number (Randomness).

While the main cons of the proposed generator in my point of view: this scheme is not efficient because of the relatively large number of iteration steps involved in its implementation. Difficult hardware implementation. The learning rate which has critical effect of the neural network performance didn't adopt in this architecture. This makes the weight adaptation relatively unstable or oscillated. It's difficult to estimate the period of the sequence because the time of number iterations in each layer depends on the initial conditions which is generated by the key. In other words, the key and seed values effect on the performance of the generator.

ACKNOWLEDGEMENTS

I acknowledge with thanks to Middle Technical University, Electrical Engineering Technical College for valuable guidance, keen interest and encouragement at various stages of my papers period research. Also, special thanks are extended to the Head of Department of Engineering Computer Science in UPB University polytechnic Bucharest Prof. Dr. Eng. Paul Cristea for sharing his pearls of wisdom with me during learning the algorithms of this study. We acknowledge assistance provided by Ass. Prof. Dr. Ammar Hussien Mutlaq, H.O.D Electrical Engineering Technical College and the rest of the staff for their coments on an earlier version of the manuscript, although, any error are our own and should not tarnish the reputation of these esteemed persons.

REFERENCES

Akhshani, A., A. Akhavan, A. Mobaraki, S.C. Lim and Z. Hassan, 2014. Pseudo random number generator based on quantum chaotic map. Commun. Nonlinear Sci. Numer. Simul., 19: 101-111.

Anonymous, 2014. Lorenz system. Wikimedia Foundation, ýSan Francisco, Californiaý, USA. https://en.wikipedia.org/wiki/Lorenz_system.

Anonymous, 2019. Chaotic system. WebFinance, Inc., Washington, DC., USA. http://www.businessdictionary.com/definition/chaotic-system.html#ixzz3LujgV3Th.

Arvandi, M., S. Wu and A. Sadeghian, 2008. On the use of recurrent neural networks to design symmetric ciphers. IEEE. Comput. Intell. Mag., 3: 42-53.

Bafghi, A.G., R. Safabakhsh and B. Sadeghiyan, 2008. Finding the differential characteristics of block ciphers with neural networks. Inf. Sci., 178: 3118-3132.

Behnia, S., A. Akhavan, A. Akhshani and A. Samsudin, 2011. A novel dynamic model of pseudo random number generator. J. Comput. Appl. Math., 235: 3455-3463.

Chatzidakis, S., P. Forsberg and L.H. Tsoukalas, 2014. Chaotic neural networks for intelligent signal encryption. Proceedings of the 5th International Conference on Information, Intelligence, Systems and Applications (IISA 2014), July 7-9, 2014, IEEE, Chania, Greece, ISBN:978-1-4799-6171-9, pp: 100-105.

Ertugrul, O.F., 2014. A novel approach to synchronization problem of artificial neural network in cryptography. Am. Assoc. Sci. Technol., 1: 27-32.

Guler, U. and S. Ergun, 2010. A high speed IC random number generator based on phase noise in ring oscillators. Proceedings of the 2010 IEEE International Symposium on Circuits and Systems, May 30-June 2, 2010, IEEE, Paris, France, ISBN:978-1-4244-5308-5, pp: 425-428.

Hsiao, F.H., Y.T. Tsai, K.P. Hsieh and Z.H. Lin, 2014. Fuzzy control for exponential H^8 synchronization of chaotic cryptosystems using an improved Genetic algorithm. Proceedings of the 11th IEEE International Conference on Control and Automation (ICCA), June 18-20, 2014, IEEE, Taichung, Taiwan, pp: 192-197.

Jhajharia, S., S. Mishra and S. Bali, 2013. Public key cryptography using neural networks and genetic algorithms. Proceedings of the 2013 6th International Conference on Contemporary Computing (IC3), August 8-10, 2013, IEEE, Noida, India, ISBN:978-1-4799-0192-0, pp: 137-142.

Joshi, S.D., V.R. Udupi and D.R. Joshi, 2012. A novel neural network for digital image data encryption/decryption. Proceedings of the 2012 International Conference on Power, Signals, Controls and Computationn (EPSCICON), January 3-6, 2012, IEEE, Thrissur, India, ISBN:978-1-4673-0446-7, pp: 1-4.

Kostenko, P.Y., A.N. Barsukov, A.V. Antonov and S.I. Sivachinko, 2009. Recovery of binary message, masked with derivative of Mackey-Glass chaotic process. Radioelectronics Commun. Syst., 52: 89-92.

Kotlarz, P. and Z. Kotulski, 2007. Neural Network as a Programmable Block Cipher. In: Advances in Information Processing and Protection, Pejas, J. and K. Saeed (Eds.). Springer, Boston, Massachusetts, ISBN:978-0-387-73136-0, pp: 241-250.

Li, Y., D. Xiao, S. Deng, Q. Han and G. Zhou, 2011. Parallel Hash function construction based on chaotic maps with changeable parameters. Neural Comput. Applic., 20: 1305-1312.

Lian, S., 2009. A block cipher based on chaotic neural networks. Neurocomputing, 72: 1296-1301.

Mansingka, A.S., A.G. Radwan and K.N. Salama, 2012. Fully digital 1-D, 2-D and 3-D multiscroll chaos as hardware pseudo random number generators. Proceedings of the 2012 55th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), August 5-8, 2012, IEEE, Boise, Idaho, ISBN:978-1-4673-2526-4, pp: 1180-1183.

McEvoy, E., 2009. Using matlab to integrate ordinary differential equations (ODES). Carnegie Mellon University, Pittsburgh, Pennsylvania.

Rukhin, A., J. Soto, J. Nechvatal, M. Smid and E. Barker *et al*., 2001. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publication 800-22, May 15, 2001, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD., USA.

Singla, P., P. Sachdeva and M. Ahmad, 2014. A chaotic neural network based cryptographic pseudo-random sequence design. Proceedings of the 2014 4th International Conference on Advanced Computing and Communication Technologies (ACCT), February 8-9, 2014, IEEE, Rohtak, India, ISBN:978-1-4799-4910-6, pp: 301-306.

Stallings, W., 2011. Cryptography and Network Security: Principles and Practice. 5th Edn., Prentice Hall, USA., ISBN: 9780136097044, Pages: 719.

Viswanath, D., 2004. The fractal property of the Lorenz attractor. Phys. D. Nonlinear Phenom., 190: 115-128.

Yayýk, A. and Y. Kutlu, 2014. Neural network based cryptography. Neural Network World, 2: 177-192.

Yee, L.P. and L.C. De Silva, 2002. Application of multilayer perceptron networks in symmetric block ciphers. Proceedings of the 2002 International Joint Conference on Neural Networks IJCNN'02 (Cat. No.02CH37290), May 12-17, 2002, IEEE, Honolulu, Hawaii, pp: 1455-1458.

Zoghabi, A.E., A.H. Yassin and H.H. Hussien, 2013. Survey report on based on neural network. Intl. J. Emerging Technol. Adv. Eng., 6: 456-462.