VOL. 14, NO. 13, JULY 2019 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

PSEUDO RANDOM NUMBER GENERATOR BASED ON NEURO-FUZZY MODELS

Layla M. Mohammed Ali and Sarab M. Hameed
Department of Computer Science, College of Science, University of Baghdad, Iraq
E-Mail: sarab_majeed@scbaghdad.eud.iq

ABSTRACT

Producing pseudo-random numbers (PRN) with high performance is one of the important issues that attract many researchers today. This paper suggests pseudo-random number generator models that integrate Hopfield Neural Network (HNN) with fuzzy logic system to improve the randomness of the Hopfield Pseudo-random generator. The fuzzy logic system has been introduced to control the update of HNN parameters. The proposed model is compared with three state-of-the-art baselines the results analysis using National Institute of Standards and Technology (NIST) statistical test and ENT test shows that the projected model is statistically significant in comparison to the baselines and this demonstrates the competency of neuro-fuzzy based model to produce a pseudo-random number.

Keywords: fuzzy logic, Hopfield neural network, pseudo random number.

1. INTRODUCTION

Today, telecommunication technologies grow rapidly, especially in the Internet, mobile networks and the domain of information transmission. These developments generate new challenges in the field of protecting information from an unauthorized intruder. It has strengthened the activities of the researchers in the cryptography domain to obtain strong secure cryptographic techniques [1]. Several cryptographic protocols such as electronic payment schemes, non-repudiation, authentication, integrity, privacy or key management require random components [2]. Therefore, random number generators (RNGs) are essential in the cryptographic system. Security in these systems is based on the inability to predict future numbers from observed sequences [3].

The literature shows that there has been a growing attention in the utilization of various types of neural networks and fuzzy logic system regarding pseudo number generation. Yuhua *et al.* used the back propagation neural network (BPNN) for improving the conventional random number generator (RNG). The secure hash algorithm (SHA-2) hash function has been employed to guarantee the randomness of the generated random numbers. The random number quality made by the architecture suggested in this work agreed with the cryptographic system security based on the consequences of standardized by the U.S test suites [4]. Kayvan and Alireza introduced a pseudo-random number generator by using a distinctive feature of Hopfield neural network (HNN) including random performance under definite conditions. A comparison has prepared with ideal random number generators. The national institute of standards and technology (NIST) statistical tests have adopted for determining the performance and for calculating quality of the projected HNN random number generator [5]. Wen *et al.* investigated the exponential lag synchronization mechanism of memristive neural networks (MNNs) problem through a fuzzy procedure for pseudo-random number generator. The MNNs model has been accomplished using a recurrent neural network and the knowledge of memristor. Then, a fuzzy process of MNNs has been used concerning the state-dependent memristor properties providing means for perceiving the complex MNNs by means of dual subsystems only. Slave systems and controller gain connection weights have updated to arrange for the slave systems exponentially lag synchronized with the master systems [6]. Yayik and Yakup implemented a neural network based cryptology. They considered a system with dual stages. In the initial stage, pseudo-random numbers have produced by a neural network. The cryptosystem has been developed using the produced random number in the 2nd stage. This neural network based cryptosystem encrypted and decrypted the data by the two identical artificial neural networks (ANNs) [7]. Sattar *et al.* suggested a fuzzy system for generating a sequence of frequencies for a spread spectrum communication system. The fuzzy rules described the old frequencies distribution by new output frequencies. The proposed fuzzy system experimented with many frequencies and compared with other types of pseudo random number generator (PRNG). The result showed that the proposed fuzzy system provided a more uniform spread than did the other methods [8]. Igor and Khaled proposed a fuzzy synchronous stream cipher system using a composition of a non-linear feedback shift register to attain a pseudo-random. The suggested system was straightforward and appropriate for a number of telecommunication cipher employment. The produced pseudo-random sequence successfully passed the tests of NIST package [9]. Igor and Khaled studied the parameters that affect a pseudo-random number generator based on fuzzy logic (FRNG). FRNG, the size of the buffer and number of membership functions parameters played a crucial role and directly affected the sequence randomness. The best randomness was attained by selecting the best values of these parameters. The results were tested using the NIST test suite and examined with Matlab random generator and sixteen random generators from the Diehard bundle [10].

The proposed work is an extension to the work of [11] and the main contribution in this work is to integrate

Hopfield neural network and fuzzy logic system to solve pseudo random number generation problem. The proposed fuzzy logic system controls the adjustment of Hopfield neural network's parameters and emphasizes the strength of Hopfield neural network in terms of randomness.

This paper is organized as follows: in section 2, background of HNN and fuzzy logic system are presented. The proposed model for pseudo number generation is introduced in section 3. Results and the analysis of the proposed PRNG model are presented and discussed in section 4. Finally, section 5 concludes the proposed work and suggests some further ramifications.

2. PRELIMINARY CONCEPTS

This section gives a brief description to the HNN and fuzzy logic system that have been adopted in this work for solving pseudo number generation problem.

2.1 Hopfield neural network

Hopfield neural networks stand for a solitary layer recurrent networks that have a cycle and the output from a neuron is input for other neurons in the network. However, they don't have self-back feedback. The Hopfield Neural Network has auto-associative memory algorithm to supply advantageous information in memory and later it can replicate this information. HNN employs Hebbian equation to memorize several patterns by relating them with the inputs of the network in the weight matrix, and after some iterations, the network can influence the stable point that is in accordance with the pattern which as been memorized [12].

The output of Hopfield network is set by Equation 1, which is the update equation for the solitary neuron [13].

$$x_i(t+1) = sign(\sum_{j=1}^{n} x_j(t) w_{ij} - \theta + I_i) \quad (1)$$

Where:

x_j: is the output magnitude of the neuron from the preceding iteration.
$\boldsymbol{w_{ij}}$: is the weight matrix unit that links neuron i to neuron j.
$\boldsymbol{I}$: is constant external Input.
$\boldsymbol{n}$: is the neurons number in the network.
$\boldsymbol{sign}$: is an activation function.
$\boldsymbol{t}$: is the iteration number and θ is the threshold.

Some of the important characteristics of a Hopfield network are shown as follow [12] [4]:

a) The weight matrix of an HNN has been symmetric with zero-valued diagonals, and just single neuron is activated per iteration then by means of Energy function that lessens after every iteration until a local optimal objective of this function is found.

b) After the convergence, the output of network has the smallest distance or is precisely equivalent to one pattern put in storage in the network throughout its weight matrix definition.

c) The patterns that put in storage in the network are orthogonal to one another and their number M ≤ 0.15N, where N is the number of the network neurons.

2.2 Fuzzy logic controller

L. Zadeh 1965 first proposed the fuzzy set theory that is essentially concerned with quantifying and reasoning using natural language. It is considered ~~of~~ as an expansion of the traditional crisp set [14]. Fuzzy logic produces a simplistic method to perform a certain conclusion based on imprecise, ambiguous, noisy, or missing input information. [15].

The fuzzy logic controller predefines membership functions and fuzzy inference rules to map numeric data into linguistic variable terms [14]. A typical fuzzy logic controller (FLC) contains three basic components: fuzzification, inference engine, and defuzzification modules. The fuzzification module transforms the crisp value of the input signal into suitable linguistic values that represent terms or sentences from a natural language, rather than numerical values based on the fuzzy set [16].

The second module of FLC is the inference engine that makes the decision based on a fuzzy rule according to the information from the fuzzification module. A fuzzy inference system involves linguistic variables, fuzzy rules, and a fuzzy inference mechanism. Fuzzy rules are a collection of rules, which produce an association between input and output data. A fuzzy rule is a single IF-THEN rule, which holds a condition and a conclusion. Finally, the defuzzification model converts a fuzzy output into a crisp value. There are several defuzzification methods and selecting a suitable one is based on the given application [17].

3. THE PROPOSED NEURO-FUZZY BASED PRNG

The most important target for any PRNG is producing the pseudo-random number that is high in randomness and in degree of independence. The proposed PRNG based on HNN in [11] has some restrictions in the length of sequence that depends on the number of neurons, and the seed of generator. In the Hopfield, the weight matrix is the seed that means the network with N input neurons has weight matrix with $N \times N$ units and this seed is huge. Therefore, this work is an extension of the work in [11] to override HNN limitation. The proposed model utilizes fuzzy logic controller to support the Hopfield network as in what follows:

a) Satisfying the challenge represented by randomness handling of PRN.

b) Generating the same sequence length by a less number of neurons.

c) Extracting the five last digits of the fraction part for each neuron output and the mod 8 operation is performed for each digit. Then the output is converted to 3 bits, binary representation that helps to produce a large number of bits with the same number of neurons without affecting randomness performance.

In the proposed fuzzy logic, the outputs of the Hopfield neural network are passed through a buffer. The purpose of the buffer is to make the output of the network to be convenient for FLC. Two statistical metrics: mean (m) and run test (r)have been adopted to assess each buffer**.**

The mean metric denotes the average number of 1's in the buffer and r represents the $p-value$[18] of run test in the buffer. The main steps for calculating the run test are described in Algorithm 1.

Algorithm 1: calculation the P-Value of Run test	
Input: ▪ The sequence of bits in the buffer with length l.	
Output: ▪ P-Value of Run test.	
1	Calculate the number of 1's, n_{one}.
2	Calculate the pre-test proportion(p_{test}): $p=\frac{n_{one}}{l}$.
3	Calculate the test statistic $v_l=\sum_{k=1}^{l-1} r(k)+1, r(k)=\begin{cases}0 & if\ bit_k=bit_{k+1}\\ 1 & Otherwise\end{cases}$
4	Calculate $p-value=erfc\left(\frac{\lvert v_l-2lp_{test}(1-p_{test})\rvert}{2\sqrt{2l}p(1-p_{test})}\right)$, where $erfc$ is error function.

After that, the resulting two statistical metrics, namely, r and m of the output neuron are used as inputs to the fuzzification component. The first input variable, r, is assigned in terms of its linguistic variable into three fuzzy sets that stand for low (L), medium (M) and high (H). The second input variable, m, is categorized into five fuzzy sets, namely, very low (VL), low (LW), medium (MM), high (HH) and very high (VH). The output variable o, is assigned in terms of its linguistic variable by using three fuzzy sets that are represented by bad (BD), good (GD) and best (BT). The triangular membership function is presented in Equation (2) [19]. The two inputs $r$and$m$ and output variable o are depicted in Figure-1.

$$trimf(x;a,b,c)=\max\left(\min\left(\frac{x-a}{b-a},\frac{c-x}{c-b}\right),0\right) \quad (2)$$

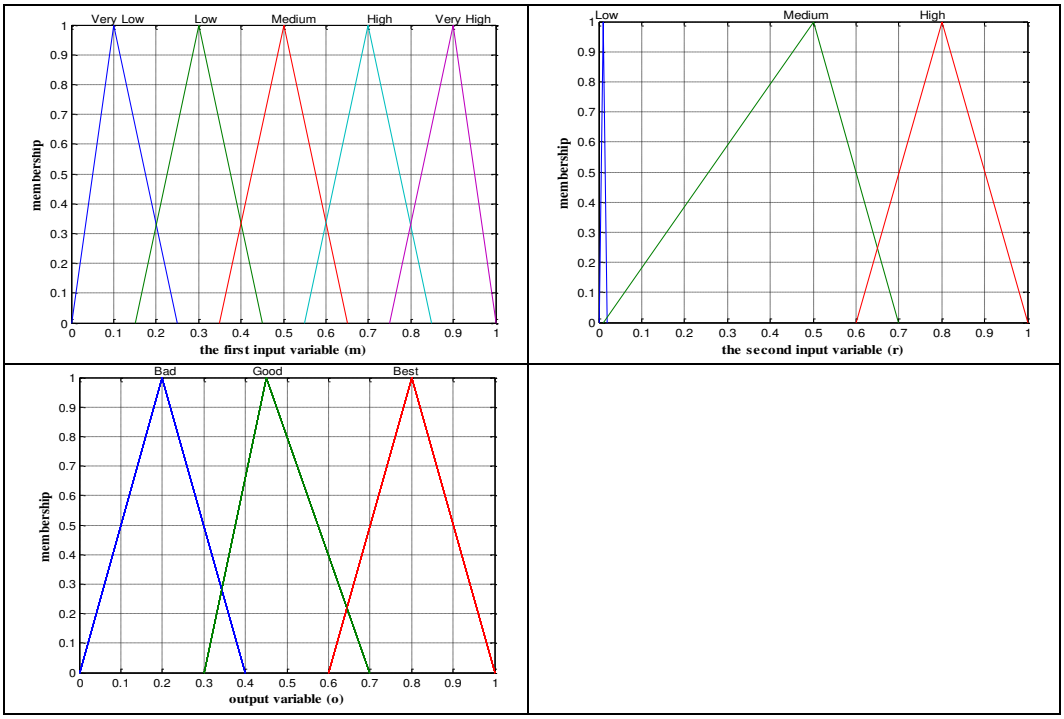

Figure-1. Membership functions of a. the first input variable(m), b. the second input variable (r) and c. the output variable(o).

Then, at inference engine component, the following if-then rules have been proposed to examine the output of the neuron as summarized in Table-1 with r and m as inputs whereas o as the output. Finally, in the defuzzification component, the center of gravity function in Equation (3) [16] is applied to calculate the crispy output, c.

$$COG(x) = \frac{\sum_{i=1}^{n} x_i.\mu_c(x_i)}{\sum_{i=1}^{n} \mu_c(x_i)} \quad (3)$$

Table-1. The proposed fuzzy rules

Second input (m)	First input (r)		
	L	**M**	**H**
LW	BD	BD	BD
VL	BD	BD	GD
MM	BD	GD	BT
HH	BD	BD	GD
VH	BD	BD	BD

Two models that combine HNN to the fuzzy logic controller have been suggested for updating the HNN parameters. The first model named as FSHNN-PRNG attempts to control the HNN parameters synchronously (i.e. all neurons in the network are updated). While the second model named as FAHNN-PRNG controls the parameters asynchronously, (i.e. one-neuron is updated per iteration).

3.1 The proposed FSHNN-PRNG model

When the Hopfield updates their neurons in a synchronous way, all neurons inputs and their weights are updated in each iteration. for the proposed FSHNN-PRNG model, the updating of HNN parameters is under the domination of fuzzy logic controller by passing the new and previous output of all neurons of HNN to the two buffers. The first buffer holds the new output and the second one contains the previous output. Then, checking the validity of a neural output is determined by evaluating the quality of buffers content using two metrics: run test $r$and mean$m$. These values are passed to the proposed fuzzy logic controller to determine which one of them has the best statistical characteristics. After that, the "good" or "best" quality output that is fittest is selected to be the output of the HNN in the next iteration and the weight of "bad" output is updated to upgrade that output as shown in Figure-2. The crispy value generated as an output of the fuzzy logic controller is used to control the updating of the HNN weighs as in Equation 4.

$$\forall i \in \{1, \dots, N\}, \forall j \in \{1, \dots, N\}$$

$$w_{ij} = \begin{cases} w_{ij} + \beta & if\ i \geq j \\ w_{ij} - \beta & Otherwise \end{cases} \quad (4)$$

Where

$\beta = \frac{c}{1000} \times \frac{y_i}{y'_i}$

y_i is the output of neuron i.

y'_i is the previous output of neuron i.

Updating neural network parameters under the control of fuzzy logic continues until a maximum number of iterations (Max_{itr}) is reached or the output of fuzzy logic becomes "best" as revealed in Figure-2.

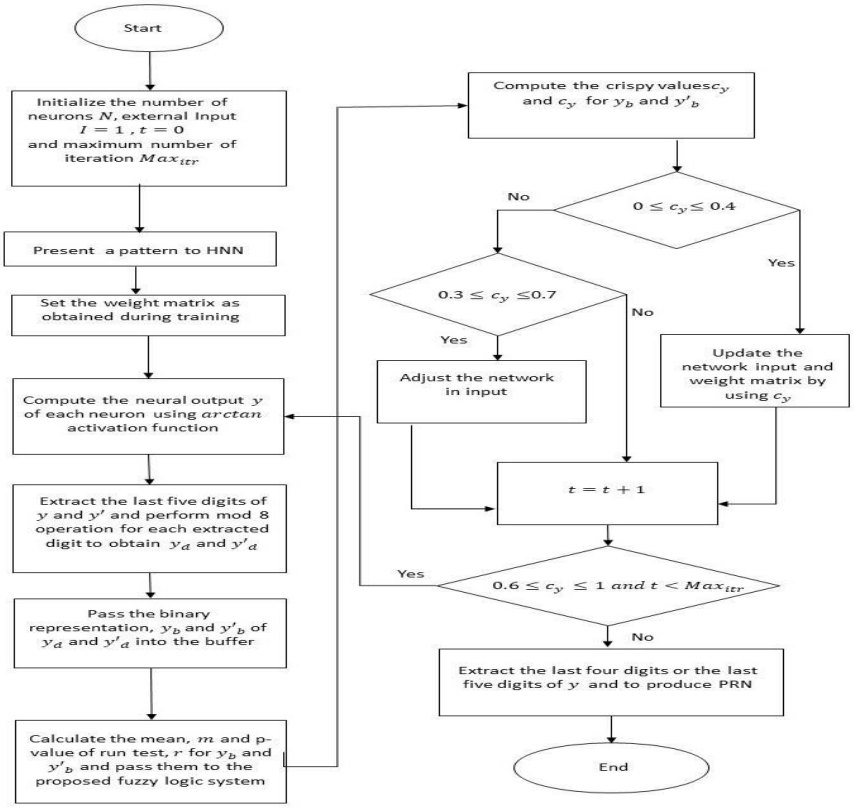

Figure-2. Flowchart of the proposed FSHNN-PRNG model.

3.2 The proposed FAHNN-PRNG model

In the asynchronous updating, one neuron is picked at each iteration. Then the output of this neuron is calculated and its state is updated accordingly (i.e. state of other neurons remains unchanged). The proposed FAHNN-PRNG model works almost in the same way as the FSHNN-PRNG model. The differences between these two models are as follows:

FAHNN-PRNG model picks one neuron at a time and compare it with other neurons. Then the outputs of those two neurons are passed into two buffers. Size of the buffer in this model depends on the number of extracted digits from each output, unlike FSHNN-PRNG that has buffer size equal to sequence length of network's output.

In the updating step, the crispy value of each neuron is compared with each one of the other neurons, if the active neuron is better than other neurons. Then, the active neuron output passes as an input for the next iteration. On the other hand, if the other neurons are better than the active neuron, the HNN weight matrix is updated as in Equation 5.

$\forall i \in \{1, \dots, N\} \wedge k$ is an active neuron.

$$w_{ki} = \begin{cases} w_{ki} + \beta & if\ k \geq i \\ w_{ki} - \beta & Otherwise \end{cases} \quad (5)$$

Where

$\beta = \frac{c}{1000} \times \frac{y_k}{y'_i}$

y_k is the output of active neuron.

y'_i is the output of i^{th} neuron.

The network continues until a maximum number of iterations (Max_{itr}) is reached or the network becomes stable. Number of iterations of this model is more than FSHNN-PRNG model because the active neuron requires to be compared with all other neurons. In this model, each neuron is compared with one neuron each time and the size of the buffer for this model depends on the number of the extracted digits. This means that the sequence is divided into small blocks and each time FLC tries to improve the block sequence and in the overall effect, FLC becomes clearer and stronger than the previous model.

4. EXPERIMENTAL RESULTS AND DISCUSSION

This section examines the performance of the two models proposed in this paper for generating pseudo-random number. The performance of the proposed models is evaluated against other existing methods, Blum Blum Shub (BBS), Linear Congruential generator (LCG) and Quadratic Congruential-I generator(QCG-I)using the national institute of standards and technology (NIST) [19] statistical test and ENT [20] test. Furthermore, six tests of NIST test specifically: frequency, block frequency, cusum-forward, cusum-reverse, run, and FTT have been adopted as an evaluation metrics and the remaining tests are not used since they need more than 1500 bits for satisfactory operation. The experiments have been conducted under five runs and the average of five runs is reported for each of the proposed models.

The performance of FSHNN-PRG model that updates the HNN parameters synchronously and FAHNN-PRNG, which uses the asynchronous updating in the HNN under control the fuzzy logic system clearly are pointed out that the both models provide good performance in terms of randomness and they passes all tests successfully compared with other baseline methods as shown in Table-2. As obviously noticed, the proposed FSHNN-PRG and FAHNN-PRNG models have produced more randomness in comparison with other baseline methodswith varying sequence length, $l = \{1500,1200,900,600\}$.

Results in Table-3 illustrate and summarizes the performance of the proposed models with varying sequence length, $l = \{1500,1200,900,600\}$concerning ENT test. Moreover, as illustrated previously, FAHNN-PRNG model in most tests has the best performance compared to FSHNN-PRNG model except the block frequency test. The results show that reducing the number of neurons affects the performance of FSHNN-PRNG model and it makes it performs better than FAHNN-PRNG model and it is observed that this is associated with buffer size. In FAHNN-PRNG, the size of the buffer does not change and does not depend on the number of neurons. Though, in FSHNN-PRNG, the size of the buffer depends on the sequence length. Therefore, decreasing the number of neurons leads to reducing the generated sequence length. This means that the fuzzy logic controller copes with a short buffer size.

Also, the results clearly elucidate that the proposed models with fewer number of neurons could generate good sequences in terms of randomness which is better than the sequences that are generated with more number of neurons when using HNN alone for generating random number.

Table-2. Comparison regarding p-value of the FSHNN-PRNG model and FAHNN-PRNG against BBS, LCG and QCG-I.

l	Model	Frequency	Block Frequency	Cusum-Forward	Cusum-Reverse	Runs	FTT
1500	FAHNN-PRNG	0.504222	0.671187	0.572812	0.565956	0.801661	0.079265
	FSHNN-PRNG	0.396514	0.733063	0.497414	0.479928	0.783373	0.105828
	BBS	0.213309	0.739918	0.122325	0.035174	0.739918	0.035174
	LCG	0.911413	0.739918	0.534146	0.066882	0.350485	0.035174
	QCG-I	0.534146	0.534146	0.350485	0.350485	0.534146	0.000199
1200	FAHNN-PRNG	0.739918	0.529224	0.760506	0.593442	0.584145	0.002169
	FSHNN-PRNG	0.627741	0.655176	0.730629	0.453771	0.415247	0.00232
	BBS	0.122325	0.739918	0.739918	0.035174	0.534146	0.000199
	LCG	0.350485	0.350485	0.350485	0.035174	0.350485	0.000001*
	QCG-I	0.350485	0.350485	0.534146	0.122325	0.739918	0.000199
900	FAHNN-PRNG	0.739918	0.739918	0.066882	0.350485	0.739918	0.066882
	FSHNN-PRNG	0.350882	0.614022	0.758229	0.705486	0.760506	0.007804
	BBS	0.350485	0.534146	0.739918	0.213309	0.534146	0.002043
	LCG	0.122325	0.122325	0.350485	0.350485	0.739918	0.000954
	QCG-I	0.534146	0.350485	0.534146	0.122325	0.213309	0.035174
600	FAHNN-PRNG	0.739918	0.213309	0.350485	0.911413	0.911413	0.000000*
	FSHNN-PRNG	0.33	0.648321	0.584145	0.627741	0.877114	0.000001*
	BBS	0.534146	0.350485	0.739918	0.350485	0.911413	0.000003*
	LCG	0.350485	0.213309	0.066882	0.350485	0.911413	0.000003*
	QCG-I	0.350485	0.739918	0.350485	0.350485	0.534146	0.000000*

Table-3. ENT test results of the proposed models for different sequence length.

l	N	d	Model	ENT Test				
				Entropy	Chi-square	Mean	Monte-Carlo-Pi	Serial-Correlation
1500	100	5	FSHNN-PRNG	0.89012	2374.681	0.307375	4.000	0.069967
		5	FAHNN-PRNG	0.891798	2339.370250	0.308812	4.000	0.065636
1200	80	5	FSHNN-PRNG	0.891342	1879.174562	0.308422	4.000	0.066813
		5	FAHNN-PRNG	0.892378	1861.729375	0.309313	4.000	0.064131
900	100	3	HNN-PRNG	0.889652	1430.71575	0.306979	4.000	0.071161
	60	5	FSHNN-PRNG	0.890898	1414.979667	0.308042	4.000	0.067958
		5	FAHNN-PRNG	0.891914	1401.806917	0.308938	4.000	0.06526
600	100	2	HNN-PRNG	0.891318	939.783875	0.308406	4.000	0.066862
	40	5	FSHNN-PRNG	0.889082	958.6	0.3065	4.000	0.072608
		5	FAHNN-PRNG	0.892194	932.413625	0.309718	4.000	0.064602
d is the number of extracted digits from output								

5. CONCLUSIONS

Any pseudo-random number generator has the challenge of producing sequence numbers with high quality in terms of randomness and independence. The main contribution of this paper is to develop pseudo-random number generator models with two significant aims: generation of numbers with high randomness and with large sequence. Two models for generating pseudo-random number based on combining have been proposed. The overall results reveal that the proposed models surpass other baseline methods. The proposed models built on coupling HNN and fuzzy logic have ensured their convenience to produce a large sequence of bits equivalent to 1500 bits with high performance in terms of randomness. In addition, they have the capability of producing the same sequence length produced by the other models using less number of neurons and gaining best performance in terms of randomness. In the proposed fuzzy Hopfield models, it is shown that altering the number of neurons doesn't have an impact on the degree of randomness and the randomness doesn't affect by the reduction of neurons number. For future work, an interesting research direction that can be adopted is to investigate other features of HNN such as activation function to generate a pseudo- random number. Also, one possible direction is to apply other statistical tests to control the HNN parameter.

REFERENCES

[1] Patidar V., Krishan K. and Narendra K. 2009. A pseudo random bit generator based on chaotic logistic map and its statistical testing. Informatica. 33(4): 441-452.

[2] Dimitriis A. and Vasilios Z. 2003. Improving pseudorandom bit sequence generation and evaluation for secure Internet communications using neural network techniques. The International Joint Conference on. 2: 1367-1372. IEEE.

[3] Chen S. and Cenjun Z. 2018. FPGA implementation of SRAM PUFs based cryptographically secure pseudo-random number generator. Microprocessors and Microsystems. 59, pp. 57-68.

[4] Yuhua W., Guoyin W. and Huanguo Z. 2010. Random Number Generator Based on Hopfield Neural Network and SHA-2 (512). Advancing Computing, Communication, Control and Management. pp. 198-205.

[5] Kayvan T. and Alireza S. 2010. Hopfield Neural Networks as Pseudo Random Number Generators. in North America Fuzzy Information Processing Society (NAFIPS), Toronto. pp. 1-6.

[6] Wen S., Zhigang Z., Tingwen H., and Yide Z. 2014. Exponential adaptive lag synchronization of memristive neural networks via fuzzy method and applications in pseudorandom number generators. IEEE Transactions on Fuzzy Systems. 22(6): 1704-1713.

[7] Yayik A. and Yakup K. 2014. Neural network based cryptography. Neural Network World. 24(2): 177.

[8] Sattar B., Sawsan K. and Najwan A. 2013. Fuzzy Based Pseudo Random Number Generator used for Wireless Networks. Journal of Al-Nahrain University-Science. 16(2): 210-216.

[9] Anikin, I. and Khaled A. 2015. Fuzzy stream cipher system. International Siberian Conference on Control and Communications, SIBCON, 2015.

[10] Anikin I. and Khaled A. 2016. Pseudo-random number generator based on fuzzy logic. In Control and Communications (SIBCON), 2016 International Siberian Conference on, pp. 1-4. IEEE.

[11] Sarab M. and Layla M. 2018. Utilizing Hopfield Neural Network for Pseudo-Random Number Generator. 15th ACS/IEEE International Conference on Computer Systems and Applications, Jordan Aqaba, pp 1-5 AICCSA.

[12] Shi X., Shukai D., Lidan W., Tingwen H. and Chuandong L. 2015. A novel memristive electronic synapse-based Hermite chaotic neural network with application in cryptography. Neurocomputing. 166, pp. 487-495.

[13] Freeman A. and David M. S. 1991. Algorithms, Applications, and Programming Techniques. Addison-Wesley Publishing Company, USA.

[14] Hong, Tzung-Pei and Chai-Ying Lee. 1996. Induction of fuzzy rules and membership functions from training examples. Fuzzy sets and Systems. 84(1): 33-47.

[15] Qaid Gamil R. and Sanjay N. 2013. Encrypting image by using fuzzy logic algorithm. International Journal of Image Processing and Vision Sciences (IJIPVS). 2(1): 2278-111.

[16] Barnabas B. 2013. Mathematics of fuzzy sets and fuzzy logic. Springer.

[17] Subiyanto S., Azah M. and Hannan M. 2012. Intelligent maximum power point tracking for PV system using Hopfield neural network optimized fuzzy logic controller. Energy and Buildings. 51, pp. 29-38.

[18] NIST. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications (2010, April). [Online]. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22.pdf

[19] Passino K., Stephen Y. and Michael R. 1998. Fuzzy control. Vol. 20. Menlo Park, CA: Addison-wesley.

[20] J. Walker. 2018. ENT. Fourmilab Switzerland, 28 January (2008). Available: http://www.fourmilab.ch/random/ (accessed 1-2-2018).