NEURAL NETWORK BASED CRYPTOGRAPHY

*Apdullah Yayık, Yakup Kutlu**

Abstract: In this paper, neural network based cryptology is performed. The system consists of two stages. In the first stage, neural network-based pseudo-random numbers (NPRNGs) are generated and the results are tested for randomness using National Institute of Standard Technology (NIST) randomness tests. In the second stage, a neural network-based cryptosystem is designed using NPRNGs. In this cryptosystem, data, which is encrypted by non-linear techniques, is subject to decryption attempts by means of two identical artificial neural networks (ANNs). With the first neural network, non-linear encryption is modeled using relation-building functionality. The encrypted data is decrypted with the second neural network using decision-making functionality.

Key words: *Artificial neural network, asymmetric cryptology, pseudo-random number generator*

Received: November 11, 2013 **DOI:** 10.14311/NNW.2014.24.011
Revised and accepted: April 14, 2014

1. Introduction

Cryptography uses mathematical techniques for information security. Information security is now a compulsory component of commercial applications, military communications and also social media implementation. This is a result of the many threats and attacks that can be made to these networks by people with malicious intent. Cyber-terrorists, crackers, hackers, so-called 'script kiddies' and industrial spies are all masters in the manipulation of information systems [17]. Cryptography is, furthermore, the most significant part of communication security [3]. It maintains the confidentiality that is the core of information security. Any cryptography requires confidentiality, authentication, integrity and non-repudiation from those authorized to have it. Authentication relates to the identification of two parties entering into communication, while integrity addresses the unauthorized modification of an element inserted into the system [23]. To date, there has been a large number of studies intended to advance robust cryptosystems and use them in communications. Some of these studies concerned the usage of neural networks in

*Apdullah Yayık, Yakup Kutlu, Mustafa Kemal University, Department of Computer Engineering, Turkey, E-mail: apdullahyayik@gmail.com, ykutlu@mku.edu.tr

cryptography. The concept of neural-based cryptography was first introduced by Laurie in 1990 [15]. In 1993, neural networks' weights were used as secret keys by Tanrıverdi [28]. Neural cryptography applications were researched by Pointcheval in 1994 [20]. Following this, a practical study of neural cryptography appeared in 1996 [24]. Cin also used neural networks for authentication [5]. A symmetric probabilistic encryption scheme based on the chaotic attractors of neural networks was studied by Guo et al. [11]. Chaotic neural encryption and decryption was studied by Sue et al. [26]. The Hopfield network and its applications were studied by Chi-Kwong and Cheng [4]. The implementation of an MLP network using its one-way property in the hash function was investigated by Yee and Silva [33]. Karras and Zorkadis presented an ANN-based technique to strengthen traditional generators such as IDEA and ANS X.9. Neural network-based pseudo-random stream generators have also been evaluated [14]. Godhavari et al. applied neural synchronization and shared weights values as a symmetric secret key in the DES algorithm [10]. Ruttor performed the neural synchronization of two tree parity networks with mutual outputs by means of the hebbian learning rule; weight and bias values were then used as a secret key [22]. Arvandi et al. presented an ANN-based symmetric chipper method [3]. Sağıroğlu and Özkaya studied ANN security for electronic communications with an algorithm developed by DELPHI, which uses ANNs as a secret key [23]. Gnanam and Munukur presented a method that aimed to remove the need for encoding to follow a general rule through use of ANNs. According to Gnanam and Munukur, there is a way to decode the encrypted data. However, the plain texts are encoded [17]. İlker and Kenan presented an ANN-based chaotic generator to overcome the weakness of chaotic cryptosystems [7]. Sivagurunathan presented a neural network that classifies chipper texts encrypted with Playfair, Vigenere and Hill chipper [25]. Mohammad and Babak proposed an ANN-based S-box design scheme [18]. Othman and Jammas developed stream chipper cryptosystem software and hardware in FPGA, a pseudo-random number generator designed using ANNs [19]. Yayık and Kutlu proposed a neural network model for improving the randomness of a classical random number generator in 2013 [31]. Yayık and Kutlu proposed a neural network model for hash function that can be used in electronic signature, in 2013 [30].Yayık and Kutlu proposed a neural network model for asymmetric cryptosystem model that weights, bias and topology of NN are shared between receiver and sender as secret key, in 2013 [32]. In this study, a neural network-based pseudo-random number generator and a neural network-based non-linear cryptosystem are developed. This study is implemented in five chapters. The first chapter gives information about information security, cryptology and existing literature. In the second chapter, the artificial neural network (ANN), the advantages of neural networks for cryptology, pseudo-random number generators, neural-based pseudo-random number generators and randomness tests are all explained. In the third chapter, the implementation of neural cryptology is explained step by step. In the fourth chapter, the results and conclusions are given. Finally, in the fifth chapter, discussions are presented.

2. Material and Methods

2.1 Artificial Neural Network

The human body renovates itself and generates a variety of reactions against the incidents that occur during physical, biological or chemical changes, in order to become used to them. The structures of this system, which complement with each other, have inspired a great number of scientists. Variations occurring in our environment are detected by neurons in our bodies, which are then transmitted to our brains. The brain works as a decision-maker, alerting sub-systems to produce the optimum reaction. In other words, the neuron system in the human body is a wonderful structure that carries out sensing, decision-making and practicing functions [6]. As in nature, connections between artificial neural networks determine the network's function. A new neural network can be designed by changing the connections between elements (weights). The weights of this network can be changed; in other words, the system can be trained until it obtains optimum output values by using optimum input values [1]. Neural networks are, therefore, adjusted or trained based on a comparison of the desired target and the output, until the network output matches the target [12]. Multilayer perceptron (MLP) networks are feed-forward and supervisor-training algorithm structures that have several middle layers. In this network, a large number of training algorithms can be used. The amount of neurons in the input, output and middle layers change according to problems or complications, so that the number is defined by experience. In MLP networks, error between expected and achieved target (output) value is minimized by changing weights until a determined iteration number is reached [6].

2.2 Advantages of Neural Network for Cryptology

Neural networks' most important property is their generalization capability. This ability ensures they produce reasonable results when they are fed with inputs not previously encountered. This makes them extremely useful for many applications. Assuming that x_k denotes inputs of a network and denotes targets of a network, it is easy to compute y_k from x_k. But if target y_k is different from input x_k, it is difficult to compute the input from the target. As a result of this property, hash functions can be generated using ANNs.

$$y_k = \theta \left(\sum_{j=1}^{m} w_k x_i + b_k \right) \tag{1}$$

The other important property of ANNs is parallel implementation. Each layer is paralleled, so they can independently implement certain functionality. A special property of neural networks is confusion, which is caused by the non-linear structure of networks. The output, therefore, depends on the input in non-linear and complicated cases. Thus, it is not easy to define the exact input. As a result of this confusion property, NNs could be preferable for cipher design [18].

2.3 Pseudo-Random Number Generators

Pseudo-random number generators (PRNGs) are deterministic functions. A state is generally mapped to a new state x using an update function in order to generate pseudo-random data. Pseudo-random numbers are used in many fields; these include stochastic physical and statistical simulation, computer science, cryptography, etc. The resultant numbers cannot be said to be truly random on account of the deterministic nature of these functions. Therefore, the primary aim of PRN generation is to create truly random numbers which have statistically identical values [13, 16]. There are many studies in the literature on this subject. Vernam invented a simple one-time pad in which the secret key is a sequence of randomly generated hits [35]. PRNG is an algorithm used to generate a sequence of numbers which approximates the properties of randomness. The security of algorithms which use PRNGs are based on the assumption that it is infeasible to distinguish when a random sequence is being used and when a PRNG.

2.4 Neural-Based Pseudo-Random Number Generator

The neural network is a well-known method that has function approximation capabilities. This makes ANNs a useful tool in many scientific disciplines. For example, the over-fitting of an ANN could be used as a method for the generation of strong pseudo-random bit sequences [14]. In this paper, an approach for creating effective random number generators for use in the security mechanisms of cryptology is described. It is based on ANN techniques. Pseudo-random numbers that are generated by a modified subtract with borrow generator [2] have a long period sequence and are used as the inputs in a neural network structure. After training with initial values (weights and bias), the reached output is called neural-based pseudo-random number. A sequence of pseudo-random numbers is generated using an ANN. They are evaluated by utilizing the statistical tests presented in the next section.

2.5 NIST (National Institute of Standard Technology) Test For Randomness

A statistical randomness test is developed to test a null hypothesis (H_0) which states that the input sequence is random. The test takes a binary sequence as an input and "accepts" or "rejects" the hypothesis. Randomness tests are probabilistic. There are two types of error: if the data is random and H_0 is rejected, a type I error has occurred; if the data is non-random and H_0 is accepted, a type II error has occurred. The probability of a type I error is called the level of significance and is denoted by α. A statistical test produces a real number between 0 and 1 which is called p-value. If p value $> \alpha$ then H_0 is accepted; otherwise it is rejected. Therefore, the level of significance varies according to application. However, for cryptographic applications it is usually set to 0.01 [27].

There are several statistical tests, such as the Diehard test suite [13], John Walkers ENT [29], Test01 [29] and the NIST statistical test suite [21]. The NIST statistical test suite is the most popular one. It was developed by the National Institute of Standards and Technology. The NIST test suite is the preferred choice

among these randomness tests because of its proven results and its popularity in many studies [31, 27, 34, 8, 9]. Randomness is a probabilistic property. A random sequence's properties can be characterized in term of probability.

The NIST test suite [21] consists of 15 tests. It is used to test the randomness of binary sequences. These binary sequences are generated by either a hardware- or a software-based generator. The tests focus on a variety of different types of non-randomness which could occur in a sequence. Some tests can be broken down into a variety of subtests.

The tests can be separated into the following:

The Frequency Test: The goal of the test is to define whether zero and one bits appear in the tested sequence with approximately the same probability. It is a simple test that can show the clearest deviations from randomness. Further tests, therefore, depend on this result [27].

Frequency Test Within a Block: This test is a generalization of the frequency test. The frequency of zeros and ones within M-bit blocks is determined. It establishes whether zeros and ones are uniformly distributed throughout the tested sequence [27].

Runs Test: The purpose of this test is to ascertain whether transitions between zeros and ones in the sequence appear as frequently as expected, based on a random sequence. The total number of runs of various lengths is counted. A run is an uninterrupted sequence of identical bits [27].

Longest Run of Ones in a Block Test: The purpose of the test is to define whether the length of the longest run of ones in a block is consistent with the length expected, based on a random sequence. The sequence for this test is processed in M-bit blocks [27].

Cumulative Sum Test: This test focuses on the maximal excursion (from zero) of the random walk, which is determined by the cumulative sum of adjusted (-1, +1) digits in the sequence. The aim is to define whether the cumulative sum of the partial sequences is too large or too small, relative to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. The excursions of the random walk should be near zero in order to define a random sequence. The excursions of this random walk from zero will be large in certain types of non-random sequences [21].

Discrete Fourier Test: This test focuses on the peak heights in the discrete Fourier transform of the sequence. The purpose is to detect periodic features in the selected sequence during testing. The tested sequence may show a deviation from the assumed randomness. The aim of the test is to detect whether the number of peaks exceeding the 95% threshold is significantly different than 5% [21].

3. Implementation of Neural Cryptology

In neural cryptology, two neural networks that have the same topology (layer size, transfer function, neuron number in each layer, weight and bias values) can achieve the same output when trained for the same input. In other words, two networks which are trained on their mutual input can synchronize with mutual synaptic weights. This has been applied in several studies [17, 3, 10, 22, 7, 18, 19, 35]. To implement this ability for cryptosystems, two partners (receiver and sender) have

to share mutual topological data and chipper text as a secret key. In this study, in order to decrypt, the ability to predict unforeseen situations using an artificial neural network is utilized and a neural-based cryptosystem is constructed. There are six steps in the proposed system: the input step; the neural-based pseudo-random number; non-linear encryption; design of the neural network topology; sending neural network topology and chipper text, and simulation of the neural network and decryption. The first step is the input step in which data or files including strings, numbers or punctuations are entered. In the next step, neural-based pseudo-random numbers are generated and tested using NIST randomness tests. The third step is non-linear encryption, which includes the processes of converting plain text to ASCII codes, converting ASCII codes to binary codes, mixing digit numbers of each string's binary codes and mixing digit numbers of all strings' binary codes, in order to create neural-based pseudo-random numbers. In this step, chipper text is created. The next step is designing the neural network topology relating to neural-based pseudo-random numbers. The fifth step is the sending neural network topology and chipper text. The final step is the simulation of neural network and decryption. This entails designing the topological neural network using sent data, simulating chipper text with sent weights and bias and converting output first to decimal numbers and then to strings or punctuations. The flowchart of the proposed system is depicted in Fig.1.The details of the neural-based cryptology system are described in the following subsections.

3.1 Generating Neural Based Pseudo-Random Numbers

The aim is to improve the randomness of the random numbers generated by any algorithm using an NN. In order to improve pseudo-random numbers via a neural network, random numbers are generated by a modified subtract with borrow algorithm in MATLAB. The random numbers generated by the modified subtract with borrow algorithm are tested for randomness by NIST. Then, these random numbers are used as input values, initial weight, bias values and the neuron number of hidden layers. The network's output values are evaluated without training. The output values of the NN are neural network-based pseudo-random numbers. Therefore, the algorithm can be called a neural-based pseudo-random numbers generator (PRNG). The random numbers generated by the NN-based pseudo-random numbers generator are also tested for randomness by NIST. The results are shown in Chapter 4. The proposed neural-based PRNG stages are simulated in Fig. 2. These improved random numbers will be used to construct the cryptosystem.

3.2 Non-Linear Encryption

Plaintext is divided into blocks and the American Standard Code for Information Interchange (ASCII) values of each block is calculated. These numbers are converted to 7-bit binary forms. Then non-linear encryption is applied to the binary form of plaintext. Each block's local digit number is mixed with itself as shown in Fig. 3. Then, all of the digits are mixed generally as shown in Fig. 4, for a random number of iterations, like shuffling playing cards. This shuffling is carried out using the neural-based random number generator.

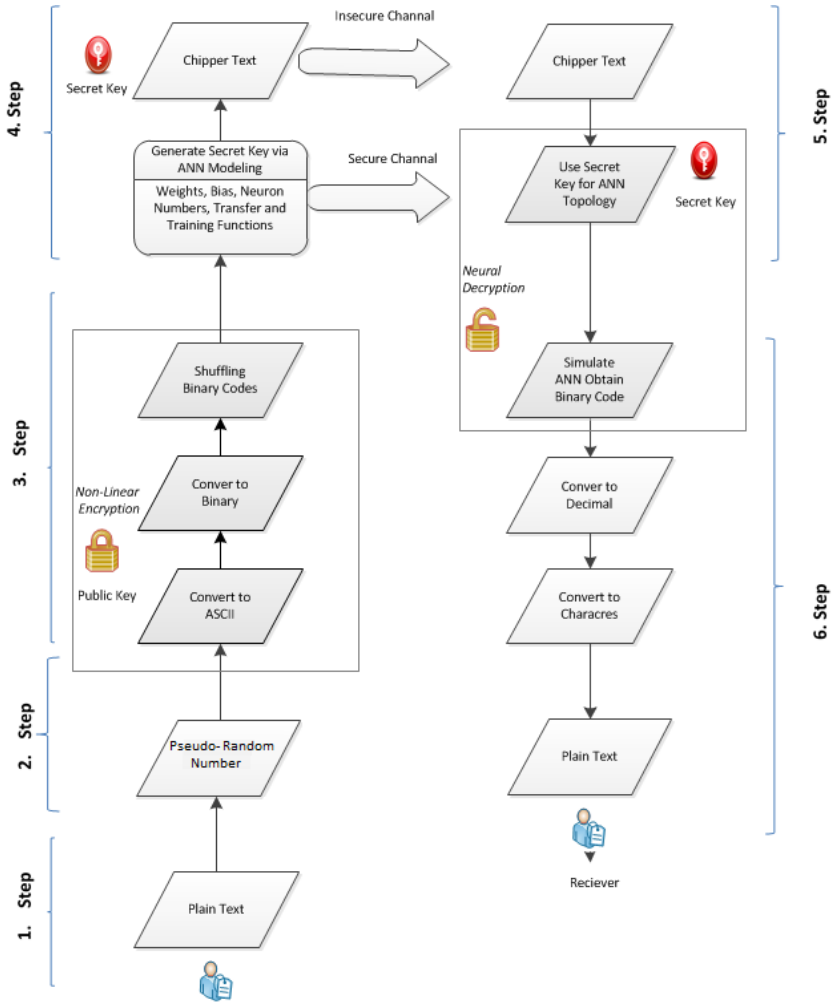

Fig. 1 *Flowchart of proposed neural cryptosystem.*

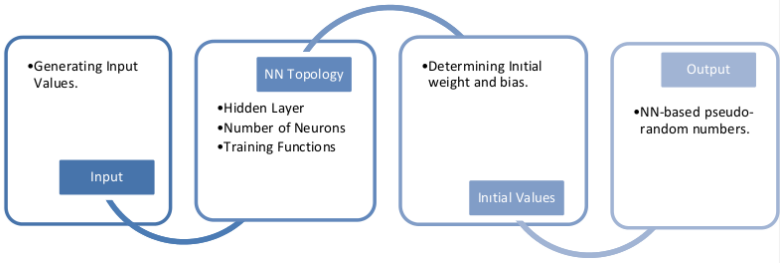

Fig. 2 *Neural network based pseudo-random number generator.*

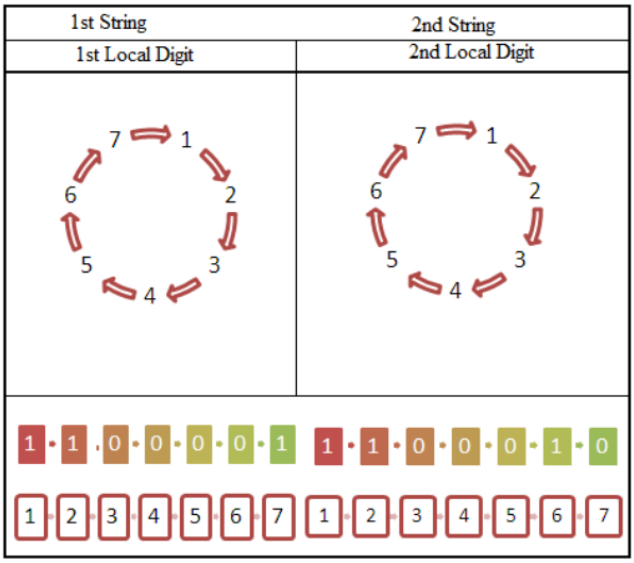

Fig. 3 *Local mixing of each strings' binary codes as random times.*

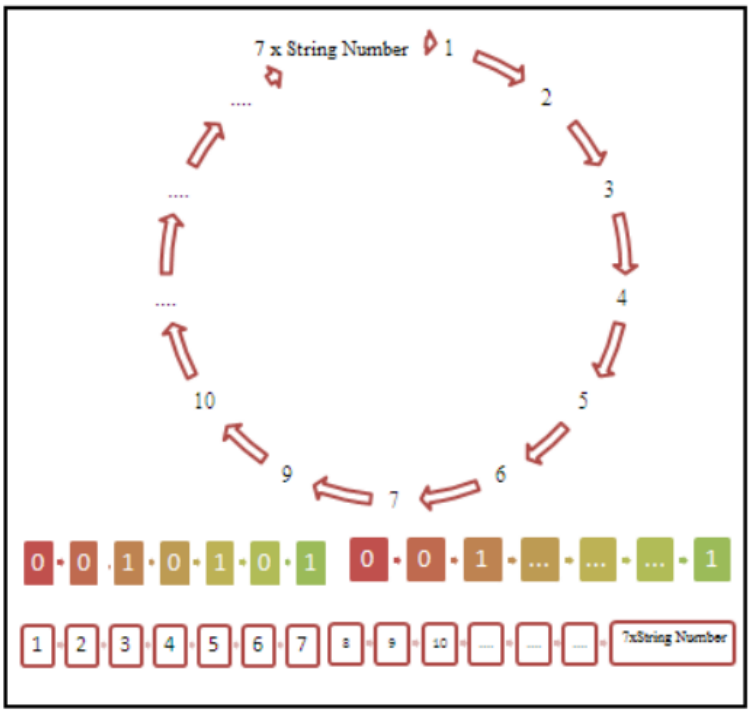

Fig. 4 *Global mixing of all strings binary codes together as random times.*

3.3 Designing Neural Network Topology

Cryptography is the practice and study of hiding information through techniques based on randomness. So, in neural cryptology, the ANN has to be a form of random topology. In this study, the structure of networks changes randomly. It means that the layer size and neuron numbers of each layer are generated by the neural-based pseudo-random number generator for each network structure. The training and transfer functions of the network are also selected randomly. A four-layer ANN with random topology is depicted in Fig. 5. Its input value is chipper text that is encrypted by non-linear encryption described in chapter 3.2. The neuron numbers of the layers are generated by the NN-based pseudo-random number generator. The transfer functions and training algorithms are also selected according to the NN-based pseudo-random number generator.

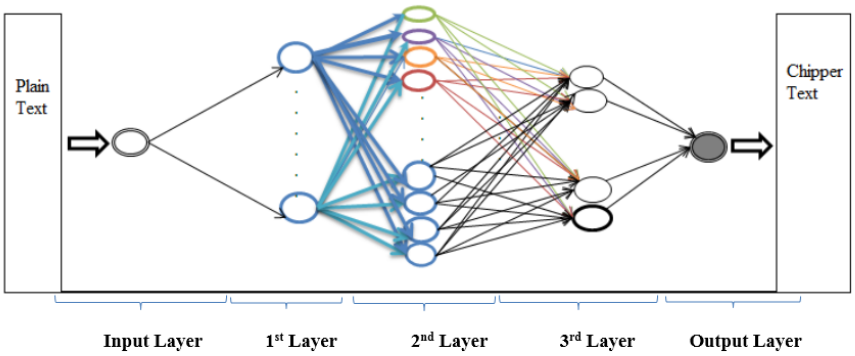

Fig. 5 *Designed random topological neural network.*

3.4 Sending NN Topology and Chipper Text

The steps described above are carried out in order to defend encrypted text from cryptanalysts and provide secure communication. The input weights, hidden layer weights, neuron numbers for each layer, information about transfer function and information about training functions are sent in a format that can only be understood by the receiver, in a secure channel. Thus, neural network topology and chipper text are sent to the receiver.

3.5 Simulating Neural Network and Decrypting

The receiver designs a new neural network which is almost a replica of the sender's. By simulating chipper text values using sent weight and bias values with the new neural network, the receiver achieves the ASCII values of plain text's binary codes. Then, the binary codes are converted to the ASCII values. The ASCII values are then converted to strings. Finally, the receiver will be able to create plaintext after combining these strings side by side.

3.6 Prepared Interfaces of Neural Cryptosystem

Using the proposed algorithm, the neural network-based crypto-tool is developed in MATLAB graphical user interface (GUI). Fig. 6 shows the prepared interfaces of the advanced cryptosystem. The user's initial log-in screen is shown in Fig. 6(a). End users are allowed to access restricted services or to access all the properties as system administrators. Then the end user decides the type of encryption: real-time encryption or file encryption, as shown in Fig. 6(b). In Fig. 6(c), the user decides to encrypt or to decrypt for the purposes of file encryption. Text files in any folder of a PC can be browsed and encrypted easily, producing an encrypted file. Encryption time and ANN topology can also be seen, subject to authorization. The encrypted file can be sent as an e-mail attachment using Simple Mail Transfer Protocol (SMTP) across Internet Protocol (IP) networks, as shown in Fig. 6(e). The real-time encryption module is shown Fig. 6(d). It can be used for any chat room or real-time conversation tool, such as Line, WhatsApp, Facebook or Twitter.

4. Results and Conclusion

Neural-based pseudo-random numbers and traditional pseudo-random numbers were explained in Chapter 3 and tested with NIST randomness tests. These tests are believed to be useful in detecting deviations of a binary sequence from randomness [21]. The results of the randomness tests are shown in Tab. I and Tab. II. In Tab. I, the randomness test results and the details of NN-based PRNGs are explained. The proposed NN-based PRNG successfully passed the test, as described in Tab. I. In Tab. II, results for the randomness of numbers generated by a modified subtraction with borrow random number generator are shown. Frequency test, runs test, longest run of ones in a block test and cumulative sum test were all failed. The other tests were passed. Comments on the passed test are described in Tab. II and the failure results are explained below. The frequency frequency test is the first randomness testing step. It defines whether zero and one bits appear in the tested sequence with approximately the same probability. Clearly, classic RNGs could not pass the basic randomness test. Indeed, there is no need to analyze other tests when the frequency test is failed [21]. Furthermore, 0.000233 p value ($>$ 0.01) describes the correlation between zeros and ones in the sequence. However, pseudo-random number sequences must not be in correlation with each other. The runs test defines whether or not transitions between zeros and ones in the sequence appear as frequently as expected in a random sequence. 0.000212 p value ($>$ 0.01) indicates that runs in a random sequence generated using a modified subtract with borrow random number generator are too slow or too fast, suggesting a non-random walk. The longest run of ones in a block test determines whether or not the length of the longest run of ones in a block is consistent with the length expected from a random sequence. 0.000001 p value ($>$ 0.01) indicates that the length of the longest run of ones in a random sequence generated using a modified subtract with borrow random number generator is not consistent according to the x^2 test. This indicates a non-random walk. The cumulative sum test determines whether the cumulative sum of the partial sequences, consisting of the tested sequence, is too large or too small relative to the expected behaviour of the cumulative sum for

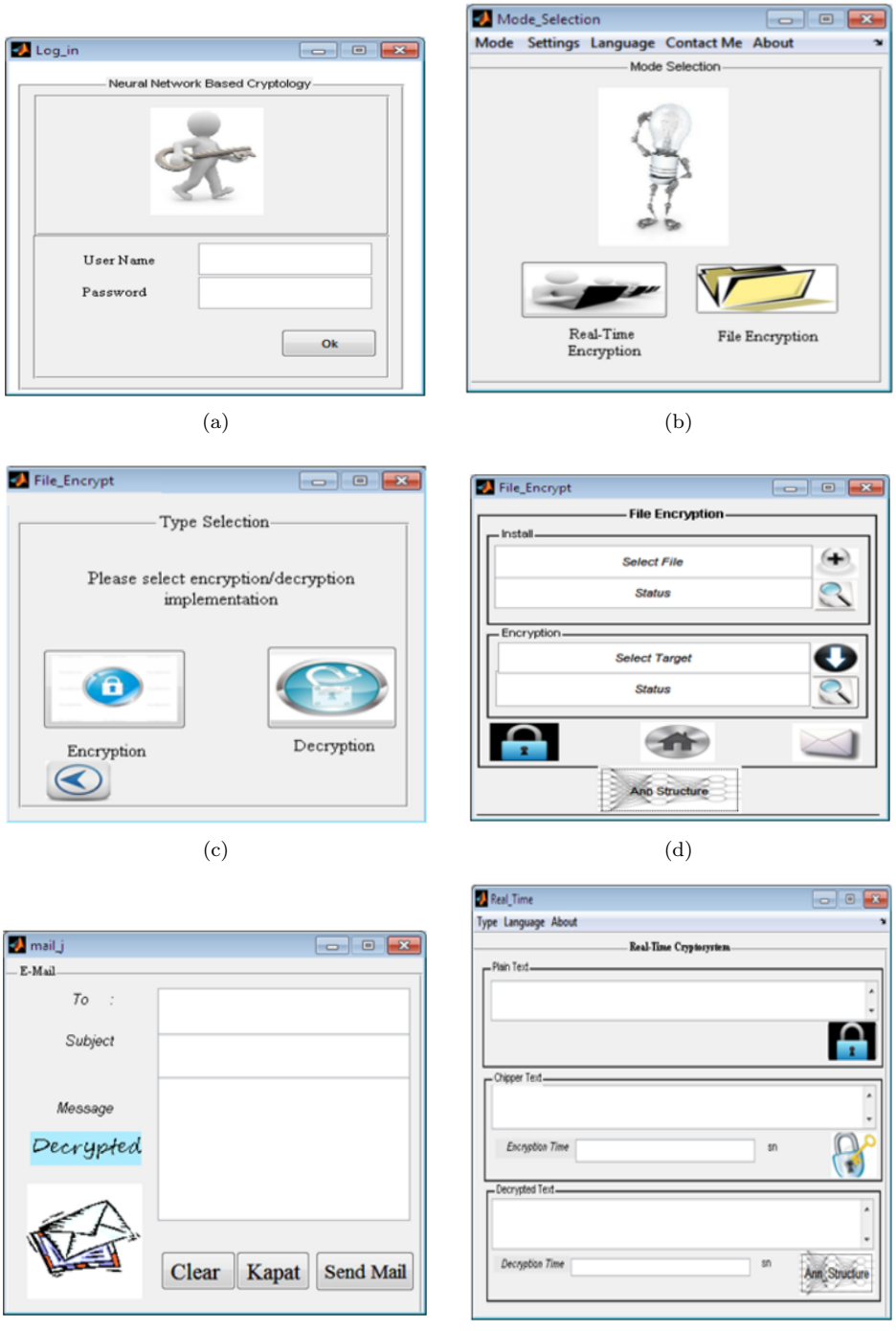

Fig. 6 *Prepared interfaces of the neural cryptosystem.*

random sequences. A 0.000368 p value (> 0.01) suggests that the cumulative sum in a random sequence generated by a modified subtract with borrow random number generator is too large or too small, indicating a non-random walk. Tabs. I and Tab. II show that the neural-based pseudo-random generator is much more successful than the modified subtract with borrow algorithm. It can, therefore, be said that the neural network can be used to improve randomness [31].

Tests	**p Value**	**Result**	**Comments**
Frequency Test	0.14986	Success	0 and 1 bits appear in the sequence with approximately the same probability.
Block Frequency Test	0.911733	Success	0 and 1 bits appear in the blocks of sequence with approximately the same probability.
Runs Test	0.85160	Success	Transitions between 0 and 1 bits in the sequence appear as often as expected from a random sequence.
Longest Run of Ones in a Block Test	0.093350	Success	The length of the longest run of 1 bits in a block is consistent with the length expected from a random sequence.
Cumulative Sum Test	0.911733	Success	The cumulative sum of the partial sequences consist of the tested sequence is not too large or too small relative to the expected behavior of that cumulative sum for random sequences.
Discrete Fourier Test	0.646355	Success	The number of peaks exceeding the 95% threshold is not significantly different than 5%.
Rank Test	0.741908	Success	The deviation of the rank distribution of the sequence can be ignored.

Tab. I *Randomness testing results of number generated by ANN based pseudo-random number generator.*

Some experimental results relating to the cryptosystem are shown in Tab. III. Each iteration of the cryptosystem uses a different neural network structure, which is generated randomly. This means that each time a special model of a key is generated. The text is encrypted using only its own key. Some neural networks with different neuron numbers were also trained; encryption and decryption times and training performance are shown in the Tab. III.

Training with scaled conjugate gradient back propagation, which updates weight and bias values according to the scaled conjugate gradient method, is much faster than other approaches.

Tests	p Value	Result	Comments
Frequency Test	0.000233	Failure	
Blok Frequency Test	0.234600	Success	0 and 1 bits appear in the blocks of sequence with approximately the same probability.
Runs Test	0.000212	Failure	
Longest Run of Ones in a Block Test	0.000001	Failure	
Cumulative Sum Test	0.000368	Failure	
Discrete Fourier Test	0.408863	Success	The number of peaks exceeding the 95% threshold is not significantly different than 5%.
Rank Test	0.741908	Success	The deviation of the rank distribution of the sequence can be ignored.

Tab. II *Randomness testing results of numbers generated by modified subtract with Barrow random number generator.*

Neuron Size	Encryption Time (sec)	Decryption Time (sec)	Performance (MSE)
4-3-18-97-1	0.3105	0.1215	2,14E-01
4-132-91-86-1	11.816	0.1111	3,60E-32
4-14-7-121-1	10.510	0.1618	1,36E-04
4-196-13-4-1	13.147	0.1978	2,14E-01
4-23-48-51-1	6.114	0.1874	3,15E-12

Tab. III *Training informations for some neural network structures.*

5. Discussion

By advancing information technology, it is possible to decrypt in a few minutes a cryptosystem which would previously have been expected to take centuries to decrypt. Therefore, it can be said that classical cryptography applications are no longer sufficient. In order to advance cryptographic security, other branches of science must support it. Quantum mechanics, artificial neural networks and chaotic systems are some of these. Many neural cryptosystems are created using chaotic generator [26, 7], a pseudo-random generator [17, 14, 19], over-fitting [14] or S-box [18]. The proposed system consists of two parts, which are the neural-based pseudo-random number generator and the neural-based cryptosystem. Firstly, the neural-based pseudo-random number generator shows that neural networks improve any PRNG randomness using the ability of the neural network. The plaintext is encrypted by non-linear encryption techniques which are supported by a neural-based PRNG. Data encrypted randomly is decrypted in the neural network

using its decision-making ability. Secondly, the neural network, which is also prepared randomly, is used to create a cryptosystem. Neural-based cryptology is also strengthened by the NN-based pseudo-random number generator. Different neural network structures are constructed randomly for each occasion. The hidden layer size, number of neurons in the hidden layers, transfer function and training function randomly change in every encryption application. If a cryptanalyst wants to generate a neural network which is the same as one already used, an ANN model that has the same weights, biases, training function and transfer functions must be determined from the text, which is sent to the receiver. As a result, this algorithm is complicated and random enough for any cryptanalyst. Consequently, the proposed system indicates two scenarios. In the first scenario, the created encryption method is supported by an NN and is completely random. It can be evaluated as a one-way hash function. The encrypted test by the random encryption method is decrypted with a neural network-based model. This model is a cryptosystem which is created independently from the encryption algorithm. The second scenario is that an NN-based secret key is created from completely custom text. Each time the parameters of the NN are generated completely afresh. The NN-based crypto-tool is a useful tool for anyone. It will be developed for embedding real-time conversation tools such as Line, Whats Up, Facebook, Twitter or any chat application. When wireless networks are used in the home or office, the information can be stolen by a sniffer attack. The NN-based crypto-tool is useful for protecting sensitive information in the home or office against theft of this kind.

Acknowledgement

The study was supported by 8702 numbered Scientific Research Project in Mustafa Kemal University.

References

[1] Abdi H.: A neural Network Primer. Journal of Biological Systems, 1994.

[2] Annonimous: Matlab toolbox release Notes for mathematics. MathWorks inc, 2013.

[3] Arvandi M., Wu S., Sadeghian A., Melek W. W., Woungang I.: Symmetric Cipher Design Using Recurrent Neural Networks. International Joint Conference on Neural Networks, pp. 2039–2046, 2006.

[4] Chi-Kwong C., Cheng L. M.: The convergence properties of a clipped Hopfield network and its application in the design of key stream generator. IEEE Trans. Neural Networks, 12, pp. 340–348, 2001.

[5] Cin İ.: Sifre Sorgulamada Yapay sinir Aglarının Kullanılması. Msc thesis, Osman Gazi Universitesi, 1996.

[6] Dalkıran İ.: Yapay Zeka Teknigi Kullanan Bilgisayar Tabanlı Yuksek Hassasiyetli Sıcaklık Olcme Birimi Tasarımı. Msc thesis, Erciyes Universitesi, 2003.

[7] Dalkıran İ., Danısman K.: Artificial neural network based chaotic generator for cryptology. Turk J Elec Eng & Comp Sci, 18(2), pp. 225–240, 2010.

[8] Desai V., Patil R., Rao D.: Using Layer Recurrent Neural Network to Generate Pseudo Random Number Sequences. International Journal of Computer Science Issues, 9(2), pp. 324–334, 2012.

[9] Fidan M., Nezih Gerek ́O.: Randomness analysis of Antimycielski number generator. IEEE 16th Signal Processing, Communication and Applications Conference, pp. 1–4, 2008.

[10] Godhavari T., Alainelu N. R., Soundararajan R.: Cryptography Using Neural Network. IEEE Indicon 2005 Conference,, (I), pp. 11–13, 2005.

[11] Guo D., Cheng L.-M., Cheng L. L.: A new symmetric probabilistic encryption scheme based on chaotic attractors of neural networks. Appl. Intell., 10(1), pp. 71–84, 1999.

[12] Hagan M. T., Beale M. H., Demuth H. B.: Neural Network Toolbox User's Guide. The MathWorks, Inc, 2009.

[13] Hughes J. M.: Pseudo-random Number Generation Using Binary Recurrent Neural Networks. Project Report, Kalamazoo College, 2007.

[14] Karras D. A., Zorkadis V.: On neural network techniques in the secure management of communication systems through improving and quality assessing pseudorandom stream generators. Neural networks : the official journal of the International Neural Network Society, 16(5-6), pp. 899–905, 2003.

[15] Lauria F. E.: On Neurocrytology. Proceedings of the Third Italian Workshop on Parallel Architectures and Neural Networks, pp. 337–343, 1990.

[16] Marsaglia G., Zaman A.: A New Class of Random Number Generators. Ann. Applied Prob, pp. 462–480, 1991.

[17] Kannan Munukur R., Gnanam V.: Neural network based decryption for random encryption algorithms. 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, pp. 603–605, August 2009.

[18] Awal Noughabi M. N., Sadeghiyan B.: Design of S-boxes based on neural networks. International Conference on Electronics and Information Engineering, 2:V2–172–V2–178, August 2010.

[19] Othman K. M. Z., Jammas M. H. A. L.: Implementation of Neural-Cryptographic System Using FPGA. Journal of Engineering Science and Technology, 6(4), pp. 411–428, 2011.

[20] Pointcheval D.: Neural Networks and their Cryptographic Applications. Pascale Charpin Ed. India, 1994.

[21] Rukhin A., Soto J., Nechvatal J., Smid M., Barker E., S. Leigh, Levenson M., Vangel M., Banks D., Heckert A., Dray J., Vo S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Special Publication 800-22 National Institute Standart Technology, 2010.

[22] Ruttor A.: Neural Synchronization and Cryptography. PhD thesis, Bayerischen Julius-Maximilians-Universitat at Wurzburg, 2006.

[23] Sağıroğlu S., Ozkaya N.: Neural Solutions for Information Security. Journal of Polytechnic, 10(1), pp. 21–25, 2007.

[24] Schneider B.: Applied Cryptography. Protocols, Algorithms, and Source codes in C, 1996.

[25] Sivagurunathan G., Rajendran V., Purusothaman T.: Classification of Substitution Ciphers using Neural Networks. International Journal of Computer Science and Network Security, 10(3), pp. 274–279, 2010.

[26] Su S., Lin A., Yen J.: Design and realization of a new chaotic neural encryption/decryption network. IEEE Asia-Pasific Conf.Cir and Syst., pp. 335–338, 2000.

[27] Sulak F.: Statistical analysis of block ciphers and hash functions. Msc thesis, Middle East Technical University, 2011.

[28] Tanrıverdi H.: Yapay Sinir Aglarının Kriptolojide Kullanılması. Msc thesis, Middle East Technical University, 1993.

[29] Walker, J.: A pseudorandom number sequence test program, 1998, available in http://www.fourmilab.ch/random/, 04.14.14.

[30] Yayık A., Kutlu Y.: Metin iç in Yapay Sinir Ağı Tabanlı Hash Fonksiyonu. International Conferance on Cryptology and Information Security, 2013.

[31] Yayık A., Kutlu Y.: Sozde Rastsal Sayı Uretecinin Yapay Sinir Agları ile Guclendirilmesi. Sinyal İsleme ve İletisim Uygulamaları (SIU) Kurultayı (SIU2013), 2013.

[32] Yayık A., Kutlu Y.: Yapay Sinir Ağı Tabanlı Kriptoloji Uygulamaları. Msc thesis, Mustafa Kemal University, 2013.

[33] Pol Yee L., De Silva L. C.: Application of MultiLayer Perceptron Network as a one-way hash function. Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No.02CH37290), pp. 1459–1462, 2002.

[34] Yılmaz R.: Kriptolojik Uygulamalarda Bazı İstatistik Testler. PhD thesis, Konya Selcuk University, 2010.

[35] Zeng K., Yang C.-H., Wei D.-Y., Rao T. R. N.: Pseudorandom bit generators in stream-cipher cryptography. IEEE Computer, 24(2), pp. 8–17, 1991.