Cryptography using Artificial Intelligence

Jonathan Blackledge
School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
Durban, South Africa
dvcresearch@ukzn.ac.za

Sergei Bezobrazov
Intelligent Information Technologies Department
Brest State Technical University
Brest, Belarus
bescase@gmail.com

Paul Tobin
School of Electrical and Electronic Engineering
Dublin Institute of Technology
Dublin, Ireland
paul.tobin@dit.ie

***Abstract*—This paper presents and discusses a method of generating encryption algorithms using neural networks and evolutionary computing. Based on the application of natural noise sources obtained from data that can include atmospheric noise (generated by radio emissions due to lightening, for example), radioactive decay, electronic noise and so on, we 'teach' a system to approximate the input noise with the aim of generating an output nonlinear function. This output is then treated as an iterator which is subjected to a range of tests to check for potential cryptographic strength in terms of metric such as a (relatively) large positive Lyapunov exponent, high information entropy, a high cycle length and key diffusion characteristics, for example. This approach provides the potential for generating an unlimited number of unique Pseudo Random Number Generator (PRNG) that can be used on a 1-to-1 basis. Typical applications include the encryption of data before it is uploaded onto the Cloud by a user that is provided with a personalized encryption algorithm rather than just a personal key using a 'known algorithm' that may be subject to a 'known algorithm attack' and/or is 'open' to the very authorities who are promoting its use.**

Keywords—coding and encryption; artificial intelligence; multiple algorithms; personalised encryption engines; artificial neural networks; evolutionary computing

I. INTRODUCTION

Protective software (i.e. antiviruses, intrusion detection systems etc.) is as an integral part of an operating system, e.g. file managers, browsers, text processors etc. However, even with use of the latest protective software products there is no guarantee that the computer and/or computing environment is 'safe'. Such infamous malware for cyber espionage as Stuxnet, Flamer, Duqu, Gauss, Regin and others gives proof that up-to-date protective solutions are powerless in the face of sophisticated cyber-weapons. The listed malware may not only still undetectable for years but also perform cyber espionage by controlling infected computers. The existing situation opens a possible arena in regard to personalized encryption algorithms and data protection systems. Personal encryption algorithms allow users to protect data even if protective software fails to detect a threat.

Previous work in this area [1] presented an approach to using evolutionary computing for generating personal ciphers using input data streams consisting of natural noise. Working with a system called *Eureqa*, designed by Nutonian Inc. [2], the system was 'seeded' with natural noise sources obtained from data based atmospheric noise generated by radio emissions due to lightening. The purpose of this is to 'force' the system to output a result (a nonlinear function) that is an approximation to the input noise. This output is then treated as an iterated function whose output (floating-point) data stream is subjected to a range of tests to check for potential cryptographic strength in terms of a positive Lyapunov exponent, high entropy (uniform probability density function), a uniform power spectrum, high cycle length and key diffusion characteristics. This approach provides the potential for generating an unlimited number of unique PRNG that can be used on a 'one-to-one' basis.

In this paper we present the implementation of an artificial neural network for generating personal ciphers. We show that neural networks are powerful tools in cryptography and provide a route to producing a strong personal cipher.

The paper is organized as follows. A state-of-the art is given in Section 2. Section 3 gives a background of how evolutionary computing can be used to generate a strong cypher. The implementation of an Artificial Neural Network, in particular, a Radial Basis Function Neural Network is given in Section 4. The final section concludes this paper.

II. STATE-OF-THE-ART

Neural cryptography is a direction in cryptography that applies Artificial Neural Networks (ANN) for encryption and cryptanalysis. Recent works in the use of neural networks in cryptography can be categorized into three major parts:

978-1-4799-1959-8/15/$31.00

a) Implementation of synchronized neural networks.

b) Cryptography based on use of chaotic neural networks.

c) Cryptography based on multi-layer neural networks.

A. Synchronization neural networks

The first approach of applying an ANN in cryptography is through the creation of two topologically identical neural networks. Neural networks can synchronize by learning from each other and full synchronization can be achieved in a finite number of steps. The main idea is as follows: the user A wants to communicate with user B, but they cannot exchange a secret key through a secure channel; so they create two topologically identical neural networks but with different random weights and evaluate them with the same inputs until the weights of both ANN's match [3] – [5]. The main algorithm of such a system consists of the following steps:

1) Initialize random weight values.

2) Execute the following steps until full synchronization is achieved.

a) Generate a random input vector.

b) Compute the values of the hidden and output neurons.

c) Compare the values of both neural networks. If the outputs are different then go to *2(a)* else update the weights of the networks according to learning rules.

3) After full synchronization is achievd, A and B can use their weights as keys.

The process of synchronization of the networks can be considered as the key generation process in cryptography. The common identical weights of synchronized networks for two partners can be used as a key for encryption.

B. Chaos-based neural cryptography

Chaos-based cryptography combines two research fields, i.e., chaos (nonlinear dynamic systems) and cryptography and becomes more practical in the secure transmission of large multi-media files over public data communication networks. In this field, the implementation of neural networks composed of chaotic neurons opens a wide area for developing strong cryptosystems. For example, W. Yu proposed an encryption technique based on the chaotic Hopfield neural network with time varying delay [6]. The proposed chaotic neural network is used for generating binary sequences for masking the plaintext. The binary value of the binary sequence chooses the logistic map randomly, used for generated the binary sequences. The plaintext is masked by switching the chaotic neural network maps and the permutation of generated binary sequences [7].

In the area of image cryptography, a triple key chaotic neural network was used by S. Suryawanshi et al. [8]. A Triple Key means that three parameters are used as control parameters producing a hexadecimal sequence. The triple parameters are used to perform the various operations on an image so as to scramble the data. Experimental results show that the algorithm performs successfully and can be applied on different colour image size.

T. Fadil et al. presented an algorithm coupled with a chaotic neural network to encrypt MPEG-2 video codecs [9]. The algorithm supports quality and bit rate control that is required by many video transmission applications. The logistics map is used with a neural network to produce a combination of chaotic neural networks based on a binary sequence generated from the logistic map, the biases and weights of neurons are modified and taken to be the secret key. It has been shown from analysis results that the proposed algorithm has high security with low cost, and also supports quality and bit rate control.

C. Multilayer neural networks in the cryptography

K. Noaman et al. presented an encryption system based on General Regression Neural Network [10]. Here a neural network is used to construct an efficient encryption system by using a permanently changing key. The encryption function consists of the three steps: 1) the creation of the keys; 2) the input message is broken into blocks equal to the number of keys and processed, one block at a time, as input to the neural network; 3) the neural network is composed of three layers (each layer consisting of a number of neurons, depending on the process type) and is used to encrypt each block of data producing the encrypted message. The encryption process divides the input message into 3-bit data sets, and produces 8-bit data after the encryption process. The simulation results provide a relatively better performance than traditional encryption methods.

Finally, E. Volna et al. studied the use of back-propagation neural networks in cryptography [11]. The model converts the input message in to ASCII code and then gets the sequence of bits for each code which is divided into 6 bit blocks which are used as input for the encryption process. The cipher key is the neural network structure containing an input layer, hidden layer, output layer, and updated weights. The proposed system has been tested for various numbers of plain text and the simulation results have shown that the system is secure.

In this paper we present a different approach from those that have been briefly described and referenced above and is based on evolutionary computing. Here we show how an ANN can be used to generate a cipher by simulating natural noise once it has been trained to do so. To use an ANN in this way, the cryptographer requires knowledge of the ANN algorithm and the weights that have been generated through the training process (i.e. the input of the noise sources used to generate the weights).

III. EVOLUTIONARY COMPUTATION IN CRYPTOGRAPHY

Evolutionary Computing is associated with the field of Computational Intelligence, and, like Artificial Intelligence, involves the process of continuous and combinatorial optimizations. It is inspired from biological mechanisms of evolution and uses iterative processes in a parallel manner to achieve the goal.

Evolutionary algorithms are a part of Evolutionary Computation and simulate different biological mechanisms such as reproduction, mutation, recombination, natural selection and so on. For designing Pseudo Random Number Generators (PRNGs), an evolutionary algorithms based approach can be used in which a population-based, stochastic search engine is required that mimics natural selection. Due to their ability to find excellent solutions for conventionally difficult and dynamic problems within acceptable time, evolutionary algorithms have attracted interest from many areas of science and engineering.

The application of evolutionary algorithms to cryptology as presented in this paper is, to the best of the author's knowledge, an original concept. The proposed technology is the first of its kind to break away from the Kerckhoff principle, a principle which states that *A cryptosystem system should be secure even if everything about the system, except the key, is public knowledge*. In this paper we use the principles of evolutionary computing to automatically generate an algorithm that is unique to the user of the algorithm and the keys that initiate it. The technology generates ciphers of a high cryptographic strength using input data streams consisting of natural noise source such as atmospheric noise, cosmic noise, radioactive decay and so on. Fig. 1 demonstrates the basic steps of the proposed approach in schematic form.

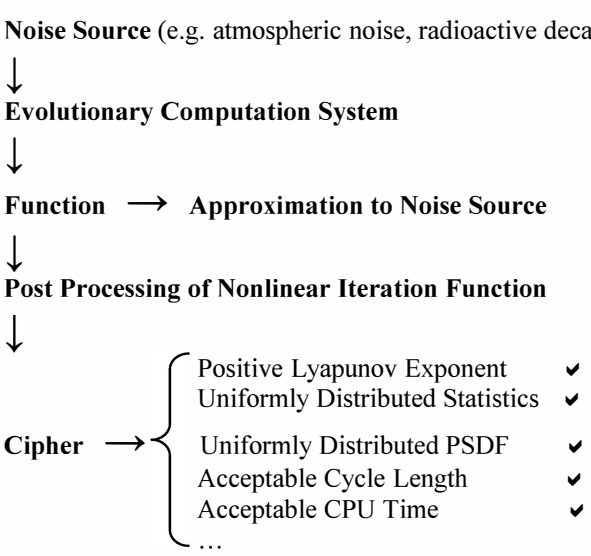

Fig. 1. Schematic of the processes for evolving a cipher.

Evolutionary computing systems, which use mechanisms such as reproduction, mutation, recombination, selection etc., try to find the best equation for an optimal approximation to a given data stream. We use the *Eureqa* system developed by Nutonian Inc. for detecting equations and hidden mathematical relationships in the data. This system uses evolutionary computations (in particular, symbolic regression) and iteratively develops a nonlinear function to described stochastic input signals usually associated with experimental data output from a chaotic system, for example. We input a noise source to the system with the purpose of 'forcing' the system to output a result (a nonlinear function) that is an approximation to the input noise. If genuine random (delta uncorrelated) noise is input into the system, then, from a theoretical point of view, no nonlinear function should be found on an evolutionary basis. Thus, inputting natural noise is a way of 'cheating' the system to 'force' it to provide a result that may be suitable (on an iterative basis) as a PRNG. The output is then treated as an iterated function which is subjected to a range of tests to check for potential cryptographic strength.

For this study, we use the data available from RANDOM.ORG which, to date, has generated 1.92 trillion random bits for the Internet community [12]. Fig. 2 shows an example screen shot of the *Eureqa* system used to generate the following iteration function (1) for cipher generation

$$c_{i+1} = 118.6 + 49.9\cos(0.4c_i^2 + \tan(0.3c_i) + \sin(2.3c_i + \cos(0.9 + 0.4c_i^2))) - 29.2\cos(4.4c_i^2). \quad (1)$$

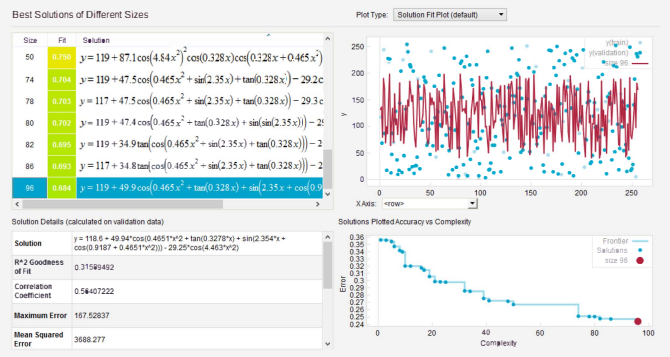

Fig. 2. Screen shot of *Eureqa* used for evolving nonlinear functions suitable for cipher generation.

The non-linear iteration function given by equation (1) (Fig. 3 demonstrates some cryptographic strength characteristics, e.g. uniform distribution, power spectrum and autocorrelation function) is the result of Eureqa undertaking over 100 iterations (using 255 noise samples randomly selected from the data bases available at RANDOM.ORG) to evolve the result, taking approximately 27 hours using an Intel Core i3 1.7x2 GHz CPU to do so.

While equation (1) provides a valuable iterator (subject to normalization so that $c(i) \in (0,1]\ \forall i$ and post-processing based on the tests described in Fig. 1), it is not provable that this equation is structurally stable, i.e. that a cryptographically strong cipher is guaranteed for any floating point value of c_0 between *0* and *1*, say, irrespective of the precision of c_0. This is important because c_0 (which seeds and thereby initiates the cipher stream) could, for example, be generated by a Hash function from a low bit private key and possibly fail at some point in the future for lack of structural stability. However, this is in keeping with many other PRNGs especially those generated by nonlinear iterations for which the structural stability problem remains an issue.

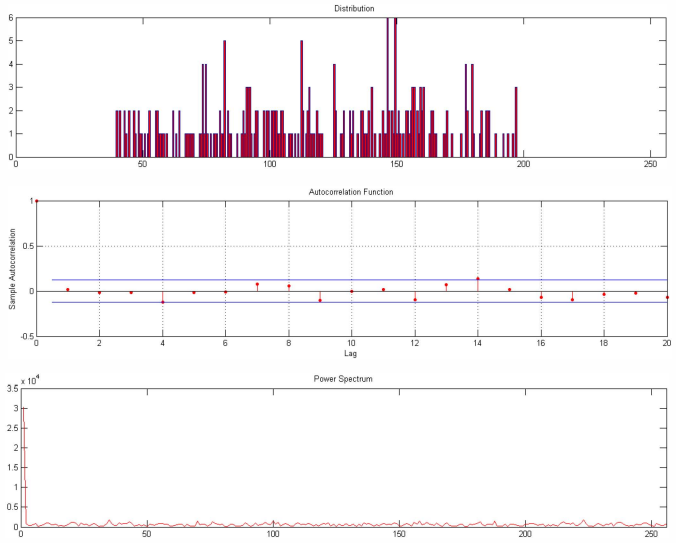

Fig. 3. Cryptographic strength of the iterated function.

In this way, the proposed approach pays no attention to the algorithmic complexity of the iterator which is one of the main problems in the application of chaos to cryptography. Neither does it consider the structural stability of the iterator or its algorithmic complexity. However, it does provide a practical solution to the problem of developing a large database of PRNG for the application of personalizing encryption algorithms for strictly 'one-to-one' communications or 'one-to-Cloud' (encrypted) data storage. By using evolutionary computing systems such as Eureqa seeded with noise, it is possible to generate a nonlinear function with appropriate control parameters. Using this function in an iterative form with an additional transformation say, and, a partition function, a PRNG suitable for encrypting data can be constructed. The combined effect is that of a hard-core predicate.

IV. Neural Cryptography

Artificial Neural Networks (ANN) as well as Evolutionary Computation are a subfield of Artificial Intelligence and involve various architectures of neural networks and rules of training. ANNs were inspired by biological neural networks (the central nervous systems, in particular, the brain) and can be implemented in various complex control engineering problems such as function approximation pattern recognition, data mining, prediction, machine learning and so on. Generally, ANNs consist of several layers of interconnected nodes or "neurons" that form a network that mimics a biological neural network. Through iterative training process an ANN computes a set of optimal weights between neurons that determines the flow of information (the amplitude of a signal at a given node) through a network that simulates a simple output subject to a complex input.

Artificial Neural Networks have, relatively recently, found a successful implementation in cryptography (in particular for cipher generation) and form a separate branch of cryptography. At least one algorithm based on the implementation of an ANN in cryptography is known using two neural networks that are trained on their mutual output bits and are able to synchronize. The common identical weights of the two neural networks can be used as a key for encryption [4].

In this paper we propose different technique that is based on ability of an ANN to simulate a high entropy input with the aim of transforming the result into a low entropy output. However, this process can be reversed to generate a high entropy output from a low entropy input. In this sense, an ANN can be used to generate a cipher by simulating natural noise once it has been trained to do so. To use an ANN in this way, the cryptographer requires knowledge of the ANN algorithm and the weights that have been generated through the iterative training process (i.e. the input of the noise sources used to generate the weights). Fig. 4 shows an example of the input noise provide by RANDOM.ORG and the ANN simulation.

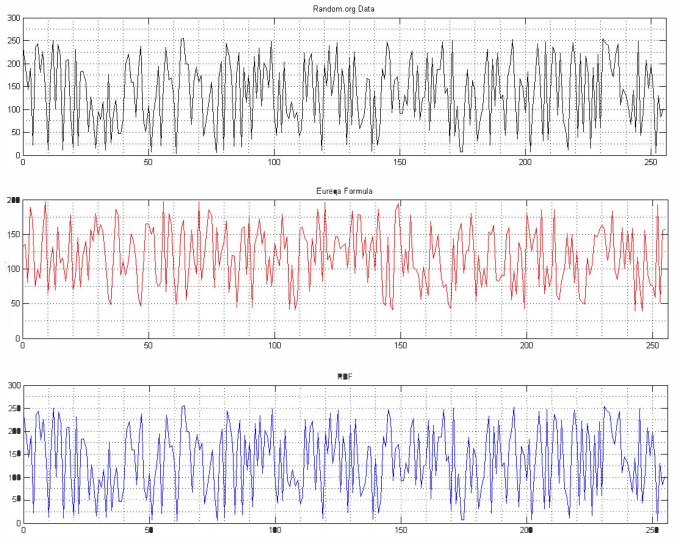

Fig. 4. Example of training an ANN to simulate a genuine random number stream: Original noise (above), 'Eureqa equation' (middle) and ANN approximation (below).

This demonstrates that an ANN can produce highly nonlinear behavior. Given this statement, the precise ANN algorithm becomes analogous to a PRNG in conventional cryptography and the weights are equivalent to the key.

In this work we use a classical Radial Basis Function (RBF) network. This neural network uses radial basis functions as activation function and provides optimal performance in the function approximation tasks. The RBF consists of three layers. The first layer consists of linear neurons and just feeds the values to each neuron in the hidden layer. The weights of each neuron in a hidden radial basis layer define the position and width of a redial basis function. An output linear layer forms a weighted sum of the radial basis function. With enough neurons in the hidden layer, a radial basis network can fit any function with any desired accuracy. Fig. 5 represents in schematic form the proposed ANN algorithm for cipher creation.

Initially, the stochastic data stream inputs to an ANN that attempts to fit the proposed data. After this, an ANN can be used for strong cypher generation.

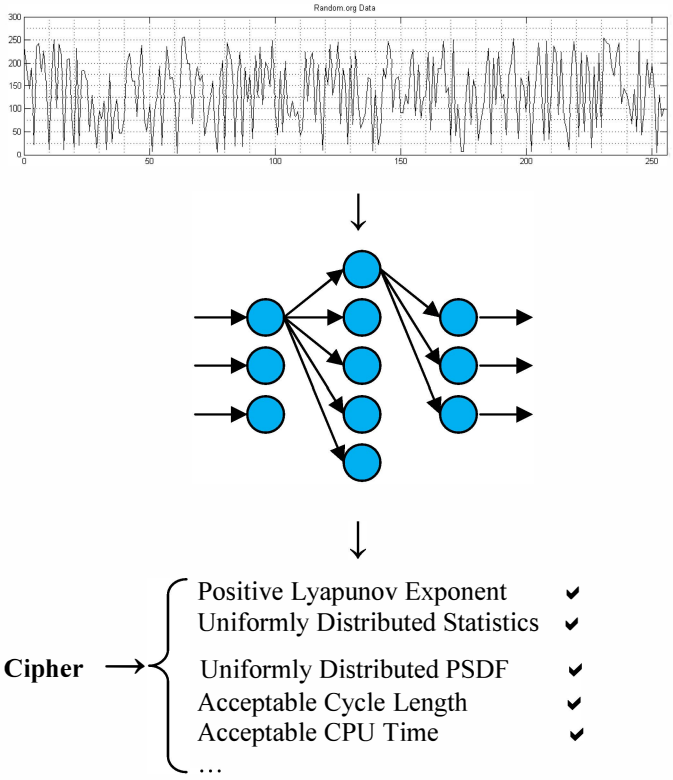

Fig. 5. The ANN algorithm for cipher creation.

The output characteristics of the ANN are represented in Fig. 6.

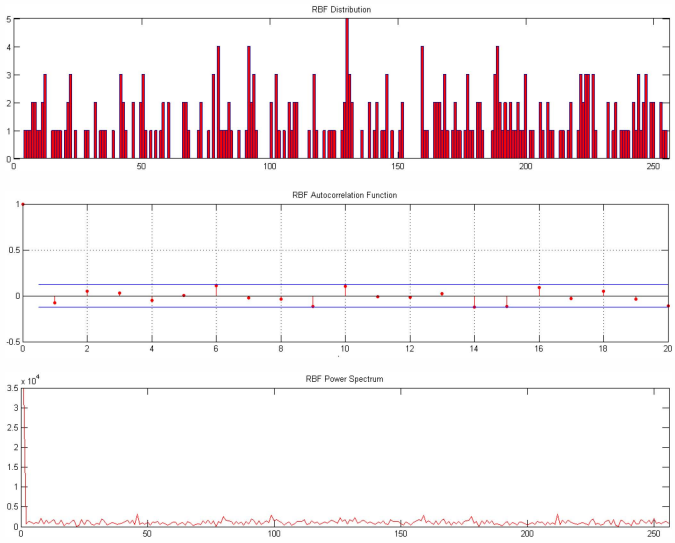

Fig. 6. The characteristic of the output of the ANN.

In comparison with the Evolutionary Commutating approach, described in the Section II, an ANN provides analogous results (uniformly distributed statistics, uniformly distributed power spectrum, positive Lyapunov exponent, information entropy etc.). However, with the same configuration of a computer system, an ANN provides time advantage. For generation of one cipher an ANN takes less than one minute while the *Eureqa* tool takes many hours to find the approximation equation for the same data stream. The second advantage of implementing an ANN for cipher generation lies in its parallel structure and provides high-distributed computational abilities.

V. Conclusions

Practical cryptography is based on passing known statistical tests which is designed to ensure the pseudo-random property of a generator, pseudo-random sequences being taken to be used instead of truly random sequences in most cryptographic applications. This paper introduces a way of designing algorithms for generating pseudo-random (chaotic) sequences using truly random strings to evolve an iterator that is taken to be an approximation to these sequences. This approach pays no attention to the algorithmic complexity of the iterator which is one of the main problems in the application of chaos to cryptography. Neither does it consider the structural stability of the iterator or its algorithmic complexity. However, it does provide a practical solution to the problem of developing a large database of PRNGs for the application of personalizing encryption algorithms for strictly 'one-to-one' communications or 'one-to-Cloud' (encrypted) data storage. By using evolutionary computing systems such as *Eureqa* seeded with noise, it is possible to generate a nonlinear function with appropriate control parameters. However, the one-step unpredictability does not guarantee that the output sequence will be unpredictable when an adversary has access to a sufficiently long sequence. In other words, the vast number of samples can, on a theoretically basis at least lead to predictability. With these provisos, the work reported in this paper demonstrates that evolutionary computing and artificial neural networks provide the potential for generating an unlimited number of ciphers which can be personalized for users to secure their 'Data on the Cloud', for example. Algorithms can be published so that the approach conforms to the Kerckhoff-Shannon principle as in the example provided, i.e. equation (1), in the knowledge that a new set of evolutionary computed algorithms can be developed. Since 2012 over 300 ciphers have been produced in this way, and, in summary, the technique may present a technical solution to the 'democratization of the cipher bureaux'.

References

[1] J. Blackledge, S. Bezobrazov, P. Tobin and F. Zamora, "Cryptography using Evolutionary Computing", IET ISSC13, LYIT Letterkenny, 2013.

[2] Eureqa, "A software tool for detecting equations and hidden mathematical relationships in your data", Cornell Creative Machine Labs, USA, 2013, http://creativemachines.cornell.edu/eureqa.

[3] W. Kinzel and I. Kanter, "Neural Cryptography," TH2002 Supplement, vol. 4, pp. 147-153, 2003.

[4] E. Klein, R. Mislovaty, I. Kantor, A. Ruttor and W. Kinzel, "Synchonization of neural networks by mutual learning and its aoolication to cryptography", Adwances in Neural Information Processing Systems, NIPS, 2004.

[5] R. Jogdand and S. Bisalapur, "Design of an efficient neural key generation", International Journal of Artificial Intelligence and Applications (IJAIA), vol. 2, no. 1, 2011, pp. 60-69.

[6] W. Yu and J. Cao, "Cryptography based on delayed chaotic neural networks", Physics Letters A, Vol. 356, (4) Elsevier, pp. 333-338, 2006.

[7] A. El-Zoghabi, A. Yassin and H. Hussien, "Survey Report on Cryptography based on Neural Network" International Journal of Emerging Technology and Advanced Engineering, (ISSN 2250-2459), Vol. 3, Issue 12, December 2013.

[8] S. Suryawanshi and D. Nawgaje, "A triple-key Chaotic neural network for cryptography in image processing", International Journal of Engineering Sciences & Emerging Technologies, Vol. 2, Issue. 1, pp. 46-50, 2012.

[9] T. Fadil, S. Yaakob, B. Ahmad and A. Yahya, "Encryption of mpeg-2 video signal based on chaotic neural network", Journal of Engineering and Technology, Vol. 3, pp. 35-42, 2012.

[10] K. Noaman and H. Jalab, "Data security based on neural networks", Task Quarterly 9, No. 4, pp. 409-414, 2005.

[11] E. Volna, M. Kotyrba, V. Kocian and M. Janosek, "Cryptography based on neural network", Proceedings 26th European Conference on Modelling and Simulation, 2012.

[12] RANDOM.ORG: True random number Service, 2015, http://www.random.org.