Key Exchange Management by using Neural Network Synchronization

Dr. Buthainah F. AL-Dulaimi* **Amer Abdulmejeed Abdurehman***
Alla W. Abud Al Kader**
College of Education for Women/University of Baghdad
**College of Baghdad for Economic Science

Abstract.

The paper presents a neural synchronization into intensive study in order to address challenges preventing from adopting it as an alternative key exchange algorithm. The results obtained from the implementation of neural synchronization with this proposed system address two challenges: namely the verification of establishing the synchronization between the two neural networks, and the public initiation of the input vector for each party. Solutions are presented and mathematical model is developed and presented, and as this proposed system focuses on stream cipher; a system of LFSRs (linear feedback shift registers) has been used with a balanced memory to generate the key. The initializations of these LFSRs are neural weights after achieving the synchronization.

Keywords: Neural synchronization, LFSRs, Public Key, Stream cipher, Key exchange, cryptographic system.

ادارة تبادل مفاتيح التشفير بأستخدام الشبكة العصبية

د.بثينة فهران عبد* **م. عامر عبد المجيد عبد الرحمن***
الاء وجيه عبد القادر**
* كلية التربية للبنات/ جامعة بغداد
** كلية بغداد للعلوم الاقتصادية

المستخلص:

يقدم هذا البحث تصميم وتنفيذ نظام لتبادل المفتاح إعتماداً على قابلية الشبكة العصبية على التزامن مع شبكة عصبية اخرى ،علاوة على ذلك فإن الشبكات العصبية تستطيع أن تتزامن بواسطةِ بدء عملية تهيئة إبتدائية عشوائية لأوزانها.

أُستخدم التزامن العصبي كجزء أساسي من طبقة إدارة تبادل المفتاح لكي يوفر قناة آمنة عبر وسط عام أقل موثوقية. إن النتائج المستحصلة من تنفيذ التزامن العصبي أشرت تحديين :هما التأكد من تحقيق التزامن بين الشبكتين العصبيتين والتحدي الأخر هوعملية التهيئة الإبتدائية لمتجهات الإدخال لكل طرف بصورة معلنة . وبما أن النظام المقترح ركَز على التشفير الإنسيابي فقد أُستخدم نظام متكون من مسجلات إزاحة خطية إسترجاعية التغذية وأُستخدمت تلك المسجلات مع ذاكرة متوازنة لتوليد المفتاح كي تكون الأوزان العصبية للشبكتين بعد تحقيق التزامن بينهما هي القيم الأولية لتلك المسجلات.

1.Introduction

The goal of any cryptographic system is the exchange of information among the intended users without any leakage of information to others, who may have unauthorized access to it. A common secret key could be created over a public channel accessible to any opponent, as a result to that, many public key cryptography have been presented which are based on number theory and they require large computational power. Also the process involved in generating public key is very complex and time consuming. A user input key has to transmit over the public channel all the ways to the receiver for performing the decryption procedure. So there is a likelihood of attack at the time of key exchange. To overcome these disadvantages and to defeat this insecure secret key generation technique a neural network based on secret key generation technique has been proposed[1,2,3].

There has been an increasing interest in the application of different classes of NNs to problems related to cryptography in the past few years. Recent works have examined the use of NNs in cryptosystems. Typical examples include key management, generation and exchange protocols; visual cryptography; pseudorandom generators; digital watermarking; and steganalysis [4,5,6]. Cryptography is the study of information security and the feasibility of communication over an insecure channel while preserving the secrecy of the information transmitted [7], [8]. Cryptographic techniques (such as stream cipher which will be used in this proposed system) should offer at least the following three security features concerning data transmission: confidentiality, authentication and integrity. As the sophistication of cryptanalytic attacks increases and their cost decreases, there is constant pressure to improve cryptographic methods on all three of these fronts [9], [10].

In cryptography, a **stream cipher** is a symmetric key cipher where plaintext bits are combined with a pseudorandom cipher bit stream (key stream), typically by an exclusive-or (XOR) operation. In a stream cipher the plaintext digits are encrypted one at a time, and the transformation of successive digits varies during the encryption [11, 12]. To send a secret message over a public channel one needs a secret key either for encryption, decryption or both. Through the last ten years, it has been shown how to use synchronization of neural network to generate secret keys over public channel. This algorithm called neural cryptography is not based on number theory but it contains a physical mechanism [13].

2.The Proposed System

The proposed system presents neural synchronization scheme where two neural networks are mutually learning each other and eventually stabilized at matched weight vector. This technique has been used to exchange secret keys over public channels as shown in Figure (1). The degree of security (complexity) of exchanging keys using neural network synchronization phenomenon is

proportional to two factors; first is the bidirectional interactivity (response to others output) of neural networks being synchronized, and second is the architecture of the neural networks being synchronized especially the number of hidden layers.

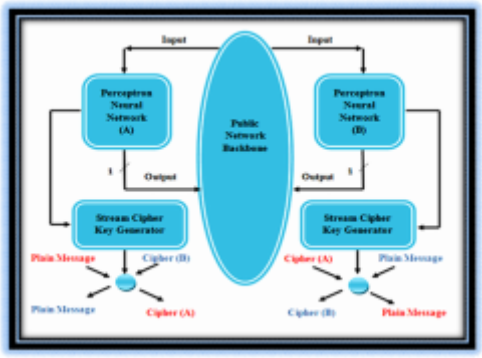

Figure 1: General skeleton scheme of the proposed system

NNs can be synchronized by learning from each other. For that purpose they receive common inputs and exchange their outputs. Adjusting discrete weights according to a suitable learning rule then leads to fully synchronization in finite steps. When synchronization is achieved the weights of each neural network will be the key exchanged over that public channel, and in this proposed system the key is an initialization vector to a stream cipher key generator. Stream key generator has to have identical initialization to be able to decrypt transmitted ciphers from certain source, in other words this approach is a point to point transmission, where parties have to participate in synchronization session to be able to encrypt and decrypt data in the right way.

The proposed system uses Linear Feedback Shift Registers (LFSRs) as the main component to build the stream cipher key generator, thus initialization in this case mean filling these registers with the exact 1's and 0's. The output of the shift registers does not represent the key but it is a memory address to memory which has balanced, randomly distributed 1's and 0's. This balance will maintain the randomness of the generated key. As it will be explained later, this proposed system has 5 shift registers that compose the address to 32 memory address, and since this proposed system does not focus upon the performance of the key generator rather than focusing on initializing it securely; this small amount is enough to generate random key with suitable period length.

3.Building Neural Network

The proposed system uses the perception neural network to be synchronized due to its features that match the tree parity structure needed to accomplish secure synchronization environment. The structure of the neural network is composed of one hidden layer and 12 input nodes as it is shown in Figure (2). This structure has to be possessed by all parties which want to exchange

keys based on synchronization of NNs. Each party, for example A and B, the project uses its own tree parity machine to get its weights to be the key received remotely. Synchronization of the tree parity machines (TPMs) is achieved in these steps:

Algorithm 1: Tree parity synchronization algorithm

1. Initialize random weight values for all nodes within the tree parity.
2. Execute these steps until the full synchronization is achieved:
    1. Generate random input vector X.
    2. Compute the values of the hidden neurons.
    3. Compute the value of the output neuron.
    4. Compare the values of both tree parity machines.
    5. IF outputs are different go to step 2.1.
    6. IF outputs are same: one of the suitable learning rules is applied to the weights.

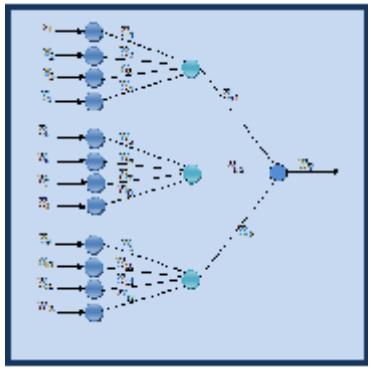

Figure 2: Perceptron NN used by this proposed system

After the full synchronization is achieved (the weights w_{ij} of both TPMs are same), A and B can use their weights as keys. This method is known as bidirectional learning. Random walk rule has been used to achieve synchronization, it is given by the following equation 1

$$w_i(t+1) = w_i + x_i\, \theta(\alpha\sigma)\theta(\sigma_A \sigma_B) \quad \ldots.1$$

where w_{t+1} : the next weight

w_i is the previous weight, x_i is the input element, α is the learning rate

σ_A and σ_B are outputs of tree parity in A and B.

4.Semantic Model for NN Synchronization

Semantic model is represented in figure (3) for the tree parity neural network which is ready to accomplish the synchronization. In this figure many more components are declared such as the networking modules and the addresses used to access other parties within the synchronization process.

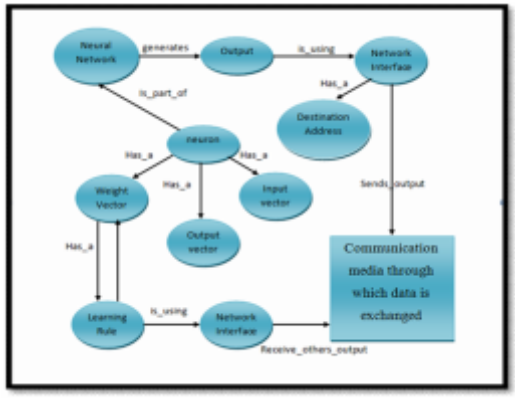

Figure 3: Semantic Model for NN Synchronization

5.Building Neuron Structure

The basic structure in NN is the neuron which represents the architectural and processing kernel component of the NN. Conceptually, a neuron has three components, as presented in Figure (4), where the output is the outcome of the activation components (i.e., $\sigma_i \in [-1, \Xi]$).

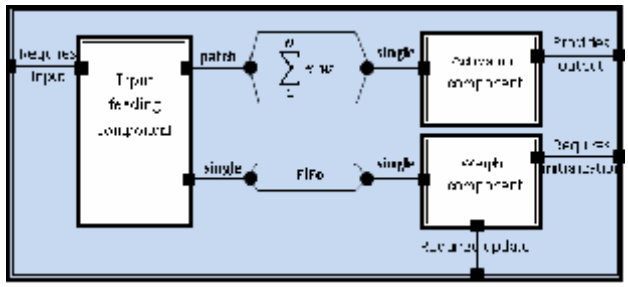

Figure 4 Conceptual View of Neuron

This data structure is going to be initialized and used as the main component of constructing NN. Figure (5) represents the flowchart of initializing neuron data structure.

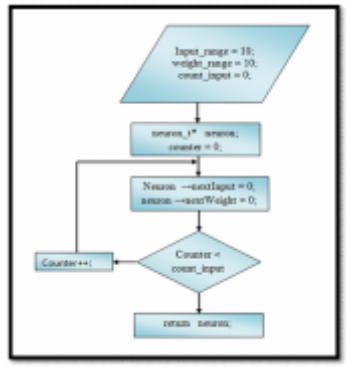

Figure 5: Neuron construction and initialization flowchart

After initializing the neuron; tree parity neural network is composed and the output of the neuron is calculated in the same manner at all levels. The method used to calculate the output is:

$$\text{sgn}(y) = \begin{cases} -1 & \text{if } y < 0 \\ 0 & \text{if } y = 0 \\ 1 & \text{if } y > 0 \end{cases}$$

6.Sequential Diagram

The complete synchronization session and data transfer is shown in figure (6), where the synchronization session has to be accomplished first and then the data can be exchanged securely. The synchronization session initiated between two parties (i.e., two tree parties) need only identical input vector, and the weights can be initialized to random values. NNs can't be synchronized with different input vectors.

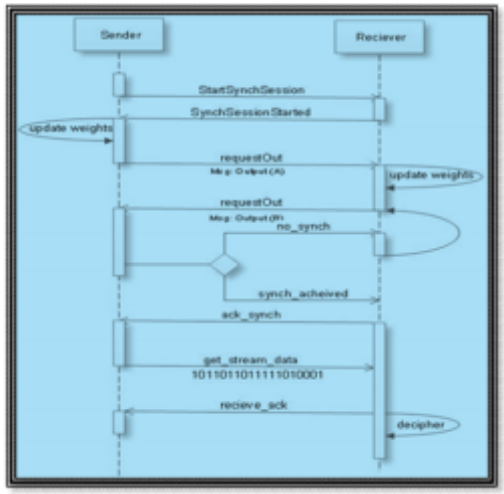

Figure 6: NNs synchronization and deploying Sequential Diagram

7.Synchronization Verification

Synchronization has to be verified before sending encrypted data, otherwise the data would not decrypt in the right way, and this is a crucial challenge in implementing NN synchronization in real world. Since tree parity produces only one value as output and sends it to other side; it will not be enough to get analogous output after certain count of iterations to confirm reaching synchronization state, even the number of iterations is huge, thus this proposed system is presenting a technique to confirm synchronization establishment. The used technique is based on simple vector mathematics as shown in figure (7).

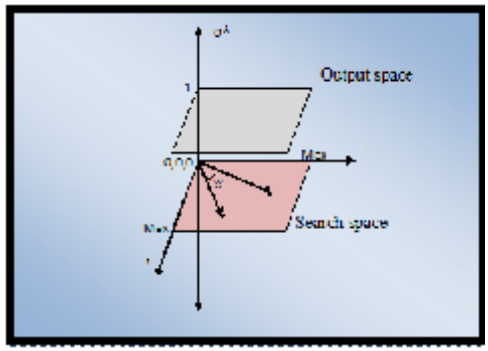

Figure 7: Geometrical representation of two NNs

$$w^{A}_{t+1} = w^{\rightarrow}_{t} + O^{\rightarrow} . X . f(\theta, \Box) \quad \ldots 2$$

$$w^{B}_{t+1} = w^{\rightarrow}_{t} + O^{\rightarrow} . X . f(\theta, \Box) \quad \ldots 3$$

$$w^{\rightarrow}_{t} = \vec{i}\, w^{\rightarrow}_{t} + \vec{j}\, w^{\rightarrow}_{t} \quad \ldots 4$$

$$\Box = \cos^{-1} \frac{w^{B}_{t+1} \cdot w^{B}_{t}}{|w^{B}_{t+1}|\,|w^{B}_{t}|} \quad \ldots 5$$

$$R = \sqrt{\left(\vec{i}\right)\left(w^{B}_{t+1} - w^{B}_{t}\right)^{2} + \left(\vec{j}\right)\left(w^{B}_{t+1} - w^{B}_{t}\right)^{2}} \quad \ldots 6$$

where R is the distance between 2 points,

$\Box$ is the angle between 2 vectors,

Synchronization will be achieved when:

$\theta, \Box = 0$

$$f(\theta, \Box) = R \cos \Box \quad \ldots 7$$

$$O^{\rightarrow} = \sum_{i=1}^{N} \vec{f}(x^{\rightarrow}_{i}, w_{i}) \; x \; f(\theta, \Box) \quad \ldots 8$$

$$O^{\rightarrow} = \sum_{i=1}^{N} \vec{f}(x^{\rightarrow}_{i}, w_{i}) \; x \; f(\theta, \Box) \quad \ldots 9$$

8.Socket Based Communication Layer

Socket networking is used to conduct the communication between parties that want to synchronize with each other. The only data transferred in the synchronization session is the output of each tree parity; which is in this case an integer value, thus no heavy traffic is thrown to the network. After accomplishing the synchronization, the same network layer can be used to transfer the cipher, and many more security layers can be added at this level.

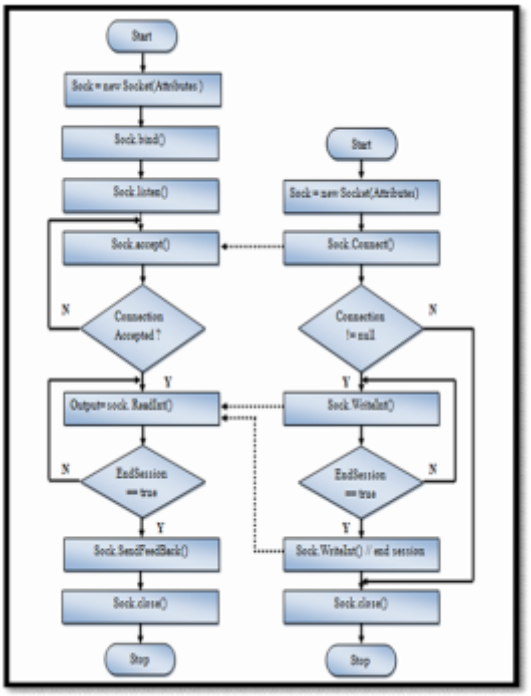

Figure 8: Network session initiation and exchanging neural output

9.Stream Cipher Algorithm

NN objectivity is to transfer secure key to remote destination over public channel, thus, eventually the key has to be used in some security algorithm to accomplish the task of transferring. In the proposed system the weights of the neural networks at each party represent the key used to initiate encryption algorithm.

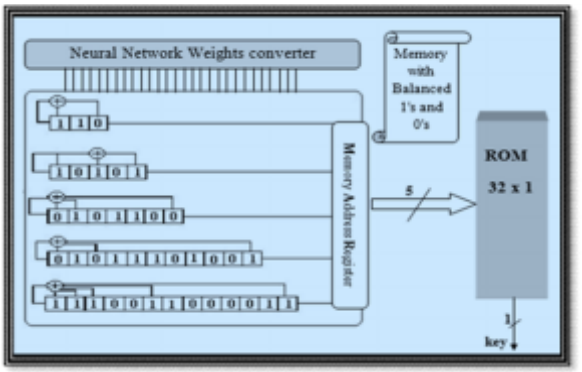

Figure 9: Stream cipher key generator used in the proposed system

In stream cipher every bit within the message should have key to be encrypted

$$\forall m_i \in M \quad \exists k_i \in Key \qquad \ldots 10$$

where

$M = \{m_1\ m_2 \dots \__n\}$ is the message to be encrypted.

$Key = \{k_1\ k_2 \dots k_n\}$ is the key used to encrypt message data.

Table 1: Maximum periods for LFSRs composing key generator

Bits	Tabs	Maximum Period
3	$x^3 + 1$	7
5	$x^5 + x^3 + 1$	31
7	$x^7 + x^6 + 1$	127
11	$x^{11} + x^{\_} + 1$	2047
13	$x^{13} + x^{12} + x^{11} + x^8 + 1$	8191

Maximum period of the system will be gained by multiplying all values of LFSRs composing the system; which yields 2^{39} (512 GBit). Neural synchronization process should be used to transfer new initialization for the system of the shift register before the expiry of the current period.

10.Key Exchange Management

NN synchronization basically depends on exchanging output to adjust the behavior of other remote NN; this will hold true if the input vector used to generate the output is identical in both networks. Two parties are a straight forward implementation where input vectors at each side is constant and previously setup as the input of the NN. In multi parties scenario the case is different; there is a compulsory need for more sophisticated policy to map destination address to a stream of binaries which represents the input vector agreed on both parties to synchronize upon. Hash function is used to construct hash table to map destination address to input vector as shown in figure (10).

Figure 10: Mapping destination to Input vector using hash table

11. Hash Table

Uni-cast and multi-cast have not been discussed and some aspects are left for future researches due to the restrictions imposed by this scheme, many parameters have to be involved such as the performance, network traffic, impersonating parties within sessions established to exchange keys and operability of the tree parity. Synchronizing multiple trees can produce weak points regarding the flexibility of the margins of the weights.

Holding hash table at each side rather than receiving it every time two parties need to synchronize can increase the performance of the overall system, previous efforts in NN synchronization focused on this technique; the proposed system uses different technique by holding hash table that binds approved parties to their input vectors. Hash table is very much like the ARP (Address Resolution Protocol) used by the TCP/IP to map physical addresses to IP (Internet Protocol) logical addresses.

12.Hacking Synchronization Session

Neural synchronization has a more significant privileged feature than other message exchange protocols which is continuous interactivity along the synchronization session while traditional key exchange algorithms rely heavily on the computation rather than continuous interactivity with simple computation; in the other hand the data transferred in neural synchronization does not possess the key. Data exchanged in neural synchronization is only the stepping value toward the key; other traditional algorithms exchange encapsulated packet, well encrypted

in sophisticated algorithm but yet hold the data. Figure (11) shows hacker GUI (Graphic User Interface) designed to hack neural synchronization session.

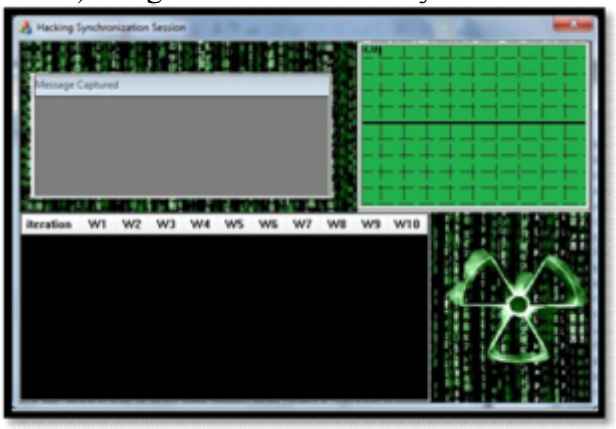

Figure 11: Hacker GUI to eavesdrop neural synchronization session

Data exchange in synchronization session is only one value that represents the output of the tree parity, and this amount of data gives nothing regarding the weights, Input vectors and tree parity structure are used to generate this output value. For an attacker: the only way to get the key is to be exchanged or to be transferred from point to point over neural synchronization session and to synchronize himself/herself with that session.

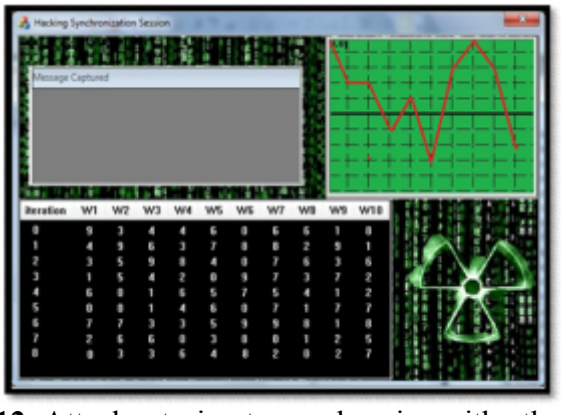

Figure 12: Attacker trying to synchronize with other parties

An attacker will not get his output to affect others' system due to the authentication used to protect the system from unauthorized participation. At the end of the synchronization session and after passing through all iterations, the hacker will not get his weights set to the appropriate values; Figure (13) demonstrates a screen shot from authorized participant and attacker program.

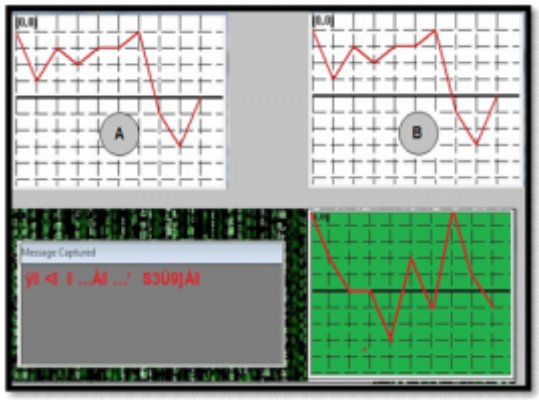

Figure 13: Attacker has not managed to synchronize

As it could be easily seen from Figure (13), the attacker could not manage to synchronize with others due to uni-direction learning algorithm used by the attacker. Many challenges are facing the attacker such as neural architecture at each party and furthermore; the balanced memory is not known for the attacker. This memory is intentionally mounted in order to harden hijacking synchronization session.

13. Conclusions

- Using Neural network synchronization in key exchange protocols due to its simple computation and less network traffic exchanged among parties who want to synchronize.
- The network session is using UDP (User Datagram Protocol) is due to the discrete behavior of neural synchronization session where next value would not be transmitted till other party's packet has arrived. This will expose this approach to many network threats such as denial of service, and other threats. This will eventually prevent parties from exchanging their keys.
- In case of losing the synchronization if some error occurs while transferring output to other party. This will grant attackers new approach to mess up key arrival by only damaging few packets transferred between parties who want to synchronize over random periods.
- When weights' seed range is high, the iterations will be too much and produce very heavy network communication and processing. In the same manner, tree parity architecture will affect the processing time due to the fact that increasing the number of nodes will increase the complexity in time and space.

References

[1] Simon Haykin, **"Neural Networks: a Comprehensive Foundation"**. Prentice Hall, 1999.

[2] Richard P.Lippman, **"An Introduction to Computing with Neural Nets"**. IEEE ASSP magazine1987.

[3] R. Hecht-Nielsen, **"Neurocomputing"**. HCN, Inc. and University of California, San Diego, 1990.

[4] Christos S. and Dimitrios S., **"Neural Networks''**. Available at www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html.

[5] Rolf Pfeifer, Dana Damia, Rudolf Fuchslin, **"Neural networks"**. 2007. Available at http://alib.ifiüzh.ch/images/stories/teaching/nn10/script/NN03032010.

[6] T. Schmidt, H. Rahnama, **"A Review of Applications of Artificial Neural Networks in Cryptosystems**". IEEE Xplore.2011

[7] Douglas Stinson, **"Cryptography, Theory and Practice"**. 2nd edition, CRC Press, 2002.

[8] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone, **"Handbook of Applied Cryptography"**. CRC Press, 1997.

[9] W. Stallings, **"Cryptography and Network Security: Principles and Practices"**. Prentice Hall, 2003.

[10] B. Schneier, **"Applied Cryptography"**. John Wiley & Sons Inc., New York, 1996.

[11] "Stream Cipher". Available at en.wikipedia.org/wiki/Stream_cipher.

[12] Henk C.A. van Tilborg, **"Fundamentals of Cryptology. A Professional Reference and Interactive Tutorial"**. Kluwer Academic Publishers, 2000.

[13] N.Prabakaran, P.Loganthan and P.Vivekanandan, **"Neural Cryptography with Multiple Transfers Functions and Multiple Learning Rule".** International journal of soft computing 3 (3): 177-281, © Medwell Journals, 2008.