Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 3159-0040
Vol. 2 Issue 1, January - 2015

An Analysis of Email Encryption using Neural Cryptography

Sudip Kumar Sahana
Department of Computer Science & Engineering
Birla Institute of Technology, Mesra
Ranchi , India
sudipsahana@bitmesra.ac.in

Prabhat Kumar Mahanti
Department of Computer Science & Applied Stat.
University of New Brunswick
Canada
pmahanti@unb.ca

***Abstract*— Encrypted Email using Neural Cryptography is a way of sending and receiving encrypted email through public channel. Neural key exchange, which is based on the synchronization of two tree parity machines, can be a secure replacement for this method. This paper present an implementation of neural cryptography for encrypted E-mail where Tree Parity Machines are initialized with random inputs vectors and the generated outputs are used to train the neural network. Symmetric key cryptography is used in encryption and decryption of messages.**

Keywords— Cryptography; Tree Parity Machine; Neural Key; Encryption; Decryption; Neurons.

I. INTRODUCTION

Cryptography [2] is used to encrypt the information and send it over public communication channel so that unauthorized users cannot have the access to the information and thus the confidentiality of the message is not leaked. Encryption converts plain text into cipher text and decryption revert back the cipher text into original plain text. The neural cryptography [5], a special type of cryptography, can be used to send encrypted emails from the sender's side and then decrypt the email at the receiver's end. Thus, the emails can be sent over unsecured channels and the chance of getting intercepted and message leaking is minimized. In neural cryptography, the neural key generated from the Tree Parity Machine (TPM) is used for the encryption and decryption of messages. To encrypt and decrypt the messages Rijndael standard is used. The neural key is generated from the Tree Parity Machine by synchronization rather than by learning because learning is always slower than synchronization. The electronic mails sent over the communication channel are generally scanned by the email service providers before delivering them to the recipients who helps them to categorize as spam and others. So, it is important that emails need to be sent in encrypted form. It is essential when highly confidential message are to be sent to dedicated users which cannot revealed to unauthorized users as it contains confidential data. Neural Cryptography has been used in various fields like Neural Key Exchange [1], Pseudo Random Number Generation [3], DES Cryptanalysis [6] and AES [9][11]. In this type of cryptography, Neural Key Exchange is used to synchronize the two neural networks by synchronization and Mutual learning is used in the neural key exchange protocol. The neural key can be generated once the network is synchronized.

II. METHODOLOGY

To increase the confidentiality [7], Encrypted Emails using Neural Cryptography can be used to send and receive encrypted emails over unsecured channels. The generation of neural key is done through the weights assigned to the Tree Parity Machine [10][13][14][15] either by a server or one of the systems. These inputs are used to calculate the value of the hidden neuron and then the output is used to predict the output of Tree Parity Machine, as shown in Fig 1.

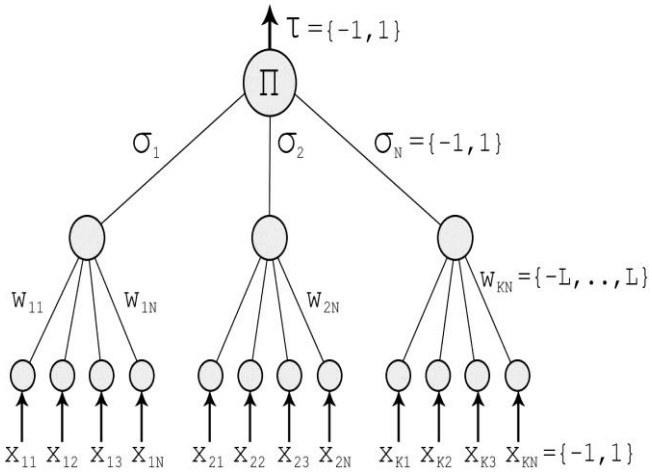

Fig.1. *Tree Parity Machine*

$$\text{sgn}(x)=\begin{cases}-1 \; if \; x<0\\ 0 \; if \; x=0\\ 1 \; if \; x>0\end{cases} \quad (1)$$

$$\sigma_i=\text{sgn}\left(\sum_{j=0}^{n} W_{ij}\, X_{ij}\right) \quad (2)$$

$$\tau=\prod_{i=1}^{n} \sigma_i \quad (3)$$

The outputs of the Tree Parity Machine are compared. If they are different then new inputs are assigned and the process is continued. And if the outputs of the Tree Parity Machine are same then the weights are adjusted according to one of the learning rules i.e.

Hebbian learning rule [1][4], Anti-Hebbian learning rule, or random walk method. The synchronization is continued till the weights become same in both the Tree Parity Machines [13]. Once the weights are synchronized, they are used to generate the neural key which is used to encrypt and decrypt the emails using Rijndael method [8]. In electronic mails service, generally all mails are scanned by the email service providers before delivering them to the recipients. So it is risky to send confidential in unencrypted form.

In our process, an extra layer of security is added to the emails and thus it can be also prevented from attackers from retrieving the emails by masquerading. This is beneficial when some confidential email is to be sent by companies, governments, etc and the confidentiality and integrity is to be preserved. The proposed flow diagram for Email Encryption using Neural Cryptography is shown in Fig 2.

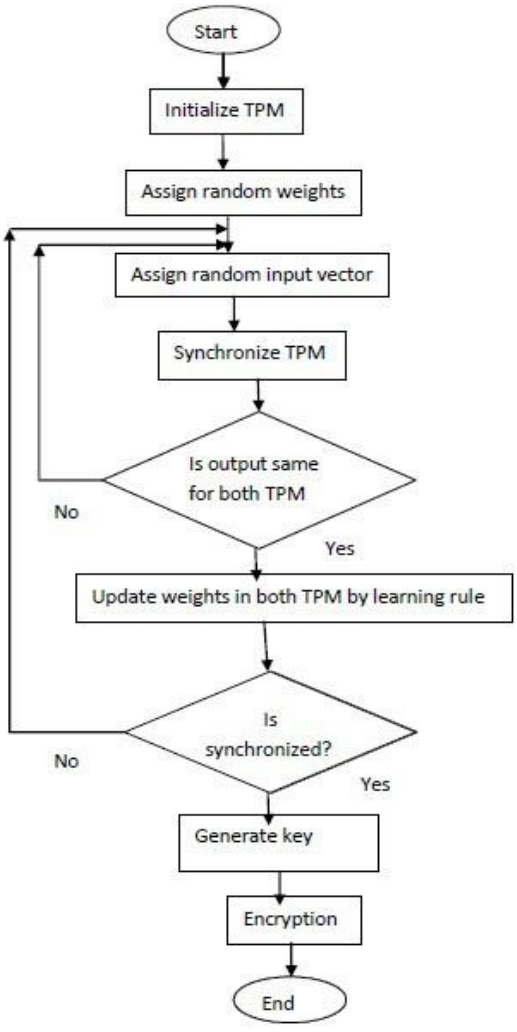

Fig.2. *Flow diagram for proposed Neural Cryptography*

The Algorithm used in neural key generation as follows:

1. Assign weights to the TPM's randomly.
2. Repeat till both the TPMs are synchronized.
   2.1 Assign the input vector X randomly
   2.2 Calculate the values of the hidden neurons using (2).
   2.3 Calculate the values of the output neurons using (3).
   2.4 The output of the TPM is compared
   If outputs are different, go to 2.1
   Else apply a learning rule (e.g. Hebbian, Anti-Hebbian and Random Walk) to the weights using equation (4), (5), (6) and (7).

Hebbian Learning Rule:

$$W_i^+ = W_i + \sigma_i X_i \, \Theta(\sigma_i \tau) \; \Theta(\tau^A \tau^B) \quad (4)$$

Anti-Hebbian Learning Rule:

$$W_i^+ = W_i - \sigma_i X_i \Theta(\sigma_i \tau) \, \Theta(\tau^A \tau^B) \quad (5)$$

Random Walk:

$$W_i^+ = W_i + X_i \, \Theta(\sigma_i \tau) \, \Theta(\tau^A \tau^B) \quad (6)$$

The function $\Theta_N(x)$ is a threshold function, which returns either 0 or 1:

$$\Theta_N(x) = \begin{cases} 0 \;\; if \; x \leq \frac{N}{2} \\ 1 \; if \; x > \frac{N}{2} \end{cases} \quad (7)$$

III. Implementation

Hardware and Software Specifications:

The application is implemented in java on Intel® Core™ i7 3667U CPU @2.00GHz 2.5 GHz processor with Internal RAM of 4GB with the operating system as Windows 8. The essential software required for running the application properly are Java platform with the JDK version 1.7. The IDE used for building the application is Netbeans 7.1.

The input, learning rule used and the output produced for our implementation is as shown below:

Input: The input weights vector to the Tree Parity Machine.

Learning Rule: Anti-Hebbian Learning Rule.

Output: The Neural Key which will be used to encrypt and decrypt the emails.

IV. Results and Discussions

The Neural Network has been used to generate the neural key. The key has been generated by synchronization of the two Tree Parity Machines. The security of the system depends on the synaptic depth of the weights used to generate the neural key [10]. The more the synaptic depth, the more it is difficult to crack the key. The total possible neural keys from a given value of Input vectors N, number of hidden neurons K, and the synaptic depth L is $(2L+1)^{(K*N)}$. The length of the neural key depend on the values of L, K and N. It is almost impossible to break the security of

the system. The strength of the neural key can be greatly increased by increasing the value of input vectors N and the synaptic depth L.

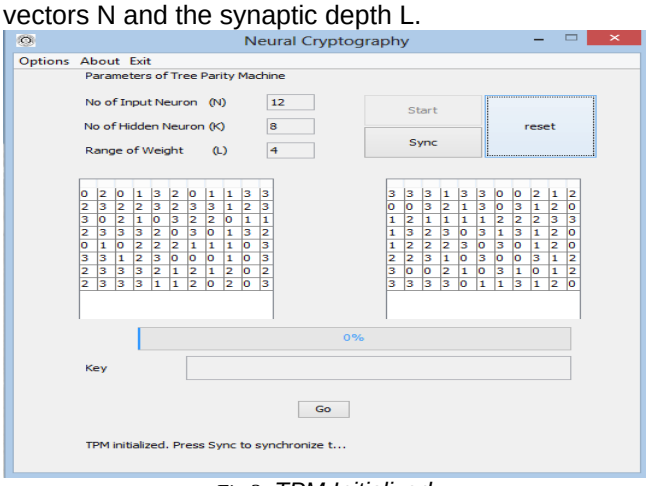

Fig.3. *TPM Initialized*

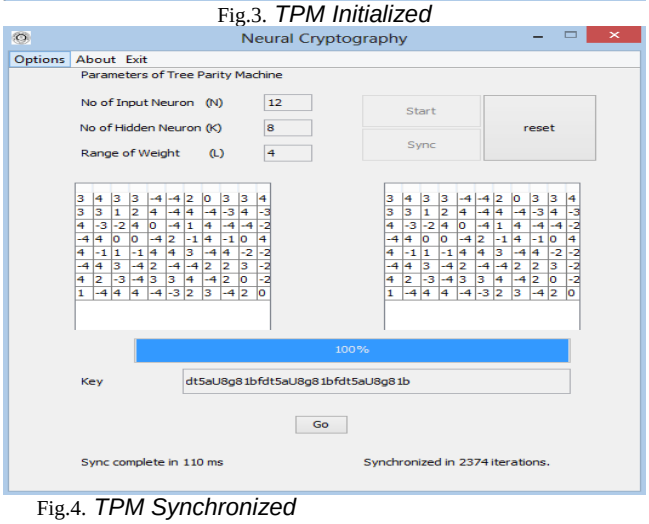

Fig.4. *TPM Synchronized*

The design interface allows the users to vary the number of input neurons (N), the hidden layer (K) and the range of weight (L). For a set of N, K and L, the initial Tree Parity Machine and the synchronized one is shown as in Fig 3 and 4 respectively for the generation of the neural key. Fig 5 indicates the time consumed for different inputs in milliseconds. Fig 6 and Fig 7 shows the set of inputs and the set of corresponding outputs obtained.

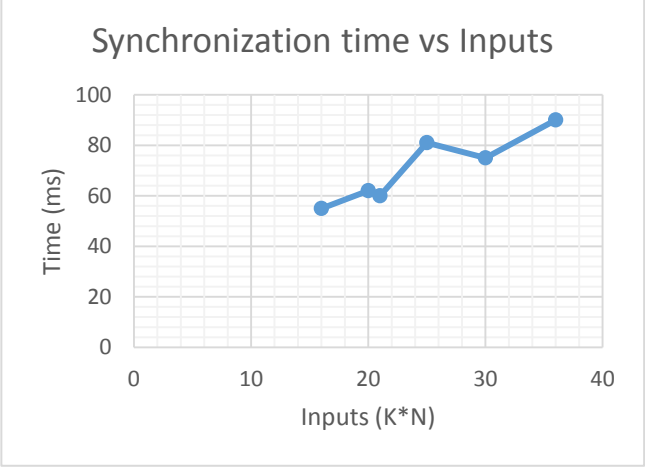

Fig.5. *Synchronization Time vs Inputs*

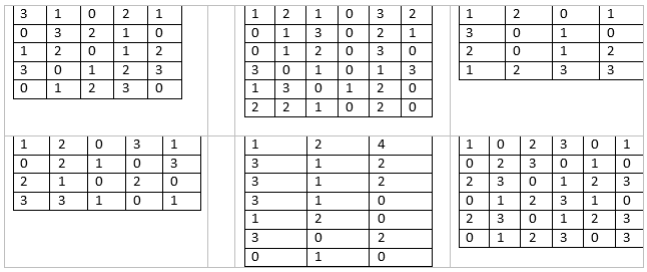

Fig.6. *Inputs(K*N)*

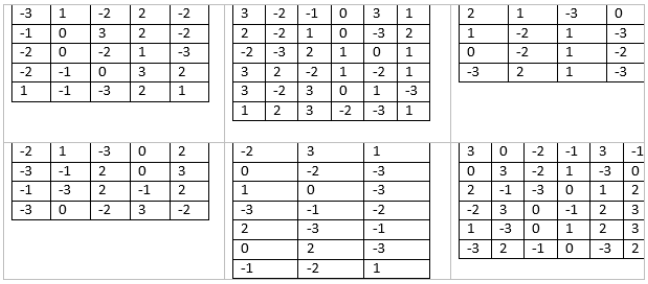

Fig.7. *Outputs(K*N)*

The system may be vulnerable to geometric and genetic attacks, but possibly these attacks cannot be successful because the learning is always slower than synchronization.

A. Geometric Attack

This attack is generally based on the geometric interpretation [10] of the action of a perception.
The procedure is as follows:

1) If Output A!= Output B,the attacker doesn't update output C

2) If Output A = Output B = Output C, attacker updates using the learning rule

3) If Output A= Output B! = Output C; The attacker does not update the weights but the two parties A and B update their weights. Since in this case the attacker does not update the weights so the synchronization of parties is faster than his learning.

B. Genetic Attack

Genetic attack [12] is dangerous and spreading in nature.

1) A biologically-inspired attack for a biologically-inspired cryptosystem

2) A large population of TPMs trained with same inputs are which is same as the weights of the two parity machines.

3) At each steps of synchronization, the TPM whose outputs are same as those of the TPM of the parties continue synchronization while others die away.

The final encrypted email and decrypted email for our interface is shown in Fig 8 and 9 respectively.

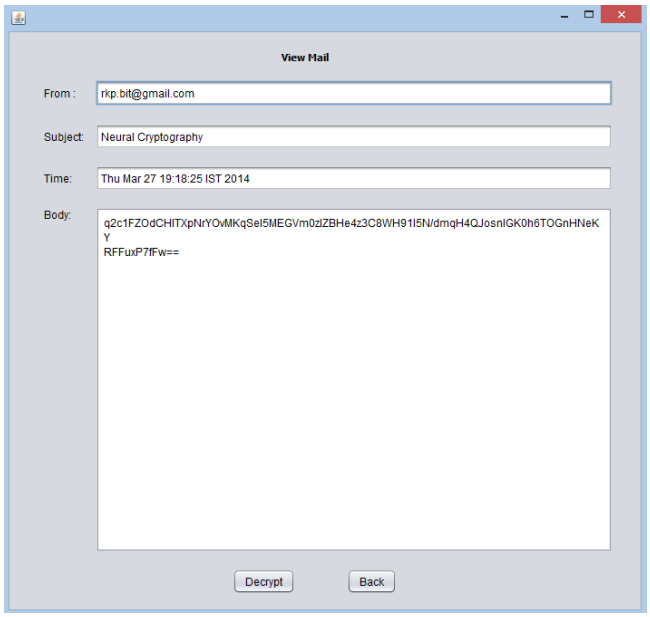

Fig.8. *Encrypted Email*

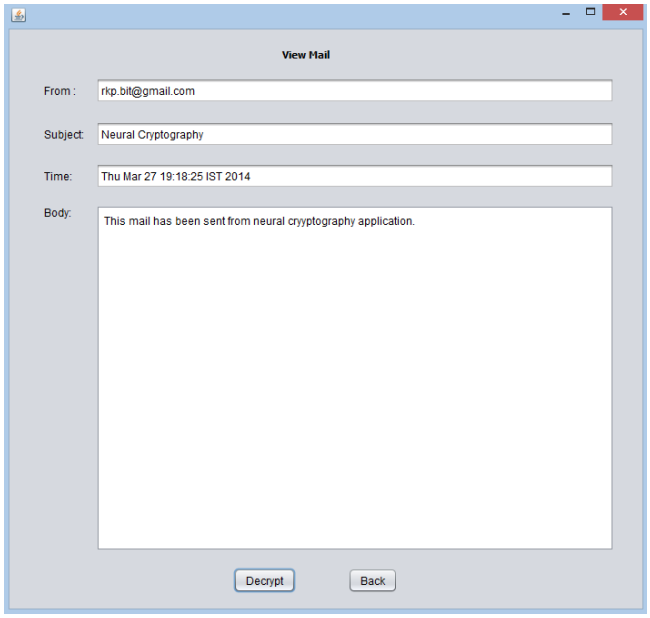

Fig.9. *Decrypted Email*

V. CONCLUSION

Encrypted Email using Neural Cryptography method proved it's superiority in terms of security and simplicity. In this approach the internal representation of the Tree Parity Machine is only known to the communicating parties. The weights of the Tree Parity Machine are used to generate the neural key after the synchronization is complete. As the intruder don't have the knowledge of the internal representation and the weights are evolved by the learning rules based on the output of the tree Parity Machines, so it becomes difficult for the attacker to guess the hidden neuron values and generate the key to decrypt the message sent over unsecured channels. The novelty of the approach is use of symmetric key concept which is very simple and easy to implement. Finally, this paper contributed an easy and more secure email encryption method.

Further, the security of the key can be increased by increasing the synaptic depth (L) of the key and the number of inputs to the Tree Parity Machine (N).

REFERENCES

[1] A.Singh and A. Nandal," Neural Cryptography for Secret Key Exchange and Encryption with AES", Int. Journal of Advanced Research in CS and SE,Vol3(5) , pp376-381,May 2013.

[2] A.Forouzan, Cryptography *and Network Security* , First Edition. McGraw-Hill, USA, 2007.

[3] D.A.Karras and V. Zorkadis, "Recurrent Neural Network Models for improved (Pseudo) Random Number Generation in computer security applications", Publisher: World Scientific and Engineering Academy and Society, ISBN: 9608052122

[4] A. Ruttor, "Neural Synchronization and Cryptography". PhD thesis, Bayerische Julius-Maximilians-Universität Würzburg, 2006.

[5] A.Klimov, A. Mityaguine and A. Shamir, "Analysis of Neural Cryptography", Proceedings: ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, Vol-2501, pp 288-298, December , 2002.

[6] M. Matsui , "Linear Cryptanalysis Method for DES Cipher", Advances in Cryptology-Eurocrypt-93,LNCS,Vol765 ,pp 386-397, 1994.

[7] A. Tardy-Corfdir and H. Gilbert, "A Known plaintext attck of FEAL-4 and FEAl-6", Advances in cryptology-CRYPTO'91,Lecture Notes in Computer Science, Vol.658,1992.

[8] J. Fran¸ C. Raymond1 and A. Stiglic2, "Security Issues in the Diffie-Hellman Key Agreement Protocol",2009. Available on: http://instantlogic.net/publications/DiffieHellman.pdf

[9] A. Jagadev, "Advanced Encryption Standard (AES) Implementation", M.Tech Thesis National Institute of Technology, Rourkela. May, 2009.

[10] Neural_cryptography, Available on: en.wikipedia.org/wiki/ .

[11] Advanced Encryption Standard, Available on: en.wikipedia.org/wiki/.

[12] A. Ruttor, W. Kinzel and I. Kanter, "Genetic attack on neural cryptography", *Phys. Rev.* E 73, 036121, 2006.

[13] M.Volkmer and S.Wallner, "Entity Auth-entication and Authenticated Key Exchange with Tree Parity Machines", Available on : http://eprint.iacr.org /*2006/112*, 2006.

[14] P. Revankar, W.Z. Gandhare and D. Rathod , ”Private inputs to Tree parity machine”, International Journal of Computer Theory and Engineering, Vol. 2, No. 4, pp 1793-8201,2010.

[15] T. Godhavari, N.R. Alainelu and R. Soundararajan, “Cryptography using neural network”, IEEE Indicon 2005 Conference, Chennai, India, pp.258-261, Dec. 2005.

Sudip Kumar Sahana, male, is currently working as Asst. Prof. in the Department of Computer Science and Engineering, B.I.T(Mesra), Ranchi, India. He received the B.E. Degree in Computer Technology from Nagpur University, India in 2001, the M.Tech. Degree in Computer Science in 2006 and Ph.D in Engineering in 2013 from the B.I.T (Mesra), Ranchi, India. His major field of study is in Computer Science. His research and teaching interests include Soft Computing, Evolutionary Algorithms, Artificial Intelligence and High Performance Computing. He is author of number of research papers in the field of Computer Science.

Prabhat Kumar Mahanti, , male, is Professor of Dept of Computer Science and Applied Statistics (CSAS), University of New Brunswick Canada. He obtained his M.Sc. from IIT-Kharagpur, India, and Ph.D. from IIT-Bombay India. His research interests include Software engineering, Software Metrics, Reliability Modelling, Modelling and Simulation, Numerical Algorithms, Finite Elements, Mobile and Soft computing, Verification of Embedded Software, Neural Computing, Data Mining, and Multi-Agent Systems. He has published research papers, technical reports, etc to his credit in national and International journals of repute.