ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

# Secure Key Exchange using Neural Network

**Vineeta Soni[*1], Mrs.Sarvesh Tanwar[2]**

[*1] Computer Science and Engineering, M.I.T.S University, Laxmangarh (Rajasthan), India

[2] Assistant Professor , Computer Science and Engineering, M.I.T.S University, Laxmangarh (Rajasthan), India

vineeta.soni89@yahoo.com

## Abstract

Any cryptographic system is used to exchange confidential information securely over the public channel without any leakage of information to the unauthorized users. Neural networks can be used to generate a common secret key because the processes involve in Cryptographic system requires large computational power and very complex. Moreover Diffi hellman key exchange is suffered from man-in –the middle attack. For overcome this problem neural networks can be used.Two neural networks which are trained on their mutual output bits. The networks synchronize to a state with identical time dependent weights. .secret key exchange over a public channel and this key can be used in implementing any encryption algorithm.

**Keywords**: Neural Network ;Cryptography ; Tree Parity Machine; Mutual Learning; Time dependent weights; Synchronizatio

## Introduction

Now a days information Security is becoming crucial aspect in every organization. In our work we are combining neural networks and cryptography to achieve a robust System against Man -in –the-middle attack.

Cryptography is the art of mangling information into apparaent unintelligibility in a manner allowing a secret method of unmangling. Cryptography is the ability to send information between participants in a way that prevents others from reading it. Original Message is known as plain text and it mangled from is known as cipher text. With the context of any application-to-application communication there are some security requirements like Authentication, Confidentiality, Integrity, Non-Repudiation etc.

Neural Network Artificial neural networks are parallel adaptive networks of consisting of simple nonlinear computing elements called neurons which are intended to abstract and model some of the functionalities of human nervous system in an attempt to partially capture some of its computational strength. They can be used to model complex relationships between inputs and outputs or to find patterns in data. A trained neural network can be a thought of as an “expert” in the category of information it has been given to analyse.

So by combining both technologies we can generate better results using less complex functions and providing better security.

## Neural Cryptography

Two identical systems, starting from different initial conditions can be synchronized by a common external signal which is coupled to the two systems. Both of the networks will be trained on their mutual output and can synchronized to a time independent state of identical synaptic weights. This rarity is applied to cryptography. In this case two partners in communication does not have a common secret key but they use their identical weights as a secret key for communication. This common weights can be used to form a key needed for encryption and decryption.

Synchronization by mutual learning can be faster than learning by listening. Neural cryptography is simpler than conventional cryptography and it is fast converging and secure also.

For this we have used a different type of neural network called Tree parity machine.

## Tree Parity Machine

For this work we have used a simple neural network which is called tree parity machine (TPM).these are special type of neural network.

ISSN: 2277-9655
Impact Factor: 1.852

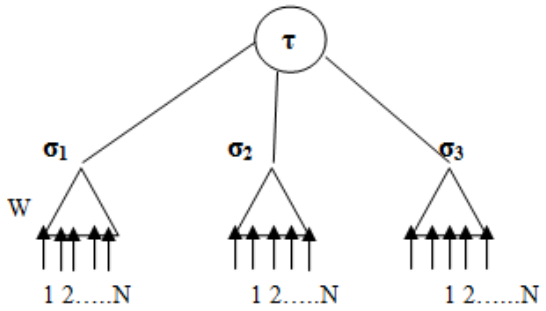


**Figure 1: Tree Parity machine**

This can be described by three parameters: K hidden layers, each of which can be seen as a single layer perceptron with N input neurons in each hidden layer, L is the maximum value for weight{-L…+L}. All input values are binary i.e.

$x_{i,j} \in \{-1,+1\}$ (1)

And the weights, which define the mapping from input to output, are discrete Numbers between −L and +L

$W_{i,j} \in \{-L,-L+1,\ldots,+L\}$ (2)

Output value of each hidden neuron is calculated as a sum of all multiplications of input neurons and these weights:

$\sigma_i = \mathbf{sgn} \sum_{j=1}^{N} W_{ij} X_i$ (3)

Two partners have the same neural machines and their output value is calculated by:

$$\tau = \prod_{i=1}^{K} \sigma_i \quad (4)$$

both networks receive common inputs vector X and select random initial weight vectors W. both the networks trained by their output bits τ(A)=τ(B).

the following learning rules can be applied:

1. If τ(A)≠τ(B) nothing is changed.
2. If τ(A)=τ(B)=τ only the hidden unit is changed for σk(A/B)=τ(A/B)
3. Three different rules can be considered for learning.

*a)Hebbian learning rule* :

$w_i^+ = w_i + \sigma_i x_i \theta(\sigma_i \tau) \theta(\tau^A \tau^B)$ (5)

*b) Anti-Hebbian learning rule:*

$w_i^+ = w_i - \sigma_i x_i \theta(\sigma_i \tau) \theta(\tau^A \tau^B)$ (6)

*c) Random-walk learning rule:*

$w_i^+ = w_i + x_i \theta(\sigma_i \tau) \theta(\tau^A \tau^B)$ (7)

Here, Theta is a special function. Theta (a, b) = 0 if a<>b; else Theta=1. The g(...) function keeps the weight in the range {- L..+L}

## Secret Key Generation

1. First of all determine the neural network parameters i.e. k, the number of hidden layer units n, the input layer units for each hidden layer unit l, the range of synaptic weight values is done by the two machines A and B.
2. The network weights to be initialized randomly.
3. Repeat 4 to 7 until synchronization occurs.
4. The inputs of the hidden units are calculated.
5. The output bit is generated and exchanged between the two machines A and B.
6. If the output vectors of both the machines are same i.e. $\tau_A = \tau_B$ then the corresponding weights are modified using the Hebbian learning rule, Anti-Hebbian learning rule and Random-walk learning rule.
7. After complete synchronization, the synaptic weights are same for both the networks. And these weights are used as secret key.

In this work we have increase the key size without increasing the weight range as a result we get maximum security with less synchronization time.

This key can be utilized to encrypt a sensitive message transmitted over an insecure channel using AES algorithm with Key size 128,192,256 bits.

## Security Analysis

By using Neural network we can show that An attacker E can eavesdrop messages between the parties A and B, but does not have an opportunity to change them.

*a) Brute force Attack*

To provide a brute force attack, an attacker has to test all possible keys (all possible values of weights $w_{ij}$ ). By K hidden neurons, K*N input neurons and boundary of weights L, this gives $(2L+1)^{KN}$ possibilities, making the attack impossible with today's computer power.

b) Learning with own tree parity machine

If an attacker owns the same tree parity machine same as the parties A and B. He wants to synchronize his tree parity machine with these two parties. In each step there are three situations possible:

. In each step there are three situations possible:

1. Output(A) ≠ Output(B): None of the parties updates its weights.
2. Output(A) = Output(B) = Output(E): All the three parties update weights in their tree parity machines.
3. Output(A) = Output(B) ≠ Output(E): Parties A and B update their tree parity machines,

but the attacker cannot do that. Because of this situation his learning is slower than the synchronization of parties A and B.

It has been proven, that the synchronization of two parties is faster than learning of an attacker. It can be improved by increasing of the synaptic depth L of the neural network. That gives this protocol enough security and an attacker can find out the key only with small probability.

## Implementation and Results

All work have been done in MATLAB and some data sets are obtained for synchronization time by varying number of input units.

| S.No. | Different issues | Without NN | With NN |
|---|---|---|---|
| 1. | Randomness | no | More |
| 2. | Security | less | More |
| 3. | Synchronization time | Not required | required |

The number of iterations required for synchronization by varying number of input units. If the value of n increases, the synchronization time and number of iteration also increases

## Conclusion and Future Work

We presented a bridge between cryptography and network security.It provides greater security and also great speed than complex cryptographic algorithm which requires large computational power.our future work will make use of secret key generated by neural network in advance encryption algorithm like triple DES,IDEA. We can use a key distribution centre which can generate as well as distribute the secret key among several parties securely

Moreover, we can use this key in generating hash functions. Neural networks can be used in generating one way hash function by using its confusion and diffusion and compression properties. This hash function will be less complex than in cryptographic hashes.and we can also analyze the security of system against meetin middle attach and birthday attacks

## *References*

1. *Dr. Ajit Singh, Aarti nandal ,"Neural Cryptography for Secret Key Exchange and Encryption with AES", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 5, May 2013 ISSN: 2277 128X*
2. *Wright, Jason L., and Milos Manic. "Neural network approach to locating cryptography in object code." Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on. IEEE, 2009.*
3. *Arvandi, M., S. Wu, and A. Sadeghian. "On the use of recurrent neural networks to design symmetric ciphers." Computational Intelligence Magazine, IEEE 3.2 (2008): 42-*
4. *Hen, Tieming, and Rongrong Jiang. "Designing Security Protocols Using Novel Neural Network Model." Natural Computation, 2007. ICNC 2007. Third International Conference on. Vol. 1. IEEE, 2007.*
5. *Liu, Niansheng, and Donghui Guo. "Security analysis of public-key encryption scheme based on neural networks and its implementing"Computational Intelligence and Security. Springer Berlin Heidelberg, 2007. 443-450..*
6. *Arvandi, Maryam, and Alireza Sadeghian. "Chosen Plaintext Attack against Neural Network-Based Symmetric Cipher." Neural Networks, 2007. IJCNN 2007. International Joint Conference on. IEEE, 2007*
7. *William, S., & Stallings, W. (2006). "Cryptography and Network Security, 4/E". Pearson Education India.*
8. *Godhavari, T., N. R. Alamelu, and R. Soundararajan. "Cryptography using neural network." INDICON, 2005 Annual IEEE. IEEE, 2005.*
9. *Mislovaty, R., et al. "Security of neural cryptography." Electronics, Circuits and Systems, 2004. ICECS 2004. Proceedings of the 2004 11th IEEE International Conference on. IEEE, 2004.*
10. *Wolfgang, and Ido Kanter. "Interacting neural networks and cryptography, Springer Berlin Heidelberg, 2002. 383-391.*
11. *Klimov, Alexander, Anton Mityagin, and Adi Shamir. "Analysis of neural cryptography." Advances in Cryptology—ASIACRYPT 2002. Springer Berlin Heidelberg, 2002. 288-298.*
12. *Haykin, Simon. "Neural networks: a comprehensive foundation. Prentice Hall PTR, 1994"*