Comparative Analysis of Tree Parity Machine and Double Hidden Layer Perceptron Based Session Key Exchange in Wireless Communication

Arindam Sarkar and J. K. Mandal

Department of Computer Science & Engineering,
University of Kalyani, Kalyani-741235,
Nadia, West Bengal, India
{arindam.vb,jkm.cse}@gmail.com

Abstract. In this paper, a detail analysis of Tree Parity Machine (TPM) and Double Hidden Layer Perceptron (DHLP) based session key exchange tecchnique has been presented in terms of synchronization time, space complexity, variability of learning rules, gantt chart, total number of threads and security. TPM uses single hidden layer in their architecture and participated in mutual learning for producing the tuned weights as a session key. DHLP uses two hidden layers instead of single. Addition of this extra layer enhances the security of the key exchange protocol. Comparisons of results of both techniques has been presented along with detail analysis.

Keywords: Tree Parity Machine (TPM), Double Hidden Layer Perceptron (DHLP), session key, wireless communication.

1 Introduction

These days a range of techniques are available to exchange session key. Each technique has its own advantages and disadvantages. In key exchange the main security intimidation is Man-In-The-Middle (MITM) attack at the time of exchange the secret session key over public channel. Diffie-Hellman key exchange algorithm suffers from this MITM attack. Most of the key generation algorithms in Public-Key cryptography suffer from MITM attack [1]. Where intruders can reside middle of sender and receiver and tries to capture all the information transmitting from both parties. Another noticeable problem is that most of the key generation algorithms need large amount of memory space for storing the key but now-a-days most of the handheld wireless devices have a criterion of memory constraints. In proposed DHLPSKG, problem MITM attack of Diffie-Hellman Key exchange [1] has been set on. In TPM and DHLP based session key generation procedure, both sender and receiver use identical architecture. Both of these DHLP [2] and TPM's [3,4,5,6,7] start with random weights and identical input vector.

The organization of this paper is as follows. Section 2 of the paper deals with the TPM synchronization algorithm. DHLP based protocol for generation of session key has been discussed in section 3. Section 4 deals experimental results and discussions. Section 5 provides conclusions and future scope and that of references at end.

2 Tree Parity Machine (TPM)

In recent times it has been discovered that Artificial Neural Networks can synchronize. These mathematical models have been first developed to study and simulate the activities of biological neurons. But it was soon discovered that complex problems in computer science can be solved by using Artificial Neural Networks. This is especially true if there is little information about the problem available. Neural synchronization can be used to construct a cryptographic key-exchange protocol. Here the partners benefit from mutual interaction, so that a passive attacker is usually unable to learn the generated key in time. If the synaptic depth (L) is increased, the complexity of a successful attack grows exponentially, but there is only a polynomial increase of the effort needed to generate a key. TPM based synchronization steps are as follows [3, 4, 5, 6, 7].

TPM Synchronization Algorithm

Step 1. *Initialization of random weight values of TPM.*

Where, $w_{ij} \in \{-L, -L+1, \ldots, +L\}$ (1)

Step 2. *Repeat step 2 to 5 until the full synchronization is achieved, using Hebbian-learning rules.*

$$w_{i,j}^{+} = g\left(w_{i,j} + x_{i,j}\tau\Theta(\sigma_i\tau)\Theta\left(\tau^A\tau^B\right)\right) \quad (2)$$

Step 3. *Generate random input vector X. Inputs are generated by a third party or one of the communicating parties.*

Step 4. *Compute the values of the hidden neurons using (eq. 3)*

$$h_i = \frac{1}{\sqrt{N}} w_i x_i = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} w_{i,j} x_{i,j} \quad (3)$$

Step 5. *Compute the value of the output neuron using (eq. 4)*

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

Compare the output values of both TPMs by exchanging the system outputs.

if Output (A) ≠ Output (B), Go to step 3

else if Output (A) = Output (B) then one of the suitable learning rule is applied only the hidden units are trained which have an output bit identical to the common output.

Update the weights only if the final output values of the TPM's are equivalent. When synchronization is finally occurred, the synaptic weights are same for both the system.

3 Double Hidden Layer Perceptron (DHLP)

In DHLP [2] based key exchange method each session after accepting an identical input vector along with random weights both DHLP's, compute their outputs, and communicate to each other. If both outputs are same then both DHLP's starts synchronization steps by updating their weights according to an appropriate learning rule. After the end of the full synchronization procedure weight vectors of both DHLP's become identical. These indistinguishable weight vector forms the session key for a particular session. So, as a substitute of transferring the session key through public channel DHLP based synchronization process is carried out and outcomes of this used as a secret session key for that entire session. That actually helps to get rid of famous Man-In-The-Middle attack.

- DHLP offers two hidden layers instead of one single hidden layer in TPM [3, 4, 5, 6, 7]
- Instead of increasing number of hidden neurons in a single hidden layer DHLP introduces an additional layer (second hidden layer) which actually increased the architectural complexity of the network that in turn helps to make the attacker's life difficult to guessing the internal representation of DHLP.
- Secondly DHLP uses Hopfield Network for generation of random input vector.
- DHLP offers weight vectors of discrete values for faster synchronization.
- DHLP uses frames for connection establishment and synchronization procedure.
- DHLP introduces authentication steps along with synchronization.

DHLP Synchronization Algorithm

Input: - Random weights and Hopfield Network based PRNG generated input vector
Output: - Secret session key through synchronization.
Method:-

Step 1. Initialization of synaptic links between input layer and first hidden layer and between first hidden layer and second hidden layer using random weights values. Where, $w_{ij} \in \{-L, -L+1, \ldots, +L\}$

Repeat step 2 to step 12 until the full synchronization is achieved,

Step 2. The input vector (x) is created by the sender using 128 bit seed of Hopfield Network based PRNG.

Step 3. Computes the values of hidden neurons by the weighted sum over the current input values.

Step 4. Compute the value of the final output neuron by computing multiplication of all values produced by K2 no. hidden neurons at the second hidden layer:

$$\tau = \prod_{p=1}^{K2} \sigma_p^2$$

Step 5. Sender utilizes its 128 first weights as key for encryption of T variable (formerly stored in its memory) $Encrypt_{\text{sender\_weight}}(T)$.

Step 6. Sender constructs a SYN frame and transmitted to the receiver for handshaking purpose in connection establishment phase. SYN usually comprises of command code, ID, Secret Seed, Sender output (τ^{Sender}), $Encrypt_{\text{sender\_weight}}(T)$ and CRC (Cyclic Redundancy Checker).

Step 7. Receiver performs Integrity test after receiving the SYN frame and then Receiver utilize its 128 first weights as key for decryption of $Encrypt_{\text{weight}}(T)$ that was received from the sender.

Step 8. If $(Decrypt_{\text{receiver\_weight}}\,(Encrypt_{\text{sender\_weight}}(T)) = T$ then networks are synchronized. Go to step 12.

Step 9. If $(Decrypt_{\text{receiver\_weight}}\,(Encrypt_{\text{sender\_weight}}(T)) \neq T$ then receiver use the secret seed (SS) received from sender to produce the receiver inputs (x) identical to sender input (x) and calculates the output τ^{Receiver} using step 3 and step 4.

Step 10. If $(\tau^{\text{Receiver}} = \tau^{\text{Sender}})$ then performs the following steps

Step 10.1 Receiver should update their weights where $\sigma_k^{Sender/\mathrm{Re}ceiver} = \tau^{Sender/\mathrm{Re}ceiver}$ using any of the following learning rules:

Anti-Hebbian:

$$W_k^{A/B} = W_k^{A/B} - \tau^{A/B} x_k \Theta(\sigma_k \tau^{A/B})(\tau^A \tau^B)$$

Hebbian:

$$W_k^{A/B} = W_k^{A/B} + \tau^{A/B} x_k \Theta(\sigma_k \tau^{A/B})(\tau^A \tau^B)$$

Random walk:

$$W_k^{A/B} = W_k^{A/B} + x_k \Theta(\sigma_k \tau^{A/B})(\tau^A \tau^B)$$

Step 10.2 At the end of receivers weights update, the receiver sends ACK_SYN to instruct the sender for updating the weights using step 10.1.

Step 10.3 Sender then transmits encrypted message to receiver.

Step 10.4 Receivers then checks

If $(Decrypt_{\text{receiver\_updated\_weight}}(Encrypt_{\text{sender\_updated\_weight}}(T)) = T$ then networks are synchronized. Go to step 12.

Step 10.5 if $(Decrypt_{\text{receiver\_updated\_weight}}(Encrypt_{\text{sender\_updated\_weight}}\,(T) \neq T$

then networks are still not synchronized. Go to step 10.1.

Step 11. If $(\tau^{\text{Receiver}} \neq \tau^{\text{Sender}})$ then the receiver sends the message NAK_SYN to notify the sender. Go to step2.

Step 12. Finally, the receiver sends the frame FIN_SYN to inform the sender regarding the index vector of the weight to form the final session key.

4 Result and Analysis

In this section result and analysis of the TPM and DHLP based key exchange approach has been presented.

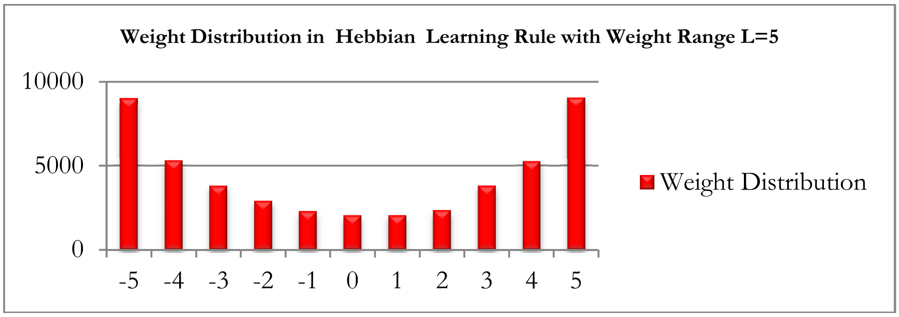

Fig. 1. TPM based weight distribution

Figure 1 shows the weight distribution of TPM based method where are not uniformly distributed.

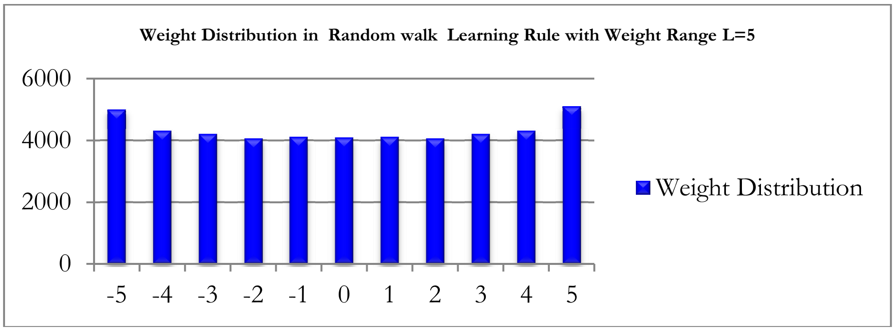

Fig. 2. DHLP based weight distribution

Figure 2 shows the weight distribution of DHLP based method where weight are uniformly distributed.

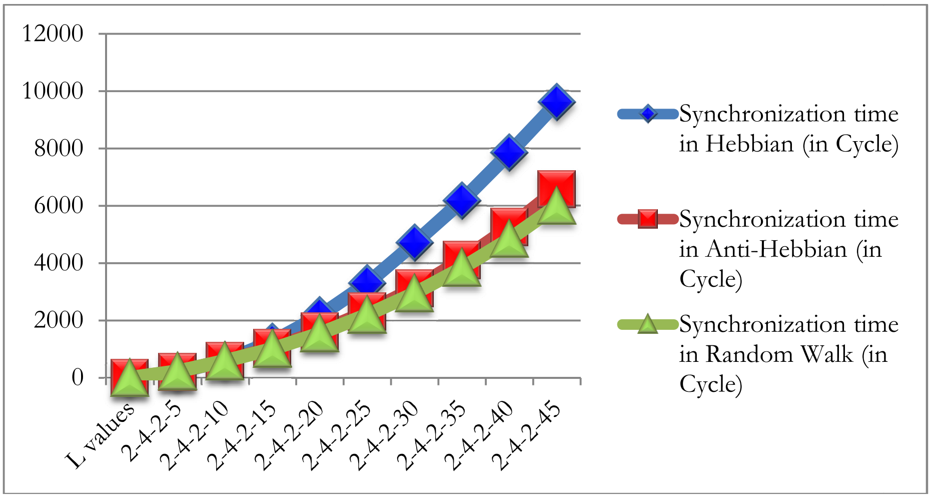

Fig. 3. DHLP based synchronization vs. learning rules

Figure 3 shows the Random-Walks takes minimum synchronization time than other two.

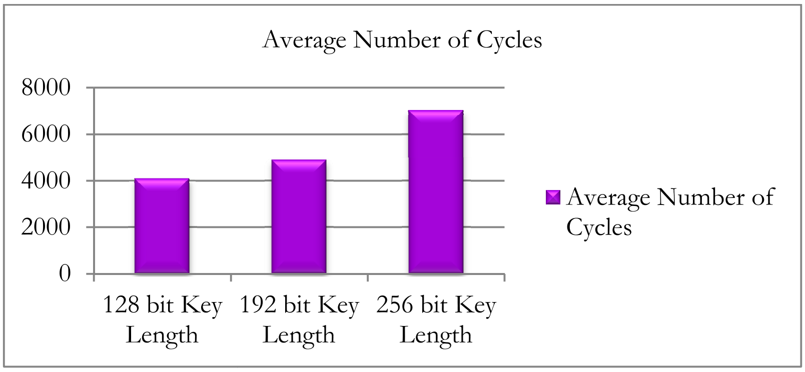

Fig. 4. Key length Vs. Average number of iterations

Figure 4 shows the increase of average number of cycles required for DHLP synchronization for different key length.

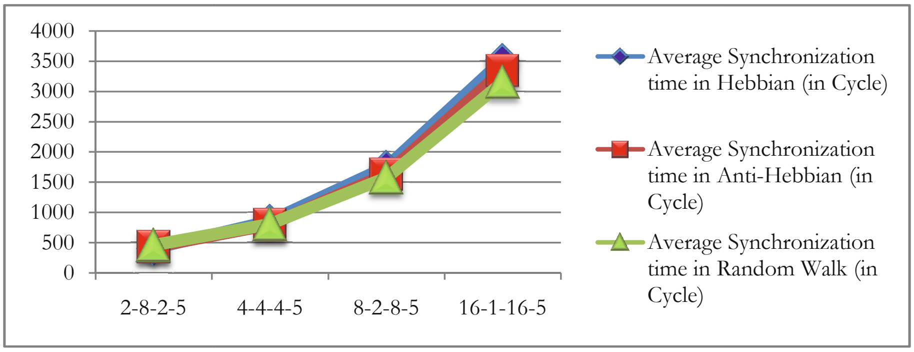

Fig. 5. Generation of 256 bit Session Key using fixed weight Range ($L = 5$) and different number of Neurons in Input and Hidden Layer

Figure 5 shows the generation of 256 bits key using fixed weight range and different number of neurons.

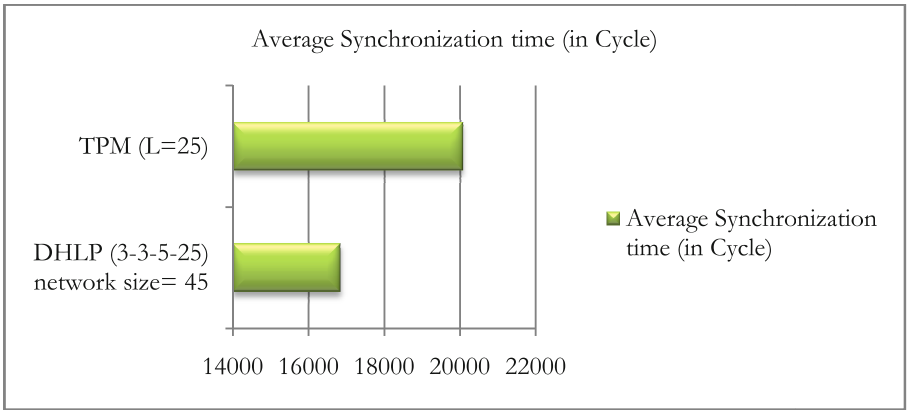

Fig. 6. Comparisons of 192 bit key length vs. average synchronization time (in cycle) for group synchronization (Group size= 4)

Figure 6 shows the Comparisons of 192 bit key length vs. average synchronization time (in cycle) for group synchronization (Group size= 4)

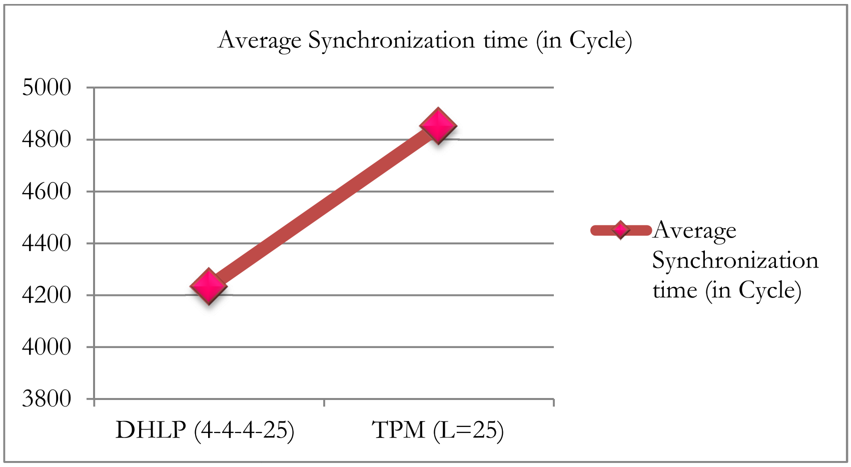

Fig. 7. Comparisons of 256 bit key length vs. average synchronization time (in cycle)

Figure 7 shows the Comparisons of 256 bit key length vs. average synchronization time (in cycle) .

5 Future Scope and Conclusion

Till now all session key generation techniques already devised [1] in concentrated only in session key generation mechanism but the process of generating keys does not guarantee the information security. Therefore, any attacker can also synchronize with an authorized device, because the protocol is a public knowledge. Thus, to ensure that only entities authorized have access to information is necessary authentication service. The function of the authentication service is to ensure the recipient that the message is from the source that it claims. There are several authentication methods, differentiated mainly by the use of secret-keys or public-keys. DHLP performs secret keys authentication where both entities must have a common secret code. Where as TPM has no such facility in terms of authentication.

Both the technique does not cause any storage overhead. Both of them needs a minimum amount of storage for storing the key which greatly handles the memory constraints criteria of wireless communication. The implementation on practical scenario is well proven with positive outcomes. In future group key exchange mechanism DHLP based protocol can be applied

Acknowledgment. The author expressed deep sense of gratitude to the Department of Science & Technology (DST) , Govt. of India, for financial assistance through INSPIRE Fellowship leading for a PhD work under which this work has been carried out, at the department of Computer Science & Engineering, University of Kalyani.

References

1. Kahate, A.: Cryptography and Network Security, Eighth reprint. Tata McGraw-Hill Publishing Company Limited (2006)
2. Arindam, S., Mandal, J.K.: Group Session Key exchange Multilayer Perceptron based Simulated Annealing guided Automata and Comparison based Metamorphosed Encryption in Wireless Communication (GSMLPSA). International Journal of Wireless & Mobile Networks (IJWMN) 5(4), 203–222 (2013), doi:10.5121/ijwmn.2013.5415, ISSN 0975 - 3834 (Online), 0975 - 4679 (Print); Mislovaty, R., Perchenok, Y., Kanter, I., Kinzel, W.: Secure key-exchange protocol with an absence of injective functions. Phys. Rev. E 66, 066102 (2002)
3. Ruttor, A., Kinzel, W., Naeh, R., Kanter, I.: Genetic attack on neural cryptography. Phys. Rev. E 73(3), 036121 (2006)
4. Engel, A., Van den Broeck, C.: Statistical Mechanics of Learning. Cambridge University Press, Cambridge (2001)
5. Godhavari, T., Alainelu, N.R., Soundararajan, R.: Cryptography Using Neural Network. In: IEEE Indicon 2005 Conference, India, December 11-13 (2005)
6. Kinzel, W., Kanter, L.: Interacting neural networks and cryptography. In: Kramer, B. (ed.) Advances in Solid State Physics, vol. 42, p. 383 arXiv- cond-mat/0203011. Springer, Berlin (2002)
7. Kinzel, W., Kanter, L.: Neural cryptography. In: Proceedings of the 9th International Conference on Neural Information Processing (ICONIP 2002) (2002)