# MUTUAL LEARNING IN TREE PARITY MACHINES USING CUCKOO SEARCH ALGORITHM FOR SECURE PUBLIC KEY EXCHANGE

**Article** *in* ICTACT Journal on Soft Computing · January 2018

DOI: 10.21917/ijsc.2018.0231

CITATIONS
2

READS
4

**5 authors**, including:

Shikha Gupta
Netaji Subhas Institute of Technology
**8** PUBLICATIONS **20** CITATIONS
SEE PROFILE

Nishi Gupta
Netaji Subhas Institute of Technology
**8** PUBLICATIONS **32** CITATIONS
SEE PROFILE

ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2018.0231



# MUTUAL LEARNING IN TREE PARITY MACHINES USING CUCKOO SEARCH ALGORITHM FOR SECURE PUBLIC KEY EXCHANGE

**Shikha Gupta[1], Nalin Nanda[2], Naman Chhikara[3], Nishi Gupta[4] and Satbir Jain[5]**
[1,4,5]*Department of Computer Science and Engineering, Netaji Subhash Institute of Technology, India*
[2,3]*Department of Electronics and Communication Engineering, Netaji Subhash Institute of Technology, India*

***Abstract***

***In Neural Cryptography, Artificial Neural Networks are used for the process of key generation and encryption. Tree Parity Machine (TPM) is a single layer neural network that approaches symmetric key exchange using the process of mutual learning. This method is exploited to design a secure key exchange protocol, where the sender and the receiver TPMs are synchronized to obtain an identically tuned weight vectors in both the networks. The synchronized TPMs are then capable of generating a key stream. The time required for synchronization depends on the initial weight vectors which are randomly initialized. In the proposed method, the process of synchronization is expedited using Cuckoo Search (CS) Algorithm used for the generation of optimal weights.***

***Keywords:***

***Neural Synchronisation, Tree Parity Machine, Cuckoo Search Algorithm, Key Exchange, Security***

## 1. INTRODUCTION

Cryptography is the practice of establishing secure communication between two parties by preventing unauthorised access by an adversary to maintain the confidentiality and integrity of data. It describes the procedure to transmit data between involved parties such that any eavesdropper is unable to retrieve the original message. Cryptographic techniques are based on private key and public key cryptography. In private key cryptography single key is used for both encryption and decryption and in public key two keys are used one for encryption and other for decryption. Though public key cryptography is not considered god for security but still it plays an important role in maintaining the key exchange between the two parties. Public key exchange protocols have a special place in cryptography ever since it has been introduced by Diffie and Hellman [1]. This protocol enables the party to share the common secret key on public communication channel without compromising the security concerns of procurement of key by an adversary. Neural cryptography creates a channel for secure key exchange by mutual learning for synchronisation of special kind of neural networks called Tree Parity Machines (TPM) [2]. The communicating networks receive identical inputs, and are trained on the basis of the outputs they generate. This process leads to synchronisation of the TPMs wherein the synaptic weights of the networks update and converge to an identical set of weight vectors, once the synchronisation process is completed [3]. These identical weights serve as the secret key crucial for the transmission of data. In this paper a new model for synchronisation is proposed where Cuckoo Search Algorithm is exploited to expedite the process of synchronisation.

The remaining paper is divided in following sections: Section 2 and 3 describes a brief of tree parity machine and cuckoo search algorithm. In section 4, proposed model have been explained and analysed. In section 5 results are analysed along with future directions followed by section 6 that concludes the paper.

## 2. TREE PARITY MACHINE (TPM)

TPMs are special breed of feed forward neural networks that possess the ability of synchronisation. The Fig.1 shows the general structure of TPM.

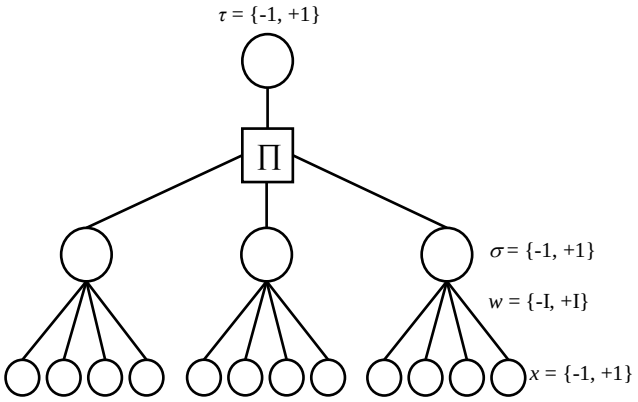



Fig.1. An example of Tree Parity Machine with $K = 3$ and $N = 4$

The neural network has $K$ hidden units, which each have their own receptive fields i.e. each hidden unit has $N$ inputs while there is only one output neuron. The network accepts binary input,

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

The weight vector associated with the inputs of the network, is a set of discrete numbers ranging from $-L$ to $+L$,

$$w_{i,j} \in \{-L, -L+1, ..., L-1, L\} \quad (2)$$

where $i = 1, 2, ..., K$ denotes the $i^{th}$ hidden unit of the TPM and $j = 1, 2, \ldots, N$ denotes the corresponding input to the hidden unit. Now the output of the hidden neurons of the neural network is defined by the weighted sum of its input vector. A function is defined as shown in Eq.(3)

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

The Eq.(3) shows the local field of hidden neurons in a network. The output $\sigma_i$ uses the signum function as the activation function with $h_i$ as the input shown in Eq.(4).

$$\sigma_i = \text{sgn}(h_i) \quad (4)$$

The network output is defined as the product of the hidden units (or parity), given by the Eq.(5)

$$\tau = \prod_{i=1}^{K} \sigma_i \tag{5}$$

So, the output $\tau$ is the measure of the number of inactive hidden units ($\sigma_i$ = -1) in the neural network i.e. $\tau$ = +1 when even number of hidden neurons are not active (even parity) and $\tau$ = -1 when odd number of hidden neurons are not active (odd parity).

## 3. CUCKOO SEARCH ALGORITHM

Cuckoo Search Algorithm is defined as a meta-heuristic search algorithm proposed by Yang and Deb [4, 5]. It is inspired by the behaviour of some of the cuckoo species. The algorithm simulates the obligate brood parasitic behaviour of some cuckoo species in combination with Lévy flight patterns of some birds and fruit flies. The cuckoo bird is known to not construct its nest; rather it lays its eggs in the nest of another host bird. Few host birds have the ability to discern their own eggs from others. A host bird that identifies the eggs of unknown origins can either throw away the egg or abandon the nest altogether and build a nest somewhere else. To prevent this, some species of female parasitic cuckoos have evolved to mimic the distinctive features (i.e. pattern, colour etc.) of the eggs of a few chosen host species. This increases the survivability of the eggs by reducing the probability of the eggs being spotted and abandoned. CS algorithm is a robust algorithm that can be applied to various mathematical models and optimisation problems [6].

The basic rules of the cuckoo search algorithm are defined as follows [4]

- Each cuckoo selects a nest and lays only one egg at a time into the randomly chosen nest.
- The best nests with high quality of eggs will be taken over to the next generation.
- The availability of host nests is fixed and then a host bird identifies the cuckoo egg with the probability of $P_a$ = {0, 1} then in this case the host bird can either throw the egg away or abandon the nest and build a new nest somewhere else.

The new solution $x^{t+1}$ is generated using the Eq.(6),

$$\therefore\ x_i^{t+1} = x_i^t + \alpha \oplus Levy(\lambda) \tag{6}$$

where, $i$ is used to denote the cuckoo, while $\oplus$ mean entry wise multiplication. $\alpha$ is step size used in the problem. Levy function is defined by the Eq.(7)

$$Levy \sim u = t^{-\lambda}, 1 < \lambda \le 3. \tag{7}$$

This distribution with infinite mean and variance is used to provide a random walk for the cuckoo at each consecutive step. The pseudocode of CS algorithm is given in below:

**Algorithm 1** Cuckoo Search (CS) Algorithm

**Begin**

Initialise the population of $n$ host nests $x_i$ where $i$ =1,2,…,$n$

Calculate individual fitness of the nests, $F_i$

**While** (stop condition)

Generate a new solution $x_j$ using Lévy flight for a randomly chosen cuckoo

Determine the corresponding fitness value $F_j$

Select a nest randomly (say $x_k$)

**If** ($F_j > F_k$)

Replace $x_j$ by $x_k$

**End if**

Abandon worst nests by $P_a$($0 < P_a < 1$) and build a new solutions (using Lévy flight)

Rank the solutions and maintain a best solution

**End while**

**End**

## 4. PROPOSED MODEL

Two TPMs with similar models starting from different initial weight vectors synchronise to achieve identical weights when given same initial vectors as input and are trained based upon their output bit. The synchronised weights of the network are used as the secret keys for encryption. This means that neural networks like TPM achieve synchronisation via the process of mutual learning to create an ephemeral channel for key exchange. In the proposed model, a combination of CS Algorithm and TPM is used to generate keys required for encryption. The generation of optimal weights using CS algorithm leads to faster convergence of TPM. The architecture of which is given in Fig.2.

The optimal weights are generated using CS algorithm. The population of host nests $x_i$ is initialised randomly between the range [$-L$, $+L$]. The fitness function is taken as parabolic and is defined as:

$$F(x) = \mathrm{sgn}(x) \cdot L + G(x) \tag{8}$$

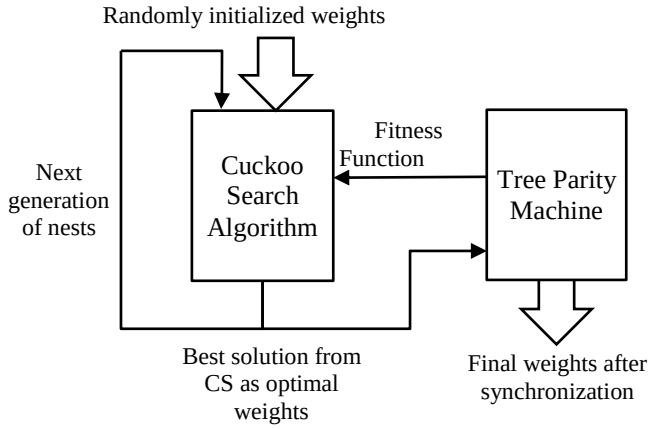



Fig.2. Proposed model for a Cuckoo Search algorithm based Tree Parity Machine

where, $G(x)$ is defined as,

$$G(x) = \begin{cases} 0 & x < -L \\ x^2 & -L \le x < L \\ 0 & x \ge L \end{cases} \tag{9}$$

Based on the fitness function of each nests, worst nests are abandoned while new nests are built via Lévy flight. This cycle is repeated till coincident weights are received after consecutive iterations.

**Algorithm 2** Process of Synchronisation

**Begin**

Initialise weight vector using CS algorithm in range [-*L*, +*L*]

**While** (not synchronised)

Generate a random input vector $x_i$

Compute the values of $h_i$ and $\sigma_i$

Compute the value of the output (i.e. parity) of the network $\tau$

**If** ($\tau_A = \tau_B$)

Apply one of the learning rules to the weights

**End if**

**End while**

**En;d**

One of the following learning rules can be employed for the process of synchronisation:

- When using Hebbian learning rule [7], the networks update weight according to the output bit as shown in Eq.(10):

$$w_{i,j}^{+} = g(w_{i,j} + x_{i,j}\tau\Theta(\sigma_i\tau)\Theta(\tau^A\tau^B)) \quad (10)$$

- When using Anti-Hebbian learning rule [8], the networks are trained opposite to the output

$$w_{i,j}^{+} = g(w_{i,j} - x_{i,j}\tau\Theta(\sigma_i\tau)\Theta(\tau^A\tau^B)) \quad (11)$$

- If the synchronisation is independent of output, one can use random-walk learning rule [9]

$$w_{i,j}^{+} = g(w_{i,j} + x_{i,j}\tau\Theta(\sigma_i\tau)\Theta(\tau^A\tau^B)) \quad (12)$$

The learning rules have to make sure to be in agreement with the original premise of weights being in range [-*L*, +*L*]. This is done by resetting the value of weight outside the range to the nearest boundary value by the function $g(x)$ defined as,

$$g(x) = \begin{cases} \text{sgn}(x)\cdot L & |x| > L \\ x & otherwise \end{cases} \quad (13)$$

This process of updating weights is repeated until synchronisation is finished. Further application of the learning rule is unable to destroy the synchronisation since process of updating uses current weight vector and input which are identical.

## 4.1 GENERATION OF KEY STREAM

Once the networks are synchronised by the proposed algorithm mentioned, the weight vectors of the networks act as the secret key for both parties. To generate a key stream of desired length, a feedback is employed in which the synchronised weights become the new input vector xi to the network as defined by,

$$x_i = \begin{cases} -1, & w_i < 0 \\ +1, & w_i \geq 0 \end{cases} \quad (14)$$

A new set of optimal weights are generated by the process of CS algorithm and the process of mutual learning is repeated to receive a new set of synchronised weights which are appended to the original key stream of larger length. The process is repeated till a key stream of desired length is produced and is ready to be used. The key generated now can be used for encryption by exploiting any stream cipher. The key stream generated can also be used in one-time pad by computing a XOR of it with the plain text to produce the desired cipher text.

## 4.2 SECURITY ATTACKS

The exchange of secret key is considered secure only when an adversary cannot reproduce the secret key when the algorithm of exchange is known as well as the information transferred between the involved parties. It is generally computationally impossible for the adversary to determine the secret key. In the case of tree parity machines the initial weights vectors generated by CS algorithm are kept secret. Only the input vector $x_i$ and the output $\tau$ are interchange over the public network. Keeping the internal network configuration of both parties secret is the basis of the security involved in key-exchange protocol in TPMs. The major attacks on the TPM involves Regular Flipping Attack (RFA) and Majority Flipping Attack (MFA) [10]. In RFA, an adversary spoofs one of the parties during the synchronisation process. In the event of the differing output between adversary and the spoofed party, the adversary changes the sign of one of its hidden unit (i.e. sign is "flipped"). In MFA, success is dependent on the cooperation among a group of attackers [11]. This cooperation is the reason for increase in probability of the attack.

# 5. RESULTS

The above algorithm was implemented using both randomly initialised weights as well as using CS algorithm. Here number of input units, $N$ = 3 and the weight range, $L$ = 4. The number of iterations averaged over 100 observations are calculated while varying the number of hidden layers and shown in Table.1.

Table.1. Number of iterations with varying number of hidden units using random and CS to initialise weights averaged over 100 observations

| Number of hidden neurons (K) | Using Random Weights | Using weights initialised by CS algorithm |
|---|---|---|
| 4 | 134 | 84 |
| 10 | 371 | 276 |
| 20 | 742 | 673 |
| 30 | 1624 | 1212 |

The number of iterations are decreased significantly by using CS algorithm for weight vector initialisation. It has resulted in ~30% decrease in number of iterations on an average.

Table.2. Number of iterations for synchronisation when varying the range of weights using random and CS initialise weights averaged over 100 observations

| Weight Range (*L*) | Using Random Weights | Using weights initialised by CS algorithm |
|---|---|---|
| 4 | 312 | 176 |
| 5 | 742 | 293 |
| 6 | 987 | 322 |
| 7 | 1423 | 404 |

Similarly the effect of CS algorithm was observed by changing the weight range from 4 to 7 with $N$ = 3, $K$ = 100. The results of the same can be observed from Table.2. We can observe that using CS, the synchronisation steps are decreased

tremendously i.e. >50% decrease is observed. In Fig.3 documents the change in synchronisation steps while changing $L$ and $N$ with $K = 100$.

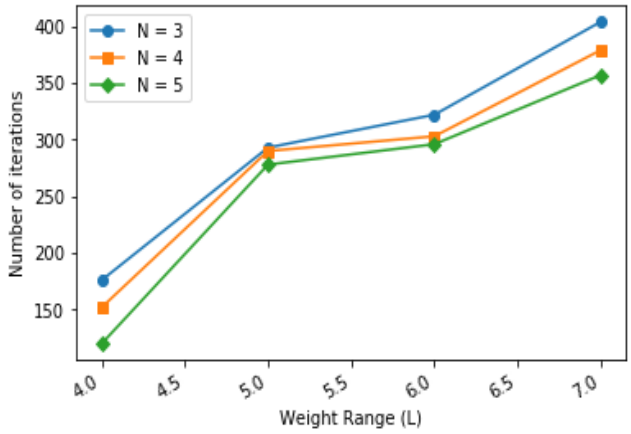
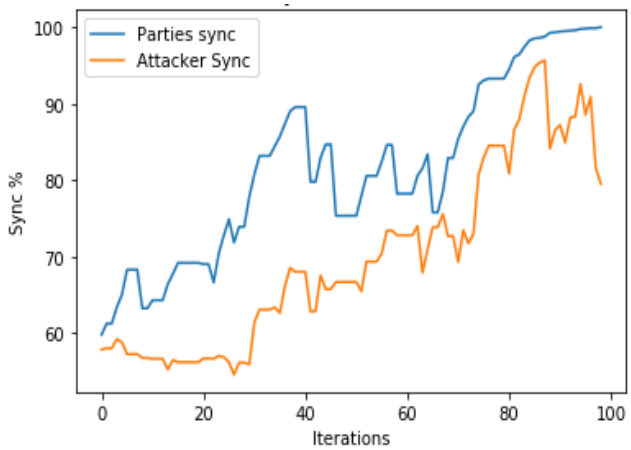
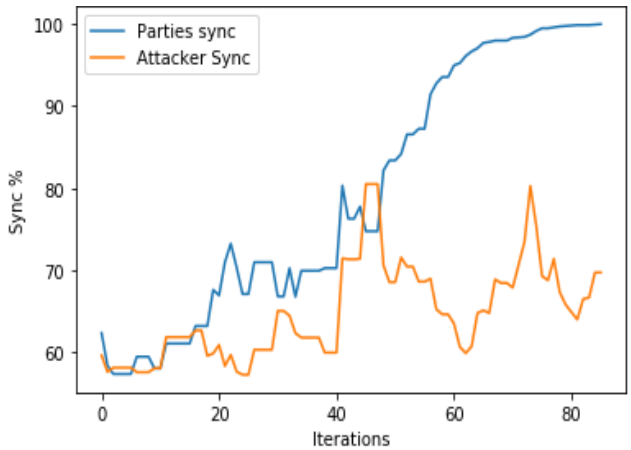



Fig.3. Iterations for synchronisation with varying weight range ($L$) and input ($N$) using CS initialisation




(a)




(b)

Fig.4. Parties and attacker synchronisation with (a) Random weights (b) CS weights

The difference between naive attack from an attacker for random and CS initialisation is compared with $N = 100$, $K = 3$, $L = 3$ and shown in Fig.4. Again, CS weights synchronise faster thereby decreasing the success probability of naive attacks. The success probability of MFA is shown in Fig.5. Comparing both weight initialisations. It can be seen that increasing the weight range significantly decreases the success probability of an attack. In addition to that CS implementation greatly boosts the security of the key exchange protocol.

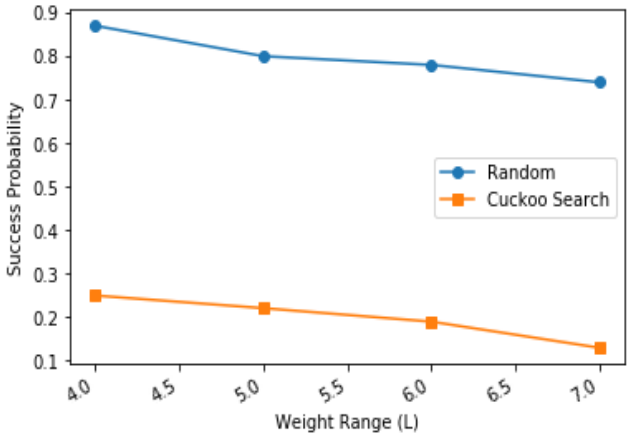



Fig.5. Majority flipping attack (MFA) success probability while varying weight range ($L$) with both initialisations

The success probability of RFA is shown in Fig.6 for varying weight range and input neurons. We can see that apart from increasing weight range, even increasing the input neurons decreases the probability of a successful attack.

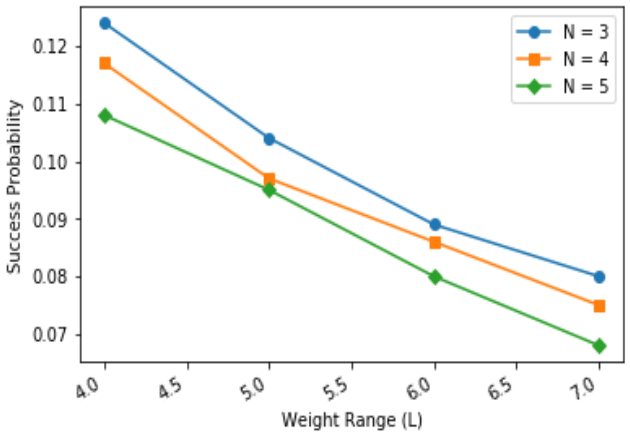



Fig.6. Random Flipping Attack (RFA) for varying weight range ($L$) and input neurons ($N$)

It is clearly observable that CS weights make the implementation converge faster and more secure from security attacks.

## 6. CONCLUSION

Neural networks like TPM with their ability to synchronize with each other through the process of mutual learning open a new field of study in cryptography. The inner configuration and architecture of these networks prove to be fascinating. Incorporating these networks with meta-heuristic techniques like Cuckoo Search Algorithm improves the overall security of these networks. In this paper a novel approach is defined for the process

of synchronization of TPM. The use of CS algorithm proves to provide faster convergence and synchronization of TPMs using mutual learning. There is considerable decrease in number of iteration required for synchronization when weight vector of the network are initialized with CS algorithm improving the viability of these networks in cryptography. Therefore neural cryptography could be the future of secure key exchange and a meta-heuristic approach could prove to be useful. In future the use of mutual learning in neural cryptography leads to be a novel applications for generating key streams.

## REFERENCES


[1] Whitfield Diffie and Martin Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644-654, 1976.

[2] Ido Kanter, Wolfgang Kinzel and Eran Kanter, "Secure Exchange of Information by Synchronization of Neural Networks", *Europhysics Letters*, Vol. 57, No. 1, pp. 141-148, 2002.

[3] Pravin Revankar, W.Z. Gandhare and Dilip Rathod, "Neural Synchronization with Queries", *Proceedings of International Conference on Signal Acquisition and Processing*, pp. 233-239, 2010.

[4] Xin-She Yang and Suash Deb, "Cuckoo Search via Levy flights", *Proceedings of International Conference on Nature and Biologically Inspired Computing*, pp. 331-336, 2009.

[5] Xin-She Yang and Suash Deb, "Engineering Optimization by Cuckoo Search", *International Journal of Mathematical Modelling and Numerical Optimization*, Vol. 1, No. 4, pp. 330-343, 2010.

[6] Natalia Caporale and Yang Dan, "Spike Timing-Dependent Plasticity: A Hebbian Learning Rule", *Annual Review of Neuroscience*, Vol. 31, pp. 25-46, 2008.

[7] Wolfgang Kinzel, "Theory of Interacting Neural Networks", *Proceedings of International Conference on Disordered Systems and Neural Networks*, pp. 311-318, 2002.

[8] Wolfgang Kinzel and Ido Kanter, "Neural Cryptography", *Proceedings of 9th International Conference on Neural Information Processing*, Vol. 3, pp. 688-695, 2002.

[9] Alexander Klimov, Anton Mityagin and Adi Shamir, "Analysis of Neural Cryptography", *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, pp. 23-29, 2002.

[10] Lanir N. Shacham, et al., "Cooperating Attackers in Neural Cryptography", *Physical Review E*, Vol. 69, No. 6, pp. 137-146, 2004.

[11] Jiawei Yuan and Shucheng Yu, "Privacy Preserving Back-Propagation Neural Network Learning made Practical with Cloud Computing", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 1, pp. 212-221, 2014.

[12] Ning Cao et al., "Privacy-Preserving Multi-Keyword Ranked Search over Encrypted Cloud Data", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 1, pp. 222-233, 2014.

[13] R. Tso, X. Huang and W. Susilo, "Strongly Secure Certificate Less Short Signatures", *Journal of Systems and Software*, Vol. 85, No. 6, pp. 1409-1417, 2012.

[14] Ahmed M. Allam, Hazem M. Abbas and M. Watheq El-Kharashi, "Authenticated Key Exchange Protocol using Neural Cryptography with Secret Boundaries", *Proceedings of International Conference on Neural Networks*, pp. 23-34, 2013.

[15] Thuan Thanh Nguyen, Anh Viet Truong and Tuan Anh Phung, "A Novel Method based on Adaptive Cuckoo Search for Optimal Network Reconfiguration and Distributed Generation Allocation in Distribution Network", *International Journal of Electrical Power and Energy Systems*, Vol. 78, pp. 801-815, 2016.

[16] R. Rao, "Review of Applications of TLBO Algorithm and a Tutorial for Beginners to Solve the Unconstrained and Constrained Optimization Problems", *Decision Science Letters*, Vol. 5, No. 1, pp. 1-30, 2016.