IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 1, 2021, accepted January 12, 2021, date of publication January 19, 2021, date of current version January 28, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3052884*

# Artificial Neural Synchronization Using Nature Inspired Whale Optimization

**ARINDAM SARKAR[1], MOHAMMAD ZUBAIR KHAN[2], MOIRANGTHEM MARJIT SINGH[3], (Senior Member, IEEE), ABDULFATTAH NOORWALI[4], (Member, IEEE), CHINMAY CHAKRABORTY[5], AND SUBHENDU KUMAR PANI[6]**

[1]Department of Computer Science and Electronics, Ramakrishna Mission Vidyamandira, Howrah 711202, India
[2]Department of Computer Science, College of Computer Science and Engineering Taibah University, Madinah 41477, Saudi Arabia
[3]Department of Computer Science and Engineering, North Eastern Regional Institute of Science and Technology, Nirjuli 791109, India
[4]Department of Electrical Engineering, Umm Al-Qura University, Makkah 24381, Saudi Arabia
[5]Department of Electronics and Communication Engineering, Birla Institute of Technology at Mesra, Ranchi 814142, India
[6]Department of Computer Science and Engineering, Orissa Engineering College, Biju Patnaik University of Technology (BPUT), Odisha 752050, India

Corresponding authors: Arindam Sarkar (arindam.vb@gmail.com), Mohammad Zubair Khan (mkhanb@taibahu.edu.sa), and Moirangthem Marjit Singh (marjitm@gmail.com).

This work was supported by the Deanship of Scientific Research at Umm Al-Qura University under Grant 19-ENG-1-01-0015.

**ABSTRACT** In this article, a whale optimization-based neural synchronization has been proposed for the development of the key exchange protocol. At the time of exchange of sensitive information, intruders can effortlessly perform sniffing, spoofing, phishing, or Man-In-The-Middle (MITM) attack to tamper the vital information. Information needs to be secretly transmitted with high level of encryption by preserving the authentication, confidentiality, and integrity factors. Such stated requirements urge the researchers to develop a neural network-based fast and robust security protocol. A special neural network structure called Double Layer Tree Parity Machine (DLTPM) is proposed for neural synchronization. Two DLTPMs accept the common input and different weight vectors and update the weights using neural learning rules by exchanging their output. In some steps, it results in complete synchronization, and the weights of the two DLTMs become identical. These identical weights serve as a secret key. There is, however, hardly any research in the field of neural weight vector optimization using a nature-inspired algorithm for faster neural synchronization. In this article, whale optimization-based DLTPM is proposed. For faster synchronization, this proposed DLTPM model uses a whale algorithm optimized weight vector. This proposed DLTPM model is faster and has better security. This proposed technique has been passed through a series of parametric tests. The results have been compared with some recent techniques. The results of the proposed technique have shown effective and has robust potential.

**INDEX TERMS** Neural synchronization, tree parity machine (TPM), session key, neural network, mutual learning, double layer tree parity machine (DLTPM), whale optimization.

## I. INTRODUCTION

A Diffie and Hellman key distribution algorithm [1] is the public-key exchange algorithm. It helps two communication systems to agree on the same encryption key by exchanging a key between them through an insecure medium. In a number of fields such as identification, authentication, data encryption and security, the secret keys are used. Innovative methods for the generation/exchange of cryptographic keys for a secure and low-cost protocol therefore must be required. An attacker E is perhaps unable to deduce the secret while he may be able to follow the structure of the algorithm. Chen *et al.* [2]; Liu *et al.* [3]; Chen *et al.* [4]; Wang *et al.* [5], [6]; Xiao *et al.* [7]; Zhang and Cao [8]; Wang *et al.* [9]; Dong *et al.* [10] described that the neural network synchronization approach offers the chance to solve enormous exchange problems. Rosen-Zvi *et al.* [11]; Lakshmanan *et al.* [12] ; Ni and Paul [13] recently showed that neural cryptography has the capability of achieving key exchange through neural synchronization of Artificial Neural Networks (ANN).

This proposed technique uses an ANN called the Double Layer Tree Parity Machine (DLTPM). By generating common inputs and sharing the outputs of such networks,

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

two DLTPM networks of the same configurations will synchronize and keep the synaptic weight secret between them. Two users of A and B will create a cryptographic key that is difficult for the attacker to infer, even though the attacker is aware of the algorithm structure and the communication channel. Since the method of neural synchronization requires less computational power, neural cryptography is suitable for the rapid exchange of information amongst communicators.

The rest of this article is organized accordingly. Section II deals with related works. Section III deals with the proposed methodology. Section IV, and V deal with security analysis and the results respectively. Conclusions and future scopes are given in section VI and references are given at the end.

## II. RELATED WORKS

Rosen-Zvi *et al.* [11] and Kanter *et al.* [14] described that upon training two ANNs by a specific learning law, the same states of their internal synaptic weight are successfully developed. Kinzel and Kanter [15] and Ruttor *et al.* [16] identified that if the weight of the network increases, the attack probability decreases and the assailant's computational cost grows exponentially as the user effort becomes polynomial. Sarkar and Mandal [17] and Sarkar *et al.* [18]–[21] proposed schemes to enhance the security of the protocol by improving the synaptic depths of TPM and henceforth counteracting the attacks of the brute strength of the attacker. It is found that the amount of security provided by TPM synchronization can also be improved by inserting a large collection of neurons and entries of each neuron into the hidden layers. Allam *et al.* [22] identified a previously shared secret authentication algorithm. As a result, the algorithm obtains a very high degree of security without increasing synchronization time. Ruttor [23] described that lower values of the hidden unit have negative safety implications. Klimov *et al.* [24] calculated whether or not two networks synchronize their weights. A frequency analysis technique was proposed by Dolecki and Kozera [25] that enables two TPM networks to be evaluated with a defined value that is not related to their differences in synaptic weights before the synchronization steps are completed. Santhanalakshmi *et al.* [26]; Dolecki and Kozera [27] evaluated the efficiency of coordinated usage of genetic algorithm and the Gaussian distribution respectively. As a consequence, the substitution of random weights with optimum weights reduces the synchronization period. The timing distribution of the two TPM networks is adapted from the Poisson distribution by Dolecki and Kozera [27]. Pu *et al.* [28] developed an algorithm that blends ''true random sequences'' with a TPM network that demonstrates more complex dynamic behaviors that increase the efficiency of encryption and resistance to attacks. According to their study, the synaptic weight values of the TPM networks are generated in a secure, standardized, and distributed. In the light of rules leading to the creation of TPM synchronization, Mu and Liao [29] and Mu *et al.* [30] describes the heuristic of minimum hamming distances. To test the security level of the final TPM network structure, the hamming distance based heuristic rule was used. Concerning improvements to initial TPM network infrastructure, Gomez *et al.* [31] observed that the synchronization period is reduced from 1.25 ms to less than 0.7 ms with an initial assignment of the weights between 15% to 20%. Niemiec [32] proposed a new concept for the main quantity reconciliation process using TPM networks. Dong and Huang [33] proposed a complex value-based neural network for neural cryptography. Here all the inputs and the outputs are complex value. But, this technique takes a significant amount of time to complete the synchronization process.

The issue of the current key exchange's agreement can be resolved using the proposed technique. This work proposes changes to the TPM framework. The proposed DLTPM approach uses two hidden layers. DLTPMs also take different random weight vectors and share their outputs. For faster synchronization of two DLTPMs, this article proposes optimization of weight vector. This proposed methodology uses a nature-inspired Whale optimization algorithm for weight optimization to reduce the tuning time of two DLTPM networks for generating cryptographic keys for encryption purposes. It results in complete synchronization after certain steps by adjusting weights using traditional rules of learning. In this way, two DLTPMs will generate a cryptographic key that is difficult to guess for the attacker even when the algorithm is known to the attacker. For the study of the proposed method, various parametric tests are performed. Python is used for the implementation of the methodology and R is used for statistical analysis.

## III. PROPOSED METHODOLOGY

### A. OPTIMAL WEIGHT VECTOR GENERATION ALGORITHM

A whale based optimization [1], [34] is used on the weight values of the DLTPM for faster synchronization. As the optimal weights contribute to faster convergence, a greater weight range can be taken into consideration to enhance the security of existing TPM. Inspiration and foraging behaviors of whales are considered in the whale optimization tehchnique. Whales are the world's largest mammals. Their intelligence is due to the presence of spindle cells in brain. They do live in groups, and they can develop their dialect. They have a special hunting mechanism which is called bubble-net feeding method. This foraging behavior is done by creating a special bubble in a spiral shape. Humpback whales do know the best location of their preys and are encircled. They do consider the current best candidate solution is a best-obtained solution and near to the optimal solution. Once the best candidate solution has been assigned, the other agents try to update their positions in close to the best search agent as shown in the following equations 1 and 2.

$$R = \left|Q.S^*\left(itr\right) - S(itr)\right| \tag{1}$$

$$S\left(itr+1\right) = S^*\left(itr\right) - P.R \tag{2}$$

where present iteration is represented by $itr$, coefficient vectors are represented by $P$ and $Q$, position vector of solution and optimal solution is denoted by $S$ and $S^*$ respectively. $P$ and $Q$ are represented as $P = 2x.y.x$ and $Q = 2.y$. Here $x$ and $y$ are random vectors belonging to [0, 1]. $x$'s components are decreasing linearly. Now the helix-shaped movement of the humpback is designed and represented using equation 3.

$$S\,(itr+1) = R'.e^{z.m}.cos\,(2\pi m) + S^*(itr) \tag{3}$$

Here, $R' = |S^*\,(itr) - S(itr)|$ is the difference between the best and current solution, $z$ is fixed value, $m$ is a random number in the interval $[-1, +1]$. In order to update the positions of the whales, equation 4 is used.

$$S(itr+1) = \begin{cases} S^*(itr) - P.R & if\ n < 0.5 \\ R'.e^{z.m}.cos\,(2\pi m) + S^*(itr) & if\ n \geq 0.5 \end{cases} \tag{4}$$

Here, $n$ is an arbitrary number between 0 and 1. The mathematical model of the exploration phase is given as equation 5 and 6.

$$R = |Q.S_{random} - S| \tag{5}$$

$$S\,(itr+1) = S_{random} - P.R \tag{6}$$

Here, the arbitrarily generated position vector is $S_{random}$ which is selected from the current population. The fitness of each whale has been evaluated using the following fitness function given in equation 7.

$$f(S_i^{itr}) = \begin{cases} +\kappa\ or\ -\kappa & if\ S_i^{itr} < -\kappa\ or\ if\ S_i^{itr} \geq \kappa \\ weight^2 & if\ -\kappa \leq S_i^{itr} < \kappa \end{cases} \tag{7}$$

Here, $\kappa$ is the weight range of DLTPM. The population size and the maximum number of iterations considered in this algorithm are 1000 and 100 respectively. The complete process of optimal weight vector generation using whale optimization is illustrated in algorithm 1.

Compared to the various metaheuristics' algorithms, whale optimization has the highest significance in terms of exploitation capability, exploration capacity, ability to get rid of local minima. When evaluated on unimodal functions, whale optimization has a higher exploitation efficiency. It works well on multimodal functions in exploration. Furthermore, checking the optimization of whales on composite functions can be seen as the ideal way to stabilise exploration and exploitation. The optimization of whales has an important exploration potential due to the use of the position updating process of whales. Whales must be arbitrarily shifted around each other in the initial phase of the algorithm. The whales easily update their locations in the next steps and travel down a spiral-shaped path in the direction of the best route that has been discovered so far. Although these two steps are conducted separately and in half iteration each, the optimization of whales prevents local optima and reaches convergence speed across the iterations at the same time.

**Algorithm 1**

**Input:** Set the parameters *PopulationSize*($sz$). *Parameter*($x$). *CoefficientVector*($P$, $Q$). *MaximumIteration*($Max_{itr}$).

**Output:** The best weight vector (whale) for neural synchronization.

*Initialisation* : counter $itr := 0$.

**for** $i = 1$ to $i <= sz$ **do**
  {/*Initial population of weight vector*/}
  Generate an initial population of weight (whale) $S_i^{itr}$ randomly
  Fitness function for each weight (whale) is evaluated using $f(S_i^{itr})$
**end for**
Assign the best search agent $S_i^*(itr)$
**repeat**
  Set $itr = itr + 1$
  **for** $i = 1$ to $i <= sz$ **do**
    Update $x$, $P$, $Q$, $m$, $n$
    **if** $n < 0.5$ **then**
      **if** $|P| < 1$ **then**
        Update the current search agent position $S_i^{itr}$ by $S\,(itr+1) = S^*$ -$P.R$
      **else**
        **if** $|P| \geq 1$ **then**
          Choose an arbitrary search agent $S_r(itr)$
          Update the position of the current search agent $S_i^{itr}$ by $S(itr+1) = S_{random} - P.R$
        **end if**
      **end if**
    **else**
      **if** $n \geq 0.5$ **then**
        Update the position of the currrent search agent $S_i^{itr}$ by $S(itr+1) = R'.e^{z.m}.cos(2\pi m) + S^*(itr)$
      **end if**
      Fitness function for each weight (whale) is evaluated by $f(S_i^{itr})$
    **end if**
  **end for**
  Replace $S^*$ with a better solution (if found)
**until** $itr > Max_{itr}$
Produce the best solution vector $S^*$ as optimal weight for *DLTPM*

It can also be said that whale optimization will improve the speed of convergence during iterations, whereas most optimization algorithms (such as PSO and GSA) do not have operators to dedicate a particular iteration to exploration or exploitation because they use only one format to change the location of search agents. It is fair to assume that whale optimization achieves convergence speed and prevents local optima by iterations at the same time due to the presence of two different phases (exploration and exploitation). In each iteration, all explorations and exploitations are achieved.

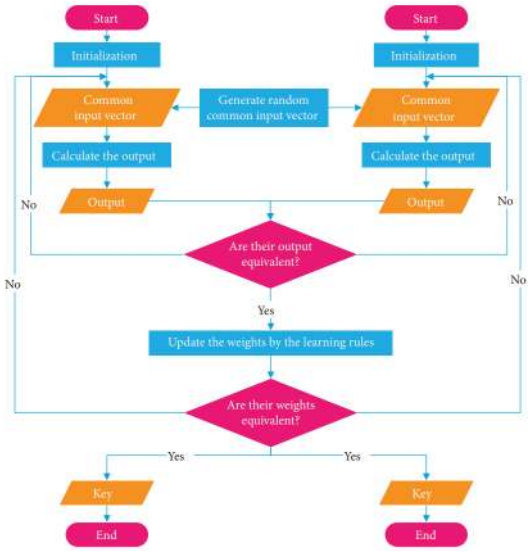


**FIGURE 1.** Flow diagram of the DLTPM synchronization.

### B. DOUBLE LAYER TREE PARITY MACHINES SYNCHRONIZATION

The proposed neural network DLTPM is composed of $M$ no. of input neurons for each $H$ no. of hidden neurons. DLTPM has only one neuron in its output. DLTPM works with binary input, $\alpha_{u,v} \in \{-1, +1\}$. The mapping between input and output is described by the discrete weight value between $-\kappa$ and $+\kappa$, $\beta_{u,v} \in \{-\kappa, -\kappa + 1, \ldots, +\kappa\}$.

Figure 1 displays the flow diagram of the DLTPM synchronization and explains each step of it.

1) *Step of initialization:* Let us suppose Arindam(A) and Marjit(B) initialize the same parameters of their tree parity unit. Then the weights are generated randomly and their weights are initialized.
2) *The phase of estimation:* The identical input vector is obtained by Arindam and Marjit at each learning stage. They calculate the tree parity machines' output and give each other the result.
3) *Step of upgrading:* They then evaluate if the two outputs are identical after they obtain another party's output. Arindam and Marjit will adjust the respective weight vector according to basic learning laws. Otherwise, the estimation process will be done.
4) *Step of evaluation:* Arindam and Marjit must determine if maximum synchronization is achieved if the weights are updating successfully. The protocol goes into the next step if the complete synchronization is achieved. Otherwise, the estimation process will be done.
5) *Finishing process:* Arindam and Marjit are producing their keys by weight. They're finishing the neural key exchange process.

Table 1 lists the variables used in this DLTPM technique.

In DLTPM $u$-th hidden unit is described by the index $u = 1, \ldots, H$ and that of $v = 1, \ldots, M$ denotes input neuron corresponding to the $u$-th hidden neuron of the DLTPM.

**TABLE 1.** Symbol Description of DLTPM.

| Symbol | Description |
|---|---|
| $H$ | Total number of hidden neurons |
| $M$ | Total number of input neurons to each hidden neuron |
| $H1$ | Total number of hidden neurons in first hidden layer |
| $H2$ | Total number of hidden neurons in second hidden layer |
| $\kappa$ | Weight range |
| $\zeta$ | Final output of the DLTPM |
| $Q_u$ | The normalized overlap |
| $\rho_u$ | The normalized overlap |
| $h_u$ | $u$-th hidden unit's local field |
| $pb^u_{a,b}$ | The likelihood distribution of the weight values in $u$-th hidden neuron of a's and b's DLTPM |
| $S$ | Weight's entropy |
| $\alpha_u$ | The input vector of the $u$-th hidden unit |
| $\beta_u$ | The weight vector of the $u$-th hidden unit |
| $\alpha_{u,v}$ | $v$-th element of $\alpha_u$ |
| $\beta_{u,v}$ | $v$-th element of $\beta_u$ |
| $\gamma_u$ | $u$-th hidden unit's output |

Consider there are $H1$, $H2$ numbers of hidden units in the first and second hidden layers respectively. Each hidden unit of the first layer calculates its output by performing the weighted sum over the present state of inputs on that particular hidden unit. Similarly, each hidden unit of the second layer calculates its output by performing the weighted sum over the hidden units of first layer on that particular hidden unit. The calculation for the first hidden layer is given by equation 8.

$$h_u = \frac{1}{\sqrt{M}}\alpha_u.\beta_u = \frac{1}{\sqrt{M}}\sum_{v=1}^{M}\alpha_{u,v}\beta_{u,v} \tag{8}$$

$signum(h_u)$ define the output $\gamma_u$ of the $u$-th hidden unit. (in equation 9),

$$\gamma_u = signum(h_u) \tag{9}$$

If $h_u = 0$ then $\gamma_u$ is set to $-1$ to make the output in binary form. If $h_u > 0$ then $\gamma_u$ is mapped to $+1$, which represents that the hidden unit is active. If $\gamma_u = -1$ then it denotes that the hidden unit is inactive (in equation 10).

$$signum(h_u) = \begin{cases} -1 & if\ h_u \leq 0 \\ +1 & if\ h_u > 0 \end{cases} \tag{10}$$

The product of the hidden neurons of the third layer denotes the ultimate result of DLTPM. This is represented by $\zeta$ is (in equation 11),

$$\zeta = \prod_{u=1}^{H2}\gamma_u \tag{11}$$

The value of $\zeta$ is mapped in the following way (in equation 12),

$$\zeta = \begin{cases} -1 & if\ \gamma_u = -1,\ is\ odd \\ +1 & if\ \gamma_u = -1,\ is\ even \end{cases} \tag{12}$$

$\zeta = \gamma_1$, if only one hidden unit ($H = 1$) is there. $\zeta$ value can be the same for $2^{H-1}$ different ($\gamma_1, \gamma_{2,\ldots,}\gamma_H$) representations.

If the output of two parties disagrees, $\zeta^A \neq \zeta^B$, then no update is allowed on the weights. Otherwise, follow the following rules:

TPM be trained from each other using Hebbian learning rule [15] (in equation 13).

$$\beta^+_{u,v} = fn(\beta_{u,v} + \alpha_{u,v}\zeta\Theta(\gamma_u\zeta)\Theta(\zeta^A\zeta^B)) \tag{13}$$

In the Anti-Hebbian learning rule, both TPM is learned with the reverse of their output [15] (in equation 14).

$$\beta^+_{u,v} = fn(\beta_{u,v} - \alpha_{u,v}\zeta\Theta(\gamma_u\zeta)\Theta(\zeta^A\zeta^B)) \tag{14}$$

If the set value of the output is not imperative for tuning given that it is similar for all participating TPM then random-walk learning rule, is used (in equation 15).

$$\beta^+_{u,v} = fn(\beta_{u,v} + \alpha_{u,v}\Theta(\gamma_u\zeta)\Theta(\zeta^A\zeta^B)) \tag{15}$$

If $X = Y$ then $\Theta(X, Y) = 1$ Otherwise, if $X \neq Y$ then $\Theta(X, Y) = 0$. Only weights are updated which are in hidden units with $\gamma_u = \zeta$. $fn(\beta)$ is used for each learning rule (in equation 16).

$$fn(\beta) = \begin{cases} signum(\beta)\kappa & for\ |\beta| > \kappa \\ \beta & otherwise \end{cases} \tag{16}$$

The likelihood distribution of the weight values in $u$-th hidden neuron of two DLTPM is represented by $(2\kappa + 1)$ (in equation 17).

$$pb^u_{a,b} = P(\beta^A_{u,v} = a \ \wedge \ \beta^B_{u,v} = b) \tag{17}$$

The standard order parameters [35] can be calculated as functions of $pb^u_{a,b}$ shown in equation 18, 19, and 20.

$$\begin{aligned} Q^A_u &= \frac{1}{M}\beta^A_U\beta^A_U \\ &= \sum_{a=-\kappa}^{\kappa}\sum_{b=-\kappa}^{\kappa} a^2\, pb^u_{a,b} \end{aligned} \tag{18}$$

$$\begin{aligned} Q^B_u &= \frac{1}{M}\beta^B_U\beta^B_U \\ &= \sum_{a=-\kappa}^{\kappa}\sum_{b=-\kappa}^{\kappa} b^2\, pb^u_{a,b} \end{aligned} \tag{19}$$

$$\begin{aligned} Q^{AB}_u &= \frac{1}{M}\beta^A_U\beta^B_U \\ &= \sum_{a=-\kappa}^{\kappa}\sum_{b=-\kappa}^{\kappa} ab\, pb^u_{a,b} \end{aligned} \tag{20}$$

Tuning is represented by the normalized overlap [35] given in equation 21.

$$\begin{aligned} \rho^{AB}_u &= \frac{\beta^A_u}{\sqrt{\beta^A_u.\beta^A_u}}\frac{\beta^B_u}{\sqrt{\beta^B_u.\beta^B_u}} \\ &= \frac{R^{AB}_u}{\sqrt{Q^A_u.Q^B_u}} \end{aligned} \tag{21}$$

Calculate the entropy [36] using equation 22.

$$S^{AB}_u = -M\ \sum_{a=-\kappa}^{\kappa}\sum_{b=-\kappa}^{\kappa} pb^u_{a,b} \ln\ pb^u_{a,b} \tag{22}$$

The weight's entropy in a single hidden neuron is represented by equations 23 and 24.

$$S^A_u = -M\ \sum_{a=-\kappa}^{\kappa}\left(\sum_{b=-\kappa}^{\kappa} pb^u_{a,b}\right) ln\left(\sum_{b=-\kappa}^{\kappa} pb^u_{a,b}\right) \tag{23}$$

$$S^B_u = -M\ \sum_{b=-\kappa}^{\kappa}\left(\sum_{a=-\kappa}^{\kappa} pb^u_{a,b}\right) ln\left(\sum_{a=-\kappa}^{\kappa} pb^u_{a,b}\right) \tag{24}$$

Using equations 22, 23, and 24 the common information [36] of A's and B's represented using equation 25.

$$In^{AB} = \sum_{u=1}^{H}\left(S^A_u + S^B_u - S^{AB}_u\right) \tag{25}$$

The likelihood to observe $\gamma_u\alpha_{u,v} = +1$ or $\gamma_u\alpha_{u,v} = -1$ are not equal, but depend on the related weight $\beta_{u,v}$ (in equation 26).

$$P\left(\gamma_u\alpha_{u,v} = 1\right) = \frac{1}{2}\left(1 + erf\left(\frac{\beta_{u,v}}{\sqrt{MQ_u - \beta^2_{u,v}}}\right)\right] \tag{26}$$

$\gamma_u\alpha_{u,v} = signum(\beta_{u,v})$ occurs more frequently than $\gamma_u\alpha_{u,v} = -signum(\beta_{u,v})$, the stationary likelihood distribution of the weights for $t \to \infty$, is computed using equation 19 for the transition likelihood [16]. This is represented using equation 27.

$$P\left(\beta_{u,v} = \beta\right) = p_0\prod_{i=1}^{|\beta|}\frac{1 + erf\left(\frac{i-1}{\sqrt{MQ_u-(i-1)^2}}\right)}{1 - erf\left(\frac{i}{\sqrt{MQ_u-i^2}}\right)} \tag{27}$$

Here the normalization constant $p_0$ is given by equation 28.

$$p_0 = \left(\sum_{\beta=-\kappa}^{\kappa}\prod_{i=1}^{|\beta|}\frac{1 + erf\left(\frac{i-1}{\sqrt{MQ_u-(i-1)^2}}\right)}{1 - erf\left(\frac{i}{\sqrt{MQ_u-i^2}}\right)}\right)^{-1} \tag{28}$$

For $M \to \infty$, the parameter of the error functions will not be considered so that the weights remain consistent as shown in equation 29.

$$\sqrt{Q_u(t=0)} = \sqrt{\frac{\kappa(\kappa+1)}{3}} \tag{29}$$

Otherwise, if $M$ is finite, the likelihood distribution represented using order parameter $Q_u$ will be as shown in equation 30.

$$Q_u = \sum_{\beta=-\kappa}^{\kappa}\beta^2 P\left(\beta_{u,v} = \beta\right) \tag{30}$$

Intensifying it in terms of $M^{-1/2}$ results in equation 31.

$$Q_u = \frac{\kappa(\kappa+1)}{3} + \frac{8\kappa^4 + 16\kappa^3 - 10\kappa^2 - 18\kappa + 9}{15\sqrt{3\pi\kappa(\kappa+1)}}\frac{1}{\sqrt{M}} + O\left(\frac{\kappa^4}{M}\right) \tag{31}$$

In the case of $1 \ll \kappa \ll \sqrt{M}$, the asymptotic performance of the order parameter is represented using equation 32.

$$Q_u \sim \frac{\kappa(\kappa+1)}{3}\left(1+\frac{8}{5\sqrt{3\pi}}\frac{\kappa}{\sqrt{M}}\right) \tag{32}$$

First-order approximation of $Q_u$ is given by equation 33.

$$Q_u = \frac{\kappa(\kappa+1)}{3} - \frac{8\kappa^4+16\kappa^3-10\kappa^2-18\kappa+9}{15\sqrt{3\pi\kappa(\kappa+1)}}\frac{1}{\sqrt{M}} + O\left(\frac{\kappa^4}{M}\right) \tag{33}$$

This systematically converges to equation 34.

$$Q_u \sim \frac{\kappa(\kappa+1)}{3}\left(1-\frac{8}{5\sqrt{3\pi}}\frac{\kappa}{\sqrt{M}}\right) \tag{34}$$

If $\zeta^A = \zeta^B$, but $\gamma_u^A \neq \gamma_u^B$, the weight of one hidden neuron is changed. The weights execute an anisotropic diffusion in case of attractive steps that takes to equation 35.

$$pb_{a,b}^{u+} = \frac{1}{2}\left(pb_{a+1,b+1}^{u} + pb_{a-1,b-1}^{u}\right) \tag{35}$$

Repulsive steps, as an alternative, are equal to normal diffusion steps (in equation 36)

$$pb_{a,b}^{u+} = \frac{1}{4}\left(pb_{a+1,b}^{u} + pb_{a-1,b}^{u} + pb_{a,b+1}^{u} + pb_{a,b-1}^{u}\right) \tag{36}$$

on the same lattice. $\triangle\rho s_{attr}(\rho)$ and $\triangle\rho s_{repu}(\rho)$ are random variables. $\triangle\rho s_{attr}$ and $\triangle\rho s_{repu}$ are the step size for attractive and repulsive respectively as shown in equations 37 and 38.

$$\triangle\rho s_{attr} = \frac{3}{\kappa\,(\kappa+1)}(1-\sum_{v=-\kappa}^{+\kappa}(2v+2)\,pb_{\kappa,v} + pb_{\kappa,\kappa}) \tag{37}$$

$$\triangle\rho s_{repu} = -\frac{3}{\kappa\,(\kappa+1)}\left(\sum_{v=-\kappa}^{+\kappa}\frac{v}{2}\,\left(pb_{\kappa,j} - pb_{-\kappa,v}\right)\right) \tag{38}$$

At the initial state of the synchronization, it has its highest effect (in equation 39),

$$\triangle\rho s_{attr}\,(\rho=0) = \frac{12\kappa}{(\kappa+1)\,(2\kappa+1)^2} \sim \frac{3}{\kappa^2} \tag{39}$$

Weights are uncorrelated as shown in equation 40.

$$pb_{a,b}\,(\rho=0) = \frac{1}{(2\kappa+1)^2} \tag{40}$$

Highest consequence (in equation 41)

$$\triangle\rho s_{repu}\,(\rho=1) = \frac{3}{(\kappa+1)\,(2\kappa+1)} \sim -\frac{3}{2\kappa^2} \tag{41}$$

is achieved for complete harmonized weights (in equation 42).

$$pb_{a,b}(\rho=1) = \begin{cases} (2\kappa+1)^{-1} & for\ a=b \\ 0 & for\ a\neq b \end{cases} \tag{42}$$

The weights are updated if $\zeta^A = \zeta^B$. $\epsilon_u$ is the generalization error. In common overlap for $H$ hidden neurons, $\epsilon_u = \epsilon$, the likelihood is represented using equation 43.

$$Pb_{id} = Pb\left(\zeta^A = \zeta^B\right) = \sum_{u=0}^{H/2}\left(\frac{H}{2u}\right)(1-\epsilon)^{H-2u}\,\epsilon^{2u} \tag{43}$$

For synchronization of DLTPM with $H > 1$, the likelihood of attractive as well as repulsive are represented using by equations 44 and 45.

$$Pb_{attr}^{B} = \frac{1}{2Pb_{id}}\sum_{u=0}^{\left(\frac{H-1}{2}\right)}\left(\frac{H-1}{2u}\right)(1-\epsilon)^{H-2u}\,\epsilon^{2u} \tag{44}$$

$$Pb_{repu}^{B} = \frac{1}{Pb_{id}}\sum_{u=1}^{H/2}\left(\frac{H-1}{2u-1}\right)(1-\epsilon)^{H-2u}\,\epsilon^{2u} \tag{45}$$

For $H = 3$, this leads to equations 46 and 47.

$$Pb_{attr}^{B} = \frac{1}{2}\frac{(1-\epsilon)^3+(1-\epsilon)\epsilon^2}{(1-\epsilon)^3+3(1-\epsilon)\epsilon^2} \tag{46}$$

$$Pb_{repu}^{B} = \frac{2(1-\epsilon)\epsilon^2}{(1-\epsilon)^3+3(1-\epsilon)\epsilon^2} \tag{47}$$

Harmonization time $Tm_{r,s}$ for the two random walk beginning at position $s$ and distance $r$, $Rf_{r,s}$ is the time of the first reflection. $Tm_{r,s}$ is given by equation 48.

$$Tm_{r,s} = Rf_{r,s} + \sum_{v=1}^{r-1} Rf_{v,1} \tag{48}$$

Replacing equation 41 with equation 48 leads to equation 49.

$$\left\langle Tm_{r,s}\right\rangle = \langle m-r+1\rangle\, s - s^2 + \frac{1}{2}(r-1)(2m-r) \tag{49}$$

Tuning time $Tm$ for arbitrarily selected beginning positions of the 2 random walks represented using equations 50 and 51.

$$\begin{aligned}\langle Tm\rangle &= \frac{2}{m^2}\sum_{r=1}^{m-1}\sum_{s=1}^{m-r}\left\langle Tm_{r,s}\right\rangle \\ &= \frac{(m-1)^2}{3} + \frac{(m-1)}{3m}\end{aligned} \tag{50}$$

$$\begin{aligned}\left\langle Tm^2\right\rangle &= \frac{2}{m^2}\sum_{r=1}^{m-1}\sum_{s=1}^{m-r}\left\langle Tm^2{}_{r,s}\right\rangle \\ &= \frac{(17m^5-51m^4+65m^3-45m^2+8m+6)}{90m}\end{aligned} \tag{51}$$

The mean attractive steps needed to achieve a harmonized state increases nearly proportional to $m^2$ (equation 52).

$$\langle Tm\rangle \sim \frac{1}{3m^2} \sim \frac{4}{3\kappa^2} \tag{52}$$

This result is fixed with the scaling performance $\langle tm_{synch}\rangle \propto \kappa^2$ found for TPM harmonization.

The standard deviation of $Tm$ is shown in equation 53.

$$SD_{Tm} = \sqrt{\frac{(7m^6-11m^5-15m^4+55m^3-72m^2+46m-10)}{90m^2}} \tag{53}$$

Here, system size $m = 2\kappa + 1$. $SD_{Tm}$ is proportional asymptotically to $\langle Tm \rangle$ (equation 54).

$$SD_{Tm} \sim \sqrt{\frac{7}{10}} \langle Tm \rangle \tag{54}$$

## IV. SECURITY ANALYSIS

### A. GEOMETRIC ATTACK

In this proposed technique a geometric attack is considered on DLTPM. The likelihood of $\gamma_u^E \neq \gamma_u^A$ is represented using the prediction error [37] using equation 55,

$$\varepsilon_u^p = \frac{1}{2}\left[1 - erf\left(\frac{\rho_u}{\sqrt{2(1-\rho_u^2}}\frac{|h_u|}{\sqrt{Q_u}}\right)\right] \tag{55}$$

of the perceptron. When the $u$-th hidden neuron does not agree and rest of all hidden units has a condition of $\gamma_v^E \neq \gamma_v^A$, then the likelihood of a modification using the geometric attack is shown using equation 56.

$$P_k^+ = \int_0^\infty \left(\frac{2}{\sqrt{2\pi Q}}\right)^H \left(\int_{h_u}^\infty \frac{1-\epsilon^P(h)}{1-\epsilon} e^{-\frac{h^2}{2Q}} dh\right)^{H-k} \times \left(\int_{h_u}^\infty \frac{\epsilon^P(h)}{\epsilon} e^{-\frac{h^2}{2Q}} dh\right)^{k-1} \frac{\epsilon^P(h_u)}{\epsilon} e^{-\frac{h_u^2}{2Q}} dh_u \tag{56}$$

For identical order parameters $Q = Q_v^E$ and $R = R_j^{AE}$ and various outputs $\gamma_v^A \neq \gamma_v^E$. Then the likelihood for a modification of $\gamma_u^E \neq \gamma_u^A$ is shown using equation 57.

$$P_k^+ = \int_0^\infty \left(\frac{2}{\sqrt{2\pi Q}}\right)^H \left(\int_{h_u}^\infty \frac{1-\epsilon^P(h)}{1-\epsilon} e^{-\frac{h^2}{2Q}} dh\right)^{H-k} \times \left(\int_{h_u}^\infty \frac{\epsilon^P(h)}{\epsilon} e^{-\frac{h^2}{2Q}} dh\right)^{k-1} \frac{\epsilon^P(h_u)}{\epsilon} e^{-\frac{h_u^2}{2Q}} dh_u \tag{57}$$

Using the same equation the likelihood for an incorrect modification of $\gamma_u^E = \gamma_u^A$ is shown with the help of equation 58.

$$P_k^- = \int_0^\infty \left(\frac{2}{\sqrt{2\pi Q}}\right)^K \left(\int_{h_i}^\infty \frac{1-\epsilon^P(h)}{1-\epsilon} e^{-\frac{h^2}{2Q}} dh\right)^{K-k-1`} \times \left(\int_{h_i}^\infty \frac{\epsilon^P(h)}{\epsilon} e^{-\frac{h^2}{2Q}} dh\right)^{k} \frac{1-\epsilon^P(h_u)}{1-\epsilon} e^{-\frac{h_u^2}{2Q}} dh_u \tag{58}$$

$\gamma_u^E \neq \gamma_u^A$ condition is satisfied and in total there is an even number of hidden units that satisfied this condition then no geometric modification is done. Equation 59 represents this.

$$P_{r,1}^E = \sum_{u=1}^{H/2} \left(\frac{H-1}{2u-1}\right) (1-\epsilon)^{H-2u} \epsilon^{2u} \tag{59}$$

Second part of $P_r^E$ can be represented using equation 60.

$$P_{r,2}^E = \sum_{u=1}^{H/2} \left(\frac{H-1}{2u-1}\right) P_{2u-1}^- (1-\epsilon)^{H-2u+1} \epsilon^{2u-1} \tag{60}$$

Third part of $P_r^E$ can be represented using equation 61.

$$P_{r,3}^E = \sum_{u=1}^{(H-1)/2} \left(\frac{H-1}{2u}\right) \left(1 - P_{2u+1}^+\right) (1-\epsilon)^{-2u-1} \epsilon^{2u+1} \tag{61}$$

If $H > 1$ then the probability value of attractive steps and repulsive steps in the $u$-th hidden unit are represented using equations 62 and 63.

$$P_a^E = \frac{1}{2}\left(1 - \sum_{v=1}^{3} P_{r,v}^E\right) \tag{62}$$

$$P_r^E = \sum_{v=1}^{3} P_{r,v}^E \tag{63}$$

Attractive steps are performed when $H = 1$. If $H = 3$ probability value can be calculated using equation 56, that forms equation 64 and 65.

$$P_a^E = \frac{1}{2}\left(1 + 2P_g\right)(1-\epsilon)^2 \epsilon + \frac{1}{2}(1-\epsilon)^3 + \frac{1}{2}(1-\epsilon)^2 + \frac{1}{6}\epsilon^3 \tag{64}$$

$$P_r^E = 2\left(1 - P_g\right)(1-\epsilon)^2 \epsilon + 2(1-\epsilon)\epsilon^2 + \frac{2}{3}\epsilon^3 \tag{65}$$

### B. SECRET KEY SPACE ANALYSIS

Consider $n$ number of cascading encryption/decryption technique is used to encrypt/decrypt the plaintext with the help of neural synchronized session key. Then a session key of length [(number of cascaded encryption technique in bits) + (three bits combinations of encryption/ decryption technique index) + (length of $n$ number of encryption/decryption keys in bits) + (length of $n$ number of session keys in bits)] i.e. [ 8 + (3 × n) + (128 × n) + (128 × n) ] bits to [ 8 + (3 × n) + (256 × n) + (256 × n)] number of bits. So,

$$\frac{[\,8 + (3\times n) + (128\times n) + (128\times n)]}{8} = \left[1 + \frac{(3\times n)}{8} + 16n + 16n\right] = 32n \text{ to } \frac{[\,8 + (3\times n) + (256\times n) + (256\times n)]}{8} = \left[1 + \frac{(3\times n)}{8} + 32n + 32n\right]$$

$= 64n$ numbers of characters.

Therefore, the total number of keys $= 256^{64n}$. Attacker checks with half of the possible keys on an average, the time needed at 1 decryption/ $\mu s = 0.5 \times 256^{64n}$ $\mu s = 0.5 \times 2^{8\times 64n}$ $\mu s = 0.5 \times 2^{512n}$ $\mu s = 2^{(512n-1)}$ $\mu s$.

Consider any single encryption using the neural key of size 512 bits which is hypothetically approved and needed to be analyzed in the context of the time taken to crack a ciphertext with the help of the fastest supercomputers available at present. In this neural technique to crack a ciphertext, the number of permutation combinations on the neural key is $2^{512} = 1.340780 \times 10^{154}$ trials for a size of 512 bits only. IBM Summit at Oak Ridge, U.S. invented the fastest supercomputer in the world with 148.6 PFLOPS i.e. means $148.6 \times 10^{15}$ floating-point computing/second. Certainly, it can be considered that each trial may require 1, 000 FLOPS to undergo its operations. Hence, the total test

**TABLE 2. Time Taken for Brute Force Attack.**

| n | Key Space | Brute Force attack time using IBM Summit Supercomputer at Oak Ridge with a speed of 148.6 PFLOPS |
|---|---|---|
| 1 | $256^{64\times1} = 2^{512}$ | $2.86109\times10^{132}$ years |
| 2 | $256^{64\times2} = 2^{1024}$ | $3.83610\times10^{286}$ years |
| 3 | $256^{64\times3} = 2^{1536}$ | $5.14337\times10^{440}$ years |
| 4 | $256^{64\times4} = 2^{2048}$ | $6.89613\times10^{594}$ years |
| 5 | $256^{64\times5} = 2^{2560}$ | $9.24620\times10^{748}$ years |
| 6 | $256^{64\times6} = 2^{3072}$ | $1.23971\times10^{903}$ years |
| 7 | $256^{64\times7} = 2^{3584}$ | $1.66218\times10^{1057}$ years |
| 8 | $256^{64\times8} = 2^{4096}$ | $2.22862\times10^{1211}$ years |

**TABLE 3. Comparison of p_Value Between Proposed DLTPM and the Existing CVTPM.**

| NIST Test | p_Value of Proposed DLTPM | p_Value of CVTPM [33] |
|---|---|---|
| Frequency | 0.572896 | 0.512374 |
| Frequency within a Block | 0.560235 | 0.538721 |
| Runs | 0.526750 | 0.489451 |
| Longest Run of Ones in a Block | 0.096128 | 0.042081 |
| Binary Matrix Rank | 0.651278 | 0.417393 |
| Discrete Fourier Transform | 0.405736 | 0.392762 |
| Non-overlapping Template Matching | 0.451278 | 0.431759 |
| Overlapping (Periodic) Template Matching | 0.201957 | 0.170426 |
| Maurer's "Universal Statistical" | 0.765192 | 0.689478 |
| Linear Complexity | 0.634265 | 0.594917 |
| Serial | 0.504568 | 0.478902 |
| Approximate Entropy | 0.263472 | 0.204175 |
| Cummulative Sums | 0.679248 | 0.587290 |
| Random Excursions | 0.319684 | 0.299875 |
| Random Excursions Variants | 0.213586 | 0.180542 |

needed per second is $148.6 \times 10^{12}$. No. of seconds in a year $= 365 \times 24 \times 60 \times 60 = 31,536,000$ sec. Total number of years for Brute Force attack: $(1.340780 \times 10^{154})/(148.6 \times 10^{12} \times 31,536,000) = 2.86109 \times 10^{132}$ years. Table 2 shows how much time is involved for the brute force attack using IBM Summit Supercomputer at Oak Ridge with a speed of 148.6 PFLOPS.

## V. RESULTS AND ANALYSIS

For results and simulation purposes, the Intel Core i7 10th Generation processor, 2.6 GHz, 16 GB RAM is used. In the proposed transmission technique, true randomness is assured by qualifying the fifteen tests found in the NIST statistical tests [38] suite. For such a proposed approach with elevated robustness, these tests are very useful. The acceptance or rejection of the input vector is determined by a probability value (p-Value). Table 3 contains the results of NIST statistical tests [38] on the generated random input vector. The p-Value comparison between the proposed and the existing CVTPM(Complex Valued TPM) system is also present in this table [33]. From the table, it was shown that the p-Value of the proposed technique surpassed CVTPM in the NIST statistical test.

The result of the frequency test indicates a ratio of 0 and 1 in the generated random sequence. Here, the value of the frequency test is 0.572896 which is quite average [39] and better than the result of frequency test 0.1329 in [40], and 0.632558 in [41] 0.629806 in [42]. A comparison of p_Value of NIST frequency test is given in Table 4.

**TABLE 4. Comparison of p_Value of NIST Frequency Test.**

| Technique | p_Value of NIST Frequency Test |
|---|---|
| Proposed DLTPM | 0.572896 |
| CVTPM [33] | 0.512374 |
| Karakaya et al. [40] | 0.1329 |
| Patidar et al. [41] | 0.632558 |
| Liu at al. [42] | 0.629806 |

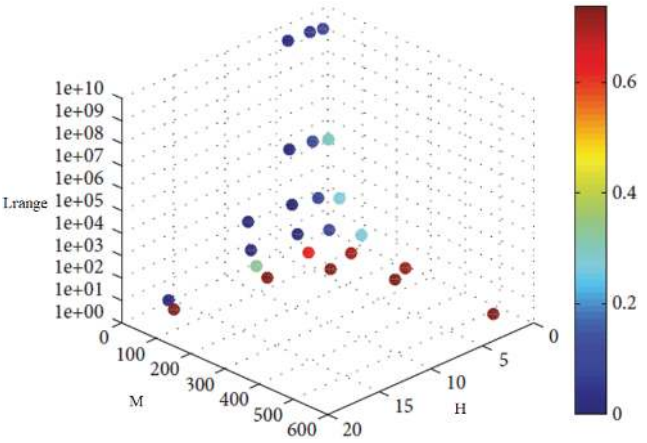


**FIGURE 2. Shows how the probabilities differ with $H1 - H2 - M - \kappa$ values.**

The results of different $H1 - H2 - M - \kappa$ simulations are shown in Table 5. The number of minimum and maximum synchronization steps shows several minimum and maximum steps to synchronize the weights of the two networks. The column for the average steps represents the sum of all the simulations performed. The minimum and maximum time synchronization columns shall signify the minimum and maximum time in seconds needed to synchronize the weights of the two networks. Effective synchronization of the attacker column E reveals how many times the attacking network has been imitating the actions of the other two networks for complete simulations. Finally, the percentage of effective synchronization of the attacker column E (%) successfully indicates the percentage of total simulations shown in the previous column.

As shown in Table 5, the combinations (8-8-16-8) and (8-8-8-128), respectively, are the best results for defense against the attacking network. E did not mimic the actions of A and B's DLTPM in any of the 500,000 simulations. The best combination of values for the trio was determined from Table 6 (8-8-16-8).

This is evaluated from Table 5 that the success probability of the attacking network depends on the $H1 - H2 - M - \kappa$ combination. With all possible combinations, their respective probabilities have been established. Using the Octave GNU scatter3 function, a scatter diagram was drawn to do this.

**TABLE 5. Results After 500,000 Simulations With Different Values of $H1-H2-M-\kappa$.**

| $H1-H2-M-\kappa$ | No. of Min Sync. Steps | No. of Max Sync. Steps | Min Sync. time (sec.) | Max Sync. time (sec.) | Successful Sync. of Attacker E | % of Successful Sync.of Attacker E (%) |
|---|---|---|---|---|---|---|
| (1-1-1-512-1) | 8 | 34 | 0.0729 | 3.4398 | 358976 | 71.795 |
| (2-2-2-256-1) | 10 | 30 | 0.0698 | 2.6739 | 356757 | 71.351 |
| (4-4-4-128-1) | 10 | 39 | 0.0810 | 3.1908 | 358768 | 71.753 |
| (8-8-8-64-1) | 8 | 26 | 0.0749 | 2.8183 | 357862 | 71.572 |
| (1-1-1-256-2) | 9 | 45 | 0.0702 | 2.9652 | 358965 | 71.793 |
| (2-2-2-128-2) | 10 | 35 | 0.0686 | 621.8954 | 338978 | 67.795 |
| (4-4-4-64-2) | 10 | 234 | 0.0719 | 108.7835 | 310878 | 62.175 |
| (8-8-8-32-2) | 11 | 973 | 0.0628 | 32.7614 | 117894 | 23.578 |
| (1-1-1-128-8) | 14 | 87 | 0.0643 | 1.9856 | 119089 | 23.817 |
| (2-2-2-64-8) | 18 | 510 | 0.0698 | 3.3908 | 39796 | 7.959 |
| (4-4-4-32-8) | 25 | 820 | 0.0520 | 6.2578 | 208 | 0.0416 |
| **(8-8-8-16-8)** | **41** | 749 | **0.0346** | **0.0711** | **0** | **0.0** |
| (1-1-1-64-128) | 10 | 118 | 0.0268 | 0.7658 | 138678 | 27.735 |
| (2-2-2-32-128) | 17 | 428 | 0.0513 | 3.3679 | 35726 | 7.1452 |
| (4-4-4-16-128) | 26 | 930 | 0.0386 | 3.8674 | 593 | 0.1186 |
| **(8-8-8-8-128)** | **39** | **1078** | **0.0517** | **5.1786** | **0** | **0.0** |

**TABLE 6. Results After 1000,000 Simulations for the (8-8-16-8) and (8-8-8-128) Combinations.**

| $H1-H2-M-\kappa$ | No. of Min Sync. Steps | No. of Max Sync. Steps | Min Sync. time (sec.) | Max Sync. time (sec.) | Successful Sync. of Attacker E | % of Successful Sync.of Attacker E (%) |
|---|---|---|---|---|---|---|
| (8-8-16-8) | 42 | 820 | 0.0825 | 2.5840 | 0 | 0 |
| (8-8-8-128) | 32 | 897 | 0.0395 | 0.9734 | 2 | 0.0002 |

The proposed approach starts with values like $H1-H2-M-\kappa$ and the probability of the attacker's success in any combination. By using a different combination of $H1-H2-M-\kappa$ values, we then draw a 3D dispersion diagram. The scale of each point plotted here is 16. The probabilities describe the likelihood value of each color dot (see Figure 2). To enhance visualization, the Z-axis ( $\kappa$ ) scale to a logarithmic one. The *xlabel* , *ylabel* and *zlabel* are assigned to their corresponding values $H1-H2-M-\kappa$ respectively. Figure 2 shows how the probabilities differ with a different combination of $H1-H2-M-\kappa$ values. Each point

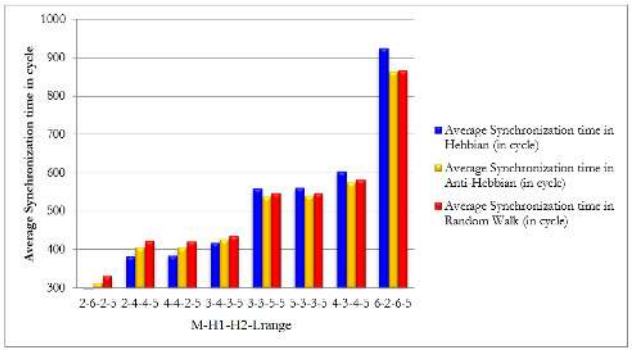


**FIGURE 3. 128-bit session key generation with variable DLTPM architecture and fixed synaptic depth.**

color displays the attacking network's probability of success. Good is a small likelihood (Blue), while bad is a high probability (Red). The value of hidden neuron influences probability greatly, shown in figure 2. A high value of M and a very small value of ( $\kappa$ ) has an adverse impact. Therefore, a fairly low value of M is suggested for proposing values of $H1-H2-M-\kappa$, but M value must be higher than hidden neuron value. ( $\kappa$ ) value shouldn't be too low. It means that a successful passive assault is quite unlikely.

Figure 3 shows that several DLTPM configurations (in terms of different neurons in different layers) can be used to generate a 128-bit session key with a fixed weight range $(\kappa) = 5$. Weight value acts as a key to the session. Among the three learning rules, Hebbian rules outperform two other rules when the network size is small.

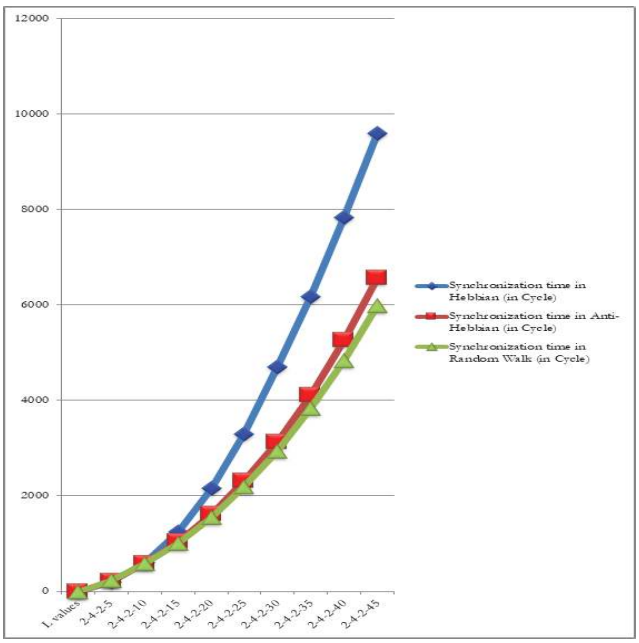


**FIGURE 4. Generation of 256-bit session key using different weight Range to and fixed number of neurons in input and hidden layer.**

In the following Figure 4, the graph shows a trend towards increase in the synchronization steps as the range for weight range $\kappa$ increases for a fixed number of neurons in input and

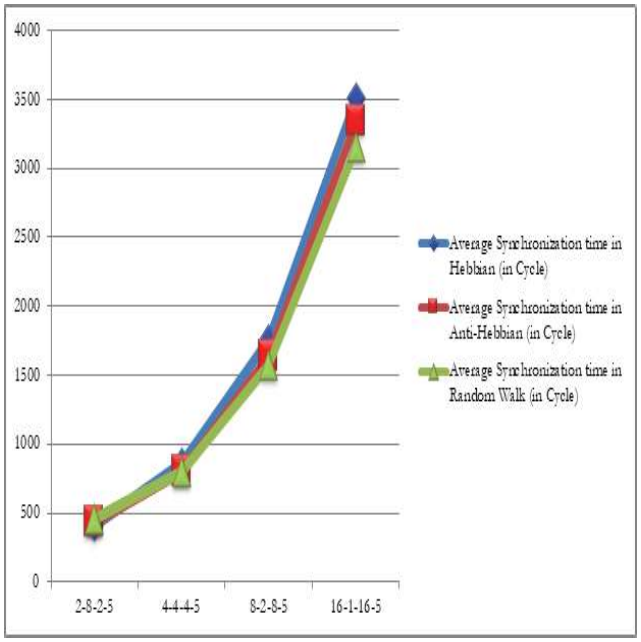


**FIGURE 5.** Generation of 512-bit session key using fixed $\kappa = 5$ and different number of neurons in input and hidden layer.

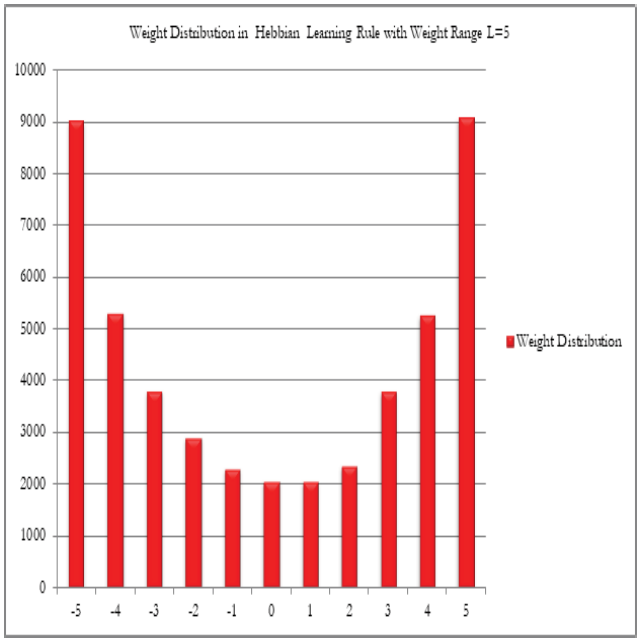


**FIGURE 6.** Weight distribution in Hebbian learning rule with weight range $\kappa = 5$.

hidden layers in all three learning rules for generation of a session key of 256-bit. For small $\kappa$ values Hebbian takes less synchronization steps than other two learning rules in the range of 2-4-2-5 to 2-4-2-15 but as the $\kappa$ value increases Hebbian rule takes more steps to synchronize than other two learning rules. Here, Anti-Hebbian rules take less time than Hebbian and Random Walk learning rules in the range of 2-4-2-20 to 2-4-2-30. Random Walk outperforms from

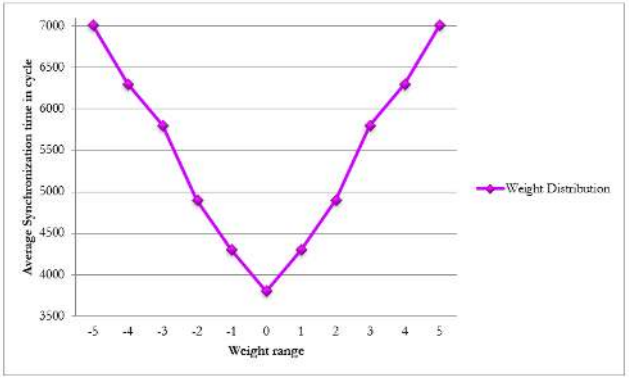


**FIGURE 7.** Weight distribution in random walk learning rule with weight range $\kappa = 5$.

**TABLE 7.** Comparison of Synchronization Time for Fixed Network Size and Variable Learning Rules and Synaptic Depth in Proposed DLTPM and Existing CVTPM Method.

| $\kappa$ Value | Sync. time in Hebbian Learning Rule (in Cycle) in Proposed DLTPM | Sync. time in Anti-Hebbian Learning Rule (in Cycle) in Proposed DLTPM | Sync. time in Random Walk Learning Rule (in Cycle) in Proposed DLTPM | Sync. time in Hebbian Learning Rule (in Cycle) in Existing CVTPM [33] | Sync. time in Hebbian Learning Rule (in Cycle) in Existing CVTPM [33] | Sync. time in Random Walk Learning Rule (in Cycle) in Existing CVTPM [33] |
|---|---|---|---|---|---|---|
| 5 | 287,81 | 309,23 | 320,52 | 348,16 | 389,26 | 415,63 |
| 10 | 664,59 | 682,25 | 675.47 | 813,64 | 753,09 | 713,94 |
| 15 | 1356,06 | 1136,92 | 1104,92 | 1769,57 | 1198,47 | 1104,81 |
| 20 | 2271,32 | 1703,26 | 1648,13 | 2971,04 | 1857,24 | 1696,35 |
| 25 | 3407,18 | 2411,15 | 2302,83 | 4186,59 | 2671,17 | 2540,62 |
| 30 | 4836,97 | 3220,86 | 3074,10 | 5982,41 | 3582,75 | 3318,49 |
| 35 | 6319,72 | 4192,53 | 3959,65 | 7310,37 | 4793,62 | 4518,27 |
| 40 | 7987,63 | 5341,28 | 4978,72 | 8953,50 | 5981,43 | 5372,11 |
| 45 | 9753,02 | 6662,07 | 6147,49 | 11207,26 | 8138,39 | 7363,29 |
| 50 | 12980,17 | 8137,47 | 7423,34 | 14952,38 | 9452,57 | 8720,93 |

2-4-2-35 and beyond that. The most vital findings is that if the synaptic depth i.e. weight range $\kappa$ 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. So, increasing the $\kappa$ value security of the system can be increased.

Figure 5 shows the synchronization time needed for different DLTPM configurations with a fixed weight range $\kappa = 5$. It shows the comparisons of synchronization time to generate the 512-bit session key using fixed $\kappa = 5$ and different number of neurons in input and hidden layers. Among the three learning rules, here also it is observed that Random Walk rules outperform the other two rules for large size of network.

Figure 6 shows the weight distribution in Hebbian learning rule with weight range $\kappa = 5$.

In Random Walk, the weights are well distributed as the Hebbian and Anti-Hebbian rules shown in Figure 7. So, a network with a size greater than 256, Random Walk makes synchronization faster, but for this range Hebbian and Anti-Hebbian take a lot more synchronization steps.

Table 7 provides a comparison of synchronization time for fixed network size and variable learning rules and $\kappa$ in the proposed DLTPM and the existing CVTPM methods. Table 7 shows a trend towards an increase in synchronization steps as the range of weights $\kappa$ rises in all three learning rules. For small $\kappa$ values, Hebbian takes fewer synchronization steps than the other two learning rules, but as the $\kappa$ value increases then more steps are taken to synchronize than the others. Here, the Anti-Hebbian requires less time than the other two learning rules for mid $\kappa$ values. Random Walk outperforms for high $\kappa$ values.

## VI. CONCLUSION AND FUTURE SCOPE

For the cryptographic public-key exchange protocol, this article proposes the synchronization of Double Hidden Layer Neural Networks using nature-inspired whale optimization. For the generation of different key lengths using the neural synchronization process, various combinations of $H1$, $H2$, $M$ and $\kappa$ with variable network sizes are considered. The article analyses the optimization of the weight vector of two DLTPMs using whale optimization algorithms for faster synchronization. DLTPM's security and synchronization time is also examined. It has been shown that geometric attacks have a lower rate of success. It has been found that DLTPM security is higher than the current TPM with the same set of parameters. Finally, in order to verify the experimental findings, a variety of outcomes and evaluations are carried out. A more detailed risk evaluation is expected for future studies to be carried out. In addition, for the optimization of weights for faster neural synchronization purposes, various nature-inspired optimization algorithms will be considered.

## REFERENCES

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976, doi: 10.1109/tit.1976.1055638.

[2] H. Chen, P. Shi, and C.-C. Lim, "Cluster synchronization for neutral stochastic delay networks via intermittent adaptive control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3246–3259, Nov. 2019, doi: 10.1109/tnnls.2018.2890269.

[3] P. Liu, Z. Zeng, and J. Wang, "Global synchronization of coupled fractional-order recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 8, pp. 2358–2368, Aug. 2019, doi: 10.1109/TNNLS.2018.2884620.

[4] H. Chen, P. Shi, and C.-C. Lim, "Exponential synchronization for Markovian stochastic coupled neural networks of neutral-type via adaptive feedback control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1618–1632, Jul. 2017, doi: 10.1109/TNNLS.2016.2546962.

[5] J.-L. Wang, Z. Qin, H.-N. Wu, and T. Huang, "Passivity and synchronization of coupled uncertain reaction–diffusion neural networks with multiple time delays," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 8, pp. 2434–2448, Aug. 2019, doi: 10.1109/TNNLS.2018.2884954.

[6] J. Wang, L.-M. Cheng, and T. Su, "Multivariate cryptography based on clipped hopfield neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 2, pp. 353–363, Feb. 2018, doi: 10.1109/tnnls.2016.2626466.

[7] Q. Xiao, T. Huang, and Z. Zeng, "Global exponential stability and synchronization for discrete-time inertial neural networks with time delays: A timescale approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 6, pp. 1854–1866, Jun. 2019, doi: 10.1109/TNNLS.2018.2874982.

[8] Z. Zhang and J. Cao, "Novel finite-time synchronization criteria for inertial neural networks with time delays via integral inequality method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1476–1485, May 2019, doi: 10.1109/TNNLS.2018.2868800.

[9] A. Wang, T. Dong, and X. Liao, "Event-triggered synchronization strategy for complex dynamical networks with the Markovian switching topologies," *Neural Netw.*, vol. 74, pp. 52–57, Feb. 2016.

[10] T. Dong, A. Wang, H. Zhu, and X. Liao, "Event-triggered synchronization for reaction–diffusion complex networks via random sampling," *Phys. A, Stat. Mech. Appl.*, vol. 495, pp. 454–462, Apr. 2018, doi: 10.1016/j.physa.2017.12.008.

[11] M. Rosen-Zvi, I. Kanter, and W. Kinzel, "Cryptography based on neural networks analytical results," *J. Phys. A, Math. Gen.*, vol. 35, no. 47, pp. L707–L713, Nov. 2002, doi: 10.1088/0305-4470/35/47/104.

[12] S. Lakshmanan, M. Prakash, C. P. Lim, R. Rakkiyappan, P. Balasubramaniam, and S. Nahavandi, "Synchronization of an inertial neural network with time-varying delays and its application to secure communication," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 195–207, Jan. 2018, doi: 10.1109/tnnls.2016.2619345.

[13] Z. Ni and S. Paul, "A multistage game in smart grid security: A reinforcement learning solution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2684–2695, Sep. 2019, doi: 10.1109/tnnls.2018.2885530.

[14] I. Kanter, W. Kinzel, and E. Kanter, "Secure exchange of information by synchronization of neural networks," *Europhys. Lett. (EPL)*, vol. 57, no. 1, pp. 141–147, Jan. 2002, doi: 10.1209/epl/i2002-00552-9.

[15] W. Kinzel and I. Kanter, "Interacting neural networks and cryptography," in *Advances in Solid State Physics*, vol. 42, K. B., Ed. Berlin, Germany: Springer, 2002, pp. 383–391, doi: 10.1007/3-540-45618-X_30.

[16] A. Ruttor, W. Kinzel, R. Naeh, and I. Kanter, "Genetic attack on neural cryptography," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 73, no. 3, Mar. 2006, Art. no. 036121, doi: 10.1103/physreve.73.036121.

[17] A. Sarkar and J. K. Mandal, *Artificial Neural Network Guided Secured Communication Techniques: A Practical Approach*. Sunnyvale, CA, USA: LAP LAMBERT Academic Publishing Germany, 2012.

[18] A. Sarkar, J. Dey, M. Chatterjee, A. Bhowmik, and S. Karforma, "Neural soft computing based secured transmission of intraoral gingivitis image in e-health care," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 14, no. 1, p. 178, Apr. 2019, doi: 10.11591/ijeecs.v14.i1.pp178-184.

[19] A. Sarkar, J. Dey, and A. Bhowmik, "Multilayer neural network synchronized secured session key based encryption in wireless communication," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 14, no. 1, p. 169, Apr. 2019, doi: 10.11591/ijeecs.v14.i1.pp169-177.

[20] A. Sarkar and J. K. Mandal, "Key generation and certification using multilayer perceptron in wireless communication (KGCMLP)," *Int. J. Secur., Privacy Trust Manage.*, vol. 1, no. 5, pp. 27–43, 2012.

[21] A. Sarkar, J. Dey, A. Bhowmik, J. K. Mandal, and S. Karforma, "Computational intelligence based neural session key generation on E-health system for ischemic heart disease information sharing," in *Contemporary Advances in Innovative and Applicable Information Technology* (Advances in Intelligent Systems and Computing), vol. 812, M. J., S. D., and B. J., Eds. Singapore: Springer, 2019.

[22] A. M. Allam, H. M. Abbas, and M. W. El-Kharashi, "Authenticated key exchange protocol using neural cryptography with secret boundaries," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–8.

[23] A. Ruttor, "Neural synchronization and cryptography," 2007, *arXiv:0711.2411*. [Online]. Available: http://arxiv.org/abs/0711.2411

[24] A. Klimov, A. Mityagin, and A. Shamir, "Analysis of neural cryptography," *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2002, pp. 288–298.

[25] M. Dolecki and R. Kozera, "Threshold method of detecting long-time TPM synchronization," in *Computer Information Systems and Industrial Management*, vol. 8104. Kraków, Poland: Springer, 2013, pp. 241–252.

[26] S. Santhanalakshmi, K. Sangeeta, and G. K. Patra, "Analysis of neural synchronization using genetic approach for secure key generation," in *Security in Computing and Communications. SSCC* (Communications in Computer and Information Science), vol. 536, J. Abawajy, S. Mukherjea, S. Thampi, and A. Ruiz-Martínez, Eds. Cham, Switzerland: Springer, 2015, doi: 10.1007/978-3-319-22915-7_20.

[27] M. Dolecki and R. Kozera, "The impact of the TPM weights distribution on network synchronization time," in *Computer Information Systems and Industrial Management*, vol. 9339. Cham, Switzerland: Springer, 2015, pp. 451–460.
[28] X. Pu, X.-J. Tian, J. Zhang, C.-Y. Liu, and J. Yin, "Chaotic multimedia stream cipher scheme based on true random sequence combined with tree parity machine," *Multimedia Tools Appl.*, vol. 76, no. 19, pp. 19881–19895, Oct. 2017.
[29] N. Mu and X. Liao, "An approach for designing neural cryptography," in *Proc. Int. Symp. Neural Netw.* Berlin, Germany: Springer, 2013, pp. 99–108.
[30] N. Mu, X. Liao, and T. Huang, "Approach to design neural cryptography: A generalized architecture and a heuristic rule," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 87, no. 6, Jun. 2013, Art. no. 062804, doi: 10.1103/physreve.87.062804.
[31] H. Gomez, Ó. Reyes, and E. Roa, "A 65 nm CMOS key establishment core based on tree parity machines," *Integration*, vol. 58, pp. 430–437, Jun. 2017, doi: 10.1016/j.vlsi.2017.01.010.
[32] M. Niemiec, "Error correction in quantum cryptography based on artificial neural networks," *Quantum Inf. Process.*, vol. 18, no. 6, p. 174, Jun. 2019, doi: 10.1007/s11128-019-2296-4.
[33] T. Dong and T. Huang, "Neural cryptography based on complex-valued neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4999–5004, Nov. 2020, doi: 10.1109/TNNLS.2019.2955165.
[34] N. Rana, M. S. A. Latiff, S. M. Abdulhamid, and H. Chiroma, "Whale optimization algorithm: A systematic review of contemporary applications, modifications and developments," *Neural Comput. Appl.*, vol. 32, no. 20, pp. 16245–16277, Oct. 2020, doi: 10.1007/s00521-020-04849-z.
[35] A. Engel and C. V. den Broeck, *Statistical Mechanics of Learning*. Cambridge, U.K.: Cambridge Univ. Press, Jun. 2012, doi: 10.1017/CBO9781139164542.
[36] T. M. Cover and J. A. Thomas, "Elements of information theory," in *Wiley Series in Telecommunications and Signal Processing*, 2nd ed. New York, NY, USA: Wiley, Sep. 2006.
[37] L. Ein-Dor and I. Kanter, "Confidence in prediction by neural networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 60, no. 1, pp. 799–802, Jul. 1999, doi: 10.1103/physreve.60.799.
[38] NIST. (2020). *NIST Statistical Test*. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/stats_tests.html
[39] A. Kanso and N. Smaoui, "Logistic chaotic maps for binary numbers generations," *Chaos, Solitons Fractals*, vol. 40, no. 5, pp. 2557–2568, Jun. 2009, doi: 10.1016/j.chaos.2007.10.049.
[40] B. Karakaya, A. Gülten, and M. Frasca, "A true random bit generator based on a memristive chaotic circuit: Analysis, design and FPGA implementation," *Chaos, Solitons Fractals*, vol. 119, pp. 143–149, Feb. 2019, doi: 10.1016/j.chaos.2018.12.021.
[41] V. Patidar, K. K. Sud, and N. K. Pareek, "A pseudo random bit generator based on chaotic logistic map and its statistical testing," *Informatica*, vol. 33, no. 4, pp. 441–452, 2009.
[42] L. Liu, S. Miao, H. Hu, and Y. Deng, "Pseudorandom bit generator based on non-stationary logistic maps," *IET Inf. Secur.*, vol. 10, no. 2, pp. 87–94, Mar. 2016, doi: 10.1049/iet-ifs.2014.0192.

**ARINDAM SARKAR** received the B.C.A. degree from The University of Burdwan, India, in 2005, the M.C.A. degree (Hons.) from VISVA-BHARATI University, Santiniketan, India, in 2008, the M.Tech. degree (Hons.) in computer science and engineering from the University of Kalyani, India, in 2011, and the Ph.D. degree in engineering from the Department of Computer Science and Engineering, University of Kalyani, India, in July 2015, under the Innovation in Science Pursuit for Inspired Research (INSPIRE) Fellowship of the Department of Science and Technology (DST), Government of India. He is currently an Assistant Professor with the Department of Computer Science and Electronics, Ramakrishna Mission Vidyamandira (A Residential Autonomous College under the University of Calcutta with CPE status), Howrah, India. He had secured 2nd Rank in The West Bengal College Service Commission (WBCSC) examination for general degree colleges ref: advt no. 1/2015 in computer science. He has more than 70 publications in different SCI and Scopus indexed International journals and conferences. His research interests include neural cryptography, GAN cryptography, deep learning, machine learning, soft computing, and telehealth. Dr. Sarkar is a Life Member of the Computer Society of India, a member of "The Science and Information Organization" (SAI), Hillside Avenue, New York, USA, and the International Association of Engineers (IAENG). He is currently serving as a Convener, a Chairman, a Moderator, a Question Paper Setter, a Reviewer, and an Editor for different examinations of Calcutta University, Kalyani University, West Bengal State University Vidyasagar University, Burdwan University, and many others since 2009.

**MOHAMMAD ZUBAIR KHAN** received the M.Tech. degree in computer science and engineering from U. P. Technical University, Lucknow, India, and the Ph.D. degree in computer science and information technology from the Faculty of Engineering, M. J. P. Rohilkhand University, Bareilly, India. He was the Head and an Associate Professor with the Department of Computer Science and Engineering, Invertis University, Bareilly. He is currently an Associate Professor with the Department of Computer Science, College of Computer Science and Engineering, Taibah University. He has published more than 60 journals and conference articles. He has more than 15 years of teaching and research experience. His current research interests include the IoT, machine learning, parallel and distributed computing, and computer networks. He has been a member of the Computer Society of India since 2004.

**MOIRANGTHEM MARJIT SINGH** (Senior Member, IEEE) received the B.Tech. degree in computer science and engineering from North-Eastern Hill University, Shillong, India, in 2001, the M.Tech. degree in computer science and engineering from the North Eastern Regional Institute of Science and Technology (NERIST), Arunachal Pradesh, India, in 2010, and the Ph.D. (Engg.) degree in computer science and engineering from the University of Kalyani, India in 2017. He has been the Head of the Department of Computer Science and Engineering, North Eastern Regional Institute of Science and Technology (NERIST), since 2018. He has been the Honorary Joint Secretary of the Institution of Engineers, India, Arunachal Pradesh State Centre, since 2019. He also secured 1st position in X and 2nd position in XII Examinations conducted by CBSE, New Delhi, India, amongst the candidates sent up from Jawahar Navodaya Vidyalayas(JNVs) of North Eastern region states of India in 1995 and 1997, respectively. He has more than 18 years of teaching and research experience. His research interests include mobile adhoc networks, wireless sensor networks, network security, AI, machine learning, and deep learning. He is a Fellow of IETE New Delhi, India. He was awarded the IE(I) Young Engineers Award 2014–2015 in the Computer Engineering Division, Institution of Engineers, India, and received the Best Paper Award in ICACCT 2016 International Conference (published by Springer). He was awarded the Gold Medal for getting top position in the M.Tech. program at NERIST, India. He has been a reviewer to reputed journals such as IEEE Access, *Wireless Networks* (Springer), *Microsystem Technologies* (Springer), and *Innovations in Systems and Software Engineering* (Springer).

**ABDULFATTAH NOORWALI** (Member, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2017. The title of his thesis was Modeling and Analysis of Smart Grids for Critical Data Communication. He is currently the Chairman of the Electrical and Computer Engineering Department, Faculty of Engineering and Islamic Architecture, UmmAl-Qura University, where he is also an Assistance Professor. He is also a Senior Consultant with Umm Al-Qura Consultancy Oasis, Institute of Consulting Research and Studies (ICRS), Umm Al-Qura University, where he is also the Chairman of Vision office of consultancy. He has authored many technical articles in journals and international conferences. His research interests include smart grid communications, cooperative communications, wireless networks, the Internet of Things, crowd management applications, and smart city solutions.

**CHINMAY CHAKRABORTY** is currently an Assistant Professor (Sr.) with the Department of Electronics and Communication Engineering, Birla Institute of Technology at Mesra, India. He has published 70 articles at reputed international journals, conferences, book chapters, and books. His current research interests include the Internet of Medical Things, wireless body area networks, wireless networks, telemedicine, m-health/e-health, and medical imaging. He was a publicity chair member of renowned international conferences including IEEE Healthcom and IEEE SP-DLT.

**SUBHENDU KUMAR PANI** received the Ph.D. degree from Utkal University, Odisha, India, in 2013. He is currently a Professor with the Department of Computer Science Engineering and also a Research coordinator with the Orissa Engineering College (OEC), Bhubaneswar. He has published 51 International Journal articles (25 Scopus index). His professional activities include roles as Book Series Editor (CRC Press, Apple Academic Press, and Wiley-Scrivener), associate editor, editorial board member, and/or a reviewer of various International Journals. He is an associate with number of conference societies. He has more than 150 international publications, five authored books, 15 edited and upcoming books; 20 book chapters into his account. He has more than 17 years of teaching and research experience. In addition to research, he has guided 31 M.Tech. students and two Ph.D. students. His research interests include data mining, big data analysis, web data analytics, fuzzy decision making, and computational intelligence. He is a Fellow in SSARSC and a Life Member in IE, ISTE, ISCA, OBA.OMS, SMIACSIT, SMUACEE, and CSI. He was a recipient of five researcher awards.