Chaos-guided neural key coordination for improving security of critical energy infrastructures

Arindam Sarkar¹ · Mohammad Zubair Khan² · Abdulfattah Noorwali³

Received: 1 April 2021 / Accepted: 21 June 2021 / Published online: 28 July 2021
© The Author(s) 2021

Abstract

In this paper, chaos-guided artificial neural learning-based session key coordination for industrial internet-of-things (IIoT) to enhance the security of critical energy infrastructures (CEI) is proposed. An intruder might pose several security problems since the data are transferred across a public network. Although there have been substantial efforts to solve security problems in the IIoT, the majority of them have relied on traditional methods. A wide range of privacy issues (secrecy, authenticity, and access control) must be addressed to protect IIoT systems against attack. Owing to the unique characteristics of IIoT nodes, existing solutions do not properly address the entire security range of IIoT networks. To deal with this, a chaos-based triple layer vector-valued neural network (TLVVNN) is proposed in this paper. A chaos-based exchange of common seed value for the generation of the identical input vector at both transmitter and receiver is also proposed. This technique has several advantages, including (1) it protects IIoT devices by utilizing TLVVNN synchronization to improve CEI security. (2) Here, artificial neural coordination is utilized for the exchange of neural keys between two IIoT nodes. (3) Using this suggested methodology, chaotic synchronization can be achieved, enabling the chaos-based PRNG seed exchange. (4) Vector-valued inputs and weights are taken into consideration for TLVVNN networks. (5) The deep internal architecture is made up of three hidden layers of the neural network and a vector value as input. As a result, the attacker would have great difficulty interpreting the internal structure. Experiments to verify the performance of the proposed technique are conducted, and the findings demonstrate that the proposed technique has greater performance benefits than the existing related techniques.

Keywords Triple layer vector-valued neural network (TLVVNN) · Industrial internet-of-things (IIoT) · Critical energy infrastructures (CEI) · Session key · Neural network

Introduction

The energy industry's digitization is transforming transportation, telecommunications, healthcare, banking, and defense.

✉ Arindam Sarkar
arindam.vb@gmail.com
Mohammad Zubair Khan
mkhanb@taibahu.edu.sa
Abdulfattah Noorwali
amnoorwali@uqu.edu.sa

- ¹ Department of Computer Science and Electronics, Ramakrishna Mission Vidyamandira, Belur Math, Howrah, West Bengal 711202, India
- ² Department of Computer Science, Taibah University, Medina, Saudi Arabia
- ³ Department of Electrical Engineering, Umm Al-Qura University, Mecca 24381, Saudi Arabia

Critical energy infrastructures (CEI) are rapidly dispersed, resulting in complex cyber-physical networks that require vigilant protection and fast recovery to prevent cyberattacks. Reliable and consistent energy infrastructure is essential for modern societies. Electricity, gas, and oil are used not only in our everyday lives but also in the service of other CEI. This is the most important infrastructure for deep industrialization and information integration. The latest technological advancement in the industrial internet of things (IIoT) era promotes better approaches to magnify the monitoring of energy quality and reliability problems, as well as smart grid initiatives. Resource use, on the other hand, is becoming more 'smart,' thanks to internet-connected smart appliances, smart houses, and smart electricity and gas meters. These would help with grid management and better aligning electricity production and demand, but they also open the door to cyberattacks. A smart meter in a smart grid infrastructure uses an insecure channel to access services securely. Since the infor-

mation is transmitted over a public channel, an intruder may pose several security threats. Distributed wind and solar electricity generation, energy storage systems, and electric cars that are connected to both the internet and the electricity grid are all examples of this. Hackers have made these systems a favorite target. With the spread of the IIoT allowed by the rollout of 5G wireless communication networks, the number of networked devices in the energy system is expected to increase.

The majority of the existing methodology fails to offer the necessary security features, that are deemed to be significant needs in the IIoT security. The foregoing facts point to the development of a robust key exchange strategy that can meet many security requirements while avoiding the security flaws and disadvantages that currently exist in IIoT key exchange techniques. As a result, this paper proposed a robust triple layer vector-valued neural network (TLVVNN) enabling key exchange between the two IIoT nodes. Since this TLVVNN's input and weight values are vector in nature, a complete neural synchronization can generate multiple pairs of session keys, improving security and maintaining performance. Although this proposed approach would regulate the length of the session key by changing the size of the vectors, the TLVVNN can preserve an appropriate protection standard to apply to real-life systems.

This article addresses the following issues:

1. IIoT safety and reliability toward Woman-In-The-Middle attacks must be enhanced. Whenever an assailant intercepts communication between two systems. It is a dangerous assault because the assailant is masked as the actual originator of the message. Because the assailant has the first line of communication, they would confuse the receiver into trusting that they are getting a piece of authentic information.
2. It is necessary to create a reliable key exchange method. Rather than transferring the complete key over a dedicated channel, a key must be automatically created at both network nodes via synchrony over a public network.
3. A cryptographic keys exchange scheme with high-security toughness which can withstand geometric and majority threats must be constructed.

The following is a summary of the several contributions made by this paper:

1. This paper aims to explain basic principles and reveal alternatives for efficient, cost-effective IIoT network security to overcome the problems posed by security complexity.
2. It proposes a neural key exchange method for enhancing the security of IIoT nodes. The cryptographic keys are not provided directly to the other node; instead, it is produced

by swapping only a few variables over the unprotected network while both nodes use the neural coordination process.

3. This proposed method offers vector-valued inputs and weights to the TLVVNN. This technique prevents geometric and majority attacks.
4. A chaos-based seed exchange technique is used for generating the TLVVNN's input vector using common seed value.
5. Using chaotic synchronization, both parties may create identical secret seed values (z). The same seed value is utilized to produce the common input vector. Two TLVVNNs synced with each other to generate a common seed value (z) in the sender and receiver sides by sharing specific parameters between sender and recipient. Since some of the variables that serve important components for the shared seed value (z) are not communicated via the public network, the sender keeps these variables hidden.
6. The proposed strategy addresses many attacks such as majority, replay, geometric, and session hijacking threats.
7. The key exchange mechanisms proposed by Dong and Huang [3], Jeong et al. [6], and Teodoro et al. [30] were first looked at in this study. Their shortcomings were also mentioned in this study. This present study provides a chaos-based TLVVNN synchronized key exchange strategy to provide a scalable length secret key to address the existing difficulties.
8. The proposed TLVVNN's synchronization periods in random walk, anti-Hebbian, and Hebbian learning are significantly shorter than the present CVTPM, VVTPM, and TPM-FPGA methods.

The next section explains related work. The proposed method is discussed in the subsequent section followed by which the computational complexity analysis is discussed. Then TLVVNN's security attacks are discussed. In the penultimate section, the results and analysis are given. The final section contains the conclusions and future scope, with references at the end.

Related work

Pham et al. [20] presented the foundations of the Whale Optimization Algorithm (WOA) and its applicability in wireless and communications system allocation of resources. They explored three challenges to show the application of WOA in wireless and communication systems, and initial findings are reported. WOA might be used to optimize allocation of resources in wireless networks and other technical areas, according to this article. Numan et al. [17] have emphasized the cloned nodes assault mechanism and given a quick

study of clone detection mechanisms employed in fixed wireless sensor. The sensor networks are vulnerable to numerous assaults such as clone node or node duplication due to the characteristics of WSNs like restricted processor, storage, power, and lack of tamper resistance hardware, among others. They also offered a theoretical and conceptual analysis of current centralized and decentralized strategies for detecting clone nodes, as well as their limitations and problems. Moon-samy et al. [13] highlighted that several Android apps reveal facts about the user's device without the user's permission. They investigate why certain apps breach, how they leak, and where the information is released. They discovered that three most popular Games apps, that were previously classed as safe, exposed critical device identifiers, such as the IMEI and the IMSI, to third parties. They discovered that third-party advertisement modules were the primary source of all breaches found in their dataset. Lanir et al. [28] presented a strategy that might be used in concert with stochastic, genetic, and geometric threats to ensure that the length of the synaptic does not affect the assailants' likelihood of succeeding. TPM synchronization, in particular, was demonstrated as a cryptographic key exchange method [21]. Ruttur et al. [24], Rosen-Zvi et al. [22], and Lanir et al. [28] demonstrated that cooperative learning laws can be used to synchronize two tree parity machines (TPMs). Neural cryptography has been the focus of considerable research. Ruttur et al. [23] suggested the coordination technique in the overall TPM system and discovered two major stages: attractive and repulsive steps. Neural networks are being frequently utilized in cybersecurity for a number of years [1,5,21]. According to Lu et al. [10], the blockchain enables multi-party collaborative secure data-sharing technology. In Niemiec et al. [15], the impact of changing parameters on the security and availability of various quality training laws was investigated. Edgar et al. [25] suggested a technique for identifying the appropriate TPM setup, which would result in a far more accurate and scalable neural key exchange method. Niemiec [14] develop a simple solution involving the use of a TPM to solve inaccuracies in the quantum key exchange procedure during transfer. Pal et al. [18] developed a learning algorithm which might improve key effectiveness while also speeding up coordination. For the swap of keys, Chourasia et al. [2] proposed a vectored neuronal syncing.

Sarkar et al. [26,27] presented techniques that enhanced TPM's weight range, thus improving the privacy of the key exchange. The data exchange problem was turned into ML problem by incorporating learning with privacy. AAS-IoTSG, an anonymous session key agreement mechanism for the smart grid system, was proposed by Jo et al. [7].

Makkar et al. [11] suggested IoT device protection by identifying spam using machine learning. Gao et al. [4] presented a graph-based procedure to collect the flow of data and flow control information. Shishniashvili et al. [29]

recommended using components of the coordinated synaptic weight vector as a cryptographic key rather than a complete weight vector. In quantum key management procedures, coordination of two systems could be used to eliminate errors [12]. The complex-valued tree parity machine (CVTPM) was suggested by the authors in Dong and Huang [3], which utilizes complex numbers for all control parameters. Although they only examined only the geometric attack, it is uncertain whether the CVTPM can withstand a majority attack. Jeong et al. [6] have proposed a VVTPM scheme. However, this method does not provide an effective syncing evaluation. To perform key exchange through reciprocal training of these systems, Teodoro et al. [30] recommended to put TPM architecture on an FPGA.

Proposed methodology

Generation and exchange of TLVVNN's seed through chaos synchronization

Chaos helps to generate identical secret seed values (z) at both ends using chaos synchronization. This identical seed value is used to generate an identical input vector. For generating common seed value (z) on the sender and receiver side, two TLVVNNs synchronized with each other by exchanging certain parameters between sender and receiver. The sender holds these parameters secret because some of the parameters that play important functions for the common seed value (z) are not transmitted over the public channel. This handling mechanism prevents any attack while exchanging parameters such as sniffing, spoofing, phishing attack. Here, the common seed is produced by synchronizing two TLVVNNs by following the Pecora and Carroll (PC) method. This technique exchanges tuning parameters (σ , b , r , x_1 , y_2 and z_2) between sender and receiver TLVVNNs for synchronization purposes. Using Eqs. (1), (2), and (3), two secure TLVVNNs, i.e., sender and receiver, has been designed

$$\dot{x} = \sigma(x - y) \quad (1)$$

$$\dot{y} = rx - y - xz \quad (2)$$

$$\dot{z} = xy - bz. \quad (3)$$

The organization of two TLVVNNs is the main objective of this technique. This refers to a process in which, due to the pairing or to a forcing (periodic or noisy), two (or more) chaotic-TLVVNN (either identical or not identical) control their motion to a similar result. The proposed technique uses the PC method to assume a dynamical system characterized by Eq. (4)

Fig. 1 Initial parameter exchange

Fig. 2 Exchange of calculated values

$$\dot{\bar{x}} = f(\bar{x}), \quad (4)$$

where $\dot{x} = (x_1, x_2, \dots, x_n)$ is the vector and f is a random mapping. This scheme is broken down into two sub-systems of Eqs. (5) and (6).

$$\left. \begin{aligned} \dot{\bar{k}} &= f(\bar{k}, \bar{l}) \\ \dot{\bar{i}} &= g(\bar{k}, \bar{l}) \end{aligned} \right\} \text{Driver} \quad (5)$$

$$\dot{\bar{m}} = h(\bar{k}, \bar{m}) \text{ Response.} \quad (6)$$

The response mechanism is drive by the driver $\dot{\bar{k}}(t)$. The chaotic coordination between the driver and the response system is possible with the Lyapunov exponents and the driver's adverse behavior. The next equations are to be used using Lorenz system for two secure sub-systems i.e. initiator and responder. Equations (7) and (8) can be described as initiators ($\dot{x}_1, \dot{z}_1$).

$$\dot{x}_1 = \sigma(x_1 - y) \quad (7)$$

$$\dot{z}_1 = x_1 y - b z_1. \quad (8)$$

Equations (9) and (10) can be identified as the receiver ($\dot{y}_2, \dot{z}_2$)

$$\dot{y}_2 = r x - y_2 - x z_2 \quad (9)$$

$$\dot{z}_2 = x y_2 - b z_2. \quad (10)$$

Response sub-systems for sender and receiver are driven by $y(t)$ and $x(t)$. If $t \rightarrow \infty$ then the value of $|z_2 - z_1| \rightarrow 0$. Since all systems have been synchronized, a shared value for the device is received.

The sender initializes the value of σ and b in this method, after which the value of b is sent to the receiver. The receiver initializes the r value and sends it to the sender. For points x_1 and z_1 , the sender produces a random value. The sender gives x_1 to the receiver. Initially, the receiver produces random y_2, z_2 value and sends them to the sender. Therefore, the sender receives y_2, z_2 from the receiver and receives b and x_1 from the sender. Figure 1 shows the initial parameters exchange.

The receiver calculates the new value of $\dot{y}_2$ and $\dot{z}_2$ with the help of r and b and x_1 (received from sender) using Eqs. (11) and (12) and returns the value of $\dot{y}_2$ and $\dot{z}_2$ to the sender. In Eqs. (11) and (12) the value of $x = x_1$ (current value received from sender).

$$\dot{y}_2 = r x - y_2 - x z_2 \quad (11)$$

$$\dot{z}_2 = x y_2 - b z_2. \quad (12)$$

Sender calculates the new value of $\dot{x}_1$ and $\dot{z}_1$ with the help of receives value y_2 from the receiver and own generated values σ and b using Eqs. (13) and (14) and sends the value of new $\dot{x}_1$ to the receiver and so on. In Eqs. (13) and (14), the value of $y = y_2$ (current value received from the receiver). Figure 2 illustrates the exchange of calculated values between sender and receiver.

$$\dot{x}_1 = \sigma(x_1 - y) \quad (13)$$

$$\dot{z}_1 = x_1 y - b z_1. \quad (14)$$

The sender produces a Nonce which is a random number after a predefined amount of parameter exchange. This Nonce is encrypted with the symmetric z_1 cipher and sends the results using Eq. (15).

$$\text{En_Nonce} = \text{Encrypt}_{z_1}(\text{Nonce}). \quad (15)$$

The receiver receives En_Nonce from the sender. Then decrypts En_Nonce using z_2 and performs $f(\cdot)$ using Eqs. (16) and (17), respectively.

$$\text{De_Nonce} = \text{Decrypt}_{z_2}(\text{En_Nonce}) \quad (16)$$

$$\text{Fn_Nonce} = f(\text{De_Nonce}). \quad (17)$$

The receiver encrypts the result of the previous step using z_2 and transmits to the sender using Eq. (18). Figure 3 shows the encrypted transmission Nonce.

$$\text{En_Fn_Nonce} = \text{Encrypt}_{z_2}(\text{Fn_Nonce}). \quad (18)$$

The sender is sent En_Fn_Nonce message from the receiver, attempting to deciphering with z_1 and executing

Fig. 3 Transmission of encrypted nonce

reverse function $f^{-1}()$ and checking if it is obtained or not in the original Nonce using Eq. (19).

$$\text{Nonce} = f^{-1}(\text{Decrypt}_{z_1}(\text{En_Fn_Nonce})). \quad (19)$$

If we generate the original ‘Nonce,’ it is possible to infer that both TLVVNNs have the same z value, i.e. $z_1 = z_2$, where they are synchronized. z_1 is then used as a hidden seed to construct an equivalent sender and receiver input vector. When the original Nuncio is not received, a predefined sum of the message is exchanged for chaos synchronization between the sender and the recipient.

TLVVNN’s synchronization

In this article each IIoT node has a TLVVNN. The underlying framework comprises χ_1 , χ_2 , χ_3 , M , and ψ values. M denotes the amount of required nodes in the 1st input layer at each artificial node in the 1st hidden layer, and χ_1 , χ_2 , χ_3 represents the set of nodes in the 1st, 2nd, and 3rd hidden layers, respectively. Consequently, α_u is the input vector of the u th hidden node, β_u is the weight matrix of the u th hidden node, $\alpha_{u,v}$ is the α_u ’s v th component, $\beta_{u,v}$ is the β_u ’s v th component i.e. $\beta_{u,v} \in \{-\psi, -\psi + 1, \dots, +\psi\}$, and ϑ_u is the u th hidden node’s result, the real outcomes of the VNN is δ , the localized field of the u th hidden neuron is τ_u , the order variable is G_u , the maximum number of nodes concealed in a single hidden layer is σ , and the standardized overlap is ς_u .

The TLVVNN’s internal parameters are vectorized variables. The input values in this architecture are defined in Eq. (20).

$$A^p = \begin{bmatrix} \alpha_1^p \\ \alpha_2^p \\ \vdots \\ \alpha_\eta^p \end{bmatrix} = \begin{bmatrix} \alpha_{11}^p & \alpha_{12}^p & \dots & \alpha_{1M}^p \\ \alpha_{21}^p & \alpha_{22}^p & \dots & \alpha_{2M}^p \\ \alpha_{31}^p & \alpha_{32}^p & \dots & \alpha_{3M}^p \\ \alpha_{\eta 1}^p & \alpha_{\eta 2}^p & \dots & \alpha_{\eta M}^p \end{bmatrix}, \quad (20)$$

where $p = 1, 2, \dots, \sigma$ and η denotes the amount of vectors. Equation 21 describes the weight values that are used to mapping input data to hidden units.

$$B^p = \begin{bmatrix} \beta_1^p \\ \beta_2^p \\ \vdots \\ \beta_\eta^p \end{bmatrix} = \begin{bmatrix} \beta_{11}^p & \beta_{12}^p & \dots & \beta_{1M}^p \\ \beta_{21}^p & \beta_{22}^p & \dots & \beta_{2M}^p \\ \beta_{31}^p & \beta_{32}^p & \dots & \beta_{3M}^p \\ \beta_{\eta 1}^p & \beta_{\eta 2}^p & \dots & \beta_{\eta M}^p \end{bmatrix}. \quad (21)$$

Here $\beta_{u,v}^p \in \{-\psi, -\psi + 1, \dots, +\psi\}$, $u = 1, 2, \dots, \eta$ represents the u th vector of the weight, $v = 1, 2, \dots, M$ represents the v th input value. Weight range is denoted by ψ . The p th hidden unit vector ϑ^p calculated using Eq. (22).

$$\vartheta^p = \begin{bmatrix} \vartheta_1^p \\ \vartheta_2^p \\ \vdots \\ \vartheta_\eta^p \end{bmatrix} = \text{sgn}(\tau^p) \triangleq \begin{bmatrix} \text{signum}(\tau_1^p) \\ \text{signum}(\tau_2^p) \\ \vdots \\ \text{signum}(\tau_\eta^p) \end{bmatrix}. \quad (22)$$

As a consequence, the TLVVNN and the CVTPM are the same in this situation. As a result, this paper emphasizes the TLVVNN’s ability to generalize systemic extended models such as TPM and CVTPM. Algorithm 1 explains the full TLVVNN sync procedure.

Algorithm 1:

Input : Identical vector input A^p and the arbitrary vector weight vector B^p .

Output : Both communicating IIoT devices have coordinated TLVVNN with the same session key.

Step 1: Set the arbitrary vector-valued weight vector B^p . Where, $\beta_{u,v}^p \in \{-\psi, -\psi + 1, \dots, +\psi\}$.

Steps 2 through 5 should be repeated once complete coordination is obtained.

Step 2: The result of each hidden node is determined by a weighting factor of the present state of inputs. Equation (23) provides the computation for the first hidden layer.

$$\tau_u^p = \frac{1}{\sqrt{M}} \alpha_u^p \cdot \beta_u^p = \frac{1}{\sqrt{M}} \sum_{v=1}^M \alpha_{u,v}^p \beta_{u,v}^p \quad (23)$$

In Eq. (24), $\text{signum}(\tau_u)$ denotes the output ϑ_u^p of the u th hidden unit.

$$\vartheta_u^p = \text{signum}(\tau_u) \quad (24)$$

If $\tau_u = 0$ is true, ϑ_u^p is set to -1 , resulting in binary output. If $\tau_u > 0$ is true, ϑ_u^p is mapped to $+1$, representing that the hidden neuron is operational. If the value is $\vartheta_u^p = -1$, the hidden node is disabled. This is represented in Eq. (25).

$$\text{signum}(\tau_u) = \begin{cases} -1 & \text{if } \tau_u \leq 0 \\ +1 & \text{if } \tau_u > 0 \end{cases}. \quad (25)$$

Step 3: Compute the final outcome of TLVVNN. The TLVVNN's final outcome is the product of the hidden nodes in the final layer. This is denoted by δ (Eq. 26).

$$\delta = \prod_{u=1}^{\chi^3} \vartheta_1^u \prod_{u=1}^{\chi^3} \vartheta_2^u \cdots \prod_{u=1}^{\chi^3} \vartheta_\eta^u. \quad (26)$$

Equation (27) represents how the value of δ is represented.

$$\delta = \begin{cases} -1 & \text{if } \gamma \vartheta u = -1, \text{ is odd} \\ +1 & \text{if } \vartheta u = -1, \text{ is even} \end{cases}. \quad (27)$$

If TLVVNN has a single hidden node then $\delta = \vartheta_1^p$. The δ value is same for $2^{\sigma-1}$ different $(\vartheta_1^p, \vartheta_2^p, \dots, \vartheta_\sigma^p)$ representations.

Step 4: When the outcomes of two TLVVNNs A and B are in agreement, $\delta^A = \delta^B$, then, to use one of the three rules, adjust the weight vector.

Both TLVVNN will be learned from one another using the Hebbian (Eq. 28).

$$\beta_{u,v}^+ = fn(\beta_{u,v}^p + \alpha_{u,v}^p \delta \Theta(\vartheta_u^p \delta) \Theta(\delta^A \delta^B)). \quad (28)$$

Both TLVVNNs are learned with the reversal of their output in the anti-Hebbian (Eq. 29).

$$\beta_{u,v}^+ = fn(\beta_{u,v}^p - \alpha_{u,v}^p \delta \Theta(\vartheta_u^p \delta) \Theta(\delta^A \delta^B)). \quad (29)$$

When the set value of output is not important for tuning because it is the same for both TLVVNNs, the random-walk learning rule is utilized (Eq. 30).

$$\beta_{u,v}^+ = fn(\beta_{u,v}^p + \alpha_{u,v}^p \Theta(\vartheta_u^p \delta) \Theta(\delta^A \delta^B)). \quad (30)$$

If $P = Q$ is true, then $\Theta(P, Q) = 1$ is true. Otherwise, if $P \neq Q$ is true, $\Theta(P, Q) = 0$ is true. Only weights in hidden nodes are modified with $\vartheta_u^p = \delta$, and $fn(\beta)$ are used for each learning rule. Equation (31) depicts this.

$$fn(\beta) = \begin{cases} \text{signum}(\beta)\psi & \text{for } |\beta| > \psi \\ \beta & \text{otherwise} \end{cases}. \quad (31)$$

Proceed to step 6 if the weight vectors are identical for both TLVVNN.

Step 5: If the output of both TLVVNNs differs, $\delta^A \neq \delta^B$, then the weights cannot be updated. Proceed to step 2.

Step 6: As an IIoT session key, use this synchronized weight vector.

In comparison, although the CVTPM and VVTPM can share $M \times H$ -sized and $\eta \times M \times H$ -sized secret key pairs after the full synchronization, the VNN exchange variable-length

session key (i.e., $\eta \times (M \times \chi 1 + \chi 1 \times \chi 2 + \chi 2 \times \chi 3)$ -sized keys). As a result, protection can be improved over CVTPM and VVTPM while maintaining performance.

TLVVNN synchronization is used as an IIoT key swap over technique for either block or stream ciphers. However, by changing the number of vectors η , the TLVVNN can generate different key sizes and exchange a key in polynomial time.

The probability to see $\vartheta_u^p \alpha_{u,v}^p = +1$ or $\vartheta_u^p \alpha_{u,v}^p = -1$ are not really similar, but they are influenced by the weight $\beta_{u,v}^p$ shown in Eq. (32)

$$P(\vartheta_{u,v}^p \alpha_{u,v}^p = 1) = \frac{1}{2} \left[1 + \text{erf} \left(\frac{\beta_{u,v}^p}{\sqrt{MG_u^p - \beta_{u,v}^{2p}}} \right) \right]. \quad (32)$$

The static probability representation of the weights for $t \rightarrow \infty$ is calculated for the transition probability, and it shows that $\vartheta_{u,v}^p \alpha_{u,v}^p = \text{signum}(\beta_{u,v}^p)$ happens frequently than $\vartheta_{u,v}^p \alpha_{u,v}^p = -\text{signum}(\beta_{u,v}^p)$. Equation (33) illustrates how to represent this

$$P(\beta_{u,v}^p = \beta) = R_0 \prod_{i=1}^{|\beta|} \frac{1 + \text{erf} \left(\frac{i-1}{\sqrt{MG_u^p - (i-1)^2}} \right)}{1 - \text{erf} \left(\frac{i}{\sqrt{MG_u^p - i^2}} \right)}. \quad (33)$$

Equation (34) gives the normalization constant R_0 in this case

$$R_0 = \left(\sum_{\beta=-\psi}^{\psi} \prod_{i=1}^{|\beta|} \frac{1 + \text{erf} \left(\frac{i-1}{\sqrt{MG_u^p - (i-1)^2}} \right)}{1 - \text{erf} \left(\frac{i}{\sqrt{MG_u^p - i^2}} \right)} \right)^{-1}. \quad (34)$$

The variable of the error functions would be eliminated for $M \rightarrow \infty$, ensuring that the weights are evenly distributed as illustrates in Eq. (35)

$$\sqrt{G_u(t=0)} = \sqrt{\frac{\psi(\psi+1)}{3}}. \quad (35)$$

If M is not infinite, the probability range can be described with the help of the ordering variable G_u , as given in Eq. (36)

$$G_u^p = \sum_{\beta=-\psi}^{\omega} \beta^2 P(\beta_{u,v}^p = \beta). \quad (36)$$

Equation (37) is obtained by increasing it in respect to $M^{-1/2}$

$$G_u^p = \frac{\psi(\psi+1)}{3} - \frac{8\psi^4 + 16\psi^3 - 10\psi^2 - 18\psi + 9}{15\sqrt{3\pi}\psi(\psi+1)} \frac{1}{\sqrt{M}} + O\left(\frac{\psi^4}{M}\right). \quad (37)$$

Equation (38) is used to describe the asymptotic efficiency of the order parameter in the context of $1 \ll \kappa \ll \sqrt{M}$

$$G_u^p \sim \frac{\psi(\psi+1)}{3} \left(1 + \frac{8}{5\sqrt{3\pi}} \frac{\psi}{\sqrt{M}}\right). \quad (38)$$

Equation (39) is a first-order estimate of G_u^p

$$G_u^p = \frac{\psi(\psi+1)}{3} - \frac{8\psi^4 + 16\psi^3 - 10\psi^2 - 18\psi + 9}{15\sqrt{3\pi}\psi(\psi+1)} \frac{1}{\sqrt{M}} + O\left(\frac{\psi^4}{M}\right). \quad (39)$$

This leads to a logical conclusion (Eq. 40)

$$G_u^p \sim \frac{\psi(\psi+1)}{3} \left(1 - \frac{8}{5\sqrt{3\pi}} \frac{\psi}{\sqrt{M}}\right). \quad (40)$$

Computational complexity analysis

The setup of the weights in the neural coordination approach requires $(M \times \chi_1 + \chi_1 \times \chi_2 + \chi_2 \times \chi_3)$ calculations. If $M = 3$, $\chi_1 = 3$, $\chi_2 = 4$, and $\chi_3 = 3$, the overall amount of synapse linkages (weights) is $(3 \times 3 + 3 \times 4 + 4 \times 3) = 33$. As a result, thirty-three calculations are required. It needs $(M \times \chi_1)$ calculations to generate M set of inputs for each χ_1 amount of hidden nodes. The hidden node outputs need $(\chi_1 + \chi_2 + \chi_3)$ number of operations to compute. The amount of hidden nodes in the first, second, and third layers is represented by χ_1 , χ_2 , χ_3 accordingly. The calculation of the resulting output requires a unit amount of time since it only requires a single operation.

Two parties' randomly determined weights are equal in the perfect scenario of the neural coordination technique. As a result, TLVVNNs that are coordinated at the start should not use the learning to adjust the weights. In the best scenario, $O(\text{generation of identical seed} + \text{setup of inputs} + \text{construction of weights} + \text{processing of hidden node's outputs})$ number of calculation is required. If weights generated by the two communications systems are not equivalent, the weights of the hidden nodes with a

value equal to the final outcome are modified as per the learning algorithm in each iteration. This situation provides a mean and the worst scenarios in which I iterations are required to construct matching weights on both sides. Therefore, for the mean and worst cases, the overall calculation is $((M \times \chi_1) + (M \times \chi_1 + \chi_1 \times \chi_2 + \chi_2 \times \chi_3) + (\chi_1 + \chi_2 + \chi_3)) + (I \times \text{No. of weight updation})$, that could be represented as $O(\text{time complexity during the 1st iteration} + (\text{no. of iterations} \times \text{no. of weight changes}))$.

Security attacks on TLVVNN

Majority attack

To demonstrate TLVVNN's safety, this article explores the majority assault condition, that is the most successful assault. As a consequence, Eq. (41) may be used to determine the likelihood of a successful attack.

$$P(\text{Sntm_atk} \leq \text{Sntm}) = \int_{t=0}^{\infty} P_{\text{Sntm}}^{\text{atk}}(t) \frac{d}{dt} P_{\text{Sntm}}(t) dt. \quad (41)$$

If the two sync times are believed to be non-overlapping variable elements, and what's the probability of $\text{Sntm_atk} \leq \text{Sntm}$? The intruder's and both participants' sync timings, and the coordinating period between both the two TLVVNNs, are denoted by $\text{Sntm_atk} \leq \text{Sntm}$ and Sntm . The $P_{\text{Sntm}}^{\text{atk}}(t)$ and $P_{\text{Sntm}}(t)$ are the summed likelihood distributions of each sync period. It is seen in Eq. (42)

$$P(\text{Sntm_atk} \leq \text{Sntm}) \approx 1 - \exp\left(-\frac{\text{Sntm_atk}}{\text{Sntm}}\right). \quad (42)$$

The percentage of the mean obtained from the two synchronization gaps, which are factors of the weight range ψ , determines the likelihood of a majority assault occurring. The proportion of the two coordinating times, which is a factor of ψ , may be calculated using Eqs. (41) and (42) (Eq. 43).

$$\left(\frac{\text{Sntm_atk}}{\text{Sntm}}\right) \propto \frac{\psi^2}{e^{c_1\psi + c_2\psi^2}} \quad (43)$$

Also, as consequence, the session key communication technique management involves the weight range ψ . If $\psi \gg 1$, the fraction of the two cooperation periods, is small (Eq. 51), the assailant's chances of success can be calculated as follows (Eq. 44):

$$P(\text{Sntm_atk} \leq \text{Sntm}) \propto \psi^2 e^{-c_1\psi} e^{-c_2\psi^2}. \quad (44)$$

Fig. 4 The relation between the number of rounds and the Frobenius norm of CVTPM, VVTPM, and TLVVNN

Fig. 5 Synchronization time for different weight range (average of 10,000 tests with synaptic depth 10, 15, and 20)

So the assailant's rate of success drops rapidly as the weight ranges ψ increases, the two TLVVNNs engaged would alter ψ to enhance the existing level of safety. As illustrated below, the probability of a majority assault prevailing decreases drastically in relation to ψ (Eq. 45)

$$P_{\text{attacker}} \approx e^{-0.17(\psi-2.0)}. \quad (45)$$

The TLVVNN's attack probability of a majority attack is determined using Eq. (46)

$$P_{\text{atk}}^{\text{TLVVNN}} = P_{\text{atk}}^{\delta^1} P_{\text{atk}}^{\delta^2} \dots P_{\text{atk}}^{\delta^\eta} \approx \left(P_{\text{atk}}^{\text{CVTPM}} \right)^\eta. \quad (46)$$

Here $P_{\text{atk}}^{\delta^u}$ is the likelihood of a majority assault on each of TLVVNN's weights succeeding. The probability of a majority attack on CVTPM is given by $P_{\text{atk}}^{\text{CVTPM}}$.

Replay attack

Replay attacks are a type of cyberattack that targets information sent through the web. For this assault, an adversary or another entity with identity theft accesses communication

and impersonates the originator to transmit it to the recipient. The recipient believes the communication is genuine, however, it was delivered by an assailant. The essential element of a replay assault is that the information is sent repeatedly to the recipient, hence the term "replay attack." A neuronal key is used in the suggested approach to counteract this assault. The key can only be used per transaction and never reused.

Geometric attack

This suggested methodology considers a geometric assault on TLVVNN. The adversary in this case is E . The probability of $\vartheta_u^E \neq \vartheta_u^A$ is expressed in Eq. (47) with the aid of the TLVVNN's error rate (ω).

$$\omega_u^p = \frac{1}{2} \left[1 - \text{erf} \left(\frac{\varsigma_u}{\sqrt{2(1-\varsigma_u^2)}} \frac{|\tau_u|}{\sqrt{G_u}} \right) \right]. \quad (47)$$

Equation (48) shows the likelihood of a modification using a geometric assault if the u th hidden node does not entirely agree and also the remaining of the hidden nodes are in a status of $\vartheta_v^E \neq \vartheta_v^A$.

$$P_g = \int_0^\infty \prod_{v \neq u} \left(\int_{\tau_i}^\infty \frac{2}{\sqrt{2\pi G_v}} \frac{1 - \omega_v^p}{1 - \omega_v} e^{-\frac{\tau_v^2}{2G_v}} d\tau_v \right) \frac{2}{\sqrt{2\pi G_u}} \frac{\omega_u^p}{\omega_u} e^{-\frac{\tau_u^2}{2G_u}} d\tau_u. \quad (48)$$

If the $\vartheta_v^E \neq \vartheta_v^A$ requirement is fulfilled there have been an even amount of hidden neurons that match the conditions, there will be no geometric alteration. Equation 49 is used to express this

$$P_{r,1}^E = \sum_{u=1}^{\sigma/2} \left(\frac{\sigma-1}{2u-1} \right) (1-\omega)^{\sigma-2u} \epsilon^{2u}. \quad (49)$$

Second component of P_r^E can be represented using Eq. (50)

$$P_{r,2}^E = \sum_{u=1}^{\sigma/2} \left(\frac{\sigma-1}{2u-1} \right) P_{2u-1}^- (1-\omega)^{\sigma-2u+1} \omega^{2u-1}. \quad (50)$$

The third element of P_r^E may be represented using Eq. (51)

$$P_{r,2}^E = \sum_{u=1}^{\sigma/2} \left(\frac{\sigma-1}{2u-1} \right) P_{2u-1}^- (1-\omega)^{\sigma-2u+1} \omega^{2u-1}. \quad (51)$$

Table 1 The average and median number of cycles required to achieve perfect coordination

Vector	Mean ($\psi = 10$)	Median ($\psi = 10$)	Mean ($\psi = 15$)	Median ($\psi = 15$)	Mean ($\psi = 20$)	Median ($\psi = 20$)
$\eta = 1$	6620	6294	16,784	16,113	33,752	32,517
$\eta = 2$	6795	6532	21,117	20,428	41,996	37,562
$\eta = 3$	8654	8291	23,975	23,049	35,610	34,098
$\eta = 4$	7730	7483	16,982	16,624	31,034	30,488
$\eta = 5$	7972	7747	17,423	17,036	31,882	31,224

If $\alpha > 1$, the chance of attracting and repulsion stages in the u th hidden node is expressed using Eqs. (52) and (53)

$$P_a^E = \frac{1}{2} \left(1 - \sum_{v=1}^3 P_{r,v}^E \right) \quad (52)$$

$$P_r^E = \sum_{v=1}^3 P_{r,v}^E. \quad (53)$$

Session hijacking attack

A further sort of MITM assault is session hijacking, sometimes known as cookie side-jacking. The attacker obtains the customer's session identifier and exploits it to get entry to the customer's profile in a session hijacking operation. Session hijacking occurs when a hacker takes full advantage of a breached active session by hijacking or acquiring the cookies required to administrate it. This suggested solution provides end-to-end encryption between both the sender and the recipient, preventing inadvertent session ID acquisition. The neuronal coordination mechanism generates lengthy and random session cookies, lowering the danger of an opponent guessing or forecasting what such a session cookie is. The TLVVNN begins a fresh syncing of every session to generate a fresh session key, resulting in very brief sessions. TLVVNNs are re-authenticated using a new session ID. TLVVNN also deletes a session cookie at the conclusion of each session to minimize the amount of times a session cookie is accessible in the networks.

Results and analysis

The variance of the overlap is shown in this paper using the Frobenius norm of a matrix to illustrate that the coordination periods of the proposed TLVVNN and existing VVTPM grow is proportional to the very same order ψ^2 . Using Eq. (54) the Frobenius norm is represented as follows:

$$\|B^{p^A} - B^{p^B}\|_F \triangleq \sqrt{\sum_{u=1}^n \sum_{v=1}^M |\beta_{u,v}^{p^A} - \beta_{u,v}^{p^B}|^2}. \quad (54)$$

Fig. 6 The distribution of the overlap U^{AC} with 100 pairs of networks

The Frobenius norm decreases to 0 when the two weight vectors are completely coordinated. Here, the vector values are set to 1, 2, and 5 to equate CVTPM, VVTPM, and TLVVNN without losing generality. The relation between the number of rounds and the Frobenius norm is illustrated in Fig. 4. Each line reflects a single synchronization procedure, and since the replicated experimental effects all follow the same sequence, only one outcome is shown for each η value.

Figure 5 illustrates the amount of iterations that happened for all users to complete the synchronization. Experiments were carried out on three different ranges of weight, with each reflecting the total significance of 10,000 trials.

Despite the rise in the value of η , the mean amount of iterations barely matters when considering the initial CVTPM ($\eta = 1$), VVTPM ($\eta = 2$), and TLVVNN. This reveals an identical feature not just in the mean value, as well as in the median value (Table 1). The number of vectors η has no significance on the synchronization time as each of them can be calculated separately.

Figure 6 depicts the distribution of the overlap U^{AC} with 100 pairs of networks A and C. Here, 1000 simulations are considered and $M = 100$, $\chi_1 = \chi_2 = \chi_3 = 3$, and $\psi = 3$.

Figure 7 depicts a graph depicting the likelihood of a majority attacker succeeding based on the scale of the variable η . This paper uses random walk and 100 TLVVNNs for the majority attack with 10,000 repeated tests.

Figure 8 represents the likelihood of success in various attacks for different weight ranges using the random walk. Here, 1000 simulations are considered and $M = 1000$. For genetic and geometric attacks 4096 and 100 assault networks are considered.

Fig. 7 With 10,000 repeated tests, the likelihood of a majority attack success

Fig. 8 Likelihood of success in geometric, majority and genetic attacks for different weight range using random walk learning

Table 2 The likelihood of a majority attack succeeding

Vector	($\psi = 10$)	($\psi = 15$)	($\psi = 20$)
$\eta = 1$	0.1362	0.0298	0.0042
$\eta = 2$	0.0398	0.0005	0.0002
$\eta = 3$	0.0012	0.0002	0
$\eta = 4$	0.0003	0	0
$\eta = 5$	0.0002	0	0

Table 2 shows the success probability of a majority assault. In the 10,000 attack trials, the value 0.0000 indicates that the attack was unsuccessful. In this paper, the desire security standard is set to 10^{-4} . TLVVNN with $\eta \geq 3$ has 10^{-4} attack likelihood. CVTPM with $\eta = 1$ and $\psi = 15$ has 3% attack likelihood. To achieve the standard security level the CVTPM should set $\psi = 57$.

However, if $\psi = 57$ is set, full synchronization takes a very long time, making it nearly difficult for the practical scenario. The synchronization takes approximately 1,270,000 rounds in the case of $\psi = 40$. When the TLVVNN is used with $\eta = 3$, $\psi = 15$, it requires approximately 23,000 rounds. Furthermore, by increasing the parameters η and ψ in real-world systems, while the participants use the TLVVNN, an acceptable degree of protection can be obtained.

Fig. 9 Probability of majority attack (the values of the result after 10,000 iterations)

Fig. 10 Likelihood of success in the attack using random walk learning rule for 1000 simulations

Figure 9 illustrates a graph depicting the possibility of a majority attacker progressing depending on the learning rules. In an attack situation, the TLVVNN, like previous results, will improve protection regardless of the learning rule.

Figure 10 depicts the likelihood of success in the attack using the random walk. Here, 1000 simulations are considered and $M = 1000$.

Figure 11 contrasts the speed of CVTPM and chaos-based TLVVNN coordination depending on parameters χ_1 , χ_2 , and χ_3 . Here, Hebbian learning rules are applied. Figure 12 depicts the speed of VVTPM and chaos-based TLVVNN coordination depending on parameters χ_1 , χ_2 , and χ_3 with $M = 35$. Chaos-based TLVVNN needs much fewer iterations than existing CVTPM and VVTPM, according to all of the cases. The degree of security is substantially enhanced by this feature. Present VVTPM and CVTPM methods, which employ arbitrary weights at the beginning of the coordination stage, are substantially less dependable than the suggested alternative.

The probability value (P_{value}) decides the neural session key's acceptance or rejection. Table 3 presents the results of the NIST test suite [16] on the synchronized neural session

Fig. 11 Comparison of number of iterations needed for TLVVNN coordination and existing CVTPM coordination

Fig. 12 Comparison of number of iterations needed for TLVVNN coordination and existing VVTPM coordination

key. The P_{Value} calculation is performed for the proposed TLVVNN and existing VVTPM [6] and CVTPM [3] techniques. The P_{Value} of the proposed TLVVNN has been found to exceed CVTPM.

In the synchronized neural key, the outcome of the frequency test denotes a proportion of 1 and 0. The frequency test outcome of the TLVVNN technique is 0.711478, that is

better than the outcome 0.538632 in Jeong et al. [6], 0.512374 in n Dong and Huang [3], 0.1329 in Karakaya et al. [8], 0.632558 in Patidar et al. [19], and 0.629806 in Liu et al. [9]. The frequency test's P_{Value} comparison is presented in Table 4.

Figures 13 and 14 show TLVVNN synchronizations in two different scenarios: using randomly generated identical

Table 3 P_{Value} TLVVNN, VVTPM, and CVTPM

Test	TLVVNN's p value	VVTPM's p value Jeong et al. [6]	CVTPM's p value Dong and Huang [3]
Frequency	0.711478	0.538632	0.512374
Frequency within a block	0.626898	0.551571	0.538721
Runs	0.544989	0.508936	0.489451
Longest run of ones in a block	0.099874	0.061574	0.042081
Binary matrix rank	0.768278	0.420589	0.417393
Discrete Fourier transform	0.635906	0.427985	0.392762
Non-overlapping template matching	0.556945	0.463462	0.431759
Overlapping (periodic) template matching	0.392119	0.204875	0.170426
Maurer's "universal statistical"	0.829398	0.728347	0.689478
Linear complexity	0.740825	0.657894	0.594917
Serial	0.632892	0.547216	0.478902
Approximate entropy	0.393848	0.280699	0.204175
Cumulative sums	0.736989	0.640273	0.587290
Random excursions	0.496608	0.377634	0.299875
Random excursions variants	0.412683	0.249632	0.180542

Table 4 P_{Value} measurements for frequency test

Techniques	p value
TLVVNN	0.711478
VVTPM Jeong et al. [6]	0.538632
CVTPM Dong and Huang [3]	0.512374
Karakaya et al. [8]	0.1329
Patidar et al. [19]	0.632558
Liu et al. [9]	0.629806

input vector and with chaos-guided identical input vector at the beginning of the coordination stage. 20,000 times, coordination was repeated. The mean value is shown by the dashed lines that connect the graph's points, which may be used to compare disparities. The result in the range $M = [25, 30]$, $\psi = 3$ in Hebbian learning algorithms were addressed. The amount of iterations in the coordination process is strongly affected by the value of $\chi_1 - \chi_2 - \chi_3$. Here, $\chi_1 = \chi_2 = \chi_3 = 6, 8, 12$ is considered. The amount of steps required to coordinate TLVVNNs in the chaos-guided identical input vector scenario is significantly smaller. Significantly more iterations are required to coordinate the weights of TLVVNNs using non-chaotic input vector.

The simulation results for various $\chi_1 - \chi_2 - \chi_3 - M - \psi$ are represented in Table 5. Two separate columns are used to represent the min and max coordination period. Effective attacker synchronization (E column) indicates the success of the attacker in correctly imitating the actions of the authenticated TLVVNNs. In the last column of the table, the percentage of successful attacker synchronization is shown.

Figure 15 depicts the sync time needed for 512-bit key creation using different weight ranges. It is discovered that, for big networks, Random Walk outperforms the other two learning methods.

Table 6 compares the coordination times of TLVVNN and CVTPM approaches for varied learning, weight distributions, and static network size. As the range of weight values of ψ grows in all 3 rules, a tendency towards an increase in the synchronization step has been seen. Hebbian takes less synchronization steps for small ψ values than the other two learning rules in the ψ value of 5–15, but when the ψ raises, more synchronization steps are taken by the Hebbian rule than the other rules.

Table 7 compares the TLVVNN and the VVTPM methods for synchronization time using the random walk. The coordination time of the TLVVNN employing the random walk learning is substantially better than the current VVTPM, as seen in this table.

Table 8 examines the anti-Hebbian coordination time of the TLVVNN, and TPM-FPGA approaches. As demonstrated in the table, the TLVVNN's synchronization period is significantly shorter than the present TPM-FPGA. Under the ψ of 20–30, the anti-Hebbian rule requires the least amount of time compared to the other two rules.

Conclusion and future scope

This paper first examined the key exchange schemes proposed by Dong and Huang [3], Jeong et al. [6], and Teodoro et al. [30] quite recently. This paper also indicated their limitations. Second, to address the existing issues, this paper

Fig. 13 TLVVNN coordination without chaos-based identical input vector **Fig. 14** TLVVNN coordination with chaos-based identical input vector **Table 5** Results of 70,000 simulations of varying $\chi_1 - \chi_2 - \chi_3 - M - \psi$

$\chi_1 - \chi_2 - \chi_3 - M - \psi$	No. of min coordination steps	No. of max coordination steps	Min coordination time (s)	Max coordination time (s)	Successful coordination of attacker E	% of successful coordination of attacker E (%)
(1-1-1-512-1)	10	35	0.0732	3.7285	346985	49.5
(2-2-2-256-1)	11	33	0.0711	2.9748	343652	49.09
(4-4-4-128-1)	9	49	0.0802	2.3958	348542	49.79
(8-8-8-64-1)	9	21	0.0768	3.2823	331280	47.32
(1-1-1-256-2)	10	42	0.0725	3.2825	321326	45.90
(2-2-2-128-2)	9	37	0.0695	668.4932	320917	45.84
(4-4-4-64-2)	12	234	0.0726	124.1561	292646	41.80
(8-8-8-32-2)	15	965	0.0621	75.9825	104583	14.94
(1-1-1-128-8)	16	82	0.0654	2.1632	107425	15.34
(2-2-2-64-8)	19	532	0.0819	3.9146	37985	5.42
(4-4-4-32-8)	25	846	0.0535	6.7892	145	0.0207
(8-8-8-16-8)	45	729	0.0381	0.0712	0	0.0
(1-1-1-64-128)	11	138	0.0295	0.7779	117892	16.84
(2-2-2-32-128)	19	435	0.0564	3.8723	33398	4.77
(4-4-4-16-128)	28	968	0.0416	4.1269	525	0.0750
(8-8-8-8-128)	39	1132	0.0522	5.6506	0	0.0

Fig. 15 512-bit key generation with a diverse weight range and fixed TLVVNN size

has presented a chaos-based TLVVNN synchronized key exchange scheme to produce a scalable length secret key. Chaos helps to exchange the seed value for the input vector generation of the TLVVNN. TLVVNN not only has the necessary security mechanisms, but it does have a high level of effectiveness. Third, this paper presents the systematic security assessment. The most powerful geometric and majority attacks have been considered. Also, this paper has shown that TLVVNN can achieve greater levels of security in real-world systems by altering the size of vectors. Fourthly, an analy-

sis of the effectiveness of the scheme is undertaken and the characteristics of existing CVTPM and VVTPM systems are compared. Finally, simulations that prioritize defense over an invading network are used to determine optimized values for χ_1 , χ_2 , χ_3 , M and ψ . This demonstrates that two authorized TLVVNNs will swap a key in a channel with a 0.00004% chance, permitting an invading TLVVNN to imitate the network's real behavior. A geometric technique is created and managed with a 0% accuracy. A secret key of variable sizes can be created by adjusting the TLVVNN's

Table 6 Comparison of synchronization time between TLVVNN and CVTPM

ψ value	Proposed TLVVNN's coordination time in Hebbian (cycle)	Existing CVTPM's coordination time in Hebbian (cycle) Dong and Huang [3]
5	310.58	348.16
10	678.46	813.64
15	992.95	1769.57
20	2365.15	2971.04
25	3516.28	4186.59
30	4985.46	5982.41
35	6983.12	7310.37
40	8317.36	8953.50
45	11,002.75	11,207.26
50	14,412.02	14,952.38

Table 7 Comparison of synchronization time between TLVVNN and VVTPM

ψ value	Proposed TLVVNN's coordination time in random walk (cycle)	Existing VVTPM's coordination time in random walk (cycle) Jeong et al. [6]
5	386.26	423.58
10	688.65	729.26
15	1012.29	1158.64
20	1653.95	1789.23
25	2482.12	2598.05
30	3292.76	3364.27
35	4128.25	4847.56
40	4658.34	5398.24
45	6379.06	7389.08
50	7783.45	8784.19

Table 8 Comparison of synchronization time in anti-Hebbian rule between TLVVNN and TPM-FPGA

ψ value	Proposed TLVVNN's coordination time in anti-Hebbian (cycle)	Existing TPM-FPGA's coordination time in anti-Hebbian (cycle) [30]
5	374.25	412.74
10	736.70	845.28
15	1082.49	1297.95
20	1618.42	1673.53
25	2392.74	2691.48
30	3131.08	3462.17
35	4417.52	4816.73
40	5736.85	6189.26
45	6695.43	6903.59
50	8737.55	9362.06

χ_1 , χ_2 , χ_3 , M and ψ . Finally, the results indicate that this proposed methodology is a good key exchange protocol for IIoT nodes in terms of both effectiveness and security in the CEI. There is one problem in the work: no appropriate optimization strategy for generating optimal weight vectors is investigated to improve the speed of neural coordination and improve security. A more comprehensive security

examination of the network is planned. By optimizing the weights of the TLVVNN for quicker coordination, considerable research is needed to enhance or strengthen the existing state of research in this field. Two vital components will have to be upgraded in the future. The first is to minimize the neural coordination latency even further, and the second is to perform new neural coordination instead of just a longer one.

Declarations

Conflict of interest No conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Abdalrdha ZK, AL-Qinani IH, Abbas FN (2019) Subject review: key generation in different cryptography algorithm. *Int J Sci Res Sci Eng Technol* 6(5):230–240. <https://doi.org/10.32628/ijrsrset196550>
- Chourasia S, Bharadwaj HC, Das, Q, Agarwal K, Lavanya K (2019) Vectorized Neural Key Exchange Using Tree Parity Machine. *Comput Soft* 8(5):3140–3145. 5
- Dong T, Huang T (2020) Neural cryptography based on complex-valued neural network. *IEEE Trans Neural Netw Learn Syst* 31(11):4999–5004. <https://doi.org/10.1109/TNNLS.2019.2955165>
- Gao J, Yang X, Jiang Y, Song H, Choo KKR, Sun J (2021) Semantic learning based cross-platform binary vulnerability search for IoT devices. *IEEE Trans Ind Inform* 17(2):971–979. <https://doi.org/10.1109/TII.2019.2947432>
- Hadke PP, Kale SG (2016) Use of Neural Networks in cryptography: A review. In: 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave), Coimbatore, India, IEEE, pp 1–4. 10.1109/STARTUP.2016.7583925
- Jeong S, Park C, Hong D, Seo C, Jho N (2021) Neural cryptography based on generalized tree parity machine for real-life systems. *Secur Commun Netw*. <https://doi.org/10.1155/2021/6680782>
- Jo M, Jangirala S, Das AK, Li X, Khan MK (2020) Designing anonymous signature-based authenticated key exchange scheme for IoT-enabled smart grid systems. *IEEE Trans Ind Inform*. <https://doi.org/10.1109/TII.2020.3011849>
- Karakaya B, Gülten A, Frasca M (2019) A true random bit generator based on a memristive chaotic circuit: Analysis, design and FPGA implementation. *Chaos Solitons Fractals* 119:143–149
- Liu L, Miao S, Hu H, Deng Y (2016) Pseudo-random bit generator based on non-stationary logistic maps. *IET Inf Secur* 2(10):87–94
- Lu Y, Huang X, Dai Y, Maharjan S, Zhang Y (2020) Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. *IEEE Trans Ind Inform* 16(6):4177–4186
- Makkar A, Garg S, Kumar N, Hossain MS, Ghoneim A, Alrashoud M (2021) An efficient spam detection technique for IoT devices using machine learning. *IEEE Trans Ind Inform* 17(2):903–912. <https://doi.org/10.1109/TII.2020.2968927>
- Mehic M, Niemiec M, Siljak H, Voznak M (2020) Error Reconciliation in Quantum Key Distribution Protocols. In: Ulidowski I, Lanese I, Schultz U, Ferreira C (eds) *Reversible Computation: Extending Horizons of Computing*. RC 2020. Lecture Notes in Computer Science, vol 12070. Springer, Cham. <https://doi.org/10.1007/978-3-030-47361-7-11>
- Moonsamy V, Alazab M, Batten L (2013) Towards an understanding of the impact of advertising on data leaks. *Int J Secur Netw*. <https://doi.org/10.1504/IJSN.2012.052540>
- Niemiec (2019) Error correction in quantum cryptography based on artificial neural networks. *Quantum Inf Process* 8:174
- Niemiec M, Mehic M, Voznak (2018) Security Verification of Artificial Neural Networks Used to Error Correction in Quantum Cryptography. In: 2018 26th Telecommunications Forum (TELFOR) Belgrade, Serbia, IEEE, pp 1–4. <https://doi.org/10.1109/TELFOR.2018.8612006>
- NIST (2020) NIST Statistical Test. http://csrc.nist.gov/groups/ST/810toolkit/rng/stats_tests.html. Accessed 30 Mar 2021
- Numan M, Subhan F, Khan WZ, Hakak S, Haider S, Reddy GT, Jolfaei A, Alazab M (2020) A systematic review on clone node detection in static wireless sensor networks. *IEEE Access* 8:65450–65461. <https://doi.org/10.1109/ACCESS.2020.2983091>
- Pal SK, Mishra S, Mishra S (2019) An TPM based approach for generation of secret key. *Int J Comput Netw Inf Secur* 11(10):45–50
- Patidar V, Sud KK, Pareek NK (2009) A pseudo random bit generator based on chaotic logistic map and its statistical testing. *Informatica* 33:441–452
- Pham QV, Mirjalili S, Kumar N, Alazab M, Hwang WJ (2020) Whale optimization algorithm with applications to resource allocation in wireless networks. *IEEE Trans Veh Technol* 69(4):4285–4297. <https://doi.org/10.1109/TVT.2020.2973294>
- Protic D (2016) Neural cryptography. *Vojnoteh Glas* 64(2):483–495. <https://doi.org/10.5937/vojtehg64-8877>
- Rosen-Zvi M, Kanter I, Kinzel W (2002) Cryptography based on neural networks analytical results. *J Phys A Math Gen* 35(47):L707–L713. <https://doi.org/10.1088/0305-4470/35/47/104>
- Ruttar A, Kinzel W, Naeh R, Kanter I (2006) Genetic attack on neural cryptography. *Phys Rev E*. <https://doi.org/10.1103/physreve.73.036121>
- Ruttar A, Kinzel W, Kanter I (2007) Dynamics of neural cryptography. *Phys Rev E*. <https://doi.org/10.1103/physreve.75.056104>
- Dorokhin ES, Fuertes W, Lascano E (2019) On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key. *Secur Commun Netw* 2019:1–10. <https://doi.org/10.1155/2019/8214681>
- Sarkar A (2019) Multilayer neural network synchronized secured session key based encryption in wireless communication. *Int J Artif Intell* 8(1):44–53
- Sarkar A, Mandal J (2012) Swarm intelligence based faster public-key cryptography in wireless communication (SIFPKC). *Int J Comput Sci Eng Technol (IJCSET)* 3(7):267–273
- Shacham LN, Klein E, Mislovaty R, Kanter I, Kinzel W (2004) Cooperating attackers in neural cryptography. *Phys Rev E*. <https://doi.org/10.1103/physreve.69.066137>
- Shishniashvili E, Mamisashvili L, Mirtskhulava L (2020) Enhancing IoT security using multi-layer feedforward neural network with tree parity machine elements. *Int J Simul Syst Sci Technol* 21(2):371–383. <https://doi.org/10.5013/ijssst.a.21.02.37>
- Teodoro A, Gomes O, Saadi M et al (2021) An FPGA-based performance evaluation of artificial neural network architecture algorithm for IoT. *Wirel Pers Commun*. <https://doi.org/10.1007/s11277-021-08566-1>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.