International Journal of Innovative
Computing, Information and Control
Volume **18**, Number **6**, December **2022**

 ISSN 1349-4198
pp. 1871–1883

NEURAL CRYPTOGRAPHY BASED ON QUATERNION-VALUED NEURAL NETWORK

Ying Zhang, Weihua Wang and Huisheng Zhang*

School of Science
Dalian Maritime University
No. 1, Linghai Road, Dalian 116026, P. R. China
yingzhang@dlmu.edu.cn; 1304629489@qq.com; *Corresponding author: hszhang@dlmu.edu.cn

Received March 2022; revised July 2022

Abstract. *Neural cryptography is a public key exchange protocol based on mutual learning of two identically structured neural networks. After learning each other and eventually reaching full synchronization, the two networks attain common network weights which can be used as a secret key for cryptographic purposes. The most common model used in neural cryptography is the tree parity machine (TPM). In this paper, neural cryptography is proposed based on quaternion-valued TPM (QVTPM). The mutual learning strategy is established and the security of QVTPM is theoretically analyzed. The main advantages of the proposed QVTPM are twofold: (i) the two communicating parties can exchange four group keys during one neural synchronization process; (ii) the security of QVTPM is higher than that of TPM and that of complex-valued TPM with the same number of input neurons and hidden units. Numerical simulation results ascertain the efficiency of QVTPM and our theoretical findings.*

Keywords: Neural cryptography, Tree parity machine, Quaternion-valued neural network, Neural synchronization

1. **Introduction.** The public key exchange protocol (PKEP) has become an important component of cryptosystems since it was first proposed by Diffie and Hellman [1]. PKEP allows two parties to jointly establish a shared secret key over an insecure communication channel, and the key can then be used to encrypt subsequent communication in a number of applications such as identification, authentication, and data encryption.

PKEPs are traditionally built upon number theory, thus needing a high cost in computation and memory during the process of protocol implementation. Neural synchronization provides another way to design public key exchange protocols, which are the so-called neural cryptography [2-9]. Equipped with two identically structured neural networks, two parties A and B start with randomly chosen weight vectors. During the mutual learning process, they share a common input vector in each step and update the network weights according to certain learning rules. Eventually, the two neural networks are synchronized with a common weight vector which can be used as a secret key [10-24]. Though an attacker E has the ability to access the communication channel, he cannot retrieve the key.

The most popular model used for neural cryptography is the tree parity machine (TPM) [25]. It has been proved that the synchronization of TPMs can be achieved with the Hebbian learning rules [26], thus establishing the feasibility for TPM to be used as a key exchange protocol. For the purpose of cryptanalysis, four types of attack algorithms for neural cryptography have been proposed: simple attack [25], geometric attack [27],

DOI: 10.24507/ijicic.18.06.1871

majority attack [28], and genetic attack [29]. Though TPM with one or two hidden units was found insecure, TPM with three (or more) hidden units can successfully resist the four aforementioned attack algorithms by increasing the synaptic depth of its networks [30-32]. Based on the original TPM model, several variants or improvement models of TPM have also been proposed, such as tree state classification machine (TSCM) [33], two-layer tree-connected feed-forward neural network (TTFNN) [34], and "Don't trust my partner (DTMP)" [35]. Tree committee machine (TCM) [26] and permutation parity machine (PPM) [36] are two other neural cryptography models which are different from TPM, however, they have been proved insecure under attacks [37].

Recently, Dong and Huang [38] proposed a neural cryptography model based on the complex-valued tree parity machine network (CVTPM), and they found that CVTPM is more secure than TPM and can exchange two group keys in one neural synchronization process [39,40].

The aforementioned neural cryptosystems are all based on the real-valued or complex-valued neural network models, and that based on the quaternion-valued neural networks is still lacking in the literature. In order to further improve the security and efficiency of neural cryptography, in this paper, we propose a neural cryptography model based on the quaternion-valued tree parity machine (QVTPM). The main contributions of this paper are as follows.

(i) We establish the network structure and the information flowing mechanism of QVTPM, whose input, output and network weights are all quaternion values.

(ii) We propose the mutual learning rules for QVTPM, by which four group keys can be exchanged during one neural synchronization process.

(iii) The security and time efficiency of QVTPM is theoretically investigated. Benefitting from the quaternion network structure and the quaternion mutual learning rules, QVTPM enjoys higher security than TPM and CVTPM, while remaining the same order of synchronization time as TPM and CVTPM.

The remainder of this paper is organized as follows. The network architecture and the mutual learning process of QVTPM are elaborated in the next section. In Sections 3 and 4, we analyze the security and synchronization time of the QVTPM, respectively. Several numerical examples are conducted in Section 5 to illustrate the advantages of the proposed QVTPM. This paper is concluded in Section 6.

2. Model Description.

Different from TPM [25] and CVTPM [38], the weight, input vector and output vector of QVTPM are all quaternion-valued. This new characteristic brings out the need for new learning rules to attain synchronization. In the following, we give a description of QVTPM from the viewpoints of the network architecture and the mutual learning process.

2.1. Network architecture.

As shown in Figure 1, the structure of QVTPM is inherited from TPM and can be regarded as a tree neural network constituted by three layers: input layer, hidden layer, and output layer. There are $K \times N$ input neurons in the input layer, K hidden units in the hidden layer, and one output neuron in the output layer. Each hidden unit works as a perceptron with independent receptive fields, including N input neurons and one output neuron. Each weight $\omega_{u,v}$ connecting the uth hidden unit and its vth input neuron is a quaternion value and can be described as follows:

$$\omega_{u,v} = a_{u,v} + b_{u,v}i + c_{u,v}j + d_{u,v}k, \tag{1}$$

where $a_{u,v}, b_{u,v}, c_{u,v}, d_{u,v} \in \{-L, -L+1, \ldots, L\}$, L represents the synaptic depth of the networks, $u = 1, 2, \ldots, K$, and $v = 1, 2, \ldots, N$. Let $(x_{u,v})_{1\times KN}$ be the input vector with

$$x_{u,v} = a_x + b_x i + c_x j + d_x k, \tag{2}$$

where $a_x, b_x, c_x, d_x \in \{-1, 1\}$. The input of the uth hidden unit is defined by

$$h_u = \frac{1}{\sqrt{N}} \left(\sum_{v=1}^{N} (a_{u,v} a_x) + i \sum_{v=1}^{N} (b_{u,v} b_x) + j \sum_{v=1}^{N} (c_{u,v} c_x) + k \sum_{v=1}^{N} (d_{u,v} d_x) \right), \tag{3}$$

and the output of the uth hidden unit is defined by

$$\sigma_u = a_{\sigma_u} + b_{\sigma_u} i + c_{\sigma_u} j + d_{\sigma_u} k, \tag{4}$$

where $a_{\sigma_u} = \mathrm{sgn}(\mathfrak{R}(h_u))$, $b_{\sigma_u} = \mathrm{sgn}(\mathfrak{T}(h_u))$, $c_{\sigma_u} = \mathrm{sgn}(\mathfrak{J}(h_u))$, $d_{\sigma_u} = \mathrm{sgn}(\mathfrak{K}(h_u))$, and $\mathfrak{R}$, $\mathfrak{T}$, $\mathfrak{J}$, $\mathfrak{K}$ denote the real part and the three imaginary parts, respectively. The output τ of the QVTPM can then be defined as follows:

$$\tau = \prod_{u=1}^{K} \mathfrak{R}(\sigma_u) + i \prod_{u=1}^{K} \mathfrak{T}(\sigma_u) + j \prod_{u=1}^{K} \mathfrak{J}(\sigma_u) + k \prod_{u=1}^{K} \mathfrak{K}(\sigma_u). \tag{5}$$

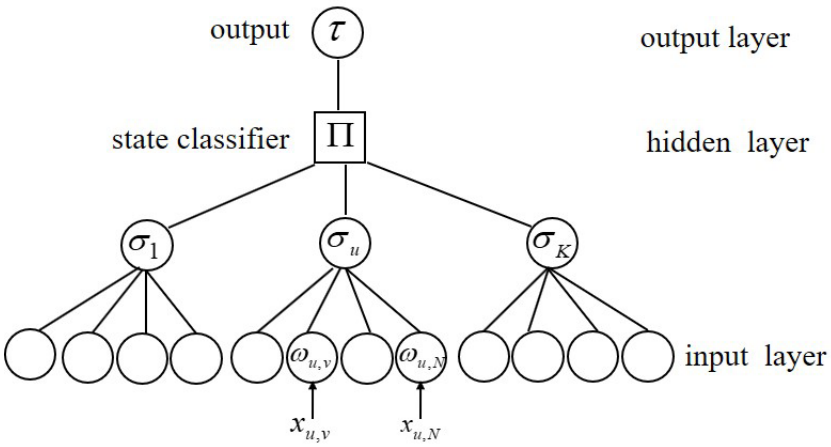


FIGURE 1. A QVTPM network with $K = 3$ and $N = 4$

Remark 2.1. *If $c_x = d_x = 0$ and $c_{u,v} = d_{u,v} = 0$, then the QVTPM reduces to the CVTPM. Thus, the CVTPM can be regarded as a special case of our model.*

2.2. **Mutual learning process.** The flow diagram for the two parties A and B to negotiate a secret key based on QVTPM is given in Figure 2. A detailed description for the negotiation procedure is provided as follows.

(I) Equip both the two parties A and B with a QVTPM. The quaternion-valued weight vectors $(\omega_{u,v})^A_{1\times KN}$ of A's QVTPM and $(\omega_{u,v})^B_{1\times KN}$ of B's QVTPM are randomly and independently initialized and then kept secretly.

(II) At each time step, a common input vector $(x_{u,v})_{1\times KN}$ is randomly generated. Once receiving $(x_{u,v})_{1\times KN}$, A and B compute the output τ_A and τ_B according to (3)-(5). Then, A and B exchange their output values on the public channel.

(III) When $A(B)$ receives $\tau_B(\tau_A)$, then $A(B)$ judges whether or not to update the weight according to the following rules.

a) If $\mathfrak{R}(\tau_A) \neq \mathfrak{R}(\tau_B)$, $\mathfrak{T}(\tau_A) \neq \mathfrak{T}(\tau_B)$, $\mathfrak{J}(\tau_A) \neq \mathfrak{J}(\tau_B)$ and $\mathfrak{K}(\tau_A) \neq \mathfrak{K}(\tau_B)$, then the weight $(\omega_{u,v})^{A/B}_{1\times KN}$ cannot be updated.

b) If $\mathfrak{R}(\tau_A) = \mathfrak{R}(\tau_B) = \mathfrak{R}\left(\sigma_u^{A/B}\right)$, $\mathfrak{T}(\tau_A) = \mathfrak{T}(\tau_B) = \mathfrak{T}\left(\sigma_u^{A/B}\right)$, $\mathfrak{J}(\tau_A) = \mathfrak{J}(\tau_B) = \mathfrak{J}\left(\sigma_u^{A/B}\right)$ and $\mathfrak{K}(\tau_A) = \mathfrak{K}(\tau_B) = \mathfrak{K}\left(\sigma_u^{A/B}\right)$, then update the weight $\omega_u^{A/B}$.

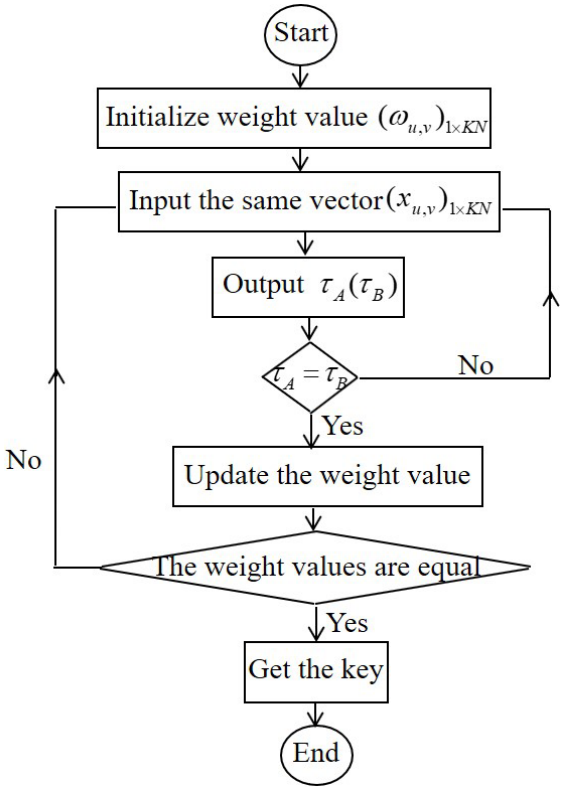


FIGURE 2. The flow diagram for the mutual learning process

c) If $\mathfrak{R}(\tau_A) = \mathfrak{R}(\tau_B) = \mathfrak{R}\left(\sigma_u^{A/B}\right)$, $\mathfrak{T}(\tau_A) = \mathfrak{T}(\tau_B) \neq \mathfrak{T}\left(\sigma_u^{A/B}\right)$, $\mathfrak{J}(\tau_A) = \mathfrak{J}(\tau_B) \neq \mathfrak{J}\left(\sigma_u^{A/B}\right)$ and $\mathfrak{K}(\tau_A) = \mathfrak{K}(\tau_B) \neq \mathfrak{K}\left(\sigma_u^{A/B}\right)$, then only update the real part $\mathfrak{R}(\bullet)$ of the weight $\omega_u^{A/B}$; other cases can be conducted similarly.

d) If $\mathfrak{R}(\tau_A) = \mathfrak{R}(\tau_B) = \mathfrak{R}\left(\sigma_u^{A/B}\right)$, $\mathfrak{T}(\tau_A) = \mathfrak{T}(\tau_B) = \mathfrak{T}\left(\sigma_u^{A/B}\right)$, $\mathfrak{J}(\tau_A) = \mathfrak{J}(\tau_B) \neq \mathfrak{J}\left(\sigma_u^{A/B}\right)$ and $\mathfrak{K}(\tau_A) = \mathfrak{K}(\tau_B) \neq \mathfrak{K}\left(\sigma_u^{A/B}\right)$, then update the real part $\mathfrak{R}(\bullet)$ and the imaginary part $\mathfrak{T}(\bullet)$ of the weight $\omega_u^{A/B}$; other cases can be conducted similarly.

e) If $\mathfrak{R}(\tau_A) = \mathfrak{R}(\tau_B) = \mathfrak{R}\left(\sigma_u^{A/B}\right)$, $\mathfrak{T}(\tau_A) = \mathfrak{T}(\tau_B) = \mathfrak{T}\left(\sigma_u^{A/B}\right)$, $\mathfrak{J}(\tau_A) = \mathfrak{J}(\tau_B) = \mathfrak{J}\left(\sigma_u^{A/B}\right)$, $\mathfrak{K}(\tau_A) = \mathfrak{K}(\tau_B) \neq \mathfrak{K}\left(\sigma_u^{A/B}\right)$, then only the imaginary part $\mathfrak{K}(\bullet)$ of the weight $\omega_u^{A/B}$ cannot be updated; other cases can be conducted similarly.

(IV) Update the weight according to one of the following learning rules.

a) Hebbian learning rule

$$\begin{cases} \mathfrak{R}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{R}} + \left[x_{u,v}^{\mathfrak{R}}\mathfrak{R}(\tau)\right] \Theta \left[\mathfrak{R}\left(\sigma_u^{A/B}\right)\mathfrak{R}(\tau)\right] \Theta \left[\mathfrak{R}\left(\tau^A\right)\mathfrak{R}\left(\tau^B\right)\right]\right), \\ \mathfrak{T}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{T}} + \left[x_{u,v}^{\mathfrak{T}}\mathfrak{T}(\tau)\right] \Theta \left[\mathfrak{T}\left(\sigma_u^{A/B}\right)\mathfrak{T}(\tau)\right] \Theta \left[\mathfrak{T}\left(\tau^A\right)\mathfrak{T}\left(\tau^B\right)\right]\right), \\ \mathfrak{J}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{J}} + \left[x_{u,v}^{\mathfrak{J}}\mathfrak{J}(\tau)\right] \Theta \left[\mathfrak{J}\left(\sigma_u^{A/B}\right)\mathfrak{J}(\tau)\right] \Theta \left[\mathfrak{J}\left(\tau^A\right)\mathfrak{J}\left(\tau^B\right)\right]\right), \\ \mathfrak{K}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{K}} + \left[x_{u,v}^{\mathfrak{K}}\mathfrak{K}(\tau)\right] \Theta \left[\mathfrak{K}\left(\sigma_u^{A/B}\right)\mathfrak{K}(\tau)\right] \Theta \left[\mathfrak{K}\left(\tau^A\right)\mathfrak{K}\left(\tau^B\right)\right]\right); \end{cases} \tag{6}$$

b) Anti-Hebbian learning rule

$$\begin{cases} \mathfrak{R}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{R}} - \left[x_{u,v}^{\mathfrak{R}}\mathfrak{R}(\tau)\right]\Theta\left[\mathfrak{R}\left(\sigma_u^{A/B}\right)\mathfrak{R}(\tau)\right]\Theta\left[\mathfrak{R}\left(\tau^A\right)\mathfrak{R}\left(\tau^B\right)\right]\right), \\ \mathfrak{T}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{T}} - \left[x_{u,v}^{\mathfrak{T}}\mathfrak{T}(\tau)\right]\Theta\left[\mathfrak{T}\left(\sigma_u^{A/B}\right)\mathfrak{T}(\tau)\right]\Theta\left[\mathfrak{T}\left(\tau^A\right)\mathfrak{T}\left(\tau^B\right)\right]\right), \\ \mathfrak{J}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{J}} - \left[x_{u,v}^{\mathfrak{J}}\mathfrak{J}(\tau)\right]\Theta\left[\mathfrak{J}\left(\sigma_u^{A/B}\right)\mathfrak{J}(\tau)\right]\Theta\left[\mathfrak{J}\left(\tau^A\right)\mathfrak{J}\left(\tau^B\right)\right]\right), \\ \mathfrak{K}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{K}} - \left[x_{u,v}^{\mathfrak{K}}\mathfrak{K}(\tau)\right]\Theta\left[\mathfrak{K}\left(\sigma_u^{A/B}\right)\mathfrak{K}(\tau)\right]\Theta\left[\mathfrak{K}\left(\tau^A\right)\mathfrak{K}\left(\tau^B\right)\right]\right); \end{cases} \tag{7}$$

c) Random walk learning rule

$$\begin{cases} \mathfrak{R}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{R}} + x_{u,v}^{\mathfrak{R}}\Theta\left[\mathfrak{R}\left(\sigma_u^{A/B}\right)\mathfrak{R}(\tau)\right]\Theta\left[\mathfrak{R}\left(\tau^A\right)\mathfrak{R}\left(\tau^B\right)\right]\right), \\ \mathfrak{T}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{T}} + x_{u,v}^{\mathfrak{T}}\Theta\left[\mathfrak{T}\left(\sigma_u^{A/B}\right)\mathfrak{T}(\tau)\right]\Theta\left[\mathfrak{T}\left(\tau^A\right)\mathfrak{T}\left(\tau^B\right)\right]\right), \\ \mathfrak{J}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{J}} + x_{u,v}^{\mathfrak{J}}\Theta\left[\mathfrak{J}\left(\sigma_u^{A/B}\right)\mathfrak{J}(\tau)\right]\Theta\left[\mathfrak{J}\left(\tau^A\right)\mathfrak{J}\left(\tau^B\right)\right]\right), \\ \mathfrak{K}\left(\omega_{u,v}^{+}\right) = g\left(\omega_{u,v}^{\mathfrak{K}} + x_{u,v}^{\mathfrak{K}}\Theta\left[\mathfrak{K}\left(\sigma_u^{A/B}\right)\mathfrak{K}(\tau)\right]\Theta\left[\mathfrak{K}\left(\tau^A\right)\mathfrak{K}\left(\tau^B\right)\right]\right). \end{cases} \tag{8}$$

Here $\Theta(y)$ is the heavyside function [26], which equals zero for $y < 0$ and 1 otherwise, and $g(\omega)$ is defined as follows:

$$g(\omega) = \begin{cases} \operatorname{sgn}(\omega)L, & \text{for } |\omega| > L, \\ \omega, & \text{otherwise.} \end{cases} \tag{9}$$

(V) Repeat procedures (II)-(IV) until synchronization $\left((\omega_{u,v})_{1\times KN}^{A} = (\omega_{u,v})_{1\times KN}^{B}\right)$ is attained. The final weight vector $(\omega_{u,v})_{1\times KN}^{A/B}$ can then be used as a common secret key between A and B.

During procedure (III), if the weight of at least one network can be updated, then the update step can be classified into two cases to characterize the behaviour of hidden nodes.

Case 1: An attractive step. This case involves four possible behaviours: $\mathfrak{R}\left(\tau^A\right) = \mathfrak{R}\left(\tau^B\right) = \mathfrak{R}\left(\sigma_u^{A/B}\right)$, $\mathfrak{T}\left(\tau^A\right) = \mathfrak{T}\left(\tau^B\right) = \mathfrak{T}\left(\sigma_u^{A/B}\right)$, $\mathfrak{J}\left(\tau^A\right) = \mathfrak{J}\left(\tau^B\right) = \mathfrak{J}\left(\sigma_u^{A/B}\right)$, and $\mathfrak{K}\left(\tau^A\right) = \mathfrak{K}\left(\tau^B\right) = \mathfrak{K}\left(\sigma_u^{A/B}\right)$. In any of those behaviours, the real part or the three imaginary parts of $(\omega_{u,v})_{1\times KN}^{A/B}$ are updated respectively in the same direction, which increases the overlap on average. Such steps help to attain synchronization.

Case 2: A repulsive step. This case involves four possible behaviours: if $\mathfrak{R}\left(\tau^A\right) = \mathfrak{R}\left(\tau^B\right)$, but $\mathfrak{R}\left(\sigma_u^A\right) \neq \mathfrak{R}\left(\sigma_u^B\right)$, only one real part of $(\omega_{u,v})_{1\times KN}^{A/B}$ is updated, while that of another party B/A is unchanged; if $\mathfrak{T}\left(\tau^A\right) = \mathfrak{T}\left(\tau^B\right)$, but $\mathfrak{T}\left(\sigma_u^A\right) \neq \mathfrak{T}\left(\sigma_u^B\right)$, only one i-imaginary part of $\omega_u^{A/B}$ is updated, while that of another party B/A is unchanged; if $\mathfrak{J}\left(\tau^A\right) = \mathfrak{J}\left(\tau^B\right)$, but $\mathfrak{J}\left(\sigma_u^A\right) \neq \mathfrak{J}\left(\sigma_u^B\right)$, only one j-imaginary part of $\omega_u^{A/B}$ is updated, while that of another party is unchanged; if $\mathfrak{K}\left(\tau^A\right) = \mathfrak{K}\left(\tau^B\right)$, but $\mathfrak{K}\left(\sigma_u^A\right) \neq \mathfrak{K}\left(\sigma_u^B\right)$, only one k-imaginary part of $\omega_u^{A/B}$ is updated, while that of another party B/A is unchanged. The above four behaviours decrease the overlap on average, and thus reduce the synchronization speed.

3. **Security Analysis of QVTPM.** A computationally secure system should satisfy the following two conditions [41-43]:

1) The average synchronization time of the mutual learning between A and B grows at a polynomial rate with the increase of the synaptic depth L;

2) The average synchronization time of the unidirectional learning for an attacker E grows at an exponential rate.

The synchronization degree of QVTPM is measured by the overlap between the corresponding hidden units of the two parties A and B in the form of

$$\bar{\rho} = \frac{\rho_{\mathfrak{R}} + \rho_{\mathfrak{T}} + \rho_{\mathfrak{J}} + \rho_{\mathfrak{K}}}{4}, \tag{10}$$

where

$$\begin{aligned}
\rho_{\mathfrak{R}} &= \frac{\mathfrak{R}\left(\omega_u^A\right)\mathfrak{R}\left(\omega_u^B\right)}{\sqrt{\mathfrak{R}\left(\omega_u^A\right)\mathfrak{R}\left(\omega_u^A\right)}\sqrt{\mathfrak{R}\left(\omega_u^B\right)\mathfrak{R}\left(\omega_u^B\right)}}, \quad \rho_{\mathfrak{R}} \in [0,1];\\
\rho_{\mathfrak{T}} &= \frac{\mathfrak{T}\left(\omega_u^A\right)\mathfrak{T}\left(\omega_u^B\right)}{\sqrt{\mathfrak{T}\left(\omega_u^A\right)\mathfrak{T}\left(\omega_u^A\right)}\sqrt{\mathfrak{T}\left(\omega_u^B\right)\mathfrak{T}\left(\omega_u^B\right)}}, \quad \rho_{\mathfrak{T}} \in [0,1];\\
\rho_{\mathfrak{J}} &= \frac{\mathfrak{J}\left(\omega_u^A\right)\mathfrak{J}\left(\omega_u^B\right)}{\sqrt{\mathfrak{J}\left(\omega_u^A\right)\mathfrak{J}\left(\omega_u^A\right)}\sqrt{\mathfrak{J}\left(\omega_u^B\right)\mathfrak{J}\left(\omega_u^B\right)}}, \quad \rho_{\mathfrak{J}} \in [0,1];\\
\rho_{\mathfrak{K}} &= \frac{\mathfrak{K}\left(\omega_u^A\right)\mathfrak{K}\left(\omega_u^B\right)}{\sqrt{\mathfrak{K}\left(\omega_u^A\right)\mathfrak{K}\left(\omega_u^A\right)}\sqrt{\mathfrak{K}\left(\omega_u^B\right)\mathfrak{K}\left(\omega_u^B\right)}}, \quad \rho_{\mathfrak{K}} \in [0,1];\\
\mathfrak{R}(\omega_u) &= \left(\mathfrak{R}(\omega_{u,1}), \mathfrak{R}(\omega_{u,2}), \ldots, \mathfrak{R}(\omega_{u,N})\right);\\
\mathfrak{T}(\omega_u) &= \left(\mathfrak{T}(\omega_{u,1}), \mathfrak{T}(\omega_{u,2}), \ldots, \mathfrak{T}(\omega_{u,N})\right);\\
\mathfrak{J}(\omega_u) &= \left(\mathfrak{J}(\omega_{u,1}), \mathfrak{J}(\omega_{u,2}), \ldots, \mathfrak{J}(\omega_{u,N})\right);\\
\mathfrak{K}(\omega_u) &= \left(\mathfrak{K}(\omega_{u,1}), \mathfrak{K}(\omega_{u,2}), \ldots, \mathfrak{K}(\omega_{u,N})\right).
\end{aligned} \tag{11}$$

As the network weights are initialized randomly, $\bar{\rho}$ approximately equals zero at the start of the synchronization process. During the mutual learning stage, ρ_u moves towards 1 and eventually stabilizes at 1 when the synchronization is attained. In order to characterize the probability that the outputs σ_u of the two corresponding hidden units are different, define the generation errors of the real and three imaginary components as

$$\varepsilon_{\mathfrak{R}} = \frac{1}{\pi}\arccos(\rho_{\mathfrak{R}}), \;\; \varepsilon_{\mathfrak{T}} = \frac{1}{\pi}\arccos(\rho_{\mathfrak{T}}), \;\; \varepsilon_{\mathfrak{J}} = \frac{1}{\pi}\arccos(\rho_{\mathfrak{J}}), \;\; \varepsilon_{\mathfrak{K}} = \frac{1}{\pi}\arccos(\rho_{\mathfrak{K}}). \tag{12}$$

From the above discussion, the QVTPM-based neural synchronization can be viewed as a stochastic process consisting of attractive steps and repulsive steps. Write the probability of the event of an attractive step and that of a repulsive step by $P_a(\bullet)$ and $P_r(\bullet)$, and the average step size of an attractive step and a repulsive step by $\Delta\rho_a(\bullet)$ and $\Delta\rho_r(\bullet)$, respectively. By jointly considering the two types of steps, the average change of overlaps can be componentwisely defined by

$$\begin{aligned}
\Delta\rho(\rho_{\mathfrak{R}}) &= P_a(\rho_{\mathfrak{R}})\Delta\rho_a(\rho_{\mathfrak{R}}) + P_r(\rho_{\mathfrak{R}})\Delta\rho_r(\rho_{\mathfrak{R}}),\\
\Delta\rho(\rho_{\mathfrak{T}}) &= P_a(\rho_{\mathfrak{T}})\Delta\rho_a(\rho_{\mathfrak{T}}) + P_r(\rho_{\mathfrak{T}})\Delta\rho_r(\rho_{\mathfrak{T}}),\\
\Delta\rho(\rho_{\mathfrak{J}}) &= P_a(\rho_{\mathfrak{J}})\Delta\rho_a(\rho_{\mathfrak{J}}) + P_r(\rho_{\mathfrak{J}})\Delta\rho_r(\rho_{\mathfrak{J}}),\\
\Delta\rho(\rho_{\mathfrak{K}}) &= P_a(\rho_{\mathfrak{K}})\Delta\rho_a(\rho_{\mathfrak{K}}) + P_r(\rho_{\mathfrak{K}})\Delta\rho_r(\rho_{\mathfrak{K}}).
\end{aligned} \tag{13}$$

According to [41], once $\Delta\rho(\bullet) > 0$, the synchronization time grows at a polynomial rate with the increase of L; otherwise, at an exponential rate. Thus, QVTPM is secure if and only if it satisfies the following two conditions.

Condition I. For A and B, during the synchronization process, there holds that

$$\Delta\rho(\rho) > 0, \;\; \rho \in (0,1). \tag{14}$$

Condition II. For E, there exists a region G in $(0,1)$, such that

$$\Delta\rho(\rho) < 0, \;\; \rho \in G. \tag{15}$$

In this paper, we only consider Condition I. By substituting (13) into (14) we have

$$P_a(\rho_{\mathfrak{R}})\Delta\rho_a(\rho_{\mathfrak{R}}) + P_r(\rho_{\mathfrak{R}})\Delta\rho_r(\rho_{\mathfrak{R}}) > 0, \tag{16}$$

where $\Delta\rho_a(\rho_{\mathfrak{R}})$ and $\Delta\rho_r(\rho_{\mathfrak{R}})$ depend on and can be obtained from the motion equations which are introduced in [33,34]. According to (12), there is a one-to-one correspondence between $\rho_{\mathfrak{R}}$ and $\varepsilon_{\mathfrak{R}}$; thus, (16) can also be described with respect to $\varepsilon_{\mathfrak{R}}$. Let $U(\varepsilon_{\mathfrak{R}}) = -\frac{\Delta\rho_a(\varepsilon_{\mathfrak{R}})}{\Delta\rho_r(\varepsilon_{\mathfrak{R}})}$ and $R(\varepsilon_{\mathfrak{R}}) = \frac{P_r(\varepsilon_{\mathfrak{R}})}{P_a(\varepsilon_{\mathfrak{R}})}$, and then (16) is equivalent to

$$U(\varepsilon_{\mathfrak{R}}) > R(\varepsilon_{\mathfrak{R}}), \tag{17}$$

where $\varepsilon_{\mathfrak{R}} \in [0, 0.5]$. According to the definitions of attractive step and repulsive step, the transition probabilities $P_r(\varepsilon_{\mathfrak{R}})$ and $P_a(\varepsilon_{\mathfrak{R}})$ take the forms

$$\begin{aligned} P_a(\varepsilon_{\mathfrak{R}}) &= P\left(\mathfrak{R}\left(\tau^{A/B}\right) = \mathfrak{R}\left(\sigma_u^A\right) = \mathfrak{R}\left(\sigma_u^B\right) \middle| \mathfrak{R}\left(\tau^A\right) = \mathfrak{R}\left(\tau^B\right)\right), \\ P_r(\varepsilon_{\mathfrak{R}}) &= P\left(\mathfrak{R}\left(\sigma_u^A\right) \neq \mathfrak{R}\left(\sigma_u^B\right) \middle| \mathfrak{R}\left(\tau^A\right) = \mathfrak{R}\left(\tau^B\right)\right). \end{aligned} \tag{18}$$

Let $P_e(\varepsilon_{\mathfrak{R}}) = P\left(\mathfrak{R}\left(\tau^A\right) = \mathfrak{R}\left(\tau^B\right)\right)$, then

$$\begin{aligned} P_a(\varepsilon_{\mathfrak{R}}) &= \frac{1}{2P_e}\sum_{l=0}^{(K-1)/2}\binom{K-1}{2l}(1-\varepsilon_{\mathfrak{R}})^{K-2l}\varepsilon_{\mathfrak{R}}^{2l}, \\ P_r(\varepsilon_{\mathfrak{R}}) &= \frac{1}{P_e}\sum_{l=1}^{K/2}\binom{K-1}{2l-1}(1-\varepsilon_{\mathfrak{R}})^{K-2l}\varepsilon_{\mathfrak{R}}^{2l}, \\ P_e = P_e(\varepsilon_{\mathfrak{R}}) &= \sum_{l=0}^{K/2}\binom{K}{2l}(1-\varepsilon_{\mathfrak{R}})^{K-2l}\varepsilon_{\mathfrak{R}}^{2l}. \end{aligned} \tag{19}$$

Taking $K = 3$, which used to be a common choice for neural cryptosystem [42], we have

$$\begin{aligned} P_a(\varepsilon_{\mathfrak{R}}) &= \frac{\frac{1}{2}(1-\varepsilon_{\mathfrak{R}})^3 + \frac{1}{2}(1-\varepsilon_{\mathfrak{R}})\varepsilon_{\mathfrak{R}}^2}{(1-\varepsilon_{\mathfrak{R}})^3 + 3(1-\varepsilon_{\mathfrak{R}})\varepsilon_{\mathfrak{R}}^2}, \\ P_r(\varepsilon_{\mathfrak{R}}) &= \frac{2(1-\varepsilon_{\mathfrak{R}})\varepsilon_{\mathfrak{R}}^2}{(1-\varepsilon_{\mathfrak{R}})^3 + 3(1-\varepsilon_{\mathfrak{R}})\varepsilon_{\mathfrak{R}}^2}. \end{aligned} \tag{20}$$

As a result, we have

$$R(\varepsilon_{\mathfrak{R}}) = \frac{P_r(\varepsilon_{\mathfrak{R}})}{P_a(\varepsilon_{\mathfrak{R}})} = \frac{4\varepsilon_{\mathfrak{R}}^2}{(1-\varepsilon_{\mathfrak{R}})^2 + \varepsilon_{\mathfrak{R}}^2}. \tag{21}$$

During the bidirectional synchronization for TPM, $\Delta\rho(\rho)$ is always positive until the process almost reaches an absorbing state at $\rho = 1$. When $\rho_{\mathfrak{R}} \to 1$ ($\varepsilon_{\mathfrak{R}} \to 0$), based on the motion equations we can obtain $U(\varepsilon_{\mathfrak{R}}) \sim (7/12)\pi^2\varepsilon_{\mathfrak{R}}^2$ [33,34]. Thus, we have

$$\lim_{\varepsilon_{\mathfrak{R}}\to 0}\frac{R(\varepsilon_{\mathfrak{R}})}{U(\varepsilon_{\mathfrak{R}})} = \frac{48}{7\pi^2} < 1. \tag{22}$$

Similarly, we also have

$$\lim_{\varepsilon_{\mathfrak{T}}\to 0}\frac{R(\varepsilon_{\mathfrak{T}})}{U(\varepsilon_{\mathfrak{T}})} < 1, \quad \lim_{\varepsilon_{\mathfrak{J}}\to 0}\frac{R(\varepsilon_{\mathfrak{J}})}{U(\varepsilon_{\mathfrak{J}})} < 1, \quad \lim_{\varepsilon_{\mathfrak{K}}\to 0}\frac{R(\varepsilon_{\mathfrak{K}})}{U(\varepsilon_{\mathfrak{K}})} < 1. \tag{23}$$

Thus, Condition I is satisfied for $\rho_{\mathfrak{R}}$, $\rho_{\mathfrak{T}}$, $\rho_{\mathfrak{J}}$, and $\rho_{\mathfrak{K}}$.

Remark 3.1. *According to the above discussion, the synchronization process of the real part (three imaginary parts) of QVTPM evolves just like that of TPM. Thus, the success probability* $P_E^{\mathfrak{R}}$ $\left(P_E^{\mathfrak{T}}, P_E^{\mathfrak{J}}, P_E^{\mathfrak{K}}\right)$ *of an attack on the real part (three imaginary parts) of QVTPM is the same as the success probability* P_E^{TPM} *of an attack on TPM. Recalling*

that the synchronization processes of the real part and the three imaginary parts are independent of each other, the success probability for an attacker to know the group keys at synchronization time can then be computed by

$$P_E^{QVTPM} = P_E^{\mathfrak{R}} P_E^{\mathfrak{T}} P_E^{\mathfrak{J}} P_E^{\mathfrak{K}},$$

which is less than P_E^{TPM}*. This means that QVTPM is more secure than TPM of the same structure. Similarly, we can also deduce that QVTPM is more secure than CVTPM.*

4. **Synchronization Time of QVTPM.** In this section, we discuss the synchronization time of QVTPM. According to [42], $P_a(\bullet)$ and $P_r(\bullet)$ are immune to the change of L, while the step sizes $\Delta\rho_a(\bullet)$ and $\Delta\rho_r(\bullet)$ increase in proportional to L^{-2}. So $\Delta\rho(\rho_{\mathfrak{R}})$, $\Delta\rho(\rho_{\mathfrak{T}})$, $\Delta\rho(\rho_{\mathfrak{J}})$ and $\Delta\rho(\rho_{\mathfrak{K}})$ also increase in proportional to L^{-2}. Thus, the synchronization time of the real part and these of the three imaginary parts obey that

$$\begin{aligned} t_{\mathfrak{R}}^{sync} &\propto \frac{1}{\Delta\rho(\rho_{\mathfrak{R}})} \propto L^2, \quad t_{\mathfrak{T}}^{sync} \propto \frac{1}{\Delta\rho(\rho_{\mathfrak{T}})} \propto L^2, \\ t_{\mathfrak{J}}^{sync} &\propto \frac{1}{\Delta\rho(\rho_{\mathfrak{J}})} \propto L^2, \quad t_{\mathfrak{K}}^{sync} \propto \frac{1}{\Delta\rho(\rho_{\mathfrak{K}})} \propto L^2, \end{aligned} \tag{24}$$

which indicate that the synchronization time of QVTPM

$$t_{QVTPM}^{sync} \propto \max\left\{\frac{1}{\Delta\rho(\rho_{\mathfrak{R}})}, \frac{1}{\Delta\rho(\rho_{\mathfrak{T}})}, \frac{1}{\Delta\rho(\rho_{\mathfrak{J}})}, \frac{1}{\Delta\rho(\rho_{\mathfrak{K}})}\right\} \propto L^2. \tag{25}$$

Remark 4.1. *According to (25), we can observe that the synchronization time of QVTPM is at the same order of magnitude as TPM and CVTPM. Thus our proposed QVTPM model gains the enhanced security without increasing the order of synchronization time.*

5. **Simulations.** In this section, several numerical experiments are conducted to illustrate the performance of the QVTPM. As mentioned in Section 3, we set $K = 3$. Without losing generality, we take $N = 1000$ in our simulation (other choices of N will lead to the similar simulation results). The learning rule used during the experiments is the random walk learning rule.

Example 5.1. *In order to describe the degree of synchronization, we define the Euclidean distance as follows:*

$$ED = \sum_{u=1}^{K}\sum_{v=1}^{N} \left\|\omega_{u,v}^{A} - \omega_{u,v}^{B}\right\|. \tag{26}$$

Firstly, we take the synaptic depth $L = 9$*. In Figure 3, the horizontal axis indicates the number of iterations and the vertical axis indicates the Euclidean distance. It can be observed that the iteration numbers to attain the synchronization of QVTPM, CVTPM and TPM are in the same order of magnitude.*

Secondly, we compare the running time and the number of iterations to attain the synchronization for TPM, CVTPM, and QVTPM when increasing the synaptic depth L*.*

The comparison results are illustrated in Figure 4, where the horizontal axis indicates the synaptic depth, and the vertical axis indicates the running time in (a) and number of iterations in (b). It can be observed that the running time and the number of iterations of QVTPM grow at a polynomial rate with the increasing of L*. It is of the same order of magnitude as CVTPM and TPM. This coincides with Remark 4.1.*

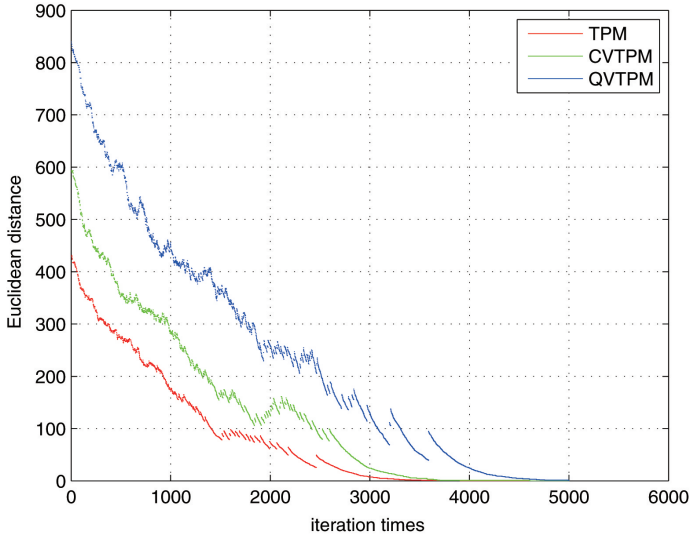
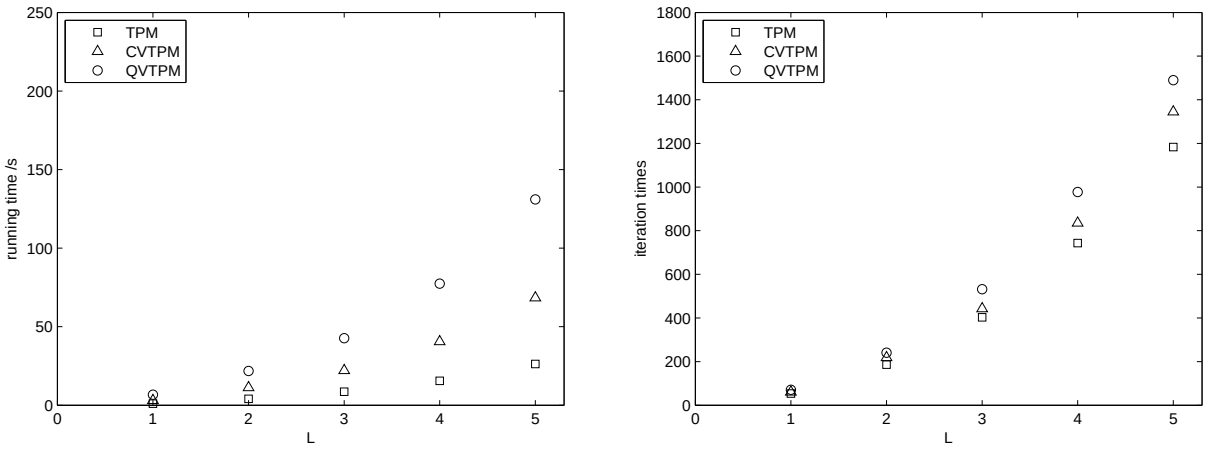


FIGURE 3. Euclidean distance between weight vectors of TPM, CVTPM, and QVTPM



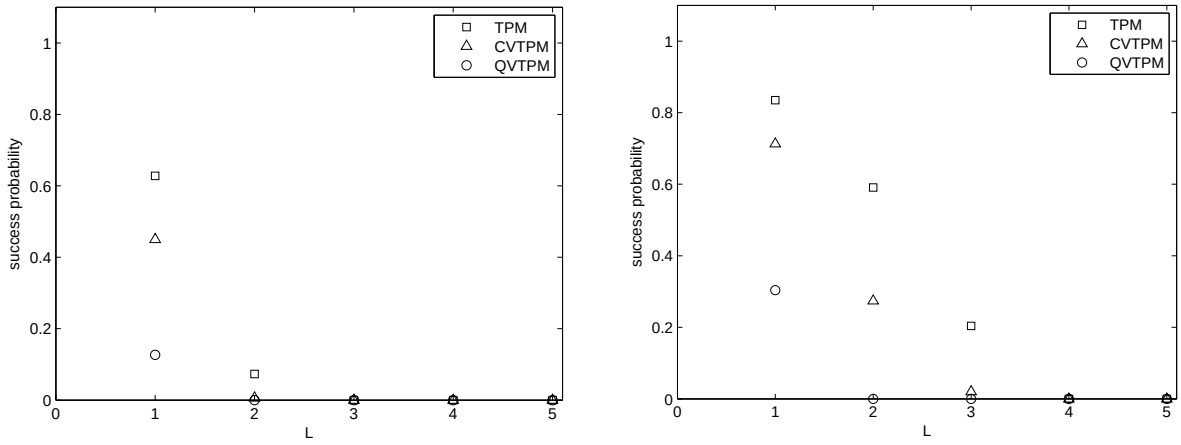
(a) Running time

(b) Number of iterations

FIGURE 4. Running time and number of iterations of TPM, CVTPM, and QVTPM

Example 5.2. *In this example, we numerically investigate the security of QVTPM under simple attacks and geometric attacks [31,33,42]. The success probabilities of simple attack and geometric attack on QVTPM, CVTPM and TPM are illustrated in Figure 5, where the horizontal axis indicates the synaptic depth and the vertical axis indicates the success probability P_E. Here we define a successful attack when the attacker knows 90 percent of the weights at synchronization time. We can observe from Figure 5 that the success probabilities of both the simple attack and the geometric attack on QVTPM are lower than those on TPM and CVTPM, which justifies our statement in Remark 3.1.*

Example 5.3. *In this example, we compare the synchronization time and the security of QVTPM with those of TPM and CVTPM when their secret keys to be generated are of the same length. The number of the input neurons for QVTPM, CVTPM and TPM are 1000, 2000, and 4000 respectively. It can be observed from Figure 6 that the synchronization*



(a) Simple attack (b) Geometric attack

FIGURE 5. Comparison of the success probabilities of the simple attack and geometric attack on TPM with those of CVTPM and QVTPM

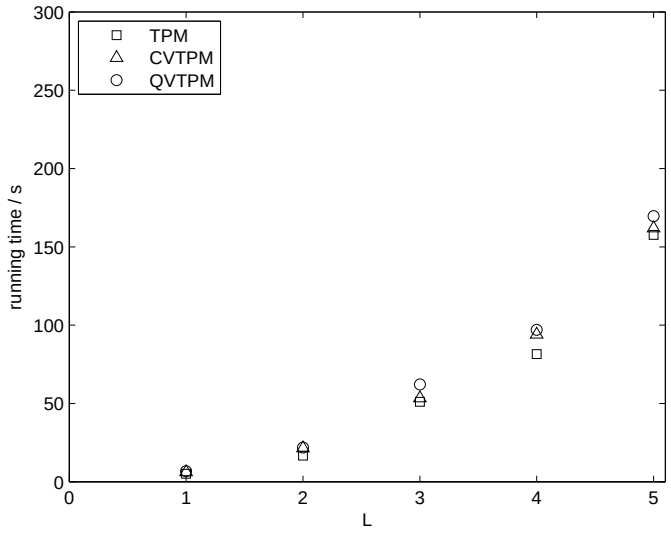
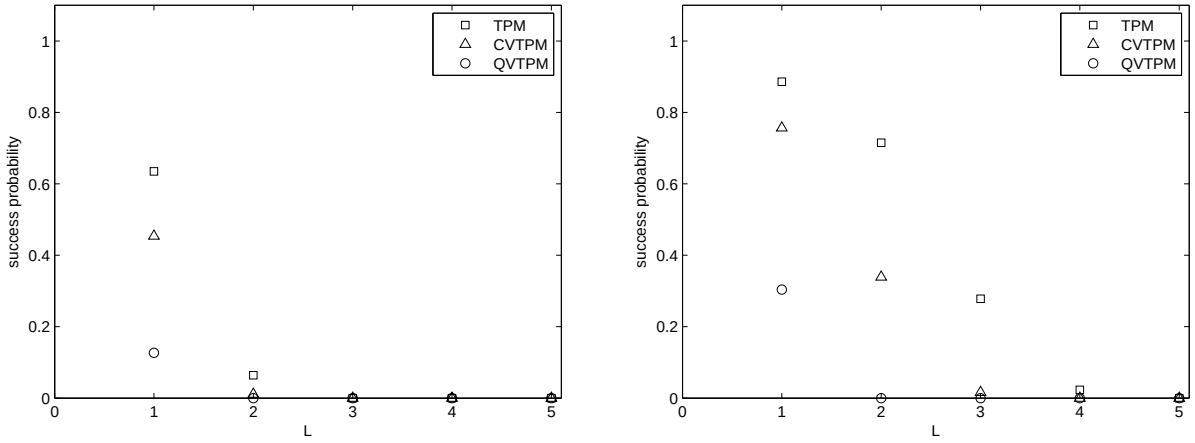


FIGURE 6. Running time of the TPM, CVTPM and QVTPM for generating keys whose length is 4000

times for QVTPM, CVTPM and TPM are very close. As shown by Figure 7, QVTPM is more secure than CVTPM and TPM in terms of the success probabilities of the simple attack and geometric attack. This validates our theoretical analysis.

6. **Conclusion.** In this paper, we proposed a neural cryptography model based on the quaternion-valued tree parity machine (QVTPM) and established the corresponding mutual learning rule. The security analysis revealed that the proposed model is more secure in theory than its real counterpart TPM and complex-valued counterpart CVTPM, while keeping the same order of the synchronization time as the two counterparts. Numerical examples verified the advantages of the proposed models and validated the theoretical analysis.



(a) Simple attack (b) Geometric attack

FIGURE 7. Success probabilities of the simple attack and geometric attack of the TPM, CVTPM and QVTPM for generating keys whose length is 4000

Acknowledgment. This work is supported by the National Natural Science Foundation of China (No. 61671099).

REFERENCES

[1] W. Diffie and M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory*, vol.22, no.6, pp.644-654, 1976.

[2] M. Rosen-Zvi, I. Kanter and W. Kinzel, Cryptography based on neural networks analytical results, *Journal of Physics A: Mathematical and General*, vol.35, no.47, pp.707-713, 2002.

[3] S. Jeong, C. Park, D. Hong, C. Seo and N. Jho, Neural cryptography based on generalized tree parity machine for real-life systems, *Security and Communication Networks*, vol.2021, no.11, pp.1-12, 2021.

[4] A. Sarkar, Neural cryptography using optimal structure of neural networks, *Applied Intelligence*, vol.51, pp.8057-8066, 2021.

[5] Y. Choi, J. Sim and L. S. Kim, CREMON: Cryptography embedded on the convolutional neural network accelerator, *IEEE Transactions on Circuits and Systems-II: Express Briefs*, vol.67, no.12, pp.3337-3341, 2020.

[6] M. Niemiec, Error correction in quantum cryptography based on artificial neural networks, *Quantum Information Processing*, vol.18, no.174, pp.1-18, 2019.

[7] J. Wang, L. M. Cheng and T. Su, Multivariate cryptography based on clipped Hopfield neural network, *IEEE Transactions on Neural Networks and Learning Systems*, vol.29, no.2, pp.353-363, 2018.

[8] X. Duan, D. Guo and N. Liu, A new high capacity image steganography method combined with image elliptic curve cryptography and deep neural network, *IEEE Access*, vol.8, pp.25777-25788, 2020.

[9] Z. Ni and S. Paul, A multistage game in smart grid security: A reinforcement learning solution, *IEEE Transactions on Neural Networks and Learning Systems*, vol.30, no.9, pp.2684-2695, 2019.

[10] A. Sarkar, Secure exchange of information using artificial intelligence and chaotic system guided neural synchronization, *Multimedia Tools and Applications*, vol.80, pp.18211-18241, 2021.

[11] Y. L. Han, Y. Li, Z. Li and S. S. Zhu, An improved method to evaluate the synchronization in neural key exchange protocol, *Security and Communication Networks*, vol.2020, no.2, pp.1-10, 2020.

[12] Y. L. Han, Y. Li and Z. Li, A key exchange optimization scheme based on tree parity machine, *Journal of Electronics and Information Technology*, vol.43, no.8, pp.2140-2148, 2021.

[13] A. Sarkar, Mutual learning-based efficient synchronization of neural networks to exchange the neural key, *Complex and Intelligent Systems*, pp.1-15, 2021.

[14] A. Sarkar, M. Z. Khan, M. M. Singh, A. Noorwali, C. Chakraborty and S. K. Pani, Artificial neural synchronization using nature inspired whale optimization, *IEEE Access*, vol.9, pp.16435-16447, 2021.
[15] S. Lakshmanan, M. Prakash, C. P. Lim and R. Rakkiyappan, Synchronization of an inertial neural network with time-varying delays and its application to secure communication, *IEEE Transactions on Neural Networks and Learning Systems*, vol.29, no.1, pp.195-207, 2018.
[16] A. Sarkar, Neural synchronization of optimal structure-based group of neural networks, *Neurocomputing*, vol.450, pp.156-167, 2021.
[17] H. Chen, P. Shi and C. C. Lim, Cluster synchronization for neutral stochastic delay networks via intermittent adaptive control, *IEEE Transactions on Neural Networks and Learning Systems*, vol.30, pp.11, pp.3246-3259, 2019.
[18] P. Liu, Z. Zeng and J. Wang, Global synchronization of coupled fractional-order recurrent neural networks, *IEEE Transactions on Neural Networks and Learning Systems*, vol.30, no.8, pp.2358-2368, 2019.
[19] 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 Transactions on Neural Networks and Learning Systems*, vol.30, no.8, pp.2434-2448, 2019.
[20] 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 Transactions on Neural Networks and Learning Systems*, vol.30, no.6, pp.1854-1866, 2019.
[21] Z. Zhang and J. Cao, Novel finite-time synchronization criteria for inertial neural networks with time delays via integral inequality method, *IEEE Transactions on Neural Networks and Learning Systems*, vol.30, no.5, pp.1476-1485, 2019.
[22] S. Prasomphan, Towards cultural heritage content retrieval by convolution neural network, *ICIC Express Letters*, vol.16, no.2, pp.137-144, 2022.
[23] X. Chen, Y. Wang, H. Dong, X. Pan and J. Li, Network representation learning based on random walk of connection number, *International Journal of Innovative Computing, Information and Control*, vol.18, no.3, pp.883-900, 2022.
[24] M. Morita, Q. Huang and M. Morishita, An update rule of parameters of online-offline integrated learning method of neural network control, *International Journal of Innovative Computing, Information and Control*, vol.18, no.3, pp.667-685, 2022.
[25] I. Kanter, W. Kinzel and E. Kanter, Secure exchange of information by synchronization of neural networks, *Europhysics Letters*, vol.57, no.1, pp.1-11, 2002.
[26] M. Rosen-Zvi, E. Klein, I. Kanter and W. Kinzel, Mutual learning in a tree parity machine and its application to cryptography, *Physical Review E*, vol.66, pp.1-14, 2002.
[27] A. Klimov, A. Mityagin and A. Shamir, Analysis of neural cryptography, *Lecture Notes in Computer Science*, vol.2501, pp.288-298, 2002.
[28] L. N. Shacham, E. Klein, R. Mislovaty, I. Kanter and W. Kinzel, Cooperating attackers in neural cryptography, *Physical Review E*, vol.69, 2004.
[29] A. Ruttor, W. Kinzel, R. Naeh and I. Kanter, Genetic attack on neural cryptography, *Physical Review E*, vol.73, 2006.
[30] W. Kinzel and I. Kanter, Disorder generated by interacting neural networks: Application to econophysics and cryptography, *Journal of Physics A: Mathematical and General*, vol.36, no.43, 2003.
[31] W. Kinzel and I. Kanter, Interacting neural networks and cryptography, *Advances in Solid State Physics*, vol.42, pp.383-391, 2002.
[32] W. Kinzel, Theory of interacting neural networks, in *Handbook of Graphs and Networks*, Wiley VCH, Berlin, 2005.
[33] N. Mu and X. Liao, Approach to design neural cryptography: A generalized architecture and a heuristic rule, *Physical Review E*, vol.87, 2013.
[34] X. Lei, X. Liao, F. Chen and T. Huang, Two-layer tree-connected feedforward neural network model for neural cryptography, *Physical Review E*, vol.87, 2013.
[35] A. M. Allam and H. M. Abbas, On the improvement of neural cryptography using erroneous transmitted information with error prediction, *IEEE Transactions on Neural Networks*, vol.21, no.12, pp.1915-1924, 2010.
[36] O. M. Reyes, I. Kopitzke and K. Zimmermann, Permutation parity machines for neural synchronization, *Journal of Physics A: Mathematical and Theoretical*, vol.42, no.19, 2009.
[37] L. F. Seoane and A. Ruttor, Successful attack on permutation-parity-machine-based neural cryptography, *Physical Review E*, vol.85, 2012.

[38] T. Dong and T. Huang, Neural cryptography based on complex-valued neural network, *IEEE Transactions on Neural Networks and Learning Systems*, vol.31, no.11, pp.4999-5004, 2020.
[39] T. Dong, A. Wang, H. Zhu and X. Liao, Event-triggered synchronization for reaction-diffusion complex networks via random sampling, *Physica A: Statistical Mechanics and Its Applications*, vol.495, pp.454-462, 2018.
[40] A. Wang, T. Dong and X. Liao, Event-triggered synchronization strategy for complex dynamical networks with the Markovian switching topologies, *Neural Networks*, vol.74, pp.52-57, 2016.
[41] A. Ruttor, W. Kinzel and I. Kanter, Dynamics of neural cryptography, *Physical Review E*, vol.75, 2007.
[42] A. Ruttor, *Neural Synchronization and Cryptography*, Ph.D. Thesis, Bavaria Würzburg, Julius Maximilians University, 2007.
[43] A. Ruttor, G. Reents and W. Kinzel, Synchronization of random walks with reflecting boundaries, *Journal of Physics A: Mathematical and General*, vol.37, no.36, pp.1-10, 2004.

Author Biography

Ying Zhang received the M.S. degree from Northeast Normal University in 2003 and Ph.D. degree from Dalian Maritime University in 2010. She is currently an associate professor at Dalian Maritime University. Her research interests include cryptography and neural networks.

Weihua Wang received the bachelor degree from Dalian Minzu University in 2019. She is currently pursuing her Master degree at Dalian Maritime University. Her research interests include neural networks and cryptography.

Huisheng Zhang received the M.S. degree from Xiamen University in 2003 and Ph.D. degree from Dalian University of Technology in 2009. From April 2014 to March 2015, he was financially supported by China Scholarship Council (CSC) to work as a visiting scholar at the Imperial College London (ICL), UK. He is currently a professor of Dalian Maritime University. His research interests include neural networks, signal processing and learning theory.