Approach to design neural cryptography: A generalized architecture and a heuristic rule

Nankun Mu,[1,*] Xiaofeng Liao,[1,†] and Tingwen Huang[2,‡]
[1]*College of Computer Science, Chongqing University, Chongqing, 400044, China*
[2]*Texas A&M University at Qatar, Doha, P.O. Box 23874, Qatar*
(Received 27 February 2013; published 12 June 2013)

Neural cryptography, a type of public key exchange protocol, is widely considered as an effective method for sharing a common secret key between two neural networks on public channels. How to design neural cryptography remains a great challenge. In this paper, in order to provide an approach to solve this challenge, a generalized network architecture and a significant heuristic rule are designed. The proposed generic framework is named as tree state classification machine (TSCM), which extends and unifies the existing structures, i.e., tree parity machine (TPM) and tree committee machine (TCM). Furthermore, we carefully study and find that the heuristic rule can improve the security of TSCM-based neural cryptography. Therefore, TSCM and the heuristic rule can guide us to designing a great deal of effective neural cryptography candidates, in which it is possible to achieve the more secure instances. Significantly, in the light of TSCM and the heuristic rule, we further expound that our designed neural cryptography outperforms TPM (the most secure model at present) on security. Finally, a series of numerical simulation experiments are provided to verify validity and applicability of our results.

DOI: 10.1103/PhysRevE.87.062804 PACS number(s): 84.35.+i, 05.45.Xt, 05.40.Fb

I. INTRODUCTION

Public key exchange protocols (PKEPs) have played an important role in modern cryptography since initially introduced by Diffie and Hellman [1]. Usually, PKEPs can enable two parties, named A and B, to share a common secret key on public channel, while an attacker E cannot retrieve the key, even if equipped with the ability to invade the communication channel. Then, the key can be utilized to deal with some cryptographic problems, such as privacy, authentication, and data integrity, to name a few. In particular, PKEPs based on number theory have been extensively studied [1–4]. However, more recently, with the broad application of neural networks [5,6], it has been recognized that neural synchronization is able to achieve the same objective, bringing about what is known as *neural cryptography* [7]. The mechanism behind neural cryptography is similar to that of "secret key agreement through public discussion" [8]. In particular, benefited from the absence of large-scale computation, which is highly suitable for the small-scale embedded systems [9,10], neural cryptography has gained considerable attention and has also been an increasingly important application of neural networks. However, by substituting for the neural networks, other synchronization systems, such as chaotic maps and coupled lasers [11–14], can also be exploited in constructing the similar PKEPs.

Neural cryptography requires that both A and B hold uniform structured networks to achieve synchronization by means of online learning [15]. At the beginning of the synchronization process, the two networks randomly initialize their discrete weighted vector, denoted by $\mathbf{w}^{\mathrm{A}}$ and $\mathbf{w}^{\mathrm{B}}$, which are kept secret, respectively. In each learning step, A and B calculate their outputs according to a common input vector and then exchange the outputs with each other. If they achieve the same outputs, their weighted vectors will be updated by a certain learning rule. After a finite iteration of such learning steps, synchronization is achieved eventually, which means $\mathbf{w}^{\mathrm{A}} = \mathbf{w}^{\mathrm{B}}$. And then the identical weighted vector $\mathbf{w}^{\mathrm{A/B}}$ can represent secret key between A and B. Clearly, the update behaviors of A and B are influenced by the outputs of each other; therefore, such learning steps correspond to mutual learning. On the other hand, an attacker E can train the third network using the examples consisting of input vectors and the outputs. Note that E cannot influence on the update behavior of A or B, so the learning steps of E belong to unidirectional learning. Because the synchronization speed of unidirectional learning cannot keep pace with that of mutual learning, E cannot achieve timely synchronization and, thus, the key cannot be recovered.

Based on the above mechanism, a number of neural cryptographies have been proposed; for example, permutation parity machine (PPM) [16,17], TPM [7], and TCM [18] (neural cryptography is often represented by the name of the structure). Meanwhile, a probabilistic attack algorithm targeted for PPM with a high success rate has also been presented [19] and the TCM is confirmed as an insecure case [18]. However, only TPM containing three hidden units ($K = 3$) can resist several kinds of attacks through increasing the synaptic depth of its networks [20–22]. Then, applicability and security of TPM ($K = 3$) have been carefully studied [23–30]. In Ref. [23], three similar learning rules are analyzed in detail, namely Hebbian learning rule, anti-Hebbian learning rule, and random-walk learning rule. The dynamical process of neural synchronization in TPM ($K = 3$) has been carefully studied [24]. Moreover, the model of the classical ruin problem is used to examine the average synchronization time of TPM ($K = 3$) [25]. In Ref. [26], in order to guarantee relevant input vectors partially unavailable to the attacker, a feedback algorithm is designed. In Ref. [27], in order to speed up the synchronization, the author introduced a queries algorithm, which is based on generating input vectors by queries instead of

*nankun.mu@qq.com
†xfliao@cqu.edu.cn
‡tingwen.huang@qatar.tamu.edu

randomizer. Recently, an error prediction algorithm called "Do not trust my partner"(DTMP) was presented [28]. It largely relies on one party sending some erroneous bits while the other is capable of predicting and correcting some corresponding errors. Meanwhile, Four kinds of attack algorithms were also experimentally investigated in detail, i.e., simple attack [7], geometric attack [29], majority attack [30], and genetic attack [23].

However, how to design neural cryptography is not a solved problem yet. Motivated by this situation, this paper establishes a practical approach, in which we can formulate instances on a large scale. In this way, the more secure neural cryptography, hopefully, can be designed.

The main contributions of this paper are as follows:

(i) A generalized architecture of neural cryptography named as TSCM is put forward. The proposed framework can be utilized as a guide in designing neural cryptography.

(ii) A significant heuristic rule is further provided by analyzing the conditions for the security of TSCM-based neural cryptography. Taking advantage of the heuristic rule, the security of such neural cryptography can be improved effectively.

(iii) Several new instances are presented in the light of TSCM and the Heuristic Rule. In addition, one of these cases is proven to be better than TPM ($K = 3$) on security.

The remainder of this paper is organized as follows. Section II shows the TSCM and the description of mutual learning algorithm. In Sec. III, the heuristic rule is proposed by investigating the security of neural cryptography. Then, we formulate several instances and illustrate the security of the proposed neural cryptography by simulation experiment in Sec. IV. Conclusions are presented in Sec. V.

II. GENERALIZED ARCHITECTURE AND MUTUAL LEARNING

Neural cryptography consists of network architecture and mutual learning algorithm.

A. The description of TSCM

Figure 1 is the principle graph of TSCM. It can be regarded as a tree-connected network consisting of three layers. More precisely, a TSCM has K hidden units, $K \times N$ input neurons, and a unique output neuron (we refer to the output neuron as state classifier). Each hidden unit works like an independent

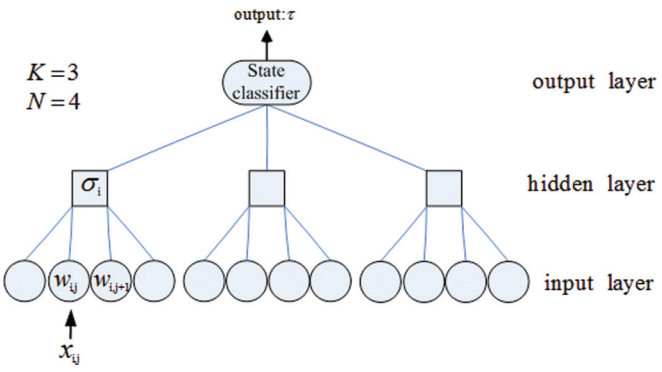

FIG. 1. (Color online) A TSCM network with $K = 3$, $N = 4$.

TABLE I. State classifier of TPM ($K = 3$).

Name	τ	d	State vector σ
TPM	τ_1	$+1$	$(+1,+1,+1),(+1,-1,-1)$ $(-1,+1,-1),(-1,-1,+1)$
	τ_2	-1	$(-1,-1,-1),(-1,+1,+1)$ $(+1,-1,+1),(+1,+1,-1)$

perceptron, and elements of the weighted vector are integral numbers and take values as

$$w_{i,j} \in \{-L, \ldots, 0, \ldots, +L\}. \tag{1}$$

Here, L represents synaptic depth of the networks; the index $i = 1, \ldots, K$ denotes the ith hidden unit of the networks, and $j = 1, \ldots, N$ denotes the jth input neuron for each hidden unit. Meanwhile, input vector is indicated by $\mathbf{x}$ and its elements are binary; i.e.,

$$x_{i,j} \in \{-1, +1\}. \tag{2}$$

When TSCM receives an input vector, the value of the ith hidden unit is defined by:

$$\begin{aligned} \sigma_i &= \operatorname{sgn}(h_i), \\ h_i &= \frac{1}{\sqrt{N}} \mathbf{w}_i \mathbf{x}_i = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} w_{i,j} x_{i,j}. \end{aligned} \tag{3}$$

In order to ensure σ_i binary, $h_i = 0$ is mapped to $\sigma_i = -1$. Then the state vector σ is classified by the state classification; i.e.,

$$\tau = \text{StateClassifier}(\sigma). \tag{4}$$

Finally, the state classifier generates an output, denoted by τ. Generally, the τ is required to be marked by a directional flag d, i.e., $+1$ or -1. For the reason, readers can refer to the learning rules, which are described in detail in the next subsection.

In particular, TSCM generalizes and unifies the existing structures, i.e., TPM and TCM. When the state classifier is defined as Tables I and II, it is observed that TCM and TPM are two special cases of TSCM. In Ref. [31], the authors present several instances that combine TCM and TPM.

B. Mutual learning algorithm

The mutual learning algorithm of TSCM is illustrated as follows:

(1) The two parties A and B start with a uniform TSCM-based network and randomly choose weighted vectors $\mathbf{w}^{\mathrm{A}}$ and

TABLE II. State classifier of TCM ($K = 3$).

Name	τ	d	State vector σ
TCM	τ_1	$+1$	$(+1,+1,+1),(+1,+1,-1)$ $(+1,-1,+1),(-1,+1,+1)$
	τ_2	-1	$(-1,-1,-1),(-1,-1,+1)$ $(-1,+1,-1),(+1,-1,-1)$

$\mathbf{w}^{\mathrm{B}}$, which are kept secret. This can certainly guarantee the uncorrelation between $\mathbf{w}^{\mathrm{A}}$ and $\mathbf{w}^{\mathrm{B}}$ at the beginning.

(2) In each learning step, the two parties receive a common input vector $\mathbf{x}$ at the same time. Upon receiving $\mathbf{x}$, A and B obtain σ^{A} and σ^{B}, respectively. And then τ^{A} and τ^{B} can also be generated according to σ^{A} and σ^{B}, respectively. This computation process is well defined in the above subsection. Afterwards, τ^{A} and τ^{B} are exchanged with each other on the public channel, while σ^{A} and σ^{B} are secretly kept.

(3) All weights are iteratively adjusted by one of the following learning rules:

(a) Hebbian learning rule:

$$w_{i,j}^{\mathrm{A/B}} = g\left[w_{i,j}^{\mathrm{A/B}} + x_{i,j}d^{\mathrm{A/B}}\Theta(\sigma_i d^{\mathrm{A/B}})\Theta(\tau^{\mathrm{A}}\tau^{\mathrm{B}})\right]. \quad (5)$$

(b) Anti-Hebbian learning rule:

$$w_{i,j}^{\mathrm{A/B}} = g\left[w_{i,j}^{\mathrm{A/B}} - x_{i,j}d^{\mathrm{A/B}}\Theta(\sigma_i d^{\mathrm{A/B}})\Theta(\tau^{\mathrm{A}}\tau^{\mathrm{B}})\right]. \quad (6)$$

(c) Random-walk learning rule:

$$w_{i,j}^{\mathrm{A/B}} = g\left[w_{i,j}^{\mathrm{A/B}} + x_{i,j}\Theta(\sigma_i d^{\mathrm{A/B}})\Theta(\tau^{\mathrm{A}}\tau^{\mathrm{B}})\right]. \quad (7)$$

Here, function $g(w)$ is introduced to ensure that each element of the weighted vector $\mathbf{w}^{\mathrm{A/B}}$ remains in the range $[-L, +L]$. It is defined as

$$g(w) = \begin{cases} \mathrm{sgn}(w)L, & \text{for} \quad |w| > L; \\ w, & \text{otherwise.} \end{cases} \quad (8)$$

(4) Repeating procedure 2 and 3 until synchronization ($\mathbf{w}^{\mathrm{A}} = \mathbf{w}^{\mathrm{B}}$) is achieved. The final identical weighted vector $\mathbf{w}^{\mathrm{A/B}}$ can be used as the common secret key between A and B.

While in the whole synchronization process, the state vector $\sigma^{\mathrm{A/B}}$ is consistently inaccessible to any others. So, two possible real update behaviors can be defined as follows:

(i) An attractive step ($\tau^{\mathrm{A}} = \tau^{\mathrm{B}} = \sigma_i^{\mathrm{A/B}}$): The weighted vectors of the ith corresponding hidden units are updated in the same direction. And a series of such steps leads to synchronization eventually.

(ii) A repulsive step ($\tau^{\mathrm{A}} = \tau^{\mathrm{B}}, \sigma_i^{\mathrm{A}} \neq \sigma_i^{\mathrm{B}}$): Only one weighted vector of the ith corresponding hidden units in A or B is updated, while B or A remains unchanged. A sequence of these repulsive steps may reduce synchronization speed.

III. STUDY ON SECURITY

Required by a computationally secure system [1], a secure neural cryptography needs to hold [24]:

(i) Increasing the synaptic depth L of the networks, the averaged synchronization time by mutual learning for A and B grows at a polynomial rate.

(ii) Meanwhile, the averaged synchronization time by unidirectional learning for E grows at an exponential rate.

Hence, L can represent the security parameter of neural cryptography. And such neural cryptography can resist attacks by increasing L.

In this section, a heuristic rule is proposed by theoretically analyzing the different security between TPM ($K = 3$) and TCM ($K = 3$). It is worth paying special attention to the fact that TPM ($K = 3$) is a secure neural cryptography [29] and TCM ($K = 3$) is insecure [18]. The heuristic rule can enable this neural cryptography to meet the second point as much as possible.

For presentation convenience, the following computations are based on one fixed pair of corresponding hidden units.

The dynamics of synchronization process are of huge impact on the security of neural cryptography. The level of synchronization is indicated by the normalized overlap between the two corresponding hidden units [32]:

$$\rho = \frac{\mathbf{w}_i^A \mathbf{w}_i^B}{\sqrt{\mathbf{w}_i^A \mathbf{w}_i^A}\sqrt{\mathbf{w}_i^B \mathbf{w}_i^B}}, \quad \rho \in [0, +1]. \quad (9)$$

At the beginning of synchronization, ρ locates approximately at $\rho = 0$ because of the random initial weighted vectors. Through a series of learning steps, synchronization is achieved and ρ is stable at $\rho = +1$. In these learning steps, it is possible that the two corresponding hidden units have different σ. The probability of this event is defined by ε, which is known as the generation error [32]:

$$\varepsilon = \frac{1}{\pi}\arccos(\rho), \quad \varepsilon \in [0, 0.5]. \quad (10)$$

For studying the dynamics of ρ in the synchronization process, it is very necessary to introduce the average change of the overlap [24] in each learning step:

$$\langle \Delta\rho(\rho) \rangle = P_a(\rho)\langle \Delta\rho_a(\rho) \rangle + P_r(\rho)\langle \Delta\rho_r(\rho) \rangle. \quad (11)$$

Here, $\langle \Delta\rho_a(\rho) \rangle$ ($\langle \Delta\rho_r(\rho) \rangle$) denotes the average change size of ρ in each attractive (repulsive) step; $P_a(\rho)$ ($P_r(\rho)$) indicates the probability of the event that an attractive (repulsive) step occurs between one pair of corresponding hidden units.

Remarkably, $\langle \Delta\rho(\rho) \rangle > 0$ in (0,1) enables synchronization time grows at a polynomial rate with increasing L [24]; otherwise, at a exponential rate. So, a conclusion can be draw that a neural cryptography is secure, if and only if:

(a) Condition I. In synchronization process, for A and B,

$$\langle \Delta\rho(\rho) \rangle > 0, \quad \rho \in (0,1) \quad (12)$$

is true;

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

$$\langle \Delta\rho(\rho) \rangle < 0, \quad \rho \in G. \quad (13)$$

In particular, Condition II will not be considered in the present investigation, which is only focused on Condition I. Substituting Eq. (11) into Eq. (12) yields the following:

$$P_a(\rho)\langle \Delta\rho_a(\rho) \rangle + P_r(\rho)\langle \Delta\rho_r(\rho) \rangle > 0, \quad \rho \in (0,1). \quad (14)$$

In Ref. [33], the authors have noted that $\langle \Delta\rho_a(\rho) \rangle$ and $\langle \Delta\rho_r(\rho) \rangle$ are only dependent on the equations of motion, which are carefully introduced in Ref. [34]. According to the equations of motion, we can obtain two constant curves to represent $\langle \Delta\rho_a(\rho) \rangle$ and $\langle \Delta\rho_r(\rho) \rangle$ in Fig. 2.

Then, let us transform Eq. (14) to

$$-\frac{\langle \Delta\rho_a(\rho) \rangle}{\langle \Delta\rho_r(\rho) \rangle} > \frac{P_r(\rho)}{P_a(\rho)}, \quad \rho \in (0,1). \quad (15)$$

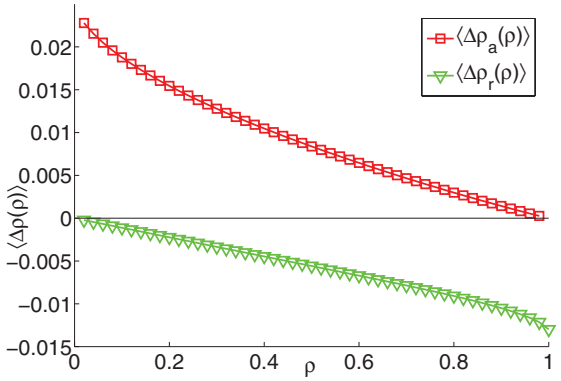

FIG. 2. (Color online) The dynamics of $\langle \Delta\rho_a(\rho)\rangle$ and $\langle \Delta\rho_r(\rho)\rangle$, while the lines have been calculated using equations of motion.

For the convenience of the following analysis, Eq. (15) can be rewritten as

$$\begin{aligned} &U(\rho) > R(\rho),\\ &U(\rho) = -\frac{\langle \Delta\rho_a(\rho)\rangle}{\langle \Delta\rho_r(\rho)\rangle},\\ &R(\rho) = \frac{P_r(\rho)}{P_a(\rho)},\\ &\rho \in (0,1). \end{aligned} \tag{16}$$

Here, $U(\rho)$ is constant unless changing the equations of motion. So, it can be easily obtained that the security of neural cryptography is all dependent on $R(\rho)$. Taking Eq. (10) to replace ρ in $R(\rho)$, we have

$$R(\varepsilon) = \frac{P_r(\varepsilon)}{P_a(\varepsilon)}, \quad \varepsilon \in [0,0.5]. \tag{17}$$

While $P_a(\varepsilon)$ and $P_r(\varepsilon)$ can be computed as

$$\begin{aligned} &P_a(\varepsilon) = P(\tau^{A/B} = \sigma^A = \sigma^A | \tau^A = \tau^B);\\ &P_r(\varepsilon) = P(\sigma^A \neq \sigma^A | \tau^A = \tau^B);\\ &P_u(\varepsilon) = P(\tau^A = \tau^B). \end{aligned} \tag{18}$$

Here, $P_u(\varepsilon)$ denotes the probability of the occurrence of an agreement on outputs between A and B. Then we describe the computational procedure of $P_u(\varepsilon)$ in TPM ($K = 3$). The following always holds:

$$P_u^{\mathrm{TPM}(K=3)}(k,\varepsilon) \begin{cases} P_u^{\mathrm{TPM}(K=3)}(0,\varepsilon) = \binom{3}{0}\varepsilon^0(1-\varepsilon)^3;\\ P_u^{\mathrm{TPM}(K=3)}(1,\varepsilon) = 0\binom{3}{1}\varepsilon^1(1-\varepsilon)^2;\\ P_u^{\mathrm{TPM}(K=3)}(2,\varepsilon) = \binom{3}{2}\varepsilon^2(1-\varepsilon)^1;\\ P_u^{\mathrm{TPM}(K=3)}(3,\varepsilon) = 0\binom{3}{3}\varepsilon^3(1-\varepsilon)^0. \end{cases} \tag{19}$$

Here, $P_u^{\mathrm{TPM}(K=3)}(k,\varepsilon)$ denotes the probability of the occurrence of an agreement on outputs between two TPM when k pairs of corresponding hidden units disagree. According to Eqs. (18) and (19), we can get

$$P_a^{\mathrm{TPM}(K=3)}(k,\varepsilon) \begin{cases} P_a^{\mathrm{TPM}(K=3)}(0,\varepsilon) = \frac{\frac{1}{2}\binom{3}{0}\varepsilon^0(1-\varepsilon)^3}{P_u^{\mathrm{TPM}(K=3)}(k,\varepsilon)};\\ P_a^{\mathrm{TPM}(K=3)}(1,\varepsilon) = 0;\\ P_a^{\mathrm{TPM}(K=3)}(2,\varepsilon) = \frac{\frac{1}{6}\binom{3}{2}\varepsilon^2(1-\varepsilon)^1}{P_u^{\mathrm{TPM}(K=3)}(k,\varepsilon)};\\ P_a^{\mathrm{TPM}(K=3)}(3,\varepsilon) = 0. \end{cases} \tag{20}$$

and

$$P_r^{\mathrm{TPM}(K=3)}(k,\varepsilon) \begin{cases} P_r^{\mathrm{TPM}(K=3)}(0,\varepsilon) = \frac{0\binom{3}{0}\varepsilon^0(1-\varepsilon)^3}{P_u^{\mathrm{TPM}(K=3)}(k,\varepsilon)};\\ P_r^{\mathrm{TPM}(K=3)}(1,\varepsilon) = 0;\\ P_r^{\mathrm{TPM}(K=3)}(2,\varepsilon) = \frac{\frac{2}{3}\binom{3}{2}\varepsilon^2(1-\varepsilon)^1}{P_u^{\mathrm{TPM}(K=3)}(k,\varepsilon)};\\ P_r^{\mathrm{TPM}(K=3)}(3,\varepsilon) = 0. \end{cases} \tag{21}$$

Here, $P_a^{\mathrm{TPM}(K=3)}(k,\varepsilon)$ [$P_r^{\mathrm{TPM}(K=3)}(k,\varepsilon)$] denotes the probability of the occurrence of an attractive (repulsive) step in TPM if k pairs of corresponding hidden units disagree. Sum them up to derive the $P_a^{\mathrm{TPM}(K=3)}(\varepsilon)$ and $P_r^{\mathrm{TPM}(K=3)}(\varepsilon)$:

$$\begin{aligned} P_a^{\mathrm{TPM}(K=3)}(\varepsilon) &= \frac{\frac{1}{2}(1-\varepsilon)^3 + \frac{1}{2}\varepsilon^2(1-\varepsilon)^2}{P_u^{\mathrm{TPM}(K=3)}(\varepsilon)};\\ P_r^{\mathrm{TPM}(K=3)}(\varepsilon) &= \frac{2\varepsilon^2(1-\varepsilon)}{P_r^{\mathrm{TPM}(K=3)}(\varepsilon)}. \end{aligned} \tag{22}$$

Then, it follows that

$$R^{\mathrm{TPM}(K=3)}(\varepsilon) = \frac{P_r^{\mathrm{TPM}(K=3)}(\varepsilon)}{P_a^{\mathrm{TPM}(K=3)}(\varepsilon)} = \frac{4\varepsilon^2}{(1-\varepsilon)^2+\varepsilon^2}. \tag{23}$$

Similarly, we can obtain:

$$R^{\mathrm{TCM}(K=3)}(\varepsilon) = \frac{\frac{1}{2}\varepsilon(1-\varepsilon)+\varepsilon^2}{\frac{3}{4}(1-\varepsilon)^2+\varepsilon(1-\varepsilon)+\frac{1}{2}\varepsilon^2}. \tag{24}$$

However, the equations of motion in TPM and TCM are the same, and this means $U^{\mathrm{TPM}}(\rho) = U^{\mathrm{TCM}}(\rho)$. Therefore, the difference on security between TPM ($K = 3$) and TCM ($K = 3$) is determined due to the different $R(\varepsilon)$.

Remarkably, simulation experiment displayed in Fig. 3 indicates that $\langle \Delta\rho(\rho)\rangle < 0 (U(\rho) > R(\varepsilon))$ is most likely to occur as $\rho \to 1, \varepsilon \to 0$, i.e., nearly synchronization. Inspired by Condition I, we can obtain:

Essential condition. Closing to synchronization, a secure TSCM also needs to satisfy

$$U(\rho) > R(\varepsilon), \quad \rho \to 1, \quad \varepsilon \to 0. \tag{25}$$

When $\rho \to 1, \varepsilon \to 0$, we can obtain the approximate values of $R^{\mathrm{TPM}}(\varepsilon)$ and $R^{\mathrm{TCM}}(\varepsilon)$. By Eqs. (23) and (24), as $\varepsilon \to 0$, it

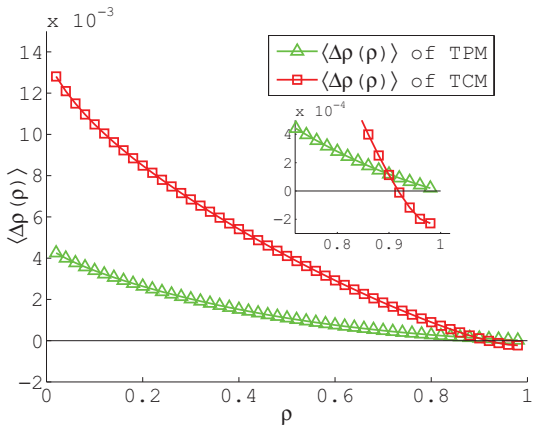

FIG. 3. (Color online) The dynamics of $\langle \Delta\rho(\rho)\rangle$ of TPM and TCM, while the lines have been calculated using Eq. (14).

holds that

$$R^{\mathrm{TPM}(K=3)}(\varepsilon) = \frac{4\varepsilon^2}{(1-\varepsilon)^2+\varepsilon^2} \sim \frac{4\varepsilon^2}{1+0} \sim 4\varepsilon^2, \tag{26}$$

and

$$R^{\mathrm{TCM}(K=3)}(\varepsilon) = \frac{\frac{1}{2}\varepsilon(1-\varepsilon)+\varepsilon^2}{\frac{3}{4}(1-\varepsilon)^2+\varepsilon(1-\varepsilon)+\frac{1}{2}\varepsilon^2} \sim \frac{\frac{1}{2}\varepsilon+0}{\frac{3}{4}+0+0} \sim \frac{2}{3}\varepsilon. \tag{27}$$

From Fig. 3, we can derive the following:

$$R^{\mathrm{TPM}(K=3)}(\varepsilon) < U^{\mathrm{TPM}}(\rho), \quad \rho \to 1, \quad \varepsilon \to 0, \tag{28}$$

and

$$U^{\mathrm{TCM}}(\rho) < R^{\mathrm{TCM}(K=3)}(\varepsilon), \quad \rho \to 1, \quad \varepsilon \to 0. \tag{29}$$

Equations (28) and (29) indicate that TPM can meet the essential condition; however, TCM cannot meet it. The $R(\varepsilon)$ is closely affected by the state classifier, and we make an assumption that the factors of state classifier can impact on the security of TSCM. The definition used in the following is previewed here.

Definition 1. (i) For two different state vectors in one class, i.e., σ and σ' in c_1, the Hamming distance (HD) between them is defined by:

$$\mathrm{HD}^{c_1} = \mathrm{HD}(\sigma,\sigma') = \sum_{i=1}^{K}(\sigma_i \oplus \sigma_i'). \tag{30}$$

(ii) The smallest Hamming distance (SHD) in class c_1:

$$\mathrm{SHD}^{c_1} = \min\left\{\mathrm{HD}_1^{c_1}, \mathrm{HD}_2^{c_1}, \ldots\right\}. \tag{31}$$

(iii) The minimum Hamming distance (h) in the classifier:

$$h = \min\{\mathrm{SHD}^{c_1}, \mathrm{SHD}^{c_2}, \ldots\}. \tag{32}$$

By Definition 1 and the state classifier of TPM, the computation process of $h^{\mathrm{TPM(K=3)}}$ for TPM is described as follows:

$$\begin{aligned}
\mathrm{HD}_1^{c_1} &= \mathrm{HD}((+1,+1,+1),(+1,-1,-1)) = 2,\\
\mathrm{HD}_2^{c_1} &= \mathrm{HD}((+1,+1,+1),(-1,+1,-1)) = 2,\\
&\ldots,\\
\mathrm{SHD}^{c_1} &= \min\left\{\mathrm{HD}_1^{c_1}, \ldots, \mathrm{HD}_6^{c_1}\right\} = 2,\\
\mathrm{SHD}^{c_2} &= \min\left\{\mathrm{HD}_1^{c_2}, \ldots, \mathrm{SHD}_6^{c_2}\right\} = 2,\\
h^{\mathrm{TPM(K=3)}} &= \min\{\mathrm{SHD}^{c_1}, \mathrm{SHD}^{c_2}, \ldots\} = 2.
\end{aligned}$$

In the same manner as above, we have $h^{\mathrm{TCM(K=3)}} = 1$ for TCM.

Note that Eqs. (26) and (27) can be represented by a uniform formula:

$$R(\varepsilon) \sim \lambda\varepsilon^h, \quad \varepsilon \to 0, \quad \lambda \in \mathbb{R}^+. \tag{33}$$

Here, h is mentioned by Definition 1 and λ is a real number related to the state classifier.

The following illustrates the reason for Eq. (33):

(1) In the synchronization process, the probability of the occurrence of an attractive step between one pair of corresponding hidden units is calculated by

$$P_a(\varepsilon) = \frac{\sum_{i=0}^{n} a_i \binom{n}{i}\varepsilon^i(1-\varepsilon)^{n-i}}{P_u(\varepsilon)}. \tag{34}$$

In fact, attractive step can surely happen when all pairs of corresponding hidden units agree. So, $a_0 \neq 0$ and the item

$$\frac{a_0\binom{n}{0}\varepsilon^0(1-\varepsilon)^n}{P_u(\varepsilon)} \tag{35}$$

must exist.

(2) Similarly, we can obtain

$$P_r(\varepsilon) = \frac{\sum_{i=0}^{n} r_i \binom{n}{i}\varepsilon^i(1-\varepsilon)^{n-i}}{P_u(\varepsilon)}. \tag{36}$$

However, repulsive step cannot exist in $\sigma^{\mathrm{A}} = \sigma^{\mathrm{B}}$. According to Definition 1, if $\sigma^{\mathrm{A}} \neq \sigma^{\mathrm{B}}$, it is impossible that $\tau^{\mathrm{A}} = \tau^{\mathrm{B}}$ occurs as $i < h$. This also means the repulsive step can not occur as $i < h$. Therefore, in Eq. (36), $r_i = 0, i < h$. Meanwhile, the item

$$\frac{r_h\binom{n}{h}\varepsilon^h(1-\varepsilon)^{n-h}}{P_u(\varepsilon)} \tag{37}$$

exists.

(3) Combining Eqs. (34) and (36), it can be easily obtained

$$R(\varepsilon) = \frac{P_r(\varepsilon)}{P_a(\varepsilon)} = \frac{\sum_{i=h}^{n} r_i\binom{n}{i}\varepsilon^i(1-\varepsilon)^{n-i}}{\sum_{i=0}^{n} a_i\binom{n}{i}\varepsilon^i(1-\varepsilon)^{n-i}}. \tag{38}$$

As $\varepsilon \to 0$, taking the approximate value of the numerator and the denominator in Eq. (38), it can be derived

$$R(\varepsilon) \sim \frac{r_h\binom{n}{h}\varepsilon^h(1-\varepsilon)^{n-h}}{a_0\binom{n}{0}\varepsilon^0(1-\varepsilon)^n} = \frac{r_h\binom{n}{d}\varepsilon^h}{a_0\binom{n}{0}}. \tag{39}$$

Simplifying Eq. (39), we can obtain Eq. (35).

According to Eqs. (12), (29), (33) and Definition 1, a heuristic rule impacting on the security of TSCM can be found.

Heuristic rule. Keeping the equations of motion constant, a state classifier with a larger h predicts a higher probability that the TSCM-based neural cryptography can meet Condition I.

IV. NOVEL NEURAL CRYPTOGRAPHY

A generic framework is established to design neural cryptography in Sec. II. Moreover, Sec. III puts forward a significant heuristic rule for improving the security of the TSCM. Here, we show how to design neural cryptography by applying these results. Meanwhile, the comparisons with TPM ($K = 3$) on security are carefully illustrated.

A. Applications

Remarkably, considering the heuristic rule for Condition II, excessively large h is probably prejudicial to the security of neural cryptography. Consequently, we need the state classifications with $h = 2$. In order to guarantee this, some state vectors are removed, which is indicated by $d = 0$. And the state classifications are formulated in Tables III and IV.

TABLE III. The state classification of CASE A ($K = 3$).

Name	τ	d	State vector σ
CA	τ_1	$+1$	$(+1,+1,-1),(+1,-1,+1)$ $(-1,+1,+1)$
	τ_2	-1	$(-1,-1,+1),(-1,+1,-1)$ $(+1,-1,-1)$
	τ_3	0	$(+1,+1,+1)(-1,-1,-1)$

Notice that $U(\rho)$ is constant as long as the learning rule stays the same, so we have

$$U^{\mathrm{TPM}}(\rho) = U^{\mathrm{CA}}(\rho) = U^{\mathrm{CB}}(\rho). \tag{40}$$

While R^{CA} and R^{CB} are calculated as

$$\begin{aligned} P_a^{\mathrm{CA}}(\varepsilon) &= \frac{\frac{1}{2}\varepsilon^0(1-\varepsilon)^3 + \frac{1}{2}\varepsilon^2(1-\varepsilon)}{P_u^{\mathrm{CA}}}; \\ P_r^{\mathrm{CA}}(\varepsilon) &= \frac{\varepsilon^2(1-\varepsilon)}{P_u^{\mathrm{CA}}}; \\ P_u^{\mathrm{CA}}(\varepsilon) &= \frac{3}{4}\varepsilon^0(1-\varepsilon)^3 + \frac{3}{2}\varepsilon^2(1-\varepsilon); \\ R^{\mathrm{CA}}(\varepsilon) &= \frac{2\varepsilon^2}{(1-\varepsilon)^2+\varepsilon^2}; \end{aligned} \tag{41}$$

and

$$\begin{aligned} P_a^{\mathrm{CB}}(\varepsilon) &= \frac{\frac{3}{8}(1-\varepsilon)^5 + \frac{3}{2}\varepsilon^2(1-\varepsilon)^3 + \frac{3}{8}\varepsilon^4(1-\varepsilon)}{P_u^{\mathrm{CB}}}; \\ P_r^{\mathrm{CB}}(\varepsilon) &= \frac{\frac{3}{2}\varepsilon^2(1-\varepsilon)^3 + \frac{3}{2}\varepsilon^4(1-\varepsilon)}{P_u^{\mathrm{CB}}}; \\ P_u^{\mathrm{CB}}(\varepsilon) &= \frac{5}{8}(1-\varepsilon)^5 + \frac{15}{4}\varepsilon^2(1-\varepsilon)^3 + \frac{15}{8}\varepsilon^4(1-\varepsilon); \\ R^{\mathrm{CB}}(\varepsilon) &= \frac{4\varepsilon^2(1-\varepsilon)^2 + 4\varepsilon^4}{(1-\varepsilon)^4 + 4\varepsilon^2(1-\varepsilon)^2 + \varepsilon^4}. \end{aligned} \tag{42}$$

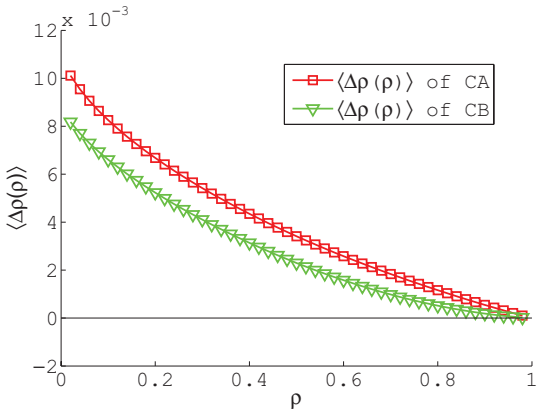

FIG. 4. (Color online) The dynamics of $\langle \Delta\rho(\rho) \rangle$ of CA and CB, while the lines have been calculated using Eq. (14).

Significantly, simulation experiment displayed in Fig. 4 indicates that the two state classifiers enable CA and CB to meet Condition I.

We remark that a series of neural cryptography can be formulated in such way, and also we can hopefully search for some better cases by comparing their security.

B. Attacks on neural cryptography

Suppose that the attacker E is equipped with the same structured networks as A's and B's. In addition, E can eavesdrop all the information on the public channels but cannot tamper or intercept them, so that only passive attacks are considered.

For the simple attack [7], E aspires for synchronization with A just by imitating the learning behavior. So E trains its networks with the examples, including the input vector x and the output τ^{A}, by applying one of the following rules:

(1) Hebbian learning rule:

$$w_{i,j}^{\mathrm{E}} = g\left[w_{i,j}^{\mathrm{E}} + x_{i,j} d^{\mathrm{A}} \Theta\left(\sigma_i^{\mathrm{E}} d^{\mathrm{A}}\right) \Theta(\tau^{\mathrm{A}} \tau^{\mathrm{B}})\right]. \tag{43}$$

TABLE IV. The state classification of CASE B ($K = 5$).

Name	τ	d	State vector σ
CB	τ_1	$+1$	$(+1,+1,+1,-1,-1),(+1,+1,-1,-1,+1)$ $(+1,-1,-1,+1,+1),(-1,-1,+1,+1,+1)$ $(+1,+1,-1,+1,-1),(+1,-1,+1,-1,+1)$ $(-1,+1,-1,+1,+1),(+1,-1,+1,+1,-1)$ $(-1,+1,+1,-1,+1),(-1,+1,+1,+1,-1)$
	τ_2	-1	$(-1,-1,-1,+1,+1),(-1,-1,+1,+1,-1)$ $(-1,+1,+1,-1,-1),(+1,+1,-1,-1,-1)$ $(-1,-1,+1,-1,+1),(-1,+1,-1,+1,-1)$ $(+1,-1,+1,-1,-1),(-1,+1,-1,-1,+1)$ $(+1,-1,-1,+1,-1),(+1,-1,-1,-1,+1)$
	τ_3	0	$(-1,-1,-1,-1,+1),(-1,-1,-1,+1,-1)$ $(-1,-1,+1,-1,-1),(-1,+1,-1,-1,-1)$ $(+1,-1,-1,-1,+1),(-1,+1,+1,+1,+1)$ $(+1,+1,-1,+1,+1),(+1,-1,+1,+1,+1)$ $(+1,+1,+1,+1,-1),(+1,+1,+1,-1,+1)$ $(+1,+1,+1,+1,+1),(-1,-1,-1,-1,-1)$

(2) Anti-Hebbian learning rule:

$$w_{i,j}^{\mathrm{E}} = g\big[w_{i,j}^{\mathrm{E}} - x_{i,j}d^{\mathrm{A}}\Theta\big(\sigma_i^{\mathrm{E}}d^{\mathrm{A}}\big)\Theta(\tau^{\mathrm{A}}\tau^{\mathrm{B}})\big]. \tag{44}$$

(3) Random-walk learning rule:

$$w_{i,j}^{\mathrm{E}} = g\big[w_{i,j}^{\mathrm{E}} + x_{i,j}\Theta\big(\sigma_i^{\mathrm{E}}d^{\mathrm{A}}\big)\Theta(\tau^{\mathrm{A}}\tau^{\mathrm{B}})\big]. \tag{45}$$

It's not hard to see that the simple attack is appropriate for all neural cryptography despite low efficiency.

If E is an intelligent attacker, it may try to predict the $\sigma^{\mathrm{A/B}}$ in each learning step by taking h_i^{E} into account. Geometric attack [29] is one kind of guess attack using single networks. The geometric attack is the same as simple attack in the case of $\tau^{\mathrm{A}} = \tau^{\mathrm{B}} = \tau^{\mathrm{E}}$. But, as long as $\tau^{\mathrm{A}} = \tau^{\mathrm{B}} \neq \tau^{\mathrm{E}}$, E attempts to change τ^{E} by means of correcting σ^{E} in consideration that A will still update the w^{A}. Note that the lower absolute value of $|h_i^{\mathrm{E}}|$ represents the higher probability of $\sigma_i^{\mathrm{A}} \neq \sigma_i^{\mathrm{E}}$. So, the attacker can flip one σ_i^{E} with the smaller $|h_i^{\mathrm{E}}|$ to enable $\tau^{\mathrm{A/B}} = \tau^{\mathrm{E}}$. Then, the attacker can utilize the same learning procedure as the case of $\tau^{\mathrm{A}} = \tau^{\mathrm{B}} = \tau^{\mathrm{E}}$. In fact, even the geometric attack cannot predict the $\sigma_i^{\mathrm{A/B}}$ perfectly; it always outperforms simple attack.

Remarkably, the original geometric attack is only appropriate for TPM and TCM. So, as a result, this paper puts forward a modified geometric attack for the instances based on TSCM. The modified geometric attack tries to enable $\tau^{\mathrm{A/B}} = \tau^{\mathrm{E}}$ by means of flipping multiple σ_i^{E} with the smaller $|h_i^{\mathrm{E}}|$ instead of one. It is verified by the simulation in the next subsection that the modified geometric attack is more effective than the original geometric attack for TSCM-based neural cryptography, including the cases presented in Ref. [31].

Majority attack [23] and genetic attack [30] are two other kinds of guess attack. They attempt to improve the E's ability to predict the $\sigma_i^{\mathrm{A/B}}$ by using multiple networks instead of single networks. In particular, a better resistance to attack based on single networks indicates a better resistance to attack using multiple networks.

C. Security

Consider that the majority attack and genetic attack are based on simple attack, original geometric attack, or modified geometric attack as an element; therefore, we only involve the attack using single networks here.

In the case of simple attack for all TSCM-based neural cryptography, a repulsive step occurs between one pair of corresponding hidden units with the probability

$$P_r^{\mathrm{E}}(\varepsilon) = \varepsilon. \tag{46}$$

and an attractive step with the probability

$$P_a^{\mathrm{E}}(\varepsilon) = a(1-\varepsilon), \quad 0 < a \leqslant 1. \tag{47}$$

Here, a is the percentage of the updating hidden units. Meanwhile, $U^{\mathrm{E}}(\rho)$ for any attacker is same as $U^{\mathrm{TPM}}(\rho)$, so

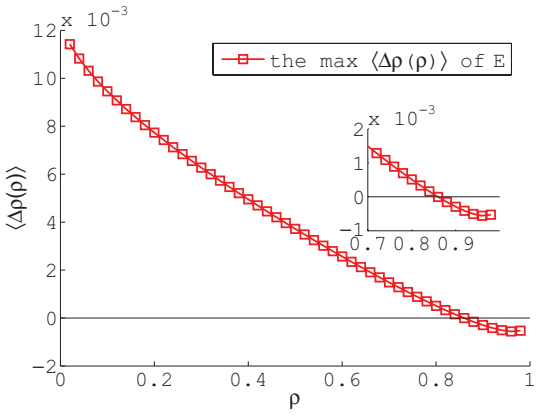

FIG. 5. (Color online) The dynamics of the max $\langle\Delta\rho(\rho)\rangle$ of E, while the lines have been calculated using Eq. (48).

we can obtain

$$\begin{aligned}\langle\Delta\rho(\rho)\rangle^{\mathrm{E}} &= P_a^{\mathrm{E}}(\varepsilon)\langle\Delta\rho_a(\rho)\rangle + \mathrm{P}_{\mathrm{r}}^{\mathrm{E}}(\varepsilon)\langle\Delta\rho_r(\rho)\rangle \\ &\leqslant (1-\varepsilon)\langle\Delta\rho_a(\rho)\rangle + \varepsilon\langle\Delta\rho_r(\rho)\rangle.\end{aligned} \tag{48}$$

As shown in Fig. 5, it demonstrates that the simple attack is not sufficient to break neural cryptography.

In the case of original geometric attack for TPM ($K = 3$), the probability for a successful correction of the hidden units is given by Ref. [23]:

$$\begin{aligned}P_g = \int_0^\infty \prod_{j\neq i}\Bigg(\int_{h_i}^\infty \frac{2}{\sqrt{2\pi\,Q_j}}\frac{1-\epsilon_j^p}{1-\epsilon_j} \\ e^{-\frac{h_j^2}{2Q_j}}dh_j\Bigg)\frac{2}{\sqrt{2\pi\,Q_i}}\frac{\epsilon_i^p}{\epsilon_i}e^{-\frac{h_i^2}{2Q_i}}dh_i.\end{aligned} \tag{49}$$

Here, ϵ_i^p is the prediction error [35]:

$$\epsilon_i^p = \frac{1}{2}\left[1 - \mathrm{erf}\left(\frac{\rho_i}{\sqrt{2\big(1-\rho_i^2\big)}}\frac{|h_i|}{\sqrt{Q_i}}\right)\right], \tag{50}$$

and Q_i is the initial length of the weight vectors [29]:

$$Q_i \sim \frac{L(L+1)}{3}\left(1 + \frac{8}{5\sqrt{3\pi}}\frac{L}{\sqrt{N}}\right). \tag{51}$$

In this case, the probabilities of the attacker can be computed as

$$\begin{aligned}P_a^{\mathrm{E\_TPM}}(\varepsilon) &= \tfrac{1}{2}(1-\varepsilon)^3 + \tfrac{1}{2}(1+2P_g)(1-\varepsilon)^2\varepsilon \\ &\quad + \tfrac{1}{2}(1-\varepsilon)\varepsilon^2 + \tfrac{1}{6}\varepsilon^3; \\ P_r^{\mathrm{E\_TPM}}(\varepsilon) &= 2(1-P_g)(1-\varepsilon)^2\varepsilon \\ &\quad + 2(1-\varepsilon)\varepsilon^2 + \tfrac{2}{3}\varepsilon^3.\end{aligned} \tag{52}$$

However, modified geometric attack faces the challenge of flipping more than one hidden unit, so the probability of flipping correctly is confirmed to be less than P_g. In this case, one can calculate the probability of the occurrence of an attractive and a repulsive step for the attacker in attacking

CA:

$$\begin{aligned} P_a^{\text{E\_CA}}(\varepsilon) &< \tfrac{2}{3}(1-\varepsilon)^3 + (1+P_g)(1-\varepsilon)^2\varepsilon \\ &\quad + \tfrac{1}{3}(3+P_g)(1-\varepsilon)\varepsilon^2 + \tfrac{1}{3}\varepsilon^3; \\ P_r^{\text{E\_CA}}(\varepsilon) &> 2(1-P_g)(1-\varepsilon)^2\varepsilon \\ &\quad + \tfrac{2}{3}(3-P_g)(1-\varepsilon)\varepsilon^2 + \tfrac{2}{3}\varepsilon^3; \end{aligned} \tag{53}$$

and CB

$$\begin{aligned} P_a^{\text{E\_CB}}(\varepsilon) &< \tfrac{3}{5}(1-\varepsilon)^5 + (2+P_g)(1-\varepsilon)^4\varepsilon \\ &\quad + \tfrac{2}{5}(10+2P_g)(1-\varepsilon)^3\varepsilon^2 \\ &\quad + \tfrac{1}{5}(11+10P_g)(1-\varepsilon)^2\varepsilon^3 \\ &\quad + \tfrac{1}{5}(5+2P_g)(1-\varepsilon)\varepsilon^4 + \tfrac{1}{5}\varepsilon^5; \\ P_r^{\text{E\_CB}}(\varepsilon) &> 2(1-P_g)(1-\varepsilon)^4\varepsilon \\ &\quad + \tfrac{2}{5}(10-4P_g)(1-\varepsilon)^3\varepsilon^2 \\ &\quad + \tfrac{2}{5}(19-10P_g)(1-\varepsilon)^2\varepsilon^3 \\ &\quad + \tfrac{1}{5}(20-4P_g)(1-\varepsilon)\varepsilon^4 + \tfrac{4}{5}\varepsilon^5. \end{aligned} \tag{54}$$

One can easily obtain

$$\frac{P_a^{\text{E\_CI}}}{P_r^{\text{E\_CI}}} > \frac{P_a^{\text{E\_TPM}}}{P_r^{\text{E\_TPM}}}. \tag{55}$$

This indicates that CI is not more secure than TPM under geometric attack. But for CB, it is hard to get an accurate inequality with TPM, so we cannot select the better one directly. Then the simulations results in Tables V and VI can be show the comparison on security clearly. Successful attack probability P is defined as the probability that the attacker knows 98 percent of the weights at synchronization time. CI is the case considered better than TPM on security [31].

It is worth noting that all the existing algorithms enhancing the security of TPM ($K = 3$) can also improve all TSCM-based neural cryptography.

From Tables V and VI, one can see that the comparison of security between these instances is

$$\textit{Simple attack} : \text{CI} > \text{CB} > \text{TPM} > \text{CA}, \tag{56}$$

$$\textit{Geometric attack} : \text{CB} > \text{TPM} > \text{CI} > \text{CA}. \tag{57}$$

CI performs well under simple attack, because CI is based on bidirectional synchronization, which brings about more repulsive steps for simple attack. But in the face of geometric attack, CI is unreliable because of its defective state classifier. Each class of the state classifier in CI only contains two vectors, so the probability that the CI's state is predicted correctly by E is very high. So, disagreeing with the opinion of Ref. [31], CI is not better than TPM.

TABLE V. Success probability P of the simple attack as a function of L. Results are obtained in 10 000 simulations with random-walk learning rule using $K \times N = 3000$.

Name	$L = 1$	$L = 2$	$L = 3$	$L = 4$
TPM ($K = 3$)	0.4617	0.0393	0.0006	0
CI ($K = 3$)	0.3176	0.0073	0	0
CA ($K = 3$)	0.4901	0.0936	0.0030	0
CB ($K = 5$)	0.4061	0.0315	0.0004	0

TABLE VI. Success probability P of the geometric attack (original or modified) as a function of L. Results are obtained in 10 000 simulations with random-walk learning rule using $K \times N = 3000$.

Name	$L = 1$	$L = 2$	$L = 3$	$L = 4$
TPM ($K = 3$)	0.7673	0.5883	0.4276	0.2799
CI ($K = 5$)	0.9879	0.8999	0.7080	0.4926
CA ($K = 3$)	0.9251	0.9038	0.8759	0.8552
CB ($K = 5$)	0.7271	0.5497	0.3955	0.2642

Similar to CI, the state classifier in CA is also unsatisfactory. Meanwhile, CA is not based on bidirectional synchronization. So CA is the worst case.

However, the state classifier in CB contains more vectors in one class, and also means that the attackers have to correct more hidden units as $\tau^{\text{A}} = \tau^{\text{B}} \neq \tau^{\text{E}}$. Owing to these, the attackers will have much more trouble to perfectly correct. So, CB outperforms TPM.

Furthermore, the success probability of the attacker in neural cryptography is confirmed as $e^{-y(L-L_0)}$ [23]. This scaling behavior is the same for all attacks, and the constants y and L_0 are different for each attack. Clearly, y and L_0 can be determined by enough simulations.

Based on all of the above, we conclude that CB is a more secure case against all attacks known up to now. It means that the attackers are unable to break the security of CB in the limit $L \to \infty$. Meanwhile, this verifies the validity and applicability of our approach to designing neural cryptography.

V. CONCLUSION

Different from most of the previous works that are focused on the existing neural cryptography, we have carefully elaborated how to design and secure neural cryptography. The main result, namely the TSCM with a heuristic rule, provides an effective approach for designing neural cryptography. In addition, CB proposed in this paper has also demonstrated that the better instances can be achieved from our results. These results have also been verified by simulations.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (Grants No. 60973114 and No. 61170249), Natural Science Foundation project of CQCSTC (No. 2009BA2024), the State Key Laboratory of Power Transmission Equipment & System Security and New Technology, Chongqing University (No. 2007DA10512711206), the program for Changjiang scholars, the Specialized Research Fund for priority areas for the Doctoral Program of Higher Education, the National Priority Research Project No. NPRP 4-1162-1-181, funded by Qatar National Research Fund, Qatar, the Science and Technology Project of Chongqing Education Committee under Grant (KJ130519) and the Teaching & Research Program of Chongqing Education Committee (KJ131401).

[1] W. Diffie and M. Hellman, IEEE Trans. Inf. Theory **22**, 644 (1976).
[2] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography* (CRC Press, Boca Raton, FL, 1996).
[3] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C,* 2nd ed. (Wiley, New York, 1995).
[4] W. Stallings, *Cryptography and Network Security: Principles and Practice* (Pearson, Upper Saddle River, NJ, 2002).
[5] H. Li, X. Liao, C. Li, H. Huang, and C. Li, Commun. Nonlin. Sci. Numer. Simul. **16** (2011).
[6] M. J. Er, W. Chen, and S. Wu, IEEE Trans. Neural Networks **16**, 679 (2005).
[7] I. Kanter, W. Kinzel, and E. Kanter, Europhys. Lett. **57**, 141 (2002).
[8] U. Maurer, IEEE Trans. Inf. Theory **39**, 733 (1993).
[9] M. Volkmer and S. Wallner, IEEE Trans. Comput. **54**, 421 (2005).
[10] M. Volkmer and S. Wallner, Cryptology ePrint Archive Report (2005).
[11] E. Klein, N. Gross, M. Rosenbluh, W. Kinzel, L. Khaykovich, and I. Kanter, Phys. Rev. E **73**, 066214 (2006).
[12] E. Klein, R. Mislovaty, I. Kanter, and W. Kinzel, Phys. Rev. E **72**, 016214 (2005).
[13] E. Klein, N. Gross, E. Kopelowitz, M. Rosenbluh, L. Khaykovich, W. Kinzel, and I. Kanter, Phys. Rev. E **74**, 046201 (2006).
[14] E. Klein, N. Gross, M. Rosenbluh, W. Kinzel, L. Khaykovich, and I. Kanter, Phys. Rev. E **73**, 066214 (2006).
[15] D. Saad, *On-line Learning in Neural Networks* (Cambridge University Press, Cambridge, England, 1998).
[16] O. M. Reyes, I. Kopitzke, and K. Zimmermann, J. Phys. A: Math. Theor. **42**, 195002 (2009).
[17] O. M. Reyes and K. H. Zimmermann, Phys. Rev. E **81**, 066117 (2010).
[18] M. Rosen-Zvi, E. Klein, I. Kanter, and W. Kinzel, Phys. Rev. E **66**, 066135 (2002).
[19] L. F. Seoane and A. Ruttor, Phys. Rev. E **85**, 025101 (2012).
[20] W. Kinzel and I. Kanter, J. Phys. A: Math. Gen. **36**, 11173 (2003).
[21] W. Kinzel and I. Kanter, Adv. Solid State Phys. **42**, 383 (2002).
[22] W. Kinzel, *Theory of Interacting Neural Networks* (Wiley VCH, Berlin, 2003).
[23] A. Ruttor, W. Kinzel, R. Naeh, and I. Kanter, Phys. Rev. E **73**, 036121 (2006).
[24] A. Ruttor, W. Kinzel, and I. Kanter, Phys. Rev. E **75**, 056104 (2007).
[25] A. Ruttor, G. Reents, and W. Kinzel, J. Phys. A: Math. Gen. **37**, 8609 (2004).
[26] A. Ruttor, W. Kinzel, L. Shacham, and I. Kanter, Phys. Rev. E **69**, 046110 (2004).
[27] A. Ruttor, W. Kinzel, and I. Kanter, J. Stat. Mech. (2005) P01009.
[28] A. M. Allam and H. M. Abbas, IEEE Trans. Neural Networks **21**, 1915 (2010).
[29] A. Klimov, A. Mityagin, and A. Shamir, Lect. Notes Comput. Sci. **2501**, 288 (2002).
[30] L. N. Shacham, E. Klein, R. Mislovaty, I. Kanter, and W. Kinzel, Phys. Rev. E **69**, 066137 (2004).
[31] X. Lei, X. Liao, F. Chen, and T. Huang, Phys. Rev. E **87** 032811 (2013).
[32] A. Engel and C. V. den Broeck, *Statistical Mechanics of Learning* (Cambridge University Press, Cambridge, UK, 2001).
[33] W. Kinzel, R. Metzler, and I. Kanter, J. Phys. A: Math. Gen. **33**, L141 (2000).
[34] A. Ruttor, Ph.D. dissertation, Wurzburg University, German, 2006.
[35] L. Ein-Dor and I. Kanter, Phys. Rev. E **60**, 799 (1999).