Hindawi
Security and Communication Networks
Volume 2020, Article ID 8869688, 10 pages
https://doi.org/10.1155/2020/8869688

Research Article

An Improved Method to Evaluate the Synchronization in Neural Key Exchange Protocol

Yi Liang Han, Yu Li, Zhe Li, and Shuai Shuai Zhu

School of Cryptology Engineering, Engineering University of PAP, Xi'an 710086, China

Correspondence should be addressed to Yi Liang Han; yilianghan@hotmail.com

Received 3 September 2020; Revised 10 October 2020; Accepted 17 October 2020; Published 29 October 2020

Academic Editor: Amir Anees

The synchronization between two neural networks by mutual learning can be used to design the neural key exchange protocol. The critical issue is how to evaluate the synchronization without a weight vector. All existing methods have a delay in evaluating the synchronization, which affects the security of the neural key exchange. To evaluate the full synchronization of neural networks more timely and accurately, an improved method for evaluating the synchronization is proposed. First, the frequency that the two networks have the same output in previous steps is used for assessing the degree of them roughly. Second, the hash function is utilized to judge whether the two networks have achieved full synchronization precisely when the degree exceeds a given threshold. The improved method can find the full synchronization between two networks with no information other than the hash value of the weight vector. Compared with other methods, the full synchronization can be detected earlier by two communication partners which adopt the method proposed in this paper. As a result, the successful probability of geometric is reduced. Therefore, the proposed method can enhance the security of the neural exchange protocol.

1. Introduction

It is vital to ensure the information and communication security in the network society. There are a lot of techniques that can achieve the purpose. The most commonly used are symmetric cryptography and asymmetric cryptography. The efficiency of asymmetric cryptography is lower than the efficiency of symmetric cryptography. Hence, the asymmetric cryptography algorithm is used to construct a key exchange protocol, and the obtained key is used to encrypt and decrypt with symmetric cryptography algorithm. However, the algorithms of key exchange are mainly based on number theory, which requires massive computation and memory. Therefore, there is a novel approach to obtain a symmetric key by neural networks, which is called the neural key exchange. Two neural networks can achieve synchronization through mutual learning. They get the common input vector and send the output of their own to each other. Then, after synchronization, the equivalent weights of the two networks can be used as the symmetric key, which can be used to encrypt and decrypt in the later session.

One of the critical issues in neural key exchange is how to evaluate the degree of the synchronization of the two neural networks while with no weights information of the other party. The methods to calculate the cosine of the weights vector and the Euclidean distance of the weights vector cannot be used in practice because there is no information about the weights vector of the other party. Deolecki et al. [1] proposed a method to evaluate the degree of synchronization by calculating the frequency of the equivalent output in fixed previous learning steps of the synchronization process. However, there is a delay to find full synchronization between the two networks in their method, which means that the two networks still continue learning while they have achieved full synchronization. The active attacker will reveal extra information from the additional learning steps. Liu [2] proposed a method to evaluate the synchronization of neural networks by using hash function. Their method can find the full synchronization precisely. However, their method will increase the communication traffic and decrease efficiency. Exceptionally, if the hash function is used untimely, there will be substantial communication traffic. It is necessary to

find the full synchronization of the two networks early and finish the learning. The earlier the full synchronization is found, the less information the attacker can obtain.

The motivation of this work is to enhance the security and improve the efficiency of neural key exchange protocol by improving the method for evaluating the synchronization between two neural networks. In the existing synchronization evaluation methods, such as the hash function method and the frequency method, there is a large delay in the judgment of the full synchronization. Due to the large delay, the probability of the neural key exchange protocol being broken by the geometric attacker increases. The efficiency of neural key exchange protocol is also reduced by the large delay. What is more, the frequency method also has the possibility of misjudgment. When misjudgment occurs, both communication partners have to perform the neural key exchange protocol again. Consequently, the synchronization evaluation method plays an important role in neural key exchange. In this paper, an improved method to reduce the delay and misjudgment rate is proposed. The advantages of the hash function and the method based on the output of neural networks are combined. Therefore, the contributions of this paper are as follows:

(1) It proposes an improved method to evaluate the synchronization of the neural networks. The improved method can find full synchronization precisely and timely. The improved method can reduce the delay.

(2) It proposes an algorithm to search for the optimal parameter of the improved evaluation method. The binary search is also used. Thanks to the optimal parameters, the misjudgment rate of the improved method is reduced. The experimental results show that the delayed steps of the proposed method are lower than other methods, and the misjudgment rate is almost zero. The security of neural key exchange which adopts the improved method can be enhanced.

The remainder of this paper is organized as follows. The related work is given in Section 2. The neural key exchange protocol is presented in Section 3. The method proposed in this paper is presented in Section 4. The results and discussion of the proposed method are presented in Section 5. The conclusion is given in Section 6.

2. Related Work

Neural networks have been applied to cryptography widely [3–7]. A lot of work has been carried out for it. In this section, the development and current state of the neural key exchange are reviewed as follows.

Key exchange protocol based on neural networks was first proposed in [8] and implemented with simple parameters. The authors analyzed the security of the protocol with naive attackers. To accelerate the synchronization of neural networks, the authors also presented the bit-packages techniques and generalized the protocol. Klimov et al. [9] analyzed the protocol proposed in [8] and explained why the two communication partners could achieve full synchronization of weights. Their results show that the naive attacker cannot achieve full synchronization with the two communication partners even though the same structure of networks, input, and learning rules is used. Then, they proposed three types of attacks, which were the genetic attack, the geometric attack, and the probabilistic attack. The three types of attacks can break the protocol with simple parameters in [8]. However, the security of the neural key exchange protocol depends on the structure parameters of the neural networks. Hence, increasing the parameters can resist the genetic attack, geometric attack, and probabilistic attack. Shacham et al. [10] proposed a cooperating strategy, which can cooperate with the genetic attack, geometric attack, and probabilistic attack, so that the success probability of attackers is not affected by the depth of the synapse.

Ruter et al. [11] made a comparative analysis among genetic attack, geometric attack, and majority attack. The simulation results show that the neural key exchange protocol is a security with increasing the depth of the synapse. Besides, they proposed a method that queries were used instead of the random common input, which can improve the security of neural key exchange protocol. Santhanalakshmi et al. [12] applied the genetic algorithm in the synchronization of neural networks to search optimal weights as the initial weights. This method can accelerate synchronization by reducing learning time and steps. Later, the authors analyzed the performance of the protocol proposed in [12]. They analyzed the parameters of the genetic algorithm and tree parity machine [13]. Their results show that security can be improved by increasing the number of the hidden layer of the tree parity machine. Allam et al. [14] proposed an authenticated key exchange protocol by using a preshared key as the boundary of learning, which is called the neural cryptography secret boundaries (NSCB) protocol. The NCSB protocol adopted the dynamic learning rate and random walk learning rules. The advantage of NCSB is that it can improve the security of neural key exchange without reducing efficiency. Chourasia et al. [15] proposed a vectorized neural key exchange protocol. Pal et al. [16] proposed a new learning rule, which can accelerate the synchronization and increase the randomness of the key. Dong and Huang [17] generalized the tree parity machine by using the complex value. Édgar et al. [18] proposed a method to search the optimal structure of the tree parity machine, which makes the neural key exchange protocol more efficient and secure.

Ruttor et al. [19] adopted the cosine similarity to evaluate the degree of the synchronization between two neural networks. It can judge full synchronization precisely by using the weights of Alice and Bob. However, Alice or Bob has no information about the weights of the other party in practice. The Euclidean distance method which needs weight cannot be used in practice too. Doleci et al. [1] show that the frequency with which both communication partners have the same output can be used to evaluate the synchronization. The simulation results show that the relationships among the frequency method, cosine similarity, and Euclidean distant are very high. Gupta and Deshmukh [20] applied the

protocol proposed in [8] to encrypt and decrypt the image secret sharing. Sarkar [21] applied the session key generated by the synchronization of multilayer perceptron to wireless communications. Shishniashvili et al. [22] proposed a technique to use parts of the weight vector instead of the entire weight vector as the session key. The synchronization between two neural networks can also be used as error reconciliation in quantum key distribution protocols [23]. Niemiec [24] proposed a novel method that the tree parity machine is used to correct errors taken place during transmission in quantum key distribution protocol. The influence of parameter variation on security and the influence of different learning rules on efficiency are analyzed in [25].

These schemes introduced above mainly adopted the following four methods for evaluating the synchronization: the cosine similarity, the Euclidean distance, the hash function, and the frequency that the outputs of both partners are equal. The strengths and weaknesses of the four methods are listed in Table 1.

We analyzed the methods from four aspects: delay, extra traffic, misjudgment, and practicality. The indicator delay means that the method cannot find the full synchronization in the first place. Only after many steps of true synchronization have taken place, the synchronization is determined, and the neural key exchange is completed. The indicator extra traffic means that the communication traffic is increased by the delay. The indicator misjudgment means that the neural key exchange is completed, while the true synchronization is not achieved, which fail to exchange the key. The indicator practicality means that whether the method can be used in practice. In the cosine similarity method and Euclidean distance method, there is no delay, no misjudgment, and no extra traffic. Although they have so many advantages, their fatal disadvantage is that they are not practical. Because the weight vector of the other network is not available. The hash function method and the frequency method are practical. But there is also delay and extra traffic when the two practical methods are used. The key issue of the hash function method is when to use the hash function. If the hash function is used too early, it will lead to inefficiency. If the hash function is used too late, it will reduce the security of neural key exchange. There is also misjudgment when using the frequency method.

In conclusion, the evaluation of the synchronization of neural networks is a serious matter affecting the security of the neural key exchange. In this paper, we combined the advantage of the output of neural networks and the hash function. We presented a synchronization evaluation method which can improve the security of the neural key exchange.

3. Neural Key Exchange Protocol

There are two communication partners called Alice and Bob in the neural key exchange protocol. Both of them own a tree parity machine, which is a special feedback neural network [26]. The two tree parity machines can achieve full synchronization by mutual learning. And then, the identical weights of the tree parity machines can be used as a session key to encrypt/decrypt or as a seed to generate a secret key. The neural key exchange protocol contains five main phases, which are the initialization phase, calculating phase, updating phase, evaluating phase, and completion phase. The flowchart of the neural key exchange protocol is shown in Figure 1, and the detail of each phase is described as follows.

Table 1: The strengths and weaknesses.

Method	Delay	Extra traffic	Misjudgment	Practicality
Cosine similarity	×	×	×	×
Euclidean distance	×	×	×	×
Hash function	√	√	×	√
Frequency	√	√	√	√

3.1. Initialization Phase. Alice and Bob initialize their tree parity machine with the same parameters. Then, they randomly generate the weights vector and initialize their weights, respectively.

3.2. Calculating Phase. Alice and Bob receive the common input vector at each learning step. They calculate the output of the tree parity machine and send the result to each other.

3.3. Updating Phase. After receiving the output of another party, they compare whether the two outputs are equal. If equal, Alice and Bob update their weights according to the special learning rules, such as Hebbian learning rules, anti-Hebbian learning rules, and random walk learning rules. Otherwise, it goes to the calculating phase.

3.4. Evaluation Phase. If the weights are updating successfully, Alice and Bob need to judge whether they have achieved full synchronization. The evaluation algorithm of the synchronization degree between Alice and Bob is used here. If the full synchronization is achieved, the protocol goes to the next phase. Otherwise, it goes to the calculating phase.

3.5. Completion Phase. Alice and Bob generate their key according to the weights. They finish the process of neural key exchange.

4. Proposed Method

In this section, we propose an improved method to evaluate the synchronization between two neural networks and an algorithm of searching for the optimal parameter of the method. Both the improved method and the algorithm are described in phases. The algorithm for each phase is also presented.

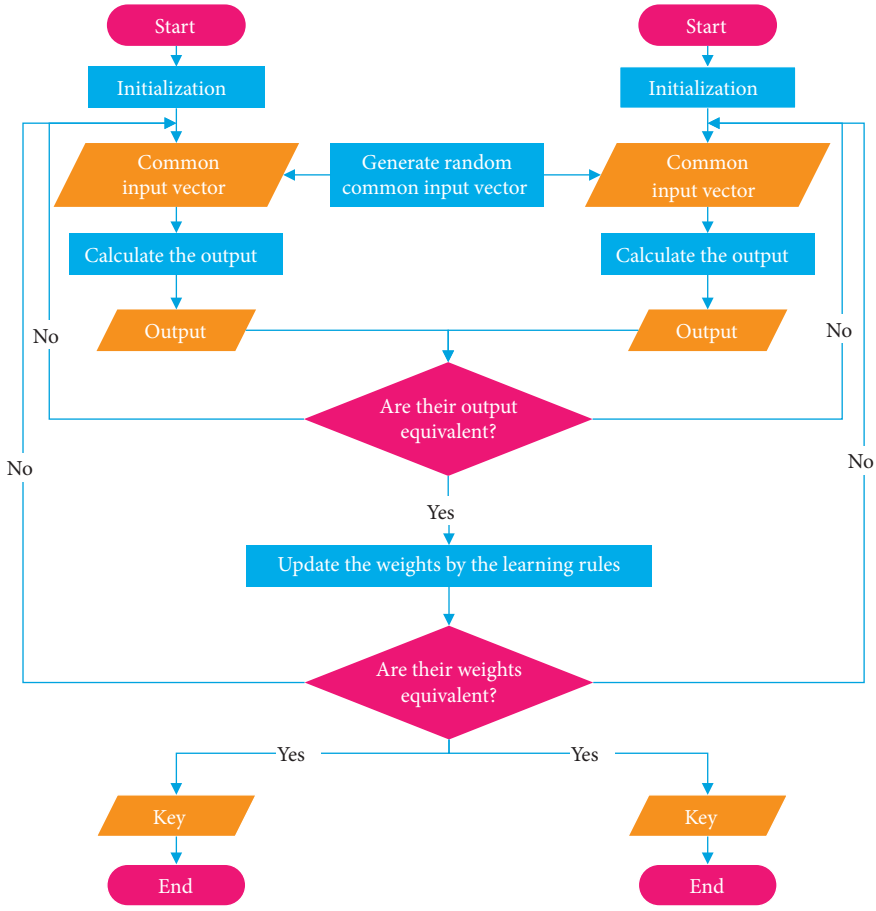

Figure 1: Neural key exchange protocol.

4.1. The Improved Method to Evaluate the Synchronization. There are four methods that can evaluate the synchronization, which are the cosines of the weight vector, the Euclidean distance of the weight vector, the frequency of the equivalent outputs, and the hash value of the weight vector. The shortcomings of the methods above are discussed in Section 1. And it is analyzed through simulation in Section 5. Therefore, it is important to evaluate the synchronization of the neural networks more accurately and timely. An improved method which can find the full synchronization in the first place is presented here. The parameters used in this paper are listed in Table 2.

The main idea of the improved method is judging the degree of synchronization roughly by calculating the frequency that both Alice and Bob have equal output. The hash values of the weights are calculated based on the degree. And then, whether the weights are equal is judged precisely by comparing the hash values. Hence, the process of evaluating the synchronization between Alice and Bob can be divided into two main phases, which are the rough evaluation phase and precise evaluation phase. The flowchart of the evaluating algorithm is given in Figure 2. The detail of each phase is described as follows.

4.1.1. Rough Evaluation Phase. Both Alice and Bob have no information about the weight vector of each other. The only information is the output delivered on a public channel. However, the frequency that Alice and Bob have the same output can be used to evaluate the degree of synchronization. The calculating algorithm is shown in Algorithm 1.

Table 2: Parameter description.

Parameter	Description
K	The number of hidden layer neurons
N	The number of input neurons of the perceptron
L	The depth of the synapse
s	Previous learning steps
W	The weight vector of neural networks
m	The number of learning steps in synchronization
τ_m^A	The result of Alice in m^{th} steps
τ_m^B	The result of Bob in m^{th} steps
a_m	The result of whether $\tau_m^A = \tau_m^B$
b_m	The average $a_m, a_{m-1}, \ldots, a_{m-s+1}$
t	The threshold of the degree of synchronization
sync	The status of synchronization
A	The communication partner Alice
B	The communication partner Bob
H	The hash function
h_A	The hash value of W of Alice
h_B	The hash value of W of Bob
left	The low limit of binary search
right	The high limit of binary search

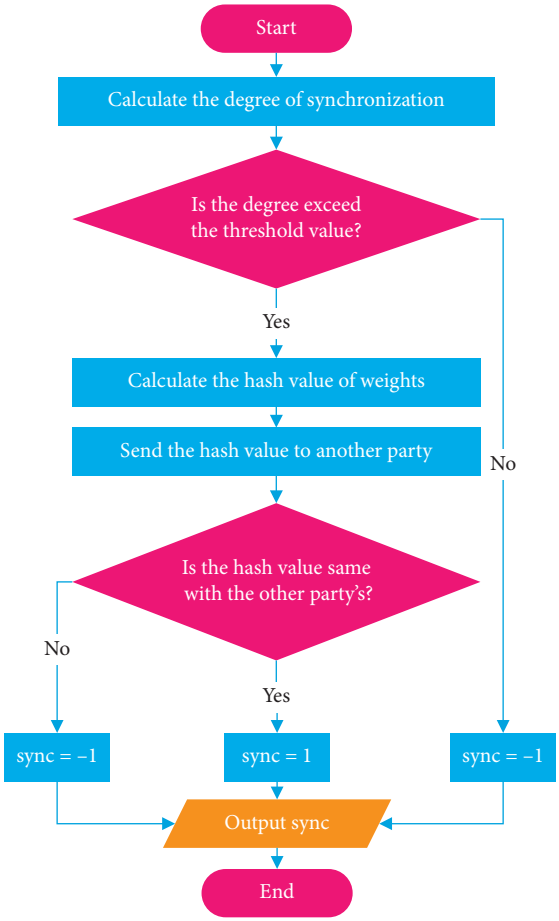

Figure 2: : The evaluation of the synchronization.

The Algorithm 1 accepts s, m, and a_m as the input and output the frequency b_m. The variable s is a natural number that should be set a proper value. The variable m is the learning steps. If Alice and Bob have the same output in m^{th} steps, then $a_m = 1$. If Alice and Bob have different outputs in m^{th} steps, then $a_m = 0$. If the learning steps m is less than s, the frequency is calculated by equation $b_m = (1/s)\sum_{j=1}^{m} a_m$. If the learning steps m is larger than s, the frequency is calculated by equation $b_m = (1/s)\sum_{j=m-s+1}^{m} a_m$.

Then, we compare b_m with the threshold value t. If $b_m < t$, the process of evaluation will be terminated and return the status sync with -1. sync $= -1$ means that Alice and Bob have not achieved full synchronization and will continue to learn with each other. sync $= 1$ means that Alice and Bob have made full synchronization and will stop the process of synchronization. If $b_m > t$, it will go to the next phase.

4.1.2. Precise Evaluation Phase. Only if the degree of synchronization exceeds the threshold value, it starts this phase. Then, Alice and Bob evaluation whether they have achieved full synchronization with the help of the hash function. Here, we take Alice as an example to show how the full synchronization is judged. The other party, Bob, should perform the same operation as Alice. The evaluation algorithm is shown in Algorithm 2.

Input: s, m, a_m
Output: b_m
Step 1: **if** $s < m$, **then**
Step 2: calculate $b_m = (1/s)\sum_{j=m-s+1}^{m} a_m$
Step 3: **else**
Step 4: calculate $b_m = (1/s)\sum_{j=1}^{m} a_m$
Step 5: **end**

Algorithm 1: Rough evaluation.

Input: t, b_m, **W**, h_B
Output: sync
Step 1: **if** $b_m \geq t$, **then**
Step 2: calculate $h_A = H(\mathbf{W})$
Step 3: **if** $h_A = h_B$ **then**
Step 4: sync = 1
Step 5: **else**
Step 6: sync = −1
Step 7: **end**
Step 8: **else**
Step 9: sync = −1
Step 10: **end**

Algorithm 2: Precise evaluation.

Input: none
Output: left, right
Step 1: let $s = 25$, left = 25
Step 2: calculating the number of misjudgment
Step 3: **while** the number does not exceed 0
Step 4: left = s
Step 5: $s = s + 50$
Step 6: calculating the number of misjudgments
Step 7: **end**
Step 8: right = s

Algorithm 3: Boundary searching.

The Algorithm 2 accepts t, b_m, W, h_B as input and output the synchronization status sync. The variable t is a threshold value that should be adequately selected and set. The variable b_m is the output of Algorithm 1. The variable W here represents the weight vector of Alice. The variable h_B is the hash value of the weight vector of Bob. At the beginning of Algorithm 2, Alice compares the output of Algorithm 1 and the threshold value. If $b_m \geq t$, Alice calculates the hash value of its own weight vector and compares it with the hash value of the weight vector of Bob. If $h_A = h_B$, the status sync will be returned with 1. Otherwise, the status sync will be returned with -1. If $b_m < t$, the status sync will also be returned with -1.

While the precise evaluation phase is completed, the neural key exchange protocol obtains the status of

Input: left, right
Output: the optimal s
Step 1: **while** right − left < 2
Step 2: s = (left + right)/2
Step 3: calculating the number of misjudgment
Step 4: **if** the number exceeds 0, **then**
Step 5: left = s
Step 6: **else**
Step 7: right = s
Step 8: **end**
Step 9: **end**

Algorithm 4: Binary searching.

synchronization. Then, if sync = 1, which means that Alice and Bob have achieved full synchronization, the neural key exchanged protocol will be finished. If sync = −1, which means that Alice and Bob have not achieved full synchronization, the neural key exchange protocol will continue until the two communication partners have achieved full synchronization.

4.2. The Algorithm for Finding the Optimal Parameter. The optimal s must satisfy two conditions. First, there is no misjudgment. Second, the delay in finding the full synchronization is kept as small as possible. In the later section, we have shown that the number of delayed steps in finding full synchronization increases linearly with the increment of s, and the number of misjudgments decreases with the increment of s. Hence, the minimum s which makes no misjudgment occur can be used as the optimal s. Here, an algorithm for searching the optimal s is proposed. The algorithm contains two phases: boundary searching phase and binary searching phase. The detail of each phase is described as follows.

4.2.1. Boundary Searching Phase. At the beginning, we let s start with 25. And then, if the number of misjudgments for given s exceeds 0, s will be increased by 50 each time until the quantity equals 0. The detail can be seen in Algorithm 3. It accepts no input and output the parameters left and right, which will be used in the next phase.

4.2.2. Binary Searching Phase. Then, the binary searching method is adopted to select the optimal s. The detail can be seen in Algorithm 4. It accepts the parameters left and right as the input and output the optimal s. The purpose of the algorithm is to find the minimum value, which makes no misjudgment occur between left and right quickly.

5. Results and Discussion

In this section, we analyzed the method proposed in [1] and the method proposed in [2] through simulation experiments. And we also analyzed the performance of the method proposed in this paper. The comparative analysis between the proposed method and other methods has also been made. All the algorithms are implemented in python. The code environment used in this paper is shown as follows: the computer is Sugon W560-G30. The operating system is Ubuntu 20.04 LTS with 64 bits. The memory is 62.6 GiB. The CPU is Intel® Xeon(R) Sliver 4110CPU@2.10 GHz with 4 core and 32 threads.

5.1. The Optimal Parameter. The idea of the method proposed by Dolecki et al. [1] is to calculate the frequency that Alice and Bob have the same output in previous s steps. Here, we show that the misjudgment will occur if the parameter s is not chosen properly. This is because there is some probability that the case where all of the outputs are equal between Alice and Bob in previous s steps, though Alice and Bob have not achieved full synchronization. In addition, the less the s is, the larger the number of misjudgment is. For example, as s increases from 25 to 225, we simulated 10000 samples for different combinations of K, N, and L: 3-101-1, 3-101-2, 3-101-3, 3-101-4, 3-101-5, 3-101-6, and 3-101-7 (Figure 3). For each combination of K, N, and L, the number of misjudgment decreases with the increment of s. While for a fixed s, the number of misjudgment increases with the increment of L.

We also show that the method proposed by Dolecki et al. [1] has a delay in finding the full synchronization of two neural networks. Therefore, if the full synchronization is achieved at the current step, while the two communication partners have one or more output that is different in previous s steps, the frequency will be less than 1. Hence, the learning between Alice and Bob will continue until there is no different output between Alice and Bob in previous s steps. For example, for $K = 3$, $N = 101$, and $L = 3$, if $s = 75$, the steps using cosine of weight vectors are 558, while the steps using frequency are 603. There are 45 delayed steps (Figure 4(a)). If $s = 125$, the steps using the cosine of weight vectors are 397, while the steps using frequency are 496. There are 99 steps that delayed (Figure 4(b)).

We use the cosine of the weight vector to find full synchronization at the first place and record the current steps. We also record the steps when their method judges full synchronization. Then, we compare the two steps and calculate the delay. We set five different values for s: 25, 75, 125, 175, and 225. We set several different values for K, N, and L: 3-101-1, 3-101-2, 3-101-3, 3-101-4, 3-101-5, 3-101-6, and 3-101-7. We simulate 10000 samples and calculate the average delayed steps for each case. In Figure 5, when K, N, and s are fixed, the number of average delayed steps decreases with the increment of L. This is because of the misjudging that has been analyzed before. When K, N, and L are fixed, the number of average steps increase linearly with the increment of the parameter s.

Therefore, selecting an optimal s is very important for evaluating the degree of synchronization. We use the algorithm proposed in Section 3 to select the optimal s for each combination of K, N, and L. The results are shown in Table 3.

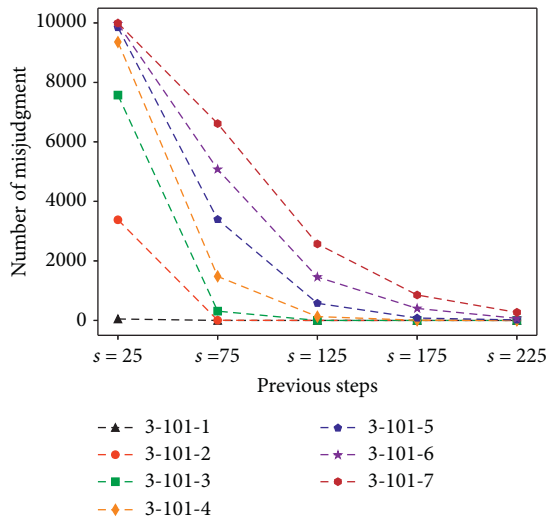

Figure 3: The number of misjudgment.

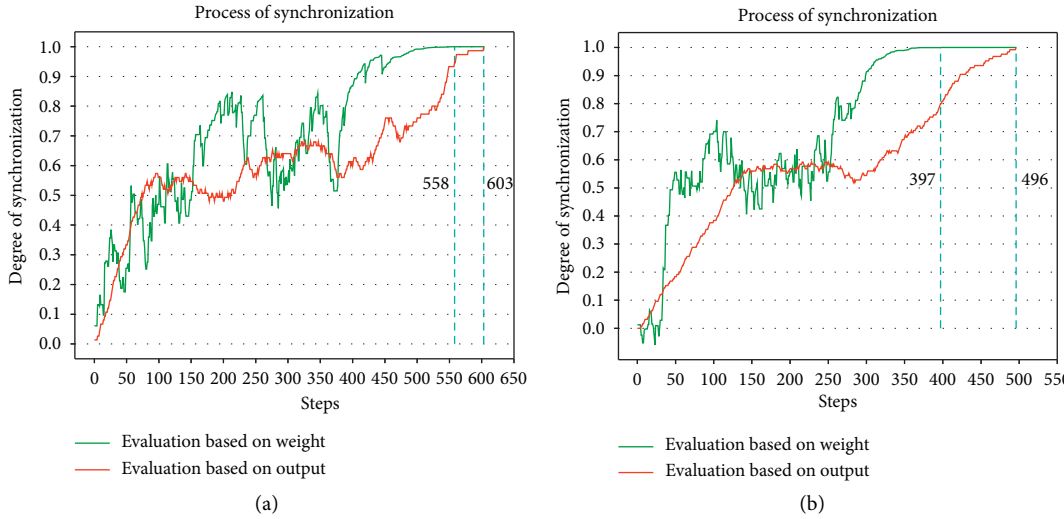

Figure 4: The delay using frequency ($K = 3$, $N = 101$, and $L = 3$). (a) $s = 75$; (b) $s = 125$.

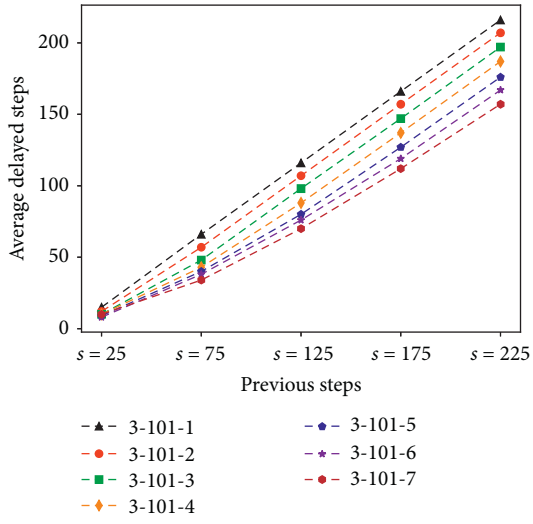

Figure 5: The average delayed steps.

5.2. Security Analysis. The delay in finding the full synchronization of neural networks may cause a security issue of neural key exchange. And the higher the number of delayed steps is, the less secure the neural key exchange is. That is, because the delayed steps can enhance the ability of the attacker. In this section, we made a comparative analysis of delayed steps and security among the method proposed in [1, 2] and this paper.

The evaluating method proposed in [1] is to use the hash function when the learning steps exceed the step threshold, which is the average learning steps of the fixed K, N, and L. The method proposed in this paper is to use the hash function when the rough degree of synchronization exceeds the degree threshold. The degree threshold means the average degree calculated by the rough evaluation phase while full synchronization occurs. In Table 4, the step threshold and the degree threshold were obtained by averaging 10000 simulations.

Table 3: The optimal s.

K	N	L	The optimal s
3	101	1	43
3	101	2	101
3	101	3	158
3	101	4	217
3	101	5	351
3	101	6	454
3	101	7	518

Table 4: The parameters used in the experiment.

K	N	L	Step threshold	Degree threshold
3	101	1	42.78	0.59
3	101	2	141.97	0.72
3	101	3	301.94	0.80
3	101	4	517.77	0.85
3	101	5	791.12	0.85
3	101	6	1106.95	0.89
3	101	7	1482.51	0.90

The delayed steps of each method are compared in Table 5. The parameter K is fixed as 3 and N is fixed as 101. The corresponding thresholds used in the simulation are presented in Table 4. The delayed steps are calculated by averaging 10000 simulations. However, the number of delayed steps in all three methods increases with the increment of L. The number delayed steps of the method proposed in this paper is the smallest among the three methods for each L (Table 5).

Then, we use the geometric attack to test the security of the neural key exchange protocol which adopts the evaluating method proposed in [1, 2] and this paper, respectively. We execute 10000 simulations for each method. The success probability of a geometric attack is shown in Table 6. For each method, the success probability decreases with the increment of L. When L increases to 4, the probability is very low. When L exceeds 4, the geometric attacker cannot break the neural key exchange protocol. For the fixed value of L, the probability of our method is the lowest, while the probability of the method in [1] is the highest. For example, the probability of our method is 25.52% lower than the method in [1] and 4.66% lower than the method in [2].

However, the probability gap between our method and the method in [1] decreases with the increment of L. The same phenomenon also occurs between the method in [2] and our method. This is also consistent with the conclusion in [9] that the success probability of geometric attack is reduced exponentially as L increases. The main reason is that the average delayed steps of each method grow more slowly than the average learning steps required for the two neural networks to achieve full synchronization. When the value of L is increased from 1 to 7 by 1 in each step, the corresponding average learning steps required are 42.87, 141.97, 301.94, 517.77, 791.12, 1106.95, and 1482.51, and the average delayed steps is showed earlier in Table 5. The average learning steps required for the two neural networks to achieve full synchronization go up exponentially as L increases (Figure 6).

Table 5: Comparison of the delayed steps.

Method	The delayed steps						
	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$	$L = 6$	$L = 7$
Reference [1]	33.67	82.69	129.80	178.72	302.39	393.23	446.70
Reference [2]	12.34	33.41	64.27	99.00	139.42	177.20	225.26
Ours'	5.77	10.51	17.72	25.44	42.40	71.73	79.87

Table 6: Success probability of geometric attack.

Method	Probability						
	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$	$L = 6$	$L = 7$
Reference [1]	53.05%	5.90%	0.48%	0.01%	0.0%	0.0%	0.0%
Reference [2]	32.19%	4.01%	0.37%	0.01%	0.0%	0.0%	0.0%
Ours'	27.53%	3.41%	0.32%	0.01%	0.0%	0.0%	0.0%

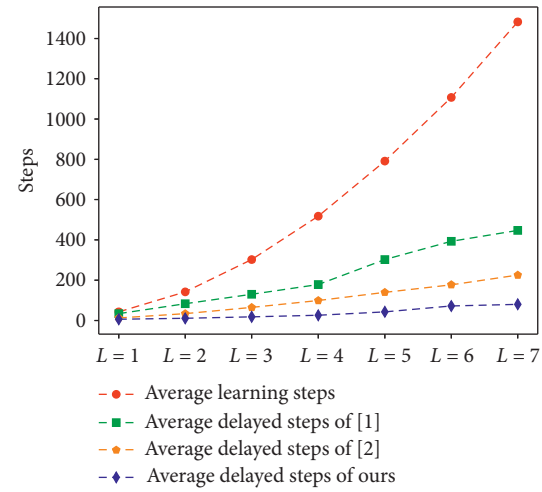

Figure 6: The average steps.

While the average delayed steps of each method only go up linearly as L increases. Since the average delayed steps grow more slowly than the average learning steps, the proportion of average delayed steps in average learning steps is decreasing. For example, when L is 1, the percentage of average delayed steps in the number of averaging learning steps in [1, 2] and methods are 78.54%, 28.78%, and 13.46%, respectively. When L is 5, the percentage of average delayed steps in the number of averaging learning steps in [1, 2] and methods are 38.17%, 17.62%, 5.36%, respectively. The impact of the delayed steps on the security of neural key exchange is decreasing. In Figure 6, our method has the slowest increase in the number of average delayed steps. While the number of average delayed steps in [2] is faster than that in our method and slower than that in [1]. This is why the success

probability of geometric attack of our method is lower than that in [1, 2].

6. Conclusion

The evaluation of synchronization is a significant part of the neural key exchange. The timing of finding full synchronization has a substantial impact on security. In this paper, an improved method to evaluate synchronization between two neural networks has been proposed. The combination of frequency that both the two communication partners have the same output and the hash function is used. The algorithm for finding the optimal critical parameter is also presented. The proposed method has been implemented and tested in the neural key exchange protocol. The experimental results show that the delay in finding full synchronization using the method in this paper is less than the method in [1, 2]. The neural key exchange protocol using the proposed method is more resistant to the geometric attack than them. Therefore, the security of neural key exchange protocol can also be improved by using the proposed method. However, there are still a few delays in finding full synchronization. In the future, two main aspects need to be further improved. One is to reduce the delay further. The other is to stop the long-time synchronization and start a new synchronization, which is a short-time synchronization with high probability.

Data Availability

The simulation data used to support this study are included within this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61572521) and the Innovative Team Foundation of Engineering University of PAP (KYTD201805).

References

[1] M. Dolecki, R. Kozera, and K. Lenik, "The evaluation of the TPM synchronization on the basis of their outputs," *Journal of Achievements in Materials and Manufacturing Engineering*, vol. 57, no. 2, pp. 91–98, 2013.

[2] Y. L. Liu, "Schemes for neural synchronization based on tree parity machine," Chongqing University, Chongqing, China, 2014.

[3] B. ÖzÇakmak, A. Özbİlen, U. Yavanoğlu, and K. CİN, "Neural and quantum cryptography in big data: a review," in *Proceedings of the 2019 IEEE International Conference on Big Data*, pp. 2413–2417, Los Angeles, CA, USA, December 2019.

[4] M. M. Alani, "Applications of machine learning in cryptography: a survey," in *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, pp. 23–27, Kuala Lumpur, Malaysia, January 2019.

[5] Z. K. Abdalrdha, I. H. Al-Qinani, and F. N. Abbas, "Subject review: key generation in different cryptography algorithm," *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 6, no. 5, pp. 230–240, 2019.

[6] D. Protic, "Neural cryptography," *Vojnotehnicki Glasnik*, vol. 64, no. 2, pp. 483–495, 2016.

[7] P. P. Hadke and S. G. Kale, "Use of neural networks in cryptography: a review," in *Proceedings of the 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pp. 1–4, Coimbatore, India, February 2016.

[8] I. Kanter, W. Kinzel, and E. Kanter, "Secure exchange of information by synchronization of neural networks," *Europhysics Letters (EPL)*, vol. 57, no. 1, pp. 141–147, 2002.

[9] A. Klimov, A. Mityagin, and A. Shamir, "Analysis of neural cryptography," in *Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 288–298, Queenstown, New Zealand, December 2002.

[10] L. N. Shacham, E. Klein, R. Mislovaty, I. Kanter, and W. Kinzel, "Cooperating attackers in neural cryptography," *Physical Review E*, vol. 69, no. 6, Article ID 066137, 2004.

[11] A. Ruttor, W. Kinzel, and I. Kanter, "Neural cryptography with queries," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2005, no. 1, Article ID P01009, 2005.

[12] S. Santhanalakshmi, T. Sudarshan, and G. K. Patra, "Neural synchronization by mutual learning using genetic approach for secure key generation," in *Proceedings of the International Conference on Security in Computer Networks and Distributed Systems*, pp. 422–431, Thiruvanathapuram, India, March 2014.

[13] S. Santhanalakshmi, K. Sangeeta, and G. K. Patra, "Analysis of neural synchronization using genetic approach for secure key generation," in *Proceedings of the International Symposium on Security in Computing and Communication*, pp. 207–216, Kochi, India, August 2015.

[14] A. M. Allam, H. M. Abbas, and M. W. El-Kharashi, "Authenticated key exchange protocol using neural cryptography with secret boundaries," in *Proceedings of the 2013 International Joint Conference on Neural Networks*, pp. 1–8, Dallas, TX, USA, December 2013.

[15] S. Chourasia, H. C. Bharadwaj, Q. Das, K. Agarwal, and K. Lavanya, "Vectorized neural key exchange using tree parity machine," *Compusoft*, vol. 8, no. 5, pp. 3140–3145, 2019.

[16] S. K. Pal, S. Mishra, and S. Mishra, "An TPM based approach for generation of secret key," *International Journal of Computer Network and Information Security*, vol. 11, no. 10, pp. 45–50, 2019.

[17] T. Dong and T. Huang, "Neural cryptography based on complex-valued neural network," *IEEE Transactions on Neural Networks and Learning Systems*, p. 1, 2020.

[18] S. D. Édgar, F. Walter, and L. Edison, "On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key," *Security and Communication Networks*, vol. 2019, no. 3, pp. 1–10, 2019.

[19] A. Ruttor, W. Kinzel, and I. Kanter, "Dynamics of neural cryptography," *Physical Review E*, vol. 75, no. 5, Article ID 056104, 2007.

[20] M. Gupta and M. Deshmukh, "Single secret image sharing scheme using neural cryptography," *Multimedia Tools and Applications*, vol. 79, no. 17-18, pp. 12183–12204, 2020.

[21] A. Sarkar, "Multilayer neural network synchronized secured session key based encryption in wireless communication," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 1, pp. 44–53, 2019.

[22] E. Shishniashvili, L. Mamisashvili, and L. Mirtskhulava, "Enhancing IoT security using multi-layer feedforward neural network with tree parity machine elements," *International Journal of Simulation Systems, Science & Technology*, vol. 21, no. 2, pp. 371–383, 2020.

[23] M. Mehic, M. Niemiec, H. Siljak, and M. Voznak, "Error reconciliation in quantum key distribution protocols," in *Proceedings of the International Conference on Reversible Computation*, pp. 222–236, Oslo, Norway, July 2020.

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

[25] M. Niemiec, M. Mehic, and M. Voznak, "Security verification of artificial neural networks used to error correction in quantum cryptography," in *Proceedings of the 26th Telecommunications Forum (TELFOR)*, pp. 1–4, Belgrade, Serbia, November 2018.

[26] A. Ruttor, "Neural synchronization and cryptography," pp. 279–345, University of St Gallen, St. Gallen, Switzerland, 2006, Business dissertations.