Mutual learning-based efficient synchronization of neural networks to exchange the neural key

Arindam Sarkar¹

Received: 1 December 2020 / Accepted: 17 March 2021 / Published online: 29 April 2021
© The Author(s) 2021

Abstract

Synchronization of two neural networks through mutual learning is used to exchange the key over a public channel. In the absence of a weight vector from another party, the key challenge with neural synchronization is how to assess the coordination of two communication parties. There is an issue of delay in the current techniques in the synchronization assessment that has an impact on the security and privacy of the neural synchronization. In this paper, to assess the complete coordination of a cluster of neural networks more efficiently and timely, an important strategy for assessing coordination is presented. To approximately determine the degree of synchronization, the frequency of the two networks having the same output in prior iterations is used. The hash is used to determine if both the networks are completely synchronized exactly when a certain threshold is crossed. The improved technique makes absolute coordination between two communication parties using the weight vectors' has value. In contrast, with existing approaches, two communicating parties who follow the proposed approach will detect complete synchronization sooner. This reduces the effective geometric likelihood. The proposed method, therefore, increases the safety of the protocol for neural key exchange. This proposed technique has been passed through different parametric tests. Simulations of the process show effectiveness in terms of cited results in the paper.

Keywords Neural synchronization · Session key · Three layer tree parity machine (TLTPM) · Neural network · Hash function

Introduction

Ensuring the security of information in the modern age is very important. There are a variety of strategies that can accomplish this goal. Symmetric key cryptography and asymmetric key cryptography are the most widely used. Asymmetric cryptographic efficiency is lower than symmetric cryptographic efficiency. Therefore, from a symmetric cryptography algorithm, a neural synchronization is built and the neural key produced being used for encryption and decryption with an asymmetric algorithm. The key exchange technique, however, is mostly depended on numerical computation, that involves immense memory and processing. There is also a fresh solution through the neural networks known as the neural key exchange to obtain a symmetric key. Through mutual learning, two neural networks can achieve

synchronization. They get a shared input vector and give their respective output to one another. The corresponding weight vectors of both networks are, however, utilized as an asymmetric key after synchronization that could be used to encode and decode. One of the significant issues in the neural exchange is the degree of synchronization to which the two networks are synchronized without the other party's weight information. Methods for calculating the cosine and the distance between the weights using the Euclidean rule are not used due to the knowledge about the weight vector of the other party is not known. Dolecki and Kozera [9] proposed a method for assessing the synchronization degree in fixed prior synchronization learning measures, by measuring the frequent equivalent performance. But there is a delay in finding complete synchronization between the two networks, so the two networks continue to learn while they are synchronized entirely. The successful intruder discloses additional details from the further steps of learning. Liu et al. [18] suggested a method for evaluating neural network synchronization with hash. Their method will precisely find complete synchronization. However, information traffic will increase. In exceptional cases, there will be considerable

✉ Arindam Sarkar
arindam.vb@gmail.com

¹ Department of Computer Science and Electronics,
Ramakrishna Mission Vidyamandira, Belur Math, Howrah,
West Bengal 711202, India

communication traffic if the hash feature is used inappropriately. The two networks' synchronization must be completed early and the learning must be stopped.

The main problem domain is stated as follows:

1. The method for evaluating the synchronization of the neural key exchange protocol must be established to improve security and reliability.
2. In conventional synchronization evaluation techniques, such as the hashing algorithm and the frequency solution, there is a significant lag in the calculation of full synchronization. The geometric attacker disrupts neuronal synchronization due to the long delay.
3. The efficacy of neural synchronization is primarily harmed by long delays. To reduce the rate of delay and error, a better solution is needed.
4. For the frequency approach, there is still the risk of making a mistake. If any communicating partner makes a mistake, the neural key exchange protocol must be re-performed.

The paper's contributions, thus are:

1. This paper aims to increase protection and reliability by developing the procedure for assessing the synchronization of the neural key exchange protocol.
2. The synchronization assessment method, therefore, plays a key role in the key exchange.
3. This paper proposes a new method for reducing the rate of delay and error.
4. The merits of the hashing technique and the efficiency of the neural network are combined.
5. It implies a more accurate way of assessing neural network synchronization. The proposed method will rapidly and efficiently achieve full synchronization. Delays would be reduced as a result of the improved technology.
6. It introduces a technique for the enhanced assessment process to look for an ideal parameter. The improved method's misappreciation rate is minimized by the optimum parameters. The experimental findings show that in the proposed process, the delayed steps are smaller than the other existing approaches and that the rate of error is almost negligible. It is possible to enhance the security and privacy of the neural synchronization which adopts this efficient algorithm.

The rest of this paper is organized accordingly. Section 2 deals with related works. Neural synchronization is explained in Sect. 3. Section 4 deals with the proposed methodology. Sections 6 and 7 deal with security analysis and the results respectively. Conclusions are given in Sect. 7 and references are given at the end.

Related works

Artificial neural networks have been used for cryptography Zçakmak et al. [39], Alani [2], Abdalrdha et al. [1], Protic [26], Hadke and Kale [12]. A lot of work has been done on this. A neural synchronization was first proposed in Kanter et al. [14] and implemented with simple parameters based on neural networks. The authors analyzed with naïve attackers the protection of the protocol. The authors also introduced bit-package techniques and extended the protocol to speed up the synchronization of neural networks. Klimov et al. [16] reviewed the protocol proposed in Kanter et al. [14] and described why the two communicating parties could reach complete weight synchronization. The findings reveal that although the same network configuration, input, and learning rules are implemented, the naïve attacker is unable to reach absolute synchronization with the two communicating partners. Then three kinds of attacks, including genetics, geometric attacks, and probabilistic attacks, were introduced by them. Therefore, the security of the protocol relies on neural network structure parameters. Shacham et al. [37] suggested a cooperative technique that can co-operate with genetic, geometric, and probabilistic attacks, to prevent the depth of the synapse from influencing the likelihood of success of attackers. Ruttur et al. [28] performed comparative analyses between genetic, geometric, and majority attacks. The results of the simulation show that the neural synchronization ensures that the depth of the synapse increases. Unlike the random common input, they have suggested a system that uses queries that can boost the protection of the key exchange protocol. In the assessment of the degree of coordination between two communicating neural networks, cosine similarity was implemented by Ruttur et al. [29]. The protocol proposed in Kanter et al. [14] was introduced by Desai et al. [8] to perform secret sharing to encode and decode the image. The authenticated key exchange protocol was suggested by Allam et al. [5], using a shared key as a learning boundary. The dynamic learning rate and random learning rules are adopted in this protocol. This protocol has the advantage that the protection of neural key exchanges can be strengthened without compromising efficiency. The efficiency of this protocol was later evaluated by the authors Santhanalakshmi et al. [31]. An effective and novel solution to malware identification has been suggested by Alazab et al. [3]. For malware feature discovery, the proposed methodology uses a hybrid wrapper-filter model that incorporates Maximal Importance filter heuristics and Artificial Neural Net Input Gain Measurement Approximation (ANNIGMA) wrapper heuristic for sub-set selection by capitalizing on the strengths of each classifier. It injects into the wrapper process the intrinsic features of data collected by the filter and integrates this with the heuristic score of the wrapper. In exchange, this will decrease the search space and direct the search for the most relevant

Table 1 The strengths and drawbacks

Technique	Delay	Additional traffic	Misconduct	Convenience
Frequency	Yes	Yes	Yes	Yes
Similarity of cosine	No	No	No	No
Euclidean distance	No	No	No	No
Hash function	Yes	Yes	No	Yes

detection-assisting malware functionality. Dolecki and Kozera [9] showed that synchronization can be measured using the frequency at which both communication parties have the same output. The results of the simulation show that relationships are very strong between the frequency system, cosine similarity, and Euclidean distance. In the synchronization of neural networks, Santhanalakshmi et al. [32] implemented the genetic algorithm to find optimal weights as initial weights. By reducing learning time and phases, this approach will accelerate synchronization. The genetic algorithm and tree parity machine parameters were studied in Santhanalakshmi et al. [32]. Their findings show that the number of hidden layers of the TPM will boost protection. Chourasia et al. [7] have suggested a vectored neural synchronization for key exchange. Pal et al. [24] suggested a rule of learning that could speed up synchronization and improve key randomness. Dong and Huang [10] proposed a complex value-based neural network for neural cryptography. Here all the inputs and the outputs are complex values. But, this technique takes a significant amount of time to complete the synchronization process. Édgar Salguero Dorokhin et al. [30] have proposed a method for finding the optimal TPM configuration, thus allowing a more effective and stable neural key exchange protocol. It will assess complete synchronization by using Alice and Bob's weights. They do not have any real information on the weight vector of each other. The measurement of the distance between weights using the rule of Euclidean is not even technically applicable. Sarkar and Mandal [34], Sarkar [33], Sarkar and Mandal [35], Mandal and Sarkar [19], Sarkar et al. [36] proposed schemes that enhanced the security of the protocol by enhancing the synaptic depths of TPM and henceforth counteracting the attacks of the brute power of the attacker. It is found that the amount of security provided by the TPM synchronization also can be improved by introducing a large set of neurons and entries of each neuron in the hidden layers. To forecast smart grid reliability, a new Multidirectional Long Short-Term Memory (MLSTM) model has been proposed by Alazab et al. [4]. Compared with conventional Machine Learning (ML) models such as Long Short Term Memory (LSTM), Gated Recurrent Units (GRU), Recurrent Neural Networks, the output of MLSTM (RNN). The proposed model attained 99.07% accuracy in training and testing, 3% better than other conventional deep learning models. Gadekallu et al. [11] proposed a method to proactively take the necessary steps to tackle agricultural

crises, the emphasis was on applying the machine learning model to identify the tomato disease image dataset. The dataset is gathered from the freely accessible plant-village dataset in this article. Using the hybrid-principal component analysis-Whale optimization algorithm, the essential features are extracted from the dataset. To distinguish intrusion detection system (IDS) datasets, a hybrid principal component analysis (PCA)-firefly focused machine learning model was suggested by Bhattacharya et al. [6]. From Kaggle, the dataset used in the analysis is obtained. For the transformation of the IDS datasets, the model first performs One-Hot encoding. Then, for dimensionality reduction, the hybrid PCA-firefly algorithm is used. For classification, the XGBoost algorithm is applied to the reduced dataset. Shishniashvili et al. [38] suggested a strategy to make use of the portions of the synchronized weight as a session key rather than the whole weight vector. In quantum key distribution protocols, Mehic et al. [20] synchronization of two neural nets can serve as a reconciliation of errors as well. Niemiec [21] suggested the new approach of using the tree parity machine to address the mistakes in the quantum key distribution protocol during transmission. The effect of the variance of parameters on security and the performance of the various efficiency learning rules was analyzed in Niemiec et al. [22].

The following four methods of synchronization were mainly implemented in these schemes; frequency of the identical output from the two partners, the similarity of cosine, Euclidean distance, and hash function. Table 1 lists the strengths and drawbacks of the four approaches.

The methods were evaluated by considering four different aspects: additional traffic, misconduct, convenience, and delay. Here, the delay in the indicator indicates that the whole synchronization cannot be identified in the first step. The synchronization has been deterred and the neural key swap over is only accomplished after several phases of true synchronization. By the delay, the additional traffic indicator increases the communication traffic. Indicator error of judgment implies that while true synchronization is not accomplished, the key can not be transferred. Whether the approach can be used in practice is a practical predictor.

In the methods of calculating the cosine for measurement of similarity index and the distance between the weights using the Euclidean rule, there are no problems of misconceptions, nor even delay, or increased traffic. Even though they have several benefits, their crushing downside is they

are not realistic. This is because the other network's weights are not accessible. The frequency and hashing scheme are helpful. But the use of both realistic approaches often leads to delays and additional traffic. The important problem with the hash function is when the hash function is being used. It will cause inefficiency when the hash function is used too early. When the hash function is used too late, the protection of neural synchronization will be reduced. The frequency approach also involves misjudgment.

Finally, for the protection of neural key exchange, the assessment of neural network synchronization is a critical matter. This paper integrated the advantage of the hash function with the neural networks' performance.

Python is used for the implementation of the technique and statistical analysis is done using R.

Neural synchronization

Two neural networks can participate in the synchronization procedure at a time. Both possess a special structure of neural network called tree parity machine (TPM) Ruttur [27]. Both tree parity machines can be completely synchronized by reciprocal learning. Then the same weight vector of a TPM can either be utilized as a seed for secret key generation, an encryption/decryption purpose, or as a session key. The protocol of neural synchronization includes five important phases: initialization, calculation, update, evaluation, and completion phase. Figure 1 displays the flow diagram of the neural synchronization and explains the specifics of each step as follows.

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

Proposed methodology

In this section, a method of evolution of synchronization, optimum parameter finding strategy have been analyzed.

Proposed method of evolution of synchronization

Four main methods can determine the synchronization of the weight vector cosines, the distance measure of the weight vector using Euclidean rule, the identical outputs' frequency, and the weight vector's hash function. Table 1 deals with the drawbacks of the aforementioned methods. In the results and discussion section, it is assessed by simulation. It is therefore essential to assess in a more suitable and timely manner the synchronization of the neural networks. An improved approach that can find complete synchronization in the first place is presented here. Table 2 lists the parameters used in this article.

The core principle of the proposed solution is to approximately determine the degree of synchronization by estimating the identical output's frequency for both Arindam and Moumita. Depending on the degree, the weights' hash value are computed. And then it is specifically calculated by evaluating the values of the hash if the weights are equal.

The phase of rough assessment

Both Arindam and Moumita do not know each other's weight vectors. The only information available is the output transmitted over the public channel. However, to assess the degree of coordination, the frequency at which both parties have the identical output will then be utilized. Algorithm 1 illustrates the algorithm for estimation. Algorithm 1 takes z , n , and a_n as input and produces the frequency b_n as an output. The z parameter is a natural number that has to be set to the correct value. n is the step of learning. If Arindam and Moumita in n th steps have the same output, then $a_n = 1$. If in n th steps Arindam and Moumita have different outputs, then $a_n = 0$. The frequency is determined by the equation $b_n = \frac{1}{z} \sum_{j=1}^n a_n$, if the learning step n is less than z . The frequency is determined by the equation $b_n = \frac{1}{z} \sum_{j=n-z+1}^n a_n$, if the learning steps n is greater than z .

The algorithm equates b_n to the threshold value of t . If $b_n < t$, the evaluation process will be discontinued and the status syn will be restored to 1. $syn = -1$ denotes that both parties yet not attained absolute coordination and will need to continue the learning further. The $syn = 1$ indicates that both parties have achieved complete synchronization and the synchronization procedure would then stop. If $b_n > t$, in the next step, it goes to. Algorithm 1 illustrates the rough assessment process.

Fig. 1 Neural synchronization process

Algorithm 1:

Input: z, n, a_n

Output: b_n

if $z < n$ **then**

$$b_n = \frac{1}{z} \sum_{j=n-z+1}^n a_n$$

else

$$b_n = \frac{1}{z} \sum_{j=1}^n a_n$$

end if

The precise process of assessment

This phase will begin only after the degree of coordination crosses the preset limit. Then with the aid of the hash function, both parties decide if they have reached an absolute correlation. This paper uses Arindam here, as an illustration, to demonstrate how to test full synchronization. It is assumed that another party, Moumita, will conduct the same protocol as Arindam. The assessment algorithm is seen in algorithm 2.

Table 2 Symbol description

Symbol	Description
M	The number of neurons in input layer
$Lrange$	Synaptic depth
H_1	The number of neurons in hidden layer 1
H_2	The number of neurons in hidden layer 2
H_3	The number of neurons in hidden layer 3
W	The neural network's weight vector
z	Earlier learning steps
n	In neural synchronisation, the cumulative number of learning steps
τ_n^A	Output of Arindam in n th steps
τ_n^B	Output of Moumita in n th steps
syn	Synchronization status
low	Lower limit of Binary search
high	Higher limit of Binary search
h_A	The hash value of Arindam's weight
h_B	The hash value of Moumita's weight
a_n	Value of checking whether $\tau_n^A = \tau_n^B$
b_n	The average value of a_n, a_{n-1}, a_{n-s+1}
t	Threshold value of the degree of synchronization
H	Hash Function
A	Communicating party Arindam
B	Communicating party Moumita

This algorithm accepts t, b_n, W, h_B as input, and outputs synchronization status. The factor t is a preset value that must be chosen and define accordingly. The outcome of Algorithm 1 is b_n . The W denotes Arindam's weight vector. The attribute h_B is Moumita's weight vector's hash value. Arindam contrasts the Algorithm 1 outcomes and the threshold value at the initiation of Algorithm 2. Following algorithm 2 illustrates the precise process of assessment.

Algorithm 2:**Input:** t, b_n, W, h_B **Output:** syn **if** $b_n \geq t$ **then** $h_A = H(W)$ **if** $h_A = h_B$ **then** $syn = 1$ **else** $syn = -1$ **end if****else** $syn = -1$ **end if****The optimum parameter finding algorithm**

Two conditions must be met for optimum z . First of all, no mistake. Second, as little time as possible to find complete

synchronization. In the following section, this paper demonstrated that a linear increase in z increases in the amount of delayed steps in the discovery of complete synchronization and a decrease in errors as z increases. Therefore, as ideal z , the minimum z that does not lead to error can be used. An algorithm is proposed here to look for the optimal z . There are two stages in the algorithm: the boundary search process and the binary search phase. Each phase is defined in detail as follows.

Boundary search process

Let z start with 25 at the beginning. Then, if the number of errors for z is greater than 0, z is increased by 50 each time the sum is equal to 0. In algorithm 3 one can see the details. It does not accept any input and provides low and high values as an output that will be used in the next step. Algorithm 3 illustrates the boundary search process.

Algorithm 3:**Input:** *none***Output:** *low, high*Let $z = 25, low = 25$

Calculate the total number of misjudgment

while $number \leq 0$ **do** $low = z$ $z = z + 50$

```

    syn = -1
    Calculate the total number of misjudgment
end while
high = z

```

The binary search process

The binary search method is then used to find the best z . In algorithm 4, details can be seen. As an input, it detects the low and high parameters and generates the optimum z as the output. The procedure aims are to search for the optimum value, that does not cause a rapid misjudgment between low and high to occur. Algorithm 4 illustrates the binary search process.

Algorithm 4:

Input: low, high

Output: optimal z

while high – low < 2 **do**

$$z = \frac{(low+high)}{2}$$

Calculate the total number of misjudgment

if number of misjudgment > 0 **then**

low = z

else

high = z

end if

end while

This special neural network TLTPM is composed of M no. of input neurons for each H no. of hidden neurons. TLTPM has only neuron in its output. TLTPM works with binary input, $x_{input_{u,v}} \in \{-1, +1\}$. The mapping between input and output is described by the discrete weight value between $-Lrange$ and $+Lrange$, $weight_{u,v} \in \{-Lrange, -Lrange + 1, \dots, +Lrange\}$.

In TLTPM u -th hidden unit is described by the index $u = 1, \dots, H$ and that of $v = 1, \dots, M$ denotes input neuron corresponding to the u -th hidden neuron of the TLTPM. Consider there are H_1, H_2, H_3 numbers of hidden units in the first, second, and third hidden layers respectively. Each hidden unit of the first layer calculates its output by performing the weighted sum over the present state of inputs on that particular hidden unit. Each hidden unit of the second layer calculates its output by performing the weighted sum over the hidden units of the first layer on that particular hidden unit. Similarly, Each hidden unit of the third layer calculates its output by performing the weighted sum over the hidden units of the second layer on that particular hidden unit.

Time complexity analysis

In neural synchronization technique initialization of the weight vector takes $(M \times H_1 + H_1 \times H_2 + H_2 \times H_3)$ amount of computations. For example, if $M = 2$, $H_1 =$

2, $H_2 = 3$, and $H_3 = 2$ then total numbers of synaptic links (weights) are $(2 \times 2 + 2 \times 3 + 3 \times 2) = 16$. So, it takes sixteen amounts of computations. Generation of M number of input vectors for each H_1 number of hidden neurons takes $(M \times H_1)$ amount of computations. Computation of the hidden neuron outputs takes $(H_1 + H_2 + H_3)$ amount of computations. Where H_1 , H_2 , and H_3 are the number of hidden units in 1st, 2nd, and 3rd layer respectively. Computation of final output value takes a unit amount of computation because it needs only a single operation to compute the value.

In the best case of the neural synchronization algorithm, Arindam's and Moumita's arbitrarily chosen weight vectors are identical. So, networks that are synchronized at the initial stage do not need to update the weight using the learning rule. Here, $((M \times H_1) + (M \times H_1 + H_1 \times H_2 + H_2 \times H_3) + (H_1 + H_2 + H_3))$ amount of computation is needed in the best case which is in form of $O(\text{Generation of common seed value} + \text{initialization of input vector} + \text{initialization of weight vector} + \text{Computation of the hidden neuron outputs})$. If the Arindam's and Moumita's arbitrarily chosen weight vectors are not identical then in each iteration the weight vectors of the hidden unit which has a value equivalent to the final output are updated according to the learning rule. This scenario leads to an average and worst-case situation where I number of iteration to be performed to generate the identical weight vectors at both ends. So, the total computation for the average and worst case is $((M \times H_1) + (M \times H_1 + H_1 \times H_2 + H_2 \times H_3) + (H_1 + H_2 + H_3)) + (\text{No. of iteration} \times \text{No. of weight updation})$ which is can be expressed in $O(\text{Time complexity in the first iteration} + (\text{No. of iteration} \times \text{No. of weight update}))$.

Security analysis

The delay in the search for complete neural network synchronization may result in a neural key exchange security problem. And the more delayed steps are less reliable in the exchange of neural key. In other words, the delayed steps will increase the attacker's ability. The delay steps and security of the existing approach in Dolecki and Kozera [9], Liu et al. [18] and this paper have been contrasted in this section.

The solution implied in Dolecki and Kozera [9] is for the hash function to be used where the learning steps surpass the step limit, i.e. the expected learning steps of specified H_1 , H_2 , H_3 , M , and $Lrange$. The proposed solution utilizes the hashing technique. The degree threshold corresponds to an average grade determined by the rough synchronization process.

Table 3 compares the delayed steps of each process. The parameter H_1 , H_2 , and H_3 are all set to 3, and the parameter M is set to 101. However, with the increase of $Lrange$, for these three strategies, the number of delayed steps increases.

Table 3 Delayed steps comparisons with proposed and existing techniques

Technique	$Lrange = 1$	$Lrange = 2$	$Lrange = 3$	$Lrange = 4$	$Lrange = 5$	$Lrange = 6$	$Lrange = 7$
Proposed	5.85	11.02	18.26	25.97	43.51	71.65	82.64
Dolecki and Kozera [9]	35.64	85.32	132.07	182.53	305.29	397.03	453.98
Liu et al. [18]	14.76	34.87	66.38	109.63	142.23	179.03	230.73

Table 4 Geometric attack's success probability

Technique	$Lrange = 1$ (%)	$Lrange = 2$ (%)	$Lrange = 3$ (%)	$Lrange = 4$ (%)	$Lrange = 5$ (%)	$Lrange = 6$ (%)	$Lrange = 7$ (%)
Proposed	28.32	4.12	0.51	0.01	0.0	0.0	0.0
Dolecki and Kozera [9]	55.73	6.1	0.57	0.01	0.0	0.0	0.0
Liu et al. [18]	33.49	4.24	0.39	0.01	0.0	0.0	0.0

Fig. 2 Number of misjudgment

There are three methods of each $Lrange$. The proposed method has a minimum amount of delayed steps (Table 3).

The geometric attack is then considered measuring the protection of the synchronization protocol using the assessment approach suggested in Dolecki and Kozera [9], Liu et al. [18] and this paper, respectively. For each process, 10,000 simulations have been performed.

Table 4 indicates the likelihood of the success of a geometric attack. The likelihood of success with the increase of $Lrange$ decreases with each process. The likelihood is very low as $Lrange$ rises to 4. If $Lrange$ is more than four, the neural synchronization protocol cannot be violated by the geometric attack. For $Lrange$'s fixed value, the proposed method's probability is the lowest, whereas the method's probability is the highest in Dolecki and Kozera [9]. For

Fig. 3 Delay using $H_1 = 3$, $H_2 = 3$, $H_3 = 3$, $M = 101$, and $z = 75$

instance, the proposed method's probability is 28.32% lower than in Dolecki and Kozera [9] and 4.12% lower than in the method Liu et al. [18]. However, with the increase of $Lrange$, the probability difference between the proposed method and the method Dolecki and Kozera [9] decreases. Between the Liu et al. [18] approach and the proposed method, the same phenomenon occurs.

This is also associated with the observation in Kanter et al. [14] that as $Lrange$ rises, the probability of success of a geometric attack is decreasing exponentially. The primary reason is that each approach's mean delayed steps are increasing very slowly than the expected learning steps taken to reach complete synchronization. The mean delayed steps are seen earlier in Table 5 if the $Lrange$ value is hiked from value 1 to 7. As $Lrange$ increases, the expected steps of learning needed for the two neural networks to reach optimal synchronization increases exponentially. Since each process's average deferred phases are only linear as $Lrange$ rises.

Consider n number of cascading encryption/decryption techniques are used to encrypt/decrypt the plaintext with the help of a neural synchronized session key. Then a session key of length [(number of cascaded encryption technique in bits) + (three bits combinations of encryption/ decryption technique index) + (length of i number of encryption/decryption keys in bits) + (length of n number of session keys in bits)]

i.e. $[8 + (3 \times n) + (128 \times n) + (128 \times n)]$ bits to $[8 + (3 \times n) + (256 \times n) + (256 \times n)]$ number of bits. So,
$$\frac{[8 + (3 \times n) + (128 \times n) + (128 \times n)]}{8} = \left[1 + \frac{(3 \times n)}{8} + 16n + 16n\right] = 32n$$
 to
$$\frac{[8 + (3 \times n) + (256 \times n) + (256 \times n)]}{8} = \left[1 + \frac{(3 \times n)}{8} + 32n + 32n\right] = 64n$$
 numbers of characters.

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

Consider any single encryption using the neural key of size 512 bit which is hypothetically approved and needed to be analyzed in the context of the time taken to crack a ciphertext with the help of the fastest supercomputers available at present. In this neural technique to crack a ciphertext, the number of permutation combinations on the neural key is $2^{512} = 1.340780 \times 10^{154}$ trials for a size of 512 bits only. IBM Summit at Oak Ridge, U.S. invented the fastest supercomputer in the world with 148.6 PFLOPS i.e. means 148.6×10^{15} floating-point computing/second. Certainly, it can be considered that each trial may require 1000 FLOPS to undergo its operations. Hence, the total test needed per second is 148.6×10^{12} . Total no. of sec. a year have = $365 \times 24 \times 60 \times 60 = 31,536,000$ s. Total number of

Fig. 4 Delay using $H_1 = 3$, $H_2 = 3$, $H_3 = 3$, $M = 101$, and $z = 125$

years for Brute Force attack: $(1.340780 \times 10^{154}) / (148.6 \times 10^{12} \times 31,536,000) = 2.86109 \times 10^{132}$ years.

Results and analysis

For result and simulation purposes, an Intel Core i7 10th Generation, 2.6 GHz processor, 16 GB RAM is used. Comprehensive and needful security views have been focused to affect acquaintance security and robustness issues. The precision of 10^{-15} decimal has been used in arithmetic operations according to the authenticated IEEE Standard 754. In this section, using simulation experiments both the method proposed in Dolecki and Kozera [9] and the method proposed in Liu et al. [18] are examined. Comparative analysis has been carried out between the proposed and existing approaches. Both algorithms have been implemented in python.

The idea of the approach proposed by Dolecki and Kozera [9] is to quantify the frequency at which previous phases had the very same output for Arindam and Moumita. Here, it is observed that if the parameter z is not properly chosen, an error can appear. As there is some likelihood that the case of all the outcomes being similar between two parties would not be entirely synchronized with Arindam and Moumita in previous phases. Also, the smaller the z ,

the higher the number of errors of judgment. For starters, we simulated 10,000 samples for different combinations of H_1 , H_2 , H_3 , M and $Lrange$ as z rises from 25 to 225: 3-3-3-101-1, 3-3-3-101-2, 3-3-3-101-3, 3-3-3-101-4, 3-3-3-101-5, 3-3-3-101-6, and 3-3-3-101-7 (Fig. 2). For each combination of H_1 , H_2 , H_3 , M and $Lrange$, with an increment of z , the volume of misjudgment decreases. This improves the efficiency of the proposed technique. Although the volume of error of judgment grows with the increase in $Lrange$ for specified z .

It has been seen it is delayed to find complete synchronization of two networks by the technique proposed by Dolecki and Kozera [9]. Therefore, if complete synchronization is accomplished at the present level when both communicating parties have one or more outputs that are dissimilar in the previous phase, the frequency value would be smaller than one. Both the parties will proceed the learning unless the previous steps do not yield any other results between Arindam and Moumita. The steps with the cosine of the vector are 558, for example, for $H_1 = 3$, $H_2 = 3$, $H_3 = 3$, $M = 101$, and $Lrange = 3$, while $z = 75$, for frequency steps, are 603. There are 45 delayed steps (Fig. 3).

If $z = 125$, the steps using the weight vector cosine is 397, but those steps are made use of frequency are 496. The total number of delayed steps is 99 (Fig. 4).

Fig. 5 Average delayed steps

This proposed technique utilizes the cosine method to first locate complete coordination and save the latest measures. When their process judges are fully coordinated, the steps are recorded. Then process equate the delay with the two phases and compute it. For z : 25, 75, 125, 175, and 225, the proposed method has set five different values. For H_1 , H_2 , H_3 , M , and $Lrange$, the process has specified a number of variables: 3-3-3-101-1, 3-3-3-101-2, 3-3-3-101-3, 3-3-3-101-4, 3-3-3-101-5, 3-3-3-101-6 and 3-3-3-101-7. The mean delayed steps of each case are determined using 10000 samples. In Fig. 5, when H_1 , H_2 , H_3 , M , and z are set, the amount of mean differed steps minimizes if the value of $Lrange$ increased. This is due to the previously analyzed misunderstandings. The number of mean steps is incremented linearly with the increase in the parameter z when H_1 , H_2 , H_3 , M , and $Lrange$ are fixed.

Therefore, it is necessary to select an optimal z for synchronization assessment. The optimum z for each combination of H_1 , H_2 , H_3 , M , and $Lrange$ are chosen using the proposed algorithm. The findings are presented in Table 5.

By passing the fifteen tests in that suite, real randomness was assured in the proposed transmitting technique. These tests are extremely helpful for a technique with such a high level of robustness as the one proposed. A probability value

Table 5 The optimal value of z

H_1	H_2	H_3	M	$Lrange$	Optimal z
3	3	3	101	1	45
3	3	3	101	2	103
3	3	3	101	3	161
3	3	3	101	4	222
3	3	3	101	5	357
3	3	3	101	6	459
3	3	3	101	7	520

(p_Value) determines the acceptance or rejection of the input vector. Table 6 contains the results of the NIST Statistical tests NIST [23] on the generated random input vector. This table also represents the comparison of p_Value between the proposed TLTPM and the existing CVTPM technique Dong and Huang [10]. From the Table 6, it has been seen that in the NIST statistical test p_Value of proposed TLTPM has outperformed than CVTPM.

The proportion of 1 and 0 in the coordinated neural session key is the result of the frequency test. The TLTPM procedure achieves a frequency test result of 0.696413, compared to 0.512374 for Kanso and Smaoui [13], 0.1329 for Karakaya

Table 6 Comparison of p _Value between proposed TLTPM and the existing CVTPM

NIST test	p _Value of proposed TLTPM	p _Value of CVTPM Dong and Huang [10]
Frequency	0.696413	0.512374
Frequency within a block	0.613725	0.538721
Runs	0.527912	0.489451
Longest run of ones in a block	0.098364	0.042081
Binary matrix rank	0.736108	0.417393
Discrete fourier transform	0.595301	0.392762
Non-overlapping template matching	0.519630	0.431759
Overlapping (periodic) template matching	0.369312	0.170426
Maurer's "universal statistical"	0.793612	0.689478
Linear complexity	0.698362	0.594917
Serial	0.594207	0.478902
Approximate entropy	0.364871	0.204175
Cumulative sums	0.716983	0.587290
Random excursions	0.459321	0.299875
Random excursions variants	0.368273	0.180542

Table 7 Comparison of p _Value of NIST Frequency Test

Techniques	p _Value of NIST Frequency test
Proposed TLTPM	0.696413
CVTPM Dong and Huang [10]	0.512374
Karakaya et al. [15]	0.1329
Patidar et al. [25]	0.632558
Liu et al. [17]	0.629806

et al. [15], 0.632558 for Patidar et al. [25], and 0.629806 for Liu et al. [17]. The frequency measure's p _Value analysis is shown in Table 7.

The results of simulations for different H_1 – H_2 – H_3 – M – $Lrange$ are shown in Table 8. The number of minimum and maximum synchronization steps indicates a number of minimum and maximum steps to synchronize the weight of the two networks. The column for the average steps indicates the sum of all simulations conducted. The minimum and maximum synchronization time columns indicate the minimum

and maximum time in seconds required to synchronize the weights of the two networks. The Successful synchronization of the attacker E column shows successfully how many instances that the attacking network imitated the behaviors of the other two networks for total simulations performed. Finally, the percentage of successful synchronization of the attacker E (%) column shows successfully the percentage of total simulations shown by the earlier column.

As shown in Table 8, the best results for the defense against the attacking network are the combinations (8-8-8-16-8) and (8-8-8-8-128), respectively. E did not mimic the behavior of A and B's TLTPM in any of the 700,000 simulations. The results of 1200,000 simulations for the (8-8-8-16-8) and (8-8-8-8-128) configurations as shown in Table 9.

Table 10 shows the comparison of synchronization time for fixed network size and variable learning rules and synaptic depth in the proposed TLTPM and existing CVTPM method. From Table 10, it has been seen that a trend towards an increase in the synchronization steps as the range of weight values ($Lrange$) increases in all three learning rules. For small ($Lrange$) values, Hebbian takes fewer synchronization steps than the other two learning rules in the range of 5–15, but as the ($Lrange$) value increases the Hebbian rule, more steps are taken to synchronize than the other two learning rules. Here, the Anti-Hebbian rules take less time than the other two learning rules in the 20–30 range. Random Walk outperforms 35 and beyond. As a result, an increase in the value of the ($Lrange$) security of the system can be increased.

Conclusion and future scope

The assessment of synchronization is a significant feature of neural synchronization. Time taken to complete synchronization has a major security impact. This paper has proposed an advanced approach for evaluating coordination between two neural networks. For generating different key lengths using the neural synchronization process various combinations of H_1 , H_2 , H_3 , M , and $Lrange$ with variable network size are considered. This paper provides the method for the evolution of synchronization and optimum parameter finding algorithm. The proposed approach has been introduced and evaluated under neural synchronization. The results suggest that the proposed method has less delay in achieving complete synchronization using the approach used in this article. Using the proposed approach, the neural synchronization is much more robust to geometric attacks than the existing methods. Therefore, by using the proposed method, the security of neural synchronization can also be improved. This proposed method has several practical implications. This key exchange technique can be used in wireless communication for the secured exchange of information. To validate

Table 8 Results after 700,000 simulations with different values of $H_1-H_2-H_3-M-Lrange$

$(H_1-H_2-H_3-M-Lrange)$	No. of min sync. Steps	No. of max sync. Steps	Min sync. time (s)	Max Sync. time	Successful sync. of attacker E	% of Successful sync. of attacker E (%)
(1-1-1-512-1)	11	32	0.0718	3.7731	341,576	48.79
(2-2-2-256-1)	10	31	0.0708	2.9865	341,482	48.78
(4-4-4-128-1)	10	39	0.0819	3.5782	342,154	48.87
(8-8-8-64-1)	9	28	0.0768	3.1968	349,247	49.89
(1-1-1-256-2)	10	46	0.0732	3.2375	335,685	47.95
(2-2-2-128-2)	11	32	0.0638	671.7826	330,275	47.18
(4-4-4-64-2)	11	241	0.0745	129.7825	298,654	42.66
(8-8-8-32-2)	12	978	0.0668	35.2320	102,383	14.62
(1-1-1-128-8)	17	92	0.0660	2.2273	107,358	15.33
(2-2-2-64-8)	21	543	0.0698	3.3814	38,759	5.537
(4-4-4-32-8)	23	862	0.0534	6.7885	165	0.023
(8-8-8-16-8)	44	781	0.0382	0.0779	0	0.0
(1-1-1-64-128)	10	132	0.0275	0.7901	116,892	16.69
(2-2-2-32-128)	18	495	0.0541	3.8471	345,256	49.32
(4-4-4-16-128)	26	985	0.0398	4.1265	557	0.079
(8-8-8-8-128)	37	1102	0.0515	5.6384	0	0.0

Table 9 Results after 1200,000 simulations for the (8-8-8-16-8) and (8-8-8-8-128) combinations

$(H_1-H_2-H_3-M-Lrange)$	No. of min sync. steps	No. of max sync. steps	Min sync. time (s)	Max sync. time (s)	Successful sync. of attacker E	% of Successful Sync. of attacker E (%)
(8-8-8-16-8)	45	845	0.0812	2.8256	0	0
(8-8-8-8-128)	34	931	0.0408	1.3814	2	0.0002

Table 10 Comparison of synchronization time for fixed network size and variable learning rules and synaptic depth in proposed TLTPM and existing CVTPM method

Lrange value	Sync. time in Hebbian learning rule (in cycle) in proposed TLTPM	Sync. time in Anti-Hebbian learning rule (in cycle) in proposed TLTPM	Sync. time in random walk learning rule (in cycle) in proposed TLTPM	Sync. time in Hebbian learning rule (in cycle) in existing CVTPM dong and Huang [10]	Sync. time in Anti-Hebbian Learning rule (in Cycle) in existing CVTPM Dong and Huang [10]	Sync. time in Random walk learning rule (in cycle) in existing CVTPM Dong and Huang [10]
5	312,56	385,35	394,03	348,16	389,26	415,63
10	681,34	739,46	696,05	813,64	753,09	713,94
15	997,36	1085,03	1017,31	1769,57	1198,47	1104,81
20	2374,63	1623,39	1650,35	2971,04	1857,24	1696,35
25	3512,09	2393,52	2483,18	4186,59	2671,17	2540,62
30	4998,46	3152,27	3294,10	5982,41	3582,75	3318,49
35	6983,27	4418,79	4128,23	7310,37	4793,62	4518,27
40	8312,03	5718,39	4673,71	8953,50	5981,43	5372,11
45	11,016,29	6684,32	6393,49	11,207,26	8138,39	7363,29
50	14,439,59	8736,26	7786,38	14,952,38	9452,57	8720,93

the experimental findings, several results and analyses are conducted. There are a few delays in achieving absolute synchronization. To extend or improve the current state of knowledge in this field further research is needed to optimize the weight of the neural network for faster synchronization. Any nature-inspired optimization algorithm can be considered for this purpose. The study's limitation is that this paper does not consider any strong PRNG with good statistical property for input vector generations to boost the protection of the neural synchronization method. Two key things require more development in the future. One is further decreased the delay and the other is to start the fresh synchronization instead of carrying out a lengthy one.

Acknowledgements The author expressed deep gratitude for the moral and congenial atmosphere support provided by Ramakrishna Mission Vidyamandira, Belur Math, India.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors. his research received no external fundings.

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

References

1. Abdalrda ZK, AL-Qinani IH, Abbas FN (2019) Subject review: key generation in different cryptography algorithm. *Int J Res Sci Eng Technol* 6(5):230–240. <https://doi.org/10.32628/ijrsrset196550>
2. Alani M (2019) Applications of machine learning in cryptography: a survey. In: *Proceedings of the 3rd International Conference on cryptography, security and privacy*, pp 23–27
3. Alazab M, Huda S, Abawajy J, Islam R, Yearwood J, Venkatraman S, Broadhurst R (2014) A hybrid wrapper-filter approach for malware detection. *J Netw* 9(11):2878–2891
4. Alazab M, Khan S, Krishnan SSR, Pham Q, Reddy MPK, Gadekallu TR (2020) A multidirectional LSTM model for predicting the stability of a smart grid. *IEEE Access* 8:85454–85463. <https://doi.org/10.1109/ACCESS.2020.2991067>
5. Allam AM, Abbas HM, El-Kharashi MW (2013) Authenticated key exchange protocol using neural cryptography with secret boundaries. In: *Proceedings of the 2013 International Joint Conference on neural networks, IJCNN 2013*, pp 1–8
6. Bhattacharya SSSRK, Maddikunta PKR, Kaluri R, Singh S, Gadekallu TR, Alazab M, Tariq U (2020) A novel PCA-firefly based XGBoost classification model for intrusion detection in networks using GPU. *Electronics* 9(2):219. <https://doi.org/10.3390/electronics9020219>
7. Chourasia S, Bharadwaj HC, Das Q, Agarwal K, Lavanya K (2019) Vectorized neural key exchange using tree parity machine. *Computsoft* 8:3140–3145
8. Desai V, Deshmukh V, Rao D (2011) Pseudo random number generator using Elman neural network. In: *Recent Advances in Intelligent Computational Systems (RAICS)* pp 251–254
9. Dolecki M, Kozera R (2015) The impact of the TPM weights distribution on network synchronization time. In: Saeed K, Homenda W (eds) *Computer information systems and industrial management. CISIM 2015. Lecture Notes in Computer Science*, vol 9339. Springer, Cham. https://doi.org/10.1007/978-3-319-24369-6_373
10. Dong T, Huang T (2020) Neural cryptography based on complex-valued neural network. *IEEE Trans Neural Netw Learn Syst* 31(11):4999–5004. <https://doi.org/10.1109/TNNLS.2019.2955165>
11. Gadekallu TR, Rajput DS, Reddy MPK, Lakshmana K, Bhattacharya S, Singh S, Jolfaei A, Alazab M (2020) A novel PCA-whale optimization-based deep neural network model for classification of tomato plant diseases using GPU. *J Real-Time Image Proc.* <https://doi.org/10.1007/s11554-020-00987-8>
12. Hadke PP, Kale SG (2016) 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
13. Kanto A, Smaoui N (2009) Logistic chaotic maps for binary numbers generations. *Chaos Solitons Fractals* 40(5):2557–2568. <https://doi.org/10.1016/j.chaos.2007.10.049>
14. Kanter I, Kinzel W, Kanter E (2002) Secure exchange of information by synchronization of neural networks. *Europhys Lett (EPL)* 57(1):141–147. <https://doi.org/10.1209/epl/i2002-00552-9>
15. Karakaya B, Gülden A, Frasca M (2019) A true random bit generator based on a memristive chaotic circuit: analysis, design and FPGA implementation. *Chaos Solitons Fractals* 119:143–149
16. Klimov A, Mityagin A, Shamir A (2002) Analysis of neural cryptography. In: *Proceedings of the 8th International Conference on the theory and application of cryptology and information security*, pp 288–298
17. Liu L, Miao S, Hu H, Deng Y (2016) Pseudo-random bit generator based on non-stationary logistic maps. *IET Inf Secur* 2(10):87–94
18. Liu P, Zeng Z, Wang J (2019) Global synchronization of coupled fractional-order recurrent neural networks. *IEEE Trans Neural Netw Learn Syst* 30(8):2358–2368
19. Mandal J, Sarkar A (2012) Neural weight session key based encryption for online wireless communication (NWSKE). In: Mandal J (ed) *Research and Higher Education in Computer Science and Information Technology. (RHECSIT- 2012)*, pp 90–95
20. Mehic M, Niemiec H, Siljak M, Voznak (2020) Error reconciliation in quantum key distribution protocols. In: *Proceedings of the International Conference on reversible computation*, pp 222–236
21. Niemiec (2019) Error correction in quantum cryptography based on artificial neural networks. *Quantum Inf Process* 8:174–174
22. Niemiec M, Mehic M, Voznak (2018) Security verification of artificial neural networks used to error correction in quantum cryptography. In: *Proceedings of the 26th Telecommunications Forum (TELFOR)*, pp 1–4
23. NIST (2020) NIST statistical test. http://csrc.nist.gov/groups/ST/toolkit/rng/stats_tests.html. Accessed 28 Nov 2020
24. Pal SK, Mishra S, Mishra S (2019) An TPM based approach for generation of secret key. *Int J Comput Netw Inf Secur* 11(10):45–50
25. Patidar V, Sud KK, Pareek NK (2009) A pseudo random bit generator based on chaotic logistic map and its statistical testing. *Informatica* 33:441–452
26. Protic D (2016) Neural cryptography. *Vojnotehnicki glasnik* 64(2):483–495. <https://doi.org/10.5937/vojtehg64-8877>

27. Ruttur A (2006) Neural synchronization and cryptography. University of St Gallen, St. Gallen
28. Ruttur A, Kinzel W, Naeh R, Kanter I (2006) Genetic attack on neural cryptography. *Phys Rev E*. <https://doi.org/10.1103/physreve.73.036121>
29. Ruttur A, Kinzel W, Kanter I (2007) Dynamics of neural cryptography. *Phys Rev E*. <https://doi.org/10.1103/physreve.75.056104>
30. Salguero Édgar D, Fuertes W, Lascano E (2019) On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key. *Secur Commun Netw* 2019:1–10. <https://doi.org/10.1155/2019/8214681>
31. Santhanalakshmi S, Sudarshan T, Patra GK (2014) 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
32. Santhanalakshmi S, Sangeeta K, Patra GK (2015) Analysis of neural synchronization using genetic approach for secure key generation. *Commun Comput Inf Sci* 536:207–216
33. Sarkar A (2019) Multilayer neural network synchronized secured session key based encryption in wireless communication. *Int J Artif Intell* 8(1):44–53
34. Sarkar A, Mandal J (2012) Secured wireless communication using fuzzy logic based high speed public-key cryptography (FLH-SPKC). *Int J Adv Comput Sci Appl (IJACSA)* 3(10):137–145
35. Sarkar A, Mandal J (2012) Swarm intelligence based faster public-key cryptography in wireless communication (SIFPKC). *Int J Comput Sci Eng Technol (IJCSET)* 3(7):267–273
36. Sarkar A, Khan MZ, Singh MM, Noorwali A, Chakraborty C, Pani SK (2021) Artificial neural synchronization using nature inspired whale optimization. *IEEE Access* 9:16435–16447. <https://doi.org/10.1109/ACCESS.2021.3052884>
37. Shacham LN, Klein E, Mislovaty R, Kanter I, Kinzel W (2004) Cooperating attackers in neural cryptography. *Phys Rev E*. <https://doi.org/10.1103/physreve.69.066137>
38. Shishniashvili E, Mamisashvili L, Mirtskhulava L (2020) Enhancing IoT security using multi-layer feedforward neural network with tree parity machine elements. *Int J Simul Syst Sci Technol* 21(2):371–383. <https://doi.org/10.5013/ijssst.a.21.02.37>
39. Zçakmak BO, ilen AOZ, lu UY, in KC (2019) Neural and quantum cryptography in big data: a review. In: *Proceedings of the 2019 IEEE International Conference on big data*, pp 2413–2417

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