Hindawi
Security and Communication Networks
Volume 2019, Article ID 8214681, 10 pages
https://doi.org/10.1155/2019/8214681

Research Article

On the Development of an Optimal Structure of Tree Parity Machine for the Establishment of a Cryptographic Key

Édgar Salguero Dorokhin, Walter Fuertes, and Edison Lascano

Department of Computer Sciences, Universidad de las Fuerzas Armadas ESPE, Sangolquí, P.O. Box 171-5-231-B, Ecuador

Correspondence should be addressed to Walter Fuertes; wmfuertes@espe.edu.ec

Received 30 December 2018; Revised 20 February 2019; Accepted 26 February 2019; Published 18 March 2019

Academic Editor: Kuo-Hui Yeh

When establishing a cryptographic key between two users, the asymmetric cryptography scheme is generally used to send it through an insecure channel. However, given that the algorithms that use this scheme, such as RSA, have already been compromised, it is imperative to research for new methods of establishing a cryptographic key that provide security when they are sent. To solve this problem, a new branch known as neural cryptography was born, using a modified artificial neural network called Tree Parity Machine or TPM. Its purpose is to establish a private key through an insecure channel. This article proposes the analysis of an optimal structure of a TPM network that allows generating and establishing a private cryptographic key of 512-bit length between two authorized parties. To achieve this, the combinations that make possible to generate a key of that length were determined. In more than 15 million simulations that were executed, we measured synchronization times, the number of steps required, and the number of times in which an attacking TPM network manages to imitate the behaviour of the two networks. The simulations resulted in the optimal combination, minimizing the synchronization time and prioritizing security against the attacking network. Finally, the model was validated by applying a heuristic rule.

1. Introduction

Cryptography is the mathematical science and the discipline of writing messages in encoded text. Its purpose is to protect secrets from adversaries, interceptors, intruders, opponents, and attackers [1]. It also pays special attention on mechanisms that guarantee information integrity and deals with techniques for exchanging users authentication keys and protocols [2]. The most known method is the combination of the symmetric and asymmetric encryption and decryption algorithms. The asymmetric algorithm is used for the exchange of cryptographic keys and the symmetric algorithm is used for information encryption and decryption.

The RSA public key cryptosystem and the systems based on elliptic curves are the most common forms for public-key cryptography in the encryption and digital signature standards [3]. However, the security of the RSA algorithm depends on the length of the prime numbers used for factoring [4]. Thus, one of the main concerns of RSA is the demand for large keys in today's cryptographic algorithms, since the product of two long prime numbers must be factored. This generates a bigger value than the original. Increasing the length of prime numbers increases security; however, the computational cost of factoring these numbers is also increased. For this reason, it is important to look for new methods of exchanging cryptographic keys in a secure way and with a relatively low computational cost. Hence, neural cryptography is born; this method uses a neural network called Tree Parity Machine (TPM). In this way, through two TPM networks with exactly the same structure, two users can establish a cryptographic key by exchanging the inputs and outputs of these networks, keeping the synaptic weights secret.

The aim of our study is to search for an optimal structure that minimizes the time and number of steps that two TPM networks require to establish a cryptographic key of 512-bit length. In addition, we also emphasize on an attacking network whose behaviour is difficult to imitate. For this, we created an algorithm using Python; this algorithm allows performing the simulations to test and find the optimal structure. A total of 15'041,100 simulations were conducted with all the possible structures of the network. Additionally,

we used R and GNU Octave for statistical analysis. Finally, a heuristic rule was applied to validate the computed values (proposed by [5, 6]). As a result, it was possible to determine the optimal structure of the TPM network with an average amount of 211 steps and a percentage of 0.00004% of success of a passive attack network and with a successful 0% of a geometric attack.

The main contribution of this study is that two users will be able to generate a cryptographic key based on certain probability according to the number of steps established at the beginning of the synchronization and with a very low probability of a successful passive attack.

The remainder of this document is structured as follows. Section 2 mentions the related works that supports the present study. Section 3 describes the procedure that helps determine the best combination. Section 4 shows the results obtained from the experiment. Finally, Section 5 details conclusions and proposes future work.

2. Related Works

Concerning the studies related to this research, Kanter et al. [7] and Rosen-Zvi et al. [8] demonstrated that when two artificial neural networks are trained on their outputs according to a learning rule, they are able to develop equivalent states of their internal synaptic weights. Kinzel and Kanter [9] and Ruttor et al. [10] revealed that the possibility of a successful attack decreases as the synaptic depth of the network increases and the computational cost of the attacker increases, since its effort grows exponentially while the effort required by the users grows polynomially. Similarly, Sarkar and Mandal [11] and Sarkar [12] mention that the performance is improved by increasing the synaptic depth of the TPM networks, henceforth counteracting the brute force attacks with the current computation. In comparison with the present study, we used the structure of the TPM network proposed by the latter. However, we determined that the security of the TPM network can also be increased by increasing the number of neurons in the hidden layer and also the number of entries of each neuron. In addition, the range for the synaptic depth value has been modified to adapt the need to generate a key of 512 bits.

With respect to the proposed algorithms for the design of the TPM network, Lei et al. [13] developed a two-layer, prepowered TPM network model. A fast synchronization can be achieved by increasing the minimum value of the internal representations Hamming distance and by reducing the probability of a step that does not modify networks weights. Allam et al. [14] proposed an algorithm that increases the security of neural cryptography by authenticating communications using previously shared secrets; in this way, it increases the security of neural cryptography. As a result, it shows that the algorithm reaches a very high security level without increasing synchronization time. Ruttor [15] mentions that the value of K is 3, since lower values have negative consequences from a safety point of view and higher values have negative consequences in terms of synchronization time. Klimov et al. [16] calculate the probability that two networks take to either synchronize their weights or not. Although the probability rises with a low value of K, the success of the attacking network also increases, and consequently the value of K must be greater. In comparison with our study, the initial structure of the TPM networks is random and different and has a single output. In addition, because the goal is to generate a 512-bit key, it was not arithmetically adequate to use odd numbers for K, N, and L values.

In relation to measurements on TPM networks performance, Dolecki and Kozera [17] present a method of frequency analysis that allows evaluating the synchronization level of two TPM networks before they finish up, with the calculated value not related to the difference of their synaptic weights. As a result, the selection of the appropriate range for the count frequency and the threshold allow them to specify whether it is a short or a long synchronization. Santhanalakshmi et al. [18] and Dolecki and Kozera [19] analyse and compare the performance of synchronization by employing, respectively, a genetic algorithm and a Gaussian distribution instead of uniform random values for the weights of the TPM networks. As a result, they found that replacing the random weights with optimal weights helps reduce the synchronization time. In addition, increasing the number of hidden and input neurons accelerates convergence and also reduces the probability of success of the "Majority Flipping Attack" attack. Dolecki et al. [20] perform an adjustment of the timing distribution of two TPM networks to a Poisson distribution. Pu et al. [21] perform an algorithm that combines "true random sequences" (generated by artificial and validated circuits with randomness tests) with TPM networks, demonstrating more complex dynamic behaviours that offer better performance as an encryption tool and resistance to attacks. In comparison with our study, the synaptic weights values of the TPM networks are generated according to a discrete uniform distribution. Additionally, the Poisson distribution adjustment was made with the results in the number of steps in each of the simulations.

With reference to rules that contribute in the design of TPM networks, Mu and Liao [5] and Mu et al. [6] define the following heuristic rule: "Keeping the equations of motion constant, a high value in the state classifier with respect to the minimum values of the smallest Hamming distances between the state vectors of the networks, has a high probability of fulfilling the condition that the average change in the percentage difference between synaptic weights is greater than zero, improving the security of neuronal cryptography". For our study, we used the proposed heuristic rule to determine the level of security of the final structure of the TPM network.

In regard to modification of initial structures of a TPM network, Gomez et al. [22] state that, with an initial misalignment in the weights between 15% and 20%, the synchronization time is reduced from $1.25ms$ to less than $0.7ms$. This also reduces the number of steps from 220 to less than 100. Within this context, Niemiec [23] presents a new idea for a key reconciliation method in quantum cryptography using TPM networks, correcting errors that occur during transmission in the quantum channel. The

Table 1: Possible values of L.

Decimal value L	Binary length L
2^0	1
2^1	2
2^3	4
2^7	8
2^{15}	16
2^{31}	32
2^{63}	64
2^{127}	128
2^{255}	256
2^{511}	512

number of steps necessary to establish the key is significantly reduced with a low value in the error rate of the quantum bit and a high value in the initial percentage of synchronization of the two networks. In comparison with our study, we did not use initial structures that presented an initial alignment with an established percentage. This initial alignment should be delivered and shared by the two users, which implies that an attacker has more initial information about the structure of the TPM networks.

3. Materials and Methods

3.1. Background. A TPM is a neural network that is formed by a hidden layer and a single output. The general structure consists of the triad of values K, N, and L, where K is the number of neurons in the hidden layer, N is the number of entries of each neuron in the hidden layer, and L sets the limit of the range of possible integer values related to the synaptic weights.

To generate and establish a key of 512 bits, some variations of the TPM structure whose synaptic weights allow this key length were tested. To determine it, the value of L was taken as a base, which, in its binary notation, establishes the final length of the key. To establish pair values in the key length, the limits of the range of values of $[-L, L]$ to $[-L, L-1]$ whose maximum value in decimal notation is $2L - 1$ were modified, and the length of its binary value is a multiple of 512. Thanks to this, there are 10 possible values of L that allow generating keys of 512 bits; see Table 1.

For each value of L, there are values of K and N such that $len(L_{BIN}) * K * N$ =512, considering the restriction $K <= N$, as shown in Table 2.

The restriction of $K <= N$ is due to the fact that if $K > N$, the neural network will be able to reach values that make the synchronization not to be possible. Then, there are 30 possible combinations of K, N, and L values that allow us to generate a key of 512-bit length.

To find the optimal combination that allows rapid synchronization in a minimum number of steps, which prevents an attacker from being able to copy their behaviour, 500,000 simulations were performed for each combination. Each simulation consisted of three TPM networks, two networks synchronized their weights in an authoritative way (Alice and Bob), and another unauthorized attacking network (Eve),

Table 2: Possible combinations with the values of L.

L	K	N
2^0	1	512
	2	256
	4	128
	8	64
	16	32
2^1	1	256
	2	128
	4	64
	8	32
	16	16
2^3	1	128
	2	64
	4	32
	8	16
2^7	1	64
	2	32
	4	16
	8	8
2^{15}	1	32
	2	16
	4	8
2^{31}	1	16
	2	8
	4	4
2^{63}	1	8
	2	4
2^{127}	1	4
	2	2
2^{255}	1	2
2^{511}	1	1

they tried to imitate the behaviour of the other two networks to discover the key they tried to establish. For simulation, the values of L greater than 2^{31} (2^{63}, 2^{127}, 2^{255} y 2^{511}) were not considered, due to its length and complexity in the calculation.

3.2. Implementation of the Algorithm and the Attack Model. We used Python to implement an algorithm to measure the time and the number of steps needed for synchronization, in addition to calculating the cumulative values of the number of steps during simulations (see Algorithm 1).

As can be seen, in Algorithm 1, the second line defines the total of simulations that will be executed. Lines 3 to 5 initialize the three TPM networks with K, N, and L. Line 6 initializes the step counter to 0. Line 7 establishes that the simulation will not finish while the weights of both networks are not equal. In line 8, we create a random vector that will be the input for the networks. Lines 9 to 11 compute the outputs of the three networks. Line 12 analyses whether the outputs of Alice and Bob networks are the same. In this case, lines 13 and 14 update weights of both networks according to the

```
Data: K, N, L, number of simulations
Result: none
1 initialization;
2 for i ⟵ 0 to numberSimulations do
3     Alice ⟵ TPM (K, N, L);
4     Bob ⟵ TPM (K, N, L);
5     Eve ⟵ TPM (K, N, L);
6     steps ⟵ 0;
7     while Alice.weights != Bob.weights do
8         input ⟵ randomVector ();
9         outA ⟵ Alice(input);
10        outB ⟵ Bob(input);
11        outE ⟵ Eve(input);
12        if outA = outB then
13            Alice.update(outB);
14            Bob.update(outA);
15        end
16        if outA = outB = outE then
17            Eve.update(outA);
18        end
19        steps ⟵ steps+1;
20    end
21    saveToFile (steps);
22 end
```

Algorithm 1: Pseudocode of the simulations.

previously computed outputs. Line 16 allows Eve to update her weights as shown by line 17, but only when the outputs of the three networks are equal. Line 19 increases the step counter by 1. Finally, line 21 saves the number of steps in a file.

At a glance from the algorithm, attacks occur when the outputs of the three TPM networks match. Eve can only listen to messages that are sent between Alice and Bob, and its learning process is slower since it only updates her weights if its output matches the other two outputs. According to [24], a geometric attack has better results when an attacker uses a single TPM network to mimic the behaviour of the other two authorized networks. Therefore, additional 180,000 simulations were conducted in a geometric attack. The condition is taken into account when the outputs of the two authorized networks are equal, and the output of the attacking network is different. In this case, the attacking network can modify its internal representation by adjusting the partial output with the lowest absolute value by its counterpart.

For this purpose, we designed and included Algorithm 2. In this algorithm, the third line calculates the absolute values of the partial outputs of each hidden neuron, using the input values and the weights of the attacking network. Line 4 calculates the sign of each of the partial values of each hidden neuron. In line 5, the sign corresponding to the position of the minimum absolute value of the partial output is changed. Line 6 calculates the value of the output of the network with the new partial outputs. Finally, line 7 updates the network weights.

```
Data: outA, outB, outE, input, weightE
Result: none
1 initialization;
2 if outA = outB != outE then
3     sigmaAbs ⟵ abs(sum(input ∗ weightE));
4     sigmaSign ⟵ sign(sum(input ∗ weightE));
5     sigmaSign[sigmaAbs.min()] ⟵
        -sigmaSign[sigmaAbs.min()];
6     tau ⟵ prod(sigmaSign);
7     update(tau);
8 end
```

Algorithm 2: Pseudocode geometric attack.

4. Results and Discussion

4.1. Results of the Simulations. To perform and obtain the results of the simulations, a DELL Inspiron 5759 computer with a 2.50GHz sixth generation Intel Core i7-6500U CPU (model 78) was used. It has four CPUs, one socket, two cores per socket, and two processing threads per core, cache memory L1d of 32K, L1i of 32K, L2 of 256K, and L3 of 4096K.

Table 3 shows the results obtained from the simulations performed for each combination of K, N, and L. Columns Min Steps and Max Steps show the minimum and the maximum number of steps that took for the two networks to synchronize their weights. The Average Steps column shows the average of steps of all the executed simulations. Columns Min Time and Max Time show the minimum and maximum time (in seconds) that took for the two networks to synchronize their weights. The Average Time column shows the average time (in seconds) of all the executed simulations. Column Eve sync successfully shows the number of occurrences in which the attacking network managed to imitate the behaviour of the other two networks of the total of executed simulations. Finally, column % Eve sync successfully shows the percentage represented by the previous column with respect to the total of simulations.

As can be seen in Table 3, items 14 and 18 corresponding to the combinations $(8, 16, 2^3)$ and $(8, 8, 2^7)$, respectively, got the best results from the point of view of security against the attacking network (Eve). None of the 500,000 simulations allowed Eve to imitate the behaviour of the other two networks with the same number of steps of Alice and Bob's networks. However, two combinations took a greater number of steps to achieve synchronization (211.659 and 218.2211, respectively) than the other combinations, which involves a longer average time.

For synchronization time, item 22 corresponding to $(1, 16, 2^{31})$ combination produced the best result, with an average synchronization time of 0.0549 seconds. However, it got an attacking network sync percentage of 9.2488%. Given that security was prioritized in the present study, we considered the combinations mentioned above as good results. It should also be noted that the combinations, where $L = 2^0 = 1$, despite having the best times in synchronization, produced the worst results because the attacking network Eve

Table 3: Results obtained from 500,000 simulations for each combination.

Item	Combination (K, N, L)	Minimum steps	Maximum steps	Average steps	Minimum time (sec)	Maximum time (sec)	Average time (sec)	Eve's successful sync	% Eve's successful sync
1	$(1, 512, 2^0)$	5	28	9.3367	0.0657	2.2718	0.4259	368680	73.736
2	$(2, 256, 2^0)$	5	25	9.3312	0.0596	1.5588	0.4253	368229	73.6458
3	$(4, 128, 2^0)$	5	32	9.3374	0.0680	2.1275	0.4263	368471	73.6942
4	$(8, 64, 2^0)$	4	27	9.3370	0.0656	1.9089	0.4233	368652	73.7304
5	$(16, 32, 2^0)$	5	32	9.3356	0.0661	1.4625	0.4249	368389	73.6778
6	$(1, 256, 2^1)$	8	45	15.5450	0.0573	1.4665	0.3824	355397	71.0794
7	$(2, 128, 2^1)$	8	25371	15.7774	0.0523	537.5299	0.3857	349495	69.899
8	$(4, 64, 2^1)$	8	4104	18.0226	0.0491	95.8512	0.3965	308691	61.7382
9	$(8, 32, 2^1)$	9	814	31.6046	0.0403	19.6915	0.4597	161148	32.2296
10	$(16, 16, 2^1)$	15	587	73.4867	0.0289	7.3628	0.5381	8102	1.6204
11	$(1, 128, 2^3)$	11	78	25.3284	0.0472	1.1114	0.2983	132715	26.543
12	$(2, 64, 2^3)$	15	289	65.2706	0.0411	2.7924	0.4699	41475	8.295
13	$(4, 32, 2^3)$	26	616	128.1582	0.0309	4.5353	0.5325	487	0.0974
14	**$(8, 16, 2^3)$**	**38**	**769**	**211.6590**	**0.0262**	**6.0476**	**0.5741**	**0**	**0.0**
15	$(1, 64, 2^7)$	10	71	25.0318	0.0180	0.6389	0.1679	132666	26.5332
16	$(2, 32, 2^7)$	15	372	74.9943	0.0362	1.9875	0.3599	39111	7.8222
17	$(4, 16, 2^7)$	23	712	137.9569	0.0193	2.7179	0.4572	759	0.1518
18	**$(8, 8, 2^7)$**	**30**	**955**	**218.2211**	**0.0342**	**3.4490**	**0.5109**	**0**	**0.0**
19	$(1, 32, 2^{15})$	7	68	21.3921	0.0105	0.2913	0.0847	144461	28.8922
20	$(2, 16, 2^{15})$	10	451	69.8188	0.0196	1.2301	0.2161	61628	12.3256
21	$(4, 8, 2^{15})$	14	777	132.0805	0.0181	2.7993	0.3200	4080	0.816
22	$(1, 16, 2^{31})$	6	84	24.9428	0.0020	0.2366	0.0549	46244	9.2488
23	$(2, 8, 2^{31})$	10	804	100.5999	0.0121	1.4193	0.1842	25006	5.0012
24	$(4, 4, 2^{31})$	9	837	134.2786	0.0032	2.0644	0.2281	9168	1.8336

Table 4: Results obtained from 1'000,000 simulations for the two combinations.

Item	Combination (K, N, L)	Minimum steps	Maximum steps	Average steps	Minimum time (sec)	Maximum time (sec)	Average time (sec)	Eve's successful sync	% Eve's successful sync
1	$(8, 16, 2^3)$	40	785	211.5888	0.0780	1.5155	0.4058	0	0
2	$(8, 8, 2^7)$	27	861	218.3696	0.0312	0.9229	0.2385	2	0.0002

managed to imitate the behaviour of the other two networks in approximately 73.7% of the total simulations.

To determine the best combination of the two previous situations, specifically, with better results from the security point of view, we performed one million simulations for each, and we got the following results; see Table 4.

From Table 4, we can determine the best combination for the triad (K, N, L) is $(8, 16, 2^3)$. From Table 4, it was determined that the best combination for the trio of values (K, N, L) is $(8, 16, 2^3)$. Simulations were carried out using geometric attack and the results were obtained as shown in Table 5.

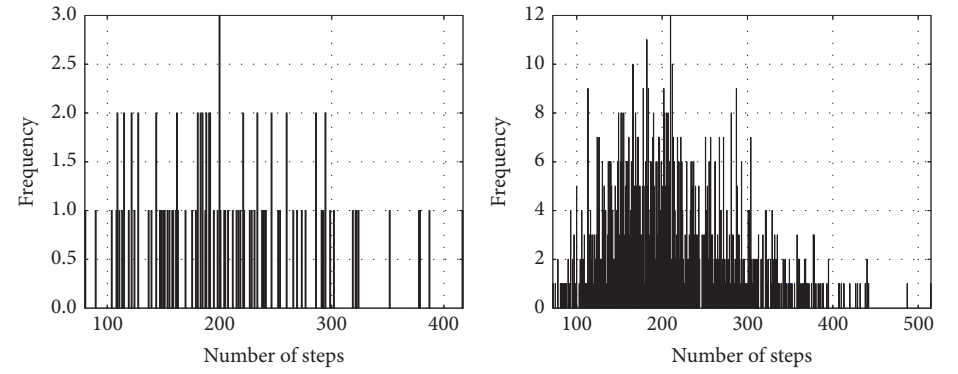
More simulations with the combination of $(8, 16, 2^3)$ were conducted to determine the evolution of the distribution in the number of steps. Of all simulations, we totalled the number of steps that were necessary for the two TPM networks to synchronize. See the histograms shown in Figure 1.

Table 5: Success probabilities of an attacking TPM using geometrical attack.

Combination (K, N, L)	Number of simulations	Eve's successful sync	% Eve's successful sync
$(8, 16, 2^3)$	180,000	0	0.0

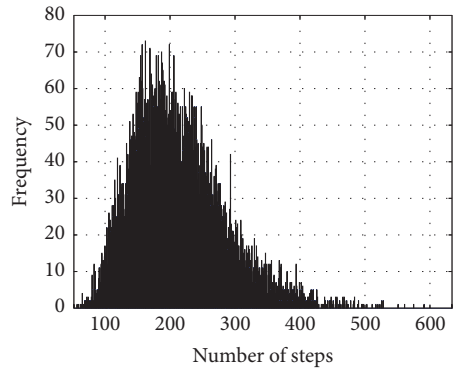
During additional simulations, on a single occasion, Eve's attacking network managed to synchronize its weights with the other networks. Table 6 shows the probabilities of a successful attack of both combinations, compared to the total of performed simulations.

4.2. Adjustment of Probabilities Distribution and Calculation. As can be seen in Figure 1, the number of steps used in all simulations follows a positive asymmetric discrete

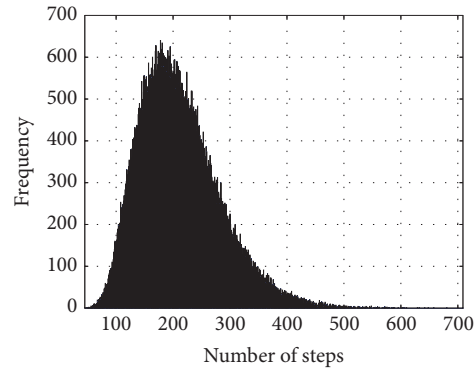


(a) 100 simulations' histogram

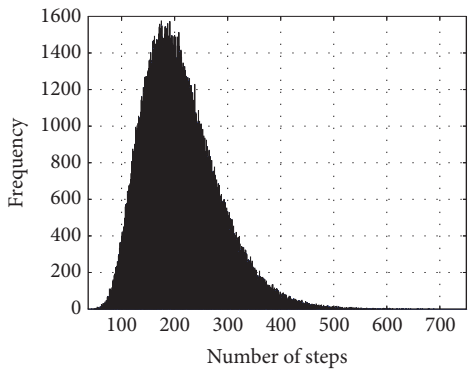
(b) 1,000 simulations' histogram



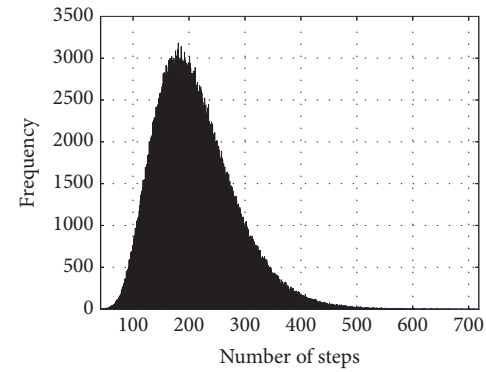
(c) 10,000 simulations' histogram



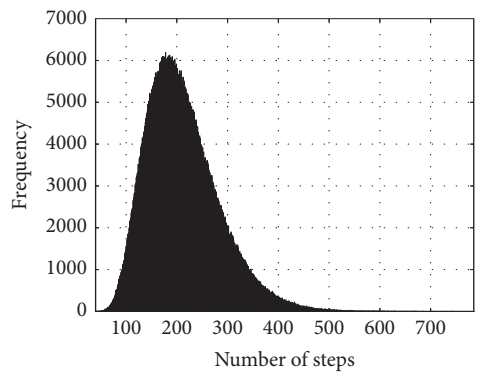
(d) 100,000 simulations' histogram



(e) 250,000 simulations' histogram



(f) 500,000 simulations' histogram



(g) 1'000,000 simulations' histogram

Figure 1: Illustrations derived from the evolution of the distribution in the number of steps.

Table 6: Success probabilities of an attacking TPM of the two combinations.

Combination (K, N, L)	Number of simulations	Eve's successful sync	% Eve's successful sync
$(8, 16, 2^3)$	2'361,100	1	0.00004
$(8, 8, 2^7)$	1'500,000	2	0.0001

Table 7: Poisson adjust results.

Number of simulations	Max. limit	Delta	Probability	Function
1,000	515	0.5150	0.4787	f
10,000	634	0.0634	0.8147	g
100,000	709	0.00709	0.9625	h
250,000	750	0.0030	0.9814	i
500,000	718	0.0014	0.9902	j
1'000,000	785	0.0007	0.9946	k

distribution or to the right. For the present work, the Poisson distribution was considered by the characteristics of the mathematical parameters, such as lambda and delta, where delta was considered the maximum value of the limit as a function of the total simulations. Delta is the probability of occurrences of the event. Poisson has the formula as shown by the equation ref Poisson.

$$f(\lambda, d) = \frac{e^{-\lambda}\lambda^{d}}{d!} \tag{1}$$

Then, for the first test, we used the results obtained from the 1,000 simulations. The maximum value of the limit is 515, where

$$delta = \frac{515}{1,000} = 0.515; 51.5\% \tag{2}$$

Using (1), taking as delta value = 0.515 of the equation ref delta 1 (lambda equal to delta to obtain the highest probability), a probability of 0.4787 was obtained. Similarly, the probabilities were calculated for the rest of the simulations. For the 10 thousand simulations, a delta value = 0.0634 (6.34 %) was obtained and its probability is 0.8147. Therefore, lambda varies by 34 %. In an analogous way, the same test was carried out with the results of the other simulations. Table 7 shows the obtained values.

As the number of simulations increases, the probability has a tendency towards 1 (maximum probability value), which implies that the tests performed are correct. The same can be seen in Figure 2. As can be seen, the data tend to adjust mathematically to a Poisson distribution.

4.3. Generation of the Cryptographic Key of 512 Bits. When the networks finish their synchronization, their synaptic weights will have values in the range $[-8, 7]$. To obtain the key of 512 bits, the value of L is added to each synaptic weight; this produces that the range is $[0, 15]$. Then, each synaptic weight is transformed to its binary equivalent of 4 bits in length. Each of these values is concatenated and the key of 512-bit length is obtained $(8 * 16 * 4 = 512)$.

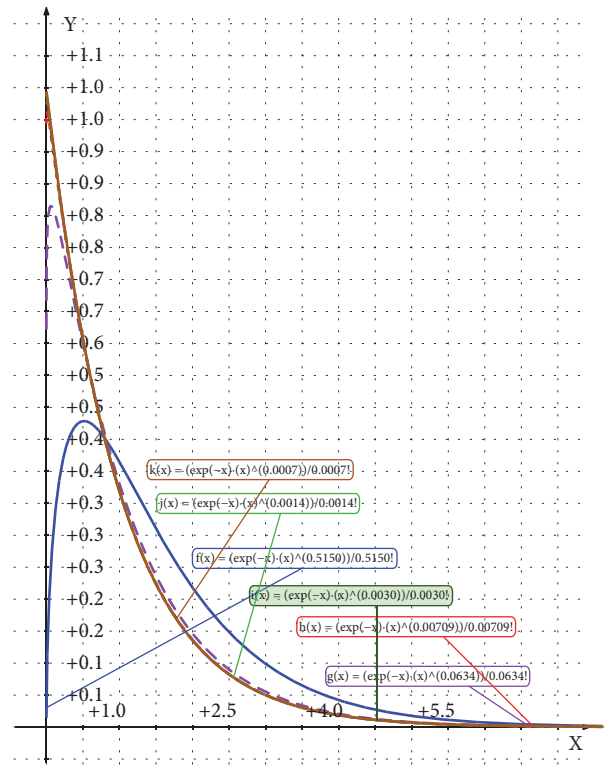


Figure 2: Poisson distribution for all simulations.

4.4. Process of Validation. To validate the security of the proposed TPM network structure with the values of $K = 8$, $N = 16$, and $L = 2^3 = 8$, and, according to the heuristic rule proposed by Mu and Liao [5] and Mu et al. [6], the value of the state classifier h was computed. For this, the Hamming distances were calculated between all the possible vectors whose outputs are equal (1 and −1). Since $K = 8$, there are 2^8 possible vectors (2^7 vectors with output 1 and 2^7 vectors with output −1). The combinations for each group according to their output are given by the binomial coefficient, as shown by

$$\binom{2^7}{2} = \binom{128}{2} = \frac{128!}{2!\,(128-2)!} = 8128 \tag{3}$$

Therefore, there are 8128 combinations for each group. The value of the state classifier was computed as follows. First, Hamming distances were calculated between each pair of vectors whose output is equal to

$$HD_1^{c1} = HD((1,1,1,1,1,1,1,1)(1,1,1,1,1,1,-1,-1)) = 2$$

$$HD_2^{c1} = HD((1,1,1,1,1,1,1,1)(1,1,1,1,1,-1,1,-1)) = 2$$

$$\cdots$$

Data: K, N, L, *probabilities*
Result: Graph
1 initialization;
2 `scatter3` (K(:), N(:), L(:), [16], *probabilities*(:), '.');
3 set(gca,'zscale','log');
4 xlabel "K";
5 ylabel "N";
6 zlabel "L";
7 colorbar;

Algorithm 3: Code in GNU Octave to graph a scatter plot.

$$\begin{aligned} & HD_1^{c2} \\ & \quad = HD\left((1,1,1,1,1,1,1,-1)\,(1,1,1,1,1,1,-1,1)\right) \\ & \quad = 2 \\ & HD_2^{c2} \\ & \quad = HD\left((1,1,1,1,1,1,1,-1)\,(1,1,1,1,1,-1,1,1)\right) \\ & \quad = 2 \\ & \ldots \end{aligned} \tag{4}$$

Then, for each group, the minimum value of the said distances is calculated:

$$\begin{aligned} SHD^{c1} &= \mathtt{min}\left\{HD_1^{c1}, HD_2^{c1}, \ldots\right\} = 2 \\ SHD^{c2} &= \mathtt{min}\left\{HD_1^{c2}, HD_2^{c2}, \ldots\right\} = 2 \end{aligned} \tag{5}$$

Finally, the value of the state classifier is calculated as the smallest value between the minimum values of the Hamming distances:

$$h^{TPM(K=8)} = \mathtt{min}\left\{SHD^{c1}, SHD^{c2}\right\} = 2 \tag{6}$$

The value of $h = 2$ shows that it meets the probability that the average change in the percentage difference between synaptic weights is greater than zero. This implies that the synchronization time is increased polynomially by increasing the value of L. This means that the proposed structure is secure.

4.5. Process of Analysis of the Values of K, N, and L. The probability of success of an attacking network has a very noticeable dependence with respect to the values of K, N, and L, as shown in Table 3. Therefore, an analysis of the values of K, N, and L was conducted with respect to their probability. All possible combinations were plotted with their respective probabilities. To do this, we used the GNU Octave `scatter3` function to draw a scatter diagram. Algorithm 3 presents the code that generated this graph.

Algorithm 3 starts with the values of K, N, L and the probabilities of Eve in each combination. Line 2 draws a 3D scatter diagram with the values of K, N, and L. The value of 16 indicates the size of the points to be plotted. The value of *probabilities*(:) reveals the value of the probabilities that will be every point colour. Line 3 changes the scale of the z axis (value of L) to a logarithmic scale to improve visualization. Lines 4 to 6 change the labels of the axes to their respective values. Finally, line 7 discloses a colour bar. Figure 3 displays the result of Algorithm 3.

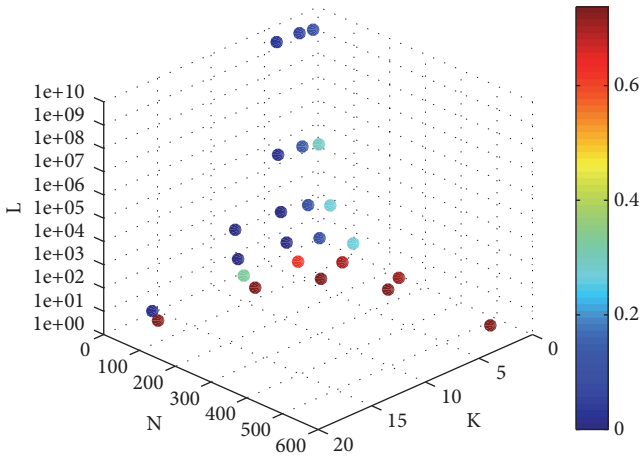


Figure 3: Scatter plot of the values of K, N, and L.

Figure 3 shows how the probabilities change according to the values of K, N, and L. On the x axis are the values of K, on the y axis the values of N, and on the z axis the values of L. The colour of each point displays the probability of success of the attacking network. A low probability is better (blue) and a high probability is worse (red).

As can be seen in Figure 3, the value of K does influence much on probability. On the other hand, a high value of N has a negative impact on the result. Finally, a very low value of L has a negative impact. For this reason, to propose values of K, N, and L, a relatively low value of N is recommended, but it has to be higher than K. The value of L should not be too low. These conditions ensure a very low probability of a successful passive attack.

5. Conclusions

This article has been developed in the context of the design of a TPM neural network that allows generating a 512-bit key between two authorized parties. To do this, the optimal values of K, N, and L were computed through simulations that allowed prioritizing security against an attacking network. This attacking network, through a passive attack, tried to imitate the behaviour of the other two networks, achieving the synchronization with the other two networks on a single occasion. This demonstrates that a secret key can be exchanged between two authorized parties in a network with a probability of 0.00004% so that an attacking network can mimic the original network behaviour. In addition, a geometric attack was designed and executed, with successful results of 0%. Furthermore, if the values of K, N, and L of a TPM neural network are modified, cryptographic keys of various lengths can be obtained. These values also determine

how easy or how difficult it might be for an attacking network to mimic the behaviour of the other two networks. According to our results, it is recommended that the values of K, N, and L meet the following conditions: $K <= N$ and $L > 1$.

As future work, we plan the analysis of L values that were not considered in this study due to software and hardware limitations (2^{63}, 2^{127}, 2^{255}, and 2^{511}), in which security is also prioritized in front of an attacking network. Also we plan studying the combination of neural cryptography using the values calculated in the study, along with symmetric cryptographic systems that need a cryptographic key established between two parties (such as DNA-based cryptography). This in order to perform secure communications and without the need to have sent its keys through insecure means. Finally, it is recommended to perform a more exhaustive security analysis, to the values found for the structure of the TPM neural network, namely, specific attack models designed to manage the cryptographic key, especially in its distribution.

Data Availability

The data used to support the findings of this study (CSV, TXT, and PNG) have been deposited in the Google Drive repository. In the following link, you can find the data used in our manuscript: https://drive.google.com/drive/folders/1MOf7CfXFrFB-gtK6ue4JkHURzga1o8zA.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

The funding of this scientific research is provided by the Ecuadorian Corporation for the Development of Research and the Academy (RED CEDIA) and also from the Mobility Regulation of the Universidad de las Fuerzas Armadas ESPE, from Sangolquí, Ecuador. The authors would like to thank the financial support of the Ecuadorian Corporation for the Development of Research and the Academy (RED CEDIA) in the development of this study, within the Project Grant GT-Cybersecurity.

References

[1] F. L. Bauer, "Cryptology," in *Encyclopedia of Cryptography and Security*, pp. 283-284, Springer, 2011.

[2] Y. Lindell and J. Katz, "Introduction to Modern Cryptography," *Chapman and Hall/CRC*, 2014.

[3] X. Zhou and X. Tang, "Research and implementation of RSA algorithm for encryption and decryption," in *Proceedings of the 6th International Forum on Strategic Technology, IFOST 2011*, pp. 1118–1121, IEEE, China, August 2011.

[4] F. Meneses, W. Fuertes, J. Sancho et al., "RSA Encryption Algorithm Optimization to Improve Performance and Security Level of Network Messages," *IJCSNS*, vol. 16, no. 8, p. 55, 2016.

[5] N. Mu and X. Liao, "An approach for designing neural cryptography," in *International Symposium on Neural Networks*, pp. 99–108, Springer, Heidelberg, Berlin, Germany, 2013.

[6] N. Mu, X. Liao, and T. Huang, "Approach to design neural cryptography: A generalized architecture and a heuristic rule," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 87, no. 6, Article ID 062804, 2013.

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

[8] M. Rosen-Zvi, I. Kanter, and W. Kinzel, "Cryptography based on neural networks—analytical results," *Journal of Physics A: Mathematical and General*, vol. 35, no. 47, pp. L707–L713, 2002.

[9] W. Kinzel and I. Kanter, "Interacting neural networks and cryptography," in *Advances in Solid State Physics*, pp. 383–391, Springer, 2002.

[10] A. Ruttor, W. Kinzel, R. Naeh, and I. Kanter, "Genetic attack on neural cryptography," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 73, no. 3, Article ID 036121, 2006.

[11] A. Sarkar and J. K. Mandal, "Cryptanalysis of key exchange method in wireless communication," *Compare*, vol. 1, 2015.

[12] A. Sarkar, "Genetic Key Guided Neural Deep Learning based Encryption for Online Wireless Communication (GKNDLE)," *International Journal of Applied Engineering Research*, vol. 13, no. 6, pp. 3631–3637, 2018.

[13] X. Lei, X. Liao, F. Chen, and T. Huang, "Two-layer tree-connected feed-forward neural network model for neural cryptography," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 87, no. 3, Article ID 032811, 2013.

[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, IJCNN 2013*, pp. 1–8, USA, August 2013.

[15] A. Ruttor, "Neural synchronization and cryptography," *Disordered Systems and Neural Networks*, 2007.

[16] A. Klimov, A. Mityagin, and A. Shamir, "Analysis of neural cryptography," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pp. 288–298, Springer, 2002.

[17] M. Dolecki and R. Kozera, "Threshold method of detecting long-time TPM synchronization," in *Computer Information Systems and Industrial Management*, vol. 8104 of *Lecture Notes in Computer Science*, pp. 241–252, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[18] S. Santhanalakshmi, K. Sangeeta, and G. K. Patra, "Analysis of neural synchronization using genetic approach for secure key generation," *Communications in Computer and Information Science*, vol. 536, pp. 207–216, 2015.

[19] M. Dolecki and R. Kozera, "The Impact of the TPM Weights Distribution on Network Synchronization Time," in *Computer Information Systems and Industrial Management*, vol. 9339 of *Lecture Notes in Computer Science*, pp. 451–460, Springer International Publishing, Cham, Swizerland, 2015.

[20] M. Dolecki and R. Kozera, "Distribution of the tree parity machine synchronization time," *Advances in Science and Technology – Research Journal*, vol. 7, no. 18, pp. 20–27, 2013.

[21] X. Pu, X.-J. Tian, J. Zhang, C.-Y. Liu, and J. Yin, "Chaotic multimedia stream cipher scheme based on true random sequence combined with tree parity machine," *Multimedia Tools and Applications*, vol. 76, no. 19, pp. 19881–19895, 2017.

[22] H. Gomez, Ó. Reyes, and E. Roa, "A 65 nm CMOS key establishment core based on tree parity machines," *Integration, the VLSI Journal*, vol. 58, pp. 430–437, 2017.

[23] M. Niemiec, "Error correction in quantum cryptography based on artificial neural networks," *Cryptography and Security*, 2018.

[24] A. Ruttor, W. Kinzel, and I. Kanter, "Dynamics of neural cryptography," *Physical Review E: Statistical, Nonlinear, and Soft Matter Physics*, vol. 75, no. 5, 056104, 9 pages, 2007.