The Impact of the TPM Weights Distribution on Network Synchronization Time

Michal Dolecki, Ryszard Kozera

The impact of the TPM weights distribution on network synchronization time

Michał Dolecki[1] and Ryszard Kozera[1,2]

[1] The John Paul II Catholic University of Lublin
ul. Konstantynów 1 H; 20-708 Lublin, Poland
michal.dolecki@kul.pl
[2] Warsaw University of Life Sciences – SGGW
ul. Nowoursynowska 159, 02-776 Warsaw, Poland
ryszard.kozera@gmail.com, ryszard_kozera@sggw.pl

Abstract. Two neural networks with randomly chosen initial weights may achieve the same weight vectors in the process of their mutual learning. This phenomenon is called a network synchronization, and can be used in cryptography to establish the keys for further communication. The time required to achieve consistent weights of networks depends on the initial similarity and on the size of the network. In the previous work related to this topic the weights in TPM networks are randomly chosen and no detailed research on used distribution is performed. This paper compares the synchronization time obtained for the network weights randomly chosen from either a uniform distribution or from Gaussian distribution with different values of standard deviation. The synchronization time of the network is examined here as a function of different numbers of inputs and of various weights belonging to the intervals with varying sizes. The standard deviation of Gaussian distribution is selected depending on this interval size in order to compare networks with different weights intervals, which also constitutes a new approach for selecting the distribution's parameters. The results of all analyzed networks are shown as a percentage of the synchronization time of a network with weights drawn from uniform distribution. The weights drawn from the Gaussian distribution with decreasing standard deviation have shorter synchronization time especially for a relatively small network.

Keywords: neural networks, Tree Parity Machine, key exchange protocol.

1 Introduction

Cryptographic algorithms permit the data encryption and decryption with the use of the cryptographic keys [1]. These cryptographic keys are special data forming additional input to the encryption and decryption functions. Depending on the keys applied, the cryptography can be divided into symmetric or asymmetric one [2, 3]. In the first approach, the sender and receiver use the same secret key to encrypt and decrypt data. In the asymmetric cryptography the pair of keys is used, where one of them is

kept secret whereas the other is available publicly. In order to use a secret key, a secure communication channel is needed facilitating a possible transfer of the secure key to the both communicating partners. Consequently, this leads to the well-known paradox which in essence amounts to the necessity of creating a secure communication channel based on the assumption of the existence of secure channel. Such difficulty is solved by the Diffie – Hellman [2, 4], where a key exchange algorithm in an open communication channel is proposed. Alternatively, another recently investigated interesting approach is to generate a relatively secure [5, 6] cryptographic key exchange protocol, which is based on special artificial neural network tool [7-11].

Artificial intelligence methods are used in cryptography in various ways. For example genetic algorithms can be used in crypto-analysis or protocol design [12]. Similarly to the artificial neural networks constituting an important element of artificial intelligence, the neuro-cryptography forms an interesting fragment of the cryptography. The term itself was first time used by Dourlens [13] in 1995. The neural networks are exploited to the realization of the cryptographic operations such as the S-BOX operation [14]. In addition they can be used in cryptographic context as a supporting tool e.g. in cryptographic keys exchange procedure [7-9]. The key exchange protocol over an open communication channel using artificial networks is particularly interesting as it is based on a new phenomenon of the so-called TPM network synchronization. Two special artificial neural networks during the mutual learning are able to establish consistent values of weights, which can in turn be used as cryptographic keys [15].

The key exchange protocol resorts to the Tree Parity Machine (TPM) network [7, 8, 16]. TPM is a specific artificial multilayer, feed-forward neural network. Its input topology is characterized by disjointed inputs. In conventional neural networks, each input reaches each neuron of the first hidden network's layer [17-19]. In this case, each neuron has its own impulses received from the fixed collection of input neurons. Values of all the first layer neurons inputs can be seen together as the one input vector for the entire network. This modification is introduced due to the cryptographic usage of the TPM network. If every impulse had an impact on the result of each hidden layer neuron, there would be a potential danger of reasoning about the results of the inner layer based on input pulses. Such separation gives the network distinctive tree-like look. Figure 1 shows the topological structure of the typical TPM network. Each input $x_{ij} \in \{-1, 1\}$ and weight is an integer number $w_{ij} = -L, -L+1, \dots, 0, \dots, L-1, L$.

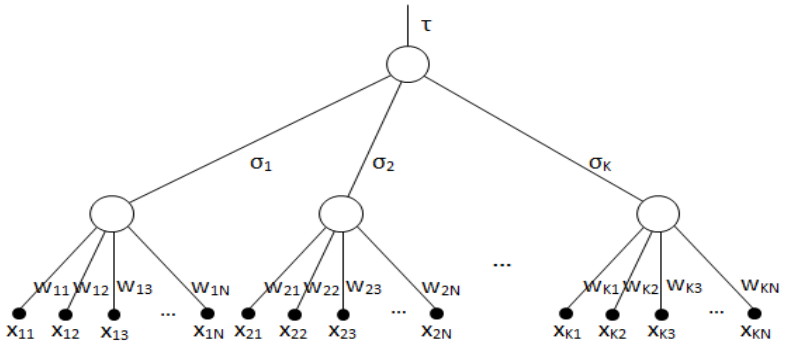

Fig. 1. Structure of Tree Parity Machine network.

The first layer networks are typical neurons which input pulses are multiplied by the corresponding weights and determine the value of its activation function [20]. According to the following formula:

$$\sigma_i = f\left(\sum_{j=1}^{N} x_{ij} w_{ij}\right),$$

where $1 \leq i \leq K$. Activation function f is taken as a signum function given by the formula:

$$f(x) = \begin{cases} -1, for\ x \leq 0 \\ 1, for\ x > 0 \end{cases}.$$

Next, the results of the neurons of the first hidden layer are passed as inputs to the last layer neuron, which calculates the product of all incoming impulses:

$$\tau = \prod_{i=1}^{K} \sigma_i,$$

since $\sigma_i \in \{-1, 1\}$, a result of the network $\tau \in \{-1, 1\}$.

TPM networks are trained according to one of three methods: Anti-Hebbian, Hebbian or Random Walk rule [5, 7, 8, 11]. The synchronization process of the TPM network with discrete weights involves $(2L + 1) \times (2L + 1)$ parameters, which variations in turn forms the Markov chain. The pertinent theoretical analysis can be found in [9].

In key exchange protocol TPM networks undergo a process of mutual learning, in which both play simultaneously the role of a teacher and a student. This learning process decomposes into the following consecutive steps:

1. Trusted parties A and B shall determine e.g. through an open channel, the parameters K, N and L describing the TPM topology and interval which contains a network weight. Both parties establish also a common method for TPM learning.
2. Each parity creates its own TPM network (according to the previously agreed topology) with randomly chosen initial weights w^A and w^B, kept in secret.
3. Both A and B obtain the same, publicly known input vector $x \in X^{KN}$ and compute TPM's results τ^A and τ^B, respectively.
4. Next two involved parities exchange their networks results.

5. A treats received value τ^B as an expected result for its TPM and analogously B uses τ^A as an expected result for its own network.
6. Parities modify their TPM's weights according to chosen learning method.
7. Both parties compare the results of both networks τ^A and $\tau^B$to determine the number of consecutively occurring consistent results. If this value is greater than the pre-determined threshold, the algorithm ends and the network are considered as synchronized, otherwise it is necessary to continue such learning/teaching process and return to step 3.

The synchronization status achieved by both networks is a state, in which they have compatible weight vectors (which may change on each iteration but remain equal in a given pair). Such networks remain synchronized regardless of the time of their further learning. Their weight's values can be used directly as cryptographic keys or as a starting point for pseudo-random number generator that will be used as the keys.

In the first case, the generated key is a sequence of $K \cdot N$ weights within the range of -L to L, which involves $K \cdot N \cdot \log_2(2L + 2)$ bits. For example, synchronization of the networks of 3 neurons with 100 entrances to each of them and weights ranging from -15 to 15 (represented by 5 bits) enables to generate 1500 bit key. Similarly, in general the keys with the required length can be achieved by modifying the networks' parameters. It is important to note that once the networks are synchronized they change their weights (equal in each pair) at each subsequent learning step. Consequently, the latter immediately permits to generate next keys with a priori specified length.

The main problem for the potential attacker is the lack of knowledge of the inner layer's outputs so that he/she has no idea how to modify the weights of the attacking TPM. During the learning process, the respective weights are changed only when the networks produce either the same results (Hebbian and Random Walk Rule) or the opposite results (Anti-Hebbian). In addition, only neurons that have the same result as the entire network modify their weights. For each output of the network (-1 or 1) it is possible 2^{K-1} variants of inner layer network, which attacker should consider in each learning step when weights are modified.

At the beginning of the synchronization procedure both TPM networks have randomly chosen weights. The authors of this key exchange protocol have identified various types of learning steps for each pair of the corresponding neurons in both networks. More specifically, three steps are distinguished: an attracting, a repulsive and quiet steps. The respective names of each step emphasize its impact on the weights of both networks. In attracting step both TPMs have the same output and the weights of both neurons move along the same direction. If one would exceed the limit value -L or L, it stays on the border, and the other moves to this value decreasing distance. In the repulsive step the weight of one network is not changed whereas the other can increase the distance between both weights. There are also cases in which weight reduce distance in repulsive step [21]. The last category is a silent step, when neurons results are different from the results for the entire network so the weight of neurons does not change. The synchronization time is dependent on the frequency of

occurrence of the attractive and repulsive steps. Usually, at the beginning of synchronization process, when both weights are drawn and their compliance is low, more frequently the repulsive steps appear. In contrast at the end of synchronization they occur sporadically. Therefore, the closer to each other the weight vectors are, the more frequently synchronization step occur permitting both weights getting closer.

2 Results

In this paper the relationship between the random distribution from which the initial weights are drawn and the network synchronization time is analyzed. Short time synchronizations are more secure and occur more often than long ones [22-24]. The research performed here is focused on two commonly used distributions: namely a uniform one and a normal Gaussian distribution. Due to the symmetry of the weight's interval [-L, L], a Gaussian distribution is taken with the mean zero, whereas different values of standard deviation are tested instead. Since the simulations in questions are conducted on networks with different sizes (i.e. different topologies dependent on parameter N and weights interval size determined by the parameter L) a standard deviation depends here on the maximum/minimum weight values of TPM examined. This in turn as indicated above is determined by the parameter L. Here various distributions with selected standard deviation $s \in \left\{L, \frac{L}{2}, \frac{L}{5}\right\}$ are admitted and tested. The latter enables to compare different networks of various sizes. The networks tested here use different topologies parameterized here by $K = 3$ and $N \in \{11,50,100\}$. As outlined above, the parameter L is related to a standard deviation s taken for the Gaussian distribution as $s = L$. The value of L is assumed to belong to the following set $L \in \{2,3,4,\ldots,10\}$. In addition, for the normal distribution with $s = \frac{L}{2}$ only even values of L are tested and for a standard deviation $s = \frac{L}{5}$ only values for L which are divisible by 5 are considered i.e. $L \in \{5,10\}$. This gives 75 possible variants of parameters of the examined networks. Each network is tested with random initial weights draw and synchronization 5000 times. The latter renders a total number of 375 000 investigated cases. The results presented below are obtained upon using own simulation program, which enables to draw the initial weights for the TPM network and its synchronization.

The distribution, from which the initial weight values of both learned TPMs are drawn affects their weights compatibility [25]. In addition the compliance at the beginning of synchronization affects the duration of the learning process. An increase of the number of each first layer neuron's input pulses determined by the parameter N together with widening the interval [-L,...,L] containing the respective weights, extends the average synchronization time in accordance with the exponential growth. The analyzed time is expressed in the number of the networks' learning cycles as described above in seven step procedure. This relationship is evident for all analyzed distributions of initial weights. Table 1 presents detailed results for the uniform distribution for TPMs with different values of parameter N and L. The graphs enclosed in

the figure 2 represent the results for the uniform distribution and Gaussian distribution with standard deviation set to either s = 0 and s = L / 2.

Table 1. An average synchronization time in learning cycles for uniform distribution.

L	UNIFORM		
	N=11	N=50	N=100
2	106,11	143,50	154,10
3	235,95	320,48	345,52
4	421,31	585,87	629,55
5	668,64	946,04	1013,96
6	972,76	1393,71	1512,05
7	1342,04	1926,97	2100,38
8	1775,86	2611,55	2839,96
9	2296,15	3354,97	3654,78
10	2858,83	4251,63	4608,29

Noticeably, small differences in the obtained results visibly follow for different distributions. However, for all analyzed cases, a characteristic exponential growth in time required for reaching a full synchronization of the networks is conspicuously transparent.

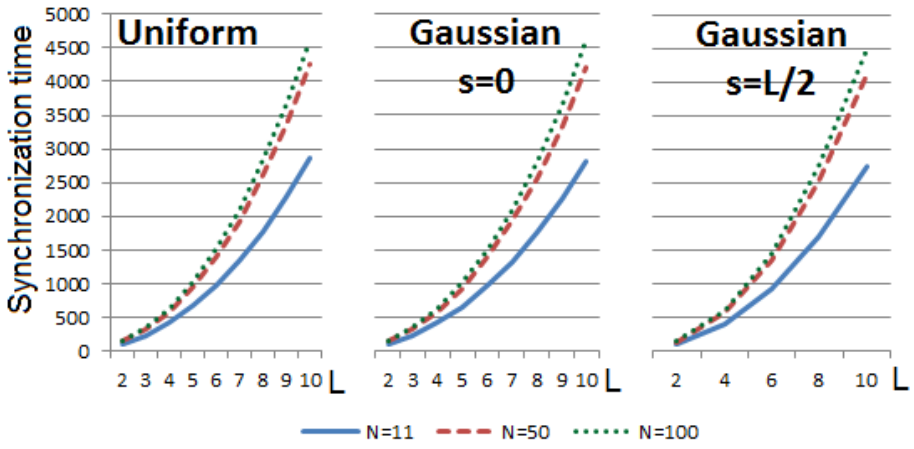

Fig. 2. Synchronization time in learning cycles for different distribution and TPM parameters.

Due to the clarity of the results of detailed synchronization time, the presented values are expressed in percentage, where the time obtained for uniform distribution as a reference point with 100% is considered.

Table 2. Synchronization time details in percentage.

L	UNIFORM	GAUSS 0	GAUSS 2
	N=11		
2	100%	100%	94%
4	100%	99%	95%
6	100%	99%	96%
8	100%	99%	96%
10	100%	98%	96%
	N=50		
2	100%	99%	95%
4	100%	99%	97%
6	100%	100%	97%
8	100%	99%	97%
10	100%	99%	97%
	N=100		
2	100%	99%	95%
4	100%	99%	97%
6	100%	100%	96%
8	100%	99%	97%
10	100%	100%	98%

The result of our analysis shows that the closer the initial weights are, the shorter synchronization time is. Table 2 enlists detailed results of the corresponding average synchronization time for each network obtained upon 5 000 simulations. In particular, these outcomes emphasize the impact of the selected initial weights distributions on the length of networks learning. Visibly, given a fixed distribution, if the initial weights values are drawn closer to the distribution mean, then the networks synchronize faster.

The results obtained for a fixed distribution in percentage are similar despite the change of parameter L. Note that synchronization time significantly increases (see Table 1), but presenting the normal distribution with various standard deviation in relation to the synchronization time for uniform distribution permits to show the nature of the variability of synchronization time. This allows for averaging the results obtained for networks with different parameter L. Significant impact of this parameter's value on the distribution of the TPM synchronization time is studied in [21, 23]. Moreover, the analysis of the geometry of the distribution's histograms forms a further research topic, which is to be conducted in the context of numerical methods [26, 27]. Table 3 shows an average synchronization time for each analyzed distribution and weights interval which is determined by the parameter L.

Table 3. An average synchronization time for different distribution and number of inputs.

N	UNIFORM	GAUSSIAN s=0	GAUSSIAN s=L/2	GAUSSIAN s=L/5
11	2858,83	2814,28	2745,47	2422,38
50	4251,63	4219,30	4131,35	3714,64
100	4608,29	4594,18	4500,68	4133,94

Table 4 shows how the synchronization time decreases in comparison with the uniform distribution after 5000 simulations performed. The latter is conducted for the networks with three first layers and for the number of input signals given by parameter N. The average synchronization time expressed in percentages takes the time obtained for the uniform distribution as 100%.

Table 4. An average synchronization time percetage.

N	UNIFORM	GAUSSIAN s=0	GAUSSIA N s=L/2	GAUSSIAN s=L/5
11	100%	98%	96%	85%
50	100%	99%	97%	87%
100	100%	100%	98%	90%

The drawing of the initial weights from the Gaussian distribution enables to reduce the respective network synchronization time. Comparing the results for a network of fixed parameter L, it can be seen that for the same range of weights the resulting weights' reduction is similar. In addition, having reduced the standard deviation for fixed number of TPM inputs the synchronization time can be even reduced by 15% for the relatively small networks.

3 Conclusions

Neural cryptography like the other new methods such as quantum cryptography [28, 29] is an interesting alternative to the currently used algorithms [2]. The TPM network' synchronization time is the most important issue built in neural cryptography. In this paper a new topic is analyzed, namely the impact on synchronization time of the distribution, from which initial network's weights are drawn. We analyzed here a uniform and Gaussian distributions with a standard deviation depending on the parameters of the synchronized TPM network. Clearly random drawing of the initial weights based on different distributions impacts on their compliance level. By setting uniform distribution as a reference point, it can be observed that the choice of Gaussian distribution results in diminishing the networks' synchronization time. Additionally, the smaller standard deviation is, the shorter synchronization time follows. Generally, the greater synchronized network is, the whole process takes longer. Particularly noteworthy is the fact that the results are similar for networks of various sizes. The

use of standard deviation dependent on the networks parameter enables to compare results obtained for TPMs with different sizes. Upon analyzing the results of a percentage of the synchronization time of network with the initial weights drawn from the uniform distribution it can be seen that similar decreases of synchronization time for networks with different numbers of inputs occurs. The percentage of the results and the dependence of the standard deviation on the network parameters permits to compare the results for networks with various sizes. In this paper we demonstrated that the choice of initial weights distribution, according to which they are drawn, has an impact on the synchronization time. Its reduction in turn may well facilitate the practical use of the TPM network synchronization to determine the encryption keys. These keys can be used in various sensitive data encryption including image [30, 31] or sound encryption [32].

4 References

1. Barker E., Barker W., Burr W., Polk W., Smid M.: Recommendation for key management – part 1: general (revision 3), National Institute of Standards and Technology Special Publication 800-57 (2012)
2. Menezes, A., Vanstone, S., Van Oorschot, P., Handbook of Applied Cryptography, CRC Press (1996)
3. Stinson, D. R.: Cryptography, Theory and Practice, CRC Press (1995)
4. Stokłosa, J., Bilski, T., Pankowski, T.: Data Security in Informatical Systems (in Polish), PWN (2001)
5. Klimov, A., Mityagin, A., Shamir, A.: Analysis of Neural Cryptography, In Y. Zheng, editor, Advances in Cryptology - ASIACRYPT 2002, Springer 287-298 (2003)
6. Ruttor, A., Kinzel, W., Naeh, R., Kanter, I.: Genetic Attack on Neural Cryptography, Phys. Rev. E 73(3) 036121-036129 (2006)
7. Kanter, I., Kinzel, W., Kanter, E.: Secure Exchange of Information by Synchronization of Neural Networks, Europhys. Lett. 57 141-147 (2002)
8. Kanter, I., Kinzel, W.: The Theory of Neural Networks and Cryptography, Proceeding of the XXII Solvay Conference on Physics, The Physics of Communication 631-644 (2003)
9. Rosen-Zvi, M., Klein, E., Kanter, I., Kinzel, W.: Mutual Learning in a Tree Parity Machine and its Application to Cryptography. Phys. Rev. E, 66:066135 (2002)
10. Klein, E., Mislovaty, R., Kanter, I., Ruttor, A., Kinzel, W.: Synchronization of Neural Networks by Mutual Learning and its Application to Cryptography, Advances in Neural Information Processing Systems, MIT Press Cambridge 17 689-696 (2005)
11. Ruttor, A.: Neural Synchronization and Cryptography, Ph.D. thesis, Wurzburg (2006)
12. Ibrachim S., Maarof M.: A review on biological inspired computation in cryptology, Jurnal Teknologi Maklumat, 17 90-98 (2005)
13. Dourlens, S.: Neuro-Cryptography. MSc Thesis, Dept. of Microcomputers and Microelectronics, University of Paris, France (1995)
14. Kotlarz, P., Kotulski, Z.: On Application of Neural Networks for S-Boxes Design, LNCS vol. 3528 243-248, Springer (2005)
15. Bisalapur S.: Design of an efficient neural key distribution center, International Journal of Artificial Intelligence & Applications, 2 60-69 (2011)
16. Volkmer M., Wallner S.: Tree parity machine re-keying architectures, IEEE Transactions on Computers, 54 421-427 (2005)

17. Hassoun, M.: Fundamentals of Artificial Neural Networks, MIT Press (1995)
18. Osowski, S.: Neural Networks in Algorithmic Approach (in Polish), WNT (1996)
19. Rutkowski, L.: Methods and Technics of Artificial Intelligence (in Polish), PWN, Warsaw (2006)
20. McCulloch W.: A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5 115-133 (1943)
21. Dolecki M.: Klasyfikacja czasu synchronizacji sieci Tree Parity Machine używanych do uzgadniania kluczy kryptograficznych (in Polish), Ph.D. thesis, Gliwice (2014)
22. Dolecki M., Kozera R.: Threshold method of detecting long-time TPM synchronization, Proceedings of the 12th International Conference on Computer Information Systems and Industrial Management Applications, Lecture Notes in Computer Science 8104 241-252, Springer – Verlag Berlin (2013)
23. Dolecki M., Kozera R.: Distribution of the Tree Parity Machine synchronization time, Advances in Science and Technology Research Journal, 18 20-27 (2013)
24. Dolecki M., Kozera R., Lenik K.: The evaluation of the TPM synchronization on the basis of their outputs, Journal of Achievements in Materials and Manufacturing Engineering, 57 91-98 (2013)
25. Dolecki M., Kozera R.: Distance of tree parity machine initial weights drawn from different distributions, Advances in Science and Technology Research Journal, 26 137-142 (2015)
26. Kozera R., Noakes L.: More-or-less-uniform sampling and lengths of curves, Quarterly Of Applied Mathematics, 61 475-484 (2003)
27. Kozera R., Noakes L.: C-1 interpolation with cumulative chord cubics, Fundamenta Informaticae, 61 285-301 (2004)
28. Gerdt V. P., Prokopenya A. N.: Some algorithms for calculating unitary matrices for quantum circuits, Programming and Computer Software, 36 111-116 (2010)
29. Gerdt V. P., Prokopenya A. N.: Simulation of quantum error correction by means of QuantumCircuit package, Programming and Computer Software, 39 143-149 (2013)
30. Brooks M. J., Chojnacki W., Kozera R.: Shading without shape, Quarterly Of Applied Mathematics, 50 27-38 (1992)
31. Kozera R.: Uniqueness in shape from shading revisited, Journal Of Mathematical Imaging And Vision, 7 123-138 (1997)
32. Homenda W.: Optical music recognition: the case study of pattern recognition, Computer Recognition Systems, Edited by Kurzynski M., Puchala E., Wozniak M. et al. Book Series Advances in Soft Computing 835-842 (2005)