Synchronization of Tree Parity Machines Using Nonbinary Input Vectors

Miłosz Stypiński and Marcin Niemiec

***Abstract*— Neural cryptography is the application of artificial neural networks (ANNs) in the subject of cryptography. The functionality of this solution is based on a tree parity machine (TPM). It uses ANNs to perform secure key exchange between network entities. This brief proposes improvements to the synchronization of two TPMs. The improvement is based on learning ANN using input vectors that have a wider range of values than binary ones. As a result, the duration of the synchronization process is reduced. Therefore, TPMs achieve common weights in a shorter time due to the reduction of necessary bit exchanges. This approach improves the security of neural cryptography.**

***Index Terms*— Artificial neural networks (ANNs), key agreement, mutual learning, neural cryptography, security.**

I. Introduction

Secure key agreement is one of the basic steps in secure channel establishment. The algorithms responsible for the key exchange must ensure that no eavesdroppers are able to reproduce the secure key. Applied key agreement protocols are based on mathematical operations which have no computationally efficient inversion, for example, factorization of a large number of problems or other derived problems.

Quantum computing poses a real threat to applied cryptography systems. Currently used algorithms, based on the public-key cryptography approach, offer conditional security. Efficient derivation of a secure key from exchanged fragmentary information may break the security of the key agreement protocol. Currently, there is one known algorithm—Shor's algorithm—capable of factorizing large numbers. Hence, it can extract exchanged keys and break all applied asymmetric cipher cryptography [1]. However, the successful implementation of this algorithm requires a quantum computer with a sufficient number of qubits.

Some modern cryptography techniques—such as quantum cryptography and neural cryptography—are able to overcome this problem and provide a variety of quantum-proof algorithms. The tree parity machine (TPM) is one such solution. It achieves a key agreement functionality by mutual learning of two artificial neural networks (ANNs). Mutual learning cannot be reduced to either primes factorization or discrete logarithm problems, hence it is not susceptible to quantum computing.

This brief proposes an acceleration to the mutual learning process of TPMs. The following novelties are considered:

1) introduction of nonbinary input vectors used for the synchronization of TPM;
2) proposal of a new parameter M describing input vector;
3) verification of the proposed solution in the insecure environment (with eavesdropper);
4) analysis of observed phenomenon called extrema values effect.

The proposed improvement results in a shorter synchronization, which at the same time enhances security properties.

This brief is structured as follows. Section II presents related works relevant to the subject of this brief. Section III outlines the architecture of TPMs, the process of mutual learning, secure key agreement protocol, exchanged key length, and security of TPMs. Section IV describes the nonbinary input TPM architecture, entropy, and its appliance in terms of quality assessment of exchanged key. Additionally, Section IV presents observed phenomena called extrema values effect. Section V describes the methodology of the performed simulations and an analysis of the gathered results.

II. Related Works

Kanter *et al.* [2] and Rosen-Zvi *et al.* [3] in 2002 successfully performed key agreement via mutual learning. After that, various efforts took place to enhance the proposed TPMs as a competitive security solution. Santhanalakshmi *et al.* [4] proposed the usage of a genetic algorithm in finding optimal weight vectors for training TPM. This solution reduces the synchronization time but increases the computational complexity. Allam *et al.* [5] proposed improvement based on a shared secret. As a result, the complexity of known attacks is increased while maintaining the same synchronization time. Another usage scenario was described in [6] where TPM was used as an error correction mechanism for key agreed in quantum cryptography system. Another approach significantly changing the TPM architecture is introduced in [7]. By using the original whale optimization algorithm for synchronization and by adding another hidden layer into the neural network, authors achieve higher security compared to standard TPM. Dong and Huang [8] describe enhancement achieved by learning TPM with complex-valued input vectors. This solution is generalized in [9], where vector-valued inputs are proposed. A different approach is described by Sarkar [10], where chaos-generated input vectors are utilized.

III. Tree Parity Machine

ANNs are increasingly popular, finding applications in fields including security. Kanter *et al.* [2] and Rosen-Zvi *et al.* [3] introduce a novel approach for the key agreement functionality implemented with neural networks, which are explained in more detail in this section.

A. Tree Parity Machine Architecture

A TPM is a two-layered perceptron-structured ANN with discrete weights, binary input, and binary output [11]. The input vector $X = [x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{k1}, \ldots, x_{kn}]$, $K, N, k, n \in \mathbb{N} \wedge k \leq K \wedge n \leq N$ has KN elements, where K denotes the number of inputs for each neuron in the first layer, and N indicates the number of neurons in the first layer. Every element x_{kn} of input vector X can have one of two possible values, either -1 or 1.

The first layer consists of neurons similar to the McCulloch–Pitts model [12]. Every input x_{kn} is connected to the kth neuron and has its corresponding weight. The values of the weights are the only difference from the former model. Every weight w_{ij} can take a value

Manuscript received April 22, 2021; revised October 30, 2021 and February 28, 2022; accepted May 22, 2022. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme through the ECHO Project under Grant 830943. *(Corresponding author: Miłosz Stypiński.)*

The authors are with the Institute of Telecommunications, AGH University of Science and Technology, 30-059 Kraków, Poland (e-mail: stypinski@agh.edu.pl; niemiec@agh.edu.pl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3180197.

Digital Object Identifier 10.1109/TNNLS.2022.3180197

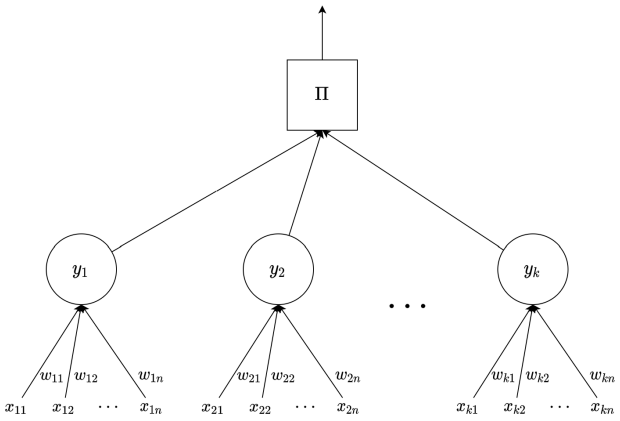

Fig. 1. Architecture of the TPM.

between $-L$ and L, where $L \in \mathbb{Z}$ is the parameter of the TPM and denotes the minimum/maximum possible weight value of the input neurons.

The output of the aforementioned neurons is based on the slightly changed signum function σ. The formula of the function is presented in (1). It differs from the regular signum function in that it never returns zero. The value of 0 is mapped either to 1 or -1, based on whether the side is the sender or the recipient of the communication [13]. The recipient and sender side is denoted by r and s, respectively. The parties decide beforehand which side is the sender and the recipient

$$\sigma\left(x^{r/s}\right) = \begin{cases} 1, & x^r \geqslant 0 \vee x^s > 0 \\ -1, & x^r < 0 \vee x^s \leqslant 0. \end{cases} \tag{1}$$

The argument for the neuron's activation function is the sum of the products of the input vector's elements with corresponding weight. The exact formula is presented in the following equation:

$$y_k = \sigma\left(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}\right). \tag{2}$$

The final result O of the TPM is the product of each of the outputs from hidden neurons from the first layer (3)

$$O^{r/s} = \prod_{k}^{K} y_k^{r/s}. \tag{3}$$

The overall architecture of the TPM is shown in Fig. 1.

B. Key Agreement Protocol

The parties performing the key agreement execute the protocol, which results in the secure shared key known only to the participating parties. This is usually achieved through the exchange of some information through an unsecured channel and by performing mathematical operations whose results are only known to the authorized parties [14]. The first and most popular key agreement protocol was proposed by Diffie and Hellman [15].

TPM offers functionality that can be adopted for key exchange purposes. The protocol for two parties consists of the following steps [13].

1) Both participants must agree on all the parameters for the TPM (K, L, N) and initialize their TPMs with random weights.
2) The key agreement participants publicly exchange a previously chosen binary random input vector X.
3) Each party computes the output from their TPM and publishes the results.
4) If the outputs match, both participants apply the appropriate learning rule that updates the weights of TPM accordingly.
5) Steps 2–4 are repeated until full synchronization of both TPMs is achieved.

The full synchronization is equivalent to every corresponding weight of both TPMs being equal to each other, at which point both TPMs are the same.

The aforementioned learning rules are responsible for updating the weights of each TPM in such a way that the synchronization process finishes in finite time [16]. There are three different learning rules that can be used in the process of updating weights [17].

1) *Hebbian Learning Rule:*

$$w_{kn}(t+1) = w_{kn}(t) + O(t)x_{kn}(t)\Theta(y_k(t), O(t)). \tag{4}$$

2) *Anti-Hebbian Learning Rule:*

$$w_{kn}(t+1) = w_{kn}(t) - O(t)x_{kn}(t)\Theta(y_k(t), O(t)). \tag{5}$$

3) *Random Walk Learning Rule:*

$$w_{kn}(t+1) = w_{kn}(t) + x_{kn}(t)\Theta(y_k(t), O(t)) \tag{6}$$

where $\Theta(a, b)$ denotes the function returning 1 if $a = b$ and 0 otherwise, and parameter t denotes the iteration in the key agreement algorithm.

The synchronization process of two parity machines is not a deterministic algorithm. The number of iterations is not fixed and depends on the size and parameters of the TPM. However, it is shown that the time is finite and can be easily estimated by users [18]. The process takes longer for larger TPM sizes (K and N) and maximum weight value (L). Other factors that affect the number of iterations required for two TPMs to finish mutual learning include distribution of initial weights and learning rule [19].

C. Security of Tree Parity Machines

Security of key agreement protocol is crucial for communication. Any eavesdropper being able to reproduce the key based on the messages exchanged between parties or any other source breaks the security of the channel. Subsequently, such a situation depreciates the secure key exchange protocol. Hence, it is crucial to assess the security of any novel algorithm or protocol.

TPMs has been studied extensively. Javurek and M. Turčaník [18] and Martínez Padilla *et al.* [20] identify four distinct types of attacks that TPM may be vulnerable to as follows.

1) *Brute force attack:* Research shows that it is impossible to find the exact key as a result of a brute force attack against TPMs in polynomial time.
2) *Genetic algorithm for weight prediction:* It has been shown that only TPMs with a single neuron in the second layer are vulnerable to this type of attack.
3) *Man-the-middle interception attack:* Studies show that on average 60% of weights were synchronized in the eavesdropper's TPM.
4) *Sign of weight classification using neural networks:* Martínez Padilla *et al.* [20] demonstrate that classification using ANNs has near 100% accuracy in determining the sign of the weight in the TPM, which reduces the time needed by the brute force attack by almost half.

The studies show that, by utilizing these attack vectors, it is possible to gain some information about the key. Hence, cryptosystems should be aware of this threat and counteract it in order to minimize the likelihood of key reconstruction.

D. Man-in-the-Middle Attack

Synchronization of two TPMs without additional layers of security is a process prone to man-in-the-middle attacks. This attack relies on the possibility of placing a node C between parties A and B performing a key agreement. The node eavesdrops on all the messages shared between A and B. Based on information collected, node C may be able to gain unauthorized access to information sent between A and B. Moreover, if the nodes are not mutually authenticated, the adversarial party may be able to alter the messages accordingly to attempt an attack with a higher probability of success.

In terms of TPMs, man-in-the-middle attacks come down to capturing all the input vectors X and outputs of parties being intercepted. An adversarial TPM performs the learning process on acquired data. There are three scenarios to be considered while intercepting the key exchange. Let A and B be the parties wishing to exchange the key and let C be an intruder able to perform a man-in-the-middle attack. The three scenarios are as follows.

1) If $\Pi_A \neq \Pi_B$, no TPMs is synchronized during this step.
2) If $\Pi_A = \Pi_B \neq \Pi_C$, only TPMs A and B are synchronized, while TPM C (attacker) does not update its weights.
3) If $\Pi_A = \Pi_B = \Pi_C$, all the TPMs update their weights accordingly.

The last scenario brings the adversarial party closer to obtaining the exchanged key. Hence, this situation should be avoided at all costs.

IV. Nonbinary Input Vectors

The TPMs use binary vectors X for input [2] during the synchronization process. This brief introduces a new approach: nonbinary input vectors used to synchronize TPMs for a secure key agreement protocol. The authors propose that the mutual learning process that uses the vectors with a greater range of possible values of every element influence the synchronization time of two TPMs. Simulations performed in Section IV verify this proposition and indicate that this approach can significantly increase the security of neural cryptography.

A. Nonbinary Vector Tree Parity Machine Architecture

So far, the exact TPM was defined by parameters K, L, N. In this brief, the authors introduce a new parameter M, denoting the minimum/maximum value of each element of input vector X. Hence, the input vector will have the following form: $X = [x_{11}, x_{12}, \ldots, x_{1n}, \ldots, x_{k1}, \ldots, x_{kn}]$, where $x_{kn} \in \{x : x \in \mathbb{Z} \wedge -M \leq x \leq -1 \vee 1 \leq x \leq M\}$. Thus, during the synchronization process, the entities can use nonbinary input vectors, instead of binary vectors which are currently used in practical implementations.

Introducing the M parameter does not affect the architecture of the TPM. The new learning process slightly differs from the original. The changes are in the first two steps of the key agreement protocol presented in Section III-B. The first two steps now read as follows.

1) Both synchronization participants must agree on all the parameters for the TPM (K, L, M, N) and initialize their TPMs with random weights.
2) The key agreement participants publicly exchange a previously chosen random input vector X, which now consists of values ranging from $-M$ to M.

Points 3–5 remain unchanged.

Furthermore, formulas shown in Section III are still valid despite more divergent values of the learning vectors. However, simulations presented in Section V show that as the input vectors are more differentiated, the distribution of settled keys is less similar to the uniform distribution. Therefore, unbiased estimation of key length is required.

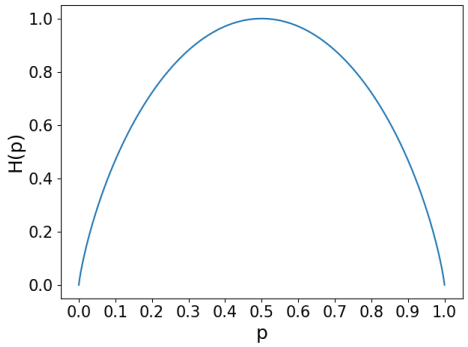

Fig. 2. Entropy of the source generating two different values with the same probability.

TABLE I
Synchronization Time of TPMs With Differentiated Input Vectors

M	N	Synchronization time			
		Average	**Minimum**	**Maximum**	**Median**
1	40	709 ± 490	313	2176	648
2		290 ± 216	103	965	273
3		172 ± 138	75	484	156
4		114 ± 82	39	314	105
5		84 ± 64	28	249	78
1	50	714 ± 453	333	1981	666
2		316 ± 209	145	769	296
3		172 ± 126	69	455	158
4		118 ± 88	39	280	108
5		87 ± 64	28	216	82
1	60	733 ± 398	391	1763	715
2		320 ± 202	140	652	306
3		182 ± 122	79	421	172
4		121 ± 87	43	284	117
5		90 ± 70	34	233	81

B. Synchronized Key Quality

The quality of random numbers generation has a significant impact on the final security of the cryptosystem. A true random number generator produces every available output with equal probability. Unfortunately, computers are incapable of generating fully random numbers. Frequently, numbers are generated based on a pseudorandom number generator. This requires a seed supplied beforehand, which is the starting point of the pseudorandom number sequence, and each further number depends on it. Many contemporary implementations lack important features like good mathematical foundations, lack of predictability, and cryptographic security [21].

Entropy is one of the measures which assesses the quality of the generated numbers. Let us assume the random source generates I different numbers $\alpha_1, \alpha_2, \ldots, \alpha_i$ with corresponding probabilities $p_1, p_2, \ldots, p_i$. Entropy for such a defined source is presented in the following equation [22]:

$$H(p_1, p_2, \ldots, p_i) = -\sum_{i=1}^{I} p_i \log_j p_i. \tag{7}$$

The base of logarithm j denotes the units in which entropy is measured, for example, for 2 and e units are bits and nats, respectively [23].

Let us consider a random source that produces two outputs with, either 0 or 1 with corresponding probabilities $P(X = 0) = p$ and $P(X = 1) = 1 - p$. The entropy for the described source is presented

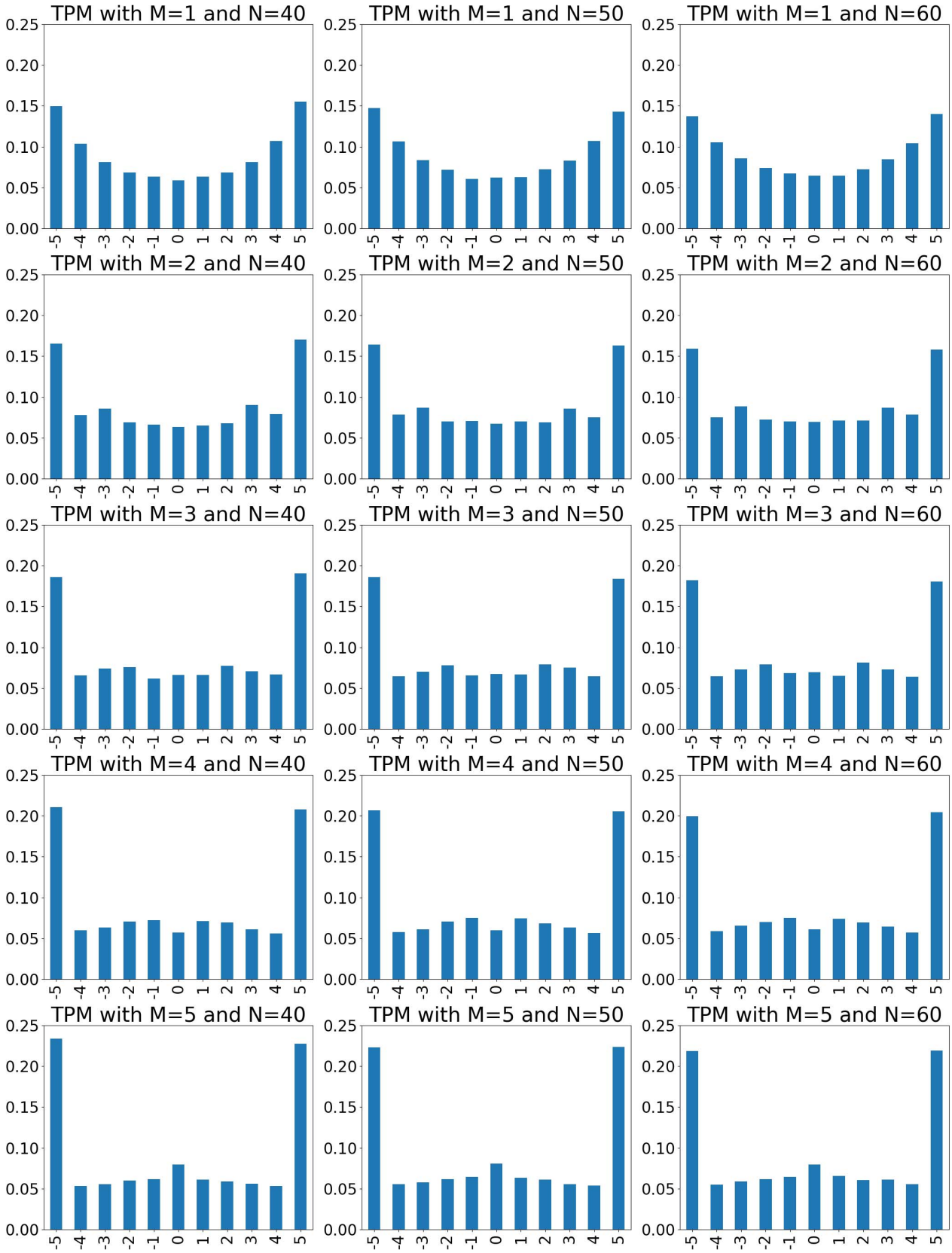

Fig. 3. Probability distribution of weights in TPMs using nonbinary input vectors.

in the following equation [23]:

$$H(p) = -p\log_j p - (1-p)\log_j(1-p). \quad (8)$$

Fig. 2 shows the plot of the entropy of the aforementioned two-value random source. The maximum of the function is reached for $p = 0.5$, where $H(p) = 1$, which is the equal probability for values 0 and 1.

Hence, entropy values increase as the probability distribution of X gets closer to the uniform distribution. This can be generalized to sources producing more outcomes.

The entropy function can be used later to assess the quality of the keys generated by different types of TPM. Taking into account (9), the effective length of a key should depend on the entropy of the synchronized weights (not just their values).

C. Agreed Key Length

After the synchronization process, both parties share identical keys. The keys are distilled from weights of the TPM, which are the same after the mutual learning process. The key length depends on the size of the TPM as well as the parameter L, which indicates the minimum/maximum value the weights may reach during synchronization. Assuming the ideal uniform distribution of the weights, the key length is equal to $K \cdot N \cdot log_2(2L+1)$. However, the distribution of the weights differs from the uniform distribution [17]. Hence, the entropy should be used to measure the quality of the key exchanged between the parties. The updated key length is defined as follows:

$$\text{length}_{\text{key}} = K \cdot N \cdot E(W) \tag{9}$$

where $E(W)$ indicates an average entropy of the weights. Entropy itself is presented in Section IV-B.

However, the exact distribution of weights is not known beforehand. Taking this fact into consideration, (9) should be updated. The estimated effective key length shown in (10) uses the estimated entropy based on the simulation results. Additionally, we propose using the floor function in the equation since the unit of effective key are bits

$$\hat{\text{length}}_{\text{key}} = \lfloor K \cdot N \cdot \hat{E}(W) \rfloor. \tag{10}$$

It should be noted that (9) indicates the theoretical maximum key length that can be extracted from mutual weights. However, a dedicated algorithm that equalizes the probability can be used to obtain a cryptography key from an unevenly distributed numerical sequence. This algorithm must be deterministic since both parties retrieve the cryptographic key from weights simultaneously.

V. Verification

This section presents the impact of the new parameter M, indicating the maximum/minimum value of the input vectors during the synchronization process and how it affects the required iterations in the learning process and the quality of the output key.

A. Methodology

To validate the proposed improvement, a dedicated TPM simulation framework has been created. The proposed framework allows mimicking a real network scenario where two parties willing to synchronize their TPMs exchange all the intermediate information via a public channel (parties perform actions from Section IV-A). Hence, the agreed TPM parameters and all the input vectors are available to eavesdroppers. The simulation framework is available in the public domain.[1]

The quality of the output key is measured in its effective length. The effective length is calculated on the basis of (10). Furthermore, simulation scenarios cover multiple sets of TPMs sizes. For each scenario, statistical analysis was prepared based on 1000 simulations. The presented confidence intervals are calculated with a 95% probability. These scenarios include all possible combinations of parameters $N \in \{40, 50, 60\}$ and $M \in \{1, 2, 3, 4, 5\}$. For all simulation scenarios, parameters K and L are equal to 3 and 5, respectively. Additionally, all the simulations are performed using the Hebbian learning rule (4). Synchronization time, entropy, and effective key length are measured in order to compare the chosen scenarios. Furthermore, we performed man-in-the-middle attack scenarios during which we measured the average synchronization score of the malicious TPM.

[1]The framework source code is available at *https://github.com/mstypinski/tpm*

B. Results

The synchronization process becomes longer as the size of the TPM increases; it also generates a longer key for cryptographic purposes. However, simulations presented in Table I reveal that the TPM size and parameters are not the only elements that have an impact on the duration of the synchronization process. Multiple simulations were performed with different values of parameter M. An increase in the parameter M value, which limits the maximum and minimum possible values x_i of the input vector X, reduces the synchronization time significantly. The synchronization time in Table I is expressed as a number of output bits exchanged between the parties to achieve full synchronization between the two TPMs (learning iterations). Thus, the volume of data exchanged between the parties performing key agreement decreases as the value of parameter M increases.

It should be noted that faster synchronization increases security. This is because as the value of parameter M increases, the key agreement process takes less time, hence a longer and more secure key is obtained in a shorter period of time. This makes this solution more competitive among other key exchange protocols.

Salguero Dorokhin *et al.* [24] conducted research regarding the optimal TPM structure for establishing a 512-bit cryptographic key. Based on their research, TPM parameters (K, L, N) providing the most security are $(8, 16, 8)$. On average, the full synchronization of such a TPM took 218.37 iterations. Comparing the above to the conducted research in this brief, TPMs having parameters (M, N) set to one of the following pairs (3, 50), (4, 50), (4, 60) perform better in terms of synchronization time while allowing to obtain cryptographic keys longer than 512 bits.

It is worth mentioning the synchronization time grows insignificantly with the increase of N. This allows extending distilled cryptographic keys without significantly extending synchronization time. The parameter having the highest impact on synchronization time is L as synchronization time grows proportionally to L^2. However, the same parameter is responsible for improving the security features of TPM [25]. The introduction of parameter M enables preserving comparable security features but results in a decrease in synchronization time.

C. Extrema Values Effect

Numerous simulations of the TPM learning process using nonbinary input vectors led to the discovery of an effect named by the authors the extrema value effect. A similar effect is shown in [10], however, only binary input vectors are considered in this brief.

Faster synchronization times and lower numbers of messages exchanged between users have an impact on the distribution of weights. As the minimum/maximum x_i increases, the probability $P(w_{kn} = M)$ and $P(w_{kn} = -M)$ also increases. As a result, the probability distribution of weights becomes less similar to the uniform distribution. Hence, every weight of the TPM carries less random information. The exact distribution of weights is presented in Fig. 3.

TABLE II

ENTROPY AND EFFECTIVE KEY LENGTH FOR DIFFERENT TPM PARAMETERS

M	N	Entropy	Estimated effective key length
1	40	3.374	404
2		3.354	402
3		3.305	396
4		3.238	388
5		3.158	378
1	50	3.386	507
2		3.368	505
3		3.315	497
4		3.248	487
5		3.186	477
1	60	3.402	612
2		3.379	608
3		3.325	598
4		3.263	587
5		3.204	576

TABLE III

SYNCHRONIZATION TIME OF TPMS WITH DIFFERENTIATED INPUT VECTORS

M	N	Adversarial TPMs synchronization score (S_{score})			
		Average	Minimum	Maximum	Median
1	40	0.174 ± 0.047	0.058	0.425	0.167
2	40	0.18 ± 0.055	0.058	0.592	0.167
3	40	0.205 ± 0.079	0.075	0.933	0.183
4	40	0.243 ± 0.101	0.083	0.983	0.208
5	40	0.28 ± 0.136	0.083	1	0.233
1	50	0.17 ± 0.039	0.073	0.373	0.16
2	50	0.179 ± 0.047	0.08	0.513	0.167
3	50	0.204 ± 0.06	0.087	0.527	0.187
4	50	0.232 ± 0.085	0.087	0.993	0.207
5	50	0.255 ± 0.107	0.067	1	0.213
1	60	0.168 ± 0.037	0.072	0.361	0.167
2	60	0.183 ± 0.048	0.061	0.417	0.172
3	60	0.204 ± 0.066	0.067	0.65	0.183
4	60	0.231 ± 0.084	0.078	0.956	0.206
5	60	0.252 ± 0.117	0.089	1	0.211

Assuming that in tth iteration, the update of all weights for all the neurons is successful and the distribution of all the weights $w_{kn}(t)$ and elements of input vector $x_{kn}(t)$ is uniform, let us estimate the probability of the weight $w_{kn}(t+1)$ being equal to L or $-L$ in the next iteration. We can only consider one of these values since the situation is analogous to the other. The probability $P(w_{kn}(t+1) = L)$ is equal to $P(w_{kn}(t+1) = L|x_{kn} = 0) + P(w_{kn}(t+1) = L|x_{kn} = 1) + \cdots + P(w_{kn}(t+1) = L|x_{kn} = M-1) + P(w_{kn}(t+1) = L|x_{kn} = M)$. Let us consider $P(w_{kn}(t+1) = L|x_{kn} = 0)$. This event only occurs providing $w_{kn} = L$, and therefore the probability is equal to $(1/(2M+1)(2L+1))$, as there is only one satisfactory event for all combinations of input and weight values. For $\mathrm{P}(w_{kn}(t+1) = L|x_{kn} = 1)$, there are two satisfactory events $w_{kn} = L$ and $w_{kn} = L-1$. In conclusion, the probability of the event A where $w_{kn}(t+1) = L$ or $w_{kn}(t+1) = -L$ is presented in the following equation:

$$\begin{aligned} P(A) &= 2 \times (P(A|x_{kn} = 0) + \cdots + P(A|x_{kn} = M)) \\ &= 2 \times \left(\frac{1 + \cdots + (M+1)}{(2M+1)(2L+1)} \right) \\ &= \frac{(M+1)(M+2)}{(2M+1)(2L+1)}. \end{aligned} \tag{11}$$

As the parameter M increases the probability of the weights being equal to either L or $-L$ raises in the next iteration, resulting in diminished TPM robustness. This fact makes parameter selection a key element of TPM security.

The unequal distribution of weights in the TPM results in a reduction of the effective key length since as the entropy value becomes lower, the less secure bits might be retrieved from the key. Entropy values and effective key lengths are presented in Table II. To visualize the proportion between effective key length, the results for the considered M and N parameters are presented in Fig. 4.

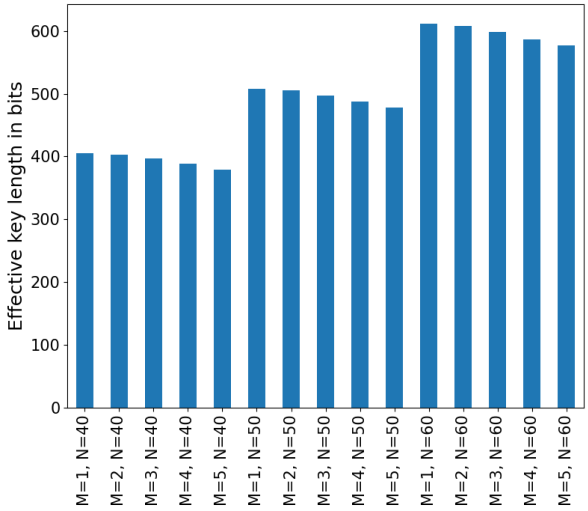

Fig. 4. Effective key length of TPMs with different parameters.

D. Susceptibility to a Man-in-the-Middle Attack

Many research considerations address TPM vulnerability to man-in-the-middle attacks. Therefore, simulations with adversarial TPMs have been conducted while utilizing learning by nonbinary input vectors.

We assumed the worst-case scenario in which the adversarial neural network was able to eavesdrop on all of the data exchanged between the parties performing the key exchange. During the simulations, the final synchronization score S_{score} was gathered for the adversarial neural network. The synchronization score measures the similarity between two TPMs. The more common weights there are, the higher the score value is assigned. Hence, the formula needs to return higher values with the progress of the learning process. For equal TPMs, the synchronization score is equal to 1. The formula for calculating the end score is presented in (12). In the following equation, w^A denotes weights of adversarial TPMs and function $\Theta(a, b)$ is defined in Section III:

$$S_{\text{score}} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \Theta\left(w_{kn}, w_{kn}^{A}\right)}{K \times N}. \tag{12}$$

In terms of security, the attacker's TPM should have the lowest synchronization score possible.

The synchronization score of adversarial TPMs are presented in Table III. Additionally, simulation results are shown in Fig. 5 to visualize the relationship between scenarios with different TPMs. Increased values of parameter M result in higher median synchronization scores, hence the TPM is more prone to man-in-the-middle attacks. When parameter M was equal to L, we observed situations where the synchronization score was equal to 1. This means that the relationship between parameters $M < L$ should be preserved to ensure security. Additionally, the median is

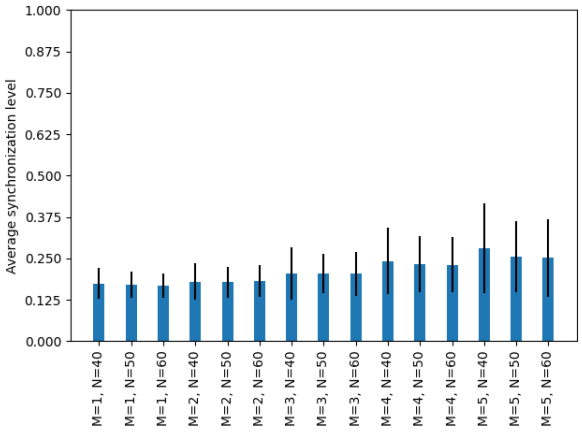

Fig. 5. Synchronization score of adversarial TPMs.

inversely proportional to the number of inputs N, and therefore the impact of nonbinary input vectors on the synchronization score is less clear for larger TPMs. Furthermore, the confidence intervals are considerable. This variability makes it difficult to predict the attacker's malicious TPM weights.

VI. Summary

Correct selection of TPM parameters is a key issue in implementing secure key agreement protocols for neural cryptography. It is crucial to find a tradeoff between effective key length, synchronization time, and security of the final key which is used by users to protect data in the network environment. This comes down to selecting the appropriate network size, extreme values of the weights, and learning vectors.

This brief proposes an improved way of learning TPMs. A significant acceleration of the key agreement process was achieved by utilizing a nonbinary input vector. This reduces the volume of data exchanged between the parties performing key agreements. Faster synchronization increases security levels; in particular, it mitigates the risk of the key being obtained by an intruder using a man-in-the-middle attack. However, speeding up the process results in an unequal distribution of weights in the TPM. This was measured by calculating the effective key length based on the entropy of each weight. The proposed solution was also verified in an insecure environment in which two TPMs are subject to a man-in-the-middle attack.

We envisage that future work will explore the development of a secure key exchange protocol using nonbinary input vectors in TPMs during mutual learning. This work will be focused on studying the extrema values effect thoroughly and minimizing the reduction of effective key length.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, Jan. 1999.

[2] I. Kanter, W. Kinzel, and E. Kanter, "Secure exchange of information by synchronization of neural networks," *Europhys. Lett.*, vol. 57, no. 1, pp. 141–147, Jan. 2002.

[3] M. Rosen-Zvi, I. Kanter, and W. Kinzel, "Cryptography based on neural networks analytical results," *J. Phys. A, Math. Gen.*, vol. 35, no. 47, pp. L707–L713, Nov. 2002, doi: 10.1088/0305-4470/35/47/104.

[4] S. Santhanalakshmi, T. S. B. Sudarshan, and G. K. Patra, "Neural synchronization by mutual learning using genetic approach for secure key generation," in *Recent Trends in Computer Networks and Distributed Systems Security*, S. M. Thampi, A. Y. Zomaya, T. Strufe, J. M. A. Calero, and T. Thomas, Eds. Berlin, Germany: Springer, 2012, pp. 422–431.

[5] A. M. Allam, H. M. Abbas, and M. W. El-Kharashi, "Authenticated key exchange protocol using neural cryptography with secret boundaries," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–8.

[6] M. Niemiec, "Error correction in quantum cryptography based on artificial neural networks," *Quantum Inf. Process.*, vol. 18, no. 6, p. 174, Jun. 2019, doi: 10.1007/s11128-019-2296-4.

[7] A. Sarkar, M. Z. Khan, M. M. Singh, A. Noorwali, C. Chakraborty, and S. K. Pani, "Artificial neural synchronization using nature inspired whale optimization," *IEEE Access*, vol. 9, pp. 16435–16447, 2021.

[8] T. Dong and T. Huang, "Neural cryptography based on complex-valued neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4999–5004, Nov. 2020.

[9] S. Jeong, C. Park, D. Hong, C. Seo, and N. Jho, "Neural cryptography based on generalized tree parity machine for real-life systems," *Secur. Commun. Netw.*, vol. 2021, Feb. 2021, Art. no. 6680782.

[10] A. Sarkar, "Secure exchange of information using artificial intelligence and chaotic system guided neural synchronization," *Multimedia Tools Appl.*, vol. 80, no. 12, pp. 18211–18241, Feb. 2021, doi: 10.1007/s11042-021-10554-3.

[11] S. Marsland, *Machine Learning: An Algorithmic Perspective*, 2nd ed. Boca Raton, FL, USA: CRC Press, 2014.

[12] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943, doi: 10.1007/BF02478259.

[13] M. Volkmer and S. Wallner, "Tree parity machine rekeying architectures," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 421–427, Apr. 2005.

[14] M. Just, *Key Agreement*. Boston, MA, USA: Springer, 2005, p. 325, doi: 10.1007/0-387-23483-7_218.

[15] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.

[16] W. Kinzel, *Theory of Interacting Neural Networks*. Hoboken, NJ, USA: Wiley, 2002, ch. 9, pp. 199–217, doi: 10.1002/3527602755.

[17] A. Ruttor, "Neural synchronization and cryptography," Ph.D. dissertation, Dept. Phys. Astron., Univ. Würzburg, Würzburg, Germany, 2007.

[18] M. Javurek and M. Turcanik, "Synchronization of two tree parity machines," in *Proc. New Trends Signal Process. (NTSP)*, Oct. 2016, pp. 1–4.

[19] M. Dolecki and R. Kozera, "Distance of the initial weights of tree parity machine drawn from different distributions," *Adv. Sci. Technol. Res. J.*, vol. 9, no 26, pp. 137–142, 2015, doi: 10.12913/22998624/2380.

[20] J. Martínez Padilla, U. Meyer-Baese, and S. Foo, "Security evaluation of tree parity re-keying machine implementations utilizing side-channel emissions," *EURASIP J. Inf. Secur.*, vol. 2018, no. 1, pp. 1–16, Apr. 2018, doi: 10.1186/s13635-018-0073-z.

[21] M. E. O'Neill, "PCG: A family of simple fast space-efficient statistically good algorithms for random number generation," Harvey Mudd College, Claremont, CA, USA, Tech. Rep., HMC-CS-2014-0905, Sep. 2014.

[22] E. Simion, "Entropy and randomness: From analogic to quantum world," *IEEE Access*, vol. 8, pp. 74553–74561, 2020.

[23] T. M. Cover and J. A. Thomas, *Elements of Information Theory* (Wiley Series in Telecommunications and Signal Processing). Hoboken, NJ, USA: Wiley, 2006.

[24] E. S. Dorokhin, W. Fuertes, and E. Lascano, "On the development of an optimal structure of tree parity machine for the establishment of a cryptographic key," *Secur. Commun. Netw.*, vol. 2019, Mar. 2019, Art. no. 8214681, doi: 10.1155/2019/8214681.

[25] R. Mislovaty, Y. Perchenok, I. Kanter, and W. Kinzel, "Secure key-exchange protocol with an absence of injective functions," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 66, no. 6, Dec. 2002, Art. no. 066102.