energies

MDPI

Article

Security of Neural Network-Based Key Agreement Protocol for Smart Grids

Miłosz Stypiński * and Marcin Niemiec *

Institute of Telecommunications, AGH University of Krakow, Mickiewicza 30, 30-059 Krakow, Poland
* Correspondence: stypinski@agh.edu.pl (M.S.); niemiec@agh.edu.pl (M.N.)

Abstract: Recent developments in quantum computing pose a significant threat to the asymmetric cryptography currently in use. Neural cryptography offers a potential alternative that is resistant to attacks of known quantum computer algorithms. The considered solution is lightweight and computationally efficient. If a quantum computer algorithm were successfully implemented, it could expose IoT sensors and smart grid components to a wide range of attack vectors. Given the lightweight nature of neural cryptography and the potential risks, neural cryptography could have potential applications in both IoT sensors and smart grid systems. This paper evaluates one of the suggested enhancements: the use of integer-valued input vectors that accelerate the synchronization of the Tree Parity Machine. This enhancement introduces a new parameter M that indicates the minimum and maximum values of input vector elements. This study evaluates the nonbinary version of the mutual learning algorithm in a simulated insecure environment. The results indicate that, while the Nonbinary Tree Parity Machine may involve some trade-offs between security and synchronization time, the speed improvement is more substantial than the decrease in security. The impact of this enhancement is particularly significant for smaller adjustments to parameter M.

Keywords: cybersecurity; key agreement; neural networks; mutual learning; smart grids

Citation: Stypiński, M.; Niemiec, M. Security of Neural Network-Based Key Agreement Protocol for Smart Grids. *Energies* **2023**, *16*, 3997. https://doi.org/10.3390/en16103997

Academic Editor: Abu-Siada Ahmed

Received: 19 April 2023
Revised: 3 May 2023
Accepted: 6 May 2023
Published: 9 May 2023

1. Introduction

Asymmetric cryptography and key agreement protocols are necessary for communicating parties over public networks to authenticate each other and perform secure data exchange. The well-known solutions capable of achieving this goal are the RSA algorithm and the Diffie–Hellman protocol. The security of both and their successors is guaranteed by the current inability to effectively solve certain number theory problems, such as prime factorization or discrete logarithm problems. The security of these two and similar algorithms was undisputed until the discovery of Shor's algorithm. This algorithm needs a quantum computer with a sufficient number of qubits. However, it is capable of successfully solving the fundamental problem of number theory: finding the prime factors of an integer. Therefore, modern communication requires new solutions to ensure data security [1].

Numerous research studies have focused on finding algorithms that are resistant to Shor's algorithm. One of the proposed solutions is mutual learning by two neural networks, called Tree Parity Machines (TPMs) [2,3]. TPMs performs key agreement by exchanging arbitrary input vectors and applying learning rules. Properly selected learning rules ensure that the mutual learning process finishes in a finite time [2]. Furthermore, mutual learning algorithms do not require computationally intensive numerical calculations, making them suitable for a wide range of applications, including smart grids. However, maintaining a proper level of security using this solution is still a challenge.

Smart grid systems are no exception here and would become prone to attacks providing successful implementation of Shor's algorithm. Additionally, power systems are being challenged to become more flexible due to factors such as energy distribution from renewable sources [4]. The vastness of interconnected sensors magnify the potential attack vectors

that could be exploited. Hence, power systems become more distributed, and this poses additional threats to the system as a whole. Authorization, authentication and accounting need to be provided by design to prevent data integrity attacks. Further, appropriate cybersecurity mechanisms prevent some physical layer attacks, particularly man-in-the-middle attacks, which require the attacker to access the communication medium. A key agreement protocol plays a crucial role in the cybersecurity system, ensuring that smart grids can resist a wide range of potential attacks. Another important branch in securing smart grid systems is immediate threat detection [5]. Neural cryptography could find its potential application in every component of smart grid systems, such as SCADA components, IoT sensors and all smart appliances [6].

A special case of TPM is Nonbinary Tree Parity Machine (NBTPM), which uses integer-valued input vectors instead of binary ones. This improvement results in reduced synchronization time of TPM and introduces the parameter M, which is responsible for the variability of input vectors and creates another dimension for finding the optimum TPM structure.

Given the difficulty of finding and verifying the security of TPMs, this article aims to evaluate the security of TPM and its nonbinary variant in various attack scenarios. Furthermore, it aims to offer guidance on determining the optimal structure of TPM for a given cryptosystem. The paper presents the results of a security analysis of NBTPM and is organized as follows. Section 2 reviews related literature. Section 3 introduces the TPM model, mutual learning and potential attacks. Section 4 details the research methodology, the results of simulations and the analysis of the findings. The last section concludes the paper.

2. Related Work

The Diffie–Hellman protocol was the first proposed algorithm for two parties to exchange a cryptographic key based on the difficulty of efficiently solving a mathematical problem, specifically the discrete logarithm problem [7]. Further, the algorithm was improved by utilizing elliptical curves [8]. However, both solutions are susceptible to being compromised by quantum computing.

As a response to the potential threat, a new family of post-quantum algorithms was proposed. This family consist of algorithms based on mathematical problems that are resistant to Shor's algorithm, such as supersingular elliptic curve isogeny, error correction codes and lattice-based and multivariate cryptosystems [9]. This paper focuses on another quantum-proof algorithm: TPM and its nonbinary variant. TPMs serve as an alternative to currently known secure key-exchange protocols. Currently, no known algorithms make TPM vulnerable to quantum computing. However, there is no mathematical proof of the security of TPM, and the computational power required to break the mutual learning is also unknown [10].

TPM has been the subject of much research [2,3,10–25]. Metzler et al. [2], Kinzel et al. [3] and Ruttor et al. [11] have shown that interacting neural networks can synchronize efficiently by using the mutual learning algorithm. In [10], the application of TPM in the field of cybersecurity was presented, wherein the synchronization of two neural networks allowed the performing of secure key agreement. The findings from these studies were consolidated in [12].

In [13–19], various improvements to TPMs were proposed. Researchers proposed an improved variant of the learning algorithm in scenarios where both parties are able to share a secret beforehand [13]. Furthermore, Santhanalakshmi et al. [14] and Sarkar et al. [15] proposed improvements based on a genetic algorithm application and whale optimization algorithms, respectively. In [16], the authors proposed a Gaussian distribution-based selection of initial weights that resulted in shorter synchronization time. Another set of improvements is based on altering the values an input vector can take. The first of them is Complex-Valued Tree Parity Machine (CVTPM), which proposes the usage of complex binary numbers during the learning process [17]. Vector-Valued Tree Parity

Machine (VVTPM) is the generalization of this idea and uses binary vectors as inputs [18]. The last improvement from this group is NBTPM, wherein the authors proposed the usage of integers instead of binary numbers [19]. NBTPM speeds up the synchronization of TPMs and introduces a new parameter M, which controls the variability of input vectors and adds an additional dimension for determining the most suitable TPM structure for users. Moreover, in [19], the authors discuss the effect of extrema of values and the reason for synchronization time reduction caused by the introduction of parameter M.

The application of TPM is presented in [20–22]. Sarkar et al. proposed the usage of TPMs in wireless networks and examined the energy consumption of the proposed solution [20]. The research conducted in [21] presents an error reconciliation protocol based on the mutual learning of neural networks in quantum cryptography. Gomez et al. presented hardware implementation of TPM [22].

Additionally, the security of TPMs has been studied extensively [23–25]. Ruttor et al. [23] and Shacham et al. [24] defined two different attacks and evaluated TPM security under them. The first attack is a standard man-in-the-middle genetic attack, and the second one is a geometric attack in which the attacker updates the weight of the neural network even when the outputs of benign parties do not match. Furthermore, in [25], the authors defined a probabilistic approach that could lead the evil party to gain some knowledge about a shared key.

Related research that involves the application of machine learning in quantum communication to achieve cryptographic keys is also presented in [26,27]. In [26], the authors define continuous-variable quantum key distribution and the framework for estimating the phase of the pilot signal. The framework is further expanded in [27]

3. Tree Parity Machine and Mutual Learning

Mutual learning in terms of TPMs is the key agreement process wherein two TPMs share information over a public medium and synchronize their weights. Once both neural networks achieve full synchronization, their weights can be used as a shared secret in further security operations. This section describes the details of the network's model, TPM learning and possible attack scenarios.

3.1. Model

TPM has a unique architecture. The network consists of two layers: an input layer and an output layer. Similar to the feed-forward network model, all of the input neurons are connected to exactly one output neuron. TPM inputs are grouped equally in number and are associated with only one hidden unit (σ_k). The outputs of all hidden units are connected to the output neuron (τ), which creates an output layer. The number of hidden units and the number of inputs per hidden unit are denoted by K and N, respectively. Every input has its associated weight (w_{kn}) that is bounded by range $< -L; L >$, where L is a positive integer. The learning rules guarantee that the values of the weights stay within the specified range. The activation function of every hidden unit is a modified signum function, and its formula is presented in (1). A and B denote the parties taking part in the mutual learning algorithm.

$$sgn'(x^{A/B}) = \begin{cases} 1, & x^A \geqslant 0 \vee x^B > 0 \\ -1, & x^A < 0 \vee x^B \leqslant 0 \end{cases} \tag{1}$$

Having defined the activation function, we are able to define the output of the k-th hidden unit σ_k, which is the sum of the products of weights and inputs. The formula is presented in (2).

$$\sigma_k = sgn'(\sum_{n=1}^{N} x_{kn} \cdot w_{kn}) \tag{2}$$

The final output τ of TPM is a product of all hidden layer outputs and is defined in (3).

$$\tau = \prod_{k=1}^{K} \sigma_k \tag{3}$$

An example TPM model is presented in Figure 1.

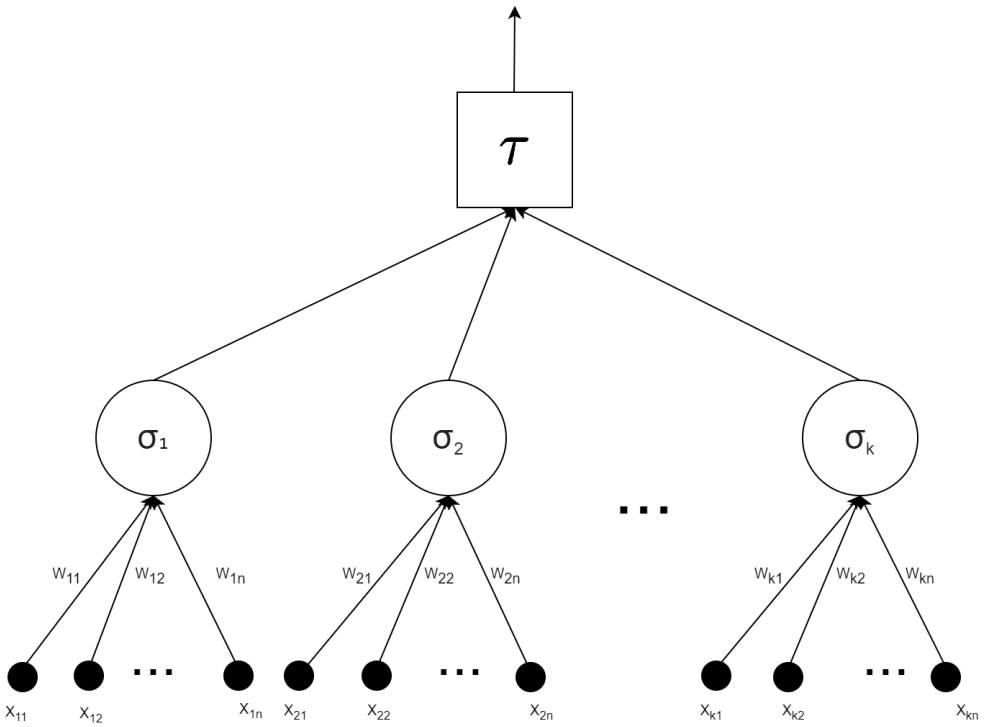

Figure 1. Structure of TPM model.

3.2. Synchronization and Learning Rules

Successful synchronization of TPM is a necessary part of key agreement protocol [28]. Both parties need to execute the following steps to distill the cryptographic key as a result of the whole process.

1. Both parties initialize their own TPM with randomly chosen weights. These neural networks must be the same variant of TPM and must share the same set of parameters uniquely defining its behavior, shape and size (i.e., K, L, M, N parameters).
2. Synchronizing parties randomly choose an input vector X consisting of $K \times N$ elements and share it via a public channel. Based on the selected TPM variant, the input vector consists either of binary values, complex binary values/vectors or integers.
3. Both participants calculate the output of their neural network and share it publicly.
4. The matching outputs are preliminary for updating the TPM weights. The learning rules are responsible for updating the weights, and users can choose one of the following:
   - Hebbian learning rule

$$w'_{kn} = w_{kn} + \tau x_{kn} \Theta(\sigma_k, \tau) \tag{4}$$

   - Anti-Hebbian learning rule

$$w'_{kn} = w_{kn} - \tau x_{kn} \Theta(\sigma_k, \tau) \tag{5}$$

- Random walk learning rule

$$w'_{kn} = w_{kn} + x_{kn}\Theta(\sigma_k, \tau) \tag{6}$$

where Θ is a function returning 1 if all of its arguments are equal and returns 0 otherwise, x_{kn}/w_{kn} denotes the k-th neuron n-th input/weight accordingly, and w'_{kn} denotes the updated weight value.

5. The parties repeat Steps 2–4 until full synchronization of their TPMs is achieved.

Once synchronization is complete, both neural networks are exactly the same. This results in the same weight vector that can be further applied in selected security operations, e.g., as a cryptographic key to encrypt confidential data using symmetric ciphers. All the notations used in the paper are presented in Appendix A in Table A1. The remaining part of this Section presents the training process of TPMs, which is later called synchronization, and attack vectors that might be used against cryptosystems using TPM.

3.3. Attack Scenarios

A man-in-the-middle attack is inherently the most efficient way of exploiting flaws in key agreement protocols. With regard to TPM, the described attack comes down to placing an eavesdropper between benign parties and learning an adversarial neural network based on captured communication. Three possible scenarios can happen when parties A and B are under a man-in-the-middle attack.

- $\tau_A \neq \tau_B$—neither benign TPMs nor the adversarial TPM proceed with synchronization.
- $\tau_A = \tau_B \neq \tau_C$—only TPMs A and B synchronize to each other, while the attacker cannot proceed.
- $\tau_A = \tau_B = \tau_C$—all the TPMs, including the adversarial one, update their weights accordingly.

The last of the described scenarios brings the evil TPM close to performing a successful attack. However, these conditions occur significantly less frequently than the others.

While neural cryptography is quantum-proof, there exist other attack vectors that pose a threat to the security of TPM. The prerequisite for all the attacks presented in this subsection is the sharing of input vectors and neural network outputs via an insecure public channel. This allows an evil party E to eavesdrop on the exchanged information. Possible attacks under these circumstances are as follows [25].

1. Man-in-the-middle attack—the simplest attack, wherein an evil party eavesdrops on the communication between benign parties. Based on this, entity E is able to synchronize the TPM. The synchronization only happens when $\tau_A = \tau_B = \tau_E$, where τ_A and τ_B denote the outputs of the benign parties' TPMs. The attacker may increase the odds of a successful attack by synchronizing more than one neural network.
2. Geometric/flipping attack—an attack that uses a geometric representation of inputs and a weight vector in N-dimensional space. This is an improved man-in-the-middle attack using an additional step that happens when $\tau_A = \tau_B$ but $\tau_A \neq \tau_E$. During this step, the attacker finds the i-th hidden unit with the lowest $\sum_{n=1}^{N} x_{in} \cdot w_{in}$. Once such a hidden unit is found, the evil party flips the output σ_i and applies the learning rule while pretending the outputs τ_A, τ_B, τ_E are equal.
3. Majority attack—an attacker uses a set of TPMs (I) and performs a geometric attack on all of them. However, just before the weight update, the voting process begins on the most common output vector $(\sigma_1^i, \sigma_2^i, \cdots, \sigma_K^i)$ that will be used in the weight update step. All the evil TPMs become more and more similar to each other as the attack progresses, which might be a flaw of the attack scenario. Using the majority attack and the flipping attack interchangeably might mitigate this issue.
4. Genetic attack—a genetic attack uses at most I TPMs to intercept key establishment. While the attacker has fewer than $I/2^{K-1}$, after every step, new 2^{K-1} are created with all possible $(\sigma_1^i, \sigma_2^i, \cdots, \sigma_K^i)$ hidden unit values. Once the limit $I/2^{K-1}$ has been

reached, only the fittest TPMs should remain. The algorithm's success is highly dependent on the fitness function, which determines which neural networks remain between generations.

4. Results

The following section presents the results of testing the security of TPM using non-binary input vectors. This section is split into two parts. The first presents the research methodology, and the second presents the collected results and their analysis.

4.1. Methodology

The conducted research consists of simulations that run attack scenarios on two benign TPMs. Benign TPMs perform the key agreement protocol described in Section 3.2. The attack scenario assumes that an attacker is able to eavesdrop on the communication. Based on this, we performed the attacks defined in Section 3.3. To mimic the conditions of an insecure channel, all the exchanged information between benign parties is also delivered to the malicious party responsible for the attack. Every scenario is repeated 1000 times. Many different sizes and parameters of TPMs are tested against man-in-the-middle attacks. The considered parameters are as following:

- $K = 3$,
- $L \in \{9, 10, 11, 12\}$,
- $M \in \{1, 2, 3, 4, 5\}$,
- $N = 50$.

By using these parameter selections, it is possible to evaluate how different values of parameter M affect various TPM forms (for $M = 1$ NBTPM becomes standard TPM defined in [2]) and to determine if parameter M improves the overall security. Furthermore, utilizing the selected parameters enables the attainment of a secure key length of approximately 500 bits [28].

The selected attacks against TPMs seem to be the most popular and successfully performed ones. Additionally, the attack vectors consist solely of man-in-the-middle attacks, as this is the main attack vector for key agreement protocols. For every attack, the eavesdropper is allowed to have a maximum of 50 neural networks. We consider the synchronization as failed when the eavesdropper is able to achieve a synchronization score equal to 1 for any of the evil TPMs. This means at least one neural network has perfectly synchronized its weights to either of the two TPMs performing mutual learning. The source code of the simulation framework is release to the public space (the source code of the simulation framework is available at https://github.com/mstypinski/tpm (accessed on 8 May 2023).)

During simulations, the number of total iterations, number of weight update steps, synchronization scores of benign TPMs and synchronization scores of the best-synchronized adversarial TPMs are gathered. The formula for synchronization score is defined in (7) [19]. Additionally, the index of synchronization when S_{score} of one evil TPMs is equal to 1 is also collected. In cases where the adversarial party does not fully synchronize any neural network, this value is equal to the total number of synchronization iterations in a particular scenario. All the results are presented along with 95% confidence intervals.

$$S_{score} = \frac{\sum_{k=1}^{K} \sum_{n=1}^{N} \Theta(w_{kn}, w_{kn}^{A})}{K \times N} \tag{7}$$

A genetic attack is a special case of man-in-the-middle attacks. It needs a fitness function that selects the proper individuals across whole populations for the next iteration of the genetic algorithm. The function used for this purpose is:

$$f = max(S_{score_A}, S_{score_B}). \tag{8}$$

In other words, this is a perfect fitness function since it selects the neural networks that have the most common weights with either of the two benign TPMs. In real-world scenarios, this function cannot be implemented because the attacker would need to know the exact weights of the legitimate neural networks. Moreover, the algorithm is allowed to select at most 50 neural networks that persist between iterations of attacks. This means that, internally, the population can be larger and can reach up to $50 \times 2^{K-1}$ individuals.

4.2. Results

Similar to the research conducted by the authors in [19], the benign TPMs synchronize in a faster manner by increasing parameter M. Figure 2 presents the number of required iteration steps and successful synchronizations for all considered neural networks. An iteration in the context of a mutual learning algorithm refers to a single cycle of the algorithm. Synchronization, on the other hand, refers to an iteration that results in an update of the weights for benign TPMs.

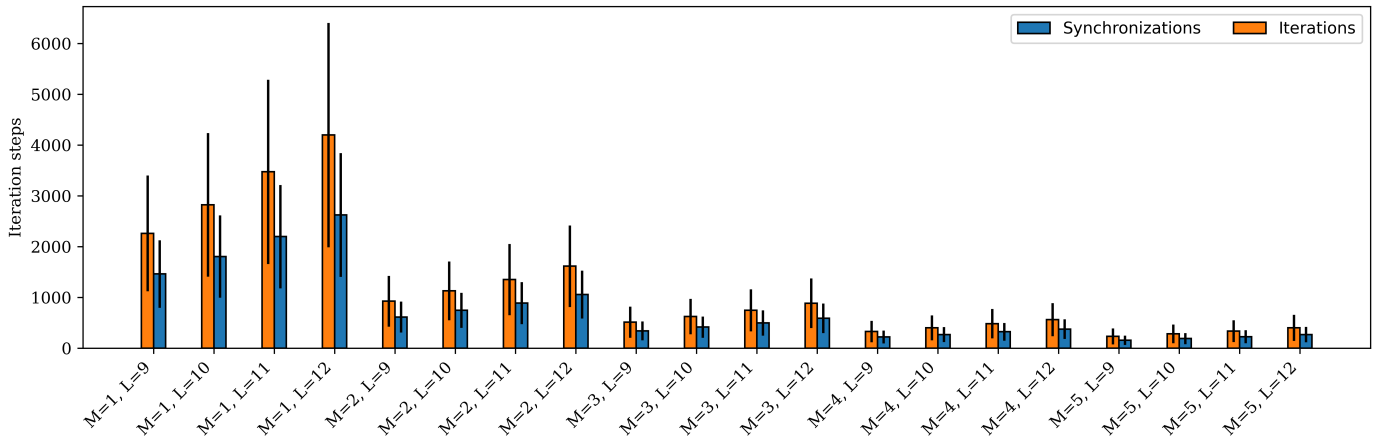

Figure 2. Number of key-exchange algorithm synchronizations and iterations of benign TPMs.

Increasing parameter M improves the security features of TPMs in terms of how quickly the parties are able to synchronize; hence, they can achieve a longer cryptographic key in the same amount of time. However, increasing parameter M also improves the eavesdropper's ability to retrieve the hidden state of the neural network. Figure 3 depicts the average index of iterations at which a malicious neural network executed the attack successfully, later called the adversarial synchronization point. The most notable is the average synchronization index difference between $M = 1$ and $M = 2$, which means the eavesdropper is able to synchronize more effectively with increases to parameter M. However, taking into consideration the decrease in the number of required iterations (Figure 2), the ratio between the median adversarial synchronization point and the median number of required synchronizations stays approximately the same.

Furthermore, it can be observed that the genetic attack performs significantly better than the other three attacks. This is due to the fact that the fitness function acts as an oracle-like determinant, selecting the TPMs most similar to the benign neural networks. To create a fitness function that is applicable in real-world scenarios, one may maximize or minimize the difference between the product of hidden outputs (σ_k) and weights and the input dot product ($w_k \cdot x_k$). The impact of different fitness functions on attack success probability requires further research. Additionally, while the attacker selects only the 50 most-fit TPMs that persist between generations, the generation may consist of more than 50 neural networks. This results in a meaningful advantage over the other attacks.

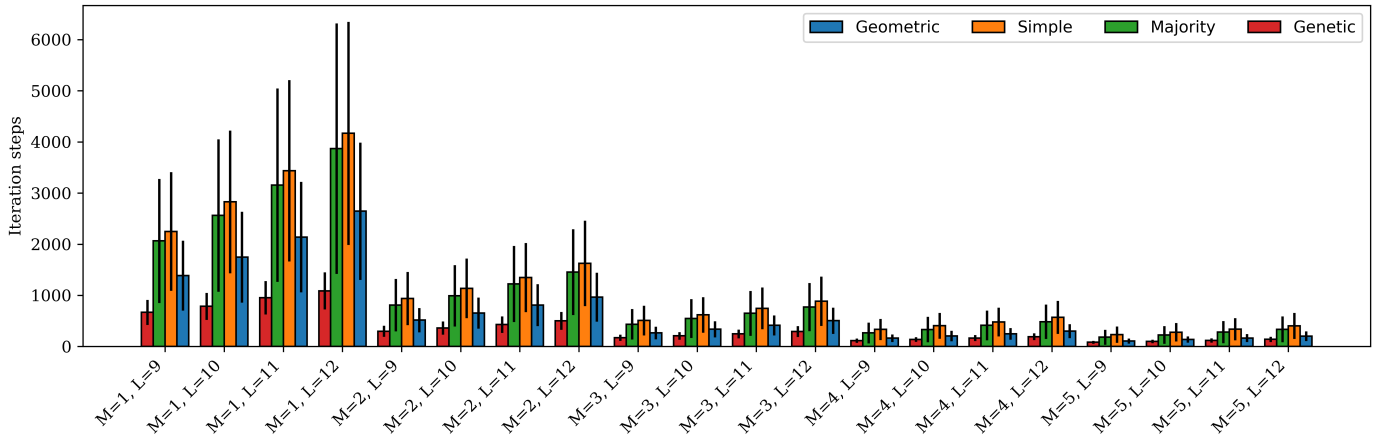

Figure 3. Average synchronization index at which evil party fully synchronizes its TPM.

The attacks against the examined TPM with the aforementioned set of parameters are effective. The least effective attack is the simple attack performed with many adversarial neural networks. Only when parameter $M = 5$ was the malicious party able to retrieve the cryptographic key. This result follows the outcomes from the initial research of NBTPM [19].

The majority attack had a success rate varying from almost 0.2 for $M = 1$, $L = 13$ up to 0.45 for $M = 5$, $L = 9$. The attack success probability rises with increases to parameter M; however, at the same time, the synchronization time decreases. The median of mutual learning iterations decreases more than twice for parameters $M = 1$ and $M = 2$, while the attack probability rises by at most 20.6%. Further, the attack success probability decreases with the increase to parameter L. This is true for all the attacks except the genetic attack. However, the greater values of parameter L result in more key-exchange algorithm iterations.

The geometric attack is successful with 0.8 probability in almost all scenarios. The conclusions from the majority attack are similar to those from the geometric attack. While the attack probability increases by at most 10%, the median number of required iterations decreases by at least 60% for parameters $M = 1$ and $M = 2$. Unfortunately, the success rate being at least 0.8 renders TPMs with the considered parameters ineffective in real-world scenarios. To avoid such situations, it is necessary to make more parameter adjustments, such as increasing the value of parameter L, which leads to a decrease in the median probability of successful geometric attacks. Unfortunately, these adjustments may result in an unwanted increase in synchronization time, which can be observed in the results.

The final attack evaluated is the genetic attack. This attack is successful in the majority of scenarios. However, it is important to note that this level of performance can only be achieved through the use of an ideal fitness function, which is not practical in real-world situations. The minimal increase in resistance to attacks observed for $M = 5$ is due to the greater standard deviation of the average number of mutual learning iterations. This implies that in certain scenarios, a benign TPM may synchronize at a significantly faster rate than an adversarial TPM.

The introduction of parameter M enhances the flexibility of TPMs in practical situations. A value of $M = 1$ corresponds to the standard version of a TPM. As M increases, the required number of iterations drops significantly, particularly for values of $M = 2$ and $M = 3$. The worst-case scenario for $M = 3$ occurs when $L = 11$. Despite a 35% increase in median attack success rate when compared to $M = 2$, there is a 45% reduction in median mutual learning algorithm iterations. The specific results for all simulated scenarios are presented in Table 1.

Table 1. Synchronization time and attack success probability.

		Mutual Learning Iterations				Attacks (S_{score})			
M	**L**	**Mean**	**Median**	**Max**	**Min**	**Simple**	**Majority**	**Geometric**	**Genetic**
1	9	2263.19 ± 1139.781	2182	5711	890	0.0	0.228	0.816	0.999
1	10	2823.69 ± 1413.413	2739	8478	1152	0.0	0.204	0.822	1.0
1	11	3475.25 ± 1814.419	3331	9087	1440	0.0	0.211	0.819	0.999
1	12	4198.83 ± 2209.911	4052	12608	1869	0.0	0.196	0.799	1.0
1	13	5064.37 ± 2820.405	4826	13067	1751	0.0	0.186	0.763	1.0
2	9	927.72 ± 500.816	893	2342	302	0.0	0.271	0.899	0.995
2	10	1131.29 ± 579.344	1098	2847	445	0.0	0.246	0.882	1.0
2	11	1353.59 ± 700.158	1306	3690	558	0.0	0.221	0.850	0.997
2	12	1617.92 ± 802.934	1568	4584	651	0.0	0.215	0.871	0.999
2	13	1908.82 ± 951.497	1839	4434	799	0.0	0.206	0.832	0.999
3	9	514.64 ± 308.012	489	1559	200	0.0	0.330	0.938	0.996
3	10	625.88 ± 348.143	598	1509	238	0.0	0.291	0.917	0.998
3	11	749.19 ± 413.828	714	1783	259	0.0	0.298	0.911	0.999
3	12	888.02 ± 487.737	853	2626	359	0.0	0.282	0.900	0.997
3	13	1031.67 ± 534.667	991	2616	435	0.0	0.235	0.892	0.999
4	9	331.62 ± 209.76	314	986	128	0.0	0.395	0.952	0.998
4	10	403.55 ± 245.221	383	1020	138	0.0	0.360	0.949	0.997
4	11	485.4 ± 289.757	461	1298	177	0.0	0.327	0.941	0.996
4	12	566.62 ± 325.635	544	1568	229	0.0	0.316	0.926	0.998
4	13	661.08 ± 367.354	631	1707	270	0.0	0.322	0.915	0.998
5	9	235.25 ± 158.482	219	695	85	0.003	0.452	0.963	0.998
5	10	285.76 ± 184.754	267	830	96	0.001	0.410	0.966	0.995
5	11	339.3 ± 214.747	320	1043	121	0.001	0.364	0.961	0.997
5	12	403.74 ± 254.932	379	1099	151	0.001	0.348	0.948	0.997
5	13	459.22 ± 272.12	438	1449	171	0.0	0.338	0.938	0.997

5. Conclusions

Quantum-proof cryptography is a growing problem in modern communications. The recent advancements in efficient factorization using quantum algorithms have highlighted the need for secure key agreement protocols. TPMs such as NBTPM have the potential to meet this need; however, further testing and optimization of parameters are necessary to apply it to production-ready smart grid systems.

In this paper, the authors investigated the security of NBTPM under different types of man-in-the-middle attacks: simple, geometric/flipping, majority and genetic. While the TPMs we considered did not attain perfect security, the parameter M provides another dimension for optimizing the TPM architecture for increased security. In particular, a significant improvement might be noticed by increasing parameter M from a value of 1 to 2 and from 2 to 3. Accordingly, while the median attack success rate increases by at most 21% and 35%, respectively, the median number of mutual learning iterations is reduced by at least 58% and 45%, respectively; hence, the conclusion can be drawn that increasing parameter M along with fine-tuning other parameters improves the overall security of the considered solution. Choosing a specific value for parameter M must be done by considering all the pros and cons that follow specific selection. Further, while the increase to M improves efficiency, it cannot be increased infinitely, as security features degrade for large M values that approach the L parameter value. It is worth mentioning that some applications of TPMs require even fewer interactions than assumed (e.g., TPMs used for error correction in quantum cryptography).

In order to implement the proposed solution in a real-world scenario, further parameter adjustment is necessary. It is important to approach the parameter M with caution, as it improves synchronization time but can also enhance the efficiency of adversarial synchronization. The number of inputs per neuron and the number of hidden neurons in

the intermediate layer of TPMs should also be adjusted carefully. Another approach to applying the described solution successfully in a smart grid could be an implementation of authenticated TPMs. Further research in this field should focus on exploring a wider range of parameters, including the number of hidden neurons in the intermediate layer, in order to find more secure variants of TPMs.

Appendix A

Table A1 presents the list of all notations used in the paper and their descriptions.

Table A1. Table of notations.

Variable	Description
K	TPM parameter denoting the number of neurons in a hidden layer
L	TPM parameter denoting the maximum value the weight modulus can take
M	NBTPM parameter denoting the maximum value the input vector element modulus can take
N	TPM parameter denoting the number of inputs per neuron in a hidden layer
x_{kn}	Input value of n-th input of k-th neuron
τ	Output of TPM
σ_k	Output of k-th hidden neuron
Θ	Function returning 1 if all its arguments are equal and 0 otherwise
w_{kn}	Weight value corresponding to n-th input of k-th neuron
w'_{kn}	Next iteration value of w_{kn}
S_{score}	Synchronization score

Author Contributions: Conceptualization, M.S. and M.N.; methodology, M.S.; software, M.S.; validation, M.S.; formal analysis, M.S.; investigation, M.S. and M.N.; writing—original draft preparation, M.S.; writing—review and editing, M.S. and M.N.; visualization, M.S.; supervision, M.N.; project administration, M.N.; funding acquisition, M.N. All authors have read and agreed to the published version of the manuscript.

Funding: Research project partly supported by the program "Excellence initiative—research university" for the AGH University of Krakow.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study—the complete results collected during the simulations—are available online at: https://github.com/mstypinski/tpm_attack_results (accessed on 5 May 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Niemiec, M.; Dziech, A.; Stypiński, M.; Derkacz, J. Quantum-Based Solutions for the Next-Generation Internet. *Inf. Secur. Int. J.* **2019**, *43*, 62–72. [CrossRef]
2. Metzler, R.; Kinzel, W.; Kanter, I. Interacting neural networks. *Phys. Rev. E Stat. Phys. Plasmas Fluids Relat. Interdiscip Top.* **2000**, *62*, 2555–2565. [CrossRef]
3. Kinzel, W.; Metzler, R.; Kanter, I. Dynamics of interacting neural networks. *J. Phys. A Math. Gen.* **2000**, *33*, L141. [CrossRef]
4. Ghiasi, M.; Niknam, T.; Wang, Z.; Mehrandezh, M.; Dehghani, M.; Ghadimi, N. A comprehensive review of cyber-attacks and defense mechanisms for improving security in smart grid energy systems: Past, present and future. *Electr. Power Syst. Res.* **2023**, *215*, 108975. .: 10.1016/j.epsr.2022.108975. [CrossRef]
5. Ghiasi, M.; Dehghani, M.; Niknam, T.; Kavousi-Fard, A.; Siano, P.; Alhelou, H.H. Cyber-Attack Detection and Cyber-Security Enhancement in Smart DC-Microgrid Based on Blockchain Technology and Hilbert Huang Transform. *IEEE Access* **2021**, *9*, 29429–29440. [CrossRef]
6. Mohan, A.M.; Meskin, N.; Mehrjerdi, H. A Comprehensive Review of the Cyber-Attacks and Cyber-Security on Load Frequency Control of Power Systems. *Energies* **2020**, *13*, 3860. [CrossRef]
7. Diffie, W.; Hellman, M. New directions in cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]

8. Barker, E.; Chen, L.; Roginsky, A.; Vassilev, A.; Davis, R. *Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018. [CrossRef]
9. Fernández-Caramés, T.M. From Pre-Quantum to Post-Quantum IoT Security: A Survey on Quantum-Resistant Cryptosystems for the Internet of Things. *IEEE Internet Things J.* **2020**, *7*, 6457–6480. [CrossRef]
10. Kanter, I.; Kinzel, W.; Kanter, E. Secure exchange of information by synchronization of neural networks. *Europhys. Lett.* **2002**, *57*, 141. [CrossRef]
11. Ruttor, A.; Reents, G.; Kinzel, W. Synchronization of random walks with reflecting boundaries. *J. Phys. A Math. Gen.* **2004**, *37*, 8609. [CrossRef]
12. Ruttor, A. Neural Synchronization and Cryptography. Ph.D. Thesis, University of Würzburg, Würzburg, Germany, 2007,
13. Allam, A.M.; Abbas, H.M.; El-Kharashi, M.W. Authenticated key exchange protocol using neural cryptography with secret boundaries. In Proceedings of the The 2013 International Joint Conference on Neural Networks (IJCNN), Dallas, TX, USA, 4–9 August 2013; pp. 1–8. [CrossRef]
14. Santhanalakshmi, S.; K., S.; Patra, G.K. Analysis of Neural Synchronization Using Genetic Approach for Secure Key Generation. In *Proceedings of the Security in Computing and Communications*; Abawajy, J.H., Mukherjea, S., Thampi, S.M., Ruiz-Martínez, A., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 207–216.
15. Sarkar, A.; Khan, M.Z.; Singh, M.M.; Noorwali, A.; Chakraborty, C.; Pani, S.K. Artificial Neural Synchronization Using Nature Inspired Whale Optimization. *IEEE Access* **2021**, *9*, 16435–16447. [CrossRef]
16. Dolecki, M.; Kozera, R. The Impact of the TPM Weights Distribution on Network Synchronization Time. In *Proceedings of the Computer Information Systems and Industrial Management*; Saeed, K., Homenda, W., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 451–460.
17. Dong, T.; Huang, T. Neural Cryptography Based on Complex-Valued Neural Network. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4999–5004. [CrossRef] [PubMed]
18. Jeong, S.; Park, C.; Hong, D.; Seo, C.; Jho, N. Neural Cryptography Based on Generalized Tree Parity Machine for Real-Life Systems. *Secur. Commun. Netw.* **2021**, *2021*, 6680782. [CrossRef]
19. Stypiński, M.; Niemiec, M. Synchronization of Tree Parity Machines Using Nonbinary Input Vectors. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–7. [CrossRef] [PubMed]
20. Sarkar, A.; Dey, J.; Bhowmik, A. Multilayer neural network synchronized secured session key based encryption in wireless communication. *Indones. J. Electr. Eng. Comput. Sci.* **2019**, *14*, 169. [CrossRef]
21. Niemiec, M. Error correction in quantum cryptography based on artificial neural networks. *Quantum Inf. Process.* **2019**, *18*, 174 . [CrossRef]
22. Gomez, H.; Óscar, R.; Roa, E. A 65nm CMOS key establishment core based on tree parity machines. *Integration* **2017**, *58*, 430–437. .: 10.1016/j.vlsi.2017.01.010. [CrossRef]
23. Ruttor, A.; Kinzel, W.; Naeh, R.; Kanter, I. Genetic attack on neural cryptography. *Phys. Rev. E* **2006**, *73*, 036121. [CrossRef]
24. Shacham, L.N.; Klein, E.; Mislovaty, R.; Kanter, I.; Kinzel, W. Cooperating attackers in neural cryptography. *Phys. Rev. E* **2004**, *69*, 066137. [CrossRef]
25. Klimov, A.; Mityagin, A.; Shamir, A. Analysis of Neural Cryptography. In *Proceedings of the Advances in Cryptology—ASIACRYPT 2002*; Zheng, Y., Ed.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 288–298.
26. Chin, H.M.; Jain, N.; Zibar, D.; Andersen, U.L.; Gehring, T. Machine learning aided carrier recovery in continuous-variable quantum key distribution. *NPJ Quantum Inf.* **2021**, *7*, 20. [CrossRef]
27. Liu, Z.P.; Zhou, M.G.; Liu, W.B.; Li, C.L.; Gu, J.; Yin, H.L.; Chen, Z.B. Automated machine learning for secure key rate in discrete-modulated continuous-variable quantum key distribution. *Opt. Express* **2022**, *30*, 15024–15036. [CrossRef] [PubMed]
28. Stypiński, M.; Niemiec, M. Impact of Nonbinary Input Vectors on Security of Tree Parity Machine. In *Proceedings of the Multimedia Communications, Services and Security*; Dziech, A., Mees, W., Niemiec, M., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 94–103.