Date of publication SEP-30, 2025, date of current version AUG-08, 2025.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.24.3.4193

Neural Cryptography Based on Tree Parity Machine to Generate OTP

VERA WATI[1], NUR FITRIANINGSIH HASAN[2], ACHMAD NUGRAHANTORO[3], NISRINA YULIA SETYAWATI[4]

[1] Sistem Informasi Kota Cerdas, Politeknik Negeri Indramayu, Indramayu, Indonesian
[1,4] Sistem Informasi Kota Cerdas, Universitas Tunas Pembangunan Surakarta, Surakarta, Indonesian
[2] Ilmu Komputer, Universitas Muhammadiyah Papua, Jayapura, Indonesian
[3] Bisnis Digital, Universitas Madani Yogyakarta, Yogyakarta, Indonesian
[3] PT Egref Telematika Nusantara, Indonesian

Corresponding author: Vera Wati (e-mail: vera.w@polindra.ac.id).

ABSTRACT: The rapid development of cloud computing increases cybersecurity risks, including hacktivism, phishing, fraud, and OTP theft. In addition to user education, further security technologies are required, such as one-time authentication at the main gateway or as an extra layer within the application system. OTP, generated through cryptographic techniques, is an effective security method because it can only be used once and does not require additional device installation. If implemented correctly, OTP provides a high level of confidentiality. Artificial Neural Networks (ANNs) are an innovation in neural cryptography. TPM ANNs, which apply synchronized learning to parity machines, can learn independently based on input, hidden, and output parameters. This study proposes the implementation of OTP ANN with TPM to improve system security. The integration of OTP with TPM on the login menu using a web server aims to generate more random keys with ideal parameters $K \geq 4$, $N \geq 5$, and $L \geq 5$. As a result, real-time OTPs can be sent via Telegram and Email, offering a more secure and efficient encryption solution in real-world applications. Compared to deterministic OTP approaches that rely on hashes of fixed time values and parameters, TPM-based stochastic approaches offer advantages in terms of entropy and cryptographic uncertainty. Deterministic OTPs are time-efficient, but are vulnerable to prediction if the seed is not accompanied by an additional secret. In contrast, TPM-based stochastic OTPs are more resistant to predictive attacks due to their complex synchronization properties and independence from system time, making them more suitable for high-risk authentication scenarios.

KEYWORDS Cryptography, Neural Cryptography, One Time Pad, Neural Network, Tree Parity Machine.

I. INTRODUCTION

The rapid advancement of cloud computing technology in recent times has compelled companies to enhance their capabilities in the field of information technology to be more scalable and accessible [1]. In this regard, the cybersecurity risks are increasing with unauthorized interventions, such as the extraction of secrets and hacktivist actions that can compromise a company's vital data [2]. Hacking techniques like phishing, scamming, and other forms of attack can lead to the occurrence of One-Time Password (OTP) theft. OTP theft is an illegal act aimed at gaining unauthorized access to the victim's accounts and information [3]. Manipulative actions by OTP theft perpetrators often involve social engineering tactics to trick victims into revealing information like OTP codes [4]. Therefore, these attacks can pose a serious threat to cybersecurity in safeguarding user data privacy. In addition to preventing social engineering through user education to promote a strong information security culture, the security of the technology itself should also be considered.

Efforts to thwart security breaches within cloud services involve the use of cryptographic techniques [5]. OTP is known as a password that is often used for one-time authentication. Another characteristic of OTP is that it consists of several unique digits or characters [6, 7]. OTP doesn't require the installation of any software and is relatively easy to implement anywhere and at any time, as this token model is typically embedded in personal devices such as social media, digital wallets, and similar applications. It is one of the most efficient data security systems for safeguarding against hacktivism [8]. OTP is considered part of the symmetric cryptography algorithms [9]. OTP provides an additional layer of authentication to web servers, protecting against fraudulent

attempts and ensuring that information is secure between the user and the recipient [10]. Research on neural network-based artificial intelligence authentication is becoming an innovative topic [11]. Modeling with neural cryptography and bit-based mutual learning synchronization shows promising new phenomena.

Neural cryptography is a branch of cryptography that employs artificial neural network (ANN) algorithms for the encryption process. The characteristics of ANN itself include strong computing capabilities and the ability to explore problems through self-learning as well as mutual synchronization among neurons. Furthermore, the stochastic behavior of ANN, characterized by random probability distributions, encourages the generation of the required solution possibilities [12]. The concept of synchronization in ANN introduces new opportunities in the field of cryptography for generating keys [13]. Typically, in many modern cryptography methods, algebraic number theory is involved in data security, identity authentication, and message encryption to safeguard the content of information [14, 15].

With the advent of digital technology, the role of cryptography has become a discipline that can transform plain text into text that is difficult to decipher [13]. Therefore, the presence of a key plays a crucial role in the transformation of this text [16]. The concept of cryptography with ANN, known as neural cryptography, utilizes a secret key for mutual synchronization between two communicating parties through tree parity machines to generate encryption [16-18]. The exchange of secret keys using the Diffie-Hellman protocol has become a practical method for implementation in the field of cryptography, particularly in the synchronization of tree parity machines. The keys generated can be further used as symmetric cipher keys. Neural parity machines offer advantages such as time complexity, low memory usage, and non-deterministic characteristics, making them resistant to patterns that could be exploited by cryptanalysis.

A Tree Parity Machine (TPM) is a specific type of multilayer feedforward neural network [19]. This type of network consists of one output neuron and input neurons that are directly fed into the output through a weighting process [9]. Feedforward artificial neural networks (ANNs) are the simplest type of neural networks and are relatively easy to design [19]. The performance adopted for the parity machine results in an encryption key by processing the parameter values of neurons that move in one direction. The TPM procedure begins with two communicators who have the same network topology, with bipolar data representation having two values: 1 and -1 [17].

The feasibility of utilizing neural cryptography with the synchronization process using TPM has been explored by several researchers. For example, a study conducted by [20][21] demonstrated that the synchronization of two TPMs can be achieved through a collaborative learning rule. Specifically, it has been shown that TPM synchronization can be used as a cryptographic key exchange protocol [22]. In terms of efficiency and security, it has been demonstrated that the synchronization time of TPM is in the same order as that of the basic TPM model, and it can be more secure than traditional generation methods with the same synaptic depth and architecture [13]. The research results conducted by [23] The research conducted by the author, who developed neural cryptography with multiple hidden layers using TPM elements, provides evidence that the number of weight mutations in the neuron layers increases exponentially, resulting in almost no identical keys. This was proven through a simulation involving 10,000 attacker machines attempting to mimic the key but failing. In addition to its high-security level and complex structure, this algorithm also boasts great efficiency, with execution times of less than 1 second observed on an Intel® Core processor in the experiments.

As the research findings by the author indicate, protecting the OTP keys does not eliminate the risk if the channel or cloud storage is compromised. Other research points to theoretical limitations of OTP, such as the requirement for keys that are as long as the message and the challenge of securely distributing these keys. Technologies like Quantum Key Distribution (QKD) offer promising solutions, but they remain expensive and difficult to implement at scale. This highlights the need for a new approach that can bridge the gap between theoretical security and practical deployment [24].

Studies such as [25] and [26] demonstrated that while the application of TOTP/HOTP algorithms within moving target defense strategies in sensitive networks improved security, the reliance on deterministic hash functions (e.g., HMAC-SHA) and static keys makes these systems susceptible to compromise if the key is exposed or if time synchronization fails. Similarly, integrates TOTP with biometrics for electronic payments, yet the OTP generation remains deterministic and dependent on a shared seed [27]. If this seed is leaked, it opens the system to brute-force or spoofing attacks.

To address these gaps, this research proposes an OTP-ANN method that utilizes the Tree Parity Machine (TPM) algorithm to generate random keys through synchronization without explicit exchange. Rather than replacing modern RNGs, this approach introduces an added layer of security through stochastic key synchronization, which is inherently unpredictable to third parties, even under full communication observation. This represents a promising solution to the unresolved limitations of conventional OTP approaches identified in prior studies.

Therefore, in this research, the proposed method using OTP-ANN and the TPM algorithm is implemented to generate random keys for enhanced security. The implementation of OTP-ANN will be applied to physical devices in a real-world context, rather than just modeling, utilizing the login authentication process on websites that are integrated with notifications through Telegram and Email. The choice of OTP functionality for login is an ideal option because, in the context of a website, login is a common necessity as a security layer to reduce the risk of hacking [28]. As a result, the implementation of OTP-ANN with TPM can be applied in real-life situations.

II. MATERIAL AND METHODS

In the proposed method, as shown in Figure 1, the process starts with initiating the login process by entering the username and password on the website, using CAPTCHA for verification, and processing for OTP integrated with Telegram and Email for authentication.

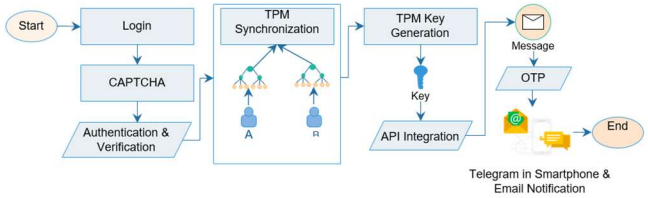

Figure 1. Proposed System Stage

The CAPTCHA process is not a secret because it is used only for verifying actions performed by humans rather than automated entities. Its generation involves processing the 26 letters of the alphabet to display 6 random letters on the login page. After successful verification, the OTP authentication process is initiated. It begins with the synchronization of TPM between both neural machines to generate a secret key. Based on this processing, the key is sent as a message through the Telegram and Email interfaces to appear in the message interface. This continues until successful notifications appear on both interfaces.

A. LOGIN PAGE

Referring to Figure 2. In the verification process by logging in, the user enters the username and password which must match the data stored in the database. Public CAPTCHA must be entered in the form provided, then click the Login option. If verification is successful, then the OTP authentication process with TPM is as shown in Figure 3. On the website page you will be waiting for the process of entering the 6 OTP codes that have been generated The authentication OTP code will send a notification to Telegram with a display as shown in Figure 4 and the registered email looks like in **Помилка! Джерело посилання не знайдено.**.

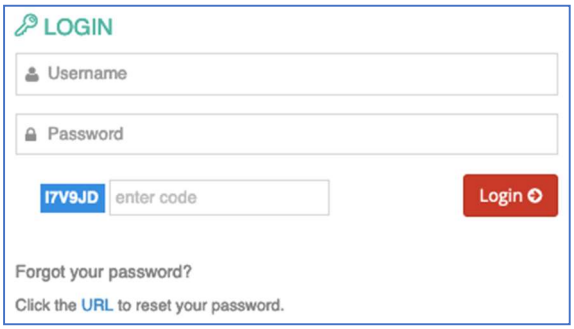

Figure 2. CAPTCHA Verification Login Page

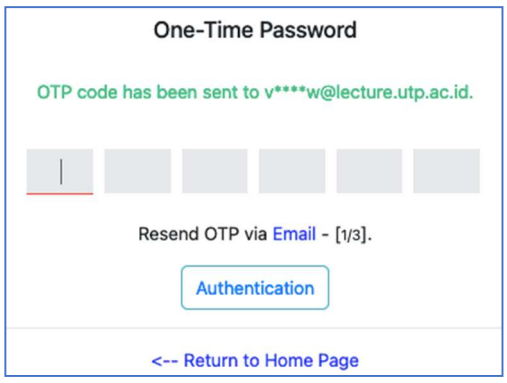

Figure 3. OTP TPM Authentication Email and Telegram Integration

OTP authentication in Figure 4 and Figure 5. Has a period of 30 seconds, and automatically sends a random code to Email and Telegram addresses. The TPM mutual learning synchronization process uses ideal parameters with a value of K=4 as the hidden neuron, N=5 as the input neuron, and a weight of 5 as L. These parameters are considered to be the minimum parameters that are most suitable in terms of functionality and can achieve the best results in a time short.

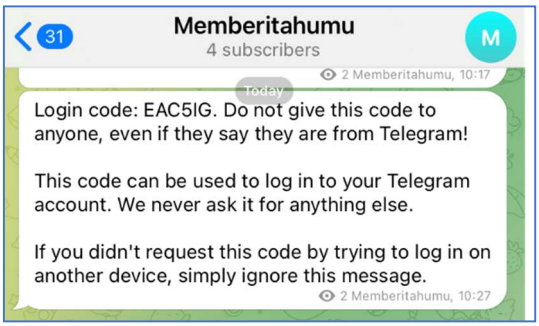

Figure 4. TPM OTP notification on Telegram

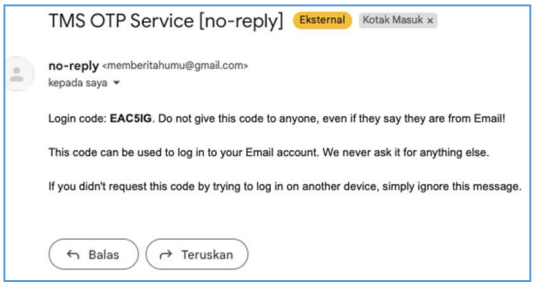

Figure 5. TPM OTP Notification in Email

B. TPM MUTUAL LEARNING SYNCHRONIZATION

TPM synchronization uses the principle of symmetric key cryptography; however, in this case, the TPM generates keys that are distributed along with the ciphertext. Mutual learning on both machines, as depicted in Figure 6, consists of K as hidden neurons, N input neurons, and one output neuron. Consideration of 2 parties A and B agreeing on a key via a secret channel, such that the basic performance of the ANN is identical to performing mutual learning for synchronization. The process of generating the secret key involves input neurons $X_{K,N}$, X_1, X_2, X_3, ... X_n, which enters the nerve cell layer by collecting weight values $W_{K,N}$, W_1, W_2, W_3, ... W_n, at each node. The weight initialization is completely random with a value range of -L to L so that the cryptanalyst does not recognize patterns to reveal the contents of the ciphertext. Mathematical function $y(x)$ (1) to ensure that the weight value formed is always within the L value range.

$$y(x) = \begin{cases} L \; if \; x > L \\ -L \; if \; x < -L \end{cases} \quad (1)$$

The weight value for the formation of a new node is calculated based on the equation of the product of each weight adjusting the activation function, with mathematical equation (2)

$$\sum_{i=1}^{n} X_i W_i - \theta \quad (2)$$

The neuron's input value is multiplied by X_i with weight value W_i in creating new nodes with the process of generating an internal Threshold (threshold value) of the activation function Σ as the next process for the output value. The Threshold function has provisions as in equation (3). The Threshold function used is the bipolar Threshold function which produces a value of 1 or – 1.

$$f(x) = \begin{cases} 1 \; if \; x \geq 0 \\ -1 \; if \; x < 0 \end{cases} \quad (3)$$

It is important to emphasize that although the synchronization process in a Tree Parity Machine (TPM) neural network involves repeated exchanges of output bits, it does not expose internal weight values or the resulting shared secret key. Unlike traditional key exchange protocols, which typically transmit partial or transformed representations of key material, TPM synchronization only exchanges the sign of aggregated weighted inputs, which does not reveal any meaningful information about the internal state of the system. Even if a passive adversary is capable of capturing the entire communication between two legitimate parties (e.g., neural machines A and B), the communication remains stochastic due to the TPM's weight update rules. When Hebbian or anti-Hebbian learning mechanisms are applied, synchronization becomes virtually unattainable for the attacker. Legitimate parties benefit from mutual feedback, which enables coordinated weight updates, whereas the attacker can only observe unidirectional communication without the ability to influence or respond to the synchronization process.

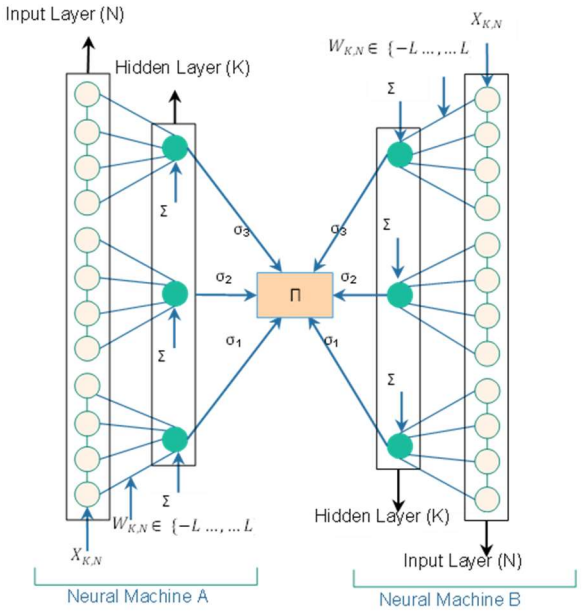

Figure 6. Mutual Learning Tree Parity Machine

Empirical studies have demonstrated that the average synchronization time for an attacker increases exponentially with the size of the TPM, while for legitimate participants, the synchronization time increases linearly with network parameters such as the number of hidden units (K), the number of input neurons per hidden unit (N), and synaptic depth (L) [29]. This asymmetry creates a computational gap that can be exploited for secure key generation. Increasing the TPM's complexity by selecting larger values for K, N, and L not only enhances the entropy of the generated key but also significantly increases the difficulty of synchronization for any eavesdropper. This asymmetry has been validated through simulations, which show that for sufficiently large TPM, the probability of successful synchronization by an attacker within practical time frames becomes negligible [30].

Importantly, although the synchronization process involves repeated public exchanges of output bits, these bits do not leak internal weight values or the final secret key. Since the attacker lacks feedback from either party and can only passively observe the exchanges, their neural network is unable to reliably align its weights. The attacker's learning is unidirectional, unlike the bidirectional feedback loop enjoyed by the legitimate parties, making convergence virtually impossible within a realistic timeframe, as confirmed by previous studies [31]. Therefore, the TPM-based mutual learning protocol offers a key advantage: the ability to establish a shared secret key over a public channel without any direct key exchange, while maintaining strong security even under full passive interception.

The weight updating process will continue to repeat until it reaches the same weight, starting with the first iteration by comparing the two neural values of the same $\sigma_K^{A/B} = \tau^{A/B}$, where neural machine A is the first party, for example$(\tau^A = \Pi_{k=1}) = (\tau^B = \Pi_{k=1})$. The length of time for the synchronization process is certainly influenced by the input values, weight values, and the network structure formed. So the larger the TPM network, the longer the synchronization time.

C. GENERATE TPM KEY FOR OTP

As illustrated in Figure 7, TPM network topologies will use the same structure. The key-generating process to produce the OTP key value involves the following stages:

1) Neural machines A and B have the same input value parameters, namely 2 input neurons (N=2), hidden neurons have a value of 3 (K=3), and a weight limit of 4 (L=4).
2) Random weight update process $w_{xy}^{A/B}$ where its valuable $\in \{-L, \dots L\}$, then it will form $\in$ {-4,-3,-2,-1,0,1,2,3,4}, taking into account the value formed x, y, with range $(1 \leq x \leq K)$ and $(1 \leq y \leq N)$ so that based on the K, N, L values, each weight for both parties forms a matrix $K * N$ or $3 * 2$ so that the equation is formed (4)

$$W^A(x) = \begin{bmatrix} 3 & 2 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \text{ sync with } W^B(y) = \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} \quad (4)$$

3) Update weight in neural is done if while $w_{xy}^A \neq w_{xy}^B$, i++ starting with the first iteration, **do**
   a) Generate input vector $Z_{xy} \in \{-1,1\}$
   b) Calculating hidden neuron units with mathematical functions (5)

$$\sigma_x^{A/B} = sgn(\sum_{x=y}^{N} w_{xy}^{A/B} Z_{xy} \quad (5)$$

This function returns $-1, 0$ or 1

$$sgn(z) \begin{cases} -1 \; if \; x < 0, \\ 0 \; if \; x = 0, \\ 1 \; if \; x > 0. \end{cases} \quad (6)$$

If the hidden neuron value is 0 then it is mapped to -1 for binary output consistency, this process is related to the process of calculating the next stage. Suppose the resulting iteration is as in matrix (7)

$$Z^{A/B} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \quad (7)$$

c) Calculating output neurons based on hidden neurons, equation (5) is used. to compare the values of the two neural machines such as by calculating each weight (8)

$$\begin{aligned}
&\sigma_1^A = \sum_3^2 Z^A W^A = (-1*3)+(1*2) = -1\\
&\sigma_2^A = \sum_1^2 Z^A W^A = (1*1)+(-1*1) = 0\\
&\sigma_3^A = \sum_2^2 Z^A W^A = (1*2)+(1*2) = 4\\
&\dots\\
&\dots\\
&\sigma_1^B = \sum_2^2 Z^A W^A = (-1*2)+(1*1) = -1\\
&\text{Etc.}
\end{aligned} \tag{8}$$

The resulting weights will undergo continuous iteration by paying attention to the return of the threshold function in equation (9)

$$\begin{aligned}
&\textbf{if } \tau^A \neq \tau^B \textbf{ then}\\
&\textbf{goto (a) } \sigma_1^A = -1, \sigma_2^A = 0, \sigma_3^A = 1, \dots, \dots \sigma_1^B = -1, \dots\\
&\textbf{else}\\
&\text{if } \tau^A = \tau^B \textbf{ then}\\
&\qquad \tau^A = \prod_{K=1}^{1} \sigma_1^A = \sigma_1^B
\end{aligned} \tag{9}$$

As an iterative process to produce the same value, use the synchronization learning rule with the Hebbian learning rule

$$w_{ij}^{A/B,++} = g(w_{xy}^{A/B} + \sigma_x^{A/B} x_{xy}\ \theta\ (\sigma_x^{A/B}, \tau^{A/B}),\ (\tau^A, \tau^B)) \tag{10}$$

End

The iteration process will be carried out optimally until both nerves are the same $w_{xy}^{A=B}$ and generate a key for the next OTP process. The depiction in Figure 6 with predetermined K, N, L values gives the probability $(2L+1)^{KN}$ generates a possible key of 3*26*** for alphabetically so that the resulting key has a small chance of being attacked by computer power.

4) OTP key generation stage for neural cryptography.
   a) Secret key value i $((w_{xy}^A = w_{xy}^B) + 4/1024.Kib)$

```
if key length >= 6
      exp date + 30 seconds
i++
End
```

   b) The generated key will be sent a notification via Telegram and Email, with several data considered such as:

```
INSERT INTO (kode OTP, generate exp
date, iterasi max, nilai kunci i, key
length, key, nilai K, N, L, status)
```

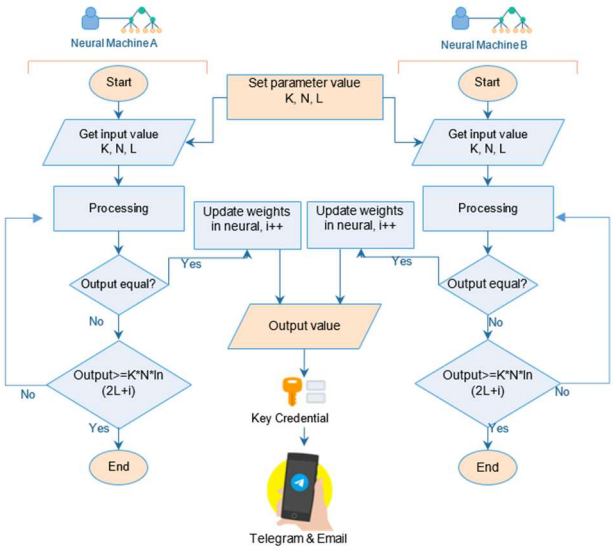

Figure 7.TPM Network for OTP Key Generator

D. OTP NOTIFICATION ON TELEGRAM AND EMAIL

In the process of creating an OTP notification in an email, you need to send a code and set the validity period for using the OTP. The first setting requires specifying the sender address and subject of the email. Variables used in email settings include the SMTP protocol on the host, port and the importance of initiating the sender's email and password. In the recipient's email settings, the message that will be sent is the text and OTP that has been successfully generated. The pseucode for using OTP for email is as follows:

```
start
text_email_otp(text, otp, exp)

initialize variables:
protocol= 'SMTP',
smtp_host= 'ssl://smtp.googlemail.com',
smtp_port= 465,
smtp_user= sender@gmail.com,
smtp_password= sender_password,
receiver= 'receiver@gmail.com'
subject= 'OTP-ANN TPM Service'
text= text_email_otp,
mailtype= 'html',
charset = 'iso-8859-1'
end.
```

Apart from utilizing email notifications, the research used the Telegram application which is already installed on mobile devices, even though it is still in the form of a bot. The main requirement of course requires the Telegram application to be installed on the device. Commands are required to create a new bot by following the instructions from the official bot provided by Telegram, namely BotFather to get the API. API as a bridge for distributing information between OTPs generated from neural cryptography and installed Telegram devices, both on mobile phones and on the Telegram web. The message will generate an OTP notification notification with a time limit by working on the following pseudocode:

```
start
text_telegram_otp(text, otp, exp)
```

```
initialize variables:
     telegram_token: 'bot_token';
     chatID: 'target_chatID';

construct the API URL:
     url= 'https://api.telegram.org/bot'+
     token +
     '/sendmessage_chat_id'+chatID;
     message_text; text_telegram_otp

set cURL options:
     curl_init();
     CURLOPT_URL = url,
     CURLOPT_RETURNTRANSFER = true
end.
```

III. RESULT AND DISCUSSION

A. NEURON VALUE PARAMETERS FOR GENERATING OTP

To test the feasibility of ideal parameter values in determining K, N, and L neurons, an interface needs to be created to make it easier for the generator process to see the ability to synchronize learning between the two neural machines. This is important to test and be able to determine the minimum K, N, and L value parameters that are suitable for use in OTP login authentication. Test generating an OTP by filling in the values in the blank form on the interface Figure 8.

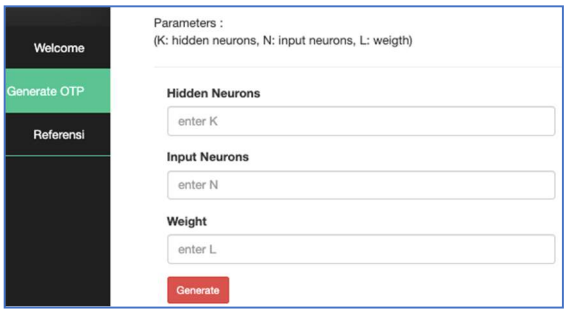

Figure 8. Interface Design for Generating OTP

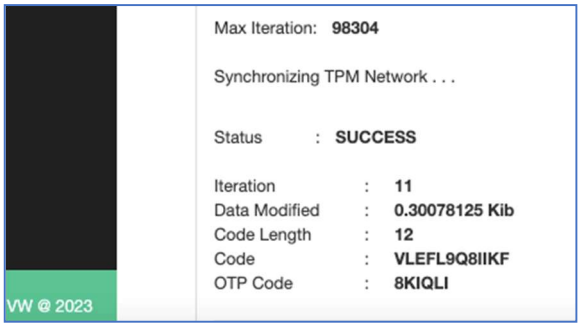

Figure 9. TPM OTP Synchronization Information

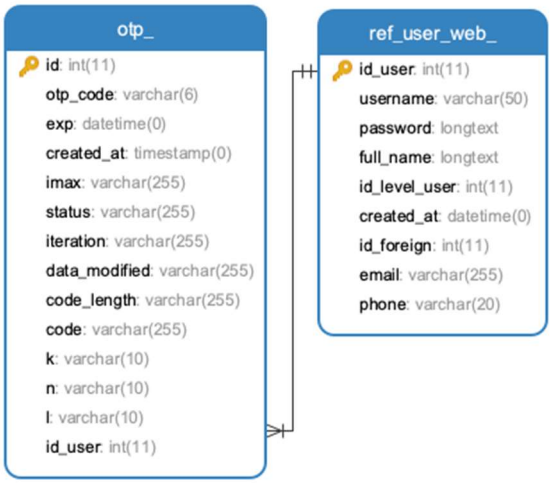

Figure 10. OTP TPM Login Database Modeling

Based on the neuron synchronization data in Figure 9, this is the first step in the concept that will be implemented in OTP-ANN with TPM for login authentication, so that a database storage model is formed as in Figure 10. Database modeling is implemented in OTP TPM by processing information such as user_id, and producing generators. The key corresponds to the K, N, and L neuron values. Testing of ideal parameter values based on minimum input values can be seen in **Помилка! Джерело посилання не знайдено.** with the lowest K, N, and L neuron values from grades 2–6. The maximum average iteration produced is 9562.16, provided that the value of K is≥4 if N is≥5 and L is≥5. Therefore, based on these conclusions, the implementation of neural synchronization for OTP-ANN TPM yields ideal parameters for use.

Table 1. Testing the Use of K, N, and L Neuron Parameters

Parameter			Iteration Max	Iteration	Data Modified (Kib)	Code Length	All OTP's	OTP Code	Status
K	N	L							
2	2	2	-	-	-	-	-	-	Failed
3	3	3	-	-	-	-	-	-	Failed
4	3	2	-	-	-	-	-	-	Failed
3	4	4	3072	73	1,140625	-	-	-	Failed
4	4	4	4096	27	0,52734375	-	-	-	Failed
3	4	3	972	2	0,03125	-	-	-	Failed
3	4	5	7500	4	0,0625	-	-	-	Failed
3	5	5	9375	852	15,80859375	-	-	-	Failed
4	5	4	5120	11	0,2578125	-	-	-	Failed
4	5	3	1620	1620	37,96875	-	-	-	Failed
4	4	5	1000	1	0,01953125	-	-	-	Failed
4	5	5	12500	11	0,2578125	7	LO8DNAG	GN8OLA	Success
5	5	5	15625	443	12,5458984375	8	ECFCI8MI	IM8CEC	Success
5	4	3	1620	377	8,8359375	-	-	-	Failed

Parameter			Iteration Max	Iteration	Data Modified (Kib)	Code Length	All OTP's	OTP Code	Status
K	N	L							
5	3	5	-	-	-	-	-	-	Failed
5	4	4	5120	5	0.1171875	-	-	-	Failed
5	4	5	12500	770	18,046875	7	GIIH7DF	FHIDGI	Success
6	3	3	-	-	-	-	-	-	Failed
6	4	3	1944	742	20,2890625	-	-	-	Failed
6	4	4	6144	7	0,19140625	6	NGHAGJ	GNHGAJ	Success
6	5	5	18750	1	0,033203125	10	QCDC9LFEIF	IFDC9F	Success
6	5	4	7680	2	0,06640625	8	CJHLMLI8	ILLC8H	Success
6	5	6	38880	13171	437,318359375	15	8CN9EICD58BNLG9	8NDC9B	Success
6	6	6	46656	93	3,6328125	18	G6CC6BJ8AG8EHB5B47	C8BJ46	Success
5	6	6	38880	7	0,232421875	15	L3M886A9DL6C3DK	AD98C6	Success

B. GENERATE OTP SYNCHRONIZATION WEIGHT VALUE UPDATE

The stochastic characteristic of TPM synchronization means that the epoch process in iterative training has different weight values for each training, even though the input K, N, L have the same value. When the synchronization process is carried out, the weight -atching process in the two parity neural machines will continue to change in a mutually interesting manner (attractive step) and weight indicate sthe steps of pushing each other (repulsive step) [32] [29].

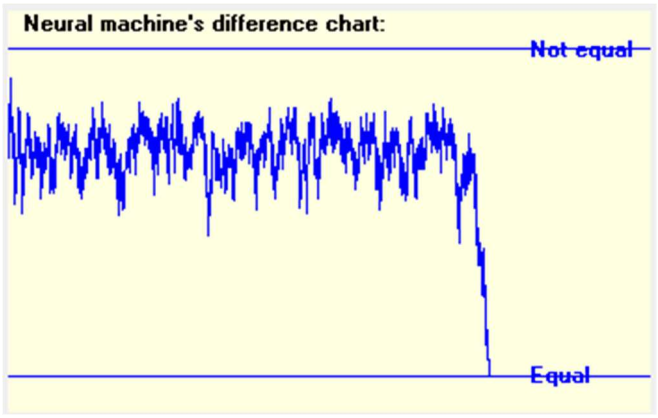

Figure 11. TPM Sync Weight Value Update Graph

As shown by the graph in Figure 11Помилка! Джерело посилання не знайдено., shows a curve line that rises and falls on synchronization using the values K=4, N=5, L=5. Illustration Figure 11. meaning that if the two weight values in the neural machine have a closer or smaller difference, it will go to the same value or the graph will go down. The condition of the two weights experiences an attractive step with a probability value that occurs as in equation (11).

$$\text{if } P(A) = (\tau^A = \prod_{K=1}^{1} \sigma_1^A = \sigma_1^B \,||\, \tau^A = \tau^B) \quad (11)$$

The attractive step state occurs when one or more of the weight values are hidden neurons $\boldsymbol{\sigma_1^A}$ has a slight difference or is equal to $\boldsymbol{\sigma_1^B}$. On the other hand, if the two weights have a difference in pushing each other or a repulsive step so that the values get further apart then the probability of what will happen is like equation (12).

$$\text{if } P(A) = (\tau^A = \prod_{K=1}^{1} \sigma_1^A \neq \sigma_1^B \,||\, \tau^A \neq \tau^B) \quad (12)$$

Equation (11) (12) will have an effect on the graph ***Помилка! Джерело посилання не знайдено.***., where during the synchronization process the parameter values K, N, L have an effect that causes an attractive step to occur with both weight values being the same so the synchronization is faster [13][33]. So, conversely, a repulsive step will occur and cause the synchronization to take longer [34]. Simulation of weight update values for synchronization to generate OTP as in Table 2. Referring to Table 2**,** the process of updating the weight values in iterations 1, 2, and 3 is the process of the two neural machines experiencing mutual pushing or repulsive steps, causing the weight values of the two TPMs to move further apart. Next, in the 2500th iteration, the process pulls each other's attractive steps on both neural machines, thus finding the same weight value.

C. GENERATE OTP SYNCHRONIZATION ITERATION CYCLE

The concept of epoch in the research carried out refers to training a model based on iteration cycles in generating synchronization of the two neural machines to produce OTP.

Table 2. Simulation of Weight Update Values

Iteration	Z	σ_1^A	σ_1^B	$Z^{A/B}$	W^A	W^B
1	$\begin{bmatrix} -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$	1	$\begin{bmatrix} 5 & 4 & -3 & -3 & 2 \\ 1 & -1 & 0 & 2 & 3 \\ -4 & -3 & -3 & 2 & 5 \\ 3 & 2 & 1 & 3 & 5 \end{bmatrix}$	$\begin{bmatrix} 5 & 4 & -2 & -2 & 0 \\ 1 & -1 & 0 & 1 & 2 \\ -3 & -3 & -3 & 2 & 4 \\ 2 & 3 & 1 & 3 & 4 \end{bmatrix}$
2	$\begin{bmatrix} 1 & 1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$	-1	$\begin{bmatrix} 2 & -3 & 2 & 1 & -5 \\ 1 & -2 & -1 & 0 & -5 \\ -1 & 2 & 1 & 5 & -5 \\ 0 & 0 & 3 & 4 & -5 \end{bmatrix}$	$\begin{bmatrix} 2 & -3 & 2 & 1 & -5 \\ 1 & 4 & -3 & 0 & -5 \\ -2 & 3 & 2 & 5 & -4 \\ 0 & 1 & 3 & 4 & -3 \end{bmatrix}$

Iteration	Z	σ_1^A	σ_1^B	$Z^{A/B}$	W^A	W^B
3	$\begin{bmatrix} 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ -1 & 1 & -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$	-1	$\begin{bmatrix} -4 & 5 & -5 & 4 & 0 \\ -5 & -5 & -5 & 5 & 4 \\ 5 & -4 & 1 & 5 & 3 \\ 5 & 4 & -5 & -4 & 2 \end{bmatrix}$	$\begin{bmatrix} 5 & -5 & 4 & -4 & 0 \\ -5 & -5 & -5 & 5 & 3 \\ -4 & 4 & -1 & -4 & 5 \\ 5 & 4 & -3 & -1 & 2 \end{bmatrix}$
n	$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$	$\begin{bmatrix} \cdots \\ \cdots \\ \cdots \\ \cdots \end{bmatrix}$	$\begin{bmatrix} \cdots \\ \cdots \\ \cdots \\ \cdots \end{bmatrix}$	…	$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$	$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$
2500	$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}$	1	$\begin{bmatrix} -4 & 5 & 3 & 3 & 5 \\ -5 & 4 & -1 & 5 & 5 \\ 5 & 3 & -5 & 5 & 5 \\ -4 & -4 & 5 & 4 & 4 \end{bmatrix}$	$\begin{bmatrix} -4 & 5 & 3 & 3 & 5 \\ -5 & 4 & -1 & 5 & 5 \\ 5 & 3 & -5 & 5 & 5 \\ -4 & -4 & 5 & 4 & 4 \end{bmatrix}$

As can be seen in *Помилка! Джерело посилання не знайдено.*, the parameter values in the network topology values K, N, L influence the overall length of the code digits. This proves that the characteristics of the TPM topology will update the weight values when synchronization occurs to achieve the same value on both parity neural machines [35].

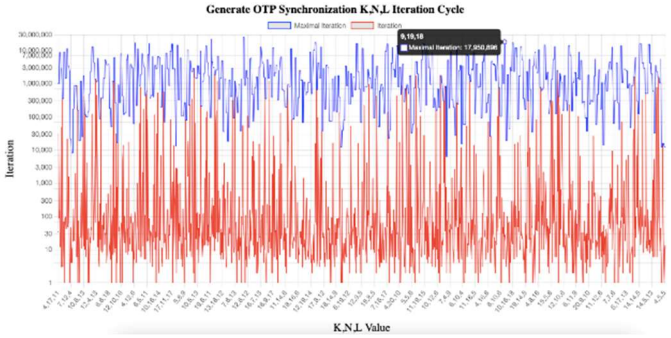

Figure 12. Generate OTP Synchronization Iteration Cycle

Figure 12 illustrates the iteration cycles required to generate a synchronized OTP under various configurations. These findings serve as a basis for modeling the synchronization process by varying K, N, and L within a range of 4 to 20. To address concerns related to synchronization overhead, it is important to emphasize that TPM synchronization does not require all iterations to be completed. As shown in Table 1 and Figure 12, synchronization can terminate early once the parity machines reach identical weight vectors, regardless of any pre-defined maximum iteration threshold. Moreover, only lightweight output bits are exchanged during synchronization, not full keys or weight states, thus maintaining communication efficiency and minimizing overhead. The average time per iteration is approximately 0.036 ms. For typical values of K, N, and L, synchronization is completed within an acceptable iteration range, ensuring that OTP generation remains practical and responsive for login-scale authentication scenarios.

The training process illustrated in Figure 12, based on 1252 simulation runs, shows that synchronization may require up to 30.000.000 times iterations in the worst case. However, the OTP generation process does not need to reach this maximum, as synchronization halts automatically upon convergence. While some instances may show longer synchronization times (as indicated by the red graph line), most OTP generations complete far earlier (as shown in the blue line), confirming that full iteration cycles are rarely needed. Therefore, although larger K, N, and L values increase synchronization time, they are not optimal for time-sensitive OTP use cases. In this implementation, a 6-digit OTP is randomly selected from the synchronized weight vectors and delivered through email and Telegram as part of the login authentication process.

The ability of TPM to dynamically conclude synchronization well before reaching the maximum iteration limit demonstrates its capacity to operate within efficient security design principles, particularly for lightweight authentication scenarios like web logins that demand both high throughput and minimal latency.

D. EXPERIMENTAL COMPARISON BETWEEN DETERMINISTIC OTP AND STOCHASTIC OTP TPM MODELS

This experimental test compares two approaches to One-Time Password (OTP) generation: a deterministic model representing Time-based OTP (TOTP), and a stochastic model based on a Tree Parity Machine (TPM). The deterministic OTP is generated by performing SHA-256 hashing on a seed consisting of a 30-second system time and the configuration parameters K, N, and L, then converted into OTP digits via a modulus operation. In contrast, the stochastic model generates an OTP through the synchronization of two simple neural networks (TPMs) that interactively update their weights until they are identical, and then the final weights are used as the source of the OTP. As illustrated in the following pseudocode:

Pseudocode for deterministic:

```
FUNCTION deterministic_otp(k, n, l):
    timestep ← current_time_in_seconds
// 30
    seed_string ← f"{k}-{n}-{l}-
{timestep}"
    hash_digest ← SHA-256(seed_string)
    digits ← convert each 2 hex chars →
int →
    mod 10
    RETURN first 6 digits as OTP
```

Pseudocode for stochastic TPM:

```
FUNCTION sync_tpm(k, n, l, max_iter):
    w_a, w_b ← random weights in range
[-l,l]
    for each k × n

    FOR step IN 1 TO max_iter:
        x ← random input matrix [-1, 1]
        of size k × n
        out_a, h_a ← TPM_output(w_a, x)
        out_b, h_b ← TPM_output(w_b, x)

        IF out_a == out_b:
            FOR i, j in k, n:
                IF h_a[i] == x[i][j] AND
```

```
                    h_b[i] == x[i][j]:
                    update w_a[i][j] and
                    w_b[i][j] by x[i][j]
                    (clamped)

        IF w_a == w_b:
            BREAK

    otp ← flatten w_a, convert abs(bit)
mod 10
    RETURN step, first 6 digits as OTP
```

Experiments such as those in Figures 13-18, showing experimental results at various iterations (50, 100, 150, 200, 500, 10.000) show a consistent pattern: the deterministic model is far superior in terms of time efficiency, as seen from the very small and stable distribution of execution times. However, from a security perspective, the stochastic approach shows higher entropy, especially at low iterations (50–150), which do not produce duplicate OTPs, in contrast to the deterministic model, which tends to produce duplications from the start. Although at high iterations (≥200) the stochastic model begins to show duplications, this is still within tolerable limits and depends on the size of the key space.

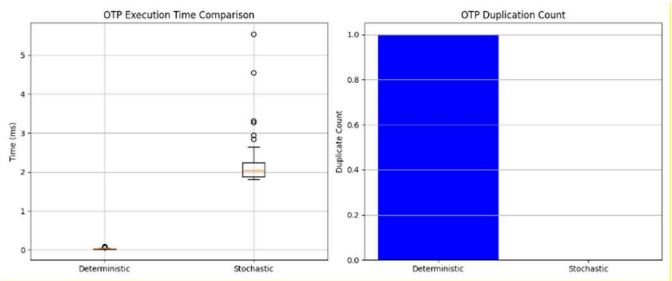

Figure 13. Comparison of deterministic and stochastic OTP at 50 iterations

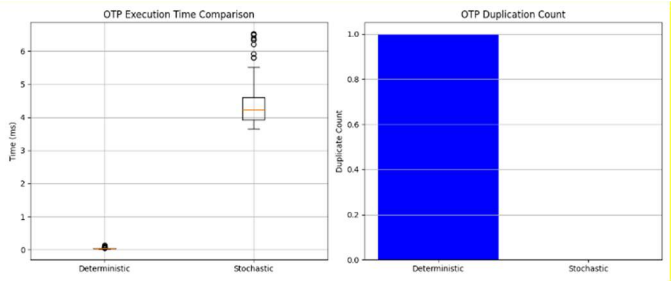

Figure 14. Comparison of deterministic and stochastic OTP at 100 iterations

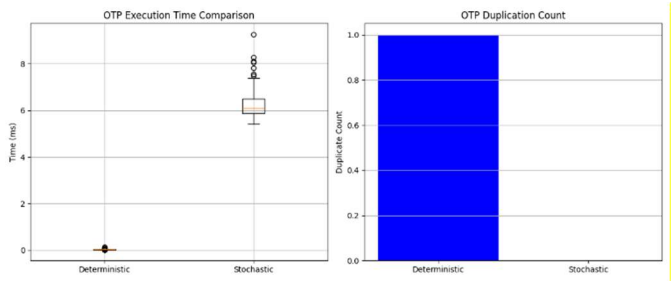

Figure 15. Comparison of deterministic and stochastic OTP at 150 iterations

These findings indicate that deterministic OTP is suitable for systems with time and resource constraints, but suffers from predictability weaknesses if the seed is not truly secret. In contrast, TPM-based OTP offers advantages in the context of cryptographic security and uncertainty, making it a strong alternative for high-risk authentication scenarios.

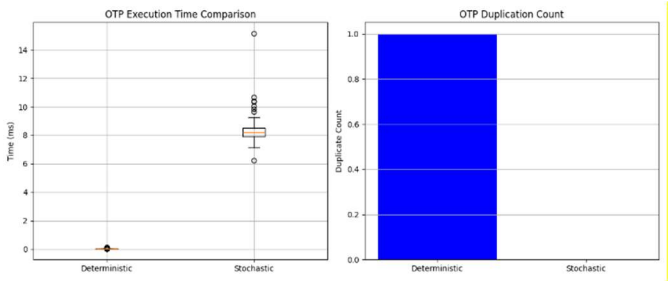

Figure 16. Comparison of deterministic and stochastic OTP at 200 iterations

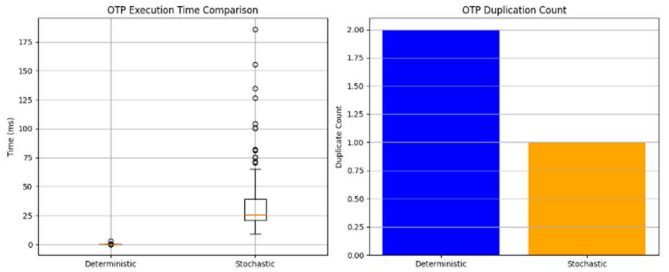

Figure 17. Comparison of deterministic and stochastic OTP at 500 iterations

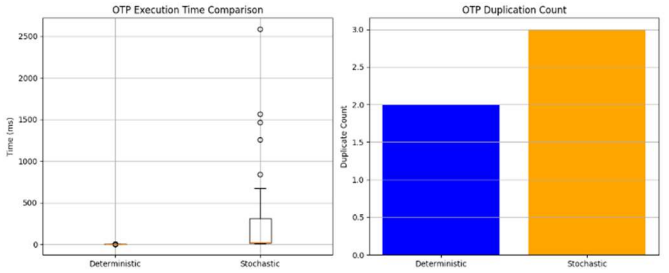

Figure 18. Comparison of deterministic and stochastic OTP at 10.000 iterasions

IV. CONCLUSION

The topology size in neural cryptography significantly influences the synchronization process of OTP generation using Tree Parity Machines (TPM). The effectiveness of ANN-based OTP generation for login authentication relies heavily on selecting optimal parameters. Experimental results indicate that stable synchronization can be achieved when $K \geq 4$, $N \geq 5$, and $L \geq 5$, with a maximum iteration threshold of 12.500. Importantly, the synchronization process often converges before reaching this limit, as it terminates once both TPMs reach identical weight configurations.

Each iteration takes approximately 0.036 ms, and during synchronization, both networks perform alternating attractive and repulsive weight adjustments. Despite using the same K, N, and L values, the stochastic nature of TPM ensures that each training session results in unique weight states, yielding highly unpredictable OTPs. This randomness makes TPM-based OTP generation particularly advantageous for high-security

authentication contexts, as the likelihood of generating duplicate codes is extremely low.

Optimal implementation prioritizes configurations that balance fast synchronization with security. Excessively large topologies, while theoretically expanding the key space, tend to increase computational cost without proportional gains in performance, especially considering that the tested upper iteration bound reaches up to 30,000,000 iterations. Therefore, moderately sized TPMs are more practical.

Understanding the underlying mechanism of TPM and its integration with ANN facilitates practical deployment in real-world systems. With proper configuration, the OTP output can be securely transmitted via web servers and integrated with platforms such as Telegram and email services. Upon successful generation, a 6-digit alphanumeric code is extracted from the synchronized weights, producing a valid and secure OTP for login authentication.

Acknowledgements

This research was funded by DAPTV Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi based on Surat Keputusan nomor 0793/D4/AL.04/2023 July 8, 2023, through a grant scheme for Penelitian Dosen Pemula (PDP) fiscal year 2023. The author would like to thank the Direktorat Jenderal Pendidikan Vokasi, LLDIKTI wilayah VI and PT. Egref Telematika Nusantara for the material and non-material support provided so that all activities run smoothly and the expected goals have been achieved.

References

[1] N. Sanchiga Nandhini and P. Arumugam, "Digital currency banking using block chain technology," *World J. Adv. Eng. Technol. Sci.*, vol. 8, no. 1, pp. 053–061, 2023, https://doi.org/10.30574/wjaets.2023.8.1.0011.

[2] M. Hammed and A.B. Adesi, "Authentication scheme using tree parity artificial neural networks for fraud detection in an on-line banking system," *Proceedings of the 4th National Development Conference of the School of Pure and Applied Science*, 2019, pp. 257–265.

[3] S. Matelski, "Universal key to authentication authority with human-computable OTP generator," *Proceedings of the 2022 17th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2022, pp. 663–671. https://doi.org/10.15439/2022F71.

[4] R. A. Grimes, *One-Time Password Attacks*, Wiley, 2021.

[5] K. Brindhashree and S. J. Prakash, "Data security based on cryptography, steganography combined with OTP algorithm and Huffman coding in the cloud environment," *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 2, no. 10, pp. 441–447, 2020.

[6] F. Ramadhani, U. Ramadhani, and L. Basit, "Combination of hybrid cryptography in one time pad (OTP) algorithm and keyed-hash message authentication code (HMAC) in securing the WhatsApp communication application," *J. Comput. Sci. Inf. Technol. Telecommun. Eng.*, vol. 1, no. 1, pp. 31–36, 2020, https://doi.org/10.30596/jcositte.v1i1.4359.

[7] M. Ziaurrahman, E. Utami, and F. W. Wibowo, "Modifikasi Kriptografi Klasik Vigenere Cipher Menggunakan One Time Pad Dengan Enkripsi Berlanjut," *J. Inform. dan Teknol. Inf.*, vol. 4, no. 1, p. (halaman 2), 2019. (in Indonesian).

[8] M. Ashiqul Islam, A. A. Kobita, M. Sagar Hossen, L. S. Rumi, R. Karim, and T. Tabassum, "Data security system for a bank based on two different asymmetric algorithms cryptography," *Lect. Notes Data Eng. Commun. Technol.*, vol. 53, no. January, pp. 837–844, 2021, https://doi.org/10.1007/978-981-15-5258-8_77.

[9] M. Volkmer and S. Wallner, "Tree parity machine rekeying architectures," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 421–427, 2005. https://doi.org/10.1109/TC.2005.70.

[10] A. P. Veera Manindra and B. Karthikeyan, "OTP camouflaging using LSB steganography and public key cryptography," *Proceedings of the 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 2022, pp. 109–115. https://doi.org/10.1109/ICESC54411.2022.9885542.

[11] B. T. Hammad, A. M. Sagheer, I. T. Ahmed, and N. Jamil, "A comparative review on symmetric and asymmetric DNA-based cryptography," *Bull. Electr. Eng. Informatics*, vol. 9, no. 6, pp. 2484–2491, 2020, https://doi.org/10.11591/eei.v9i6.2470.

[12] N. G. N. Amma and F. R. Dhanaseelan, "Optimal privacy preserving scheme based on modified ANN and PSO in cloud," *Res. Anthol. Priv. Secur. Data*, pp. 773–793, 2021, https://doi.org/10.4018/978-1-7998-8954-0.ch035.

[13] 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. Networks*, vol. 2021, 2021, https://doi.org/10.1155/2021/6680782.

[14] S. Y. Yan, *Computational Number Theory and Modern Cryptography*, Google Books, [Online]. Available at: https://books.google.co.id/books?hl=id&lr=&id=74oBi4ys0UUC&oi=fnd&pg=PR9&dq=Typically,+in+many+modern+cryptography+methods,+algebraic+number+theory+is+involved+&ots=fR44FTmYIO&sig=2DazU2MVXoRT-kb7vI6G65tFt_c&redir_esc=y#v=onepage&q=Typically%2C in many modern cryptography methods%2C algebraic number theory is involved&f=false.

[15] Z. Zheng, Modern Cryptography Volume 1, Springer, 2022, 359. https://doi.org/10.1007/978-981-19-0920-7.

[16] S. Chourasia, C. H. Bharadwaj, Q. Das, K. Agarwal, and K. Lavanya, "Vectorized neural key exchange using tree parity machine," *Compusoft*, vol. 8, no. 5, pp. 3140–3145, 2019.

[17] M. Dolecki and R. Kozera, "Distribution of the tree parity machine synchronization time," *Adv. Sci. Technol. – Res. J.*, vol. 7, no. 18, pp. 20–27, 2013, https://doi.org/10.5604/20804075.1049490.

[18] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography (1st ed.)*. CRC Press, 2022. https://doi.org/10.1201/9780429466335.

[19] C. Biswas, M. M. Haque, and U. Das Gupta, "A modified key sifting scheme with artificial neural network based key reconciliation analysis in quantum cryptography," *IEEE Access*, vol. 10, no. July, pp. 72743–72757, 2022, https://doi.org/10.1109/ACCESS.2022.3188798.

[20] Y. R. Mislovaty, I. Perchenok, W. Kanter, and Kinzel, "Secure key-exchange protocol with an absence of injective functions," *Phys. Rev.*, vol. 66, no. 6, 2002. https://doi.org/10.1103/PhysRevE.66.066102.

[21] I. Kanter, W. Kinzel, and E. Kanter, "Secure exchange of information by synchronization of neural networks," *Euro-physics Lett.*, vol. 57, no. 1, p. 141, 2002. https://doi.org/10.1209/epl/i2002-00552-9.

[22] M. Rosen-Zvi, E. Klein, I. Kanter, and W. Kinzel, "Mutual learning in a tree parity machine and its application to cryptography," *Phys. Rev. E*, vol. 66, no. 6, p. 66135, 2002, https://doi.org/10.1103/PhysRevE.66.066135.

[23] E. Shishniashvili, L. Mamisashvili, and L. Mirtskhulava, "Enhancing IoT security using multi-layer feedforward neural network with tree parity machine elements," *Int. J. Simul. Syst. Sci. Technol.*, pp. 6–10, 2020, https://doi.org/10.5013/IJSSST.a.21.02.37.

[24] E. M. M. Manucom, B. D. Gerardo, and R. P. Medina, "Analysis of key randomness in improved one-time pad cryptography," *Proceedings of the 2019 IEEE 13th International Conference on Anti-Counterfeiting, Security, and Identification (ASID)*, 2019, pp. 11–16. https://doi.org/10.1109/ICASID.2019.8925173.

[25] R. Manjupargavi, M. V. Srinath, "Efficient OTP generation with encryption and decryption for secure file access in cloud environment," *J. Commun. Technol.*, vol. 13, no. 2, pp. 2683–2688, 2022, https://doi.org/10.21917/ijct.2022.0397.

[26] L. C. B. C. Ferreira, P. R. Chaves, R. M. Assumpção, O. C. Branquinho, F. Fruett, and P. Cardieri, "The three-phase methodology for IoT project development," *Internet of Things*, vol. 20, p. 100624, 2022, https://doi.org/10.1016/j.iot.2022.100624.

[27] É. Salguero 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. Networks*, vol. 2019, 2019, https://doi.org/10.1155/2019/8214681.

[28] S. Matelski, "Human-computable OTP generator as an alternative of the two-factor authentication," pp. 64–71, 2022, https://doi.org/10.1145/3528580.3532842.

[29] X. Lei, X. Liao, F. Chen, and T. Huang, "Two-layer tree-connected feed-forward neural network model for neural cryptography," *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 87, no. 3, p. 032811, 2013, https://doi.org/10.1103/PhysRevE.87.032811.

[30] A. Ruttor, "Neural synchronization and cryptography," 2007, [Online]. Available: http://arxiv.org/abs/0711.2411.

[31] A. Ruttor, W. Kinzel, and I. Kanter, "Dynamics of neural cryptography," *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 75, no. 5, p. 056104, 2007, https://doi.org/10.1103/PhysRevE.75.056104.

[32] M. A. Budiman, Handrizal, and William, "A neural cryptography

approach for digital image security using Vigenère cipher and tree parity machine," *J. Phys. Conf. Ser.*, vol. 1898, no. 1, p. 012039, 2021, https://doi.org/10.1088/1742-6596/1898/1/012039.

[33] I. Meraouche, S. Dutta, H. Tan, and K. Sakurai, "Learning asymmetric encryption using adversarial neural networks," *Eng. Appl. Artif. Intell.*, vol. 123, p. 106220, 2023, https://doi.org/10.1016/j.engappai.2023.106220.

***VERA WATI** – is an Assistant Professor since 2020 in the field of Smart City and Information Technology, previously at Universitas Tunas Pembangunan Surakarta and now at Politeknik Negeri Indramayu. Research interest include artificial intelligence field, smart systems, and cyber cryptography.*

***NUR FITRIANINGSIH HASAN** – is an Assistant Professor of Computer Science at Universitas Muhammadiyah Papua. Her research interests include artificial intelligence, neural networks, and machine learning.*

***ACHMAD NUGRAHANTORO** – is a lecturer in Digital Business at Universitas Madani Yogyakarta and a full-stack developer practitioner at PT Egref Telematika Nusantara. His research interests include cybersecurity, information security, cryptography, and programming.*

***NISRINA YULIA SETYAWATI** – Received her Bachelor's degree in Smart City Information Systems from Universitas Tunas Pembangunan Surakarta in 2024. Her interests include system development, enterprise solutions, and optimizing business processes through technology.*