A neural cryptography approach for digital image security using Vigenère cipher and tree parity machine

M A Budiman, Handrizal, William
Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Jl. Universitas No. 9-A, Kampus USU, Medan 20155, Indonesia
mandrib@usu.ac.id, handrizal_tanjung@yahoo.com, william1099@gmail.com

Abstract. As with other form of digital media, image can also contain private information that should be kept secret. Commonly, symmetric and asymmetric cryptosystem can be used to secure an image. Symmetric-key cryptography is not a best solution as the secret key has to be transmitted through private channel. Asymmetric-key cryptosystem, however, allows transmission of key through public channel, but requires high computational power and long computational time. This paper proposes a neural cryptography approach to protect image by using tree parity machine and Vigenère cipher. By using this approach, the secret key will never be transmitted. Moreover, the whole process does not require high computational power and long computational time as in public-key cryptosystem. To overcome the weakness of Vigenère cipher, change is made to the algorithm by multiplying the key value with y-axis of image. The effectiveness is validated by the value of pearson correlation that is close to 0. Lastly, the result of experiments show neural synchronization time depends on the size of network, initial weights and randomly generated input vector.

1. Introduction
Data security is one of the most fundamental aspects of digital media communication. One form of digital media that has been increasingly popular due to the growing number of social media platforms is digital images. While it is publicly shared over the internet, image can contain private information that should be kept secret. Thus, an image protection mechanism is necessary to prevent unauthorized access to private image such as ID card or any private document in the form of digital image.

One of the most frequently used approaches to secure data is cryptography. Normally, the implementation of cryptography can be achieved by using symmetric-key and asymmetric-key cryptography. In symmetric cryptosystem, the same key is used to perform encryption and decryption. It is also generally faster than other cryptographic schemes as it only uses simple mathematical operation [1]. However, the downside is the security depends heavily on secrecy of the key. Thus, the key distribution has to be done through private channel to ensure that the key is known only to both trusted parties. Another approach is to use asymmetric cryptosystem. In asymmetric cryptography, both parties use a pair of keys, such as public key and private key, to perform encryption and decryption respectively. Data encrypted with public key can only be decrypted with the corresponding private key. Thus, the public key may be publicly shared as only the user who hold the corresponding private key can decrypt the data. By using this method, key transmission problem as in symmetric cryptosystem is no longer an issue. However, using asymmetric cryptosystem generally requires high
computational power and long computational time [1]. Furthermore, the encrypted file size is usually much higher than the original file especially if it is applied to file with large size such as image and video [2]. Consequently, in practice it is often paired with symmetric-key algorithm to encrypt session key before transmitting the key into a public network [3].

Considering the limitations of both cryptosystems, another method of securing data is needed to solve the problems. Recently, there has been a growing number of researches in the field of artificial neural network. Some researches conducted in [5] [6] indicate that some artificial neural network, with specific learning rule and activation function, can be incorporated with cryptography to potentially create a secure secret key exchange. Such network is called tree parity machine (TPM).

This paper proposes an approach to protect image by utilizing tree parity machine neural network to establish a more effective cryptography scheme. There are many cryptography algorithms available for image encryption, but in this study, we will be using Vigenère cipher as the image encryption algorithm.

By using this approach, the secret key is kept secret and will never be transmitted. Only input vector and output value is transmitted over public network. As opposed to asymmetric cryptosystem, this approach involves simple mathematical operation that is much faster and does not require high computational power. In addition, study conducted in [4] shows that the final weights generated from this neural synchronization process is undeterministic. Thus, it is suitable to be used as a secret key.

2. Methods

2.1 Vigenère Cipher

Vigenère cipher is one of the classic polyalphabetic substitution cipher that is often used to encrypt alphabetic character. It involves the use of various caesar ciphers based on the letters of some keyword [7]. Vigenère cipher operates by using shifting mechanism, where each character is shifted by some amount of places [7]. This process requires a Vigenère table to match the letters of plaintext with the corresponding key's character. Mathematically, the encryption and decryption can be written, respectively, as follows:

\[ C_i = P_i + K_i \pmod{256} \]  
\[ P_i = C_i - K_i \pmod{256} \]

Where \( C_i \) and \( P_i \) are color of \( i \)-th pixel of cipherimage and plainimage respectively. And \( K_i \) represents the value of key.

2.2 Tree Parity Machine

Tree parity machine (TPM) is a unique type of neural network in which it has only one output neuron in the output layer [8]. TPM network is classified as multi-layer network, which means it consists of more than two layers of computational units. Each units are connected to another units of the subsequent layer in a feed-forward way [8], meaning the network information moves forward in one direction from input layer to output layer. TPM hidden layer is composed by \( K \) hidden neuron, where each hidden neuron is connected to \( N \) input neuron leading to \( K \times N \) size of network. Neurons in TPM is built upon McCulloch-Pitts model with bipolar data representation [9]. For that reason, the value of input neuron \( x_{ij} \), output of hidden neuron \( \sigma_i \) and output neuron \( \tau \) consists only two values, namely 1 and -1.

\[ x_{ij} \in \{-1,+1\}, \sigma_i \in \{-1,+1\}, \tau_i \in \{-1,+1\} \]

Whereas the weights, which are associated with the input neuron and hidden neuron, are numbers in the range of \(-L\) to \(L\).

\[ w_{ij} \in \{-L, \ldots, 0, \ldots, +L\} \]
Here $i \ (1 \leq i \leq K)$ denotes $i$-th hidden neuron in hidden layer and $j \ (1 \leq j \leq N)$ denotes the element of vectors. In TPM, specific activation function is used, and it is known as signum or step activation function. Below is the formula of the activation function.

$$Sgn(x) = \begin{cases} 
1 & \text{if } x > 0 \\
0 & \text{if } x = 0 \\
-1 & \text{if } x < 0 
\end{cases} \quad (5)$$

If the output of function above is 0, the value will be mapped to -1 to ensure that the output is either 1 or -1. Output of hidden neuron $\sigma_i$ and output neuron $\tau$ can be obtained, respectively, by using the following formula:

$$\sigma_i = sgn\left(\sum_{j=1}^{N} w_{ij}x_{ij}\right) \quad (6)$$

$$\tau = \prod_{i=1}^{K} \sigma_i \quad (7)$$

2.3 Proposed Method

In this paper, a neural cryptography approach is proposed for a digital image security as in Figure 1. Note that the cryptanalysis of the proposed method will not be discussed in this paper. There are two main procedures involved in this method: key generation stage and encryption/decryption.

2.3.1 Key Generation. In this stage, two parties are involved in neural synchronization process. First, TPM initialization phase occurs, in which both sender and receiver initialize their own TPM network with the same K, N and L value. During this phase, initial weights of each TPM $w_{ij}^{A/B}$ are randomly chosen and are kept secret. Then, in each synchronization phase, both parties receive the same randomly generated input vector $x_{ij}$ and compute their own output neurons $\tau^{A/B}$. Afterwards, the output neuron computed on each TPM is exchanged with each other.

Weights’ adjustment decision will be made by comparing the value of output neuron. If both output neurons are different, $\tau^A \neq \tau^B$, the weights are not updated. Otherwise, the weights are adjusted based on the selected learning rules. In this paper, Hebbian learning rule will be used on each TPM. The synchronization phase is running until both parties reach the synchronization state, which is marked by having identical weights on both TPM, $w_{ij}^A = w_{ij}^B$. These identical weights are then used as key. The exact algorithm is shown below.

1. Input K, N, L
2. Initialize TPM weights $w_{ij}^{A/B}$ randomly where $w_{ij}^{A/B} \in \{-L, ..., 0, ..., L\}$
3. **While** $w_{ij}^A \neq w_{ij}^B$ for all $i \ (1 \leq i \leq K)$ and $j \ (1 \leq j \leq N)$ **do**
(a) Generate input vector $x_{ij}$ randomly where $x_{ij} \in \{-1, 1\}$

(b) Compute hidden units, $\sigma_i^{A/B} = \text{sgn}\left(\sum_{j=1}^{N} w_{ij}^{A/B} x_{ij}\right)$

(c) Compute output unit, $\tau_i^{A/B} = \prod_{i=1}^{K} \sigma_i^{A/B}$

(d) if $\tau^A \neq \tau^B$ then
   goto (a)
else
   if $\tau^A = \tau^B$ then
      Apply Hebbian learning rule
      \[ w_{ij}^{A/B} + g(\sigma_i^{A/B} x_{ij} \Theta(\sigma_i^{A/B}, \tau_i^{A/B}) \Theta(\tau^A, \tau^B)) \]
   end if
end if

End while

4. Since $w_{ij}^A = w_{ij}^B$, use $w_{ij}^{A/B}$ as a common secret key

2.3.2 Encryption and Decryption. Encryption and decryption will be performed by using Vigenère cipher. In the Vigenère cipher, if the key length is shorter than the length of plaintext, the key will be repeated until both lengths are the same. However, due to the repeating nature of the key, the colors of pixels will likely be shifted by the same key value. As a result, there is little visual change between plainimage and cipherimage, as can be seen in Table 1.

| Table 1. Plainimage and cipherimage comparison |
|-----------------------------------------------|
| Plainimage | Cipherimage | Pearson correlation coefficient |
| ![Plainimage](image1.jpg) | ![Cipherimage](image2.jpg) | 0.584 |

In consequence, change in vigenère cipher algorithm is necessary to solve this problem. Such change is made by multiplying the key value with the $y$-axis (vertical axis) of the image. The idea behind this is to have different key value in each row of image matrix, thus reducing the likelihood of same color of pixel being encrypted with the same key value.

3. Results and discussions
The proposed method was implemented as a desktop application using Java programming language. The simulations were run on an Intel Core i3-4030U 1.90 GHz CPU with 16 GB of RAM. First, both sender and receiver initialized their own TPM with $K = 4$, $N = 4$ and $L = 50$. Then, both parties are involved in neural synchronization process. This process took about 0.217 second to complete and the learning steps required for both TPM to achieve full synchronization state are 4363 steps. At the end of neural synchronization phase, both TPM achieve identical weights. These identical weights are then used as a common secret key. The generated key is shown below.

Key: 49 24 50 50 50 27 50 48 49 50 41 50 50 5 49 50

As stated previously, the weakness of vigenère cipher comes down to the repeating nature of the key. To address the issue, change is made to the algorithm by multiplying the key value with the $y$-axis of the image. To verify the effectiveness of the algorithm, an experiment is conducted to two plainimages. Those plainimages will be encrypted using both standard vigenère cipher and modified vigenère cipher. The results are later compared by using pearson correlation coefficient to measure the
statistical association between plainimage and cipherimage. Mathematically, the Pearson correlation coefficient is defined as follows:

\[
    r_{x,y} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}
\]

(8)

Table 2. Pearson correlation coefficient comparison

| Plain image | Vigenère Cipher | \(r_{x,y}\) | Modified Vigenère Cipher | \(r_{x,y}\) |
|-------------|-----------------|-------|-------------------------|-------|
| ![Plain image](image1) | ![Cipher image](image2) | 0.58 | ![Cipher image](image3) | 0.0022 |
| ![Plain image](image4) | ![Cipher image](image5) | -0.548 | ![Cipher image](image6) | 0.0004 |

As can be seen in Table 2, the Pearson coefficient of modified Vigenère cipher is much lower than Pearson coefficient of conventional Vigenère cipher. The \(r\) value of modified Vigenère cipher is close to 0, thus making it effective to ensure the confidentiality aspect of cipherimage. In order to evaluate the performance of the proposed method, experiments with variables that affect synchronization time, were conducted. Table 3 displays neural synchronization time for each TPM with different size.

Table 3. Neural synchronization time

| No | Tree Parity Machine | Epoch / steps | Synchronization Time (ms) |
|----|---------------------|---------------|--------------------------|
|    | \(K\) \(N\) \(L\) |               |                          |
| 1  | 3 4 10              | 169           | 21                       |
| 2  | 5 7 10              | 662           | 29                       |
| 3  | 5 8 10              | 84            | 6                        |
| 4  | 6 9 10              | 7153          | 147                      |
| 5  | 10 10 10            | 42504         | 541                      |

In the table above, it can be seen that in general, the larger the size of network, the number of training steps it took to achieve full synchronization state is also higher, thus requiring longer time to synchronize. However, notice that at the third test, TPM took less time to synchronize than the previous two tests despite having a larger network size. This is due to the fact that, though it does affect the synchronization time to some degree, the network size is not the only variable that affects synchronization time.

In order to identify another variable that influences the synchronization time, we will be measuring the similarity of initial randomly chosen weights. The measurement being used is cosine similarity, which is a metric to measure the similarity between two vectors. The formula of cosine similarity can be written as follows:

\[
    \text{Similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}
\]

(2)
Where $A_i$ and $B_i$ are component of vector A and B respectively. The result of experiment is shown in Table 4 below.

**Table 4. Initial weights comparison**

| No | Tree Parity Machine | Epoch / steps | Cosine similarity | Synchronization Time (ms) |
|----|---------------------|---------------|------------------|--------------------------|
| 1  | 3 4 10              | 169           | 0.77414          | 21                       |
| 3  | 5 8 10              | 84            | 0.78627          | 6                        |

In the table 4, it is clear that both TPM at third test will synchronize faster than both TPM at the first test since the initial weights of both TPM are much more similar. The more similar the initial weights are, the number of steps required to reach synchronization state is lower, thus resulting to faster synchronization time. However, during the synchronization phase, both TPM are not guaranteed to always perform attractive step. This can be seen from the Figure 2 below.

![Figure 2. Attractive and repulsive steps of TPM](image)

In the figure 2 above, two TPM with $K = 3$, $N = 12$, $L = 10$, are involved in neural synchronization. The process took 472 steps to complete which is marked by the value of cosine similarity, 1. Notice that in the figure above, the weights of both TPM are not guaranteed to always be closer to each other. This can be seen from the downward slope in some learning steps.

In general, attractive step occur if both TPM have the same hidden units representation $\{\sigma_1^{A/B}, \sigma_2^{A/B}, \sigma_3^{A/B}, \ldots, \sigma_n^{A/B}\}$ [10]. In contrast, repulsive step occur if both TPM have different representation of hidden units [10]. Hidden unit can be calculated by summing all the product of weights and input vectors. Therefore, the chance attractive or repulsive step will occur depends on the randomly generated input vectors. The frequency of attractive or repulsive step occurring will affect the synchronization time. The more the attractive step occurs, the faster the synchronization time will be and vice versa.

4. Conclusion

To conclude, a neural cryptography approach can be used to enhance the effectiveness of digital image security. The weakness of vigenère cipher can be overcome by multiplying the key value with the y-axis of the image. The effectiveness is proven by the value of pearson correlation that is close to 0. The result of experiments show that neural synchronization time depends on the size of network, initial weights and randomly generated input vector. The cryptanalysis of the proposed method is not discussed in this paper. Therefore, it is strongly encouraged for other researchers to explore the security and the reliability of the proposed method.
References

[1] Maqsood F, Ahmed M, Ali M M, Shah M A 2017 Cryptography: A Comparative Analysis for Modern Techniques *Int. J. of Adv. Comp. Sci. and Applications*

[2] Alese B K, Philemon E D, Falaki S O 2012 Comparative Analysis of Public-Key Encryption Schemes *Int. J. of Eng. and Technology* 2 9

[3] Rachmawati D, Sharif A, Jaysilen and Budiman M A 2018 *IOP Conf. Ser.: Mater. Sci. Eng.* 300 012042

[4] Sowmya D, Priyanka B R 2016 Design of Word Based Stream Cipher using Tree Parity Machine *Imperial J. of Interdisciplinary Research* 2 10

[5] Pattanayak S, Ludwig S A 2018 Encryption Based on Neural Cryptography (eds) Hybrid Intelligent Systems, Advances in Intelligent Systems and Computing 734 pp. 321–330, March 2018.

[6] Fadil A, Riadi I, Nugrahantoro A 2020 Kombinasi Sinkronisasi Jaringan Syaraf Tiruan dan Vigenere Cipher untuk Optimasi Keamanan Informasi Digitalzone: Jurnal Teknologi Informasi dan Komunikasi 11 1

[7] Kester Q 2012 Cryptosystem based on Vigenère cipher with varying key *Int. J. of Adv. Res. in Comp. Eng. & Technology* 1 10

[8] Klein E, Mislovaty R, Kanter I, Ruttor A, and Kinzel W 2005 Synchronization of neural networks by mutual learning and its application to cryptography *Advances in Neural Information Processing Systems*, 17 (MIT Press: Cambridge) pp 689-696

[9] Dolecki M and Ryszard K 2013 Distribution of the Tree Parity Machine synchronization time *Adv. in Sci. and Technology Res. J.* 7 18 pp 20–27

[10] Klimov A, Mityaguine A, and Shamir A 2005 Analysis of Neural Cryptography Advances in Cryptology—ASIACRYPT p 288