JSR Publications

Available Online

J. Sci. Res. **14** (2), 471-481 (2022)

JOURNAL OF SCIENTIFIC RESEARCH

www.banglajol.info/index.php/JSR

An Artificial Neural Network Technique of Modern Cryptography

S. K. Pal[1*], B. Datta[2], A. Karmakar[1]

[1]Department of CSE, Maulana Abul Kalam Azad University of Technology, Nadia, West Bengal, India

[2]Department of CSE, Budge Budge Institute of Technology, Kolkata, West Bengal, India

Received 15 September 2021, accepted in final revised form 17 February 2022

Abstract

An enormous volume of data transmitted through the internet in every second nowadays, so data protection is still a big challenge as communication channels are mostly public. Several approaches are there for protection of data when transmitted through the network channel, cryptography is one of them. Cryptography is a technique to hide original and confidential information from intruder. Considering information security principles i.e. confidentiality, integrity, and availability (CIA-triad) an efficient, portable, dynamic, and simple, artificial neural network (ANN) based cryptography algorithm have been presented in this paper. A multi-substitution tree parity machine (TPM) approach is used here to generate dynamic single digit secret key. The secret key created by the tree parity machine will be different in each phase of execution. This single digit dynamic secret key is used in the proposed encryption and decryption techniques. The complexity and execution time taken by the proposed algorithm is improved comparing to the existing algorithms. The proposed algorithm provides higher reliability and robustness in terms of security because of it dynamic nature.

Keywords: Cryptography; Encryption; Decryption; Information security; Artificial neural network; Tree parity machine.

doi: http://dx.doi.org/10.3329/jsr.v14i2.55669 J. Sci. Res. **14** (2), 471-481 (2022)

1. Introduction

Cryptography is the process of hiding information that allows only the sender and intended recipient of a message to view its contents. Several approaches are there for development of cryptography algorithms. Some of those cryptography approaches are traditional, mathematical, graph, artificial intelligence etc. Cryptography methods are employed in various applications in today's internet-driven communication. The Cryptography techniques are applied in modern network communications for several reasons. It provides security of the data during communication through a public channel and also increases reliability, robustness, efficiency of the communication channel. It is added to the network as it has many purposes such as reliability i.e. it prevents nodes to

* *Corresponding author*: sarbojay@gmail.com

perform task ambiguously, fault tolerance i.e. when certain nodes are unable then someone else will perform the task, and protection i.e. the confidence needed to do the job is exchanged among nodes [1].

For any cryptography procedure, encryption and decryption are core operations in which the key generation is a vital task in operational processes. The reliability and robustness of the algorithm is based on efficiency of the key generation. In real world, the confidential information needs high security and privacy by preventing from malicious activities. Data confidentiality, sender-side data protection and non-repudiation should be preserved well during data communication [2].

The Symmetric Key Encryption and Asymmetric Key Encryption are two major categories of cryptographic methods. This paper is concerned with the symmetric key methodology which uses a similar key between the communicating entities involved. The key generation guarantees the reliability and protection of the encrypted data, how well it was immune to attacks, and demonstrates its performance in various parameters. In asymmetric key technique, both public and private keys are used by the sender and receiver for encryption and decryption of data [3].

Asymmetric cryptographic algorithms originated in the modern era of cryptography. During 1970, the theory of Diffie and Hellmann key exchange [3] first introduced this form of cryptography technique. The idea is, at the time of key exchange the third party Eve, intercepts the public value of the Sender and sends its public value to the Receiver. As Receiver conveys the public value, Eve replaces it with its own and sends it to Sender. Eve and Sender then settle on one common key and Eve and Receiver decide on a mutual key for another. During this exchange, Eve actually decodes all messages received by Sender or Receiver and then read and probably alter the message with the correct key before re-encryption and to pass them on to the other party. But this concept is complex and time-consuming.

In the year 2021, Dwivedi [4] has introduced a new block cipher, BRISK, with a block size of 32-bit. The BRISK is a block cipher from the family of Feistel Cipher. The cipher design is straightforward due to the use of simple round of BRISK operations. These operations can be efficiently run in available hardware and suitable for software. Another major concept used with this cipher is dynamism during encryption process for each session; that is, instead of using the similar encryption algorithm, participants use different ciphers for each session. There are several traditional, non-traditional and artificial intelligence approaches are existing for implementation of the symmetric key and asymmetric key cryptography for block cipher.

In the year 2021, Thabit *et. al.* [5] published a paper on New Lightweight Cryptographic Algorithm(NLCA) for enhancing information security in cloud environment. The NLCA is a symmetric data encryption approach. The NLCA algorithm is a 16 bytes (128-bit) block cipher method and uses 16 bytes (128-bit) key to encrypt the data. It is enthused by Feistel, and substitution permutation architectural method was used to improve the complexity of the encryption. The algorithm attains Shannon's theory of diffusion and confusion by the associating of logical operations, such as XOR, XNOR,

shifting, swapping etc.

Balaji *et. al.* [6] reported another technique, 'Data Security and Deduplication Framework' for securing data in public and private cloud environment. The data security and deduplication framework has a cloud service which provides security service by means of symmetric cryptographic encryption. This is known as Enhanced Convergent Encryption. The Enhanced Convergent Encryption method comprises of two cryptographic symmetric encryption techniques: first is Enhanced Data Convergent Encryption (EDCE) and second is Enhanced Symmetric Convergent Encryption (ESCE). In the ESCE method data is divided into 128-bit blocks, and key is used 128-bit length with each block of data for further operations.

In this paper the concept of Tree Parity Machine has used for single-digit secret key generation, and multi-substitution of characters is introduced in the encryption and decryption scheme. The proposed algorithm uses symmetric key concept for encryption and decryption process. The XOR operation is used in the proposed encryption and decryption scheme for betterment and strengthen of its internal structure. Due to the dynamic nature of the proposed algorithm the reliability, efficiency, and security are much better comparing to the existing similar category algorithms. Significantly, the proposed cryptography algorithm creates different secret key in each session due to it vibrant nature. Therefore, revealing of the secret key is almost impossible.

2. Terminology

In this section some terminologies have defined which is used throughout the paper.

2.1. *Artificial neural network*

Artificial Neural Networks (ANN) is unique system of computing that simulate the approach of human brain functionalities and signal processing techniques. The biological neuron's structure of the human brain is represented using the concept of a graph in ANN [7]. The human brain is built-up integrating 86 billion nerve cells called neurons. In an artificial neural network, each neuron is represented by a node. All neurons are connected by edges or links and it has some weight; one side of the link shows the output of one neuron and another side of the link shows the input of another neuron. The input layer nodes of the artificial neural network take input data and then process it. The outcomes of the input layer are finally forwarded to the nodes of output layer through hidden layers, and produce output [8].

2.2. *Tree parity machine*

Tree Parity Machine (TPM) is a unique model of a multilayer feed-forward neural network. In this model, connections between the nodes never form circuits. A TPM consists of one output neuron, H hidden neurons, and (H × N) input neurons. TPM is the

oldest model of artificial neural networks. In this method, the data is moved from the input neurons to the output neuron through hidden neurons.

2.3. *Secret key*

Cryptography is the process of data encryption and then decryption using some standard algorithm and key. The key may be private or public. The key is generated by the key generator or key may be decided confidentially. When a similar key is used by the sender and receiver for the encryption and decryption process, the process is called symmetric-key cryptography. The key is secret in this case, the sender and receiver only know the key. In asymmetric key cryptography, the sender and receiver uses a secret key as well as a public key for encryption and decryption process [9].

2.4. *Symmetric key*

In the symmetric key cryptosystem, same key is used by the sender and receiver for encryption and decryption process. The key is extremely secret in this method, other than the sender and receiver nobody knows it. In private key cryptography, the sender and receiver share a secret key confidentially and separately before communication. Therefore, both parties must have a copy of common secret key [10].

2.5. *Cypher text*

The Cipher text is the outcomes of the encryption process. A set of plain text is encrypted using a secret key and create cipher text as output. Only an authenticated person can decipher the cipher text [11].

2.6. *Plain text*

It is a text available in the book, newspaper, etc. and everybody can read it. A plain text is the outcome of the decryption process. A set of cipher text is decrypt using a secret key to recover the plain text [12].

3. Secret Key Generation Using TPM

In the proposed encryption and decryption algorithm, the secret key is generated using the Tree Parity Machine (TPM) scheme. In TPM technique, the user input is feed into the input layer nodes and weight is assigned to the edges. In first layer, the input string is transformed into ASCII code string and then it is equally divided into two halves. This output is then feed into next layer of the TPM, where both halves are passes through the sigma function shown in equation (1) [13].

$$\sigma_i = sgn\left(\sum_{j=1}^{N} W_{ij}\, X_{ij}\right), \quad (1)$$

The output provided by sigma (σ) function is feed into the final layer. In the final layer, the output is calculated using the function tau (τ) shown in equation (2).

$$\tau = \prod_{i=1}^{k} \sigma, \quad (2)$$

If the value of tau is equal for both halves, it is considered that the input value is synchronized. The synchronized input value will be further considered as per the rules to generate secret key. The complexity of the secret key generation algorithm is, O(n).

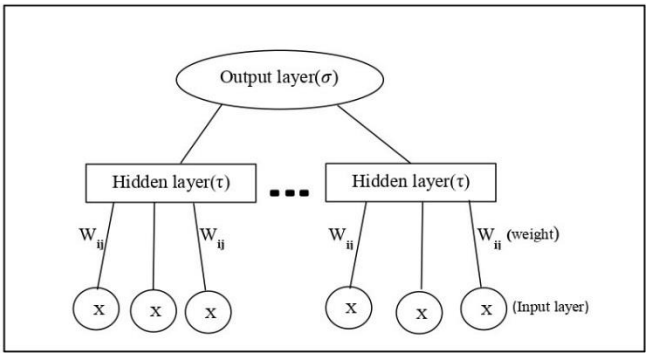

Fig. 1. TPM approach of secret key generation.

The single-digit secret key is calculated by adding all the digits obtained from the TPM algorithm. This single-digit secret key is used for the encryption and decryption process, in this paper.

3.1. *Key generation process*

In the proposed secret key generation algorithm two tree parity machines(TPM) are considered: TPM-A and TPM-B. The weight of the edges having in between input node and hidden node of TPM is assigned a values in the range {-L,...0..., L}. The user shall enter the input values of the input layer nodes. Calculated outcomes of the hidden layer using equation (1), this output is denoted by sigma (σ). The signum function will returns the values in the range {-1,0,1}, i.e., sgn(x) = {-1 if x<0, 0 if x=0, 1 if x>1}. The values of hidden layer obtained from both the TPM's are then used to find the values of the output layer using equation (2). The outcome of the output layer is denoted by, tao (τ). If, the output values of both the TPMs is same then update the weight with the proposed rule shown in equation (3).

$$W_{ki} = W_{ki} + \tau, \quad (3)$$

If updated weight of both TPM's gets the same then we consider that both the weights are synchronized. But, if the updated weights are not same then we have to update the weights further until the weight becomes same.

When, weight of both the TPM's are same i.e. synchronized to each other, applied modulus operation to make the weights random. The updated synchronized weight is modulus by the current system time shown in equation (4).

$$W_{ij} = W_{ij} \,\%\, current\ system\ time, \tag{4}$$

The key obtained in this method is random and unique every time. Each digit of the key is then added to get the single digit secret key. This single digit secret key is used in proposed encryption and decryption algorithms.

3.2. *Secret key generation algorithm*

Step 1: Taken two Tree Parity Machines, TPM-A and TPM-B.
Step 2: Initialize the weight of the edges having in between input layer nodes and the hidden layer nodes, in the range, {-L ...., 0,....., L}.
Step 3: Enter input value of the input nodes, 1,2,3,4,…, N.
Step 4: Find out value of the hidden layer using the values of input (X) and weight (W) in the equation (1), which is denoted by sigma (σ),
Step 5: Find out value of the output layer using equation (2) for both the TPMs, using corresponding values of sigma (σ). The output layer value is denoted by, tao (τ).
Step 6: When output of TPM-A and output of TPM-B is same then move to step 7, otherwise, move to step 2, of this algorithm.
Step 7: Update weight of the TPM's using the rule proposed in equation (3).
Step 8: When updated weight of TPM-A and TPM-B becomes synchronised then, calculate random weight using the equation (4). This operation will make the weight random each time. This updated random weight becomes the secret key.
Step 9: Add all the digits of the secret key to form single-digit secret key (K_s).
Step 10: Stop.

4. Encryption Process

In this section proposed encryption technique and algorithm have been described with a block diagram.

4.1. *Encryption technique*

In the proposed encryption algorithm, the plain text entered as input to be converted into the ASCII format. In the next step, the ASCII value of the lower case and upper case letter converted to the upper case and lower case, respectively. All available spaces present into the text filled with anyone ASCII value having a range in between 128-255 and keep the value of space for further use, this is encoded text, E_1. In each even position (indexed from 0) of the encoded text E_1 add the single-digit key value, it will form the encoded text E_2. Any key generator may be used to make a single digit secret key K_s. In this paper a TPM based key generation algorithm is used. A multiple digit key converted to single digits by

adding each digit of the key until it becomes a single digit. Perform XOR operation between encoded text E_2 and secret key K_s for getting encoded text E_3. Finally, encode text E_3 and UTF-8 encoding scheme generate the cipher text.

4.2. *Encryption algorithm*

Step 1: Taken input messages from the user.
Step 2: Switch the cases of each character, i.e. lowercase convert to uppercase and vies versa, a→A, B→b, etc.
Step 3: Reciprocate each letters in the modified message to their respective character of the English alphabets and then convert each character to ASCII Code i.e., A→Z→90, B→Y→89, etc.
Step 4: Convert the spaces of the message to any random value of ASCII code having in the range: 128 to 255, and keep that value of space (say, Spc). This modified text forms (E_1).
Step 5: In each even position (indexed from 0) of the encoded text E_1, add the single-digit key value which was generated by the secret key generation algorithm, given in section 3.2. This will form encoded text(E_2).
Step 6: Perform XOR operation between the single-digit secret key (K_s) and encoded text (E_2) to produce encoded text (E_3).
Step 7: Append space value (Spc) at the right side end of the encoded text (E_3), encode it to UTF-8 to forms cipher text.
Step 8: Stop.

4.3. *Encryption process diagram*

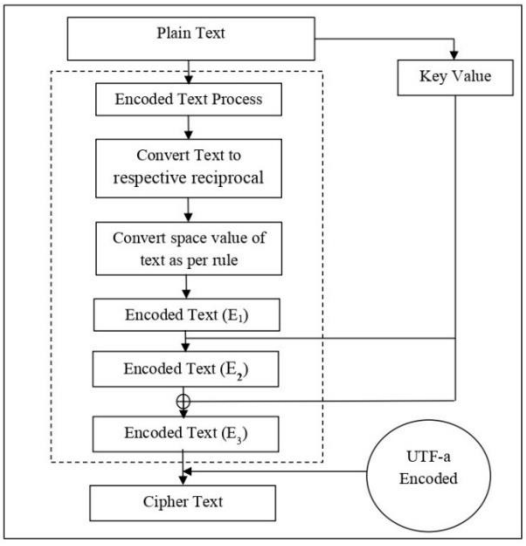

Fig. 2. Encryption process flow diagram.

5. Decryption Process

In this section proposed decryption technique and algorithm have been described with a block diagram.

5.1. *Decryption technique*

The cipher text generated by the encryption algorithm is used as input in the decryption algorithm. Decoded cipher text from the UTF-8 encoding scheme and get the first decrypted text D_1. Then perform XOR operation with D_1 and secret key K_s for getting decrypted text D_2. The key is same as it is used in the encryption process. Subtract the key value from decoded text D_2's even position (index from 0) for getting the decrypted text D_3. For any ASCII value which is greater than 127, replace it with ASCII value of space i.e. 32, the outcome is the decoded text D_4. Finally, switch the lower case and upper case letters of decrypted text D_4 to the upper case and lower case letters respectively. The output obtained is the desired plain text.

5.2. *Decryption algorithm*

Step1: Taken cipher text as input in decryption algorithm.
Step2: Decode the cipher text from UTF-8 encoding, gets decoded text (D_1).
Step3: Perform XOR operation between the decoded text D_1 and single-digit secret key K_s to gets decoded text (D_2). The Key value same as it was used in the encryption process.
Step4: Subtract the key value from the decoded text D_2's even position to gets decrypted text (D_3).
Step5: Replace each characters of the decoded text D_3 with ASCII value, if ASCII value is greater than 127 replace it with ASCII value 32 i.e. Spc.
Step 6: Swap the cases of the characters of D_3 as mentioned in an encryption algorithm (section 4.2, step 2). This operation will create the desired plain text.
Step 7: Stop.

5.3. *Decryption process diagram*

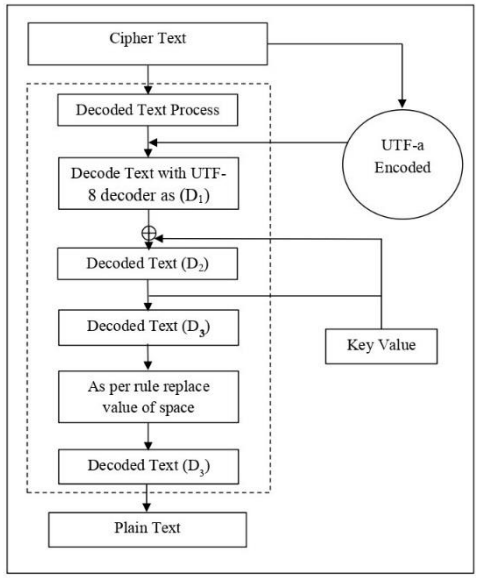

Fig. 3. Decryption process flow diagram.

6. Complexity Analysis

In this section complexity of the proposed algorithm has described and experimental results are analysed with the existing algorithms.

6.1. *Complexity of the algorithm*

The time complexity of the algorithms is measured assuming the length of the input string, n. In key generation algorithm the values of sigma (σ) and tau (τ) are calculated using the same input. Therefore, the time complexity for generating the secret key is, O(n). For each operation in the encryption algorithm like swapping and reciprocating of alphabets, convert them into ASCII code, change spaces into random characters, adding keys to each even elements of index places, and combine them to generate the cipher are done by one pass of the control. Therefore, time complexity for all these operations done in the encryption process is, O(n). Similarly, the time complexity of the decryption algorithm is, O(n) as well, because it will follow all the operations which were performed in the encryption algorithm but all operations are done in the reverse order. Thus, the overall time complexity of the algorithm presented in this paper is, O(n).

6.2. *Experimental result analysis*

The existing algorithms of Pal *et. al.* [7] and Pujeri *et. al.* [14], presented algorithms in this paper are examined using the Intel i3, 6th generation processor. The supported system is a 64-bit Windows machine with an 8 GB RAM capability. All algorithms are implemented using the Python programming language. The performance analysis is measured quantitatively and experimental result has been shown graphically as well for comparative analysis of performance of the algorithms. The programs were executed based on different input length of the messages (n). In the Table 1, the first column represents the length of the input string, second column represents the execution time taken by the existing ASCII based symmetric key algorithm [14], third column represents execution time taken by the existing ANN symmetric key algorithm, and fourth column represents execution time taken by the presented algorithm.

Table 1. Comparative execution time (in second) of existing and presented algorithm.

Input length (n)	ASCII algorithm [14]	ANN algorithm [7]	Presented TPM algorithm
100	0.141346693	0.091879121	0.010140181
300	0.142423391	0.071731341	0.008810997
500	0.147154093	0.077543721	0.033656120
700	0.135841131	0.110223311	0.075269222
900	0.141376019	0.092693821	0.034378767
1100	0.157432079	0.095462903	0.060044765
1300	0.162011862	0.113371165	0.052880526
1500	0.127288580	0.100365541	0.091072559
1700	0.139026880	0.101453907	0.078531981
1900	0.207770109	0.122415505	0.073513508
2100	0.155461311	0.115621658	0.075035334
2300	0.155148745	0.131752346	0.109886885
2500	0.147355556	0.121183224	0.101118565
2800	0.662622234	0.435288006	0.415288121
7300	0.826183156	0.372337094	0.332337453

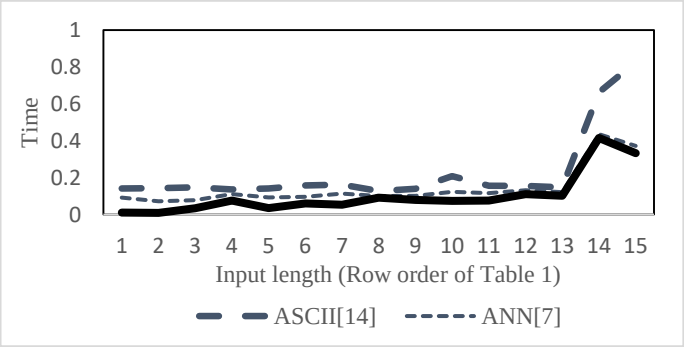

Fig. 4. Comparative chart for execution time of three algorithms given in Table 1.

5. Conclusion

The cryptography algorithm presented in this paper generated different cipher text every time for the same input string due to the dynamic nature of key generator, and padding with different special and non-printed characters. Due to this dynamism it is almost impossible for the intruder to detect the actual information even the information is passing through an insecure public channel. We believe in the very famous and well-known quote, "Morning shows the day". Our future target is to improve performance of the algorithm in a manner such that the complexity becomes, O (log n) or less.

Acknowledgment

I am sincerely thankful to the NSHM College of Management and Technology, NSHM Knowledge Campus, Kolkata, for providing me opportunity of the research work and necessary support in all aspect to complete the experiment.

References

1. B. F. Cruz, K. Domingo, F. D. Guzman, and J. Cotiangco, Int. J. Comp. Sci. Mobile Comput. **6**, 133 (2017). https://doi.org/10.13140/RG.2.2.36392.72969
2. S. Kumari, Int. J. Engg. Comp. Sci. **6**, 20915 (2017). https://doi.org/10.18535/ijecs/v6i4.20
3. P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, Proc. Comput. Sci. **78**, 617 (2016). https://doi.org/10.1016/j.procs.2016.02.108
4. A. D. Dwivedi, Sensors **21**, 1 (2021). https://doi.org/10.3390/s21175744
5. F. Thabit, S. Alhomdy, H. A. Al-Ahdal, and S. Jagpat, Global Transitions Proc. **2**, 91 (2021) https://doi.org/10.1016/j.gltp.2021.01.013
6. K. Balaji, and S. S. Manikandasaran, J. Sci. Res. **14**, 153 (2022).
7. S. K. Pal, B. Datta, and A. Karmakar - *Proc. Emerging Technologies in Data Mining and Inform. Security* (Springer Nature, Singapore) **3** (2021) pp. 47-56. https://doi.org/10.1007/978-981-15-9774-9_5
8. J. M. Padilla, U. M. Baese, and S. Foo, EURASIP J. Informat. Security **3** (2018). https://doi.org/10.1186/s13635-018-0073-z
9. S. A. Dar, Int. J. Modern Trends Engg. Res. **5**, 73 (2018). https://doi.org/10.21884/IJMTER.2018.5013.DAYGS
10. S. K. Pal and N. Chakraborty, Int. J. Comp. Network Informat. Security **5**, 11 (2017). https://doi.org/10.1016/S1353-4858(17)30103-4
11. S. K. Pal and S. De, Int. J. Comp. Network Informat. Security **7**, 50 (2015).
12. S. K. Pal and S. Mishra, Invertis J. Renewable Energy **9**, 43 (2019). https://doi.org/10.5958/2454-7611.2019.00008.0
13. S. K. Pal and S. Mishra, Int. J. Comp. Network Informat. Security **11**, 45 (2019). https://doi.org/10.5815/ijcnis.2019.10.06
14. U. Pujeri, and R. Pujeri, Int. J. Recent Technol. Engg. **8**, 2355 (2020). https://doi.org/10.35940/ijrte.E5980.018520