International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3631-3637
© Research India Publications. http://www.ripublication.com

Genetic Key Guided Neural Deep Learning based Encryption for Online Wireless Communication (GKNDLE)

Arindam Sarkar

Department of Computer Science & Electronics, Ramakrishna Mission Vidyamandira, Belur Math-711202, West Bengal, India.

Abstract

In this paper, a neural deep learning guided genetic secret key based technique for encryption (GKNDLE) has been proposed for online transmission of data/information through wireless communication. In this approach both the communicating networks receive an indistinguishable input vector, produce an output bit and are trained based on the output bit. Synchronized weight vectors become the chromosome of the 1st generation Genetic Algorithm based secret key used to encrypt the plain text. Intermediate encrypted stream goes through another triangular cipher process. A secret key is also fabricated in the process of encryption as cascaded manner. This intermediate cipher text is again encrypted to form the final cipher text through chaining and cascaded xoring of identical weight vector with the identical length intermediate cipher text block. If size of the last block of intermediate cipher text is less than the size of the key then this block is kept unchanged. Receiver will use identical weight vector for performing deciphering process for getting the triangular encrypted cipher text and secret key for decoding. Finally GA secret key helps to generate ultimate plain text. A session key based transmission has also been proposed using 161-bit key format of 14 different segments [7,8]. Parametric tests are done and results are compared in terms of Chi-Square test, response time in transmission with some existing classical techniques, which shows comparable results for the proposed system.

Keywords: Deep Learning, weight vector, input vector, sub key, session key, chaining, wrapping technique.

INTRODUCTION

These days a variety of techniques are available to defend data and information from eavesdroppers [2-9, 11, 20, 21, 22]. Every algorithm has its own advantages and disadvantages. For Example in DES, AES algorithms [1] the cipher block length is inflexible. RBCMPCC [14] allow only one cipher block encoding. In RPSP algorithm [15] any intermediate blocks during its cycle taken as the encrypted block and this number of iterations acts as secret key.

The organization of this paper is as follows. Section II of the paper deals with the proposed synchronization and key generation technique and also random block length based cryptographic techniques. Session key based encryption technique has been discussed in section III. Example of this technique is given in section IV. Complexity of the algorithm is presented in section V. Results are presented in section VI. Analysis regarding various aspects of the technique has been presented in section VII. Conclusions are drawn in section VIII and that of references at end.

THE TECHNIQUE

In proposed technique Genetic key is used to encrypt the plain text. Then this intermediate GA encrypted stream goes through another round of triangular encryption process. Secret key is padded with the last block of intermediate triangular encrypted text. This intermediate and padded cipher text is again encrypted to form the final cipher text using chaining of cascaded xoring of intermediate cipher text with the identical neural secret key. Using the same weight vector receiver performs the reverse technique. Section II.A deals with key generation technique and that of section II.B and II.C describes the encryption and decryption techniques respectively.

A. Neural synchronization scheme & secret key

At the beginning of the transmission, a neural network based secret key generation is performed between receiver and sender. The same may be done through the private channel also or it may be done in some other time, which is absolutely free from encryption. Figure 1 shows the tree based generation process using random number of nodes (neurons) and the corresponding algorithm is given in two-sub section II.A.1 and II.A.2.

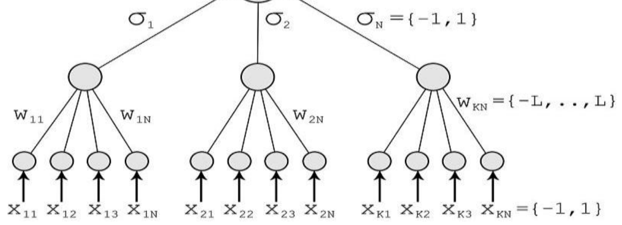
$\tau = \{-1, 1\}$
Π
σ_1
σ_2
$\sigma_N = \{-1, 1\}$
w_{11}
w_{1N}
w_{2N}
$w_{KN} = \{-L, .., L\}$
x_{11} x_{12} x_{13} x_{1N}
x_{21} x_{22} x_{23} x_{2N}
x_{K1} x_{K2} x_{K3} $x_{KN} = \{-1, 1\}$

Figure 1. A tree parity machine with K=3 and N=4

.A.1. Neural Synchronization Algorithm

Input: - Random weights, input vectors for both neural networks

Output: - Secret key through synchronization of input and output neurons as vectors.

Method: - Each party (A and B) uses its own (same) tree parity machine. Neural network parameters: K, N, L values will be identical for both parties and synchronization of the tree parity machines is achieved.

Parameters:

K - The number of hidden neurons.

N - The number of input neurons connected to each hidden neuron, total (K*N) input neurons.

L - The weight value {-L..+L}

Step 1. Initialize random weight values.

Step 2. Repeat step 3 to 6 until the full synchronization is achieved, using hebbian-learning rules (eq. 3).

Step 3. Generate random input vector X. Inputs are generated by a third party (say the server) or one of the communicating parties (fig. 2)

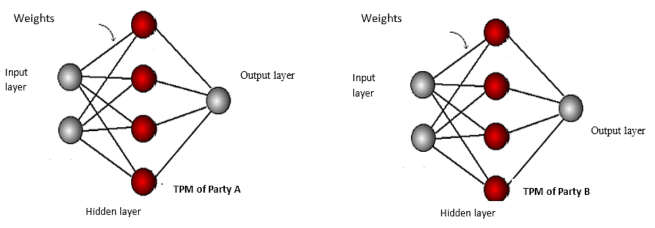

Figure 2. Neural network for sender and receiver

Step 4. Compute the values of the hidden neurons

$$\sigma_i = \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} x_{ij}\right) \quad \operatorname{sgn}(x) = \begin{cases} -1 & if \quad x < 0, \\ 0 & if \quad x = 0, \\ 1 & if \quad x > 0. \end{cases} \tag{1}$$

Step 5. Compute the value of the output neuron

$$\tau = \prod_{i=1}^{K} \operatorname{sgn}\left(\sum_{j=1}^{N} w_{ij} x_{ij}\right) \tag{2}$$

Step 6. Compare the output values of both tree parity machines by exchanging between the networks Outputs are not same: Go to step 3 if Output (A) ≠ Output (B) else if Output (A) = Output (B) then one of the suitable learning rule is applied to the weights. Update the weights only if the final output values of the neural machines are equivalent. In this paper we have used deep hebbian-learning rule for synchronization (eq. 3). When synchronization is finally occurred, the synaptic weights are same for both the networks. In each step there may be three possibilities:

$$w_i^{+} = w_i + \sigma_i x_i \theta(\sigma_i \tau) \theta(\tau^A \tau^B) \tag{3}$$

1. Output (A) ≠ Output (B): None of the parties updates its weights.

2. Output (A) = Output (B) = Output (E): All the three parties update weights in their tree parity machines. **3. Output (A) = Output (B) ≠ Output (E):** Parties A and B update their tree parity machines, but the attacker cannot do that. Because of this circumstance his learning is slower than the synchronization of parties A and B. The synchronization of two parties is faster than learning of an attacker.

Complexity: O(N) computational steps are needed to generate a key of length N. The average synchronization time up to N=1000 asymptotically one expects an increase like O (log N).

A.2. Final Key generation using Genetic Algorithm

The technique considers the synchronized weight vector from the in the form of blocks of bits with dissimilar size like 8/16/32/64/128/256 becomes the 1st chromosome of the 1st generation.

Steps of Random numbers generation (eq. 4) using genetic functions, i.e., crossover and mutation respectively.

Assumption about the generation of random numbers: -

Chromosomes representation in Binary, Population Size is 10 (could be varied), Hence 5 generations are required to generate 50chromosomes if the number of block of characters is 50.The sequence of random numbers (chromosome) is generated via the following iterative equation.

$$X_{n+1} = (a * X_{n+c}) \bmod m \tag{4}$$

Where, **Xn** denotes the randomly selected 8 bits synchronized weight vector which becomes first chromosome of the first generation, **m** -modulus (m > 0), **a -** multiplier (0 <= a < m),**c** -increment (0 <= c < m).

After crating the 1st generation of the chromosomes, next generation chromosomes are generated using the GA operators CROSSOVER followed by MUTATION.

In case of even generation crossover is taken place from top to bottom in a sequential way by pairing up chromosomes like (1,2) (3,4) (5,6).In case of odd generation crossover is take place from bottom to top in a sequential way by pairing up chromosomes like (50,49) (48,47) (46,45)

Consider 284, 7000, 9024, 4025, 1235, 2564, 654, 6526, 3652, 124, as first generation chromosomes.

2nd generation chromosomes are obtained by Pairing up like (284, 7000), (9024, 4025), (1235, 2564), (654, 6526), (3652, 124)

Here, crossover and mutation will be perform let at 4th locus of the gene of chromosome.

Consider binary representation of chromosomes as

284= 0000100011100 7000= 1101101011000

The output stream on crossover is 0000**101011000** 1101**100011100**. The output generated on mutation is 000**1**101011000=856 110**0**100011100=6428 (Mutation of 4th bit). Up to 5th generation same technique is use to generate 50 chromosomes (Each generation 10 populations) and get the final set of numbers.

Encryption

1. Once all the numbers are generated then let this array of numbers be called GENETIC_ARRAY and select the LAST digit of each number from GENETIC _ARRAY and a new collection of numbers is generated and let this collection is called CODED_ARRAY.
2. Use this numbers from CODED _ARRAY sequentially for substituting on a one-to-one basis for the characters of the plain text.
3. Use ASCII values of the plain text characters and ADD the numbers of CODED_ARRAY from the ASCII values. For example the message "ARINDAM" the CIPHER TEXT will be calculated according to following method. LET GENETIC _ARRAY = {2365, 1211, 8526, 1258, 7639, 4584,3982}

Table I: Genetic Algorithm Based Encryption

Character	ASCII Value	GENETIC_ARRAY Number Taken sequentially	Addition	Result
A	65	5	65+5	70(F)
R	82	1	82+1	83(S)
I	73	6	73+6	79(O)
N	78	8	78+8	86(V)
D	68	9	68+9	77(M)
A	65	4	65+4	69(E)
M	77	2	77+2	79(O)

The intermediate cipher text is "FSOVMEO" from table I. Now this intermediate encrypted stream is consider as an input source block to the next level of encryption procedure.

B.Ttraingularization

During encryption, consider a block S = $s^0_0\ s^0_1\ s^0_2\ s^0_3\ s^0_4\ s^0_5 \dots s^0_{n-2}\ s^0_{n-1}$ of size n bits, where $s^0_i = 0$ or 1 for $0 <= i <= (n-1)$. Now, starting from MSB (s^0_0) and the next-to-MSB (s^0_1), bits are pair-wise XORed, so that the 1st intermediate sub-stream $S^1 = s^1_0\ s^1_1\ s^1_2\ s^1_3\ s^1_4\ s^1_5 \dots s^1_{n-2}$ is generated consisting of (n-1) bits, where $s^1_j = s^0_j \oplus s^0_{j+1}$ for $0 <= j <= n-2$, $\oplus$ stands for the exclusive OR operation. This 1st intermediate sub-stream S^1 is also then pair-wise XORed to generate $S^2 = s^2_0\ s^2_1\ s^2_2\ s^2_3\ s^2_4\ s^2_5 \dots s^2_{n-3}$, which is the 2nd intermediate sub-stream of length (n-2). The method iterates (n-1) times to produce $S^{n-1} = s^{n-1}_0$, which is a single bit only. So, the size of the 1st midway sub-stream is one bit fewer than the source sub-stream; the size of each of the intermediate sub-streams starting from the 2nd one is one bit less than that of the sub- stream wherefrom it was generated; and finally the size of the final sub-stream in the process is one bit less than the final intermediate sub-stream. Figure 3 shows the process.

Table II: Options for choosing Target Block from Triangle

Option Serial No.	Target Block	Method of Formation
001	$s^0_0\ s^1_0\ s^2_0\ s^3_0\ s^4_0\ s^5_0 \dots s^{n-2}_0\ s^{n-1}_0$	Taking all the MSBs starting from the source block till the last block generated
010	$s^{n-1}_0\ s^{n-2}_0\ s^{n-3}_0\ s^{n-4}_0\ s^{n-5}_0 \dots s^1_0\ s^0_0$	Taking all the MSBs starting from the last block generated till the source block
011	$s^0_{n-1}\ s^1_{n-2}\ s^2_{n-3}\ s^3_{n-4}\ s^4_{n-5} \dots s^{n-2}_1\ s^{n-1}_0$	Taking all the LSBs starting from the source block till the last block generated
100	$s^{n-1}_0\ s^{n-2}_1\ s^{n-3}_2\ s^{n-4}_3\ s^{n-5}_4 \dots s^1_{n-2}\ s^0_{n-1}$	Taking all the LSBs starting from the last block generated till the source block

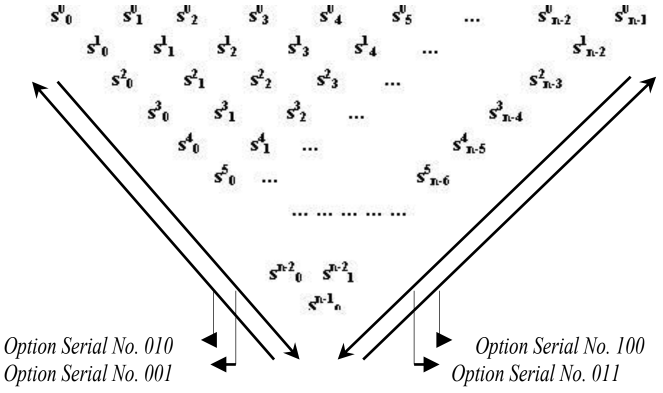
s^0_0 s^0_1 s^0_2 s^0_3 s^0_4 s^0_5 … s^0_{n-2} s^0_{n-1}
s^1_0 s^1_1 s^1_2 s^1_3 s^1_4 … s^1_{n-2}
s^2_0 s^2_1 s^2_2 s^2_3 … s^2_{n-3}
s^3_0 s^3_1 s^3_2 … s^3_{n-4}
s^4_0 s^4_1 … s^4_{n-5}
s^5_0 … s^5_{n-6}
… … … … …
s^{n-2}_0 s^{n-2}_1
s^{n-1}_0
Option Serial No. 010
Option Serial No. 001
Option Serial No. 100
Option Serial No. 011

Figure 3. Diagrammatic Representation of Options for choosing Target Block from Triangle

Table II is for options for choosing target block from triangle. Now size of each encrypted block under consideration and options chosen for this particular block merged together. In this way a intermediate key is formed and padded with the encrypted block. Then the neural secret key is use to xored with the same length first intermediate cipher text block to produce the first final cipher block (neural secret key XOR with same length cipher text). This newly generated block again xored with the immediate next block and so on. This chaining of cascaded xoring mechanism is performed until all the blocks get exhausted. If the last block size of intermediate cipher text is less than the require xoring block size (i.e. weight vector size) then this block is kept unchanged

C. Decryption Technique

During decryption, the encrypted message is converted into the corresponding stream of bits and then this stream is decomposed into a finite set of blocks, each consisting of a finite set of bits using same rule of encryption. It is to be noted that the receiver will take identical weight vectors and same neural key generation algorithm is use to generate the neural key. Then cascaded xoring operation is performed using identical neural secret key with the cipher text. The technique of performing xoring is same that was in encryption process. Finally from the outcomes intermediate encrypted block (E) and key block is extracted and now key is use to decipher the E to get the intermediate source stream. To ease the explanation of decryption technique, let us consider, $e^0_{i-1} = s^{i-1}_{n-i}$ for $1 <= i <= n$, so that the encrypted block becomes $E = e^0_0\ e^0_1\ e^0_2\ e^0_3\ e^0_4\ \ldots\ e^0_{n-2}\ e^0_{n-1}$. Now, following the same approach as mentioned in section 3.2.1, a triangle is to be formed. After the formation of the triangle, for the purpose of decryption, the block $e^{n-1}_0\ e^{n-2}_0\ e^{n-3}_0\ e^{n-4}_0\ e^{n-5}_0\ \ldots\ e^1_0\ e^0_0$, i.e., the block constructed by taking all the MSBs of the blocks starting from the finally generated single-bit block E^{n-1} to E, are to be taken together and it is to be considered as the decrypted block. Figure 4 show the triangle generated and hence the decrypted block obtained. Here the intermediate blocks are referred to as E^1, E^2, …, E^{n-2} and the final block generated as E^{n-1}.

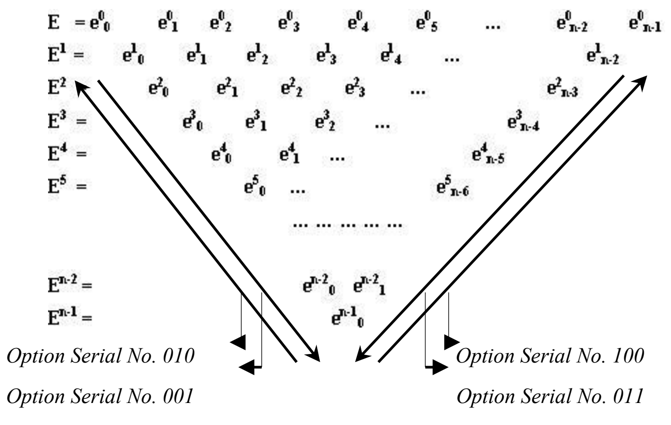

Figure 4. Generation of Source Block from Target

Finally, table III Genetic Algorithm based decryption procedure is use to get original plain text.

Table III: Genetic Algorithm Based Decryption

Result (R)	**For loop I =0 to 255 Do (I – R) & Compare With CODED _ARRAY & Choose I**	**Character**
70	65	A
83	82	R
79	73	I
86	78	N
77	68	D
69	65	A
79	77	M

SESSION KEY GENERATION

To ensure secured encryption of the proposed technique with varying size of blocks, a 161 bit key format consisting of 14 different segment has been proposed here [7,8]. The section of rank the R, there can exist a utmost of N=218-R blocks, each of unique size of S=218-R, R starting from 1 and moving till 14.Full input is divided into blocks of various sizes based on the session key.

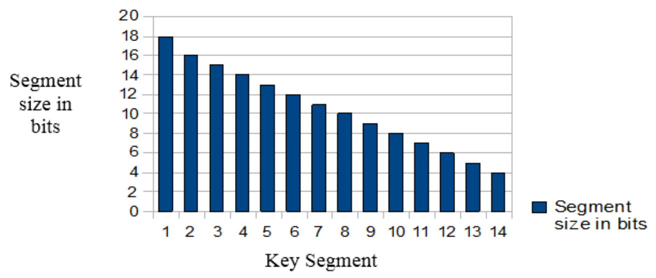

Figure 5. 161 bit key format

- Segment with R=1 formed with the first maximum 131072 blocks, each of size 131072 bits
- Segment with R=2 formed with the first maximum 65536 blocks, each of size 65536 bits
- Segment with R=3,4,5,----,14 formed with the next maximum 32768, 16384,8192,----,16 blocks, each of size 32768, 16384,8192,-------,16 bits respectively.

With such a structure, the key space becomes 147 bits long and a stream of the maximum size 2.6 GB can be encrypted using the proposed technique. The key structure is represented in Figure 5.This session key may be used during each session as a random manner to enhance the security of wireless communication.

EXAMPLE

In this section, we consider Genetic Algorithm based encrypted stream as a source stream. Section IV.A shows how the GA based encrypted text is further encrypted and section IV.B describes the process of decryption.

A The Process of Encryption:

1 0 0 1 0 1 0 1

1 0 1 1 1 1 1

1 1 0 0 0 0

0 1 0 0 0

1 1 0 0

0 1 0

1 1

0

Figure 6. Formation of Triangle for S = 10010101

Figure 6 shows the triangle formation of 1st 8 bits source block.

Table IV: Different Target Blocks for S = 10010101

Source Block S	Target Block Corresponding to Serial No.	Target Block T
10010101	001	T_1 =11101010
	010	T_2 =01010111
	011	T_3 =11000010
	100	T_4 =01000011

Table IV shows that different target (decrypted) block for a particular source block depending on different option serial no (001,010,011,100). For different 8/16/32/64/128 bit blocks different option serial no can be chosen.

B The Process of Decryption:

For target blocks T_1 = 11101010 and T_4 = 01000011, the same approach is to be followed to generate the corresponding source blocks. But blocks T_2 = 01010111 and T_3 = 11000010 require different techniques. Figure 7 show the generations of source block from target blocks T_2.

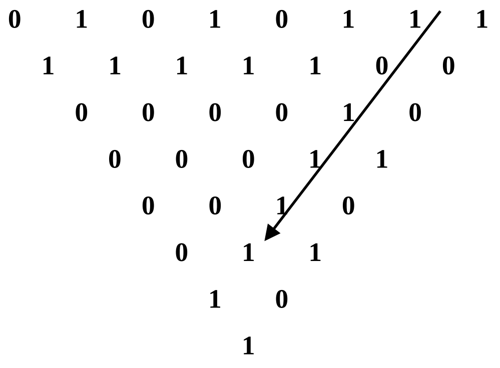

Figure7. Source Block S = 10010101 from Target Block T_2 = 01010111

After stuffing the key with the above midway cipher text and xoring with the neural secret key, final encrypted cipher text is generated. At the destination point, receiver's secret neural key is use to xoring the cipher text to get back the key and intermediate cipher text. Now, by that secret key, the receiver gets the information on different block lengths.In this way all the source blocks of bits are regenerated and combining those blocks in the same sequence and performing GA based decryption the source stream of bits are obtained to get the source message or the plaintext.

RESULTS

In this section the results of implementation of the proposed technique has been presented in terms of encryption decryption time, Chi-Square test, source file size vs. encryption time along with source file size vs. encrypted file size. The results are also compared with existing RSA technique.

Table V: ENCRYPTION / DECRYPTION TIME VS. FILE SIZE

Encryption Time (s)			Decryption Time (s)		
Source Size (bytes)	**Proposed ANNGKE**	**RPSP**	**Encrypted Size (bytes)**	**Proposed ANNGKE**	**RPSP**
18432	5. 76	7.85	18432	5.92	7.81
23044	8. 51	10.32	23040	7.73	9.92
35425	14. 23	15.21	35425	13. 87	14.93
36242	14. 98	15.34	36242	14. 91	15.24
59398	23. 40	25.49	59398	22. 98	24.95

Table V shows encryption and decryption time with respect to the source and encrypted size respectively. It is also observed the alternation of the size on encryption.

In figure 8 stream size is represented along X axis and encryption / decryption time is represented along Y-axis. This graph is not linear, because of different time requirement for finding appropriate random number for different block size. The decryption time is approximately linear, since there is no random number generation during decryption.

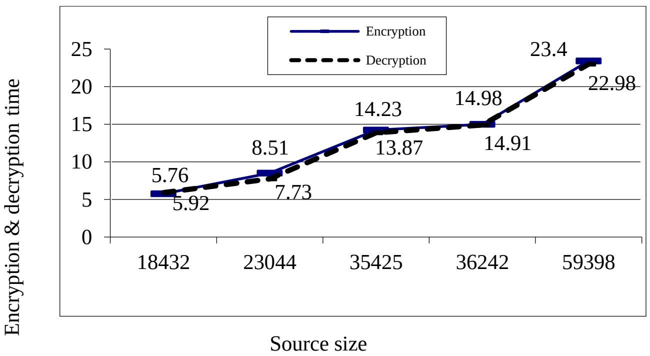

Figure 8. Source size vs. encryption time & decryption time

Table VI shows Chi-Square value for different source stream size after applying different encryption algorithms. It is seen that the Chi-Square value of ANNGKE is better compared to the algorithm RBCMPCC and comparable to the Chi-Square value of the RSA algorithm.

Table VI: Source size vs. Chi-Square value

Stream Size (bytes)	**Chi-Square value (TDES)**	**Chi-Square value (Proposed ANNGKE)**	**Chi-Square value (RBCM CPCC)**	**Chi-Square value (RSA)**
1500	1228.5803	2627.7534	2464.0324	5623.14
2500	2948.2285	5719.8522	5642.5835	22638.99
3000	3679.0432	6739.73621	6714.6741	12800.355
3250	4228.2119	7009.2813	6994.6189	15097.77
3500	4242.9165	11624.2315	10570.4671	15284.728

Figure 9 shows graphical representation of table VI.

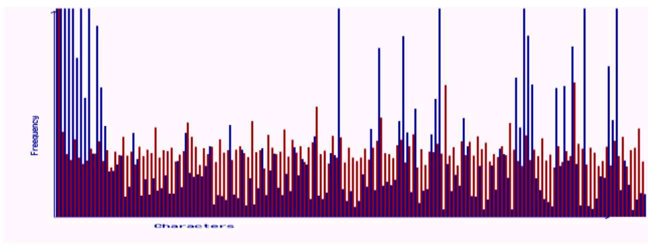

Figure 9. Source size vs. Chi-Square value

COMPLEXITY OF THE TECHNIQUE

The proposed technique has a complexity of O(L), which can be computed using following three steps.

Step 1: To generate a key of length N needs O(N) Computational steps. The average synchronization time is almost independent of the size N of the networks, at least up to N=1000.Asymptotically one expects an increase like O (log N).

Step 2: Complexity of the encryption technique is O(L).

Step2.1:Genetic Algorithm based encryption technique of the source block S = $s_0\ s_1\ s_2\ s_3\ s_4 \ldots\ s_{L-1}$, takes O (L) amount of time.

Step 2.2:Triangular encryption process takes O(L).

Step 2.3: Neural weight encryption technique is

carried out which generates the complexity as O (L).

Step 3: Complexity of the decryption technique is O(L).

Step 3.1: Neural weight decryption technique. So time complexity is O(L).

Step 3.2: Triangular decryption process takes O(L).

Step3.3:In Genetic Algorithm based decryption technique, complexity to convert T into the corresponding stream of bits S = $s_0\ s_1\ s_2\ s_3\ s_4 \ldots s_{L-1}$, which is the source block is O(L) as this step also takes constant amount of time for merging $s_0\ s_1\ s_2\ s_3\ s_4 \ldots s_{L-1}$. So, overall time complexity of the entire technique is O(L).

ANALYSIS OF RESULT

From experimental results it is clear that the proposed technique may achieve optimal performances. Encryption time and decryption time varies almost linearly with respect to the block size For the proposed algorithm Chi-Square value is very high compared to some existing algorithms. In the RBCMCPCC [14] a user input key (minimum length of eight byte) and a set of arbitrarily generated session keys are used to encrypt a source block. But this key generation technique is not sustainable against attack. Because RBCMCPCC key potency thoroughly depend upon strengthness of user input key. A user input key has to transmit over the public channel all the way to the receiver for performing the decryption procedure. So there is a likelihood of attack at the time of key exchange. To defeat this insecure secret key generation technique a neural network based secret key generation technique has been proposed. The security issue of RBCMCPCC and RPSP [15] algorithm can be improved by using neural secret sub key generation technique. In this case, the two partners A and B do not have to share a common secret but use their indistinguishable weights as a secret key needed for encryption. The fundamental conception of neural network based key exchange protocol [10,12,13,16,17,18,19] focuses mostly on two key attributes of neural networks. Firstly, two nodes coupled over a public channel will synchronize even though each individual network exhibits disorganized behavior. Secondly, an outside network, even if identical to the two communicating networks, will find it exceptionally difficult to synchronize with those parties, those parties are communicating over a public network. An attacker E who knows all the particulars of the algorithm and records through this channel finds it thorny to synchronize with the parties, and hence to calculate the common secret key. Synchronization by mutual learning (A and B) is much quicker than learning by listening (E) [17]. For usual cryptographic systems, we can improve the safety of the protocol by increasing of the key length. In the case of neural cryptography, we improve it by increasing the synaptic depth L of the neural networks. For a brute force attack using K hidden neurons, K*N input neurons and boundary of weights L, gives (2L+1)KN possibilities. For example, the configuration K = 3,L = 3 and N = 100 gives us 3*10253 key possibilities, making the attack unfeasible with today's computer power. E could start from all of the (2L+1)3N initial weight vectors and calculate the ones which are consistent with the input/output sequence. It has been shown, that all of these initial states move towards the same final weight vector, the key is unique. This is not true for simple perceptron the most unbeaten cryptanalysis has two supplementary ingredients first; a group of attacker is used. Second, E makes extra training steps when A and B are quiet. So increasing synaptic depth L of the neural networks we can make our neural network safe.

CONCLUSIONS

This paper presents a novel approach for generation of secret key proposed algorithm using neural cryptography. This technique enhances the security features of the RBCMCCC and RPSP algorithm by increasing of the synaptic depth L of the neural networks. In this case, the two partners A and B do not have to exchange a common secret key over a public channel but use their indistinguishable weights as a secret key needed for encryption or decryption. So likelihood of attack proposed technique is much lesser than the simple RBCMCCC algorithm.

REFERENCES

[1] Kahate, A. (2010). *Cryptography and Network Security*, 2nd edition, Tata McGraw Hill.

[2] Diffie, W., & Hellman, M. (1976). New directions in cryptography, *IEEE Trans. Inform. Theory, 22*(6), pp.644-654.

[3] Zhang, R., Shen, J., Wei, F., Li, X., & Sangaiah, A. K. (2017). Medical image classification based on multi-scale non-negative sparse coding. Artificial Intelligence in Medicine.

[4] Anurag Roy and Asoke Nath, “DNA Encryption Algorithms: Scope and Challenges in Symmetric Key Cryptography”, IJIRAE 2016.

[5] Kalsi, S., Kaur, H. & Chang, V. J Med Syst (2018) 42: 17. https://doi.org/10.1007/s10916-017-0851-z, Springer US, ISSN: 0148-5598.

[6] Liao, X., Yin, J., Guo, S., Li, X., & Sangaiah, A. K. (2017). Medical JPEG image steganography based on preserving interblock dependencies. Computers & Electrical Engineering.

[7] Anusudha, K., Venkateswaran, N. & Valarmathi, J. Multimed Tools Appl (2017) *76* (2), pp 2911–2932, https://doi.org/10.1007/s11042-015-3213-1, Springer US, Print ISSN 1380-7501, Online ISSN 1573-7721

[8] Al-Haj, A., Mohammad, A. & Amer(2017) A. J Digit Imaging *30*(1) , pp 26–38https://doi.org/10.1007 /s10278-016-9901-1

[9] Chen, C., Xiang, H., Qiu, T., Wang, C., Zhou, Yang., Chang, V. A rear-end collision prediction scheme based on deep learning in the Internet of Vehicles, Journal of Parallel and Distributed Computing, 2017.

[10] Sarkar, A., & Mandal, J. K. (2014). Cryptanalysis of Key Exchange method in Wireless Communication (CKE). *International Journal of Network Security (IJNS)*, 17(4), 484-493, ISSN 1816 – 3548 [Online]; 1816 – 353X [Print].

[11] Sarkar, A., & Mandal, J. K. (2014). Computational Science guided Soft Computing based Cryptographic Technique using Ant Colony Intelligence for Wireless Communication (ACICT). *International Journal of Computational Science and Applications (IJCSA)*, *4*(5), 61-73, DOI: 10.5121/ijcsa.2014.4505, ISSN 2200 – 0011.

[12] Sarkar, A., & Mandal, J. K. (2014). Soft Computing based Cryptographic Technique using Kohonen's Self-Organizing Map Synchronization for Wireless communication (KSOMSCT). *International Journal in Foundations of Computer Science & Technology (IJFCST), 4*(5), 85-100, DOI: 10.5121/ijfcst.2014.4508, ISSN 1839 – 7662.

[13] Sarkar, A., & Mandal, J. K. (2013). Computational Intelligence based Triple Layer Perceptron Model Coordinated PSO guided Metamorphosed based Application in Cryptographic Technique for Secured Communication (TLPPSO). In *Proceedings of the First International Conference on Computational Intelligence: Modeling, Techniques and Applications (CIMTA-2013)*, Vol.10, pp. 433-442, DOI: 10.1016/j.protcy.2013.12.380, ISSN: 2212-0173, September 27-28 2013, Department of Computer Science & Engineering, University of Kalyani, Kalyani, India, Procedia Technology, Elsevier.

[14] Jayanta Kumar Pal, J. K. Mandal “A Random Block Length Based Cryptosystem through Multiple Cascaded Permutation Combinations and Chaining of Blocks”Fourth International Conference on Industrial and Information Systems, ICIIS 2009, 28 – 31December2009, SriLanka.

[15] J. K. Mandal, et al, “A 256-bit Recursive Pair Parity Encoder for Encryption”, Advances in Modeling, D; Computer Science & Statistics (AMSE), vol. 9, No. 1, pp. 1-14, France, 2004.

[16] T. Godhavari, N. R. Alainelu and R. Soundararajan “Cryptography Using Neural Network ” IEEE Indicon 2005 Conference, Chennai, India, 11-13 Dec. 2005.

[17] Wolfgang Kinzel and ldo Kanter, "Interacting neural networks and cryptography", Advances in Solid State Physics, Ed. by B. Kramer (Springer, Berlin. 2002), Vol. 42, p. 383 arXiv- cond-mat/0203011.

[18] Wolfgang Kinzel and ldo Kanter, "Neural cryptography" proceedings of the 9th international conference on Neural Information processing(ICONIP 02).

[19] Dong Hu "A new service based computing security model with neural cryptography"IEEE07/2009.

[20] Jha P. K., Mandal, J. K., “Encryption Through Cascaded Recursive Key Rotation of a Session Key withTransposition and Addition of Blocks (CRKRTAB)”Proceedings of National conference on Recent Trends in Information Systems 2006, organized by IEEE Kolkata section, Jadavpur University, Kolkata held on July 14-15th, 2006. pp 69-72.

[21] Jha P. K., Nayak, A.K., “An Encryption Technique through Bitwise Operation of Blocks (BOB)” accepted tothe 97th Indian Science Congress, to be held in Kerala, Jan 2010, India.

[22] Saurabh Dutta, Ph. D. Thesis “An Approach Towards Development of Efficient encryption Techniques” The University of North Bengal, India, 2004.