GROUP SESSION KEY EXCHANGE MULTILAYER PERCEPTRON BASED SIMULATED ANNEALING GUIDED AUTOMATA AND COMPARISON BASED METAMORPHOSED ENCRYPTION IN WIRELESS COMMUNICATION (GSMLPSA)

Arindam Sarkar[1] and J. K. Mandal[2]

[1]Department of Computer Science & Engineering, University of Kalyani, W.B, India
arindam.vb@gmail.com
[2]Department of Computer Science & Engineering, University of Kalyani, W.B, India
jkm.cse@gmail.com

ABSTRACT

In this paper, a group session Key Exchange multilayer Perceptron based Simulated Annealing guided Automata and Comparison based Metamorphosed encryption technique (GSMLPSA) has been proposed in wireless communication of data/information. Both sender and receiver uses identical multilayer perceptron and depending on the final output of the both side multilayer perceptron, weights vector of hidden layer get tuned in both ends. As a results both perceptrons generates identical weight vectors which is consider as an one time session key. In GSMLPSA technique plain text is encrypted using metamorphosed code table for producing level 1 encrypted text. Then comparison based technique is used to further encerypt the level 1 encrypted text and produce level 2 encrypted text. Simulated Annealing based keystream is xored with the leve2 encrypted text and form a level 3 encrypted text. Finally level 3 encrypted text is xored with the MLP based session key and get transmitted to the receiver. GSMLPSA technique uses two keys for encryption purpose. SA based key get further encrypted using Automata based technique and finally xored with MLP based session key and transmitted to the receiver. This technique ensures that if intruders intercept the key of the keystream then also values of the key not be known to the intruders because of the automata based encoding. Receiver will perform same operation in reverse order to get the plain text back. Two parties can swap over a common key using synchronization between their own multilayer perceptrons. But the problem crop up when group of N parties desire to swap over a key. Since in this case each communicating party has to synchronize with other for swapping over the key. So, if there are N parties then total number of synchronizations needed before swapping over the actual key is $O(N^2)$. GSMLPSA scheme offers a novel technique in which complete binary tree structure is follows for key swapping over. Using proposed algorithm a set of N parties can be able to share a common key with only $O(\log_2 N)$ synchronization. Parametric tests have been done and results are compared with some existing classical techniques, which show comparable results for the proposed technique.

KEYWORDS

Metamorphosed, Automata, Session Key

1. INTRODUCTION

The sturdiness of the key is calculated in terms of linear complexity, randomness and correlation immunity. To devise a key following 4 basic features like large period of key, large linear

complexity of the key, good random key sequence, high order of correlation immunity of the key sequence is required. The most important hazard for private key cryptography is how to firmly swap over the shared secrets between the parties. As a result, key exchange protocols are mandatory for transferring private keys in a protected manner. The first available key exchange protocol is known as Diffie-Hellman key exchange and it depends on the stiffness of computing discrete logarithms [1]. These days a range of techniques are available to preserve data and information from eavesdroppers [1]. Each algorithm has its own advantages and disadvantages. Security of the encrypted text exclusively depends on the key used for encryption. In cryptography the main security intimidation is man-in-the-middle attack at the time of exchange the secret session key over public channel.

In recent times wide ranges of techniques are available to protect data and information from eavesdroppers [1, 2, 3, 4, 5, 6, 7, 8, 9]. These algorithms have their virtue and shortcomings. For Example in DES, AES algorithms [1] the cipher block length is nonflexible. In NSKTE [4], NWSKE [5], AGKNE [6], ANNRPMS [7] and ANNRBLC [8] technique uses two neural network one for sender and another for receiver having one hidden layer for producing synchronized weight vector for key generation. Now attacker can get an idea about sender and receiver's neural machine because for each session architecture of neural machine is static. In NNSKECC algorithm [9] any intermediate blocks throughout its cycle taken as the encrypted block and this number of iterations acts as secret key. If n number of iterations are needed for cycle formation and if intermediate block is chosen as an encrypted block after $n/2^{th}$ iteration then exactly same number of iterations i.e. n/2 are needed for decode the block which makes easier the attackers life. As the same time as key exchange protocols are developed for exchanging key between two parties, many applications do necessitate the need of swapping over a secret key among group of parties. A lot of proposals have been proposed to accomplish this goal. As Multilayer perceptron synchronization proposal is a fresh addition to the field of cryptography, it does not provide a group key exchange mechanism. To solve these types of problems in this paper we have proposed a key swap over mechanism among cluster of multilayer perceptrons for encryption/decryption in wireless communication of data/information. This scheme implements the key swap over algorithm with the help of complete binary tree which make the algorithm scales logarithmically with the number of parties participating in the protocol. In this paper, additional sub key generation mechanism has been also proposed using local random search algorithm i.e. Simulated Annealing (SA) for encryption. This SA based key generation methods generates key which satisfies all 4 basic features. Proposed technique is also better than SATMLP [17] algorithm because of Metamorphosed guided Automata based comparison encryption rather than simple triangularized encryption in case of SATMLP [17].

The organization of this paper is as follows. Section 2 and 3 of the paper deals with character code table and metamorphosed code generation. Proposed encryption and decryption has been discussed in section 4 and 5. SA based key generation discussed in section 6. Section 7 deals with MLP based session key generation. GSMLPSA password based certificate generation scheme is given in section 8 and 9. Complexity analysis of the technique is given in section 10. Experimental results are described in section 11. Analysis of the results presented in section 12. Analysis regarding various aspects of the technique has been presented in section 13. Conclusions and future scope are drawn in section 14 and that of references at end.

2. CHARACTER CODE TABLE GENERATION

For plain text "tree" figure 1 shows corresponding tree representation of probability of occurrence of each character in the plain text. Characters't' and 'r' occur once and character 'e' occurs twice. Each character code can be generated by travelling the tree using preorder traversal. Character

values are extracted from the decimal representation of character code. Left branch is coded as '0' and that of right branch '1'. Table 1 shows the code and value of a particular character in the plain text.

Table1. Code table

Plain text	Code	Value
t	10	2
r	11	3
e	0	0

3. METAMORPHOSED CHARACTER CODE TABLE GENERATION

From the original tree metamorphosed tree is derived using mutation. Figure 2, 3 and 4 are the metamorphosed trees. After mutation new code values as obtained are tabulated in table 2. Tree having (n-1) intermediate nodes can generate 2^{n-1} mutated trees. In order to obtain unique value, the code length is added to the character if the value is identical in the table.

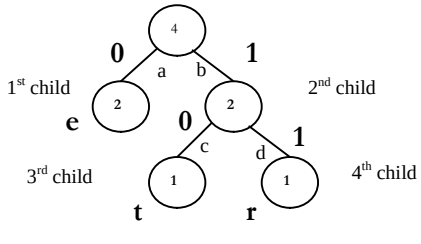


Figure 1. Character Code Tree

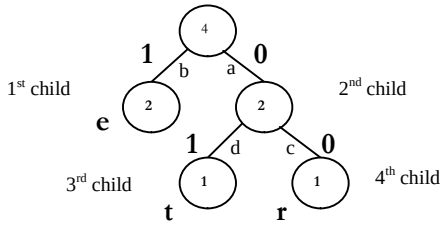


Figure 2. Swap the edges between a and b and between c and d.

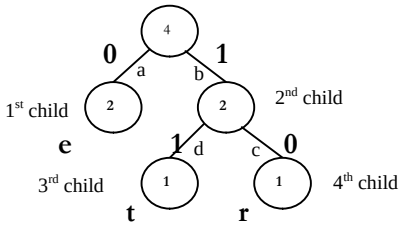


Figure 3. Swap the edges between c and d.
Edges a and b get unaltered.

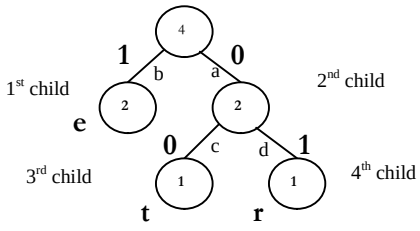


Figure 4. Swap the edges between a and b.
Edges c and d get unaltered.

Table2. Mutated code table

Character	Code	Value	Code	Value	Code	Value
t	01	1	11	3	00	0
r	00	0	10	2	01	1
e	1	2	0	0	1	2

4. ENCRYPTION ALGORITHM

Step 1. Plain texts are firstly encoded using the help of metamorphosed code table to get an level1 encoded text for enhancing the security perform the following comparisons based encryption for each block to get level2 encoded text.

Step 1.1 For the characters in the block perform following calculations. For the very first character in the block weight value of the previous character will be 0.

$$Char\_Weight = Positionof\_Char\_in\_Alphabet$$

$$\psi_{Char} = \frac{1}{Current\_Char\_Weight - Previous\_Char\_Weight} + \frac{\pi}{100}$$

$$Char\_Offset = \left| \psi_{first\_character} \times 10 \right|$$

$$Char\_Total\_Value = Char\_Offset \times Char\_Weight$$

Step 1.2 Calculate the Block _Value using following equation

$$Block\_Value = \left(\sum_{i=1}^{n} Char_i\_Total\_Value \right) - Length\_of\_Block$$

Step 1.3 Perform the following operation on each block depending on the block value.

$$if\ (0 \le Block\_Value < 100) then$$

Reverse the block

$$if\ (100 \le Block\_Value < 150) then$$

Block is circular right shifted by $\frac{(Length\_of\_Block)}{2}$

$$if\ (150 \le Block\_Value < 200) then$$

Block is circular left shifted by $\frac{(Length\_of\_Block)}{2}$

Step 2. Perform XOR operation between level 2 encoded plain text and SA based keystream to generate the level 3encoded text.

Step 3. Finally perform xor operation between MLP based session key and level 3 encoded plain text and then transfer to the receiver.

Step 4. Perform the Automata based encryption to get level1 encoded SA based keystream. This technique ensures that if intruders intercept the key of the keystream then also values of the key not be known to the intruders because of the Automata based encoding.

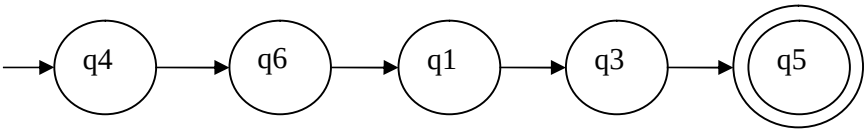
Step 4.1 Break the SA based keystream into 3 bit binary code and use the following table to represent value of each code and their octal, gray code representation and state number.

Table 3. Octal, gray code representation and state number table

Binary (3 bit)	Octal	Gray Code	State Number
000	0	000	q0
001	1	001	q1
010	2	011	q2
011	3	010	q3
100	4	111	q4
101	5	110	q5
110	6	100	q6
111	7	101	q7

For example if SA based key stream is in the form 100 110 001 011 101 then state representation will be Q= set of finite state i.e. {q4, q6, q1, q3, q5} where q4 is the initial state and q5 is the final state. State transition operator δ is use to denote transition from one state to another and $\sum$ denotes alphabet set. Then state transition is done using following representations:

$\delta(q4, tranmsition\_variable) = q6$ i.e. (q4 ⟶ q6)
$\delta(q6, tranmsition\_variable) = q1$ i.e. (q6 ⟶ q1)
$\delta(q1, tranmsition\_variable) = q3$ i.e. (q1 ⟶ q3)
$\delta(q3, tranmsition\_variable) = q5$ i.e. (q3 ⟶ q5)



Step 4.2 Perform the following operation

$$\{\max(q_4, q_6)\ in\ \_octal\_form \times \max(q_4, q_6)\ in\ \_gray\_code\} +$$
$$\{\max(q_6, q_1)\ in\ \_octal\_form \times \max(q_6, q_1)\ in\ \_gray\_code\} +$$
$$\{\max(q_1, q_3)\ in\ \_octal\_form \times \max(q_1, q_3)\ in\ \_gray\_code\} +$$
$$\{\max(q_3, q_5)\ in\ \_octal\_form \times \max(q_3, q_5)\ in\ \_gray\_code\}$$

Step 4.3 Result of this operation get converted in binary form and then get XOR-ed with SA based keystream value in reverse order to produce level1 encoded keystream.

Step 5. Perform xoring between MLP based session key and level 1 encoded key stream to produce level 2 based key and transmitted to the receiver.

5. DECRYPTION ALGORITHM

Step 1. Receive the cipher text and level2 based encoded keystream from the sender.
Step 2. Generate level1 based encoded keystream by performing xoring between MLP based session key and level 2 encoded key stream.
Step 3. Generate SA based keystream by performing Automata based decryption on level1 encoded keystream.

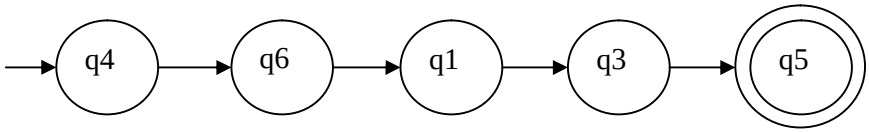
Step 3.1 Break the level1 encoded keystream into 3 bit binary code and use the following table to represent value of each code and their octal, gray code representation and state number.

Table 4. Octal, gray code representation and state number table

Binary (3 bit)	Octal	Gray Code	State Number
000	0	000	q0
001	1	001	q1
010	2	011	q2
011	3	010	q3
100	4	111	q4
101	5	110	q5
110	6	100	q6
111	7	101	q7

For example if level1 encoded key stream is in the form 100 110 001 011 101 then state representation will be Q= set of finite state i.e. {q4, q6, q1, q3, q5} where q4 is the initial state and q5 is the final state. State transition operator δ is use to denote transition from one state to another and $\sum$ denotes alphabet set. Then state transition is done using following representations:

$\delta(q4, tranmsition\_variable) = q6$ i.e. (q4 ⟶ q6)
$\delta(q6, tranmsition\_variable) = q1$ i.e. (q6 ⟶ q1)
$\delta(q1, tranmsition\_variable) = q3$ i.e. (q1 ⟶ q3)
$\delta(q3, tranmsition\_variable) = q5$ i.e. (q3 ⟶ q5)



Step 3.2 Now, perform the following operation

$$\{\max(q_4, q_6)\ in\_octal\_form \times \max(q_4, q_6)\ in\_gray\_code\} +$$
$$\{\max(q_6, q_1)\ in\_octal\_form \times \max(q_6, q_1)\ in\_gray\_code\} +$$
$$\{\max(q_1, q_3)\ in\_octal\_form \times \max(q_1, q_3)\ in\_gray\_code\} +$$
$$\{\max(q_3, q_5)\ in\_octal\_form \times \max(q_3, q_5)\ in\_gray\_code\}$$

Step 3.3 Result of this operation get converted in binary form and then get XOR-ed with level1 encoded keystream value in reverse order to produce SA based keystream.

Step 4. Perform xor operation between MLP encoded text and MLP based session key to generate level 3 encoded text.

Step 5. Perform xor operation between level 3 encoded text and SA based key to generate level 2 encoded text.

Step 6. Generate level1 based encoded plain text by performing Comparison based decryption on level2 encoded plain text

Step 6.1 For the characters in the block perform following calculations. For the very first character in the block weight value of the previous character will be 0.

$$Char\_Weight = Positionof\_Char\_in\_Alphabet$$

$$\psi_{Char} = \frac{1}{Current\_Char\_Weight - Previous\_Char\_Weight} + \frac{\pi}{100}$$

$$Char\_Offset = \left| \psi_{first\_character} \times 10 \right|$$

$$Char\_Total\_Value = Char\_Offset \times Char\_Weight$$

Step 6.2 Calculate the Block _Value using following equation

$$Block\_Value = \left(\sum_{i=1}^{n} Char_i\_Total\_Value \right) - Length\_of\_Block$$

Step 6.3 Perform the following operation on each block depending on the block value.

$$if\ (0 \le Block\_Value < 100)\ then$$

Reverse the block

$$if\ (100 \le Block\_Value < 150)\ then$$

Block is circular right shifted by $\frac{(Length\_of\_Block)}{2}$

$$if\ (150 \le Block\_Value < 200)\ then$$

Block is circular left shifted by $\frac{(Length\_of\_Block)}{2}$

Step 7. Generate plain text by performing metamorphosed character code based decryption on level1 encoded plain text

6. SIMULATED ANNEALING (SA) BASED KEY GENERATION

Simulated Annealing (SA) algorithm which is based on the analogy between the annealing of solids and the problem of solving combinatorial optimization problems SA provides local search technique that helps to escape from local optima. In this scheme SA is used to generate key after satisfying some desired features such as good statistical properties, long period, large linear complexity, and highly order degree of correlation immunity.

In proposed SA based key generation technique Section A deals with initialization procedure and that of section B describes the fitness calculation, section C describes cooling mechanism and finally section D deals with SA based key generation algorithm.

Initialization Procedure

At the preliminary state of SA, each sequence is represented as a binary string of an equal number of 0's and 1's of a given length. So, each generated keystream (solution) coded as a binary string.

Fitness Calculation

For each sequence fitness value is calculated examining the keystream. Using the following sequence of steps fitness is calculated:

Frequency test of 0's & 1's: Proportion of 0's & 1's in the total sequence are being checked using eq. $Frequency\_Fault = \left|\psi_0 - \psi_1\right|$ Ψ_0: No. of 0's in total sequence. Ψ_1: No. of 1's in total sequence.

Binary Derivative Test: Binary Dervative test is applied to the sequence by taking the overlapping 2 tupels in the original bit stream.

Table 5. Binary xor Operation Table

First Bit	Second Bit	Result
0	0	0
0	1	1
1	0	1
1	1	0

Here, an equal proportion of 1's and 0's in the new bit stream is checked by eq.

$$Binary\_Derivative\_Fault = \left|\left(C\_\psi_0 - C\_\psi_1\right) - 1\right|$$

Where, $C\_\Psi_{0:}$ Count no. of 0's in new bit stream. C_Ψ_1: Count no. of 1's in new bit stream.

Change Point Test: In this test a check point is created for observing maximum difference between, the proportions of 1's including the check point and the proportions of 1's after the check point using following eq.

$D[C_p] = \Psi * K[C_p] - C_{p*} K[\Psi]$

$$\rho_r = \exp\left(-2M^2/\psi * K[\psi] * (\psi - K[\psi])\right)$$

$$Change\_Point\_Fault = \rho_r$$

$\Psi_{:}$ Total no. of bit in the stream. $K[\Psi]$: Total no. of 1's in the bit stream. $C_{p:}$ Change Point. $K[C_p]$: Total no. of 1's to bit C_p (Change Point). $D[C_p]$: Difference respect to the Change point.
M: MAX (ABS ($D[C_p]$)), for $C_{p=1..}\Psi$.
ρ_r Probability of statistics that smaller value of ρ_r more significant the result. Finally, fault of every test is summed up for calculating fitness function. Using following eq. fitness is calculated.

$$Fitness\_Function = \frac{1}{1 + Fault}$$

Cooling Procedure

To determining the cooling schedule in case of optimization problem in annealing process requires some parameters for initial value of control parameter, decrement function of control parameter, length of individual distance parameter, stopping criteria

SA based Key Generation Algorithm

Step 1*: Set T as a starting temperature.Set Itr (no. of iteration) =0. Set Inner_Cycle=0, Set Finish=False.*
Step 2: *Randomly generate sequence of binary bits and calculate the fitness function.*
Where, $F_{current}$= fitness function of current state.
Step 3: *While Finish=False do (a) to (d) Repeat inner loop iteration n times.*
(a) Generate next sequence using swapping & flipping operators.
(b) Evaluate the fitness value F_{new} of the new sequence.
(c) Evaluate the energy changes in fitness function. $\Delta E = F_{new} - F_{current}$

(d) If $F_{new} > F_{current}$ *then new solution has a better fitness than the current one so accept the move.* $F_{current} = F_{new.}$

Else evaluate $\rho_r = \exp\left(\frac{-\Delta E}{T}\right)$

If ρ_r *> Threshold value then accept the move.*

*Else Reject. In the recent inner loop if no move is accepted then Inner_Cycle= Inner_Cycle+1; T=T*α; Itr=Itr+1; If (Inner_Cycle > Max_Cycle) or (Itr>Max_Itr) then Finish=True*

Step 4: $F_{current}$ *becomes the final solution.*

Now, the final solution i.e. stream of 0's & 1's becomes the SA based sub key for 2nd level encryption.

7. STRUCTURE OF MULTILAYER PERCEPTRON

In multilayer perceptron synchronization scheme secret session key is not physically get exchanged over public insecure channel. At end of neural weight synchronization strategy of both parties' generates identical weight vectors and activated hidden layer outputs for both the parties become identical. This identical output of hidden layer for both parties can be use as one time secret session key for secured data exchange. A multilayer perceptron synaptic simulated weight based undisclosed key generation is carried out between recipient and sender. Figure5 shows multilayer perceptron based synaptic simulation system. Sender and receivers multilayer perceptron select same single hidden layer among multiple hidden layers for a particular session. For that session all other hidden layers goes in deactivated mode means hidden (processing) units of other layers do nothing with the incoming input. Either synchronized identical weight vector of sender and receivers' input layer, activated hidden layer and output layer becomes session key or session key can be form using identical output of hidden units of activated hidden layer.

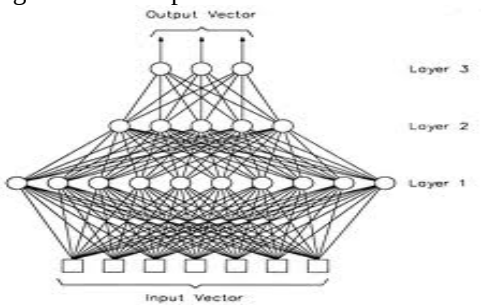


Figure 5. A Multilayer Perceptron with 3 Hidden Layers

Sender and receiver multilayer perceptron in each session acts as a single layer network with dynamically chosen one activated hidden layer and K no. of hidden neurons, N no. of input neurons having binary input vector, $x_{ij} \in \{-1,+1\}$, discrete weights, are generated from input to output, are lies between -L and +L, $w_{ij} \in \{-L,-L+1,\ldots,+L\}$. Where i = 1,…,K denotes the i^{th} hidden unit of the perceptron and j = 1,…,N the elements of the vector and one output neuron. Output of the hidden units is calculated by the weighted sum over the current input values . So, the state of the each hidden neurons is expressed using (eq.1)

$$h_i = \frac{1}{\sqrt{N}} w_i x_i = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} w_{i,j} x_{i,j} \quad (1)$$

Output of the i^{th} hidden unit is defined as

$$\sigma_i = \text{sgn}(h_i) \tag{2}$$

But in case of $h_i = 0$ then $\sigma_i = -1$ to produce a binary output. Hence a, $\sigma_i = +1$, if the weighted sum over its inputs is positive, or else it is inactive, $\sigma_i = -1$. The total output of a perceptron is the product of the hidden units expressed in (eq. 2)

$$\tau = \prod_{i=1}^{K} \sigma_i \tag{3}$$

The learning mechanism proceeds as follows ([6, 7]):

1. If the output bits are different, $\tau^A \neq \tau^B$, nothing is changed.
2. 2. If $\tau^A = \tau^B = \tau$, only the weights of the hidden units with $\sigma_k^{A/B} = \tau^{A/B}$ will be updated.
3. The weight vector of this hidden unit is adjusted using any of the following learning rules:

Anti-Hebbian:

$$W_k^{A/B} = W_k^{A/B} - \tau^{A/B} x_k \Theta(\sigma_k \tau^{A/B})(\tau^A \tau^B) \tag{4}$$

Hebbian :

$$W_k^{A/B} = W_k^{A/B} + \tau^{A/B} x_k \Theta(\sigma_k \tau^{A/B})(\tau^A \tau^B) \tag{5}$$

Random walk:

$$W_k^{A/B} = W_k^{A/B} + x_k \Theta(\sigma_k \tau^{A/B})(\tau^A \tau^B) \tag{6}$$

During step (2), if there is at least one common hidden unit with $\sigma_k = \tau$ in the two networks, then there are 3 possibilities that characterize the behaviour of the hidden nodes:

1. An attractive move: if hidden units at similar k positions have equal output bits, $\sigma_k^A = \sigma_k^B = \tau^{A/B}$

2. A repulsive move: if hidden units at similar k positions have unequal output bits, $\sigma_k^A \neq \sigma_k^B$

3. No move: when $\sigma_k^A = \sigma_k^B \neq \tau^{A/B}$

The distance between hidden units can be defined by their mutual overlap, ρ_k,

$$\rho_k = \frac{w_k^A w_k^B}{\sqrt{w_k^A w_k^A}\sqrt{w_k^B w_k^B}} \tag{7}$$

where $0 < \rho_k < 1$, with $\rho_k = 0$ at the start of learning and $\rho_k = 1$ when synchronization occurs with the two hidden units having a common weight vector.

7.1 Multilayer Perceptron Simulation Algorithm

Input: - Random weights, input vectors for both multilayer perceptrons.
Output: - Secret key through synchronization of input and output neurons as vectors.
Method:-

Step 1. *Initialization of random weight values of synaptic links between input layer and randomly selected activated hidden layer.*

$$\text{Where, } w_{ij} \in \{-L, -L+1 \ldots +L\} \tag{8}$$

Step 2. *Repeat step 3 to 6 until the full synchronization is achieved, using Hebbian-learning rules.*

$$w_{i,j}^{+} = g\left(w_{i,j} + x_{i,j}\tau\Theta(\sigma_i\tau)\Theta\left(\tau^A\tau^B\right)\right) \tag{9}$$

Step 3. *Generate random input vector X. Inputs are generated by a third party or one of the communicating parties.*

Step 4. *Compute the values of the activated hidden neurons of activated hidden layer using (eq. 10)*

$$h_i = \frac{1}{\sqrt{N}} w_i x_i = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} w_{i,j} x_{i,j} \tag{10}$$

Step 5. *Compute the value of the output neuron using*

$$\tau = \prod_{i=1}^{K} \sigma_i \tag{11}$$

Compare the output values of both multilayer perceptron by exchanging the system outputs.
if Output (A) ≠ Output (B), Go to step 3
else if Output (A) = Output (B) then one of the suitable learning rule is applied
only the hidden units are trained which have an output bit identical to the common output.

Update the weights only if the final output values of the perceptron are equivalent. When synchronization is finally achieved, the synaptic weights are identical for both the system.

7.2 Multilayer Perceptron Learning rule

At the beginning of the synchronization process multilayer perceptron of A and B start with uncorrelated weight vectors $w_i^{A/B}$. For each time step K, public input vectors are generated randomly and the corresponding output bits $\tau^{A/B}$ are calculated. Afterwards A and B communicate their output bits to each other. If they disagree, $\tau^A \neq \tau^B$, the weights are not changed. Otherwise learning rules suitable for synchronization is applied. In the case of the Hebbian learning rule [10] both neural networks learn from each other.

$$w_{i,j}^{+} = g\left(w_{i,j} + x_{i,j}\tau\Theta(\sigma_i\tau)\Theta\left(\tau^A\tau^B\right)\right) \tag{12}$$

The learning rules used for synchronizing multilayer perceptron share a common structure. That is why they can be described by a single (eq. 4)

$$w_{i,j}^{+} = g\left(w_{i,j} + f\left(\sigma_i, \tau^A, \tau^B\right)x_{i,j}\right) \tag{13}$$

with a function $f\left(\sigma_i, \tau^A, \tau^B\right)$, which can take the values -1, 0, or +1. In the case of bidirectional interaction it is given by

$$f\left(\sigma_i,\tau^A,\tau^B\right)=\Theta\left(\sigma\tau^A\right)\Theta\left(\tau^A\tau^B\right)\begin{cases}\sigma & \textit{Hebbian learning}\\ -\sigma & \textit{anti-Hebbian learning}\\ 1 & \textit{Random walk learning}\end{cases} \tag{14}$$

The common part $\Theta\left(\sigma\tau^A\right)\Theta\left(\tau^A\tau^B\right)$ of $f\left(\sigma_i,\tau^A,\tau^B\right)$ controls, when the weight vector of a hidden unit is adjusted. Because it is responsible for the occurrence of attractive and repulsive steps [6].

The equation consists of two parts:

1. $\Theta\left(\sigma\tau^A\right)\Theta\left(\tau^A\tau^B\right)$: This part is common between the three learning rules and it is responsible for the attractive and repulsive effect and controls when the weight vectors of a hidden unit is updated. Therefore, all three learning rules have similar effect on the overlap.
2. $(\sigma,-\sigma, 1)$: This part differs among the three learning rules and it is responsible for the direction of the weights movement in the space. Therefore, it changes the distribution of the weights in the case of Hebbian and anti-Hebbian learning. For the Hebbian rule, A's ad B's multilayer perceptron learn their own output and the weights are pushed towards the boundaries at $-L$ and $+L$. In contrast, by using the anti- Hebbian rule, A's and B's multilayer perceptron learn the opposite of their own outputs. Consequently, the weights are pulled from the boundaries $\pm L$. The random walk rule is the only rule that does not affect the weight distribution so they stay uniformly distributed. In fact, at large values of N, both Hebbian and anti-Hebbian rules do not affect the weight distribution. Therefore, the proposed algorithm is restricted to use either random walk learning rule or Hebian or anti-Hebbian learning rules only at large values of N. The random walk learning rule is chosen since it does not affect the weights distribution
   regardless of the value of N.

7.5 Hidden Layer as a Secret Session Key

At end of full weight synchronization process, weight vectors between input layer and activated hidden layer of both multilayer perceptron systems become identical. Activated hidden layer's output of source multilayer perceptron is used to construct the secret session key. This sessionkey is not get transmitted over public channel because receiver multilayer perceptron has same identical activated hidden layer's output. Compute the values of the each hidden unit by

$$\sigma_i=\operatorname{sgn}\left(\sum_{j=1}^{N}w_{ij}x_{ij}\right)\quad \operatorname{sgn}(x)=\begin{cases}-1 & if\ x<0,\\ 0 & if\ x=0,\\ 1 & if\ x>0.\end{cases} \tag{15}$$

For example consider 8 hidden units of activated hidden layer having absolute value (1, 0, 0, 1, 0, 1, 0, 1) becomes an 8 bit block. This 10010101 become a secret session key for a particular session and cascaded xored with recursive replacement encrypted text. Now final session key based encrypted text is transmitted to the receiver end. Receiver has the identical session key i.e. the output of the hidden units of activated hidden layer of receiver. This session key used to get the recursive replacement encrypted text from the final cipher text. In the next session both the machines started tuning again to produce another session key.

Identical weight vector derived from synaptic link between input and activated hidden layer of both multilayer perceptron can also becomes secret session key for a particular session after full weight synchronization is achieved.

8. THE GSMLPSA TECHNIQUE

Our proposed group key swap over technique offers two novel procedures for exchanging group key among different multilayer perceptron. Both procedures are based on the structure of complete binary tree. In the multilayer perceptron group key exchange algorithm, N multilayer perceptrons need to synchronize together and they are represented by an M number of leaves of a complete binary tree where M is defined as $M = 2^{(\log_2 N)}$.

Complete binary tree based proposed procedures are

a) Complete Binary Tree with Vote (CBTV)
b) Complete Binary Tree with Exchange (CBTE)

Both procedures have the same end results but their implementations are different.

8.1. Synchronization using Complete Binary Tree with Vote (CBTV)

In the CBTV method, the N multilayer perceptrons are represented by the M leaves. For every step j (starting at j = 1) of the algorithm the binary tree is divided into $\frac{M}{2^j}$ subtrees each with 2^j leaves. Then each pair of leaves sharing the same parent involved in mutual learning. Next, j is incremented and in each subtree, a node is nominated and the mutual learning algorithm is executed by the nominated nodes and the rest if the nodes follow. When the algorithm reaches the root, then it terminates and hence, all the multilayer perceptrons are synchronized and share the same weight vectors.

Algorithm 1 –The CBTV method

1: loop {$for\ j = \log_2 M; j \geq 0; j$ --}
2: Nominate a leader in each left and right subtree in level j
3: Apply Mutual learning between nominated leaders
4: Every leader sends the τ bits to its group.
5: end loop

8.2. Synchronization using Complete Binary Tree with Exchange (CBTE)

In the CBTE method, the mutual learning algorithm is take place between every two parties having the same parent in the binary tree structure. Let the $max\ depth$ is the depth of the complete binary tree and $cur\ depth$ is the current depth where the algorithm is functioning. Starting from a $cur\ depth = max\ depth-1$, apply the mutual learning algorithm between each pair of leaves having the same parent. Following the synchronization, one level up is marked ($cur\ depth = cur\ depth - 1$) and a exchange method is applied between the right leaves of both right and left branches for all subtrees in that $cur\ depth$. Once the $cur\ depth$ becomes equal to zero, all leaves will be synchronized together. For sake of simplicity the group of parties will be represented as vector with indices {0, 1, . . . , $M-1$} Fig.6 shows the scenario of synchronization. Fig.6a shows the preliminary configuration of unsynchronized parties. In Fig.6b, pairs of parties are synchronized together, {(0, 1), (2, 3), (4, 5), (6, 7)}. Then, the exchange operation is performed, {(0, 2), (1, 3), (4, 6), (6, 7)}, and the mutual learning is applied again. This results in

synchronization of two groups each with four parties, {(0, 1, 2, 3), (4, 5, 6, 7)}, as shown in Fig.6c. After that, the exchange operation is applied again and the vector takes the form {(0, 4), (1, 5), (2, 6), (3, 7)}. The algorithm terminates when pairs in the new vector apply mutual learning that produces full synchronization between all parties (Fig.6d). The CBTV method needs to transmit the data between the nominated nodes to other nodes in order to be followed. On the other hand, the CBTE algorithm applies the mutual learning algorithm between each pair of nodes separately.

At the same time as the proposed key exchange protocol is scalable; it remains susceptible to active attacks. An attacker can take part in the protocol and synchronize with the group and finally obtain the shared key which endangers the secret communication between the group later. As a result, it is compulsory to build up an authenticated key exchange protocol to permit only certified users to get hold of the mutual secret.

Algorithm 2 –The CBTE method

Require: l, m *variables.*
1: loop $\{for\ j = 0;\ j \leq log_2 M;\ j++\}$
2: $m = 2j$
3: A node participates just the once per iteration.
4: Apply mutual learning between nodes $l,\ l + m$
5: end loop

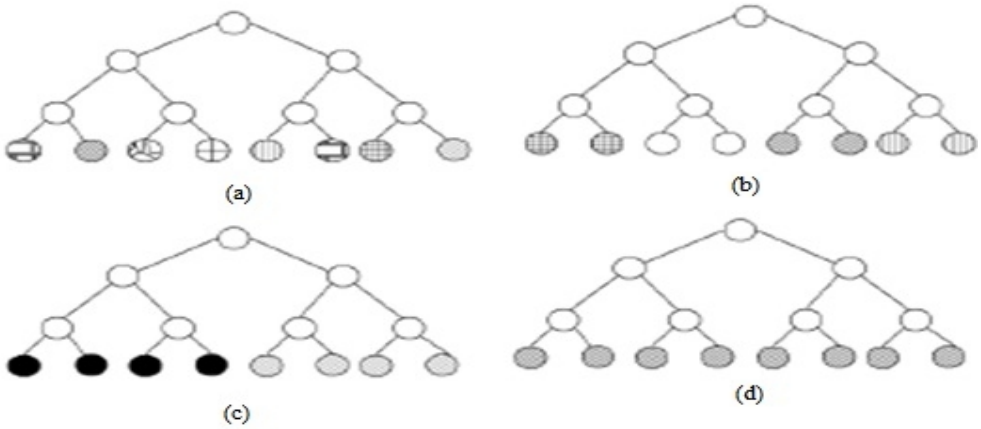


Figure 6. (a) Shows the preliminary configuration of unsynchronized parties.
(b) Pairs of parties are synchronized together, {(0, 1), (2, 3), (4, 5), (6, 7)}.
(c) Synchronization of two groups each with four parties, {(0, 1, 2, 3), (4, 5, 6, 7)}.
(d) After that, the exchange operation is applied again and the vector takes the form {(0, 4), (1, 5), (2, 6), (3, 7)}.

9. GSMLPSA CERTIFICATE GENERATION

While proposed GSMLPSA method offers rapid key exchange between groups of users, it is susceptible to malicious attacks where an challenger can participate in the protocol and hence obtains the group top secret key. So, the group requires an authentication certificate to safeguard it against such types of attacks. In order to construct an authentication certificate, GSMLPSA assumes that the group obtains a secret password which can be used to authenticate the exchange protocol. This password can be mapped to multilayer perceptron guided cryptographic public parameter which can be used as an initial seed for a random number generator which encrypts the

output bits τ in a fashion similar to that was proposed in [12, 13]. Assume a random number generator (RNG), R_i , $R_i^p = \mathrm{Re}\,m((a * R_{i-1}^p \_ c), m)$ with the set λ = {a, c,m} being the RNG parameters.

Algorithm 3 –GSMLPSA Password Authentication Scheme

***Require**: n is the number of iterations needed to synchronize between two parties and a nonlinear function F.*

loop {for each pair}
  Generate random number R_0 and publicly exchange between each pair.
  Compute $R_1 = R_0 \oplus$ password
  loop {for n}
    Generate random number R_i
    if $F(R_i^A > 0)$ then
      $\tau_{sent}^{A} = -\tau_{computed}^{A}$
    end if
    if $F(R_i^B, R_i^B) > 0$ then
      $\tau_{used}^{A} = -\tau_{sent}^{A}$
    end if
  end loop
end loop

10. COMPLEXITY ANALYSIS

The complexity of the Synchronization technique will be O(L), which can be computed using following three steps.

Step 1. To generate a MLP guided 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).
Step 2. 1. Recursive replacement of bits using prime nonprime recognition encryption process takes O(L).
Step 2. 2. MLP based encryption technique takes O(L) amount of time.

Step 3. Complexity of the decryption technique is O(L).
Step 3. 1. In MLP based decryption technique, complexity to convert final cipher text into recursive replacement cipher text T takes O(L).

Key exchange algorithm has complexity of logarithmic proportional to the number of the parties need to synchronize together. Because key exchange protocols works on a structure of a complete binary tree. Algorithm works form leaf level to the root i.e. the height of a complete binary tree which is O (log N).

11. EXPERIMENT RESULTS

In this section, CBTV is applied between group of parties and some simulation results are presented. For simplicity, the number of communicating parties is taken to be four. i.e., (M = 4). Assuming four parties A, B, C and D need to share a common key so they apply the CBTV algorithm. As shown in Fig.7 curve 1, A and B apply the ordinary mutual learning algorithm till they synchronize. At the same time both C and D do the same as shown in curve 2. Then the swapping mechanism is applied and hence, A and C apply the ordinary mutual learning algorithm and the same scenario repeats for B and D. It is evident that curves 3 and 4 are identical which indicates that the four parties have synchronized at common weight vectors. If another party requires to share a key with previously N synchronized parties, it does not need to repeat the entire algorithm again. Instead, the N synchronized parties are dealt with as a single partner and the mutual learning algorithm is applied between an elected party of the N partners

and the new party. Then the other (N − 1) parties apply the learning rules without sending their output bits over the public channel.

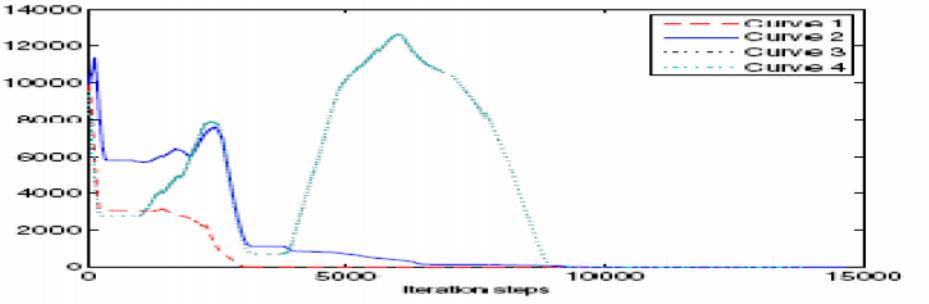


Figure 7. Synchronization between 4 Multilayer Perceptrons.

In this section the results of implementation of the proposed GSMLPSA encryption/decryption 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 [1] technique, existing ANNRBLC [8] and NNSKECC [9].

.

Table 6. Encryption / decryption time vs. File size

Encryption Time (s)			**Decryption Time (s)**		
Source Size (bytes)	**GSMLPSA**	**NNSKECC [9]**	**Encrypted Size (bytes)**	**GSMLPSA**	**NNSKECC [9]**
18432	6. 42	7.85	18432	6.99	7.81
23044	9. 23	10.32	23040	9.27	9.92
35425	14. 62	15.21	35425	14. 47	14.93
36242	14. 72	15.34	36242	15. 19	15.24
59398	25. 11	25.49	59398	24. 34	24.95

Table 6 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.

Table 7 shows Chi-Square value for different source stream size after applying different encryption algorithms. It is seen that the Chi-Square value of GSMLPSA is better compared to

the algorithm ANNRBLC [8] and also better than SATMLP [17] algorithm because of Metamorphosed guided Automata based comparison encryption rather than simple triangularized encryption in case of SATMLP [17].

Table 7. Source size vs. Chi-Square value

Stream Size (bytes)	Chi-Square value (TDES) [1]	Chi-Square value (ANNRBLC) [8]	Chi-Square value in (SATMLP) [17]	Chi-Square value Proposed GSMLPSA
1500	1228.5803	2471.0724	2627.7534	2718.2517
2500	2948.2285	5645.3462	5719.8522	5935.0471
3000	3679.0432	6757.8211	6739.73621	6938.7203
3250	4228.2119	6994.6198	7009.2813	7114.0649
3500	4242.9165	10572.4673	11624.2315	11729.8530

Figure 8 shows graphical representation of table 7.

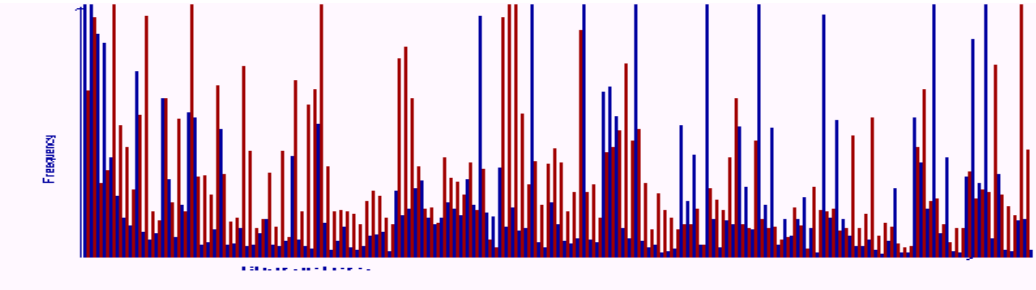


Figure 8. Chi-Square value against stream size

Table 8 shows total number of iteration needed and number of data being transferred for GSMLPSA key generation process with different numbers of input(N) and activated hidden(H) neurons and varying synaptic depth(L).

Table 8. Data Exchanged and No. of Iterations For Different Parameters Value

No. of Input Neurons(N)	No. of Activated Hidden Neurons(K)	Synaptic Weight (L)	Total No. of Iterations	Data Exchanged (Kb)
5	15	3	624	48
30	4	4	848	102
25	5	3	241	30
20	10	3	1390	276
8	15	4	2390	289

12. ANALYSIS OF RESULTS

From results obtained it is clear that the technique will achieve optimal performances. Encryption time and decryption time varies almost linearly with respect to the block size. For the algorithm presented, Chi-Square value is very high compared to some existing algorithms. 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 devised. The security issue of existing algorithm can be improved by using GSMLPSA secret session 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 or output of activated hidden layer as a secret key needed for encryption. The fundamental conception of GSMLPSA based key exchange protocol focuses mostly on two key attributes of GSMLPSA. Firstly, two nodes coupled over a public channel will synchronize even though each individual network exhibits disorganized behaviour. 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) [10]. For usual cryptographic systems, we can improve the safety of the protocol by increasing of the key length. In the case of GSMLPSA, we improved 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 [10]-[12]. So increasing synaptic depth L of the GSMLPSA we can make our GSMLPSA safe.

13. SECURITY ISSUE

The main difference between the partners and the attacker in GSMLPSA is that A and B are able to influence each other by communicating their output bits τ^A & τ^B while E can only listen to these messages. Of course, A and B use their advantage to select suitable input vectors for adjusting the weights which finally leads to different synchronization times for partners and attackers. However, there are more effects, which show that the two-way communication between A and B makes attacking the GSMLPSA protocol more difficult than simple learning of examples. These confirm that the security of GSMLPSA key generation is based on the bidirectional interaction of the partners. Each partener uses a seperate, but identical pseudo random number generator. As these devices are initialized with a secret seed state shared by A and B. They produce exactly the same sequence of input bits. Whereas attacker does not know this secret seed state. By increasing synaptic depth average synchronize time will be increased by polynomial time. But success probability of attacker will be drop exponentially Synchonization by mutual learning is much faster than learning by adopting to example generated by other network. Unidirectional learning and bidirectional synchronization. As E can't influence A and B at the time they stop transmit due to synchrnization. Only one weight get changed where, = T. So, difficult to find w_i weight for attacker to know the actual weight without knowing internal representation it has to guess.

14. FUTURE SCOPE & CONCLUSION

This paper presented a novel approach for group key exchange. GSMLPSA algorithms are proposed as extensions to the ordinary mutual learning algorithm. Also it has been shown that the complexity of the algorithms is logarithmic proportional to the number of the parties need to

synchronize together. This algorithm can be used in many applications such as video and voice conferences. This technique enhances the security features of the key exchange algorithm by increasing of the synaptic depth L of the GSMLPSA. Here two partners A and B do not have to exchange a common secret key over a public channel but use their indistinguishable weights or outputs of the activated hidden layer as a secret key needed for encryption or decryption. So likelihood of attack proposed technique is much lesser than the simple key exchange algorithm.

Future scope of this technique is that this GSMLPSA model can be used in wireless communication and also in key distribution mechanism.

ACKNOWLEDGEMENTS

The author expresses deep sense of gratitude to the Department of Science & Technology (DST) , Govt. of India, for financial assistance through INSPIRE Fellowship leading for a PhD work under which this work has been carried out, at the department of Computer Science & Engineering, University of Kalyani.

REFERENCES

[1] Atul Kahate, Cryptography and Network Security, 2003, Tata McGraw-Hill publishing Company Limited, Eighth reprint 2006.

[2] Sarkar Arindam, Mandal J. K, "Artificial Neural Network Guided Secured Communication Techniques: A Practical Approach" LAP Lambert Academic Publishing (2012-06-04), ISBN: 978-3-659-11991-0, 2012

[3] Sarkar Arindam, Karforma S, Mandal J. K, "Object Oriented Modeling of IDEA using GA based Efficient Key Generation for E-Governance Security (OOMIG) ", International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.2, March 2012, DOI : 10.5121/ijdps.2012.3215, ISSN : 0976 - 9757 [Online] ; 2229 - 3957 [Print]. Indexed by: EBSCO, DOAJ, NASA, Google Scholar, INSPEC and WorldCat, 2011.

[4] Mandal J. K., Sarkar Arindam, "Neural Session Key based Traingularized Encryption for Online Wireless Communication (NSKTE)", 2nd National Conference on Computing and Systems, (NaCCS 2012), March 15-16, 2012, Department of Computer Science, The University of Burdwan, Golapbag North, Burdwan –713104, West Bengal, India. ISBN 978- 93-808131-8-9, 2012.

[5] Mandal J. K., Sarkar Arindam, "Neural Weight Session Key based Encryption for Online Wireless Communication (NWSKE)", Research and Higher Education in Computer Science and Information Technology, (RHECSIT- 2012) ,February 21-22, 2012, Department of Computer Science, Sammilani Mahavidyalaya, Kolkata , West Bengal, India. ISBN 978-81- 923820-0-5,2012

[6] Mandal J. K., Sarkar Arindam, "An Adaptive Genetic Key Based Neural Encryption For Online Wireless Communication (AGKNE)", International Conference on Recent Trends In Information Systems (RETIS 2011) BY IEEE, 21-23 December 2011, Jadavpur University, Kolkata, India. ISBN 978-1-4577-0791-9, 2011

[7] Mandal J. K., Sarkar Arindam, "An Adaptive Neural Network Guided Secret Key Based Encryption Through Recursive Positional Modulo-2 Substitution For Online Wireless Communication (ANNRPMS)", International Conference on Recent Trends In Information Technology (ICRTIT 2011) BY IEEE, 3-5 June 2011, Madras Institute of Technology, Anna University, Chennai, Tamil Nadu, India. 978-1-4577-0590-8/11, 2011

[8] Mandal J. K., Sarkar Arindam, "An Adaptive Neural Network Guided Random Block Length Based Cryptosystem (ANNRBLC)", 2nd International Conference on Wireless Communications, Vehicular Technology, Information Theory And Aerospace & Electronic System Technology" (Wireless Vitae 2011) By IEEE Societies, February 28- March 03, 2011,Chennai, Tamil Nadu, India. ISBN 978-87-92329-61-5, 2011

[9] Mandal J. K., Sarkar Arindam, "Neural Network Guided Secret Key based Encryption through Cascading Chaining of Recursive Positional Substitution of Prime Non-Prime (NNSKECC)", International Confference on Computing and Systems, ICCS – 2010, 19–20 November, 2010,Department of Computer Science, The University of Burdwan, Golapbag North, Burdwan – 713104, West Bengal, India.ISBN 93-80813-01-5, 2010

[10] R. Mislovaty, Y. Perchenok, I. Kanter, and W. Kinzel. Secure key-exchange protocol with an absence of injective functions. Phys. Rev. E, 66:066102,2002.

[11] A. Ruttor, W. Kinzel, R. Naeh, and I. Kanter. Genetic attack on neural cryptography. Phys. Rev. E, 73(3):036121, 2006.

[12] A. Engel and C. Van den Broeck. Statistical Mechanics of Learning. Cambridge University Press, Cambridge, 2001.

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

[14] 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, 2002

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

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

17. Sarkar, A, Mandal , J. K. , " Secured wireless communication through Simulated Annealing Guided Triangularized Encryption by Multilayer Perceptron generated Session Key(SATMLP)", CCSIT-2013, Proceedings of Third International Conference on Computer Science & Information Technology(CCSIT-2013), Computer Science & Information Technology (CS & IT) ISSN : 2231 - 5403 [Online].

AUTHORS

Arindam Sarkar

INSPIRE Fellow (DST, Govt. of India), MCA (VISVA BHARATI, Santiniketan, University First Class First Rank Holder), M.Tech (CSE, K.U, University First Class First Rank Holder). Total number of publications 19.

Jyotsna Kumar Mandal

Jyotsna Kumar Mandal, M. Tech.(Computer Science, University of Calcutta), Ph.D.(Engg., Jadavpur University) in the field of Data Compression and Error Correction Techniques, Professor i n Computer Science and Engineering, University of Kalyani, India. Life Member of Computer Society of India since 1992 and life member of cryptology Research Society of India. Ex-Dean Faculty of Engineering, Teachnology & Management, working in the field of Network Security, Steganography, Remote Sensing & GIS Application, Image Processing. 26 years of teaching and research experiences. Nine Scholars awarded Ph.D. one submitted and eight are pursuing. Total number of publications is more than two hundred eighty one in addition of publication of five books from LAP Lambert, Germany.