NEURO GENETIC KEY BASED RECURSIVE MODULO-2 SUBSTITUTION USING MUTATED CHARACTER FOR ONLINE WIRELESS COMMUNICATION (NGKRMSMC)

Arindam Sarkar and J. K. Mandal

Department of Computer Science & Engineering, University of Kalyani, W.B, India

ABSTRACT

In this paper, a neural genetic key based technique for encryption (NGKRMSMC) has been proposed through recursive modulo-2 substitution using mutated character code generation for online wireless communication of data/information. Both sender and receiver get synchronized based on their final output. The length of the key depends on the number of input and output neurons. Coordinated matching weight vectors assist to generate chromosomes pool. Genetic secret key is obtained using fitness function, which is the hamming distance between two chromosomes. Crossover and mutation are used to add elitism of chromosomes. At first mutated character code table based encryption strategy get perform on plain text. . Then the intermediate cipher text is yet again encrypted through recursive positional modulo-2 substitution technique to from next level encrypted text. This 2nd level intermediate cipher text is again encrypted to form the final cipher text through chaining and cascaded xoring of neuro genetic key with the identical length intermediate cipher text block. Receiver will perform same symmetric operation to get back the plain text using identical key.

KEYWORDS

Neuro Genetic Key based base Recursive Modulo-2 Substitution using Mutated Character (NGKRMSMC), weight vector, input vector, chaining, neuro genetic key.

1. INTRODUCTION

In cryptography there are wide variety of techniques are available with some pros and cons to protect data [1]-[3], [7]. These algorithms have their merits and shortcomings. For Example in DES, AES algorithms [9] the cipher block length is nonflexible. ANNRPMS [1] and ANNRBLC [2] allow only one cipher block encoding. In this paper a neural training based genetic algorithm guided online encryption technique for wireless communicationhas been proposed.

The organization of this paper is as follows. Section 2 of the paper deals with the proposed neuro synchronization and genetic key generation technique and also random block length based cryptographic techniques using mutated character code generation. Session key based technique has been discussed in section 3. Example of recursive modulo-2 technique is given in section 4. Complexity of the algorithm has been presented in section 5. Results are described in section 6. Conclusions are presented in section 7 and references at end.

2. THE NGKRMSMC TECHNIQUE

In proposed technique mutated character code table and recursive positional modulo-2 substitution algorithm is used to encrypt the plain text. This intermediate cipher text is again encrypted to form the final cipher text using chaining of cascaded xoring of the neuro genetic secret key. Using identical weight vector receiver performs deciphering process to regenerate the plain text.

2.1 Neural synchronization scheme & secret key generation

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. Fig. 1. shows the tree based generation process using arbitrary number of nodes (neurons). Corresponding algorithm and genetic key generation is given in two-sub section 1 and.2.

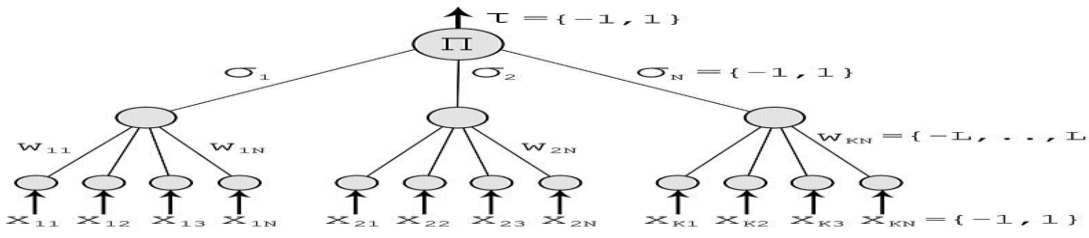


Figure1 A tree parity machine with K=3 and N=4

Neural synchronization algorithm

Input: - Random weights, input vectors for both neural networks
Output: - Session key .
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 maximum value for weight {-L..+L}
Step 1. Initialize random weight values.
Step 2. Repeat step 3 to 6 until both sender and receiver get synchronized.
Step 3. Produce random input vector X. Inputs are generated by a third party (say the server) or one of the communicating parties (figure. 2)

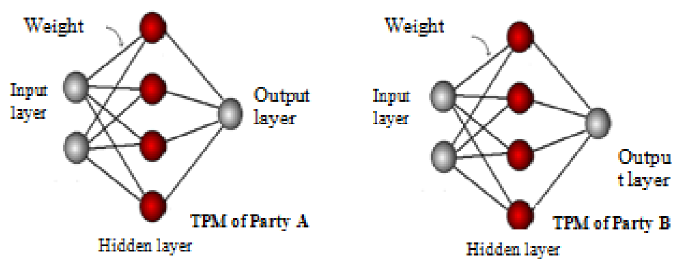


Figure 2 Neural network for machine A and machine B with K=4 & N=2

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. If Output (A) ≠ Output (B) then go to step 3 else if Output (A) = Output (B) then Update the weights. In this paper we have used hebbian-learning rule for synchronization (eq. 3).

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

In each step there may be three possibilities:

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.

3. Output (A) = Output (B) ≠ Output (E): Parties A and B update their weights, but the attacker cannot do that as the synchronization of two parties are faster than learning of an attacker.

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

3 SESSION KEY GENERATION USING GENETIC ALGORITHM

The method considered indistinguishable matched neural weight vector from the first segment in the form of blocks of bits with dissimilar size like 8/16/32/64/128/256 to generate the dynamic chromosomes pool. If neural key is exhausted from the formation of pool at some stage, the key can be circular right shifted by multiple of 8 bits to generate new key stream.

Step 1. Initially, 200 chromosomes are considered as seeds (every chromosomes has number of bits identical to 8/16/32/64/128/256 bit) from dynamic pool. Parameters of the genetic algorithm used are: population size = 200, Crossover probability(CP) = 0.6-0.9 and Mutation probability(MP) =0.001

Step 2. Hamming distance function is used as a fitness function to evaluate the chromosomes to get the best individuals for forming the key.

Step 3. Perform one-point crossover, where a crossover point on the chromosome are elected randomly, and two parent individuals are interchanged at these points.
Step 4. Mutation is performed by bit flipping.
Step 5. Genetic processing is continued until the optimized key (the master key) is obtained.

3.1 Character Code Table Generation

For the plain text "tree" Fig 3 shows corresponding tree representation of probability of occurrence of each character in the plain text. Character 't' and 'r' occur once and character 'e' occurs twice. Preorder traversal get used to find out the character code. 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 tabulated the code and value of a particular character in the plain text.

Mutated tree is generated using mutation. Fig 4, 5 and 6 are the mutated 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.

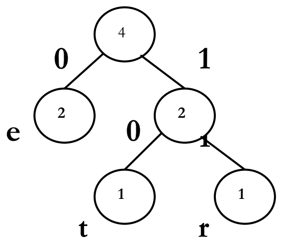


Figure3 Character Code Tree

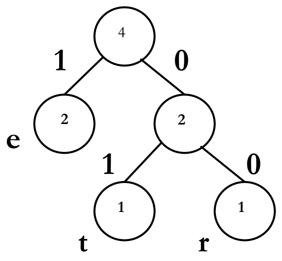


Figure4 Mutated at Position 1,2 & 3,4

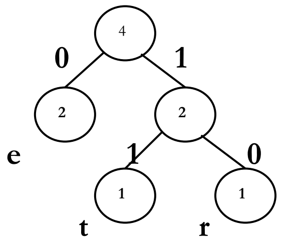


Figure5 Mutated at Position 3,4

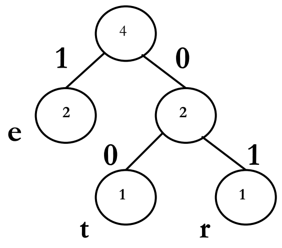


Figure6 Mutated at Position 1,2

Table 1. Code table

Character of plain text	Code	Value of that Character
t	10	2
r	11	3
e	0	0

Table 2. 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 RECURSIVE MODULO-2 ENCRYPTION TECHNIQUE

Now, plain text characters are already converted into the mutated code using the mutated code table. Then it is divided into blocks. The decimal equivalent of the block of bits under consideration is one integral value from which the recursive modulo-2 operation starts. The modulo-2 operation is performed to check whether integral value is even or odd and the position of that integral value in the series of natural even or odd numbers is evaluated. Process is carried out Recursively to a finite number of times, which is exactly the length of the source block. After each modulo-2 operation, 0 or 1 is pushed to the output stream in MSB-to-LSB order; depending on the fact whether the integral value is even or odd. In this way intermediate encrypted text is generated.

```
Set: $P = 0$.
LOOP:
        Evaluate: Temp = Remainder of $D_{L-P} / 2$.
        If Temp = 0
                Evaluate: $D_{L-P-1} = D_{L-P} / 2$.
                Set: $t_P = 0$.
        Else If Temp = 1
                Evaluate: $D_{L-P-1} = (D_{L-P} + 1) / 2$.
        Set: $t_P = 1$.
        Set: $P = P + 1$.
        If $(P > (L – 1))$
                Exit.
END LOOP
```

Figure7 Algorithm for Recursive Modulo-2 encryption technique.

Thus overall complexity of encryption algorithm is O(L).The key is padded with the recursive modulo-2 encrypted text block to form the 2nd level intermediate cipher block. Then the neuro genetic secret key is use to xored with the same length first 2nd level intermediate cipher text block to produce the first final cipher block (neuro genetic 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. neuro genetic vector size) then this block is kept unchanged.

Recursive Modulo-2 Decryption Technique

During decryption the encrypted message is xored with the neuro genetic key to extract the intermediate encrypted stream. Then the intermediate encrypted stream is decomposed into a set of blocks, each consisting of a fixed number of bits using same rule of encryption. Then perform the recursive modulo-2 decryption operation. Finally, character code value is decrypted using character code table to get the plain text. Complexity of this decryption algorithm is also O(L).

```
Set: $P = L - 1$ and $T = 1$.
LOOP:
If $t_P = 0$
        Evaluate: $T = T^{th}$ even number in the series of natural numbers.
Else If $t_P = 1$
        Evaluate: $T = T^{th}$ odd number in the series of natural numbers.
Set: $P = P - 1$.
If $P < 0$
        Exit.
END LOOP
```

Figure 8 Algorithm for Recursive Modulo-2 decryption technique.

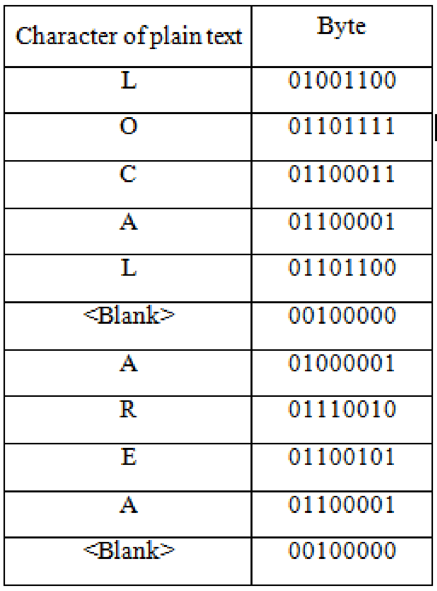
Example of Recursive Modulo-2 Operation

Consider a separate plaintext (P) as: “Local Area Network”.

The Process of Recursive Modulo-2 Encryption

To generate the source stream of bits, we take the reference of Table 3.

Table 3. Character-to-Byte Conversion for the Text “Local Area Network”



Character of plain text	Byte
L	01001100
O	01101111
C	01100011
A	01100001
L	01101100
<Blank>	00100000
A	01000001
R	01110010
E	01100101
A	01100001
<Blank>	00100000

Message stream of bits S is:

S=01001100/01101111/01100011/01100001/01101100/00100000/01000001/01110010/0110010
1/01100001/00100000/01001110/01100101/01110100/01110111/01101111/01110010/01101011.

Decompose S into a set of 5 blocks, each of the first four being of size 32 bits and the last one being of 16 bits. During this process of decomposition, we scan S along the MSB-to-LSB direction. Thus we obtain

S_1=01001100011011110110001101100001, S_2=01101100001000000100000101110010, S_3=01100101011000010010000001001110, S_4=01100101011101000111011101101111, S_5=0111001001101011. For the block S_1, corresponding to which the decimal value is $(1282368353)_{10}$, the process of encryption is shown below:

1282368353 $\rightarrow$ 641184177^1 $\rightarrow$ 320592089^1 $\rightarrow$ 160296045^1 $\rightarrow$ 80148023^1 $\rightarrow$ 40074012^1 $\rightarrow$ 20037006^0 $\rightarrow$ 100018503^0 $\rightarrow$ 5009252^1 $\rightarrow$ 2504626^0 $\rightarrow$ 1252313^0 $\rightarrow$ 626157^1 $\rightarrow$ 313079^1 $\rightarrow$ 156540^1 $\rightarrow$ 78720^0 $\rightarrow$ 39135^0 $\rightarrow$ 19568^1 $\rightarrow$ 9784^0 $\rightarrow$ 4892^0 $\rightarrow$ 2446^0 $\rightarrow$ 1223^0 $\rightarrow$ 612^1 $\rightarrow$ 306^0 $\rightarrow$ 153^0 $\rightarrow$ 77^1 $\rightarrow$ 39^1 $\rightarrow$ 20^1 $\rightarrow$ 10^0 $\rightarrow$ 5^0 $\rightarrow$ 3^1 $\rightarrow$ 2^1 $\rightarrow$ 1^0 $\rightarrow$ 1^1**.** From this we generate the target block T_1 corresponding to S_1 as: T_1=11111001001110010000100111001101.

Applying the similar process, we generate target blocks T_1, T_2, T_3, T_4 and T_5 corresponding to source blocks S_1, S_2, S_3, S_4 and S_5 respectively.

T_2=01110001011111011111101111001001
T_3=01001101111110110111100101011001
T_4=10001001000100011101000101011001
T_5=1110100110110001.
Now, combining target blocks in the same sequence, we get the target stream of bits T as the following:

T=11111001/00111001/00001001/11001101/01110001/01111101/11111011/11001001/0100110
1/11111011/01111001/01011001/10001001/00010001/11010001/01011001/11101001/10110001.
After padding the key with the above intermediate cipher text and xoring with the neural secret key, final encrypted cipher text is generated. This stream of bits, in the form of a stream of characters, is transmitted as the encrypted message.

The Process of Recursive Modulo-2 Decryption

At the destination end, receiver's secret neural key is used to xoring the cipher text to get back the key and intermediate cipher text. Using secret key, the receiver gets the information on different block lengths. Using that secret key, all the blocks T_1, T_2, T_3, T_4 and T_5 are formed as follows:

T_1=11111001001110010000100111001101
T_2=01110001011111011111101111001001
T_3=01001101111110110111100101011001
T_4=10001001000100011101000101011001
T_5=1110100110110001.

Apply the process of decryption to generate source blocks S_i for all T_i, $1 \leq i \leq 5$. 32-bit stream is regenerated S1=01001100011011110110001101100001."

All the source blocks of bits are regenerated and combining those blocks in the same sequence, the source stream of bits obtained to get the source message or the plaintext.

5 RESULTS

Results are 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 4. Encryption / decryption time vs. File size

Encryption Time (s)			Decryption Time (s)		
Source (bytes)	NGKRMSMC	ANNRPMS	Encrypted (bytes)	NGKRMSMC	ANNRPMS
18432	5. 98	5. 32	18432	5.52	4.85
23044	8.04	7. 37	23040	7.63	6.96
35425	14.59	13. 98	35425	14.08	13. 37
36242	15.17	14. 53	36242	14.70	14. 01
59398	23.02	22. 39	59398	22.51	21. 88

Table 4 shows encryption and decryption time with respect to the source and encrypted size respectively.

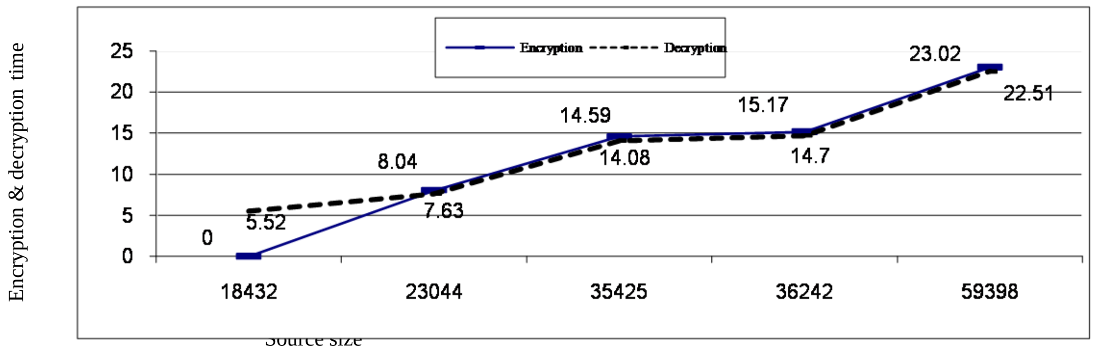


Figure9 Source size vs. encryption time & decryption time

Encryption /decryption time is marginally high because of incorporation of genetic technique and mutated character coding.

Table 5 shows Chi-Square value for different source stream size on applying various encryption algorithms. It is seen that the Chi-Square value of NGKRMSMCC is better compared to the RBCMPCC algorithm and comparable with values of the RSA technique.

Table 5. Source size vs. Chi-Square value

Stream Size (bytes)	Chi-Square value (TDES)	Chi-Square value Proposed NGKRMS MCC	Chi-Square value (RBCM CPCC)	Chi-Square value (RSA)
1500	1228.5803	2465.0749	2464.0324	5623.14
2500	2948.2285	5643.5271	5642.5835	22638.99
3000	3679.0432	6759.2956	6714.6741	12800.355
3250	4228.2119	6997.6173	6994.6189	15097.77
3500	4242.9165	10572.7263	10570.4671	15284.728

Figure 10 shows graphical representation of table 5.

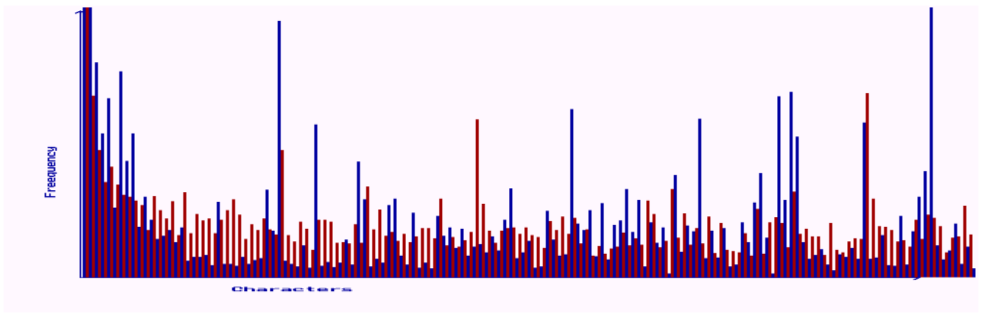


Figure10 Source size vs. Chi-Square value

6 Complexity Of The Technique

The complexity of the technique will be 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 independent of the size N of the networks (up to N=1000). The complexity of synchronization is O (log N).

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

Step 2.1: Corresponding to the source block S = s_0 s_1 s_2 s_3 s_4… s_{L-1}, evaluate the equivalent decimal integer, D_L. It takes O (L) amount of time.

Step 2.2: Step 3 and step 4 executed exactly L number of times for the values of the variable P ranging from 0 to (L-1) increasing by 1 after each execution of the loop where the complexity is O (L) as L no. of iterations are needed.

Step 2.3: Apply modulo-2 operation on D_{L-P} to check if D_{L-P} is even or odd, takes constant amount of time i.e. O(1).

Step 2.4: If D_{L-P} is found to be even, compute $D_{L-P-1} = D_{L-P} / 2$, where D_{L-P-1} is its position in the series of natural even numbers. Assign $t_P = 0$. If D_{L-P} is found to be odd, compute $D_{L-P-1} = (D_{L-P} + 1) / 2$, where D_{L-P-1} is its position in the series of natural odd numbers. Assign t_P=1 which generates the complexity as O (1).

Step 2.5: evaluate: D_L, the decimal equivalent, corresponding to the source block S = s_0 s_1 s_2 s_3 s_4 … s_{L-1} which takes O (L) amount of time.

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

Step 3.1: Set P=L-1and T=1.So time complexity of assignments is O (1).
Step 3.2: Complexity of loop is O (L) because L iterations are needed.
Step 3.3: If $t_P = 0$,T = T^th even number in the series of natural even numbers. If $t_P = 1$,T = T^th odd number in the series of natural even numbers. Complexity is O (1) as it takes constant amount of times.
Step 3.4: Set P=P-1. This step takes constant amount of time i.e. O(1).
Step 3.5: 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).

7 CONCLUSION

The paper presents a novel approach for generation of secret key using neural synchronization. This proposed technique allows key exchange through public channel. So likelihood of attack of the technique is much less than the simple RBCMCCC [4]algorithm.

ACKNOWLEDGEMENTS

The author expresses deep sense of gratitude to the DST, Govt. of India, for financial assistance through INSPIRE Fellowship leading for a PhD work under which this work has been carried out.

REFERENCES

[1] 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)”, in Proc. International Conference on Recent Trends In Information Technology (ICRTIT 2011) Conf. BY IEEE, 3-5 June 2011, Madras Institute of Technology, Anna University, Chennai, Tamil Nadu, India. 978-1-4577-0590-8/11

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

[3] Mandal, J. K., Sarkar Arindam “Neural Network Guided Secret Key based Encryption through Cascading Chaining of Recursive Positional Substitution of Prime Non-Prime (NNSKECC)” [TP-48][PID-63], in Proc. of International Confrence of Computing and Systems-2010 Conf by ICCS-2010, Novembar 19-20, 2010, The University of Burdwan,pp 291-297.

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

[5] T. Godhavari, N. R. Alainelu and R. Soundararajan “Cryptography Using Neural Network “in Proc.by Conf IEEE Indicon 2005, Chennai, India, 11-13 Dec. 2005.ggggjggggggggggg

[6] 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.

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

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

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

Author

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 25.

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 in 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. Dean Faculty of Engineering, Technology & Management, working in the field of Network Security, Steganography, Remote Sensing & GIS Application, Image Processing. 25 years of teaching and research experiences. Eight Scholars awarded Ph.D. and 8 are pursuing. Total number of publications 267.