CRYPTANALYSIS OF KEY EXCHANGE METHOD USING COMPUTATIONAL INTELLIGENCE GUIDED MULTILAYER PERCEPTRON IN WIRELESS COMMUNICATION (CKEMLP)

Arindam Sarkar and J. K. Mandal

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

ABSTRACT

In this paper, a cryptanalysis of key exchange method using multilayer perceptron (CKEMLP) has been proposed in wireless communication of data/information. In this proposed CKEMLP technique both sender and receiver uses an identical multilayer perceptrons for synchronization between them. After achieving the full synchronization weights vectors of both the parties' becomes identical and this identical weight vector is used as a secret session key for encryption/decryption. Different types of possible attacks during synchronization phase are introduced in this paper. Among different types of attacks some of them can be easily prevented by increasing the synaptic depth L. But few attacks are also there which has a great success rate.

KEYWORDS

Cryptanalysis, Encryption, Wireless Communication.

1. INTRODUCTION

In this paper CKEMLP technique has been proposed to analyzed variety of attacks that can be possible in key generation phase using multilayer perceptron and also provides some way out from these attacks.

The organization of this paper is as follows. Section 2 of the paper deals with different types of attacks in CKEMLP. Conclusions and future scope are drawn in section 3 and that of references at end.

2. DIFFERENT TYPES OF ATTACKS ON MULTILAYER PERCEPTRON

The security of multilayer perceptron based key generation protocol is based on a contest between attractive and repulsive forces. Two multilayer perceptrons interacting with each other synchronize much faster than an attacker network only trained with their inputs and outputs. The dissimilarity between the two parties and the attacker is that the two parties synchronize in a polynomial time of synaptic depth L, while the complexity of the attacker scales exponentially. However, the process is stochastic and depends on the random attractive and repulsive forces. As a result, there is a small probability that an attacker succeeds to synchronize with one of the parties. The difficulty an attacker faces with the organization of multilayer perceptron is the lack of information about the internal representation of A's or B's machine[1, 2, 3]. Most of attacks depend on estimating the state of the hidden units. Following are the different possible attacks on multilayer perceptron during key generation phase.

2.1 Type1 Attack

In this type1attack replicate a huge population of multilayer perceptrons with the identical arrangement as the two parties, and teach them with the same inputs. At each stage about half the replicated networks produces an output of +1, and half produces an output of -1. Successful multilayer perceptrons whose outputs imitate those of the two parties raise and multiply, while unsuccessful multilayer perceptrons gets ruled out. Attack starts with one network with haphazardly chosen weights. At each step a population of networks grow according to 3 potential scenarios:[8,9,10,11,12,13,14]

- A and B have dissimilar outputs $\tau^A \neq \tau^B$, and therefore do not change their weights. Then all the attacker's networks stay unaffected as well.

- A and B have the equivalent outputs $\tau^A = \tau^B$, and the sum of attacking networks is lesser than some predefined limit. In this case there are 4 possible combinations of the hidden outputs agreeing with the final output. So, the attacker replaces each network N from the population by 4 variants of itself, $\{N_1, \ldots ,N_4\}$ which are the results of updating N with the standard learning rule but pretending that the hidden outputs were equal to each one of these combinations.

- A and B have the identical outputs $\tau^A = \tau^B$ but the total number of simulated networks is larger than predefined value. In this case the attacker computes the outputs of all the networks, deletes the unsuccessful networks whose output is different from τ^A, and updates the weights in the successful networks by using the standard learning rule with the actual hidden outputs of the perceptrons [4,5,6,7].

2.2 Type2 Attack

In type2 attack the attacker imitates one of the parties, but if attacker output disagrees with the imitated party's output $\tau^c \neq \tau^A$, attacker certainly knows that either one or all three of his hidden units are mistaken. In order to get $\tau^c = \tau^A$ attacker negates the sign of one of attacker's hidden units. As $\sigma = \operatorname{sgn}(h)$ the unit most likely to be wrong is the one with the minimal $|h|$, therefore that is the unit which is negate. This policy results a immense enhancement in the attacker's achievement. It can be seen that the success rate is quite high for all L values presented, but it drops exponentially as L increases. On the other hand parties' synchronization time increases like L^2, and therefore it can be conclude that in the boundary of large L values the proposed technique is secure against the type2 attack. Each input can be viewed as K random hyperplanes

$X_1, \ldots, X_K$ corresponding to K hidden units. Each X_i is a hyperplane $f_i(z_1,\ldots,z_n) = \sum_{j=i}^{N} x_{ij}.z_j = 0$

in the N-dimensional discrete space $U = \{-L,\ldots,L\}^N$. The weights of a network could be also viewed as K points $W_1, \ldots, W_K$ in U, $W_i = \{w_{i1},\ldots,w_{ik}\}$, while the i-th hidden output is just the side of the half-space (with respect to X_i) which contains W_i. Consider an attacking network E that is close enough to the unknown network A but has a different output for a given input. In fact they have either 1 or 3 different hidden outputs. The second case is less likely to occur so we assume that only one hidden output of the network E is different from the corresponding hidden output of A. Consequently, only one pair (W_i^A, W_i^E) is separated by the known input hyperplane X_i. Of course, we are interested in detecting its index i. If the points W_i^E and W_i^A are separated by X_i then the distance between them is greater than the distance from W_i^E to the hyperplane X_i. W_i^E and W_i^A are close to each other, so the distance from W_i^E to X_i has to be small. On the other hand, if W_i^E and W_i^A are in the same half-space with respect to X_i then they are more likely to be far away from the random input X_i (even though we know that they are close to each other). We thus guess that the index of the incorrect hidden output is the i for which W_i^E is closest to the corresponding hyperplane X_i, where we compute the distance by

$\rho(W_i^E, X_i) = \left| f_i(W_i^E) \right|$. Formally, the attacker constructs a single neural network E with the same structure as A and B, and randomly initializes its weights. At each step attacker's trains E with the same input as the two parties, and updates its weights with the following rules:

- If A and B have different outputs $\tau^A \neq \tau^B$, then the attacker doesn't update E.
- If A and B have the same outputs $\tau^A = \tau^B$ and $\tau^E = \tau^A$, then the attacker updates E by the usual learning rule.
- If A and B have the same outputs $\tau^A = \tau^B$ and $\tau^E \neq \tau^A$, then the attacker finds $i_0 \in \{1,..., K\}$ that minimizes $\left| \sum_{j=0}^{N} w_{ij}^E . x_{ij} \right|$.The attacker negates $\tau_{i_0}^E$ and updates E assuming the new hidden bits and output τ^A.

2.3 Type3 Attack

In this type3 attack a huge collection of M attackers work together. The type2 attacker's likelihood to supposition correctly A's interior representation is some function *Pcorrect*(þ) of its overlap þ with A, starting from *Pcorrect*(þ = 0) = 0.25. Assume there are group of M independent type2 attackers, each having overlap þ with A. They will split into 4 groups, one for each possible internal representation. Since *Pcorrect* > 0.25 for all þ > 0, the number of attackers having the correct internal representation,*M .Pcorrect*, will be bigger than the number of attackers in the other 3 groups, for all þ > 0. Therefore, the internal representation resulted from the majority discussion of M independent type2 attackers would always be the correct one! From this argument we conclude that the attack should use M >> 2^{k-1} attackers, which would simultaneously develop an overlap with the parties, trying to remain as independent as possible.

2.4 Type4 Attack

The type4 attack procedure is to start from independent type2 attackers and let them act disjointedly for some preliminary number of time steps. Then, the majority procedure is applied: we count how many attackers have each of the 4 possible internal representations, and assign the majority's internal representation to all the M attackers. To prevent the similarity between the attackers from developing too quickly, this majority procedure is applied only on even time steps. However, the attackers make many coherent moves, and unavoidable overlap is developed between them as well. Therefore we do not have a group of *independent* attackers, but of attackers with an overlap between them. This overlap diminishes the efficiency of the attack, and it is not *always* successful as a majority attack of M independent attackers would be.

2.5 Type5 Attack

It is much easier to predict the position of a point in a bounded multidimensional box after several moves in its random walk than to guess its original position. A simple way to do it is to consider each coordinate separately, and to associate with each possible value i in the interval $\{-L,...,L\}$ the probability $p_t(i) = \Pr[X_t = i]$. Initially $\forall i, p_0(i) = \frac{1}{2L+1}$ and after each move $p_{t+1}(i) = \sum_j p_t(j)$, where j are such that if $x_t = j$ then $x_{t+1} = i$ .Applying this technique to the original scheme we face the problem that the moves are not known | the attacker does not know which perceptrons are updated in each round. Fortunately, if we know the distribution of the probabilities $P_{k,n,i} = \Pr[w_{k,n} = i]$ then using dynamic programming we can calculate the distribution of $\overrightarrow{w_k}\overrightarrow{x_k}$ for a given vector $\overrightarrow{x_k}$ and thus the probabilities $u_k(s) = \Pr[\tau_k = s]$. Using these probabilities we can calculate the conditional probabilities $U_k = \Pr[\tau_k = 1|\tau]$ $U_k = \frac{\sum_{(\alpha_1,...,\alpha_k):\Pi_i\alpha_i=\tau,\alpha_k=1} \Pi_i u_i(\alpha_i)}{\sum_{(\alpha_1,...,\alpha_k):\Pi_i\alpha_i=\tau} \Pi_i u_i(\alpha_i)}$,because τ is publicly known. We can now update the distribution of the weights: $P^{t+1}{}_{k,n,i} = \sum_j P^t_{k,n,j} \Pr[w^t{}_{k,n} = j \Rightarrow w^{t+1}{}_{k,n} = i]$ is calculated using U$_k$. Experiments show that in most cases, when A and B converge to a common $\hat{w}_{k,n}$ the probabilities $\Pr[w_{k,n} = \hat{w}_{k,n}] \approx 1$ and thus the adversary can easily find $\hat{w}_{k,n}$ when A and B decide to stop the protocol.

2.6 Type 6 Attack

Here the attacker E's neural network has the same structure of A's and B's. All what E has to do is to start with random initial weights and to train with the same inputs transmitted between A and B over the public channel. Then, the attacker E learns the mutual output bit $\tau^{A/B}$ between them and applies the same learning rule by replacing τ^E with $\tau^{A/B}$, i.e.

$$W_k^E = W_k^E - \tau^{A/B} x_k \Theta(\sigma^E{}_k \tau^{A/B})(\tau^A \tau^B)$$

Uncorrelated hidden units, e. g. at the beginning of the synchronization process, have ρ$_i$ = 0, while the maximum value ρ$_i$ = 1 is reached for fully synchronized weights. Consequently, ρi is the most important quantity for analyzing the process of synchronization. But it is also interesting to estimate the mutual information gained by the partners during the process of synchronization. For this purpose one has to calculate the entropy

$$S_i^{AB} = -N \sum_{a=-L}^{L} \sum_{b=-L}^{L} P_{a,b}^{i} \ln P_{a,b}^{i}$$

of the joint weight distribution of A’s and B’s neural networks. Similarly the entropy of the weights in a single hidden unit is given by

$$S_i^A = -N \sum_{a=-L}^{L} \left(\sum_{b=-L}^{L} P_{a,b}^{i} \right) \ln \left(\sum_{b=-L}^{L} P_{a,b}^{i} \right)$$

$$S_i^B = -N \sum_{b=-L}^{L} \left(\sum_{a=-L}^{L} P_{a,b}^i \right) \ln \left(\sum_{a=-L}^{L} P_{a,b}^i \right)$$

Of course, these equations assume that there are no correlations between different weights in one hidden unit. This is correct in the limit N → ∞, but not necessarily for small systems. Using above three equations the mutual information of A’s and B’s

Multilayer Perceptrons can be calculated as

$$I^{AB} = \sum_{i=1}^{K} \left(S_i^A + S_i^B - S_i^{AB} \right)$$

At the beginning of the synchronization process, the partners only know the weight configuration of their own neural network, so that $I^{AB} = 0$. But for fully synchronized weight vectors this quantity is equal to the entropy of a single Multilayer Perceptron, which is given by

$$S_0 = KN \ln (2L + 1)$$

in the case of uniformly distributed weights.

The neural key-exchange protocol is an application of neural synchronization. Both partners A and B use a Multilayer Perceptron with the same structure. The parameters K, L and N are public. Each neural network starts with randomly chosen weight vectors. These initial conditions are kept secret. During the synchronization process, only the input vectors x_i and the total outputs τ^A, τ^B are transmitted over the public channel. Therefore each participant just knows the internal representation ($\sigma_1, \sigma_2, \ldots, \sigma_K$) of his own Multilayer Perceptron. Keeping this information secret is essential for the security of the key-exchange protocol. After achieving full synchronization A and B use the weight vectors as common secret key. The main problem of the attacker E is that the internal representations ($\sigma_1, \sigma_2, \ldots, \sigma_K$) of A’s and B’s Multilayer Perceptrons are not known to her. As the movement of the weights depends on σi, it is important for a successful attack to guess the state of the hidden units correctly. Of course, most known attacks

use this approach. But there are other possibilities and it is indeed possible that a clever attack method will be found, which breaks the security of neural cryptography completely. However, this risk exists for all cryptographic algorithms except the one-time pad.

2.7 Type 7 Attack

For the simple attack E just trains a third Multilayer Perceptron with the examples consisting of input vectors xi and output bits τ^A. These can be obtained easily by intercepting the messages transmitted by the partners over the public channel. E’s neural network has the same structure as A’s and B’s and starts with random initial weights, too.

In each time step the attacker calculates the output of her neural network. Afterwards E uses the same learning rule as the partners, but τ^E is replaced byτA. Thus the update of the weights is given by one of the following equations:

• Hebbian learning rule:

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

Anti-Hebbian learning rule:

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

Random walk learning rule:

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

So E uses the internal representation (σ_1, σ_2, . . . , σ_K) of her own network in order to estimate A's, even if the total output is different. As $\tau^A \neq \tau^E$ indicates that there is at least one hidden unit with $\sigma_i^A \neq \sigma_i^E$, this is certainly not the best algorithm available for an attacker.

2.8 Type 8 Attack

The geometric attack performs better than the simple attack, because E takes τ^E and the local fields of her hidden units into account. In fact, it is the most successful method for an attacker using only a single Multilayer Perceptron. Similar to the simple attack E tries to imitate B without being able to interact with A. As long as $\tau^A = \tau^E$, this can be done by just applying the same learning rule as the partners A and B. But in the case of $\tau^E \neq \tau^A$ E cannot stop A's update of the weights. Instead the attacker tries to correct the internal representation of her own Multilayer Perceptron using the local fields $h_1^E, h_{2,}^E, ..., h_K^E$ as additional information. These quantities can be used to determine the level of confidence associated with the output of each hidden unit. As a low absolute value $\left|h_i^E\right|$ indicates a high probability of $\sigma_i^A \neq \sigma_i^E$, the attacker changes the output σ_i^E

Of the hidden unit with minimal $\left|h_i^E\right|$ and the total output τ^E before applying the learning rule. Of course, the geometric attack does not always succeed in estimating the internal representation of A's Multilayer Perceptron correctly. Sometimes there are several hidden units with $\sigma_i^A \neq \sigma_i^E$. In this case the change of one output bit is not enough. It is also possible that $\sigma_i^A = \sigma_i^E$ for the hidden unit with minimal $\left|h_i^E\right|$, so that the geometric correction makes the result worse than before.

2.9 Type 9 Attack

With the majority attack E can improve her ability to predict the internal representation of A's neural network. For that purpose the attacker uses an ensemble of M Multilayer Perceptrons instead of a single neural network. At the beginning of the synchronization process the weight vectors of all attacking networks are chosen randomly, so that their average overlap is zero. Similar to other attacks, E does not change the weights in time steps with $\tau^A \neq \tau^B$, because the partners skip these input vectors, too. But for $\tau^A = \tau^B$ an update is necessary and the attacker calculates the output bits $\tau^{E,m}$ of her Multilayer Perceptrons. If the output bit $\tau^{E,m}$ of the m-th attacking network disagrees with τ^A, E searches the hidden unit i with minimal absolute local field $\left|h_i^{E,m}\right|$

.Then the output bits $\sigma_i^{E,m}$ and $\tau^{E,m}$ are inverted similarly to the geometric attack. Afterwards the attacker counts the internal representations $\left(\sigma_1^{E,m},...,\sigma_K^{E,m}\right)$ of her Multilayer Perceptrons and selects the most common one. This majority vote is then adopted by all attacking networks for the application of the learning rule. But these identical updates create and amplify correlations between E's Multilayer Perceptrons, which reduce the efficiency of the majority attack. Especially if the attacking neural networks become fully synchronized, this method is reducedto a geometric attack. In order to keep the Tree ParityMachines as uncorrelated as possible, majority attack and geometric attack are used alternately. In even time steps the majority vote is used for learning, but otherwise E only applies the geometric correction. Therefore not all updates of the weight vectors are identical, so that the overlap between them is reduced. Additionally, E replaces the majority attack by the geometric attack in the first 100 time steps of the synchronization process.

2.10 Type 10 Attack

The genetic attack offers an alternative approach for the opponent, which is not based on optimizing the prediction of the internal representation, but on an evolutionary algorithm. E starts with only one randomly initialized Multilayer Perceptron, but she can use up to M neural networks. Whenever the partners update the weights because of $\tau^A = \tau^B$ in a time step, the following genetic algorithm is applied:

• As long as E has at most $M/2^{K-1}$ Multilayer Perceptrons, she determines all 2^{K-1} internal representations $\left(\sigma_1^{E},...,\sigma_K^{E}\right)$ which reproduce the output τ^A. Afterwards these are used to update the weights in the attacking networks according to the learning rule. By doing so E creates 2^{K-1} variants of each Multilayer Perceptron in this mutation step.

• But if E already has more than $M/2^{K-1}$ neural networks, only the fittest Multilayer Perceptrons should be kept. This is achieved by discarding all networks which predicted less than U outputs τA in the last V learning steps, with $\tau^A = \tau^B$, successfully. A limit of U = 10 and a history of V = 20 are used as default values for the selection step. Additionally, E keeps at least 20 of her Multilayer Perceptrons. The efficiency of the genetic attack mostly depends on the algorithm which selects the fittest neural networks. In the ideal case the Multilayer Perceptron, which has the same sequence of internal representations as A is never discarded. Then the problem of the opponent E would be reduced to the synchronization of K perceptrons and the genetic attack would succeed certainly. However, this algorithm as well as other methods available for the opponent E are not perfect.

Table 1. Source size vs. Chi-Square value

Stream Size (bytes)	**Chi-Square value (TDES) [1]**	**Chi-Square value (CKEMLP)**	**Chi-Square value (ANNRBLC) [8]**	**Chi-Square value (RSA) [1]**
1500	1228.5803	2856.2673	2471.0724	5623.14
2500	2948.2285	6582.7259	5645.3462	22638.99
3000	3679.0432	7125.2364	6757.8211	12800.355
3250	4228.2119	7091.1931	6994.6198	15097.77
3500	4242.9165	12731.7231	10572.4673	15284.728

Figure 1 shows graphical representation of table 2.

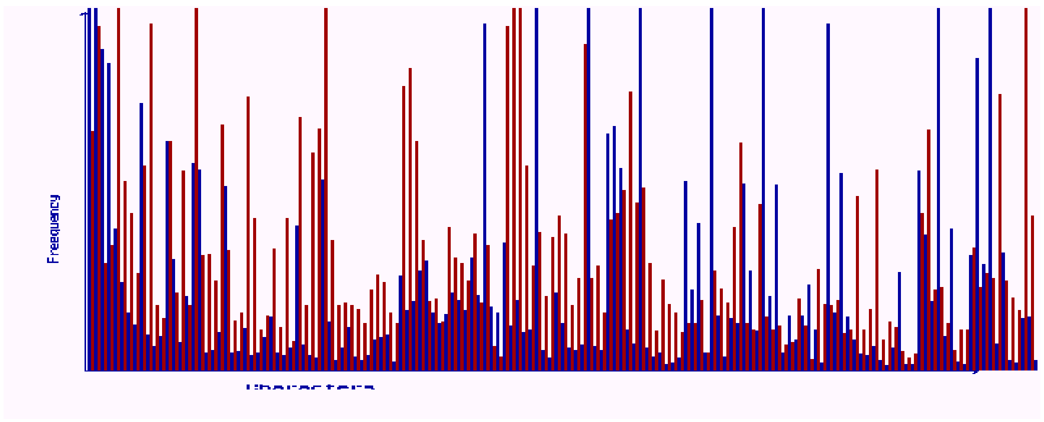

Figure 1. Chi-Square value against stream size

3. Future Scope & Conclusion

This paper presented a novel approach for cryptanalysis of key exchange using multilayer perceptron. In this paper 10 types of attacks has been presented along with analysis. In future attacks in group key exchange protocol with analysis will be considered.

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] 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

[2] 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.

[3] 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

[4] 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

[5] 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

[6] 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

[7] 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

[8] 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.

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

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

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

[12] 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

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

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

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 25.

Jyotsna Kumar Manda

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. one submitted and 8 are pursuing. Total number of publications 267.