Securing MANET using Artificial Neural Networks

Manisha, Partibha Yadav

Abstract- In present paper, we are providing enhanced security to Manet using the Back Propagation Method of Artificial Neural Networks. Here we eliminate the use of files for storing the passwords or other encrypted content used in the network , by replacing them with much secure weight matrix of Back Propagation. The neural network system is trained for some data and then file used for storing the data is eliminated, resulting confidentiality of connection. Thus the method makes the system more secure.

Key Words: Mobile Adhoc Network, Artificial Neural Network, Pattern Mapping Technique.

I. INTRODUCTION

A Mobile Ad hoc Network (MANET) is a collection of wireless nodes that can dynamically be setup anywhere & anytime without using any pre-existing network infrastructure. Mobile Ad hoc Networks are a new field of research of wireless communication for mobile nodes. There is no fixed infrastructure such as base stations in such networks. Nodes within each other radio range communicate directly via wireless links while those which are far apart rely on other nodes to pass on messages. Node mobility causes frequent changes in topology. MANETs became a popular subject for research as laptops and 802.11/Wi-Fi wireless networking became widespread in the mid to late 1990s. Different protocols are then evaluated based on the packet drop rate, the overhead introduced by the routing protocol, and other measures. The Mobile Ad hoc Network is a collection of mobile nodes that are dynamically and arbitrarily located in such a manner that the interconnection between nodes keeps changing frequently.

Mobile Ad hoc Network Devices

- Mobiles
- Laptops
- Personal Computers

Characteristics of Mobile Ad hoc Network

- Does not rely on a fixed infrastructure for its operation autonomous transitory association of mobile nodes.
- It can be rapidly deployed with user intervention.
- Need not to operate in a standalone fashion but can be attached to the internet or cellular networks.
- Devices are free to join or leave the network and they may randomly, possibly resulting in rapid and unpredictable changes.

Manuscript received July , 2013

Manisha, Student, Department of Computer Science & Engg , P.D.M. College of Engg. For Women, Bahadurgarh (Haryana), India.
Partibha Yadav, Lecturer, Department of Computer Science & Engg , P.D.M. College of Engg. For Women, Bahadurgarh (Haryana), India.

Mobile Ad hoc Network Working

A Mobile Ad hoc Network (MANET) is a network architecture that can be rapidly deployed without relying on existing fixed wireless network infrastructure. This means that network nodes should be able to communicate to each other even if no static infrastructure, such as back bone network, base station, and centralized network management function are available. Under these situations a node itself provides these functions. The transmission range of a node is limited to a circular region around the node. If the destination node is not in the transmission range of the source node, then the mobile ad hoc network works like a multihop network with one or more node acting as routing node. All the active nodes of the MANET need to transmit a "hi" message at regular intervals, to indicate their presence. Other nodes, in the transmission range of a node, can use that node as next hop to forward their packets toward the destination.

Security Threat on MANETs

With high speed network technologies, large amount of data that is considered confidential, is being transferred over the Internet and other public channels. This has increased our concern for protection of information which includes our passwords and data.
In eavesdropping , the node simply observes the private information. This information can be later used by the malicious node. The secret information like location, public key, private key, password etc. can be fetched by eavesdropper.
The idea of the present research work started with the simple concept on how various encryption algorithms work, how they are utilized for securing the machines and the networks, and how artificial neural networks can help these encryption algorithms to make machines and networks more secure.

II. NEURAL NETWORK ARCHITECTURE AND METHODOLOGY

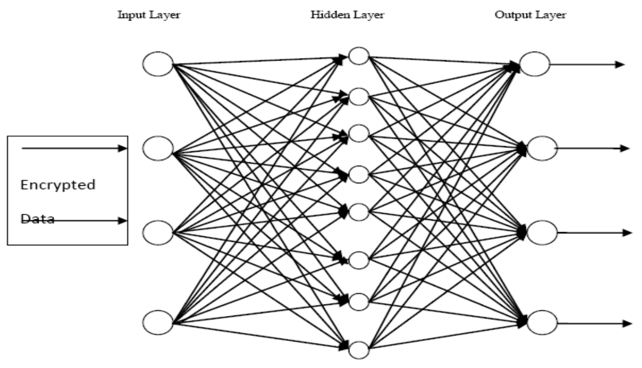

Fig 1. Neural Network Architecture for a Feed Forward Network

Table 1: Parameters used for Training of Network using Back Propagation Model

Parameter	**Value**
Neurons in Input Layer	4
Number of Hidden Layers	1
Neurons in Hidden Layer	8
Neurons in Output Layer	4
Minimum Error Exist in the Network	0.001
Initial Weights and biased term values	Values between 0 and 1

The basic network architecture is a completely connected feed forward adaptive network, based on model of Back Propagation, that performs static mapping from input string to output sting with its error propagation model. Each input pattern thus corresponds to one key encryption. The network has one input layer having 4 neurons with two hidden layers having 4 neurons each, there is one output layer having 4 neurons corresponding to four neurons in input layer and in the network architecture each node of input and output layer represents one bit of the four bit string. Since we have taken four bit pattern, there can be at most 16 pattern data sets. We trained the network for the different 4-bit input pattern say (x_1,x_2,x_3,x_4) and observed 4-bit output pattern say (y_1,y_2,y_3,y_4)

The input and output values of the network are represented as x_i and s_h for ith neurons in input layer and hth neuron in output layer, and w_{ih} is the weight that connects input of ith neuron to the output of hth neuron.

The output values are given by the relation,

$$s_h = f\left[y_1^h\right]$$

in which y_1^h is the activation vector of any layer j and is given by:

$$y_1^h = \sum_{j=1}^{I} w_{ih} x_i$$

The error expression introduced for a single perceptron is generalized to include all squared errors for the outputs j=1,2,……..J for a specific pattern p=1,2,3,…..P that is at the input and gives the output error.

$$E^p = \frac{1}{2}\sum_{j=1}^{J}\left(d_j - y_j\right)^2$$

where, d is the desired output vector $[d_1 d_2 d_3 ....d_J]^t$

Now the weight adjustment is given as:

$$\Delta w_{hj} = -\eta \frac{\partial E^p}{\partial w_{hj}}$$

Where, η is the Back Propagation learning rate

This expression is adjusted as:

$$\frac{\partial E^p}{\partial w_{hj}} = \frac{\partial E^p}{\partial y_j^h} \cdot \frac{\partial y_1^h}{\partial w_{hj}} = \frac{\partial E^p}{\partial y_j^h} .\left[s_h\left(y_1^h\right)\right] = \frac{\partial E^P}{\partial s_j\left(y_1^h\right)} \cdot \frac{\partial s_j\left(y_j^h\right)}{\partial y_j^h}\left[s_h\left(y^h\right)\right]$$

Since $\frac{\partial S_j\left(y_i^h\right)}{\partial y_j^h} = \dot{S}_j(y_j^h)$

Hence the weight adjustment is now denoted as:

$$\Delta w_{ih} = -\eta \frac{\partial E^P}{\partial s_j\left(y_j^h\right)} .\dot{S}_j(y_j^h).s_h\left(y_j^h\right) = -\eta \sum_{j=1}^{J}\left(d_j - y_j\right)\dot{S}_j(y_j^h).s_h\left(y_j^h\right)$$

The final adjusted weight is given by:

$$\Delta w_{hj}(t+1) = w_{hj}(t) + \eta \sum_{j=1}^{J}\left(d_j - y_j\right)\dot{S}_j(y_j^h).s_h\left(y_j^h\right)$$

$$\Delta w_{ih} = \eta \sum_{j=1}^{N}\left(d_j - y_j\right)y_j^p.w_{hj}.\dot{S}_h(y_h^j).x_i^j$$

III. EXPERIMENTAL DESIGN

In this segment we are performing the actual training and experiment conducted on test data. Here we memorize the input pattern sets by minimizing the error between the desired and actual output for the 13 sets out of total possible 16 pattern sets between pattern set (0,0,0,0) to (1,1,1,1). In this experiment, we estimate the data storage with standard Back Propagation Model. The experiment described is designed to achieve the encryption pattern storage in the network.

3.1 Input Pattern Sets

Training of the system was carried out for following 13 input pattern sets:

0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0
0 1 1 0	0 1 1 1	1 0 0 0	1 0 1 0	1 0 1 1
1 1 0 0	1 1 1 0	1 1 1 1		

3.2 Test Pattern Sets

The following three pattern sets were chosen to be test patterns for validation:

0 1 0 1	1 0 0 1	1 1 0 1

3.3 Determination of Coupling Strength between the Neuron Layers

For estimating the encryption pattern of link keys, we first need to define the functional dependency between the encrypted link keys in the form of weights in the neural network.

Table 2. Initial Weights for the Network

	Initial Weights			
Between Input Layer to Hidden Layer	0.34130	0.67609	0.62793	0.20288
	0.03032	-0.11165	0.67340	-0.37649
	0.64180	0.47350	0.00865	0.04078
	0.48416	-0.64196	-0.11188	-0.8026
	0.18010	0.08688	-0.04788	-0.0884
	-0.21870	0.06733	-0.99540	-0.63465
	0.28370	0.77028	0.16267	0.42277
	0.30940	0.54900	0.00164	0.04050
Between Hidden Layer to Output Layer	-0.61014	-0.19135	-0.25260	0.33321
	-0.13367	-0.18642	-0.26149	0.66970
	0.05072	0.25839	0.54230	-0.47125
	0.10362	0.25053	0.23261	0.19510
	-0.72159	0.11920	0.06990	-0.38859
	0.24076	-0.97050	0.67718	-0.02120
	-0.51510	-0.07900	-0.09670	-0.42853
	-0.48730	-0.00450	-0.56635	0.10257

3.4 Algorithmic Implementation of Storage of the Memorized Input Pattern Sets

Step 1: First find the input pattern set from the encryption mechanism.

Step 2: Initialize the Feed Forward Network, which is to be trained using Model of Back Propagation.

Step 3 : Insert the input pattern set in the network.

Step 4: Train the above feed forward network using the Gradient Descent method and calculate the activation values and the output values,

$$\frac{\partial E^{p}}{\partial w_{hj}} = \frac{\partial E^{P}}{\partial s_j\left(y_1^h\right)} \cdot \frac{\partial s_j\left(y_j^h\right)}{\partial y_j^h}\left[s_h\left(y^h\right)\right]$$

Step 5: Adjustment of coupling strength between keys is made, so that the sample input pattern converges to the desired output pattern, that possess the same storage as possessed by the encryption algorithm. For this, adjust the weights of each link using equations:

$$\Delta w_{ih} = -\eta \sum_{j=1}^{J}\left(d_j - y_j\right)^2 . \left[s_h\left(y_1^h\right)\right]$$

Step 6 : Update the final weight using equation:

$$\Delta w_{hj}(t+1) = w_{hj}(t) + \eta \sum_{j=1}^{J}\left(d_j - y_j\right) . \dot{S}_j(y_j^h) . s_h\left(y_j^h\right)$$

$$\Delta w_{ih} = \eta \sum_{j=1}^{N}\left(d_j - y_j\right) y_j^p . w_{hj} . \dot{S}_h(y_h^j) . x_i^j$$

Step 7 : End.

IV. RESULTS

1.1 Memorized Pattern Set

Table 3. Memorized Pattern Set for Feed Forward Network trained by Back Propagation

S.No.	Input Data Set	Output Data Set
1.	0,0,0,0	0,0,0,0
2.	0,0,0,1	0,0,0,1
3.	0,0,1,0	0,0,1,0
4.	0,0,1,1	0,0,1,1
5.	0,1,0,0	0,1,0,0
6.	0,1,1,0	0,1,1,0
7.	0,1,1,1	0,1,1,1
8.	1,0,0,0	1,0,0,0
9.	1,0,1,0	1,0,1,0
10.	1,0,1,1	1,0,1,1
11.	1,1,0,0	1,1,0,0
12.	1,1,1,0	1,1,1,0
13.	1,1,1,1	1,1,1,1

4.2 Validation by Test Pattern:

Test Pattern	Output for Test Pattern (bit by bit) By The Network			
0 1 0 1	0.00910	0.99999	0.01202	0.99999
1 0 0 1	0.99999	-0.00116	0.00778	0.99999
1 1 0 1	0.99999	0.99999	0.17402	0.99999

5.3 Training Performance:

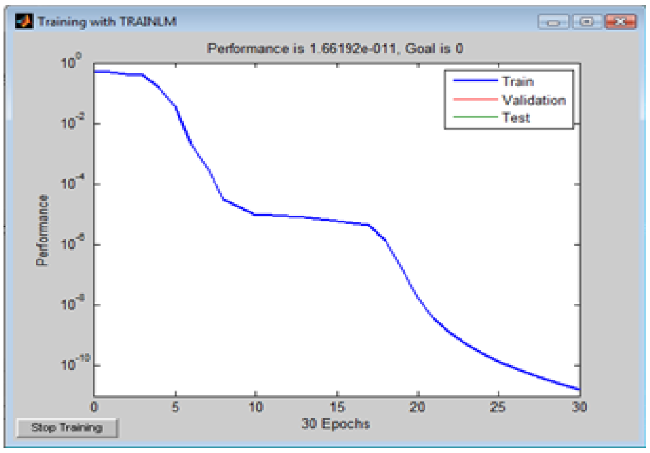

Fig 2: Graph for Network Performance during Memorized Pattern Set for Feed Forward Network trained by Back Propagation

V. DISCUSSION

With the above experiment we proved that the encrypted data files can be stored in the network which indicates that the model is working properly for finding data validation and storing them using the model of Back Propagation of Artificial Neural Networks. The strength of data depends on number of neurons in the model. Table 2 informs about the initial coupling strengths for the network. Table 3 shows the results for storing the memorized input patterns of the network. The mapping of every input pattern to an output class has been performed in about 1850 iterations each is also shown. Table 4 shows the convergence of sample input pattern sets into corresponding desired output class.

The simulation program was developed in MATLAB. It generates the initial weights randomly with the help of random generator. It helped with different epochs for the same algorithm within same network structure and same training data set.

VI. CONCLUSION

The results thus obtained shows that the encrypted files, which were easily cracked by hackers are more secure if the model of Back Propagation is used.

This model worked successfully for finding the storing and validation of data. The data security was maintained as the model mapped the pattern between the clear text and cipher text. These patterns were then used by the Feed Forward Network, trained using the model of Back Propagation. The model used Gradient Descent algorithm that generated the iterations using various mathematical equations and matched the input values with the desired values. Further calculations were minimized as the future data entries were passed through the trained network and the entries were matched with the output pattern wherein the matched patterns were allowed the access to the system. Moreover the error has been taken at 0.0001 units thus increasing the coupling strengths between the layers of network.

Thus it proves that Back Propagation Model of Artificial Neural Network by far, is a secure method than the traditional approaches of encryption and hashing.

REFERENCES

1. Preet Inder Singh, Gaur Sunder Mitra Thakur, "Enhanced Password Based Security System Based on User Behaviour using Neural Network," International Journal Information Engineering and Electronic Business,vol 2, pp 29-35, 2012.
2. Dr. B.S. Pradeep, S.Soumya, "Role of ANN in Secured Wireless Multicast Routing during Dynamic Channel Allocation for User Demanded Packet Optimality," Int. J.Advanced Networking and Applications,vol 3, issue 2, pp 1135-1139, 2011.
3. Khalil Shihab, "A Cryptographic Scheme Based on Neural Network," Proceedings of the 10th WSEAS International Conference on COMMUNICATIONS, Vouliagmeni, Athens, Greece, July 10-12, 2006, pp 7-12.
4. F.Chabaud, A. Joux.," Differential Collisions in SHA-0," Advances in Cryptology – Crypto'98, pringer-Verlag, , pp.56-71, August 1998.
5. National Institute of Standards and Technology, NIST FIPS PUB 186," Digital Signature Standard," U.S. Department of Commerce, 1994.
6. Kohn R., Van Hemmen J. L., "Self-Organizing Maps and Adaptive Filters", 1991.
7. Y.F.Yam and T.W.S. Chow, "Determining initial weights of feedforward neural networks based on least square method," Neural Processing Lett., vol.2, pp. 13-17,1995.
8. "A new method in determining the Initial weights of feedforward neural networks,"
9. Bruce Schneier,"Applied Cryptography", General reference book about crypto algorithms and protocols, aimed at implementers, 2nd edition.
10. John Cannaddy," Artificial Neural Networks for Misuse Detection", School of Computer and Information Sciences Nova Southeastern University Fort Lauderdale, FL 33314