Journal of Computer Science 2 (9): 710-715, 2006
ISSN 1549-3636

A Backpropagation Neural Network for Computer Network Security

Khalil Shihab
Department of Computer Science, SQU, Box 36, Al-Khod, 123, Oman

Abstract: In this paper, an efficient and scalable technique for computer network security is presented. On one hand, the decryption scheme and the public key creation used in this work are based on a multi-layer neural network that is trained by backpropagation learning algorithm. On the other hand, the encryption scheme and the private key creation process are based on Boolean algebra. This is a new potential source for public key cryptographic schemes which are not based on number theoretic functions and have small time and memory complexities. This paper along with test results show that the possibility of guessing keys is extremely weaker than using the Data Encryption Standard method (DES), which is a widely-used method of data encryption. The presented results are obtained through the use of MATLAB 6.5.1 software.

Key words: Security, encryption, decryption, neural networks

INTRODUCTION

The problem of protecting information has existed since information has been managed. However, as technology advances and information management systems become more and more powerful, the problem of enforcing information security also becomes more critical[1]. The massive use of the communication networks for various purposes in the past few years has posed new serious security threats and increased the potential damage that violations may cause. As organizations are increasing their reliance on computer network environments, they are becoming more vulnerable to security breaches. Private and public sectors more than ever today depend on the information they manage. A violation to the security of the information may jeopardize the whole system working and cause serious damages. Advances in artificial neural networks (ANNs) provide effective solutions to this problem[2], section 5 provides more details on ANNs.

The security problem is considered here as the problem of keeping communications over the network private. In other words, a secure network allows only the intended recipient to intercept and read a message addressed to her/him. Thus, protection of information is required against possible violations that can compromise its secrecy (or confidentiality). Secrecy is compromised if information is disclosed to users not authorized to access it. While the encryption scheme used in this work is based on Boolean algebra, the decryption scheme here is based on a neural network techniques that uses backpropagation learning algorithm.

Data encryption: The data transferred over Public infrastructure should be unreadable for illegal purposes. The fundamental of encryption technique is to map the data to a domain in a manner that is safe from sniffing. Two major techniques used in encryption are: Symmetric encryption and Asymmetric encryption[3, 4]. In Symmetric encryption method, a common key is shared among participants. This is used in both encoding and decoding processes. The sender encrypts the message (M) using a key (K) to generate the codeword (E), i.e.,

$$E = \text{Encrypt}(K, M) \quad (1)$$

The resulting codeword is decrypted using the common key after being sent through the network and is received by the receiver, i.e.

$$M = \text{Decrypt}(K, E) \quad (2)$$

In Asymmetric encryption model, two keys are assigned to each member: Private key, which is exclusively confined to the user and Public key, which is published among other members by the user. Theoretically, the encryption function is such that the message encrypted with the Public key, is not decrypted unless by the means of corresponding private key and the message encrypted with the private key, is not decrypted unless by the means of corresponding Public key. The relation between encryption and decryption by these two keys can be mathematically shown. If M is the message, Pub_Ui shows the ith user's Public key and Prv_Ui is the ith user's private key, then:

$$M=\text{Decrypt}[\text{Pub\_Ui}, \text{Encrypt}(\text{Prv\_Ui}, M)] \quad (3)$$

iff $<\text{Pub\_Ui}, \text{Prv\_Ui}> \in Ui$
where Ui is the ith user's key set.
Also:

$$M=\text{Decrypt}[\text{Prv\_Ui}, \text{Encrypt}(\text{Pub\_Ui}, M)] \quad (4)$$

iff $<\text{pub\_Ui}, \text{prv\_Ui}> \in Ui$

Corresponding Author: Khalil Shihab, Department of Computer Science, SQU, Box 36, Al-Khod, 123, Oman

Therefore, to send a private message, the sender encrypts the message using the receiver's Public key and transfers it through Public infrastructure. On the other side, the receiver decrypts the encoded message by the help of its own private key. An encryption mechanism can also be used to authenticate the sender of a message. The technique is known as a digital signature. To sign a message, the sender encrypts the message using his or her private key. The recipient uses the inverse function and the sender's Public key to decrypt the message. The recipient knows who has sent the message; because, only the sender has the key needed to perform the encryption. To ensure that encrypted messages are not copied and resent later, the original message can contain the time and date that the message was created. Interestingly, two level of encryption can be used to guarantee that the message is both authentic and private. First, the message is signed by using the sender's private key to encrypt it. Second, the encrypted message is encrypted again using the recipient's Public key. Mathematically, double encryption can be expressed as:

X=Encrypt[Pub_U2, Encrypt(Prv_U1,M)] (5)

Where M denotes a message to be sending, X denotes the string resulting from the double encryption, Prv_U1 represents the sender's private key and Pub_U2 denotes the recipient's Public key.

At the receiving terminal, the decryption process is the reverse of the encryption process. First, the recipient uses his or her private key to decrypt the message. The decryption removes one level of encryption, but leaves the message digitally signed. Second, the recipient uses the sender's Public key to decrypt the message again. The process can be expressed as:

M=Decrypt [Pub_U1, Decrypt(Prv_U2,X)] (6)

Where X denotes the encrypted string that was transferred across the network, M denotes original message, Prv_U2 denotes the recipient's private key and Pub_U1 denotes the sender's Public key. If a meaningful message results from the double decryption, it must be true that the message was confidential and authentic. The message must have reached its intended recipient because only the intended recipient has the correct private key needed to remove the outer encryption. The message must have been authentic, because only the sender has the private key needed to encrypt the message so that sender's Public key will correctly decrypt it[3, 4].

Model design: Suppose M is some N-bit initial unipolar data, i.e.,

$Mi = \{0, 1\}, 0 \le i \le N-1$ (7)

The encryption process includes two functions: Permutation and Doping. In the following subsections, we will describe these two functions.

Permutation function: This function contains a vector P of 2N elements, whose elements are in [0, 2 N-1] interval. This vector should not have repeated elements.

$P = < p0, p1, \ldots, p2N-1 >$ (8)

$0 \le pi \le 2N - 1$

$pi \ne pj$ If $i \ne j$

Where < p0, p1, … ,p2N- 1 > is an n-Tuple.

If we let Val (M) be as follows:

Val(M)=2N-1*MN-1+2N-2*MN-2+…+20*M0 (9)

Then the Permutations (Perm) can be defined as follows:

Perm (M) = Bin [Pval(M)] (10)

Where Bin (X) returns the binary form of X. In other words, the Permutation function maps the string M of value V onto a string located at Vth position of the P vector. Note that vector P includes 2N unrepeated elements reveals that the Permutation is a bijective function, i, e.,

$M \ne M' <==> Perm (M) \ne Perm (M')$ (11)

Doping function: This function includes an N'-element vector, D, whose elements are in [0, (N+ N'- 1)] interval. N' is a selective number. The vector D should contain no repeated elements as well. The Doping function makes the (N + N')-bit string E, from N-bit string S as follows:

* For each $i \in D$: Ei = Fi (S).
  In which Fi can be any Boolean function.
* N non-permutated elements of E are correspondingly permutated with S elements.

For example, suppose:

S = [0 1 0].

D = [0 2 5].

E0 = F0 (S) = S0 AND S1.

E2 = F2 (S) = S0 OR S1 OR S2.

E5 = F5 (S) = E0 OR E2.

Therefore,

E = [1 0 1 1 0 0].

Model description: The permutation function operates over an N-bit message M to produce an N-bit result S, i.e.,

S = Perm (M) (12)

S will be presented to Doping function, then the (N + N')- bit result, i. e, E, will be generated.

E = Dop (S) (13)

E is represented as the encrypted data. In this model, the private key is:

Prv_U = {P, D, N', Fi} (14)

Guessing the private key: For the 2N-element vector P, there are (2 N!) states and for the N'-element vector D whose elements are in the [0, (N + N'-1)] interval, there are (N+N')! /N! states. Also, there are 22N states for each of N-variable Fi functions. Consequently, even under the assumption that N' is known, the overall number of states is as follows:

Total States=2N!*(N+N')!/N!*2N'*2N (15)

It is clear that this space is much larger than the 256 state spaces through Data Encryption Standard (DES), which has been used widely, even by choosing small values for N and N'. As an example, if N=5 and N' = 1, then these values generate a space of 1029 times larger than that of DES.

Artificial neural networks (ANNs): A neural network is a massively parallel-distributed processor made up from simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. The use of neural network offers the Input-Output Mapping property and capability[2,5-9].

The ANNs learning algorithms can be divided into two main groups that are supervised (or Associative learning) and unsupervised (Self-Organization) learning[2, 5, 10]. Supervised learning learns based on the target value or the desired outputs. During training the network tries to match the outputs with the desired target values. It is presented with an example picked at random from the set and the synaptic weights of the network are modified to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an appropriate statistical criterion. The training of the network is repeated for many examples in the set until the network reaches a steady state, where there are no further significant changes in the synaptic weights. The previously applied training example may be reapplied during the training session but in a difference order. Thus the network learns from the examples by constructing an input-output mapping for the problem at hand[5].

Unsupervised learning method is not given any target value. A desired output of the network is unknown. During training the network performs some kind of data compression such as dimensionality reduction or clustering. The network learns the distribution of patterns and makes a classification of that pattern where, similar patterns are assigned to the same output cluster. The Kohonen Self-Organizing Map (SOM) network is the best example of unsupervised learning network[5].

SOM has been used to provide a graphical representation of the analysis, highlighting outliers that may suggest suspicious activity[6, 7, 10]. In our cryptography process, we used a feed-forward network implementing the back propagation algorithm[11, 12].

Using neural network to learn the public key: An encrypted message has (N+N') bits. However, it will have only 2N valid states. No other state is generated. To learn the Public key, the valid states are fed to a supervised neural network. We will expect that the initial message M will show up as the output. In other word, training set will be the following pairs:

{(E0, M0), (E1, M1)…(E2N-1, M2N-1)} (16)

Where Ej and Mj are encrypted string of length (N+N') and N-bit initial string, respectively.

Having been trained in this way, the structure and the weights of the network are presented as a Public key.

Pub_U=< Net, W > (17)

The backpropagation neural network: One of the most commonly used supervised ANN model is backpropagation network that uses backpropagation learning algorithm[2, 12, 13]. Backpropagation (or backprop) algorithm is one of the well-known algorithms in neural networks. The introduction of backprop algorithm has overcome the drawback of previous NN algorithm in 1970s where single layer perceptron fail to solve a simple XOR problem. The backpropagation neural network is essentially a network of simple processing elements working together to produce a complex output. These elements or nodes are arranged into different layers: input, middle and output. The output from a backpropagation neural network is computed using a procedure known as the forward pass[2, 5, 14, 16]:

* The input layer propagates a particular input vector's components to each node in the middle layer.
* Middle layer nodes compute output values, which become inputs to the nodes of the output layer.
* The output layer nodes compute the network output for the particular input vector.

The forward pass produces an output vector for a given input vector based on the current state of the network weights. Since the network weights are initialized to random values, it is unlikely that reasonable outputs will result before training. The weights are adjusted to reduce the error by propagating the output error backward through the network. This process is where the backpropagation neural network gets its name and is known as the backward pass:

* Compute error values for each node in the output layer. This can be computed because the desired output for each node is known.
* Compute the error for the middle layer nodes. This is done by attributing a portion of the error at each output layer node to the middle layer node, which feed that output node. The amount of error due to each middle layer node depends on the size of the weight assigned to the connection between the two nodes.
* Adjust the weight values to improve network performance using the Delta rule.
* Compute the overall error to test network performance.

The training set is repeatedly presented to the network and the weight values are adjusted until the overall error is below a predetermined tolerance. Since the Delta rule follows the path of greatest decent along the error surface, local minima can impede training. The momentum term compensates for this problem to some degree.

Cipher block chaining: In order to complicated decryption for illegal people, cipher block chaining can be used so that each plaintext block is XORed with the previous cipher block before being encrypted. Thus, the encryption will not be context free. The first block is XORed with the initial vector that is randomly selected. In other word, encryption steps will be as follows:

C0 = Encrypt (P0 Xor IV).
C1 = Encrypt (P1 Xor Ĉ0).
C2 = Encrypt (P2 Xor Ĉ1).
C3 = Encrypt (P3 Xor Ĉ2).
In general, Ci is as follows:

$$Ci = Encrypt (Pi Xor \hat{C}i-1) \quad (18)$$

Where IV is initial vector, Pi is ith plaintext, Ci is ith cipher text and Ĉi is a window cut of Ci so that the length of Ĉi be equal to the length of Pi. Decryption is also done via the following procedure:

P0 = IV Xor Decrypt(C0).
P1 = Ĉ0 Xor Decrypt(C1).
P2 = Ĉ1 Xor Decrypt(C2).
P3 = Ĉ2 Xor Decrypt(C3).
In general, Pi can be represented as follows:

$$Pi = \hat{C}i-1 \text{ Xor } Decrypt(Ci) \quad (19)$$

It is observed that encryption of ith block is a function of all plaintexts in block 0 through (i-1). Hence, depending on where the plaintext is located, different cipher texts are generated from the same text.

Implementation: In our simulation, an initial 12-bit data set has been used, (N=12) and 4 bits are doped through encryption process (N' =4). P & D vectors are produced randomly after considering essential conditions (unrepeated elements). We use the following Boolean functions:

F3 = S3 Xor S1.
F8 = S8 Xor S4 Xor S2 Xor S0.
F12 = S11 Xor S6 Xor S3.
F13 = S11 Xor S9 Xor S7 Xor S5.
Table 1 shows a few examples.

Table 1: Some of the encryption results

M	Perm((M)	Dop(S)
000000000000	101010111101	1000101101111101
000000011000	011011111000	0110101111111000
010101001100	111110110111	1101111001101111
010101010010	001101110111	0001110011101111
010101010111	100111010011	1000011110101011
010101011010	010111010101	0111011010100101
010101011110	000100101000	0011010101011000
010101100100	001100000101	0010110100000101

The neural network used in the decryption process is a 3-layer feed-forward network implementing the back propagation algorithm. There are 16 neurons in input layer, 24 neurons in the hidden layer and 12 neurons in the output layer. Figure 1 shows the architecture of the neural network. To implement our neural network we used the Neural Network Toolbox in MATLAB.

At the beginning of the learning process, the weight matrices between input and hidden layer (IW {1,1}) and between hidden and output layer (IW {2,1}) are initialized with the random values in the [-0.5, 0.5] interval. Vectors for hidden neuron biases (b {1}) and output neuron biases (b {2}) are also initialized with random values. In the hidden and output layers, the linear activation functions have been used. After several iterations, when the difference between the calculated output and the desired output is less than the threshold value, the iteration is stopped.

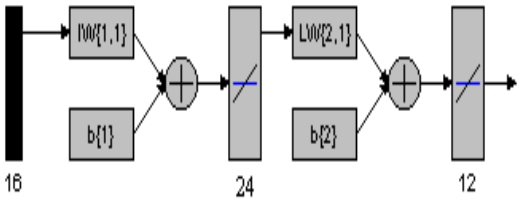

Fig. 1: Neural network architecture in decryption process

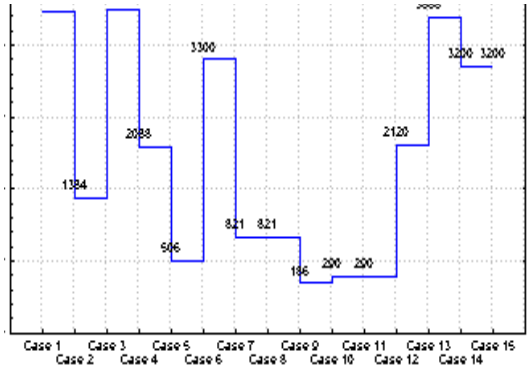

Fig. 2: Original signal before encryption

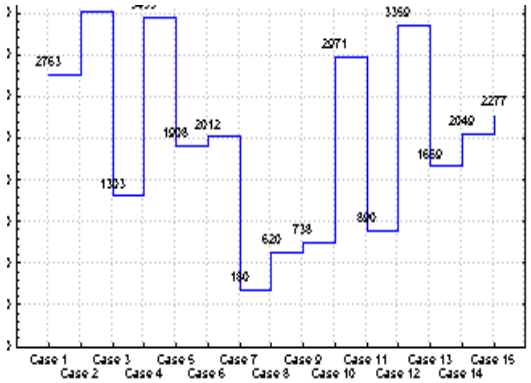

Fig. 3: XORed signal

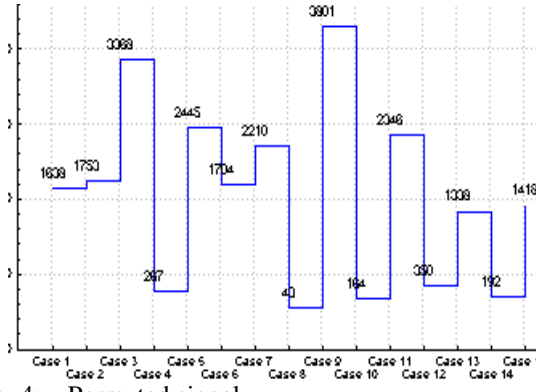

Fig. 4: Permuted signal

RESULTS

In order to evaluate the discussed mechanism, the encryption and decryption steps of a typical digital signal are shown below. Figure 2 shows original signal in a plain form. Figure 3 shows the signal in a chained form. The value of this signal in each time sample is the XORed of original signal value in the same time sample and previous time sample of encrypted signal. In our experiment, IV is "10101001100" and the window is put over the first twelve bits. Figure 4 and 5 shows the permuted and doped signal respectively. The result shows that the encryption mechanism is not a contextfree process. It is seen that, although that the original signal has same values in 7th and 8th time samples, but the encrypted signal has different values right in the same time samples. This condition is repeated in 11th and 12th time samples. Figure 6 and 7 show the artificial neural network outputs and decryption signals respectively.

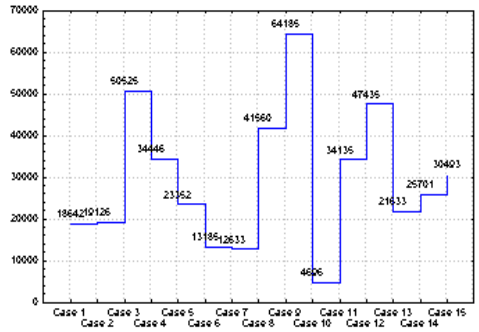

Fig. 5: Encrypted signal

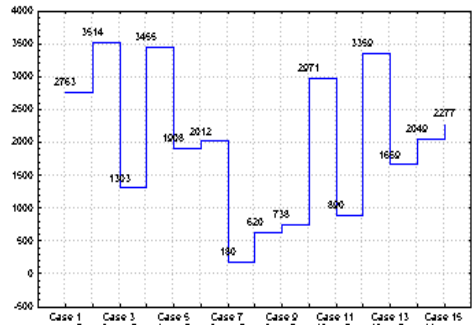

Fig. 6: Neural network output

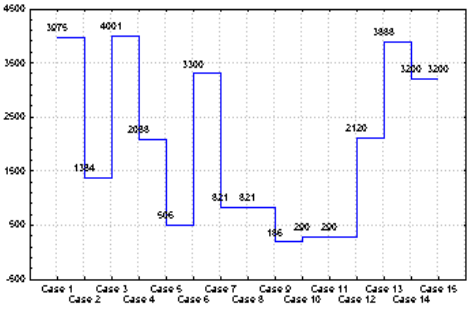

Fig. 7: Decrypted signal

CONCLUSION

As the computer network grow, the encryption mechanisms are of notable importance. In particular, the asymmetric encryption models have been always deeply considered because of their wide range of usage. However, finding two pair functions for encryption and decryption that satisfy the necessary conditions for providing computational strength and safety that has always been a serious problem.

In this work, we provide a new asymmetric encryption mechanism based on artificial neural networks. First, we presented the overall methods of encryption, and then we explored the necessary conditions of asymmetric methods. Next, we presented a model for the encryption mechanism that is based on Boolean algebra. We then used a neural network to learn the decryption mechanism. Finally, the simulation results showed that after training the artificial neural networks, it can be used effectively as a decryption function.

REFERENCES

1. Modaressi, A.R. and S. Mohan, 2000. Control and management in Next-Generation Networks: challenges and opportunities. IEEE Commun. Mag., 38: 94-102.
2. Russell, S. and P. Norvig, 2003. Artificial Intelligence: A Modern Approach. 2nd Edn. Prentice Hall, Inc.
3. Comer, D., 2001. Computer Networks and Internets with Internet Applications. 3rd Edn. Prentice Hall, Inc..
4. Alfred, J.M., P.C. van Oorschot and Scott, 2001. Handbook of Applied Cryptography. 5th Edn. CRC Press.
5. Kohonen, T., 1997. Self-Organizing Maps. Springler-Verlag, Berlin.
6. Sin, Y.L., W.L. Low and P.Y. Wong, 2002. Learning fingerprints for a database intrusion detection system. Proc. 7th Eur. Symp. Research in Computer Security, Zurich, Switzerland, pp: 264-280.
7. Lee, S.C. and D.V. Heinbuch, 2001. Training a neural network-based intrusion detector to recognize novel attacks. IEEE Trans. Systems, Man and Cybernetics Part A: Systems and Humans, 31: 294-299.
8. Zhou, T., X. Liao and Y. Chen, 2004. A novel symmetric cryptography based on chaotic signal generator and a clipped neural network. Advances in Neural Networks-ISNN, Intl. Symp. Neural Networks Proc., Part II. Lecture Notes in Computer Science, 3174: 639-644

9. Dokas, P., L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava and P. Tan, 2002. Data mining for network intrusion detection. Proc. NSF Workshop on Next Generation Data Mining.
10. Leonid, P., E. Eskin and S.J. Stolfo, 2001. Intrusion detection with unlabeled data using clustering. Proc. ACM CSS Workshop on Data Mining Applied to Security.
11. Wang, K. and S.J. Stolfo, 2004. Anomalous payload-based network intrusion detection. Proc. 7th Intl. Symp. Recent Advanced in Intrusion Detection (RAID), pp: 201-222.
12. Nong, Y., S. Vilbert and Q. Chen, 2003. Computer intrusion detection through EWMA for auto correlated and uncorrelated data. IEEE Trans. Reliability, 52: 75-82.
13. Li, C., S. Li, D. Zhang and G. Chen, 2004. Cryptanalysis of a chaotic neural network based multimedia encryption scheme. Advances in Multimedia Information Processing PCM 2004 Proc., Part III, Lecture Notes in Computer Science., Springer-Verlag, 3333: 418-425
14. Yen, J.C. and J.I. Guo, 2002. The design and realization of a chaotic neural signal security system. Pattern Recognition and Image Analysis (Advances in Mathematical Theory and Applications), 12: 70-79.
15. Li, S. and X. Zheng, 2002. Cryptanalysis of a chaotic image encryption method. Proc. IEEE Intl. Symp. Circuits and Systems, 2: 708-711
16. Lian, S., G. Chen, A. Cheung and Z. Wang, 2004. A chaotic-neural-network-based encryption algorithm for JPEG2000 encoded images. Advances in Neural Networks, Intl. Symp. Neural Networks Proc., Part II, Lecture Notes in Computer Science, 3174: 627-632.