ISSN: 2321-7782 (Online)
Volume 2, Issue 11, November 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Cryptography Algorithms using Artificial Neural Network

Arijit Ghosh[1]
Department of Computer Science
St. Xavier's College (Autonomous)
Kolkata – India

Dr. Asoke Nath[2]
Department of Computer Science
St. Xavier's College (Autonomous)
Kolkata – India

Abstract: ***In the recent years there has been quite a development in the field of artificial intelligence one of which has been the introduction of the artificial neural networks (ANN). The ANN can be considered as an information processing unit which to a great extent resembles the working of the human brain. Its utilization has spread through various fields namely bioinformatics, stock market predictions, medical science, weather forecasting etc. One of these fields in which ANN has been indispensable is cryptography. Recently there has been quite a study going on various encryption methods based on neural nets comprising of single layer or multilayer perceptron models. This field of cryptography is more popularly known as Neural Cryptography. The learning method of ANN architecture can be well utilized to generate more effective encryption systems based on feedback. In the present paper the authors tried to make a theoretical study on how ANN can be used in data encryption.***

Keywords: Stochastic algorithms; Cipher; Neurons; Synaptic Connections; Learning method; Layered Perceptron Models; Signum Function.

I. INTRODUCTION

The human brain has been the most efficient information processing unit known to man. The processing time for a brain is in milliseconds whereas that for a computer is in nanoseconds or microseconds. But some of the activities like face recognition is performed much faster by the brain. This can be due to two reasons:

1. Our brain has a complex and vast network of 10 billion neurons which are in turn connected to each other by 60 trillion interconnections. This can be considered as 10 billion processors working together synchronously to complete a task. Trying to imitate this type of architecture is futile as it is not feasible.
2. Another reason can be the superior learning mechanism of the brain that helps to solve problems more efficiently than the computer.

The learning mechanism can however be harnessed to create highly efficient feedback systems. As the brain consists of neurons, the Artificial Neural Networks consist of perceptron(s) that is very similar to the structure of a neuron. The comparison has been provided in fig. 1. Various learning methods have been devised based on the study of the working of the brain which can be grouped into 5 types namely:

1. Error correction method
2. Memory based learning method
3. Hebbian learning method
4. Competitive learning method
5. Boltzmann learning method

One of the most popular learning methods among them is the Hebbian learning method.

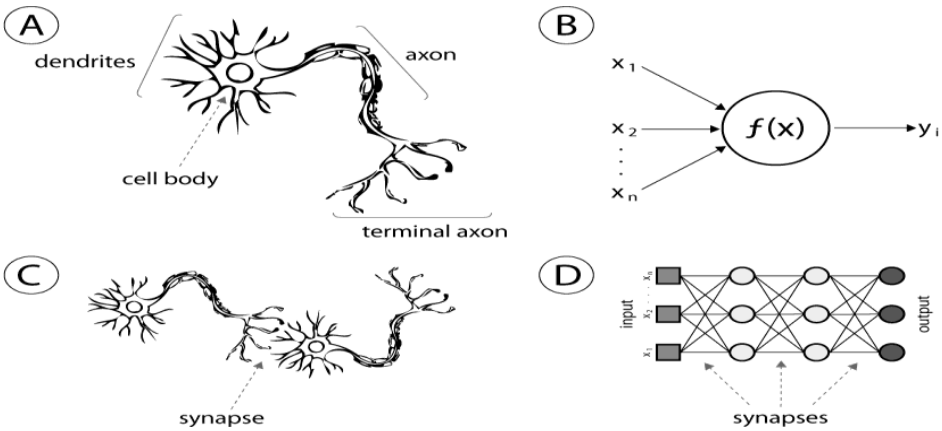

Fig. 1. Diagram to correlate between Biological and Artificial Neural Network

ANNs resemble the brain in two aspects:

1. Knowledge is acquired by the network from its environment through a learning process.

2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.

2. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.

3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

Three types of algorithms are used in cryptography in ANN:

a) Secret Key Cryptography

b) Public Key Encryption

c) Hash Functions

II. Proposed Work

The authors will try to analyze the various ANN algorithms present and then try to modify the existing system of cryptographic method to create a more reliable and effective system.

III. The Hebbian Learning Method

This is one of the most famous learning mechanisms of Neural Network. It has been named so in honor of neuropsychologist Hebb. He proposed that,

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.

This rule can be expanded into two-part rule (Stent, 1973; Changeux and Danchin, 1976):

1. If two neurons on either side of a synapse (connection) are activated simultaneously (i.e., synchronously), then the strength of that synapse is selectively increased.

2. If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

A. Mathematical model of Hebbian Learning Mechanism

In mathematical terms Hebb's hypothesis [1] can be formulated in the following manner. Let us consider the weights of the neural network to be 'w_{kj}' of neuron k with presynaptic and post synaptic signals x_j and y_k respectively. The adjustments applied to the synaptic weight w_{kj} at a time step n is expressed in the general form as

$$\Delta w_{kj} = f(y_k(n), x_j(n)) \qquad \text{(eq.1)}$$

where f(.) is a function applied to both presynaptic and postsynaptic signals. The signals $x_j(n)$ and $y_k(n)$ are often treated as dimensionless.

Hebb's hypothesis. The simplest form of Hebbian learning is described by

$$\Delta w_{kj} = \eta\, y_k(n)\, x_j(n) \qquad \text{(eq.2)}$$

where η is a positive constant that determines the rate of learning. Equation (eq.2) clearly emphasizes the correlational nature of a Hebbian synapse. It is sometimes referred to as the **activity product rule**. The repeated application of the input signal (presynaptic activity) $x_j(n)$ leads to an increase in $y_k(n)$ and therefore exponential growth that finally drives the synaptic connection into saturation. At that point no information will be stored in the synapse and selectivity is lost as shown in fig. 1. This typical property has been used to create a key exchange algorithm based on synchronization of neural networks proposed by I. Kanter and W. Kinzel[2].

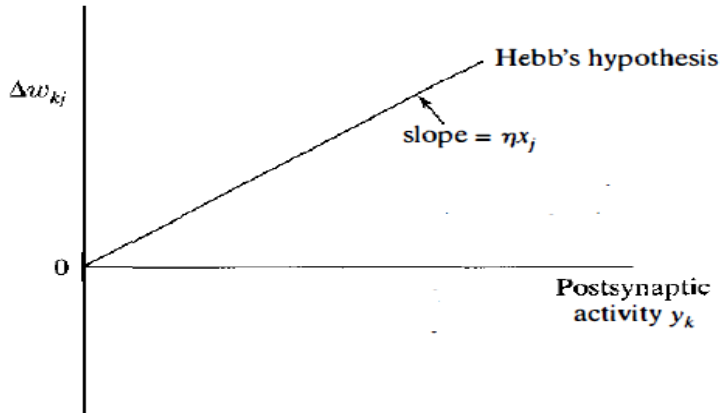

Fig. 2. Illustration of Hebb's hypothesis

IV. A REVIEW KEY EXCHANGE THROUGH SYNCHRONIZATION

I. Kanter and W. Kinzel suggested a public key exchange algorithm based on synchronization of the weights of two interconnected networks which can be considered similar to the case of pre and post synaptic neural connection.

The architecture used by the recipient and the sender is a two-layered perceptron, exemplified here by a parity machine (PM) with K hidden units. More precisely, the size of the input is KN and its components are denoted by x_{kj}, k = 1, 2,……. , K and j = 1,……, N. For simplicity, each input unit takes binary values, x_{kj} = +1/-1. The K binary hidden units are denoted by y_1, y_2 ,..., y_K.

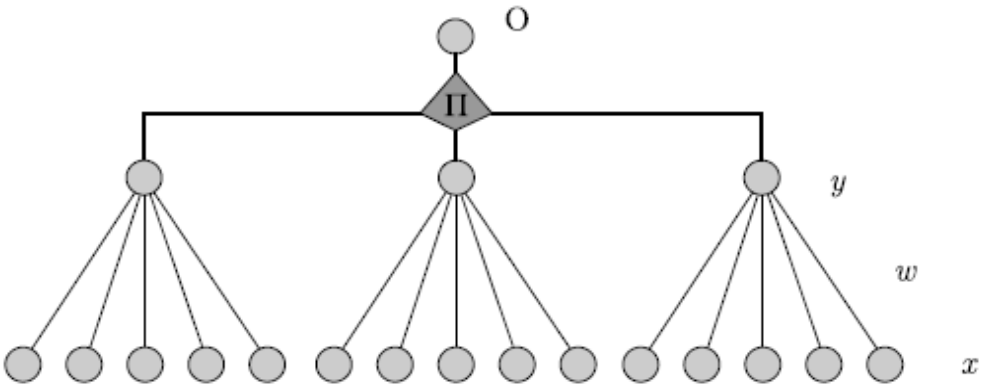

Fig. 3. Architecture of the networks: 3N input units x are transmitted by three weight vectors w to three hidden units y. The final output bit O is the product of the hidden units.

The architecture is characterized by non-overlapping receptive fields (a tree), where the weight from the jth input unit to the kth hidden unit is denoted by w_{kj} , and the output bit O is the product of the state of the hidden units (see Fig. 2). For simplicity the PMs are discussed with three hidden units K = 3. Integer weights bounded by L are used, i.e., w_{kj} can take the values -L, -L + 1, …..., L.

The initial weights w_{kj}^{S} and w_{kj}^{R} for the sender(S) and receiver(R) is selected by them which in turn acts as their common secret key. As in fig. 2, the sender and receiver may have 6N (3N for each) integer numbers. The sender and receiver network is then fed with random common public inputs (x_{kj}). The output bit O of each is sent to the other for the purpose of learning and synchronization.

The output bit O is calculated in the following steps:

Step 1: The state of the hidden fields is calculated from the following expression:-

$$y_k^{S/R} = \text{sign}[\Sigma_{j=1 \text{ to } n}\, w_{kj}^{S/R}\, x_{kj}] \qquad \text{(eq.3)}$$

In case of a zero field the corresponding hidden unit is set to 1/-1.

Step 2: In this step the output is calculated from the hidden units as

$$O^{S/R} = y_1^{S/R}\, y_2^{S/R}\, y_3^{S/R} \qquad \text{(eq 4)}$$

Both the sender and receiver is trained on the basis of their partner's output bit. If their outputs do not agree i.e. $O^S O^R < 0$, then the following learning method is applied,

$$\text{if } \left(O^{S/R} y_k^{S/R} > 0\right) \text{ then } w_{kj}^{S/R} = w_{kj}^{S/R} - O^{S/R}\, x_{kj}$$
$$\text{if } \left(|w_{kj}^{S/R}| > L\right) \text{ then } w_{kj}^{S/R} = \text{sign}(w_{kj}^{S/R})\, L$$

Fig. 3 The Hebbian Learning Method for the key exchange

Only those weights which belong to the hidden units whose state is same as that of the output unit are updated in both the networks.

V. Demerits of the Key Exchange Algorithm

The threat this algorithm faces is from **fabrication** or **masquerading**. Masquerading is a type of attack where the attacker presents himself to be someone else which in this case is the sender or the receiver. This is also called **the man-in-the-middle attack**.

Fig. 4 Illustration of the man-in-the-middle attack

Considering that the attacker can very well hide his/her identity for a considerable amount of time the attacker can try to synchronize with both the sender and the receiver being the middle man. As per the algorithm the random inputs are public as is the output bit. All the attacker needs to do is synchronize its weight bits with both the sender's and receiver's outputs. But instead of sending the sender's output to the receiver it can send its own output. This is called **denial-of-service**.

So, practically what happens is that the middle man can create its own antiparallel synchronized weight bits with respect to both the sender and receiver, but the weights of the sender and receiver will not be synchronized.

An even more threatening attack can be to let the sender's and receiver's output bits reach each other. This requires less effort for the attacker and the partners involved will never identify the snooping involved. The key exchange algorithm requires at most 400 bits approx. (for very large numbers) or even less to synchronize with each other. So, the attacker has to fabricate himself for a very limited time.

VI. Possible Solution to the Attack

A middle man attack can only be stopped using digital signatures or digital certificates or other methods like NONCE. Exchange of these has to be done on behalf of both the participants to ensure both are safe from any type of masquerading.

This can possibly be done in two ways:

1. If it is a denial-of-service attack, an encrypted dummy text can be sent by the sender to the receiver along with its own digital certificate after the synchronization is complete. The receiver is also needed to send the decrypted text back to the sender to ensure its authenticity. This has to be done both ways simultaneously to ensure the channel is secure. The NONCE method is also applicable.

   If the channel is insecure the receiver is bound to have a different key than that of the sender as it is a denial-of-service attack. So, whoever sends the reply has to have a wrong decrypted text or an invalid digital certificate. The digital certificate can be verified by a third party as well.

2. If the output of the sender is forwarded to the receiver then the above way won't work as the sender, receiver and the attacker will have the same key. In this case the sender and receiver has to agree on a secure digital signature algorithm like DSA, ECDSA etc. Each time a change occurs to the weights of two or more (can be changed according to requirement) hidden units, a digital signature verification must be done between the two to ensure that there is no third party trying to synchronize with them. If so happens the reply to the digital signature verification will be a wrong one and insecurity of the channel can be realized.

VII. Proposed Modification to the Key Exchange Algorithm

A. Exchanging String as a Key

The key exchange algorithm suggested by I. Kanter and W. Kinzel can be effectively used for bit encryption methods. But many a times one requires keys for simple text encryption or byte encryption. The authors propose the present algorithm in a form that can be very easily used to exchange keys involving characters and integers (including special characters).

The parity machine is designed as a two layer perceptron- the first layer involving the public random inputs and second layer involving the hidden units. The calculation of the hidden units and the final output bit is to be done in the very same way as that of the previous algorithm. The change is to be made in the signum function involving the calculation of the hidden units (see eq. 3) in a way as follows:

$$\text{if } y_k^{S/R} = 0,\ y_k^{S/R} = 0$$

$$\text{if } y_k^{S/R} > 0,\ y_k^{S/R} = 1$$

$$\text{if } y_k^{S/R} < 0,\ y_k^{S/R} = 1/0$$

Instead of using 1/-1, 1/0 is used as weights. The network can be designed in a way which consists of N number of hidden layers (depends on the size of the key needed) and each hidden layer consists of eight input units. The learning method (fig. 3) is applied when O^S **XOR** $O^R = 1$ i.e. when the outputs do not match.

The weights after synchronization can be divided into groups of eight to get the ASCII codes of the characters. These characters can be used as key for any kind of encryption. In case one predefines the size of the key, salting of the key or use of hash function can be done to increase or decrease the size of the key.

A simpler method of key exchange could be to change the weights in the state of -1 to a state of 0. With this mechanism one does not need to make any changes to the initial formula (eq.3) or the condition of the learning rule activation.

B. Increasing the range of the weights

Brute force on keys with binary value makes the key quite vulnerable. A possible solution to the problem can be to increase the range of values that the weights can take. This can be done using a **sigmoid** activation function instead of a signum activation function. This enables the weights to have a varied range of values which makes it difficult for the attacker to apply a brute force attack. The activation function [1] can be of the form,

$$\varphi(v) = \frac{1}{1+\exp^{(-av)}}$$

where 'a' controls the slope of the graph and v is the output.

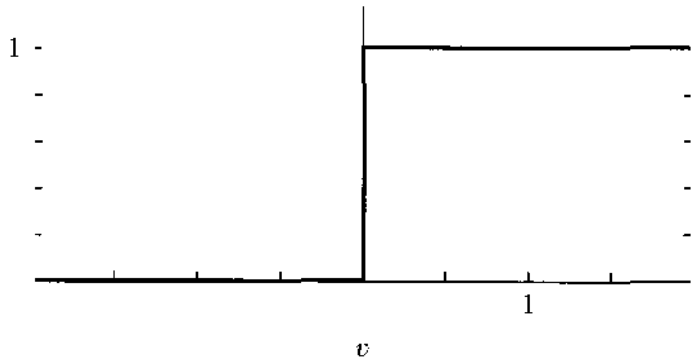

Fig.4 Graph for a signum function

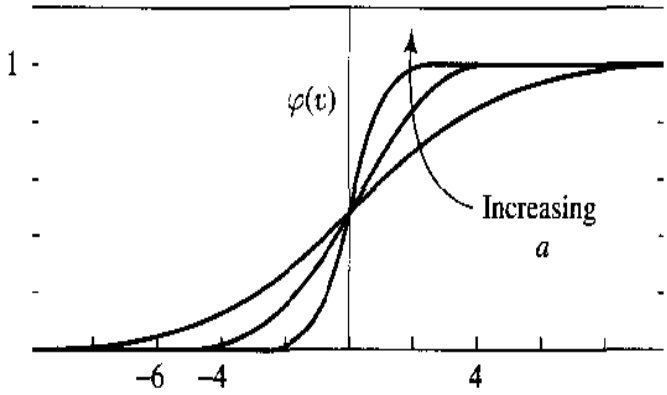

Fig.5 Graph for a sigmoid function with parameter 'a' controlling the slope

As the graph suggests, sigmoid functions are capable of generating a large range of weights including floating point weights which adds to the complexity of the problem. The value of 'a' can be controlled to make activation function act as a signum function (extreme case) as well.

VIII. Conclusion and Future Scope

The proposed key exchange algorithm can have varied applications in cryptography. One can apply ANN in Block Ciphers or Stream Cipher or Bit Level or Byte Level Encryption methods. The ANN method can be used in efficient manner to make the encryption process very hard to break and at the same time very reliable. The common attacks in cryptography such as known plain text attack, Brute Force Attack, Differential Attack may not be so easy in case of data encryption using Neural Networks.

References

1. Simon Haykin, Neural Networks A Comprehensive Foundation, Prentice Hall International Inc., ISBN: 0-13-908385-5
2. I. Kanter, W. Kinzel, The Theory of Neural Networks and Cryptography, Quantum Computers and Computing, V. 5, No. 1, 2005
3. William Stallings, Cryptography and Neural Security, ISBN: 978-93-325-1877-3
4. An Introduction to the Modelling of Neural Networks, P. Peretto, ISBN 0-521-41451-2 hardback, ISBN 0-521-42487-9 paperback
5. Use of Artificial Neural Network in the Field of Security, ISSN 2230-7621, MIT International Journal of Computer Science & Information Technology Vol. 3, No. 1, Jan. 2013, pp. 42–44

Author(s) Profile

Dr. Asoke Nath, Associate Professor, Department of Computer Science. St. Xavier's College (Autonomous), Kolkata. Presently he is involved in research work in Cryptography and Network Security, Steganography, Image processing, Green Computing, MOOCs, E-learning Methodologies, Neural network. He has published more than 116 papers.

Arijit Ghosh is presently pursuing M.Sc. in Computer Science. He has completed his graduation from Jogesh Chandra Chaudhuri College. He is presently involved in research work in chaotic neural networks and Cryptography.