Cryptography Using Neural Network

T. Godhavari, N. R. Alamelu and R. Soundararajan

Abstract - **The goal of any cryptographic system is the exchange of information among the intended users without any leakage of information to others who may have unauthorized access to it. In 1976, Diffie & Hellmann found that a common secret key could be created over a public channel accessible to any opponent. Since then many public key cryptography have been presented which are based on number theory and they demand large computational power. Moreover the process involved in generating public key is very complex and time consuming. To overcome these disadvantages, the neural networks can be used to generate common secret key. This is the motivation for this present work on interacting neural networks and cryptography[1].In the case of neural cryptography, both the communicating networks receive an identical input vector, generate an output bit and are trained based on the output bit. The dynamics of the two networks and their weight vectors is found to exhibit a novel phenomenon, where the networks synchronize to a state with identical time -dependent weights. This concept of synchronization by mutual learning can be applied to a secret key exchange protocol over a public channel. The generation of secret key over a public channel has been studied and the generated key is used for encrypting and decrypting the given message using DES algorithm which is simulated and synthesized using VHDL**

Keywords - **neural cryptography, mutual learning, time-dependent weights**

I. Introduction

Artificial neural networks are massively parallel adaptive networks of simple nonlinear computing elements called neurons which are intended to abstract and model some of the functionality of the human nervous system in an attempt to partially capture some of its computational strengths.[2] A novel phenomenon of dynamic neural network is applied in cryptography systems. The limitation in the general cryptographic process led to the development of cryptographic systems with shorter keys called the secret key systems. The security of such cryptographic system depends on the secrecy of the key. Public key cryptosystems solve the distribution problem by using different secret keys for the internal users. Thus key exchange between users is not required and the security of such system depends on the computational complexity. But in large systems the distribution of key between the users and key bank is still a bottleneck. Moreover, the problem of passive eavesdropping is still not addressed. So the special characteristic of neural networks can be used in generating secret key over public channel.

Dr.M.G.R. Educational & Research Institute,
Chennai – 600 095. India
E-mail nralamelu@yahoo.com nralamelu@yahoo.com

0-7803-9503-4/05/$20.00 ©2005 IEEE

II. Neural Cryptography

A. Interacting neural network and cryptography:

Two identical dynamical systems, starting from different initial conditions, can be synchronized by a common external-signal which is coupled to the two systems.

Synchronization has recently been observed in artificial neural networks as well. Two networks which are trained on their mutual output can synchronize to a time dependent state of identical synaptic weights [3].This phenomenon has been applied to cryptography as well [4]. In this case, the two partners A and B do not have to share a common secret but use their identical weights as a secret key needed for encryption. The secret key is generated over a public channel. An attacker E who knows all the details of the algorithm and records any communication transmitted through this channel finds it difficult to synchronize with the parties, and hence to calculate the common secret key. Synchronization by mutual learning (A and B) is much faster than learning by listening(E).Neural cryptography is much simpler than the commonly used algorithms which are mainly based on number theory [5] or on quantum mechanics [6].

B. Basic Algorithm:

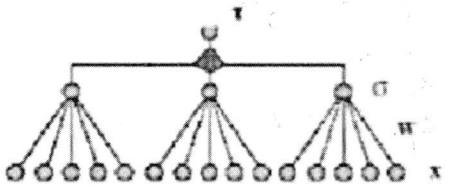

Fig 1.Tree Parity Machine

The mathematical model used for representing a party is a Tree Party machine (TPM) as shown in figure 1. It consists of k different hidden units; each of them is a perceptron with an n-dimensional weight vector wk, when a hidden unit k receives an n-dimensional input vector xk it provides input bit

$$\sigma k = \text{sign}\,(wk \,.\, xk) \qquad (1)$$

Output bit τ of the total network by

$$\tau = \prod_{i=1}^{k} \sigma_i \qquad (2)$$

The input vector is generally a binary input such that xk Î{ -1,+1 } and both discrete or continuous weights wk can be considered. However, generally discrete weights are considered for ease in explanation.

Each of the two communication parties A and B has their own network with an identical TPM architecture. Each party selects a random initial weight vector wk (A) and wk (B) at t = 0.Both the networks are trained by their output bits τ (A) and τ (B). At each training step, the two networks receive the common input vector xk and the corresponding output bit ô of its partner. The following is the learning rule

1. If the output bits are different, τ(A) ≠ τ(B) , nothing is changed
2. If τ (A) = τ (B) = τ only the hidden units are trained which have an output bit identical to the common output σk (A/B) = τ (A/B)
3. Three different rules can be considered for training
4. Anti- Hebbian learning, Hebbian learning ,Random walk.

If any component wk moves out of the interval [- L, L] it is replaced by sign (wk)L. This can treated as a random walk with reflecting boundaries. Using this algorithm the two neural networks synchronize to a common secret key.

III. Design Considerations

Method of applying Hebbian rule

The two output bits are exchanged and used for the mutual learning and synchronization. At each training step, the machines A, B etc. receive identical input vectors x1, x2 etc. The training algorithm is as follows.

Only if all output bits are equal i.e τ(A) = τ(B) =τ τhe weights can be changed. In this case, only the midden unit 'i' whose output bit is si is ι dentical to the actual output bit ô changed its weights using the following rule

$$wi(A/B)\,(t+1) = wi(A/B)\,(t)+xi \qquad (3)$$

If the learning step pushes any component weight out of the interval -L....L, then the component is replaced by ± L. The benefit of this scheme is that the partner as well as any opponent does not know which one of the weight vectors is being updated.

IV. Secret Key Generation

A. Key Generation

The different stages in the secret key generation procedure which is based on neural networks, can be stated as follows [10]

1. Determination of neural network parameters:k, the number of hidden layer units,n, the input layer units for each hidden layer unit,l, the range of synaptic weight values
2. The network weights to be initialized randomly
3. The following steps are repeated until synchronization occurs.
4. Inputs are generated by a third party (say the server) or one of the communicating parties
5. The inputs of the hidden units (ó) are calculated
6. The output bit ô is generated and exchanged between the networks
7. If the output values agree for each of ói which agree with the output value ô, the corresponding weights are modified using the Hebbian learning rule
8. When synchronization is finally occurred, the synaptic weights are same for both the networks.

B. Use of continuous weight

The range of weight values that the perceptrons take can be continuous, instead of being discrete. By using continuous weights, the protocol is enhanced because of the precision involved. [7]

V. Cryptanalysis

Any attacker who knows all the details of the protocol and all the information exchanged between A and B should not have the computational power to calculate the secret key.

Assume that the attacker E knows the algorithm, the sequence of input vectors and the sequence of output bits. In principle, E could start from all of the (2L+ 1)3N initial weight vectors and calculate the ones which are consistent with the input/output sequence. It has been shown, that all of these initial states move towards the same final weight vector, the key is unique [8]. However, this task is computationally infeasible. This is not true for the simple perceptron [9,10]. The most successful cryptanalysis has two additional ingredients: First, an ensemble of attackers is used. Second, E makes additional training steps when A and B are quiet .Therefore, in the limit of sufficiently large values of L neural cryptography is secure.

VI. Results And Discussion

A. Simulation Results

The data set obtained for synchronization time by varying number of input units (n) is shown in fig 2. The number of iterations required for synchronization by varying number of input units(n) is shown in fig 3. The two figures show that as the value of n increases, the synchronization time and number of iterations also increases. A peculiar but favorable behavior is observed at n = 101. In this case both synchronization time and number of iterations required are very less.Figure 2 &3 show the behavior before and after n = 101. Though this behavior is in explicable at present, the value of input units (n) = 101 is chosen to be optimal.

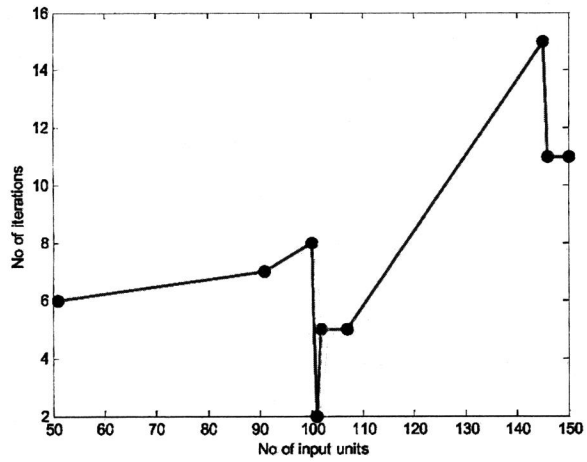

Fig.2. Simulation Result
No of Input Units Vs No of Iteration

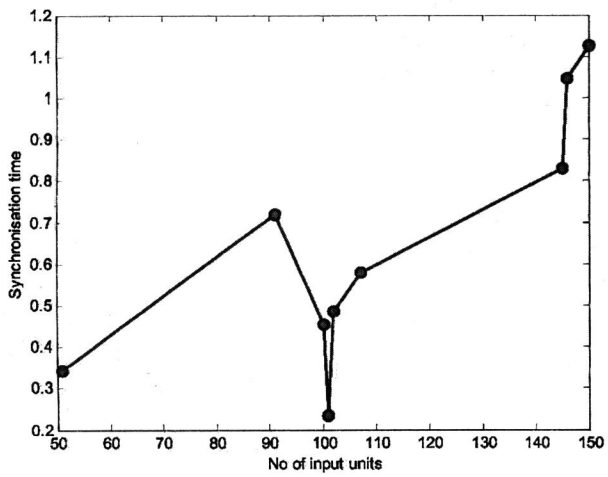

Fig. 3. Simulation Results
No of Input Units Vs Sync. time

A.VHDL Implementation - DES algorithm

The DES (Data Encryption Standard) algorithm is the most widely used encryption algorithm.DES works by encrypting groups of 64 message bits, which is the same as 16 hexadecimal numbers. In this present work 56 bit key is generated using neural cryptography and used in VHDL implementation of DES algorithm.Figure4 shows the VHDL timing diagram for encryption.

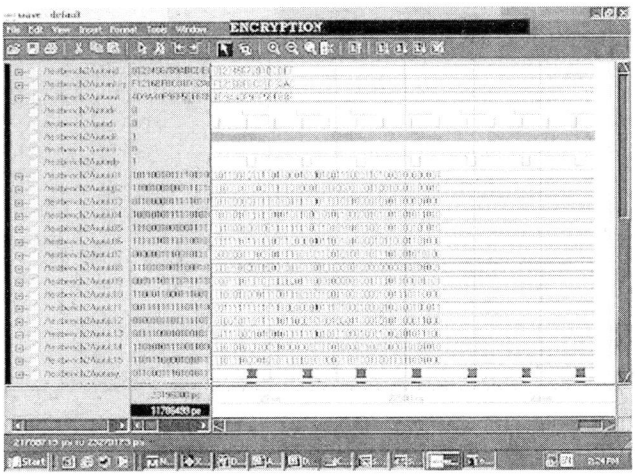

Fig.4. VHDL timing diagram for DES Encryption

VII. Conclusion

Interacting neural networks have been calculated analytically. At each training step two networks receive a common random input vector and learn their mutual output bits. A new phenomenon has been observed: Synchronization by mutual learning. The two partners can agree on a common secret key over a public channel. An opponent who is recording the public exchange of training examples cannot obtain full information about the secrete key used for encryption .This works if the two partners use multilayer networks, parity machines. The opponent has all the information (except the initial weight vectors) of the two partners and uses the same algorithms. Nevertheless he does not synchronize.

This phenomenon may be used as a key exchange protocol. The two partners select secret initial weight vectors, agree on a public sequence of input vectors and exchange public bits. After a few steps they have identical weight vectors which are used for a secret encryption key. For each communication they agree on a new secret key, without having stored any secret information before. In contrast to number theoretical methods the networks are very fast; essentially they are linear filters, the complexity to generate a key of length N scales with N (for sequential update of the weights). In fact, ensembles of opponents have a better chance to synchronize. These may be good news for a possible attacker.

In the new network based approach is faster than conventional (number theory) methods.In this paper neural network based mutual learning approach for secret key generation has been attempted as hardware implementation which would be still faster.

VIII. Future Work

The new concepts of public key cryptography, which are not based on number theory method can be used for multi party key exchange protocols. Future research can make use of the secret key generated using neural networks in advanced algorithms like triple DES to improve the security of neural cryptography.

References

[1] Wolfgang Kinzel and Ido 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

[2] Jacek M. Zurada, Introduction to Artificial Neural Systems.

[3] R. Metzler and W. Kinzel and I. Kanter, Phys. Rev. E, 62, 2555 (2000).

[4] I. Kanter, W. Kinzel and E. Kanter, Europhys. Lett, 57,141-147 (2002).

[5] D. R. Stinson, Cryptography: Theory and Practice (CRC Press 2002).

[6] C.P. Williams and S.H. Clearwater, Explorations in Quantum Computing, Springer Verlag, 1998.

[7] G. K. Patra, Tahir Ali, Anil Kumar and R. P. Thangavelu, "Multiparty Secure Key Exchange Algorithm using Neural Cryptology", National Workshop On Cryptology, 2004.

[8] R. Urbanczik, Private communication.

[9] D. R. Stinson, Cryptography: Theory and Practice, CRC Press 1995.

[10] M. Rosen-Zvi, E. Klein, I. Kanter and W. Kinzel, "Mutual learning in a treeparity machine and its application to cryptography", Phys. Rev. E (2002)