International Journal of Computer Theory and Engineering, Vol. 2, No. 4, August, 2010
1793-8201

Private Inputs to Tree Parity Machine

Pravin Revankar, W. Z. Gandhare and Dilip Rathod

***Abstract*—Neural cryptography is based on synchronization of tree parity machines by mutual learning. We extend previous key-exchange protocols by replacing random inputs with queries depending on the current state of the neural networks. The results show that queries restore the security against attackers. We further restrict amount information available to an attacker by keeping seed of pseudo random number generator private.**

***Index Terms*—Neural cryptography, query, mutual learning, tree parity machine, neural synchronization, encryption/decryption.**

I. Introduction

In case of neural cryptography, two identical dynamic systems (neural network), starting from different initial condition receive an identical input vector, generate an output bit and are trained based on the out bit. In this case two parties A and B do not have to share common secret but use their identical weights as secret key needed for encryption. The dynamics of two networks and their weight vectors found exhibit a novel phenomenon, where network synchronize to a state with identical time dependent weights. This concept of synchronization by mutual learning can be applied to secret key exchange protocol over a public channel has been studied and generated key is used for encryption and decryption given message. This method of symmetric key exchange method based on the fast synchronization of two identically structured Tree Parity Machines (TPMs) was proposed by Kanter and Kinzel [3]. The algorithm does not operate on large numbers and methods from number theory [9]. Two parties involved are allowed to perform the cryptographic process of (entity) authentication. In the area of cryptography, authentication is an important step still before key exchange or even the encryption/decryption of information with generated secret key [8].

The security of neural cryptography is still being debated. Since the method is based on a stochastic process, there is a small chance that an attacker synchronizes to the key, as well. However, it has been found that the model parameter L determines the security of the system [8, 1].

Use of query is in this work to enhance the security. Query: input vectors which are correlated with the present weight vector wk (t). The random inputs are replaced by queries, which A and B choose alternatively according to their own weight vectors. At odd (even) time steps the partner A (B) is generating an input vector which has a certain overlap to its weights wA (wB). It turns out that queries improve the security of the system.

Pravin Revankar and W. Z. Gandhare are with the Government College of Engineering,Aurangabad, India. (email: prevankar@gmail.com, wz_gandhare@yahoo.com).
Dilip Rathod is with the Dept. of Information Technology, P. E. S. College of Engineering, Aurangabad, India(email: rathod.dt@gmail.com).

II. Artificial Neural Network

Artificial Neural Networks (ANN) is the system of elements interacting by adaptive couplings which are trained from a set of examples. After training they function as content addressable associative memory, as classifiers or as prediction algorithms. A general form of an ANN is shown in fig1. It consists of three layers. One or more hidden layers might be used in the structure. Neurons in the input layer can be treated as buffer and distribute *xi* input signal to neurons in hidden layer. Output of each neuron *j* in the hidden layer is obtained from sum of multiplication of all input signals x_{kj} and weights w_{kj} that follows all these input signals. The sum can be calculated as a function of *yj* and can be expressed as:

$$y_k = f(\sum w_{ji} x_j) \qquad (1)$$

where f can be a simple threshold function, sigmoid function or any another function suitable to the problem under consideration. The outputs of the neurons in other layers are calculated with the same way. ANNs might be trained with many different learning algorithms. The weights are adopted according to the error occurred in the calculation with the help of a learning algorithm.

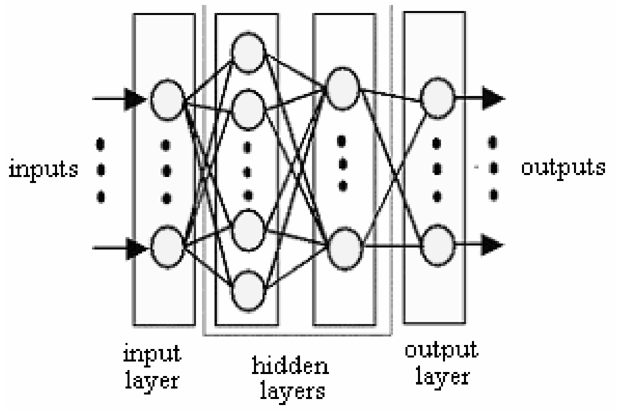

Figure 1. General form of ANN

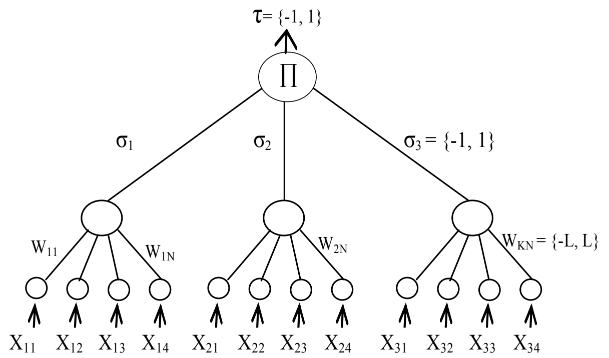

Figure 2. Tree parity machine with L=[-4,4], K=3 and N=4

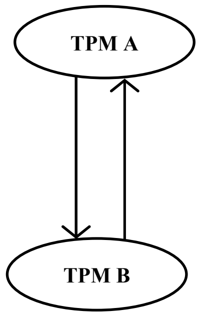

Figure 3. Outputs on commonly given inputs are exchanged between parties A and B for adaptation of their preliminary key.

Artificial neural networks have been applied to solve many problems. Learning, generalization, less data requirement, fast computation, ease of implementation, and software and hardware availability features have made ANNs very attractive for the applications. These fascinating features have also made them popular in cryptography as well.

- Neural cryptography presents a new approach based on artificial neural networks (ANNs) for data security in electronic communication.
- Neural cryptography is much simpler than the commonly used algorithms which are mainly based on number theory and have small time and memory complexities.
- Neural cryptographic algorithms are simple and fast to implement.

III. Tree Parity Machine

A multilayer feed forward newral network so called as Tree Parity Machine (TPM) as shown in fig. 2. The TPM has k hidden units ($1 \le k \le K$). Each hidden receives N different inputs ($1 \le j \le N$) leading to an input of size K×N. Each input take binary values, $x_{kj} = \pm 1$ and weight associated with inputs w_{kj} bounded by [L, -L]. The binary hidden units are denoted by $\sigma_1, \sigma_2, \ldots \sigma_K$ and the output bit τ is the product of the state of the hidden units.

IV. Generation of Queries

A query is based on input vector generated based on local field of weight vectors. The queries are generated alternatively by A & B and exchanged. The query replaces public input generated using pseudo random number generator, as used in basic neural cryptography algorithm [1].

With inclusion of query, the synchronization process now depends not only on the synaptic dept of TPM, but also on queries.

The query generation procedure is described as follows. As both weights $w_{k,j}$ and inputs $x_{k,j}$ are discrete, there are only 2L+1 possibilities for $w_{k,j} \cdot x_{k,j}$. Therefore we can describe the solution by counting the number $c_{k,l}$ of products with $w_{k,j} \cdot x_{k,j} = l$. Then the local field is given by:

$$h_k = \frac{1}{\sqrt{N}} \sum_{l=1}^{L} (c_{k,l} - c_{k,-l}) \tag{2}$$

In our simulations we use the following algorithm to generate the queries. First the output σ_k of the hidden unit is chosen randomly. Therefore the set value of the local field is given by $h_k = \sigma_k H$. Then we use either

$$c_{k,l} = \left\lfloor \frac{n_{k,l}+1}{2} + \frac{1}{2l}\left(\sigma_k H\sqrt{N} - \sum_{j=l+1}^{L} j(2c_{k,j} - n_{k,j}) \right) \right\rfloor \tag{3}$$

or

$$c_{k,l} = \left\lceil \frac{n_{k,l}-1}{2} + \frac{1}{2l}\left(\sigma_k H\sqrt{N} - \sum_{j=l+1}^{L} j(2c_{k,j} - n_{k,j}) \right) \right\rceil \tag{4}$$

to compute the values of $c_{k,L}, c_{k,L-1}, \ldots, c_{k,1}$. In each calculation one of the two equations is selected randomly with equal probability, so that rounding errors do not influence the average result. Additionally, we have to take into account that $0 \le c_{k,l} \le n_{k,l}$. Therefore we set $c_{k,l}$ to the nearest boundary value, if (3) or (4) yield a result outside this range.

Afterward the input vector x_k is generated. Inputs associated with zero weights are chosen randomly, because they do not influence the local field. The other input bits $x_{k,j}$ are divided into L groups according to the absolute value $l = |w_{k,j}|$ of their corresponding weight. In each group, $c_{k,l}$ inputs are selected randomly and set to $x_{k,j} = \mathrm{sgn}(w_{kj})$. The remaining $n_{k,l} - c_{k,l}$ input bits are set to $x_{k,j} = -\mathrm{sgn}(w_{k,j})$.

In order to achieve a secure key exchange with queries the partners have to choose the parameter H in such a way that they synchronize quickly, while an attacker is not successful. Fortunately, this is possible for all known attacks [8]. Then one finds the same scaling laws again, which parameter H one can reach a higher level of security for the neural key-exchange protocol without increasing the average synchronization time.

V. Neural Key Exchange Algorithm with Query

The only change in basic neural key exchange algorithm is public random input are replaced with query which is based on their weight vector. Algorithm is given below:

1. Each party selects a random initial weight vector w_k^A and w_k^B at time t = 0.
2. At each training step, the output each hidden unit is calculated as

$$\sigma_i^A = sign(w_i^A, x)$$
$$\sigma_i^B = sign(w_i^B, x) \tag{5}$$

The hidden unit output bits combined to an output bit τ for both the networks A & B.

$$\tau = \prod_{i=1}^{K} \sigma_i \tag{6}$$

3. If the output bits are different, $O^A \neq O^B$, nothing is changed.
4. If $\tau^A = \tau^B$ only the hidden units are trained which have an output bit identical to the common output. $\tau^{A/B} = \sigma^{A/B}$
5. The weights are adjusted during training using Hebbian rule as given below

$$w_i^A(t+1) = w_i^A(t) + x_i(t) \tag{7}$$

If any component wk moves out of the interval [-L, L] it is replaced by sign (w_k)L. Using this algorithm the two neural networks synchronize to a common time dependent secret key $w^A(t) = w^B(t)$. Surprisingly synchronization is fast.

VI. Keeping Inputs Private

In the original key exchange protocol, the structure of the network, output of the TPM (6), the adaptation-rule (7) and the common inputs xk,j are public. The only secrets involved are the different initial weights wk,j of the two parties. If they were not secret, the resulting keys could simply be calculated, because all further computations are completely deterministic.

With use of query we proposed following solution to the basic algorithm. It is divided into two rounds.

Round I

- In first neural key exchange algorithm with query is applied till two networks are fully synchronized.
- The inputs (query) are visible to an attacker, but he cannot predict what will be TPM query (input) generated by either party as it is based on weights which are never exposed.

Round II

- After synchronization, identical weight vectors are used as seed for pseudo random number generator.
- Queries are not exchanged; rather inputs to neural network (TPM) are obtained from pseudo random generator.
- TPM can now be used to generate secret key which is used for encryption/decryption of plain text using Advanced Encryption Standard (AES).

The seed of pseudo random generator are not exchanged over network, attacker cannot anticipate random generators output though its algorithm is public. Hence one more parameter is now unknown (so private) to attacker.

VII. Attack on Cryptosystem

In every attack it is considered, that the attacker E can eavesdrop messages between the parties A and B, but does not have an opportunity to change them.

a) Brute force

To provide a brute force attack, an attacker has to test all possible keys (all possible values of weights w_{kj}). By K hidden neurons, K*N input neurons and boundary of weights L, this gives $(2L+1)^{KN}$ possibilities. For example, the configuration K = 3, L = 3 and N = 100 gives us $3*10^{253}$ key possibilities, making the attack impossible with today's computer power.

b) Learning with own tree parity machine

One of the basic attacks can be provided by an attacker, who owns the same tree parity machine as the parties A and B. He wants to synchronize his tree parity machine with these two parties. In each step there are three situations possible:

- Output (A) ≠ Output (B): None of the parties updates its weights.
- Output (A) = Output (B) = Output (E): All the three parties update weights in their tree parity machines.

Output (A) = Output (B) ≠ Output (E): Parties A and B update their tree parity machines, but the attacker cannot do that. Because of this situation his learning is slower than the synchronization of parties A and B.

It has been proven, that the synchronization of two parties is faster than learning of an attacker. It can be improved by increasing of the synaptic depth L of the neural network. That gives this protocol enough security and an attacker can find out the key only with small probability. Changing this parameter increases the cost of a successful attack exponentially, while the effort for the users grows polynomially. Therefore, breaking the security of neural key exchange belongs to the complexity class NP.

There are other more sophisticated attacks against this protocol (e.g. geometric, majority, genetic attack). The most successful is majority attack. So far, none of the known attacks could break security of the neural key exchange protocol with queries.

VIII. Results

The neural key exchange algorithm with query is implemented in software on Fedora 9, using gcc. The well known Advanced Encryption Standard algorithm is used for encryption/decryption [14]. The round I & II given section VI are implemented, only partial output is given below. The entire implementation takes about 100KB and execution time is in nanoseconds. The actual output and result as are as below:

```
===================================
TREE PARITY MACHINE B
===================================
```

Note: Parity with TPM B encrypts cipher with Advanced Encryption Standard

```
/*                Round I                    */
INITIAL WEIGHT VECTORS
Wb[ ][ ]=
2 -1 3 0 4 -4 -3 -2 2 0
-3 -3 1 1 4 -1 -3 -4 1 -2
-4 3 -1 -2 -3 -4 -2 1 -1 1
```

TPM A & TPM B are exchanging outputs for mutual learning

```
Iteration: 0
----------------
Query received
1 -1 1 -1 1 1 1 -1 -1 -1
1 -1 -1 1 1 1 -1 1 1 1
1 -1 -1 1 1 1 -1 -1 1 1
Iteration: 1
----------------
Query send
1 -1 1 -1 1 -1 -1 -1 1 1
-1 1 1 1 -1 -1 1 -1 1 1
1 -1 -1 1 1 -1 1 1 1 -1
```

/* The query exchange continues till synchronization (output truncated here)*/

```
/*                Round II                   */
WEIGHT VECTORS AFTER SYNCHRONIZATION
Wb[ ][ ]=
4 3 2 1 2 -3 -3 -4 -2 3
3 4 -1 3 1 -4 0 3 -1 2
1 0 1 4 3 0 -2 -2 0 3
/*Wb[][] is used as seed for random number generater*/
No. of iterations TPMs has taken for synchronization: 439
Plaintext:
0 11 22 33 44 55 66 77 88 99 aa bb cc dd ee ff
Ciphertext:
c4 fd 5c fc cd 5e a8 1c 5a b0 28 8e e0 17 19 4
```

Key:
e6 79 d1 84 8a e9 96 f 7e 85 e8 ad f1 82 91 e5
Hash: 11E87945

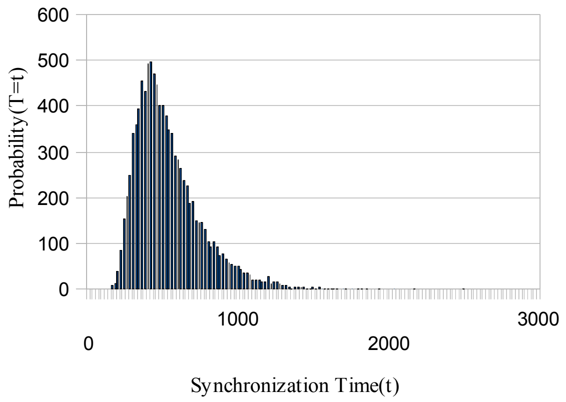

Figure 4. Distribution of Synchronization Time t_{sync} for N=100 for two Tree Parity Machines with K = 3, L = 4. The histogram shows the relative frequency of occurrence observed in 10000 runs.

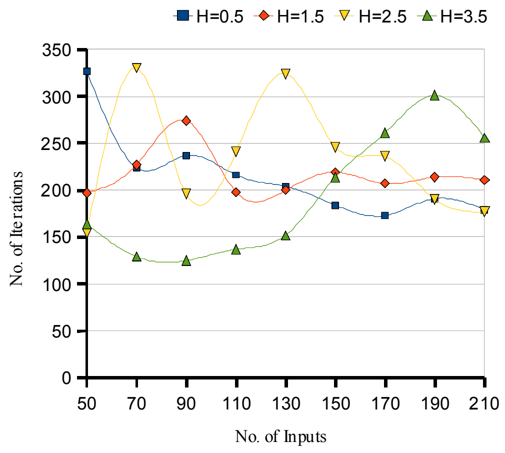

Figure 5. Difference in behavior of Tree Parity Machine with query for local field H=0.5 to H=3.5.

The average synchronization time tav does not increase with increasing size N of the system. In the case of the TPM, the complexity of the encryption/decryption processes scales linearly with the size of the transmitted message, whereas the complexity of the synchronization process does not scale with the size of the network. Hence our construction is a linear cryptosystem.

From fig 4. it seems to converge to tav 510 for infinitely large networks. Surprisingly, in the limit of large N one needs to exchange only about 500 bits to obtain agreement between 3N components. However, one should keep in mind that the two partners do not learn the initial weights of each other, they just are attracted to a dynamical state with opposite weight vectors.

The network that operates with query based on local field of random weight vectors synchronize faster, but execution time is roughly tree times more compared to network without query. Additionally performance of network for different values local field has been studied. It is found that performance of network good for H=1.5 various values of input N=10 to 100, as shown fig.5.

IX. Conclusion

The TPM that operates with query synchronize faster, but as amount of information exchanged is more that increases execution time of algorithm, but queries are only exchanged till synchronization. After synchronization of TPM inputs to TPM are taken from random number.

The seed of pseudo random generator are not exchanged over network, attacker cannot anticipate random generators output though its algorithm is public. Hence not only weight vector but also input vector is unknown (so private) to attacker thereby making his task difficult. The queries along with private inputs restore the security of neural cryptography against attackers.

X. Future Scope

The TPM that operates with query synchronize faster, but as amount of information exchanged is more that increases execution time of algorithm, but queries are only exchanged till synchronization. After synchronization of TPM inputs to TPM are taken from random number.

The seed of pseudo random generator are not exchanged over network, attacker cannot anticipate random generators output though its algorithm is public. Hence not only weight vector but also input vector is unknown (so private) to attacker thereby making his task difficult. The queries along with private inputs restore the security of neural cryptography against attackers.

References

[1] P. S. Revankar, W. Z. Gandhare, D. T. Rathod, "Neural synchronization with queries," ICSAP 10, in press.

[2] Volkmer M., Wallner S.: "Tree parity machine rekeying architectures". IEEE Transactions on Computers 54, 2005, pp. 421-427.

[3] T. Godhavari, N. R. Alainelu and R. Soundararajan, "Cryptography using neural network", IEEE Indicon 2005 Conference, Chennai, India, 11-13 Dec. 2005, pp.258-261.

[4] A. Klimov, A. Mityaguine, and A. Shamir. "Analysis of neural cryptography", In Y. Zheng, editor, Advances in Cryptology—ASIACRYPT 2002, Springer, Heidelberg, pp. 288, 2003.

[5] U. Maurer, "Secret key agreement by public discussion", IEEE Trans. Information Theory, vol. 39, pp. 733-742, 1993.

[6] N. Prabhakaran, P. Saravanam, P. Vivekanandan, "Neural cryptography with multiple transfers functions and multiple learning rule", International Journal of Soft Computing 3, pp. 177-181, 2008.

[7] Tieming Chen, and Samuel H.Huang, "Tree parity machine-based one-time password authentication", Schemes International Joint Conference on Neural Networks (IJCNN 2008), 2008, pp.257-261.

[8] Andreas Ruttor, Wolfgang Kinzel and Ido Kanter, "Neural cryptography with queries", Journal of Statistical Mechanics: Theory and Experiment, doi:1088/1742-5468/2005/01/P01009, Jan. 2005.

[9] Wolfgang Kinzel and ldo Kanter, "Interacting neural networks and cryptography", Advances in Solid State Physics, Ed. by B. Kramer (Springer, Berlin. 2002), Vol. 42, pp.383.

[10] M. Rosen-Zvi, E. Klein, 1. Kanter and W. Kinzel, "Mutual learning in a tree parity machine and its application to cryptography", Phys. Rev. E, 2002.

[11] Ido Kanter, Wolfgang Kinzel, and Eran Kanter, "Secure exchange of information by synchronization", Euriphysics Letters 57, 2002, pp 141-147 .

[12] Maurer U., "Protocols for secret key agreement by public dicussion based on common information", Advances in Cryptology- CRYPTO' 92. Vol. 740 of LNCS, Springer Verlog, 1993, pp. 461-470.

[13] Stajano F., "Security in pervasive computing", Proceeding of the 1st International Conference on Security in Pervasive Computing, Vol. 2802 of LNCS, 2003.

[14] "Advanced Encryption Standard", Federal Information Processing Standards Publication 197,November 26, NIST, Computer Security Division ,2001, pp. 5-30.