Int. J. of Advanced Networking and Applications
Volume: 02, Issue: 01, Pages: 437-444 (2010)

A New Security on Neural Cryptography with Queries

Dr. N. Prabakaran
Department of Computer Applications, Rajalakshmi Engineering College, Chennai- 602 105, India
E-mail: prabakaran_om@yahoo.com

Dr. P. Vivekanandan
Professor & Director, Knowledge Data Centre, Anna University, Chennai-600 025, India
E-mail: vivek@annauniv.edu

ABSTRACT

We can generate a secret key using neural cryptography, which is based on synchronization of Tree Parity Machines (TPMs) by mutual learning. In the proposed TPMs random inputs are replaced with queries which are considered. The queries depend on the current state of A and B TPMs. Then, TPMs hidden layer of each output vectors are compared. That is, the output vectors of hidden unit using Hebbian learning rule, left-dynamic hidden unit using Random walk learning rule and right-dynamic hidden unit using Anti-Hebbian learning rule are compared. Among the compared values, one of the best values is received by the output layer. The queries fix the security against majority flipping and geometric attacks are shown in this paper. The new parameter H can accomplish a higher level of security for the neural key-exchange protocol without altering the average synchronization time.

Keywords: Majority attacks, Neural Synchronization, Queries, Tree Parity Machines.

Date of Submission: October 12, 2009 Date of Acceptance: May 07, 2010

1. Introduction

Neural cryptography is a method for generating secret information over a public channel. Before two partners A and B can exchange a secret message over a public communication channel, they have to agree on a secret encryption key. Using number theoretic methods, such keys can be constructed over public channels without previous secret agreements of the two partners. The algorithm as well as the complete information passed between the partners is known to a possible attacker E; nevertheless the final key is secret, it is known to the two partners A and B only and an attacker E with limited computer power cannot calculate the key. The method is based on the computational difficulty of factorizing large numbers or calculating the discrete logarithm of large numbers [8].

Two identical dynamical systems, starting from different initial conditions can be synchronized by a common input values which are coupled to the two systems. Two networks which are trained on their mutual output can synchronize to a time-dependent state of identical synaptic weights. The networks receive a common input vector after calculating their outputs and update their weight vectors according to the match between their mutual outputs in every time step. The input or output relations are exchanged through a public channel until their weight vectors are identical and can be used as a secret key for encryption and decryption of secret messages. The random inputs are replaced by queries in this network. It is based on exchanging inputs between A and B which is correlated to weight vectors of the two networks [11].

The properties of the synchronization process depend on the synaptic depth L and queries. Hence there is an additional parameter H, which fixes the absolute value $|h_i|$ of the local fields in the TPMs generating the current query. As the prediction error of a hidden unit, left-dynamic hidden unit and right-dynamic hidden unit is a function of both the overlap ρ_i and the local field h_i, the partners modify the probability of repulsive steps $P_r(\rho)$ if they change H. The partners A and B are able to adjust the difficulty of neural synchronization and learning [7].

This paper is organized as follows. In Section 2, the basic algorithm for neural synchronization with queries is given. Also lower layer spy unit, upper layer spy unit, definition of the order parameters and transition probabilities of proposed TPMs are explained. In Section 3, the generation of queries is described. Synchronization time of two TPMs is briefly explained in Section 4. The security against known attacks with success probability of an attacker is presented in Sec. 5. In Section 6, the advanced attacks are discussed. Finally, conclusion is shown in Section 7.

2. Neural Synchronization

The weight vectors of the two neural networks begin with random numbers, which are generated by Pseudo-Random Number Generators (PRNGs). In these networks, random inputs are replaced by queries. That is A and B choose alternatively according to their own weight vectors. The partners A and B receive a common input vector at each time; their outputs are computed and then communicated over public channel [8]. If they agree on the mapping between the current input and the output, their weights are updated according to the learning rule.

2.1 A Structure of Tree Parity Machine

The TPMs consist of K-hidden units, Y-left dynamic hidden units [3] and Z-right dynamic hidden units [4], each of them being a perceptron with an N-dimensional weight vector **w**. The lower layer spy unit (ϑ) is associated with the N-input units **x**. The upper layer spy unit (ξ) is associated with the hidden units (σ) [5], left-dynamic hidden units (δ), right-dynamic hidden units (γ) and output unit (τ).

The lower layer spy unit receives the input from the N-input units. The upper layer spy unit receives the input from the Y-left dynamic hidden units, Z-right dynamic hidden units, K-hidden units and output unit. The network structure of this TPM is shown in fig.1.

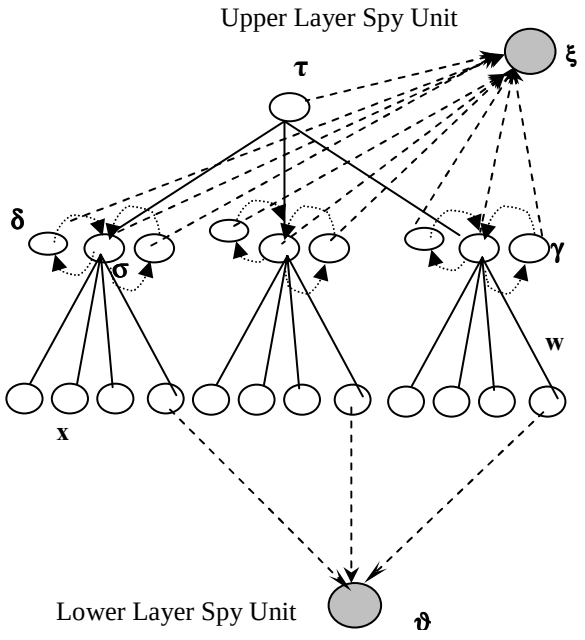

Fig. 1: A structure of Tree Parity Machine with K=3, Y=3, Z=3, ϑ=1, ξ=1 and N=4.

The components of the input vectors **x** are binary: [9]

$$x_{ij} \in \{-1, +1\},\ x_{im} \in \{-1, +1\},\ x_{ik} \in \{-1, +1\} \quad (1)$$

and the weights are discrete numbers between -L and +L :

$$w_{ij} \in \{-L, -L+1, \ldots, 0, \ldots, L-1, L\},$$
$$w_{im} \in \{-L, -L+1, \ldots, 0, \ldots, L-1, L\},$$
$$w_{ik} \in \{-L, -L+1, \ldots, 0, \ldots, L-1, L\}. \quad (2)$$

where L is the depths of the weights of the networks [2].

The TPM receives the input vectors using queries. These input vectors are correlated with the present weight vector $w_k(t)$. At odd time steps, the partner A generates an input vector which has a certain overlap to its weights w_k^A. At even time steps, the partner B generates an input vector which has a certain overlap to its weights w_k^B. It is based on the queries to improve the security of the systems.

The index i = 1,…..,K denotes the i^{th} hidden unit of TPM , m=1,…..,Y denotes the m^{th} left-dynamic hidden unit of the TPM, k=1,…..,Z denotes k^{th} right-dynamic hidden unit of the TPM and j =1,….,N denotes the N input units [6].

The different transfer functions for hidden layer are given below:

$$\sigma_i = \operatorname{sign}\left(\sum_{j=1}^{N} w_{ij} \bullet x_{ij}\right) \quad (3)$$

$$\delta_i = \tanh\left(\sum_{m=1}^{N} w_{im} \bullet x_{im}\right) \quad (4)$$

$$\gamma_i = \arctan\left(\sum_{k=1}^{N} w_{ik} \bullet x_{ik}\right) \quad (5)$$

where equation (3) is the transfer function of the hidden unit, the equation (4) the transfer function of the left-dynamic hidden unit and the equation (5) the transfer function of the right-dynamic hidden unit.

The transfer functions for lower layer spy unit and upper layer spy unit are given below:

$$\vartheta = -\operatorname{sign}\left(\sum_{i=1}^{N}\left[\sum_{j=1}^{N} x_{ij} \bullet w_{ij}\right]\right) \quad (6)$$

$$\xi = -\operatorname{sign}\left(\sum_{i=1}^{K} \delta_i \bullet \sigma \bullet_i \gamma_i \bullet \tau\right) \quad (7)$$

where the equation (6) is the transfer function of the lower layer spy unit and equation (7) the transfer function of the upper layer spy unit.

The K-hidden units of σ_i, Y-left dynamic hidden units [3] of δ_i and Z-right dynamic hidden units [4] of γ_i define common output bit of hidden layer of the network and are given by:

$$\beta_a = \prod_{i=1}^{K} \sigma_i \quad (8)$$

$$\beta_b = \prod_{i=1}^{Y} \delta_i \quad (9)$$

$$\beta_c = \prod_{i=1}^{Z} \gamma_i \quad (10)$$

where equation (8) is the output for the hidden units, equation (9) the output for the left-dynamic hidden units and equation (10) the output for the right-dynamic hidden units.

The two TPMs compare the hidden layer's output bits (hidden, left-dynamic and right-dynamic hidden units) and then update the weight vector to the output unit as well as partners A and B that are trying to synchronize their weight vectors:

$$\psi_i^{A,B} = comp(\beta_a, \beta_b, \beta_c) \quad (11)$$

$$\phi_i^A = w_{ij}^A \, x_{ij}^A \, \tau^B \psi_i^A \quad (12)$$

$$\phi_i^B = w_{ij}^B \, x_{ij}^B \, \tau^A \psi_i^B \quad (13)$$

where equation (11) represents comparison of the output of hidden, left-dynamic and right-dynamic hidden units of A and B. The equation (12) and (13) represent output of hidden, left and right-dynamic hidden units of A and B respectively.

$$\tau = \prod_{i=1}^{K} \phi_i \quad (14)$$

The equation (14) represents the output vector of the output unit of the TPM.

2.2 Learning Rules

The partners A and B initialize their random number of weight vectors before the start of the training period. At each time step t, a public input vector is generated and the bits τ^A and τ^B are switched over the public channel. In the case of indistinguishable output bits $\tau^A = \tau^B$, each TPM adjust those of its weight vectors for which the hidden unit, left-dynamic hidden unit and right-dynamic hidden unit is identical to the output $\phi^{A/B} = \tau^{A/B}$. These weights are adjusted according to a given learning rules. They are

(a) Hebbian Learning rule for hidden units:

$$w_i^A(t+1) = w_i^A(t) + x_i \tau^A \Theta(\tau^A \phi_i^A)\Theta(\tau^A \tau^B)$$
$$w_i^B(t+1) = w_i^B(t) + x_i \tau^B \Theta(\tau^B \phi_i^B)\Theta(\tau^A \tau^B) \quad (15)$$

where Θ is the Heaviside step function, if the input is positive then the output is 1 and if input is negative then the function evaluates to 0.

(b) Random walk learning for left-dynamic hidden units:

$$w_i^A(t+1) = w_i^A(t) + x_i \Theta(\tau^A \phi_i^A)\Theta(\tau^A \tau^B)$$
$$w_i^B(t+1) = w_i^B(t) + x_i \Theta(\tau^B \phi_i^B)\Theta(\tau^A \tau^B) \quad (16)$$

(c) Anti-Hebbian learning for right-dynamic hidden units:

$$w_i^A(t+1) = w_i^A(t) - \phi_i x_i \Theta(\tau^A \phi_i^A)\Theta(\tau^A \tau^B)$$
$$w_i^B(t+1) = w_i^B(t) - \phi_i x_i \Theta(\tau^B \phi_i^B)\Theta(\tau^A \tau^B) \quad (17)$$

2.3 Order Parameters

The process of synchronization itself can be described by standard order parameters. These order parameters are:

$$Q_i = \frac{W_i^A \cdot W_i^A}{N}, Q_j = \frac{W_j^A \cdot W_j^A}{N}, Q_k = \frac{W_k^A \cdot W_k^A}{N} \quad (18)$$

The equation (18) represents weight distribution of hidden units, left-dynamic hidden units and right-dynamic hidden units of A's TPM.

$$R_i^{A,B} = \frac{W_i^A \cdot W_i^B}{N}, \; R_j^{A,B} = \frac{W_j^A \cdot W_j^B}{N}, \; R_k^{A,B} = \frac{W_k^A \cdot W_k^B}{N} \quad (19)$$

The equation (19) represents overlap between two hidden units, two left-dynamic hidden units and two right-dynamic hidden units of A and B respectively.

The distance between two corresponding hidden unit, left-dynamic hidden units and right-dynamic hidden units are defined by the overlap is given below:

$$\rho_{ijk}^{A,B} = \frac{R_i^{A,B}}{\sqrt{Q_i^A Q_i^B}} + \frac{R_j^{A,B}}{\sqrt{Q_j^A Q_j^B}} + \frac{R_k^{A,B}}{\sqrt{Q_k^A Q_k^B}} \quad (20)$$

2.4 Transition Probabilities

A repulsive step can only occur in the i[th] hidden unit, j[th] left-dynamic hidden unit and k[th] right-dynamic hidden unit, if the two corresponding outputs ϕ_i are different. The probability for this event is given by the well-known generalization error for the perceptron: [1]

$$\varepsilon_p^j = \frac{1}{\pi} \arccos(\rho_{ijk}) \qquad (21)$$

where equation (21) represents the generalization error for hidden, left-dynamic and right-dynamic unit of two TPMs.

The quantity ε_ρ^i is a measure of the distance between the weight vectors of the corresponding hidden units, left-dynamic hidden units and right-dynamic hidden units and these values are independent. The values ε_ρ^i determine the conditional probability P_r for a repulsive step and P_a for a attractive step between two hidden units, left-dynamic hidden units and right-dynamic hidden units given identical output bits of the two TPMs. In the case of identical distances $\varepsilon_\rho^i = \varepsilon$, the values of K, Y and Z are found as K=3, Y=3 and Z=3.

$$P_a = \frac{1}{2}\frac{(1-\varepsilon)^9 + 3(1-\varepsilon)^3\varepsilon^2}{(1-\varepsilon)^9 + 9(1-\varepsilon)^3\varepsilon^2} \qquad (22)$$

$$P_r = \frac{6(1-\varepsilon)^3 \ \varepsilon^2}{3(1-\varepsilon)^9 + 9(1-\varepsilon)^3\varepsilon^2} \qquad (23)$$

The equation (22) and (23) represent probability of attractive and repulsive steps between two hidden units, two left-dynamic hidden units and two right-dynamic hidden units of A and B respectively.

The attacker E can assign a confidence level to each output σ_i^E, δ_i^E *and* Υ_i^E of its hidden units, left-dynamic hidden units and right-dynamic hidden units. For this task the local field is given by:

$$h_{ijk} = \frac{w_i \cdot x_i}{\sqrt{N}} + \frac{w_j \cdot x_j}{\sqrt{N}} + \frac{w_k \cdot x_k}{\sqrt{N}} \qquad (24)$$

where equation (24) represents the local field of hidden unit, left-dynamic and right-dynamic hidden units of an attacker‘s TPM.

From the below Fig. 2, we are able to predict the probability of repulsive steps occur more frequently in E’s TPM than in A’s and B’s for equal overlap $0 < \rho < 1$. So, the partners A and B have a clear advantage over a simple attack in neural cryptography.

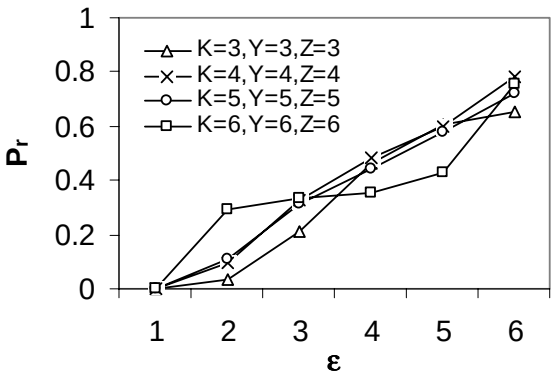

Fig. 2: Probability $P_r^B(\rho)$ of repulsive steps for synchronization with mutual interaction under the condition $\tau^A = \tau^B$.

Then the prediction error of the probability of different output bits for an input vectors ‘x’ inducing a local field h_{ijk} is given below:

$$\varepsilon(\rho_{ijk}, h_{ijk}) = \frac{1}{2}\left[1 - erf\left(\frac{\rho_{ijk}}{\sqrt{2(1-\rho_{ijk}^2)}}\frac{|h_{ijk}|}{\sqrt{Q_i}}\right)\right] \qquad (25)$$

where equation (25) represents prediction error of the local field of hidden units, left-dynamic and right-dynamic hidden units of an attacker’s TPM.

3. Generation of Queries

As both inputs $x_{i,m}$ and weights $w_{i,m}$ are discrete, there are only (2L+1) possible solutions for the product $x_{i,m} \cdot w_{i,m}$ [7]. Therefore, a set of input vectors consisting of all permutation, which do not exchange h_i, can be depicted by counting the number $c_{i,l}$ of products with $x_{i,m} \cdot w_{i,m} = l$. Then the local field is given by:

$$h_{ijk} = \frac{1}{\sqrt{N}}\left[\sum_{l=1}^{L}\left(l(c_{i,l} - c_{i,-l}) + l(c_{j,l} - c_{j,-l}) + l(c_{k,l} - c_{k,-l})\right)\right] \qquad (26)$$

where equation (26) is the number of inputs and weights in the local field of TPM.

The sum $n_{ijk,l} = c_{i,l} + c_{i,-l} + c_{j,l} + c_{j,-l} + c_{k,l} + c_{k,-l}$ is equal to the number of weights with $|w_{i,j}| = |l|$ and thus independent of ‘x’. Accordingly, one can write h_{ijk} as a function of only L variables, because the generation of queries cannot change ‘w’:

Volume: 02, Issue: 01, Pages: 437-444 (2010)

$$h_{ijk} = \frac{1}{\sqrt{N}}\left[\sum_{l=1}^{L}\left(l(2c_{i,l} - n_{i,l}) + l(2c_{j,l} - n_{j,l}) + l(2c_{k,l} - n_{k,l})\right)\right] \quad (27)$$

where equation (27) is the inputs and sum of current weights vectors of the local field of two TPMs.

The inputs vectors 'x' is connected with zero weights which are selected by randomly, because they do not determine the local field. The other input bits $x_{i,m}$ are divided into L groups according to the absolute value $l = |w_{i,m}|$ of their corresponding weight. In each group, $c_{i,l}$ inputs are selected randomly and set to $x_{i,m}$=sign($w_{i,m}$). The remaining $n_{i,l} - c_{i,l}$ input bits are set to $x_{i,m}$ = -sign ($w_{i,m}$).

The maximum possibilities of the weight vectors of an attacker's TPM is given by:

$$l_{\max} = (4L + 2)^{(K+Y+Z)\bullet N} \quad (28)$$

Then

$$\ln\ (l_{\max}) = (K+Y+Z)\cdot N\ \ln(4L+2) \quad (29)$$

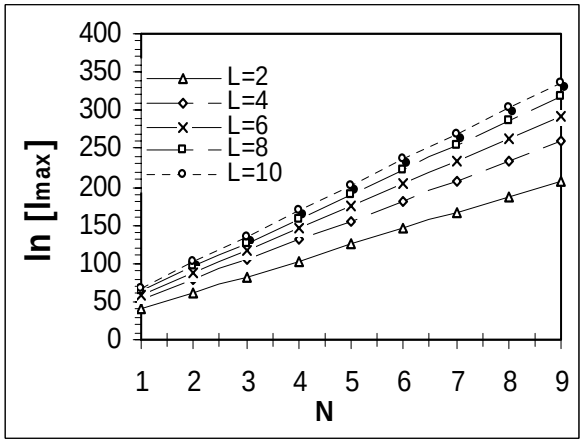

Fig. 3: The possible values of l_{max} of the weight vectors of an attack's TPM.

From the above fig.3, we are able to determine a large number of possibilities of weight vectors against an attacker, in which weights are selected by queries. In each time step, either A or B generates the input vectors. The attacker E cannot easily gain the weight vector and useful information from analyzing queries**.**

4. Synchronization Time

The generation of the inputs vectors x_{ij} has to be changed during queries synchronization. The partner A or B use the generation of queries (shown in Section 3) at each time step to generate K input vectors x_k, which result in $h_{ijk}^{A/B} \approx \pm H$. Both partners calculate the output of their TPMs with queries. Then the exchange of τ^A and τ^B the Hebbian learning rule is used to update the weights. This leads to synchronization after t_{sync} steps.

The dependence on the synaptic depth L of t_{sync}, which is induced by two effects

1. The control signals σ_i, δ_i, γ_i and τ are neglected, if weight vectors can be described as random walk with reflection boundaries: [8].
$$t_{sync} \propto L^2 \quad (30)$$
2. The probability of repulsive steps P_r depends on the overlap ρ_{ijk}, but also on the quantity $|h_{ijk}|/\sqrt{Q_i}$ using queries. Assuming uniformly distributed weights as given below:
$$Q_{ijk} = \frac{1}{N}\ w_{ijk} \bullet w_{ijk} = \frac{1}{3}L(L+1) \approx \frac{1}{3}L^2 \quad (31)$$
Therefore the length of the weight vectors improves proportional to L.

$$H = \alpha L + (\vartheta + \xi) \quad (32)$$

where α is the rescaled field of H/L. L is the synaptic depth of TPM, ϑ is the lower layer spy unit vector and ξ is the upper layer spy unit vector.

In eqn. (30) and (31), we can rescale t_{sync} in order to obtain functions $f_L(\alpha)$, which are nearly independent of the synaptic depth in the case L >>1:

$$t_{sync} = L^2 f_L\left(\frac{H}{L}\right) \quad (33)$$

The function of $f_L(\alpha)$ converges to a universal scaling function $f(\alpha)$ in the limit $L\to\infty$.

$$f(\alpha) = \lim_{L\to\infty} f_L(\alpha) \quad (34)$$

The distance $|f_L(\alpha) - f(\alpha)|$ shrinks proportion to (L – 1). Therefore the universal function $f_L(\alpha)$ can be found out by finite-size scaling.

Volume: 02, Issue: 01, Pages: 437-444 (2010)

5. Security against Known Attacks

The neural key exchange protocol using queries is efficient, secure against the attacks. The different values of the local field influence the security of the system. The results impose further restrictions upon the usable range of the parameter H [7, 8].

5.1 Success Probability

The two partners A and B use queries for the neural key-exchange. The success probability strongly depends on the parameter H [10]. This model is suitable for both the majority and the geometric attack.

$$P_E = \frac{1}{1+\exp(-\beta(H-\mu))} \qquad (35)$$

two parameter β and μ is a suitable fitting functions for describing P_E as a function of H.

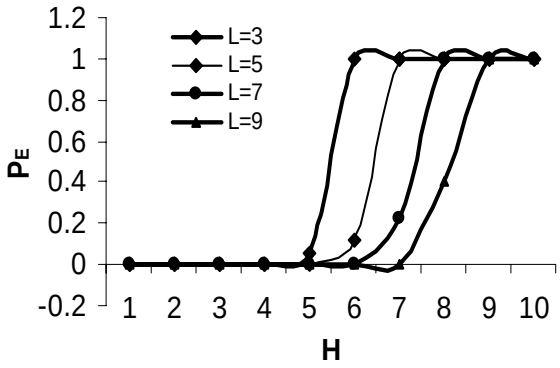

Fig. 4: Success probability of the Majority Flipping Attack as a function of H. Symbols denote the simulations results for K = 3, Y=3, Z=3, η=3, M = 100 and N = 1000 (H= η+ 0.45 * L).

The position μ of the smooth step increases linearly with the synaptic depth L as given below:

$$\mu = \alpha_s L + \eta\partial \qquad (36)$$

The parameter α_s is the maximum α for security and η is the number of hidden layer (hidden unit, left-dynamic and right-dynamic hidden units) and ∂ depend on the learning rule and the attack.

Combined eqn. (35) and (36) yields:

$$P_E = \frac{1}{1+\exp(-\beta\eta\partial)\ \exp(-\beta(\alpha_s-\alpha))L} \qquad (37)$$

for the success probability of any known attack.

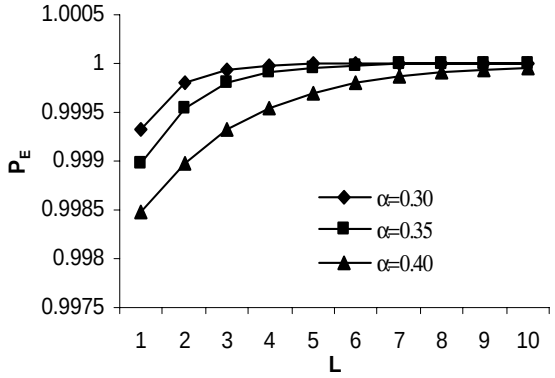

Fig. 5: Success probability of the Majority Flipping Attacks depends on the synaptic depth of L. Symbols denote the simulations results for K = 3, Y=3, Z=3, η=3, M = 100 and N = 1000 (α_s=0.45, ∂=0.40, β=8.1).

The partner A and B choose $\alpha = H/L$ according to the condition $\alpha < \alpha_s$, P_E vanishes for L→∞, its asymptotic behavior is given by:

$$P_E \sim e^{-\beta\eta\partial} e^{-\beta(\alpha_s-\alpha)L} \qquad (38)$$

Hence the success probability P_E decreases exponentially with increasing synaptic depth of L

$$P_E \sim e^{-y(L-L_0)} \qquad (39)$$

6. Advanced Attacks

The attacker may improve the success probability P_E by using additional information exposed through the algorithm generating the input vectors. The two approaches are

1. The attacker E could use to improve the internal representation of the absolute local field $(\sigma_1^E, \sigma_2^E, ..., \sigma_K^E)$, $(\delta_1^E, \delta_2^E, ..., \delta_K^E)$ and $(\gamma_1^E, \gamma_2^E, ..., \gamma_K^E)$ in the geometric correction.

2. Each weight vector **w** is correlated to the corresponding input vector **x** of the generating network.

Volume: 02, Issue: 01, Pages: 437-444 (2010)

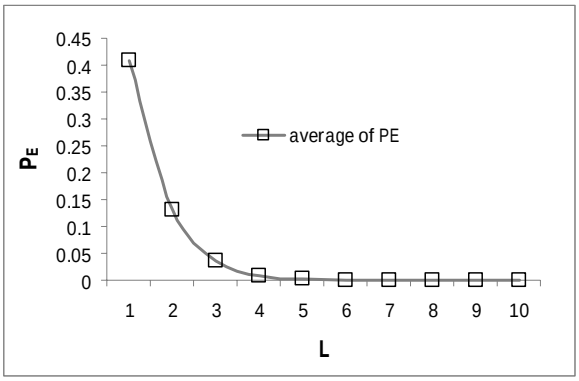

Fig. 6: the Average success probability of Majority Flipping Attack as a function of H. Symbols denote the simulations results for K = 3, Y=3, Z=3, η=3, M = 100 and N = 1000.

6.1 Known Local Field

The absolute local field value in either A's or B's hidden units, left-dynamic units and right-dynamic units is given by H using queries and E knows the local fields h_i^E in attacker's TPM.

$$P(\phi_i^E \neq \phi_i^A) = \left[1+\exp\left(\eta\left(\frac{6\rho_i^{AE}}{1-(\rho_i^{AE})^2}\ \frac{H}{\sqrt{Q_i^A}}\frac{|h_i^E|}{\sqrt{Q_i^E}}\right)\right)\right]^{-1} \quad (40)$$

From the Fig.6, we are able predict the prediction error of the local field h_i^E is increased against the geometric attack.

As there is no qualitative difference compared to synchronization with random inputs, it is not possible to improve the geometric attack by using H as additional information.

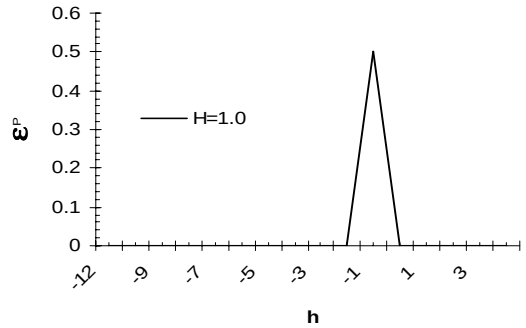

Fig. 6: Prediction error ε_i^P as a function of the local field h_i^E for Q_i^A=1, Q_i^E=1 and ρ=0.5.

6.2 Information about weight vectors

The queries give E as additional information about the weight vectors in A's and B's TPMs while H cannot be used directly in the geometric attack [7]. The absolute local field $|h_{ijk}|$ for synchronization with queries is lower that the average value.

$$|h_{ijk}| = \sqrt{\frac{2\eta Q_{ijk}}{\pi}} \approx 0.78\sqrt{Q_{ijk}} \quad (41)$$

observed for random inputs. Therefore the overlap:

$$\rho_{ijk,in} = \frac{w_{ijk} \bullet x_{ijk}}{\sqrt{w_{ijk} \bullet w_{ijk}}\sqrt{x_{ijk} \bullet x_{jki}}} = \frac{1}{3\sqrt{N}}\frac{h_{ijk}}{\sqrt{Q_{ijk}}} \quad (42)$$

between input vector and weight vector converges to zero in the limit N→∞, even if queries with $0<|h_{ijk}|<\infty$ are used. Consequently, x_{ijk} and w_{ijk} are nearly perpendicular to each other, so that the information exposed by queries is minimized.

A given value of H the number of weight vectors, which are consistent with a given query is still exponentially large.

7. Conclusion

In the proposed TPMs, we trained the three transfer functions in the hidden layer. That is, hidden unit using Hebbian learning rule, left-dynamic hidden unit using Random walk rule and right-dynamic hidden unit using Anti-Hebbian learning rule included with queries. Also, the queries increase the probability of repulsive steps for an attacker during the synchronization. In addition, the method receives a new parameter, which can be adapted to give optimal security. The attacker E is at a disadvantage compared to A and B because E needs a higher absolute value of the local field than the partners in order to synchronize on average. Hence, it is possible to adjust the new parameter H combined with lower layer spy unit vector and upper layer spy unit vector. The partners A and B synchronize fast, but E is not successful regardless of the attack method.

References

[1] A. Engel and C.Van Den Broeck, Statistical Mechanics of Learning, (Cambridge University Press, Cambridge 2001).

[2] W. Kinzeland I. Kanter, Interacting neural networks and cryptography, *Advances in Solid State Physics,* by B. Kramer (Springer, Berlin), 42, 2002, 383-391 [cond-mat/0203011].

[3] N. Prabakaran, P.Loganathan and P. Vivekanandan, Neural Cryptography with Multiple Transfer function and Multiple Learning Rule, *International Journal of Soft Computing, 3(3)*, 2008, 177-181.

[4] N. Prabakaran, P. Karuppuchamy and P. Vivekanandan, A New approach on Neural Cryptography with Dynamic and Spy units using multiple transfer functions and learning rules. *Asian Journal of Information Technology, 7(7)*, 2008, 300-306.

[5] N. Prabakaran, P. Saravanan and P. Vivekanandan P (2008), A New technique on Neural Cryptography with Securing of Electronic Medical Records in Telemedicine System, *International Journal of Soft Computing, 3*(5), 2008, 390-396.

[6] A. Ruttor, W. Kinzel, L. Shacham and I. Kanter, Neural cryptography with feedback, *Physical Review Edition, 69*, 2004,1- 8.

[7] A. Ruttor, *Neural Synchronization and Cryptography*, Ph. D. Thesis,2006.

[8] A. Ruttor, W. Kinzel and I. Kanter, Neural cryptography with queries, *Physics Rev. E. 25(01)*, 2005, 01-12.

[9] A. Ruttor, I. Kanter and W. Kinzel, Dynamics of neural cryptography, *Phy. Rev.E., 75(5)*, 2006, 056104-056113.

[10] A.Ruttor, I. Kanter, R. Naeh and W. Kinzel, Genetic attack on neural cryptography, *Phy. Rev.E., 73(3)*, 2006, 036121-036127.

[11] R. Mislovaty, E. Klein, I. Kanter and W. Kinzel, Public channel cryptography by synchronization of neural networks and chaotic maps, *Phy. Review Letter 91(11)*, 2003, 118701.

Authors Biography

PRABAKARAN N , he has received M.Sc in Computer Science from Anna University. He had done Ph.D from Anna University. His areas of interest are network security, visual programming and wireless technology. He had published Six International Journals and two papers in National Journals and he has presented five papers in international conferences.

VIVEKANANDAN PERIYASAMY received his Master of Science in Applied Mathematics from Madras University in 1978 and Doctor of Philosophy from Anna University in 1987. Also, he obtained his postgraduate degree in Master of Engineering in Computer Science and Engineering from Anna University in 1995. He is working as Professor of Mathematics, Department of Mathematics in Anna University from 1978. He visited Singapore, Malaysia, Japan, Bangladesh, Sultanate of Oman and USA for presenting research papers and Chairing Sessions. He has published more than 80 research papers in national and international journals. His areas of research are Neural Networks, Internet Security and Software Reliability.