A New Method an Enhancement on Neural Cryptography with Multiple Transfers Functions and Learning Rules

N. Prabakaran[1] *P. Loganathan* [2] *P. Vivekanandan*[3]

ABSTRACT

The goal of any cryptographic system is the exchange of information among the intended users. We can generate a common secret key using neural networks and cryptography. Neural cryptography is based on a competition between attractive and repulsive stochastic forces. A feedback mechanism is added to neural cryptography which increases the repulsive forces. The partners A and B have to use a cryptographic key exchange protocol in order to generate a common secret key over the public channel. This can be achieved by two Tree Parity Machines (TPMs). In the proposed TPMs, each output vectors are compared, then updates from hidden unit using Hebbian Learning Rule, left-dynamic hidden unit using Random Walk Rule and right-dynamic hidden unit using Anti-Hebbian Rule with feedback mechanism. We can enhance the security of the system using different learning rules with different units. A network with feedback generates a pseudorandom bit sequence which can be used to encrypt and decrypt a secret message. In this paper, the most successful attack on neural cryptography is the majority flipping attack, which is presented here. The probability of a successful attack is calculated for different model parameters using numerical simulations.

Keywords : Neural Cryptography, Tree Parity Machine, Neural Synchronization, Feedback Synchronization, Majority Flipping Attacks.

1. INTRODUCTION

Neural cryptography is based on the effect that two neural networks are able to synchronize by mutual learning (Ruttor. A et al., 2006). In each step of this online learning procedure, they receive a common input pattern and calculate their output. Then, both neural networks use those outputs present by their partner to adjust their own weights. This process leads to fully synchronized weight vectors.

Synchronization of neural networks is, in fact, a complex dynamical process. The weights of the networks perform random walks, which are driven by a competition of attractive and repulsive stochastic forces. Two neural networks can increase the attractive effect of their moves by cooperating with each other. But, a third network is only listening to the communication. Therefore, bidirectional synchronization is much faster than unidirectional learning.

Two partners A and B want to exchange a secret message over a public channel. In order to protect the content against an attacker E, who is listening to the communication, A encrypts the message, but B needs A's secret key for decryption. Without an additional private channel, A and B have to use a cryptographic

[1]Dept. of Master of Computer Application , Sri Muthukumaran Institute of Technology , Mangadu , Chennai-69, India.

[2&3]Department of Mathematics, Anna University, Chennai, India. Email : prabakaran_om@yahoo.com , vivek@annauniv.edu

key exchange protocol in order to generate a common secret key over the public channel (Ruttor A et al., 2006). This can be achieved by synchronizing two TPMs, one for A and one for B, respectively. In this paper, we introduce a mechanism, which is based on the generation of inputs by feedback.

A measure of the security of the system is the probability P_E that an attacking network is successful. We calculate P_E obtained from the best known attack for different model parameters and search for scaling properties of the synchronization time as well as for the security measure. It turns out that feedback improves the security significantly but it also increases the effort to find the common key when this effort is kept constant, feedback only yields an improvement of security (Ruttor A et al., 2004).

In this paper, we analyze the influence of learning rules and order parameters on neural synchronization of TPMs are presented in Sec. 2. Also, we explain the synchronization of two TPMs with feedback mechanism and it is described in Sec. 3. Here, we show that the analysis of the security, probability of successful attack of the Majority Flipping Attacker with simulation results and it is explained in Sec. 4.

2. Neural Synchronization

The proposed Tree Parity Machine is used by partners and an attacker in neural cryptography consists of K-hidden units and Y-left-dynamic hidden units (Prabakaran N et al., 2008) and Z-right-dynamic hidden units, each of them being a perceptron with an N-dimensional weight vector w_k (Godhavari T et al., 2005). When the hidden and dynamic hidden units receive an N-dimensional input vector x_k, these units produce the output bit.

The general structure of this network is shown in Fig.1. All inputs values are binary,

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

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

$$w_{ij} \in \{ -L, -L+1, \ldots\ldots, L-1, L \},$$

$$w_{im} \in \{ -L, -L+1, \ldots\ldots, L-1, L \},$$

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

where, L is the depths of the weights of the networks.

The index j = 1,.....,N denotes the j^{th} hidden unit of TPM, m=1,.....Y denotes the m^{th} left-dynamic hidden unit of the TPM and k=1,.....,Z denotes the k^{th} right-dynamic hidden unit of the TPM. We mainly consider three transfer functions as given below

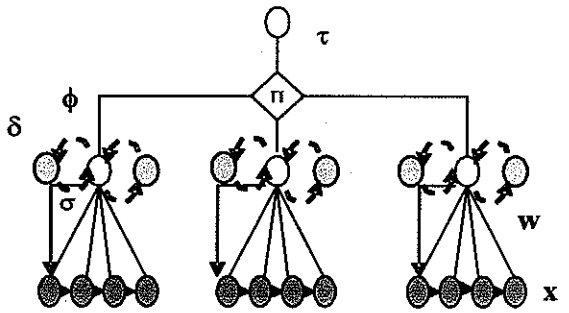

Figure 1 : A Structure of Tree Parity Machine with K=3, Y=3, Z=3 and N=4

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

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

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

where, equation (3) is the transfer function of the hidden unit (Kinzel W et al., 2000), the equation (4), is the transfer function of the left-dynamic hidden unit and

equation (5), is the transfer function of the right-dynamic hidden unit.

The K-hidden units of σ_i left-dynamic hidden units of δ_i and right-dynamic hidden units of γ_i (Prabakaran N et al., 2008) define a common output bit τ of the total network and is given by

$$\beta_a = \prod_{i=1}^{K} \sigma_i \quad (6)$$

$$\beta_b = \prod_{i=1}^{Y} \delta_i \quad (7)$$

$$\beta_c = \prod_{i=1}^{Z} \gamma_i \quad (8)$$

where, equation (6) is output for the hidden units, equation (7) output for the left- dynamic hidden units and equation (8) output for the right-dynamic hidden units.

The two TPMs compare the output bits and then update the values between hidden units and dynamic hidden units as well as two parties A and B that are trying to synchronize their weights

$$\psi_i^{A,B} = comp(\beta_a, \beta_b, \beta_c) \quad (9)$$

$$\phi_i^A = w_{ij}^A \, x_{ij}^A \, \tau^B \psi_i^A \quad (10)$$

$$\phi_i^B = w_{ij}^B \, x_{ij}^B \, \tau^A \psi_i^B \quad (11)$$

where, equation (9) represents comparison of the output of hidden, left-dynamic and right-dynamic hidden units. The equation (10) and (11) represent output of hidden, left and right-dynamic hidden units of A and B respectively.

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 vectors $w_i(A)$ and $w_i(B)$ at t=0.

2.1 Learning Rules

Both of the networks are trained by their mutual output bits τ^A and τ^B. At each training step, the two networks receive common input vectors x_i and the corresponding output bit τ of its partner (Ruttor A et al., 2004). We use the following learning rule

(i) If the output bits are different, $\tau^A \neq \tau^B$, nothing is changed.

(ii) If $\tau^A = \tau^B \equiv \tau$, the hidden and dynamic units are trained which have an output bit identical to the common output $\phi_i^{A/B} = \tau^{A/B}$.

(iii) To adjust the weights, we consider two different learning rules.

(a) Hebbian Learning rule for hidden units

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

$$w_i^B(t+1) = w_i^B(t) + x_i \tau^B \Theta(\tau^B \phi_i^B) \Theta(\tau^A \tau^B) \quad (12)$$

where, Θ is the Heaviside step function (Engel A and Van Den Brock C, 2001), 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 (Prabakaran N et al., 2008)

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

$$w_i^B(t+1) = w_i^B(t) + x_i \Theta(\tau^B \phi_i^B) \Theta(\tau^A \tau^B) \quad (13)$$

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

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

$$w_i^B(t+1) = w_i^B(t) - \phi_i x_i \Theta(\tau^B \phi_i^B) \Theta(\tau^A \tau^B) \quad (14)$$

2.2 Order Parameters

The size of a matrix F is (2L+1) × (2L+1) in TPMs. Their elements are $F^i(\mu)$, $F^j(\mu)$ and $F^k(\mu)$ where, 'μ' is the state of the machines in the time step, 'i' is hidden units, where, 'j' is the left-dynamic hidden units and 'k' is the right-dynamic hidden units. The element f_{qr}^{i} of matrix stands for the matching components in the i^{th} weight-vector in which the A's components are equal to 'q' and the matching components of B are equal to 'r'. The element $f_{s,t}^{j}$, matching components of j^{th} weight-vector in which the A's components are equal to 's' and the matching components of B are equal to 't' (Rosen-Zvi M, et al.,2002). The element f_{uv}^{k}, matching components of k^{th} weight-vectors in which the A's components are equal to 'u' and the matching components of B are equal to 'v'. The values of q, r, s, t, u, v are equal to -L, ... -1, 0, 1,....., L. The overlap of the weights belonging to the i^{th} hidden unit, j^{th} left-dynamic hidden unit and k^{th} right-dynamic hidden unit in the two parties are given below:

$$R_i^{A,B}=\frac{w_i^A \cdot w_i^B}{N},\ R_j^{A,B}=\frac{w_j^A \cdot w_j^B}{N},\ R_k^{A,B}=\frac{w_k^A \cdot w_k^B}{N} \tag{15}$$

Also their norms $Q_i=\frac{w_i^A \cdot w_i^A}{N}$, $Q_j=\frac{w_j^A \cdot w_j^A}{N}$ and $Q_k=\frac{w_k^A \cdot w_k^A}{N}$ are hidden, left and right-dynamic hidden units of A's TPM respectively. $Q_i=\frac{w_i^B \cdot w_i^B}{N}$, $Q_j=\frac{w_j^B \cdot w_j^B}{N}$ and $Q_k=\frac{w_k^B \cdot w_k^B}{N}$ are hidden, left and right-dynamic hidden units of B's TPM respectively. They are given by the matrix elements

$$R_i^{A,B} = \sum_{q,r} qr\, f_{\mathrm{qr}}^{\mathrm{i}} \tag{16}$$

$$R_j^{A,B} = \sum_{s,t} st\, f_{\mathrm{st}}^{\mathrm{j}} \tag{17}$$

$$R_k^{A,B} = \sum_{u,v} uv\, f_{\mathrm{uv}}^{\mathrm{k}} \tag{18}$$

The equation (16-17) represent overlap between two hidden units, two left-dynamic hidden units and two right-dynamic hidden units of A and B respectively.

$$Q_i^A=\sum_{q,r} q^2 f_{qr}^{i}\ , \qquad Q_i^B=\sum_{q,r} r^2 f_{qr}^{i} \tag{19}$$

$$Q_j^A=\sum_{s,t} s^2 f_{st}^{j}\ , \qquad Q_j^B=\sum_{s,t} s^2 f_{st}^{j} \tag{20}$$

$$Q_k^A=\sum_{u,v} u^2 f_{uv}^{k}\ , \qquad Q_k^B=\sum_{u,v} v^2 f_{uv}^{k} \tag{21}$$

The equation (19-21) represent weight distribution of hidden units, left-dynamic hidden units and right-dynamic hidden units of A and B respectively.

These overlaps and norms fixed the probabilities of deriving the same internal representation via the normalized overlap, $\rho_i^{A,B}=\frac{R_i^{A,B}}{\sqrt{Q_i^A Q_i^B}}$, $\rho_j^{A,B}=\frac{R_j^{A,B}}{\sqrt{Q_j^A Q_j^B}}$ and $\rho_k^{A,B}=\frac{R_k^{A,B}}{\sqrt{Q_k^A Q_k^B}}$ then

$$\rho_{ijk}^{A,B}=\frac{R_i^{A,B}}{\sqrt{Q_i^A Q_i^B}}+\frac{R_j^{A,B}}{\sqrt{Q_j^A Q_j^B}}+\frac{R_k^{A,B}}{\sqrt{Q_k^A Q_k^B}} \tag{22}$$

More precisely, the probability of having different results in the i^{th} hidden unit, j^{th} left-dynamic hidden unit and k^{th} right-dynamic hidden unit of the two parties is given by the well-known generalization error for the perceptron is given below

$$\varepsilon_p^i = \frac{1}{\pi}\arccos\ (\rho_{ijk}) \tag{23}$$

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. Since the hidden unit, left-dynamic hidden unit, right-dynamic hidden units are independent, also 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.

$$P_a = \frac{1}{2}\frac{(1-\varepsilon)^9 + 3(1-\varepsilon)^3\varepsilon^2}{(1-\varepsilon)^9 + 9(1-\varepsilon)^3\varepsilon^2} \tag{24}$$

$$P_r = \frac{6(1-\varepsilon)^3\ \varepsilon^2}{3(1-\varepsilon)^9 + 9(1-\varepsilon)^3\varepsilon^2} \tag{25}$$

The equation (24) and (25) 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.

3. Synchronization With Feedback

The TPMs A and B start with different random weights and common random inputs. The feedback mechanism is defined as follows.

(i) After each step 't' the input is shifted, $x_{i,j}(t+1) = x_{i,j-1}(t)$ for j >1.

(ii) If the output bits agree, $\tau^A(t) = \tau^B(t)$, the output of each hidden unit is used as a new input bit, $x_{i,1}(t+1) = \phi_i(t)$ are set to common public random values.

(iii) After R steps with different outputs, $\tau^A(t) = \tau^B(t)$, all input vectors are reset to public common random vectors, $x_{i,j}^A(t+1) = x_{i,j}^B(t+1)$.

The feedback creates correlations between the weights and the inputs and therefore the system becomes sensitive to the learning rule. The components of the weights have a broad distribution in Anti-Hebbian Learning rule. The entropy per component is larger than 99% of the maximal value of ln(2L+1). For Hebbian or Random walk rule, the entropy is much smaller, because the values of the weights are pushed to the boundary values ± L (Ruttor A et al., 2004). Therefore, the network with the anti-Hebbian rule offers less information to an attack than the two other rules. In fig.2, we have numerically calculated the average synchronization time as a function of the number L of components for the Hebbian rule of hidden units, Random walk rule of left-dynamic hidden units and Anti-Hebbian rule of right-dynamic hidden units. There is a large deviation from the scaling law $t_{sync} \propto L^2$ as observed for R=0.

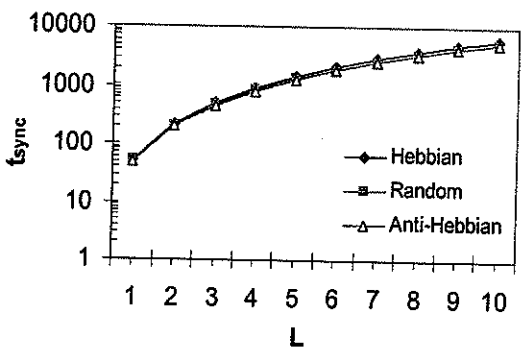

Figure 2 : Average Synchronization Time Between Hebbian, Random Walk And Anti-Hebbian Of T_{sync} And Its Standard Deviation As A Function Of L, From TPM With K=3. Simulation Results Obtained Using $N=10^3$

4. Simulation Results

The attacker E tries to learn the weight vector of one of the two machines (Mislovaty R et al., 2004). The values of N and L are public as well as all the transmission through the channel: input x_{ij} and output $\tau^{A/B}$. The information E lacks in each learning step are the values $\{\sigma_i\}$ of A's hidden units, $\{\delta_i\}$ of A's left-dynamic hidden units and $\{\gamma_i\}$ of A's right-dynamic hidden units. That is $2^{K-1} * 2^{Y-1} * 2^{Z-1}$ possible updating scenarios A performs.

The most successful attack on neural cryptography is the Majority Flipping Attack, which is an extension of the Geometric Attack. The Attacker E uses an ensemble of 'M' TPMs. At the beginning, the weight vectors of all attacking networks are chosen randomly, so that the average initial overlap between them is zero. If the two partners A and B use queries for the neural key exchange, the success probability strongly depends on the parameter H. This can be used to regain security against the Majority Flipping Attack.

$$P_E = \frac{1}{1+e^{-\beta(H-\mu)}} \qquad (26)$$

where, H is the absolute set value of the local field, β is sensitivity of P_E in regard of H and μ is increases linearly with the synaptic depth. The two parameters β and μ are a suitable fitting function for describing P_E as a function of H.

Here the probability of Majority Flipping Attack $P_{flip(C)}$ decreases exponentially with increasing synaptic depth in Hebbian Learning Rule

$$P_{flip(C)} \propto e^{-CL} \qquad (27)$$

where, C increases linearly with R,

$$C = 0.09 + 1.06 \times 10^{-3} R$$

The probability of Majority Flipping Attack $P_{flip(D)}$ in Random Walk Learning Rule

$$P_{flip(D)} \propto e^{-DL} \qquad (28)$$

where, D increases linearly with R,

$$D = 0.0908 + 1.06 \times 10^{-3} R$$

The probability of Majority Flipping Attack $P_{flip(F)}$ in Anti-Hebbian Learning Rule

$$P_{flip(F)} \propto e^{-FL} \qquad (29)$$

where, F increases linearly with R

$$F = 0.0908 + 1.06 \times 10^{-3} R$$

The addition of three transfer functions are given by

$$P_{flip(C+D+F)} \propto e^{-(C+D+F)L} \qquad (30)$$

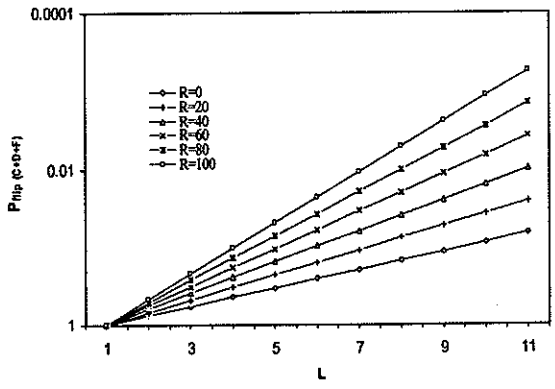

Figure 3: The Probability $P_{flip(C+D+F)}$ As A Function Of L, Averaging 1000 Simulation With K=3, Y=3, Z =3 And M=1000 In Hebbian Learning Rule, Random Walk Learning Rule And Anti-Hebbian Learning Rule

We are able to predict the improvement of the security of success of attack from fig. 5. Also, we have show that the success probability of an attacker (P_E) is decreased using Hebbian learning rule for hidden unit, Random walk learning rule for dynamic unit and right-dynamic hidden unit for Anti-Hebbian learning rule in majority flipping attacks.

From the above results, we are able to identify that the feedback improves the security of neural cryptography. The synchronization time on the other side is also increased.

5. Conclusions

In the proposed TPMs, the synchronize time of the attacker is increased by the addition of three transfer functions for 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. A feedback mechanism for hidden, left-dynamic and right-dynamic hidden units has been increased the synchronization time of two networks and decreases the probability of a successful attack of the Majority Flipping Attack. The synchronization with feedback yields an improvement of the security of the system. The TPMs generate a secret key and also encrypt and decrypt a secret message.

References

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

[2] Godhavari .T, Alamelu .N.R and Soundarajan.R, *"Cryptography Using Neural Network"*, IEEE Indicon 2005 conference, PP.258-261, 2005.

[3] Kinzel.W, Metzler.R and Kanter.I, *"Dynamics of Interacting neural networks"*, J. Phys. A: Math, Gen. 33, PP.L141-L149. [cond-mat/ 9906058], 2000.

[4] Kanter .I, Kinzel.W and Kanter .E, *"Secure exchange of informationby synchronization of neural network"*, Euro Physics Letters Vol.57, No.1, PP.141-147. [cond -mat/0202112], 2002.

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

[6] Kanter. I and Kinzel .W, *"Neural cryptography"*, In: Proc. of the 9th International Conference on Neural Information Processing, Singapore, 2002.

[7] R. Mislovaty, E. Klein, I. Kanter and W. Kinzel, *"Security of Neural cryptography"*, IEEE Transactions, Vol.5, No.4, PP.219-221,2004.

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

[9] Prabakaran .N, Karuppuchamy. P and Vivekanandan .P, *"A New Approach on Neural Cryptography with Dynamic and Spy Units using Multiple Transfer Functions and Learning Rules"*, Asian Journal of Information Technology, Vol. 7(7), PP.300-306, 2008.

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

[11] Rosen-Zvi.M, Kleign.E, Kanter.I and Kinzel.W , *"Mutual learning in a tree parity machine and its application to cryptography"*, Phys. Rev. E, Vol. 66, 066135 (13) ,2002.

[12] Ruttor .A , Kinzel .W, Shacham. L and Kanter .I, *"Neural cryptography with feedback"*, Phy. Review E, Vol. 69, PP.1- 8 , 2004.

[13] Ruttor.A, Kanter.I and Kinzel.W, *"Dynamics of neural cryptography"* [cont-mat/061257], 2006.

[14] Ruttor.A , Kanter.I , Naeh.R and Kinzel.W, *"Genetic attack on neural cryptography"*,[cond-mat/0512022v2

Author's Biography

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.

Loganathan.P, received M.Sc. degree from Presidency College, Chennai. He received his Ph.D. in Mathematics from Anna University. He is working as Assistant Professor in Department of Mathematics, Anna University. His areas of interest are Computational Fluid Dynamics, Heat and Mass transfer, Natural connection and Object Oriented Programming. He has published about ten papers in national and international conferences.

Prabakaran.N , received M.Sc in Computer Science from Anna University. He is doing research under the guidance of Dr. P. Vivekanandan. His areas of interest are network security, visual programming, computer networks and wireless technology.He had published five International Journals and six papers in national and international conferences.