Image Hash using Neural Networks

Veena Desai
Research Centre ,Dept of E&C,
GIT, Belgaum, India

D H Rao, PhD.
Jain College of Engg,
Belgaum, India

ABSTRACT
Hash functions have been used to generate hash codes for data authentication. Traditionally these functions are generated using byte oriented algorithms like MD5 and others. In our paper we propose a new method of generating hash code for images using neural networks. Three sample images namely, fingerprint, lena and football image have been considered and their hash values calculated using two neural network structures namely, 1) structure without feedback 2) structure with feedback. The original images are then subjected to bit modification,Gaussian noise and rotational noise. The hash values are recalculated for the modified images. Sensitivity and hit collision are calculated and are found to be comparable with that of MD5 algorithm.

General Terms
Neural Networks, Cryptography, Security

Keywords
Image hash, neural hash, hash sensitivity, hit collision

1. INTRODUCTION
A hash function H is a transformation that takes a variable-size input message M and returns a fixed-size hash string H, which is called the hash value. A Hash function f(M) generates a unique hash value H for a particular image and thus can be used for checking data integrity and authentication purposes. When image messages are considered preimage resistance and hit collision are parameters for evaluating the performance of hash functions.

The applications of neural networks in areas of cryptography in general are discussed in [1-8].The authors of [9-10] have used both chaos and neural networks in data encryption because of their cipher-suitable properties, such as parameter-sensitivity, time-varying, random-similarity, etc. Based on chaotic neural networks, a hash function is constructed, which makes use of neural networks' diffusion property and chaos' confusion property. This function encodes the plaintext of arbitrary length into the hash value of fixed length (typically, 128-bit, 256-bit or 512-bit).

Another demonstration of hash function implementation based on conservative chaotic system is proposed by authors of [11]. In their implementation the plaintext is divided into a group of message blocks by a fixed length and each message block is iterated some times through standard map. Both the iterations results of every round and the plaintext block determine the two initial values and the steps of iterations in next round. Some items of the result in the final round are chosen to be transformed into hash value of 128 bits. In the paper [12] a hash function construction method based on cellular neural network (CNN) with hyper-chaos characteristics is proposed. The chaos sequence generated by iterating CNN with Runge-Kutta algorithm, then the sequence iterates with every bit of the plaintext continually. Then hash code is obtained through the corresponding transform of the latter chaos sequence from iteration. Hash code with different lengths could be generated from the former hash result. In [13] The MLP network structure developed for one way hashing consists of a hidden layer and an output layer. The hidden layer contains 64 neurons with 61 input including the bias. The weights of the hidden neurons are truncated to 3 decimal places, which α set to 1000. The MLP network thus structured is shown to be pre image resistance, 2nd pre image resistance and collision resistance features.

In section 2 of our paper the neural network structures used in the proposed implementation are illustrated and explained. Section 3 provides the proposed algorithm details. Result calculation and sample data are elucidated in section 4. Conclusions are discussed in section 5.

2. NEURAL NETWORKS
Artificial neural network is an interconnected group of artificial neurons which use a mathematical model or a computational model for information processing based on connectionist approach to computation. It is a network of simple processing elements which can exhibit complex behavior determined by the connections between processing elements and element parameters. Artificial neural network is an adaptive system that changes its structure based on external or internal information that flows through the network.

The method of setting the values for the weights enables the process of learning or training. The process of modifying the weights of the connections between network layers with the expected output is called training a network. The internal process that takes place when a network is trained is called learning. Figure 1 and Fig.2 show the neural network structure implemented in this paper without feedback and with feedback respectively.

The structure of feed-forward network as in Fig. 1 is made up of layers of neurons. For the purpose of the one-way hashing function, three layers of neurons are employed. The first layer is called 'input' layer, the next is the 'hidden' layer and the last layer is the 'output' layer. In our implementation, the input and the hidden layers consist of 128 neurons and the output layer consists of 64 neurons.

A sequence of binary bits are obtained from the image as mentioned in section 3.These bits are elements of out(1,n) from the input layer and are processed further by the hidden layer. The output of hidden layer becomes the input for the output layer. Due to the use of tan-sigmoidal function, non-linearity is introduced in each neuron of the output layer. In our structure, we have used a constant weight of 0.5 for each neuron and the bias is set to 0 for each neuron. Diffusion property is also satisfied due to the use of neural structure.

The structure of feedback network is again a structure of 3 layers of neurons. The interconnect between input and hidden layer output are as in feed forward structure.

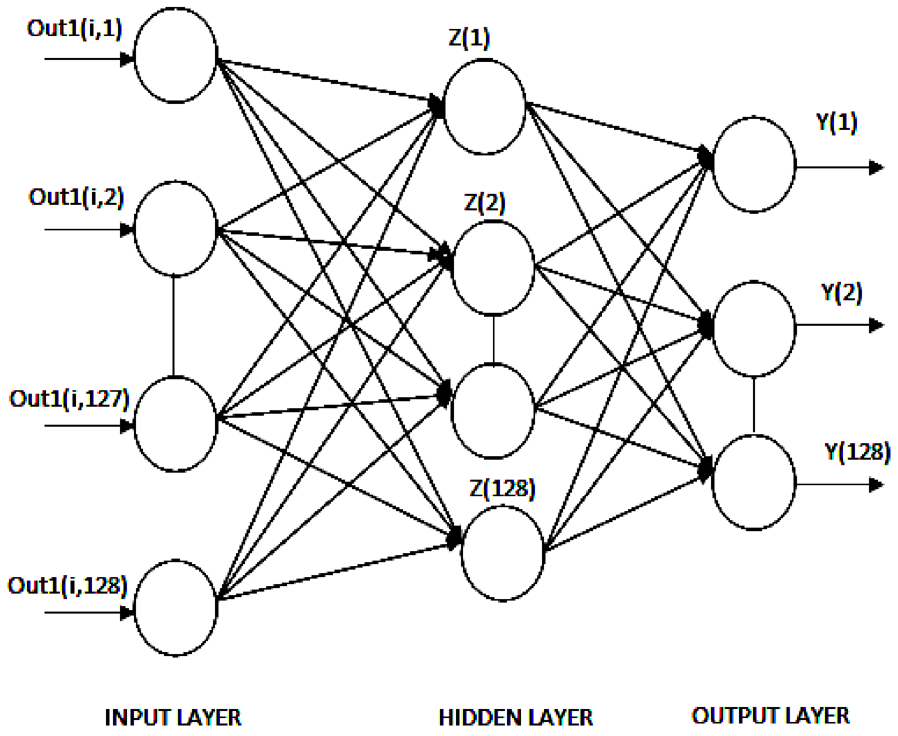

Fig 1. Neural network structure without feedback

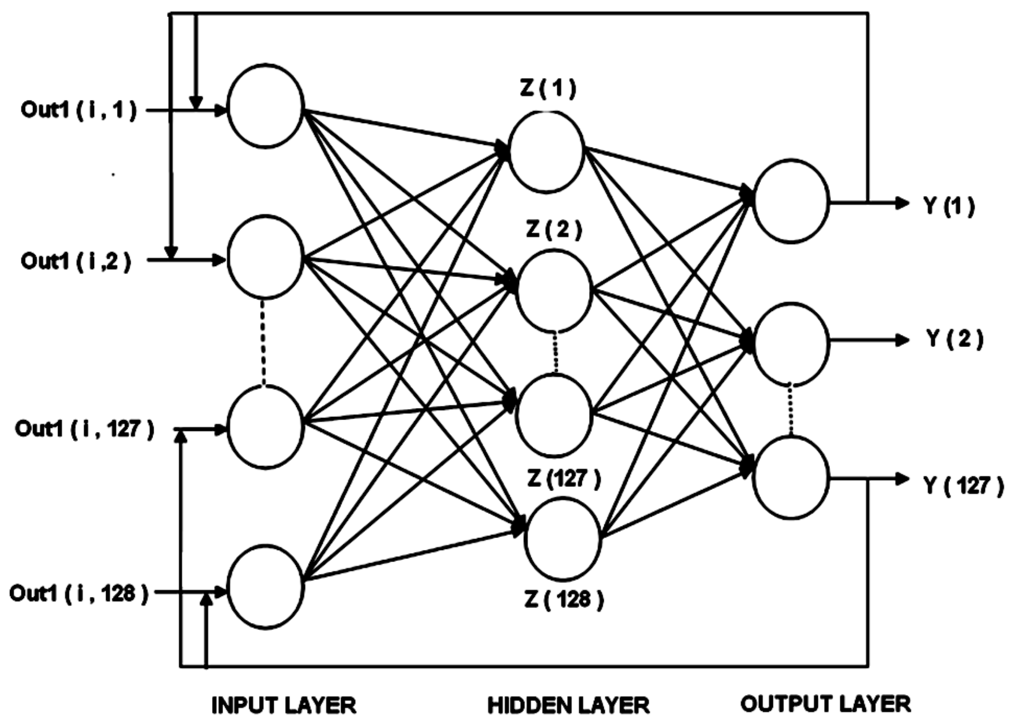

Fig 2. Neural network structure with feedback

Each output neuron is mapped from the output layer to two consecutive input layer neurons. For each iteration the input layer neuron takes into consideration the current input and previous output due to which stability is introduced in the structure thereby making it more immune to noise.

3. PROPOSED ALGORITHM

For neural network structure without feedback

1. *Read the input image*
2. *Convert the image into two dimensional pixel values*
3. *Determine the number of rows and columns of image*
4. *Create a row matrix of content values*
5. *Resize the row matrix into a matrix of size(r * 128). (r is the number of rows and it depends on the image which is being read)*
6. *Initialise each neuron in the input layer, hidden layer and output layer with a constant weight of 0.5 and a bias of 0.*
7. *Initialise the final layer output* $[y_1(1), y_1(2). . . .y_1(64)]$ *to zero*
8. *For k = 1 to r*
9. *If(k=1)*
10. *Let the final layer output be* $y_1(1), y_1(2) . . . .y_1(64)$
11. *End if*
12. *If (k>1)*
13. $h_{k-1} = [h_{k-2}(1)$ EXOR $y_k(1)],[h_{k-2}(2)$ EXOR $y_k(2)] . . . . [h_{k-2}(64)$ EXOR $y_k(64)]$
14. *End if*
15. *End for k*
16. *The hash value of image without noise is* h_{k-1}
17. *Modify the image by introducing one of the following four noises: One-bit change / Gaussian filtering / +5 degrees rotation / -5 degrees rotation.*
18. *Repeat Steps 7 to 15 for the modified image.*
19. *Let the hash of the modified image be* Hnoise $_{k-1.}$
20. *Calculate the difference between* h_{k-1} and Hnoise $_{k-1.}$

For neural network structure with feedback

Repeat Steps 1 to 7 of without feedback

1. *For k=1 to r*
2. *If(k>1)*
3. *Initialise t to1*
4. *for j = 1 to 128 in steps of 2*
5. *out1(k,j) = EXOR[y(k-1,t),out1(k,j)];*
6. *out1(k,j+1) = EXOR[y(k-1,t),out1(k,j+1)];*
7. *t=t+1;*
8. *if(t == 65)*
9. *End for j*
10. *End if*
11. *End for k*
12. *Repeat Steps 16 to 20 of Without Feedback*

The input image is converted into two dimensional pixel values. 128 values of this matrix are given at a time to the input layer of the neural network. These values are passed through the hidden layer and 64 values of 38 bits each are obtained from each neuron of the output layer.

For the without feedback neural network structure, the output obtained from the consecutive iterations are XOR'ed to get the final hash value for the particular image.

For the with feedback neural network structure, the values generated by each output neuron for the previous iteration are Xor'ed with the input values to two consecutive neurons in the current iteration. Let the hash value generated for the original image for both without feedback and with feedback structure be h1. Noise is introduced into the image and the procedure is repeated to obtain the new hash for the modified image. Let this hash be represented as h2.

Sensitivity gives the number of bits change in the original image after addition of noise and is calculated as:

%Sensitivity= (Number of bits changed/ Total number of bits) * 100

Hit collision is the number of digits (hex values) remaining same in the hash after addition of noise in the original image and is calculated as:

%Hit collision = (No. of hex values remaining same/ Total No. of hex values) * 100

4. SAMPLE DATA AND RESULTS

The following figures, 3(a) to 3(e) show the sample set of images that have been used for hash calculation. Fingerprint image 1(a) is the original image of sample 1. Fig. 4(b) to 4(e) are the modifications on sample 1 with 1 bit change, filtered image ,with +5 degrees rotation, and -5 degrees rotation respectively.

3(a)

3(b)

3(c)

3(d)

3(e) 3(f) 3(g)

Fig 3. (a)original fingerprint image (b) fingerprint image with 1 bit modification (c) fingerprimt image with gaussian noise (d)fingerprint image with +5 degrees rotation (e)fingerprint image with -5degreees rotation (f)original lena image (g)original football image

Hash value of fingerprint image :

BCC85FBEC44BB32600AA960A6EB3C1B538590D2254B4
8A4261E1290D73A67C360363684A938C2C8E7E687400CB
7FB2482D0C3BA6EB536BD379B766FF3EEE31216A0541
CA9FCBA72C0AF78F8F8EB4B87522A465C01B70B8EC74
B7CDAD8127B50E41DA38BA97BCD1DFA21A693507C6
CAFEBFE49BAD926FFA24B47615B37A99F738E0AF6CF6
7038559AE920A4DF5E76E84D20C836A353F9D889845C3
7E648D7735EBB1B672A189307C395FAE9BC0E1D8D10A
B73557321538CA765C816E5CC0B02B7238E2C5E7B57108
96A60D08DECF65D3109831F396F74FD65232153DEE2F6
4C2CD9A2FC31A38482513EECFE4D746B700F2AF6616B
A9ABD4211579786CF06ECC4EB08841CC1425A6BA3239
9518F3A1E07C69921243FFE5A835A89AD81969B3D21BA
9E47ACFC28AF19E7E40E0A0F400077B

Hash value for fingerprint image with 1-bit change without feedback structure:

BE6098D3387B95840EC11E3EF088E417A82D3FB2F660FF80C
4AA37EE60FF6D61D829561132562504561EDE81FC7A6320D
D5E72B84B62A1FC2D156B1D3002B70D4AD8D9959CB61ACE
8D6E998F9A707D0970D4D9478EB0128A21A1A73F913D2E13
F7AFA58497B827ED09CA53A0E6CD272209CF9393ABB20EF4
1FC1FC6AFD7E46F526822749978B38C07BC69E9855D253A9
5198F086DDBED2C42E6BA4A224C21081CC62EAED4E9286C
AE17550B1CE3A76E2335B9DDDAB108866B8C5AEF6D3CF90
6CB460CC3089F0CEAB176533754B4D632DF794A29CC432D3
912252C3C3140287A35367650113DDAB03EED5900DC3AA3
DA3FD2DA0A739E199169C58FE3860F549A2EBE860738260A
46026CD9A50D14FEA4963FC95E0C

Number of bit changes observed in the New hash: 1174

Sensitivity: 48.2730 %

Hit Collision: 8.8816 %

Elapsed Time: 32.7 minutes

Hash value of fingerprint image with 1-bit change with feedback structure

130F878378CE576E8E0298AE7D13489058DE5800B3DF2EFC
BD08AE41F27A2648B8079A3176DA3E8051D508B1B1EB6E92
5A383BFB45D42D6D8835C7EEBA90A8F30DFE32343D94AD4
1B7D9BE02308AB3180CA136AE3B11FFB301714FFEEC85222
D516DE64499773BC111A6F646B58E20880ED82AF655AEA89
0AC00CDA0E3785247CC83E621A928F8DBAEE4852E482E800
000002A4B942401E927101FFB3C733A65EEA3BC5C832A0AE
EC3712924FDCBC8BC4F4E55E0A33E69090F4A69A1D9DE226
C40C514052EEFA9D2F729A53409EF5E81DD1EAD829CC8FBA
9101FA2C8F1614749706130A417DEBA4E18D8B0AE0D92A59
654EC88DA7722590818EB49F4C30DDBE0828689219D480CF
F496A3FFF80C094B5DEB74A6267A568D386B66FEB3B38FDA
EC4BCAC51EABE2244D65AEC03CE5821C7

Number of bit changes observed in the New hash: 1115

Sensitivity: 45.8470 %

Hit Collision: 10.0329 %

Elapsed Time: 58.32 minutes

Table 1 provides the comparison of sensitivities for hash values obtained from the neural network without feedback and MD5 algorithm for the fingerprint image and Lena image. In both cases we find that the sensitivity of the hash obtained from the neural structure is comparable with that of MD5 algorithm.

Table 1: Comparison of sensitivities of sample images and MD5

Comparison made for image fingerprint		
Type of noise	**Sensitivity**	**Sensitivity (MD5 Tool)**
One-bit change	**48.27 %**	**57.03 %**
+5 degrees rotation	**50.25 %**	**47.65 %**
-5 degrees rotation	**49.79 %**	**48.43 %**
Filtering	**48.23 %**	**49.21 %**

Comparison made for image Lena

Type of noise	Sensitivity	Sensitivity (MD5 Tool)
One-bit change	49.09 %	53.90 %
+5 degrees rotation	47.82 %	53.90 %
-5 degrees rotation	48.97 %	49.21 %
Filtering	47.62 %	53.13 %

Figures 4,5,6 and 7 are plots of bit changes, sensitivity, hit collision and time elapsed for variations in 1 bit change, Gaussian noise,+5 degree rotation and -5 degree rotation.

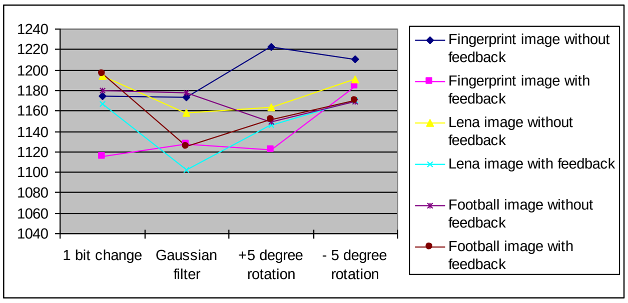

Fig 4: plot of bit changes for variations in noise types

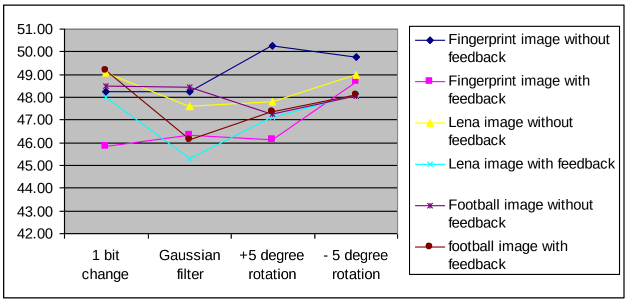

Fig 5: plot of sensitivity for variations in noise types

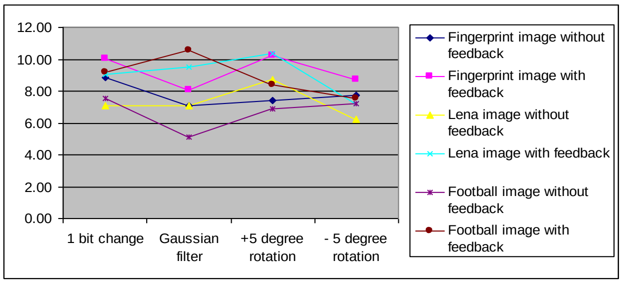

Fig 5: plot of hit collision for variations in noise types

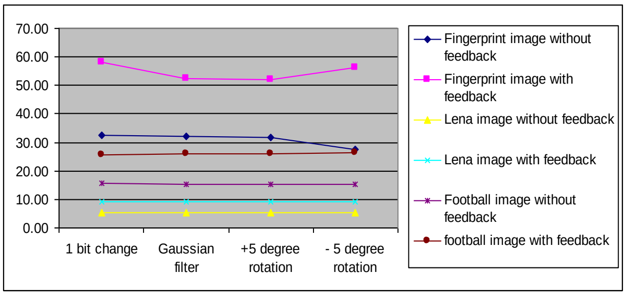

Fig 7: plot of time delay for variations in noise types

Table. 2 and Table. 3 show the results of the hash algorithm obtained from neural structures without feedback and with feedback for lena image and fingerprint image respectively. Figure 8 and fig.9 are plots of bit changes for each neuron in ouput layer for without feedback structure and with feedback structure.

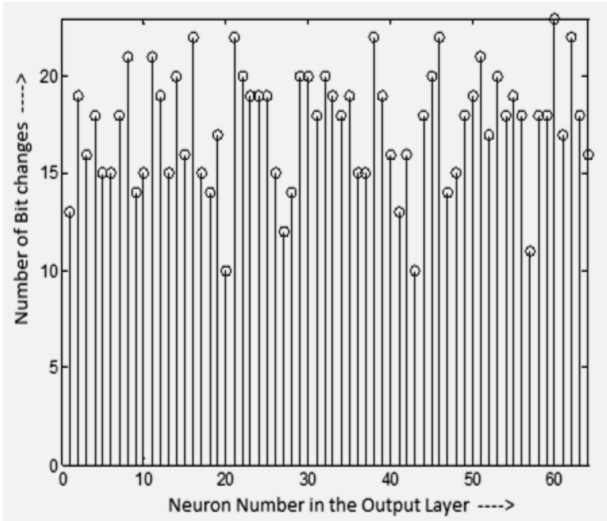

Fig. 8 plot of bit changes for each neuron in ouput layer for without feedback structure

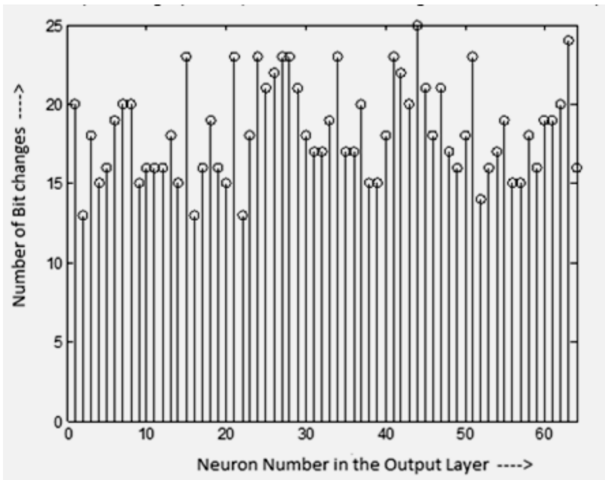

Fig. 9 plot ob bit changes for each neuron in ouput layer for with feedback structure

Table 2: Results for lena image

Structure	Type of Noise	Bit Changes	Sensitivity	Hit collision	Time elapsed
Without feedback	1 bit change	1194	49.09	7.0724	5.33
	Gaussian filter	1158	47.61	7.0724	5.52
	+5 degree rotation	1163	47.82	8.7171	5.51
	- 5 degree rotation	1191	48.97	6.2500	5.41
With feedback	1 bit change	1167	47.98	9.0461	9.35
	Gaussian filter	1102	45.31	9.5395	9.23
	+5 degree rotation	1146	47.12	10.3618	9.00
	- 5 degree rotation	1170	48.10	7.2368	9.13

Table 3: Results for fingerprint image

Structure	Type of Noise	Bit Changes	Sensitivity	Hit collision	Time elapsed
Without feedback	1 bit change	1180	48.51	7.5658	15.66
	Gaussian filter	1178	48.43	5.0985	15.38
	+5 degree rotation	1149	47.24	6.9079	15.31
	- 5 degree rotation	1169	48.06	7.2368	15.35
With feedback	1 bit change	1196	49.17	9.2105	25.58
	Gaussian filter	1125	46.125	10.5263	26.15
	+5 degree rotation	1152	47.36	8.3882	26.08
	-5 degree rotation	1170	48.10	7.5658	26.43

5. CONCLUSION

In this paper a unique hash value for a given image using neural networks is obtained. The code is implemented in MATLAB and tested for a wide range of images and a unique hash obtained for each image. From the results it can be inferred that the unique hash value obtained is sensitive to modifications made to the input image. The sensitivity obtained is in the range of 40-50%. For without feedback neural network structure sensitivity has been calculated using MD5 tool. These sensitivity values are compared with the values obtained from the proposed scheme using neural networks. The hit collisions obtained for the modified image is found to be less than 10%. Lesser the hit collision better is the efficiency of the algorithm and thereby making cryptanalysis difficult. The limitation of this implementation is that the time taken to generate the unique hash depends upon the resolution and the information carried by the image. Hence it is observed that the elapsed time is more for the fingerprints as compared to the standard lena image.

In this paper an image hash has been generated using simple feed-forward and feedback neural network structures. Other neural network structures like back propagation network(BPN), radial basis function network(RBFN),discrete Hopfield networks, continuous Hopfield networks etc can be explored and checked for better sensitivity.

6. REFERENCES

[1] Khalil Shihab, “A Back propagation Neural Network for Computer Network Security” Journal of Computer Science 2 (9): 710-715, 2006.

[2] 2. D.A. Karras and V. Zorkadis. On neural network techniques in the secure management of communication systems through improving and quality assessing pseudorandom stream generators. Neural Networks, Vol. 16, No. 5-6, June - July, 2003: 899-905.

[3] S.G. Lian, G.R. Chen, A. Cheung, Z.Q. Wang. A Chaotic-Neural-Network-Based Encryption Algorithm for JPEG2000 Encoded Images. In: Processing of 2004

IEEE Symposium on Neural Networks (ISNN2004), Dalian, China, Springer LNCS, 3174 (2004) 627-632.

[4] V.V Desai,V B Deshmukh,Rao D H “Pseudo random number generator using Elman neural nework”,RAICS, IEEE conf. proceedings,pp. 251-254, Nov 2011.

[5] V.V Desai, Ravindra..T.P, V B Deshmukh, Rao D H “Pseudo random number generator using Time Delay neural nework”,World Journal of Science Technology,2012,2(10):165-169

[6] C.-K. Chan and L.M. Cheng. The convergence properties of a clipped Hopfield network and its application in the design of key stream generator, IEEE Transactions on Neural Networks, Vol.12, No. 2, pp. 340-348, March 2001.

[7] Liew Pol Yee and De Silva, L.C. "Application of Multilayer Perceptron Networks in symmetric block ciphers” Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, HI, USA, Vol. 2, 12-17 May2002:1455-1458.

[8] Yi Du, Detang Lu, Daolun Li, “An Effective Hash-based Method for Generating Synthetic Well Log” 2008 IEEE.

[9] Shiguo Lian, Zhongxuan Liu, Zhen Ren, Haila Wang, “Hash Function Based on Chaotic Neural Networks” IEEE, 2006.

[10] Shiguo Lian, Jinsheng Sun, Zhiquan Wang, “One-way Hash Function Based on Neural Network” Journal of Information Assurance and Security, 2006.

[11] Qinghua Zhang, Han Zhang and Zhaohui Li ”One-way Hash Function Construction Based on Conservative Chaotic Systems” Journal of Information Assurance and Security, 5, pp.171-178, 2010.

[12] Qun-ting Yang, Tie-gang Gao, Li Fan, Qiao-lun Gu “Analysis of One-way Alterable Length Hash Function Based on Cell Neural Network” Fifth International Conference on Information Assurance and Security, 2009.

[13] L.P. Yee, D. L.C. Silva, "Application of Multilayer Perceptron Network as a One-way Hash Function” International Joint Conference on Neural Networks, Vol. 2, 2002