# Implementation of SDES and DES Using Neural Network

Sojwal S. Kulkarni
alias R.M. Jogdand
*Associate Professor Department of Computer Science & Engineering*
*GIT, Belgaum*

Dr. H.M.Rai
*Ex-Professor- National Institute of Technology, Kurukshetra*

Dr. Sanjay Singla
*Associate. Professor, OITM, Hisar*

**Abstract- In this work a new schedule for XOR design is going to be considered. We will first implement SDES algorithm in MATALAB they possess the most popular design criteria for block ciphers. they will replace the normal XOR function with neural network XOR function in the design process the motivation for the proposal is ability of neural networks to perform complex mapping function from one domain to another. Also the non –linearity produced by neural network mapping is desired in cryptography.**

**Keywords – Neural networks, MATALAB, Cryptography, SDES, XOR**

## I. INTRODUCTION

Computer security is a very important issue in this age of electronics connectivity because of viruses and hackers, electronic eavesdropping and electronic fraud. Explosive growth in computer systems and their interconnections via networks has increased the dependence of both the organizations and individuals on the information stored and communicated. This in turn has led to an awareness of the need to protect data and resources from disclosure, to guarantee the authenticity of data and messages and to protect systems from network based attacks.

Cryptography provides various algorithms to perform substitutions and transformation on the original text to produce unintelligible cipher text. This masking prevents security evasion. These standard algorithms generally use functions like Exclusive-OR to transform the original text. Also the keys used for encryption/decryption are generated using simple permutations. Since such functions/permutations are simple nonlinear functions it is possible to analyze the pattern of the cipher texts generated. This results in release of contents to the cryptanalysts/hackers.

To overcome some of the limitations of the traditional cryptographic methods, new design tools can be designed using Computational Intelligence (CI). The CI is a methodology involving computing that exhibits an ability to learn and/or to deal with new situations, such that the system is perceived to possess one or more attributes of reason, such as generalization, discovery, association and abstraction.

Silicon based computational intelligence systems usually comprise hybrids of paradigms such as artificial neural networks, fuzzy systems and evolutionary algorithms, augmented with knowledge elements, and are often designed to mimic one or more aspects of carbon based biological intelligence As CI systems comprise practical adaptation concepts, paradigms, algorithms and implementations that enable or facilitate appropriate actions (intelligent behavior) in complex and changing environment one of these component may be best suited to develop

this algorithm. In this work it is proposed to implement one of the CI tools, namely Artificial Neural Network (ANN) in the development and analyses of some of the cryptic algorithms.

An ANN is a massively parallel distributed processor made up of simple processing units, with natural propensity for storing knowledge and making it available for use. It emulates, in a feeble way, the features of human brain in two aspects:

1. Knowledge is acquired from its environment through the learning process and
2. Inter-neuron connections strengths, known as synaptic weights, are used to store the acquired knowledge.

Some of the Salient features of ANNs are

- Non Linearity
- Adaptability
- Contextual information
- Fault Tolerance, and
- Uniformity of Analysis and design.

It is envisaged that more cryptic algorithms can be developed by exploiting some of the salient features of ANN's, Cryptography algorithms may be made more robust by utilizing the non-linear functional approximation and learning features of ANNs.

## II. Related Work

1. Souhila Sadeg[13] *et. al* DNA cryptography is a new promising direction in cryptography research that emerged with the progress in DNA computing field. DNA can be used not only to store and transmit information, but also to perform computations. The massive parallelism and extraordinary information density inherent in this molecule are exploited for cryptographic purposes, and several DNA based algorithms are proposed for encryption, authentication and so on. The current main difficulties of DNA cryptography are the absence of theoretical basis, the high tech lab requirements and computation limitations. In this paper, a symmetric key bloc cipher algorithm is proposed. It includes a step that simulates ideas from the processes of transcription (transfer from DNA to mRNA) and translation (from mRNA into amino acids). This algorithm is, we believe, efficient in computation and very secure, since it was designed following recommendations of experts in cryptography and focuses on the application of the fundamental principles of Shannon: Confusion and diffusion. Tests were conducted and the results are very satisfactory. Tests were conducted in order to test the performances of algorithm in term of efficiency by running it on plain texts of different lengths and types (images, videos, texts…). Comparisons were also done between this algorithm and two versions of AES, and the results were very satisfactory. In term of security, this encryption algorithm, we believe, is robust .

2. Shen Guicheng[12] *et. al* in their work use object-oriented technology as tools, and divides Elliptic Curve Cryptosystem into several layers, every of which corresponds a class. The properties and methods of these classes are discussed, and some methods are implemented. In the end, the advantages are analyzed, and the cryptosystem implemented with advanced programming language is easy to transplant. As object-oriented technology have four characteristics: inheritance, polymorphism, and abstract, these characteristics enable us to design several classes, which collaborate with each other. Even if one class has changed its implementation, this does not affect other classes, therefore the flexibility and stability of the system are improved and system maintenance is very convenient. As the system is designed, developed with C++ language, it will provide a sound for the transplantation from one platform to another platform and there is no need to modify the program.

3. Albasil and Wohdan [6] discuss a block cipher based on neural networks. The Fiestal block cipher using neural network is considered.

4. Handessi, Gulliver and Sheikh [7] discuss an algorithm for the fast generation of large m x n s-box and test their cryptographic properties.

5. P.Kotlarz and Z.Kotulski [8] propose the integration of neural network playing the role of Boolean functions with appropriate properties for S-box design.

6. T.Godhavari and N.R.Alamelu [9] discuss the interaction of neural networks and cryptography to generate secret key over public channel.

7. M.Capsodi, J.Vandewalle and T.Roska [10] demonstrate the implementation of cryptography has functions on regular array of simple cellular neural network cells with periodic boundary conditions.

It has been observed in the above literature survey that ANNs have not been used yet in S-box design or any cryptographic algorithms so far. Hence, the proposed work embarks on the concept of integrating neural networks with cryptographic algorithms. It is also proposed to implement fuzzy logic techniques [11] in the algorithms, and to study the performance comparison of these two methods.

## III. IMPLEMENTATION OF XOR USING NEURAL NETWORKS.

In this section we see implementation of XOR using neural networks. It gives an idea as to how the neural network can be introduced into block ciphers to increase their nonlinearity.

*A Implementation*

*i. XOR:*

The XOR of SDES is designed using neural networks two layer neural networks. The first layer consists of 4 neurons and the second layer consists of 2 neurons. For this we used Feed Forward network and back propagation training is employed.

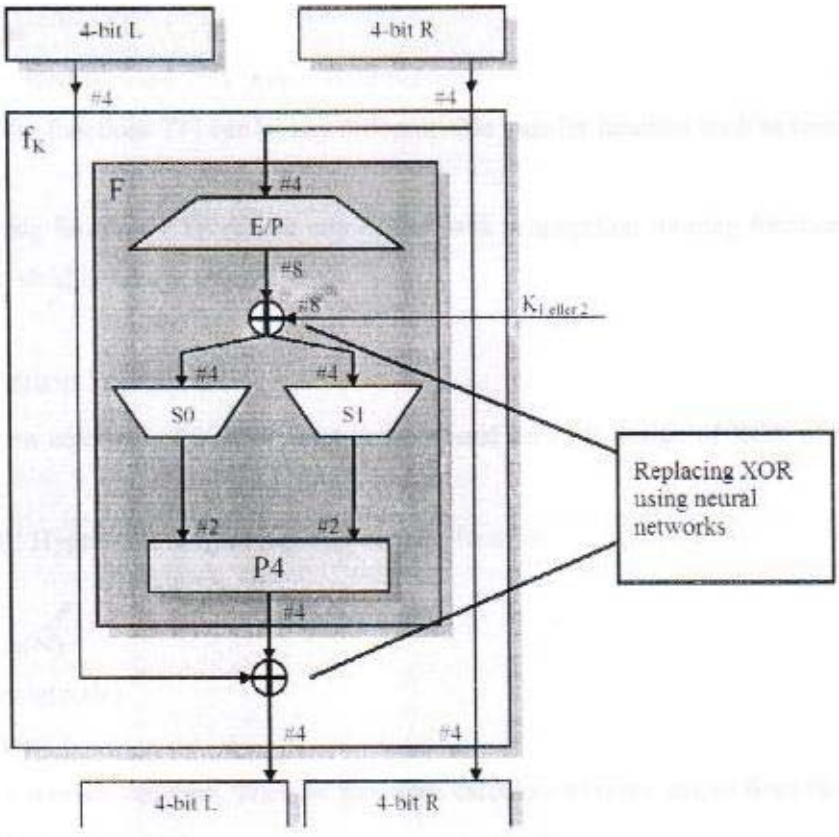


Figure 3.1 SDES Block

*ii. Description of Feed Forward network:*

The Description of Feed Forward network, which is used for creating a required neural network, is given as below:

Syntax

net=newff
net=newff(PR,[S1 S2….SN1],{TF1 TF2…TFN1},BTF,BLF,PF)
Description
net=newff creates a new network with a dialog box
newff(PR,[S1 S2….SN1],{TF1 TF2…TFN1},BTF,BLF,PF) takes,
PR—R x 2 matrix of min and max values for R input elements
Si—Size of $i^{th}$ layer,for N1 layers
TFi—Transfer function of $i^{th}$ layer, default='tansing'
BTF-Back Propagation weight/bias learning function, default='learngdm'
PF-Performance function, default='mse' and returns
an N layer feed-forward back propagation network.
The transfer functions TFi can be any differentiable transfer function such as tansig, logsig, or purelin.
The training function BTF can be any of the back propagation training functions such as trainlm, trainbfg, trainrp, traingd,etc.

*iii. Activation functions:*

The various activation functions used in the neural network design of XOR of SDES an DES are:

1. tansig: Hyperbolic tangent sigmoid transfer function

Syntax

A=tansig(N)
Info=tansig(code)
Description
Tansig is a transfer function. Transfer functions calculated a layers output from its net input tansig(N) takes one input,
N—S x Q matrix of net input (column) vectors
And returns
Each elements of N squashed between -1and 1.
Tansig(code)return useful information for each code string:
'deriv'—name of the derivative functions
'name'—Full name
'output'—output range
'active'—active input range tansig is named after the hyperbolic tangent, which has the same shape, However ,tan h may be more accurate and is recommended for application that required the hyperbolic tangent.

2. Purelin: Linear transfer function

Syntax

A=purelin(N)
Info=purelin(code)
Description
Purelin is a transfer function. Transfer functions calculate a layers output from its net input.
Purelin(N) takes one input,
N—Sx Q matrix of net input (column) vectors and returns N.

Purelin(code)returns useful information for each code string:
'deriv'—Name of derivative function
'name'—Full name
'output'—Output range
'active'—Active input range.

*iv.Training function*

The training function employee for training neural network is:
Training occurs according to the trainlm's Training parameters, shows here with their default values:
Trainlm: syntax
[net,TR] = trainlm(net,Pd,Tl,Ai,Q,TS,VV,TV)
Info =trainlm(code)
Description
Trainlm is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization.
trainlm(net,Pd,Tl,Ai,Q,TS,VV,TV) takes these inputs,
net--neural network
Pd--Delayed input vectors.
Tl--Layer target vectors.
Ai--Initial input delay conditions.
Q--Batch size.
TS--Time steps.
VV--Either empty matrix [] or structure of validation vectors
TV--Either empty matrix [] or structure of validation vectors
And returns,
Net--trained network.
Training occurs according to the trainlm's Training parameters shows here with their default values:

| | | |
|---|---|---|
| Net.trainParam.epochs | 100 | maximum number of epochs to train |
| Net.trainParam.goal | 0 | performance goal |

Validation vectors are used to stop training early if the network performance on the validation

Vectors fail to improve or remains the same for max_ fail epochs in a row. Test vectors are used as a further check that the network is generalizing well, but do not have any effect on training.

Trainlm(code) returns useful information for each code string:
'pnames' --Names of training parameters
'pdefaults--Default training parameters

*v.* Training stops when??

Training stops when any of this condition occurs:

- The maximum numbers of epochs(repetitions) is reached.
- The maximum amount of time has been exceeded.
- Performance has been minimized to the goal
- The performance gradient falls below mingra_mu exceeds mu_max
- Validation performance has increased more than max_fail times since the last time is decreased

After the implementation of XOR operation using neural network we now compare the result in next chapter .

## IV. RESULTS

In this work we are going to give comparison between the result obtained by using simple xor design for SDES and DES

The parameter considered for the comparison between normal and neural SDES AND DES as follows

Avalanche Effect:

It is property of any encryption algorithm that a small change in either plain text or the key should produce a significant change in the cipher text. In particular a change in one bit of plain text or one bit of the key should produce a change in many bits of cipher text

E.g. without neural
Plaintext = 1 1 1 1 1 1 1 1
Key (10-bit)=1111100000
Ciphertext
Round 1=11111010
Round 1=10111100
No of bits that differ after round 1=2
No of bits that differ after round 1=3
Eg with neural
Plaintext = 1 1 1 1 1 1 1 1
Key (10-bit)=1111100000
Ciphertext
Round 1=11111011
Round 1=00011010
No of bits that differ after round 1=1
No of bits that differ after round 1=5

Hamming Distance : In information theory, the hamming distance between two string s of equal length is the number of position for which the corresponding symbols are different .Put another way, it measures the number of substitution required to change one into another ,or the number of errors that transformed one string into other for

Example

- The hamming distance between 10111101 AND 1001001 is 2
- The hamming distance between 2143896 2233796 is 3
- The hamming distance between “toned” AND”roses” is 3

*B.SDES Comparison*

Graphics given below is analysis for the avalanche effects depicted by the SDES algorithm with AND without neural network. Here important point is that for 8 bit plaintext maximum possible plaintext will be 256.But during analysis we have taken 17 different plaintext each time.

**PLOT1** - This plot for the difference of number of bits in plaintext and cipher text after round 1 AND 2 with normal AND neural XOR design.

**PLOT 2**- This stem for the difference of number of bits in plaintext AND cipher text after round 1 AND 2 with normal AND neural XOR design

**PLOT 3**- This is the stem shows the comparison neural and normal SDES for round 1 AND 2

*Using Normal XOR design for round 1*

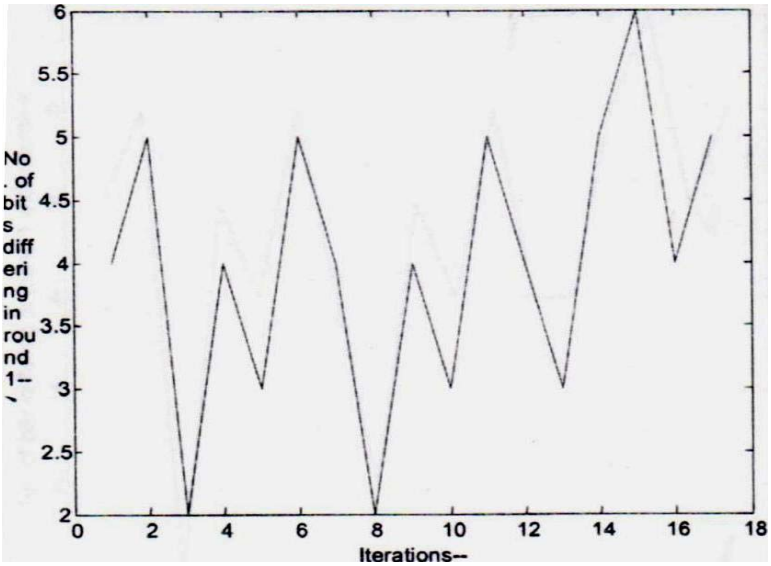


Figure 4.1

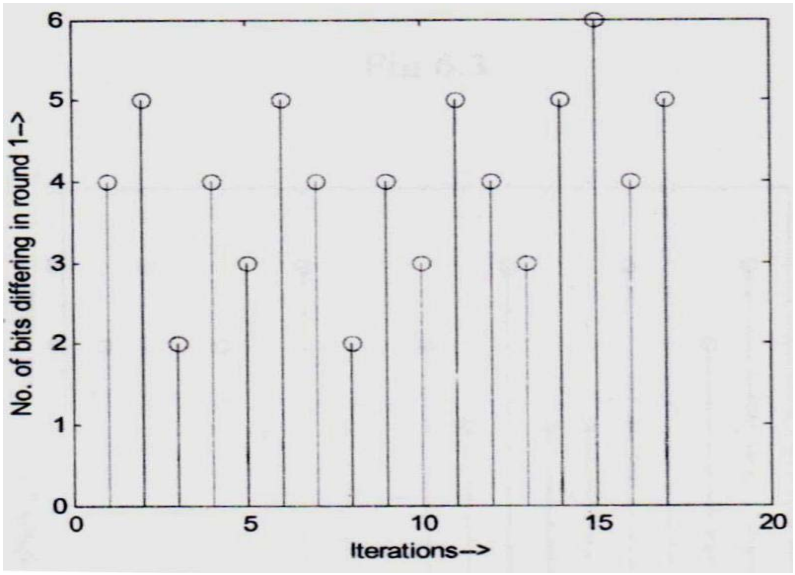


Figure 4.2

Fig 4.1 AND 4.2: Here the plot and stem show the difference in plaintext AND cipher text after first round for normal XOR design

*Using Neural XOR design for round 1*

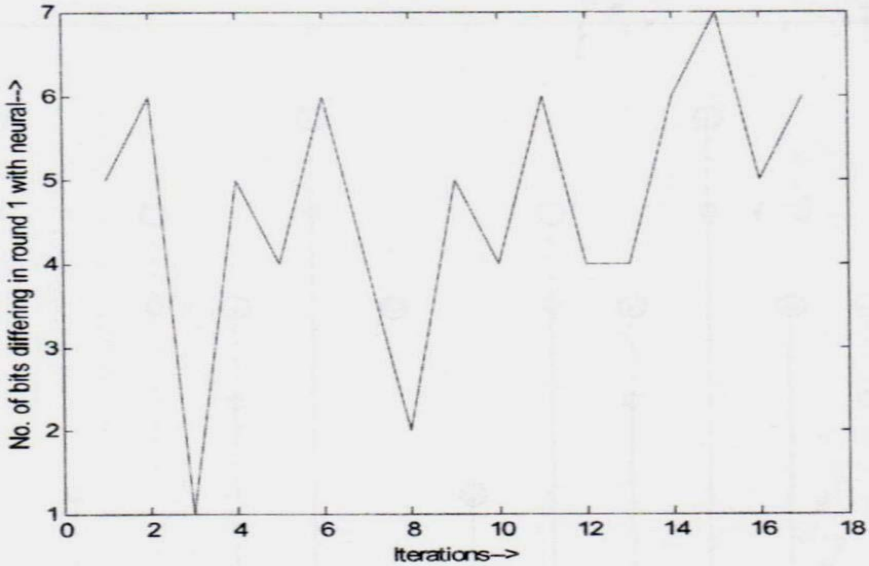


Figure 4.3

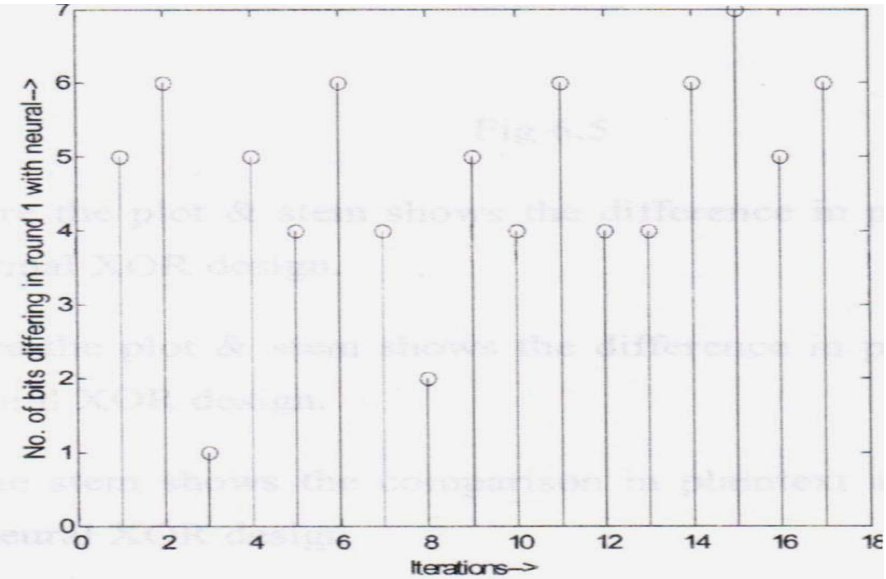


Figure 4.4

Fig 4.3 AND 4.4: Here the plot and stem show the difference in plaintext AND cipher text after first round for neural XOR design

*Comparison between neural & normal XOR design for SDES for round 1*

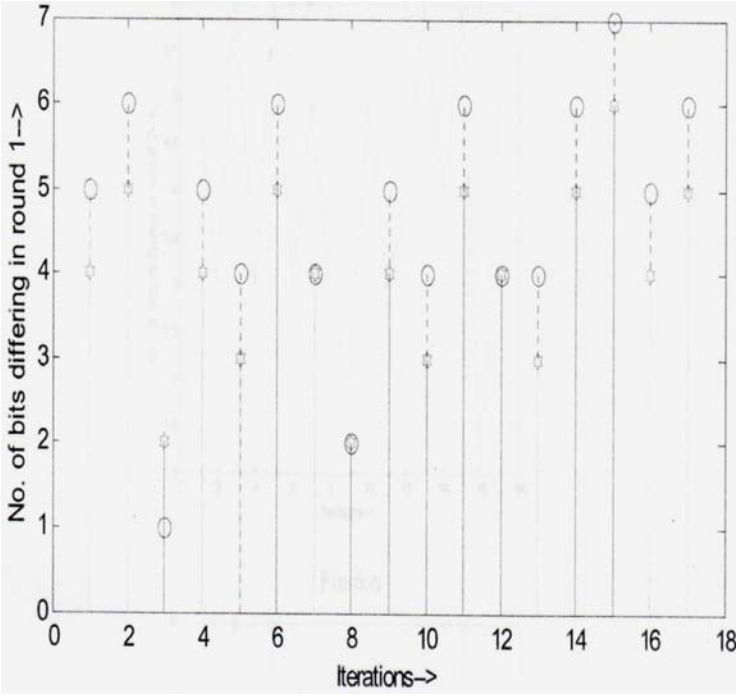


Fig 4.5

Fig 4.5: Here the plot and stem show the difference in plaintext AND cipher text after first round for normal AND neural XOR design

*Using Normal XOR design for round 2*

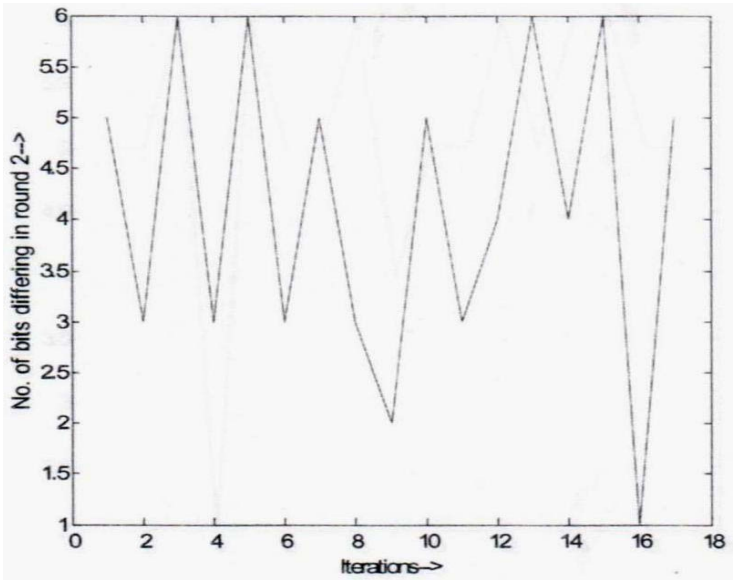


Fig 4.6

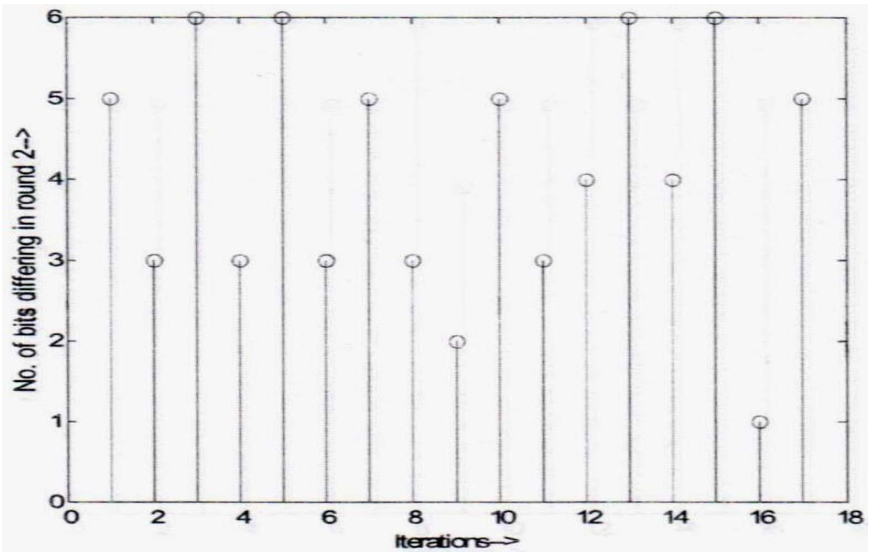


Fig 4.7

*Using neural XOR design for round2*

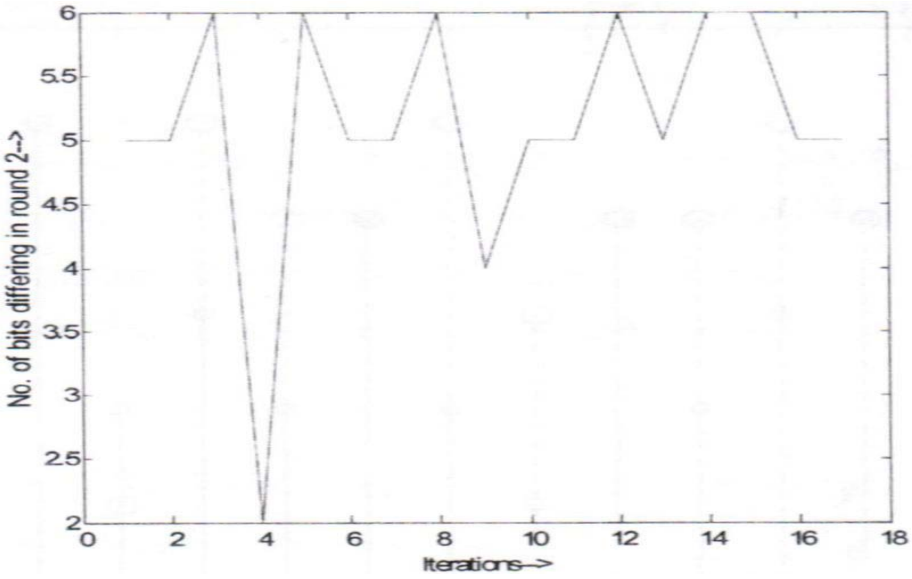


Fig 4.8

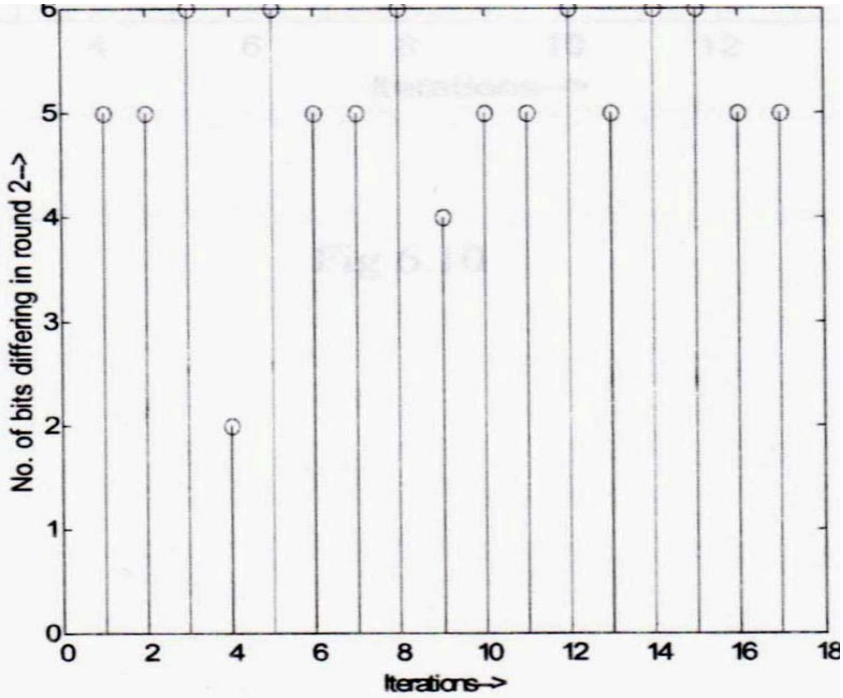


Fig 4.9

*Comparison between neural and Normal OR design for SDES for round*

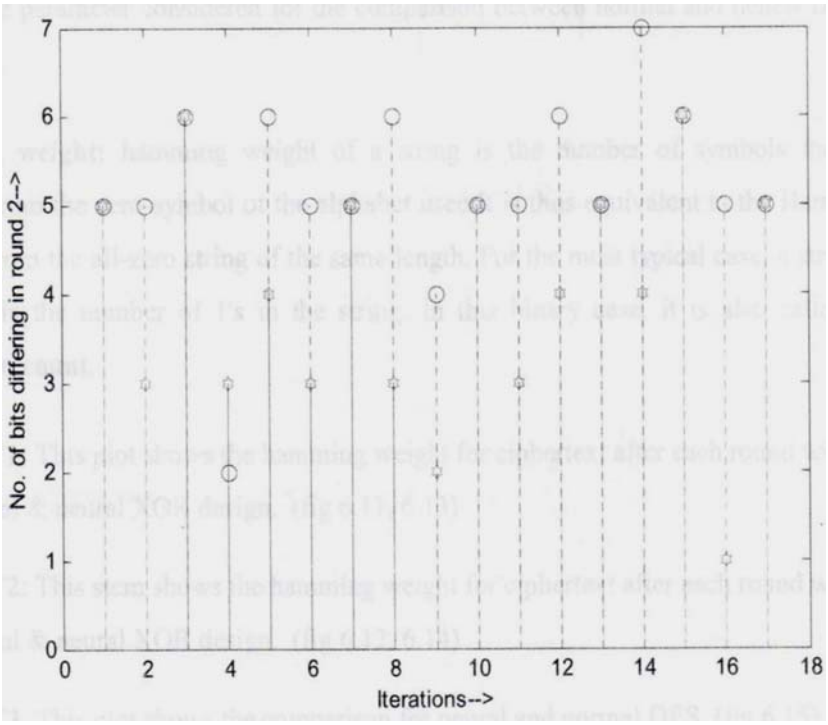


Fig 4.10

*B. DES Comparison*

The parameters considered for the comparison between normal and neural DES is as follows:

Hamming weight hamming weight of a string is the number of symbols that are different from the zero-symbol of the alphabet used. it is thus equivalent to the hamming distance from all-zero string of the same length .for the most typical case, a string of bits, this is the number of 1's in the string. in this binary case, it is also called the population count

**PLOT 1:** this plot shows the hamming weight for ciphertext after each round with normal and neural XOR design . (fig 4.11,4.13)

**PLOT 2:** this stem shows the hamming weight for ciphertext after each round with normal and neural XOR design . (fig 4.12,4.14)

**PLOT 3:** this plot shows the comparison for neural and normal DES, (fig 4.15)

**PLOT 4:** this stem shows the comparison for neural and normal DES,(fig 4.16)

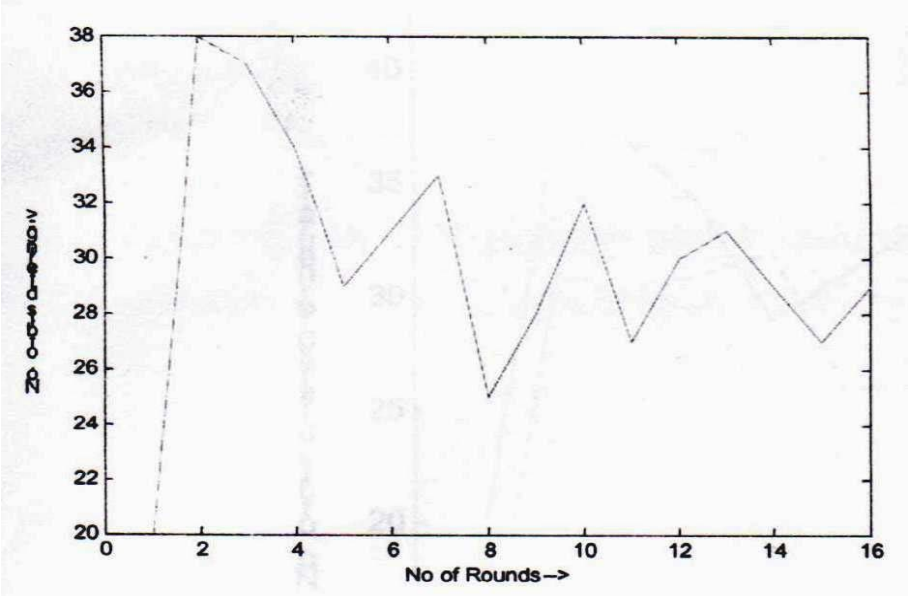


**Fig. 4.11**

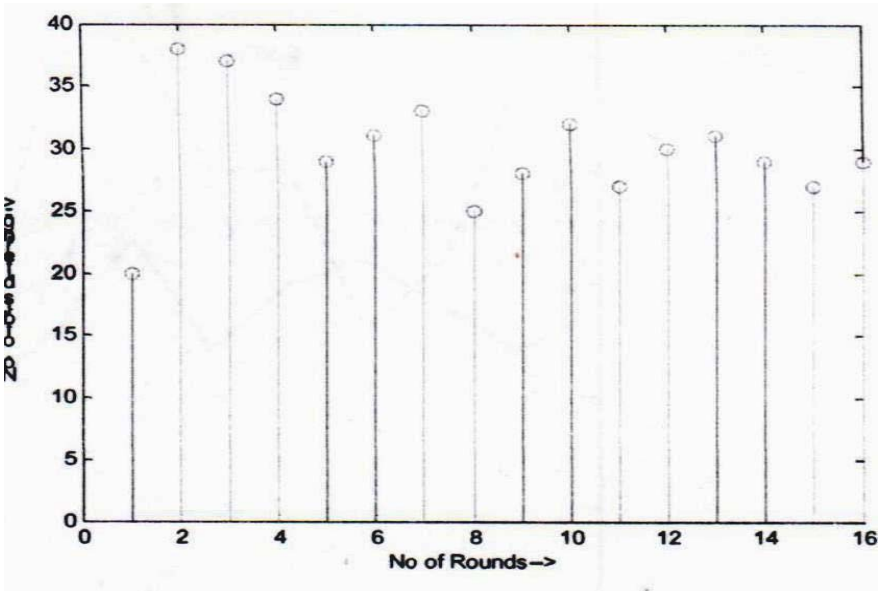


Fig. 4.12

Fig 4.11 &4.12: Here plot & stem shows the hamming weight for cipher text after each round of normal XOR design

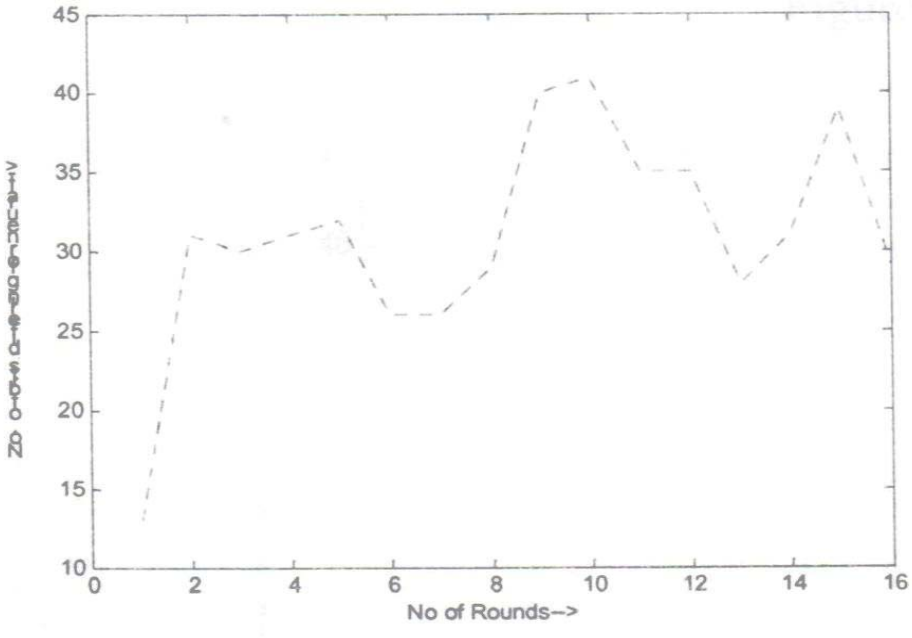


Fig. 4.13

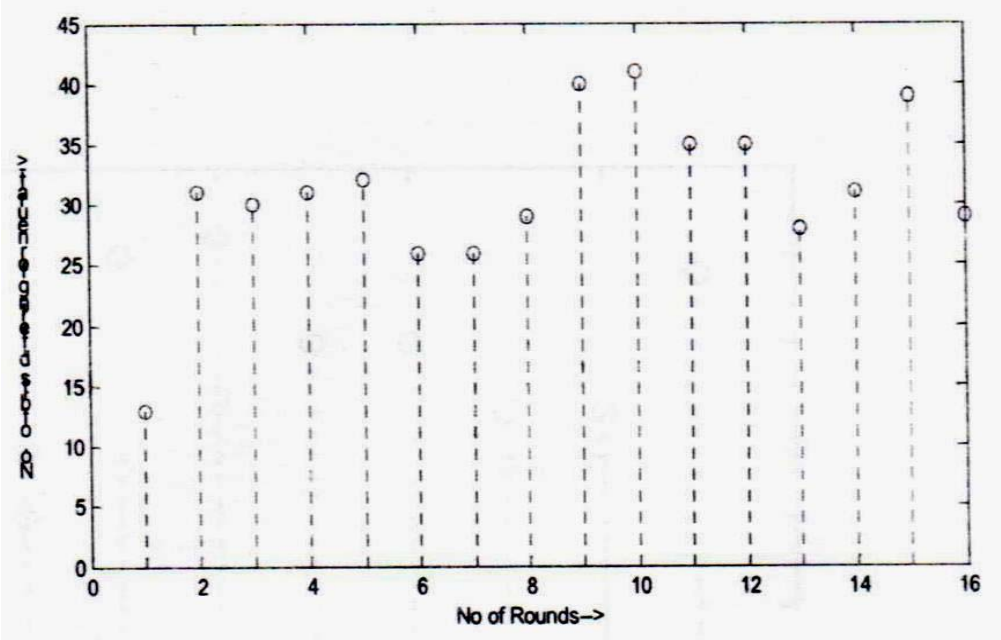


Fig. 4.14

Fig 4.13 & 4.14: Here plot & stem shows the hamming weight for cipher text after each round of DES for neural XOR design.

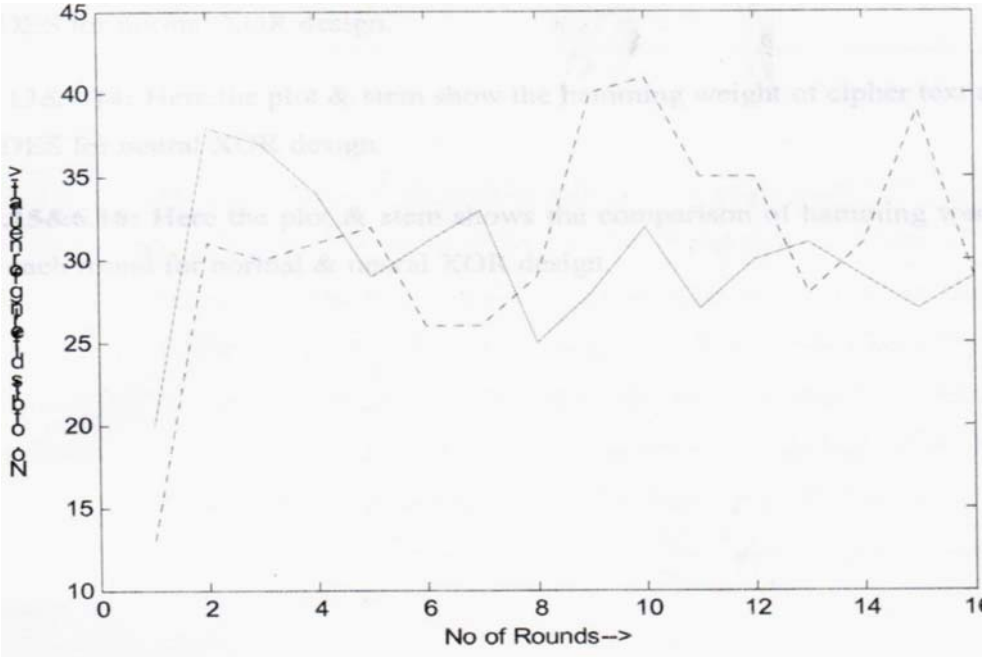


Fig 4.15

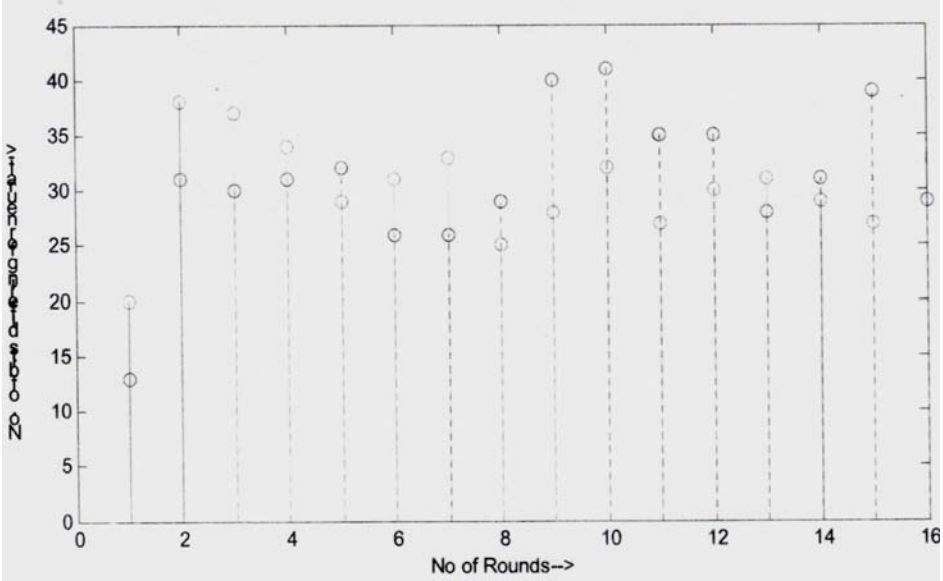


Fig 4.16

Fig 4.15 & 4.16: Here plot & stem shows the comparison of hamming weight of cipher text after each round for normal & neural XOR design.

## V. CONCLUSION AND FUTURE SCOPE

*A Conclusion*

In the project, we presented an idea of application of Neural Networks, a powerful tool for replacing XOR. We have used the back proration algorithm with feed forward networks to make the S-DES algorithm more robust thereby making cryptanalysis more difficult. We have observed the avalanche effect for S-DES algorithm i.e. A small change in either the plaintext or the key should produce a significant change in the cipher text. Using normal and natural XOR design and we have found that avalanche effect is better for natural XOR design as the no of bit change in cipher text has increased which has made our algorithm more robust and to make cryptanalysis difficult.

*B. Future Scope*

- We can use the concept of natural networks to replace the logical operation such as AND, OR, NOT etc in other cryptographic algorithms.
- We can also use this concept for other symmetric an asymmetric cryptographic algorithms.
- We can even extend its use to replace the substitution and permutation modules in the algorithm.

## VI. References

[1] W.Stalling, Cryptography and network security, July 2005[5]

[2]Computational Intelligence – An Introduction – Engelbrecht – cited by 108.

[3] M.M. Gupta and D.H. Rao, Neuro-Control systems, IEEE Press, USA, 1994.

[4] V. Desai, D.H.Rao, "S-Box Design of DES using neural network approach to achieve bent function approximation", IEEE Indicon conference, Banglore, 2007.

[5] T.Godhavari, N.R.Alamelu, "Cryptography Using Neural networks", IEEE Indicon conference, Chennai, India 2005.

[6] Albasil and Wohdan discuss a block cipher based on neural networks. The Fiestal block cipher using neural network is considered.

[7] Handessi, Gulliver and Sheikh, “Large S-Box design Using Converging Method”,1997, Ulm, Germany.

[8] P.Kotlarz and Z.Kotulski,”Applicationof Neural Network for S-Box Design”,Kezimeierz Wielki University, Bydgo, Institute of Fundamental Technology Research of the Polish Academy of Sciences, Institute of telecommunications of WUT, Warsaw.

[9] T.Godhavari and N.R.Alamelu discuss the interaction of neural networks and cryptography to generate secret key over public channel.

[10] M.Capsodi, J.Vandewalle and T.Roska, “High Speed Calculation of Cryptographic Hash functions by CNN chips”, ESAT laboratory, Leuven, Belgium.

[11] M.M. Gupta, and T. Yamakawa, Eds., Fuzzy Computing : Theory, Hardware and Applications. New York, Amsterdam, Oxford: North Holland, 1988 .

[12] S. Guicheng, Z. Xuefeng, L. Bingwu “Object – Oriented Analysis and design of elliptic curve cryptosystem”, IEEE International Conference on industrial informatics 2008, Korea.

[13] T. Kyung, K. Kim, S. Kim “ A study on information security level evaluation using fuzzy AHP”,

[14] S. Sadeg, M. Gougache, H. Drias “ An encryption algorithm inspired from DNA”,

[15] H. Wade , B. Wallace “Is information security under control?”, IEEE Computer Society, 1540 – 7993/07/@2007 IEEE.