A Symmetric Key Cryptography using Genetic Algorithm and Error Back Propagation Neural Network

Vikas Sagar
Department of Computer Science
Gurukul Kangri University
Haridwar, India
vikassagar52@gmail.com

Krishan Kumar
Department of Computer Science
Gurukul Kangri University
Haridwar, India
Kumar_krishana@yahoo.com

Abstract – In conventional security mechanism, cryptography is a process of information and data hiding from unauthorized access. It offers the unique possibility of certifiably secure data transmission among users at different remote locations. Cryptography is used to achieve availability, privacy and integrity over different networks. Usually, there are two categories of cryptography i.e. symmetric and asymmetric. In this paper, we have proposed a new symmetric key algorithm based on genetic algorithm (GA) and error back propagation neural network (EBP-NN). Genetic algorithm has been used for encryption and neural network has been used for decryption process. Consequently, this paper proposes an easy cryptographic secure algorithm for communication over the public computer networks.

Keywords – cryptography; error back propagation neural network; genetic algorithm; symmetric key.

I. INTRODUCTION

In the present time privacy and proprietary information during transmission is a tremendous task over the public network. Maintaining confidentiality is becoming increasingly important factor that requires urgent attention in order to maintain the secrecy. There are a number of security mechanisms and applications already exist.

Cryptography used from ancient time for secure communications. In traditional approaches of cryptographic algorithm mathematical functions are used for encryption and decryption process.

The basic goals of cryptography are given as:

Confidentiality: Confidentially is to keep information secret from unauthorized entity. When the data is exchanged between two users then the attackers must not be able to detect about the source and destination.

Data integrity: Data should not be manipulated by unauthorized entities. Data manipulations involves effects such as insertion, deletion, and data modification. The recipient should be able to determine if the message has been altered or not.

Authentication: Authentication is a service related to identification. The recipient should be able to verify the identity of the sender, the origin from the message.

Non-repudiation: Non-repudiation prevents either sender or receiver from denying a message. It means that, when receiver received a message, the sender can prove the alleged receiver in received that message. On the other hand, when a sender sends a message to the receiver, the receiver can prove that the message is sent by the authorized sender [1], [5].

II. OVERVIEW OF SYMMETRIC KEY CRYPTOGRAPHY

As the name indicates, a single key is used both for encryption and decryption in symmetric key cryptography [1]. The symmetric key cryptography scheme has following five components in all:

- **Plaintext**: This is the original information data that is taking as an input in the algorithm.
- **Encryption Algorithm:** The encryption algorithm takes the plaintext and performs various substitutions and permutations according to the encryption techniques to encrypt the data.
- **Secret Key:** The encryption and the decryption algorithm used the secret key in the encryption algorithm. The different keys used in substitutions and permutations at every time, and the algorithm will produce a different output depending on the specific key being used at the time.
- **Cipher text:** This is the encrypted message generated by the encryption algorithm. Every time the complexity of cipher text is different and depends on the plaintext and the key.
- **Decryption Algorithm:** Decryption algorithm takes the cipher text and the secret key and generate the original plaintext. This is just reverse process of the encryption algorithm [1], [2], [4].

The symmetric key encryption and decryption process shown in figure-1:

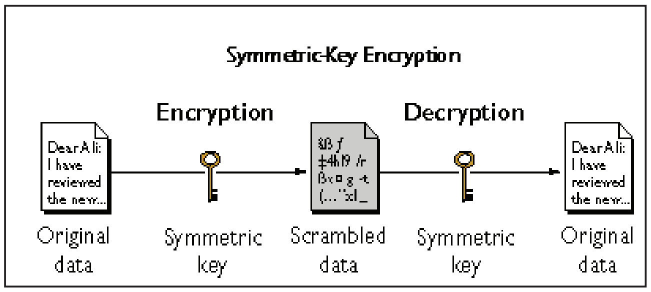


Fig. 1. Symmetric key encryption process [5]

III. GENETIC ALGORITHM

Genetic algorithm (GA) was first used by John Holland Adaptation in Natural and Artificial Systems of 1975. Genetic Algorithms are a family of computational models inspired by evolution. Genetic algorithms are often viewed as function optimizers although the range of problems to which genetic algorithms have been applied is quite broad.

Genetic Algorithm is a highly optimized procedure based on mechanics of natural selection and natural genetics. Genetic is an adaptive heuristic search algorithm for the extreme multivariable functions. A procedural implementation of a genetic algorithm shown in figure 2:

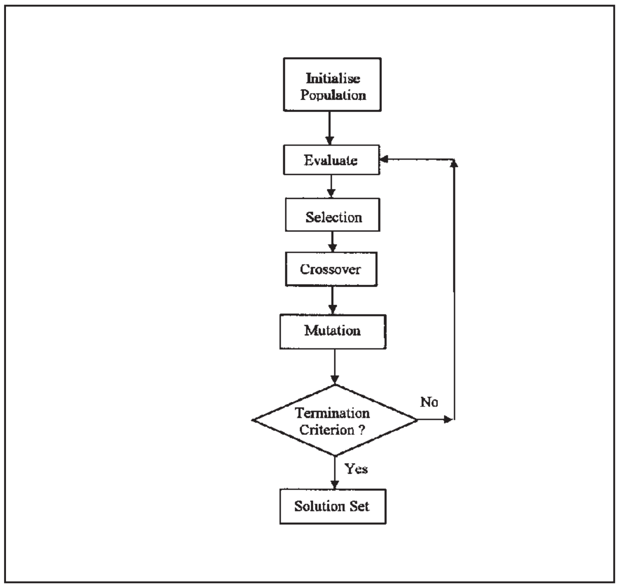


Fig. 2. Basic Architecture of Genetic Algorithm

Initialization: The initial population of is generated randomly across the search space or defined by the user.

Evaluation: After the initialization of population the fitness values of the candidate solutions are evaluated.

Selection: After evaluation, the fittest chromosomes have higher probability to be selected for the next generation. To compute fitness probability we must compute the fitness of each chromosome.

Crossover: Crossover combines two or more parts of parental solutions to create new, possibly better solutions. Crossover can be done by different methods. Randomly select a position (single or multiple positions it's depend on the crossover method) in the parent chromosome then exchanging sub-chromosome.

Mutation: Mutation randomly modifies a solution after crossover operates on two or more parental chromosomes. There are different types of method of mutation. Mutation process is done by replacing the gen at random position with a new value [10], [12].

We use GA with uniform crossover and single point mutation for the encryption in proposed cryptography algorithm.

Uniform Crossover: The Uniform Crossover takes the uniform fixed ratio between two parents. The uniform crossover makes the data complicated in this algorithm. We use this genetic method to increase the complexity of the encryption side by crossover the binary form of plaintext. The uniform crossover shown in figure 3:

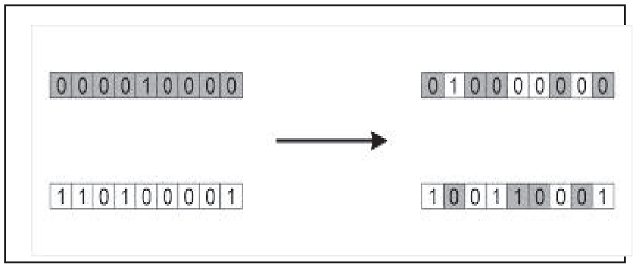


Fig.3. Uniform Crossover

Single point mutation: Point mutation is a random mutation in the deoxyribonucleic acid (DNA) that occurs at one point. Point mutation happens during DNA replication so that there is a change within a gene (binary numbers) in which one base pair in the DNA sequence is altered. We use this single point mutation over the data coming from crossover section [11], [12].

IV. ARTIFICAL NEURAL NETWORK

Artificial neural networks were introduced by Mc Culloch Pitts in 1943. ANN is inspired from biological neurons. A biologically neural network basically consists of tiny computing units, called neurons, A neuron consists of a soma (cell body), axons (sends signals), and dendrites (receives signals). Neurons interconnected with each other with some nerves connections called synapses. An Artificial Neuron is basically an engineering approach of biological neuron try to mimic the behavior of biological neurons and their interconnections. It has many inputs and one output. ANN is consisting of large number of simple processing elements that are interconnected with each other and layered also. These networks are non-linear, adaptive, simple, and robust in nature.

As we know, an artificial neural network (ANN) has been inspired by biological nervous systems, such as the brain. A neuron is a biological cell that transfers information in the form of electrical and chemical change from one neuron to another neuron. It is composed two types of out reaching tree like branches: the axon and the dendrites. In the cell body or soma it has a nucleus that store information that needed by the neurons [8].

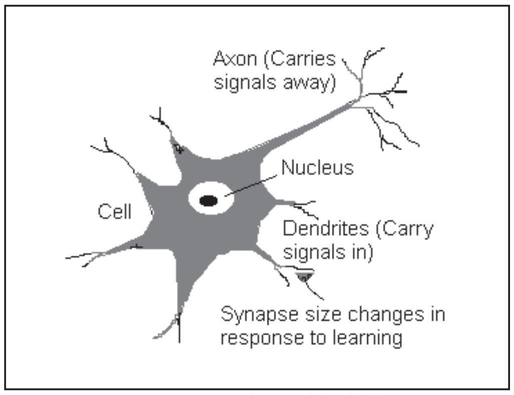

Axon (Carries signals away)
Nucleus
Cell
Dendrites (Carry signals in)
Synapse size changes in response to learning

Fig. 4. Biological Neuron

Artificial Neural Networks fully inspired this structure of neuron. ANN consists of units, connections and weights. The basic architecture of ANN is:

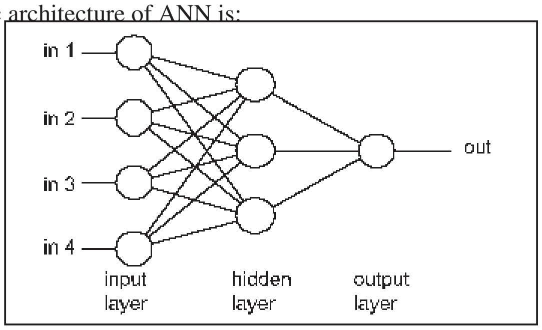


Fig. 5. Artificial Neural Network

The Comparison symmetric between the biological neuron and artificial neuron network:

Biologic NN	**Artificial NN**
Soma	Unit
axon, dendrite	Connection
synapse	weight
potential	weighted sum
threshold	bias weight
signal	activation

Normally an artificial neural network computes the things parallel which makes it powerful. It is also known as black box neural networks because we are only intended with the input and output, and we are least bothered about the hidden layers. All artificial neural networks can be divided in supervised and unsupervised learning [3].

The model work over the input and produce the output that is predefined known as supervised learning. In contrast with the supervised learning, the learning doesn't have the known output known as unsupervised learning. So that its work only on input by itself.

These networks are being widely used in different applications areas together with other technologies for solving artificial intelligent problems, such as functions approximation, modeling, brain mapping, air traffic control, stock exchange, control systems, financial modeling, data compression, classification tasks or data processing.

We have used the Error back propagation neural network for new proposed algorithm [8], [9].

ANN plays a very important role in cryptography now-a-days. Neural cryptography is a fine method to generate secret information over a public channel. Many papers and works is dedicated in the last few years towards neural cryptography, and a magnificent starting point found for this literature.

Error back propagation neural network (EBP-NN): One of the most commonly used supervised artificial neural network is Error back propagation neural network. This model is easy to understand, and can be easily implemented as a software simulation [14].

Algorithm

1. First we apply the inputs to the network and the initial weights that taken have been random numbers.

2. Next, the input pattern is applied and the output calculated (it's called *forward pass*). If the output is completely different to the target value, then calculate the *Error* of each neuron. The error is, in other words:

$\text{Error}_B = \text{Output}_B\,(1\text{-Output}_B)\,(\text{Target}_B - \text{Output}_B)$

The "*Output (1-Output)*" term is Sigmoid Function – if we were only using a threshold neuron it would just be *(Target – Output).*

3. Now change the weight. Let W^+_{AB} be the new (trained) weight and WAB be the initial weight. $W^+_{AB} = W_{AB}$ + (Error x Output). Update all the weights in the output layer in this way.

4. Calculate the Errors for the hidden layer neurons. We can't calculate these directly, so we *Back Propagate* them from the output layer. For this calculate the Errors from the output neurons and running them back through the weights to get the hidden layer errors. For example if neuron A is connected as shown to B and C then we take the errors from B and C to generate an error for A.

$\text{Error}_A = \text{Output}_A\,(1 - \text{Output}_A)(\text{Error}_B\, W_{AB} + \text{Error}_C\, W_{AC})$

Again, the factor "*Output (1 – Output)*" is present because of the sigmoid squashing function.

5. After calculating the Error for the hidden layer neurons now proceed as in next stage to change the weights in hidden layer. By repeating this method we can train a network of any number of layers [7], [14].

The working can be easily shown in the following calculations and figure:-

1. Calculate errors of output neurons

$$\delta_\alpha = out_\alpha (1 - out_\alpha) (Target_\alpha - out_\alpha)$$
$$\delta_\beta = out_\beta (1 - out_\beta) (Target_\beta - out_\beta)$$

2. Change output layer weights

$$W^+_{A\alpha} = W_{A\alpha} + \eta\delta_\alpha\, out_A,\ W^+_{A\beta} = W_{A\beta} + \eta\delta_\beta\, out_A$$
$$W^+_{B\alpha} = W_{B\alpha} + \eta\delta_\alpha\, out_B,\ W^+_{B\beta} = W_{B\beta} + \eta\delta_\beta\, out_B$$
$$W^+_{C\alpha} = W_{C\alpha} + \eta\delta_\alpha\, out_C,\ W^+_{C\beta} = W_{C\beta} + \eta\delta_\beta\, out_C$$

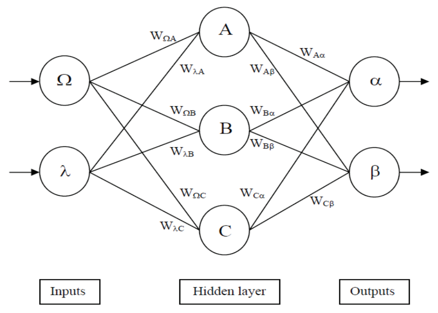


Fig.6. Architecture of EBP-NN

3. Calculate (back-propagate) hidden layer errors

$$\delta_A = out_A (1 - out_A) (\delta_\alpha W_{A\alpha} + \delta_\beta W_{A\beta})$$
$$\delta_B = out_B (1 - out_B) (\delta_\alpha W_{B\alpha} + \delta_\beta W_{B\beta})$$
$$\delta_C = out_C (1 - out_C) (\delta_\alpha W_{C\alpha} + \delta_\beta W_{C\beta})$$

4. Change hidden layer weights

$$W^+_{\lambda A} = W_{\lambda A} + \eta\delta_A\, in_\lambda,\ W^+_{\Omega A} = W^+_{\Omega A} + \eta\delta_A\, in_\Omega$$
$$W^+_{\lambda B} = W_{\lambda B} + \eta\delta_B\, in_\lambda,\ W^+_{\Omega B} = W^+_{\Omega B} + \eta\delta_B\, in_\Omega$$
$$W^+_{\lambda C} = W_{\lambda C} + \eta\delta_C\, in_\lambda,\ W^+_{\Omega C} = W^+_{\Omega C} + \eta\delta_C\, in_\Omega$$

The constant η (learning rate, nominally equal to one) is put in to speed up or slow down the learning if required [10].

V. NEW SYMMETRIC KEY CRYPTOGRAPHY

Here we proposed a new cryptographic algorithm with GA and ANN methods.

Encryption:

1. Get input data from the sender in ie plaintext.
2. Convert plain text in ASCII format.
3. Convert the ASCII value in to binary value (use as Target value for the Error back propagation neural network).
4. Take the binary string and divide in to equal size and crossover both string.
5. Now use mutation over the string get from crossover section. This string use as cipher text and send this to the receiver over the network.

Decryption:-

1. Receiver receives the cipher text sending by the sender.
2. Send that cipher text as input in to error back propagation neural network and set the target value.
3. When the output data of the error back propagation neural network is same as target value then take the output.
4. Convert that output (binary string) in to ASCII
5. And last convert ASCII to corresponding to plain text.

Example of the Algorithm:
Encryption:

STEP 1:- Very first we take a plain text.

"This is the very important data"

STEP 2:- Convert this plain text in to ASCII format.

084 104 105 115 032 105 115 032 116 104 101 032 118 101 114 121 032 105 109 112 111 114 116 097 110 116 032 100 097 116 097 013 010

STEP 3:-Convert ASCII format in to binary strings.

```
00110000  00111000  00110100  00100000  00110001  00110000  00110100  00100000
00110001  00110000  00110101  00100000  00110001  00110001  00110101  00100000
00110000  00110011  00110010  00100000  00110001  00110000  00110101  00100000
00110001  00110001  00110101  00100000  00110000  00110011  00110010  00100000
00110001  00110001  00110110  00100000  00110001  00110000  00110100  00100000
00110001  00110000  00110001  00100000  00110000  00110011  00110010  00100000
00110001  00110001  00111000  00100000  00110001  00110000  00110001  00100000
00110001  00110001  00110100  00100000  00110001  00110010  00110001  00100000
00110000  00110011  00110010  00100000  00110001  00110000  00110101  00100000
00110001  00110000  00111001  00100000  00110001  00110001  00110010  00100000
00110001  00110001  00110001  00100000  00110001  00110001  00110100  00100000
00110001  00110001  00110110  00100000  00110000  00111001  00110111  00100000
00110001  00110001  00110000  00100000  00110001  00110001  00110110  00100000
00110000  00110011  00110010  00100000  00110001  00110000  00110000  00100000
00110000  00111001  00110111  00100000  00110001  00110001  00110110  00100000
00110000  00111001  00110111  00100000  00110000  00110001  00110011  00100000
00110000  00110001  00110000
```

STEP 4:- Divide the binary string in to 2 equal size strings.

String1-

```
00110000  00111000  00110100  00100000  00110001  00110000  00110100  00100000
00110001  00110000  00110101  00100000  00110001  00110001  00110101  00100000
00110000  00110011  00110010  00100000  00110001  00110000  00110101  00100000
00110001  00110001  00110101  00100000  00110000  00110011  00110010  00100000
00110001  00110001  00110110  00100000  00110001  00110000  00110100  00100000
00110001  00110000  00110001  00100000  00110000  00110011  00110010  00100000
00110001  00110001  00111000  00100000  00110001  00110000  00110001  00100000
00110001  00110001  00110100  00100000  00110001  00110010  00110001  00100000
00110000  0011
```

String2

```
00110011  00100010  00000011  00010011  00000011  01010010  00000011  00010011
00000011  11001001  00000001  10000011  00010011  00100010  00000000  00110001
00110001  00110001  00110001  00100000  00110001  00110001  00110100  00100000
00110001  00110001  00110110  00100000  00110000  00111001  00110111  00100000
00110001  00110001  00110000  00100000  00110001  00110001  00110110  00100000
00110000  00110011  00110010  00100000  00110001  00110000  00110000  00100000
00110000  00111001  00110111  00100000  00110001  00110001  00110110  00100000
00110000  00111001  00110111  00100000  00110000  00110011  00100011  00000011
00010011  0000
```

STEP 5:- Applying uniform Crossover upon both strings.

```
00110001   00110000   00110100   00100000   00110001   00110001   00110110   00100000
00110000   00110011   00110010   00100000   00110001   00110000   00110001   00100000
00110001   00110000   00110101   00100000   00110001   00110001   00111000   00100000
00110001   00110000   00110100   00100000   00110001   00110001   00110100   00100000
00110001   00110010   00110001   00100000   00110000   00111000   00110100   00100000
00110001   00110000   00110101   00100000   00110001   00110001   00110101   00100000
00110001   00110000   00110001   00100000   00110000   00110011   00110010   00100000
00110001   00110001   00110101   00100000   00110000   00110011   00110010   00100000
00110000   00110000   00110001   00110000   00110000   00100000   00100011   00000011
00110001   00110001   00110001   00100000   00110001   00110001   00110100   00100000
00110000   00111001   00110111   00100000   00110001   00110001   00110110   00100000
00110000   00111001   00110111   00110000   00110011   00110010   00100000   00010011
00110001   00110001   00110110   00100000   00110000   00111001   00110111   00100000
00110001   00110001   00110110   00100000   00000011   01010010   00000011   00010011
00110001   00110001   00110000   00100000   00000011   11001001   00000001   10000011
00010011   00100010   00000000   00110001   00110011   00100010   00000011   00010011
00100000   00110000   00110011
```

STEP 6:- Applying mutation over the string get by crossover.

```
00110001   00110000   00110100   00100000   00110001   00110001   00110110   00100000
00110000   00110011   00110010   00100000   00110001   00110000   00110001   00100000
00110001   00110000   00110101   00100000   00110001   00110001   00111000   00100000
00110001   00110000   00110100   00100000   00110001   00110001   00110100   00100000
00110001   00110010   00110001   00100000   00110000   00111000   00110100   00100000
00110001   00110000   00110101   00100000   00110001   00110001   00110101   00100000
00110001   00110000   00110001   00100000   00110000   00110011   00110010   00100000
00110001   00110001   00110101   00100000   00110000   00110011   00110010   00100000
00110000   00110000   00110001   00110000   00110000   00100000   00100011   00000011
00110001   00110001   00110001   00100000   00110001   00110001   00110100   00100000
00110000   00111001   00110111   00100000   00110001   00110001   00110110   00100000
00110000   00111001   00110111   00110000   00110011   00110010   00100000   00010011
00110001   00110001   00110110   00100000   00110000   00111001   00110111   00100000
00110001   00110001   00110110   00100000   00000011   01010010   00000011   00010011
00110001   00110001   00110000   00100000   00000011   11001001   00000001   10000011
00010011   00100010   00000000   00110001   00110011   00100010   00000011   00010011
00100000   00110000   00110011
```

Decryption:
STEP 1:- Receiver receives the cipher text sending by the sender.

```
00110001   00110000   00110100   00100000   00110001   00110001   00110110   00100000
00110000   00110011   00110010   00100000   00110001   00110000   00110001   00100000
00110001   00110000   00110101   00100000   00110001   00110001   00111000   00100000
00110001   00110000   00110100   00100000   00110001   00110001   00110100   00100000
00110001   00110010   00110001   00100000   00110000   00111000   00110100   00100000
00110001   00110000   00110101   00100000   00110001   00110001   00110101   00100000
00110001   00110000   00110001   00100000   00110000   00110011   00110010   00100000
00110001   00110001   00110101   00100000   00110000   00110011   00110010   00100000
00110000   00110000   00110001   00110000   00110000   00100000   00100011   00000011
00110001   00110001   00110001   00100000   00110001   00110001   00110100   00100000
00110000   00111001   00110111   00100000   00110001   00110001   00110110   00100000
00110000   00111001   00110111   00110000   00110011   00110010   00100000   00010011
00110001   00110001   00110110   00100000   00110000   00111001   00110111   00100000
00110001   00110001   00110110   00100000   00000011   01010010   00000011   00010011
00110001   00110001   00110000   00100000   00000011   11001001   00000001   10000011
00010011   00100010   00000000   00110001   00110011   00100010   00000011   00010011
00100000   00110000   00110011
```

STEP 2:- Send that cipher text as input in to Error Back Propagation Neural Network and set the target value and this EBP-NN run until the target value achieved.

```
00110000   00111000   00110100   00100000   00110001   00110000   00110100   00100000
00110001   00110000   00110101   00100000   00110001   00110001   00110101   00100000
00110000   00110011   00110010   00100000   00110001   00110000   00110101   00100000
00110001   00110001   00110101   00100000   00110000   00110011   00110010   00100000
00110001   00110001   00110110   00100000   00110001   00110000   00110100   00100000
00110001   00110000   00110001   00100000   00110000   00110011   00110010   00100000
00110001   00110001   00111000   00100000   00110001   00110000   00110001   00100000
00110001   00110001   00110100   00100000   00110001   00110010   00110001   00100000
00110000   00110011   00110010   00100000   00110001   00110000   00110101   00100000
00110001   00110000   00111001   00100000   00110001   00110001   00110010   00100000
00110001   00110001   00110001   00100000   00110001   00110001   00110100   00100000
00110001   00110001   00110110   00100000   00110000   00111001   00110111   00100000
00110001   00110001   00110000   00100000   00110001   00110001   00110110   00100000
00110000   00110011   00110010   00100000   00110001   00110000   00110000   00100000
00110000   00111001   00110111   00100000   00110001   00110001   00110110   00100000
00110000   00111001   00110111   00100000   00110000   00110001   00110011   00100000
00110000   00110001   00110000
```

STEP 3:- Convert that output (binary string) in to ASCII.

084 104 105 115 032 105 115 032 116 104 101 032 118 101 114 121 032 105 109 112 111 114 116 097 110 116 032 100 097 116 097 013 010

STEP 4:- convert ASCII to corresponding to plain text.

"This is the very important data"

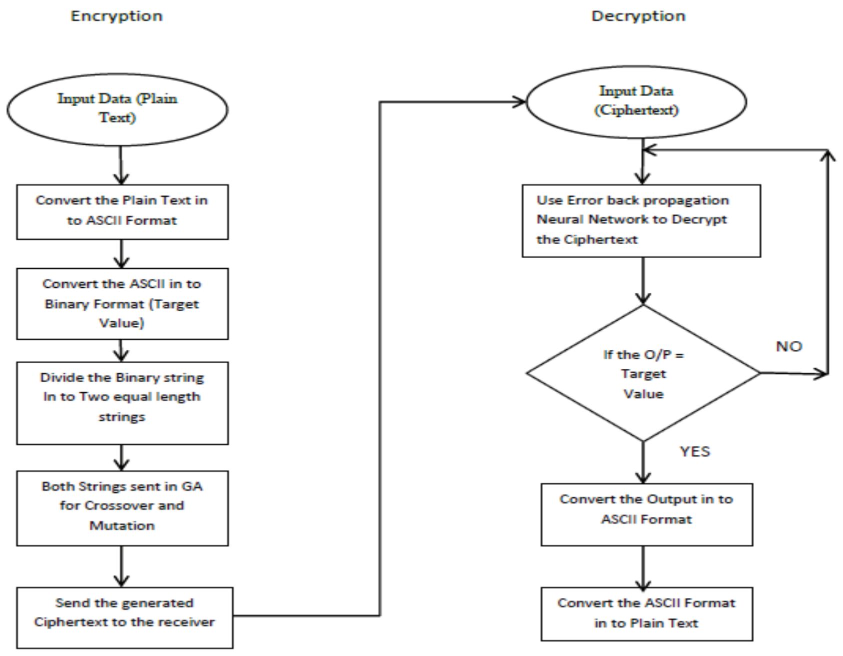
and it's the final plain text that is send by the sender.

VI. FLOWCHART OF PROPOSED ALGORITHM

(It is shown below after Conclusion)

VII. CONCLUSION

The prime objective of this research paper is to generate a secure symmetric key cryptographic algorithm using genetic algorithm and error back propagation neural networks, and hence communicate the information over the wired or wireless network medium with the effective and efficient way which fits to the required objective of sending and receiving the data securely. Further, low or high bandwidth over the network does not affect the performance of this algorithm. Consequently we can say that our hybrid approach to generate the key is easy to use, fast and simple. Hence the time complexity of this algorithm will be very less in comparison to the other traditional algorithms like DES, AES, & RSA.


Encryption
Decryption
Input Data (Plain Text)
Convert the Plain Text in to ASCII Format
Convert the ASCII in to Binary Format (Target Value)
Divide the Binary string In to Two equal length strings
Both Strings sent in GA for Crossover and Mutation
Send the generated Ciphertext to the receiver
Input Data (Ciphertext)
Use Error back propagation Neural Network to Decrypt the Ciphertext
If the O/P = Target Value
NO
YES
Convert the Output in to ASCII Format
Convert the ASCII Format in to Plain Text

REFERENCES

[1] Menezes, P. van Oorschot and S. Vanstone, "Handbook of Applied Cryptography". CRC Press, First ed., 1997J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] Eva Volna, Martin Kotyrba, Vaclav Kocian and Michal Janosek "Cryptography based on neural network" Proceedings 26th European Conference on Modelling and Simulation ©ECMS Klaus G. Troitzsch, Michael Möhring, Ulf Lotzmann (Editors) ISBN: 978-0-9564944-4-3 / ISBN: 978-0-9564944-5-0, ecms2012.

[3] Vikas Gujral, "Cryptography using Artifical neural network", Engineering National Institute of Technology Rourkela-769008 Orissa, Session 2005-2009.

[4] Adel A. El-Zoghabi, Amr H. Yassin, Hany H. Hussien, " Survey Report on Cryptography based on Neural Network" International Journal of Emerging Technology and Advanced Engineering, (ISSN 2250-2459), Vol. 3, Issue 12, December 2013.

[5] Vikas sagar, Krishan Kumar "A symmetric key cryptography using counter propagation neural network " International Conference on Information and Communication Technology for Competitive Strategies" ACM-ICPS Proceedings Volume ISBN No 978-1-4503-3216-3.

[6] O.S. Eluyode1 and Dipo Theophilus Akomolafe2 "Comparative study of biological and artificial neural networks" European journalof applied Engineering and scientific research, 2013,2,1. ISSN: 2278-0041.

[7] Khalil Shihab "A cryptographic scheam based on neural network" 10th WSEAS International Conference on COMMUNICATIONS, Vouliagmeni, Athens, Greece, July 10-12, 2006.

[8] Laurene Fausett(1994), "Fundamentals of Neural Networks , Architecutre, Algorithms and Applications', published by arrangement with Pearson Education, Inc. and Dorling Kindersley Publishing Inc.

[9] Zurada, Jacek M. (1999), 'Introduction to artificial neural systems', published by West Publishing Company, printed in United States of America.

[10] Man, K.F. "Genetic algorithms: concepts and applications" Industrial Electronics, IEEE Transactions on (Volume:43 Issue: 5), ISSN 0278-0046

[11] Mehrdad Dianati, Insop Song, and Mark Treiber, "An Introduction to Genetic Algorithms and Evolution Strategies" University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada.

[12] Mitchell Melanie, "An Introduction to Genetic Algorithms" published by A Bradford Book The MIT Press, Cambridge, Massachusetts, London, England Fifth printing, 1999.

[13] Laurene Fausett(1994), 'Fundamentals of Neural Networks , Architecutre, Algorithms and Applications', published by arrangement with Pearson Education, Inc. and Dorling Kindersley Publishing Inc.

[14] Khalil Shihab " A Back Propagation Neural Network For computer Network Security", Journal of Computer Science 2 (9): 710-715, 2006 ISSN 1549-3636.