IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. XII (Mar-Apr. 2014), PP 134-142
www.iosrjournals.org

Secure Barcode Authentication using Genetic Algorithm

[1]Dr. Poornima G. Naik, [2]Mr. Girish R. Naik
[1]Assistant Professor Department of Computer Studies Chh Shahu Institute of Business Education and Research, Kolhapur, India
[2]Associate Professor Production Department KIT's College of Engineering Kolhapur,India

Abstract: *Genetic Algorithm (GA) is an invaluable tool for solving optimization problems due to its robustness. It does not break even in the presence of a reasonable noise or even if the inputs are changed slightly. GA offers significant benefits over other optimization techniques in searching a large state space or n-dimensional surface. In todays information age information transfer and sharing has increased exponentially. With the popularization of Internet and exponential increase in e-commerce transactions security has become an inevitable and an integral part of any e-commerce application. Data integrity, confidentiality, authenticity, non-repudiation have gained tremendous importance and have become important components of information security. In this paper we have made an attempt to exploit the randomness involved in crossover and mutation processes of GA for generating a barcode for authentication process. The number of crossover points and number of mutation points is fixed and cannot be altered by the user. In the current work we have employed a single crossover point and two mutation points. We have used Code-39 and Code-128 encoding techniques for generating a barcode. The barcode data comprises of 12 randomly generated decimal digits. Each decimal digit is represented using 4 bits. Hence the length of the barcode data is 36 bits. The randomly generated data is transformed into encoded form by applying crossover, mutation and XOR operations before generating a bar code. The randomness together with encoding makes the password robust and hard to track. Finally, the algorithm is implemented in Java and applied for authentication of employee data in a hypothetical organization. The methodology is general and can be applied to any task where authentication is required.*
Keywords: *Genetic Algorithm, Cross-over, Mutation, Barcode, Encoding.*

I. Introduction

Genetic algorithms (GA) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetics [1]. They are based on the principle of natural genetics and Darwinian idea of survival of the fittest.

A. Genetic Algorithm

Generally, a Genetic Algorithm consists of three basic operations.

- Selection
- Crossover
- Mutation

The first step consists of searching individuals for reproduction. In the current work we have generated a genetic pool consisting of 50 twelve digit numbers representing the chromosomes which are randomly generated and from which a single random number with the highest fitness value as dictated by the fitness function is selected. The random number thus selected is divided into two parts and encoded using cross-over and mutation operations before generating a barcode using code-39 encoding technique.

Cross-over is the process of taking two parents and producing from them a child. In an optimization problem, crossover operator is applied to the mating pool with the hope that it creates a better offspring. For the problem under consideration, crossover is taken as one of the steps in producing a decrypted vector. We have employed four-point crossover method. In the case of optimization problem, selecting more than four crossover points will result in the disruption of building blocks whereas in the case of encryption larger the disruption better is the algorithm which makes it robust and difficult to break.

After crossover, the vectors are subject to mutation. In optimization problem, mutation prevents the algorithm from being trapped in a local minimum. Mutation plays an important role in the recovery of the lost genetic matter as well for randomly distributing the genetic information. In encryption problem, mutation is employed for inducing disorder into the vector. It introduces a new genetic structure in the population by randomly modifying some of the building blocks and maintains diversity into the population. We have employed flipping method, in which for a character 1 in mutation chromosome, the corresponding character b in the parent chromosome is flipped from b to (9-b) and corresponding child chromosome is produced. In the following example 1 occurs at two random places of mutation chromosome, the corresponding characters in parent chromosomes are flipped and the child chromosomes are generated.

Parent Chromosome	b0	b1	b2	b3	b4	b5	b6	b7
Mutation Chromosome	1	0	0	0	0	0	0	1
Child Chromosome	9-b0	b1	b2	b3	b4	b5	b6	9-b7

Structure of Code128 Bar Code

Barcodes consists of a series of lines that vary in width. They correspond to various numeric, alphanumeric, or multicode configurations readable by a laser barcode scanner. Code 128 is a very effective, high-density symbology which enables the encoding of alphanumeric data. It includes verification protection both through a checksum digit and byte parity checking. This symbology has been widely implemented in many applications where a large amount of data must be encoded in a relatively small amount of space. A Code 128 barcode consists of a leading "quiet zone", one of three start codes, the data itself, a check character, a stop character, and a trailing quiet zone as shown in Fig. 1. The Code 128 data is encoded in strips of bars and spaces. The sequences of zeros or ones simply appear as thicker bars or spaces. The checksum is included in the barcode, and is a digit that verifies that the data just read in was correct. The checksum digit is based on a modulo 103 calculation based on the weighted sum of the values of each of the digits in the message that is being encoded, including the start character.

start code 'C' 'S' 'E' '3' '7' '0' check digit stop code

Fig. 1. Code-128 Barcode

Similar structure exists for Code-39 Barcode.

II. Literature Survey

In literature to date, many GA based encryption algorithms have been proposed. A. Tragha et.al [2] have describe a new symmetric block cipher system namely, ICIGA (Improved Cryptographic Inspired by Genetic Algorithm) which generates a one time session key in a random process. The block size and key length are variables and can be fixed by the end user in the beginning of the cipher process. ICIGA is an enhancement of the system GIC (Genetic Algorithm inspired Cryptography) [3]. There are various proposed methods for image encryption such as quad tree approach, cellular automata [4, 5]. There are wide applications of GA in solving non-linear optimization problems in various domains [6,7]. But very few papers exist which exploit the randomness in the algorithm for implementation of security. Chaos theory and entropy have large application in secure data communication and the desired disorder is provided by inherent nature of genetic algorithm [8, 10]. Mohammad Sazzadul Hoque et.al [11] have presented an intrusion detection system by applying GA to efficiently detect different types of network intrusions. They have used evolutionary theory to filter the traffic data thereby reducing the complexity [12]. There are several papers related to IDS all of which use GA in deriving classification rules [13, 15]. The authors have proposed a symmetric key and asymmetric key encryption/decryption algorithm based on Genetic Algorithm which is implemented in Java for encryption and decryption of a Word document. In their work they have employed four cross-over points and three mutation points, a random factor and a permutation factor to generate a 36-bit robust key [16,17]. But to the best of our knowledge very few papers exist which exploit randomness in generating barcode for authentication purpose.

III. Proposed Method

We have used Code-39 and Code-128 encoding techniques for generating a barcode. The barcode data comprises of 12 randomly generated decimal digits. Each decimal digit is represented using 4 bits. Hence the length of the barcode data is 36 bits.The randomly generated data is transformed into encoded form by applying crossover, mutation and XOR operations before generating a bar code. The application architecture is shown in Fig. 2.

Data
Encoding
Barcode Generation
Physical File

Fig. 2. Application Architecture

A. *Pseudocode*

The pseudo code for barcode generation process using GA is depicted in Fig 3.
Step 1 : Generate a pool of 50 chromosomes consisting of twelve digit numbers which are randomly generated.
Step 2 : Select a single chromosome with the highest fitness value as dictated by the fitness function F given by

$$F= \sum_{i=1}^{12} |d_i - d_{i-1}| + [12 - \text{Max}(rj)] \text{ for } 0<=j<=9$$

where, rj refers to repetition of digit j and $|d_i - d_{i-1}|$ is the absolute numeric distance between the two digits. Store a selected twelve digit random number it in a vector.
Step 3 : Each decimal digit in step 2 can be represented using 4 binary digits. Hence the total number of binary digits required to represent the data is 4 x 12 = 48 bits. Generate a hash H, by repeating digits 0 and 1 (if the digit is > 8) and 0 and 0, otherwise, required number of times. The hash function generated is such that it enables one-to-one mapping between datasets involved. This renders the hash function reversible.
Step 4 : Perform the XOR operation between the data and a 48-bit hash computed above.
Step 5 : Split the vector into two vectors of size six each.
Step 6 : Compute 10's complement of each digit.
Step 7 : Perform the crossover operation at the midpoint.
Step 8 : Perform the mutation at the extreme positions of the vector. The mutation operation consists of flipping the digit from its original value to its complement.
Step 9 : Combine the vectors to reconstruct a 12-digit vector.
Step 10 : Perform the XOR operation between the data and a 48-bit hash computed above.
Step 11 : Use the 12-digit number generated above to generate a barcode in code-128 format.
Step 12 : End

Fig 3 Pseudo code for barcode generation using GA

B.Mathematical Formulation.

Let the original vector be represented by $V_{Original}$. Let H be the hash constructed as follows.

H= ∑' Hi where 1 <= i <= 12 and ∑' is the string concatenation operator.

$$H_i = \begin{cases} 0000, & \text{for } i = 8 \text{ or } 9 \\ 0101, & \text{otherwise.} \end{cases}$$

H is the generated hash of length 48 bits.

Compute the hash of $V_{Original}$ as shown below:

$$V_{Original}\ \theta\ H = V_{Hash}$$

Split the hash into two vectors of size six each. Let the two parts be represented by, V^1_{Hash} and $V^2_{Hash,}$ respectively.

$$V_{Hash} = V^1_{Hash} + V^2_{Hash}$$

Compute 10's complement of each digit. Let the two parts be represented by $(V^1_{Hash})'$ and $(V^2_{Hash})'$, respectively.

Perform the crossover operation at the midpoint. Let the two new parts now be represented by $C(V^1_{Hash})'$ and $C(V^2_{Hash})$, respectively, where C is the crossover operator.

Perform the mutation at the extreme positions of the vector. Let the two parts now be represented by $MC(V^1_{Hash})'$ and $MC(V^2_{Hash})'$, respectively, where M is the crossover operator. Combine the vectors to reconstruct a 12-digit vector.

Perform the XOR operation between the data and a 48-bit hash, H computed above to generate a final vector. Let it be $VT_{ransformed}$. We get,

$$VT_{ransformed} = [\, MC(V^1_{Hash})' + MC(V^2_{Hash})' \,]\ \theta\ H \quad (1)$$

Decoding Vector into original Vector

Perform XOR operation between H and $VT_{ransformed}$ given by equ(1) to get, [MC(V^1_{Hash})΄ + MC(V^2_{Hash})΄].

Split the hash into two vectors of size six each. Let the two parts be represented by, MC(V^1_{Hash})΄ and MC(V^2_{Hash})΄ respectively.

Perform reverse mutation operation and then reverse cross0ver operation on two individual parts to get, (V^1_{Hash})΄ and (V^2_{Hash})΄, respectively.

Take 10's complement of each digit in the two vectors to get, (V^1_{Hash}) and (V^2_{Hash}), respectively.

Combine the two vectors to get V_{Hash}, where

$$V_{Hash} = V_{Original} \ \theta \ H$$

Perform XOR operation between H and $V_{Hash}$to get the original vector.

The entire process of generating the barcode is illustrated below with the help of an example.

Step 1: Generate a 12 digit random number and store it in a vector. Let the number be represented by

8	6	9	3	1	8	9	8	3	9	5	3

Step 2 : Generate Hash H as shown below.

0	5	0	5	5	0	0	0	5	0	5	5

Step 3 : Perform the XOR operation between the data and a 48-bit hash computed above.

8	3	9	6	4	8	9	8	6	9	0	6

Step 4 : Split the vector into two vectors of size six each.

8	3	9	6	4	8

and

9	8	6	9	0	6

Step 5 : Compute 10's complement of each digit.

1	6	0	4	5	1

and

0	1	3	0	9	3

Step 6 : Perform the crossover operation at the midpoint.

0	9	3	4	5	1

and

0	1	3	1	6	0

Step 7 : Perform the mutation at the extreme positions of the vector.

9	9	3	4	5	8

9	1	3	1	6	9

Step 8 : Combine the vectors to reconstruct a 12-digit vector.

9	9	3	4	5	8	9	1	3	1	6	9

Step 9 :Generate Hash H as shown below..

0	0	5	5	5	0	0	5	5	5	5	0

Step 10 : Perform the XOR operation between the data and a 48-bit hash computed above

9	9	6	1	0	8	9	4	6	4	3	9

Step 11 : Use the 12-digit number generated above to generate a barcode in code-128 fromat.

CODE128- 996108946439.

Decoding the barcode

Step 1: Extract the rightmost 12 digits from the barcode.

9	9	6	1	0	8	9	4	6	4	3	9

Step 2 : Generate a hash as shown below:

0	0	5	5	5	0	0	5	5	5	5	0

Step 3 : Perform the XOR operation between the data and a 48-bit hash computed above

9	9	3	4	5	8	9	1	3	1	6	9

Step 4 : Split the vector into two vectors of size six each.

9	9	3	4	5	8

9	1	3	1	6	9

Step 5 : Perform reverse mutation at the extreme positions of the vector.

0	9	3	4	5	1

and

0	1	3	1	6	0

Step 6 : Perform the crossover operation at the midpoint.

1	6	0	4	5	1

and

0	1	3	0	9	3

Step 7 : Compute 10's complement of each digit.

8	3	9	6	4	8

and

9	8	6	9	0	6

Step 8 : Combine the vectors to reconstruct a 12-digit vector.

8	3	9	6	4	8	9	8	6	9	0	6

Step 9 : Generate a hash as shown below:

0	5	0	5	5	0	0	0	5	0	0	5

Step 10 : Perform the XOR operation between the data and a 48-bit hash computed above

8	6	9	3	1	8	9	8	3	9	5	3

which represents the original vector

IV. implementation in JAVA

The model proposed above is implemented in Java using MS Access as backend and Swing for GUI development. JDBC Type-I driver is used. The structure of the Barcode table used in the implementation is as follows :

Table 1 STRUCTURE OF BARCODE TABLE

Column Name	Column Type	Description
Name	Text	Stores User Name of an Employee
Barcode	Text	Stores encoded barcode data.
ID	Text	Stores unique identification for an Employee.

The following table depicts a pool of 20 chromosomes with the highest fitness value of 59 which is used for generating a barcode.

TABLE 2
RANDOMLY GENERATED CHROMOSOMES ALONG WITH THEIR FITNESS VALUES.

Sr.No.	Chromosome	Fitness Value	Sr.No.	Chromosome	Fitness Value
1	628104537742	44	11	642799395427	44
2	055138997499	35	12	861527055490	55
3	421335739797	34	13	310727917436	52
4	**830399068108**	**59**	14	204193991510	55
5	758782554498	32	15	379177926062	58
6	460021515333	34	16	753125934986	39
7	068026024093	59	17	518153194567	51
8	557827878970	35	18	228499422488	36
9	173359242759	48	19	954430201372	35
10	241245870304	39	20	613296011200	37

The following code snippet depicts the generation of barchar in Java.

```
import com.barcodelib.barcode.Linear;
 Linear barcode = new Linear();
        barcode.setType(Linear.CODE39);
        // barcode data to encode
        barcode.setData(finalstring);
        // wide bar width vs narrow bar width ratio
        barcode.setN(3.0f);
        // unit of measure for X, Y, LeftMargin, RightMargin,
        TopMargin, BottomMargin
        barcode.setUOM(Linear.UOM_PIXEL);
        // barcode module width in pixel
        barcode.setX(3f);
        // barcode module height in pixel
        barcode.setY(75f);
        barcode.setLeftMargin(0f);
        barcode.setRightMargin(0f);
        barcode.setTopMargin(0f);
        barcode.setBottomMargin(0f);
        // barcode image resolution in dpi
        barcode.setResolution(72);
        // disply human readable text under the barcode
```

```
barcode.setShowText(true);
// human reable text font style
barcode.setTextFont(new Font("Arial", 0, 12));
//  ANGLE_0, ANGLE_90, ANGLE_180, ANGLE_270
barcode.setRotate(Linear.ANGLE_0);
barcode.setAddCheckSum(true);
bfilename="Noname";
if (!t1.getText().equals(""))
  bfilename="C:\\"+t1.getText()+".gif";
barcode.renderBarcode(bfilename);
```

The following figures 4.1 to 4.4 show the output windows generated by Barcode tool developed in Java.

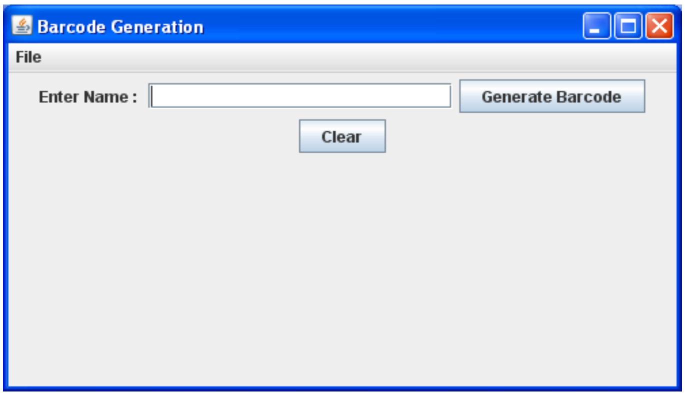

Fig. 4.1 Java Barcode Generation Tool

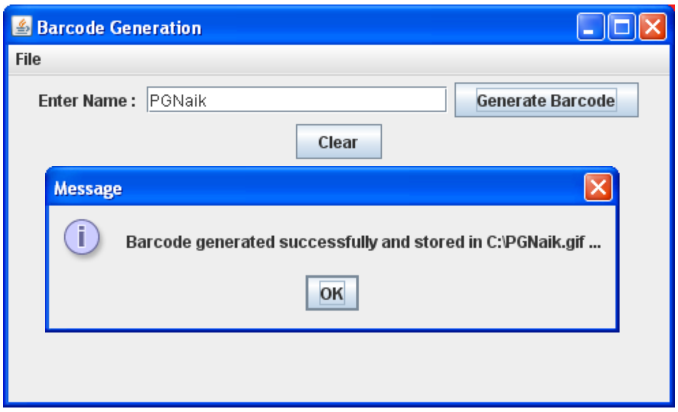

Fig. 4.2 Generation of Barcode

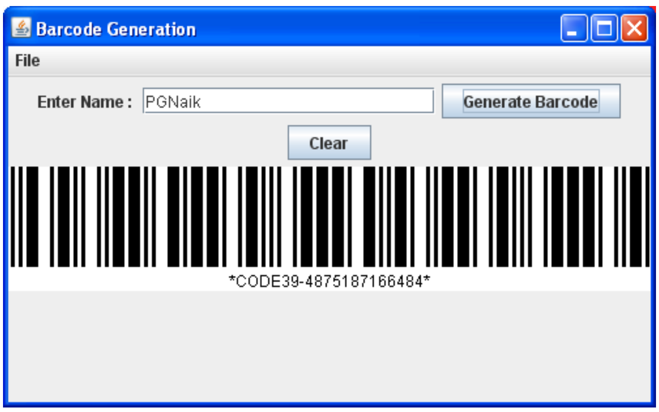

Fig. 4.3 Barcode generated in Code-39 Format

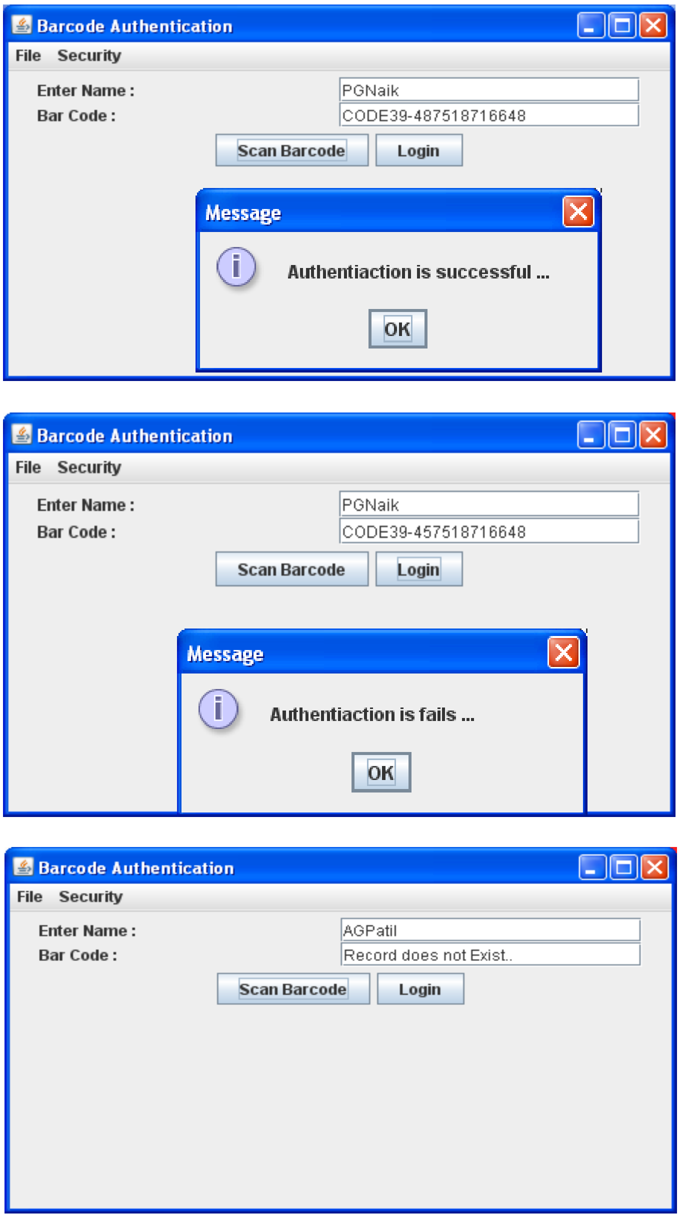

Fig. 4.4 Barcode Authentication Process.

V. Conclusion And Scope For Future Work

In this paper we have proposed a model for barcode generation based on genetic algorithm and is implemented in Java for authentication of employees in a hypothetical organization. The password is encrypted by applying crossover, mutation and XOR operations and is difficult to track. This model provides a unique security layer on top of existing barcode security layer which makes the password more robust and difficult to break. Even if the database is hacked, the password cannot be stolen because the relationship between barcode and ID is not known. The model can be employed in situations where authentication is of prime significance and can be used for secure transmission of limited data such as credit card number. It provides a cheaper solution to RFID for authentication. Due to the symmetry in the operations involved and symmetry of XOR operation, the coding and encoding processes are reversible. Our future work consists of interfacing the software with barcode scanner and study of various coding techniques with reference to their applicability. Our future work consists of interfacing the software with barcode scanner and study of various encoding techniques with reference to their applicability. In future we plan to generate a distributed 3D password incorporating textual, graphical, biometric and barcode authentication where the authentication information is distributed over multiple authentication servers. The encryption key is divided into three sub keys and stored on three different authentication servers. Based on the level of security and infrastructure availability the end user can make selection between 1 to 3 different levels and authentication methods.

References

[1] David. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Pearson Education, 1989, ISBN-13: 978-020115767.

[2] X. F. Liao, S. Y.Lai and Q. Zhou. Signal Processing. 90 (2010) 2714–2722.

[3] H. Cheng and X. Li. IEEE Transactions on Signal Processive. 48 (8) (2000) 2439–2451.

[4] O. Lafe. Engineering Applications of Artificial Intelligence. 10 (6) (1998) 581–591.

[5] R. J. Chen and J. L. Lai. Pattern Recognition. 40 (2007) 1621–1631

[6] Dr.Poornima G. Naik, Girish R. Naik, Application of Genetic Algorithm to Mass Production Line for Productivity Improvement, International Journal of Latest Trends in Engineering and Technology (IJLTET) Special Issue – IDEAS-2013 ISSN:2278-621X.

[7] S. Li, G. Chen and X. Zheng. Multimedia security handbook. LLC, Boca Raton, FL, USA: CRC Press; (2004) [chapter 4].

[8] Y. Mao and G. Chen. Handbook of computational geometry for pattern recognition, computer vision, neural computing and robotics. Springer; (2003).

[9] H. S. Kwok, W. K. S. Tang, Chaos Solitons and Fractals, (2007) 1518–1529.

[10] Mohammad SazzadulHoque, Md. Abdul Mukit and Md. Abu NaserBikas,An Implementation of Intrusion Detection System Using Genetic Algorithm, International Journal of Network Security & Its Applications (IJNSA), Vol.4, No.2, March 2012

[11] L.M.R.J Lobo, Suhas B. Chavan, Use of Genetic Algorithm in Network Security, International Journal of Computer Applications (0975 – 8887)Volume 53– No.8, September 2012

[12] W. Lu, I. Traore, "Detecting New Forms of Network Intrusion Using Genetic Programming". Computational Intelligence, vol. 20, pp. 3, Blackwell Publishing, Malden, pp. 475-494, 2004.

[13] M. M. Pillai, J. H. P. Eloff, H. S. Venter, "An Approach to Implement a Network Intrusion Detection System using Genetic Algorithms", Proceedings of SAICSIT, pp:221-228, 2004.

[14] S. M. Bridges, R. B. Vaughn, "Fuzzy Data Mining And Genetic Algorithms Applied To Intrusion Detection", Proceedings of 12th Annual Canadian Information Technology Security Symposium, pp. 109-122, 2000.

[15] M. Middlemiss, G. Dick, "Feature selection of intrusion detection data using a hybrid geneticalgorithm/KNN approach", Design and application of hybrid intelligent systems, IOS Press Amsterdam, pp.519-527, 2003.

[16] Poornima G. Naik and Girish R. Naik, Symmetric Key Encryption using Genetic Algorithm, International Journal of Information systems (A Journal of SIMCA), Vol IV, Issue II, Jan-May 2014, ISSN:2229-5429

[17] Poornima G. Naik and Girish R. Naik, Asymmetric Key Encryption using Genetic Algorithm, International Journal of Latest Trends in Engineering and Technology (IJLTET), Vol. 3 Issue 3 January 2014, 118-128, ISSN: 2278-621X.