Generating the Best Fit Key in Cryptography using Genetic Algorithm

Sania Jawaid
Department of Computer Science,
Jamia Hamdard, New Delhi, India

Adeeba Jamal
Department of Computer Science,
Jamia Hamdard, New Delhi, India

ABSTRACT
The goal of network security is to provide a secure, effective and private communication between the sender and the receiver. In order to achieve a high level of security, data is sent in the encrypted form to its intended recipient. But, tampering with the text and eavesdropping have assumed colossal proportions. This is generally done by decoding the key. Therefore, to make the key strong and almost unpredictable, a method based on the theory of natural selection has been proposed in the paper. This method finds the best fit element in the environment. as the paper uses Genetic Algorithm to accomplish the above task. The paper not only illustrates this innovative method of key generation but also demonstrates its implementation. To achieve even more high standards of security Data Encryption Standard cipher program has been used for verification and validation.

Keywords
Genetic Algorithm, Data Encryption Standard, Cryptography, Key Generation

1. INTRODUCTION
Network security embraces some policies, adopted by the network administrator. Network security precludes the misuse and modification of the network resources. Authentication is also an important part of network security [1]. To authenticate, the username and the password is used.

'Cryptography' is a Greek word which means the style of secret writing [2] [13]. It is used for altering the messages to make them more secure. Cryptography is for the receiver, so that no intruder is ingenious enough to decode the text.

Network Security norms are necessary to protect the data during transmission. Network needs security from the hackers and the attackers spread everywhere [3]. Whenever data is transferred via the networks; it can be accessed by unguarded computers. To avoid data to be lost or get corrupted, network security is very essential and at the same time pivotal [2] [14]. The major role of network security lies in avoiding the tampering of data transmitted across the network.

The plain text is reformed into cipher text when the encryption algorithm is applied [1] [7]. At the receiver's site cipher text is transformed into the plain text. Ciphering of the text is done so that Eve does not interrupt in between to hack the message.

The paper has been divided into the following section. Section 2 of the paper discusses cryptography, section 3 introduces genetic algorithms, section 4 discusses the proposed work, section 5 illustrates observations, section 6 discusses inference and the last section concludes.

2. TYPES OF CRYPTOGRAPHY
The systems based on cryptography are majorly divided into three independent categories [7] [2]:

a) The operations which are adequate for transformation of plain text to cipher text.

b) The number of keys which are used by the algorithm.

c) And lastly, how the plain text is processed [15].

Broadly there are two types of Cryptography:

2.1 Symmetric Key Cryptography
In this type, the sender and the receiver employ the equivalent key for encryption and decryption and hence the key is shared [1]. The sender uses this key and encryption algorithm to transform the plain text into cipher text [3]. The receiver uses the same key and decryption algorithm to convert the cipher text back into the plain text.

Some of the examples of symmetric cryptography are the block ciphers like AES and DES.

DES stands for Data Encryption Standard. This algorithm takes 64 bit key as the plain text and gives the output of 64 bit ciphered text [2] [14]. There are two P boxes which are used as initial and final permutations. Also there are 16 rounds and for each round a key is generated.

AES stands for Advanced Encryption Standard. This algorithm was introduced because length of the key in DES was small [1]. So, to increase the length of the key AES is implemented. It has three types of rounds with corresponding bits of text. The functioning is same but the difference lies in the order keys are used.

2.2 Asymmetric Key Cryptography
This type of cryptography is also called public key cryptography. In this, both the parties (sender and receiver) do not use the same key. The key is not shared. The sender uses the public key and encryption algorithm to encrypt the message [10]. The receiver uses the private key and decryption algorithm to convert the message back to plain text.

Some of the examples of asymmetric key cryptography are RSA and Diffie-Hellman Key Exchange [7] [12].

RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman [1]. It is a public key cryptography technique. There are two large prime numbers which are chosen. After the final calculation of numbers they are used for encryption and decryption.

Diffie- Hellman Key Exchange is also a public key cryptography technique [3] [14]. Two numbers 'x' and 'y' are chosen. Then R1 and R2 are calculated for exchange between Alice and Bob. After the exchange has been done the session key (K) is generated. The session key is shared.

3. GENETIC ALGORITHM

Genetic Algorithm (GA) is a function that imitates the process of natural selection in the field of Artificial Intelligence (AI) [11] [14]. GA involves some of the operators like crossover, mutation and selection.

3.1 Population Generation

The process of Genetic Algorithms (GA), usually, starts with a population which is randomly generated and is composed of several chromosomes [9]. The chromosomes are either binary or hex. A chromosome has cells which are 0/1 or a hex number, depending on the type of population. GA's are iterative in nature and transform the population of chromosomes, generated into a new generation. Various operators are applied when the population is generated for selection of individuals. The individuals are selected based on their probability, various genetic operations and their fitness value [8] [15].

3.2 Crossover

After the population is generated the most important parameter of the GA is applied to the generated population. Crossover operator is applied on an individual and the resulting generation is much more fit, than the previous generation. Crossover is the process of taking more than one parent solutions. These parent solutions are chosen from the generation [4]. The offspring results from those solutions. A crossover operator takes more than one parent solutions and produces a child solution. Also there can be different types of crossover such as one point crossover or two point crossover.

3.3 Mutation

Mutation is another important genetic operator. It is used for maintaining genetic diversity among the different generations [4] [8]. Mutation alters each gene independently depending on the probability. This probability is called the mutation rate [10]. After the mutation operator is applied, the population remains same. But the less fit chromosomes are replaced by the more fit chromosomes. The main motive of the mutation process is to beget such chromosomes which have least similarities among themselves [6] [15].

3.4 Selection

Selection stage comes after the population has been generated, the crossover and mutation operators have been applied and the individual fitness of each chromosome is calculated. Selection procedure is used for selecting the better individuals. More fit chromosomes are selected out of the existing population. These individuals have a higher chance over the other individuals based on the proportion to fitness [9] [13]. To implement the selection technique, Dominance Testing is used.

4. PROPOSED WORK

In this paper, the use of GA is described in order to find the best fit key for the cryptographic algorithm. In the research paper an approach of a pseudo random number generator which is used to produce unique keys further used in the various ciphers has been proposed. Since in Genetic algorithm only the fittest one survives, hence there was a need to define a fitness function which helped us identify the best keys among the rest. The key produced is shown to be non-repeating which makes the cipher text difficult to decode. The basic processes in Genetic Algorithm, such as Initial Population Generation, Crossover, Mutation, Fitness Function Calculation and Final Key Selection are used. In this paper a 48-bit key Data Encryption Standard (DES) Cipher is used in the end to show the implementation of the research.

The proposed solution involves the following steps as illustrated in Figure 1:

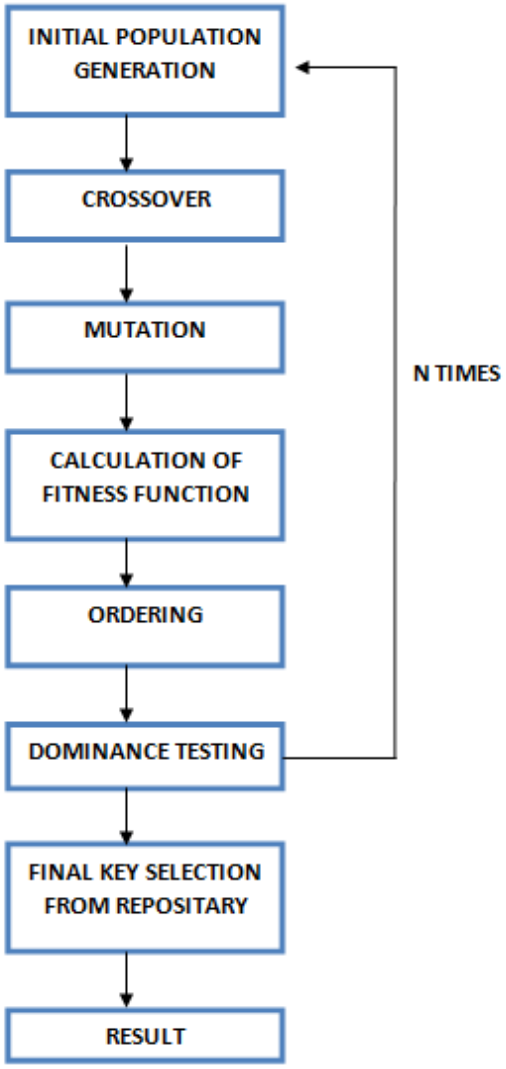

Figure 1. Steps involved in Generating the Best Fit Key in Cryptography using Genetic Algorithm.

4.1 Population Generation

GA typically starts with a population of computer generated random keys which are known as chromosomes. The number of genes will be equal to the length of the key used. Here 48-bit key size has been used. The population size depends on a large number of possible solutions. After the initial population has been generated, the population matrix undergoes various Genetic operations which increase the total number of chromosomes. These operations are applied to individual(s) selected from the population. The individuals are probabilistically selected to participate in the genetic operations based on their fitness.

4.2 Crossover

Crossover primarily simulates sexual genetic recombination. There are various ways by which it can be implemented in GAs. Sometimes crossover is applied with moderation so usually the crossover probability is defined to indicate a ratio of the number of couples selected for mating. Crossover is applied on two randomly chosen individuals.

A crossover rate is chosen and then number of crossovers is calculated with the formula:

noco = cor * m * n / 100,

Where noco = number of crossovers,
cor = crossover rate,
m = key length,
n = number of keys.

A random crossover point is then generated and the bits of the first individual from the starting till the crossover point and the bits of the second individual starting from the crossover point till the end are copied. The successor thus generated from crossover is very different from their initial parents.

4.3 Mutation

Mutation is a genetic operator used to maintain diversity from one generation of a population of chromosomes to the next. It is analogous to biological mutation. Mutation alters one or more gene values in a chromosome from its initial state. In mutation, the solution may change entirely from the previous solution. Mutation occurs during evolution according to a user-defined mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search. For example, if the mutation probability is set to 1, it means 1 out of every 100 bits will be manipulated. In the proposed solution, the bit inversion mutation operator is used. Here also number of mutations is calculated via the formula,

nom = mr * m * n / 100,

Where nom = number of mutations,
mr = mutation rate,
m = key length,
n = number of keys.

Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, so that it cannot slow or stop evolution.

4.4 Calculating Fitness Function

Fitness evaluation involves defining an objective or fitness function which tests for suitability of the chromosome. As the algorithm proceeds, the individual fitness of the "best" chromosome is increased. Also the total fitness of the population is increased as a whole. In the proposed solution, all the keys which are in binary form are first converted into their respective decimal forms.

Gap test and Frequency Test are performed on them. The proposed fitness function is calculated by the formula,

$\mathbf{F = 1 / (1 + e^{(-\lambda)})}$

Where $\lambda = C1\ \lambda_1 + C2\ \lambda_2$

λ_1 **= Frequency Test**

λ_2**= Gap Test**

C1 = C2 = 1

4.4.1 Frequency Test

The frequency of each number is then calculated. This test is performed to check the number of times a chromosome is repeated [10]. The frequency test is used for testing randomization of the key.

4.4.2 Gap Test

This test is performed to calculate the gap between the two repeating numbers [11]. The gap test is used to determine the implication of the interval between recurrences of the same digit.

4.5 Ordering

The numbers are arranged in a sorted order according to their fitness values. The key with the highest fitness value is at the top of the list and the key with the lowest fitness value is at the bottom.

4.6 Dominance Testing

After ordering has been done, there will be set of random keys to choose from. On those keys dominance testing is performed. The key with highest fitness value will be paired with the rest of the keys and hamming distance between them will be calculated. Hamming distance is calculated by performing XOR of the two binary keys, then calculating the number of 1's. Out of those keys, one key will be chosen randomly which will dominate over others. This key will be the Final key resulting from Dominance Testing.

4.7 Final Key Selection from Repository

The whole process is then repeated n times and all the keys generated from n iterations are stored in the repository. The final key is then selected on the basis of Dominance Testing.

4.8 Result

The final key selected is then used for the encryption and decryption process in the DES cipher.

5. OBSERVATION

The technique proposed in section 4 was implemented using Java Technology and observations were analyzed. These observations led to the conclusions presented in the next section. Figure 2 is shown as a screenshot from the working program of the proposed solution:

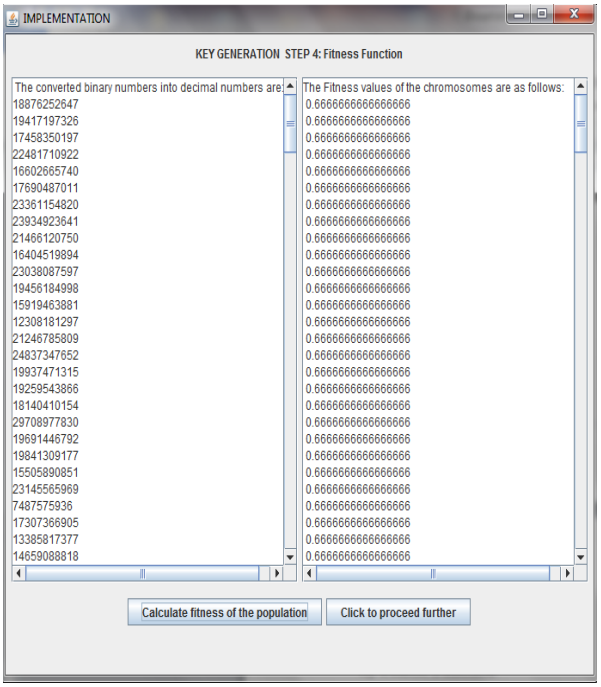

Figure 2. Screenshot of the Fitness Function Calculation

The observations from the implementation of the proposed algorithm are as follows:

5.1 Population Generation

In the implementation of the proposed algorithm, a random initial population of 100 chromosomes each having 48 genes is generated.

5.2 Crossover

The second step is to create a new child out of two randomly selected parents. Crossover Rate is taken as 2.5 which gives the number of crossovers as 120. For each iteration, a different crossover point is selected to perform single point crossover. Now the total population size becomes 220. The diagram shown in Figure 3 illustrates the single point crossover operation:

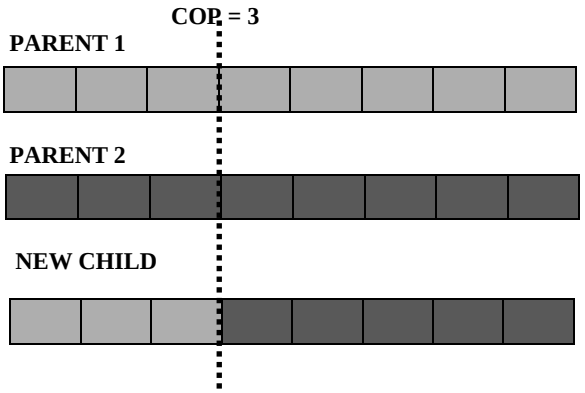

Figure 3. Single Point Crossover Operation

Table 1 shows 10% of the results generated from the crossover operation:

Table 1. Crossover Operation

S. NO.	CHROMOSOME	
1.	PARENT 1	010110010001011000111110000001001001000011110111
	PARENT 2	100111100110101000111101000110110110110100110000
	CROSSOVER POINT	27
	NEW CHILD	010110010001011000111110000101101101101001100000
2.	PARENT 1	010101001111101000010110001000011000011010000110
	PARENT 2	110010111010101111101100101101001011000110100110
	CROSSOVER POINT	15
	NEW CHILD	010101001111101111101100101101001011000110100110
3.	PARENT 1	010101111001010010100000100011111110100111111011110
	PARENT 2	100010010100001110011110010001110111010011110010
	CROSSOVER POINT	22
	NEW CHILD	010101111001010010100010010001110111010011110010
4.	PARENT 1	111110100010001011001000110001010101011111010011001
	PARENT 2	100010010100001110011110010001110111010011110010
	CROSSOVER POINT	28
	NEW CHILD	111110100010001011001000110001110111010011110010
5.	PARENT 1	111110001010011011111000000011000101001000000011
	PARENT 2	101011000010011101001010001111011000101110001110
	CROSSOVER POINT	9
	NEW CHILD	111110001010011101001010001111011000101110001110
6.	PARENT 1	110001111011111000111101110010001110010000001111
	PARENT 2	000000011010101111000010111101010011111010101111
	CROSSOVER POINT	33
	NEW CHILD	110001111011111000111101110010001110010111010111
7.	PARENT 1	001101011111100000000010010101010000111100000110
	PARENT 2	011110110100000010111100010100010011111010010011
	CROSSOVER POINT	8
	NEW CHILD	001101010100000010111100010100010011111010010011
8.	PARENT 1	000001001001000010000000101000001000011100111001
	PARENT 2	001001011010100100001100110000010100101111001
	CROSSOVER POINT	8
	NEW CHILD	000001001010100100001100110000010100101111001
9.	PARENT 1	000111100100011011110011001010110001011100101101
	PARENT 2	011111000110100111011001110111110011111100100110
	CROSSOVER POINT	20
	NEW CHILD	000111100100011011111001110111110011111100100110
10.	PARENT 1	011001011000011011100110100101101011101111010
	PARENT 2	000111100100011011110011001010110001011100101101
	CROSSOVER POINT	16
	NEW CHILD	011001011000011011110011001010110001011100101101

5.3 Mutation

The next step is to perform mutation where a random chromosome is chosen from the existing population and a mutation point is selected to invert that particular bit. Mutation Rate is selected as 0.5 which gives the number of mutations to be performed as 52. Table 2 below shows first 10 chromosomes generated after the mutation operation:

Table 2. Mutation Operation

S. NO.	CHROMOSOME	
1.	ORIGINAL CHROMOSOME	11111000101001101111100000001100010101001000000011
	MUTATION POINT	33
	MUTATED CHROMOSOME	11111000101001101111100000001100111010010000000011
2.	ORIGINAL CHROMOSOME	00000101010001111111011011100000011110000010011111
	MUTATION POINT	14
	MUTATED CHROMOSOME	00000101010000111111011011100000011110000010011111
3.	ORIGINAL CHROMOSOME	11110100100111010110010111100010100100111010101010
	MUTATION POINT	17
	MUTATED CHROMOSOME	11110100100111011110010111100010100100111010101010
4.	ORIGINAL CHROMOSOME	11000100110000111000100110001101011001000011111011
	MUTATION POINT	25
	MUTATED CHROMOSOME	11000100110000111000100100001101011001000011111011
5.	ORIGINAL CHROMOSOME	00100101100101001001111101100110100010000001101010
	MUTATION POINT	48
	MUTATED CHROMOSOME	00100101100101001001111101100110100010000001101011
6.	ORIGINAL CHROMOSOME	11111000101001101111100000001100010101001000000011
	MUTATION POINT	43
	MUTATED CHROMOSOME	11111000101001101111100000001100010101001000100011
7.	ORIGINAL CHROMOSOME	00001100010011100101110111100010101110001100011101
	MUTATION POINT	21
	MUTATED CHROMOSOME	00001100010011100101010111100010101110001100011101
8.	ORIGINAL CHROMOSOME	01111000011010000101100001000001011110101011001000001
	MUTATION POINT	2
	MUTATED CHROMOSOME	00111000011010000101100001000001011110101011001000001
9.	ORIGINAL CHROMOSOME	10010011110001111101110111101000001011111001101
	MUTATION POINT	30
	MUTATED CHROMOSOME	10010011110001111101110111101100001011111001101
10.	ORIGINAL CHROMOSOME	10100001110011000101001100100110001000100111001100011
	MUTATION POINT	30
	MUTATED CHROMOSOME	10100001110011000101001100100100001000100111001100011

5.4 Fitness Function Calculation

5.4.1 Conversion

The binary valued chromosomes from the population of size 220 were converted into the decimal number format.

5.4.2 Fitness Value

The Fitness values of the above chromosomes are shown in Table 3.

Table 3. Fitness Function

S. NO.	Frequency Test (λ_1)	Gap Test (λ_2)	$\lambda=\lambda_1+\lambda_2$	Fitness Value F = $1/(1+e^{(-\lambda)})$
1.	1	0.0	1.0	0.6666666666666666
2.	1	0.0	1.0	0.6666666666666666
3.	1	0.0	1.0	0.6666666666666666
4.	1	0.0	1.0	0.6666666666666666
5.	2	0.9181818181818182	2.918181818181818	0.29942712908773716
6.	1	0.0	1.0	0.6666666666666666
7.	1	0.0	1.0	0.6666666666666666
8.	1	0.0	1.0	0.6666666666666666
9.	1	0.0	1.0	0.6666666666666666
10.	2	0.9227272727272727	2.922727272727273	0.29864652653727375

5.5 Ordering

The chromosomes are then sorted in the decreasing order according to their fitness values as shown in Table 4:

Table 4. Ordering

S.NO.	ORDERED POPULATION
1.	00010110010010101011011010110101111101010101111
2.	11110100111011000100101111110100100110010011101
3.	01011001000101100011111100000100100100001111011
4.	11000001011100001000110110001010101000001010101
5.	10001011110011010010110110110110110011001000110
6.	10101101011111101110101000110101001111100011010
7.	01111111010000001011110001010001001111101001001
8.	00101000010100010010001010101011111101011010100
9.	11011001100001110100100111111011000110000011111
10.	00011011110010100001000011100001001101000111100

5.6 Dominance Testing

The topmost key from the sorted population, i.e. the fittest value is selected and paired with the rest of the chromosomes one at a time. XOR function is applied to all these pairs and each pair's hamming distance is calculated. The pair with the maximum hamming distance is then chosen and through random selection one of the chromosomes is selected from that final pair. In the implementation, the most dominating key came out to be as:

01101110101101000110000001010010010110001110000 0

This key is then stored in the repository and the entire process starting from the first step is repeated 100 times.

5.7 Final Key Selection from Repository

The following Table 5 shows the first 10 dominant keys generated which get stored in the repository:

Table 5. Repository

S.NO.	REPOSITARY
1.	00100000010101110101000000001010011100111 0000100
2.	10111011100011000001001111111010101100000 1110111
3.	00010110100111110011100010100000111100011 0000000
4.	11111000110110010111101110010010001111000 0111101
5.	01101001100011001101011011000001000111110 1010110
6.	11111101001010000001010000011001101110101 1011010
7.	11100111000010001011100110101001101100001 1100011
8.	01011010110011011111011111011011001010111 1100111
9.	00100111101110101110100110111101011010000 1101001
10.	11010001000100010001111000000001101011101 1110101

The Dominance Testing is again applied to all the 100 keys generated and then the final key is selected for the DES Cipher for further data encryption and decryption. This final key came out to be the 73rd key from repository which is as follows:

00100111110111011001001000000001101011000101110 1

6. RESULTS

The work has been implemented and analyzed. The implementation was done using Java Technology. Random samples were created by generating an initial random population of 100 chromosomes. Various tests were applied on the samples and the results were also satisfactory.

After generating 100 chromosomes, crossover function was applied taking the total population size to 220 chromosomes using the crossover rate as 2.5. Mutation rate was selected as 0.5. The fitness values of the keys were calculated and analyzed using the Frequency and Gap Tests. The maximum frequency which was observed in the sample was 2. This means the chromosomes were found to be repeated at most two times. This proves the randomness of the sample used. Therefore, the final result generated came out to be as random and unique as possible.

7. CONCLUSION

It can be observed from the above tables that the generated key is very random and almost difficult to decode. The Genetic Processes involved are very complex and when used together, they generate the most random and non-repeating key as possible. The implementation further involves the use of DES cipher for data encryption which is very complicated itself and it makes almost impossible for the cryptanalysts to attack the data. The proposed solution has a total of seven rounds and the whole process is again repeated 100 times. Despite this, the key gets generated in a very short duration of time. This proves to be a great advantage as the computational time of generating the key is lesser than encrypting the data using DES.

8. FUTURE SCOPE

This propounded idea can not only overcome the existing network security but also has an edge over the computational time. However, in this paper only a need for data encryption is created and its implementation is shown. Messages can be in the form of images or audio as well which are equally important to be protected against eavesdroppers and cryptanalysts [11] [15]. But implementing the same algorithm for image encryption could not be shown due to time constraints. Moreover, the proposed approach can be expanded further by using Neural Networks in Artificial Intelligence to calculate the mathematical coefficient for the Gap Test and the Frequency Test used in the fitness function. Both Neural Network and Genetic Algorithm provide with non-linear problem solving iteration [5]. So using both of the techniques together can solve a great deal of problems.

9. REFERENCES

[1] Behrouz A Forouzan, “Data Communication and Networking” Tata McGraw- Hill Publishing Company Limited, Special Indian Edition 2006.

[2] William Stallings, “Network Security Essentials,” Fourth edition.

[3] William Stallings, “Cryptography and Network security”, Fifth Edition.

[4] Harsh Bhasin and Nakul Arora, “Key Generation for Cryptography using Genetic Algorithm”.

[5] Sindhuja K and Pramela Devi S, "A Symmetric Key Encryption Technique Using Genetic Algorithm", Sindhuja K et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, ISSN: 0975-9646 Vol. 5 (1), 2014, pg 414-416.

[6] Y.V. Srinivasa Murthy, Dr. S. C. Satapathy, P. Srinivasu and A.A.S. Saranya, "Key Generation for Text Encryption in Cellular Networks using Multi-point Crossover Function", International Journal of Computer Applications (0975-8887) Volume 32-No.9, October 2001.

[7] Bethany Delman, Genetic Algorithms in Cryptography, MS Thesis 2004.

[8] A. Tragha, F. Omary, A. Kriouile, "Genetic Algorithms Inspired Cryptography", A.M.S.E Association for the Advancement of Modeling & Simulation Techniques in Enterprises, Series D: Computer Science and Statistics, November 2007.

[9] A Kumar, N Rajpal, Application of Genetic Algorithm in the Field of Steganography, in Journal of Information Technology, Vol. 2, No.1, Jul-Dec.2004, pg 12-15.

[10] Oded Goldreich, Foundations of Cryptography, Volume 1: Basic Tools, Cambridge University Press, 2001, ISBN 0-521-79172-3.

[11] A. J. Bagnall, "The Applications of Genetic Algorithms in Cryptanalysis", School of Information Systems, University Of East Anglia, 1996.

[12] N. Koblitz, "A Course in Number Theory and Cryptography", Springer-Verlag, New York, Inc., 1994.

[13] Harsh Bhasin, "Test Data Generation Using Artificial Life and Cellular Automata", ACM SIGsoft Software Engineering Notes, January 2014.

[14] Harsh Bhasin, Neha Singla, "Cellular Genetic Test data Generation": ACM Sigsoft Software Engineering Notes, September Edition, 2013.

[15] Harsh Bhasin et. al., "Cellular Automata based Test Data Generation", ACM Sigsoft Software Engineering Notes, July, 2013.