© 2015 IJSRSET | Volume 1 | Issue 1 | Print ISSN : 2395-1990 | Online ISSN : 2394-4099
Themed Section: Engineering and Technology

Key Exchange Based on Genetic Algorithm

Amanie Hasn Alhussain
Information Technology Department, Peoples' Friendship University, Moscow, Russia

ABSTRACT

In this study, has shown how to design key exchange algorithm based on the features of crossover and mutation operations of genetic algorithm (GA) and asymmetric key encryption. The number of the crossover points together with number of mutation points dictate the length of the secret key and hence the strength of the algorithm. The algorithm is further strengthened by making it difficult to break by permuting the key by a random permutation factor; the randomness together with permutation makes the algorithm robust and hard to break. The proposed algorithm solves the problem of storing and distribution of the secret key over the network; the various examples and implementation of the algorithm proves that it exchanges the keys over the channel successfully

Keywords: key exchange algorithm, Genetic Algorithm, Mutation, Crossover, public key, Asymmetric Key Encryption.

I. INTRODUCTION

The growing dependence on computers to process information and transmit it across virtually connected systems has increased the need for security. Cryptography is a fundamental technique for securing information. Any cryptographic system based on the use of cryptographic keys. In the symmetric cryptosystem sender and recipient of the message using the same secret key. It should be known to all other simultaneously and periodically updated by the sender and the recipient. The process of distribution (distribution) of secret keys between participants of information exchange in symmetric cryptosystems is very complex.

Asymmetric cryptosystem involves the use of two keys - public and private (secret). The public key can disclose, and private must be kept secret. In an exchange of messages must forward only the public key. An important requirement is to ensure the authenticity of the public key forwarded.

Key distribution is the most important process in the management of keys. It must meet the requirements of the efficiency and accuracy of distribution, and the confidentiality and integrity of the distributed keys [1].

For the distribution of keys between users of computer network, there are the following basic ways: 1. the use of one or more Key Distribution Center- 2. The direct exchange of session keys between the users of the network.

In both cases, should be provided with the authenticity of the communication session. This can be achieved by using a challenge-response mechanism or timestamps.

In literature to date, many key exchange algorithms has been proposed. Diffie-Hellman was the first public-key algorithm ever invented, way back in 1976. It gets its security from the difficulty of calculating discrete logarithms in a finite field, as compared with the ease of calculating exponentiation in the same field. Diffie-Hellman

can be used for key distribution but it cannot be used to encrypt and decrypt messages. Diffie-Hellman also works in commutative rings. Z. Shmuley and Kevin McCurley studied a variant of the algorithm where the modulus is a composite number. V. S. Miller and Neal Koblitz extended this algorithm to elliptic curves. Taher ElGamal used the basic idea to develop an encryption and digital signature algorithm [2].

Up to Genetic algorithm, it is a search heuristics that mimics the process of natural evolution. Genetic algorithms are based on the Darwinian theory of evolution. Genetic algorithms have been invented by J. Holland in 1960s [4], and since then they have been successfully applied to the variety of problems in the field of combinatorial optimization [3]

This paper is proposed an algorithm for key distribution and management and key exchange based on genetic algorithm and asymmetric key encryption.

The rest of the paper is organized as follows:

In Section II, the proposed algorithm was introduced; Section III discusses about implementation example of the algorithm; Section IV experimental results; Section V concludes the paper.

II. METHODS AND MATERIAL

THE PROPOSED ALGORITHM:

Description of the Key Exchange Algorithm:

The steps of the algorithms consist of two steps:

The First Step:

It is on the sender side:

1. Choose Block size N, numbers of crossover points R, numbers of mutation points M
2. Create two random numbers, the first called random factor which range from 1 to 15, while the second called permutation factor which range from 1 to 7.
3. Send these values to the other side.

The second Step:

It is on the second side (receiver):

Input: Block size N, numbers of crossover points R, numbers of mutation points M, random factor, permutation factor

1. Complete the length of the key with spaces to equal the maximum key length=16 (supposed in our algorithms could be different depend on the purpose, constrains and conditions which algorithms is served).
2. Convert each character of the key into its ASCII cod value.
3. Convert each ASCII cod value into binary (8bit) representation.
4. Divide the string of the bits which is obtained in step 3 into block with size blockSize N to form one row matrix.
5. Transform one raw matrix which is generated in step 4 into matrix with N columns.
6. Transform multicolumn matrix which is obtained from step5 into transport matrix.
7. Merge the rows in the previous matrix into one strings, divide it into two substrings (which formed parents in our algorithm).
8. Generate R random crossover points which range from 1 to length(parent)-1,and sort them.
9. Crossover the parent strings which is generated in step7 according to the R crossover points which is created in step8 to generate new strings of bits which formed the children (multiple-point crossover).
10. Generate M random mutation points which range from 1 to length(child)-1.
11. Mutate each string (child) which is created in step9 with M mutation points which is created in step10 respectively.
12. Divide each child into substrings with 8bits length.
13. Create random cross point range from 0 t0 7which would be used to cross each 8bits string.
14. Divide each 8bits string into two parts according to cross point, and convert each part into hexadecimal value to form encrypted key which is the sequence of hexadecimal values.
15. Formulate private key which consists of the sequence of block Size, crossover point1…crossover pointR, mutation point1…

mutation pointM , cross point, permutation factor and random factor.

16. Repeat random factor L times to form randomFactorString : L =the length of the sequence obtained in the previous step.
17. Formulate practical public key = (private key) XOR randomFactorString.
18. Create permutated practical public key by permutated practical public key according to permutation factor
19. Create Public key which is the sequence of permutated practical public key followed by permutation factor and random factor.
20. Message to send is encrypted key which is created in step15, followed by public key which is generated in step20.

The public key would be sent directly from the sender to the receiver by inserting it at the end of the encrypted key to form the message which would be sent to the receiver (Fig.1), so the receiver would retrieve the private key from the public key by reverse the steps of obtaining the public key and based on the symmetry of the operations involved and the symmetry of XOR operations [5].

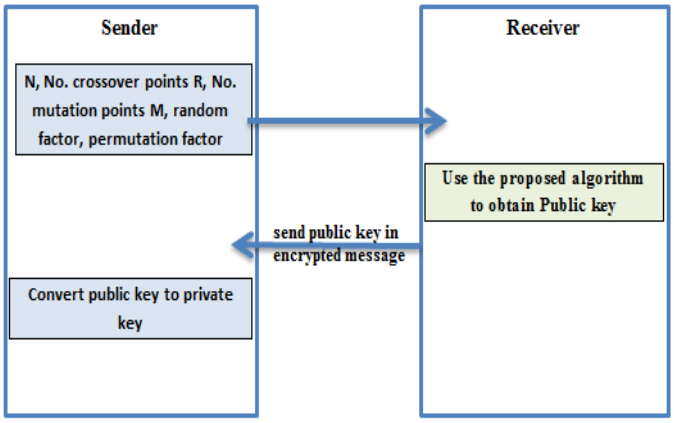

Figure 1. The schema for exchange the key

EXAMPLE OF APPLYING THE PROPOSED ALGORITHM:

On the sender side:

If the sender chooses the following parameters:
Block size N=4, numbers of crossover points R=2, numbers of mutation points M=4,
Random factor=14, permutation factor=2

On the receiver side:

Input: Block size N, numbers of crossover points R, numbers of mutation points M, random factor, permutation factor

Suppose input key: Agt8374HmC9

1. Complete the length of the key with spaces to equal the key length=16

   Agt8374HmC9

2. Convert each character into its ASCII cod.

   65 103 116 56 51 55 52 72 109 67 57 32 32 32 32 32

3. Convert each ASCII cod into binary (8bit).

   01000001 01100111 01110100
   00111000 00110011 00110111
   00110100 01001000 01101101
   01000011 00111001 00100000
   00100000 00100000 00100000 00100000

4. Divide the string of the bits into block with size blockSize N=4.

   0100 0001 0110 0111 0111 0100 0011 1000
   0011 0011 0011 0111 0011 0100 0100 1000
   0110 1101 0100 0011 0011 1001 0010 0000
   0010 0000 0010 0000 0010 0000 0010 0000

5. Transform one raw into matrix with N columns.

   0100 0001 0110 0111
   0111 0100 0011 1000
   0011 0011 0011 0111
   0011 0100 0100 1000
   0110 1101 0100 0011
   0011 1001 0010 0000
   0010 0000 0010 0000
   0010 0000 0010 0000

6. Transform multicolumn matrix into transport matrix.

   0100 0111 0011 0011 0110 0011 0010 0010
   0001 0100 0011 0100 1101 1001 0000 0000
   0110 0011 0011 0100 0100 0010 0010 0010
   0111 1000 0111 1000 0011 0000 0000 0000

7. Merge the rows into one strings, divide it into two substrings.

   parent1: 0100011100110011011000110010001000010100001101001101
   100100000000

   parent2: 0110001100110100010000100010001001111000011110000011
   000000000000

8. Generate two random crossover points, and sort them.

   crossover1: 29 crossover2: 38

9. Crossover the parent strings according to the two crossover points

Child1:

0100011100110011011000110010001001111000001101001101100100100000000

Child2: 0110001100110100010000100010001000010100011110000000000000

10. Generate four random mutation points. Mutation point1:13; Mutation point2:32; Mutation point3:17; Mutation point4:16;
11. Mutate each string (child) with two mutation points respectively.

Mutated Child1: 0100011100110111011000110010001011111000001101011101100100000000

Mutated Child2: 0110001100110100100000100010001000010100011111000000110000000000000

12. Divide each child into substrings with 8bits length.

01000111 00110111 01100011 00100010 11111000 00110100 11011001 00000000 01100011 00110100 10000010 00100010 00010100 01111000 00110000 00000000

13. Create random cross point. crossPoint : 2
14. Divide each 8bits string into two parts according to cross point, and convert each part into hexadecimal value.

1 7 0 37 1 23 0 22 3 38 0 34 3 19 0 0 1 23 0 34 2 2 0 22 14 1 38 0 30 0 0

15. Create two random numbers: random factor and permutation factor .Random Factor: 14; permutation Factor: 2;
16. Formulate private key. 4 29 38 13 32 17 16 2 2 14
17. Repeat random factor 8 times to form randomFactorString randomFactorString: 14 14 14 14 14 14 14 14
18. Formulate practical public key = (the first eight parts of private key) XOR randomFactorString.10 3D 2C 7 26 3 2 16
19. Create permutated practical public key 2C 7 26 3 2 16 10 3D
20. Create Public key 2C 7 26 3 2 16 10 3D 2 14
21. Message to send

1 7 0 37 1 23 0 22 3 38 0 34 3 19 0 0 1 23 0 34 2 2 0 22 0 14 1 38 0 30 0 0 2C 7 26 3 2 16 10 3D 2 14

Graphic user interface of this example is presented in the experimental results.

III. RESULTS AND DISCUSSION

The first step is included sending the parameters from Bob to Anna

If the values of the parameters are chosen as: Block size N=4, numbers of crossover points R=2, numbers of mutation points M=4, Random factor=14, permutation factor=2

The second step: In Anna side, when the key is entered, it would be encrypted after that the private and public key would be produced; the message to be sent is the encrypted key followed by produced public key.as shown in fig 2; the basic internal parameters that would be used is shown in fig 3; while the description of the encrypted key process is shown in fig 4.

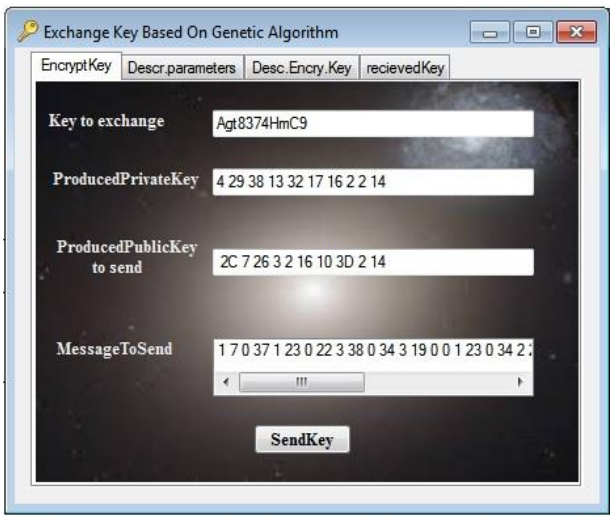

Figure.2 public and private key and the message to be sent to the sender side (receiver side)

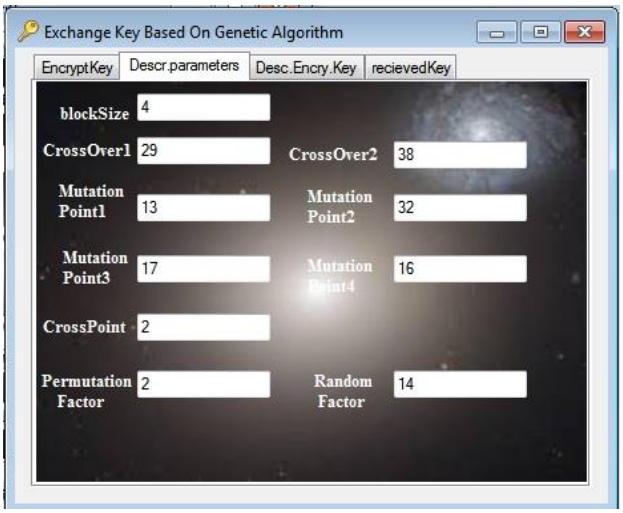

Figure.3 parameters which are generated internally (receiver side)

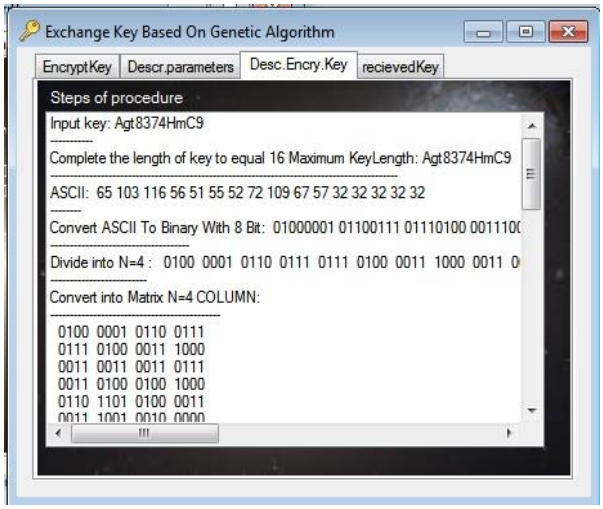

Figure. 4 the internal steps of the proposed system on the receiver side

On the Bob side, Bob would receive the encrypted key as shown in fig. 5; the steps to retrieve the original key are shown in fig. 6.

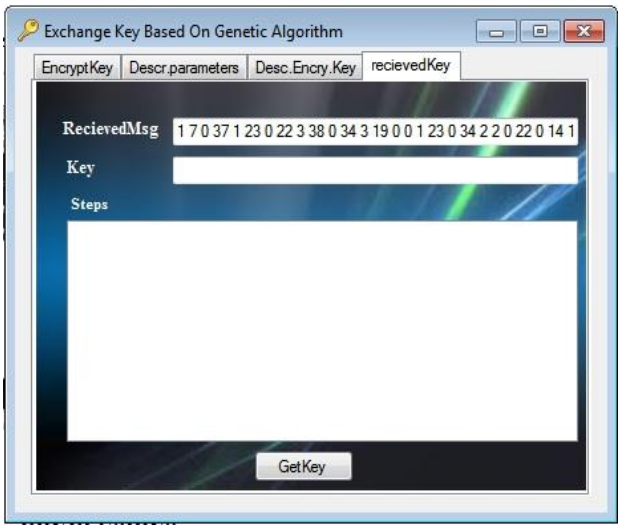

Figure.5: Received encrypted key(on the sender side)

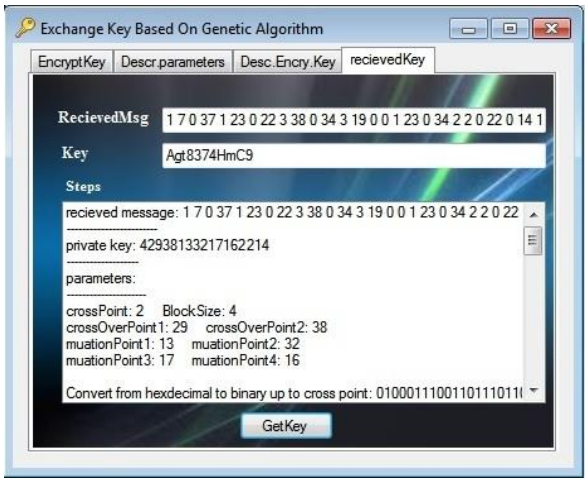

Figure 6: steps of retrieving original key

IV. CONCLUSION

In the present study a key exchange algorithm has been designed using the concept of genetic algorithms with the randomness and by the help of public key encryption. The algorithm solves the problem of key distribution and storing by using the genetic algorithm operators. The algorithm has been implemented using C#, and MATLAB is used as simulation platform. It has been tested and work successfully via network. This algorithm enhances the quality, efficiency and effectiveness of the algorithm being used for the cryptography and key exchange.

V. REFERENCES

[1] Handbook of Applied Cryptography, by A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996.

[2] Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C by Bruce Schneier Wiley Computer Publishing, John Wiley & Sons, Inc.

[3] Poonam Garg, "Genetic Algorithm, Tabu Search & Simulated Annealing Attack on Transposition Cipher", proceeding of third AIMS International conference on management at IIMA – 2006, 983-989

[4] Holland, J., "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, 1975.

[5] Alhussain Amanie Hasn "Cryptosystem for Providing Secured Application based on Genetic Algorithm" International Journal of Emerging Technology and Advanced Engineering Certified Journal, Volume 4, Special Issue 5, June 2014, 8-14; International Research Conference on Engineering, Science and Management 2014 (IRCESM 2014)