Asymmetric Key Encryption using Genetic Algorithm

Dr. Poornima G. Naik
Assistant Professor
Department of Computer Studies
Chh Shahu Institute of Business Education and Research, Kolhapur, India
Girish R. Naik
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 if the inputs are changed slightly or in the presence of a reasonable noise. GA offers significant benefits over other optimization techniques in searching a large state space or n-dimensional surface. In todays information age information sharing and transfer has increased exponentially. With the ever increasing growth of multimedia applications security has become an important issue in the communication of text and images. Encryption has extensive applications in preserving confidentiality of data in Internet applications. 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. There are many risks involved in communication of plain text over Internet. Cryptography is a technique of encoding and decoding messages so that they cannot be interpreted by anybody except the sender and the intended recipient. In this paper we have made an attempt to exploit the randomness involved in crossover and mutation processes for generating a asymmetric key pair for encryption and decryption of messages. The number of crossover points and number of mutation points together with permutation factor and random byte to be used in the generation of a private key dictate the length of the secret key and hence the strength of the algorithm. In the current work we have employed four crossover points, three mutation points and a single random byte and a permutation factor. The former three parameters are in the range 0-15 whereas the last parameter is generated in the range 1-7. For uniformity each parameter is represented using 4 bits. Hence the length of the key is 36 bits. The algorithm is further strengthened by making it difficult to break by permuting the asymmetric key by a predefined permutation factor agreed upon by both the sender and the intended receiver. The randomness together with permutation makes the algorithm robust and hard to break. Finally, the algorithm is implemented in Java and applied for the encryption and decryption of a text file and a Word Document. The methodology is general and can be applied to any file for secure transmission of data.

Keywords **– Asymmetric key, Cross-over, Cryptography, Decryption, Encryption, Genetic Algorithm, Mutation**,.

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 Darwinian idea of survival of the fittest and natural genetics. In a symmetric key encryption or secret key encryption only one key is used by both the sender and the intended receiver for both the encryption and decryption of the message. Both the sender and the intended receiver must agree upon the key before any communication begins. In the asymmetric key encryption, at the sender's end the public key/private key is used to encrypt the original message into a form known as a cipher text. At the receiver's end the corresponding private key/public key is used to decrypt the cipher text and restore a plain text from it. In practical situations, symmetric key encryption has number of problems. One such problem is that of key agreement and distribution which is overcome in asymmetric key encryption. The second problem is more serious. Since the same key is used for both encryption and decryption, one key per set of communicating parties is required. This limitation can be overcome by generating a key pair, a public key and a private key where a public key is freely distributed and the private key is kept confidential known only to the owner of the key pair. In this paper, we use Genetic Algorithm for generating an asymmetric key pair which is entirely a new approach and is not publicized like RSA and DES algorithms. Hence, even if the key becomes available to an unauthorized user it is difficult to break an algorithm. Figure. 1 depicts working of Asymmetric key encryption.

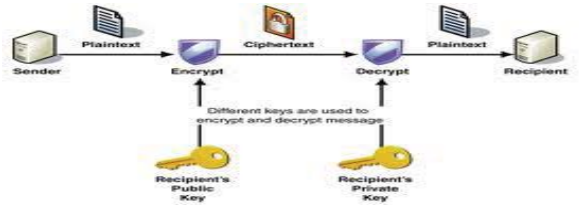

Figure 1. Working of Asymmetric Key Encryption

A. CRYPTOGRAPHY

Cryptography plays an important role in network security. Cryptography is the science of writing in secret code. The purpose of cryptography is to protect transmitted information from being read and understood by anyone except the intended recipient. In the ideal sense, unauthorized individuals can never read an enciphered message Cryptographic systems are generally classified among two independent dimensions.

- Types of Operations – All encrypted algorithms are based on two general principles, substitution and transposition. The fundamental requirements are that no information is lost and all operations are reversible.
- Key used – The length of the key determines the strength of the security.

In our current work, GA algorithm transfers 32-byte plain text blocks into 32-byte cipher blocks. Each parent is a 16-byte plain text block. On applying various crossover and mutation operations using a randomly generated asymmetric key the corresponding cipher child blocks are generated. The asymmetric key is generated randomly and consists of the following components.

- Four randomly generated cross-over points in the range 0-15
- Three randomly generated mutation points in the range 0-15.
- Randomly generated permutation factor in the range 1-7.
- A single random byte to be used in the generation of a private key

The strength of the key depends on the number of cross-over points and mutation points. The entire process is depicted in Figure. 2.

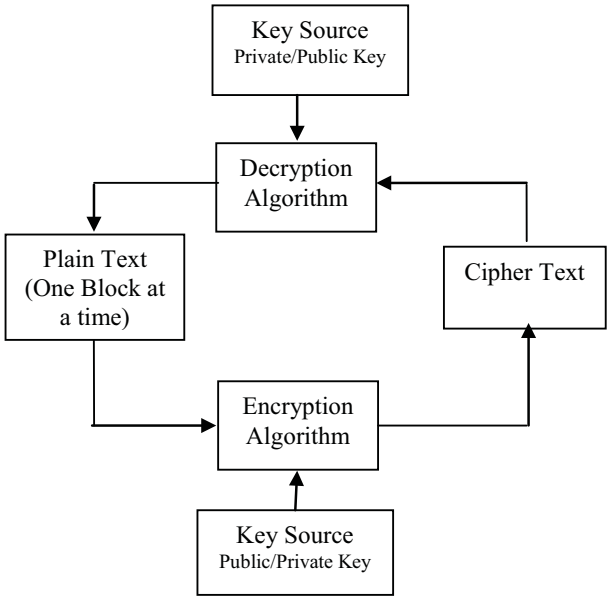

Figure. 2 Block Diagram of ASymmetric Key Encryption

B. GENETIC ALGORITHM

Generally, a Genetic Algorithm consists of three basic operations.

- Selection
- Crossover
- Mutation

The first step consists of searching individuals for reproduction. In our problem, we have selected two vectors of 16 bytes each as parents for reproduction. Since the problem is of encryption, there is no special preference given to any particular selection method. All the vectors are selected sequentially based on their order of appearance in a text file.

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 the role of recovering the lost genetic matter as well for randomly distributed 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 (128-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	128-b0	b1	b2	b3	b4	b5	b6	128-b7

The paper is organized as follows. The Section I gives an introduction to Genetic Algorithm and Cryptography under the heading of Introduction. Section II covers the literature survey and the current scenario of application of soft computing in implementing security. Section III focuses on the proposed method of asymmetric key encryption using Genetic Algorithm. Section IV covers implementation of the algorithm in Java. Finally, Section V is devoted for conclusion and scope for future enhancements.

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 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]. 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 [7, 9]. Mohammad Sazzadul Hoque et.al [10] have presented an intrusion detection system by applying GA to efficiently detect various types of network intrusions. They have used evolutionary theory to filter the traffic data and thus reduce the complexity [11]. There are several papers related to IDS all of which use GA in deriving classification rules [12, 14].

III. PROPOSED ALGORITHM

The pseudo code for encryption process using GA is given below.

Step 1 : Generate four crossover points and three mutation points in the range 0-15.

Step 2 : Sort the crossover points in ascending order.

Step 3 : Generate the random number in the range 1-7 used for the permutation of the key for encrypting subsequent blocks. This number is referred to as a permutation factor.

Step 4 : Generate a random number in the range 0-15. This random number is used in generating the corresponding private key in the key pair and is referred to as private random number.

Step 5 : Generate the public key based on crossover points, mutation points, permutation factor and private random number generated above.

Step 6 : Generate a private key based on the public key using the formula

$$K_{Private} = [\, K'_{Public}\ \theta\ R_p]' \qquad (1)$$

Where, R_p is a 36 bit binary no. generated by repeating private random number 9 times. Due to the symmetry of the operations involved and the symmetry of XOR operation, the corresponding public key can be generated from equ (1) and is given by

$$K_{Public} = [\, K'_{Private}\ \theta\ R_p]' \qquad (2)$$

Hence if the text is encrypted using public key then it can be decrypted using the corresponding private key and vice versa due to the reversibility of the operations involved.

Step 7 : Write the key pair to a file.

Step 8 : Read two blocks of 16 byte each from a text file.

Step 9 : If sufficient number of bytes are available then go to step 4.

Step 10 : Append the requisite no. of spaces so as to create even number of 16-byte blocks.

Step 11 : Apply translation on the blocks generated.

Step 12 : Perform the crossover and mutation operations on the blocks generated in step 11.

Step 13 : Write the encrypted blocks to a file.

Step 14 : If a text file contains more data then go to Step 8.

Step 15 : End

The working of the algorithm is illustrated for two 16-byte blocks extracted from a text file to be encrypted.

Step 1: Extract two 16-byte blocks from the text file to be encrypted. Let the two blocks be represented by

b0	b1	b2	b3	b4	b5	b6	b7	b8	b9	b10	b11	b12	b13	b14	b15

and

c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15

where each bi and ci is a character in a file.

Step 2 : Perform translation on selected blocks. Let the translated blocks be represented by replacing each bi by bi' and each ci by ci'.

Generate four random numbers in the range 0-15. Let the four random numbers generated be 2,7, 10 and 14 . These random numbers will serve as cross-over points.

Hence, Crossover Point1 = 2 Crossover Point2 = 7

Crossover Point3 = 10 Crossover Point4 = 14

Step 3 : Perform crossover operation.

Perform the crossover between two crossover points generated above.

b0’	b1’	b2’	b3’	b4’	b5’	b6’	b7’	b8’	b9’	b10’	b11’	b12’	b13’	b14’	b15’
c0’	c1’	c2’	c3’	c4’	c5’	c6’	c7’	c8’	c9’	c10’	c11’	c12’	c13’	c14’	c15’

The blocks after performing crossover operation are

b0’	b1’	b2’	**c3’**	**c4’**	**c5’**	**c6’**	**c7’**	b8’	b9’	b10’	**c11’**	**c12’**	**c13’**	**c14’**	b15’
c0’	c1’	c2’	**b3’**	**b4’**	**b5’**	**b6’**	**b7’**	c8’	c9’	c10’	**b11’**	**b12’**	**b13’**	**b14’**	c15’

Step 4 : Perform mutation operation.

Generate three random numbers in the range 0 to 15. These three random numbers serve as the mutation points for encryption. Let the three random numbers generated be 1,7 and 12.

Hence, Mutation Point1 = 1

Mutation Point2 = 7

Mutation Point3 = 12

Perform mutation operation on two blocks obtained in Step 2.

Parent Chromosome	b0’	**128-b1’**	c2’	c3’	c4’	c5’	c6’	**128-c7’**	b8’	b9’	b10’	c11’	**128-c12’**	c13’	c14’	b15’
Mutation Chromosome	0	**1**	0	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0
Child Chromosome	c0’	**128-c1’**	b2’	b3’	b4’	b5’	b6’	**128-c7’**	c8’	c9’	c10’	b11’	**128-b12’**	b13’	b14’	c15’

Step 5 : Generate the permutation factor randomly in the range 1-7 Let the permutation factor be 4.

Step 6 : Generate the random factor randomly in the range 0-15 Let the random factor be 9.

Step 7 : Generate a random key based on crossover points, mutation points, permutation factor and random factor generated above.
Hence the symmetric key in an hexadecimal form is

2	7	A	E	1	7	C	4	9

and in binary form is

0	0	1	0	0	1	1	1	1	0	1	0	1	1	1	0	0	0	0	1	0	1	1	1	1	1	0	0	0	1	0	0	1	0	0	1

Hence the length of the key is 36 bits in our case which depends on the number of crossover points and mutation points. The length of the symmetric key can be computed using a general formula

Key Length = 4 (c + m + 2)

Where c-> No. of Crossover points
m-> No. of Mutation points

Each hexadecimal digit in a asymmetric key can be represented using 4 bits. Hence a asymmetric key in a binary format is given by

In the above example,

c = 4
m = 3

Hence Key Length = 36.

The same operation is repeated for the next set of blocks after permuting the digits to the left of the permutation factor of a symmetric key by a permutation factor.

The new key generated is

1	7	C	2	7	A	E	4	9

Applying the same operation on the next two blocks , we get,

Step 1: Extract next two 16-byte blocks from the text file to be encrypted. Let the two blocks after translation be represented by

d0’	d1’	d2’	d3’	d4’	d5’	d6’	d7’	d8’	d9’	d10’	d11’	d12’	d13’	d14’	d15’

and

f0’	f1’	f2’	f3’	f4’	f5’	f6’	f7’	f8’	f9’	f10’	f11’	f12’	f13’	f14’	f15’

Step 2 : Perform crossover operation. New crossover points are,

Crossover Point1 = 1 Crossover Point2 = 7

Crossover Point3 = 12 Crossover Point4 = 2

Sorted crossover points are

Crossover Point1 = 1 Crossover Point2 = 2

Crossover Point3 = 7 Crossover Point4 = 12

d0’	d1’	d2’	d3’	d4’	d5’	d6’	d7’	d8’	d9’	d10’	d11’	d12’	d13’	d14’	d15’

f0’	f1’	f2’	f3’	f4’	f5’	f6’	f7’	f8’	f9’	f10’	f11’	f12’	f13’	f14’	f15’

The blocks after performing crossover operation are

d0’	d1’	**f2’**	d3’	d4’	d5’	d6’	d7’	**f8’**	**f9’**	**f10’**	**f11’**	**f12’**	d13’	d14’	d15’

f0’	f1’	**d2’**	f3’	f4’	f5’	f6’	f7’	**d8’**	**d9’**	**d10’**	**d11’**	**d12’**	f13’	f14’	f15’

Step 3 : Perform mutation operation. New mutation points are,

Mutation Point1 = 7 Mutation Point2 = 10 Mutation Point2 = 14

d0’	d1’	**f2’**	d3’	d4’	d5’	d6’	128-d7’	**f8’**	**f9’**	**128-f10’**	**f11’**	**f12’**	d13’	128-d14’	d15’

f0’	f1’	**d2’**	f3’	f4’	f5’	f6’	128-f7’	**d8’**	**d9’**	**128-d10’**	**d11’**	**d12’**	f13’	128-f14’	f15’

The process is repeated for all pairs of blocks in a text file.

IV. EXPERIMENTS AND RESULTS

A. IMPLEMENTATION IN JAVA

The cryptographic algorithm developed above is implemented in Java 1.7 which is used for encrypting/decrypting a text file. The work is further extended to encrypt the content Word document. The Word document is accessed in Java application using the following packages.

- org.apache.poi.hwpf
- org.apache.poi.hwpf.extractor
- org.apache.poi.hwpf.usermodel
- org.apache.poi.xwpf.usermodel

Files used in the program are shown in Table 1.

Table 1.Files used in the program

Filename	Description
ga.txt	Contains the text to be encrypted
Encrypt.txt	Contains the encrypted data
Decrypt.txt	Contains the decrypted data
KeyPair.txt	Contains randomly generated public key/private key pair.

Java source files generated during implementation are shown in Tab le 2.

Table 2. Java source files generated during implementation

Filename	Description
KeyPair.java	Used for generating public key/private key pair.
EncryptWord.java	Reads the contents of the word document, encrypts and writes back to the same file.
EncryptWord.java	Decrypts the contents of the Word file using the corresponding key in the key pair.

Swing GUI for Asymmetric Encryption using GA is shown in the Figure. 3.

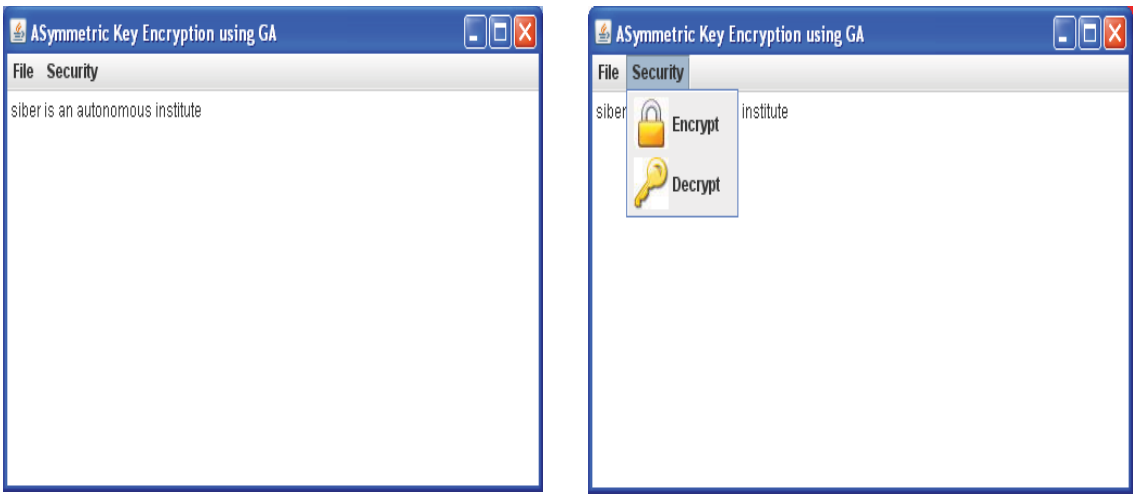

Figure 3. GUI for Asymmetric Encryption using GA.

The content of the encrypted file is shown in Figure. 4

Figure 4. Content of the Encrypted File.

The public key/ private key pair generated is shown in Figure 5.

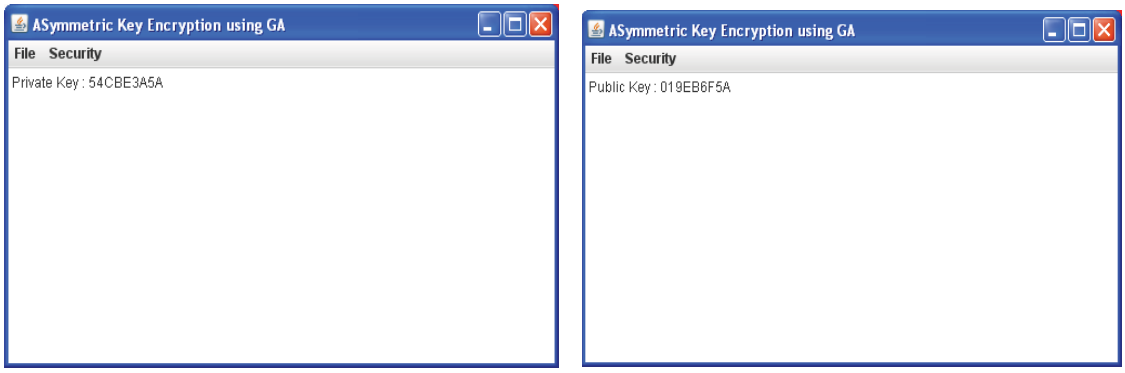

Figure 5. Public key/Private Key pair generated randomly.

The Word Document before and after encryption is depicted in Figure. 6 (a) and 6 (b), respectively.

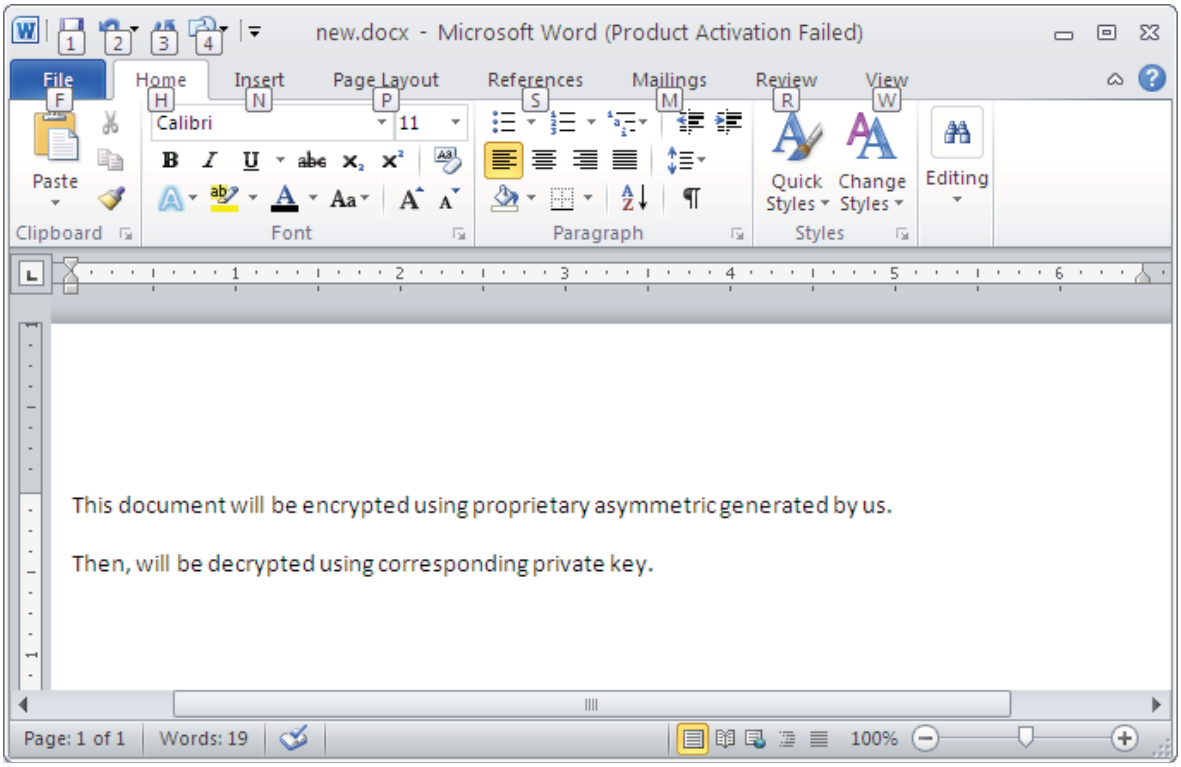

Figure. 6 (a) Word Document Before Encryption

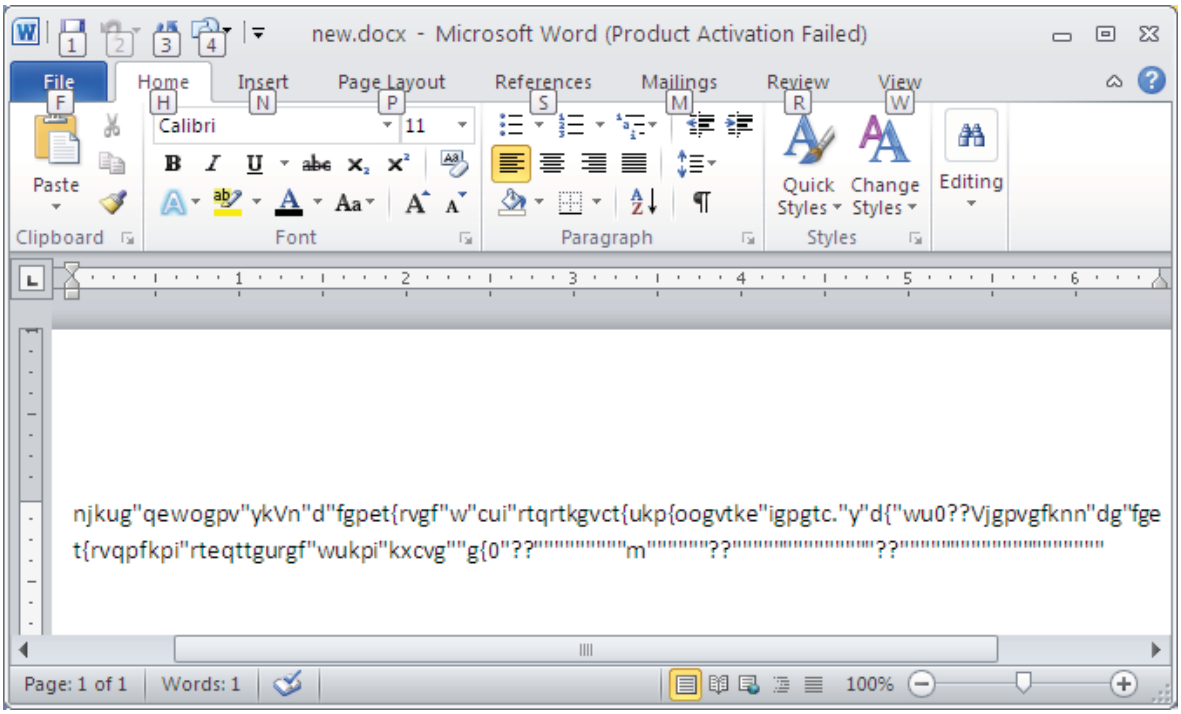

Figure. 6 (b) Word Document After Encryption

After decryption using the corresponding private key, the same plain text is restored back. The hexadecimal key pair used in the cryptographic process is shown in the Table 3.

Table 3. Cryptographic Key Pair.

Public key	014634D69
Private key	672052B69

V. CONCLUSION

In this paper we have proposed a new algorithm exploiting the randomness involved in crossover and mutation processes for generating a asymmetric key pair for encryption and decryption of messages. The number of crossover points and number of mutation points together with permutation factor and random byte to be used in the generation of a private key dictate the length of the secret key and hence the strength of the algorithm. In the current work we have employed four crossover points, three mutation points and a single random byte and a permutation factor. The length of the key is 36 bits. The algorithm is further strengthened by making it difficult to break by permuting the asymmetric key by a predefined permutation factor agreed upon by both the sender and the intended receiver. The randomness together with permutation makes the algorithm robust and hard to break. Finally, the algorithm is implemented in Java and applied for the encryption and decryption of a text file and a Word Document. Our future work consists of devising a formula to measure the strength of the algorithm using the displacement of each character in the original string.

REFERENCES

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

[2] Tragha A., Omary F., Mouloudi A., "ICIGA:Improved Cryptography Inspired by Genetic Algorithms", Proceedings of the International Conference on Hybrid Information Technology (ICHIT'06), pp. 335-341, 2006.

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

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

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

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

[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