International Journal of Mathematics and
Computer Science, **15**(2020), no. 4, 1109–1115

Key Generation for DNA Cryptography Using Genetic Operators and Diffie-Hellman Key Exchange Algorithm

E. Vidhya, R. Rathipriya

Department of Computer Science
Periyar University
Tamilnadu, Salem, India

email: vidhya11tamilarasi@gmail.com, rathipriyar@gmail.com

(Received July 9, 2020, Accepted August 28, 2020)

Abstract

Information security is one of the significant research challenges in the IT world. Most of the time, information is transmitted through a network that is not in a secured format. In this paper, the proposed work is to improve the critical strength of the DNA algorithm. The key strength is improved with the combination of the genetic algorithm and the Diffie-Hellman key exchange algorithm. The Shannon entropy measures the data compression while the critical entropy measures the key strength.

1 Introduction

Cryptography is a technique to secure the information in the transmit/storage stage. This technique contains two types: 1) Symmetric key cryptography, 2) Asymmetric key cryptography. The cryptography techniques transmit information with abundant space and with a less key strength. To overcome this problem, DNA cryptography is used which is one of the new techniques to secure information with less memory space. The base of DNA cryptography (Jash, 2018), (Kale, 2016) is DNA which is represented as a Deoxyribonucleic Acid. DNA cryptography secures information with DNA sequences which are A-Adenine, G-Guanine,

Key words and phrases: DNA cryptography, Genetic algorithm, The Diffie-Hellmen key exchange algorithm.
AMS (MOS) Subject Classifications: 11Z05.
ISSN 1814-0432, 2020, http://ijmcs.future-in-tech.net

C-Cytosine-Thymine (A, G, C, T). DNA cryptography stores a vast amount of information with the help of a complementary pair. So it takes less space. In DNA cryptography, 108 terabytes of information can be stored in 1 gram of DNA. In DNA Cryptography, the key generated (Souza, 2017) uses a Genetic algorithm. The keys are not in stable format because the genetic algorithm uses only an operator of crossover and mutation. The values are not in a random format. To overcome this problem, the Diffie-Hellman key exchange algorithm is used. In this algorithm, a key is generated with a mathematical form like a Discrete-Logarithm problem.

2 Related Works

This study focuses on DNA cryptography and the genetic algorithm methods show some of the famous works in this field. In the following table, the methods column describes the DNA cryptography which was used before. The Description column describes the methods used and the remarks column represents the pros and cons of the methods.

2.1 DNA Cryptography with Genetic Algorithm

In this paper, the authors describe a selective medical Image and encrypted the medical image using DNA cryptography and the dual hyperchaotic map and give a higher-level security. The proposed work is to reduce the computational time for the encryption of medical images. The DNA Cryptography techniques are applied to the cloud data and secure the data at a higher level. In DNA cryptography, the binary numbers are created with two algorithms. The first algorithm is to transmit the binary numbers to a DNA sequence using public-key encryption. The second algorithm is to create a binary number with an XOR, One-time pad cryptography This paper described DNA cryptography, and the key generated using a genetic algorithm. In the genetic algorithm, there are four operators in the four operators, the proposed used only two operators to generate a key. The key is in a stable format. The key is generated with a strong using the genetic algorithm and the N-W algorithm. A DNA cryptography processes the encryption and decryption. In the proposed work, the DNA cryptography is hybrid with the Genetic algorithm and the Diffie-Hellman key exchange algorithm which produces the strong encryption key. The proposed work is measured by an entropy and execution time and with a throughput.

3 Methodology

DNA SEQUENCES	A	G	C	T
BINARY VALUE	00	01	10	11

Table 1: DNA Sequence

Algorithm for Genetic Algorithm and Diffie-Helmen Key Exchange Algorithm

Algorithm 1: Genetic Algorithm	**Algorithm 2: Diffie-Helmen Key Exchange Algorithm**

1. Generate a random population with an 'n' number of chromosomes.
2. The fitness function f(x) evaluated for each chromosome, where 'x' represented as **a** population
3. The following steps create the new population. The steps repeated until the new population is complete.
  1. The two-parent chromosomes selected from the population according to their fitness value.
  2. If the crossover probability presented, the new offspring created else; it created a copy of the parent chromosome.
  3. The mutation is to swap the bits in the new offspring with a random position and generate another new offspring.
4. In the new population, the new offspring placed.
5. The new population used for the algorithm.
6. If the end condition becomes real, it stops the process, and the best solution is to return in the current population
7. If the end condition becomes false, Go to step 2.
	Input: Given a prime number and primitive root
	Output: Compute the key.
	Step 1: Input the large prime numbers as (n) and the primitive root as(g).
	Step 2: X_A has randomly selected a large positive integer by User A, which is less than n
	(X_A<n).
	Step 3: X_B has randomly selected a large positive integer by User B, which is less than n
	(X_B<n).
	Step 4: User A creates the public key
	$Y_A = (g^{X_A}) \bmod n$
	Step 5: User B creates the public key
	$Y_B = (g^{X_B}) \bmod n$
	Step 6: Public key of User A and User B has been exchanging through the insecure channel.
	Step 7: Compute User A Key.
	$K = (Y_B^{X_A}) \bmod n$
	Step 8: Compute User B Key.
	$K = (Y_A^{X_B}) \bmod n$

4 Proposed Work

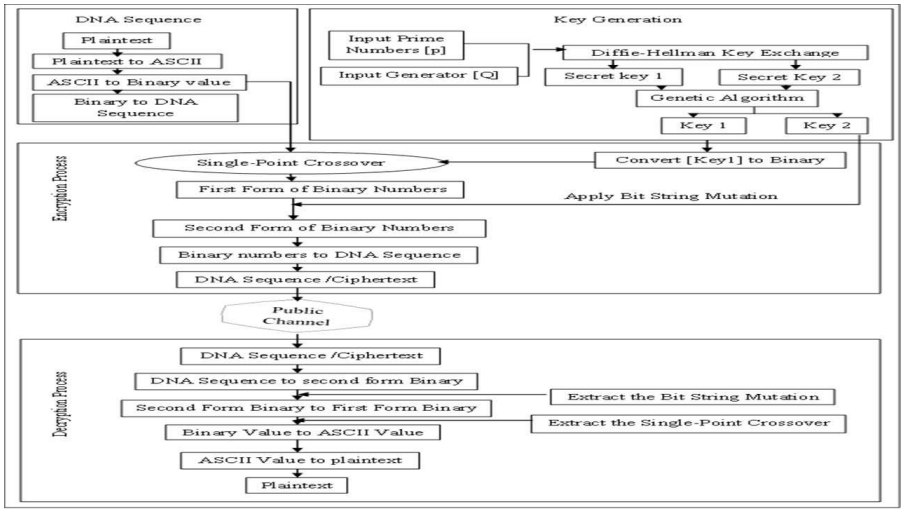

Figure 1: Proposed algorithm flowchart for Encryption and Decryption

4.1 Proposed Work algorithm for DNA Sequence and Key Generation

Phases I: DNA Sequence creation:
Begin

1. Read the plaintext as t_i where i >= 1 ***Example: DNA***
2. Split the plaintext into 'n' number of bits.
   n=1; st_i= (t_i/n); st_i= {t_1, t_2, t_3, …. ,t_i} ***Example: D N A***
3. Convert the split text to their related ASCII value as {a_1, . .,a_n}
   for i= 1 to n: a_i= ord(st_i); return { a_1, a_2, …., a_n}

Example: D= 68, N= 78, A= 65.

4. Convert the ASCII value to the binary values as {b_1, b_2, …, b_n}
   for i= 1 to n: b_i= (a_i)$_2$ return {b_1, b_2, ……, b_n}
   Example: 68= 01000100, 78= 01001110, 65= 01000001
5. Join all binary values into single form.
   for i= 1 to n: jo_i= " ".join(b_i); return {jo_1, jo_2, ……, jo_n}
   Example: 010001000100111001000001
6. Split the binary values into 'n' number of bits.
   n=2: sb_i= (jo_i%n); sb_i={sb_1, sb_2, ……, sb_n}
   Example: 01 00 01 00 01 00 11 10 01 00 00 01 1
7. Convert the binary numbers in to the DNA Sequence represented in an table 1.

```
da=[ ]; for n, i in (sbi): if i == '00': sb[n] = 'A'
        elif i == '10': sb[n] = 'C'
        elif i == '01': sb[n] = 'G'
        elif i == '11': sb[n] = 'T'
       da.append(sb) return{da}
```

Example: GAGAGATCGAAGC
End:

Phases II: Key Generation:
Begin:

1. In Diffie-Hellman key exchange Algorithm, Read the input value of prime numbers [p] and the generator [g].
2. Using the steps 2 to 8 given in algorithm 2. Create two secret keys as secret key1 and the secret key2.
3. Read the two Secret keys in to the Genetic Algorithm using the steps 3.1 and 3.2 in algorithm 1 and create two keys,
   3.1 The key 1 is used for a Single-Point crossover.
   3.2 The key 2 is used for a Bit-String Manipulation.
   Example: key1 = 43 and key2 = 4
4. Split the binary numbers into Phases I: step 5. by 'n' number of bits.

```
n=8
sb1= (jo%n)
print sb1.
```

   Example: 01000100, 01001110, 01000001
5. Split the sb1 into n bits.

```
n=4
sb11i= (sb1i%n)
print sb11i.
```

   Example: 0100 0100, 0100 1110, 0100 0001
6. Convert the First key into a binary value.
   fk=(secret key1)$_{10}$
   Example: 43=00101011

End

4.2 Proposed Work algorithm for Encryption and Decryption

Phases III: Encryption
Begin

Read the first key in algorithm 2: step6, Split the fk in to 'n' bits.
n=4; $fk1_i$= (fk/n); print $fk1_i$. ***Example: 0010, 1011***

Using the Single-Point Crossover, step 5 in Phases II and step 1 in Phases III swapped with an 'n' number of bits.
return as 'spc' ***Example: 01001011, 01001011, 01001011***

Read the key 2 value, in algorithm 2: step 3.
Example: Key2 =4

The Bit-String Mutation used. For every 'kth' position of the 'spc' the bit value is converted to another bit[0 to 1 or 1 to 0]. Return as 'nspc.'
Where k= key2.
Example: spc= 010010110100101101001011
Example: nspc= 010110100101101010110 10

Split the binary value into two bits.
n=2 ; da1= (nspc%n); print da1.
Example: 01 01 10 10 01 01 10 10 01 01 10 10

Convert the binary numbers into the DNA Sequence represented in table

```
fda=[ ]
for n, i in (da1):
   if i == '00': da1[n] = 'A'
   elif i == '10': da1[n] = 'C'
    elif i == '01': da1[n] = 'G'
     elif i == '11': da1[n] = 'T'
     fda.append(da1) return{fda}
```

Example: GGCCGGCCGGCC
End:

Phases IV: Decryption
Begin:

1. Read the DNA Sequence as fda.
   Example: GGCCGGCCGGCC
2. Convert the DNA Sequence to the binary values
   Example: 01 01 10 10 01 01 10 10 01 01 10 10
3. The Bit-String Mutation used, the bit extracted from the 'kth' position.
   Example: 010010110100101101001011
4. The single-point crossover used, the values are extracted and form an original binary value
   Example: 01000100, 01001110, 01000001
5. The binary value converted to ASCII value {a_1, a_2, …., a_n}
   Example: 01000100=68, 01001110=78, 01000001=65
6. The ASCII value converted to a text t_i
   Example: DNA

End:

5 Results

5.1 Experimental setup

The data run using an HPC. HPC means High-Performance computing most commonly refers to the performance of aggregating computing power in a method that delivers much-advanced performance than an individual can obtain out of a personal desktop computer or workplace in classify to solve significant problems in science, engineering, or business.

5.2 Shannon Entropy

$$H(X) = -\sum_{i=1}^{n} P(x_i \ \ )log_b P(x_i) \tag{1}$$

where P is the Probability of a given symbol and b is the base of the logarithm.

File Size in GB	DNA with Genetic	DNA with Genetic and Diffie-Hellman key exchange algorithm
24.1	1.764	1.695
5.43	1.999	1.548
26.7	1.580	1.580
46.5	1.964	1.711
87.0	1.850	1.850
87.6	1.865	1.749
94.3	1.799	1.478

Table 2 Entropy values for text files.

File Size in GB	DNA with Genetic	DNA with Genetic and Diffie-Hellman key exchange algorithm
120	1.620	1.854
190	1.599	1.412
24.8	1.765	1.393
33.5	1.765	1.382

Table 3 Entropy values for Image files.

5.3 Key Entropy

$$Throughput = \frac{\sum (input file)}{\sum (execution\ time)} \tag{2}$$

Key(bit)	DNA with Genetic (in bits)	DNA with Genetic and Diffie-Hellman key exchange algorithm (in bits)
24	4.857	7.507
32	7.057	11.754
40	9.614	12.462

Table 4 Key Entropy

6 Conclusion

The main objective of this paper was to improve the security level to information by using DNA cryptography with the Genetic algorithm and the Diffie-Hellman key exchange algorithm. The proposed work is to encrypt the data with a DNA sequence and with a Genetic algorithm and the Diffie-Hellman key exchange algorithm. The unauthorized person cannot reach the original information. This proposed work is to secure the text data and image data with a high-level of confidently by the measure of entropy values.

Acknowledgment. The authors would like to acknowledge and thank URF (University Research Fellowship) for supporting this work in the Department of Computer Science, Periyar University, Salem, Tamil Nadu, India.

References

[1] M. A. Ahlawat, A survey of DNA Based Cryptography. International Journal of Scientific Engineering and research, **3,** (2015), 132–134.

[2] O. A. AI-Harbi, Security Analysis of DNA based Steganography techniques, SN Applied Sciences, (2020).

[3] P. Akkasaligar, Selective medical image encryption using DNA cryptography, Information Security Journal: A global perspective, **29,** (2020), 91–101.

[4] G. Alexander, DNA Based cryptography and steganography, Global Research and Development Journal for Engineering, **2,** (2017), 249–253.

[5] Raj Anushree, DNA Cryptography algorithm using Genetic algorithm, International Journal of Latest Trends in Engineering and Technology,(2017), 34–39.

[6] D. H. Barnekow, DNA-Based watermarks using the DNA-Crypt algorithm, BMC Bioinformatics, (2007), 1–11.

[7] E. Vidhya, Two Level text Data Encryption using DNA Cryptography, International Journal of Computational Intelligence and Informatics, **8,** (2018), 106–118.

[8] E. Vidhya, Hybrid Key Generation for RSA and ECC. International Conference on Communication and Electronics Systems, (2019).

[9] J. P. George, An efficient 2-step DNA symmetric cryptography algorithm based on dynamic data structures, International Journal of Engineering and Technology, **7,** nos. 2-6, (2018), 141–146.

[10] X. Guozhen, The new field of cryptography: DNA Cryptography, Chinese Science Bulletin, **51,** (2006), 1413–1420.