ISSN: 2321-7782 (Online)
e-ISJN: A4372-3114
Impact Factor: 6.047

Volume 5, Issue 7, July 2017

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

3-Dimensional Bit Level Encryption Algorithm Version-2 (3DBLEA-2)

Dr. Asoke Nath[1]
Department of computer science
St. Xavier's College
Kolkata – India

Ayan Ghosh[2]
Department of computer science
St. Xavier's College
Kolkata – India

Enakshi Ghosh[3]
Department of computer science
St. Xavier's College
Kolkata – India

Jayisha Saha[4]
Department of computer science
St. Xavier's College
Kolkata – India

Abstract: ***In the last one decade many symmetric as well as public key encryption algorithms have been developed by the researchers in India and in abroad. The researchers developed several bit level encryption algorithms which are almost impossible to break without knowing the actual key and the actual procedure. There is a quite a number of research papers published on three dimensional encryption algorithm using DNA encryption and Genetic algorithms. These methods are quite complex to decrypt by using any kind of standard attacks such as Brute force attack method, known plain text attack method, statistical attack method and differential attack method etc. In the present study the authors have introduced a multilevel encryption algorithm which employs the use few bit level encryption techniques, transposition operations, DNA encryption and finally Genetic operators. It is a completely new idea which may be implemented to send any kind of confidential data over internet. The detailed operation is controlled by the algorithm and a key. This key is a secret parameter, ideally known only to the communicants for a specific message exchange context. The objective of this method is to evaluate the security of any confidential data. By using this algorithm one can encrypt any file such as .txt file, .doc file, .jpg file, .exe file, .wav file, .pdf file or any other file. After encryption or decryption the original size of the file will remain unaltered. A thorough investigation made on change in single bit in any position of the cipher text and it was found that decryption will not work. It is not possible to get back original plain text file if there is one change in bits in encrypted text. The testing is done on almost all types of files and it was found that the method is working satisfactorily.***

Keywords: ***Plain text, cipher text, encryption, decryption, key, Symmetric key, Genetic algorithm, DNA cryptography, Transposition, Mutation, Crossover.***

I. INTRODUCTION

The current scenario is such that the assurance of security in large open networks has become the need of the hour. With increase in the rate of crimes, one needs to take precautions to protect the data in an efficient manner from all possible attacks. This application plays an important role in providing security for military communications, financial transactions, corporate and political issues. Cryptography is one of the major concerned areas of computer and data security and a very promising direction in cryptography research is known as DNA Cryptography. DNA computational logic can be used in cryptography for encrypting, storing and transmitting the information, as well as for computation. Although in its primitive stage, DNA Cryptography is shown to be very effective. In this the concept of DNA is being used in the encryption and decryption process. The theoretical analysis and implementations shows this method to be efficient in computation, storage and transmission and it

is very powerful against certain attacks. This also proposes a unique cipher text generation procedure. In cryptography, cipher text is the result of encryption performed on plaintext using an Algorithm, called a cipher. Cipher text is also known as encrypted or encoded information because it contains a form of the original plaintext that is unreadable by a human or computer without the proper cipher to decrypt it. Decryption, the inverse of encryption, is the process of turning cipher text into readable plaintext. Substitution and transposition are ways of encrypting the plain text. Poly-alphabetic substitution is useful and is less vulnerable to cryptanalysis attacks-both Brute Force and Statistical attacks.

In Brute Force attacks, the cryptanalyst tries to break into the cipher text by trying out all possible keys on the cipher text to generate the plain text.

In Statistical attack, the cryptanalyst uses some inherent characteristics of the language to interpret the cipher text. For ex: if the language used is English language, then the most frequently appearing letter is the letter 'E' in an English writing.

Other types of attacks may be Known cipher text, Chosen Plain text, Chosen cipher text, Cipher text only.

In contrast, our algorithm proposes to find a solution to these problems of cryptanalysis attacks. We device the algorithm that converts the plain text file into bits and then uses encryption mechanisms on it. This provides the advantage of hiding the inherent characteristics of the language. Thus, even if the same characters appear in the plain text, using the same key also, the cipher text generated is unique in all cases.

II. Literature Survey

A. DNA Based Cryptography

DNA cryptography is a relatively new paradigm that has attracted great interest in the field of information security. DNA coding technology is used to convert binary data to DNA strings. Since scientists found that binary computers have many physical limitations, especially in data storage and computation, they have concentrated on DNA computers and tried to implement this new science in the information security field. DNA cryptography is a new concept that needs many improvements. Although there are still problems with DNA cryptography, many scientists are trying to solve them because they believe that, with the characteristics of DNA computers they have more advantages than conventional cryptography. DNA coding technology is another concept in cryptography that is intended to encode binary data to a DNA strand and vice versa. Binary data can be encoded in DNA by using sequence of alphabet. It is known that DNA sequences contain four basic letters Adenine (A), Cytosine (C), Guanine (G) and Thymine (T).

00 is converted to A. 01 is converted to C.

10 is converted to G. 11 is converted to T.

For example, a binary string like '00011011' is converted to 'ACGT'. Here the authors have first implemented few bit level encryption techniques and then they converted the bits to DNA sequences to apply genetic operations like crossover and mutation.

B. Genetic Algorithm

In computer science and operational research, a genetic algorithm (GA) is a meta-heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection. The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations,

the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear.

A meta-heuristic is a higher level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity. The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

- *Selection rules* select the individuals, called *parents* that contribute to the population at the next generation.
- *Crossover rules* combine two parents to form children for the next generation.
- *Mutation rules* apply random changes to individual parents to form children

Here in our algorithm we have used only crossover and mutation operator.

1. *Crossover*

After constructing the chromosome population, the next operation is crossover. These may be sequentially used in all the rounds. In the first one, the parents are selected in the mating pool. A single crossover point is selected between the entire length of the parents' chromosomes, thereby creating two new offspring by exchanging the head of parent1 and parent2. The first offspring is created by concatenating the head of parent1 and the tail of parent2 and the second offspring is created by concatenating the head of parent2 and the tail of parent1. Consequently each offspring contains portions of the DNA codes of both parents.

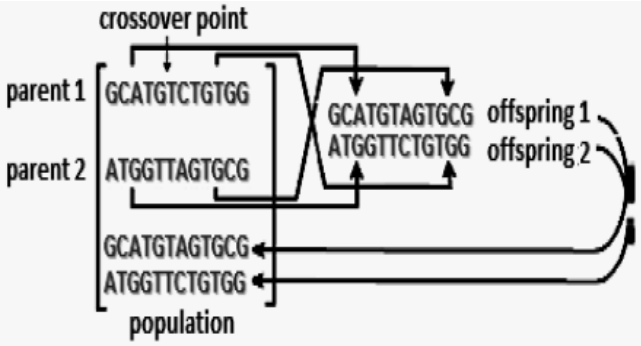


Fig1: Two parents' crossover to produce two offspring
Reference: DNA genetic encryption technique, Hamdy M. Mousa, I. J. Computer Network and Information Security, 2016, 7, 1-9

2. *Mutation*

After crossover process, mutation process is applied on the chromosome population. Mutation is the alteration of string elements. Two types of mutation are used. In the first one, two mutation points are selected between the entire length of the bit string. Then the bits in between these two points are complemented that is, single point mutation changes a 1 to a 0, and vice versa.

Example: Before mutation:

…… 11 0110 0100 1001 0010 11 …….

After mutation:

…… 11 0101 1011 0110 1010 11 ……..

In the second mutation type, four bits are converted to two bases of DNA (1010 -> CG).

TABLE I Mutation table

DNA	Bits	DNA	Bits	DNA	Bits	DNA	Bits
TA	0000	**GA**	0100	**CA**	1000	**AA**	1100
TC	0001	**GC**	0101	**CC**	1001	**AC**	1101
TG	0010	**GG**	0110	**CG**	1010	**AG**	1110
TT	0011	**GT**	0111	**CT**	1011	**AT**	1111

Example:

Before mutation:-

…… G G A C T G C G A T ……

After mutation:-

…… A A G T C A T A G C ……..

Here each DNA base is treated as two bits and the first bit is complemented for alter mutation. Thus G=’10’ is mutated to ‘00’=A and vice-versa.

III. Encryption Algorithm

Step-1: Input Plain text file.

Step-2: Convert Plain Text file to Bits.

Step-3: Input key from user. This key can be any text key that is randomly input.

Step4: Generate the key which will be later used in transposition methods.

Let the key be=”ABCD”.

After conversion of ASCII codes of secret key to bits=

01000001 01000010 01000011 01000100 which is 32bits in length.

Prime numbers from 1 to 32 are-2,3,5,7,11,13,17,19,23,29,31 which are used later as bases.

Position of 1’s in the secret key bits to be used as exponents:

2,8,10,15,18,23,24,26,30.

Sum(s)=$2^2+3^8+5^{10}+7^{15}+11^{18}+13^{23}+17^{24}+19^{26}+23^{30}+29^2+31^8$

=710943505599442715264808072267741784274 4.

Column, row and depth key selection for transposition method

From this sum the key for column, row and depth wise transposition can be found:

S=710943505599442715264808072267741784274 4

1. **Column/Row:** Let the column/row number be 6. (0 to row/column of plain text).

Then key for column and row transposition is 1, 0, 4, 3, 5, 2, and 6 (this key is selected from the sum as the digits appear in order).

2. **Depth:** Let the depth number be 12. (0 to depth of plain text).

Then the key for depth transposition is 7,1,0,9,4,3,5,2,6,8. But in the sum there is number 10, 11, 12. So we add these two numbers 10, 11, 12 in the key. Thus the key will be 7,1,0,9,4,3,5,2,6,8,10,11,12

Step5:- Calculate the size of the bit pattern of plain text.

Step6:- Complement the prime position bits of the entire bit stream. For example:- Bit stream 0111100110101100.

Prime bit positions 2, 3,5,7,11,13

Bits in these positions are 1, 1, 1, 0, 1, 1

Complemented bit stream 0001001110000100.

Step7:- Reverse the entire bit pattern.

Step8:- Again complement the prime positions.

Step9:-Perform bitwise XOR operation on bit1 and bit n and substitute in n^{th} bit position.

Step10:- Leave the first n/2 bits as it is.

Step11:- Repeat step 9 and 10 till all bits are exhausted.

Step12:- Store the bits into two dimensional array of size n*n. (Here n is chosen to be the nearest perfect square lesser than the size of the bit stream). And residual bits are stored into a one dimensional array. Shifting operations are performed on the two dimensional array.

Step13:- Perform bitwise left shift. This is done by shifting all bits in each row by one unit on the left side.

Step14:- Perform bitwise up shift. This is done by shifting all bits in each column by one unit in upward direction.

Step15:- Perform bitwise diagonal shift. This is done by shifting all bits in each of the two diagonals by one unit.

Step16: Perform bitwise right shift. This is done by shifting all bits in each row by one unit to right.

Step17: Perform bitwise downshift. This is done by shifting all bits in each column by one unit in downward direction.

Step18: The residual bits from the 1D array are shifted into the 2D array and the same numbers of bits are shifted from the 2D array to the 1D array.

Step-19: Convert 2 consecutive bits of Plain text file to DNA sequence. These bits are then taken into groups of 2bits at a time and converted to DNA sequence. Two bits can have any of the following configurations: 00->'A', 01->'C' 10->'G' 11->'T'. By this conversion, we manage to reduce the size of the file to some extent.

Example- Binary form of secret data:

….00 01 10 01 10 11….

Converted DNA sequence: ….A C G C G T….

Step20:- Calculate the row, column and depth for 3dimensional transposition method.

For Example:

Let the plain text bits be = 010000110100000101000010010001010100010001000101011000010110001101100100

Then the DNA sequence will be:

CAATCAACCAAGCACCCACACACCCGACCGAGCGCA

Size of DNA sequence= 36

Let n=36/2=18 (since all factors of 36 lie between, 1 to 18).

Calculate all perfect square from 1 -18

Here the perfect squares are 4, 9 & 16.

Then store the perfect squares into an array and take the middle position of the array.

Here we take 9.so column=3 & row=3

Depth= size of plain text bit stream/ (column*row) =36/9=4

Now, the DNA sequence are arranged in a 3D array with column=3, row=3 and depth=4.

Step 21:- From the sum s, the key for column, row and depth wise transposition is again calculated:

S=710943505599442715264808072267741784 2744

Column/Row: Let the column/row number be 3. (0 to row/column of DNA sequence).

Then key for column and row transposition is 1, 0, 3, and 2 (this key is selected from the sum as the digits appear in order).

Depth: Let the depth number be 4. (0 to depth of DNA sequence).

Then the key for depth transposition is 1, 0, 4, 3, and 2.

Step22:- Perform character wise columnar transposition on the DNA sequence.

Step23:- Perform character wise row transposition on the DNA sequence.

Step24:- Perform character wise depth transposition on the DNA sequence.

Step25:- Perform crossover operations.

Here two rows are selected from the DNA sequence encrypted 3D array called parent chromosomes. A single crossover point is selected and each parent chromosome is identified by a HEAD (H) and a TAIL (T). Two offsprings are formed by combining the Head of first parent with the Tail of other parent and the Head of the second parent with the Tail of the first parent respectively.

For Example: let the two parent chromosomes be ACA and GTA.

Let the crossover position be 2.

Offspring1 =ATA.

Offspring2=GCA.

Step 26:- Perform mutation operations.

Here each DNA base is treated as two bits and the first bit is complemented for alter mutation. Thus G=’10’ is mutated to ‘00’=A and vice versa.

For example: if the DNA sequence is ATA, A is complemented to G and T is complemented to C.

ATA is mutated to GCG.

Step27:- This encrypted DNA string is then converted to bits.

For example:-…..ATAGCACCTAACGGTATCGG…..

Gets converted to

…..00110010010001011100000110101100110110 10….

Step 28:-This bit pattern is finally converted back to bytes. This file is the cipher text obtained from the input plain text.

IV. CONCLUSION

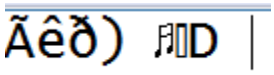
In this decade information is being considered to be most valuable entity. So the proper protection of information is highly needed. Huge amount of sensitive information are stored in computer which are now available and accessible through internet. As more information is made available electronically, it can be assumed that threats and vulnerabilities to the integrity of that information will increase as well. We need to protect our vital information from adversaries or any person who may use our important data to benefit them directly. Using this algorithm we secure our information so that anybody cannot hack it easily. This method can be applied on various types of files such as .txt, .doc, .exe etc. In the present method encrypted text cannot be decrypted without knowing the initial key. The present method is free from any kind of brute-force attack or known-plaintext attack.

If the encrypted text gets modified by the attacker by just one bit then he or she will not be able to get the proper original plain text as the algorithm is sensitive.

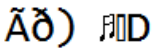
TEST CASE:-

Input plain text- Good Day.

Cipher text-



Change in cipher text by attacker-



Change in plain text- 

By the above example we can understand how much sensitive our algorithm is.

Every application has its merits and demerits. The project has covered almost all requirements. Further requirements and improvements can easily be done since coding is mainly structured or modular in nature. By implementing another layer into the transposition method we can extend this algorithm to 4 dimensions. The present algorithm can be applied to files like .txt, .png, .jpg, .dll, .exe etc. But the authors have implemented the method in basic text files and the results were quite satisfactory .Currently it is not feasible to use this method for large files as execution time is high.

ACKNOWLEDGEMENT

Apart from the efforts of us, the success of any work depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this paper. We are very grateful to Dr. Asoke Nath who is our mentor and guide for giving us this pristine idea of data encryption.

References

1. Asoke Nath, Madhumita Santra, Supriya Maji and Kanij Fatema Aleya,"3-Dimensional Bit Level Encryption Algorithm Ver-1(3DBLEA-1)". International Journal of Innovative Research in Computer and Communication Engineering(IJIRCCE), Vol. 4, page: : 8611-8618 Issue 5, MAY 2016.
2. Hamdy M. Mousa, "DNA-Genetic Encryption Technique". I.J. Computer Network and Information Security, Vol-7,2016.
3. https://docs.oracle.com/javase/8/docs/technotes/guides/install/windows_system_requirements.html,17/5/2017, accesstime: 11:15
4. Behrouz A. Forouzan, "Cryptography and Network Security", Special Indian edition 2007, Tata Mc-Graw Hill publishing company limited, pages - 2-13 and 56-58
5. William Stallings, "Cryptography and Network Security Principles and Practices", Fourth edition 2005, Prentice Hall publishers

6. Asoke Nath, Saima Ghosh, Meheboob Alam Mallik, "Symmetric Key Cryptography using Random Key generator", Proceedings of International conference on security and management (SAM-10) held at Las Vegas, USA, July 12-15, 2010, Vol-2, Page: 239-244(2010).
7. Dripto Chatterjee, Joyshree Nath, Suvadeep Dasgupta and Asoke Nath , "A new Symmetric key Cryptography Algorithm using extended MSA method: DJSA symmetric key algorithm", Proceedings of IEEE International Conference on Communication Systems and Network Technologies, held at SMVDU(Jammu) 03-06 June,2011, Page-89-94(2011).

