International Journal of Latest Trends in Engineering and Technology
Vol.(10)Issue(2), pp.340-346
DOI: http://dx.doi.org/10.21172/1.102.55
e-ISSN:2278-621X

3-DIMENSIONAL BIT LEVEL ENCRYPTION ALGORITHM VERSION-3 (3DBLEA-3)

Asoke Nath[1], Soumyadip Ray[2], Salil Anthony Dhara[3], Sourav Hazra[4]

Abstract- Research has developed several bit level encryption algorithms which are near impossible to break without having the actual key and knowing the actual procedure. There are quite a number of research publications on three dimensional encryption algorithm using DNA encryption and Genetic algorithms. These methods are quite complex to decrypt by using any known standard attacks such as Brute force, statistical and differential attack methods and such. In this paper the authors have introduced a multilevel encryption algorithm which uses some bit level encryption techniques, transposition operations, DNA encryption and finally genetic operations. Also, the algorithm takes the help of asymmetric key cryptographic algorithms to transfer the key .It may be implemented to encrypt and send any kind of confidential data.
Keywords – Plain text, Cipher text, Genetic Algorithm, DNA cryptography, Transposition, Mutation, Message Digest, Crossover.

1. INTRODUCTION

After its human resources, information is an organization's most important asset. Security and risk management is data centric. All efforts to protect systems and networks attempt to achieve three outcomes: data availability, integrity, and confidentiality. No infrastructure security controls are 100% effective. Thus layered security models are used, where one final prevention control wrapped around sensitive information: encryption. Achieving strong encryption, the hiding of data's meaning, also requires intuitive leaps that allow creative application of known or new methods.

DNA cryptography is one of the rapid emerging technology which works on concepts of DNA computing. A new technique for securing data was introduced using the biological structure of DNA called DNA Computing (aka molecular computing or biological computing). It was invented by Leonard Max Adleman in the year 1994, for solving the complex problems such as directed Hamilton path problem, NP-complete problem similar to The Travelling Salesman problem. DNA can be used to store and transmit data. The concept of using DNA computing in the fields of cryptography and steganography has been identified as a possible technology that may bring forward a new hope for unbreakable algorithms. Strands of DNA are long polymers of millions of linked nucleotides. These nucleotides consist of one of four nitrogen bases, a five carbon sugar and a phosphate group. The nucleotides that make up these polymers are named after the nitrogen base that it consists of; Adenine (A), Cytosine (C), Guanine (G) and Thymine (T).

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, the proposed algorithm proposes to find a solution to these problems of cryptanalysis attacks. The proposed algorithm 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.

Also, for ensuring the integrity of data, we have included the concept of Message Digest. This will enable the receiver to know if some attacker in the middle has tampered with the message or not.

[1] Department of Computer Science, St. Xavier's College(Autonomous), Kolkata, West Bengal, India
[2] Department of Computer Science, St. Xavier's College(Autonomous), Kolkata, West Bengal, India
[3] Department of Computer Science, St. Xavier's College(Autonomous), Kolkata, West Bengal, India
[4] Department of Computer Science, St. Xavier's College(Autonomous), Kolkata, West Bengal, India

1.1 DNA Cryptography

It is a new technology that attracted the attention of many programmers and developers in the field of information security. DNA coding aims at converting a binary data into DNA string. Due to some limitations in data storage and computing power many scientists have adopted DNA computers. Although problems are there in this technology scientist are trying to solve these problems because they believe DNA cryptography will be much more effective and efficient than the traditional cryptography. 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 some bit level encryption techniques are applied and then genetic operations like mutation, crossover, selection by converting the bits into DNA string are applied.

1.2. Genetic Algorithm

Genetic algorithms (GAs) are a meta-heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). GAs are used to produce high quality results in an optimization and search based problems with the help of some bio-inspired operators like mutation, crossover, 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. In GAs, we have a pool or a population of possible solutions to the given problem. These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations. Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals. This is in line with the Darwinian Theory of "Survival of the Fittest". Some of the terminologies related to the genetic algorithms are as follows: -

a. Population: It is defined as the subset of all the possible solutions for a given problem.
b. Chromosomes: It is one of the solutions to the given problem.
c. Gene: One element position of a chromosome.
d. Allele: It is the value a gene takes for a particular chromosome.

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.

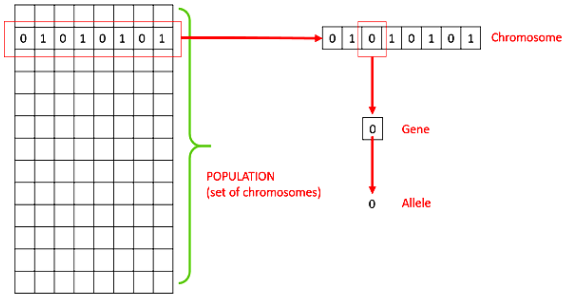


Fig 1

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 three rules that are used in genetic algorithm are as follows: -

Selection rule: This rule selects two individuals called as parents who will lead to the next generation by mating and recombining. The selection should be efficient as better parent will lead to a better solution.

Crossover rule: Process of combining two parents to produce two off springs. 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.

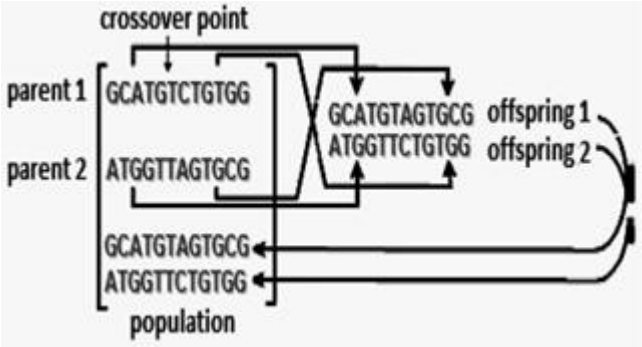


Fig 2: Two parents crossover to produce two off springs

Mutation rule: Application of some changes to the chromosomes to form a new one. 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. E.g.

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.

1.3 Message Digest
Message digests are designed to protect the integrity of data or media to detect changes and alterations in them. They are a type of cryptography which utilize hash values that can warn the copyright owner of any modifications that may have been applied to their work. The particular message digest will change if any changes are made to the file. Not only can message digests help to determine modifications in files, but it can also assist in locating duplicate files. File sharing programs, such as peer-to-peer (P2P), utilize the message digests. Message digests are encrypted with private keys which create a digital signature. This results in a type of validation which ensures that the appropriate user is accessing protected information. Message digests protect one-way hash algorithms by taking random data and transmitting a set length hash value. To begin the process a message digest is initialized, then the data is processed through the message digest by using updates. Final operations include padding, during which the message digest completes the hash computation and resets itself.

2. PROPOSED 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. It can also be generated from the plain text itself.
Step 4: 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)=22+38+510+715+1118+1323+1724+1926+2330+292+318
=710943505599442715264808072267741784 2744.
Step 5: Calculate the message digest. For this find the sum of all the digits in the sum as 7+1+0+9+4+3+……...=1472
Now find the modulus mod(sum, length of the sum/2)=112.
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=710943505599442715264808072267741784 2744
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).
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
Step 6:- Calculate the size of the bit pattern of plain text.
Step 7:- 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.
Step 8:- Reverse the entire bit pattern.
Step 9:- Again complement the prime positions.
Step 10:-Perform bitwise XOR operation on bit1 and bit n and substitute in nth bit position.
Step 11:- Leave the first n/2 bits as it is.
Step 12:- Repeat step 9 and 10 till all bits are exhausted.
Step 13:- 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.
Step 14:- Perform bitwise left shift. This is done by shifting all bits in each row by one unit on the left side.
Step 15:- Perform bitwise up shift. This is done by shifting all bits in each column by one unit in upward direction.
Step 16:- Perform bitwise diagonal shift. This is done by shifting all bits in each of the two diagonals by one unit.
Step 17: Perform bitwise right shift. This is done by shifting all bits in each row by one unit to right.
Step 18: Perform bitwise downshift. This is done by shifting all bits in each column by one unit in downward direction.
Step 19: 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 20: Convert 2 consecutive bits of the bit pattern obtained by concatenating the bits form the 2D array followed by 1D residual array 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….

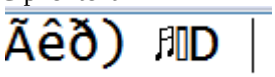
Step 21:- Calculate the row, column and depth for 3dimensional transposition method.
For Example:
Let the plain text bits be = 0100001101000001010000100100010101000100010001010110000101100011011001 00
Then the DNA sequence will be:
CAATCAACCAAGCACCCACACACCCGACCGAGCGCA
Size of DNA sequence= 36
Let n=36/2=18 (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 22:- 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.
Step 23:- Perform character wise columnar transposition on the DNA sequence.
Step 24:- Perform character wise row transposition on the DNA sequence.
Step 25:- Perform character wise depth transposition on the DNA sequence.
Step 26:- Perform crossover operations.
Here two random 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. The selected random rows are saved.
For Example: let the two parent chromosomes be ACA and GTA.
Let the crossover position be 2.
Offspring1 =ATA.
Offspring2=GCA.
Step 27:- 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.

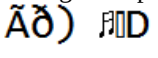
Step 28:- This encrypted DNA string is then converted to bits.
For example:-…..ATAGCACCTAACGGTATCGG…..
Gets converted to
…..00110010010001011100000110101100110110 10….
Step 29:-This bit pattern is finally converted back to bytes. This file is the cipher text obtained from the input plain text.
Step 30: Take the message digest obtained in Step 4 and convert it into its binary equivalent.
Step 31: Perform a bitwise right shift on the binary equivalent of the message digest.
Step 32: Perform bitwise XOR between the consecutive bits of the output of Step 31 .
Step 33: Perform a right shift on the output of Step 32.
Step 34: Embed the encrypted message digest in the middle position of the encrypted plain text.
Step 35: This will give the final cipher text with message digest encrypted in it.
Step 36: Allot two random large prime numbers to the sender and the receiver. Let the numbers be x and y. Calculate ‘n’ which is some function of x and y. Thus, n=f(x, y).
Step 37: Calculate the multiplicative inverses in Zn.
Step 38: Choose any one of the multiplicative inverses.
Step 39: Perform an XOR operation of the transposition and selection key with the selected multiplicative inverse.
Step 40: Send the selected multiplicative inverse and the encrypted transposition and selection keys to the receiver.
For decryption just the opposite of the above given procedure is to be followed. The message digest is to be extracted from the cipher text and saved so that it can used later to verify the integrity of the message. For decryption of the two keys, one for transposition and the other for selection, calculate the multiplicative inverse of the received key.

3. RESULTS AND DISCUSSION

3.1 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.

Since the algorithm uses random selection each time the encryption operation is performed different results are obtained as follows:
Input plain text: My name is Soumyadip
I am a Xaverian
I live in Kolkata
I am a Bengali

Output cipher text when run for the first time:뷔땀詖□韛趧둲鉴隴ฅ핺ʒ 憘萿珩액鞅騡쵞趦붆鞥爚□땞ᆙ 徶葠ҁ□銹v̇ 蘧옏閾敼뉯훰酥趴□홑鍩誟□傷핵ꯍ ?□?趦刄韛爔驦誄□휥?훙降陫
Output cipher text when run for the second time:뷡떡詖雽韛趑□鍾隱ฅ휼ㄅ輓篐玽햪鞥褁?越刄鞥爪馆뗭ᆕ 嵢靇ᄆ□錗□藏閰□ᄀ 휼蝕趦랒휠녙誅□容핵₵ ?⟵?訏賫鞦爩覥誄靠훈?핵?陫
Output cipher text when run for the third time:
뷘艶詹雥鞥 & 趧둲鉴隴ꦙ휼□鞄籍珩햊輓桲趦迋韛爨□땞ᆗ 嶢□ᄆ陏鋧□蘧흸閰□ᆃ 휩酥趧蛻휠酙誶□핵₵ ?بحجي越軻鞶
爔驦諺華훌?홨?陌
Now let’s demonstrate the fact with another example:
Input plain text: AAAAaaaa
Output cipher text when run for first time:몺刨몺刨駺魄
Output cipher text when run for second time:몺慤랢僨魄홱驫
Output cipher text when run for third time:랬战뫕戩鷘훑魄
Thus, on carrying such tests, it has been found that there is approximately 11.2 % chance that two same characters in the plain text are replaced by the same cipher text character. Also, it has been found that on applying the encryption process repeatedly, there is almost 6.39% chance that the same plain text character is replaced by the same cipher text character.
This algorithm is applicable to all kinds of file formats. For .txt files, the file can be directly input into the encryption processes however for different file types, each file type requires its own preprocessing step.
To encrypt image files, first the bitmap of the image is extracted. Then, the FFT(Fast Fourier Transform)is applied to the image, and the discrete values of the image is computed. In the next step each of these values is passed through the encryption process. The values are then stored as a text file in the server. During the decryption process, the values are extracted from the text file, decrypted and then IFFT(Inverse FFT) is performed to get back the original image. It must be noted that the image height has to be a power of 2.In cases where this is not possible, the values closest to(less than) the height but of power of 2 is taken ,i.e. if the height is less than 27 then height is considered as 26.The rest of the values are kept unchanged during the encryption and decryption process.
To encrypt files of .pdf, the bitmap of the file is extracted. During the encryption process these values are encrypted and stored in a text file in the server. For decryption, this file is decrypted to get the original bitmap version of the initial file.

4. CONCLUSION AND FUTURE SCOPE

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.

5. ACKNOWLEDGEMENT

The authors are very much grateful to Prof. Shalabh Agarwal, Head, Department of Computer Science, St. Xavier’s College(Autonomous) to give opportunity to carry out this project work. Without his support it would not been possible to complete this work. One of the authors AN is grateful to Dr. Dominic Savio, Principal , St. Xavier’s College(Autonomous), Kolkata for his all time support to do research work in Network security and Cryptography.

6. REFERENCES

[1] Asoke Nath, Ayan Ghosh, Enakshi Ghosh and Jayisha Saha,”3-Dimensional Bit Level Encryption Algorithm Ver-2(3DBLEA-2)”. International Journal of Innovative Research in Computer and Communication Engineering(IJIRCCE), Vol. 4, page: : 8611-8618 Issue 5, MAY 2017.

[2] 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.

[3] Hamdy M. Mousa, "DNA-Genetic Encryption Technique". I.J. Computer Network and Information Security, Vol-7,2016.
[4] https://docs.oracle.com/javase/8/docs/technotes/guides/install/windows_system_requirements.html,17/5/2017, accesstime: 11:15
[5] Behrouz A. Forouzan, "Cryptography and Network Security", Special Indian edition 2007, Tata Mc-Graw Hill publishing company limited, pages - 2-13 and 56-58
[6] William Stallings, "Cryptography and Network Security Principles and Practices", Fourth edition 2005, Prentice Hall publishers
[7] 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).
[8] 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).