A Study of Encryption and Decryption Technique Using Genetic Algorithm

Lalit Chaurasiya, Manoj Ughade & S. S. Shrivastava
Department of Mathematics
Institute for Excellence in Higher Education, Bhopal (M.P.)

ABSTRACT

The purpose of this paper is to introduced an encryption and decryption algorithm that uses Genetic algorithm to improve data communication security and efficiency. The technique adds complexity and randomness to the cryptographic process by combining XNOR-based logical transformations with one-point and two-point crossover operations because it transforms bits in relation to another string of binary data (the key) that bears no relation to the primary data. This fact leads to considerable ambiguity on the part of attackers trying to guess the value of the key or the plain text, making the system more robust against security attacks.

Keywords: Genetic algorithm, XNOR cipher, Encryption and Decryption, one-point and two-point crossover.

I. INTRODUCTION:

Protection against data transmission is now a necessity because of the growth in digital communication and the growing threat of unauthorized access. On mathematical grounds, cryptography defends information primarily by encrypting it (making it unreadable form) and decrypting it (restoring its original state). Cryptographic algorithms are typically divided into two categories: symmetric and asymmetric. Symmetric encryption relies on a single key for encoding and decoding data, whereas asymmetric encryption relies on a pair of keys—a public key to encrypt and a private key to decrypt. With digital interactions characterizing modern life, cryptography is the keystone of protecting data from unauthorized access and ensuring confidentiality. Various encryption and decryption methods are employed to protect digital data, deterring unauthorized alteration upon receipt. Apart from conventional approaches, we propose a new encryption and decryption process using a genetic algorithm in conjunction with logical operator XNOR to improve data protection by a unique and advanced mechanism.

Genetic algorithm:

A genetic algorithm in cryptography is a smart problem-solving method that works like natural evolution in nature. It begins with many possible solutions (such as keys or codes), checks which ones work best, and then improves them by keeping the best, mixing them, and making small changes. By repeating this process many times, it moves closer to the correct key, cracks the code, or strengthens the encryption system.

XNOR Operator:

XNOR operations are of greatest significance in cryptography because they have the potential to alter bits according to another binary string (the key) without any direct relation to the original information. The uncertainty makes it difficult for attackers to deduce the key or plaintext, especially when rounds of XNOR are called multiple times. One of the most important benefits of XNOR is that it changes every bit individually, so encryption works and The XNOR function produces 1 if both input bits are identical (both 0 or both 1) and 0 if the bits are not.

One-Point Crossover:

It's a simple mixing technique where two pieces of data (like encrypted text or keys) are cut at one random point and their parts are swapped to create new, mixed data. This adds randomness and strength to encryption.

Two-Point Crossover:

It's a technique where two data blocks (like encrypted text or keys) are cut at **two random points**, and the middle segments are swapped to create new mixed data. This adds more randomness and complexity than one-point crossover.

II. LITERATUEW REVIEW:

(1) Mittal Et. Al. [3] Established an Algorithm for Encrypting and Decrypting Messages Using a Genetic Algorithm. In Their Proposed Algorithm, They Used a Genetic Algorithm (Crossover and Mutation Technique) Which Is Based on Symmetric Key Cryptosystem

(2) Mittal et. al. [5] established an algorithm for encrypting and decrypting messages using a symmetric key cryptosystem based on a Genetic Algorithm. In their proposed algorithm, they used a substitution method along with genetic crossover and mutation techniques.

(3) P Srikanth [6] Established a New Encryption Technique Using Genetic Algorithm Operations and Pseudorandom Number. The Method First We Use the Genetic Algorithm Operation Such as Crossover and Mutation Functions, Genetic Algorithm Concepts with Pseudorandom Function Are Being Used to Encrypt and Decrypt Data. The Encryption Process Is Applied Over a Binary File Therefore the Algorithm Can Be Applied Over Any Type of Data.

III. METHODOLOGY:

(1) To encrypt and decrypt the message, we will use single point crossover operator and XNOR operator, which gives the intermediate cipher and then incorporate two-point (at 25th and 40th points) crossover. We get the final cipher text.

(2) To decrypt the message firstly we will use two-point (at 25th and 40th points) crossover, after that applying XNOR operator and matrix subtractive operation and single point crossover, we get the original plain text.

Additionally, we have used the tables which are as follows:

Table

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

(XNOR Operation)

ASCII table is also used in this paper.

IV. ALGORITHM:

Encryption:

1. Generate a key matrix.
   (a) Take block size of matrix (say n = 4)
   (b) Choose the input key.
   (c) We convert the input key which is taken by us into equivalent to ASCII table (decimal number).
   (d) After this we convert the decimal number into equivalent to binary code.
   (e) Now we apply the right shift operation by 2 bits on the binary string.
   (f) After performing the operation, we get the binary set which we convert into equivalent to decimal number.
   (g) After doing this we get the key matrix (say K).
2. Convert the chosen plaintext into a text matrix.
   (a) Consider the plain text.
   (b) Convert the plaintext into corresponding equivalent ASCII table.
   (c) Convert decimal numbers into equivalent ASCII table binary form.
   (d) Divide the binary string into 16 blocks with 8- bit/block.
   (e) apply the single point crossover operator on blocks after this we get another block.
   (f) Now convert binary blocks into equivalent ASCII table (decimal number) after this we get plain text matrix (say P).
3. By adding key matrix from plain text matrix produce an additive matrix (say C).
4. Apply logical operator XNOR on additive matrix.
   (a) Convert each element of matrix C into their corresponding binary equivalent (say Di).

(b) Calculate Zi = Ki XNOR Di, where i=1,2, 3…,16.
(c) Convert each 8-bit group into their corresponding hexadecimal equivalent.

5. Apply Genetic Crossover and Mutation.
   (a) Divide the Zi binary values into two segments.
   (b) Apply the two-point (at 25th and 40th points) crossover.
   (c) Apply the mutation function i.e. a '1' would mutate into '0' and vice versa.
6. Divide the binary streams into group of 8-bits and convert each 8-bit group into their corresponding hexadecimal equivalent.

Decryption:

1. Consider the cipher text.
2. Convert each hexadecimal value to 8-bit binary.
3. Now apply the mutation function, i.e. a '1' would mutate into '0' and vice versa.
   (a) Divide the binary streams into two segments.
4. Apply the two-point (at 25th and 40th points) crossover.
5. Combine P1 and P2 to get the Z values.
6. Calculate Di = Ki XNOR Zi, i = 1, 2, 3, …, 16.
7. Now convert 8-bit binary group into their corresponding decimal equivalent and arrange them in a square matrix of order 4.
8. Obtaining the Subtractive Matrix.
9. Convert above plaintext matrix into equivalent ASCII table binary form (say B).
10. Now reverse the single-point crossover (swap the last 4 bits back).
11. We get Final Plaintext.

V. ILLUSTRATION:

This example is based on the above algorithm involving genetic algorithm and logical operator XNOR.

Encryption:

1. **Generation of Key Matrix:**

(a) Take block size of matrix (say n = 4).
(b) Choose the input key

MISUNDERSTANDING

(c) Convert the input key into corresponding equivalent ASCII table:
77, 73, 83, 85, 78, 68, 69, 82, 83, 84, 65, 78, 68, 73, 78, 71
(d) Convert above numeric code into equivalent binary form:

01001101	01001001	01010011	01010101
01001110	01000100	01000101	01010010
01010011	01010100	01000001	01001110
01000100	01001001	01001110	01000111

(e) Now we apply the right shift operation by 2 bits on the above binary streams, as follows:

00010011	00010010	00010100	00010101
00010011	00010001	00010001	00010100
00010100	00010101	00010000	00010011
00010001	00010010	00010011	00010001

(f) After performing the operation, we get the binary set. Now convert this binary set into equivalent ASCII TABLE, we get the key matrix (say K).

$$K = \begin{bmatrix} 19 & 18 & 20 & 21 \\ 19 & 17 & 17 & 20 \\ 20 & 21 & 16 & 19 \\ 17 & 18 & 19 & 17 \end{bmatrix}$$

2. **Convert the Chosen Plaintext into a Text Matrix:**

(a) Consider the plaintext:

MISCOMMUNICATION

(b) Convert the plain text into corresponding equivalent ASCII table:
77, 73, 83, 67, 79, 77, 77, 85, 78, 73, 67, 65, 84, 73, 79, 78
(c) Convert above numeric code into equivalent ASCII table binary form:

01001101010010010101001101000011010011110100110101001101010101010100111001001001
010000110100000101010100010010010100111101001110

(length is = 128)

(d) Divide the above binary string into 16 blocks with 8- bit/block, denote each 8-bit group by A_i (say), i = 1, 2, 3, …. Choose 8 is the first key value. Therefore:

A1= 01001101 A2= 01001001 A3= 01010011 A4= 01000011
A5= 01001111 A6= 01001101 A7= 01001101 A8= 01010101
A9= 01001110 A10= 01001001 A11=01000011 A12=01000001
A13=01010100 A14=01001001 A15=01001111 A16=01001110

(e) Now apply the single point crossover operator on A1 and A2, A3 and A4, A5 and A6, A7 and A8, A9 and A10, A11 and A12, A13 and A14, A15 and A16. We get another 16 blocks B1, B2, B3, B4, B5, B6, B7, B8, B9, B10, B11, B12, B13, B14, B15, B16 as follows:

A1= 0100 1101 A2= 0100 1001

B1= 0100 1001 B2= 0100 1101

A3= 0101 0011 A4= 0100 0011

B3= 0101 0011 B4= 0100 0011

A5= 0100 1111 A6= 0100 1101

B5= 0100 1101 B6= 0100 1111

A7= 0100 1101 A8= 0101 0101

B7= 0100 0101 B8= 0101 1101

A9= 0100 1110 A10= 0100 1001

B9= 0100 1001 B10= 0100 1110

A11= 0100 0011 A12= 0100 0001

B11= 0100 0001 B12= 0100 0011

A13= 0101 0100 A14= 0100 1001

B13= 0101 1001 B14= 0100 0100

A15= 0100 1111 A16= 0100 1110

B15= 0100 1110 B16= 0100 1111

Therefore:

B1= 01001001 B2= 01001101 B3= 01010011 B4= 01000011
B5= 01001101 B6= 01001111 B7= 01000101 B8= 01011101
B9= 01001001 B10= 01001110 B11= 01000001 B12= 01000011
B13= 01011001 B14= 01000100 B15= 01001110 B16= 01001111

(f) After performing the operation, we get the binary set. Now convert this binary set into equivalent ASCII TABLE, we get the plain text matrix (say P):

$$P = \begin{bmatrix} 73 & 77 & 83 & 67 \\ 77 & 79 & 69 & 93 \\ 73 & 78 & 65 & 67 \\ 89 & 68 & 78 & 79 \end{bmatrix}$$

3. **Obtaining the Additive Matrix:**
By adding key matrix K from plaintext matrix P. produce an additive matrix (say C) as follows:

$$C = \begin{bmatrix} 19 & 18 & 20 & 21 \\ 19 & 17 & 17 & 20 \\ 20 & 21 & 16 & 19 \\ 17 & 18 & 19 & 17 \end{bmatrix} + \begin{bmatrix} 73 & 77 & 83 & 67 \\ 77 & 79 & 69 & 93 \\ 73 & 78 & 65 & 67 \\ 89 & 68 & 78 & 79 \end{bmatrix}$$

$$C = \begin{bmatrix} 92 & 95 & 103 & 88 \\ 96 & 96 & 86 & 113 \\ 93 & 99 & 81 & 86 \\ 106 & 86 & 97 & 96 \end{bmatrix}$$

4. **Appling logical operator XNOR:**
(a) Convert each element of matrix C into their corresponding binary equivalent as follows, say them D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14, D15, D16 as follows:

D1= 01011100 D2= 01011111 D3= 01100111 D4= 01011000
D5= 01100000 D6= 01100000 D7= 01010110 D8= 01110001
D9= 01011101 D10= 01100011 D11= 01010001 D12= 01010110
D13= 01101010 D14= 01010110 D15= 01100001 D16= 01100000

(b) Calculate Zi = Ki XNOR Di, i = 1, 2, 3, …, 16:
Z1= 10110000 Z2= 10110010 Z3= 10001100 Z4= 10110010
Z5= 10001100 Z6= 10001110 Z7= 10111000 Z8= 10011010
Z9= 10110110 Z10= 10001001 Z11= 10111110 Z12= 10111010
Z13= 10000100 Z14= 10111011 Z15= 10001101 Z16= 10001100

(c) Convert each 8-bit group into their corresponding hexadecimal equivalent:
B0, B2, 8C, B2, 8C, 8E, B8, 9A, B6, 89, BE, BA, 84, BB, 8D, 8C

5. **Genetic Crossover and Mutation:**

(a) Divide the above binary streams into two segments as follows:
1011000010110010100011001011001010001100100011101011100010011010
1011011010001001101111101011101010000100101110111000110110001100

(b) On the above input, apply the two-point (at 25th and 40th points) crossover, we get:
P1=
(1-24) = 101100001011001010001100
(25-40) = 1011001010001100 (swapped showed in P2)
(41-64) = 100011101011100010011010
P2=
(1-24) = 101101101000100110111110
(25-40) = 1011101010000100
(41-64) = 101110111000110110001100
(swapped the middle parts 25 to 40 in P1 and P2, we get C1 and C2)
C1=1011000010110010100011001011101010000100100011101011100010011010
C2=1011011010001001101111101011101010000100101110111000110110001100

(c) Now apply the mutation function. Here we use the flipping of bits technique i.e. a '1' would mutate into '0' and vice versa, therefore we get:
01001111010011011011100110100010101111011011100010100011101100101
0100100101110110010000010100010101111011010001000111001001110011

6. **Divide the above binary streams into group of 8-bits, we get as follows:**

01001111	01001101	10111001	01000101
01111011	01110001	01000111	01100101
01001001	01110110	01000001	01000101
01111011	01000100	01110010	01110011

7. **Convert each 8-bit group into their corresponding hexadecimal equivalent:**
4F, 4D, B9, 45, 7B, 71, 47, 65, 49, 76, 41, 45, 7B, 44, 72, 73

Decryption:

1. **Consider the ciphertext:**
   4F, 4D, B9, 45, 7B, 71, 47, 65, 49, 76, 41, 45, 7B, 44, 72, 73
2. **Convert each hexadecimal value to 8-bit binary:**

01001111	01001101	10111001	01000101
01111011	01110001	01000111	01100101
01001001	01110110	01000001	01000101
01111011	01000100	01110010	01110011

3. **Now apply the mutation function. Here we use the flipping of bits technique i.e. a '1' would mutate into '0' and vice versa, therefore we get:**

10110000	10110010	01000110	10111010
10000100	10001110	10111000	10011010
10110110	10001001	10111110	10111010
10000100	10111011	10001101	10001100

   Divide the above binary streams into two segments as follows:
   C1 = 1011000010110010100011001011101010000100100011101011100010011010
   C2 = 1011011010001001101111101011101010000100101110111000110110001100

4. **On the above input, apply the two-point (at 25th and 40th points) crossover, we get:**
   To reverse:
   - Take first 24 bits of C1 + middle 16 bits of C2 + remaining bits of C1
   - Take first 24 bits of C2 + middle 16 bits of C1 + remaining bits of C2

   P1=
   (1-24) = 101100001011001010001100
   (25-40) = 1011001010001100 (swapped showed in P2)
   (41-60) = 100011101011100010011010
   P2=
   (1-24) = 101101101000100110111110
   (25-40) = 1011101010000100
   (41-64) = 101110111000110110001100
   P1=1011000010110010100011001011001010001100100011101011100010011010
   P2=1011011010001001101111101011101010000100101110111000110110001100

5. **Combine P1 and P2 we get the Z values:**
   Z1= 10110000 Z2= 10110010 Z3= 10001100 Z4= 10110010
   Z5= 10001100 Z6= 10001110 Z7= 10111000 Z8= 10011010
   Z9= 10110110 Z10= 10001001 Z11= 10111110 Z12= 10111010
   Z13= 10000100 Z14= 10111011 Z15= 10001101 Z16= 10001100

6. **Calculate Di = Ki XNOR Zi, i = 1, 2, 3, …, 16:**
   D1= 01011100 D2= 01011111 D3= 01100111 D4= 01011000
   D5= 01100000 D6= 01100000 D7= 01010110 D8= 01110001
   D9= 01011101 D10= 01100011 D11= 01010001 D12= 01010110
   D13= 01101010 D14= 01010110 D15= 01100001 D16= 01100000

7. **Now convert above 8-bit binary group into their corresponding decimal equivalent and arrange them in a square matrix of order 4, we get:**

$$C = \begin{bmatrix} 92 & 95 & 103 & 88 \\ 96 & 96 & 86 & 113 \\ 93 & 99 & 81 & 86 \\ 106 & 86 & 97 & 96 \end{bmatrix}$$

8. Obtaining the Subtractive Matrix:

By subtracting matrix C from key matrix K. produce a plaintext matrix (say P) as follows:

$$P = \begin{bmatrix} 73 & 77 & 83 & 67 \\ 77 & 79 & 69 & 93 \\ 73 & 78 & 65 & 67 \\ 89 & 68 & 78 & 79 \end{bmatrix}$$

9. Convert above plaintext matrix into equivalent ASCII table binary form (say B):

B1= 01001001 B2= 01001101 B3= 01010011 B4= 01000011
B5= 01001101 B6= 01001111 B7= 01000101 B8= 01011101
B9= 01001001 B10= 01001110 B11= 01000001 B12= 01000011
B13= 01011001 B14= 01000100 B15= 01001110 B16= 01001111

10. Now reverse the single-point crossover (swap the last 4 bits back):

For each pair (B1, B2), (B3, B4), etc., swap the last 4 bits between them.
A1 = first 4 of B1 + last 4 of B2 = 0100 + 1101 = 01001101
A2 = first 4 of B2 + last 4 of B1 = 0100 + 1001 = 01001001
Similarly for all pairs:
A1=01001101 (B1 first 4 + B2 last 4)
A2=01001001 (B2 first 4 + B1 last 4)

A3=01010011 (B3 first 4 + B4 last 4)
A4=01000011 (B4 first 4 + B3 last 4)

A5=01001111 (B5 first 4 + B6 last 4)
A6=01001101 (B6 first 4 + B5 last 4)

A7=01001101 (B7 first 4 + B8 last 4)
A8=01010101 (B8 first 4 + B7 last 4)

A9=01001110 (B9 first 4 + B10 last 4)
A10=01001001 (B10 first 4 + B9 last 4)

A11=01000011 (B11 first 4 + B12 last 4)
A12=01000001 (B12 first 4 + B11 last 4)

A13=01010100 (B13 first 4 + B14 last 4)
A14=01001001 (B14 first 4 + B13 last 4)

A15=01001111 (B15 first 4 + B16 last 4)
A16=01001110 (B16 first 4 + B15 last 4)

Now convert each 8-bit A value to its ASCII character:

A1=01001101 = 77 = 'M'
A2=01001001 = 73 = 'I'
A3=01010011 = 83 = 'S'
A4=01000011 = 67 = 'C'
A5=01001111 = 79 = 'O'
A6=01001101 = 77 = 'M'
A7=01001101 = 77 = 'M'
A8=01010101 = 85 = 'U'
A9=01001110 = 78 = 'N'
A10=01001001 = 73 = 'I'
A11=01000011 = 67 = 'C'
A12=01000001 = 65 = 'A'
A13=01010100 = 84 = 'T'
A14=01001001 = 73 = 'I'
A15=01001111 = 79 = 'O'
A16=01001110 = 78 = 'N'

11. Final Plaintext:

Combining these: M, I, S, C, O, M, M, U, N, I, C, A, T, I, O, N

Which spells: "MISCOMMUNICATION"
This matches the original plaintext

V. RESULT AND DISCUSSION:

In this paper, we introduce a novel encryption scheme that effectively incorporates Genetic Algorithm operations, such as crossover and mutation, with the XNOR logical operator to safely encrypt and decrypt information, as illustrated using the plaintext "MISCOMMUNICATION." The multi-stage process—consisting of matrix addition, bit-shifting, and swapping of data—completely scrambles and diffuses the information so much that it becomes highly challenging for an intruder to detect any pattern or correlation between the initial message and the encrypted message without the proper key. Because the key is created by converting text to ASCII code and shifting its bits, the system becomes very complex. This makes it extremely strong against common hacking methods like brute force (guessing all keys) or frequency analysis (looking for patterns). The process is so complex that trying to figure out the key without already knowing it is pretty much impossible for a computer. Therefore, this combined approach creates a very strong and safe way to protect important information.

These methods complicate decryption even further. Brute force attacks also fail to decrypt it because the encryption employs a 4×4 matrix as a key. decryption of big messages is not feasible. This keeps stored and sent messages secret, particularly for secret information such as military information. Because of these security properties, the likelihood of attacks such as ciphertext attack, chosen plaintext attack, brute force attack, and known plaintext attack is very slim. Hence, this suggested algorithm is much more secure than other encryption algorithms designed by researchers.

VI. CONCLUSION:

In conclusion, the genetic algorithm and logical operator XNOR stands as an essential tool for simplifying complex problems and enhancing security, with potential for continued growth and adaptation in the future.

REFERENCES

1. Chowdhury Somalina, Das Sisir Kumar, Das Annapurna: Application of Genetic Algorithm in Communication Network Security, International Journal of Innovative Research in Computer and Communication Engineering, Volume-03, Issue-01, January 2015.
2. Fanyo Marilyn, Sharma Tanuja Kumari: A New Data Encryption Model Based On Genetic Inspired Cryptography, JETIR, Volume-06, Issue-06, June 2019.
3. Mittal Ayush: A New Cryptographic Technique Involving Genetic Algorithm, International Journal of Scientific & Technology Research, Volume-9 Issue-03, March, 2020.
4. Muhammad Irshad Nazeer, Ghulam Ali Mallah, Noor Ahmed Shaikh: Implication of Genetic Algorithm in Cryptography to Enhance Security, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 9, No. 6, 2018.
5. Mittal Ayush and Gupta Ravindra Kumar: Encryption and Decryption of a Message Involving Genetic Algorithm, International Journal of Engineering and Advanced Technology (IJEAT), Volume-9 Issue-2, December, 2019.
6. P Srikanth, Mehta Abhinav, Yadav Neha, Singh Sahil, Singhal Shubham: Encryption and Decryption Using Genetic Algorithm Operations and Pseudorandom Number, International Journal of Computer Science and Network (IJCSN), Volume 6, Issue 3, June 2017.
7. Sindhuja K, Pramela Devi S: A Symmetric Key Encryption Technique Using Genetic Algorithm, International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 5, Issue 1, 2014.