A Multi-Level Random Key Cryptosystem based on DNA Encoding and State-Changing Mealy Machine

by

Towshik Anam Taj
21366026

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
BRAC University
February 2023

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Towshik Anam Taj
21366026

Approval

The thesis/project titled "A Multi-Level Random Key Cryptosystem based on DNA Encoding and State-Changing Mealy Machine" submitted by

1. Towshik Anam Taj (21366026)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on February 28, 2023.

Examining Committee:

Supervisor:
(Member)

Muhammad Iqbal Hossain, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi
Chairperson and Associate Professor
Department of Computer Science and Engineering
BRAC University

Examining Committee: (Cont.)

External:
(Member)

Mohammad Zahidur Rahman, PhD
Professor
Department of Computer Science and Engineering
Jahangirnagar University

Internal:
(Member)

Md. Golam Rabiul Alam, PhD
Professor
Department of Computer Science and Engineering
BRAC University

Internal:
(Member)

Amitabha Chakrabarty, PhD
Associate Professor
Department of Computer Science and Engineering
BRAC University

Ethics Statement (Optional)

This is optional, if you don't have an ethics statement then omit this page

Abstract

Cryptography allows our data to be transmitted without giving sensitive information away. This is the art of hiding the information from the malicious third party and make the data accessible to only the sender and the receiver. Building a complex cryptosystem has always been a challenge which can provide relentless security and is infeasible to break. This paper discusses about a hybrid cryptosystem which is inspired from the concepts of DNA cryptography and it is further strengthened using multiple components. A random key is used which is generated using run test of randomness, different DNA encoding combinations and a state changing random state mealy machine is used for further strengthening the security. This paper provides a detailed discussion regarding every components the authors used to build the system and also discusses about the combined system that has been built. This paper also discusses about the effectiveness and the performance of the proposed system to give an overview of its security measures and also provides some comparative analysis with existing works to back the claim on improved security features.

Keywords: Cryptography, DNA Cryptography, Key Generation, Run Test of Randomness, Encryption, Decryption, Mealy Machine, State Changing Mealy Machine, DNA Encoding

Dedication (Optional)

A dedication is the expression of friendly connection or thanks by the author towards another person. It can occupy one or multiple lines depending on its importance. You can remove this page if you want.

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption.
Secondly, to my thesis supervisor Dr. Muhammad Iqbal Hossain sir for his kind support and advice in this work. He helped me whenever any issue was required solving.
And finally to my parents without their throughout support it may not be possible. With their kind support and prayer we are now on the verge of our graduation.

Table of Contents

Declaration i

Approval ii

Ethics Statement iv

Abstract v

Dedication vi

Acknowledgment vii

Table of Contents viii

List of Figures x

List of Tables xi

Nomenclature xii

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Research Objective . . . 2
- 1.4 Paper Outline . . . 3

2 Related Works 4

3 Methodology 6
- 3.1 Key Generation . . . 6
- 3.2 Mealy Machine . . . 9
- 3.3 In-Shuffle . . . 12
- 3.4 Encryption and Decryption . . . 12

4 Case Study 17
- 4.1 Encryption Process . . . 17
  - 4.1.1 Key Generation . . . 17
  - 4.1.2 Encryption . . . 18
- 4.2 Decryption Process . . . 20

5 Result & Analysis **23**
5.1 Experimental Setup 23
5.2 Prevention Against Attacks 23
5.2.1 Brute-Force Attack 23
5.2.2 Known Plaintext Attack 24
5.2.3 Ciphertext Only Attack 24
5.2.4 Man in The Middle Attack 26
5.2.5 Differential Cryptanalysis Attack 26
5.3 Encryption and Decryption Time 27
5.4 Avalanche Effect 27
5.4.1 Changing 1 Key Bit 28
5.4.2 Modifying State Change Operation 28
5.4.3 Modifying Encoding Operation 29

6 Conclusion **31**

Bibliography **33**

List of Figures

3.1 A Random State Mealy Machine 9
3.2 Mealy Machine After Changing State 10

5.1 Frequency of DNA Bases in Selected Ciphertexts for Small Data . . . 24
5.2 Frequency of DNA Bases in Selected Ciphertexts for Medium Data . 25
5.3 Frequency of DNA Bases in Selected Ciphertexts for Big Data 25

List of Tables

3.1 Transition Tables 10
3.2 Transition Tables After Changing Configuration 11
3.3 Encoding Sequences 11

4.1 Transition Tables After Quantity 19
4.2 Transition Tables After Quantity2 19
4.3 Transition Tables After Quantity 21
4.4 Transition Tables After Quantity2 22

5.1 Time taken for Encryption and Decryption Part 1 26
5.2 Time taken for Encryption and Decryption Part 2 26
5.3 Changes in Decrypted Text for 1-bit Change in Key for Small Data . 27
5.4 Changes in Decrypted Text for 1-bit Change in Key for Medium Data 27
5.5 Changes in Decrypted Text for 1-bit Change in Key for Big Data . . 28
5.6 Changes in Decrypted Text for Modifying A State Change Operation for Small Data 28
5.7 Changes in Decrypted Text for Modifying A State Change Operation for Medium Data 29
5.8 Changes in Decrypted Text for Modifying A State Change Operation for Big Data 29
5.9 Changes in Decrypted Text for Modifying An Encoding Operation for Small Data 29
5.10 Changes in Decrypted Text for Modifying An Encoding Operation for Medium Data 30
5.11 Changes in Decrypted Text for Modifying An Encoding Operation for Big Data 30

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

δ	Delta
μ	Expected No of Runs
$\prod$	Product
σ	Standard Deviation
$\sum$	Summation
mod	Modulus

Chapter 1

Introduction

Cryptography is the art of hiding necessary information from the third party to whom the information is not intended but make the information available for the desired party at the same time. In the modern era of vast technological advancement the cryptographic procedures are also required to be much more complex so that the security of information is unharmed. DNA cryptography comes with the idea of being computationally complex while ensuring secure transmission of data. While various work on DNA cryptography has been done, the research on this field of cryptography is also required to be evolving on a regular basis to stay in sync with the modern technology and the code breakers. In this work the authors built a system inspired from the concepts of DNA cryptography and combined them with other concepts to build a secure and complex cryptosystem. The authors also provided comparative analysis with existing works of DNA cryptography to give an overview of the system's security trait and also necessary analysis were given regarding the system's unique security traits to support the argument of the system being secure.

1.1 Background

Deoxyribonucleic Acid (DNA) is the building element of animal kingdom. DNA stores all the basic information of all humankind and other animals, which passes on from one generation to another. DNA is composed of nucleotides which contains four types of bases, Adenine (A), Guanine (G), Cytosine (C) & Thymine (T). While all these seems very biological, in the field of computer science DNA has its own usefulness. Hameed [5] stated that 10 trillion DNA molecules can be fitted into a space of a marble while only 1 gram of DNA can store 215 petabytes (215 million gigabytes) of data. So it is possible to fit all the data centers into test-tubes if harnessing the computational power of DNA is possible while it would also save tons of electricity as DNA operations are done using chemical reactions. To fit the idea of using DNA in the field of computer science, the idea of DNA computing came and from DNA computing came DNA cryptography. The field of DNA cryptography is not new in the field of cryptographic research. The idea of solving the computationally complex problems using DNA computing fueled many researchers into looking into it. While this type of research requires expensive and appropriate laboratory setup, not everyone has access to these kind of setup. That is why the modern era researchers mostly adopted the principles of DNA operations, properties and in some cases mixed them with already established algorithms to build strong and

hybrid cryptographic algorithms. They are also the building element of the resulting ciphertext after encryption in DNA cryptography. The researchers also combined several ideas and properties and in some cases various works of other researchers to build hybrid DNA cryptosystem which are also proven as effective and can be used for transmitting sensitive information through an open channel without having any risk of exposure as those systems are found to be quite complex to break.

1.2 Problem Statement

The main goal of this work is to build an effective and complex cryptosystem which will be resistant to various types of attacks and which will be able to transmit data securely. The understanding of the researchers was multiple security features are necessary to provide relentless security for the sensitive data so that data does not go into the hand of the third party if some of the features get compromised. So the authors decided to build an effective system whose security trait will not depend on any single feature entirely but a collection of features will protect the data. The idea of using several security features came so that even if one feature may get compromised, the other features will still be able to provide necessary security which is the key motivation for the authors.

1.3 Research Objective

The major contribution of authors in this work is the authors ensured security using multiple layers and each layer is not dependant on other and is able to provide relentless security even if one layer gets compromised. The authors broke the whole system into multiple components and decided to make each component secure autonomously, so that each component of the system can protect sensitive data alone. The whole system is an integrated version of these security components that protects the data even if one component gets compromised. In order to build a system that provides multiple layers of security, the authors took inspiration from various works, came up with some original ideas and finally combined several features to build a hybrid system that provides better security than existing works in the field of DNA cryptography. In this work the authors introduced a new cryptographic algorithm which is inspired by the DNA cryptography's principle of being computationally complex and infeasible to break. The authors used genetic algorithm to generate a random cryptographic key which was evaluated using the run test of randomness to ensure the key is generated randomly. The authors broke the plaintext into fixed sized blocks, encrypted each block using different DNA encoding sequences which turned the plaintext into DNA sequences, entered the DNA sequences through a randomly generated state changing mealy machine to generate different DNA sequences. And finally changed the mealy machine's configuration several times using several mathematical operations to make it ready for encrypting the next block of plaintext. The authors ensured the same set of operations and calculations for both the sender and the receiver side for both encryption and decryption operation. The authors mainly focused on increasing the security of the entire system, some compromise has been done on other attributes such as encryption and decryption time. But the authors showed that even after compromising a bit on time, this system

provides better result in terms of encryption and decryption time than most systems and this system's unique security traits provide better security than similar existing work. The authors discussed the entire process briefly and provided a case study for better understanding of the proposed work. The authors also provided appropriate analysis to evaluate the work which includes how the system can resist various types of attacks and comparative analysis with existing works that ensures the authors has been successful in building a more secure and effective cryptosystem.

1.4 Paper Outline

- **Chapter 2:** This chapter includes overview of the works that are similar and relevant to proposed scheme. That includes the works that used the principles of DNA cryptography, works that used similar principles for different format of data and also the works that introduces hybrid methods based on DNA cryptography and other schemes to build strong cryptosystems.

- **Chapter 3:** This chapter gives a detailed overview of the system that has been built. This chapter discusses about the components that has been used to build the system as a whole such as key generation procedure, measuring randomness of the key, mealy machine and related operations, in-shuffle and finally discusses about the system as whole, the encryption and the decryption procedure and also includes key generation, encryption and decryption algorithm.

- **Chapter 4:** This chapter provides a detailed case study that demonstrates how the system works using a simulation of both the encryption and the decryption operation.

- **Chapter 5:** This chapter discusses about how the system is resistant of various types of attacks and also it provides analysis regarding its security traits based on three types of data which are small, medium and big. This chapter also includes evaluation of the system's security traits that also includes comparative analysis and evaluation of the security traits of the system's unique components.

- **Chapter 6:** This chapter draws conclusion of all the discussions regarding this work and also discusses about a few areas on which some works can be done in the future.

Chapter 2

Related Works

The basic idea of DNA Cryptography comes from DNA computing which is an active field of research. Adleman [1] intended to use the properties of molecular biology to solve one of the most known problems in the field of computer science and applied mathematics, "The Hamiltonian Path". After the success of Adleman, so many researchers such as Lipton [2] intended to adopt his techniques to solve constraint satisfaction and NP complete problems. Their research on DNA computation created possibilities for expanding it further into the field of cryptography. The modern era researchers adopted properties of DNA translation, DNA transcription, molecular biology and other principles to create strong cryptographic algorithms which are computationally complex. Pramanik et al. [6] proposed a simple idea where they used "The Watson-Crick Complementary" to encrypt the data and transmit the encrypted data sequentially through an open channel. Their method needs a secure channel to transmit the key that is required for the encryption. Their method focused mostly on biological properties and principles to adopt a novel cryptosystem and they proposed that one of the practical implementation of their work can be to use it for One Time Pads. Rafiul et al. [12] on the other hand introduced a complex cryptosystem which used a dynamic sequence table to convert plaintext to DNA bases where the table was continuously changed using the fibonacci or any other mathematical series. After the initial conversion, dynamic DNA encoding is used to break the received output into different chunks of DNA bases which is also done using the same mathematical series which was chosen in the first step of the encryption process. And finally they performed a sequential merge to unite all the chunks which serves as the final ciphertext. Sayantani et al. [11] included machine learning upon building a DNA cryptographic algorithm. They used Bi-directional Associative Memory Neural Network (BAMNN) for key generation. They adopted DNA transcription and translation techniques into algorithms to convert the plaintext into encrypted DNA base and used the generated key to add an extra layer of security. Kalsi et al. [9] used Genetic Algorithm to generate the key where they used Run Test of Randomness to ensure the generated key's randomness. They went further and used Needleman- Wunsch (NW) Algorithm to find the most dissimilar keys and used an XORed version of two of the most dissimilar keys. Like Sayantani et al. [11] they also adopted DNA Transcription and Translation into algorithms to convert the plaintext into DNA bases although their method stayed more true to the biological operations that occur and the components that are used in those operations. Sakr et al [21] used a novel approach for encryption. Their work used

genetic algorithm to generate a random security key where randomness was measured using run test of randomness [10]. After that they used amino acid encryption technique which used amino acid based playfair cipher technique which generates a protein cipher as cipertext. While all the mentioned work has been used for text data encryption, there are methods that are introduced for encrypting images based on the concepts of DNA cryptography. Chirakkarottu et al. [17] introduced a novel technique that encrypts medical images using two-dimensional Zaslavski map and DNA cryptography. Their system can work on any type of medical image. They used two phases, permutation and diffusion while the diffusion phase used the principles of DNA cryptography where they converted each pixel of the image into DNA sequences using DNA digital encoding. This proves that the principles of DNA cryptography can be used widely and they also can be used on different formats of data. Hazra et al. [18] combined the concepts of One Time Pad and DNA cryptography to create a two layered security system that uses both the aforementioned concepts on textual data to encrypt and transfer the data securely. They used a DNA permutation table to encrypt the data which they receive after completing the first step of encryption using one time pad. This shows that multiple concepts can be used sequentially to create a strong and hybrid cryptosystem that may use DNA cryptography as one of its security feature. Hassan et al. [20] used DNA encoding scheme along with Huffman-Coding that allows to further modify the already encoded DNA sequence using the frequency of DNA bases so that the ciphertext becomes more systematically secure with added resistance against pattern analysis schemes. Paul et al. [8] proposed a DNA cryptography technique that uses symmetric key change method. Their work used XOR operation with a one time pad DNA sequence for encrypting and transmitting data. However their method is not found to be appropriate for large messages and the length of the key their work used is relatively small. Kaundal et al. [7] took inspiration from Fiestel-Cipher which they combined with the idea of DNA Cryptography that they found from Pramanik et al. [6]. The later mentioned work used a digital DNA encoding, Kaundal [7] used DNA hybridization technique along with that DNA encoding and combined them with Fiestel-Cipher to create their own hybrid algorithm. This algorithm is computationally complex but takes hefty amount of time on encyption and decryption. Pavithran et al. [19] presented a different kind of idea which involved introducing a randomly generated finite state mealy machine in their system which helped in building a very strong cryptographic algorithm. They did not focus on improving the security much but focused on other attributes such as time and throughput of their system. Their work used the same configuration of mealy machine over and over again without any change in it, used a single encoding scheme that they choose for a term or session. This work took inspiration from their work but the authors modified their idea to improve the overall security of their system. This work focused on changing the mealy machine's configuration at a fixed amount of interval, used different encoding scheme for different block of plaintext and used a randomly measured and generated key for the encryption and decryption operation which improves overall security of the system than the aforementioned work. The detailed methodology of this work will be discussed in the following section.

Chapter 3

Methodology

In this work the authors followed a few steps to complete the entire encryption and decryption process. Before demonstrating them the authors would like to discuss about several attributes that has been used in the whole process such as key generation mechanism, mealy machine, in-shuffle and corresponding details.

3.1 Key Generation

The authors used a proven technique to generate an n-bit length binary key which is used in the encryption process. That same key was shared by both the sender and receiver for both encryption and decryption for a particular session. The authors used Genetic Algorithm to generate the key as the usage of an evolutionary algorithm generates an appropriate outcome based on specific constraints. The authors ensured the key's randomness using run test of randomness [10]. In this particular test number of runs is calculated for a string. For example in a binary sequence there is only two types of digits, 0 and 1. So in the following binary sequence "00011000" the number of runs will be the number of uninterrupted sequences, which is 3. Run test of randomness is measured using the Z value where Z can be calculated using Equation 3.1. In the equation R refers to number of runs in the sequence, μ refers to the expected number of runs and σ refers to the standard deviation. The value of μ can be calculated using Equation 3.2 and the value of *sigma* can be calculated using Equation 3.3 where n_1 refers to the total number of 0's in the sequence and n_2 refers to the total number of 1's in the sequence. Having lesser number of runs or having very much greater number of runs can both make a sequence non-random. That is why hypothesis test is done to declare the sequence is random or not. In this work the authors used two tailed 5% hypothesis test to ensure the randomness of the key. The two tailed test refers to two open boundaries, while 5% test ensures the value of the boundaries which are 1.96 and -1.96. Anything that is scored in between is declared as random and anything outside this boundary is declared as not random.

$$Z = \frac{R - \mu}{\sigma}. \tag{3.1}$$

$$\mu = \frac{2n_1n_2}{n_1 + n_2} + 1 \tag{3.2}$$

$$\sigma = \sqrt{\frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}} \tag{3.3}$$

This 5% test ensures a sequence to be found random at a confidence level of 95%. Using a random key is necessary so that the hacker may not be able to guess it easily. And further randomizing the key using an already established method gives the authors huge advantage on building a secure cryptographic algorithm.
Genetic algorithm is used to find new generations of key sequences and run test of randomness is used to calculate the fitness score of the key sequences. The process starts with generating two random binary sequences of length N. After that the two sequences are modified using random selection, multi-point crossover and bit-flip mutation operation for a fixed number of generations. In this work the maximum generation number is taken as 100000. Before the crossover operation two random regions from both binary sequences of same length are selected and an arbitrary value q is selected where q is any arbitrary value between 1 to 50. After that the selected sequences are swapped which is referred as the crossover operation. This operation is repeated q times in each generation. After the crossover operation, any arbitrary value r is selected range same as q. Then single random element is selected from both the binary sequences and that element is changed to 1 if the selected element is 0 and vice versa. This operation is referred as the mutation operation and it is repeated r times for both the binary sequences. Crossover and mutation operation are done to bring variety in the binary sequences in each generation so that it is further randomized in each generation. After the operations are done, the fitness score are calculated for both the binary sequences using the run test of randomness. If the score of both binary sequences are within the randomness threshold, in this work which is in between -1.96 to 1.96, then any one of them is selected as the final key randomly. And if only one of them is found to have scored in between the threshold then that sequences is selected as the final key. This whole process is repeated until the maximum number of generations is reached and the final key is modified countless times in between the process randomizing the key further. After the maximum number of generations is reached, the final key is returned and that key is used between the sender and the receive for a particular session to securely transmit the data. Algorithm 1 gives an overview of the whole operation of the key generating process. The key can be shared using a secure channel and also using the Diffie–Hellman Key Exchange Algorithm to securely transmit the key through an open channel. The key ensures one layer of security in this work, there are other layers of security that makes the algorithm very strong against various attacks. The other components are discussed further in this paper.

Algorithm 1 Key Generation Algorithm

1. START
2. $Key1 \leftarrow Random\ Binary\ String\ of\ Length\ N$
3. $Key2 \leftarrow Random\ Binary\ String\ of\ Length\ N$
4. **while** $generation < max\_generation$
5. $\quad Q \leftarrow Any\ Random\ Arbritary\ Value$
6. $\quad$ **while** $iteration < Q$
7. $\quad\quad Key\_Chunk1, Key\_Chunk2 \leftarrow selection(Key1, Key2)$
8. $\quad\quad Key1, Key2 \leftarrow crossover(Key\_Chunk1, Key\_Chunk2) \cup Unselected\ Key\ Region$
9. $\quad\quad Bit1, Bit2 \leftarrow selection(Key1, Key2)$
10. $\quad\quad Key1, Key2 \leftarrow mutation(Bit1, Bit2) \cup Unselected\ Key\ Region$
11. $\quad\quad iteration \leftarrow iteration + 1$
12. $\quad$ **end while**
13. $\quad score1, score2 \leftarrow fitness\_function(Key1, Key2)$
14. $\quad$ **if** $(-1.96 < score1 < 1.96)$ **and** $(-1.96 < score < 1.96)$
15. $\quad\quad final\_key \leftarrow random\_selection(Key1, Key2)$
16. $\quad$ **else**
17. $\quad\quad final\_key \leftarrow Key\ That\ Matches\ Threshold$
18. $\quad$ **end if**
19. $\quad generation \leftarrow generation + 1$
20. **end while**
21. $return\ final\_key$
22. END

3.2 Mealy Machine

Mealy Machine (Hopcroft et al.) [4] is a finite state machine which is consist of b no of states where b is any positive finite number. If a string of length n is given as an input to a mealy machine then an output of length n will be provided where the output is dependant on both the current state and current input. Mealy machine is defined using six symbols, M = (Q, $\sum$, O, δ, f, q_0) where,

- Q = A non-empty finite set of states
- $\sum$ = A non-empty finite set of input symbols
- O = A finite set of output symbols
- δ = Input transition function, where δ: Q x $\sum \rightarrow$ Q
- f = Output transition function, where f : Q x $\sum \rightarrow$ O
- q_0 = Initial state

The authors used a finite-state mealy machine for the implementation of the system. Since the authors worked with DNA cryptography, that is why their mealy machine is consist of four states to match with four DNA bases. Figure 3.1 refers to a random state mealy machine which fulfills the requirements of the author's. The mealy machine has four states and has 16 transitions. The transitions are tracked using two tables. Table 3.1 refers to the state table and output table for the particular diagram.

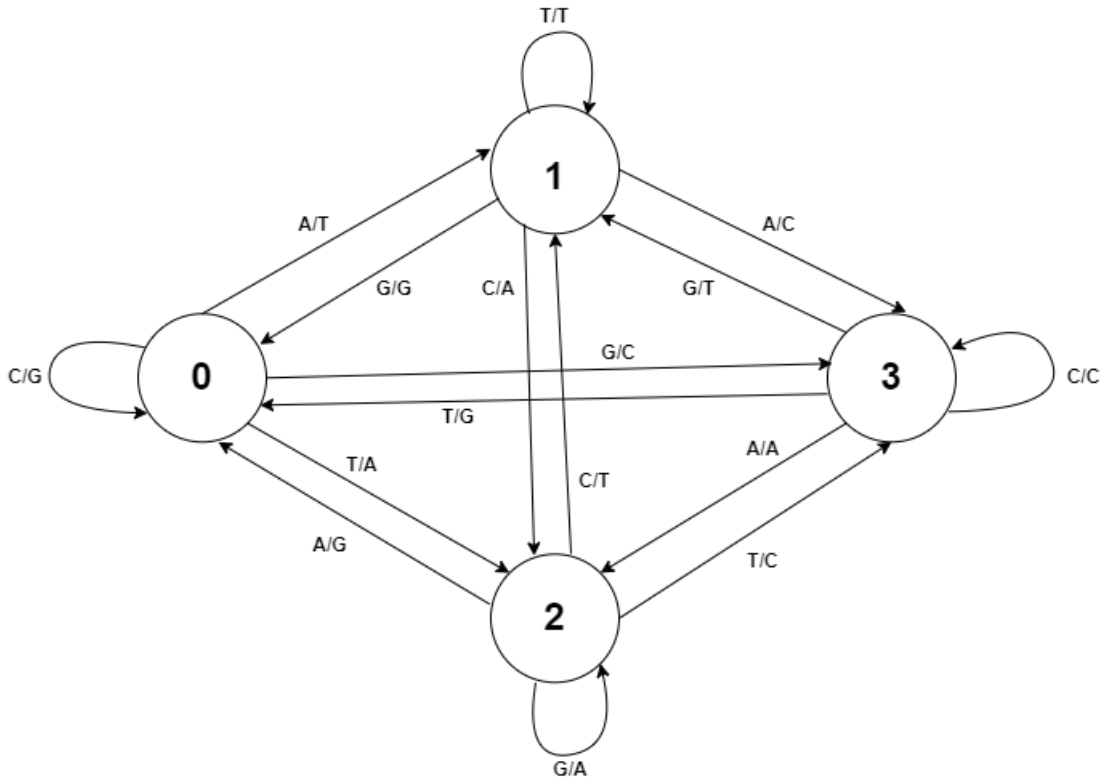


Figure 3.1: A Random State Mealy Machine

There can be ${}_4C_2 = 6$ state changes applicable for this mealy machine. State change (0,1), (0,2), (0,3), (1,2), (1,3) and (2,3) can be initiated in this particular mealy

State	A	T	C	G
0	1	2	0	3
1	3	1	2	0
2	0	3	1	2
3	2	0	3	1

(a) State Table

State	A	T	C	G
0	T	A	G	C
1	C	T	A	G
2	G	C	T	A
3	A	G	C	T

(b) Output Table

Table 3.1: Transition Tables

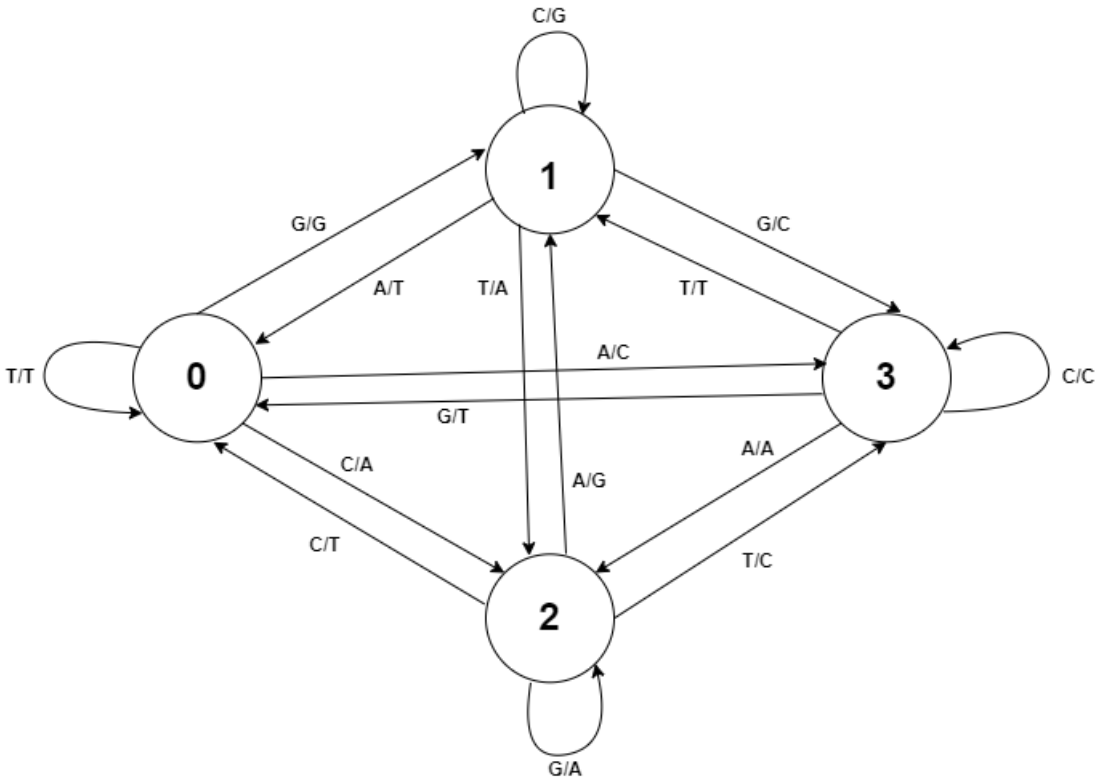


Figure 3.2: Mealy Machine After Changing State

machine. If we apply state change (0,1) then configuration will be updated. Figure 3.2 shows the updated mealy machine.
Table 3.2 shows the updated transition tables where the elements marked as red indicates that they are unchanged from the previous table despite the change in configuration. These changes in the transition table is only caused by one state change operation, it is safe to assume the configuration will be updated in a more drastic manner if n number of state change operation is done. In this work this state change concept in mealy machine strengthens the security further.

State	A	T	C	G
0	3	0	2	1
1	0	2	1	3
2	1	3	0	2
3	2	1	3	0

(a) State Table

State	A	T	C	G
0	C	T	A	G
1	T	A	G	C
2	G	C	T	A
3	A	G	C	T

(b) Output Table

Table 3.2: Transition Tables After Changing Configuration

Combination No	00	01	10	11
0	A	T	C	G
1	A	T	G	C
2	A	C	T	G
3	A	C	G	T
4	A	G	T	C
5	A	G	C	T
6	T	A	C	G
7	T	A	G	C
8	T	C	A	G
9	T	C	G	A
10	T	G	A	C
11	T	G	C	A
12	C	A	T	G
13	C	A	G	T
14	C	T	A	G
15	C	T	G	A
16	C	G	A	T
17	C	G	T	A
18	G	A	T	C
19	G	A	C	T
20	G	T	A	C
21	G	T	C	A
22	G	C	A	T
23	G	C	T	A

Table 3.3: Encoding Sequences

3.3 In-Shuffle

In-Shuffle [23] technique is a successful sequential shuffling technique where the shuffling contents are always divided into two halves and later it is merged in a sequential manner. It is widely used in casino's and other gambling chains to keep track of cards. In mathematics it has sheer significance as well. In this work the possible state change sequences list is shuffled using In-Shuffle technique Q times so that the synchronization between the sender and the receiver is intact during the encryption and decryption process as both sender and receiver are required to repeat same calculation with 100 percent accuracy. An example is provided using this work's state change sequences list for better understanding the concepts of In-Shuffle technique,
State Change Sequences List = (0,1),(0,2),(0,3),(1,2),(1,3),(2,3)
Now the list will be divided into two halves,
State Change Sequences List = (0,1),(0,2),(0,3) || (1,2),(1,3),(2,3)
Then the first element of the second half will become the first element of the list and the first element of first half will follow as the second element in the shuffled list.
State Change Sequences List = [(1,2),(0,1)] (0,2),(0,3) || (1,3),(2,3) The bracketed elements above indicates that they are shuffled now. Thye same process will repeat on the not shuffled part as the first element of the second half will join at the end of the shuffled sequence and the first element of the first half will join later,
State Change Sequences List = [(1,2),(0,1),(1,3),(0,2)] (0,3) || (2,3)
State Change Sequences List = [(1,2),(0,1),(1,3),(0,2),(2,3),(0,3)]
So the final shuffled sequence stand as,
Shuffled State Change Sequences List = (1,2),(0,1),(1,3),(0,2),(2,3),(0,3)
This In-Shuffle operation is done Q times to make the sequences untrackable for the hacker as it randomizes the sequence in each shuffle in an ordered manner.

3.4 Encryption and Decryption

Initially the sender and the receiver uses the same default configuration to configure their mealy machine. Before the encryption process is started some necessary pre-processing is done to make the plaintext free of unicode characters.After that the plaintext is divided into several N sized blocks. When a block is selected, the characters of that particular block is converted into their corresponding ASCII values. Sum of that block's all ASCII values are also calculated and stored for further use. Then each ASCII value is converted into its 8 bit binary value. Then the aforementioned security key is XORed with the binary sequence, thus a new binary sequence is generated.
Then the updated binary value sequence is encoded into DNA bases using the encoding sequence from Table 3.3 that matches the current encoding sequence number which is calculated using the following formula.

$$(\sum ASCII\ Values\ of\ Current\ Block + \sum ASCII\ Values\ of\ All\ Previous\ Blocks)\ mod\ 24$$

. With each block, this encoding sequence number changes as it is dependant on both the sum of ASCII values of the current block and the sum of ASCII values

of all the blocks that has been encrypted so far. From Table 3.3 we can see that every two binary digit is converted into oe DNA base. So each character is encoded into four DNA bases. Then that DNA sequence is inserted into the mealy machine and a new DNA sequence has been generated. After that the encrypted ciphertext is obtained for that particular block. Then before encrypting the next block, the current ciphertext is stored and the configuration of the mealy machine is updated. Firstly, sum of a mathematical series is calculated for m terms where m is the value of the summation of ASCII values for the current block. The sum of the series up to m gives a very big value, that is why it is decreased by doing a mod operation, which gives an arbitrary value q_1. In this work fibonacci series is used and 500 is used as the mod value. As it is discussed before, there can be a combination of six state change operations for this work. The six state change operations are inserted into a list and the list is shuffled using In-Shuffle technique q_1 times. After that a loop is taken with the condition of it running q_1 times. In each iteration the loop shuffles the state change combinations list once, picks a state change by performing a mod using value 6, and lastly initiates the obtained state change. The loop runs q_1 times hence the the configuration of mealy machine is updated q_1 times. After the loop is ended, a newly configured mealy machine is found. Then the the state change operation is done for another q_2 times where q_2 is calculated as Total Sum of all blocks encrypted so far mod Sum of current block. If q_2 is found to be zero then an arbitrary value z is added with q_2. This process is done to prevent hacker from extracting the plaintext accurately if the hacker guesses the sum of the ASCII values of a particular block. The hacker would still be required to guess the sum of ASCII values of all the blocks encrypted so far. **Algorithm 3** shows the process of changing mealy machine configuration. After this process is done, another newly configured mealy machine is found and it is then used for encrypting the next block of text. And the whole encryption process runs until the plaintext is ended. In this encryption process, the value of q_1 and q_2 will not be same every time which makes it difficult for the hackers to keep track of how many state changes are getting initiated and in which sequence they are initiating.

The hacker needs to get it correct for straight q_1 and q_2 times separately and any mismatch in between will result in complete failure. Furthermore the hacker needs to guess the exact sum of ASCII values of the current block and also the sum of ASCII values of all the blocks that are encrypted so far to figure out the mealy machine's configuration that will be used to encrypt the next block of text. They also need to guess the correct encoding sequence that is used to convert the DNA bases into binary value and vice versa. Each block uses different encoding sequence so determining the current encoding sequence is necessary, which is also dependant on the sum of ASCII values of the current block and the sum of ASCII values of all the blocks that are encrypted so far. The hackers are also required to know the key which adds another extra layer of security. **Algorithm 2** shows the whole process that has been followed for the encryption.

As for the decryption, the whole encryption process is reversed. The receiver starts with the same mealy machine configuration as the sender, so it doesn't bother the receiver to calculate and configure the same way that the sender's side does. The block size for the receiver is 4N as it was discussed before that each character is converted into four DNA bases. With each block of ciphertext the receiver successfully decrypts, they will be able to find the sum of ASCII values for that particular

Algorithm 2 Encryption Algorithm

1. START
2. **while** *plaintext not ended*
3. **if** $N < length(remaining\ plaintext)$
4. $block \leftarrow plaintext\ of\ length\ N$
5. **else**
6. $block \leftarrow plaintext\ of\ remaining\ length$
7. **end if**
8. $block\_ascii \leftarrow convert\_ascii(block)$
9. $SUM \leftarrow \sum All\ ASCII\ Values\ of\ block$
10. $TOTAL\_SUM \leftarrow \sum All\ Previous\ SUM$
11. $block\_binary \leftarrow convert\_binary(block\_ascii)$
12. $key \leftarrow Binary\ Key\ of\ block\_binary\ Length$
13. $block\_xor \leftarrow XOR(key, block\_binary)$
14. $block\_dna1 \leftarrow dna\_encoding(Current\_Encoding\_Sequence, block\_xor)$
15. $block\_dna2 \leftarrow mealy\_machine\_sequence(block\_dna1)$
16. $Final\ Cipher\ Text \leftarrow Final\ Cipher\ Text + block\_dna2$
17. $mealy\_machine \leftarrow state\_change\_algorithm(SUM)$
18. $quantity \leftarrow TOTAL\ SUM$ **mod** SUM
19. **if** $quantity == 0$
20. $quantity \leftarrow quantity + Z$
21. **end if**
22. $mealy\_machine \leftarrow state\_change\_algorithm(quantity)$
23. $Current\_Encoding\_Sequence \leftarrow (SUM + TOTAL\_SUM)$ **mod** 24
24. **end while**
25. $return\ Final\_Cipher\_Text$
26. END

block and will be able to configure the mealy machine accordingly to decrypt the next block of text. Thus the decryption operation is the complete opposite to the encryption operation and involves the same set of operations which was done in the encryption. **Algorithm 4** shows the exact process that happens during the decryption.

Algorithm 3 State Change Algorithm

1. START
2. $Series\_sum \leftarrow \sum_{1}^{ASCII\_SUM} Fibonacci\ Series\ Sequence$
3. $Quantity \leftarrow Series\_sum$ **mod** X
4. **while** $iteration < Quantity$
5. $\quad in\_shuffle(State\ Change\ Combination\ List)$
6. $\quad iteration \leftarrow iteration + 1$
7. **end while**
8. **while** $iteration2 < Quantity$
9. $\quad in\_shuffle(State\ Change\ Combination\ List)$
10. $\quad Current\ State\ Change\ Sequence \leftarrow ASCII\_SUM$ **mod** 6
11. $\quad state\_change(Mealy\ Machine, Current\ State\ Change\ Sequence)$
12. $\quad iteration2 \leftarrow iteration2 + 1$
13. **end while**
14. END

Algorithm 4 Decryption Algorithm

1. START
2. **while** *ciphertext not ended*
3. **if** $4N < length(remaining\ ciphertext)$
4. $block \leftarrow ciphertext\ of\ length\ 4N$
5. **else**
6. $block \leftarrow ciphertext\ of\ remaining\ length$
7. **end if**
8. $block\_dna \leftarrow mealy\_machine\_sequence(block)$
9. $block\_binary \leftarrow dna\_encoding(Current\_Encoding\_Sequence, block\_dna)$
10. $key \leftarrow Binary\ Key\ of\ block\_binary\ Length$
11. $block\_xor \leftarrow XOR(key, block\_binary)$
12. $block\_ascii \leftarrow convert\_ascii(block\_xor)$
13. $SUM \leftarrow \sum All\ ASCII\ Values\ of\ block$
14. $TOTAL\_SUM \leftarrow \sum All\ Previous\ SUM$
15. $block\_plaintext \leftarrow reverse\_ascii(block\_ascii)$
16. $Full\ Plain\ Text \leftarrow Full\ Plain\ Text + block\_plaintext$
17. $Current\_Encoding\_Sequence \leftarrow (SUM + TOTAL\_SUM)$ **mod** 24
18. $mealy\_machine \leftarrow state\_change\_algorithm(SUM)$
19. $quantity \leftarrow TOTAL\ SUM$ **mod** SUM
20. **if** $quantity == 0$
21. $quantity \leftarrow quantity + Z$
22. **end if**
23. $mealy\_machine \leftarrow state\_change\_algorithm(quantity)$
24. **end while**
25. $return\ Full\_Plain\_Text$
26. END

Chapter 4

Case Study

Here an example is provided to track the step by step process in both encryption and decryption. The case study provides a detailed simulation regarding what is going on in the background in both encryption and decryption process.

4.1 Encryption Process

The authors selected "Hello_World" as the plaintext that is going into the encryption process. The entire process will start after giving the input plaintext and will end after getting the combined ciphertext. The first part of the process would be key generation.

4.1.1 Key Generation

Before going into that the key would be generated using the Key Generation Algorithm as mentioned in Algorithm 3. Since the length of the input string is 11, for this particular simulation the authors determines the block size N would be 6 for the input characters for ease of understanding the whole process. Hence the size of the key would be 8N, as each character is converted into their 8 bit binary counterpart. That is why in this case a key is generated of length 48. Two random binary sequence is generated at first, if we call them **b1** and **b2** then let,

b1 = 111001011010100011100111010010001110010010110100
b2 = 010101100011101011111010111010000010011010001101

After that genetic algorithm is deployed and the best key is determined using the fitness score which uses run test of randomness for generating scores of the two binary sequences. The process is continued for 100000 generation as mentioned in Algorithm 3 and the key is randomized countless times in the process which is not feasible to mention. After the maximum number of generation is reached, the final key is found which is mentioned below,

Final Key = 000101101000001110111000001010000001000100100000

The fitness score of the key is found to -0.213 which well in between the range -1.96 to 1.96 and can be declared as random. The key can be shared between the sender and the receiver using either a secure channel or using Diffie-Hellman Key Exchange algorithm to share it through an open channel.

4.1.2 Encryption

Now that the key is found the encryption process can begin. The input text is inserted.

Input Text = Hello_World

The configuration of the mealy machine is initially same for both sender and the receiver. The authors let the initial configuration be the configuration mentioned in Table 3.1. Firstly, the first block of plaintext is taken for the encryption process. In this case which is,

Plain Text Block1 = Hello_

After that the characters in current block is converted into their corresponding ASCII values,

ASCII Block1 = 72 101 108 108 111 95

After that the sum of the ASCII values are calculated for further usage and is also added to the total sum of all block's ASCII values so for,

Sum Block1 = 595
Total Sum = 595

After that the ASCII values are converted into corresponding 8 bit binary values and later XOR operation is done with the Final Key,

Binary Block1 = 01001000011001010110110001101100011011101011111
XORed Binary Block1 = 01011110111001101101010001000100011111100111111

After that the binary values are converted to DNA bases using an encoding sequence from Table 3.3. In the first block the combination number 0 is used,

Current Encoding Combination Number = 0
Current Encoding Combination = ['A', 'T', 'C', 'G']
Encoded DNA Block1 = TTGCGCTCGTTATATATGGCTGGG

Then the initial DNA sequence is inserted to mealy machine to find the encrypted text for block1 using the transition tables 3.1 which is also added with the total ciphertext which will be the ciphertext of the entire plaintext,

Encrypted Block1 = ACTAATTAACGTTCGTTGCCGCTG
Total Ciphertext = ACTAATTAACGTTCGTTGCCGCTG

After the encryption process is done for block1, the process of updating the configuration is started to the mealy machine and other attributes ready for the next phase of encryption. Firstly the current combination number is updated using the mentioned formula in Algorithm 2,

Current Combination Number = Sum mod 24 = 595 mod 24 = 19

Then the value of the Sum is used to modify the mealy machine's configuration using Algorithm 3. Firstly the Sum is used to calculate the sum of fibonacci series up to Sum term, then the operation fibonacci series sum mod an arbitrary value X, in this case 500 is used as the value X to find the Quantity,

Quantity = Fibonacci Series Sum mod 500 = 298

Then the In Shuffle operation is done Quantity times on the State Change Sequence

Combination List to modify the list,

Initial List = [(0,1),(0,2),(0,3),(1,2),(1,3),(2,3)]
Modified List = [(1, 2), (0, 1), (1, 3), (0, 2), (2, 3), (0, 3)]]

After that a loop is taken to run for Quantity times and in each iteration the in shuffle operation is done on that list once and a state change sequence is selected and then initiated,

Again Modified List = [(0, 2), (1, 2), (2, 3), (0, 1), (0, 3), (1, 3)]
Stage Change Sequence Number = Sum mod 6 = 595 mod 6 = 1

The element in index 1 of the modified state change sequence list is (1, 2), hence that state change operation is initiated. These operations are done Quantity times, the final configuration of the mealy machine is given in Table 4.1,

State	A	T	C	G
0	1	3	2	0
1	2	0	1	3
2	3	2	0	1
3	0	1	3	2

(a) State Table

State	A	T	C	G
0	G	C	T	A
1	T	A	G	C
2	C	T	A	G
3	A	G	C	T

(b) Output Table

Table 4.1: Transition Tables After Quantity

After that Quantity2 is calculated and the whole stage changing process is done again for Quantity2 times,

Quantity2 = Total Sum mod Sum = 0 mod 595 = 0 + 10 = 10

As Quantity2 is found to be zero, an arbitrary value Z is added with it. In this case the value of Z is 10. The modified transition tables after the same operation is done Quantity2 times are given in Table 4.2,

State	A	T	C	G
0	3	0	1	2
1	2	3	0	1
2	0	1	2	3
3	1	2	3	0

(a) State Table

State	A	T	C	G
0	C	T	A	G
1	G	C	T	A
2	T	A	G	C
3	A	G	C	T

(b) Output Table

Table 4.2: Transition Tables After Quantity2

All the configuration are now set for encrypting the next block of plaintext. The process is mentioned descriptively below,

Plain Text Block2 = World
ASCII Block2 = 87 111 114 108 100
Binary Block2 = 0101011101101111011100100110110001100100

Here the length of the block is less than N which is 5. That is why the key length

is also modified to adjust the length of the length of Block2 and is taken from the first 8N bits, in this case this is 40,

Key = 00010110100000111011100000101000000010001
XORed Binary Block2 = 0100000111101100110010100100010001110101
Current Encoding Combination Number = 19
Current Encoding Combination = ['G', 'A', 'C', 'T']
Encoded DNA Block2 = AGGATCTGTGCCAGAGATAA
Encrypted Block2 = CTGTTACTTGGGTGTGTTCA
Total Ciphertext = ACTAATTAACGTTCGTTGCCGCTGCTGTTACTTGG GTGTGTTCA

After that the configurations are modified to make it ready for encrypting the next block of plaintext, since the plaintext is finished so the authors are not inserting the details further. The final ciphertext that is found is,

Final Ciphertext = ACTAATTAACGTTCGTTGCCGCTGCTGTTACTTGG GTGTGTTCA

This encrypted text can be transmitted through an open channel as hackers are unable to retrieve the information from the encrypted text. The decryption process will take this ciphertext as input and will return the plaintext that is inserted.

4.2 Decryption Process

The encrypted text that is found at the end of the encryption process works as the input in the decryption process. Algorithm 4 is used for decryption process. All the initial configurations for the receiver has to be exactly the same as the sender's initial configuration. The input text is inserted,

Input Text = ACTAATTAACGTTCGTTGCCGCTGCTGTTACTTGGGT GTGTTCA

As mentioned before, the configuration of the mealy machine is initially same for both the sender and the receiver. The initial configuration is as mentioned in Table 3.1. Firstly, the first block of ciphertext is taken for the decryption process. In this case which is,

Cipher Text Block1 = ACTAATTAACGTTCGTTGCCGCTG

Then the DNA sequence is inserted into the mealy machine and a new sequence is generated using the transition tables 3.1,

New DNA Block1 = TTGCGCTCGTTATATATGGCTGGG

After that the DNA bases are converted into binary values using a decoding sequence from Table 3.3. The encoding operation is reversed here as the DNA bases are substituted using their binary value as per the decoding combination. As mentioned before, in the first block the combination number 0 is used,

Current Decoding Combination Number = 0
Current Decoding Combination = ['A', 'T', 'C', 'G']
Decoded Binary Block1 = 0101111011100110110101000100010001111110011111111

After that the XOR operation is done using the Final Key on the retrieved Binary sequence,

XORed Binary Block1 = 01001000011001010110110001101100011010111101011111

After that the binary sequence is divided into 8-bit binary blocks and then are converted into their corresponding ASCII values,

ASCII Block1 = 72 101 108 108 111 95

After that the sum of the ASCII values are calculated for further usage and is also added to the total sum of all block's ASCII values so for. And later the ASCII values are converted into their corresponding characters,

Sum Block1 = 595
Total Sum = 595
Plaintext Block1 = Hello_

After the decryption process is done for block1, the process of updating the configuration is started for the mealy machine and other attributes to make it ready for the next phase of encryption. Firstly the current combination number is updated using the mentioned formula in Algorithm 4,

Current Combination Number = Sum mod 24 = 595 mod 24 = 19

Then the value of the Sum is used to modify the mealy machine's configuration using Algorithm 3. The same process is used as mentioned in Encryption using the same arbitrary value 500,

Quantity = Fibonacci Series Sum mod 500 = 298

Then the In Shuffle operation is done Quantity times on the State Change Sequence Combination List to modify the list the same way,

Initial List = [(0,1),(0,2),(0,3),(1,2),(1,3),(2,3)]
Modified List = [(1, 2), (0, 1), (1, 3), (0, 2), (2, 3), (0, 3)]

After that a loop is taken to run for Quantity times and in each iteration the in shuffle operation is done on that list once and a state change sequence is selected and then initiated,

Again Modified List = [(0, 2), (1, 2), (2, 3), (0, 1), (0, 3), (1, 3)]
Stage Change Sequence Number = Sum mod 6 = 595 mod 6 = 1

The element in index 1 of the modified state change sequence list is (0, 2), hence that state change operation is initiated. These operations are done Quantity times, the final configuration of the mealy machine is given in Table 4.3 which is exactly the same as the encryption process after encryption of block1 was done,

State	A	T	C	G
0	1	3	2	0
1	2	0	1	3
2	3	2	0	1
3	0	1	3	2

(a) State Table

State	A	T	C	G
0	G	C	T	A
1	T	A	G	C
2	C	T	A	G
3	A	G	C	T

(b) Output Table

Table 4.3: Transition Tables After Quantity

After that Quantity2 is calculated and the whole state changing process is done

again for Quantity2 times. The same arbitrary value 10 is used upon Quantity2 being found as zero,

Quantity2 = Total Sum mod Sum = 0 mod 595 = 0 + 10 = 10

The modified transition tables after the same operation is done Quantity2 times are given in Table 4.4 which also matches the Encryption operation's tables,

State	A	T	C	G
0	3	0	1	2
1	2	3	0	1
2	0	1	2	3
3	1	2	3	0

(a) State Table

State	A	T	C	G
0	C	T	A	G
1	G	C	T	A
2	T	A	G	C
3	A	G	C	T

(b) Output Table

Table 4.4: Transition Tables After Quantity2

All the configuration are now set for decrypting the next block of ciphertext. The process is mentioned descriptively below,

Cipher Text Block2 = CTGTTACTTGGGTGTGTTCA
Decoded DNA Block2 = AGGATCTGTGCCAGAGATAA
Current Encoding Combination Number = 19
Current Encoding Combination = ['G', 'A', 'C', 'T']
Binary Block2 = 01000001111011001100101001000100011101 01

Here the length of the block is less than N which is 5. That is why the key length is also modified same as encryption,

Key = 00010110100000111011100000101000000010001
XORed Binary Block2 = 01010111011011110111001001101100011001 00
ASCII Block2 = 87 111 114 108 100
Decrypted Block2 = World
Total Ciphertext = Hello_World

After that the configurations are modified to make it ready for decrypting the next block of plaintext, since the plaintext is finished so the authors are not inserting the details further. The final ciphertext that is found is,

Final Ciphertext = Hello_World

The retrieved result ensures that this work is capable of transmitting data securely through an open channel.

Chapter 5

Result & Analysis

5.1 Experimental Setup

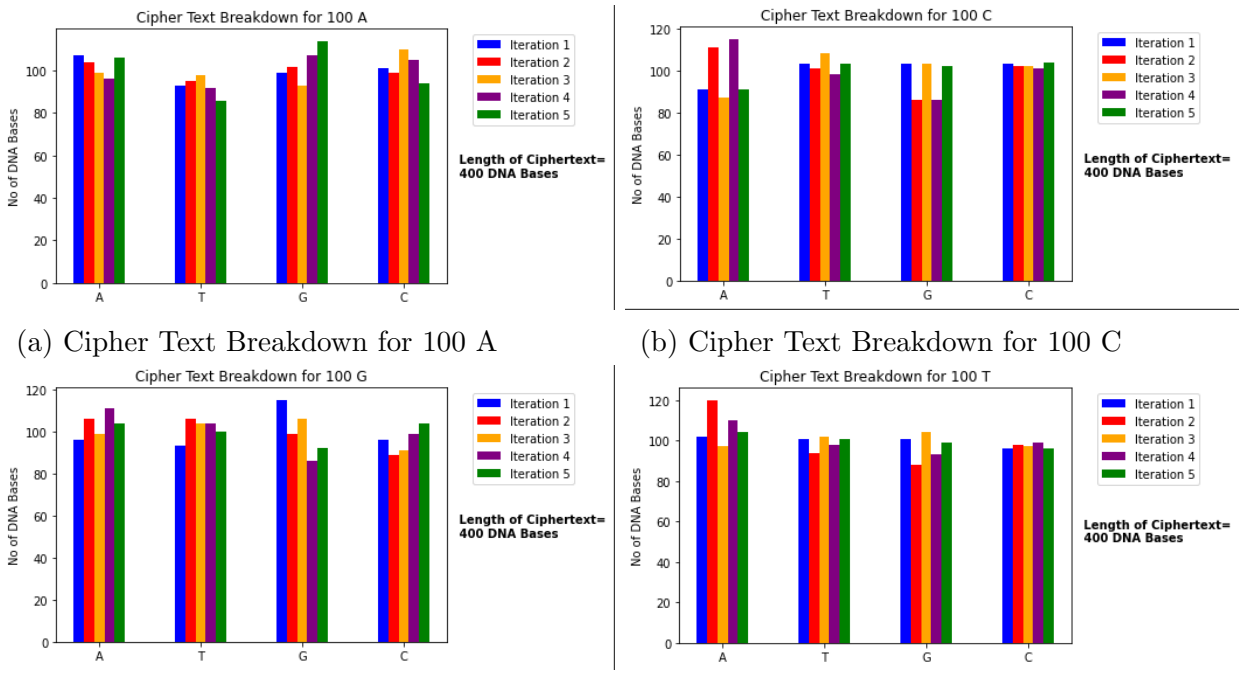
The proposed system is implemented using Python3 on Google Colab environment. The data that is used to generate test results is taken from the 20 Newsgroup Dataset [22], which is a very much widely used dataset across several research works. As this work focused on text data only, that is why that dataset is considered as suitable for generating test results

5.2 Prevention Against Attacks

The authors worked on increasing security of the system that was built. The authors also focused building a system that is going to resist various types of attack. How the system resists various types of attack and some overview of the system's security is discussed ahead.

5.2.1 Brute-Force Attack

Brute Force attack (Salamatian et al.) [15] refers to trying to break the cipher by guessing all the secret information. In this work this is infeasible because the attacker will be needing to guess 2^n combinations to find the key. The attacker also have to guess the encoding sequence for each block which can be any among the 24 encoding sequences shown in Table **3.3**. Also the attacker will have to guess the exact flow of the state change operations that happens in two sequences after encrypting each block and each sequence involves multiple state change operations. Without guessing the exact configuration of the mealy machine the hacker will not be able to decrypt the next block and so on. The exact amount of state change operation in a sequence depends on the sum of the ASCII values of a block and also the following sequence depends on all the block's combined sum of ASCII values. The probability for guessing the sum of a block is $\prod_{i=1}^{n} 1/128$, where n is the size of the block. The probability for guessing the sum of all the block's ASCII values is $\prod_{i=1}^{Z} \prod_{i=1}^{n} 1/128$, here Z is the total number of blocks up to this point, which is infeasible, not to mention the attacker also have to have the correct configuration of the mealy machine even if they guess a block's information correctly in the middle. So the probability of randomly guessing a block of plaintext stands as $\frac{1}{2^n} \frac{1}{24} (\prod_{i=1}^{Z} \prod_{i=1}^{n} 1/128)$, which is much higher than Pavithran et al. [19] and is not feasible. This proves that



(a) Cipher Text Breakdown for 100 A

(b) Cipher Text Breakdown for 100 C

(c) Cipher Text Breakdown for 100 G

(d) Cipher Text Breakdown for 100 T

Figure 5.1: Frequency of DNA Bases in Selected Ciphertexts for Small Data

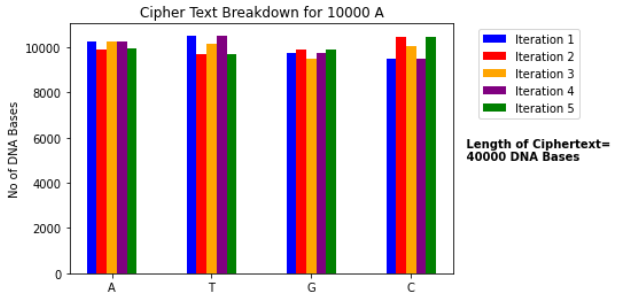
proposed scheme's security trait is not only dependant on security key, but on other features as well.

5.2.2 Known Plaintext Attack

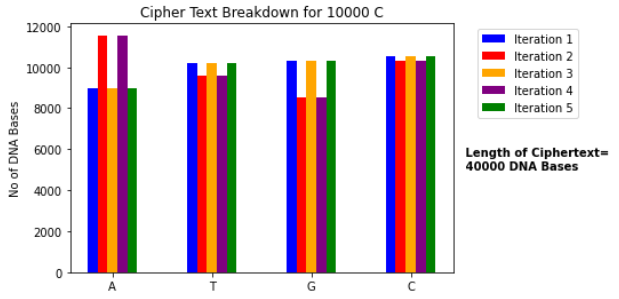
Known plaintext attack (Peng et al.) [3] is about finding a relation or pattern between the ciphertext and the plaintext. In this work this is not possible. The authors tested the system for three types of data. First one is for small data which includes 100 A's, 100 T's, 100 C's and 100 G's, second on is medium data, which includes 10000 DNA bases of each kind and last one is for big data which includes 100000 DNA bases for each kind. All types of data was encrypted five times each and the generated results were differential on quantities of DNA bases present in the ciphertext. Figure **5.1**, **5.2** and **5.3** shows that the DNA base frequency is not identical at all, each time it provides different frequencies. And it also shows that the DNA bases are mostly evenly distributed hence there is no question of any bias. Every types of data shows the same consistency on distribution which proves the effectiveness of proposed system. So finding any type of pattern or relation will be fruitless in this system.

5.2.3 Ciphertext Only Attack

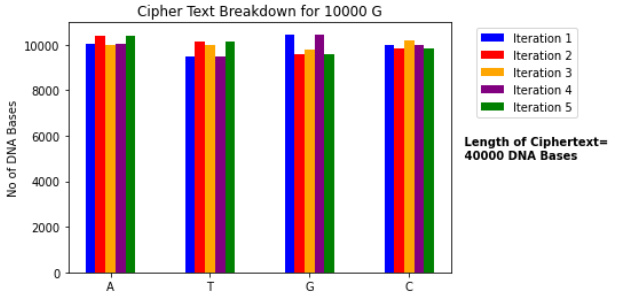
Ciphetext only attack (Chang et al.) [16] refers to trying to guess some information such as secret key with having some sort of idea about plaintext. In this work this would be futile because in Figure **5.1**, **5.2** and **5.3** the authors established that statistical analysis is fruitless in this system, also the mealy machine's configuration is continuously changing, hence making it infeasible for considering any sort of analysis even if some part of plaintext is guessed correctly.



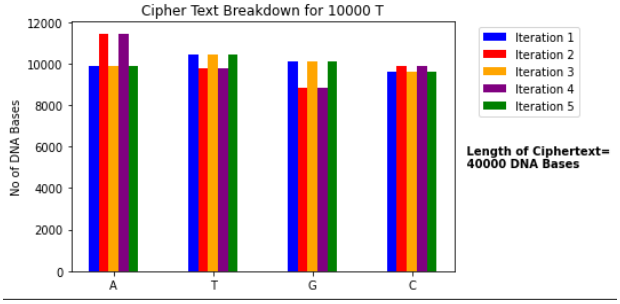
(a) Cipher Text Breakdown for 10000 A



(b) Cipher Text Breakdown for 10000 C

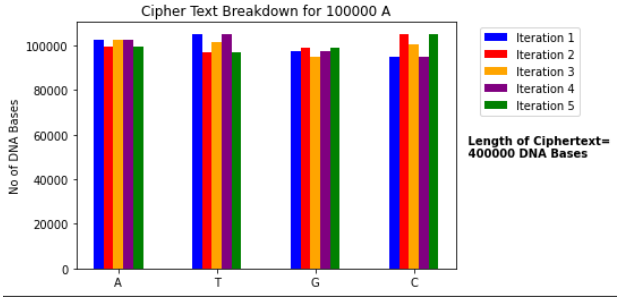


(c) Cipher Text Breakdown for 10000 G

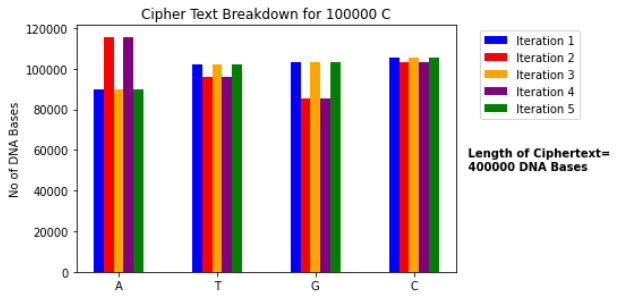


(d) Cipher Text Breakdown for 10000 T

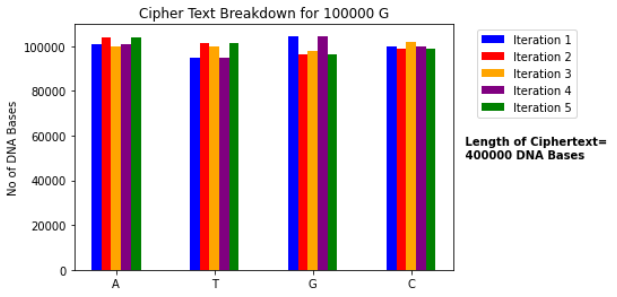
Figure 5.2: Frequency of DNA Bases in Selected Ciphertexts for Medium Data



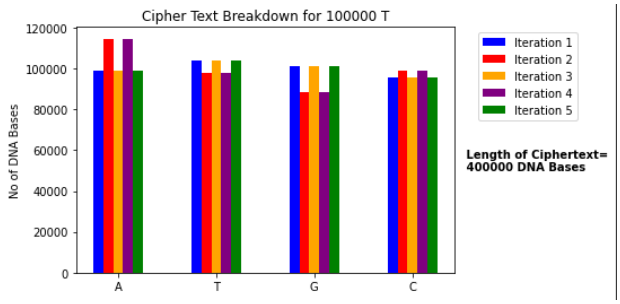
(a) Cipher Text Breakdown for 100000A



(b) Cipher Text Breakdown for 100000C



(c) Cipher Text Breakdown for 100000G



(d) Cipher Text Breakdown for 100000T

Figure 5.3: Frequency of DNA Bases in Selected Ciphertexts for Big Data

Plaintext Length	Kaundal et al. [7]		Pramanik et al [6]		Paul et al.[8]	
	Encryption Time (ms)	Decryption Time (ms)	Encryption Time (ms)	Decryption Time (ms)	Encryption Time (ms)	Decryption Time (ms)
10	15	37	20	49	18	42
20	27	44	35	68	28	57
40	38	110	49	159	42	123
80	70	270	96	414	84	310
100	96	365	105	530	112	415
500	420	1660	466	1768	486	1896

Table 5.1: Time taken for Encryption and Decryption Part 1

Plaintext Length	Pavithran et al. [19]		Proposed Work	
	Encryption Time (ms)	Decryption Time (ms)	Encryption Time (ms)	Decryption Time (ms)
10	11.31	5.87	8.17	6.38
20	16.42	9.57	25.53	24.78
40	31.11	24.75	50.61	42.98
80	62.27	49.11	74.55	73.61
100	85.45	61.25	88.60	90.07
500	312.45	245.75	385.50	392.74

Table 5.2: Time taken for Encryption and Decryption Part 2

5.2.4 Man in The Middle Attack

Man in the middle attack (Mallik et al.) [14] refers to an attacker intercepting the communication between two or multiple persons. The attacker may try to impersonate someone or try to corrupt the data anyhow, the main goal of the attacker is to mislead the people and trying to get sensitive information from them by impersonating as someone else. In this work the key is exchanged using either a secure channel or using a secure and trusted system (Diffie-Hellman Algorithm) through an open channel. This cancels out the idea that key can be intercepted by a third party. Also the mealy machine's configuration between the sender and the receiver would be exactly identical, therefore it would not be possible for anyone to guess that configuration and impersonate as someone else, this would result in decrypted message being gibberish hence they would not be able to extract any useful information.

5.2.5 Differential Cryptanalysis Attack

Differential cryptanalysis attack (Cao et al.) [13] refers to analyzing pairs of plaintext and ciphertext to retrieve the key or canceling out combinations in order to get boost in finding the key. In this work the plaintext is protected not only by the key but also the state changing mealy machine which changes states frequently and encrypts the same text differently in each time. Also different encoding combinations are used for encoding different blocks of plaintext, which makes it much more difficult. Also the key is proven to be certified random and is shared using a secured channel or a secured method hence this attack would not work in this system.

Plaintext	Change in Ciphertext
100 A's	72%
100 T's	85%
100 G's	82%
100 C's	79%

(a) Pavithran et al. [19]

Plaintext	Change in Ciphertext
100 A's	94%
100 T's	94%
100 G's	99%
100 C's	91%

(b) This Work

Table 5.3: Changes in Decrypted Text for 1-bit Change in Key for Small Data

Plaintext	Change in Decrypted Text
10000 A's	94.89%
10000 T's	94.73%
10000 G's	94.42%
10000 C's	94.23%

Table 5.4: Changes in Decrypted Text for 1-bit Change in Key for Medium Data

5.3 Encryption and Decryption Time

As mentioned before, this work focused on building a more secure system hence the core constraints face some trade off among time and security feature. The authors did their best on maximizing the security trait while keeping the time in check. Table **5.1** shows the amount of time taken by existing systems against several length of text data. This work's data along with Pavithran et al. [19] is visible in Table **5.2**, which shows this work outperforms all the existing works shown in Table **5.1** regarding time efficiency while it fell short to Pavithran et al. Point to be noted here is this work focused on increasing security trait and performs significantly well than Pavithran et al. which will be discussed in the next section. This work faces a trade off regarding time for adding the extra security traits such as adding different encoding sequences for different blocks and introducing state change operation to change the configuration of the mealy machine before encrypting a block of plaintext. Adding those traits means that the same set of operations are done for both encryption and decryption, that explains the amount of time taken for decryption for this work. While the difference is not significantly high between this work and Pavithran, this work outperforms previously mentioned work which is discussed in the next section.

5.4 Avalanche Effect

Avalanche effect is an important cryptographic property which states that for only a tiny amount of change in crucial information such as key or other properties, how much change it reflects into the resulting decrypted text. In this work the authors measured the avalanche effect for changing bits of key, modifying the state change operation and modifying encoding operation.

Plaintext	Change in Decrypted Text
100000 A's	95.26%
100000 T's	94.38%
100000 G's	94.32%
100000 C's	94.34%

Table 5.5: Changes in Decrypted Text for 1-bit Change in Key for Big Data

Plaintext	Change in Decrypted Text
100 A's	89%
100 T's	88%
100 G's	93%
100 C's	95%

Table 5.6: Changes in Decrypted Text for Modifying A State Change Operation for Small Data

5.4.1 Changing 1 Key Bit

In this test only one bit of the security key was flipped and then the decryption operation was done using the modified security key. Table 5.3 shows a comparison between Pavithran et al. [19] and this work regarding how much change occurred in the resulting decrypted text when 1-bit in key is changed during the decryption process. The lowest amount of change is recorded as 91% while it was 72% in Pavithran's work, the difference is significantly huge. The highest amount of change is recorded as 99% for this work while the highest amount in Pavithran's work being 85%. This proves that the proposed scheme provides better security than the mentioned work. The avalanche effect for 1-bit change in key is also tested for medium and big data and they also showed significant consistency. For medium data containing 10000 bases each in plaintexts the lowest amount of change being 94.23% is even higher than the lowest amount of change for small data, while the highest amount of change for medium data being 94.89%. As for the big data which contains 100000 bases for each plaintexts records the lowest amount of change being 94.32% and the highest being 95.26%. These stats record the consistency of the proposed scheme for any size of data.

5.4.2 Modifying State Change Operation

Proposed scheme also has multiple security features added, one of which is state change operation on mealy machine. This experiment is done by tweaking one state change operation during the decryption process but every other attribute like security key is kept unchanged. This experiment is also done for three types of data, small data includes 100 DNA base for each, medium data includes 10000 DNA bases for each and big data includes 100000 DNA bases each. Table **5.6**, **5.7** and **5.8** shows the changes in decrypted text if one state change operation is modified and do a different state change operation in the middle of changing the configuration of the mealy machine. For small data the minimum amount of change

Plaintext	Change in Decrypted Text
10000 A's	92.38%
10000 T's	91.10%
10000 G's	92.31%
10000 C's	93.72%

Table 5.7: Changes in Decrypted Text for Modifying A State Change Operation for Medium Data

Plaintext	Change in Decrypted Text
100000 A's	92.60%
100000 T's	92.42%
100000 G's	93.55%
100000 C's	94.16%

Table 5.8: Changes in Decrypted Text for Modifying A State Change Operation for Big Data

is 89% while the maximum is 95%. For medium data the lowest amount is 91.10% while the highest being 93.72%. For big data the lowest amount is 92.42% and the highest amount stands at 93.55%. These three types of data shows that proposed work has consistency and it does not depend on only one security feature, that the system is secure on many aspects.

5.4.3 Modifying Encoding Operation

Another security trait of this work is using different encoding sequence for different blocks. This feature also adds significant amount of security to the system and is also tested for avalanche effect. This test is also done for three types of data. One encoding operation was tweaked and another combination was used for a single time during the decryption process to run this test. Table **5.9**, **5.10** and **5.11** shows the result data of the aforementioned test. For small data the minimum amount of change is 91% while the maximum is 99%. For medium data the lowest amount is 92.25% while the highest being 93.93%. For big data the lowest amount is recorded as 90.34% and the highest amount is observed at 94.04%.

These test results prove that this work is not dependant on only one security feature

Plaintext	Change in Decrypted Text
100 A's	91%
100 T's	97%
100 G's	99%
100 C's	94%

Table 5.9: Changes in Decrypted Text for Modifying An Encoding Operation for Small Data

Plaintext	Change in Decrypted Text
10000 A's	92.25%
10000 T's	93.84%
10000 G's	93.16%
10000 C's	93.93%

Table 5.10: Changes in Decrypted Text for Modifying An Encoding Operation for Medium Data

Plaintext	Change in Decrypted Text
100000 A's	90.63%
100000 T's	90.34%
100000 G's	93.07%
100000 C's	94.04%

Table 5.11: Changes in Decrypted Text for Modifying An Encoding Operation for Big Data

but multiple security features. Confiscating each of the major security features resulted in significant amount of change in decrypted text on different sizes of data. That proves this scheme works consistently for all size of data. While facing the trade off between time and enhanced security, this work increased the security trait of the system significantly than existing works while not giving away much regarding time.

Chapter 6

Conclusion

This work proposed a hybrid cryptographic system which uses a key length of N which is generated using Genetic Algorithm and is evaluated using the run test of randomness to ensure the randomness of the key. The authors ensured that the key can be shared using both through a secure channel and through an open channel using Diffie-Hellman Key Exchange algorithm. Then the plaintext goes through various cryptographic operations which involves dividing the plaintext into block sizes on length N, converting the block into corresponding ASCII values, calculate the sum of the ASCII values, converting the ASCII values into binary and adding the key and lastly converting the binary sequence to a DNA sequence which uses different encoding combinations each time and generate a new DNA sequence using a randomly generated state changing mealy machine. After that various operations are done to change the configuration of the mealy machine by initiating state change operation Q times where Q is calculated using the current block's information and then the configuration is changed again using Q2 times where Q2 is calculated using all the block's information so far. These processes add extra layer of security to the system and after the change, the mealy machine becomes ready for encryption operation for the next block and the process continues till it runs out of plaintext to encrypt. The same set of operations are also carried out in the receiver's side in a reverse manner to decrypt the received ciphertext. A case study is provided for the better understanding of the proposed method. This work proposes a unique idea which is resistant to most attacks and necessary analysis are provided to back the claim that this algorithm provides better security. For the future, the authors are determined to make this system work for other types of data such as image, sound, video and others and they are also determined to make the system more efficient and secure and provide more analysis regarding the system.

Bibliography

[1] L. M. Adleman, "Molecular computation of solutions to combinatorial problems," *Science, JSTOR*, vol. 266, pp. 1021–1025, 1994.

[2] R. J. Lipton, "Using dna to solve np-complete problems," *Science*, vol. 268, pp. 542–545, 1995.

[3] X. Peng, P. Zhang, H. Wei, and B. Yu, "Known-plaintext attack on optical encryption based on double random phase keys," *Optics letters*, vol. 31, pp. 1044–6, May 2006. DOI: 10.1364/OL.31.001044.

[4] J. Hopcroft, *Introduction to Automata Theory, Languages, and Computation* (Always Learning). Pearson Education, 2008, ISBN: 9788131720479. [Online]. Available: https://books.google.com.bd/books?id=tzttuN4gsVgC.

[5] K. Hameed, "Dna computation based approach for enhanced computing power," *International Journal of Emerging Sciences*, vol. 1, pp. 31–17, Jan. 2011.

[6] S. Pramanik and S. K. Setua, "Dna cryptography," *2012 7th International Conference on Electrical and Computer Engineering*, pp. 551–554, 2012. DOI: 10.1109/ICECE.2012.6471609..

[7] A. K. Kaundal and A. K. Verma, "Extending feistel structure to dna cryptography," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 18, no. 4, pp. 349–362, 2015. DOI: 10.1080/09720529.2014.995975. eprint: https://doi.org/10.1080/09720529.2014.995975. [Online]. Available: https://doi.org/10.1080/09720529.2014.995975.

[8] S. Paul, T. Anwar, and A. Kumar, "An innovative dna cryptography technique for secure data transmission," *International Journal of Bioinformatics Research and Applications*, vol. 12, p. 238, Jan. 2016. DOI: 10.1504/IJBRA.2016.078235.

[9] S. Kalsi, H. Kaur, and V. Chang, "Dna cryptography and deep learning using genetic algorithm with nw algorithm for key generation," *Journal of Medical Systems*, vol. 42, p. 17, Dec. 2017. DOI: 10.1007/s10916-017-0851-z.

[10] M. A. Bujang and F. Sapri, "An application of the runs test to test for randomness of observations obtained from a clinical survey in an ordered population," *Malaysian Journal of Medical Sciences*, vol. 25, pp. 146–151, Aug. 2018. DOI: 10.21315/mjms2018.25.4.15.

[11] S. Basu, M. Karuppiah, M. Nasipuri, A. K. Halder, and N. Radhakrishnan, "Bio-inspired cryptosystem with dna cryptography and neural networks," *J. Syst. Archit.*, vol. 94, pp. 24–31, 2019.

[12] M. R. Biswas, K. M. R. Alam, S. Tamura, and Y. Morimoto, "A technique for dna cryptography based on dynamic mechanisms," *J. Inf. Secur. Appl.*, vol. 48, 2019.

[13] M. Cao and W. Zhang, "Related-key differential cryptanalysis of the reduced-round block cipher gift," *IEEE Access*, vol. 7, pp. 175 769–175 778, 2019. DOI: 10.1109/ACCESS.2019.2957581.

[14] A. Mallik, A. Ahsan, M. Shahadat, and J.-C. Tsou, "Man-in-the-middle-attack: Understanding in simple words," vol. 3, pp. 77–92, Jan. 2019. DOI: 10.5267/j.ijdns.2019.1.001.

[15] S. Salamatian, W. Huleihel, A. Beirami, A. Cohen, and M. Medard, "Why botnets work: Distributed brute-force attacks need no synchronization," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2288–2299, Sep. 2019, Publisher Copyright: © 2019 IEEE., ISSN: 1556-6013. DOI: 10.1109/TIFS.2019.2895955.

[16] X. Chang, A. Yan, and H. Zhang, "Ciphertext-only attack on optical scanning cryptography," *Optics and Lasers in Engineering*, vol. 126, 105901, p. 105 901, Mar. 2020. DOI: 10.1016/j.optlaseng.2019.105901.

[17] S. Chirakkarottu and S. Mathew, "A novel encryption method for medical images using 2d zaslavski map and dna cryptography," *SN Applied Sciences*, vol. 2, Jan. 2020. DOI: 10.1007/s42452-019-1685-8.

[18] A. Hazra, C. Lenka, A. Jha, and M. Younus, "A novel two layer encryption algorithm using one-time pad and dna cryptography," in *Innovations in Computer Science and Engineering: Proceedings of 7th ICICSE*, H. S. Saini, R. Sayal, R. Buyya, and G. Aliseri, Eds. Singapore: Springer Singapore, 2020, pp. 297–309, ISBN: 978-981-15-2043-3. DOI: 10.1007/978-981-15-2043-3_35. [Online]. Available: https://doi.org/10.1007/978-981-15-2043-3_35.

[19] P. Pavithran, S. Mathew, S. Namasudra, and P. Lorenz, "A novel cryptosystem based on DNA cryptography and randomly generated mealy machine," *Computers and Security*, vol. 104, p. 102 160, Dec. 2020. DOI: 10.1016/j.cose.2020.102160. [Online]. Available: https://hal.science/hal-03671929.

[20] S. Hassan, M. A. Muztaba, M. S. Hossain, and H. S. Narman, "A hybrid encryption technique based on dna cryptography and steganography," in *2022 IEEE 13th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2022, pp. 0501–0508. DOI: 10.1109/IEMCON56893.2022.9946512.

[21] A. S. Sakr, M. Y. Shams, A. Mahmoud, and M. Zidan, "Amino acid encryption method using genetic algorithm for key generation," *Computers, Materials & Continua*, vol. 70, no. 1, pp. 123–134, 2022, ISSN: 1546-2226. DOI: 10.32604/cmc.2022.019455. [Online]. Available: http://www.techscience.com/cmc/v70n1/44413.

[22] L. K., *20 newsgroups dataset*. [Online]. Available: http://people.csail.mit.edu/jrennie/20Newsgroups/.

[23] E. W. Weisstein, "In-shuffle," *MathWorld–A Wolfram Web Resource*, [Online]. Available: https://mathworld.wolfram.com/In-Shuffle.html.