DEBRE BIRHAN UNIVERSITY

COLLEGE OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

Security Enhancement of Playfair Cipher Using Modified BBS and Keystream Values

A Thesis Submitted to the Department of Information Technology in Partial Fulfilment for the Degree of Master of Science in Computer Network and Security

BY

TEFERA ALAGAW

Advisor

Alemu Jorgi (PhD)

Debre Birhan, Ethiopia

February, 2023

DEBRE BIRHAN UNIVERSITY

COLLEGE OF COMPUTING

DEPARTMENT OF INFORMATION TECHNOLOGY

Security Enhancement of Playfair Cipher Using Modified BBS and Keystream Values

A Thesis Submitted to the Department of Information Technology in Partial Fulfilment for the Degree of Master of Science in Computer Network and Security

BY

TEFERA ALAGAW

Advisor

Alemu Jorgi (PhD)

Debre Birhan, Ethiopia

February, 2023

Approval Page

This is to certify that the thesis prepared by Mr. Tefera Alagaw entitled " **Security Enhancement of Playfair Cipher Using Modified BBS and Keystream Values** " and submitted in partial fulfilment of the requirements for the Degree of Master of Science in Computer Network and Security complies with the regulations of the university and meets the accepted standards with respect to its originality. So, it has been approved by the following examiners:

Advisor: **Signature, Date**

External Examiner: Signature, Date

Internal Examiner: Signature, Date

DEDICATION

This thesis work is dedicated to My beloved mother.

TEFERA ALAGAW

Acknowledgments

First and foremost, I want to express my gratitude to God and his Holy Mother for providing me the strength to complete this thesis. Then, I am extremely thankful to my advisor Dr. Alemu Jorgi for his invaluable help and guidance throughout the duration of the research. I am especially grateful to him for always offering constructive suggestions which carried me to achieve the completion of the research. I thank him for taking out time from his busy schedule to help me out. Without him, I could not have been done this thesis. I also would like to express my gratitude to my families, all of my friends, and other people who have been on my side and supported me in this thesis work.

Declaration

I hereby declare that this thesis is my original work performed under the supervision of Dr. Alemu Jorgi, and it has not been presented as a thesis work for a degree program in any other university and all sources of materials used for the thesis are duly acknowledged.This thesis has been submitted for examination with the approval of university advisor.

Declared by: **Signature, Date**

Confirmed by:

Advisor:

Abstract

Today, information's are shared and stored on the cloud. Thus, security concerns with this information are a big concern, and as technology advances, hackers can use these cutting-edge technologies to attack the data. In order to protect this data, it is required to use a security mechanism called cryptography. Cryptography is the process of changing intelligible data to unintelligible data using key and encryption algorithms. There are two types of cryptography namely symmetric and asymmetric. Playfair cipher is a symmetric cipher that encrypts a pair of letters at the same time. The goal of this study is to improve the security of the Playfair cipher by using a modified Blum Blum Shub algorithm and Keystream values. The Blum Blum Shub algorithm is altered using four blum prime numbers and used to generate random number and then, it will be a key after finding the equivalent character of the generated random number sequences. In this study, we securely swapped the key and resolve the need for filler characters when related characters are combined to make a bigram. Furthermore, we have also hide the relationship between plaintext and cipher text bigrams by modifying the encryption mechanism of playfair cipher. Further, here, reciever did not recieve the key directly instead it can generate the same key with that of the sender and this overcomes the key exchange problem of playfair cipher. Finally, the proposed algorithm has been evaluated through computer simulations namely Matlab software. In order to evaluate the performance of the proposed work, we have used Avalanche effect, confusion and diffusion , frequency analaysis and brute force attack as an evaluation parameter. Finally, the simulation result shown that the proposed algorithm provides a high avalanche effect ratio , generate a complex key which is difficult to predict and it require a lot of time to be cracked by cryptanalaysis attack. A single character change in the plaintext provide an average of **98.572%** of avalanche result. So, the proposed work is secured than the extended algorithm.

Keywords:Playfair, Bigram, Blum Prime, Keystream, Digraph, Median, CCM, efficacy

Contents

List of Tables

List of Figures

List of Abbreviations

AE . . . . . Avalanche Effect
ASCII . . . . . American Standard Code for Information Interchange
BBS . . . . . Blum Blum Shub algorithm
CCM . . . . . Cipher Chaining Method
DSA . . . . . Digital Signature Algorithm
ECC . . . . . Elliptic Curve Cryptography
EPAL . . . . . Existing Playfair Algorithm
ExtePA . . . . . Extended Playfair Algorithm
FCT . . . . . Final Cipher Text
GCD . . . . . Greatest Common Divisor
IEEE . . . . . Institute of Electrical and Electronics Engineers
IRSTC . . . . . Innovation for Research, Science, Technology and Culture
LCM . . . . . Linear Congruential Method
LFSR . . . . . Left Feedback Shift Register
L˙p . . . . . Length of Plaintext
Med . . . . . Median value
MPDR . . . . . Modified Playfair Decryption Rule
MPER . . . . . Modified Playfair Encryption Rule
M˙v . . . . . Modules Value
OPA . . . . . Origional Playfair Algorithm
Pl . . . . . Plaintext
PRNG . . . . . Pseudo Random Number Generator
RSA . . . . . Rivest Shamir and Adleman
SAC . . . . . Strict Avalanche Criterion
SEPCMBBSKV . . . . . Security Enhancement of Playfair Cipher using Modified BBS and Keystream Values

Chapter One

1 Introduction

1.1 Background of the Research

Today, we are in a digital world and crucial information's are shared through an unsecured channel or the internet. This transmission of data over the internet may be vulnerable to an intruder. The intruders always try to attack the transmitted data and recover the original data. In today's world, the security of information or data is indispensable to both an organization and individuals. When any information is stored or transmitted over the internet there should be some mechanism to protect that information from unauthorized persons or attackers [1]. So, we need to hide data in such a way that unintended persons can't access the shared data. To hide data one of the fundamental methods is to use cryptography or simply encryption and decryption techniques.

In etymologically speaking, the origin of the term cryptography is Greek. It is created by combining two words which are Crypto and Graphy. The word Crypto means secret and Graphy means writing [2]. Cryptography deals with preparing files or information that can be transmitted secretly through public communication channels. Cryptography is the science of the confidentiality of the message. At present, everyone wants to encrypt the information at the sender side and decrypt it at the receiver side to preserve security. So, cryptography is the science of creating an unreadable file using encryption and decryption techniques. Encryption or enciphering is the process of converting plaintext into cipher text. It is a technique that provides a cipher text for the plaintext using a key (i.e., converting information from a readable state to a scrambled form that cannot be understood). It changes the information into an unreadable text format with the help of an algorithm or encryption technique [3]. In the encryption technique, the term plaintext is used to express the original message or the readable message which is going to be transformed into another form. On the other hand, decryption or deciphering is the process of retaining the original message from the cipher message or transformed message. In the decryption technique, the word cipher text represents the unreadable form of the message or hidden message.

The function or sequence of code which works with the key to change the plaintext into cipher text is called an encryption algorithm. The reverse process of an encryption algo-

rithm which means a function that converts the cipher text into plaintext with the help of a key is called a decryption algorithm [4]. In other words, decryption is moving from the unintelligible form of text towards the audible form of text. A defined value that is used for input for encryption and decryption algorithm is referred to as a key. The following figure shows how the encryption process and decryption processes are done.

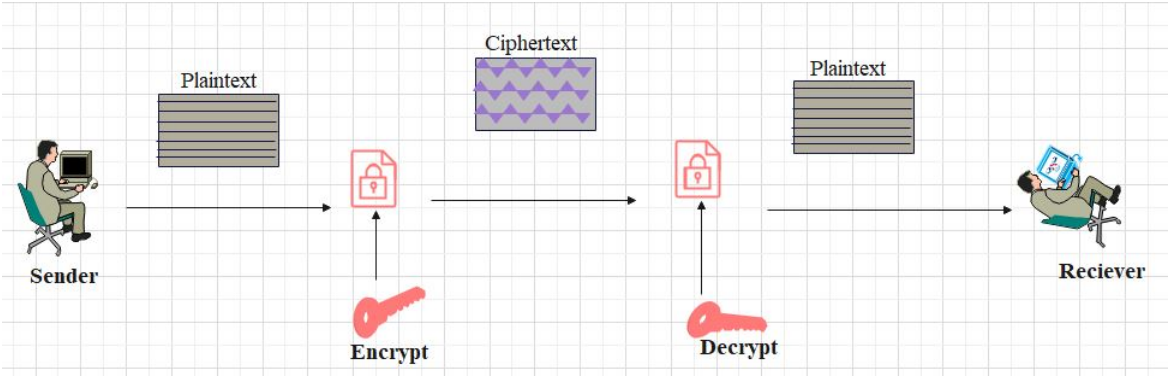


Figure 1.1: Encryption and Decryption process

Cryptology has created several types of systems for hiding texts over the last 2,500 years [5]. A cryptosystem is an algorithm that takes a key and converts intelligible data to unintelligible data and vice versa. The plaintext is what you want to hide; cipher text should appear to be random gibberish. Cryptosystems are of two types, namely symmetric and asymmetric key cryptosystems [4]. Symmetric key encryption is a cryptography technique in which both parties use a shared secret key to encrypt and decrypt the data. The shared key and encryption algorithm are used by the sender to encrypt the message. Then the receiver decrypts the message using a shared key and decryption algorithm. The main challenge with a symmetric cryptosystem is to keep the key secret while distributing it to the receiver. However, if the key is compromised by the intruder, the information becomes compromised by an attacker [6]. In an asymmetric encryption algorithm, each user uses two different but mathematically connected keys namely a public key and a private key. A public key is used to encrypt a message and is publically available and a private key is used to decrypt the message and kept safe [5]. The sender encrypts the message using the public key of the receiver and the receiver decrypts the message using his private key. The main disadvantage of asymmetric algorithms is that they are slower than symmetric algorithms. It graphically looks as follows:

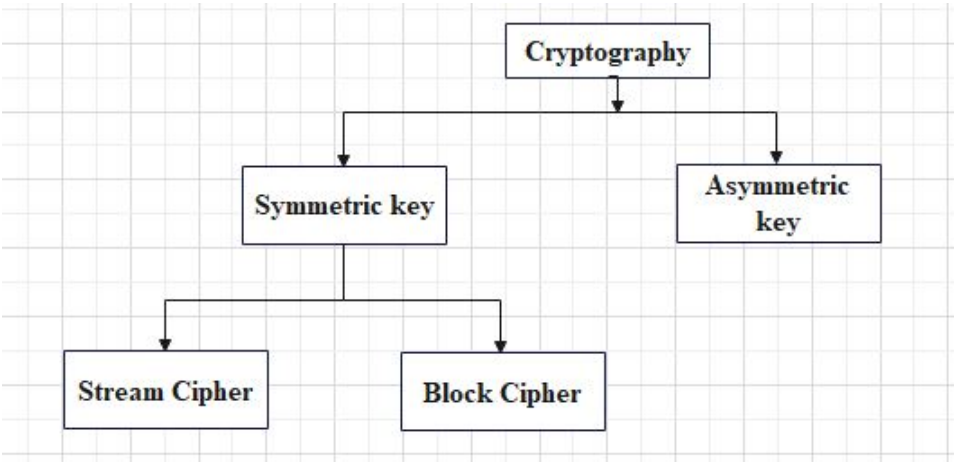


Figure 1.2: Types of Cryptography [4]

In symmetric cryptography, there are two types of cipher which are substitution cipher and transposition cipher [4]. Substitution cipher means one symbol of the plaintext is replaced by another symbol. It has further divided into two types namely monoalphabetic and polyalphabetic substitution cipher. In the monoalphabetic substitution cipher, a character in the plaintext is always changed to the same character in the cipher text. It has a one-to-one correspondence of the symbols with each other. Due to the reason of origional data frequency reflection, monoalphabetic substitution ciphers are easy to break [7]. The statistical structure of the plaintext in the cipher text is reflected because another alphabet in the cipher text is always replaced for a particular alphabet in the plaintext. CEASAR cipher is one of the well-known monoalphabetic substitution ciphers which always changed character a to character d. In a polyalphabetic substitution cipher, many characters in the cipher text are changed to a single character in the plain text. One of the well-known polyalphabetic substitution ciphers is the VIGENERE cipher which changes a single character in the plaintext into many characters in the cipher text using the encryption rule of the Vigenere cipher. In transposition cipher, the cipher character is derived by swapping the character in the plaintext which means the character retains its plaintext form but its position is changed.

Playfair cipher is one type of symmetric and polyalphabetic substitution cipher. It was primarily developed by Charles Wheatstone in 1854 from Britain but holds the name of Lord Playfair who promoted the use of this method [8]. It was valuable material during World War I and World War II [5]. In cryptography, random numbers have a great role and are

used for key generation purposes. It is possible to generate a random number using Blum Blum Shub Algorithm (BBS), Linear Congruential Method (LCM) , Left Feedback Shift Register (LFSR), etc. The whole system will be vulnerable to intruders if randomness is not generated effectively [9].

As we discussed before, today we are using the internet to share and store information since technologies are advanced. Although technologies are advanced and we use the internet, security is still a big issue. To secure this information we need to use cryptographic techniques like the playfair encryption technique. Play fair is one technique that is used to secure data but it has limitations. We become motivated to overcome this playfair limitation and develop a robust algorithm to secure information and make it difficult to break by an unauthorized person. Our study focuses on improving the security of the playfair cipher by using The Blum Blum Shub pseudo-random number generator and key stream values. We have tried to generate keys rather than the selection of keys beforehand since this leads to the compromise of keys during sharing and also we have expanded the size of the matrix used for encryption and decryption purposes.

1.2 Statement of the Problem

Playfair cipher uses a bigram or group of two characters to encrypt and decrypt messages, and it is also a symmetric encryption technique. The main challenge in symmetric encryption is to find a secured channel for the sharing of keys as both parties use the same key. Most of the time intruder attacks playfair cipher using frequency analysis attack. As we discussed in the introduction, the playfair cipher uses a 5x5 matrix and uses only uppercase characters as the content of the matrix by considering the letter I and J as one cell. From this 5x5 matrix, it is possible to generate 676 bigram combinations. However, attackers can get at least one bigram by trying all these combinations because they use a computer as an attacking tool, and trying this bigram combination becomes simple for them also putting letters I and J as one cell creates ambiguity for the receiver during decryption. In the playfair cipher, there was no possibility of encrypting alphanumeric characters and special symbols. Due to this reason, the security of the encrypted cipher was not strong and different bigrams provide similar bigram ciphers. For example, the same bigrams in small letters and capital letters provide the same cipher because the matrix only incorporates capital alphabets. This can lead to a frequency analysis attack.

In order to solve the above-listed issues, many researchers have proposed an algorithm that increases the security of the playfair cipher by extending the key matrix and by do not select the key of the matrix beforehand since the selection of the key beforehand causes compromise of the key while sharing to the receiver. Among them, the researcher in [10] proposed an extended playfair encryption technique using the Fibonacci series to encrypt small and capital letters, numbers, and two special symbols. The author increases the matrix into 8x8 key matrixes. However, there are still some drawbacks in this extended playfair algorithm. The first drawback of this extended algorithm is on generating encryption and decryption keys of the playfair algorithm using Fibonacci terms and exchanging it in an unsecured way. Since the Fibonacci term is the sum of the previous two terms, it can be easily calculated. So, an attacker can easily get the key during sharing of keys or it can easily get through calculation. Another drawback of this paper is the bigrams and their reverse provide similar cipher patterns and similar bigrams give similar cipher. This can lead to a frequency analysis attack. Another limitation of this paper is when we split the plaintext into bigrams if two similar characters make a pair, it needs to split those using

filler characters. This filler character creates ambiguity during decryption at the receiver side which means it is difficult to know whether the filler character is part of plaintext or not. In addition to this, if the plaintext length is odd then the algorithms need to add another padding letter but if that letter is a part of the plaintext, it creates ambiguity for the receiver during decryption. In playfair, it is required to hide the relationship between plaintext bigram and cipher text bigram but the existing algorithm does not hide the relationship between them. This makes the data to be vulnerable to an unauthorized person. According to Kirchhoff's principle, the secrecy of data depends on the secrecy of the key but not the algorithm. The encryption algorithm is public knowledge, while only the key remains secret [11]. So, using the Fibonacci term to generate a key makes the algorithm easily breakable since it uses a weak method to generate its key. Hence, our proposed paper intends to improve the encryption scheme of the extended playfair cipher algorithm using Fibonacci series to increase its security performance. In this paper, we have designed a 14 x 14 playfair matrix algorithm namely security enhancement of playfair cipher using modified BBS and Key stream values. This proposed algorithm answers the following questions.

1. How does the proposed algorithm enhance the security of the extended playfair cipher algorithm?
2. How can we modify the BBS algorithm using four Blum prime numbers?
3. What are the performance metrics to evaluate the proposed algorithm?

1.3 Objective

1.3.1 General Objective

The general objective of this study is to develop an algorithm that enhances the security of playfair cipher using modified BBS and key stream values.

1.3.2 Specific Objective

The specific objectives of this study include:

- To review related proposed on playfair cipher
- To investigate key stream values of each plaintext character
- To design an algorithm using modified BBS for the proposed work
- To evaluate the performance of the proposed work using performance metrics.
- To compare the proposed solution with the existing work

1.4 Scope of the study

This study is delimited to the security of playfair cipher using BBS and key stream values. The study focuses on improving the security of playfair and producing a strong cipher that would be complex for the intruder from being attacked by frequency analysis and brute force attack.

1.5 Significance of the Study

We developed an algorithm that enhances the security of the playfair cipher algorithm. This algorithm helps to:

- Provide confidentiality of information
- Provide strong cipher
- Keys would not be compromised by the intruder, because it is not selected beforehand

- Cipher would not be attacked by frequency analysis attack
- Reduce ambiguity by removing filler character
- Hide plaintext and cipher text relationship

1.6 Organization of the Thesis

The chapters of this thesis are organized as follows: Chapter 2 discussed the literature review and related works of playfair cipher. Chapter 3 introduced the proposed work and Chapter 4 illustrated the detail implementation of the improved algorithm, as well as the simulation results and discussion. Finally, Chapter 5 explained the conclusion and future works of our thesis work.

Chapter Two

2 Literature review

2.1 Overview

This section contains a list of theoretical concepts in the area of cryptography, detail description about playfair cipher algorithm, BBS algorithm and related works in our study area.

2.2 The basic concept of Cryptology

The word cryptology arises from Greek which means hidden and word or reason. It is an umbrella word that is used to express all fields of secret communications. It is an art and science of designing methods to disguise messages [12]. Cryptology is the study of cryptosystems which can be split into two branches namely cryptography and cryptanalysis. Cryptography is the art of designing cipher systems with the goal of hiding the meaning of a message and cryptanalysis is the science of breaking those systems [13].

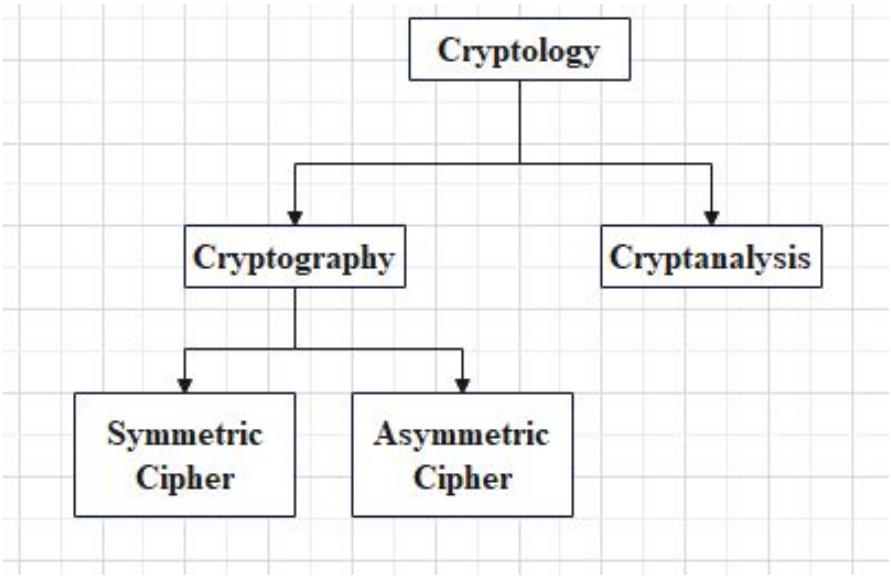


Figure 2.1: Cryptology branches [12]

2.2.1 Cryptography

Cryptography is the process of converting intelligible information into an unintelligible form in order to disguise information from unauthorized parties. This conversion of data is defined by a cryptographic algorithm or procedure with the help of a value called the cryptographic key. An artistic transformation of data into an illegible format to be used and understood by the intended recipient is called cryptography. It is the art and science of hiding valuable and secret information from being infringed upon by unauthorized persons. Hence, generally speaking, protecting and safeguarding all information from cyber criminals or anyone else other than the authorized recipient is referred to as cryptography [14]. Cryptography which is also known as cryptology helps users and institutions to cipher information to transmit safely and decipher it. The message or simply the sequence of letters or symbols which we want to transmit is called clear text or plaintext. The enciphered message or the sequence of letters or symbols that are transmitted is called cipher text [12]. The process of transferring clear text into cipher text under the control of a key is called encipherment or encryption. We can write it as follows:

$$c = e_k(m) \tag{2.1}$$

Where

- m =the plaintext
- e = the cipher function
- k= the secret key
- c = the cipher text

The reverse process of converting cipher text into clear text is called decipherment or decryption and it can be discovered as follows:

$$m = d_k(c) \tag{2.2}$$

So, the sender enciphers the clear text and the receiver deciphers the cipher text in order to get an intelligible message. The sole purpose of cryptography is to protect the

information, email, credit card details, and other personal data transmitted across a public network. Cryptography has two major techniques to convert information into an illegible format such as symmetric cryptography and asymmetric cryptography.

2.2.1.1 Symmetric Cryptography

This type of technique uses the same key for encryption as well as decryption. It is also called a secret key, personal key, or conventional key [14]. Starting from ancient times until 1976, all cryptography was exclusively based on symmetric methods. Most of the time symmetric ciphers are still in use, especially for data encryption and the integrity of messages [13]. In Symmetric encryption or secret key encryption, the two entities should have the key before transmission. In cryptography keys have a great role, if the communicated parties use a weak key in the algorithm then everyone may decrypt the data. The size of the key is the main factor for the Strength of Symmetric key encryption. Encryption using a longer key is harder to break than the one done using a smaller key for the same algorithm [15]. The problem of secrecy in symmetric cryptography is protected by using the shared secret key to transform the message in such a way that no one cannot be recovered anymore without the key.

2.2.1.1.1 Encryption Algorithms

In order to protect the secrecy of messages which are sent over an insecure channel, an encryption algorithm is performed. The general encryption algorithm contains two mathematical concepts such as an encryption function **E** and a decryption function **D=Eˆ-1**. The sender performs encryption to an original message or clear text to communicate securely and transmit the resulting cipher text over an insecure channel.

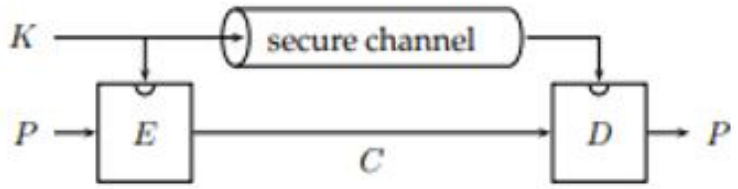


Figure 2.2: Model for symmetric encryption [14]

Symmetric cryptography is further divided into two types namely Stream ciphers and Block ciphers. In the stream cipher, a single digit or a letter of the message is encrypted. So, let's do an example of Vigenere Cipher. Let's a plaintext is "ATTACKATNINE-SHARP". The sender takes a key and repeats it till it matches the length of the plaintext. So, let's assume "SPEED" as a key then, "SPEEDSPEEDSPEEDSP" will be the key for encryption and decryption of the plaintext. Then, we wrote the plain text and the key one beneath the other for making a cipher text. Then after, we got a cipher text called "SIXE-FCPXRLFTWLDJE". In block cipher, a fixed-length group of bits is encrypted at a time. This technique removes the chance of encrypting identical blocks of message [16].

In today's world, symmetric key algorithms are still used by all modern cryptographic systems to encrypt and decrypt data [8]. Symmetric cryptography is less secure for more sensitive data than asymmetric cryptography due its less key size [17]. Some of the well-known symmetric encryption techniques are DES, 3DES, AES, Playfair, Hill, Affine, Vigenere, porta cipher, etc.

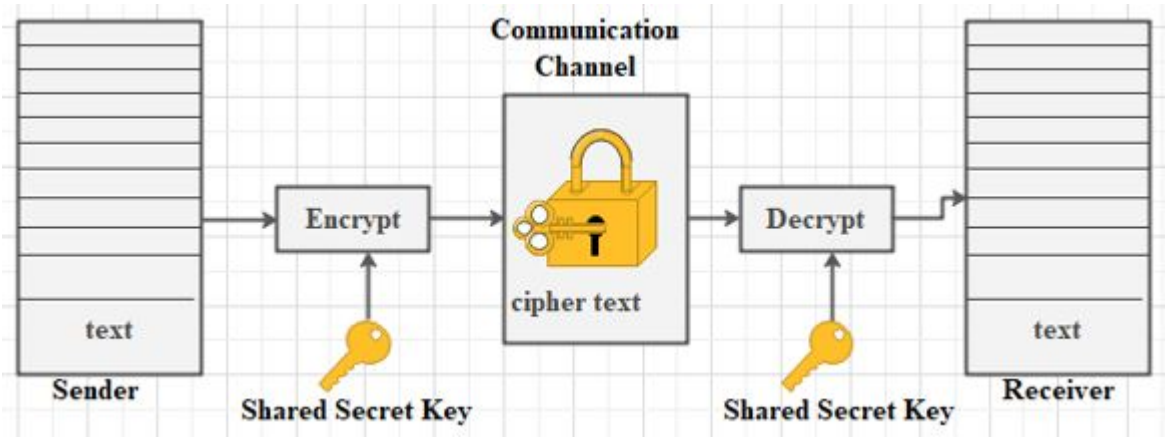


Figure 2.3: Symmetric cryptography [17]

2.2.1.2 Asymmetric Cryptography

In asymmetric or public-key cryptography two keys are used namely a private key and a public key. The message which is encrypted with the public key can be decrypted only with the corresponding private key of the set and vice-versa [18]. The receiver kept secure the private key and the public key is announced to the public [19]. This type of cryptography is used to overcome the problem of key distribution in symmetric cryptography. The public key is used for encryption and the private key is used for decryption [20]. The keys in this technique are different but complementary in function. Because of they require more computational processing power, asymmetric encryption techniques are almost 1000 times slower than Symmetric techniques [15]. Some commonly used public key cryptography techniques are RSA, Diffie-Hellman, DSA, and ECC.

In asymmetric cryptography, encryption/decryption is more complex for a large amount of data due to its computational time is greater than that of symmetric cryptography. Public key cryptography is used only once for key exchange and encryption/decryption is done by using symmetric key cryptography [17]. The computational time of cryptography techniques is also divided into encryption/decryption time, key generation, and key exchange time. The time taken for converting a plaintext or clear text into cipher text and vice versa is called Encryption/decryption time. Key generation time for symmetric and asymmetric cryptography is different and it depends on the size of the key length. The key exchange time depends on the communication channel between the sender and receiver [17].

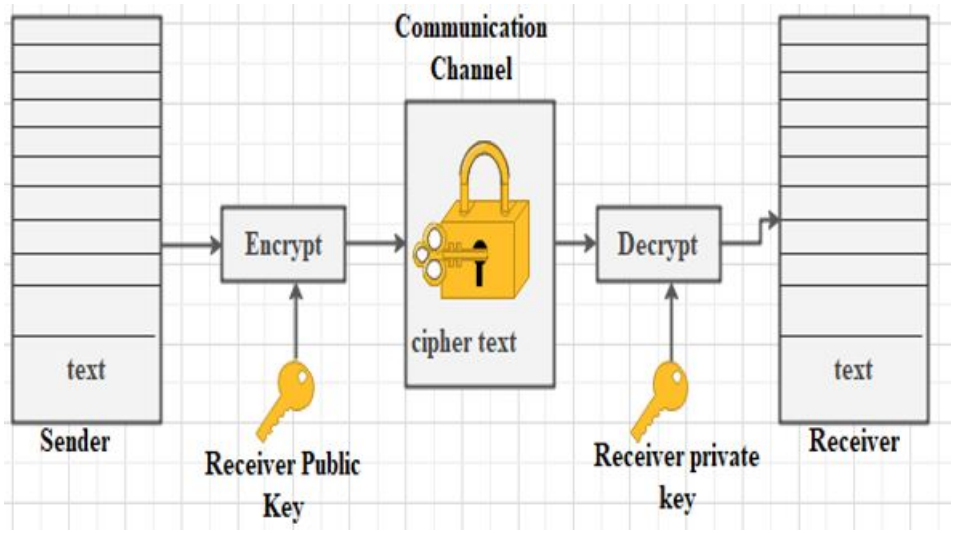


Figure 2.4: Asymmetric cryptography [17]

The below figure shown us the taxonomy of cryptography techniques.

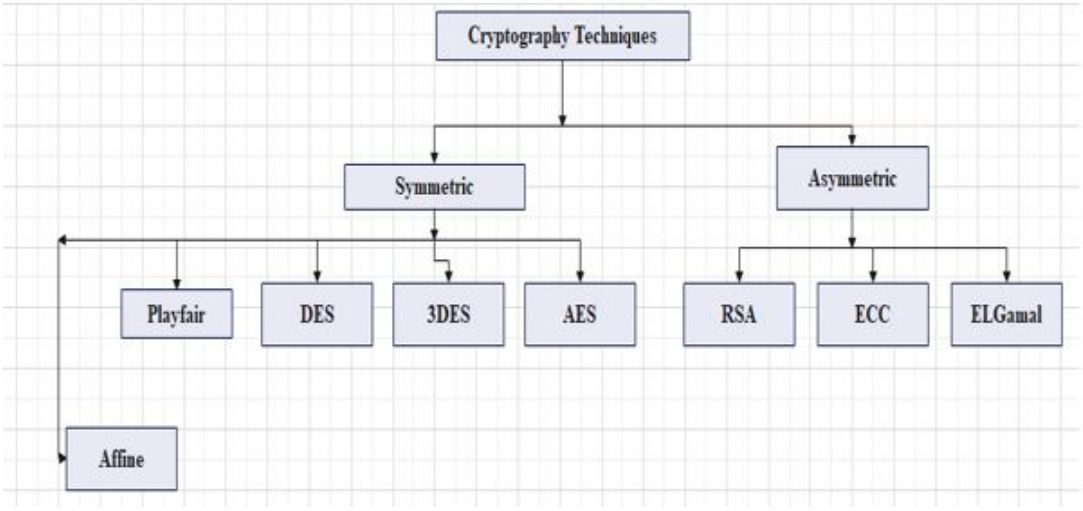


Figure 2.5: Taxonomy of cryptographic technique

2.2.1.3 Goals of Cryptography

Cryptography has four goals such as:

- **Confidentiality:** This is used to protect user identity from being read by others. This goal enables only authenticated users can access the message [20].
- **Integrity:** this goal is used to protect the data from being modified by others.
- **Authentication:** This is used to ensure the data is arise from a specified entity [21]. Before the transmission of data, the sender and receiver must be authenticated.
- **Non-repudiation:** the sender cannot deny that they had sent a message to the receiver. This goal ensures that neither the sender nor the receiver of the message can deny the transmission [22].

The below figure shown those security goals (CIA) of cryptography.

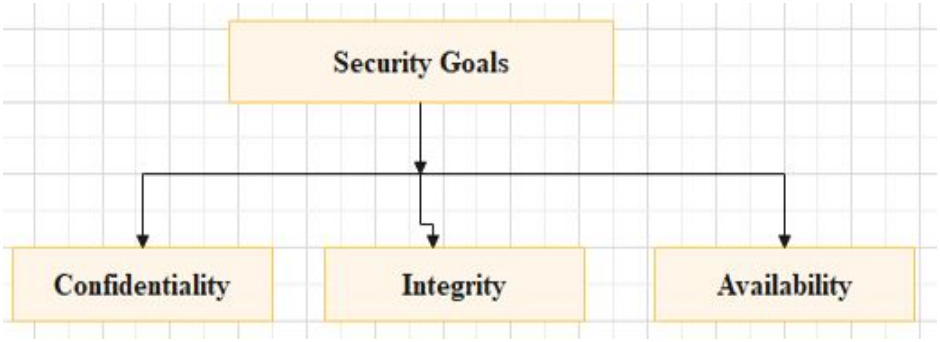


Figure 2.6: Taxonomy of security goals

2.2.1.4 Basic Terms in encryption/decryption technique

It is needed to define some basic terms of cryptography before we moved to Encryption & Decryption process, such as [22]:

- **Plaintext** - is the original text which everyone can read and understand. It is the actual message that has to be sent to other communicated parties end.
- **Encryption** - is the method used to hide a clear text to get an illegible text. In order to send a confidential message through unsecured channel cryptography use an encryption technique. It is performed on the sender side and uses an encryption algorithm and key.
- **Cipher text** - is an encrypted form of plaintext which is not able understood by anyone else. It is a meaningless message form.
- **Decryption** - is the reverse process of encryption which is used to get the original message form from the cipher text. In order to obtain plaintext from non-readable messages, cryptography uses the decryption technique on the receiver side. It uses decryption algorithm and a key.

Figure 2.7 shown the procedure which indicates where the above terminologies are performed on the encryption and decryption process of cryptography.

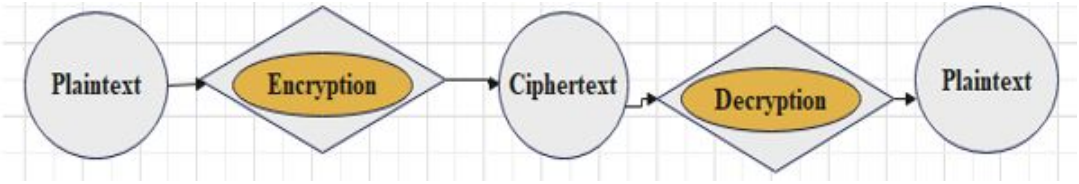


Figure 2.7: Basic terminologies in the encryption/decryption technique

2.2.1.5 Efficacy of Encryption

As we already described earlier, encryption is a way of hiding a clear text message into a cipher text. So, the key area for checking the efficacy of encryption techniques are listed below [23].

- **Large key space**: In order to resist brute force attacks, encryption needs large key space.
- **Sensitive dependence on initial conditions**: The result for Small changes in the initial values of keys should be a high degree of enciphered output.
- **Less Time Complexity/High Speed**: very less encryption time should be needed for the scheme.
- **Security**: the effect of statistical, differential, brute force, and several other attacks should be minimum.
- **Diffusion** :There should be a drastic change in enciphered output for a small change in input data.
- **Confusion** : there should be no or minimum relation between key and cipher text.

2.2.2 Cryptanalysis

The technique of deriving the original message from the unreadable form of text without any prior knowledge of secret key is known as cryptanalysis [24]. Cryptanalyst tries a lot of methods to obtain maximum information about the clear text or original data [25].

Cryptanalysis is needed not to break others people code but to learn how vulnerable our cryptosystem is. In order to create better secret codes cryptanalysis is needed.

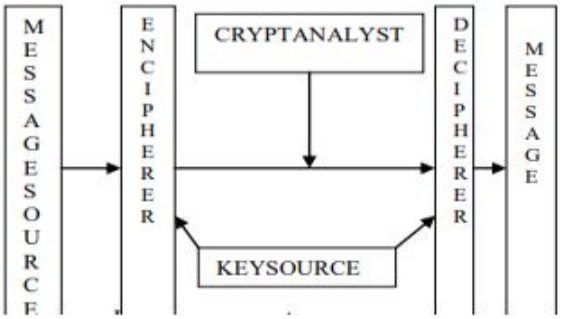


Figure 2.8: Cryptanalysis

Attacker aims to access encryption key instead of simply decrypt the data. Based on the basis of available information, attacker can be classified in to four types [25]. The following figure shown the four types of cryptanalysis methods.

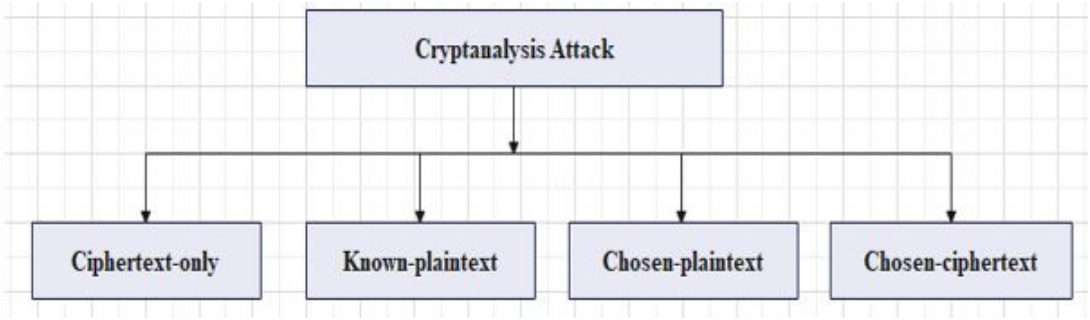


Figure 2.9: Types of Cryptanalysis attacks

2.2.2.1 Cipher text only Attack

In this type of attack, an attacker can access only the cipher text of the encrypted message. Attacker tries to find the equivalent key and plaintext from the cipher text using some mechanisms. It is the most probable attack because attacker has only access to cipher text [26]. In the below diagram Eve who is an attacker got cipher text while Alice and Bob shared the data and eve tried to analyse the equivalent plaintext. As figure 2.10 shown, eve has access to the cipher text which is sent from Alice to Bob. Various types of cipher text only attack are there. Among them, the two are described below:

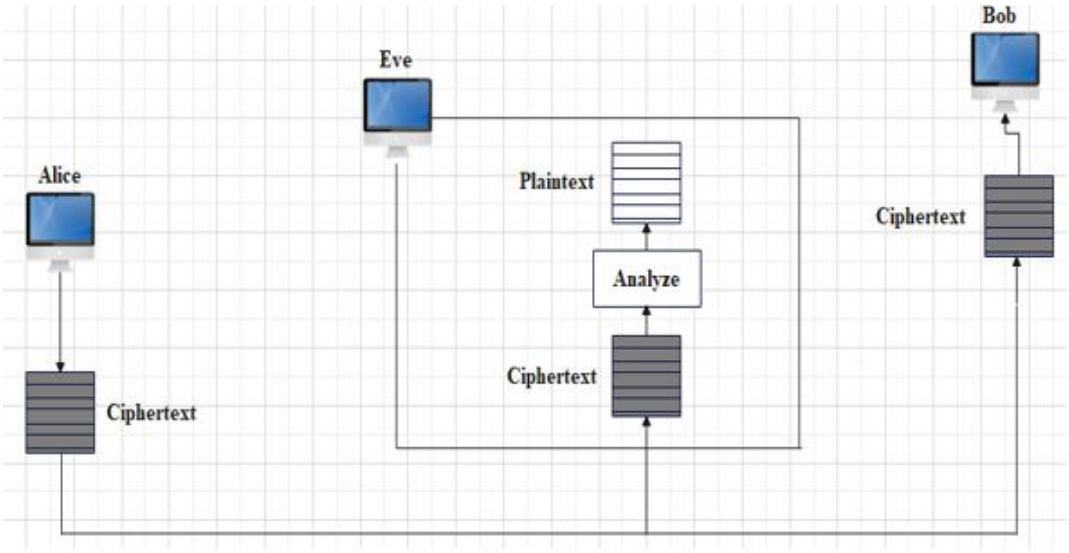


Figure 2.10: Cipher text only attack [26]

Brute Force Attack

In this types of attacks, unauthorized user tries to use all possible keys in order to decrypt the encrypted text. Here, attacker knows the algorithm and the list of all possible keys (Key domain). Attacker tries all key domains until the plaintext give sense. In the past time, using this method was so difficult than today because attackers were not used computer as an attacking tool in the past time rather it was done manually. In this time, attacker use computer as their attacking tool. The feasibility of brute force attack depends on the length of the key used in encryption process [25].

Statistical Attack

In this types of attack, an attacker finds the most frequently used characters in the cipher text and guess the corresponding character based on the frequency analysis of English characters. Based on this fact, the attacker finds the equivalent key which was used to encrypt the cipher text. In the cipher text, the statistical properties of the plaintext are preserved cause of each plain text symbol always map to the same cipher text symbol [13]. The most frequent letter in English letter is E (about 13%), Second highest is T (about 9%) and the third one is A (about 8%), and so on. In addition to characters frequency, there are also bigram characters which appear most frequently in English language. The most common bigrams are AN, AT, ED, EN, ER, ES, HE, IN, ON, OR, RE, ST, TE, TH, and TI.

The following table 2.1 contains english characters with thier probability of frequences

in english language. So, from the table, we saw that character E is the most frequented characters.

Table 2.1: English letter frequency [26]

Letter	Frequency	Letter	Frequency
A	0.0817	N	0.0675
B	0.0150	O	0.0751
C	0.0278	P	0.0193
D	0.0425	Q	0.0010
E	0.1270	R	0.0599
F	0.0223	S	0.0633
G	0.0202	T	0.0906
H	0.0609	U	0.0276
I	0.0697	V	0.0098
J	0.0015	W	0.0236
K	0.0077	X	0.0015
L	0.0403	Y	0.0197
M	0.0241	Z	0.0007

2.2.2.2 Known plain text attack

In this types of attack, an attackers can access some plaintext and their corresponding cipher text. Then, attacker finds out the relation between two to find out the encryption key [25].

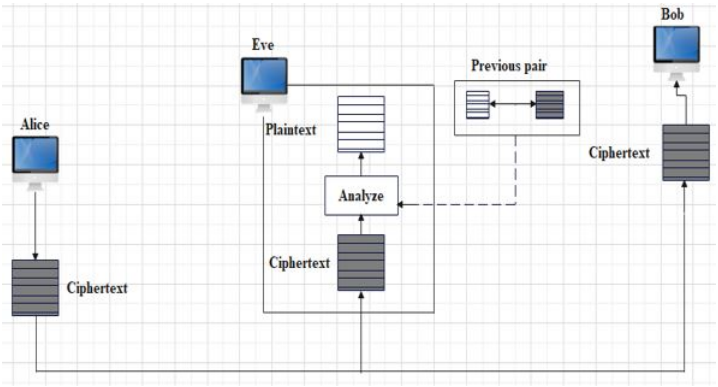


Figure 2.11: Known plaintext attack [26]

2.2.2.3 Chosen Plaintext Attack

In this type, an attacker first choses the plaint text / cipher text pairs and then tries to analyse plaintext from the transmitting cipher text. This type of attack is much less likely to happen because attackers can do this if they enable to access the sender device.

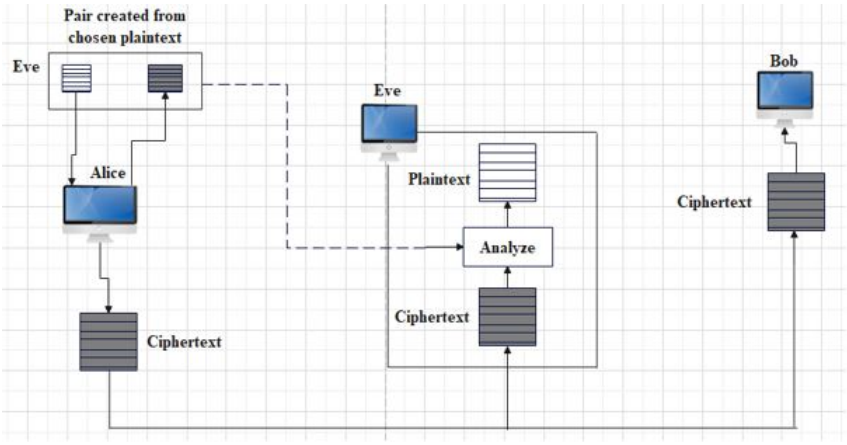


Figure 2.12: Chosen plaintext Attack [26]

2.2.2.4 Chosen Cipher Text Attack

The attacker first obtain an arbitrary set of cipher text and finds the various corresponding plaintext. This attack is happened if attacker has access to user computer.

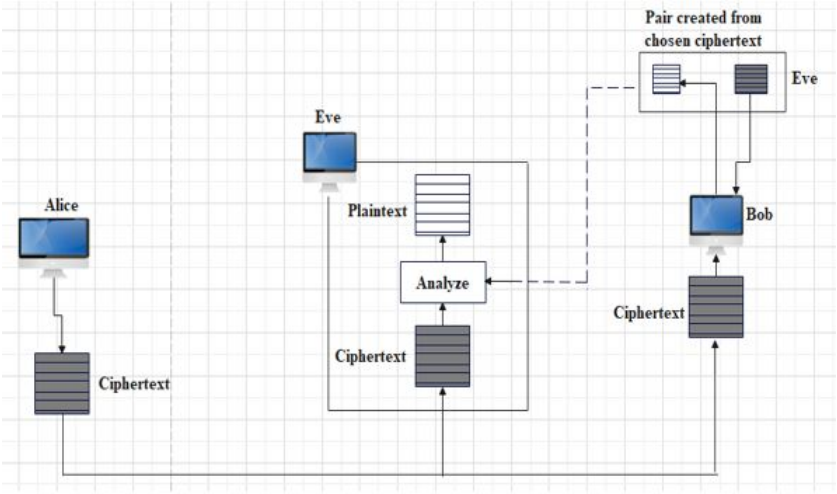


Figure 2.13: Chosen Cipher text Attack [26]

2.3 Traditional Playfair cipher

Playfair cipher is a symmetric encryption technique. It is a polyalphabetic cipher and a digraph cipher, which encrypts plain text or original data to cipher text or scrambled form of data with two characters at a time. The key of this playfair cipher is made from the 25 English alphabet letters arranged in 5*5 matrixes. It is possible to create many different secret keys by arranging letters of the matrixes in different ways. The characters of the key are entered at the start of the playfair matrix from top to bottom and written without repetition. The remaining cells of the matrixes are filled with another letter of the alphabets following its assigned value increasingly, while I and J are placed in the same cell and used interchangeably [8].

The creation of the Playfair cipher shows a great advancement over the monoalphabetic ciphers since digraphs identification is more difficult than individual letters identification. The technique encrypts pairs of letters (diagraphs) instead of single letters which are seen in substitution ciphers. Thus, makes the playfair cipher harder to break than the monoalphabetic substitution cipher since 676 digraphs (26 X 26 has to be unique) are possible whereas in the case of a simple substitution cipher technique, only 26 monographs are possible. The Playfair Cipher was considered unbreakable for a long time [7].

2.3.1 Preparing the Plaintext

In the first step, the plaintext is to be prepared and pre-processed. In order to do this, all plaintext letters should be written in uppercase letters and all spaces and punctuation characters present in the plaintext should be removed. Then, the plaintext is grouped in pairs or digraphs and all **Js** character are replaced by character **Is**. Besides, if double letters occur in a pair, it must be divided by an extra bogus letter **X** or **Z**. Most of the time, characters which are used as a bogus letter is the least frequented characters in English language as shown in table 2.1. This splitting of double letter with bogus character helps to reduce the number of visible patterns in the cipher text. Finally, after inserting the bogus letter, if the number of character in the plaintext is odd, to make the number of character even one extra bogus letter is added at the end. Once we have completed above steps, it is possible to do encryption process.

2.3.2 Preparing the Key

The 5 * 5 encryption key matrix is prepared by filling the letters of the key word (without repetition) from left to right and top to bottom in the first row the matrix. The remaining letters are filled the rest of the matrix in alphabetical order. In this case, the letters **I** and **J** are considered as one letter. If the key word we used is too long, the more secure the cipher will be generated.

2.3.3 Encryption

The possible combination of the digraph characters in the Playfair cipher algorithm are in the same row, in the same column, or in different rows and columns of the matrix. Existing playfair cipher technique encrypts the plaintext containing alphabets only but Numerical values and symbols cannot be encrypted by this technique. For the last digraph characters possible combination, we used the third rule from the following listed rules and in order to form rectangle we should start with the first letter, and move across until it is lined up with the second letter; then start with the second letter and move up or down until it is lined up with the first letter. In order to made encryption using playfair algorithm, we have to use the following three rules [1]:

- If the pairs of character are located in the same row of the key matrix, the equivalent encrypted character for each letter will be the next letter to the right in the same row (going back to the leftmost if at the right most position).
- If the pairs of character are located in the same column of the key matrix, the equivalent encrypted character for each letter will be the letter beneath it in the same column (with wrapping to the beginning of the column if the plaintext letter is the last character in the column).
- If the pairs of character are not in the same row or column of the key matrix, form a rectangle with two letters and take the letters on the horizontal opposite corner of the rectangle.

For example, the matrix formation for the key word "**PUZZLE**" would be represents as follows in the table 2.2.

Table 2.2: Matrix formation of playfair using PUZZLE as a key

P	U	Z	L	E
A	B	C	D	F
G	H	I	K	M
N	O	Q	R	S
T	V	W	X	Y

Then, using the above three rules and the constructed matrix, it is possible to encrypt the plaintext "**MEET ME AT THE MALL**". First, the plaintext should get converted to digraphs as "**ME ET ME AT TH EM AL L**". Here, the last character **L** could not became digraph because the number of the plaintext character is odd. i.e., which is fifteen. So, it is needed to add one extra bogus letter **X** to make the total length of the plaintext even and to make digraph with character **L**. once we finished, it becomes "**ME ET ME AT TH EM AL LX**". Finally, the final cipher would be "**SF PY SF GP VG FS DP DL**".

2.3.4 Decryption

Decryption is the reverse process of encryption in cryptographic algorithm. The receiver has created the key matrix with same key of the sender and made decryption by the key matrix. It follows the following rules to retain the original plaintext:

- If the letters in the digraphs are at the corners of a rectangle, then the cipher text has the letters at the opposite corners of the rectangle.
- Otherwise, if the digraphs letter are in the same row, translate each of them as the next letter on the left. If you fall off the beginning of the row, wrap around to the end.
- Otherwise, if the letters are in the same column, translate each of them as the letter above. If you climb off the top of the column, wrap around to the Bottom.

2.4 Cryptography and Random Number

Randomness is fundamental to the field of cryptography because secrets are the main thing in the field. Truly secret data should be random to the attacker. In cryptography randomness is like the air we breathe, without it , we can do nothing [27]. A sequence of integers that never show relations to each other is referred to as random number. Based on the way of generating a random numbers, there are two methods namely true random number generator and pseudo random number generator. Pseudo random number generators (PRNGs) generate an output sequence number by taking a seed as input [28]. It is widely used since it is convenient and fast. If someone can't predict the future value of a random number by observing the generated random sequence we can say that the random number is secure [29]. It is possible to generate a pseudo random numbers using algorithms, for instance, BBS, LFSR, LCG etc.

2.4.1 Blum Blum Shub Algorithm (BBS)

The Blum Blum Shub is a pseudorandom number generator created in 1986 by Lenore Blum, Manuel Blum and Michael Shub. The security of the BBS generator depends on the difficulty of factoring M [30]. In order to generate random number it uses the following formula:

$$X_{n+1} = X^2{}_n (mod M) \tag{2.3}$$

Where:

n=1,2,3,4,5,6,7,...,

X˙0= a seed value,

M= P*Q and P and Q are prime number.

The value of P and Q are both congruent to 3 *mod* 4 and **GCD** (X0, M) =1.

X˙n+1= The generated random sequences.

2.5 Extended Playfair Algorithm

This algorithm uses the Fibonacci series to generate keys for the encryption and decryption process. We have reviewed the paper in [10] and explained the strengths and limitations of this algorithm. The below figure shown the flow diagram for the key generation and encryption process of this extended algorithm. It has also made a modification to the encryption mechanism of the original playfair algorithm.

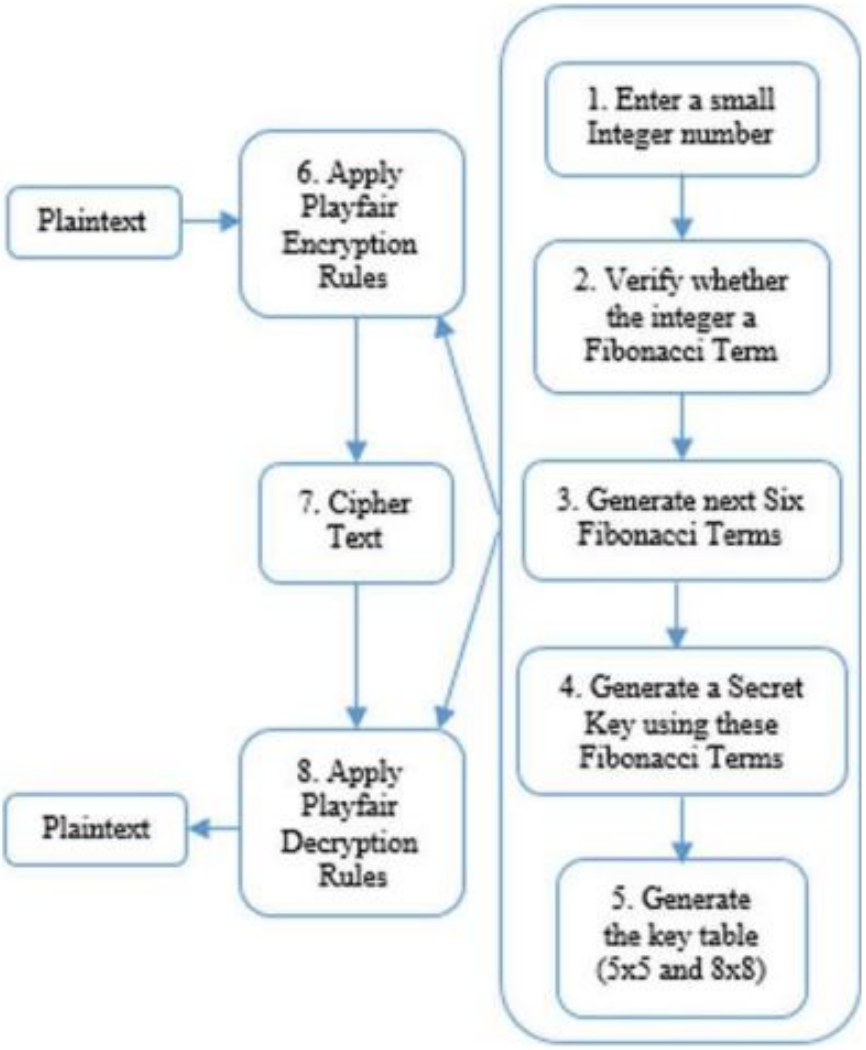


Figure 2.14: Extended Playfair Algorith Flow Diagram

2.6 Related Work

With the advancement of technologies there is a need to make information more secure. Cryptography is one of the best methods to keep the information secure from unauthorized body. Cryptography uses encryption technique and keys to secure data. Playfair is one of the encryption techniques used for securing the data safe but it has its weaknesses. So, in order to provide more robust playfair encryption algorithm several studies has been conducted. Among them some are listed below.

In [31], the author proposed a new 3D generalized playfair model which takes trigraph of text message at the same time and generates the equivalent cipher text accordingly. Here the author designs his algorithm for encryption and decryption techniques. In this research the plaintext is divided into group of three letters also called trigraph and if any of the trigraph have repeated letters, it is again separated by filler letter such as x and y. It has better security than digraph 5x5 playfair cipher because it makes alphabetic, symbolic and hybrid combination of cipher text but this algorithms provide same cipher text for same characters and low avalanche effect with one bit change and also it needs filler character when repeated character pair happens.

Muhammad Syahriza et al [32] developed a modified playfair cipher with random key of linear congruent method and changed the key matrix length consists of only 25 capital characters by counting letter I and j as one to 255 ASCII characters. This length extending of key matrix increases the security. Here the authors generate the key randomly rather than sharing of key but it is still vulnerable to attacker because it uses printable ASCII character values after finding the random key value and same digram also generates the same cipher.

The authors of [33] proposed a novel technique for enhancement of the security of playfair cipher by using vigenere cipher, playfair cipher and linear congruential methods. Here the author first encrypts the plaintext using the vigenere cipher by using first key and again encrypt the vigenere cipher result using playfair cipher by the second key. In this study the existing 5x5 matrix is modified by 6x6 matrix .The matrix and the vigenere cipher table incorporate numbers from 0 - 9 in addition to the capital English characters. This proposed technique uses linear congruential method to generate a 6x6 matrix sequence of random numbers in order to map to the cipher of the playfair cipher. This paper was secure than

the existing cipher because it uses hybrid algorithms and transmit mapped sequence number than characters but it has still limitation such as it does not include special characters and generation of random number using linear congruential method depend on the value the multiplier and increment, which needs further investigation to enhance the security. If the increment and multiplier are to low and the same, the generated sequence became not satisfactory.

Maherin Mizan Maha et al [8] was proposed an effective modification of playfair cipher with performance analysis using 6x6 matrixes. In this paper, it is possible to encrypt alphabets and numbers. The authors create a method by counting the key and find the character equivalent of the total number of keys and take the character row and column value then multiply it and get the diagonal character to the multiplication equivalent character to form digraph for the encryption of space using key matrix and increase the possible key matrix formation by extending the matrix size. The proposed paper produces strong cipher but it does not incorporate small letter and special characters and also the digraph and its reverse give the same cipher.

In 2019, authors in [34] have proposed an algorithm which enhances the key security of playfair cipher entitled with An Enhanced Key Security of Playfair Cipher Algorithm. Here, the author encrypts and sends the key to the receiver to protect the key from unauthorized attack but the key is still vulnerable to man in the middle attack because it uses ASCII values. This ASCII value is printable and everyone is aware of the characters equivalent in ASCII value.

In [35], an author was proposed algorithm to enhance security of playfair cipher. In this paper the author uses a total of 36 numbers which are uppercase letters and numbers to generate the key matrix. In this paper similar digram producing similar cipher problem of the existing playfair problem is solved. Still the paper has limitations on hiding the relationship between the cipher and the plaintext for instance in the worked example the last plaintext Y and Z produces cipher ZY. Another limitation is it did not incorporate special characters so it is not possible to encrypt plaintext which is a combination of characters and special characters.

The study of [36] proposed an algorithm to enhance the security of playfair cipher by expanding the key matrix from 5x5 to 5x19. Here, the proposed algorithm incorporates 95

characters as a content of the matrix. The authors solved ambiguity which was created on the receiver side due to the removal of character J. The authors also solved the problem of equal length of plaintext and cipher text by increasing the length of cipher text. This proposed algorithm has still limitations like similar character pair provide the same cipher text and its reverse provides the same cipher pattern. This limitation makes the algorithm to be vulnerable by attackers.

An author in [37] was proposed an algorithm which enhances the security of play fair cipher by expanding the matrix size and by modifying the matrix formation pattern. This algorithm was named an efficient modification to play fair cipher which includes 26 English alphabets, 10 numeric digits and 10 most frequently used punctuation marks. Among this 49 total elements two of them are used as a padding and filler letter. In this paper the author overcomes the problem of I/J inconsistency and padding character problem. Here, the author algorithm changes the matrix generation pattern but still it has limitation like the digraph and its reverse show similar cipher pattern and cipher text show relationship to its plaintext when the digraphs character are paired from end and beginning part of the matrix.

In [38], an author proposed an extended play fair cipher which enhances its security by expanding the matrix size to 10 x10. Here, the author uses all digits, all English lower case and upper case letters and 38 special characters including spaces as their proponents to generate the matrix. In this paper the filler character used for splitting similar character digram is special symbol instead of X. It also use special character for padding character when the length of character becomes odd. This paper reduces ambiguity of padding and filler letter and enhance security by incorporating a lot of character as their matrix proponents. This paper has low avalanche effect problem. When a single character is changed it shows low avalanche effect percentage.

In paper [39], the author proposed an improved 3D (i3D) playfair algorithm which overcomes the drawbacks of the paper 3D playfair algorithm. This paper achieves a high avalanche effect (51.11%) when a single key bit changes. Here, the paper implements CBC method and dual circular XOR method to make the paper execute better performance than other algorithms. Furthermore, the key space is also increased from 3D of 64 character keyspace to i3D of 256 characters. The paper incorporates additional character to make

the result ciphertext more secure but still it has limitations of key exchange and needs filler character while the same character paired in a triplet. This key exchange limitations make the algorithm vulnerable for cryptanalaysis like brute force attack and needs of filler character limitations create ambiguity at the receiver side.

A Modified Playfair CBC Algorithm [40] presents the digraph CBC encryption mechanism to make the cryptanalysis more secure. In this paper, the key matrix is reorganized after performing one digraph CBC encryption and this process continues until the number of digraphs are completed. This algorithm is examined using security parameters like ciphertext only attack, brute force attack and avalanche effect and produce a better performance than the original playfair cipher. Even though it provides better performance it has still limitations of requiring filler letter when same characters are paired and also it shows a correlation between cipher text and plaintext when the encrypted digraph characters are paired from the corner or beginning and end part of the key matrix. It also exchanges the key directly with the receiver and this makes the key vulnerable for attacker.

Arnold C.Licayan et al [41] designed seed based color substitution of playfair cipher which enhances the limitation of directing the text to equivalent color values. This algorithm uses original playfair algorithm, random numbers and compression techniques together to improve security and space at the same time. It evaluates the algorithm using avalanche effect and got 31% more security and saved space by 52% using Huffman compression methods. It uses middle square method to generate random numbers which are nondeterministic and show no relation with the initial values. Here, the algorithms exchange the key used for encryption and decryption process and removes redundant characters of the key instead of replacing by another characters which are not part of the key. This leads the key length to become less in length and easy to guess by attackers. It has also needs additional filler character when a bigrams are paired with the same character and this creates during decryption at the receiver side. Besides, the algorithm also shows the plaintext and cipher text relation when the bigrams are paired from the beginning and ending part of the matrix cell.

Authors in [42] designed an algorithm by the combination of Playfair cipher and RSA technique to exchange the keys between sender and receiver in a much secured manner. Here, the key square or the matrix is expanded to 16 * 16 from the existing playfair key

square of 5*5. In this algorithm, the RSA is used to encrypt the key of the playfair cipher used for encryption and decryption to protect it from being attacked by unauthorized party. This paper has still limitations of requiring filler character when the same characters are paired and it does not hide the relationship between plaintext and ciphertext when the bigrams are paired from the beginning and end part of the key square. The future scope of this method aims to decrease the decryption time of the RSA algorithm.

2.6.1 Summary of Related Works

The following table presents summary of some of the above related works.

Table 2.3: Summary of related works

No	Year	Authors	Publisher	Title	Objective	Conclusion	Gabs
1	2014	Md Ahnaf and Md Rabiul	ULAB Journal of Science and Engineering	An Efficient Modification to Playfair Cipher	To increase security by expanding matrix size and change the encryption rule of playfair	The author overcomes limitations of I/J inconsistency and increase the matrix size element from 5*5 to 7*7 matrix size. It changes the rule of encryption and decryption.	• It still uses filler character • Even if the encryption rule changed, it shows plaintext and cipher text character relationship. • Key exchange problem
2	2016	Priyanka Goyal and et al.l	IJERT	Implementation of Modified Playfair CBC Algorithm	To implement CBC with playfair algorithm to make robust algorithm	CBC is implemented on the digraphs and the key matrix is reorganized after every CBC operation is performed. It produces better performance than original playfair cipher.	• Requires filler character when same characters are paired in the digraph • The key is exchanged directly • It shows correlation between digraph plaintext character and its equivalent ciphers.

3	2017	Muhammad S. and et al.	IRSTC	Modified Playfair Cipher Using Random Key Linear Congruent Method	To improve security by increasing the number of playfair matrix elements from 25 to 255 ASCII values.	It generates random key and change it its equivalent ASCII value. Further, it is possible to encrypt alphanumeric characters.	• It's easily attacked since ASCII values are printable and can be easily known. • Digraph and its reverse provide same cipher pattern. • It also need filler character. • Plaintext and ciphertext correlations are shown.
4	2018	E.Elahi and et al	International Conference on Emerging technologies	A New 3D playfair based secure cipher generation model	To develop a new 3D algorithm and enhance security of playfair cipher	It encrypts a trigraph than digraph. It has developed the encryption and decryption rule for a trigraph. It provide better security than 5*5 playfair algorithm.	• It has low avalanche effect result. • It provides the same cipher text for same characters • It needs filler characters
5	2018	Justine Ceasar and et al.	IEEE	Extended 10x10 playfair cipher	To increase security by extending the matrix size in to 10x10 matrix.	It includes alphabets, numbers and special symbols since it increases the matrix size. It uses special symbols for filler and pad characters. It provides better security because the cipher contains alphabets, numbers and special symbols.	• Low avalanche effect • Even if it uses special symbols for filler character, requiring filler character by itself is the limitations. • Plaintext and ciphertext correlations are shown. • It exchanges the key directly.

6	2019	Pal Pathikrit and et al.	IEEE	A variation in the working of playfair cipher	To solve similar digram provide same cipher and increase security using 36 elements as the proponent of the matrix.	It uses 36 characters as the matrix size and solve similar digraph provide similar cipher.	• It shows plaintext and ciphertext correlation. • Filler characters are required when the same characters are paired. • It does not incorporate special symbols as the matrix element. So, it is not possible to encrypt special symbols. • Plaintext and ciphertext correlations are shown.
7	2019	Anshari and et al	IEEE	Expending Technique Cryptography for Plaintext Messages by Modifying Playfair Cipher Algorithm with Matrix 5 x 19l	To improve security of playfair cipher by increasing the matrix size in to 5x19	It provides better security by overcoming the problem of ambiguity created due to removal of character J and solves the problem of equal length of plaintext and ciphertext by increasing the length of the cipher.	• The plaintext digraph and its reverse digraph provide the same cipher pattern. • Key exchange problem is also there. • Filler character is also required here during the same characters are paired in the digraph.
8	2019	Ritchell S. and et al.	IEEE	i3D-Playfair: An Improved 3D Playfair Cipher Algorithm	To improve security by overcoming the limitations of new 3D algorithm	This work provide better avalanche effect and implement CBC and dual XOR method. It increases the key space from 64 of 3D to 256 of improved 3D algorithm.	• It has limitations of key exchange problem. • It needs a filler character when two of the trigraph characters are the same.

9	2020	Arnold C. and et al.	IEEE	Performance Analaysis of Playfair Cipher Color Substitution Variants	To improve security and space at the same time using original playfair cipher, random number and compression techniques	This paper produce better security performance when it is evaluated using avalanche effect and saved a lot of space using compression techniques.	• If the key characters are repeated, it removes the characters instead of replacing by another characters. This leads to the key length to become short. • It requires filler character.
10	2020	Mohd V. and et al	Springer	An Extended Playfair Encryption Technique Based on Fibonacci Seriesl	To improve security by extending the matrix in to 8x8 and using Fibonacci series	It increases the security a little bit than original playfair cipher.	• We deeply expressed in the statement of the problem since this paper is our bench mark.

From the literature, we grasped the method approaches and trends to improve the security of playfair cipher, and then the gabs to be addressed in the future. We have identified a lot of gabs that many authors did not address. Besides this, the literature also helped us to know the current state of the playfair cipher algorithm security, and it helped us to consider other methods to improve the security and to develop a robust algorithm for playfair cipher.

We have identified a lot of gaps in the literature, but among the gaps, we focused on the security improvement like key exchange, and plaintext and cipher text relationship. The other gaps we focused is filler character requirement during the same characters are paired in bigram formation.

Chapter Three

3 PROPOSED ALGORITHM

3.1 Overview of the proposed work

In this paper, we have proposed an encryption and decryption algorithm for the playfair cipher algorithm using modified BBS and keystream values. The main motif of this proposed work is to enhance the security of the cipher which was created by using an extended playfair cipher using Fibonacci series and to make it less vulnerable to cryptanalysis attacks. This chapter exhibits the details of the proposed security enhancement of the playfair cipher algorithms. Here, we discussed how we modify BBS using four Blum prime numbers, how we investigated the keystream of each character, and the proposed algorithm's basic functions like key generation, encryption, and decryption process.

3.2 Modified BBS Algorithm

As we discussed in chapter two, BBS is a pseudorandom number generator using two Blum prime numbers and seed values. The security of classical PRNG sequences using BBS depends on the difficulty of the integer factorization [43]. Once the attacker factorizes the module's value, he gets the value of the two Blum prime numbers P and Q and he enables to find the generated PRNG sequence. Here, we have modified BBS using four Blum prime numbers to make the integer factorization more complex. These modifications are performed using four Blum prime numbers instead of using two Blum prime numbers. In this study, we have derived a formula for merging two Blum prime numbers by fulfilling the condition of the merging number must be a Blum number. So, the four Blum prime number becomes two Blum prime numbers using the derived formula. For instance, we have four Blum prime numbers A, B, C, and D. So, using the derived formula we can merge into two Blum numbers as follows:

$$DerivedFormula = (B_i * B_{i+1}) + 2 \tag{3.1}$$

Where,

- B˙i is the four entered Blum prime number namely A,B,C and D

Thus, the resulting Blum prime number will be:

$$BL1 = (A * B) + 2 \quad and \quad BL2 = (C * D) + 2 \tag{3.2}$$

Where

- BL1 is the first merged Blum prime number and
- BL2 is the second merged Blum prime number.

In this paper, we need to share only the four Blum prime numbers with the receiver by keeping safe the module's value and the receiver merges it into two Blum prime numbers as we showed above, and then it generates a PRNG sequences which are used as a key for encryption and decryption purpose. Here are the steps of the modified BBS algorithm:

1. Insert four Blum prime numbers namely B1, B2, B3, and B4
2. Generate two Blum prime numbers from the four Blum numbers using equation (3.2)
3. Discover the value of N or modules value using BP1 and BP2

$$N = (BP1 * BP2) \tag{3.3}$$

4. Select the seed value (Se) between 0< Se < N and it must be relatively prime with the value of N. It also must be lessthan that of the total number of elements of a 14 x 14 matrix.
   i.e., GCD(Se,N)=1 and compute

$$R_0 = Se^2 mod N \tag{3.4}$$

5. For **i** from **1** to **n** do the following

$$R_i = R^2{}_{i-1} mod N \tag{3.5}$$

   Here,n is the number of sequences we need to generate.

6. END

7. The sequence becomes R1, R2,... Rn.

The modified BBS pseudorandom number generator flowchart can be seen in the following figure.

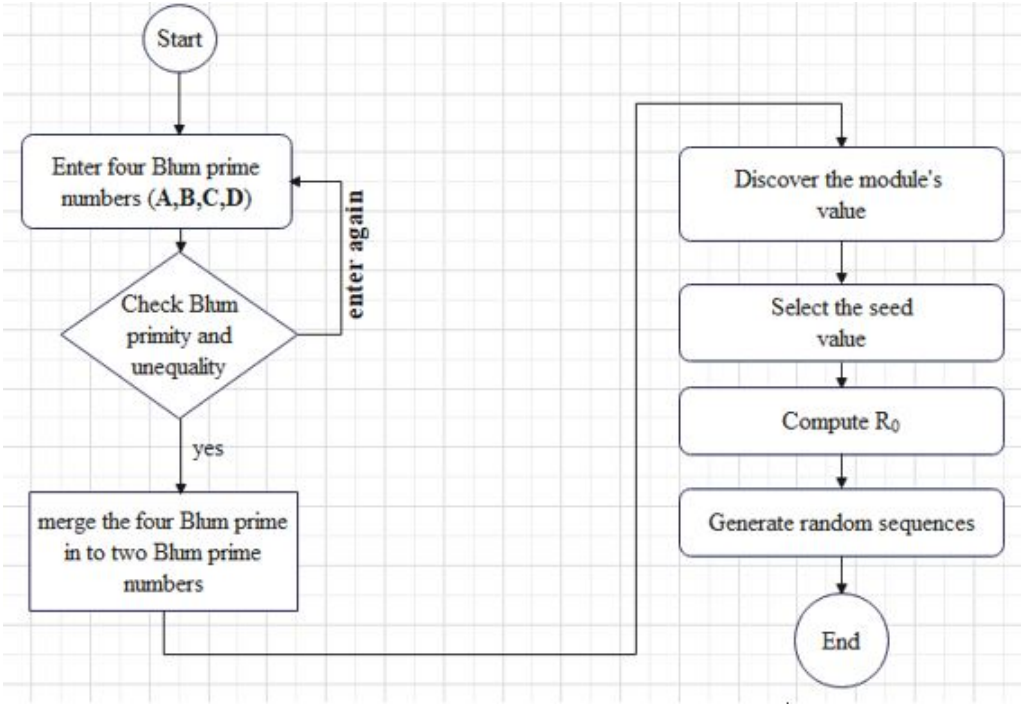


Figure 3.1: flowchart diagram of modified BBS

For instance let's take the value of the four Blum prime numbers A, B, C, and D as 3, 7, 11, and 19 respectively to generate random sequence numbers. The seed value (Se) we selected is 25. Using this given values the generated random sequences became 37 33 117 108 126 37 12 127 3 64 64 54 96 93 and 41.

3.3 Keystream Values

In cryptography, the term keystream is a sequence or pseudorandom sequences that are combined with the plaintext message to create an unintelligible form of the original message [44]. We said that a keystream is a pseudorandom sequence because the seed or an initial seed value is required to generate sequences. Here, in this paper, we used a linear equation

$$keystream = ax + c \tag{3.6}$$

to investigate the keystream value of every character of the plaintext. We will use the module's operation if the investigated keystream value of every character is greater than the total number of characters of the matrix. In this equation, the character c indicates the positional value of the plaintext characters and a is the sum of the seed value and the every character positional value (c), X is the sum of Med, a, and c. Med is the median value of the sequences which is used to encrypt the four Blum prime number. The seed value we use in modified BBS to generate random sequences will be supplied to the linear equation keystream=ax+c and it helps to generate the keystream sequence which is equal in length to the length of the plaintext character. In our algorithm, we have assigned values for every character which is incorporated under the matrix elements rather than using ASCII values because the ASCII value is easily known by many persons.

So, we use the index value we assigned for every character instead of using ASCII values. For instance, the upper case letter 'A' has an ASCII value 65, this is known and if the attacker gets the character A, then he tries to use 65 because the ASCII value is known throughout the world. In our case, the upper case letter A has assigned with an index value of 1, and this is not known by others. We have assigned an index value for uppercase letters from 1-26, digits from 26-36, lowercase letters from 37-62, and special symbols from 63-196. So, the character index value we use in our algorithm is not the same as that of the ASCII value. The following is the pseudo code for keystream generation of each plaintext character:

1. Enter the seed value (Se) and calculate the length of plaintext (Lp)
2. Investigate the median (Med) of the random number sequences and identify positional value (c) of every plaintext character
3. Calculate the value of **a** using seed value and positional value

$$i.e., a_i = Se + c_i \tag{3.7}$$

4. Find the value of X˙i

$$i.e., X_i = Med + a_i + c_i \tag{3.8}$$

5. For **I** from 1 to **Lp** do the following

$$Keystream_i = a_i x_i + c_i \tag{3.9}$$

where

- Keystream˙i is the discovered keystream value of each character
- **a** is the sum of seed value and every character position
- **X˙i** is the sum of Med,a,and c
- **c˙i** is the character position value in the message
  - if the discovered keystream is greater than 196

$$i.e., Keystream_i = mod(Keystream_i, 196) \tag{3.10}$$

6. End

7. The sequence become keystream˙i , keystream˙i+1 , . . . ,keystream˙n where **i** starts from 1,2,3. . . .and **n** is the length which is equal to the length of the plaintext.

The following figure illustrates the overall process of finding keystream value of every plaintext characters.

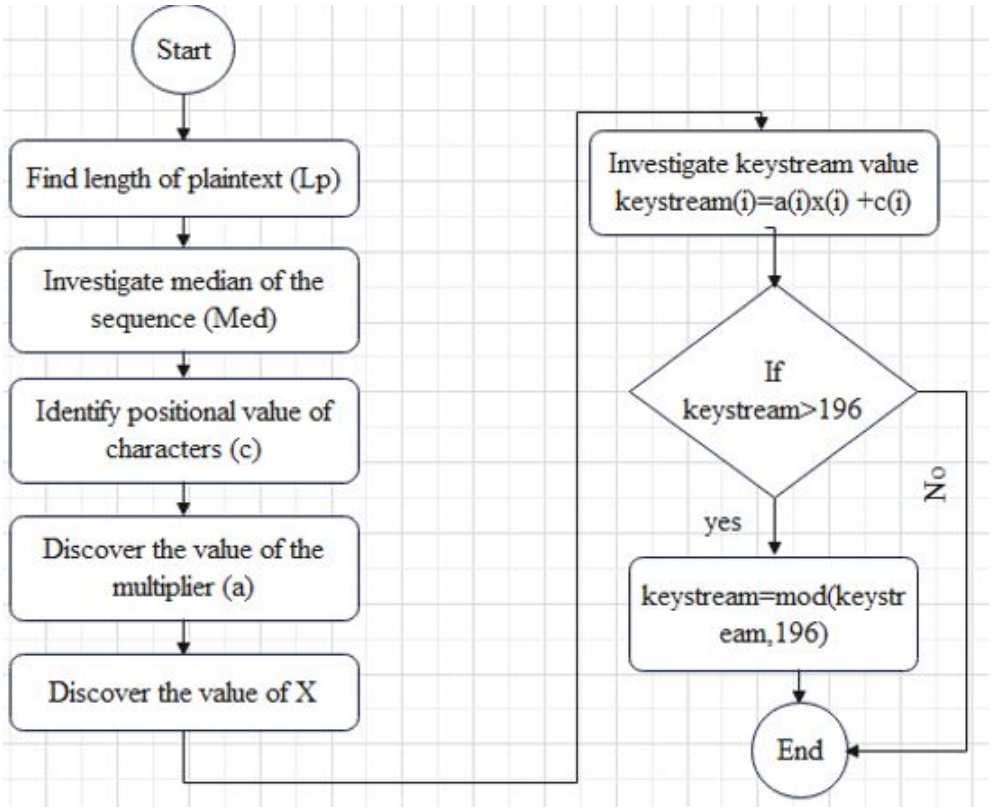


Figure 3.2: Keystream value flow diagram

For example, let us consider that the supplied seed value (Se) is 25 and using this seed let us discover the keystream value of each character for the plaintext message '**Tefera**'. First, we need to find the length of plaintext (Lp) and it becomes 6. Then we are required to investigate the median (Med) of the random number sequences which are generated using modified BBS algorithm and it becomes 64. Besides, we also need to identify the positional value of every character from the plaintext and it becomes 1, 2, 3, 4, 5, and 6 respectively as they are placed which means the first character T has a positional value of 1 and the last character has a positional value of 6. Further, let us find the value of **a** using **a=seed+ c˙i** and it becames 26, 27, 28, 29, 30, and 31 respectively. Finally, we should also find **X˙i** which are the sum of Med, a, and c as we described above and it becames 91, 93, 95, 97, 99, and 101 respectively. So, the keystream values are investigated as follows using equation (3.9) and if necessary we will use equation (3.10).

Keystream value of character **T**

T=ax + C

=26(91)+1

=2367. since it is greater than 196

=2367 mod 196

=15

Keystream value of character **e**

e=ax + C

=27(93)+2

=2513. since it is greater than 196

=2513 mod 196

=161

Keystream value of character **f**

f=ax + C

=28(95)+3

=2663. since it is greater than 196

=2663 mod 196

=115

Keystream value of character **e**

e=ax + C

=29(97)+4

=2817. since it is greater than 196

=2817 mod 196

=73

Keystream value of character **r**

$$\mathbf{r}=ax + C$$
$$=30(99)+5$$
$$=2975. \text{ since it is greater than } 196$$
$$=2975 \bmod 196$$
$$=35$$

Keystream value of character **a**

$$\mathbf{a}=ax + C$$
$$=31(101)+6$$
$$=3137. \text{ since it is greater than } 196$$
$$=3137 \bmod 196$$
$$=1$$

So, the generated keystream value of each characters are 15, 161, 115, 73, 35, and 1 respectively. These keystream value of each character will be added with the index value of each plaintext character. We will show how the keystream and index value will be combined in the encryption part later on.

3.4 How to investigate median of the sequence?

The median of a sequence is the middle number which divides the length of the sequence in equal parts after arranging it in increasing order. In this paper, we have to follow the below steps to find the median of the sequence which were generated by modified BBS as we illustrated above.

- **Step 1**:Arrange the generated random number sequences in increasing order
- **Step 2**:Discover the length of the sequence
- **Step 3**:Once we have got the length of the sequence, if the length is odd the median will be the middle number. If the length of the sequence is even, the median value

will be the average of the two middle random number sequences and when the resulting average value gives a fraction of numbers it will use the floor function of a number.

Thus, in our paper we can say that the median is the middle number because we have generated a random number sequences that have odd length as we illustrated in the example under subtitle 3.2.

3.5 Why the median value is required?

In this proposed work, we used a median value for the two basic operation purposes. The first one is to encrypt the four Blum prime numbers and seed value to securely exchange it with the receiver. As we discussed before, in order to enable the receiver to generate random sequences we only need to send the four Blum prime numbers and seed value rather than exchanging the generated random sequences. So, we encrypt the four Blum prime numbers in a cipher chaining mode method with the median value as the initialization value as the following figure 3.3 depicted and the seed value we used in the random generation process is also encrypted with the median value and appended from the last position of the encrypted four Blum prime numbers and send to the receiver. The second purpose of the median value in our proposed paper is to investigate the keystream value of each plaintext character and ciphertext character. We have shown how the median value is used in order to investigate keystream value of each plaintext character in figure 3.2. So, discovering the median value from the generated random sequence gives a great role for the listed two purposes in our proposed algorithm. From the above example what we have discussed, let's draw a graph that shows how the four Blum prime number is encrypted with median value in CCM and how the seed value is encrypted with median value.

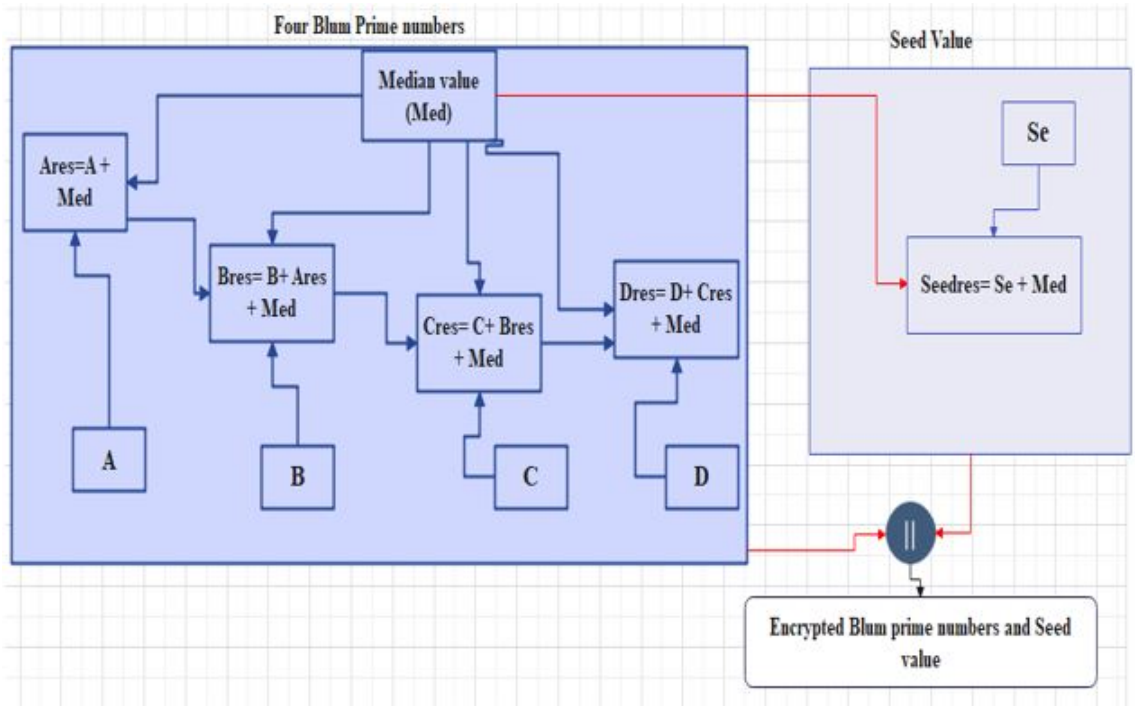


Figure 3.3: Encrypted Blum prime numbers and Seed value flow diagram

As the above figure shown, the four Blum prime numbers and seed values are encrypted with the median value in the CCM way and finally concatenated and send to the receiver. So, the receiver receives A˙res, B˙res, C˙res, D˙res, and Seedres.

3.6 PROPOSED ALGORITHM

The proposed model of enhancing the security of playfair cipher using modified BBS and keystream values is described in this section. Playfair cipher is a symmetric substitution cipher that encrypts a pair of characters or bigram at a time. It was considered a strong encryption algorithm though as technology advances through time the playfair becomes vulnerable for different types of attacks. In our method, first, we accept four Blum prime numbers and seed values from the user. Next, we used a modified BBS algorithm to generate random number sequences as depicted in figure 3.1. The generated random sequences will use a modules operator if each sequence is greater than 196 and find its equivalent character for each sequence to use as a key for the generation of a 14 * 14 matrix. Then, we investigated the median of the generated random numbers and named Med and it helps to encrypt four Blum prime numbers with CCM and seed value to exchange the four Blum prime numbers and seed value with the receiver in a secured way. Besides, we also com-

bine keystream values of each character with the index value of every character in order to get cipher text one. We have discussed how we could discover keystream values of each plaintext character in 3.3. Then after, we investigated the average index value of the plaintext characters and combined them with cipher one, and called the resulting cipher as cipher two. Further, we also changed the original playfair encryption rule. Once we have finished the above steps, we encrypt the resulting cipher text or cipher two using the key matrix generated with modified BBS algorithms depending on the modified playfair cipher encryption rule. In the end, we have appended the Med equivalent character from the beginning of the final cipher and the average index value equivalent character at the end of the final cipher and named Final˙Orginal˙Cipher. Finally, Final_Orginal_Cipher will be sent to the receiver.

One of the challenges in the symmetric encryption techniques is getting a secured channel to distribute encryption and decryption keys. Distributing keys beforehand makes the key vulnerable to the attacker. In order to solve this secure key distribution problem we exchange the four Blum prime numbers and seed value instead of the encryption/decryption keys to enable the receiver to generate the keys at their side rather than sharing. In order to share the four Blum prime numbers we encrypted them using CCM as figure 3.3 shows. In our proposed work, we have three basic functions namely key generation, encryption, and decryption. Each one of these functions will be discussed in detail in the next sub-sections.

3.6.1 Modified playfair cipher rule

As we stated in classical playfair cipher in chapter two, playfair cipher has three rules that are used for the encryption and decryption of bigrams character. Here, we have modified the three encryption and decryption rule of playfair cipher to generate a strong cipher which does not show any correlation between the plaintext and cipher text characters. One of the problems of the playfair cipher is that the encrypted bigram cipher text shows the plaintext clue when the bigram characters are paired from the beginning and end part of the key matrix as we discussed in the statement of the problem. The encryption rules of our proposed work are listed as follows:

- If the bigram character pairs are located in the same row of the key matrix, the equivalent bigram cipher character pairs will be the character below each bigram

character in other row (with wrapping to the beginning row of the matrix if the bigrams plaintext letter are from the last row of the key matrix).

- If the bigram character pairs are located in the same column of the key matrix, the equivalent bigram cipher character pairs will be the previous character to the left of each character in other column (with wrapping to the last column of the matrix if the bigrams plaintext character are from the first column or most left column of the key matrix).

- If the bigram character pairs are located neither in the same row nor the same column of the key matrix, the equivalent bigram cipher characters will be the next character to the right in the same row (with wrapping to the left most position of a row if the bigrams plaintext letter are from the right most position of the row).

In order to retain the plaintext from the cipher text, the reverse process of encryption which is also called decryption process is required. Decryption is performed by reversing the above encryption process. So, the decryption process of our proposed work follows the following three rules:

- If the cipher text character pairs are located in the same row of the key matrix, the equivalent plaintext character pairs will be the character above for each cipher text character in other row (with wrapping to the last row of the matrix if the bigrams cipher text letter are from the first row of the key matrix).

- If the cipher bigram letters are located in the same column of the key matrix, the equivalent bigram plaintext characters will be the next character to the right of each cipher character in other column (with wrapping to the first column of the matrix if the bigrams cipher text letter are from the last column of the key matrix).

- If the cipher character pairs are located neither in the same row nor the same column of the key matrix, the equivalent bigram plaintext characters will be the previous character to the left in the same row (with wrapping to the right most position of a row if the bigrams cipher text letter are from the left most position of the row).

3.6.2 Key Matrix generation of the proposed work

In order to generate the key matrix, first we generated a random number using the BBS algorithm with four Blum prime numbers. Then, the generated random sequence will be a key for the matrix generation after finding the equivalent character of every sequence. The proposed 14 x 14 key matrix is generated using the key which results from the random sequence. In our key matrix generation method, we don't remove the repeated character or symbol of the key rather we choose another symbol or character for that repeated character unlike most researchers did because removing the repeated symbol of the key makes the key length very short and easily predicted by the attacker. In our key matrix generation algorithm, first, the algorithm analyses whether any repeated symbol is present or not. Then, If a repeated character is occured in the key characters, it removes the repeated symbol and replaces that with another symbol that is not a member of the key characters.

In the matrix, the key characters are filled in the first row starting from the left to the right and top to bottom after replacing repeated key characters with other characters. The remaining symbols filled the rest of the matrix depending on the indices value we assigned. Symbols which is the member of the key and is filled in the first row of the matrix cannot be filled again in the other matrix cell. In this proposed work, we don't send the key directly to the receiver rather we share the four Blum prime numbers and the seed value by encrypting them in encryption techniques. So, the receiver generates the random sequence using those Blum prime numbers and seed value after decrypting the received cipher which is the encrypted seed value, and four Blum prime numbers. For instance, let B1, B2, B3, and B4 be the four Blum prime numbers with values 3, 7, 11, and 19 respectively, and let's generate the random sequence using the BBS algorithm. It becomes as follows:

BP1=(B1*B2)+2 where BP1 is the first combined Blum prime number =(3*7)+2 =23 BP2=(B3*B4)+2 where BP2 is the second combined Blum prime number =(11*19)+2 =211 Now,we need to calculate the modules value (Mv)by using the two combined Blum numbers namely BP1 and BP2. So, Mv=BP1*BP2 =23*211 =4853 Then,here we need to select seed value (Se) which is relative prime to Mv. i.e., GCD(Se,Mv)=1 and Let Se=24

Now, we can generate the random sequence using BBS algorithms using four Blum prime numbers, seed value, and Mv as an argument. The generated sequence is from R1, R2, R3, ..., and R15. Since, we need to generate 15 random sequences and the length of the key

also becomes 15. After performing the sequence generation process the sequences from R1 to R15 become 184, 8, 93, 72, 79, 46, 29, 65, 185, 159, 194, 24, 184, 8, and 93 respectively. So, the equivalent character of these random sequences are shown in the figure below:

ÓH¬>;j2.Õ'åXÓH¬

Figure 3.4: Random sequences equivalent characters

These symbols are the equivalent symbols of the random number sequence and the last three sequences from the fig34 are repeated and are the same with the first three symbols. So, as we stated before, here, we don't remove this repeated character instead we have changed it by other symbols as follows:

ÓH¬>;j2.Õ'åX!:G

Figure 3.5: updated random sequences equivalent characters

in figure 3.5, the last repeated characters are replaced by other characters. So, the final key is the random sequence equivalent symbol after replacing repeated sequence equivalent by other characters as I showed in the above sequence. Thus, the key became:

key = ÓH¬>;j2.Õ'åX!:G

Figure 3.6: Key characters

Now, after getting the key we can generate a 14 x 14 key matrix and fill it with the key characters and othe rcharacters accordingly with the index value. Using the above key, the key matrix became looks as follows in the figure.

Figure 3.7: Key Matrix generation

In the above figure 3.7, the key characters are filled in the first and second rows of the matrix from left to right and top to bottom as we discussed before. The following figure shows where the key characters are filled in the matrix and all are encircled.

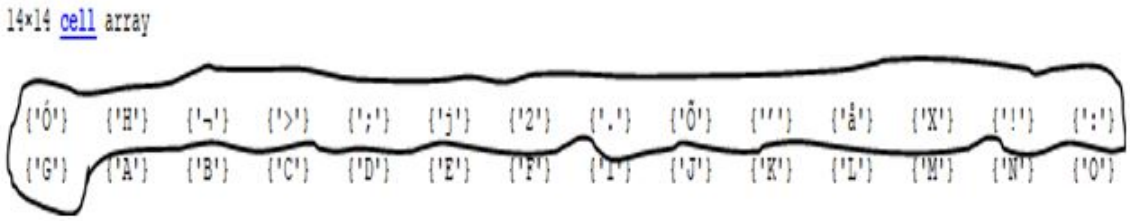


Figure 3.8: Key characters in the matrix

3.6.3 The encryption process of the proposed work

In this section, the encryption process of our proposed algorithm is discussed in detail. We have followed the following rules to encrypt plaintext message.

1. Accept the four Blum prime numbers represented as A, B, C, and D from the user.
2. Combine the Four Blum prime numbers in to two Blum prime numbers using equation (3.2) and discover the modules value (Mv) from the two combined numbers using equation (3.3).
3. Generate the seed value (Se) which is primitive root with the modules value

4. Generate random sequences with BBS algorithm using the above modules value and seed value. If the sequences are greater than 196 we will use a modules operation.

5. Investigate the median (Med) of the sequence

6. Encrypt the four Blum prime numbers and seed value using this median value as shown in figure 3.3 and sent to the receiver via any communication medium.

7. Find the equivalent character of every random sequence and named as key

8. Remove repeated character of the key and replace by another character

9. Construction of key matrix

    - The cells of the matrix are first filled with the characters of the key in row major order starting with the first row as shown in figure 3.8.
    - The remaining cell of the matrix are then filled with the remaining characters from the total set of characters depending on the indices value assigned in ascending order.

10. Accept the plaintext to be encrypted from the user

    - Here, all characters of the plaintext must be from the character set
    - Space has also considered as a one character under the character set
    - Find the length of the plaintext character and if it is odd, space is added to the end in order to make the length even.

11. Discover the positional value of each plaintext character

12. Discover the index value of each plaintext character from the character set

13. Investigate keystream value of each plaintext character. We have deeply discussed how we could find keystream values of each plaintext character under 3.3.

14. Combine the index value of each plaintext character and keystream value of each plaintext character and we called the resulting value as cipher_one.

    - If the cipher˙one value is greater than the total number of character set we will use modules operation.

15. Investigate the average index of the plaintext character

    - If the average index value of the plaintext characters became zero we will add 1.

$$i.e., Averageindexvalue = Averageindexvalue + 1 \tag{3.11}$$

16. Sum up average index value on every cipher˙one value to get cipher two and if it is greater than 196 we will use a modules operation.

17. Divide cipher two character in to bigrams

18. Encrypt the resulting bigrams using modified playfair encryption rule to get final cipher

19. Append the median equivalent character from the left most of the final cipher and average index equivalent character from the right most position of the final cipher and we named the resulting text as Final_Orginal_Cipher.

20. Once we have completed the above steps, here, Final_Orginal_Cipher will be sent to the receiver.

The following example shows the details of every steps which are listed in the above encryption process. It shows a deep explanation about the encryption process and the encryption flow diagram.

1. Accept the four Blum prime numbers from the user and let A=3, B=7, C=19, and D=23.

2. Combine the four Blum prime numbers in to two Blum prime numbers. It looks like as follows:

BP1=(A*B)+2
=(3*7)+2
=23 and
BP2=(C*D)+2
=(19*23)+2
=439

Here,the four Blum prime numbers 3,7,19,and 23 are combined and form a two Blum

prime numbers 23 and 439.

thus,

Mv=(BP1*BP2)

=(23*439)

=10,097

3. Generate the seed value which has a GCD of one with Mv and we took 59 as the value of a seed.

4. Generate random sequences using the seed value and Mv. The generated sequences are: 149 177 186 22 3 103 72 11 27 68 109 64 42 161 57

5. Investigate the median value of the above sequences and it became 68.

i.e., Med=68

6. Encrypt the four Blum prime number and seed value with the median value and the results are: 71 146 233 324 127. Sender sent 71 146 233 324 127 to the receiver rather than sending the four Blum prime numbers accepted from the user. Here, the first four numbers 71 146 233 324 are the result of the encrypted four Blum prime numbers and the last value 127 is the result of the encrypted seed value. The four Blum prime numbers are putted first and at the last encrypted seed value is appended.

7. Find the equivalent character of every random sequence.
   Here, the sequences are: 149 177 186 22 3 103 72 11 27 68 109 64 42 161 57. So, the equivalent character of this random sequence from the character set and it became the key. Thus, the key became:

…É×VC¼>K0#Ä‚f'u

Figure 3.9: key character for proposed example

8. Remove repeated characters of the key and replace by another character. For this example, this step is skipped since the key characters are not repeated. Due to this reason, we did not need to change anything and skipped this steps.

9. Construction of key matrix

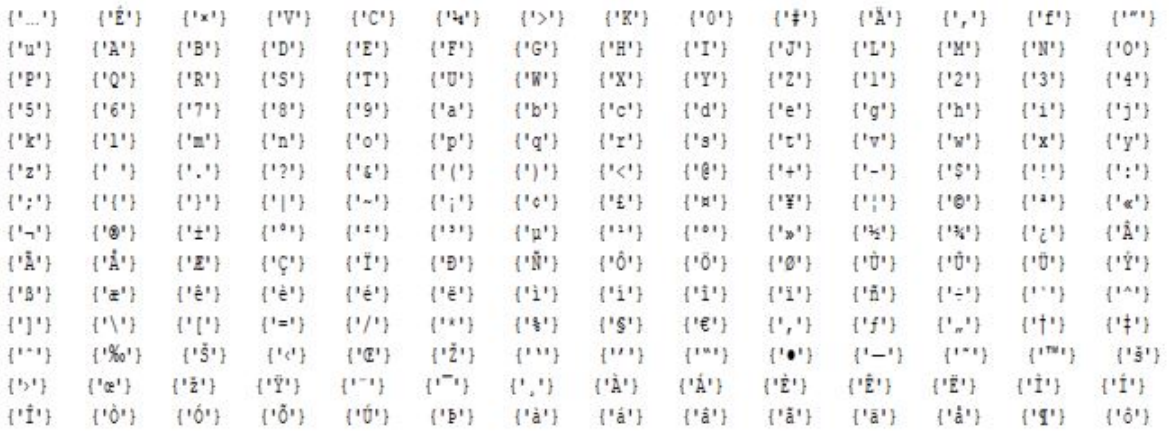


Figure 3.10: Proposed encryption key matrix

10. Accept the plaintext to be encrypted

Plaintex=Tefera Alagaw

11. Discover the positional of the plaintext characters and it became 1 2 3 4 5 6 7 8 9 10 11 12 13 14 respectively.

12. Discover the index value of each plaintext characters and it gives 20 41 42 41 54 37 63 1 48 37 43 37 59 63 respectively.

13. Investigate keystream value of each plaintext characters: the keystream value of each plaintext characters are: 97 153 17 81 149 25 101 181 69 157 53 149 53 157 14 respectively.

14. **Cipher_one**: it is the combination of keystream value of each plaintext characters and index value of each plaintext characters. It gives the following indices value:

117	**194**	**59**	**122**	**7**	**62**	**164**	**182**	**117**	**194**	**96**	**186**	**112**	**24 14**

Figure 3.11: Plaintext index value

15. **Average index value**: the average index value is the remainder of the sum of all character indices value moduled by the length of the plaintext characters. So, the above plaintext character average index value is **12**.

16. **Cipher Two**: it is the sum of cipher one value and average index value. The indices value of cipher two are: 129 10 71 134 19 74 176 194 129 10 108 2 124 36. In order to separate this values in to bigrams, first we need to change every number in to equivalent characters and it became:

ìJ<÷S+ÈåìJÃBæ9

Figure 3.12: Cipher Two index equivalent characters

17. So, the bigrams are:

ìJ
<÷
S+
Èå
ìJ
ÃB
æ9

Figure 3.13: Cipher Two bigrams

18. The above bigrams are encrypted with the modified playfair encryption rule. The following table shows the bigrams and its equivalent cipher bigrams.

No	Plaintext Bigrams	Cipher text Bigrams
1	**ìJ**	**íL**
2	**<÷**	**@`**
3	**S+**	**T-**
4	**Èå**	**Ê¶**
5	**ìJ**	**íL**
6	**ÃB**	**ÅD**
7	**æ9**	**Ëa**
Final cipher		**íL@`T-Ê¶íLÅDêa**

Figure 3.14: Bigrams plaintext character and its equivalent bigrams cipher

19. Append the median equivalent character from the beginning of the final cipher and average index value from the end position of the final cipher to get Final_Orginal˙Cipher. Final_Orginal_Cipher became:

Figure 3.15: After appending median equivalent character

As we saw here, the character # is appended as the median equivalent characters and character L as average index value equivalent character.

20. Finally, Final_Orginal_Cipher will be sent to the receiver.

In the above example, we have shown that the plaintext **Tefera Alagaw** is encrypted with the proposed algorithm and provides its equivalent ciphertext. We will take this ciphertext and decrypt to get the plaintext in the decryption process.

3.6.3.1 Proposed Encryption Flow Diagram

The following diagram depicts the flow diagram of the encryption process which is required to follow to encrypt plaintext message. It shows the overall process of proposed encryption work.

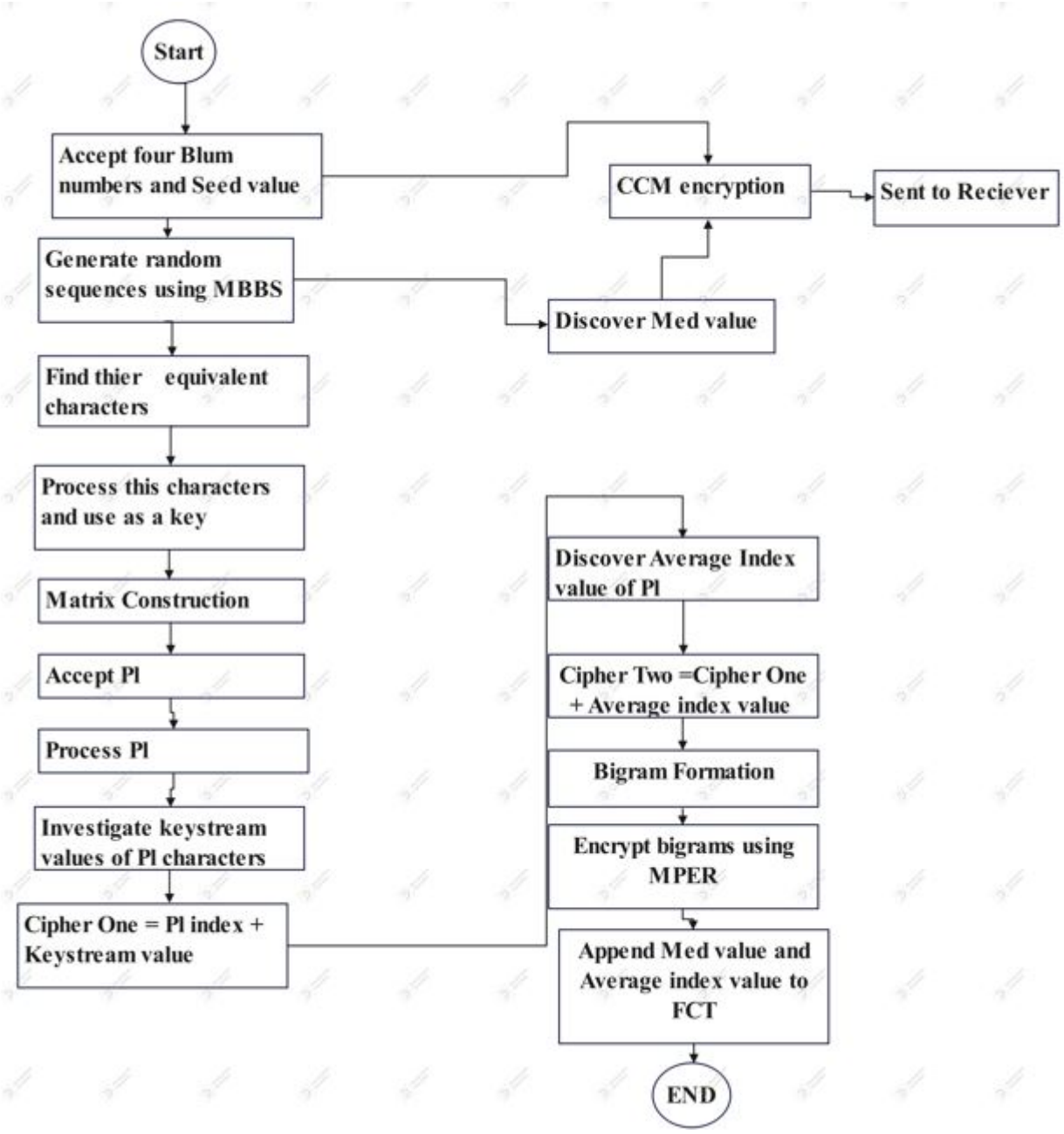


Figure 3.16: Encryption Flow Diagram

3.6.4 The Decryption process of the proposed work

Decryption is the reverse process of encryption which makes the plaintext unreadable. Decryption helps to retain the original plaintext from the cipher text. It converts unreadable form of text in to readable form of text. In our proposed paper, the receiver first receives Encry_Blum_seed_cipher. Encry_Blum_seed_cipher is the combination of encrypted four Blum prime numbers and encrypted seed value together. Next, receiver receives final cipher sent by the sender. In this section, we have discussed our proposed paper decryption processes. We have also showed an example by decrypting the example we have done in the encryption process. Thus, we have to follow the following steps to decrypt our proposed work:

1. Accept the four encrypted Blum prime numbers and seed value depending on the order of the sender sent. The first four numbers are for Blum prime numbers represented as A, B, C, D and the last number is for seed value represented as Se.

2. Accept the cipher text sent by the sender

3. From the cipher text, split the first character and last character of the cipher text since the first character is the equivalent character of the median value which was appended during encryption and the last character was the equivalent character of the average index value of the plaintext character which was appended. Therefore, the final cipher text became the character excluding the first and last character of the cipher which was sent by the sender and also its length is deduced by two.

4. Discover the equivalent value of the median character and the average index value character.

5. Decrypt the four Blum prime number and seed value using the median value

6. Once we got the original value of the four Blum prime numbers and seed value, we have to combine the four Blum prime numbers in to two Blum prime and find the modules value.

7. Generate random sequences using modules value and seed value. If the sequences are greater than 196 we will use a modules operation.

8. Find the equivalent character of the random sequences from the character sets and used as a key.

9. Replace repeated characters of the key by another characters which are not a member of a key characters. This step is done automatically by the algorithm. The algorithm analyses whether repeated character present or not and if repeated character present it will automatically change it. Therefore, the final key became the key characters after this process is done.

10. Matrix is constructed using the above final keys

    - Here, as we discussed in the encryption process, the key characters are first filled row wise.
    - The remaining cells of the matrix are filled with the remaining characters of the characters set depending on the index value we assigned.

11. Then, the processed final cipher is grouped in to bigrams

12. The bigrams are decrypted using modified decryption rule of playfair cipher.

13. The decrypted bigrams together gives cipher two. As we stated in the encryption process, cipher two is the combination of cipher one and average index value.

14. Here, we need to investigate the equivalent value of the cipher two characters. Then, to get cipher one just subtracting the average index value from every cipher two characters equivalent value.

15. Investigate keystream values of cipher one characters. We have shown how we could find keystream values in the above.

16. Subtract keystream values from equivalent value of cipher one characters.

    - If the resulting value is less than zero, we will add 196 to get positive inger value.

17. Finally, the original plaintext characters will be the equivalent characters of the above results. It simply mean that we got the equivalent index value of original plaintext

message when we subtract keystream value from cipher one equivalent value.

$$plaintextindexvalue = cipheroneequivalentvalue - keystreamvalue \quad (3.12)$$

The following example shows the details of every steps which are listed in the above decryption process. It shows a deep explanation about the decryption process and the decryption flow diagram. In this example, we are going to decrypt the examples we have done in the encryption process. Here are all the steps to decrypt it.

1. Accept encrypted Blum prime numbers and encrypted seed values. So, the receiver receives 71 146 233 324 127 number. These are encrypted four Blum prime numbers and seed value. As we stated before, the first four numbers are for the four Blum prime numbers (A, B, C, D) respectively and the last numbers are for the seed value. So,

   ENC˙A=71

   ENC˙B=146

   ENC˙C=233

   ENC˙D=324

   ENC˙SE=127

   Where,

   ENC˙A=encrypted value of A,

   ENC˙B=encrypted value of B

   ENC˙C=encrypted value of C

   ENC˙D=encrypted value of D

   ENC˙SE=encrypted value of seed value

2. Accept the cipher text sent by the sender. So, the cipher text is:

   Cipher text=#íL@`T-Ê¶íLÅDêaL

   Figure 3.17: ciphertext sent by the Sender

3. Here, we need to identify and split the first and last characters of the cipher text. As we saw in the cipher text, character# is the first character of the cipher text and used

as the equivalent characters of the median value and character **L** is the last character of the cipher and used to represent the equivalent character of the average index value. Its length is reduced by two when we separate the first and last characters.

4. Find the equivalent value of the median character and average index value character. The equivalent value of the median character # is 68 and the equivalent value for the average index value character **L** is 12.

5. In this step, it is needed to decrypt the encrypted four Blum prime numbers and encrypted seed value using the equivalent value of median character. As we illustrated before the four encrypted Blum prime numbers represented as A, B, C, and D are 71 146 233 324 and the encrypted seed value is 127. So, when the decryption is performed, it must start from the beginning as follows:

A˙org=ENC˙A-Med
=71-68
=3

B˙org=(ENC˙B-Med)-ENC˙A
=(146 -68)-71
=7

C˙org=(ENC˙C-Med)-ENC˙B
= (233-68)-146
= 19

D˙org=(ENC˙D-Med)-ENC˙C
=(324-68)-233
= 23

SE˙org=(ENC˙SE-Med)
= 127-68
= 59

Where,

Aorg=Aorg is the origional value of Blum Prime A

Borg=Borg is the origional value of Blum prime B

C˙org=Corg is the origional value of Blum prime C

D˙org=Dorg is the origional value of Blum prime D &

SE˙org=SEorg is the origional value of seed value.

Therefore, the decrypted value of the four Blum prime numbers represented as Aorg, Borg, Corg, and Dorg are 3, 7, 19, 23 and the decrypted value of the seed value represented as SEorg is 59.

6. Once we got the original value of the four Blum prime numbers , we could combine the four Blum prime numbers in to two Blum prime to find the modules value. So,

BP1=(Aorg*Borg)+2

=(3*7)+2

=23

BP2=(Corg*Dorg)+2

=(19*23)+2

=439

then,the modules value became:

Mv=BP1*BP2

=23*439

=10,097

Where,

BP1 is the first combined Blum prime number,

BP2 is the second combined Blum prime number,

Mv is the modules value

7. Here, random sequences are generated using the above modules value (Mv) and Original seed value (SEorg). The generated random sequences are 149 177 186 22 3 103 72 11 27 68 109 64 42 161 57. These sequences are generated using BBS algorithm as shown in 3.2.

8. In this step, we found the equivalent characters of the above sequences to use as a key. So, the equivalent characters is:

…É×VC¼>K0#Ä‚f”u

Figure 3.18: Equivalent characters for the sequences generated in the decryption pocess

9. Replace repeated character of the key by another character. Here, we don't have repeated characters. So, this step is skipped since no repeated characters of the key are present.

10. In this step, the decryption key matrix is constructed using the keys we got in step 8. The key matrix is constructed in similar way as we construct the encryption key matrix. Here, we can able to generate the same matrix to that of the encryption matrix without exchanging of keys.The decryption key matrix looks like as follows in the figure:

14×14 cell array

{'…'}	{'É'}	{'×'}	{'V'}	{'C'}	{'¼'}	{'>'}	{'K'}	{'0'}	{'#'}	{'Ä'}	{','}	{'f'}	{'"'}
{'u'}	{'A'}	{'B'}	{'D'}	{'E'}	{'F'}	{'G'}	{'H'}	{'I'}	{'J'}	{'L'}	{'M'}	{'N'}	{'O'}
{'P'}	{'Q'}	{'R'}	{'S'}	{'T'}	{'U'}	{'W'}	{'X'}	{'Y'}	{'Z'}	{'1'}	{'2'}	{'3'}	{'4'}
{'5'}	{'6'}	{'7'}	{'8'}	{'9'}	{'a'}	{'b'}	{'c'}	{'d'}	{'e'}	{'g'}	{'h'}	{'i'}	{'j'}
{'k'}	{'l'}	{'m'}	{'n'}	{'o'}	{'p'}	{'q'}	{'r'}	{'s'}	{'t'}	{'v'}	{'w'}	{'x'}	{'y'}
{'z'}	{' '}	{'.'}	{'?'}	{'&'}	{'('}	{')'}	{'<'}	{'@'}	{'+'}	{'-'}	{'$'}	{'!'}	{':'}
{';'}	{'['}	{']'}	{'|'}	{'~'}	{'¡'}	{'¢'}	{'£'}	{'¤'}	{'¥'}	{'¦'}	{'©'}	{'ª'}	{'«'}
{'¬'}	{'®'}	{'±'}	{'°'}	{'²'}	{'³'}	{'µ'}	{'¹'}	{'º'}	{'»'}	{'½'}	{'¾'}	{'¿'}	{'Ã'}
{'Å'}	{'Å'}	{'Æ'}	{'Ç'}	{'Ï'}	{'Ð'}	{'Ñ'}	{'Ô'}	{'Õ'}	{'Ø'}	{'Ù'}	{'Û'}	{'Ü'}	{'Ý'}
{'ß'}	{'æ'}	{'ê'}	{'è'}	{'é'}	{'ë'}	{'ì'}	{'í'}	{'î'}	{'ï'}	{'ñ'}	{'÷'}	{'`'}	{'^'}
{']'}	{'\'}	{'['}	{'='}	{'/'}	{'*'}	{'§'}	{'§'}	{'€'}	{','}	{'ƒ'}	{'„'}	{'†'}	{'‡'}
{'ˆ'}	{'‰'}	{'Š'}	{'‹'}	{'Œ'}	{'Ž'}	{'‘'}	{'’'}	{'“'}	{'•'}	{'—'}	{'˜'}	{'™'}	{'š'}
{'›'}	{'œ'}	{'ž'}	{'Ÿ'}	{'¯'}	{'¯'}	{'¸'}	{'À'}	{'Á'}	{'È'}	{'É'}	{'Ë'}	{'Ì'}	{'Í'}
{'Î'}	{'Ò'}	{'Ó'}	{'Õ'}	{'Ù'}	{'Þ'}	{'à'}	{'á'}	{'â'}	{'ã'}	{'ä'}	{'å'}	{'æ'}	{'ô'}

Figure 3.19: Decryption key matrix

11. Group the processed final cipher in to bigrams. Processed final cipher is the final cipher text after splitting the first and last characters of the received cipher. Therefore, the bigrams are:

íL
@`
T-
Ê¶
íL
ÅD
êa

Figure 3.20: Bigrams formation at the Decryption process

12. The bigrams are decrypted using modified decryption rule of playfair cipher. So, the above bigrams are decrypted as follows in the below figure.

No	Cipher Bigrams	Equivalent decrypted bigrams
1	íL	ìJ
2	@`	<÷
3	T-	S+
4	Ê¶	Èå
5	íL	ìJ
6	ÅD	ÃB
7	êa	æ9
Final decrypted result		ìJ<÷S+ÈåìJÃBæ9

Figure 3.21: Cipher text bigrams and its decrypted equivalent bigrams

13. The above figure final decrypted result is a cipher two cipher. It is the combination of cipher one and average index value. So, cipher two is:

ìJ<÷S+ÈåìJÃBæ9

Figure 3.22: Cipher Two at the decryption side

14. Here, we need to find the equivalent value of cipher two characters. The following table shows the cipher two characters and its equivalent value in the character set.

No	Cipher two character	Equivalent value in the character set
1	Ì	129
2	J	10
3	<	71
4	÷	134
5	S	19
6	+	74
7	È	176
8	Å	194
9	Ì	129
10	J	10
11	Ă	108
12	B	2
13	Æ	124
14	9	36

Figure 3.23: cipher two characters equivalent value

15. From the above figure 3.22, we could find the equivalent value of cipher one's character using Cipher two characters equivalent value and average index value.

$$Cipheronecharactervalue = ciphertwocharactervalue - Averageindexvalue \tag{3.13}$$

The below figure shown the equivalent value of cipher one and its equivalent character.

No	Cipher one equivalent value	Cipher One equivalent character
1	117	Ö
2	194	å
3	59	w
4	122	Ý
5	7	G
6	62	z
7	164	~
8	182	Î
9	117	Ö
10	194	å
11	96	°
12	186	×
13	112	Ç
14	24	X

Figure 3.24: Cipher one equivalent value and its equivalent character in the character set

16. In this step, it is needed to investigate keystream value of cipher one characters. This keystream value helps us to get the original plaintext message. The investigated keystream values are 97 153 17 81 149 25 101 181 69 157 53 149 53 157.

17. Here, to get the plaintext message it is needed to subtract keystream values from equivalent value of cipher one characters.

$$Pl \quad value = cipher \quad one \quad equivalent \quad value - keystream \quad value \tag{3.14}$$

Where,

Pl value is the plaintext equivalent value in the character set.

So, we have shown the equivalent value of the original plaintext characters in the character set and its equivalent characters in the following figure.

No	Pl value	Pl equivalent character
1	20	T
2	41	e
3	42	f
4	41	e
5	54	r
6	37	a
7	63	
8	1	A
9	48	l
10	37	a
11	43	g
12	37	a
13	59	w
14	63	

Figure 3.25: Plaintext characters and its equivalent value

In this figure, we have got the original plaintext characters from the character set using the Pl value. The unfilled space in the figure for the Pl value 63 indicates that the value in the character set is assigned for space. As we saw in the figure, the plaintext character has space at the end since it was added during encryption to make the length of the plaintext even. So, we could remove space after decrypting if we got at the end of the plaintext characters. Even though we leave the space at the end, it will not create any ambiguity. So, the plaintext which was sent by the sender to the receiver is '**Tefera Alagaw**' after removing a space which was added at the end. Without removing a space which was added

at the end, it gives the plaintext as '**Tefera Alagaw** '. Even if a space is not removed, it does not create any ambiguity or it does not show any meaning change.

3.6.4.1. Proposed Decryption Flow Diagram

The following figure shown us the overall steps of our proposed decryption process.

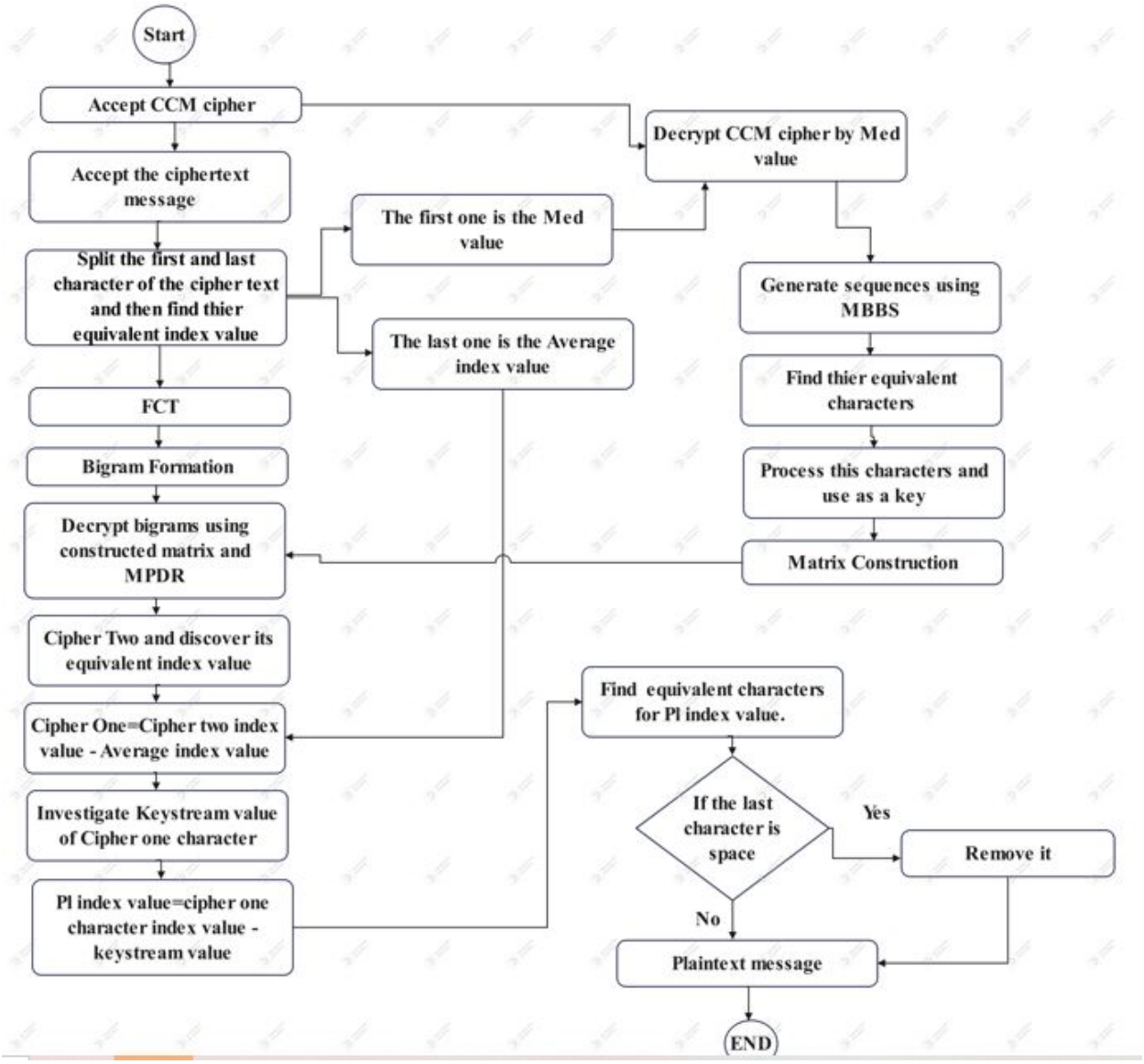


Figure 3.26: Decryption flow diagram

Chapter Four

4 IMPLEMENTATION AND PERFORMANCE EVALUATION

4.1 Chapter overview

This chapter discusses the result of the implementation process to evaluate the performance of the proposed work, the software we used to implement our work, security performance metrics, and analysis of the result we got during the implementation. All we mentioned here are discussed briefly one by one in this chapter.

4.2 Matlab Software

Matlab is a software we used to implement the proposed algorithm. It is a widely used implementation software for technical computing with a focus on matrix operation. Its name stands for "MATrix LABoratory" and primarily designed goal is for doing numerical computations with vectors and matrices. Typical areas of matlab use are:

- Math and Computation
- Modelling and Simulation
- Data Analysis and Visualization
- Application Development
- Graphical user interface development
- Cryptography coding

4.2.1 Default layout

When the matlab software is started for the first time, the desktop appears with the default layout as shown figure 4.1. It has the following parts:

- **Command Window**: is used to run matlab statements.
- **Current Directory**: is used to view, open, search for, and make changes to matlab related directories and files.

- **Command History**: the log of the function we have entered in the command window is displayed here.

- **Workspace**: is used to show each name of variables, its value, and the min and max entry if the variable is a matrix.

4.2.2 Editor

This can be used to create and edit M files, in which we can write and save matlab programs.

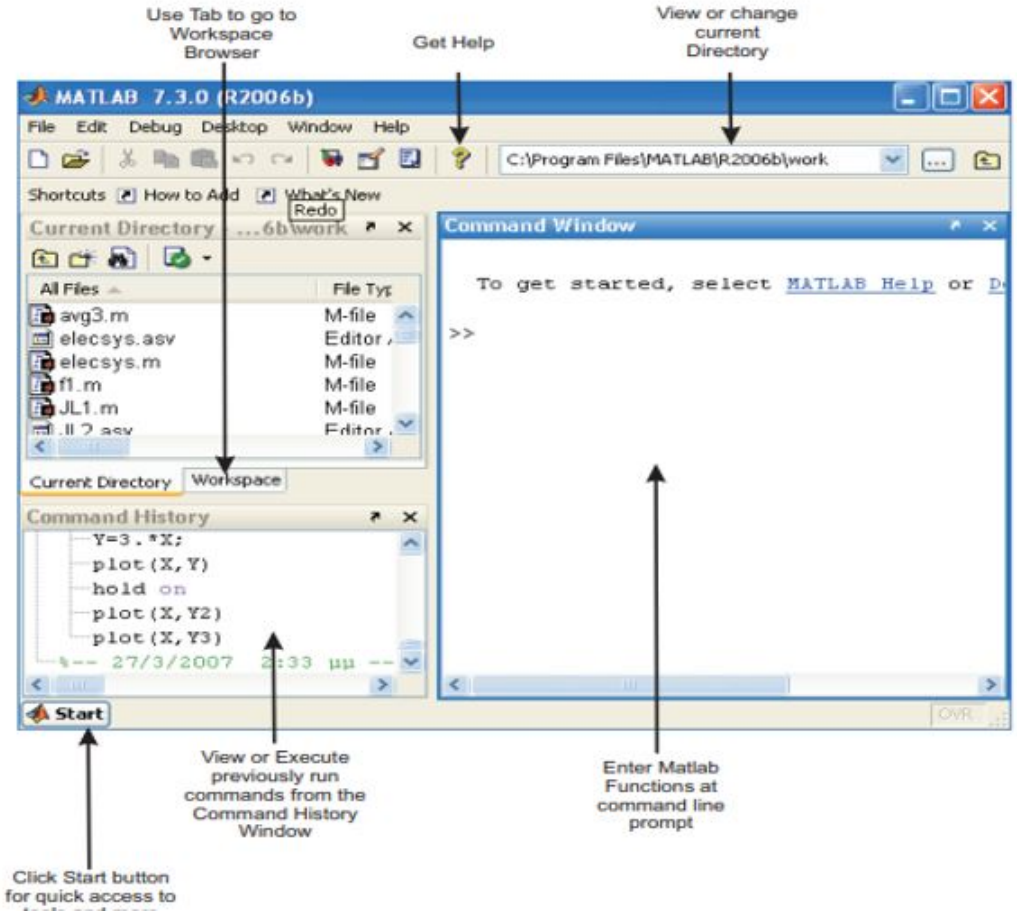


Figure 4.1: Matlab Default layout [48]

4.3 Security Performance Metrics

To evaluate the performance of our proposed work, we have to use security performance metrics since our work is focused on improving the security of Playfair cipher algorithm. So, in this section, we selected the security metrics and described one by one in detail as follows.

4.3.1 Avalanche Effect

Avalanche Effect is one of the features of a secure cryptographic algorithm wherein a slight or single change in the input will result to a drastic change in the output of the algorithm [39]. When we alter one character in the key or plaintext, the result should change within a high avalanche ratio. If the avalanche result shows above 50%, the algorithm is considered as a strong algorithm and the result of the algorithm has become complex for an attacker. The avalanche effect is calculated using the equation below:

$$Avalanche \quad Effect = \frac{No. \quad of \quad changed \quad characters \quad in \quad the \quad ciphertext}{total \quad number \quad of \quad ciphertext \quad characters} \quad * 100\% \tag{4.1}$$

So, Avalanche effect is used to test the performance of one algorithm. It is considered that the algorithm achieve a strict avalanche criterion (SAC) when the avalanche result is greater than 50% due to a single character change of a key or plaintext characters.

4.3.2 Confusion and Diffusion

Confusion and diffusion are the properties of secure cipher which was introduced by Claude Shannon to prevent cryptanalysis based on statistical analysis. They are involved to make the statistical relationship between plaintext, key, and cipher text as complex as possible [45]. Hiding the relationship between plaintext and cipher text is related to the idea of diffusion [26]. An attacker who uses the statics of cipher text to find plaintext is discouraged by the properties of diffusion. In the concept of diffusion, each symbol or character in the cipher text is dependent on some or all characters in the plaintext. In other expressions, if a single character is changed in the plaintext, several or all symbols of the cipher text will also be changed. So, diffusion hides the relationship between plaintext and cipher text.

The idea of hiding the correlation between cipher text and the key is related to the concept of confusion [26]. As we stated before, the idea of confusion also discourages attacker, who uses cipher text to find the key. If a single character or symbol in the key is changed, there will be a significant change in the cipher text or several cipher text characters will be changed. So, confusion hides the correlation between cipher text and key.

4.3.3 Brute-Force Attack

In cryptography, a brute force attack is a strategy that will be used against any encrypted information. It is the way of checking all potential keys till the proper key is found [46]. Key has a major role in determining the secrecy level of numerous cryptographic algorithms. A brute force attack systematically tries every possible key characters or symbols to get a valid key. For an attacker to make their guessing process significantly easier, only a small number of known bytes or small information about the key is necessary.
The attacker absolutely needs some factor like the key length, the way of generating keys, and the total number of character sets to be able to determine how long it would take to determine the key [47]. If we are using a cryptographically weak process to generate keys, then our whole system became weak. An attacker needs not cryptanalyze our encryption algorithm, rather attacker cryptanalyze our key generation algorithm since it is weak.
The estimated time needed to break one algorithm using brute force attack is calculated by using the following formula.

$$Estimated \quad Time = \frac{(Number \quad of \quad character \quad set^{key \quad length})}{Encryption \quad time \quad (second)} \tag{4.2}$$

4.3.4 Frequency Analysis

In cryptanalysis, frequency analysis is a way of dealing with the frequency of letters or groups of letters in a cipher text [45]. It is based on the fact that, in any given cipher text, certain letters and combinations of letters occur with varying frequencies. We have shown the frequency distribution of English alphabets in chapter two using a table. In the 5x5 matrix, the probability of occurrence of any particular element on average is:

1/26=0.0384

4.4 Analytical result

Based on the implementation of the proposed algorithm, we got results that overcome the drawbacks which are stated in chapter one under the statements of the problem. Our primary concern was solving the drawbacks of extended playfair algorithm using Fibonacci series and improving the security of the playfair algorithm. So, our proposed work implementation became successful and we have achieved our objectives. In this section, we have discussed the results in detail as follows.

4.4.1 Key generation and key exchange

Firstly, we could generate keys for the encryption and decryption process using a modified BBS algorithm. This key is strong and cannot be predicted simply by the attacker as it is generated randomly. When repetitions of character happen in the key character, it will be replaced by another character from the character set and this helps to not deduce the length of the key due to repetition of characters during key processing. Our proposed work way of generating key is better than the base paper research work and original playfair cipher algorithm since the base paper uses the Fibonacci series to generate keys. We also enabled to exchange the keys in a secure way because we did not exchange the key directly instead we shared the numbers used to generate keys with the receiver. So, the receiver can able to generate the same keys like the sender using those numbers received from the sender. We also encrypted the numbers which are sent to the receiver using a median value to make those numbers more secure. We have shown an example in chapter three which depicts how the proposed key generation looks like and how the numbers which help the receiver to generate key is encrypted by the median value. The paper done by Mohd Vasim and et al [10] exchange the keys directly to the receiver and it uses a weak method to generate keys. This limitation is solved by our proposed work. As a result, our paper is better in key generation and key exchange than an extended playfair algorithm using Fibonacci series and original playfair algorithm.

The below diagram clearly shows the keys which are generated using the proposed algorithm and the compared algorithm. The proposed algorithm key is very complex and can not be predicted by an attacker than the compared algorithm key. The keys of the proposed algorithm kept the randomization property and it became complex for the attacker to crack

through cryptanalaysis attack.

	Algorithms	
No	*Extended Algorithm*	*Proposed Algorithm*
1	89ER7YA	a6ÖÃè)LéC,-r°¬e
2	7YAjzfb	>ÄcÁx‹*À¼ªáQ.Ì4
3	YAjzfbI	]:oXÄí*È6Õ¯K†„²

Figure 4.2: Key Comparison

4.4.2 Filler character

As we discussed in the statements of problem, filler character is the character needed to split when the same characters are paired together during formation of bigram. Filler character creates ambiguity during decryption at the receiver side since receiver did not know whether that character is a part of the original message or added as a character to split bigrams of the same paired character. Further, it also increases the length of cipher text character if a lot of filler characters are needed during encryption. The paper in [10] and original playfair algorithm requires filler character if the bigrams are paired with the same character. For instance, if the plaintext to be encrypted is BALLON, the paper in [10] and original playfair cipher encrypts as **BALXLONX**. As we saw here in the example, filler character is required and also when the filler character is added, the length of plaintext message increased from 6 to 7 and it again requires additional character to make the length even. Having said this, our proposed work did not require filler character. Our paper overcomes the problem of requiring filler character when the same characters are paired in the bigram. The above example **BALLON** is encrypted as **BALLON** in our proposed work since no need of filler character or it removes necessity of filler character. As a result, our proposed work overcomes the problem of requiring filler character in the paper [10] and original playfair algorithm.

4.4.3 Pad character

In playfair algorithm, pad character is a symbol or character added at the end when the length of the plaintext is odd. In bigram formation of the playfair algorithm, the length of the plaintext must be even unless it is difficult to form bigrams since the last character cannot paired. One of the biggest advantage of pad character is making the length of plaintext even. Most of the time, characters which are not most frequently occured in English alphabet like X, Q are used as a pad character. So, without making the length of plaintext even, it is not possible to form bigram in playfair algorithm. Even though, it has advantage, pad character create ambiguity during decryption at the receiver side like filler character as receiver did not know whether it is pad character or part of the plaintext message. For instance, if the plaintext to be encrypted is **TAX**, it is required to add additional character since the length of the plaintext is odd or three. So, let take **X** as pad character and added it at the end of the plaintext and it became **TAXX**. Now, when the receiver decrypt and get **TAXX**, it became confused due to the last **XX** characters since X is considered as a pad character. If receiver consider **X** as pad and removes, the decrypted plaintext at receiver side became **TA** and it's not the original message sent from the sender. So, using character as a pad character creates ambiguity at the receiver side. The paper in [10] and original playfair cipher algorithm also uses character or symbol as a pad character when it is required. In our proposed work, we did not use character as a pad character instead we use space. Space is not visible when it is at the end and could not create confusion for the receiver. Thus, we use space as pad character in our proposed work. The above example **TAX** is encrypted as **TAX** in our proposed work. There is space next to the character **X** at the end but it is not visible and cannot confused receiver during decryption. As a result, using space as a pad character is better than that of using other symbol or character. So, the proposed work uses space as pad character and better in removing ambiguity than the paper in [10] and original playfair algorithm.

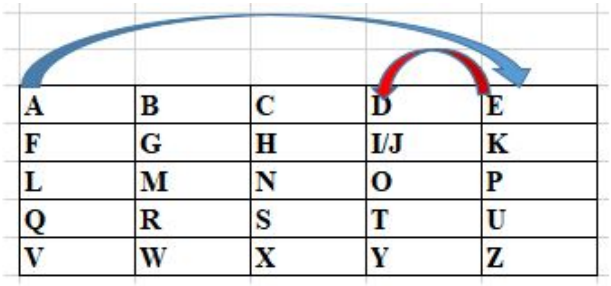
4.4.4 Bigram and its reverse

Bigram is a combination of two letters paired together. In playfair algorithm, a plaintext message is grouped in to bigrams to perform the encryption process. In paper [10] and original playfair algorithm, the bigram and its reverse provide similar cipher pattern. For

example, if the bigram **DE** provides cipher **SF**, then the bigram **ED** provides cipher **FS**. Besides this, in paper [10] and original playfair algorithm, similar bigram delivers similar cipher. For instance, if the plaintext is **EDITED**, then the first bigram **ED** and the last bigram **ED** gives similar cipher. This drawbacks makes the algorithm vulnerable for cryptanalysis attack like frequency analysis attack and pattern attack. In our proposed work, the above limitations are solved since we used position value of each character during investigation of key stream values. As a result, in our proposed work the bigram and its reverse did not deliver similar cipher pattern and also similar bigram did not deliver similar cipher. When we encrypt the plaintext message **EDITED** using our proposed algorithm, the resulting cipher became **ááåVX3**. As we saw here, the first bigram **ED** equivalent cipher is **áä** and the last bigram **ED** equivalent cipher is **X3**. This shows that in our proposed work, similar bigram cannot deliver similar cipher. Therefore, our proposed work is better by this method than the paper in [10] and original playfair algorithm.

4.4.5 Modified playfair encryption rule

Playfair cipher algorithm has three rules which is used for the purpose of encryption process. Most researchers use those three rules of playfair cipher to perform their proposed work on playfair algorithm. But, encrypting messages using those three rules made the cipher easily vulnerable for attacker since it does not hide the relationship between cipher text and plaintext when bigram characters are paired from the beginning and last part of the matrix. The paper in [10] change the three encryption rules but there is still limitations of showing the correlation between plaintext and cipher text. For instance, consider the following is the key matrix and if we want to encrypt the plaintext AE, it gives the cipher text ED. From the example, we could understand that, when we encrypt bigram messages which is paired from the first and last part of the matrix, it shown us some clue about the bigram character. So, attacker can easily know what the plaintext was from the above example. This is done when the bigram characters are on the same row. When the bigram characters are on the same column, it does not also hide the correlation of bigram plaintext and their equivalent cipher text bigrams. The following figure shown us how the given example is being done.

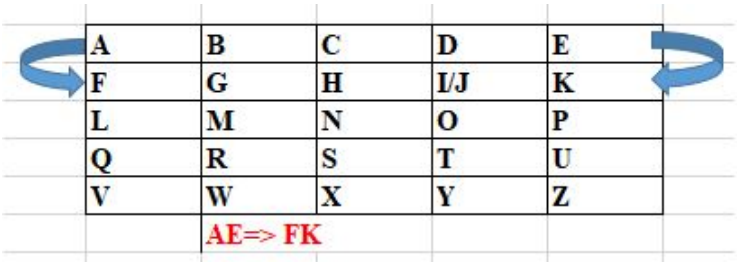


A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z

Figure 4.3: Encryption rule example

In this proposed work, the encryption rules of playfair algorithm is changed by achieving the goals of hiding plaintext and cipher text message. In the proposed work, if the bigram characters are paired from the first and last part of the matrix, the resulting cipher bigram does not show any correlation between plaintext bigram and cipher text bigram. The proposed encryption rule is stated in chapter three briefly. So, the proposed encryption rule is illustrated as follows using diagram representation. We used the above example matrix to show the proposed work encryption rule since the proposed encryption matrix is 14x14, it is difficult to show on it graphically.

- When the bigram characters are on the same row



A	B	C	D	E
F	G	H	I/J	K
L	M	N	O	P
Q	R	S	T	U
V	W	X	Y	Z
	AE=> FK			

Figure 4.4: Modified encryption rule on the same row

Figure 4.4 shown that there is no correlation between plaintext and cipher text when we encrypt bigrams where its character is paired from the first and last part of the matrix. But, in figure 4.3 we have shown its correlation when it is encrypted by the rule of the paper [10].

- When the bigram characters are on the same column

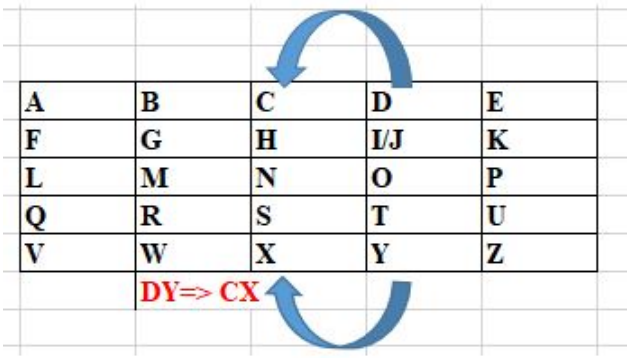


Figure 4.5: Modified encryption rule on the same column

In figure 4.5 ,the plaintext character DY is encrypted using modified encryption rule of playfair algorithm and the resulting cipher became CX. As the figure shown, the bigram character DY is paired from the first and last row of the matrix though it did not show any correlation between the plaintext bigram and cipher text bigram like the paper in [10] and original playfair cipher algorithm.

- When the bigram characters are neither on the same row nor the same column

Figure 4.6: Modified encryption rule neither in same column nor same row

In the above figure 4.6 ,we have shown that the proposed work is encrypted in this way, when the characters are neither in the same row nor column. Generally, as the above figure shown, the modified encryption rule of the playfair algorithm did not show any correlation between plaintext and cipher text. As a result, the proposed work is better in hiding correlation between cipher text and plaintext bigram which is created due to the encryption rule.

4.5 Simulation result

Besides the above analytical results, we have also evaluated the proposed algorithm using security performance metrics listed in the previous sub section. In this section, the performance of the proposed work is depicted based on the parameters as follows briefly.

4.5.1 Avalanche Effect

The proposed algorithm performance was tested using Avalanche effect result. As we stated in sub section 4.3.1, it should provide a significant change on the output when a slight change is made in the input plaintext. The following figure shown us the result of the proposed algorithm with the same key and small change on the plaintext.

Plaintext: **Cryptography!** Secret key: **/Cä*ŠS;ćFÓtzE9>** Cipher text: **²Ù°<vdagK21u¹¾**		
Plaintext	**Cipher text**	**AE**
kryptography!	ŠÜ¼AxfciMP3v¦®	**100**
Cryptography?	¹Â»<jgZ5N5Pw!¬	**92.86**
cryptography!	éä±.p74aGWVn}F	**100**
Cryptographi!	±Öµ(ub7eI0Zbµ¼	**100**
Criptography!	±Ö~(ub7eI0Zrµ¼	**100**
	Average	**98.572**

Figure 4.7: Proposed algorithm avalanche effect result

Figure 4.7 shown an average AE ration of 98.572% if there is a minimal change in the plaintext message **Cryptography!** With the same key. This shown that within a small change in the plaintext, the proposed algorithm provide a significant change in the output cipher text. The proposed algorithm also achieved a SAC property since it has a greater change within a slight change in the input.

This proposed work was then compared to the paper in [10] and original playfair algorithm to show that which algorithm has a better performance on the avalanche effect. In order to gain a more accurate evaluation, the same length of data was changed in the input plaintext.

Plaintext: playfair cipher **Secret key of SEPCMBBSKV**: AfCN\4G>£šé«2„ **Secret key ExtePA**: 7YAJzfb **Secret key OPA**:DFINVLJ						
Plaintext	**Cipher of OPA**	**Cipher of ExtePA**	**Cipher of SEPCMBBSKV**	**AE OPA**	**AE ExtePA**	**Proposed AE**
flayfair cipher	DACWAHBXB NQGBT	BiW3BXXi4kg rgds0	Ùív0‹¼?9L‰ïÒ¦<£-	14.29	12.5	100
flaifair cipher	DABFAHBXB NQGBT	BiXlBXXi4kgr gds0	ÙíxM‹¼?9L‰ïÒ¦<£-	28.57	25	100
flaifair cifer	DABFAHBXB NVAXI	BiXlBXXi4kX0 io	ÈÙ7f§¬u0Ò*Ü;-£	57.14	57.143	100
flaifair sifer	DABFAHBXR NVAXI	BiXlBXXi2kX0 io	ÈÜoN¸®w3D"Ä²ª¦	64.29	64.29	100
flaipair sifer	DABFQLBXR NVAXI	BiXltVXi2kX0i o	ÀÖkô^©sY¶ŒÙ2£<	78.57	78.57	100

Figure 4.8: Avalanche effect comparison

Figure 4.7 shown that, the proposed algorithm is better in avalanche effect than the base paper and original playfair algorithm. In the proposed work, when a single character is changed in the input plaintext, all or a lot of characters are changed in the resulting cipher text. In the above figure, when we encrypte the given plaintext, the proposed algorithm provides an AE of **100%** even a single character is changed. As a result, the proposed work has a high avalanche effect ratio as compared to extended playfair algorithm using Fibonacci series and original playfair algorithm.

As the following graph shows, the proposed algorithm has high avalanche effect result when slight change is made in the plaintext. We have shown the above figure 4.8 AE result using graphs in the below figure 4.9.

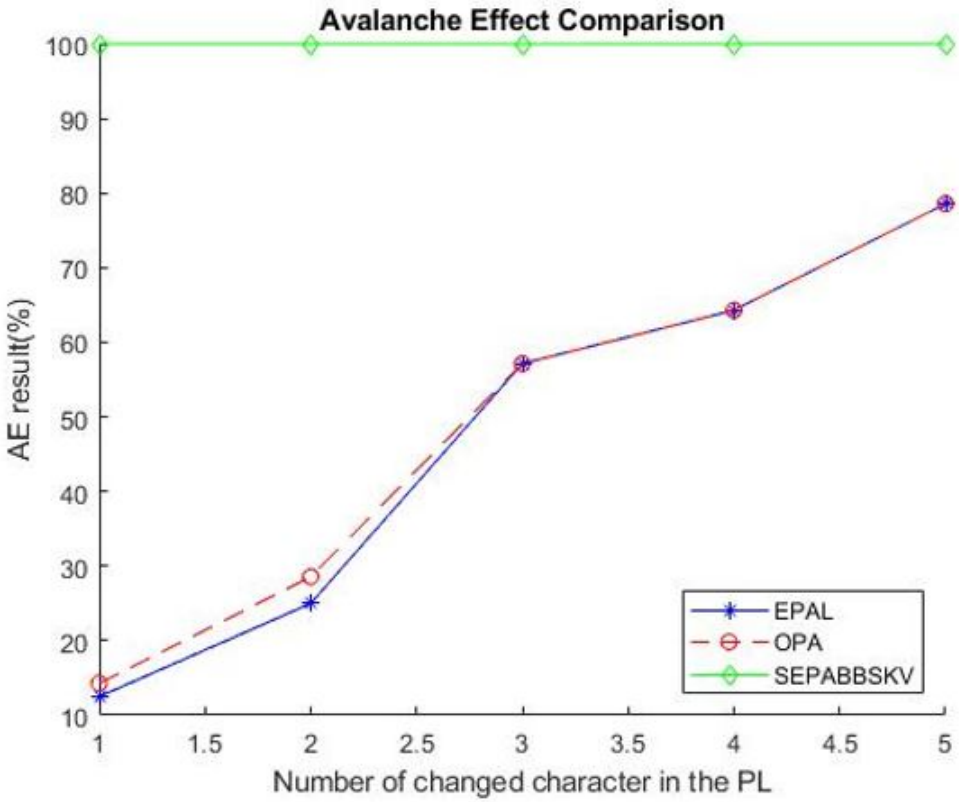


Figure 4.9: Avalanche Effect comparison graph

4.5.2 Confusion and Diffusion performance

The confusion and diffusion performance metrics were the most useful parameters which determines the proposed algorithm robustness. As we discussed before, diffusion is hiding the relationship between plaintext and cipher text and the influence of changing one character in the plaintext enforces the resulting cipher text to change in a high degree. The proposed algorithm achieved this diffusion properties as shown in the figure 4.7 AE result of the proposed algorithm. Figure 4.7 shown that a single character change in the PL enforced the cipher to change with an average of 98.572% in the resulting cipher. This indicates that the cipher text is highly dependent on the plaintext character and the proposed work has shown a high diffusion ratio. Thus, the proposed work hides the relationship between plaintext and cipher text. In the proposed algorithm, the resulting cipher text and its plaintext did not show any correlation.

Besides diffusion, the proposed algorithm has also achieved confusion properties by hiding the relationship between key and cipher text. In figure 4.9, we have shown that the influence of one character change in the key. The proposed algorithm enforced the cipher text to changes in a high degree when one character is changed in the key as compared to the paper in [10] and original playfair algorithm. Based on this fact, the proposed algorithm has high confusion rate. In the proposed algorithm and the paper in [10], when we change a key character we did not change the character itself rather we changed the number which is used to generate key. In both cases, the key is not accepted from the user rather the algorithm accepts number and generate key characters.

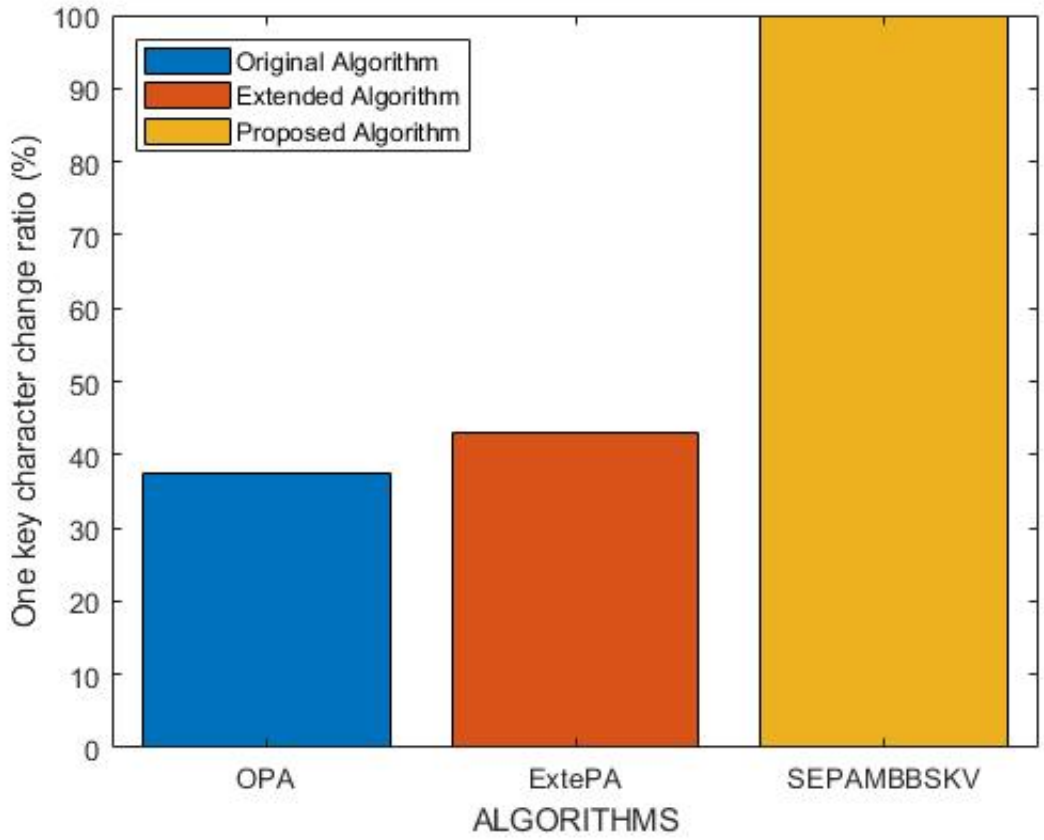


Figure 4.10: One key character change ratio

4.5.3 Brute Force Attack

The table displayed the estimation time to successfully crack the key by performing brute force attack. In this experiment, we used a 10 characters for the length of the plaintext , 15 key length characters for the proposed work and 7 key length characters for the existing work. This proposed experiments is done in Windows 10 (64bit), Intel(R) Celeron(R) CPU N3060 with 1.60GHz and 2 Core(s), and 4GB RAM. Five different tests were done for the specified length of plaintext to take the encryption time and obtained thier average. So, using this parameters, we made a calculation for the estimation time and it looks as follows:

Table 4.1: Estimation Time to Brute force attack

Unit	**Extended Algorithm**	**Proposed Algorithm**
Minutes	3.9839 * 10^11	9.5075 e + 32
Hours	6.639 * 10^9	15.846 * 10^30
Days	2.77 * 10^8	66.025 * 10^28
Years	7.58 * 10^5	18.089 * 10^26
Decades	7.58 * 10^4	18.089 * 10^25
Centuries	7.58 * 10^3	18.089 * 10^24

As the table shown, the proposed algorithm require a lot of time for brute force attack than the extended algorithm. Besides estimation time, the number of bigram permutations in the proposed work is also very large as we are using 196 characters which will take more time to apply brute force attack on each pair of bigrams. The extended playfair algorithm number of bigram permutations is less than that of our proposed work since it only uses 64 characters. The below figure shows the number of permutations of the proposed algorithm and algorithms where our works are compared with.

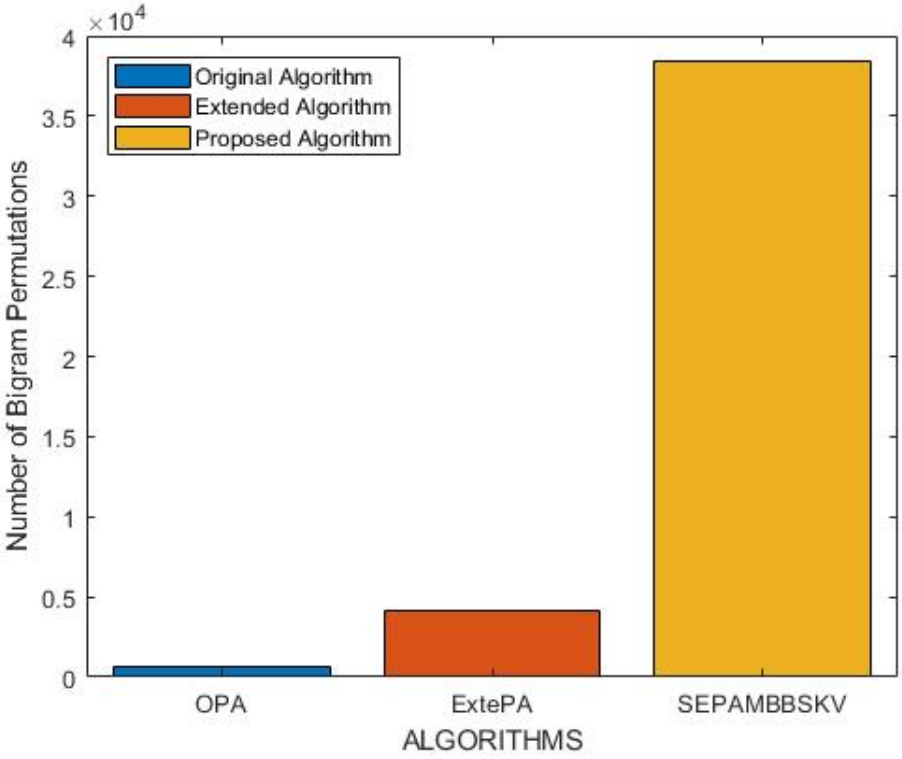


Figure 4.11: Number of Bigram Permutation

4.5.4 Number of character supported

The proposed algorithm supported a lot characters including uppercase letters, lowercase letters, digits, and 134 special characters. It is possible to encrypt alphanumeric characters and special symbols in this work. Figure 4.12 shows the number of characters supported by the three algorithms. So, from the figure, it is possible to say that our proposed work supports a lot character than the paper in [10] and original playfair algorithm.

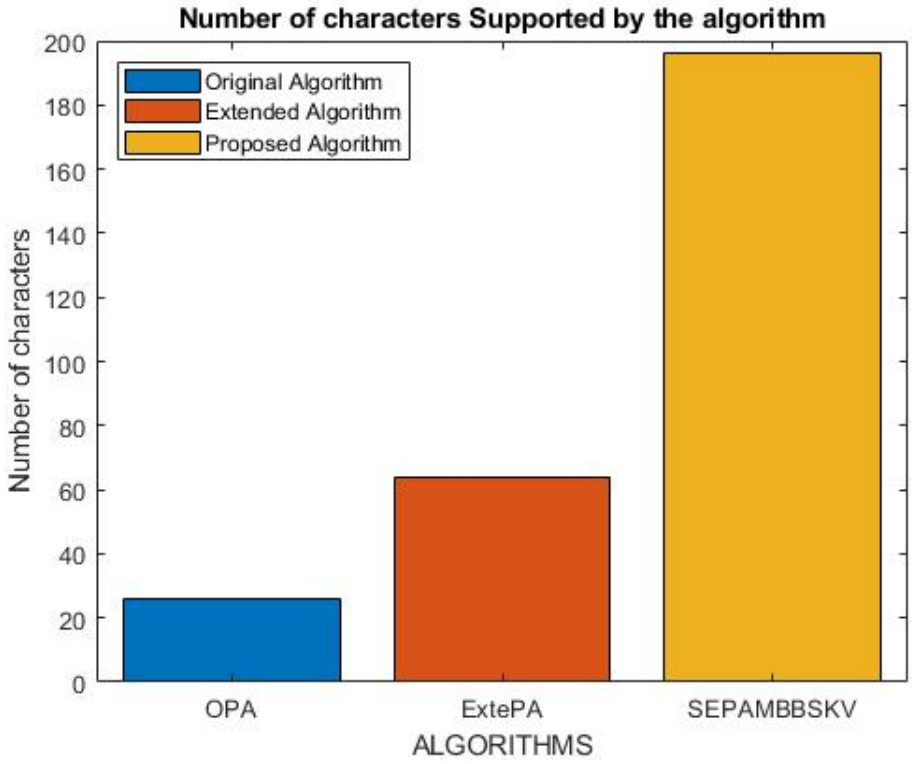


Figure 4.12: Number of characters supported

4.5.5 Frequency analysis attack

It is the analysis of the appearances of the number of characters or group of characters in the cipher text. It is a cipher text only attack. Increasing the key size reduces the probability of interrupting the cipher by frequency analysis attack [49]. So, the proposed algorithm has less probability of interrupting by frequency analysis attack since it supports a lot of character likewise the key size is also large. The chance of occurrence of a part within the proposed algorithm is,1/(196)=0.0051 , which is less when compared to the paper in [10] and original playfair algorithm. The following figure shown the probability of occurrence one character in the cipher text in the three algorithms. As a result, the proposed algorithm has less probability and became durable for cryptanalysis attack.

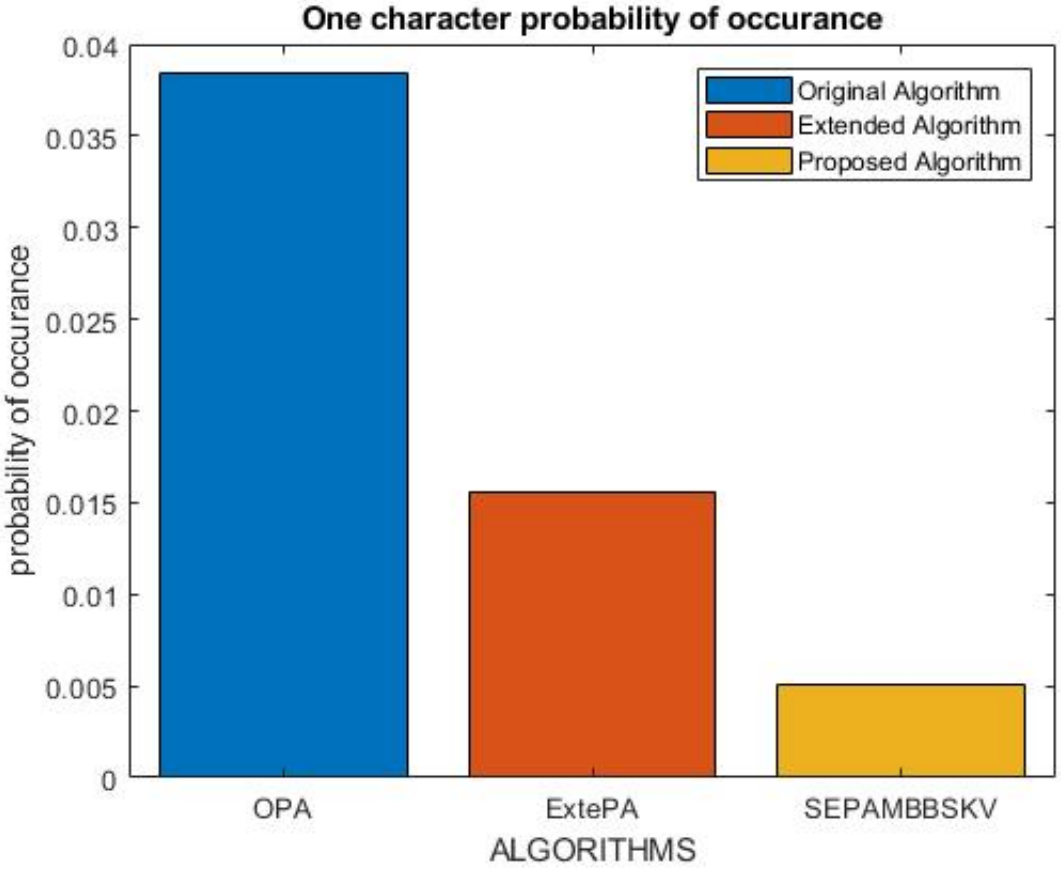


Figure 4.13: One character probability of occurrence

4.6 Analysis of the proposed work

In the proposed work, we have improved the security of playfair cipher using modified BBS and key stream values. The performance of the proposed work were evaluated using Avalanche effect, confusion and diffusion, number of character supported, filler character, etc. and it showed a better performance than the compared algorithm.

In our proposed work, we did not use filler character to separate a bigram with the same characters pair and we used space to make the length of character even, when the length of character became odd. In the proposed work 196 characters or symbols are incorporated as a proponent for a 14x14 matrix. In cryptography, Avalanche effect is a desirable property of cryptographic algorithm which determines the robustness of the algorithm. The proposed algorithm provided an average of 98.572% within the change of one character in the plaintext. It increases by an average of 81.905% with the avalanche effect result of the base paper. This shown that the security of the proposed work is strong and cannot easily attacked by cryptanalysis attack.

Chapter Five

5 Conclusion and Future Work

5.1 Conclusion

In this thesis work, a mechanism to improve the security of playfair algorithm is proposed. This algorithms aims to enhance security of playfair cipher using BBS random number generator and key stream values. In the proposed work, the limitations of the base paper like key generation and key exchange, need of filler character and showing correlation between plaintext and cipher text is solved. This work is evaluated by the security performance metrics and the result shown that the proposed algorithm is strong.

We have showed the result of the implementation based on two ways namely analytical result and simulation result. Analytical result described the result of the implementation of the proposed work in the case of overcoming the drawbacks we stated under the statement of the problem. So, this analytical result is evaluated based on the statement of problem limitation like filler character, pad character, key generation, key exchange, bigram and its reverse etc. In analytical result, we have stated the result in word and we did not show in graph, that's why we said it analytical. Whereas, simulation result is the implementation result of the proposed work and we have shown its result using graph.

Generally, based on the parameters result we used as a performance metrics to determine the security of the proposed work, we have shown that the proposed work is strong in security. We have achieved the goal of the work and successfully enhanced the security of playfair algorithm using modified BBS and key stream values. So, it can be safely conclude that the proposed algorithm is very strong than the extended playfair algorithm using Fibonacci series by the virtue of its key generation and key exchange, modified encryption mechanism, extended matrix, and solving filler character requirements.

5.2 Future Work

As a future work, we recommend someone to do research on the following points to make this algorithm more robust.

- Reduce the time taken for encryption and decryption, since we did not consider time as our concern was on security.
- Use appropriate compression technique to save space.
- Use the algorithm for the encryption and decryption of picture, sound, and video.
- Consider the proposed algorithm to integrate with system application.

References

[1] R.Deepthi", A Survey Paper on Playfair Cipher and its Variants", International Research Journal of Engineering and Technology (IRJET), Apr -2017.

[2] S.S Dhenakaran , M. llayaraja , "Extension of Playfair Cipher using 16X16 Matrix", International Journal of Computer Applications, June 2012.

[3] N. Sharma, H. Meghwal, M. Mehta and T. Kumar, "A Review on Playfair Substitution Cipher and Frequency Analysis Attack on Playfair," International Conference on Trends in Electronics and Informatics (ICOEI), 2018.

[4] A.Alam, S.Ullah, I.Wahid and et al, "Universal Playfair Cipher Using MXN Matrix", International Journal of Advanced Computer Science, Sep. 2011.

[5] J. F. Dooley, "History of cryptography and cryptanalysis: Codes, Ciphers, and their algorithms. ", Springer, 2018.

[6] M.Curtin, "Cracking the Data Encryption", Springer, 2007.

[7] A.Kumar, P.Singh Mehra, G.Gupta and et al, "Enhanced Block Playfair Cipher ", Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2013.

[8] M. Mizan Maha, Md. Masuduzzaman, and A.Bhowmik, "An Effective Modification of Play Fair Cipher with Performance Analysis using 6X6 Matrix" ,International Conference on Computing Advancements, 2020.

[9] A. chittala, T. Bhupathi, and D. Prasad Alakunta,"Random Number Generation Algorithms for Performance Testing", IEEE, 30 November 2021.

[10] M. Vasim Ahamad, MohdImran, and N. Siddiqui et al, "An Extended Playfair Encryption Technique Based on Fibonacci Series", Springer, 2020.

[11] F.A. Petitcolas, "Kirchhoff's' Principle", in Encyclopaedia of Cryptography and Security, Springer, 2011.

[12] Beutelspacher, Albrecht. "Cryptology. MAA", 1994.

[13] C. Paar and J.Pelzl, Understanding Cryptography,springer, 2010.

[14] M. Tuncay Gençoğlu, Importance of Cryptography in Information Security, IOSR Journal of Computer Engineering, Jan - Feb 2019.

[15] D. S. Abdul, H. M. Abdul Kader, and M. M Hadhoud. ”Performance evaluation of symmetric encryption algorithms.” IJCSNS International Journal of Computer Science and Network Security,2008.

[16] A. Kekunnaya , R. Gundla and S. Nanda, “A research Paper for Symmetric and Asymmetric Cryptography”, INTERNATIONAL JOURNAL OF RESEARCH IN ELECTRONICS AND COMPUTER ENGINEERING, JUNE 2019.

[17] F. Maqsood, M. Ahmed, M. M. Ali and et al, “Cryptography: A Comparative Analysis for Modern Techniques ”, International Journal of Advanced Computer Science and Applications, Vol. 8, 2017.

[18] K.S. and Meenakshi Mittal. ”Performance evaluation of various symmetric encryption algorithms.” 2014 international conference on parallel, distributed and grid computing. IEEE, 2014.

[19] N.Jirwan, A. Singh, Dr. Sandip Vijay, “Review and Analysis of CryptographyTechniques”, international Journal of Scientific & Engineering Research, 2013.

[20] G., A., and N.Kaur Walia. ”Cryptography Algorithms: a review.” (2014).

[21] S. Al Busafi and B. Kumar, “Review and Analysis of Cryptography Techniques”, IEEE Conference, 2020.

[22] R. K Sheth and Sarika P.Patel, “Analysis of Cryptography Techniques ”, International Journal of Research in advance Engineering, Volume -1 , Feb-2015.

[23] N.Tayal, R.Bansal, Shailender Gupta and et al, “Analysis of Various Cryptography Techniques: A Survey”, International Journal of Security and Its Applications, Vol. 10, 2016.

[24] M. U. Bokhari, Shadab Alam and Faheem Syeed Masoodi, “Cryptanalysis Techniques for Stream Cipher: A Survey ”, International Journal of Computer Applications, Volume 60, December 2012.

[25] A.K. Kendhe and H. Agrawal, “A Survey Report on Various Cryptanalysis Techniques ”, International Journal of Soft Computing and Engineering (IJSCE), Volume-3, May 2013.

[26] B. A. Forouzan, “Cryptography and Network Security”, Special Indian Edition, The McGraw- Hill companies, New Delhi, 2007.

[27] Y.Dodis, “Randomness and Cryptography”, Springer, 2019.

[28] M., Zhang, et al. ,”Overview of Randomness Test on Cryptographic Algorithms.” ,Journal of Physics: Conference Series, 2021.

[29] A. Khalique, A. Hamid Lone and Syed Shahabuddin Ashraf, “A Novel Unpredictable Temporal based Pseudo Random Number Generator”, International Journal of Computer Applications, 2015.

[30] R.N Sari and R.S. Hayati, “Beaufort Cipher Algorithm Analysis Based on the Power Lock-Blum Blum Shub in Securing Data”, International Conference on Cyber and IT Service Management, 2018.

[31] E.Elahi, H.Raza, and S Ali, “A New 3D playfair based secure cipher generation model”, International Conference on Emerging technologies, 2018.

[32] M.Syahriza,Murdani,S.D. Nasution et al, “Modified Playfair Cipher Using Random Key Linear Congruent Method”, Innovation for Research,Science,Technology and Culture (IRSTC),2017.

[33] S.Singh,”A Novel Technique for Enhancement of the Security of Playfair Cipher”, International Journal of Computer Engineering in Research Trends, Volume-7, 2020.

[34] M., M. Richard, and A.M. Sison. ”An enhanced key security of playfair cipher algorithm.” Proceedings of the 2019 8th International Conference on Software and Computer Applications, 2019.

[35] Pal, Pathikrit, et al., ”A variation in the working of playfair cipher.”, International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), Vol. 4,IEEE, 2019.

[36] A., Muhammad, and A.Mujahidah. ”Expending Technique Cryptography for Plaintext Messages by Modifying Playfair Cipher Algorithm with Matrix 5 x 19”, International Conference on Electrical Engineering and Computer Science (ICECOS) IEEE, 2019.

[37] Md.Ahnaf Tahmid Shakil and Md.Rabiul Islam, “An Efficient Modification to Playfair Cipher”, ULAB Journal of Science and Engineering, Vol. 5, November 2014.

[38] J.Ceasar C.Ferrer, Froilan E.De Guzeman and Kaye Louize E.Gardon et al, “Extended 10x10 playfair cipher”, IEEE, 2018.

[39] S.Ritchell villafuerte, A. M.sison and Ruji P. Medina ,” i3D-Playfair: An Improved 3D Playfair Cipher Algorithm ”, IEEE, 2019.

[40] P.Goyal, G.Sharma and S.S. Kushwah, “Implementation of Modified Playfair CBC Algorithm ”, International Journal of Engineering Research & Technology (IJERT), Vol. 5 Issue 06, June-2016.

[41] A.C. Licayan, B.D. Gerardo, and A.A.Hernandez, “ Performance Analaysis of Playfair Cipher Color Substitution Variants”, IEEE, august-2020.

[42] Mathur, S. K., & Srivastava, S, “Extended 16x16 Play-Fair Algorithm for Secure Key Exchange Using RSA Algorithm”, International Journal on Future Revolution in Computer Science & Communication Engineering, 2018.

[43] B.Aïssa and et al, “Implementation of Blum Blum Shub Generator for Message Encryption”, International Conference on Control, Engineering & Information Technology, 2014.

[44] https://resources.infosecinstitute.com/topic/stream-ciphers.

[45] S.Shakti Srivastava & N.Gupta, “Security aspects of the Extended Playfair cipher”, “IEEE”, 2011.

[46] K.Bhat, D.Mahto and D. K. Yadav, “Information Security using Adaptive Multidimensional Playfair Cipher” , International Journal of Advanced Research in Computer Science, June 2017.

[47] J. B.,Alimpia, Sison, A. M., & Medina, R. P., ” An enhanced hash-based message authentication code using BCrypt”, In Proceedings of the International Journal for Research in Applied Science and Engineering Technology, 2018.

[48] http://ides.ethz.ch.

Appendix

Appendix A: key processing and getting final key

```
% this code is to find the equivalent character for the pseudo Random sequence
for i=1:le
        key=ARR(num)
end
        key
       mm=length(key);
% this is to remove repeated key character
for i=1:mm
        for j=i+1:mm
            if(key(i)==key(j))
                key(j)=0;
           end
        end
end
% after replacing repeated character by 0, the key become
       key
% this is to replace removed repeated character by other character
for i=1:mm
        if(key(i)==0)
            Inschar=ARR(i+64)
                for j=1:mm
                   if(Inschar==key(j))
                       Inschar=ARR(j+2)
                    end
               end
                   key(i)=Inschar;
        end
end
% after replacing 0's with another character, the final key become
```

```
        disp('The final key is:');
        key
```

Appendix B: key matrix generation

```
% this is to generate key matrix using the key
         mat=zeros(14,14);
        num=[];
for j=1:196
        alphabets(j)=ARR(j);
end
for q=1:mm
         num=find(ARR==(key(q)));
             for h=1:196
                 ch=find(ARR==(alphabets(h)));
                     if (num==ch)
                         alphabets(h)=0;
                    end
            end
end
         q=1;
for m=1:196
         ff=find(ARR==(alphabets(m)));
            if(ff =0)
                 new_alpha(q)=alphabets(m);
                q=q+1;
             end
end
         len=length(new˙alpha);
        f=196;
        words˙comb=zeros(1,f);
```

```
    for m=1:f
        if(mm¿=m)
           words_comb(m)=find(ARR==(key(m)));
        elseif(m¿mm)
            for h=1:len
               words_comb(h+mm)=find(ARR==(new_alpha(h)));
           end
        end
end
for i=1:f
        ff=ARR(words_comb(i));
        mat_words(i)=ff;
end
       len_mat=length(mat_words);
       h=1;

    for i=1:14
        for j=1:14
            try
               mat(i,j)=mat_words(h);
                h=h+1;
           catch(h¡196)
                break
            end
        end
end
for i=1:14
       for j=1:14
            b=char(mat(i,j));
            words(i,j)=b;
        end
```

```
end
        words
```

Appendix C: Modified encryption rule of playfair algorithm

```
        coll=[];
        roww=[];
for i=1:le_equ

            [row,col] = find(cellfun(@(x) isequal(equiv_tex(i),x),words));


             coll(i)=col
         roww(i)=row
end
        ciph=[];
         D1=[];
        Final_cipher=[];
         j=1;
for i=1:le_equ-1
         if(j¡le_equ)
            if(roww(j)==roww(j+1))
                if(roww(j)==14)
                    roww(j)=1;
                elseif(roww(j) =14)
                    roww(j)=roww(j)+1;
                end
                if(roww(j+1)==14)
                    roww(j+1)=1;
                elseif(roww(j+1) =14)
                    roww(j+1)=roww(j+1)+1;
                end
                   ciph=char(words(roww(j),coll(j)))
                   D1(j)=find(ARR==ciph);
```

```
        Final_cipher=ARR(D1)
        ciphr=char(words(roww(j+1),coll(j+1)))
        D1(j+1)=find(ARR==ciphr);
        Final_cipher=ARR(D1);
 end
 if(coll(j)==coll(j+1))
     if(coll(j)==1)
         coll(j)=14;
    elseif(coll(j) =1)
         coll(j)=coll(j)-1;
    end
     if(coll(j+1)==1)
         coll(j+1)=14;
    elseif(coll(j+1) =1)
        coll(j+1)=coll(j+1)-1;
    end
         ciph=char(words(roww(j),coll(j)));
        D1(j)=find(ARR==ciph);
        Final_cipher=ARR(D1);
        ciphr=char(words(roww(j+1),coll(j+1)));
         D1(j+1)=find(ARR==ciphr);
        Final_cipher=ARR(D1);
end
if(roww(j) =roww(j+1) && coll(j) =coll(j+1))
     if(coll(j)==14)
         coll(j)=1;
    elseif(coll(j) =14)
         coll(j)=coll(j)+1
     end
     if(coll(j+1)==14)
        coll(j+1)=1
```

```
            elseif(coll(j+1) =14)
                coll(j+1)=coll(j+1)+1
            end
                ciph=char(words(roww(j),coll(j)))
                 D1(j)=find(ARR==ciph);
                 Final_cipher=ARR(D1)
                ciphr=char(words(roww(j+1),coll(j+1)))
                D1(j+1)=find(ARR==ciphr);
                Final_cipher=ARR(D1)
        end
    end
        j=j+2;
end
    Final_cipher
```

Appendix D: processing of plaintext

```
pp=input('enter the plaintext to be encrypted: ','S')
    Len_plaintext=length(pp)
    y=1;
% the following code is to check whether the entered character is within in % array or not
while y¡=Len_plaintext
    if(ismember(pp(y),ARR))
        y=y+1;
    else
        disp(['unknown Character :', pp(y)]);
        pp=input('Enter message withn given array : ','S');
        y=1;
        Len_plaintext=length(pp)
    end
end
% the below code shows the length of plaintext is even or odd. if it is odd
% it will add another bogus letter
```

```
        tt=rem(Len_plaintext,2);
if(tt =0)
        pp(Len˙plaintext+1)=' '
        Len_plaintext=length(pp)
end
```

Appendix E: positional value of each plaintext character

```
        ppp=[];
% the below code gives the positional value of the character
for i=1:Len_plaintext
        ppp(i)=i
end
```