# Alphabet Encoding

A Project Report submitted in partial fulfillment of the

requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING


Submitted by

Abhishek Joshi (Enroll. No. R100218003)

Devansh Raghuvanshi (Enroll. No. R970218018)

Garvit(Enroll. No. R610218015)

Vaibhav Gandhi(Enroll No. R100218070)

Under the guidance of

Mrs. Anushree Shah




**SCHOOL OF COMPUTER SCIENCE**

UNIVERSITY OF PETROLEUM & ENERGY STUDIES

Bidholi Campus, Energy Acres, Dehradun – 248007.


**DECLARATION**

I hereby declare that this submission is my own and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other Degree or Diploma of the University or other Institute of Higher learning, except where due acknowledgement has been made in the text.

- Abhishek Joshi (Enroll. No. R100218003 )
- Devansh Raghuvanshi (Enroll. No. R970218018)
- Garvit (Enroll. No. R610218015)
- Vaibhav Gandhi(Enroll No. R100218070)

# CERTIFICATE

This is to certify that the project titled **Alphabet Encoding** submitted by Abhishek Joshi (Enroll. No. R100218003), Devansh Raghuvanshi (Enroll. No. R970218018), Garvit (Enroll. No. R610218015), Vaibhav Gandhi (Enroll. No. R100218070) to the University of Petroleum & Energy Studies, for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING is a Bonafede record of project work carried out by them under my supervision and guidance. The content of the project, in full or parts have not been submitted to any other Institute or University for the award of any other degree or diploma.

Date: _____

Name of Guide

HOD Dept. of Systemics | SOCS

# ACKNOWLEDGEMENT

I owe my deep gratitude to our project guide, Mrs. Anushree Shah who took keen interest on our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all staff of Department of Systemics who helped us in successfully completing our project work.

# TABLE OF CONTENTS

# ABSTRACT

Currently, data security and privacy policy has been regarded as one of the biggest concerns in the era computing. In our project user will securely input data followed by the symmetric-key for the encryption and decryption of data. Cryptography enables the user toencrypt or decrypt the plaintextin a cipher text (or vice versa) to maintain the confidentiality of the data so that the data(plaintext) cannot be compromise. So, this scheme can make users assure about the security of data store

# CHAPTER 1

# INTRODUCTION

**Encryption and Decryption:**Cryptography is used to secure and protect data during communication. It is helpful to prevent unauthorized person or group of users from accessing any confidential data. Encryption and decryption are the two essential functionalities of cryptography. Encryption is a process which transforms the original information into an unrecognizable form. This new form of the message is entirely different from the original message. Decryption is a process of converting encoded/encrypted data in a form that is readable and understood by a human or a computer. This method is performed by un-encrypting the text manually or by using keys used to encrypt the original data.

Techniques: The techniques(algorithms) we are using are following:

- Playfair cipher
- Rail fence cipher
- Ceaser cipher

1. **Playfair cipher:** The Playfair cipher is a symmetric encryption and decryption technique. This technique was invented in 1854 by Charles Wheatstone but it carries the name of his friend Lord Playfair. This technique works upon the pairs of characters instead of single character. Hence it is harder to break, which is its one of its advantage also. It contains base structure of 5 × 5 matrix which consists of 25 cells, with which its uniqueness should be maintained. Although alphabets consist of 26 letters but one of the characters from alphabets will be adjusted in one cell so that all characters in alphabets will cover the cells. Its main disadvantage is that same key is used for both encryption and decryption technique. 4 rulesare used to encrypt the data. And to decrypt the data we use 3 rules to

implement this technique which are reverse of encryption. It is used to pass the secret information. Hence with pairing of the words we can convert plain text to cipher text.

2. **Rail fence cipher:** The rail fence cipher is also known as zig-zag. It is a transposition cipher that follows a simple rule for mixing up the characters in the plaintext to form the ciphertext. The rail fence cipher offers essentially no communication security. Although it is weak on its own, but it can be combined with other ciphers, such as a substitution cipher, the combination of which is more difficult to break than either cipher on its own.

   To encrypt a message using the Rail Fence Cipher, we have to write the message in zigzag lines across the page, and then read off each row. Firstly, we need to have a key, which for this cipher is the number of rows we are going to have. Then start writing the letters of the plaintext diagonally down to the right until we reach the number of rows specified by the key. Then bounce back up diagonally until we hit the first row again. This continues until the end of the plaintext.

   The decryption process for the Rail Fence Cipher involves reconstructing the diagonal grid used to encrypt the message. We start writing the message, but leaving a dash in place of the spaces yet to be occupied. Gradually, we can replace all the dashes with the corresponding letters, and read off the plaintext from the table.

3. **Caesar cipher:** The Caesar Cipher strategy is one of the oldest and least complex strategy for encryption method. It's essentially a sort of replacement figure, i.e., each letter of a given book is supplanted by a letter some fixed number of positions down the letters in order. For instance with a move of 1, A would be supplanted by B, B would become C, etc. The technique is obviously named after Julius Caesar, who clearly utilized it to speak with his authorities. In this way to encode a given book we need a number worth, known as move which demonstrates the quantity of position each letter of the content has been descended. The encryption can be spoken to utilizing secluded math by first changing the letters into numbers, as per the plan, A = 0, B = 1…, Z = 25.

# CHAPTER 2
# LITREATURE REVIEW

## 1. Encryption and Decryption:

Encryption is the method by which information is converted into secret code that hides the information's true meaning. The science of encrypting and decrypting information is called cryptography.

In computing, unencrypted data is also known as plaintext, and encrypted data is called ciphertext. The formulas used to encode and decode messages are called encryption algorithms, or ciphers.
To be effective, a cipher includes a variable as part of the algorithm. The variable, which is called a key, is what makes a cipher's output unique. When an encrypted message is intercepted by an unauthorized entity, the intruder has to guess which cipher the sender used to encrypt the message, as well as what keys were used as variables. The time and difficulty of guessing this information is what makes encryption such a valuable security tool.

SYMMETRIC: Symmetric ciphers, also referred to as secret key encryption, use a single key. The key is sometimes referred to as a shared secret because the sender or computing system doing the encryption must share the secret key with all entities authorized to decrypt the message. Symmetric key encryption is usually much faster than asymmetric encryption.

Symmetric key: Symmetric-key encryption are algorithms which use the same cryptographic keys for both encryption of plaintext and decryption of ciphertext.

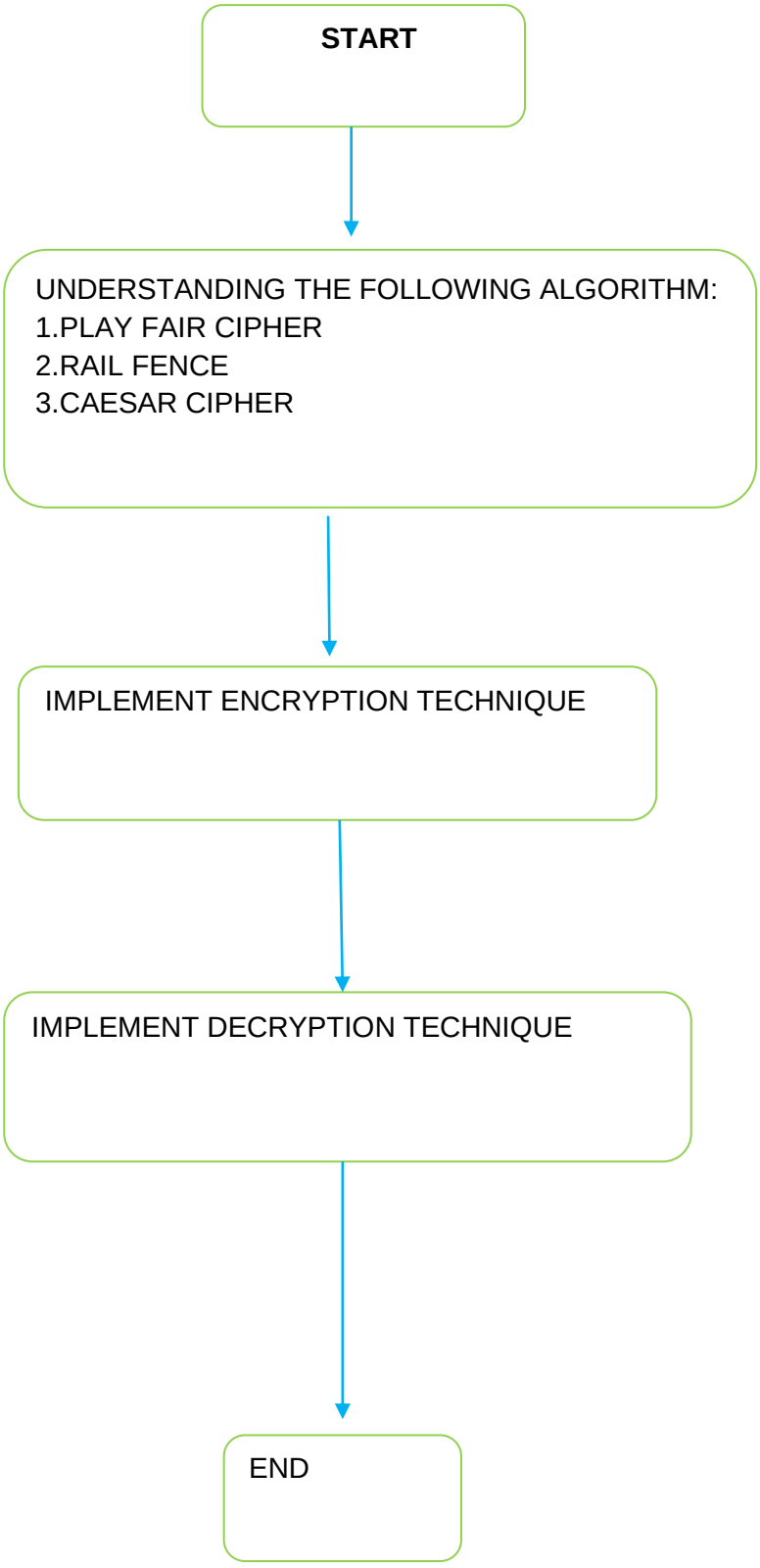
ASSYMETRIC: Asymmetric ciphers, also known as public key encryption, use two different, but logically linked keys. This type of cryptography often uses prime numbers to create keys since it is computationally difficult to factor large prime numbers and reverse-engineer the encryption. The Rivest Shamir Adleman (RSA) encryption algorithm is currently the most widely used public key algorithm. With RSA, the public or the private key can be used to encrypt a message.
Asymmetric key: Asymmetric encryption uses 2 pairs of key for encryption. Public key is available to anyone while the secret key is only made available to the receiver of the message. This boots security.

Reference book:

Cryptography and Network Security: Principles and Practice Book by William Stallings

Head First Java 2nd Edition by Kathy Sierra & Bert Bates

# Chapter-3
# Methodology



Through our minor project topic, we will implement theconcept of symmetric encryption and decryption. Symmetric encryption and decryptionare a technique for cryptography that use the same symmetric keyfor both encryption of plaintext and decryption of ciphertext. We implement this concept through the 4 algorithms which are as follows –

• Rail-fence cipher

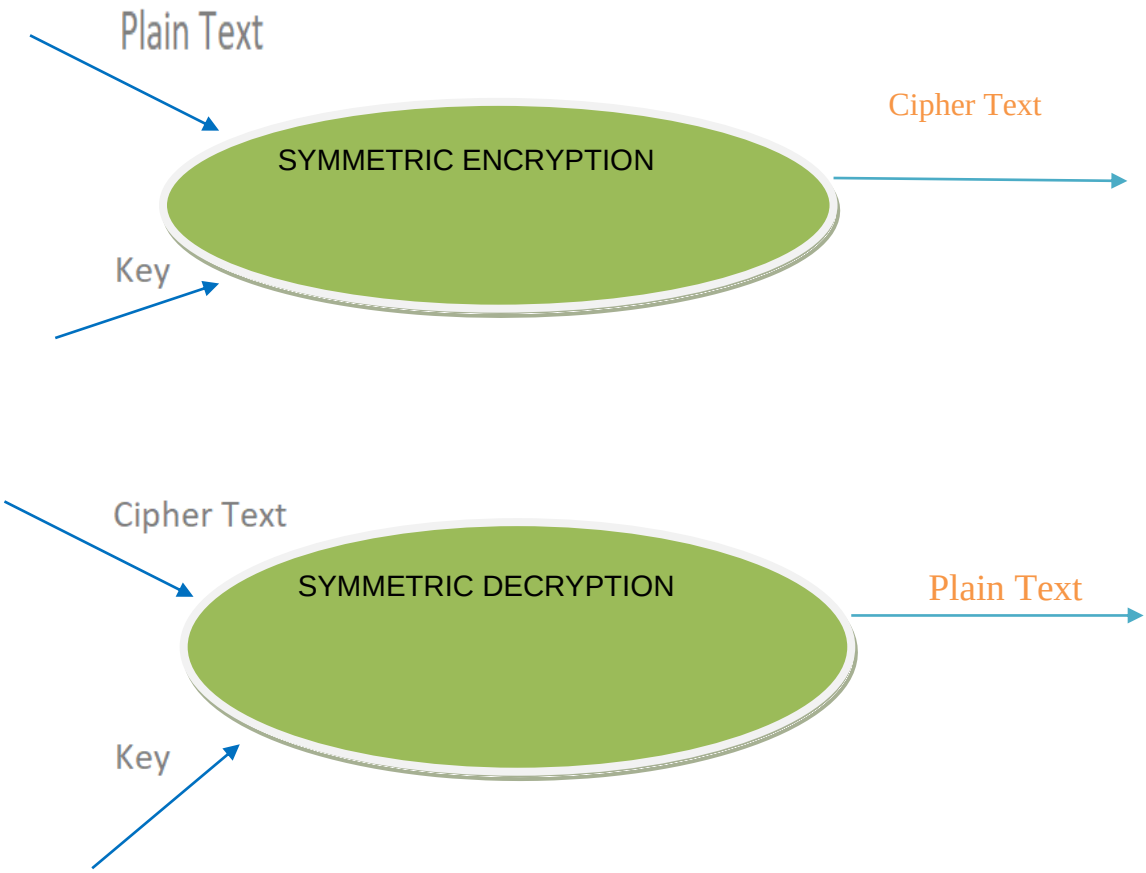
• Caesar cipher

• Play-fair cipher

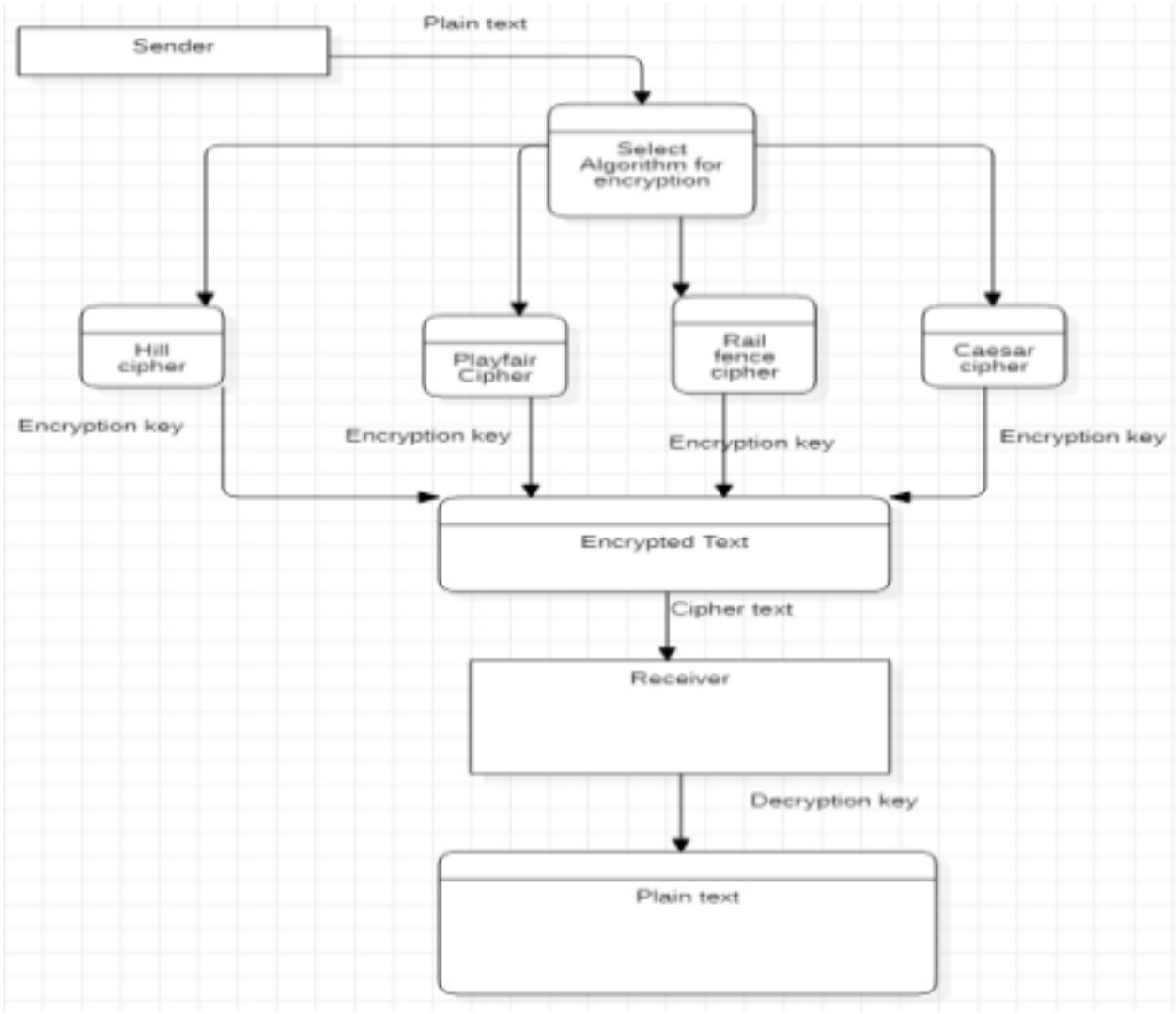
We implement the encryption and decryption by coding a menu driven program for all 4 algorithms and ask the user to enter the data to be encrypt following by a symmetric key, then our code will return the decrypted data to the user. Hence same for decryption also.

User will choose one algorithm from menu driven program and implement the encryption and decryption technique for the same using the Java language.

# LEVEL 0 DFD



# LEVEL 2 DFD


Sender
Plain text
Select Algorithm for encryption
Hill cipher
Playfair Cipher
Rail fence cipher
Caesar cipher
Encryption key
Encryption key
Encryption key
Encryption key
Encrypted Text
Cipher text
Receiver
Decryption key
Plain text

## CHAPTER 4

## SOURCE CODE

```
import java.util.Scanner;
public class ENC_DEC{
public static void main(String[] args){
Scanner sc = new Scanner(System.in);
System.out.println("enter the no from 1 to 3");
int n = sc.nextInt();
if(n==1){
 System.out.println("the algorithm for encoding and decoding is playfair cipher");
 PlayfairCipher.main(args);
}
else if(n==2){
 System.out.println("the algorithm for encoding and decoding is RailFence");
 RailFence.main(args);
}
else if(n==3){
System.out.println("the algorithm for encoding and decoding is caesarcipher");
 CaesarCipherJava.main(args);
}
else{
System.out.println("invalid error");
}
}
}

//Rail Fence
import java.util.*;
class RailFenceBasic{
 int depth;
 String Encryption(String plainText,int depth)
```

```
{
 int r=depth,len=plainText.length();
 int c=len/depth;
 char mat[][]=new char[r][c];
 int k=0;

 String cipherText="";

 for(int i=0;i< c;i++)
 {
  for(int j=0;j< r;j++)
  {
   if(k!=len)
    mat[j][i]=plainText.charAt(k++);
   else
    mat[j][i]='X';
  }
 }
 for(int i=0;i< r;i++)
 {
  for(int j=0;j< c;j++)
  {
   cipherText+=mat[i][j];
  }
 }
 return cipherText;
}

String Decryption(String cipherText,int
depth) {
```

```
  int
  r=depth,len=cipherText.length();
  int c=len/depth;
  char mat[][]=new char[r][c];
  int k=0;

  String plainText="";


  for(int i=0;i< r;i++)
  {
   for(int j=0;j< c;j++)
   {
    mat[i][j]=cipherText.charAt(k++);
   }
  }
  for(int i=0;i< c;i++)
  {
   for(int j=0;j< r;j++)
   {
    plainText+=mat[j][i];
   }
  }

  return plainText;
 }
}

class RailFence{
 public static void main(String[] args){
```

```
  RailFenceBasic rf=new RailFenceBasic();
            Scanner scn=new Scanner(System.in);
            int depth;

            String plainText,cipherText,decryptedText;

            System.out.println("Enter plain text:");
            plainText=scn.nextLine();

            System.out.println("Enter depth for Encryption:");
            depth=scn.nextInt();

  cipherText=rf.Encryption(plainText,depth);
      System.out.println("Encrypted text is:\n"+cipherText);

              decryptedText=rf.Decryption(cipherText, depth);

                          System.out.println("Decrypted text

                                      is:\n"+decryptedText);

 }
}


//Play Fair Cipher

import java.awt.Point;

import java.util.Scanner;
```

```
public class PlayfairCipher

{

//length of digraph array

private int length = 0;

//creates a matrix for Playfair cipher

private String [][] table;

//main() method to test Playfair method

public static void main(String args[])
{

PlayfairCipher pf = new PlayfairCipher();

}

//main run of the program, Playfair method

//constructor of the class

private PlayfairCipher()

{

//prompts user for the keyword to use for encoding & creates tables
```

```
System.out.print("Enter the key for playfair cipher: "); Scanner sc

= new Scanner(System.in);

String key = parseString(sc);

while(key.equals(""))

key = parseString(sc);

table = this.cipherTable(key);

//prompts user for message to be encoded

System.out.print("Enter the plaintext to be encipher: ");
//System.out.println("using the previously given keyword");

String input = parseString(sc);

while(input.equals(""))

input = parseString(sc);

//encodes and then decodes the encoded message

String output = cipher(input);

String decodedOutput = decode(output);

//output the results to user
```

```
this.keyTable(table);

this.printResults(output,decodedOutput);

}

//parses an input string to remove numbers, punctuation,

//replaces any J's with I's and makes string all caps

private String parseString(Scanner sc)

{

String parse = sc.nextLine();
//converts all the letters in upper case

parse = parse.toUpperCase();

//the string to be substituted by space for each match (A to Z)

parse = parse.replaceAll("[^A-Z]", "");

//replace the letter J by I

parse = parse.replace("J", "I");

return parse;

}
```

```
//creates the cipher table based on some input string (already parsed)

private String[][] cipherTable(String key)

{

//creates a matrix of 5*5

String[][] playfairTable = new String[5][5];

String keyString = key + "ABCDEFGHIKLMNOPQRSTUVWXYZ";

//fill string array with empty string

for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)

playfairTable[i][j] = "";

for(int k = 0; k < keyString.length(); k++)

{

boolean repeat = false;

boolean used = false;

for(int i = 0; i < 5; i++)

{

for(int j = 0; j < 5; j++)
```

```
{
if(playfairTable[i][j].equals("" + keyString.charAt(k))) {
repeat = true;
}
else if(playfairTable[i][j].equals("") && !repeat && !used) {
playfairTable[i][j] = "" + keyString.charAt(k);
used = true;
}
}
}
}
return playfairTable;
}
//cipher: takes input (all upper-case), encodes it, and returns the output
private String cipher(String in)
{
```

```
length = (int) in.length() / 2 + in.length() % 2;

//insert x between double-letter digraphs & redefines "length"



for(int i = 0; i < (length - 1); i++)

{
if(in.charAt(2 * i) == in.charAt(2 * i + 1))

{

in = new StringBuffer(in).insert(2 * i + 1, 'X').toString(); length = (int) in.length() / 2 +

in.length() % 2; }

}

//------------makes plaintext of even length---------- //creates an array of digraphs

String[] digraph = new String[length];

//loop iterates over the plaintext

for(int j = 0; j < length ; j++)

{

//checks the plaintext is of even length or not if(j == (length - 1) && in.length() / 2 ==

(length - 1)) //if not addends X at the end of the plaintext in = in + "X";
```

```
digraph[j] = in.charAt(2 * j) +""+ in.charAt(2 * j + 1);

}

//encodes the digraphs and returns the output

String out = "";

String[] encDigraphs = new String[length];

encDigraphs = encodeDigraph(digraph);

for(int k = 0; k < length; k++)

out = out + encDigraphs[k];

return out;

}

//----------encryption logic----------

//encodes the digraph input with the cipher's specifications

private String[] encodeDigraph(String di[])

{

String[] encipher = new String[length];
```

```
for(int i = 0; i < length; i++)
{

char a = di[i].charAt(0);

char b = di[i].charAt(1);

int r1 = (int) getPoint(a).getX();

int r2 = (int) getPoint(b).getX();

int c1 = (int) getPoint(a).getY();

int c2 = (int) getPoint(b).getY();

//executes if the letters of digraph appear in the same row //in such case shift columns to

right

if(r1 == r2)

{

c1 = (c1 + 1) % 5;

c2 = (c2 + 1) % 5;

}

//executes if the letters of digraph appear in the same column //in such case shift rows down

else if(c1 == c2)
```

```
{

r1 = (r1 + 1) % 5;

r2 = (r2 + 1) % 5;

}

//executes if the letters of digraph appear in the different row and different column

//in such case swap the first column with the second column

else

{

int temp = c1;

c1 = c2;

c2 = temp;

}

//performs the table look-up and puts those values into the encoded array

encipher[i] = table[r1][c1] + "" + table[r2][c2];

}
return encipher;
```

```
}

//----------------------decryption logic---------------------

// decodes the output given from the cipher and decode methods (opp. of encoding process)

private String decode(String out)

{

String decoded = "";

for(int i = 0; i < out.length() / 2; i++)

{

char a = out.charAt(2*i);

char b = out.charAt(2*i+1);

int r1 = (int) getPoint(a).getX();

int r2 = (int) getPoint(b).getX();

int c1 = (int) getPoint(a).getY();

int c2 = (int) getPoint(b).getY();

if(r1 == r2)
{
```

```
c1 = (c1 + 4) % 5;

c2 = (c2 + 4) % 5;

}

else if(c1 == c2)

{

r1 = (r1 + 4) % 5;

r2 = (r2 + 4) % 5;

}

else

{

//swapping logic

int temp = c1;

c1 = c2;

c2 = temp;

}
decoded = decoded + table[r1][c1] + table[r2][c2];
```

```
}

//returns the decoded message

return decoded;

}

// returns a point containing the row and column of the letter

private Point getPoint(char c)

{

Point pt = new Point(0,0);

for(int i = 0; i < 5; i++)

for(int j = 0; j < 5; j++)

if(c == table[i][j].charAt(0))

pt = new Point(i,j);

return pt;

}

//function prints the key-table in matrix form for playfair cipher
private void keyTable(String[][]
```

```
printTable) {

System.out.println("Playfair Cipher Key Matrix: ");

System.out.println();

//loop iterates for rows

for(int i = 0; i < 5; i++)

{

//loop iterates for column

for(int j = 0; j < 5; j++)

{

//prints the key-table in matrix form

System.out.print(printTable[i][j]+" ");

}

System.out.println();

}

System.out.println();
}
```

```
//method that prints all the results

private void printResults(String encipher, String

dec) {

System.out.print("Encrypted Message: ");

//prints the encrypted message

System.out.println(encipher);

System.out.println();

System.out.print("Decrypted Message: ");

//prints the decryted message

System.out.println(dec);

}

}
```

**OUTPUT**

```
D:\javaprograms>java ENC_DEC
enter the no from 1 to 3
1
the algorithm for encoding and decoding is playfair cipher
Enter the key for playfair cipher: Diamond
Enter the plaintext to be encipher: Abhishek
Playfair Cipher Key Matrix:

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

Encrypted Message: ICRBRKCL

Decrypted Message: ABHISHEK

D:\javaprograms>java ENC_DEC
enter the no from 1 to 3
2
the algorithm for encoding and decoding is Railfence
Enter plain text:
Hi, How are you?
Enter depth for Encryption:
3
Encrypted text is:
H wryiH eo,oa u
Decrypted text is:
Hi, How are you

D:\javaprograms>java ENC_DEC
enter the no from 1 to 3
3
the algorithm for encoding and decoding is caesarcipher
Enter a message:
welcome to upes
Enter key:
3
Encrypted Message - zhofrph wr xshv
Decrypted Message - Welcome to upes
```

Problems @ Javadoc Declaration Console

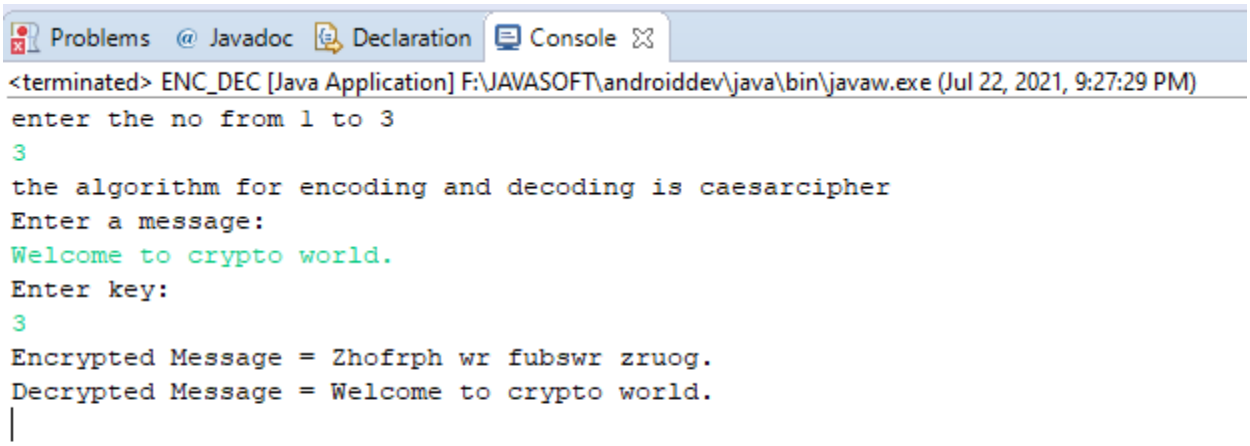
<terminated> ENC_DEC [Java Application] F:\JAVASOFT\androiddev\java\bin\javaw.exe (Jul 22, 2021, 6:39:15 PM)

```
enter the no from 1 to 3
1
the algorithm for encoding and decoding is playfair cipher
Enter the key for playfair cipher: Diamond
Enter the plaintext to be encipher: Abhishek
Playfair Cipher Key Matrix:

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

Encrypted Message: ICRBRKCL

Decrypted Message: ABHISHEK
```

Problems @ Javadoc Declaration Console

<terminated> ENC_DEC [Java Application] F:\JAVASOFT\androiddev\java\bin\javaw.exe (Jul 22, 2021, 6:45:53 PM)

```
enter the no from 1 to 3
2
the algorithm for encoding and decoding is RailFence
Enter plain text:
Hi How are you?
Enter depth for Encryption:
3
Encrypted text is:
HH eoioa u wry?
Decrypted text is:
Hi How are you?
```




## COMPARISION BETWEEN PLAY FAIR,RAIL FENCE,HILL CIPHER

| S.NO. | PLAY FAIR | RAIL FENCE | CAESAR CIPHER |
|---|---|---|---|
| 1. | It is significantly harder to break since the frequency analysis technique used to break simple substitution ciphers is difficult but still can be used on (25*25) = 625 digraphs rather than 25 monographs which is difficult. | The whole message is not to be encrypted at once. If we use simple rail fence over a block of characters then it is very easy to find out the number of rails used for encryption | Use of only a short key in the entire process |
| 2. | Frequency analysis thus requires more cipher text to crack the encryption. | In order to guess the key for 'n' characters block, the cipher breaker will have to just try out 'n' rails to find out the key whereas in the block rail fence cipher the third party has to go through approximately 'nm' numbers where 'm' is the size of the key. | Requires few computing resources |

## PERFORMACE OR EXECUTION TIME OF ALGORITHMS

**CAESAR CIPHER < RAIL FENCE CIPHER < PLAY FAIR CIPHER**

**Execution Time of Play fair is more .Hence it is harder to break.**

## REFERENCES


1. Head First Java 2nd Edition by Kathy Sierra & Bert Bates

2. Cryptography and Network Security: Principles and Practice Book by William Stallings