A Report on

Cryptoware: Polyalphabetic cipher

(Mini-Project of CSE1007-Introduction to Cryptography)

Submitted by

Manikanta Bhuvanesh Valiveti

(Registration-19BCD7088)

on

15th November 2020

School of Computer Science and Engineering

VIT-AP University, Andhra Pradesh

Abstract

Cryptography is considered to be a disciple of science of achieving security by converting sensitive information to an un-interpretable form such that it cannot be interpreted by anyone except the transmitter and intended recipient. An innumerable set of cryptographic schemes persist in which each of it has its own affirmative and feeble characteristics. In this paper we present a perspective on Polyalphabetic Cipher Techniques which are currently used for encryption and decryption purpose. This paper mainly focuses on practically use of the polyalphabetic Cipher Techniques for encryption and decryption purpose. A brief discussion about types of polyalphabetic cipher techniques.

Keywords: Polyalphabetic, Encryption, Vigenere cipher, Vernam cipher, One—time pad,Autokey,playfair,hill cipher,rotor, Decryption.

1 Introduction

Cryptography is an art and science.It plays major role in information and security division.The main aim of the cryptography is protecting the data from unauthorized users or hackers.Cryptography is subject contains two parts one is encryption and decryption.Encryption is a process converting the plain text to cipher text using some keys. Decryption is a process of converting the cipher text to plain text using the keys.Encryption is an effective way to achieve the security of data.The word of encryption came in mind of King Julius Caesar because he did not believe on his messenger so he thought to encrypt the data or message by replacing every alphabet of data by 3rd next alphabet. The process of Encryption hides the data in a way that an attacker cannot hack the data. The main purpose of encryption is to hide the data from unauthorized parties from viewing, altering the data. Encryption techniques occur or used by using the shifting techniques, mathematical operations and shifting techniques. The Simple data is known as Plain text and Data after encryption is known as Cipher text. Substitution and transposition techniques are mainly used for it. In encryption methods, two methods are used for encryption purpose

a. Substitution techniques-Change the one letter by another using secret key.

b. Transposition techniques-Replace the place of letters of plaintext.

.

In substitution techniques monoalphabetic and polyalphabetic techniques are used. In monoalphabetic, a single cipher alphabet is used per message. This techniques was easy to break because they show the frequency data of plaintext alphabet. So polyalphabetic techniques came into knowledge in which different monoalphabetic substitution as one proceeds through original message.

1.1 About

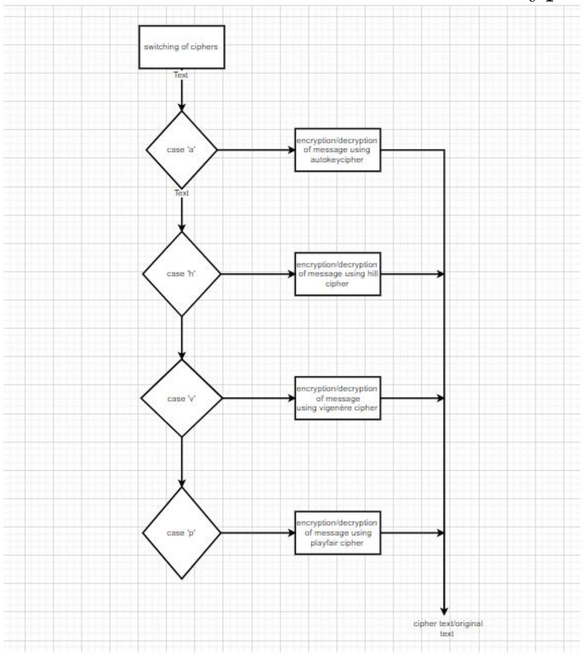
In this project I am going to do design cryptographic tool for polyalphabetic ciphers using Alice-Bob models.For Alice I am going to write an encryption driven program in which alice can choose any one of polyalphabetic ciphers(Autokey,Playfair,Vigenere,Hill) of their choice for encrypting message which has to be send to the bob using the key which is already predetermined by both I.e, Bob and Alice.And for Bob I am going to write decrytion driven program in which bob has to decyrpt cipher text using predetermined key.

1.2 Implementation Environment

For this cryptographic tool I used java program,For compiling program i used Bluej IdE.Input should be small letter alphabets,No special charaters and without spaces in message,For auto key cipher key input should be single character alphabet.For hill cipher message length should be perfect square and key should be square root message length and invertible.For vigenere cipher no specifications.For playfair cipher message length should be even.

2 Procedure

. . . . . Flow chart of alice and bob cryptographic tool . . . . .


switching of ciphers
Text
case 'a'
encryption/decryption of message using autokeycipher
Text
case 'h'
encryption/decryption of message using hill cipher
case 'v'
encryption/decryption of message using vigenère cipher
case 'p'
encryption/decryption of message using playfair cipher
cipher text/original text

2.1 Autokey encrytion and decryption:-

First of all we want a key as long as the message with the keyword added to the prefix of message and triming the added key message to the length of original message we get the key to encrypt the message c(i)=(p(i)+k(i))mod 26.In decryption with the key we can recover first letter of original text and it will be used as key for next letter of cipher text and we will do until rest of message p(i)=(c(i)-k(i))mod 26.

2.2 Hill cipher encrytion and decryption:-

In this we take meassage length which is perfect sqaure and key length is square root of message length.Genarate a message matrix of n*n and key matrix of 1*n and key matrix should be invertible.To encrypt an message we have to multiply both matices against mod 26.To decrypt the message, each block is multiplied by the inverse of the matrix used for original message.

2.3 Vigenere cipher encrytion and decryption:-

For this cipher we have to generate a key stream of length as much as the length of message using key.To encrypt message use key stream to change as cipher text c(i)=(p(i)+k(i))mod 26.And to decrypt genarate key stream and decrypt the cipher text to original text p(i)=(c(i)-k(i))mod 26.

2.4 playfair cipher encrytion and decryption:-

For this cipher we have to genarate a key matix of 5*5 filling the letters of keyword from left to right and fill remaining matix with the remaining letters in alphabetic order.The remaining letters in alphabetic order.Plaintext is encrypted two letters at a time.Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x.Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last.To encrypt Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last.Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.Decryption is just the reverse of Encryption Process.

3 Major Components

Java code for alice and bob.

```
\\For Alice to encryt message
import java.util.*;
import java.io.*;
class Autokey{
public static String autoEncryption(String msg, String key)
{
String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
int len = msg.length();
String newKey = key.concat(msg);
newKey = newKey.substring(0, newKey.length() - key.length());
String encryptMsg = "";
for (int x = 0; x < len; x++) {
int first = alphabet.indexOf(msg.charAt(x));
int second = alphabet.indexOf(newKey.charAt(x));
int total = (first + second) % 26;
encryptMsg += alphabet.charAt(total);
}
return encryptMsg;
}
}
class Hill
{
    int keymatrix[][];
    int linematrix[];
    int resultmatrix[];

    public void divide(String temp, int s)
    {
        while (temp.length() > s)
        {
            String sub = temp.substring(0, s);
            temp = temp.substring(s, temp.length());
            perform(sub);
        }
        if (temp.length() == s)
            perform(temp);
        else if (temp.length() < s)
        {
            for (int i = temp.length(); i < s; i++)
                temp = temp + 'x';
            perform(temp);
        }
    }

    public void perform(String line)
    {
        linetomatrix(line);
        linemultiplykey(line.length());
        result(line.length());
    }

    public void keytomatrix(String key, int len)
    {
        keymatrix = new int[len][len];
        int c = 0;
        for (int i = 0; i < len; i++)
        {
            for (int j = 0; j < len; j++)
            {
                keymatrix[i][j] = ((int) key.charAt(c)) - 97;
                c++;
            }
        }
    }

    public void linetomatrix(String line)
    {
        linematrix = new int[line.length()];
        for (int i = 0; i < line.length(); i++)
        {
            linematrix[i] = ((int) line.charAt(i)) - 97;
        }
    }

    public void linemultiplykey(int len)
    {
        resultmatrix = new int[len];
        for (int i = 0; i < len; i++)
        {
            for (int j = 0; j < len; j++)
            {
                resultmatrix[i] += keymatrix[i][j] * linematrix[j];
            }
            resultmatrix[i] %= 26;
        }
    }

    public void result(int len)
    {
        String result = "";
        for (int i = 0; i < len; i++)
        {
            result += (char) (resultmatrix[i] + 97);
        }
        System.out.print(result);
    }

    public boolean check(String key, int len)
    {
        keytomatrix(key, len);
        int d = determinant(keymatrix, len);
        d = d % 26;
        if (d == 0)
        {
            System.out.println("Invalid key!!! Key is not invertible because determinant=0...");
            return false;
        }
        else if (d % 2 == 0 || d % 13 == 0)
        {
            System.out.println("Invalid key!!! Key is not invertible because determinant has common factor with 26...");
            return false;
        }
        else
        {
            return true;
        }
    }

    public int determinant(int A[][], int N)
    {
        int res;
        if (N == 1)
            res = A[0][0];
        else if (N == 2)
        {
            res = A[0][0] * A[1][1] - A[1][0] * A[0][1];
        }
        else
        {
            res = 0;
            for (int j1 = 0; j1 < N; j1++)
            {
                int m[][] = new int[N - 1][N - 1];
                for (int i = 1; i < N; i++)
                {
                    int j2 = 0;
                    for (int j = 0; j < N; j++)
                    {
                        if (j == j1)
                            continue;
                        m[i - 1][j2] = A[i][j];
                        j2++;
                    }
                }
                res += Math.pow(-1.0, 1.0 + j1 + 1.0) * A[0][j1]
                        * determinant(m, N - 1);
            }
        }
        return res;
    }

    public void cofact(int num[][], int f)
    {
        int b[][], fac[][];
        b = new int[f][f];
        fac = new int[f][f];
        int p, q, m, n, i, j;
        for (q = 0; q < f; q++)
        {
            for (p = 0; p < f; p++)
            {
                m = 0;
                n = 0;
                for (i = 0; i < f; i++)
                {
                    for (j = 0; j < f; j++)
                    {
                        b[i][j] = 0;
                        if (i != q && j != p)
                        {
                            b[m][n] = num[i][j];
                            if (n < (f - 2))
                                n++;
                            else
                            {
                                n = 0;
                                m++;
                            }
                        }
                    }
                }
                fac[q][p] = (int) Math.pow(-1, q + p) * determinant(b, f - 1);
            }
        }
        trans(fac, f);
    }

    void trans(int fac[][], int r)
    {
        int i, j;
        int b[][], inv[][];
        b = new int[r][r];
        inv = new int[r][r];
        int d = determinant(keymatrix, r);
        int mi = mi(d % 26);
        mi %= 26;
        if (mi < 0)
            mi += 26;
        for (i = 0; i < r; i++)
        {
            for (j = 0; j < r; j++)
            {
                b[i][j] = fac[j][i];
            }
        }
        for (i = 0; i < r; i++)
        {
            for (j = 0; j < r; j++)
            {
                inv[i][j] = b[i][j] % 26;
                if (inv[i][j] < 0)
                    inv[i][j] += 26;
                inv[i][j] *= mi;
                inv[i][j] %= 26;
            }
        }

    }

    public int mi(int d)
    {
        int q, r1, r2, r, t1, t2, t;
        r1 = 26;
        r2 = d;
        t1 = 0;
        t2 = 1;
        while (r1 != 1 && r2 != 0)
        {
            q = r1 / r2;
            r = r1 % r2;
            t = t1 - (t2 * q);
            r1 = r2;
            r2 = r;
            t1 = t2;
            t2 = t;
        }
        return (t1 + t2);
    }
}
class vigenere
{


static String generateKey(String str, String key)
{
    int x = str.length();

    for (int i = 0; ; i++)
    {
        if (x == i)
            i = 0;
        if (key.length() == str.length())
            break;
        key+=(key.charAt(i));
    }
    return key;
}


static String cipherText(String str, String key)
{
    String cipher_text="";

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

        int x = (str.charAt(i) + key.charAt(i)) %26;

        x += 'A';

        cipher_text+=(char)(x);
    }
    return cipher_text;
}

}


class Playfair
{
   private String KeyWord        = new String();
   private String Key            = new String();
   private char   matrix_arr[][] = new char[5][5];

   public void setKey(String k)
   {
       String K_adjust = new String();
    boolean flag = false;
    K_adjust = K_adjust + k.charAt(0);
    for (int i = 1; i < k.length(); i++)
    {
        for (int j = 0; j < K_adjust.length(); j++)
        {
            if (k.charAt(i) == K_adjust.charAt(j))
            {
                flag = true;
            }
        }
        if (flag == false)
            K_adjust = K_adjust + k.charAt(i);
        flag = false;
    }
    KeyWord = K_adjust;
}

public void KeyGen()
{
    boolean flag = true;
    char current;
    Key = KeyWord;
    for (int i = 0; i < 26; i++)
    {
        current = (char) (i + 97);
        if (current == 'j')
            continue;
        for (int j = 0; j < KeyWord.length(); j++)
        {
            if (current == KeyWord.charAt(j))
            {
                flag = false;
                break;
            }
        }
        if (flag)
            Key = Key + current;
        flag = true;
    }
    matrix();
}

private void matrix()
{
    int counter = 0;
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            matrix_arr[i][j] = Key.charAt(counter);
            counter++;
        }
    }
}

private String format(String old_text)
{
    int i = 0;
    int len = 0;
    String text = new String();
    len = old_text.length();
    for (int tmp = 0; tmp < len; tmp++)
    {
        if (old_text.charAt(tmp) == 'j')
        {
            text = text + 'i';
        }
        else
            text = text + old_text.charAt(tmp);
    }
    len = text.length();
    for (i = 0; i < len; i = i + 2)
    {
        if (text.charAt(i + 1) == text.charAt(i))
        {
            text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);
        }
    }
    return text;
}

private String[] Divid2Pairs(String new_string)
{
    String Original = format(new_string);
    int size = Original.length();
    if (size % 2 != 0)
    {
        size++;
        Original = Original + 'x';
    }
    String x[] = new String[size / 2];
    int counter = 0;
    for (int i = 0; i < size / 2; i++)
    {
        x[i] = Original.substring(counter, counter + 2);
        counter = counter + 2;
    }
    return x;
}

public int[] GetDiminsions(char letter)
{
    int[] key = new int[2];
    if (letter == 'j')
        letter = 'i';
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (matrix_arr[i][j] == letter)
            {
                key[0] = i;
                key[1] = j;
                break;
            }
        }
    }
    return key;
}

public String encryptMessage(String Source)
{
    String src_arr[] = Divid2Pairs(Source);
    String Code = new String();
    char one;
    char two;
    int part1[] = new int[2];
    int part2[] = new int[2];
    for (int i = 0; i < src_arr.length; i++)
    {
        one = src_arr[i].charAt(0);
```

```
            two = src_arr[i].charAt(1);
            part1 = GetDiminsions(one);
            part2 = GetDiminsions(two);
            if (part1[0] == part2[0])
            {
                if (part1[1] < 4)
                    part1[1]++;
                else
                    part1[1] = 0;
                if (part2[1] < 4)
                    part2[1]++;
                else
                    part2[1] = 0;
            }
            else if (part1[1] == part2[1])
            {
                if (part1[0] < 4)
                    part1[0]++;
                else
                    part1[0] = 0;
                if (part2[0] < 4)
                    part2[0]++;
                else
                    part2[0] = 0;
            }
            else
            {
                int temp = part1[1];
                part1[1] = part2[1];
                part2[1] = temp;
            }
            Code = Code + matrix_arr[part1[0]][part1[1]]
                    + matrix_arr[part2[0]][part2[1]];
        }
        return Code;
    }
}

public class encrypt {

public static void main(String[] args)
{
char choice;
Scanner sc = new Scanner(System.in);
System.out.println("Enter a for Autokey encryption.");
System.out.println("Enter h for hill cipher encrytion.");
System.out.println("Enter v for vigenere encryption.");
System.out.println("Enter p for playfair encryption.");
String st = sc.next();
choice = st.charAt(0);
System.out.println("Enter message to encrypt");
String msg = sc.next();
System.out.println("Enter key to encrypt");
String key = sc.next();
switch(choice) {
case 'a':
msg = msg.toUpperCase();
 key = key.toUpperCase();
        String enc = Autokey.autoEncryption(msg, key);
System.out.println("Ciphertext : " + enc.toLowerCase());
break;
case 'h':
Hill obj = new Hill();
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        double sq = Math.sqrt(key.length());
        if (sq != (long) sq)
```

```
            System.out.println("Invalid key length!!! Does not form a square matrix...");
        else
        {
            int s = (int) sq;
            if (obj.check(key, s))
            {
                System.out.println("Ciphertext : ");
                obj.divide(msg, s);
                obj.cofact(obj.keymatrix, s);
            }
        }

     break;
case 'v':
String str = msg.toUpperCase();
    String keyword = key.toUpperCase();

    key = vigenere.generateKey(str, keyword);
    String cipher_text = vigenere.cipherText(str, key);

    System.out.println("Ciphertext : " + cipher_text.toLowerCase());
    break;
case 'p':
Playfair x = new Playfair();
        x.setKey(key);
        x.KeyGen();
        if (msg.length() % 2 == 0)
        {
            System.out.println("Ciphertext : " + x.encryptMessage(msg));

        }
        else
        {
            System.out.println("Message length should be even");
        }
        break;
}
}
}


//For bob to decrypt ciphertext

import java.io.*;
import java.util.Scanner;

class Autokey{
public static String autoDecryption(String msg, String key)
{
String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
String currentKey = key;
String decryptMsg = "";
for (int x = 0; x < msg.length(); x++) {
int get1 = alphabet.indexOf(msg.charAt(x));
int get2 = alphabet.indexOf(currentKey.charAt(x));
int total = (get1 - get2) % 26;
total = (total < 0) ? total + 26 : total;
decryptMsg += alphabet.charAt(total);
currentKey += alphabet.charAt(total);
}
return decryptMsg;
}
}
class vigenre
{
static String generateKey(String str, String key)
```

```
{
    int x = str.length();

    for (int i = 0; ; i++)
    {
        if (x == i)
            i = 0;
        if (key.length() == str.length())
            break;
        key+=(key.charAt(i));
    }
    return key;
}

static String originalText(String cipher_text, String key)
{
    String orig_text="";

    for (int i = 0 ; i < cipher_text.length() &&
                            i < key.length(); i++)
    {

        int x = (cipher_text.charAt(i) -
                    key.charAt(i) + 26) %26;


        x += 'A';
        orig_text+=(char)(x);
    }
    return orig_text.toLowerCase();
}
}
class Playfair
{
    private String KeyWord        = new String();
    private String Key            = new String();
    private char   matrix_arr[][] = new char[5][5];

    public void setKey(String k)
    {
        String K_adjust = new String();
        boolean flag = false;
        K_adjust = K_adjust + k.charAt(0);
        for (int i = 1; i < k.length(); i++)
        {
            for (int j = 0; j < K_adjust.length(); j++)
            {
                if (k.charAt(i) == K_adjust.charAt(j))
                {
                    flag = true;
                }
            }
            if (flag == false)
                K_adjust = K_adjust + k.charAt(i);
            flag = false;
        }
        KeyWord = K_adjust;
    }

    public void KeyGen()
    {
        boolean flag = true;
        char current;
        Key = KeyWord;
        for (int i = 0; i < 26; i++)
        {
```

```
        current = (char) (i + 97);
        if (current == ’j’)
            continue;
        for (int j = 0; j < KeyWord.length(); j++)
        {
            if (current == KeyWord.charAt(j))
            {
                flag = false;
                break;
            }
        }
        if (flag)
            Key = Key + current;
        flag = true;
    }
    matrix();
}

private void matrix()
{
    int counter = 0;
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            matrix_arr[i][j] = Key.charAt(counter);
            counter++;
        }
    }
}

private String format(String old_text)
{
    int i = 0;
    int len = 0;
    String text = new String();
    len = old_text.length();
    for (int tmp = 0; tmp < len; tmp++)
    {
        if (old_text.charAt(tmp) == ’j’)
        {
            text = text + ’i’;
        }
        else
            text = text + old_text.charAt(tmp);
    }
    len = text.length();
    for (i = 0; i < len; i = i + 2)
    {
        if (text.charAt(i + 1) == text.charAt(i))
        {
            text = text.substring(0, i + 1) + ’x’ + text.substring(i + 1);
        }
    }
    return text;
}

private String[] Divid2Pairs(String new_string)
{
    String Original = format(new_string);
    int size = Original.length();
    if (size % 2 != 0)
    {
        size++;
        Original = Original + ’x’;
    }
    String x[] = new String[size / 2];
    int counter = 0;
    for (int i = 0; i < size / 2; i++)
    {
        x[i] = Original.substring(counter, counter + 2);
        counter = counter + 2;
    }
    return x;
}

public int[] GetDiminsions(char letter)
{
    int[] key = new int[2];
    if (letter == 'j')
        letter = 'i';
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (matrix_arr[i][j] == letter)
            {
                key[0] = i;
                key[1] = j;
                break;
            }
        }
    }
    return key;
}


public String decryptMessage(String Code)
{
    String Original = new String();
    String src_arr[] = Divid2Pairs(Code);
    char one;
    char two;
    int part1[] = new int[2];
    int part2[] = new int[2];
    for (int i = 0; i < src_arr.length; i++)
    {
        one = src_arr[i].charAt(0);
        two = src_arr[i].charAt(1);
        part1 = GetDiminsions(one);
        part2 = GetDiminsions(two);
        if (part1[0] == part2[0])
        {
            if (part1[1] > 0)
                part1[1]--;
            else
                part1[1] = 4;
            if (part2[1] > 0)
                part2[1]--;
            else
                part2[1] = 4;
        }
        else if (part1[1] == part2[1])
        {
            if (part1[0] > 0)
                part1[0]--;
            else
                part1[0] = 4;
            if (part2[0] > 0)
                part2[0]--;
            else
                part2[0] = 4;
            }
            else
            {
                int temp = part1[1];
                part1[1] = part2[1];
                part2[1] = temp;
            }
            Original = Original + matrix_arr[part1[0]][part1[1]]
                    + matrix_arr[part2[0]][part2[1]];
        }
        return Original;
    }
}
class Hill {

int[] lm;
    int[][] km;
    int[] rm;
    int [][] invK;

    public void performDivision(String temp, int s)
    {
        while (temp.length() > s)
        {
            String line = temp.substring(0, s);
            temp = temp.substring(s, temp.length());
            calLineMatrix(line);
            multiplyLineByInvKey(line.length());
            showResult(line.length());
        }
        if (temp.length() == s){

            calLineMatrix(temp);
            this.multiplyLineByInvKey(temp.length());
            showResult(temp.length());




        }
        else if (temp.length() < s)
        {
            for (int i = temp.length(); i < s; i++)
                temp = temp + 'x';
            calLineMatrix(temp);
            multiplyLineByInvKey(temp.length());
            showResult(temp.length());


        }
    }


    public void calKeyMatrix(String key, int len)
    {
        km = new int[len][len];
        int k = 0;
        for (int i = 0; i < len; i++)
        {
            for (int j = 0; j < len; j++)
            {
                km[i][j] = ((int) key.charAt(k)) - 97;
                k++;
            }
        }
    }
```

```
public void calLineMatrix(String line)
{
    lm = new int[line.length()];
    for (int i = 0; i < line.length(); i++)
    {
        lm[i] = ((int) line.charAt(i)) - 97;
    }
}


public void multiplyLineByInvKey(int len)
{

    rm = new int[len];
    for (int i = 0; i < len; i++)
    {
        for (int j = 0; j < len; j++)
        {
            rm[i] += invK[i][j] * lm[j];
        }
        rm[i] %= 26;
    }
}


public void showResult(int len)
{
    String result = "";
    for (int i = 0; i < len; i++)
    {
        result += (char) (rm[i] + 97);
    }
    System.out.print(result);
}



public int calDeterminant(int A[][], int N)
{
    int resultOfDet;
    switch (N) {
        case 1:
            resultOfDet = A[0][0];
            break;
        case 2:
            resultOfDet = A[0][0] * A[1][1] - A[1][0] * A[0][1];
            break;
        default:
            resultOfDet = 0;
            for (int j1 = 0; j1 < N; j1++)
            {
                int m[][] = new int[N - 1][N - 1];
                for (int i = 1; i < N; i++)
                {
                    int j2 = 0;
                    for (int j = 0; j < N; j++)
                    {
                        if (j == j1)
                            continue;
                        m[i - 1][j2] = A[i][j];
                        j2++;
                    }
                }
                resultOfDet += Math.pow(-1.0, 1.0 + j1 + 1.0) * A[0][j1]
                        * calDeterminant(m, N - 1);
            }   break;
    }
    return resultOfDet;
}

public void cofact(int num[][], int f)
{
    int b[][], fac[][];
    b = new int[f][f];
    fac = new int[f][f];
    int p, q, m, n, i, j;
    for (q = 0; q < f; q++)
    {
        for (p = 0; p < f; p++)
        {
            m = 0;
            n = 0;
            for (i = 0; i < f; i++)
            {
                for (j = 0; j < f; j++)
                {
                    b[i][j] = 0;
                    if (i != q && j != p)
                    {
                        b[m][n] = num[i][j];
                        if (n < (f - 2))
                            n++;
                        else
                        {
                            n = 0;
                            m++;
                        }
                    }
                }
            }
            fac[q][p] = (int) Math.pow(-1, q + p) * calDeterminant(b, f - 1);
        }
    }
    trans(fac, f);
}

void trans(int fac[][], int r)
{
    int i, j;
    int b[][], inv[][];
    b = new int[r][r];
    inv = new int[r][r];
    int d = calDeterminant(km, r);
    int mi = mi(d % 26);
    mi %= 26;
    if (mi < 0)
        mi += 26;
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            b[i][j] = fac[j][i];
        }
    }
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            inv[i][j] = b[i][j] % 26;
            if (inv[i][j] < 0)
                inv[i][j] += 26;
            inv[i][j] *= mi;
```

```
                inv[i][j] %= 26;
            }
        }


        invK = inv;
    }

    public int mi(int d)
    {
        int q, r1, r2, r, t1, t2, t;
        r1 = 26;
        r2 = d;
        t1 = 0;
        t2 = 1;
        while (r1 != 1 && r2 != 0)
        {
            q = r1 / r2;
            r = r1 % r2;
            t = t1 - (t2 * q);
            r1 = r2;
            r2 = r;
            t1 = t2;
            t2 = t;
        }
        return (t1 + t2);
    }

    public void matrixtoinvkey(int inv[][], int n)
    {
        String invkey = "";
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                invkey += (char) (inv[i][j] + 97);
            }
        }
    }
     public boolean check(String key, int len)
    {
        calKeyMatrix(key, len);
        int d = calDeterminant(km, len);
        d = d % 26;
        if (d == 0)
        {
            System.out.println("Key is not invertible");
            return false;
        }
        else if (d % 2 == 0 || d % 13 == 0)
        {
            System.out.println("Key is not invertible");
            return false;
        }
        else
        {
            return true;
        }
    }

}

public class Decrypt{
public static void main(String[] args)
{
char choice;
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter a for Autokey decryption.");
System.out.println("Enter h for hill cipher decrytion.");
System.out.println("Enter v for vigenere decryption.");
System.out.println("Enter p for playfair decryption.");
String st = sc.next();
choice = st.charAt(0);
System.out.println("Enter ciphertext to decrypt");
String enc = sc.next();
System.out.println("Enter key to decrypt");
String key = sc.next();
switch(choice) {
case 'a':
enc = enc.toUpperCase();
key = key.toUpperCase();
String msg = Autokey.autoDecryption(enc, key);
System.out.println("Original/Decrypted Text : "+ msg.toLowerCase());
break;
case 'v':
String cipher_text = enc.toUpperCase();
    key = key.toUpperCase() ;
    key = vigenre.generateKey(cipher_text, key);
System.out.println("Original/Decrypted Text : "+ vigenre.originalText(cipher_text, key));
break;
case 'p':
   Playfair x = new Playfair();
       x.setKey(key);
       x.KeyGen();
       System.out.println("Original/Decrypted Text : "+ x.decryptMessage(enc));
       break;
case 'h':
Hill obj = new Hill();
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        double sq = Math.sqrt(key.length());
        int size = (int) sq;
        if (obj.check(key, size))
            {
                System.out.print("Original/Decrypted Text : ");
                obj.cofact(obj.km, size);
                obj.performDivision(enc, size);
            }
        break;

}
}
}
```

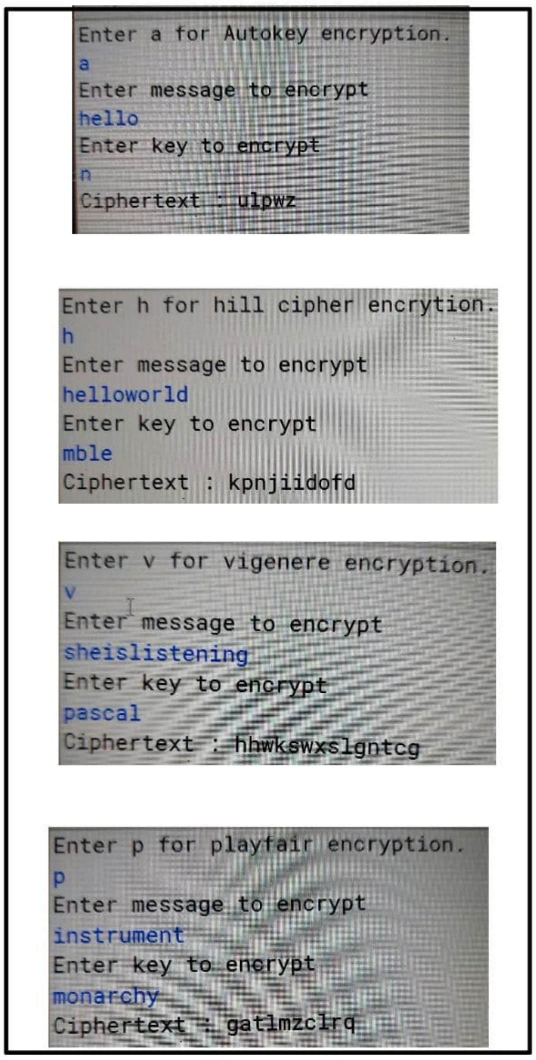
.

.

.

4 Results

1 and 2.



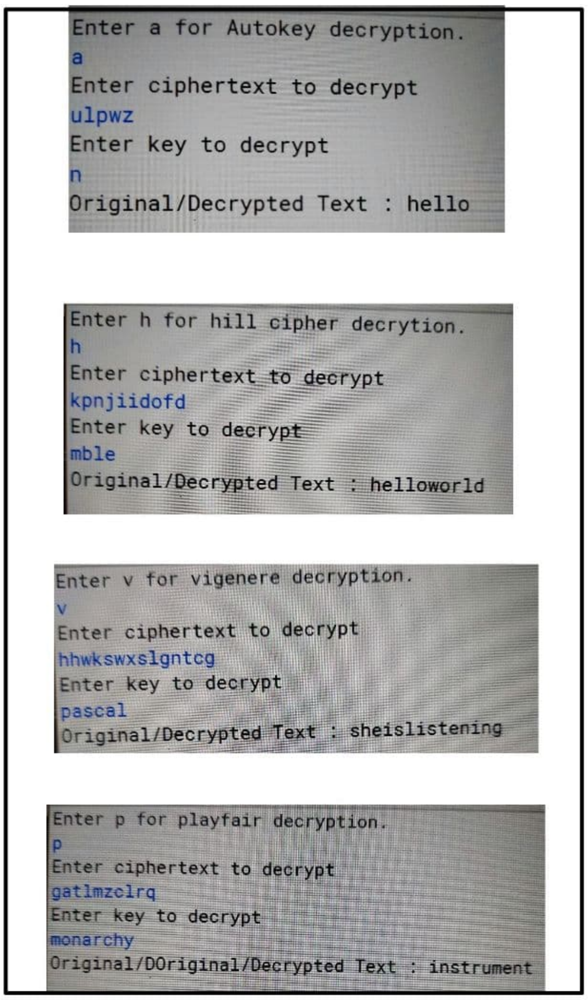
```
Enter a for Autokey encryption.
a
Enter message to encrypt
hello
Enter key to encrypt
n
Ciphertext : ulpwz
```

```
Enter h for hill cipher encrytion.
h
Enter message to encrypt
helloworld
Enter key to encrypt
mble
Ciphertext : kpnjiidofd
```

```
Enter v for vigenere encryption.
v
Enter message to encrypt
sheislistening
Enter key to encrypt
pascal
Ciphertext : hhwkswxslgntcg
```

```
Enter p for playfair encryption.
p
Enter message to encrypt
instrument
Enter key to encrypt
monarchy
Ciphertext : gatlmzclrq
```

Figure 1: Computations at Sender-side (Alice).



```
Enter a for Autokey decryption.
a
Enter ciphertext to decrypt
ulpwz
Enter key to decrypt
n
Original/Decrypted Text : hello
```

```
Enter h for hill cipher decrytion.
h
Enter ciphertext to decrypt
kpnjiidofd
Enter key to decrypt
mble
Original/Decrypted Text : helloworld
```

```
Enter v for vigenere decryption.
v
Enter ciphertext to decrypt
hhwkswxslgntcg
Enter key to decrypt
pascal
Original/Decrypted Text : sheislistening
```

```
Enter p for playfair decryption.
p
Enter ciphertext to decrypt
gatlmzclrq
Enter key to decrypt
monarchy
Original/DOriginal/Decrypted Text : instrument
```

Figure 2: Computations at Receiver-side (Bob).

References

[1] Behrouz A Forouzan, Debdeep Mukhopadhyay, “Cryptography and Network Security”, Mc Graw Hill, Third Edition, 2015.

[2] William Stallings, “Cryptography and Network Security: Principles and Practice”, Pearson Education, Seventh Edition, 2017.

[3] Overleaf: a collaborative cloud-based LaTeX editor used for writing, editing and publishing scientific documents. `http://www.overleaf.com`.