CRYPTOGRAPHY LABORATORY FILE
CS-511

Submitted to:
Dr. Samayveer Singh
Assistant Professor
Department of Computer Science

Submitted by:
Shashi Shekhar Azad
Roll No.: 23203029
M. Tech. CSE 1st Semester

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
DR. B. R. AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY
JALANDHAR

Objective 3

Write a program to implement the encryption and decryption process of the following substitution techniques:

a. Playfair Cipher

Procedure: Playfair cipher can be illustrated as follow
Let, key: CRYPTOGRAPHY
Message: COMMUNICATE

Create 5X5 matrix of alphabet with Key.

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

Add X (filler) between of repetitive characters make pair(diagraph) after that substitute the characters as per the Playfair rules.

CO MX MU NI CA TE
OD QV LV QF YO RK

Code:

```
// encodes text input using the Playfair cipher
// results (both encode and decode) are output with the table
// requires a user keyword for the cipher
// ues letter 'X' for insertion, I replace J

import java.awt.Point;
import java.util.Scanner;

public class PlayFairCipher {
  // length of digraph array
  private int length = 0;

  // table 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
private PlayFairCipher() {

  // prompts user for the keyword to use for encoding & creates tables
  String output = "";
  String decodedOutput = "";
  System.out.println("Please enter the key.");
  Scanner sc = new Scanner(System.in);
  String keyword = parseString(sc);
  while (keyword.equals(""))
    keyword = parseString(sc);
  System.out.println();
  table = this.cipherTable(keyword);

  //Print playfair table
  this.printTable(table);

  // prompts user for message to be encoded
  System.out.println("Please input the message:");
  String inputTxt = parseString(sc);
  while (inputTxt.equals(""))
    inputTxt = parseString(sc);
  System.out.println();

  // encodes and then decodes the encoded message
  System.out.println("Enter your choice:");
  System.out.println("1. Encrypt\n2. Decrypt\n3. Both");
  int ch = sc.nextInt();
  if (ch == 1) {
    output = cipher(inputTxt);
    this.printResults("Encrypted message", output);
  } else if (ch == 2) {
    decodedOutput = decode(inputTxt);
    this.printResults("Decrypted message", decodedOutput);

  } else if (ch == 3) {
    output = cipher(inputTxt);
    decodedOutput = decode(output);
    this.printResults("Encrypted message", output);
    this.printResults("Decrypted message", decodedOutput);
  }
  else System.out.println("Oh! no, Invalid choice!");
}

// parses any input string to remove numbers, punctuation,
```

```
// replaces any J's with I's, and makes string all caps
private String parseString(Scanner s) {
  String parse = s.nextLine();
  parse = parse.toUpperCase();
  parse = parse.replaceAll("[^A-Z]", "");
  parse = parse.replace("J", "I");
  return parse;
}

// creates the cipher table based on some input string (already parsed)
private String[][] cipherTable(String key) {
  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 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;
    }
```

```
  }

  // adds an x to the last digraph, if necessary
  String[] digraph = new String[length];
  for (int j = 0; j < length; j++) {
    if (j == (length - 1) && in.length() / 2 == (length - 1))
      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;
}

// encodes the digraph input with the cipher's specifications
private String[] encodeDigraph(String di[]) {
  String[] enc = 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();

    // case 1: letters in digraph are of same row, shift columns to right
    if (r1 == r2) {
      c1 = (c1 + 1) % 5;
      c2 = (c2 + 1) % 5;

      // case 2: letters in digraph are of same column, shift rows down
    } else if (c1 == c2) {
      r1 = (r1 + 1) % 5;
      r2 = (r2 + 1) % 5;

      // case 3: letters in digraph form rectangle, swap first column # with second
      // column #
    } else {
      int temp = c1;
      c1 = c2;
      c2 = temp;
```

```
    }

    // performs the table look-up and puts those values into the encoded array
    enc[i] = table[r1][c1] + "" + table[r2][c2];
  }
  return enc;
}

// 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 {
      int temp = c1;
      c1 = c2;
      c2 = temp;
    }
    decoded = decoded + table[r1][c1] + table[r2][c2];
  }
  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;
}

// prints the cipher table out for the user
```

```
  private void printTable(String[][] printedTable) {
    System.out.println("\nPlayfair Cipher Matrix (5X5).");
    System.out.println();

    for (int i = 0; i < 5; i++) {
      for (int j = 0; j < 5; j++) {
        System.out.print(printedTable[i][j] + " ");
      }
      System.out.println();
    }
    System.out.println();
  }
  // prints results (encryption and decryption)
  private void printResults(String text, String output) {
    System.out.println(text + ": " + output);
  }
}
```

Output:

```
PS D:\DATAs\NITJ\CryptoLab> java PlayFairCipher

Please enter the key: CRYPTOGRAPHY


Playfair Cipher Matrix (5X5).

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

Please enter the message: COMMUNICATE

Enter your choice:
1. Encrypt
2. Decrypt
3. Both
3
Encrypted message: ODQVLVQFYORK
Decrypted message: COMXMUNICATE

PS D:\DATAs\NITJ\CryptoLab>
```

Objective 3

Write a program to implement the encryption and decryption process of the following substitution techniques:

b. Hill Cipher

Procedure:

Plain text: cryptography
Key: sanfoundr

Code:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class HillCipher {
    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;
    }
  }
  System.out.print("\nInverse key:");
  matrixtoinvkey(inv, r);
}

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);
        }
    }
    System.out.print(invkey);
}

public static void main(String args[]) throws IOException {
    HillCipher obj = new HillCipher();
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    int choice;
    System.out.println("Enter your choice:\n1: Encryption\n2: Decryption");
    choice = Integer.parseInt(in.readLine());
    System.out.print("\nEnter the message: ");
    String line = in.readLine();
    if (choice == 1) {
        System.out.print("\nEnter the key: ");
    } else if (choice == 2) {
        System.out.print("\nEnter the inverse key: ");
    } else {
        System.out.print("\nInvalid choice, enter valid one: ");
    }
    String key = in.readLine();
    double sq = Math.sqrt(key.length());
    if (sq != (long) sq)
        System.out
                .println("\nInvalid key length!!! Does not form a square matrix...");
    else {
```

```
            int s = (int) sq;
            if (obj.check(key, s)) {
                System.out.print("\nResult: ");
                obj.divide(line, s);
                obj.cofact(obj.keymatrix, s);
            }
        }
    }
}
```

Output:

```
PS D:\DATAs\NITJ\CryptoLab> java HillCipher
Enter your choice:
1: Encryption
2: Decryption
1

Enter the message: nitjalandhar

Enter the key: hell

Result: txnwsranxgqf
Inverse key:jsrb
PS D:\DATAs\NITJ\CryptoLab> java HillCipher
Enter your choice:
1: Encryption
2: Decryption
2

Enter the message: txnwsranxgqf

Enter the inverse key: jsrb

Result: nitjalandhar
Inverse key:hell
PS D:\DATAs\NITJ\CryptoLab>
```