Information Security & Cryptography

Nehal Jhajharia

Lab Assignment 2

Implement a menu driven program for 5X5 Playfair Cipher with following functions.

1. Takes text phrases to generate key matrix.
2. Encrypt given plain text.
3. Decrypt given cipher text.

```
import java.util.*;

class Asmt2 {
    static Scanner input = new Scanner(System.in);
    static char cipherMatrix[][] = new char[5][5];
    static ArrayList<String> digrams = new ArrayList<>();
    static String cipher = new String();

    static void increRC(int rc[]) {
        if (rc[1] == 4) {
            rc[1] = 0;
            rc[0]++;
        } else {
            rc[1]++;
        }
    }

    static void setMatrix(String text) {
        boolean alpha[] = new boolean[26];
```

```
    Arrays.fill(alpha, false);

    int rc[] = new int[2];
    rc[0] = 0;
    rc[1] = 0;

    for (int i = 0; i < text.length(); i++) {
        int ch = text.charAt(i) - 97;
        if (!alpha[ch]) {
            alpha[ch] = true;
            if (ch == 'j' - 97 || ch == 'i' - 97) {
                alpha['i' - 97] = true;
                alpha['j' - 97] = true;
                ch = 'i' - 97;
            }
            cipherMatrix[rc[0]][rc[1]] = (char) (ch + 97);
            increRC(rc);
        }
    }

    int t = 0;
    while ((rc[0] < 5) && (t < 26)) {
        if (alpha[t] || t == 'j' - 97) {
            t++;
            continue;
        }

        cipherMatrix[rc[0]][rc[1]] = (char) (t + 97);
        increRC(rc);
        alpha[t] = true;
        t++;
    }
}

static void printMatrix(char matrix[][]) {
    for (int i = 0; i < matrix.length; i++) {
```

```
        System.out.print("| ");
        for (int j = 0; j < matrix[0].length; j++) {
            System.out.print(matrix[i][j]);
            System.out.print(" | ");
        }
        System.out.println();
    }
}

static void setDigrams(String text) {
    digrams.clear();
    boolean flag = false;

    for (int i = 1; i < text.length(); i++) {
        if (text.charAt(i) != text.charAt(i - 1)) {
            digrams.add(text.substring(i - 1, i + 1));
            i++;
        } else {
            digrams.add(text.substring(i - 1, i) + 'z');
            flag = !flag;
        }
    }

    if (text.length() % 2 == 0) {
        if (flag) {
            digrams.add(text.substring(text.length() - 1, text.length()) + 'z');
        }
    } else {
        if (!flag) {
            digrams.add(text.substring(text.length() - 1, text.length()) + 'z');
        }
    }

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

```
    static int[] search(char ch) {
        int coord[] = new int[2];
        char key = ch;
        if (key == 'j') {
            key = 'i';
        }

        for (int i = 0; i < 5; i++) {
            for (int j = 0; j < 5; j++) {
                if (cipherMatrix[i][j] == key) {
                    coord[0] = i;
                    coord[1] = j;
                    return coord;
                }
            }
        }

        return coord;
    }

    static void encrypt(String text) {
        setDigrams(text);
        ArrayList<String> encryptedDigrams = new ArrayList<>();

        for (int i = 0; i < digrams.size(); i++) {
            String currStr = digrams.get(i);
            int coordA[] = search(currStr.charAt(0));
            int coordB[] = search(currStr.charAt(1));
            String temp = "";

            // case 1 - same column
            if (coordA[1] == coordB[1]) {
                int rowA = (coordA[0] + 1) % 5;
                int rowB = (coordB[0] + 1) % 5;
                temp = Character.toString(cipherMatrix[rowA][coordA[1]]) +
Character.toString(cipherMatrix[rowB][coordA[1]]);
```

```
        }

        // case 2 - same row
        else if (coordA[0] == coordB[0]) {
            int colA = (coordA[1] + 1) % 5;
            int colB = (coordB[1] + 1) % 5;
            temp = Character.toString(cipherMatrix[coordA[0]][colA]) +
Character.toString(cipherMatrix[coordA[0]][colB]);
        }

        // case 3 - different column and different row
        else {
            temp = Character.toString(cipherMatrix[coordA[0]][coordB[1]]) +
Character.toString(cipherMatrix[coordB[0]][coordA[1]]);
        }

        encryptedDigrams.add(temp);
      }

      System.out.println(encryptedDigrams);

      cipher = arrayList_to_String(encryptedDigrams);
      System.out.println(cipher);
   }

   static String arrayList_to_String (ArrayList<String> al) {
      String temp = al.get(0);
      for (int i = 1; i < al.size(); i++) {
         temp = temp + al.get(i);
      }
      return temp;
   }

   static void decrypt(String cipher) {
      setDigrams(cipher);
      ArrayList<String> decryptedDigrams = new ArrayList<>();
```

```
    for (int i = 0; i < digrams.size(); i++) {
        String currStr = digrams.get(i);
        int coordA[] = search(currStr.charAt(0));
        int coordB[] = search(currStr.charAt(1));
        String temp = "";

        // case 1 - same column
        if (coordA[1] == coordB[1]) {
            int rowA = (coordA[0] + 4) % 5;
            int rowB = (coordB[0] + 4) % 5;
            temp = Character.toString(cipherMatrix[rowA][coordA[1]]) +
Character.toString(cipherMatrix[rowB][coordA[1]]);
        }

        // case 2 - same row
        else if (coordA[0] == coordB[0]) {
            int colA = (coordA[1] + 4) % 5;
            int colB = (coordB[1] + 4) % 5;
            temp = Character.toString(cipherMatrix[coordA[0]][colA]) +
Character.toString(cipherMatrix[coordA[0]][colB]);
        }

        // case 3 - different column and different row
        else {
            temp = Character.toString(cipherMatrix[coordA[0]][coordB[1]]) +
Character.toString(cipherMatrix[coordB[0]][coordA[1]]);
        }

        decryptedDigrams.add(temp);
    }
    System.out.println(decryptedDigrams);
    String decipher = arrayList_to_String(decryptedDigrams);
    System.out.println(decipher);
  }
```

```
    public static void main(String[] args) {
        System.out.print("Enter key : ");
        String key = input.nextLine();
        System.out.print("Enter text : ");
        String text = input.nextLine();

        setMatrix(key);
        printMatrix(cipherMatrix);
        encrypt(text);
        decrypt(cipher);
    }
}

jhajharia@Nehals-MacBook-Air Asmt2 % javac Asmt2.java
jhajharia@Nehals-MacBook-Air Asmt2 % Java Asmt2
Enter key : jack
Enter text : jackicecream
| i | a | c | k | b |
| d | e | f | g | h |
| l | m | n | o | p |
| q | r | s | t | u |
| v | w | x | y | z |
[ja, ck, ic, ec, re, am]
[ac, kb, ak, fa, wm, er]
ackbakfawmer
[ac, kb, ak, fa, wm, er]
[ia, ck, ic, ec, re, am]
iackicecream
jhajharia@Nehals-MacBook-Air Asmt2 % Java Asmt2
Enter key : nehal
Enter text : jhajharia
| n | e | h | a | l |
| b | c | d | f | g |
| i | k | m | o | p |
| q | r | s | t | u |
| v | w | x | y | z |
```

[jh, aj, ha, ri, az]
[mn, no, al, qk, ly]
mnnoalqkly
[mn, no, al, qk, ly]
[ih, ai, ha, ri, az]
ihaihariaz
jhajharia@Nehals-MacBook-Air Asmt2 %