Information Security & Cryptography

Nehal Jhajharia

Lab Assignment 3

```
// 1) Encryption and decryption using Hill cipher.

import java.util.*;

public class Hill {
   static Scanner input = new Scanner(System.in);

   static boolean falseKey(String text, String key) {
      if ((text.length() * text.length()) != key.length()) {
         System.out.println("Length of key should be square of the length of text.");
         return true;
      }

      return false;
   }

   static String encrypt(String text, String key) {
      int textVector[] = Matrix.generateVector(text);
      int cipherMatrix[][] = Matrix.generateMatrix(key);

      int cipherVector[] = Matrix.multiplyMatrixArray(cipherMatrix, textVector);
      System.out.println("Enrypted cipher vector : ");
      Matrix.printArray(cipherVector);

      String cipher = Matrix.generateString(cipherVector);
      System.out.print("Cipher text : " + cipher + "\n");

      return cipher;
   }
```

```
    static String decrypt(String cipher, String key) {
        int cipherVector[] = Matrix.generateVector(cipher);
        int cipherMatrix[][] = Matrix.generateMatrix(key);
        int inverseMatrix[][] = Matrix.invertMatrix(cipherMatrix);

        int textVector[] = Matrix.multiplyMatrixArray(inverseMatrix, cipherVector);
        System.out.println("Decrypted text vector : ");
        Matrix.printArray(textVector);

        String text = Matrix.generateString(textVector);

        return text;
    }

    public static void main(String[] args) {
        String text = "try";
        String key = "gybnqkurp";

        if (falseKey(text, key)) {
            return;
        }

        System.out.println("Plaintext : " + text);
        System.out.println("Key : " + key);
        System.out.println("\n\nEncrypting...");
        String cipher = encrypt(text, key);
        System.out.println("\n\nDecrypting...");
        String decipher = decrypt(cipher, key);
        System.out.println("Plaintext : " + decipher + "\n");
    }
}

import java.math.BigInteger;

// Java program to find adjoint and inverse of a matrix
```

```
public class Matrix {
    static int[][] generateMatrix(String key) {
        int n = (int)(Math.sqrt(key.length()));

        int matrix[][] = new int[n][n];
        int ptr = 0;
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = key.charAt(ptr) - 'a';
                ptr++;
            }
        }
        System.out.println("Cipher Matrix : ");
        printMatrix(matrix);

        return matrix;
    }

    static String generateString(int vector[]) {
        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < vector.length; i++) {
            sb.append((char)(vector[i] + 'a'));
        }

        return sb.toString();
    }

    static int[] generateVector(String text) {
        int n = text.length();

        int vector[] = new int[n];
        for (int i = 0; i < n; i++) {
            vector[i] = text.charAt(i) - 'a';
        }
        System.out.println("Text Vector : ");
```

```
    printArray(vector);

    return vector;
}

static void printArray(int array[]) {
    for (int i = 0; i < array.length; i++) {
        System.out.println(array[i]);
    }
}

static void printMatrix(int 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 printMatrix(float 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 getCofactor(int matrix[][], int cofactor[][], int p, int q, int N) {
    int i = 0;
    int j = 0;
```

```
    for (int row = 0; row < N; row++) {
        for (int col = 0; col < N; col++) {
            if (row != p && col != q) {
                cofactor[i][j++] = matrix[row][col];

                if (j == N - 1) {
                    j = 0;
                    i++;
                }
            }
        }
    }
}

static int getDeterminant(int matrix[][], int N) {
    int n = matrix.length;
    int D = 0;

    if (N == 1) {
        return matrix[0][0];
    }

    int cofactor[][] = new int[n][n];

    int sign = 1;

    for (int i = 0; i < N; i++) {
        getCofactor(matrix, cofactor, 0, i, N);
        D += sign * matrix[0][i] * getDeterminant(cofactor, N - 1);

        sign = -sign;
    }

    return D;
}
```

```
static int[][] getAdjoint(int matrix[][]) {
    int n = matrix.length;
    int adj[][] = new int[n][n];

    if (n == 1) {
        adj[0][0] = 1;
        return adj;
    }

    int sign = 1;
    int cofactor[][] = new int[n][n];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            getCofactor(matrix, cofactor, i, j, n);

            sign = ((i + j) % 2 == 0) ? 1 : -1;

            adj[j][i] = (sign) * (getDeterminant(cofactor, n - 1));
        }
    }

    System.out.println("Adj matrix : ");
    printMatrix(adj);

    return adj;
}

static int mod26(int num) {
    int n = num;
    while (n < 0) {
        n = 26 + n;
    }

    n %= 26;
```

```
    return n;
}

static int[][] invertMatrix(int matrix[][]) {
    int n = matrix.length;
    int inverse[][] = new int[n][n];

    int det = getDeterminant(matrix, n);
    System.out.println(det);
    if (det == 0) {
        System.out.print("Singular matrix, can't find its inverse");
        return inverse;
    }

    BigInteger big_det = new BigInteger(String.valueOf(det));
    BigInteger t6 = new BigInteger("26");

    big_det = big_det.modInverse(t6);
    det = big_det.intValue();

    int adj[][] = getAdjoint(matrix);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            int temp = adj[i][j];
            temp = mod26(temp);
            temp *= det;

            inverse[i][j] = mod26(temp);
        }
    }

    System.out.println("Inverse Matrix : ");
    printMatrix(inverse);
```

```
        return inverse;
    }

    static int multiplyArrays(int a[], int b[]) {
        int n = a.length;
        int result = 0;

        for (int i = 0; i < n; i++) {
            result += (a[i] * b[i]);
        }

        return result;
    }

    static int[] multiplyMatrixArray(int matrix[][], int array[]) {
        int n = array.length;

        int result[] = new int[n];

        for (int i = 0; i < n; i++) {
            result[i] = (multiplyArrays(matrix[i], array)) % 26;
        }

        return result;
    }
}

jhajharia@Nehals-MacBook-Air Asmt3 % javac Hill.java
jhajharia@Nehals-MacBook-Air Asmt3 % Java Hill
Plaintext : try
Key : gybnqkurp


Encrypting...
Text Vector :
19
```

17
24
Cipher Matrix :
6	24	1
13	16	10
20	17	15
Enrypted cipher vector :
0
5
15
Cipher text : afp

Decrypting...
Text Vector :
0
5
15
Cipher Matrix :
6	24	1
13	16	10
20	17	15
441
Adj matrix :
70	-343	224
5	70	-47
-99	378	-216
Inverse Matrix :
8	5	10
21	8	21
21	12	8
Decrypted text vector :
19
17
24
Plaintext : try

jhajharia@Nehals-MacBook-Air Asmt3 %

// 2) Encryption and decryption using Vigenere cipher.

```
class Vigenere {
    static String generateKey(String str, String key) {
        int l = str.length();
        for (int i = 0;; i++) {
            if (l == i) {
                i = 0;
            }
            if (key.length() == str.length()) {
                break;
            }

            key += (key.charAt(i));
        }

        return key;
    }

    static String encrypt(String str, String key) {
        String ciphertext = "";
        for (int i = 0; i < str.length(); i++) {
            int x = (str.charAt(i) + key.charAt(i)) % 26;
            x += 'A';
            ciphertext += (char) (x);
        }

        return ciphertext;
    }
```

```
    static String decrypt(String ciphertext, String key) {
        String orig_text = "";

        for (int i = 0; i < ciphertext.length() &&
                i < key.length(); i++) {
            int x = (ciphertext.charAt(i) -
                    key.charAt(i) + 26) % 26;

            x += 'A';
            orig_text += (char) (x);
        }
        return orig_text;
    }

    public static void main(String[] args) {
        String str = "PRACTICALTHREE";
        String keyword = "NEHAL";

        String key = generateKey(str, keyword);
        String ciphertext = encrypt(str, key);

        System.out.println("Ciphertext : " + ciphertext + "\n");
        System.out.println("Decrypted Text : " + decrypt(ciphertext, key));
    }
}
```

```
jhajharia@Nehals-MacBook-Air Asmt3 % javac Vigenere.java
jhajharia@Nehals-MacBook-Air Asmt3 % Java Vigenere
Ciphertext : CVHCEVGHLEUVLE

Decrypted Text : PRACTICALTHREE
jhajharia@Nehals-MacBook-Air Asmt3 %
```