Experiment 5

Aim: Write a program to implement Hill Cipher encryption-decryption.

Theory: Hill cipher is a polygraphic substitution cipher based on linear algebra. Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, …, Z = 25 is used, but this is not an essential feature of the cipher. To encrypt a message, each block of n letters (considered as an n-component vector) is multiplied by an invertible n × n matrix, against modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption. The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26)

Encryption

In order to encrypt a message using the Hill cipher, the sender and receiver must first agree upon a key matrix A of size n x n. A must be invertible mod 26. The plaintext will then be enciphered in blocks of size n. In the following example A is a 2 x 2 matrix and the message will be enciphered in blocks of 2 characters.

$$Encrypt(\text{ACT}) = POH$$

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} = \begin{bmatrix} 67 \\ 222 \\ 319 \end{bmatrix} \equiv \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} \%26$$

Decryption

Decryption follows Encryption but uses the inverse of the encryption matrix instead.

$$Decrypt(\text{"POH"}) = ACT$$

$$\begin{bmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{bmatrix}^{-1} \begin{bmatrix} 15 \\ 14 \\ 7 \end{bmatrix} = \begin{bmatrix} 260 \\ 574 \\ 539 \end{bmatrix} \equiv \begin{bmatrix} 0 \\ 2 \\ 19 \end{bmatrix} \%26$$

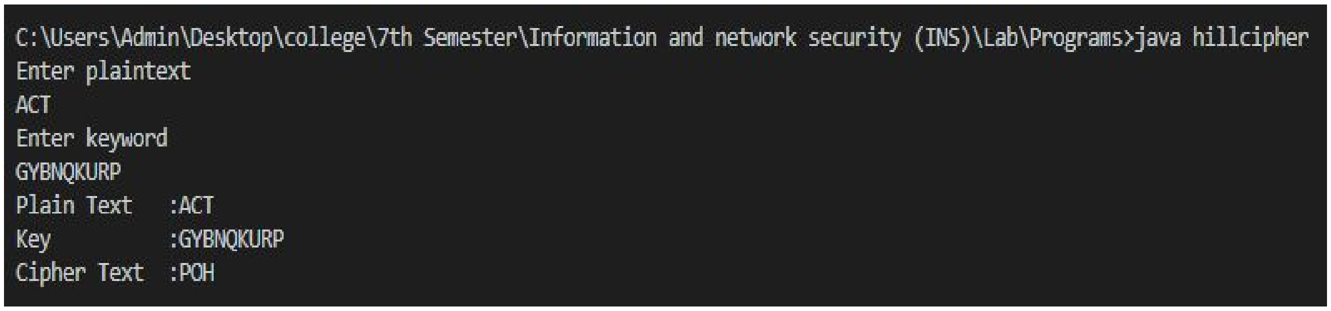
Source Code:

```
import java.util.*;
import java.util.Scanner;
public class hillcipher {
public static void getKeyMatrix(String key, int keyMatrix[][])
{
  int k = 0;
  for (int i = 0; i < 3; i++)
```

```
    {
        for (int j = 0; j < 3; j++)
        {
            keyMatrix[i][j] = (key.charAt(k)) % 65;
            k++;
        }
    }
}
public static void encrypt(int cipherMatrix[][],int keyMatrix[][], int messageVector[][])
{
    int x, i, j;
    for (i = 0; i < 3; i++)
    { for (j = 0; j < 1; j++)
        {
            cipherMatrix[i][j] = 0;
            for (x = 0; x < 3; x++)
            {
                cipherMatrix[i][j] += keyMatrix[i][x] * messageVector[x][j];
            }
            cipherMatrix[i][j] = cipherMatrix[i][j] % 26;
        }
    }
}

public static void Decrypt(int decrypt[][],int keyMatrix[][], int messageVector[][])
{
    int i, j, k;
    int [][]b = new int[3][3];
    inverse(b);
    for(i = 0; i < 3; i++)
        {
            for(j = 0; j < 1; j++)
            {
                        for(k = 0; k < 3; k++)
                        {
                                    decrypt[i][j] = decrypt[i][j] + b[i][k] * encrypt[k][j];
                        }
                        decrypt[i][j] = decrypt[i][j] % 26;
            }
        }
    }
  void inverse( int b[][] )
  {
     int i, j, k;
     float p, q;
     for(i = 0; i < 3; i++)
        for(j = 0; j < 3; j++)
           {
           if(i == j)
              b[i][j]=1;
           else
              b[i][j]=0;
           }
     for(k = 0; k < 3; k++) {
        for(i = 0; i < 3; i++) {
           p = c[i][k];
           q = c[k][k];
           for(j = 0; j < 3; j++) {
              if(i != k) {
                 c[i][j] = c[i][j]*q - p*c[k][j];
                 b[i][j] = b[i][j]*q - p*b[k][j];
              }
           }
        }
     }
     for(i = 0; i < 3; i++){
           for(j = 0; j < 3; j++) {
             b[i][j] = b[i][j] / c[i][i];
           }
       }
  }
  public static String HillCipher(String message, String key)
  {
     int [][]keyMatrix = new int[3][3];
     getKeyMatrix(key, keyMatrix);
     int [][]messageVector = new int[3][1];
     for (int i = 0; i < 3; i++)
        messageVector[i][0] = (message.charAt(i)) % 65;
     int [][]cipherMatrix = new int[3][1];
     encrypt(cipherMatrix, keyMatrix, messageVector);
     String CipherText="";
     for (int i = 0; i < 3; i++)
        CipherText += (char)(cipherMatrix[i][0] + 65);
     return CipherText;
  }
  public static void main(String[] args)
  {
     final Scanner sc = new Scanner(System.in);
     System.out.println("Enter plaintext");
     final String message = sc.nextLine();
     System.out.println("Enter keyword");
     final String key = sc.nextLine();
     String c=HillCipher(message, key);
     System.out.println("Plain Text   :"+ message);
     System.out.println("Key          :"+ key);
     System.out.println("Cipher Text  :"+ c);
     sc.close();
  }
  }
```

Output:



```
C:\Users\Admin\Desktop\college\7th Semester\Information and network security (INS)\Lab\Programs>java hillcipher
Enter plaintext
ACT
Enter keyword
GYBNQKURP
Plain Text   :ACT
Key          :GYBNQKURP
Cipher Text  :POH
```

Learning Outcomes:

The basic Hill cipher is vulnerable to plaintext attack because it's completely linear. An opponent who intercepts the plain/ciphertext can easily set up a linear system which can be solved. So finding the solution using a standard linear algebra algorithm takes less time. Also, the key domain is less as they make use of those keys which can be inverted when used in matrix format.