INS PRACTICALS

Course Code	Course Title	Credits	Lectures /Week
USCSP502	**Information & Network Security – Practical**	**1**	**3**
1	Implementing Substitution and Transposition Ciphers: Design and implement algorithms to encrypt and decrypt messages using classical substitution and transposition techniques.		
2	RSA Encryption and Decryption: Implement the RSA algorithm for public-key encryption and decryption, and explore its properties and security considerations.		
3	Message Authentication Codes: Implement algorithms to generate and verify message authentication codes (MACs) for ensuring data integrity and authenticity.		
4	Digital Signatures: Implement digital signature algorithms such as RSA-based signatures, and verify the integrity and authenticity of digitally signed messages.		
5	Key Exchange using Diffie-Hellman: Implement the Diffie-Hellman key exchange algorithm to securely exchange keys between two entities over an insecure network.		
6	IP Security (IPsec) Configuration: Configure IPsec on network devices to provide secure communication and protect against unauthorized access and attacks.		
7	Web Security with SSL/TLS: Configure and implement secure web communication using SSL/TLS protocols, including certificate management and secure session establishment.		
8	Intrusion Detection System: Set up and configure an intrusion detection system (IDS) to monitor network traffic and detect potential security breaches or malicious activities.		
9	Malware Analysis and Detection: Analyze and identify malware samples using antivirus tools, analyze their behavior, and develop countermeasures to mitigate their impact.		
10	Firewall Configuration and Rule-based Filtering: Configure and test firewall rules to control network traffic, filter packets based on specified criteria, and protect network resources from unauthorized access.		

PRACTICAL 1.

AIM : Implementing Substitution and Transposition Ciphers:
Design and implement algorithms to encrypt and decrypt messages using classical substitution and transposition techniques

Techniques: - Caesar Cipher - Monoalphabetic Cipher

CODE: Caesar Cipher

```
import java.util.Scanner;

public class CaesarCipher
{
  public static final String ALPHABET =  "abcdefghijklmnopqrstuvwxyz";
  public static String encrypt(String plainText, int shiftKey)
   {
      plainText = plainText.toLowerCase();
      String cipherText = "";
      for (int i = 0; i < plainText.length(); i++)
      {
         int charPosition = ALPHABET.indexOf(plainText.charAt(i));
          int keyVal = (shiftKey + charPosition) % 26;
          char replaceVal = ALPHABET.charAt(keyVal);
          cipherText += replaceVal;
      }
      return cipherText;
```

```
}
public static String decrypt(String cipherText, int shiftKey)
{
    cipherText = cipherText.toLowerCase();
    String plainText = "";
    for (int i = 0 ; i < cipherText.length(); i++)
    {
        int charPosition = ALPHABET.indexOf(cipherText.charAt(i));
        int keyVal =(charPosition-shiftKey) % 26;
        if (keyVal < 0)
        {
            keyVal = ALPHABET.length() + keyVal;
        }
        char replaceVal = ALPHABET.charAt(keyVal);

         plainText += replaceVal;

    }
  return plainText;
}
public static void main(String[] args)
{
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the String for Encryption: ");
   String message = new String();
   message=sc.next();
    System.out.println(encrypt (message, 3));
    System.out.println(decrypt (encrypt (message, 3), 3));
    sc.close();
```

```
    }
}
```

OUTPUT:

```
Enter the String for Encryption:
vdnvkl
sakshi
```

(II) MONOALPHABETIC CIPHER

```
import java.util.Scanner;

public class Monoprac {
    public static char
p[]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'};
    public static char
ch[]={'Q','W','E','R','T','Y','U','I','O','P','A','S','D','F','G','H','J','K','L','Z','X','C','V','B','
N','M'};
    public static String doEncryption(String s)
    {
        char c[]=new char[(s.length())];
        for(int i=0;i<s.length();i++)
        {
            for(int j=0;j<26;j++)
            {
                if(p[j]==s.charAt(i))
```

```
            {
                c[i]=ch[j];
                    break;
            }
        }
    }
    return(new String(c));
}
public static String doDecryption(String s)
{
    char p1[]=new char[(s.length())];
    for(int i=0;i<s.length();i++)
    {
        for(int j=0;j<26;j++)
        {
            if(ch[j]==s.charAt(i))
            {
                p1[i]=p[j];
                 break;
            }
        }
    }
    return(new String(p1));
}
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter a Message: ");
    String en=doEncryption(sc.next().toLowerCase());
```

```
        System.out.println("Encrypted Message: "+en);
        System.out.println("Decrypted Message: "+doDecryption(en));
        sc.close();
    }
}
```

OUTPUT:

```
Enter a Message:
Encrypted Message: LQALIO
Decrypted Message: sakshi
```

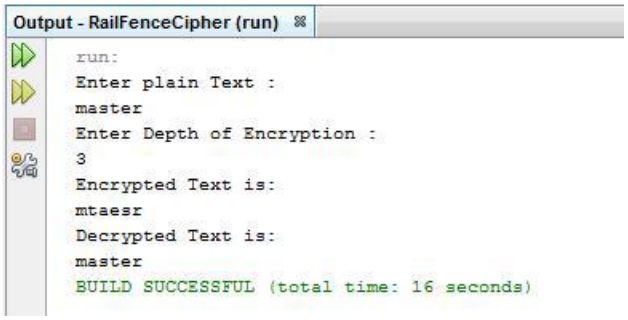
Transposition Cipher Techniques: -

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
package railfencecipher;
import java.util.Scanner;
/**
**
@author Fatema
*/
public class RailFenceCipher {
/**
* @param args the command line arguments
*/
public static void main(String[] args) {
        try
        {
                RailFenceBasic rf=new RailFenceBasic();
                Scanner sc=new Scanner(System.in);
                int depth;
                String plainText,cipherText,decryptedText;
                System.out.println("Enter plain Text : ");
                plainText=sc.nextLine();
                System.out.println("Enter Depth of Encryption : ");
```

```
                depth=sc.nextInt();
                cipherText=rf.Encryption(plainText,depth);
                System.out.println("Encrypted Text is:\n"+cipherText);
                decryptedText=rf.Decryption(cipherText,depth);
                System.out.println("Decrypted Text is:\n"+decryptedText);
        }catch(Exception e)
        {}
}
private static class RailFenceBasic {
        //int depth;

        String Encryption(String plainText, int depth)throws Exception {
        int r=depth,len=plainText.length();
        int c=len/depth;

        if(len%depth != 0){
                c = c+1;
            }

        char mat[][]=new char[r][c];
        int k=0;
        String cipherText="";
        for(int i=0;i<c;i++)
        {
                for(int j=0;j<r;j++){
                if(k!=len)
                {
                        mat[j][i]=plainText.charAt(k++);
                } else mat[j][i]='X';
                }
        }
        for(int i=0;i<r;i++)
        {
                for(int j=0;j<c;j++)
                {
                        cipherText+=mat[i][j];
                }
        }
        return cipherText;
}

String Decryption(String cipherText, int depth) {

        int r=depth,len=cipherText.length();
        int c=len/depth;
        char mat[][]=new char[r][c];
```

```
        int k=0;
        String plainText="";
        for(int i=0;i<r;i++)
        {
                for(int j=0;j<c;j++)
                {
                        mat[i][j]=cipherText.charAt(k++);
                }
        }
        for(int i=0;i<c;i++)
        {
                for(int j=0;j<r;j++)
                {
                        plainText+=mat[j][i];
                }
        }
        return plainText;

}

}}
```

OUTPUT:



Write the program to implement Simple Columnar Technique.

```
import java.util.*;
class SimpleColumn{

        public static void main(String sap[]){
                Scanner sc = new Scanner(System.in);
                System.out.print("\nEnter plaintext(enter in lower case): ");
                String message = sc.next();
```

```
                System.out.print("\nEnter key in numbers: ");
                String key = sc.next();
                int columnCount = key.length();
                int rowCount = (message.length()+columnCount)/columnCount;
                int plainText[][] = new int[rowCount][columnCount];
                int cipherText[][] = new int[rowCount][columnCount];
                System.out.print("\n-----Encryption-----\n");
cipherText = encrypt(plainText, cipherText, message, rowCount,columnCount,key);
                String ct = "";
                for(int i=0; i<columnCount; i++)
                {
                        for(int j=0; j<rowCount; j++)
                        {
                        if(cipherText[j][i] == 0)
                                ct = ct + 'x';
                        else{
                                ct = ct + (char)cipherText[j][i];
                        }
        }
        }
        System.out.print("\nCipher Text: " + ct);
        System.out.print("\n\n\n-----Decryption-----\n");
        plainText = decrypt(plainText, cipherText, ct, rowCount, columnCount, key);
        String pt = "";
        for(int i=0; i<rowCount; i++)
        {
        for(int j=0; j<columnCount; j++)
        {
        if(plainText[i][j] == 0)
        pt = pt + "";
        else{
        pt = pt + (char)plainText[i][j];
        }
        }
        }
        System.out.print("\nPlain Text: " + pt);

        System.out.println();
        }
        static int[][] encrypt(int plainText[][], int cipherText[][], String message, int
        rowCount, int columnCount, String key){
        int i,j;
        int k=0;
        for(i=0; i<rowCount; i++)
        {
        for(j=0; j<columnCount; j++)
```

```
{
if(k < message.length())
{
plainText[i][j] = (int)message.charAt(k);
k++;
}
else
{
break;
}
}
}
for(i=0; i<columnCount; i++)
{
int currentCol= ( (int)key.charAt(i) - 48 ) -1;
for(j=0; j<rowCount; j++)
{
cipherText[j][i] = plainText[j][currentCol];
}
}
System.out.print("Cipher Array(read column by column): \n");
for(i=0;i<rowCount;i++){
for(j=0;j<columnCount;j++){
System.out.print((char)cipherText[i][j]+"\t");
}
System.out.println();
}
return cipherText;
}
static int[][] decrypt(int plainText[][], int cipherText[][], String message, int
rowCount, int columnCount, String key){
int i,j;
int k=0;
for(i=0; i<columnCount; i++)
{
int currentCol= ( (int)key.charAt(i) - 48 ) -1;
for(j=0; j<rowCount; j++)
{
plainText[j][currentCol] = cipherText[j][i];
}
}
System.out.print("Plain Array(read row by row): \n");
for(i=0;i<rowCount;i++){
for(j=0;j<columnCount;j++){
System.out.print((char)plainText[i][j]+"\t");
}
```

```
System.out.println();
}
return plainText;
}
}
```

Output –

```
C:\ty>javac SimpleColumn.java
C:\ty>java SimpleColumn
Enter plaintext(enter in lower case): networksecurity
Enter key in numbers: 34152
-----Encryption-----
Cipher Array(read column by column):
t w n o e
s e r c k
i t u y r
Cipher Text: tsixwetxnruxocyxekrx
-----Decryption-----
Plain Array(read row by row):
n e t w o
r k s e c
u r i t y
Plain Text: networksecurity
```