AJAY KUMAR GARG ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

CNS Lab

(KIT 751B)

Submitted To - **Submitted By-Rahul Maurya**

Ms. Tahira Mazumder **2000270130130**

AJAY KUMAR GARG ENGINEERING COLLEGE

INDEX

7	Implement the Diffie-Hellman Key exchange mechanism using HTML and JavaScript.		
8	Implement SHA-1 algorithm in Java.		
9	Write a C/Java program to implement the BlowFish algorithm.		
10	Write a C/Java program to implement the Rijndael algorithm.		

AIM:

Program :1

Write a C program that contains a string (char pointer) with a value \Hello World’. The program should XOR each character in this string with 0 and display the result.

PROGRAM:

```
#include<stdlib.h> main()
{ char str[]="Hello World"; char str1[11]; int i,len;
len=strlen(str); for(i=0;i<len;i++)
{ str1[i]=str[i]^0; printf("%c",str1[i]);
}
printf("\n");
}
```

OUTPUT:

```
Hello World
Hello World
```

AIM:

Program :2

Write a C program that contains a string (char pointer) with a value \Hello World’. The program should AND or and XOR each character in this string with 127 and display the result.

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h> void
main() { char str[]="Hello
World"; char str1[11]; char
str2[11]=str[]; int i,len;
len = strlen(str);
for(i=0;i<len;i++)
{ str1[i] = str[i]&127;
printf("%c",str1[i]);
} printf("\n"); for(i=0;i<len;i++)
{ str3[i] = str2[i]^127;
printf("%c",str3[i]);
}
printf("\n");
}
```

Output:

```
Hello World
Hello World
Hello World
```

AIM:

Program : 3

Write a Java program to perform encryption and decryption using the following algorithms:

- Ceaser Cipher
- Vigenere Cipher
- Autokey Cipher

PROGRAM:

Ceaser Cipher

```
 import java.io.BufferedReader; import java.io.IOException; import
 java.io.InputStreamReader; import java.util.Scanner; public class
 CeaserCipher {   static Scanner sc=new
Scanner(System.in); static BufferedReader br =
new BufferedReader(new
 InputStreamReader(System.in)); public static void main(String[] args)
 throws IOException {
 // TODO code application logic here

 System.out.print("Enter any String: "); String str = br.readLine();
 System.out.print("\nEnter the Key: "); int key
 = sc.nextInt();

 String encrypted = encrypt(str, key);
 System.out.println("\nEncrypted String is: " +encrypted);
 String decrypted = decrypt(encrypted, key);
 System.out.println("\nDecrypted String is: "
 +decrypted); System.out.println("\n");
 } public static String encrypt(String str, int key)
```

AIM:

```
{
String encrypted="";
For(int i=0; i<str.length(); i++)
{
 int c=str.charAt(i); if
  (Character.isUpperCase(c)) { c = c
  + (key % 26);
 if (c > 'Z')
  }


  c = c - 26;


  else if (Character.isLowerCase(c)) { c = c + (key % 26); if

  (c > 'z')


  }
   c = c -
  26;
   encrypted += (char)
  c;
  }  return
  encrypted;
  }
     public static String decrypt(String str, int
key)
  { String decrypted = ""; for(int i
  = 0; i < str.length(); i++) { int c = str.charAt(i);
  if (Character.isUpperCase(c)) { c = c - (key % 26); if
  (c < 'A') c = c + 26;
  }

 else if (Character.isLowerCase(c)) { c = c - (key % 26);
  if (c < 'a')



  } c = c +
  26;
```

OUTPUT:

```
Enter any String: Hello World
 Enter the Key: 5
Encrypted String is: MjqqtBtwqi
Decrypted String is: Hello World
```

Vigenere Cipher

PROGRAM:

```
 class GFG  { static String generateKey(String
str, String key)
{
      int x = str.length();

      for (int i = 0; ; i++)
      {
         if (x == i)
               i = 0;
            if (key.length() == str.length())
                  break;
            key+=(key.charAt(i));
      }
      return key;
} static String cipherText(String str, String
key)
{
      String cipher_text="";

      for (int i = 0; i < str.length(); i++)
      {
            // converting in range 0-25
            int x = (str.charAt(i) + key.charAt(i)) %26;

            // convert into alphabets(ASCII)
            x += 'A';

            cipher_text+=(char)(x);
      }
      return cipher_text;
}  static String originalText(String cipher_text, String
key) {
      String orig_text="";

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

            // convert into alphabets(ASCII)
            x += 'A';
            orig_text+=(char)(x);
      }
      return orig_text;
} static String LowerToUpper(String
s)
{
   StringBuffer str =new StringBuffer(s);
   for(int i = 0; i < s.length(); i++)
      {
            if(Character.isLowerCase(s.charAt(i)))
            {
                  str.setCharAt(i, Character.toUpperCase(s.charAt(i)));
            }
      }
   s = str.toString();   return
s;
} public static void main(String[]
args)
{
      String Str = "GEEKSFORGEEKS";
      String Keyword = "AYUSH";

      String str = LowerToUpper(Str);
      String keyword = LowerToUpper(Keyword);

      String key = generateKey(str, keyword);
      String cipher_text = cipherText(str, key);

      System.out.println("Ciphertext : "
            + cipher_text + "\n");

      System.out.println("Original/Decrypted Text : "
            + originalText(cipher_text, key));
      }
}
```

OUTPUT:

```
Ciphertext : GCYCZFMLYLEIM
Original/Decrypted Text : GEEKSFORGEEKS
```

AutoKey Cipher

PROGRAM:

```
 import java.lang.*; import
java.util.*;

public class AutoKey {

      private static final String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

      public static void main(String[] args)
      {
         String msg = "HELLO";
   String key = "N";
            if (key.matches("[-+]?\\d*\\.?\\d+"))
                  key = "" + alphabet.charAt(Integer.parseInt(key));
         String enc = autoEncryption(msg, key);
            System.out.println("Plaintext : " + msg);
            System.out.println("Encrypted : " + enc);
            System.out.println("Decrypted : " + autoDecryption(enc, key));
      }

      public static String autoEncryption(String msg, String key)
      {
         int len = msg.length();
   String newKey = key.concat(msg);
         newKey = newKey.substring(0, newKey.length() - key.length());
         String encryptMsg = "";          for (int x = 0; x < len; x++) {
               int first = alphabet.indexOf(msg.charAt(x));
         int second = alphabet.indexOf(newKey.charAt(x));
               int total = (first + second) % 26;
   encryptMsg += alphabet.charAt(total);       }
            return encryptMsg;
      }

      public static String autoDecryption(String msg, String key)
      {
            String currentKey = key;
            String decryptMsg = "";

        // applying decryption algorithm            for (int x =
0; x < msg.length(); x++) {              int get1 =
alphabet.indexOf(msg.charAt(x));              int get2 =
alphabet.indexOf(currentKey.charAt(x));
              int total = (get1 - get2) % 26;
   total = (total < 0) ? total + 26 : total;
   decryptMsg += alphabet.charAt(total);
   currentKey += alphabet.charAt(total);
            }
            return decryptMsg;
      }
}   OUTPUT:
```

```
Plaintext : HELLO

Encrypted : ULPWZ

Decrypted : HELLO
```

Program : 4

AIM: Write a Java program to perform encryption and decryption using the following algorithms:

- PlayFair Cipher
- Hill Cipher

PlayFair Cipher

PROGRAM:

```
import java.io.*; import
java.util.*;

class Playfair {
String key;
String plainText;
 char[][] matrix = new char[5][5];

 public Playfair(String key, String plainText)
 {
     this.key = key.toLowerCase();
```

```
    this.plainText = plainText.toLowerCase();
}
    public void cleanPlayFairKey()
{
    LinkedHashSet<Character> set
         = new LinkedHashSet<Character>();

    String newKey = "";

    for (int i = 0; i < key.length(); i++)
         set.add(key.charAt(i));

    Iterator<Character> it = set.iterator();

    while (it.hasNext())
         newKey += (Character)it.next();

    key = newKey;
}
public void generateCipherKey()
{
    Set<Character> set = new HashSet<Character>();

    for (int i = 0; i < key.length(); i++)
    {
         if (key.charAt(i) == 'j')
              continue;
         set.add(key.charAt(i));
    }
    String tempKey = new String(key);

    for (int i = 0; i < 26; i++)
    {
         char ch = (char)(i + 97);
       if (ch == 'j')
 continue;

       if (!set.contains(ch))
 tempKey += ch;
    }
 for (int i = 0, idx = 0; i < 5; i++)
 for (int j = 0; j < 5; j++)
              matrix[i][j] = tempKey.charAt(idx++);

    System.out.println("Playfair Cipher Key Matrix:");
```

```
     for (int i = 0; i < 5; i++)
          System.out.println(Arrays.toString(matrix[i]));
 }
 public String formatPlainText()
 {
  String message = "";       int
len = plainText.length();

     for (int i = 0; i < len; i++)
     {
       if (plainText.charAt(i) == 'j')
             message += 'i';
  else
                message += plainText.charAt(i);
     }
     for (int i = 0; i < message.length(); i += 2)
     {
          if (message.charAt(i) == message.charAt(i + 1))
                message = message.substring(0, i + 1) + 'x'
                          + message.substring(i + 1);
     }
          if (len % 2 == 1)
          message += 'x'; // dummy character
  return message;
 }
public String[] formPairs(String message)
 {
  int len = message.length();
  String[] pairs = new String[len / 2];
  for (int i = 0, cnt = 0; i < len / 2; i++)
  pairs[i] = message.substring(cnt, cnt += 2);

     return pairs;
 }
 public int[] getCharPos(char ch)
 {
     int[] keyPos = new int[2];

     for (int i = 0; i < 5; i++)
     {
          for (int j = 0; j < 5; j++)
          {
                if (matrix[i][j] == ch)
                {
                   keyPos[0] = i;
        keyPos[1] = j;
  break;
                }
          }
     }
     return keyPos;
 }

 public String encryptMessage()
 {
     String message = formatPlainText();
     String[] msgPairs = formPairs(message);
     String encText = "";

     for (int i = 0; i < msgPairs.length; i++)
     {
        char ch1 = msgPairs[i].charAt(0);
  char ch2 = msgPairs[i].charAt(1);
  int[] ch1Pos = getCharPos(ch1);
  int[] ch2Pos = getCharPos(ch2);             if
(ch1Pos[0] == ch2Pos[0]) {
  ch1Pos[1] = (ch1Pos[1] + 1) % 5;
  ch2Pos[1] = (ch2Pos[1] + 1) % 5;
          }
                else if (ch1Pos[1] == ch2Pos[1])
          {
         ch1Pos[0] = (ch1Pos[0] + 1) % 5;
            ch2Pos[0] = (ch2Pos[0] + 1) % 5;
          }
          else {
             int temp = ch1Pos[1];
  ch1Pos[1] = ch2Pos[1];
                ch2Pos[1] = temp;
          }
          encText = encText + matrix[ch1Pos[0]][ch1Pos[1]]
                     + matrix[ch2Pos[0]][ch2Pos[1]];
     }

     return encText;
 }
}
```

```
 public class GFG {
  public static void main(String[] args)
  {
      System.out.println("Example-1\n");

      String key1 = "Problem";
      String plainText1 = "Playfair";

      System.out.println("Key: " + key1);
      System.out.println("PlainText: " + plainText1);

   Playfair pfc1 = new Playfair(key1, plainText1);
   pfc1.cleanPlayFairKey();   pfc1.generateCipherKey();

      String encText1 = pfc1.encryptMessage();
      System.out.println("Cipher Text is: " + encText1);

      System.out.println("\nExample-2\n");

      String key2 = "Problem";
      String plainText2 = "Hello";

      System.out.println("Key: " + key2);
      System.out.println("PlainText: " + plainText2);

   Playfair pfc2 = new Playfair(key2, plainText2);
   pfc2.cleanPlayFairKey();   pfc2.generateCipherKey();

      String encText2 = pfc2.encryptMessage();
      System.out.println("Cipher Text is: " + encText2);
 }
 }
```

```
 OUTPUT:  Example-1

Key: Problem

PlainText: Playfair

Playfair Cipher Key Matrix:

[p, r, o, b, l]

[e, m, a, c, d]
```

```
[f, g, h, i, k]
[n, q, s, t, u]
[v, w, x, y, z]
Cipher Text is: rpcxhegb

Example-2


Key: Problem
PlainText: Hello
Playfair Cipher Key Matrix:
[p, r, o, b, l]
[e, m, a, c, d]
[f, g, h, i, k]
[n, q, s, t, u]
[v, w, x, y, z]
Cipher Text is: faozpb
```

Hill Cipher

```
PROGRAM : class GFG  { 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++;
                }
        }
} 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;
               }
        }
} static void HillCipher(String message, String
key)
{
   // Get key matrix from the key string     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);
        System.out.print(" Ciphertext:" + CipherText);
```

```
} public static void main(String[]
args)
{
        String message = "ACT";
        String key = "GYBNQKURP";

        HillCipher(message, key);
        }
}
```

OUTPUT:

```
Ciphertext: POH
```

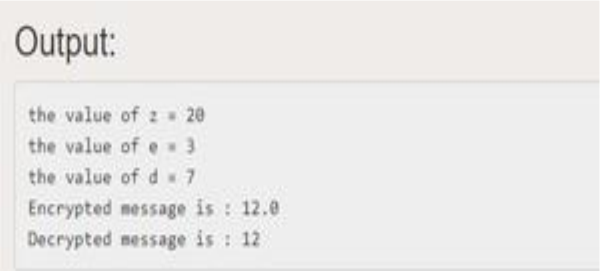
Program : 5

AIM: Implementation of RSA algorithm using java.

PROGRAM:

```
 import java.math.*; import
 java.util.*;
  class RSA {  public static void
 main(String args[])
  {
  int p, q, n, z, d = 0, e, i;
 int msg = 12;  double c;
  BigInteger msgback;
p = 3;
  q = 11;  n = p * q;  z =
 (p - 1) * (q - 1);
  System.out.println("the value of z = " + z);
   for (e = 2; e < z; e++) {
  if (gcd(e, z) == 1) {
              break;
          }
  }
  System.out.println("the value of e = " + e);
 for (i = 0; i <= 9; i++) {       int x = 1 + (i
 * z);      if (x % e == 0) {         d = x /
 e;         break;
          }
  }
  System.out.println("the value of d = " + d);  c
 = (Math.pow(msg, e)) % n;
  System.out.println("Encrypted message is : " + c);
  BigInteger N = BigInteger.valueOf(n);
  BigInteger C = BigDecimal.valueOf(c).toBigInteger();  msgback
 = (C.pow(d)).mod(N);
  System.out.println("Decrypted message is : "
                          + msgback);
  }
   static int gcd(int e, int
 z)
  {
  if (e == 0)
        return z;  else
  return gcd(z % e, e);
  }
 }
```

OUTPUT:



```
the value of z = 20
the value of e = 3
the value of d = 7
Encrypted message is : 12.0
Decrypted message is : 12
```

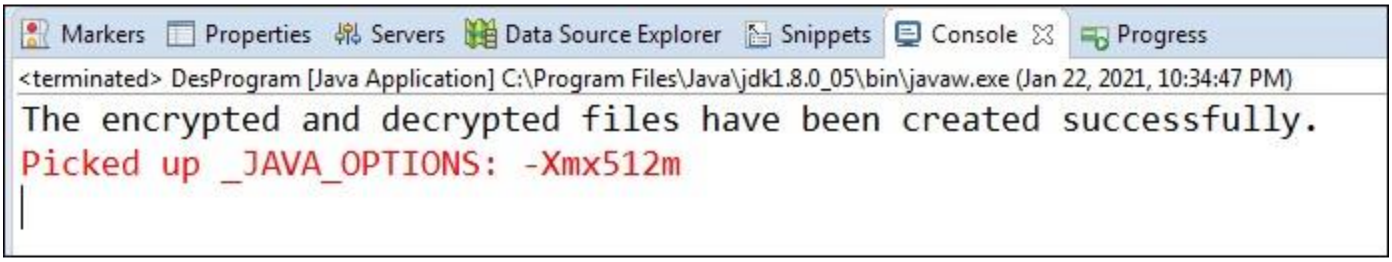
Program : 6

AIM: Write a java program to implement DES algorithm.

Program:

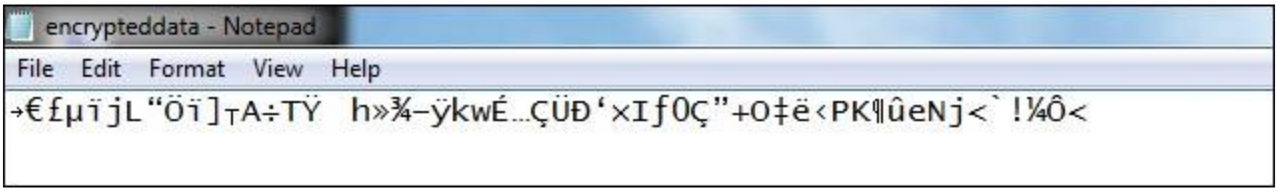
```
import java.io.FileInputStream;   import
java.io.FileOutputStream;   import java.io.IOException;
import java.io.InputStream;   import
java.io.OutputStream;   import
java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;   import
java.security.NoSuchAlgorithmException;   import
java.security.spec.AlgorithmParameterSpec;   import
javax.crypto.Cipher;   import
javax.crypto.CipherInputStream;   import
javax.crypto.CipherOutputStream;   import
javax.crypto.KeyGenerator;   import
javax.crypto.NoSuchPaddingException;   import
javax.crypto.SecretKey;   import
javax.crypto.spec.IvParameterSpec;   public class
DesProgram
{
private static Cipher encrypt;   private static Cipher decrypt;   private static
final byte[] initialization_vector = { 22, 33, 11, 44, 55, 99, 66, 77 };   public
static void main(String[] args)
{
String textFile = "C:/Users/Anubhav/Desktop/DemoData.txt";
String encryptedData = "C:/Users/Anubhav/Desktop/encrypteddata.txt";
String decryptedData = "C:/Users/Anubhav/Desktop/decrypteddata.txt";   try
{
SecretKey scrtkey = KeyGenerator.getInstance("DES").generateKey();
AlgorithmParameterSpec aps = new IvParameterSpec(initialization_vector);
encrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, scrtkey, aps);
decrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, scrtkey, aps);   encryption(new
FileInputStream(textFile), new FileOutputStream(encryptedData));   decryption(new
FileInputStream(encryptedData), new FileOutputStream(decryptedData));
System.out.println("The encrypted and decrypted files have been created successfully.");
}
catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | InvalidAlgorithmParameterException | IOException e)
{
e.printStackTrace();
}
}
private static void encryption(InputStream input, OutputStream output)
throws IOException
{
1. output = new CipherOutputStream(output, encrypt);      writeBytes(input,
output);
}
private static void decryption(InputStream input, OutputStream output)
throws IOException
{
input = new CipherInputStream(input, decrypt);   writeBytes(input,
output);
}
private static void writeBytes(InputStream input, OutputStream output)
throws IOException
{
byte[] writeBuffer = new byte[512];   int
readBytes = 0;   while ((readBytes =
input.read(writeBuffer)) >= 0)
{
output.write(writeBuffer, 0, readBytes);
}
output.close();   input.close();
}
}
```

OUTPUT :



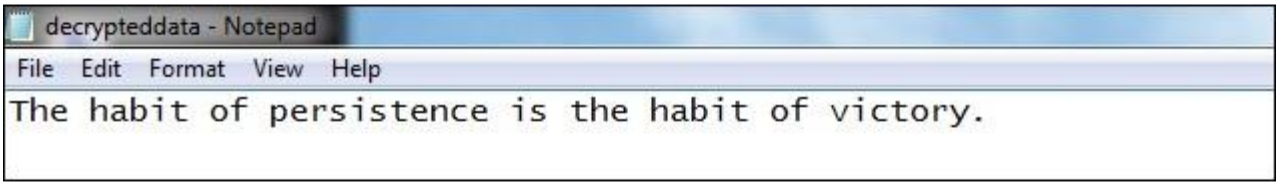
```
<terminated> DesProgram [Java Application] C:\Program Files\Java\jdk1.8.0_05\bin\javaw.exe (Jan 22, 2021, 10:34:47 PM)
The encrypted and decrypted files have been created successfully.
Picked up _JAVA_OPTIONS: -Xmx512m
```

encrypteddata.txt



```
→€£µïjL“Öï]┬A÷TŸ  h»¾–ÿkwÉ…ÇÜÐ‘×IƒOÇ”+O‡ë‹PK¶ûeNj<`!¼Ô<
```

deecrypteddata.txt



```
The habit of persistence is the habit of victory.
```