THIAGARAJAR COLLEGE OF ENGINEERING

(Govt. Aided & Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF INFORMATION TECHNOLOGY

18IT580 – INFORMATION SECURITY LAB

[LAB EXPERIMENTS - RECORD]

THIAGARAJAR COLLEGE OF ENGINEERING

(Govt. Aided & Autonomous Institution, Affiliated to Anna University, Chennai)

B.Tech INFORMATION TECHNOLOGY

18IT580 – INFORMATION SECURITY LAB

[LAB EXPERIMENTS - RECORD]

REG. NO : 18IT116 **ROLL NO : 61854**

NAME : YAJITH VISHWA S

BRANCH : INFORMATION TECHNOLOGY

THIAGARAJAR COLLEGE OF ENGINEERING

(Govt. Aided & Autonomous Institution, Affiliated to Anna University, Chennai)

This is certified to be the Bonafide Record of 18IT580 – Information Security Lab Practical work done by Mr. YAJITH VISHWA S (18IT116) for the Fifth Semester – B.Tech Information Technology Degree Course, Department of Information Technology, Thiagarajar college of Engineering, Madurai - 625015, during the academic year 2020 – 2021.

Staff-In Charge

Register No: 18IT116 **Roll No: 61854**

Submitted for the Terminal Lab Examination held on _____

EXTERNAL EXAMINER **INTERNAL EXAMINER**

TABLE OF CONTENTS

Ex.No.1	**IMPLEMENTATION AND CRYPTANALYSIS OF CAESAR CIPHER**

AIM:

To implement

- Encryption
- Decryption
- Brute force Attack
- Frequency Analysis attack

in Caesar cipher using Java.

THEORY:

Encryption and Decryption in Caesar Cipher:

The simplest monoalphabetic cipher is the additive cipher. This cipher is sometimes called a shift cipher and sometimes a Caesar cipher

Brute Force Attack in Caesar Cipher:

The technique of trying every possible decryption key is called a **brute-force attack**

Frequency Analysis Attack in Caesar Cipher:

Frequency Analysis attack is performed based on the **frequency** of letters or groups of letters in a ciphertext. The letter 'e' is the most frequent (about 12% of all letters in the book), followed by 't', and so on

Frequency of Character in English:

Letter	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*
E	12.7	H	6.1	W	2.3	K	0.08
T	9.1	R	6.0	F	2.2	J	0.02
A	8.2	D	4.3	G	2.0	Q	0.01
O	7.5	L	4.0	Y	2.0	X	0.01
I	7.0	C	2.8	P	1.9	Z	0.01
N	6.7	U	2.8	B	1.5		
S	6.3	M	2.4	V	1.0		

ALGORITHM:

1. Start.
2. Define 5 methods : encrypt,decrypt,maxoccuringchar,bruteforce, frequencyanalysis.
3. Main method receives input and key for encryption and decryption.
4. Encryption is done shifting characters using the key and vice versa for decryption.
5. The position equivalence for the plain text is added with key value and the corresponding position equivalent character is displayed as cipher text.
6. The key is subtracted from the position equivalence of the cipher text and the corresponding position equivalent character is displayed as plain text.
7. Brute force : The input is cipher text to be decrypted and the decryption method is used with all possible keys and displays all output.
8. Frequency analysis: The input is cipher text and the most occurring cipher letter is found. Its position equivalence is subtracted from the position equivalence of the most occurring letter in alphabets to get the key. Using the key, the plain text is obtained , if it is not meaningful then the next most occurring letter in alphabet is taken for consideration.
9. End.

CODING:

Ceaser Cipher:

```
package com.yajith.cipher;

import java.util.Dictionary;
import java.util.Hashtable;
import java.util.Scanner;


public class Main {
    private static String cipher;
    private static int choice,j;
    public static void main(String[] args) {
       Scanner s=new Scanner(System.in);
       System.out.println("1.encrypt\n2.decrypt");
       choice=s.nextInt();
       System.out.println("Enter the key");
       j=s.nextInt();
       if(choice==1)
        {
```

```
            Scanner s1=new Scanner(System.in);
            System.out.println("Enter The Plain Text:");
            cipher=s1.nextLine();
            cipher=cipher.toLowerCase();
                for (int i = 0; i < cipher.length(); i++) {
                    int a = (int) cipher.charAt(i);
                    if (a == 32) {
                        System.out.print(" ");
                        continue;
                    }
                    char c = (char) ( (( (a + j) - 97) % 26) + 97);
                    c = Character.toUpperCase(c);
                    System.out.print(c);
                }
                System.out.println();
         }
        else if(choice==2) {
            Scanner s2=new Scanner(System.in);
            System.out.println("Enter The Cipher Text:");
            cipher=s2.nextLine();
            cipher=cipher.toLowerCase();
            System.out.println("The Plain Text are");
                for (int i = 0; i < cipher.length(); i++) {
                    int a = (int) cipher.charAt(i);
                    if (a == 32) {
                        System.out.print(" ");
                        continue;
                    }
                    char c = (char) ( ( ((a - j) - 97) % 26) + 97);
                    System.out.print(c);
                }
                System.out.println();
            }
        else
         {
            System.out.println("Wrong Choice");
         }
    }
}
```

Brute Force Attack:

```
package com.yajith.cipher;

import java.util.Dictionary;
import java.util.Hashtable;
import java.util.Scanner;


public class Main {
    private static String cipher;
    private static int choice;
    public static void main(String[] args) {
       Scanner s=new Scanner(System.in);
       System.out.println("1.encrypt\n2.decrypt");
       choice=s.nextInt();
       if(choice==1)
        {
```

```
            Scanner s1=new Scanner(System.in);
            System.out.println("Enter The Plain Text:");
            cipher=s1.nextLine();
            cipher=cipher.toLowerCase();
            for(int j=0;j<26;j++) {
                System.out.print(j+" ");
                for (int i = 0; i < cipher.length(); i++) {
                    int a = (int) cipher.charAt(i);
                    if (a == 32) {
                        System.out.print(" ");
                        continue;
                    }
                    char c = (char) ( (( (a + j) - 97) % 26) + 97);
                    c = Character.toUpperCase(c);
                    System.out.print(c);
                }
                System.out.println();
            }
        }
        else if(choice==2) {
            Scanner s2=new Scanner(System.in);
            System.out.println("Enter The Cipher Text:");
            cipher=s2.nextLine();
            cipher=cipher.toLowerCase();
            System.out.println("The Plain Text are");
           for(int j=0;j<26;j++) {
               System.out.print(j+" ");
                for (int i = 0; i < cipher.length(); i++) {
                    int a = (int) cipher.charAt(i);
                    if (a == 32) {
                        System.out.print(" ");
                        continue;
                    }
                    char c = (char) ( ( ((a - j) - 97) % 26) + 97);
                    System.out.print(c);
                }
                System.out.println();

            }
        }
        else
        {
            System.out.println("Wrong Choice");
        }
    }
}
```

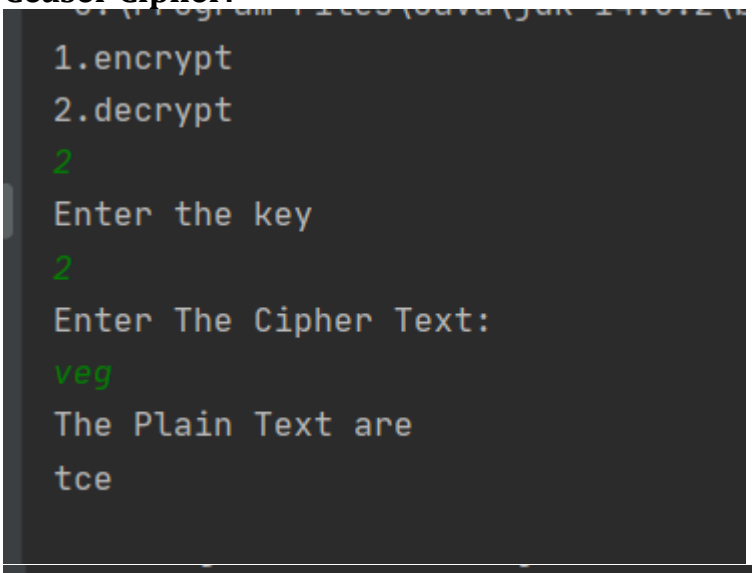
Frequency Analysis:

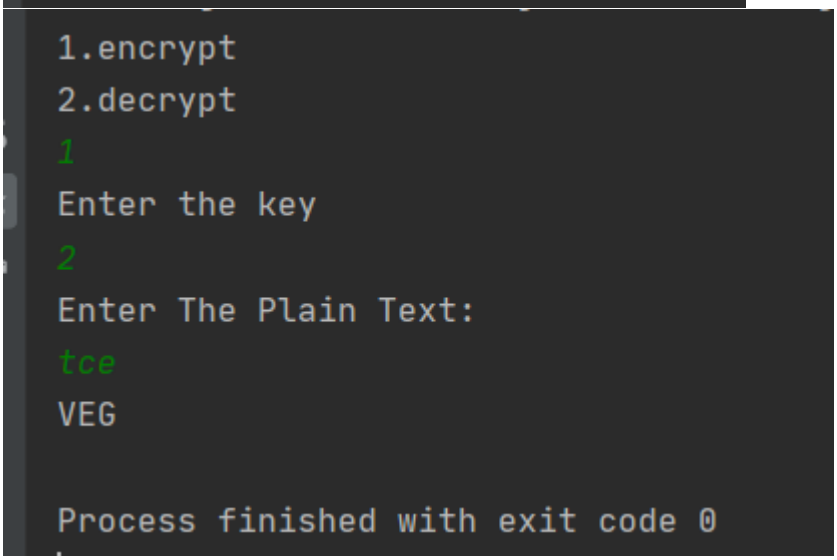
```
package com.yajith.frequency;

import java.util.Scanner;

public class Main {
    private static String input;
    public static void main(String[] args) {
        Scanner scanner=new Scanner(System.in);
```

```
        input=scanner.nextLine();
        String cipherText = "";
        String ALPHABET = "abcdefghijklmnopqrstuvwxyz";
        cipherText = input;
        char freq = getMaxOccuringChar(cipherText);
        System.out.println("The most occurring cipher letter is:"+ freq);
        System.out.println("Enter the plain letter corresponding to " + freq + ": ");
        char p = scanner.next().charAt(0);
        int cpos = ALPHABET.indexOf(freq);
        int ppos = ALPHABET.indexOf(p);
        int keyVal = (cpos - ppos) % 26;
        System.out.println("Key value: " + (keyVal*-1));
        decrypt(cipherText);
    }
    static void decrypt(String text){
        System.out.println("Key: ");
        int key = new Scanner(System.in).nextInt();
        char chars[] = text.toCharArray();
        int i = 0;
        while(i<chars.length){
            chars[i] = Character.toLowerCase(chars[i]);
            int val = ((int)chars[i]-key);
            val = (val >= 97)?val:121-(97-val);
            chars[i] = (char)val;
            System.out.print(Character.toLowerCase(chars[i]));
            i++;
        }
        System.out.println('\n');
    }
    static char getMaxOccuringChar(String str) {
        // Create array to keep the count of individual
        // characters and initialize the array as 0
        int count[] = new int[256];

        // Construct character count array from the input
        // string.
        int len = str.length();
        for (int i = 0; i < len; i++)
            count[str.charAt(i)]++;

        int max = -1;  // Initialize max count
        char result = ' ';   // Initialize result

        // Traversing through the string and maintaining
        // the count of each character
        for (int i = 0; i < len; i++) {
            if (max < count[str.charAt(i)]) {
                max = count[str.charAt(i)];
                result = str.charAt(i);
            }
        }

        return result;
    }

}
```

SCREEN SHOTS:

Ceaser Cipher:





Brute Force attack:

```
1.encrypt
2.decrypt
1
Enter The Plain Text:
attackatonce
0 ATTACKATONCE
1 BUUBDLBUPODF
2 CVVCEMCVQPEG
3 DWWDFNDWRQFH
4 EXXEGOEXSRGI
5 FYYFHPFYTSHJ
6 GZZGIQGZUTIK
7 HAAHJRHAVUJL
8 IBBIKSIBWVKM
9 JCCJLTJCXWLN
10 KDDKMUKDYXMO
11 LEELNVLEZYNP
12 MFFMOWMFAZOQ
13 NGGNPXNGBAPR
14 OHHOQYOHCBQS
15 PIIPRZPIDCRT
16 QJJQSAQJEDSU
17 RKKRTBRKFETV
18 SLLSUCSLGFUW
19 TMMTVDTMHGVX
20 UNNUWEUNIHWY
21 VOOVXFVOJIXZ
22 WPPWYGWPKJYA
23 XQQXZHXQLKZB
24 YRRYAIYRMLAC
25 ZSSZBJZSNMBD
```

```
2.decrypt
2
Enter The Cipher Text:
EXXEGOEXSRGI
The Plain Text are
0 exxegoexsrgi
1 dwwdfndwrqfh
2 cvvcemcvqpeg
3 buubdlbupodf
4 attackatonce
5 `ss`bj`snmbd
6 _rr_ai_rmlac
7 ^qq^`h^qlk`b
8 ]pp]_g]pkj_a
9 \oo\^f\oji^`
10 [nn[]e[nih]_
11 ZmmZ\dZmhg\^
12 YllY[cYlgf[]
13 XkkXZbXkfeZ\
14 WjjWYaWjedY[
15 ViiVX`VidcXZ
16 UhhUW_UhcbWY
17 TggTV^TgbaVX
18 SffSU]Sfa`UW
19 ReeRT\Re`_TV
20 QddQS[Qd_^SU
21 PccPRZPc^]RT
22 ObbOQYOb]\QS
23 NaaNPXNa\[PR
24 M``MOWM`[ZOQ
25 L__LNVL_ZYNP
```

Frequency Analysis:

```
Enter the cipher text to be decrypted:
ubzr
The most recurring cipher letter is:u
Enter the most recurring cipher letter
u
Enter the plain letter corresponding to u:
e
Key value: 16
The plain text is : eljb
```

RESULT:

Thus the programs for

- Encryption
- Decryption
- Bruteforce Attack
- Frequency Analysis attack

in Caesar cipher are implemented in Java and the results are verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Uniqueness of the Code	7	
Use of Comment lines and standard coding practices	3	
Viva	10	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.2	**IMPLEMENTATION AND CRYPTANALYSIS OF HILL CIPHER**

AIM:

To implement

- Encryption
- Decryption
- Known Plain test – cipher text attack

in Hill cipher using Java.

THEORY:

Encryption

To encrypt a message using the Hill Cipher we must first turn our keyword into a key matrix (a 2 x 2 matrix for working with digraphs, a 3 x 3 matrix for working with trigraphs, etc). We also turn the plaintext into digraphs (or trigraphs) and each of these into a column vector.

Decryption

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).

Constraints for selecting Key Matrix in Hill Cipher

If we work modulo 26 as above, the determinant must be nonzero, and must not be divisible by 2 or 13. If the determinant is 0, or has common factors with the modular base, then the matrix cannot be used in the Hill cipher, and another matrix must be chosen (otherwise it will not be possible to decrypt).

Steps for calculation of inverse of Key Matrix in Modulo arithmetic

1. First, compute the determinant of the matrix,detA. If detA is coprime to m, then you can be sure that A is invertible mod m.
2. Next, compute the matrix of cofactors of A, call this B.
3. The matrix (detA)−1×B is an inverse to A modulo m.

Euclidean and Extended Euclidean algorithm

The Euclidean algorithm is a way to find the greatest common divisor of two positive integers, a and b

Extended Euclidean algorithm also refers to a very similar algorithm for computing the polynomial greatest common divisor and the coefficients of Bézout's identity of two univariate polynomials. The extended Euclidean algorithm is particularly useful when a and b are coprime.

Known Plaintext and Cipher text attack with an example

During known-plaintext attacks, the attacker has an access to the ciphertext and its corresponding plaintext. His goal is to guess the secret key (or a number of secret keys) or to develop an algorithm which would allow him to decrypt any further messages. This gives the attacker much bigger possibilities to break the cipher than just by performing ciphertext only attacks. However, he is no able to actively provide customized data or secret keys which would be processed by the cipher.

ALGORITHM:

Encryption:

Step1: Start
Step 2: Get the order of the key matrix.
Step 3: Get the elements of the key matrix.
Step 4: Get the Plain text to be encoded
Step 5:Perform number encoding for given plain text and give it as input for matrix P.
Step 6:Perform matrix multiplication between K (key) and P.
Step 7:Do modulo m with all element of matrix.
Step 8:Perform character encoding and it results in cipher text.

Decryption:

Step1: Start
Step 2: Get the order of the key matrix.
Step 3: Get the elements of the key matrix.
Step 4: Get the Cipher text to be decoded
Step 5:Perform number encoding for given cipher text and give it as input for matrix C.
Step 6:Find determinant of the key matrix and then do modulo.
Step 7:Next find the determinant inverse and calculate the adjoint of the key (K) matrix.
Step 8:Calculate inverse of key (K) matrix.
Step 9:Perform matrix multiplication between inverse of key (K) and C.Step 10:Do modulo m with all element of matrix.
Step 11:Perform character encoding and it results in plain text.

Known Plaintext attack and Cipher text attack

Step 1:Get plain text and cipher text as inputs
Step2:Perform number encoding.
Step 3:Get their order of matrix respectively.
Step 4:Perform matrix multiplication between inverse of C and P.
Step 5:Do modulo m with all element of matrix.
Step 6:The resultant matrix is the key inverse.

DESIGN

List of classes:

1. HillCipher
2. Main

List of methods:

1. Determinant(array,size) - to find the determinant of matrix.
2. inverse(key,size) -to find inverse of the matrix.
3. chartonum(pt, key) - to convert characters to it'sasciiform
4. decode(array, order) - to convert numbers to characters.
5. getCofactor(array , order) - to get the cofactor of the respective matrix.
6. adjoint(array,size) - to calculate adjoint of the matrix.
7. Determinant(array,size) - to find the determinant of matrix.
8. matmul(array) - to do matrix multiplication between two matrices.
9. inverse(key,size) -to find inverse of the matrix.
10. encrypt(array,size) - to encrypt plain text to cipher text.
11. decrpypt(array,size) - to decrypt cipher text to plain text.
12. knownptct(array,size) - to perform known pt ct.

TASK DISTRIBUTION

Reg No	Name	Task	Support from Group Members	Learning
18IT079	G.L. Sankara Subramaniyan	Determinant, Inverse of matrix, encrypt, decrypt	Cross validated code	Out of bound exception avoidance and findings
18IT116	S. Yajith Vishwa	Matrix Multiplication, cofactor, adjoint	Cross validated code	Splitting matrix for adjoint calculation

MODULE WISE CODING AND UNIT TESTING

1)Determinant:

```
public static void main(String args[])
{
static int determinant(int mat[][], int n)
   {
int D = 0;
if (n == 1)
return mat[0][0];
int temp[][] = new int[n][n];
int sign = 1;
for (int f = 0; f < n; f++)
      {
getCofactor(mat, temp, 0, f, n);
          D += sign * mat[0][f]
            * determinant(temp, n - 1);
sign = -sign; }
      D=D%26;
if(D<0)
      {
        D=26+D;
return D;
       }
return D;

}
```

2)Inverse of matrix

```
int[][] inverse(int adjoint1[][],int g,int N)
{
int k[][]=new int[N][N];
for(int i=0;i<N;i++)
  {
for(int j=0;j<N;j++)
      {
k[i][j]=(g*adjoint1[i][j])%26;

        }
     }
for(int i=0;i<N;i++)
  {
for(int j=0;j<N;j++)
```

```
    {
if(k[i][j]<0)
        {
k[i][j]=26+k[i][j];
        }
      }
}
return k;
}
```

INTEGRATED CODE AND TESTING

```
import java.util.*;
class Hillcipher
{
  static final String ALPHABET="abcdefghijklmnopqrstuvwxyz";
  int[][] numencode(String ptext,int key)
  {
    int row=0;
    int col=0;
    int res=0;
    int extracol=0;
    int k=0;
    int i=0;
    int j=0;
    int col1=0;
    String alpha="abcdefghijklmnopqrstuvwxyz";
    int len=ptext.length();
    int[] enc=new int[100];
    for(i=0;i<len;i++)
    {
      enc[i]=alpha.indexOf(ptext.toLowerCase().charAt(i));
    }
    if(len%key==0)
    {
      row=key;
      col1=len/key;
    }
    else
```

```
    {
      row=key;
      res=len%key;
      res=key-res;
      col=res+len;
      col1=col/key;
      for(i=len;i<col;i++)
      {
        enc[i]=23;
      }
    }
    int[][] ans=new int[key][col1];
    k=0;
    for(i=0;i<col1;i++)
    {
      for(j=0;j<key;j++)
      {
        ans[j][i]=enc[k];
        k++;
      }
    }
    for(i=0;i<key;i++)
    {
      for(j=0;j<col1;j++)
      {
        System.out.print(ans[i][j]);
        System.out.print("\t");
      }
      System.out.println();
    }
    return ans;
  }
  void decode(int c[][],int len)
  {
    String p="";

    int k=c.length;
    int q=len/k;
    for(int j=0;j<=q;j++)
    {
      for(int i=0;i<k;i++)
      {
        p+=ALPHABET.charAt(c[i][j]);
```

```
        }
      }
      System.out.print(p);
   }
   void decode1(int c[][],int len)
   {
      String p="";
      int k=c.length;
      int q=len/k;
      for(int j=0;j<q;j++)
      {
         for(int i=0;i<k;i++)
         {
            p+=ALPHABET.charAt(c[i][j]);

         }
      }
      System.out.println(p.substring(0,len-1));
   }
   int[][] matmul (int key[][],int a[][])
   {
      int row=key.length;
      int col=a[0].length;
      int col1=key[0].length;
      int [][] c= new int[row][col];
      for (int i = 0; i< row; i++)
      {
         for (int j = 0; j < col; j++)
         {
            for (int k = 0; k < row; k++)
            {
               c[i][j] = c[i][j] + key[i][k] * a[k][j];
            }
         }
      }
      return c;
   }
   static int determinant(int mat[][], int n)
   {
      int D = 0; // Initialize result
      if (n == 1)
         return mat[0][0];
```

```
                                // cofactors
    int temp[][] = new int[n][n];
    int sign = 1;
    for (int f = 0; f < n; f++)
    {
      getCofactor(mat, temp, 0, f, n);
      D += sign * mat[0][f]
           * determinant(temp, n - 1);
      sign = -sign; }
    D=D%26;
    if(D<0)
    {
      D=26+D;
      return D;
    }
    return D;
  }
  int mulinverse(int a,int m)
  {
    a = a % m;
    for (int x = 1; x < m; x++)
      if ((a * x) % m == 1)
        return x;
    return 1;
  }
  int[][] inverse(int adjoint1[][],int g,int N)
  {
    int k[][]=new int[N][N];
    for(int i=0;i<N;i++)
    {
      for(int j=0;j<N;j++)
      {
        k[i][j]=(g*adjoint1[i][j])%26;

      }
    }

    for(int i=0;i<N;i++)
    {
      for(int j=0;j<N;j++)
      {
        if(k[i][j]<0)
        {
```

```
            k[i][j]=26+k[i][j];
          }
        }
      }
      return k;
    }
    int[][] adjoint(int mat[][],int N)
    {
      int si = 1;
      int[][]temp=new int[N][N];
      int[][]adjoint=new int[N][N];
      for (int i=0; i<N; i++)
      {
        for (int j=0; j<N; j++)
        {
                                    //adjoint
          getCofactor(mat, temp, i, j, N);
          si = ((i+j)%2==0)? 1: -1;
          adjoint[j][i] = (si)*(determinant(temp, N-1));

        }
      }
      return adjoint;
    }
    static void getCofactor(int mat[][],int temp[][], int p, int q, int n)
    {
      int i = 0, j = 0;
      for (int row = 0; row < n; row++)
      {
        for (int col = 0; col < n; col++)
        {
          if (row != p && col != q)
          {
            temp[i][j++] = mat[row][col];
            if (j == n - 1)
            {
              j = 0;
              i++;
            }
          }
        }
      }
    }
```

```
int[][] encrypt(int key[][] ,int a[][] ,int N)
{
   int row=a.length;
   int col=a[0].length;
   int mat1[][]=new int[N][col];
   char [][] replace=new char[N][col];
   mat1 = matmul(key,a);
   System.out.println("The encrypted matrix is: ");
   for (int i = 0; i< N; i++)
   {
      System.out.println();
      for (int j = 0; j < col; j++)
      {
         mat1[i][j]=(mat1[i][j])%26;
         System.out.print("\t"+mat1[i][j]);
      }
   }
   return mat1;
}


// decryption
int[][] decrypt(int key[][], int b[][],int N)
{
   int m=26;
   int row=b.length;
   int col=b[0].length;
   int[][] adjoint1=new int [N][N];
   int [][] keyinverse=new int[N][N];
   int[][] mat2=new int[row][col];
   int l=determinant(key, N);
   System.out.println("Determinant " + "of the matrix is : "+ l );
   int g=mulinverse(l,m);
   System.out.println("Inverse of determinant: "+g);
   adjoint1=adjoint(key,N);
   keyinverse=inverse(adjoint1,g,N);
   mat2=matmul(keyinverse,b);
   System.out.println("Decrypted matrix: ");
   for(int i=0;i<row;i++)
   {
      System.out.println();
      for(int j=0;j<col;j++)
```

```
        {
            mat2[i][j]=(mat2[i][j])%26;
            System.out.print("\t"+mat2[i][j]);
        }
    }
    return mat2;
}


// Known plain text – cipher text attack
int[][] knownptct(int a[][],int b[][],int N2)
{
    int row=a.length;
    int col=b[0].length;
    int m=26;
    int [][] adjoint1=new int[row][col];
    int [][] keyinverse1=new int[row][col];
    int [][] mat2=new int[row][col];
    int l=determinant(b, row);
    System.out.println("\n");
    System.out.println("\n");
    System.out.println("\n");
    System.out.println("\nDeterminant " + "of the matrix is : "+ l );
    int g=mulinverse(l,m);
    System.out.println("Inverse of determinant: "+g);
    adjoint1=adjoint(b,row);
    keyinverse1=inverse(adjoint1,g,row);
    mat2=matmul(a,keyinverse1);
    System.out.println("Key matrix: ");
    for(int i=0;i<N2;i++)
    {
        System.out.println();
        for(int j=0;j<N2;j++)
        {
            mat2[i][j]=(mat2[i][j])%26;
            System.out.print("\t"+mat2[i][j]);
        }
    }
    return mat2;
}
}
```

```
class Execution{

   public static void main (String[] args)
   {
      int m=26;
      Hillcipher h = new Hillcipher();
      Scanner input = new Scanner(System.in);
      Scanner sc= new Scanner(System.in);
      System.out.println("Encryption");
      System.out.println("Enter the order of key: ");
      int o=input.nextInt();
      int[][] key=new int[o][o];
      System.out.println("Enter the elements of key matrix "+"("+o*o+")"+":");
      for(int i=0;i<o;i++)
      {
         for(int j=0;j<o;j++)
         {
            key[i][j]=input.nextInt();
         }
      }
      System.out.println("Enter the plain text to encode :");
      String pt=sc.nextLine();
      int len=pt.length();
      int [][]a=h.numencode(pt,o);
      int[][]mat1=h.encrypt(key,a,o);
      System.out.println("\nThe encrypted cipher text is: ");
      h.decode(mat1,len);

      System.out.println("\nDECRYPTION");
      System.out.println("Enter the order of key matrix: ");
      int o1=input.nextInt();
      int[][] key1=new int[o1][o1];
      System.out.println("\nEnter elements of key matrix:"+"("+o1*o1+")"+":");
      for(int i=0;i<o1;i++)
      {
         for(int j=0;j<o1;j++)
         {
            key1[i][j]=input.nextInt();
         }
      }
      System.out.println("Enter the cipher text to decode: ");
      String pt1=sc.nextLine();
      int len1=pt1.length();
```

```
        int[][]a1=h.numencode(pt1,o1);
        int mat2[][]=h.decrypt(key1,a1,o1);
        System.out.println("\nThe decrypted plain text is: ");
        h.decode1(mat2,len1);
        System.out.println("KNOWN Plain Text - Cipher Text");
        System.out.println("Enter the Plain text ");
        String pt3 = sc.nextLine();
        System.out.println("Enter the cipher text ");
        String ct = sc.nextLine();
        System.out.println("Enter order of key matrix: ");
        int o2=input.nextInt();
        int [][] a2=new int[o2][o2];
        int [][] a3=new int[o2][o2];
        int [][]mat3=new int[o2][o2];
        a2=h.numencode(pt3,o2);
        a3=h.numencode(ct,o2);
        mat3= h.knownptct(a2,a3,o2);
    }
}
```

```
The decrypted plain text is:
islab
KNOWN Plain Text - Cipher Text
Enter the Plain text
apple
Enter the cipher text
bqqmf
Enter order of key matrix:
2
0   15  4
15  11  23
1   16  5
16  12  23
```

SCREEN SHOTS:

```
"C:\Program Files\Java\jdk-13.0.1\bin\java.exe" -Didea.launcher.port=55039 "-Didea.launcher
Encryption
Enter the order of key:
2
Enter the elements of key matrix (4):
12
19
15
9
Enter the plain text to encode :
islab
8   11  1
18  0   23
The encrypted matrix is:

    22  2   7
    22  9   14
The encrypted cipher text is:
wwcjho
DECRYPTION
Enter the order of key matrix:
```

```
wwcjho
DECRYPTION
Enter the order of key matrix:
2

Enter elements of key matrix:(4):
12
19
15
9
Enter the cipher text to decode:
wwcjho
22  2   7
22  9   14
Determinant of the matrix is : 5
Inverse of determinant: 21
Decrypted matrix:

    8   11  1
    18  0   23
```

```
The decrypted plain text is:
islab
KNOWN Plain Text - Cipher Text
Enter the Plain text
apple
Enter the cipher text
bqqmf
Enter order of key matrix:
2
0   15  4
15  11  23
1   16  5
16  12  23
```

```
Determinant of the matrix is : 16
Inverse of determinant: 1
Key matrix:

    20  15
    4   5
Process finished with exit code 0
```

RESULT:

Thus the programs for

- Encryption
- Decryption
- Known Plain text – Cipher text attack

in Hill Cipher are implemented in Java and the results are verified.

Evaluation

Criteria	Ratings				Pts
Correctness	5.0 pts Excellent • Program runs and completes all required tasks • Handles special cases • Executes without errors	3.0 pts Good • Program is complete in all aspects and competes most tasks appropriately • Program fails to work for special cases	2.0 pts Satisfactory • Individual modules produce expected output • Program fails to handle errors due to integration	0.0 pts Unsatisfactory • Individual modules do not execute due to errors. • No integration of modules has been performed • Incorrect results for most or all independent modules	
Coding Standards	5.0 pts Excellent • Includes name, date, and assignment title. • Excellent use of white space. • Creatively organized work. • Excellent use of variables (no global variables, unambiguous naming).	3.0 pts Good • Includes name, date, and assignment title. • Good use of white space. • Organized work. • Good use of variables (no global variables, unambiguous naming)	2.0 pts Satisfactory • Completed between 70-80% of the requirements. • Delivered on time, and in correct format (disk, email, etc.) global variables, unambiguous naming).	0.0 pts Unsatisfactory • Completed less than 70% of the requirements. • Not delivered on time or not in correct format (disk, email, etc.)	
Documentation	5.0 pts Excellent • Clearly and effectively documented including	3.0 pts Good • Clearly documented including descriptions of all	2.0 pts Satisfactory • Basic documentation has been completed	0.0 pts Unsatisfactory • No documentation included.	

Criteria	Ratings				Pts
	descriptions of all variables. • Specific purpose is noted for each function, control structure, input requirements, and output results.	variables. • Specific purpose is noted for each function and control structure.	including descriptions of all variables. • Purpose is noted for each function.		
Runtime	5.0 pts Excellent • Executes without errors excellent user prompts, good use of symbols, spacing in output. • Thorough and organized testing has been completed and output from test cases is included.	3.0 pts Good • Executes without errors. • User prompts are understandable, minimum use of symbols or spacing in output. • Thorough testing has been completed	2.0 pts Satisfactory • Executes without errors. • User prompts contain little information, poor design . • Some testing has been completed.	0.0 pts Unsatisfactory • Does not execute due to errors. • User prompts are misleading or non-existent. • No testing has been completed.	
Team Work	5.0 pts Excellent • Equal Participation and contribution • Excellent support for each other • Able to explain the logic of other modules	3.0 pts Good • Contribution from few members • Provide moderate explanation of other Modules • Moderate Support for Team members	2.0 pts Satisfactory • Contribution from one or two members in a group • No clear idea on the work of others	0.0 pts Unsatisfactory • No cooperation among team members • No support for each other • No idea on the work of other team members	
Completed on time	5.0 pts Excellent Program is completed on time	3.0 pts Good Program is one day late	2.0 pts Satisfactory Program is three day late	0.0 pts Unsatisfactory Program is late by more than three days	

Ex.No.3	**DEMONSTRATION OF TRIPLE DES AND BLOCK CIPHER MODES OF OPERATION**

AIM:

To demonstrate the working principle of Triple DES and Block cipher modes of operation

THEORY:

Meet in the middle attack

The meet-in-the-middle attack (MITM), a known plaintext attack, is a generic space–time trade off cryptographic attack against encryption schemes that rely on performing multiple encryption operations in sequence. The MITM attack is the primary reason why Double Des is not used and why a Triple Des key (168-bit) can be brute forced by an attacker with 2^{56} space and 2^{112} operations.

Triple DES Architecture

The Triple Data Encryption Algorithm is a symmetric key block cipher, which applies the DES cipher algorithm three times to each data block. The Data Encryption Standard's (DES) 56-bit key is no longer considered adequate in the face of modern cryptanalytic techniques and supercomputing power. However, an adapted version of DES, Triple DES (3DES), uses the same algorithm to produce a more secure encryption.

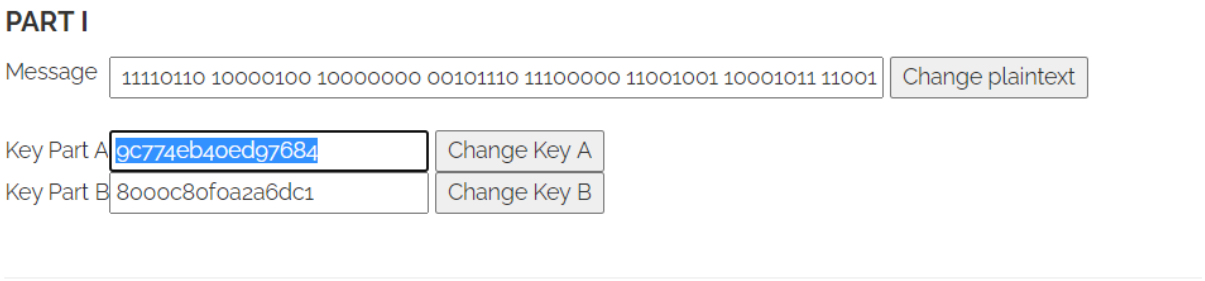
Block Cipher Modes of Operation

A block cipher is a deterministic algorithm operating on fixed-length groups of bits, called blocks. It uses an unvarying transformation, that is, it uses a symmetric key. They are specified elementary components in the design of many cryptographic protocol and are widely used to implement the encryption of large amounts of data, including data exchange protocols.

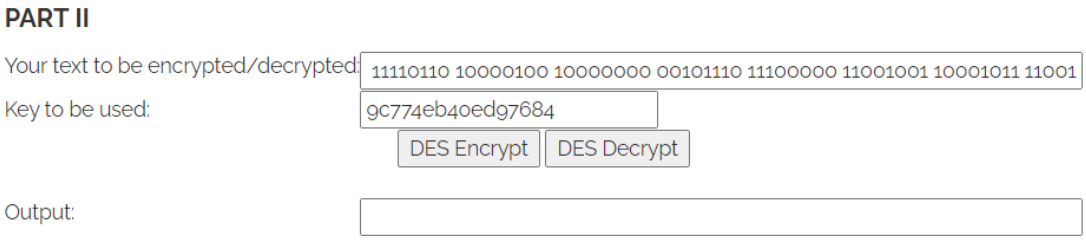
DEMONSTRATION IN VIRTUAL LABS

Triple DES:

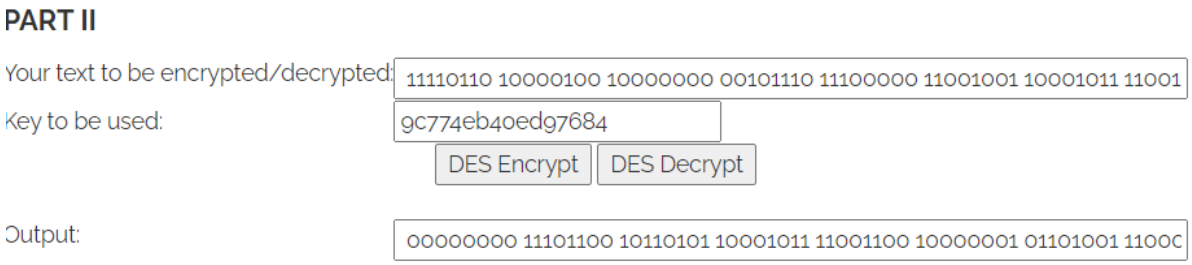
1. Generate Plain text, key1 and key2



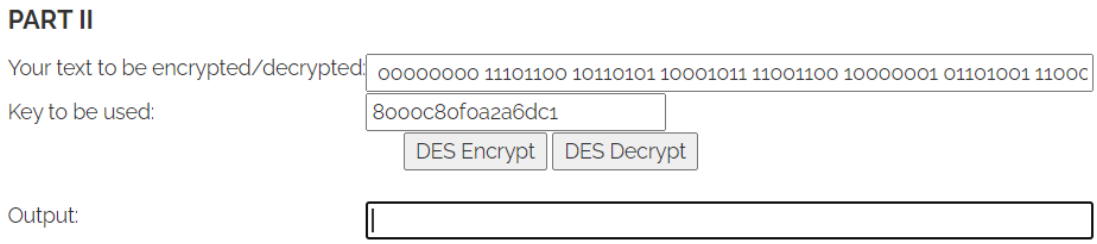
2. Use key1 and do encrypt we get



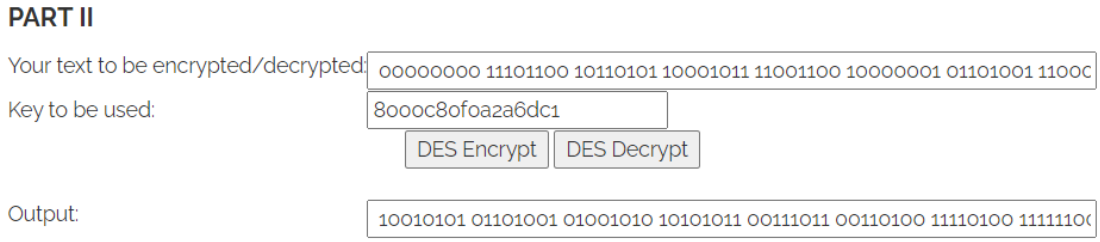
3. Click encrypt



4. Enter cipher text 1 in your text to be encrypted/decrypted textbox and key 2 is used

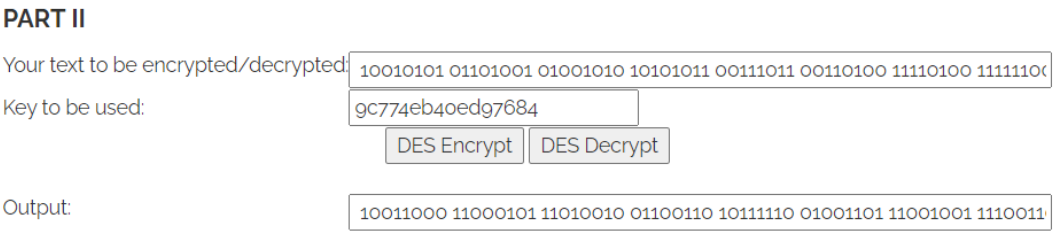


5. Click decrypt



6. Enter cipher text 2 and key 1

7. Click Encrypt



8. Check the answer

PART III

Enter your answer here:

10011000 11000101 11010010 01100110 10111110 01001101 11001001 111001

Check Answer!

CORRECT!

Block Cipher:

AES and Modes of Operation

AES (Rijndael) Encryption

PART I

Choose your mode of operation: Cipher Block Chaining

Step 2 :

PART II

Key size in bits: 128

Plaintext:
b9a72151 ea1a8458 e46e6aba 42f584b5
63e781f9 b1883c4a 94fd210c 020b6550
4422c6f6 7964b0f6 53b25994 532bea4d
90c1f07c 36d2119c 827c08ef 8be0736d
bfa83361 885309fd 78ea0c30 a354b060

Next Plaintext Key:

66ae3483 db6dfa6c 2eec4c98 7a3effde

Next Keytext

IV: b1869e56 4e7b4388 c2f5407d aaee46ae

Next IV

Step 3 :

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde
Plaintext in hex: b9a72151 ea1a8458 e46e6aba 42f584b5
Ciphertext in hex: 926ef408 83b9ff12 1dd52134 c8564d56
Encrypt Decrypt Clear

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde
Plaintext in hex: 63e781f9 b1883c4a 94fd210c 020b6550
Ciphertext in hex: 784fa4e3 32893612 0ebe6a62 9bf40cdb
Encrypt Decrypt Clear

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde
Plaintext in hex: 4422c6f6 7964b0f6 53b25994 532bea4d
Ciphertext in hex: d3a24149 4e0b6a05 d7ed5a06 4be5f281
Encrypt Decrypt Clear

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde
Plaintext in hex: 90c1f07c 36d2119c 827c08ef 8be0736d
Ciphertext in hex: d8afbb1c d5194f6c 60316742 1e26f044
Encrypt Decrypt Clear

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde
Plaintext in hex: bfa83361 885309fd 78ea0c30 a354b060
Ciphertext in hex: bb071316 6f977420 53c8aaa5 2236cd8e
Encrypt Decrypt Clear

Step 4 :

35

PART V

Enter your answer here:

926ef408 83b9ff12 1dd52134 c8564d56 784fa4e3 32893612 0ebe6a62 9k | Check Answer!

CORRECT!!

Code Block Chaining :

Step 1:

AES (Rijndael) Encryption

PART I

Choose your mode of operation: Cipher Block Chaining

Step 2 :

PART II

Key size in bits: 128

Plaintext:
b9a72151 ea1a8458 e46e6aba 42f584b5
63e781f9 b1883c4a 94fd210c 020b6550
4422c6f6 7964b0f6 53b25994 532bea4d
90c1f07c 36d2119c 827c08ef 8be0736d
bfa83361 885309fd 78ea0c30 a354b060

Next Plaintext | Key:

66ae3483 db6dfa6c 2eec4c98 7a3effde

Next Keytext

IV: f29fef64 d6d4d834 f1ddc73f f70442ef

Next IV

Step 3:

PART III

Calculate XOR:

b1869e56 4e7b4388 c2f5407d aaee46ae

b9a72151 ea1a8458 e46e6aba 42f584b5

Calculate XOR

XOR: 0821bf07 a461c7d0 269b2ac7 e81bc21b

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde

Plaintext in hex: 0821bf07 a461c7d0 269b2ac7 e81bc21b

Ciphertext in hex: f2aff4bd 84a4b302 e74b76ac 9740cd77

Encrypt Decrypt Clear

PART III

Calculate XOR:

f2aff4bd 84a4b302 e74b76ac 9740cd77

63e781f9 b1883c4a 94fd210c 020b6550

Calculate XOR

XOR: 91487544 352c8f48 73b657a0 954ba827

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde

Plaintext in hex: 91487544 352c8f48 73b657a0 954ba827

Ciphertext in hex: 8ad30e2e 3a54fe40 a6a4bfa0 9937c7be

Encrypt Decrypt Clear

PART III

Calculate XOR:

8ad30e2e 3a54fe40 a6a4bfa0 9937c7be

4422c6f6 7964b0f6 53b25994 532bea4d

Calculate XOR

XOR: cef1c8d8 43304eb6 f516e634 ca1c2df3

PART III

Calculate XOR:

04a7ad79 ae20db37 d086cf37 e1c22548

90c1f07c 36d2119c 827c08ef 8be0736d

Calculate XOR

XOR: 94665d05 98f2caab 52fac7d8 6a225625

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde

Plaintext in hex: 94665d05 98f2caab 52fac7d8 6a225625

Ciphertext in hex: ef497982 acb24c97 0faef6c8 3c1dce68

Encrypt Decrypt Clear

PART III

Calculate XOR:

ef497982 acb24c97 0faef6c8 3c1dce68

bfa83361 885309fd 78ea0c30 a354b060 Calculate XOR

XOR: 50e14ae3 24e1456a 7744faf8 9f497e08

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde

Plaintext in hex: 50e14ae3 24e1456a 7744faf8 9f497e08

Ciphertext in hex: 2cfdbbef 337bda64 5fec36a2 22a0a9b7

Encrypt Decrypt Clear

Step 4 :

PART V

Enter your answer here:

f29fef64 d6d4d834 f1ddc73f f70442ef 29633896 d6ec38d3 a43e905c bc Check Answer!

CORRECT!!

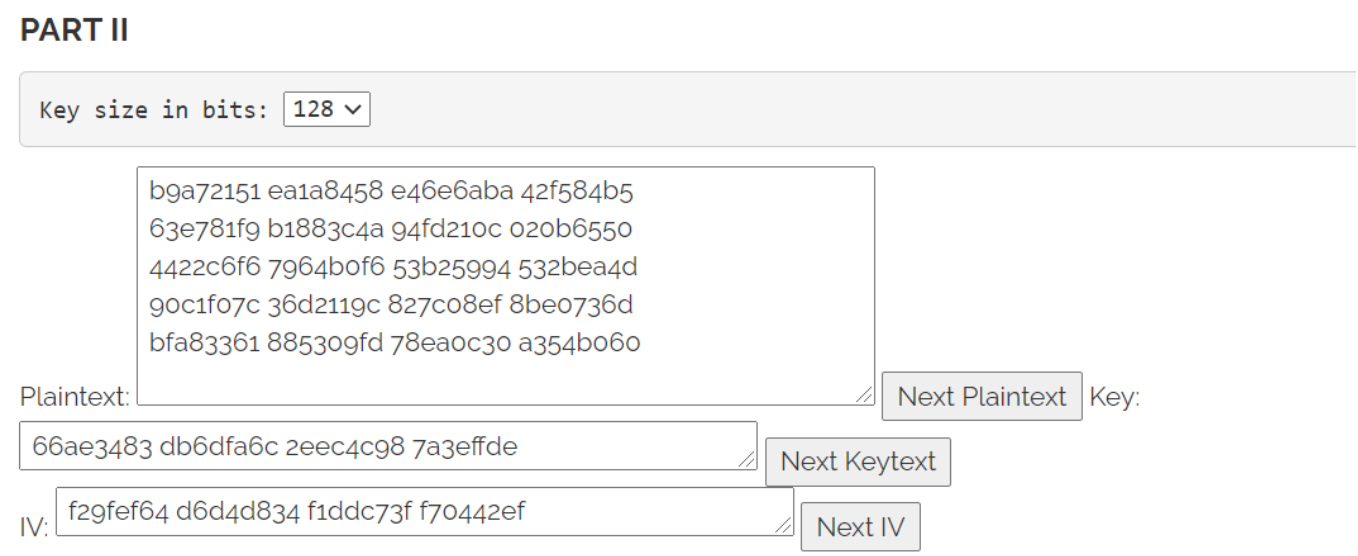
Output Feedback :

Step 1 :

PART I

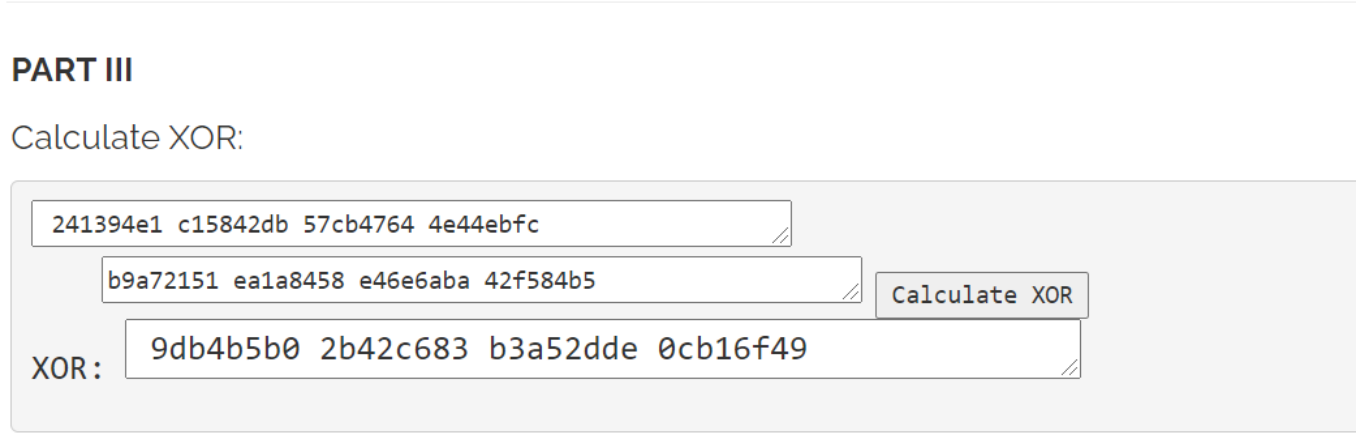
Choose your mode of operation: Output Feedback

Step 2 :


PART II
Key size in bits: 128
b9a72151 ea1a8458 e46e6aba 42f584b5
63e781f9 b1883c4a 94fd210c 020b6550
4422c6f6 7964b0f6 53b25994 532bea4d
90c1f07c 36d2119c 827c08ef 8be0736d
bfa83361 885309fd 78ea0c30 a354b060
Plaintext:
Next Plaintext
Key:
66ae3483 db6dfa6c 2eec4c98 7a3effde
Next Keytext
IV:
f29fef64 d6d4d834 f1ddc73f f70442ef
Next IV

Step 3 :


PART IV
Key in hex:
66ae3483 db6dfa6c 2eec4c98 7a3effde
Plaintext in hex:
f29fef64 d6d4d834 f1ddc73f f70442ef
Ciphertext in hex:
241394e1 c15842db 57cb4764 4e44ebfc
Encrypt
Decrypt
Clear


PART III
Calculate XOR:
241394e1 c15842db 57cb4764 4e44ebfc
b9a72151 ea1a8458 e46e6aba 42f584b5
Calculate XOR
XOR:
9db4b5b0 2b42c683 b3a52dde 0cb16f49

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde

Plaintext in hex: 241394e1 c15842db 57cb4764 4e44ebfc

Ciphertext in hex: 90b7434c 0cadc4c3 5652ff3e 8382b058

Encrypt | Decrypt | Clear

PART III

Calculate XOR:

90b7434c 0cadc4c3 5652ff3e 8382b058

63e781f9 b1883c4a 94fd210c 020b6550

Calculate XOR

XOR: f350c2b5 bd25f889 c2afde32 8189d508

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde

Plaintext in hex: 90b7434c 0cadc4c3 5652ff3e 8382b058

Ciphertext in hex: bbd552cb 70c12846 43250f16 435ec368

Encrypt | Decrypt | Clear

PART III

Calculate XOR:

bbd552cb 70c12846 43250f16 435ec368

4422c6f6 7964b0f6 53b25994 532bea4d

Calculate XOR

XOR: fff7943d 09a598b0 10975682 10752925

PART IV

Key in hex: 66ae3483 db6dfa6c 2eec4c98 7a3effde

Plaintext in hex: 2921dfa2 491ef726 91ed3b57 d80415aa

Ciphertext in hex: db34d541 aceb6125 d6ed8316 b4be9c66

Encrypt | Decrypt | Clear

PART III

Calculate XOR:

db34d541 aceb6125 d6ed8316 b4be9c66

bfa83361 885309fd 78ea0c30 a354b060 Calculate XOR

XOR: 649ce620 24b868d8 ae078f26 17ea2c06

Step 4:

PART V

Enter your answer here:

f29fef64 d6d4d834 f1ddc73f f70442ef9db4b5b0 2b42c683 b3a52dde 0c Check Answer!

CORRECT!!

Counter Mode :

Step 1 :

PART I

Choose your mode of operation: Counter mode

Step 2 :

PART II

Key size in bits: 128

Plaintext: 180b5121 02d4e3c6 46cb787a 6abb1045
6da2f3be b3a87eb8 748a022f 33f27e8a
522881fc cb23983a 736acab9 a1556f5e
ab12f8ff c886ce76 3cffc3d9 1696007a
d7e19acd 8a0ea7c0 8c6107eb 4b88e548

Next Plaintext

Key: 15597b05 32edd20f a851247e 0ec3ff79

Next Keytext

CTR: 582c08de 34db3569 6c95191d 4498fe0a

Next CTR

Step 3 :

PART III

Calculate XOR:

582c08de 34db3569 6c95191d 4498fe0a

1

Calculate XOR

XOR: 582c08de 34db3569 6c95191d 4498fe0b

PART IV

Key in hex: 15597b05 32edd20f a851247e 0ec3ff79

Plaintext in hex: 582c08de 34db3569 6c95191d 4498fe0b

Ciphertext in hex: 9ce335fe ba0d532c 9abd58fe 3f59dd88

Encrypt | Decrypt | Clear

PART III

Calculate XOR:

180b5121 02d4e3c6 46cb787a 6abb1045

9ce335fe ba0d532c 9abd58fe 3f59dd88

Calculate XOR

XOR: 84e864df b8d9b0ea dc762084 55e2cdcd

PART III

Calculate XOR:

582c08de 34db3569 6c95191d 4498fe0a

2

Calculate XOR

XOR: 582c08de 34db3569 6c95191d 4498fe08

PART IV

Key in hex: 15597b05 32edd20f a851247e 0ec3ff79

Plaintext in hex: 582c08de 34db3569 6c95191d 4498fe08

Ciphertext in hex: dab4f0ea 3c867c16 8af42257 dd460a61

Encrypt | Decrypt | Clear

PART III

Calculate XOR:

582c08de 34db3569 6c95191d 4498fe0a

3

Calculate XOR

XOR: 582c08de 34db3569 6c95191d 4498fe09

PART IV

Key in hex: 15597b05 32edd20f a851247e 0ec3ff79

Plaintext in hex: 582c08de 34db3569 6c95191d 4498fe0a

Ciphertext in hex: b077a351 fd2a35c0 902a6880 00144bd9

Encrypt Decrypt Clear

PART III

Calculate XOR:

522881fc cb23983a 736acab9 a1556f5e

b077a351 fd2a35c0 902a6880 00144bd9

Calculate XOR

XOR: e25f22ad 3609adfa e340a239 a1412487

PART III

Calculate XOR:

582c08de 34db3569 6c95191d 4498fe0a

4

Calculate XOR

XOR: 582c08de 34db3569 6c95191d 4498fe0e

PART IV

Key in hex: 15597b05 32edd20f a851247e 0ec3ff79

Plaintext in hex: 582c08de 34db3569 6c95191d 4498fe0e

Ciphertext in hex: e3d028a1 79c34f22 59ba2ad9 006ff3d3

Encrypt Decrypt Clear

PART III

Calculate XOR:

e3d028a1 79c34f22 59ba2ad9 006ff3d3

ab12f8ff c886ce76 3cffc3d9 1696007a

Calculate XOR

XOR: 48c2d05e b1458154 6545e900 16f9f3a9

PART III

Calculate XOR:

582c08de 34db3569 6c95191d 4498fe0a

5

Calculate XOR

XOR: 582c08de 34db3569 6c95191d 4498fe0f

PART IV

Key in hex:	15597b05 32edd20f a851247e 0ec3ff79
Plaintext in hex:	582c08de 34db3569 6c95191d 4498fe0f
Ciphertext in hex:	f839d143 15baa865 b425531a 182cf7a3

Encrypt Decrypt Clear

PART III

Calculate XOR:

d7e19acd 8a0ea7c0 8c6107eb 4b88e548

f839d143 15baa865 b425531a 182cf7a3

Calculate XOR

XOR: 2fd84b8e 9fb40fa5 384454f1 53a412eb

Step 4 :

PART V

Enter your answer here:

582c08de 34db3569 6c95191d 4498fe0a 84e864df b8d9b0ea dc76208 Check Answer!

RESULT:

The Electronic Code Block(ECB),Cipher Block Chaining(CBC),Output Feedback, Counter Mode are verified.

Evaluation

Parameter	Max Marks	Marks Obtained
Originality of the Work	15	
Documentation	15	
Total		
Signature of the faculty with Date		

Ex.No.4	**IMPLEMENTATION OF DIFFIE HELLMAN KEY EXCHANGE**

AIM:

To simulate the working of Diffie Hellman in Virtual lab environment and to implement the same in Client/ Server model using Java

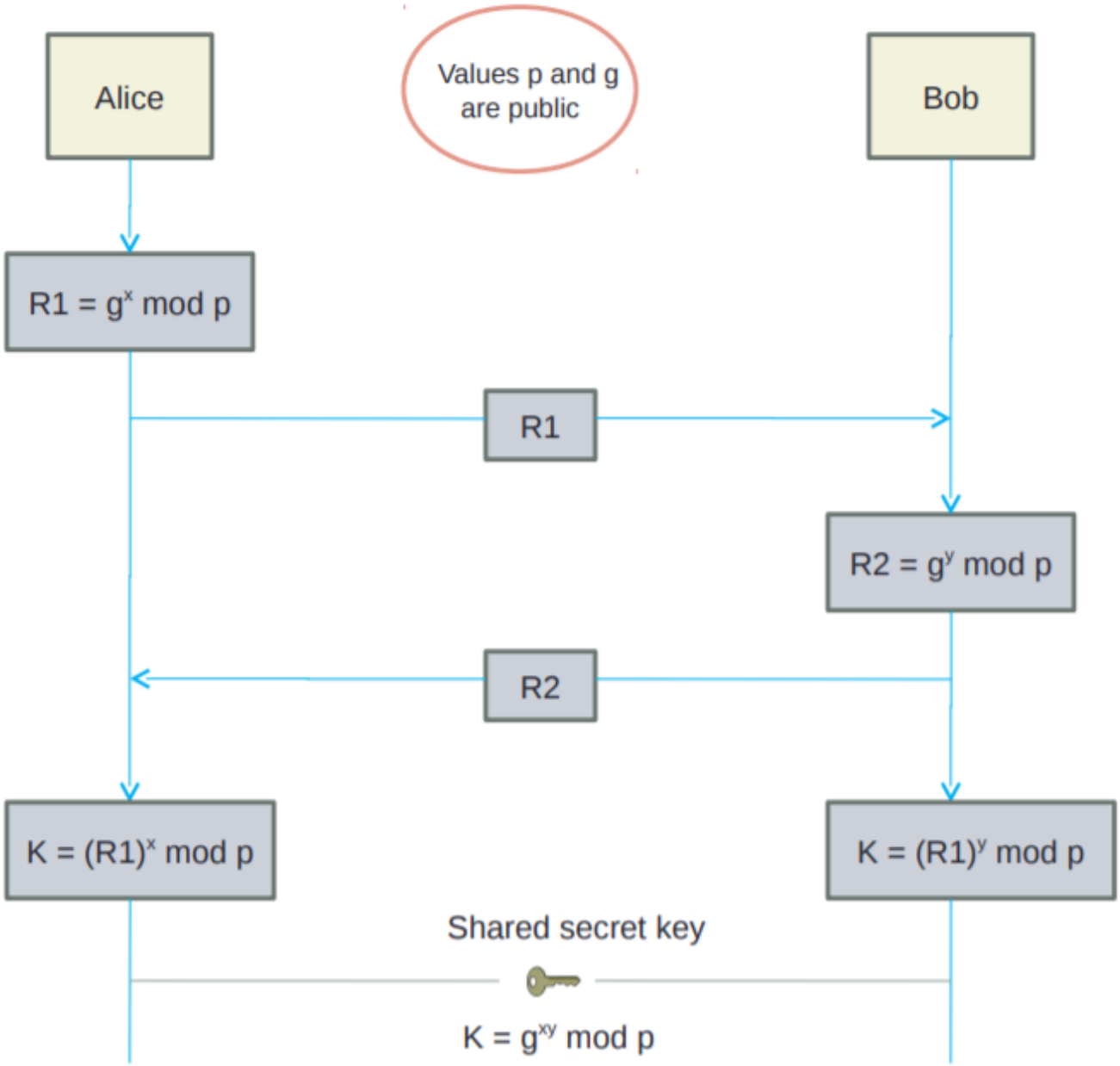
THEORY:

Diffie-Hellman protocol allows two communicating parties, say Alice and Bob, to create a symmetric session key with out the need of a KDC (Key Distribution Center)

Diffie-Hellman Protocol :

Alice and Bob chose two numbers p and g which are public.
'p' is a large prime of the order of 1024 bits. 'g' is a generator of order p-1 in the group Z p *
Alice chooses a large random number 'x' in the range 0 to p-1 and calculates R1 = g^x mod p
Bob chooses a large random number 'y' in the range 0 to p-1 and calculates R2 = g^y mod p
Alice sends R1 to Bob and Bob sends R2 to Alice Alice Calculates K = $(R2)^x$ mod p
Bob Calculates K = $(R1)^y$ mod p
K = (gx mod p)y mod p = (gy mod p)x mod p = gxy mod p
K is the symmetric key for the session

P is a large prime of the order of 1024 bits
g is a generator of order p-1 in the group Zp*



ALGORITHM:

STEP 1 : Firslty,choose a large prime number **p** and a generator **g** for that prime.

STEP 2 : Secondly,both Alice and Bob generate their respective keys **A** and **B**. And (**ga**,**gb**) for their keys respectively.

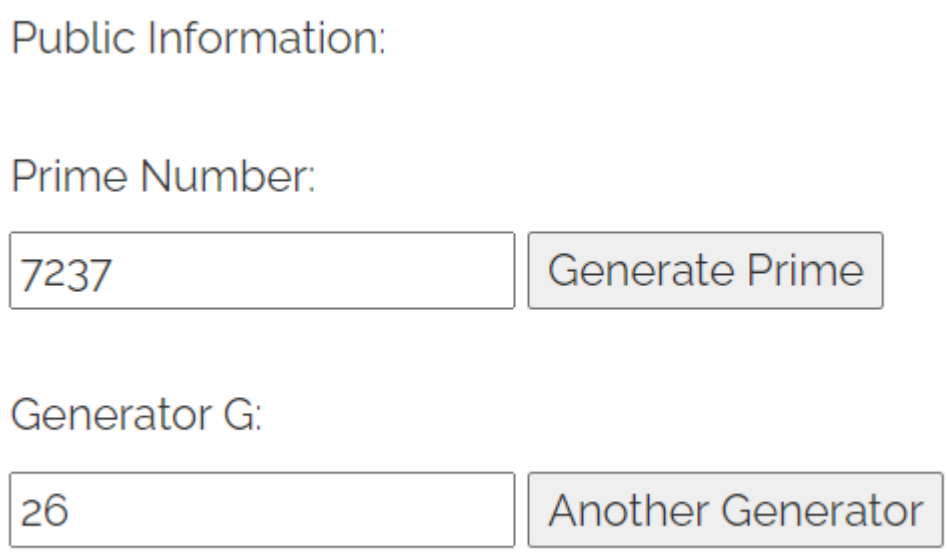
STEP 3 : Both Alice and bob send exchange their **ga**,**gb**.

STEP 4 : Finally,both calculate their public keys **gab** and **gba** repectively.

STEP 5 : If both **gab** and **gba** are equal then Deffie-Hellman key exchange is verified.

Screen Shots of simulation in Virtual labs

STEP 1 :



STEP 2 :

STEP 3 :

Coding

Alice:

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

public class Alice{
public static void main(String[] args) throws IOException
{
try {
int port = 8000;

Scanner Sc= new Scanner (System.in);
System.out.print("enter a Private key B of Bob:");
int b =Sc.nextInt();

double clientP, clientG, clientA, B, Bdash;
String Bstr;

ServerSocket serverSocket = new ServerSocket(port);
Socket server = serverSocket.accept();
System.out.println("Connected to port ");

System.out.println("The private key of Bob = " + b);

DataInputStream in = new DataInputStream(server.getInputStream());

clientP = Integer.parseInt(in.readUTF());System.out.println("From Alice: P = " + clientP);
clientG = Integer.parseInt(in.readUTF());

System.out.println("From Alice : G = " + clientG);

clientA = Double.parseDouble(in.readUTF());
System.out.println("From Alice : Public Key = " + clientA);
```

```
B = ((Math.pow(clientG, b)) % clientP);
Bstr = Double.toString(B);

OutputStream outToclient = server.getOutputStream();
DataOutputStream out = new DataOutputStream(outToclient);

out.writeUTF(Bstr);

Bdash = ((Math.pow(clientA, b)) % clientP); System.out.println("Secret key of Alice "+
Bdash);
server.close();
}

catch (SocketTimeoutException s) {
System.out.println("Timed out!");
}
catch (IOException e) {
}
}
}
```

Bob:

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

public class Bob{
static void Pr (int num)
{
boolean flag = false;
for(int i = 2; i <= num/2; ++i)
{
if(num % i == 0)
{
flag = true;
break;
}
}
if (!flag)
{
```

```
System.out.println(num + " is a prime number.");
}
else
{
System.out.println(num + " is not a prime number.");
}
for(int i=2;i < num;i++)
{
int p=0;
int[] a = new int[200];
System.out.println(" ");
for(int j=1;j < num;j++)
{
p = (int)Math.pow(i, j);
int x1=p%num;
System.out.println(+i+" ^"+ j +" mod "+num +"="+x1);
a[j]=x1;
}
find(a,i,num);
}
}

static void find ( int [] a, int i, int num)
{

int[] b = new int[200];
for(int z=1;z<num ;z++)
{
int count=0;
for(int e=1;e<num;e++)
{
b=a;
if(b[e]==a[z]){
count++;
}

}
if(count ==1 && z == num-1)
System.out.println(+i+" is primitive root of "+num);

}
```

```
}
public static void main(String[] args)
{
try {
String pstr, gstr, Astr;
String serverName = "localhost";
int port = 8000;
System.out.print("enter prime:");
Scanner S1= new Scanner (System.in);
int p = S1.nextInt();
Pr(p);
System.out.print("Enter private key of Alice:");
int a = S1.nextInt();
System.out.print("Enter Gb:");
int g = S1.nextInt();
double Adash, serverB;

Socket client = new Socket(serverName, port);
System.out.println("Connected to port ");

OutputStream outToServer = client.getOutputStream();
DataOutputStream out = new DataOutputStream(outToServer);

pstr = Integer.toString(p);
out.writeUTF(pstr);

gstr = Integer.toString(g);
out.writeUTF(gstr);

double A = ((Math.pow(g, a)) % p);
Astr = Double.toString(A);
out.writeUTF(Astr);

System.out.println("From Alice (server): Private Key = " + a);

DataInputStream in = new DataInputStream(client.getInputStream());

serverB = Double.parseDouble(in.readUTF());
System.out.println("From Bob(client) : Public Key = " + serverB);

Adash = ((Math.pow(serverB, a)) % p);
```

```
System.out.println("Secret Key of Bob "+ Adash);
client.close();
}
catch (Exception e) {
e.printStackTrace();
}
}
}
```

SCREEN SHOTS:

```
enter a Private key B of Bob:12
Connected to port
The private key of Bob = 12
From Alice: P = 11.0
From Alice : G = 9.0
From Alice : Public Key = 9.0
Secret key of Alice 4.0

Process completed.
```

```
6 ^2 mod 11=3
6 ^3 mod 11=7
6 ^4 mod 11=9
6 ^5 mod 11=10
6 ^6 mod 11=5
6 ^7 mod 11=8
6 ^8 mod 11=4
6 ^9 mod 11=2
6 ^10 mod 11=1
6 is primitive root of 11

7 ^1 mod 11=7
7 ^2 mod 11=5
7 ^3 mod 11=2
7 ^4 mod 11=3
7 ^5 mod 11=10
7 ^6 mod 11=4
7 ^7 mod 11=6
7 ^8 mod 11=9
7 ^9 mod 11=8
7 ^10 mod 11=1
7 is primitive root of 11

8 ^1 mod 11=8
8 ^2 mod 11=9
8 ^3 mod 11=6
8 ^4 mod 11=4
8 ^5 mod 11=10
8 ^6 mod 11=3
8 ^7 mod 11=2
8 ^8 mod 11=5
8 ^9 mod 11=7
8 ^10 mod 11=1
8 is primitive root of 11

9 ^1 mod 11=9
9 ^2 mod 11=4
9 ^3 mod 11=3
9 ^4 mod 11=5
9 ^5 mod 11=1
9 ^6 mod 11=9
9 ^7 mod 11=4
9 ^8 mod 11=3
9 ^9 mod 11=5
9 ^10 mod 11=1

10 ^1 mod 11=10
10 ^2 mod 11=1
```

```
enter prime:11
11 is a prime number.

2 ^1 mod 11=2
2 ^2 mod 11=4
2 ^3 mod 11=8
2 ^4 mod 11=5
2 ^5 mod 11=10
2 ^6 mod 11=9
2 ^7 mod 11=7
2 ^8 mod 11=3
2 ^9 mod 11=6
2 ^10 mod 11=1
2 is primitive root of 11

3 ^1 mod 11=3
3 ^2 mod 11=9
3 ^3 mod 11=5
3 ^4 mod 11=4
3 ^5 mod 11=1
3 ^6 mod 11=3
3 ^7 mod 11=9
3 ^8 mod 11=5
3 ^9 mod 11=4
3 ^10 mod 11=1

4 ^1 mod 11=4
4 ^2 mod 11=5
4 ^3 mod 11=9
4 ^4 mod 11=3
4 ^5 mod 11=1
4 ^6 mod 11=4
4 ^7 mod 11=5
4 ^8 mod 11=9
4 ^9 mod 11=3
4 ^10 mod 11=1


5 ^1 mod 11=5
5 ^2 mod 11=3
5 ^3 mod 11=4
5 ^4 mod 11=9
5 ^5 mod 11=1
5 ^6 mod 11=5
5 ^7 mod 11=3
5 ^8 mod 11=4
5 ^9 mod 11=9
```

```
10 ^2 mod 11=1
10 ^3 mod 11=10
10 ^4 mod 11=1
10 ^5 mod 11=10
10 ^6 mod 11=1
10 ^7 mod 11=10
10 ^8 mod 11=1
10 ^9 mod 11=10
10 ^10 mod 11=1
Enter private key of Alice:6
Enter Gb:9
Connected to port
From Alice (server): Private Key = 6
From Bob(client) : Public Key = 4.0
Secret Key of Bob 4.0

Process completed.
```

RESULT:

Thus the working of Diffie Hellman in Virtual lab environment and the implementation of the same in Client/ Server model using Java is done.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Code	15	
Completion of experiment on time	5	
Documentation	5	
Simulation in Vlabs	5	
Total	30	
Signature of the faculty with Date		

Ex.No.5	**IMPLEMENTATION OF RSA**

AIM:

To simulate the working of RSA in Virtual lab environment and to implement the same in Java/Python

THEORY:

One-way function

A Trapdoor one-way function is a one-way function which might be described as a function for which evaluation in one direction is straightforward, while computation in the reverse direction is far more difficult.

Key generation

The difficulty of determining a private key from an RSA public key is equivalent to factoring the modulus n. An attacker thus cannot use knowledge of an RSA public key to determine an RSA private key unless he can factor n. It is also a one-way function, going from p & q values to modulus n is easy but reverse is not possible.

Select two large primes, p and q.

Calculate n=p*q.

Select number e which must be greater than 1 and less than $(p - 1)(q - 1)$ and such that e and $(p - 1)(q - 1)$ are coprime.

(n, e) - RSA public key

$d \equiv e^{-1} \bmod (p - 1)(q - 1)$ – RSA private key

RSA Encryption

The sender whose public key is (n, e) represents the plaintext as a series of numbers less than n.

To encrypt the first plaintext P, which is a number modulo n. The encryption process is simple mathematical step as

$$\mathbf{c = p^e \bmod n}$$

RSA Decryption

The receiver of public-key pair (n, e) with a ciphertext c raises c to the power of his private key d. The result modulo n will be the plaintext P

$$\mathbf{p = c^d \bmod n}$$

ALGORITHM:

Key generation

1. Select two large primes, p and q.
2. Calculate n=p*q.
3. Select number e which must be greater than 1 and less than (p − 1)(q − 1) and such that e and (p – 1)(q – 1) are coprime.
4. (n, e) gives the public key pair.
5. $d \equiv e^{-1} \bmod (p - 1)(q - 1)$ gives the private key of the recipient.

RSA Encryption

1. Make the plain text as p.
2. With the public key pair (n,e) calculate cipher text where c is

   $$\mathbf{c = p^e \bmod n}$$
3. Send the cipher text to the receiver.

RSA Decryption

1. Receive the cipher text as c.
2. With the public key pair (n,e) and with private key d calculate p

   $$\mathbf{p = c^d \bmod n}$$

Screen Shots of simulation in Virtual labs

1. Enter the plain text .

Public-Key Cryptosystems (PKCSv1.5)

Plaintext (string):

yajith vishwa

encrypt

Ciphertext (hex):

decrypt

Decrypted Plaintext (string):

Status:

2. Generate RSA private key by clicking any one bit key

RSA private key

1024 bit | 1024 bit (e=3) | 512 bit | 512 bit (e=3) | Generate bits = 512

Modulus (hex):

Public exponent (hex, F4=0x10001):

3

Private exponent (hex):

P (hex):

Q (hex):

3. Generate private key.

RSA private key

1024 bit | 1024 bit (e=3) | 512 bit | 512 bit (e=3) | Generate | bits = 512

Modulus (hex):

76d112980479c3c3b561f6f7b4a88a8be22f129cf9a5b90085d13abde9b83131
2875442942658ffb2963b1c8aea90a4436fed597cd87926a5f9099c9f30d281b

Public exponent (hex, F4=0x10001):

3

Private exponent (hex):

4f360c6558512d2d2396a4a5231b0707ec1f61bdfbc3d0ab03e0d1d3f125761f
d733d8807f900c049ae6fb1c5a922bcc64aa0c34e43c88ee5b63c27cbc87312b

P (hex):

da59a90969881b1c51bafec43b67d58d571e81a2fa886ec2bc73459253217ffb

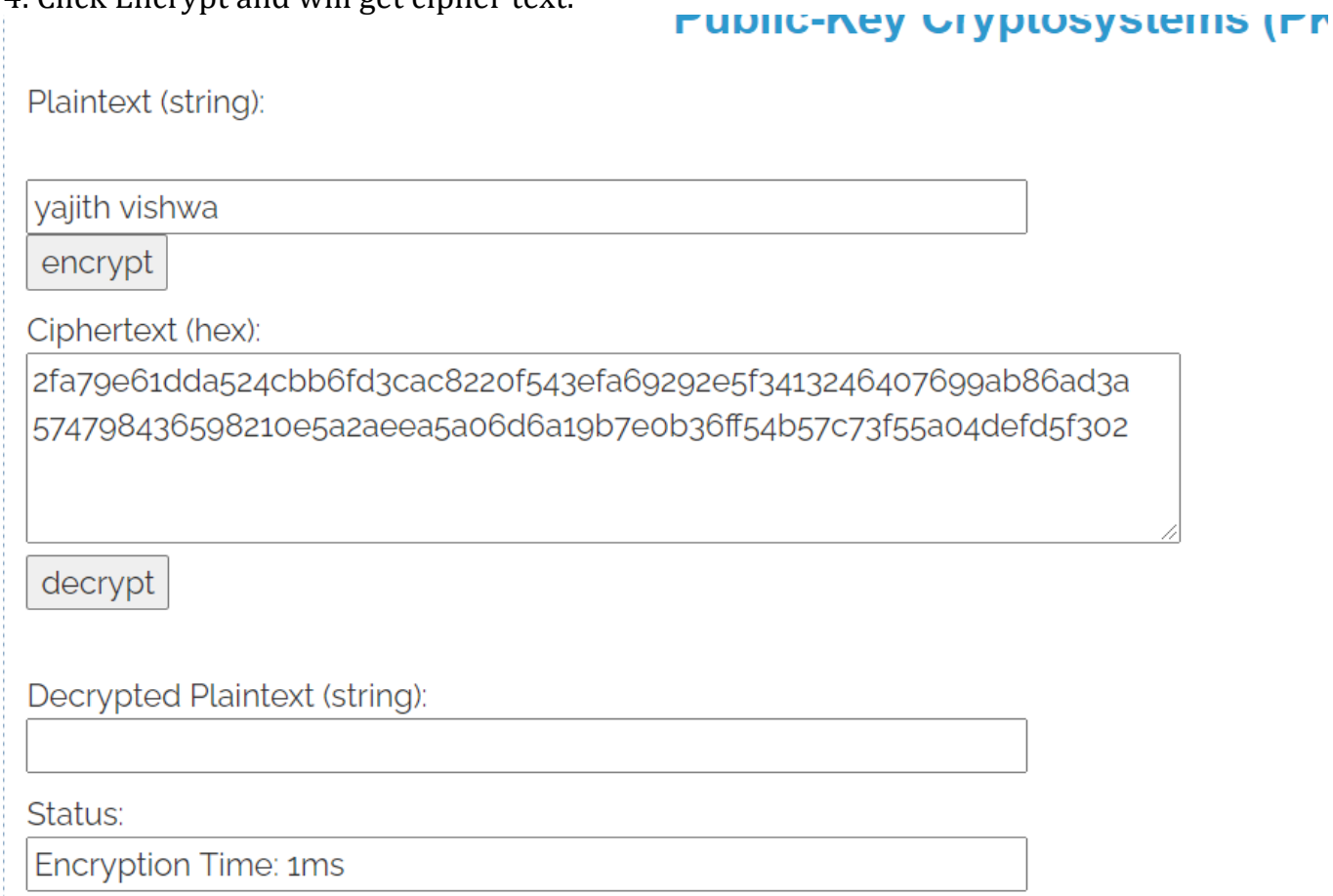
Q (hex):

8b4dd65f198562d7ef4e3a59eb65f30448e141a57ca456421a07b07c8520de61

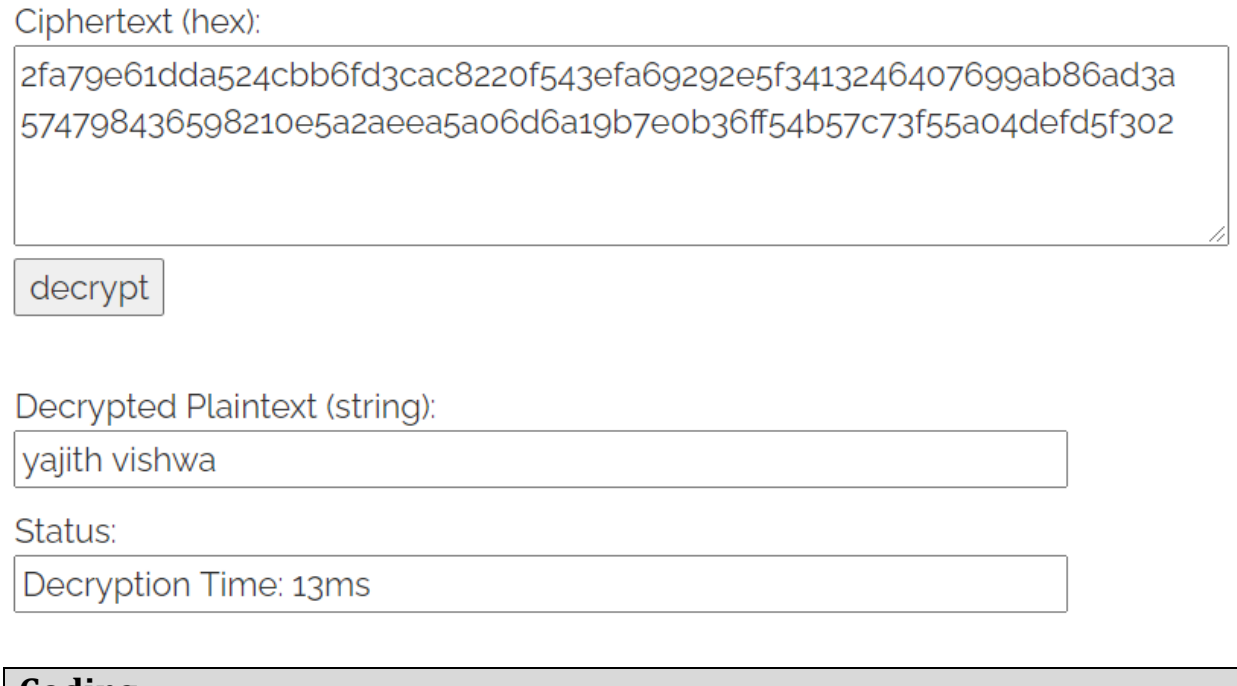
D mod (P-1) (hex):

91911b5b9bb01212e1275482d2453908e4bf011751b049d7284cd90c376baaa7

4. Click Encrypt and will get cipher text.



5. Click Decrypt to get back the plain text.



Coding

```java
package com.yajith.hillcipher;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int p,q,n,z,d=0,e,i;
        System.out.println("Enter the number to be encrypted and decrypted");
        int msg=sc.nextInt();
        double c;
        BigInteger msgback;
        System.out.println("Enter 1st prime number p");
        p=sc.nextInt();
        System.out.println("Enter 2nd prime number q");
        q=sc.nextInt();

        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)            // e is from public key pair
            {
                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 is for private key
            {
                d=x/e;
                break;
            }
        }
        System.out.println("the value of d = "+d);
        c=(Math.pow(msg,e))%n;
        System.out.println("Encrypted message is : -");
        System.out.println(c);
        BigInteger N = BigInteger.valueOf(n);
        BigInteger C = BigDecimal.valueOf(c).toBigInteger();
        msgback = (C.pow(d)).mod(N);
        System.out.println("Derypted message is : -");
        System.out.println(msgback);

    }
    static int gcd(int e, int z)
    {
        if(e==0)
            return z;
        else
            return gcd(z%e,e);
    }
}
```

SCREEN SHOTS:

```
Enter the number to be encrypted and decrypted
24
Enter 1st prime number p
7
Enter 2nd prime number q
11
the value of z = 60
the value of e = 7
the value of d = 43
Encrypted message is : -
73.0
Derypted message is : -
24

Process finished with exit code 0
```

RESULT:

Thus, the simulation of RSA algorithm in done in virtual lab and the Java program for the algorithm is also implemented and the results are obtained.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Code	15	
Completion of experiment on time	5	
Documentation	5	
Simulation in Vlabs	5	
Total	30	
Signature of the faculty with Date		

Ex.No.6	SECURITY ANALYSIS OF NETWORK TRAFFIC USING WIRESHARK

AIM:

To analyse the network traffic for security analysis using wireshark.

THEORY:

Wireshark

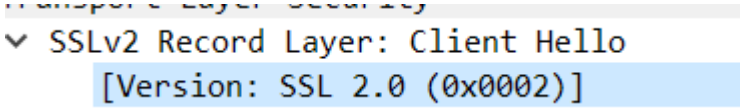
Wireshark is a packet sniffer and analysis tool. It captures network traffic on the local network and stores that data for offline analysis. Wireshark captures network traffic from Ethernet, Bluetooth, Wireless (IEEE.802.11), Token Ring, Frame Relay connections, and etc.

SSL/TLS

SSL and its successor, **TLS** are protocols for establishing authenticated and encrypted links between networked computers. Although the SSL protocol was deprecated with the release of TLS 1.0 in 1999, it is still common to refer to these related technologies as “SSL” or “SSL/TLS.” The most current version is **TLS 1.3**, defined in RFC 8446**.**

WORKSHEET (using sample.pcap)

1. What version of SSL is supported by the client?
   SSL version 2 is used by the client.



2. List the cryptographic algorithms supported by the client in Client Hello message.

```
Cipher Specs (26 specs)
   Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)
   Cipher Spec: SSL2_RC2_128_CBC_WITH_MD5 (0x030080)
   Cipher Spec: SSL2_DES_192_EDE3_CBC_WITH_MD5 (0x0700c0)
   Cipher Spec: SSL2_DES_64_CBC_WITH_MD5 (0x060040)
   Cipher Spec: SSL2_RC4_128_EXPORT40_WITH_MD5 (0x020080)
   Cipher Spec: SSL2_RC2_128_CBC_EXPORT40_WITH_MD5 (0x040080)
   Cipher Spec: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x000039)
   Cipher Spec: TLS_DHE_DSS_WITH_AES_256_CBC_SHA (0x000038)
   Cipher Spec: TLS_RSA_WITH_AES_256_CBC_SHA (0x000035)
   Cipher Spec: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x000033)
   Cipher Spec: TLS_DHE_DSS_WITH_AES_128_CBC_SHA (0x000032)
   Cipher Spec: TLS_RSA_WITH_RC4_128_MD5 (0x000004)
   Cipher Spec: TLS_RSA_WITH_RC4_128_SHA (0x000005)
   Cipher Spec: TLS_RSA_WITH_AES_128_CBC_SHA (0x00002f)
   Cipher Spec: TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (0x000016)
   Cipher Spec: TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA (0x000013)
   Cipher Spec: SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (0x00feff)
   Cipher Spec: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000a)
   Cipher Spec: TLS_DHE_RSA_WITH_DES_CBC_SHA (0x000015)
   Cipher Spec: TLS_DHE_DSS_WITH_DES_CBC_SHA (0x000012)
   Cipher Spec: SSL_RSA_FIPS_WITH_DES_CBC_SHA (0x00fefe)
   Cipher Spec: TLS_RSA_WITH_DES_CBC_SHA (0x000009)
   Cipher Spec: TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (0x000064)
   Cipher Spec: TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA (0x000062)
   Cipher Spec: TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x000003)
   Cipher Spec: TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x000006)
Challenge
```

3. List the various parameters present in the public key certificate of the server.

```
signedCertificate
    version: v3 (2)
    serialNumber: 0x01
  > signature (md5WithRSAEncryption)
  > issuer: rdnSequence (0)
  > validity
  > subject: rdnSequence (0)
  v subjectPublicKeyInfo
    v algorithm (rsaEncryption)
        Algorithm Id: 1.2.840.113549.1.1.1 (rsaEncryption)
    v subjectPublicKey: 30818902818100a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba…
        modulus: 0x00a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba734b78443dc6…
        publicExponent: 65537
  > extensions: 3 items
> algorithmIdentifier (md5WithRSAEncryption)
  Padding: 0
  encrypted: ae7979229075fda6d5c4b8c4994e1c057c9159be890d3dc68ca3cff6ba23dfb8ae44688a…
```

4. Identify the public key of the server? Can you trust the same?
   30818902818100a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba
   734b78443dc66c69a425b361029d09043f723dd827d3b05a4577b736e42623cc12b8ae
   dea7b63a823c7c24590af896438ba329363f917f5dc72394297f0ace0abd8d9b2f1917aad
   58eec66a237eb3f57533cf2aabb79194b907ea7a399fe844c89f03d0203010001

   ```
   subjectPublicKey: 30818902818100a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba…
   ```

   Yes we can trust because they have used RDN Sequence

```
v subject: rdnSequence (0)
  v rdnSequence: 7 items (pkcs-9-at-emailAddress=www@snakeoil.dom,id-at-commonName=www.snakeoil.dom,id-at-organizationalUnitName=Webserver Team,id-at-organizationName=Snake Oil, Ltd,id-at-l…
    v RDNSequence item: 1 item (id-at-countryName=XY)
      v RelativeDistinguishedName item (id-at-countryName=XY)
          Id: 2.5.4.6 (id-at-countryName)
          CountryName: XY
    > RDNSequence item: 1 item (id-at-stateOrProvinceName=Snake Desert)
    > RDNSequence item: 1 item (id-at-localityName=Snake Town)
    > RDNSequence item: 1 item (id-at-organizationName=Snake Oil, Ltd)
    > RDNSequence item: 1 item (id-at-organizationalUnitName=Webserver Team)
    > RDNSequence item: 1 item (id-at-commonName=www.snakeoil.dom)
    > RDNSequence item: 1 item (pkcs-9-at-emailAddress=www@snakeoil.dom)
```

5. Identify the length of key exchanged by the Client?
   The length of the key exchanged by the client is 128.

```
v Handshake Protocol: Client Key Exchange
      Handshake Type: Client Key Exchange (16)
      Length: 128
```

6. What algorithm is used for encrypting the session key?
   The Algorithm used for encrypting the session key is RSA Encryption algorithm.

```
v RSA Encrypted PreMaster Secret
    Encrypted PreMaster: 65512da6d4a738dfac791f0bd9b2617d738832d9f2623a8b110475ca42ff4ed9ccb9fa86…
```

7. List the various parameters specified in the Encrypted handshake message.
   The parameters are Content Type, Version, Length, Handshake protocol.

```
▸ SSLv3 Record Layer: Handshake Protocol: Encrypted Handshake Message
    Content Type: Handshake (22)
    Version: SSL 3.0 (0x0300)
    Length: 64
    Handshake Protocol: Encrypted Handshake Message
```

8. Calculate the time taken for completion of the entire handshake protocol.

   End

```
18 2.943406    127.0.0.1    127.0.0.1    SSLv3    172 Change Cipher Spec, Encrypted Handshake Message
```

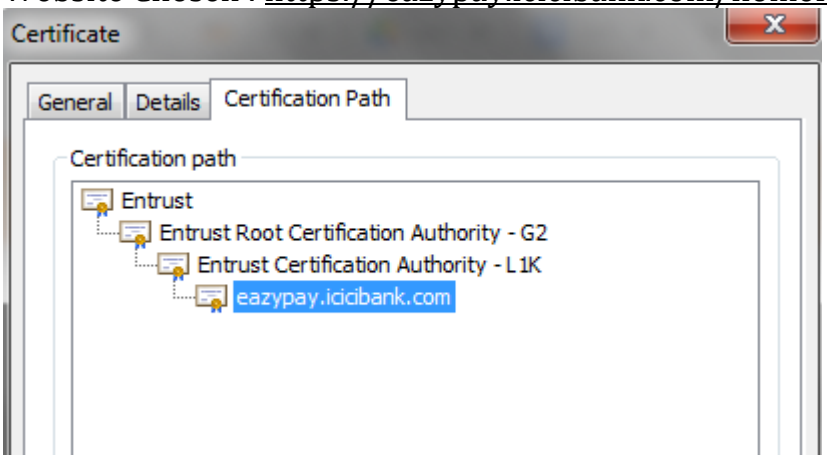
   Start

```
4 0.000158     127.0.0.1    127.0.0.1    SSLv2    171 Client Hello
```

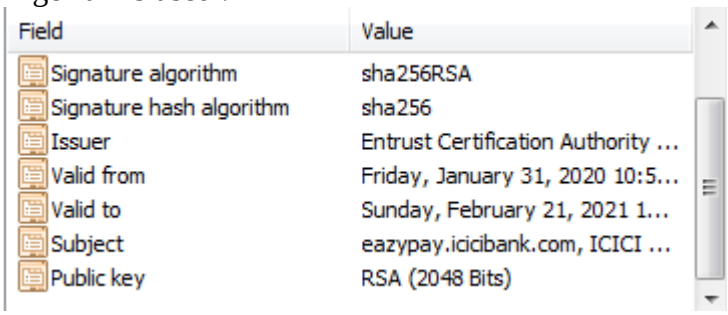
   2.943406-0.000158=2.94328

9. Examine the certificates of gmail server, any bank server. Identify the Cerification Authority, Hierarchy of Certification Authority, Validity date, crypto algorithms used for signing etc.

Website Chosen : https://eazypay.icicibank.com/homePage



Algorithms used :



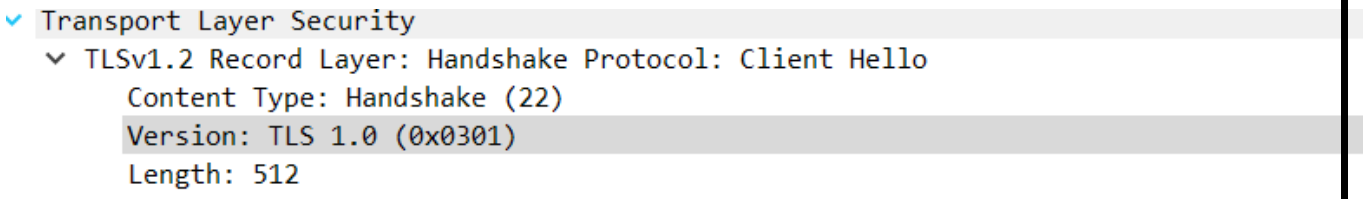
10. Enlist the differences between HTTP and HTTPS.

HTTP	HTTPS
HTTP is hypertext transfer protocol.	HTTPS stands for Hyper Text Transfer Protocol Secure.
HTTP uses port 80 for Data Communication.	HTTPS uses the port no. 443 for Data Communication.

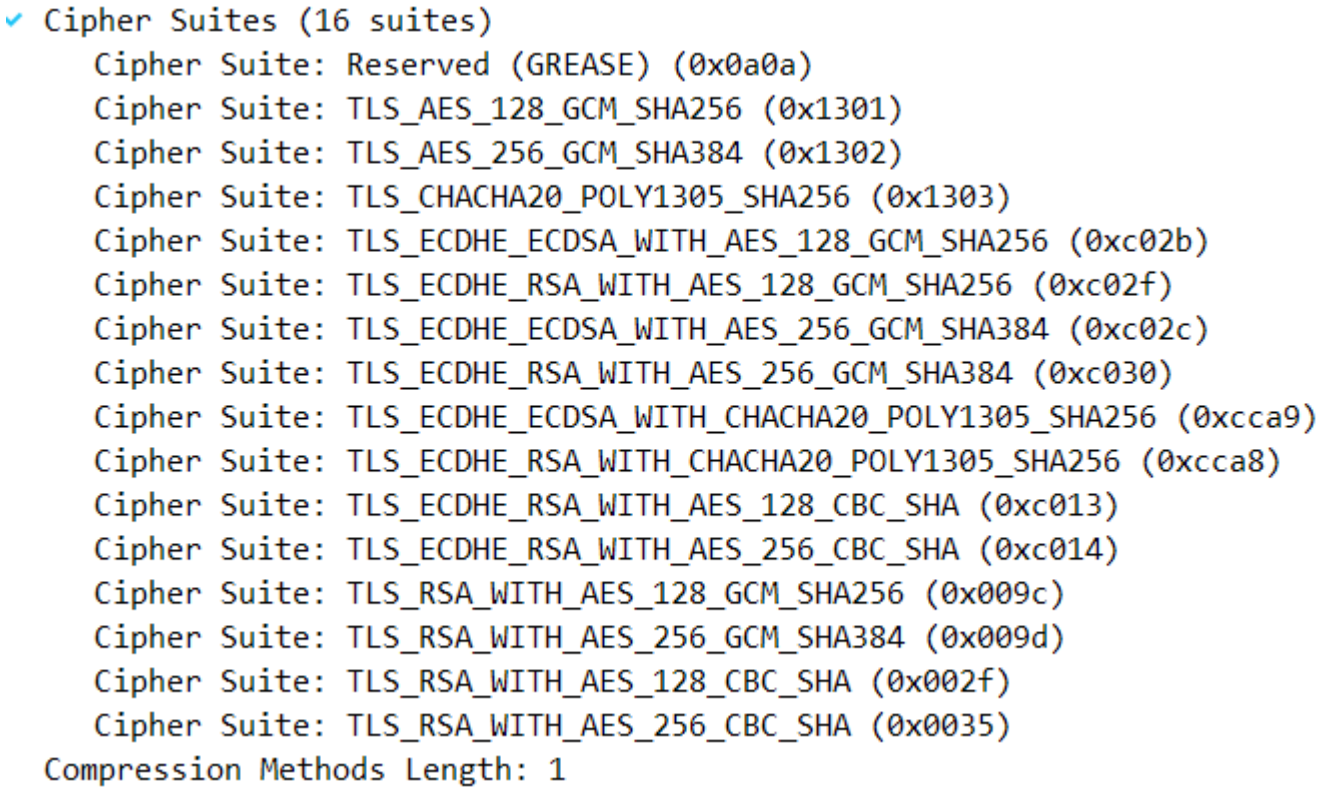
HTTP website do not need SSL.	HTTPS is a combination of SSL/TLS protocol and HTTP.
HTTP does not improve search rankings.	HTTPS helps to improve search ranking.
HTTP URLs begin with http://	HTTPs URLs begin with https://
HTTP is less secure as the data can be vulnerable to hackers.	HTTPS is designed to prevent hackers from accessing critical information. It is secure against such attacks

WORKSHEET (For packet captured for real time data

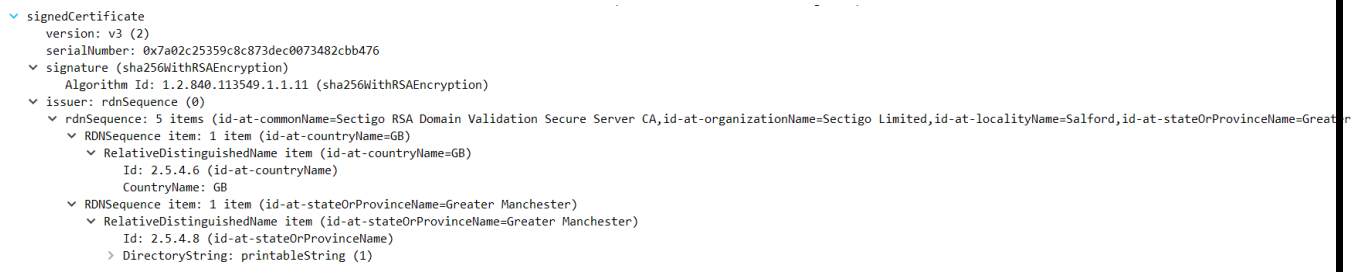
1. What version of SSL is supported by the client?
   TLS version used here is 1.0.



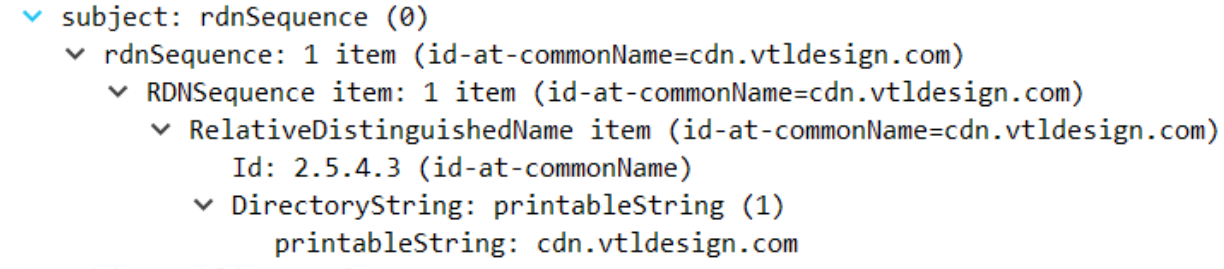
2. List the cryptographic algorithms supported by the client in Client Hello message.



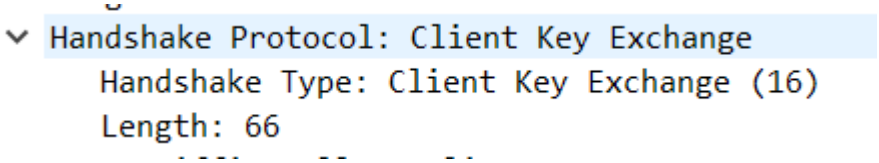
3. List the various parameters present in the public key certificate of the server.



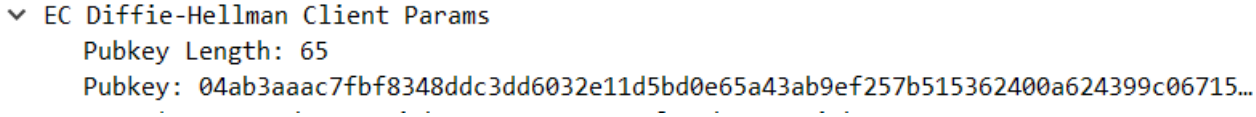
4. Identify the public key of the server? Can you trust the same?
   Yes we can trust the same because digital signature is used here.



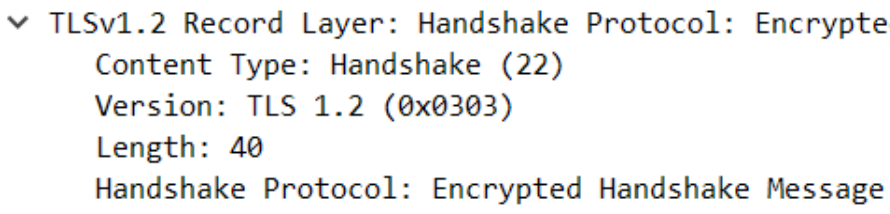
5. Identify the length of key exchanged by the Client?
   The length of the client key exchange is 66.



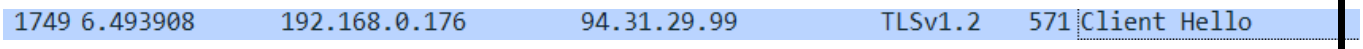
6. What algorithm is used for encrypting the session key?
   The algorithm used here is EC Diffie Hellman.



7. List the various parameters specified in the Encrypted handshake message.
   The parameters are Content Type, Version, Length, Handshake protocol.



8. Calculate the time taken for completion of the entire handshake protocol.
   Start



   End

   6.940876-6.493908=0.446968

RESULT:

The analyse the network traffic for security analysis using wireshark is verified.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Code	15	
Completion of experiment on time	5	
Documentation	5	
Simulation in Vlabs	5	
Total	30	
Signature of the faculty with Date		

Ex.No.7	**Perform password extraction, cracking and recovery from target system**

AIM:

To use open source software tools for extraction, cracking and recovery from target system.

THEORY:

Tool Selected:

John The Ripper

Office2john.py .This file helped to create hash file for the word file. This has support up to Microsoft office 2013.

About the Tool:

The tool used for

1. Cracking Linux User Password

2.Cracking Password Protected ZIP/RAR Files

3.Decrypting MD5 Hash

4.Using Wordlists To Crack Passwords

Software Requirements:

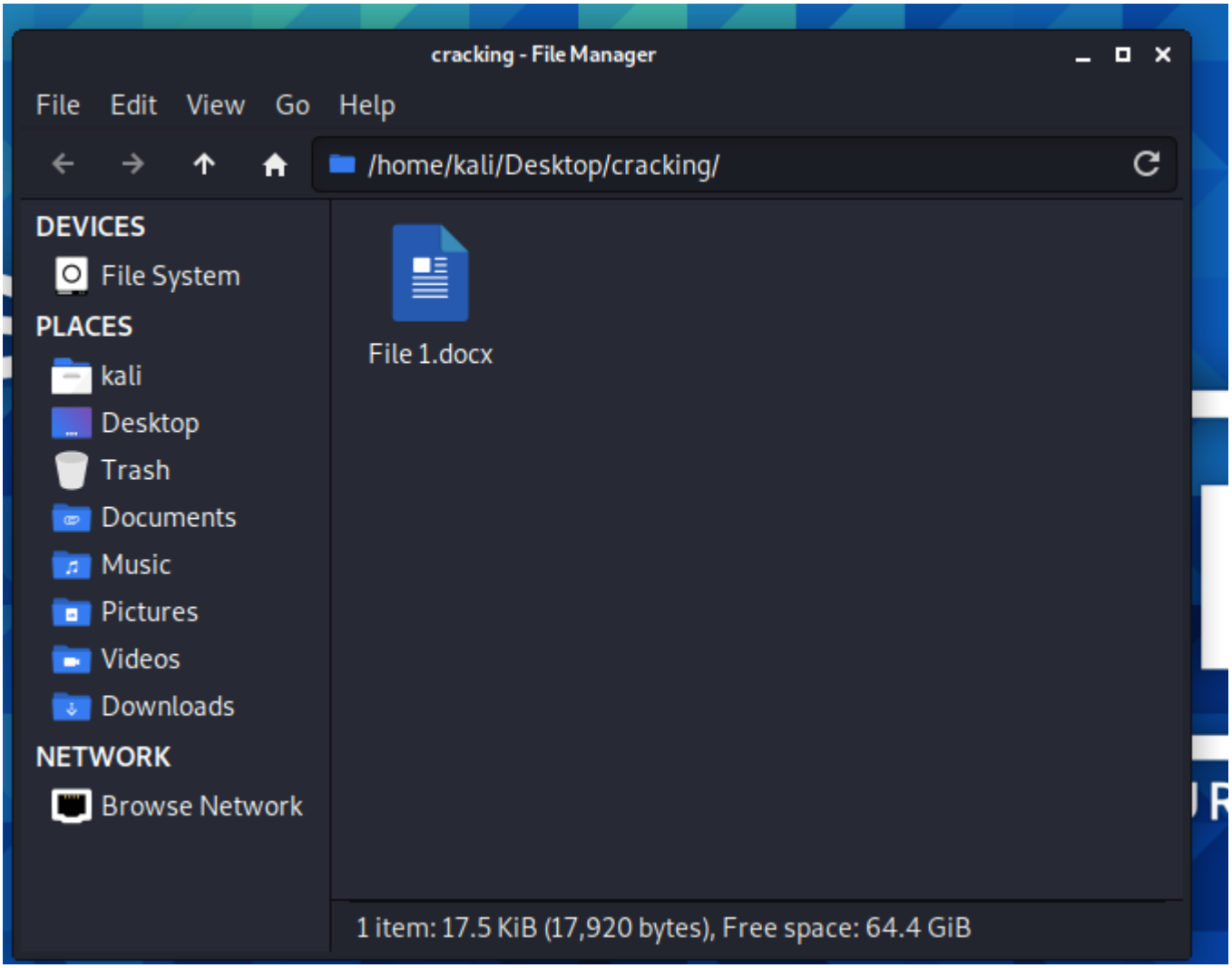
Unix based System

Dictionary attack:

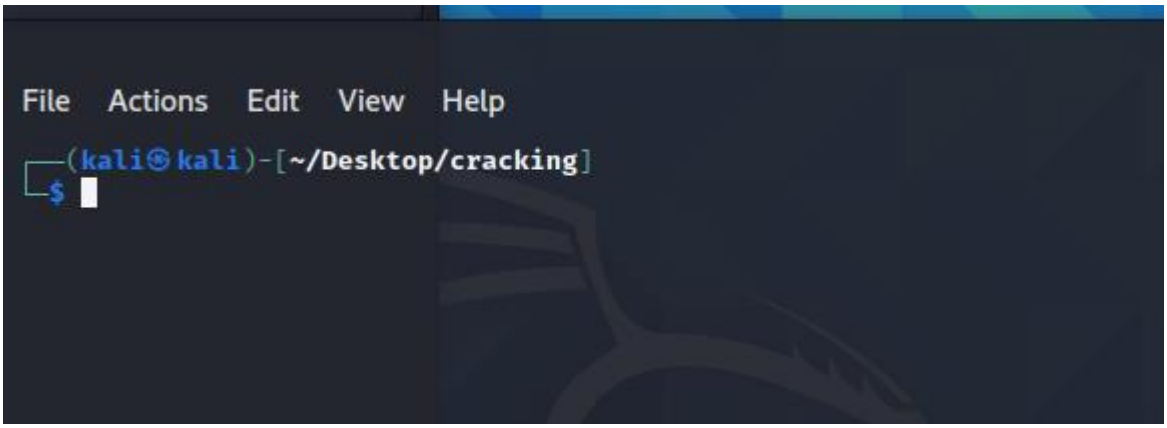
A dictionary attack is a method of breaking into a password-protected computer or server by systematically entering every word in a dictionary as a password. A dictionary attack can also be used in an attempt to find the key necessary to decrypt an encrypted message or document

Demonstrations (With Screen shots)

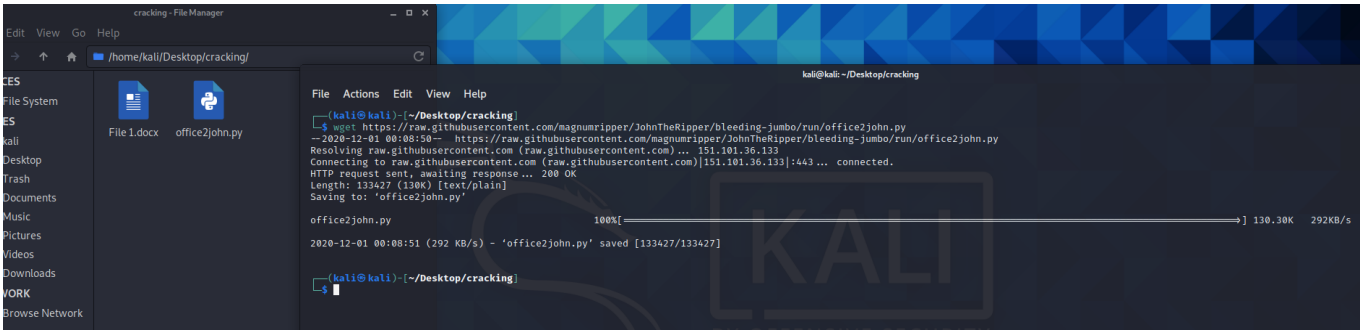
1. Create a folder and store the encrypted file in the folder.



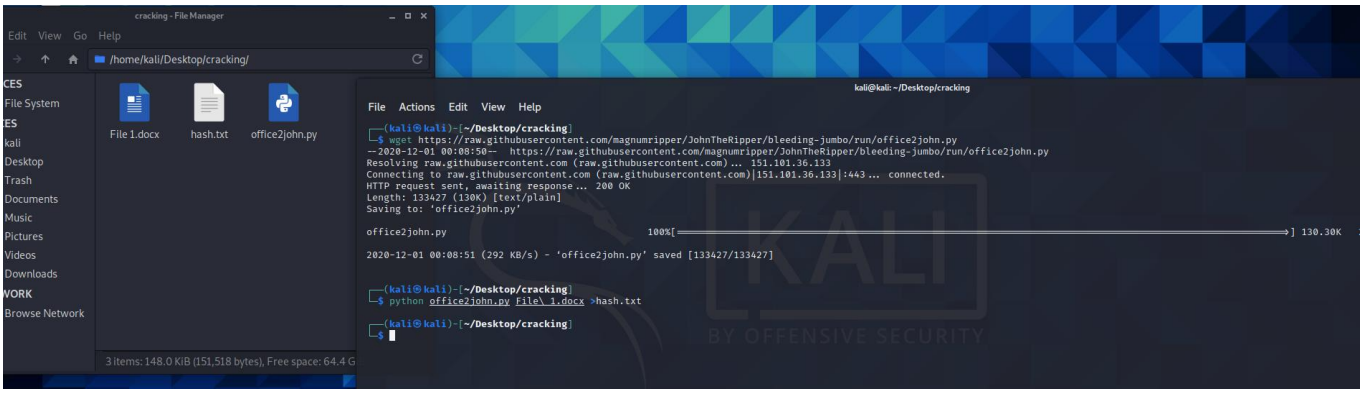
2. Open the terminal in the folder.



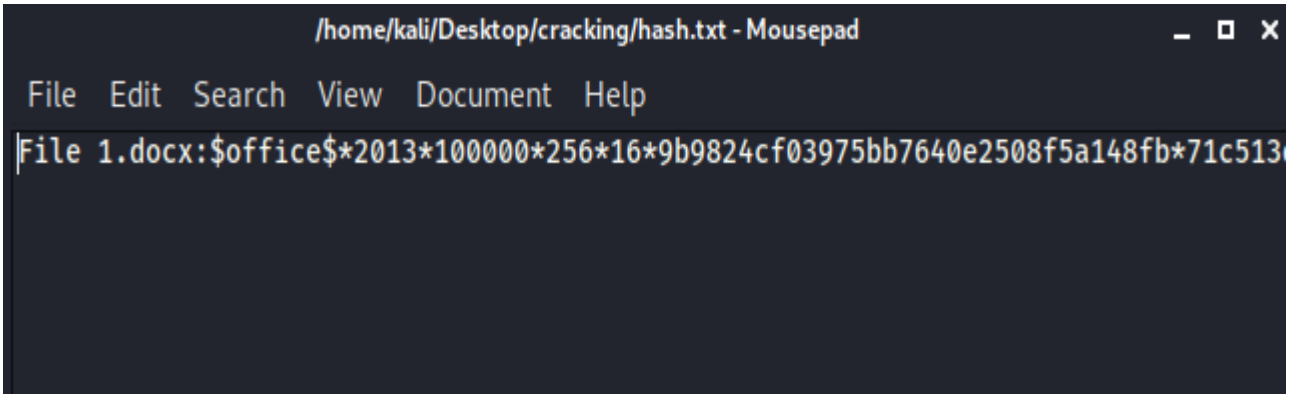
3. Download the python file using the command
   wget https://raw.githubusercontent.com/magnumripper/JohnTheRipper/bleeding-jumbo/run/office2john.py



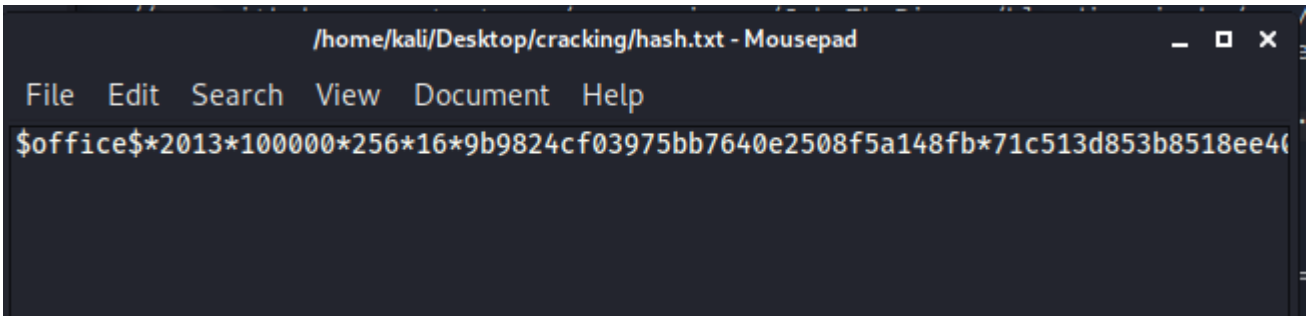
4. Run the python and pass the docx file to get hash file.



5. Open the hash file and delete upto $ office.



6. Finally hash file looks like this.



```
-a, --attack-mode             | Num  | Attack-mode, see references below
-m, --hash-type               | Num  | Hash-type, see references below
-i, --increment               |      | Enable mask increment mode
-1, --custom-charset1         | CS   | User-defined charset ?1                                | -1 ?l?d?u
-2, --custom-charset2         | CS   | User-defined charset ?2                                | -2 ?l?d?s
-3, --custom-charset3         | CS   | User-defined charset ?3                                |
-4, --custom-charset4         | CS   | User-defined charset ?4                                |

- [ Attack Modes ] -

  # | Mode
 ===+======
  0 | Straight
  1 | Combination
  3 | Brute-force
  6 | Hybrid Wordlist + Mask
  7 | Hybrid Mask + Wordlist

- [ Built-in Charsets ] -

  ? | Charset
 ===+=========
  l | abcdefghijklmnopqrstuvwxyz
  u | ABCDEFGHIJKLMNOPQRSTUVWXYZ
  d | 0123456789
  h | 0123456789abcdef
  H | 0123456789ABCDEF
  s |  !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
  a | ?l?u?d?s
  b | 0x00 - 0xff

- [ Basic Examples ] -

  Attack-          | Hash- |
  Mode             | Type  | Example command
 ==================+=======+==================================================================
  Wordlist         | $P$   | hashcat -a 0 -m 400 example400.hash example.dict
  Wordlist + Rules | MD5   | hashcat -a 0 -m 0 example0.hash example.dict -r rules/best64.rule
  Brute-Force      | MD5   | hashcat -a 3 -m 0 example0.hash ?a?a?a?a?a?a
  Combinator       | MD5   | hashcat -a 1 -m 0 example0.hash example.dict example.dict
```

7. Start cracking with hashcat function.

```
sudo hashcat -a 0 -m 9600 -o result.txt hash.txt /usr/share/wordlists/rockyou.txt
```

```
┌──(kali㉿kali)-[~/Desktop/cracking]
└─$ sudo hashcat -a 0 -m 9600 -o result.txt hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.1.1) starting...

OpenCL API (OpenCL 1.2 pocl 1.5, None+Asserts, LLVM 9.0.1, RELOC, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
==========================================================================================================================

* Device #1: pthread-Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz, 1423/1487 MB (512 MB allocatable), 4MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

Initializing backend runtime for device #1...
Host memory required for this attack: 65 MB



Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 2 secs

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit ⇒
```

After 4 mins

```
Session..........: hashcat
Status...........: Cracked
Hash.Name........: MS Office 2013
Hash.Target......: $office$*2013*100000*256*16*9b9824cf03975bb7640e250...0866e5
Time.Started.....: Tue Dec  1 03:24:58 2020 (4 mins, 28 secs)
Time.Estimated...: Tue Dec  1 03:29:26 2020 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:       91 H/s (6.02ms) @ Accel:16 Loops:1024 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 24320/14344385 (0.17%)
Rejected.........: 0/24320 (0.00%)
Restore.Point....: 24256/14344385 (0.17%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: 050291 → sentra

Started: Tue Dec  1 03:24:54 2020
Stopped: Tue Dec  1 03:29:27 2020

┌──(kali㉿kali)-[~/Desktop/cracking]
└─$ sudo cat result.txt
$office$*2013*100000*256*16*9b9824cf03975bb7640e2508f5a148fb*71c513d853b8518ee408895cfc3315af*2519c2e9167d054cc12ca951d777fae7b29a713f0b616e7abfb3d935bf0866e5:today
```

Password for File 1:
today

Hi Welcome!

RESULT:

The password cracking tool has been installed and password recovery has been done.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Tool (Installation and Exploration of Functionalities)	30	
Completion of experiment on time	10	
Documentation	10	
Total	50	
Signature of the faculty with Date		

Ex.No.8	HASHING TECHNIQUES

AIM:

To implement SHA-1 for integrity check
To explore the usage of online calculators' for hash computation

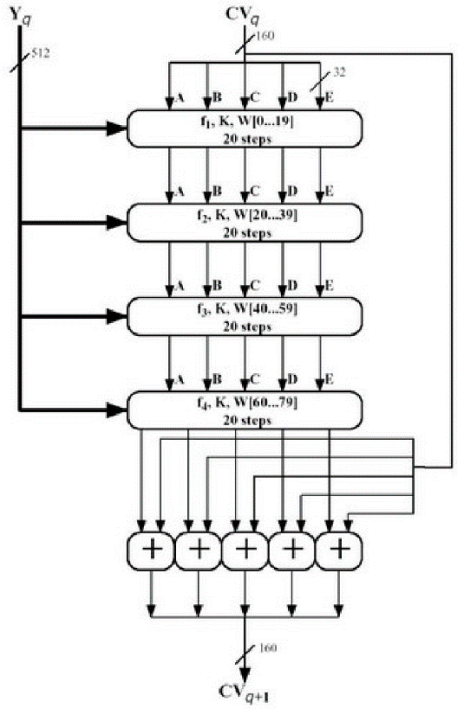
THEORY:

SHA-1 takes an input of any size and produces a 160-bit hash value known as a message digest.

Requirements of Hash Functions

1. The input can be of any length
2. The output has a fixed length of 160 bits
3. H(x) is relatively easy to compute for any given x,
4. H(x) should be one-way function

Block Diagram of One step of SHA-1



Algorithm

1. Initialize the buffer values
2. The message is then padded by appending a 1, followed by enough 0s until the message is 448 bits.

3. The length of the message represented by 64 bits is then added to the end, producing a message that is 512 bits long.
4. The padded message is then divided into 16 words each of size 32 bits
5. Start the 80 iterations of sha-1.
6. Calculate W as $W(i)=S^1(W(i-3) \oplus W(i-8) \oplus W(i-14) \oplus W(i-16))$
7. Assign

   B = A
   C = S30(B)
   D = C
   E = D
   $A = F \oplus E \oplus S^5(A) \oplus W_t \oplus K_t$
   Where K is

$$K(i) = 5A827999, \quad \text{where } 0 \le i \le 19$$

$$K(i) = 6ED9EBA1, \quad \text{where } 20 \le i \le 39$$

$$K(i) = 8F1BBCDC, \quad \text{where } 40 \le i \le 59$$

$$K(i) = CA62C1D6, \quad \text{where } 60 \le i \le 79.$$

   and F is

$$f(i;B,C,D) = (B \wedge C) \vee ((\neg B) \wedge D) \quad \text{for } 0 \ge i \ge 19$$

$$f(i;B,C,D) = B \oplus C \oplus D \quad \text{for } 20 \ge i \ge 39$$

$$f(i;B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \quad \text{for } 40 \ge i \ge 59$$

$$f(i;B,C,D) = B \oplus C \oplus D \quad \text{for } 60 \ge i \ge 79.$$

8. Repeat above for 79 times and the final result is calculated as logically OR with the buffer values. This is the message digest

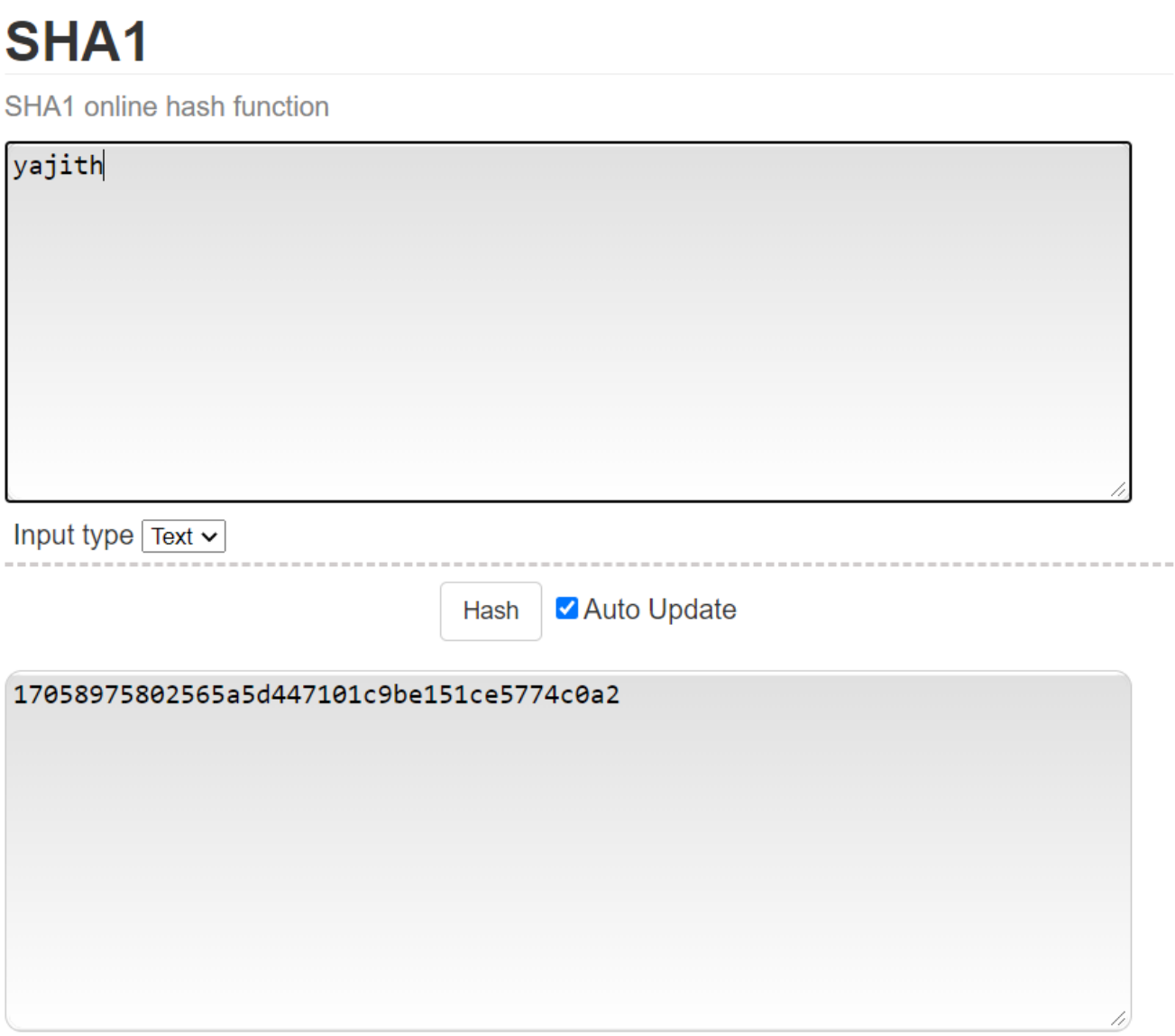
Coding

```java
package com.yajith.sha1;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
public class Main {
    public static String encryptThisString(String input)
    {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");
            byte[] messageDigest = md.digest(input.getBytes());
            BigInteger no = new BigInteger(1, messageDigest);
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            return hashtext;
        }
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }
    public static void main(String args[])
    {

        System.out.println("Enter Plain Text");
        Scanner scanner=new Scanner(System.in);
        String s1 = scanner.next();
        System.out.println("\n" + s1 + " : " + encryptThisString(s1));
    }
}
```

Output

```
Enter Plain Text
yajith vishwa s

yajith : 17058975802565a5d447101c9be151ce5774c0a2
```

Hash Computation – Use of Online Calculator



RESULT:

Thus, the program for one step of Secure Hash Algorithm -1 is implemented in Java and the results are verified.

Evaluation

Parameter	Max Marks	Marks Obtained
Originality of the code	25	
Hash Computation using Online Calculators	5	
Completion of experiment on time	10	
Documentation	10	
Total	50	
Signature of the faculty with Date		

Ex.No.9	**Exploring Standard Cryptographic libraries for Secure Communication - OpenSSL**

AIM:

To gain experience in using OpenSSL for

- Symmetric and Asymmetric encryption,
- Message digest and Hash,
- Digital signature – generation and verification

THEORY:

About OpenSSL

OpenSSL is a full-featured toolkit for the Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. It is also a general-purpose cryptography library which can be used to perform AES, RSA and hashing algorithm like SHA, MD5 and DSS. It is widely used by Internet servers, including the majority of HTTPS websites.

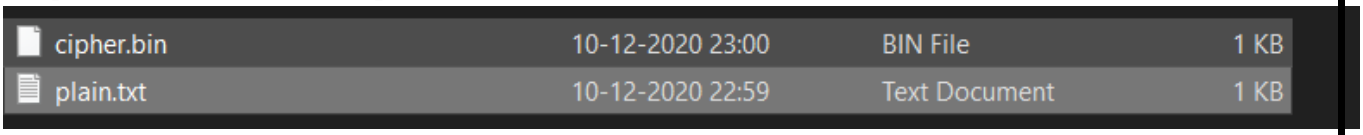
Worksheet (Answer the following ques and paste the relevant outputs)

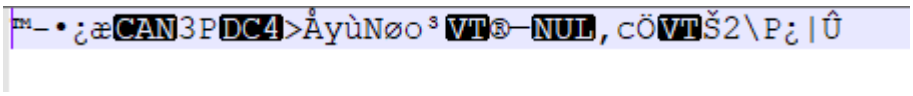
1. Create a plaintext. txt file with your name and regno.

2. Encrypt using aes in all block cipher modes of operation.
   openssl aes-128-cbc -e -in plain.txt -out cipher.bin -k "password" -nosalt

```
PS D:\5th Semester\Information Security\Lab\OpenSsl> openssl aes-128-cbc -e  -in plain.txt -out cipher.bin -k "password"
 -nosalt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
PS D:\5th Semester\Information Security\Lab\OpenSsl>
```

3. Display the contents of cipher.bin



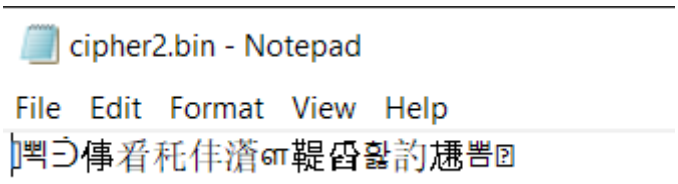


4. Decrypt the contents of cipher.bin
   openssl-1.1\x64\bin>openssl aes-128-cbc -d -in cipher1.bin -out pt.txt -k "password" –nosalt

```
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin> openssl aes-128-cbc -d  -in cipher2.bin
 -out pt.txt -k "password" -nosalt
Extra arguments given.
aes-128-cbc: Use -help for summary.
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin>
```

5. Display the contents of pt.txt

6. Examine Avalanche effect by changing one bit/charater in your plain text file



7. Generate private and public key for RSA
   openssl genrsa –out pvtkey.pem
   openssl rsa -pubout -in pvtkey.pem -out pubkey.pem

```
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin> openssl rsa -pubout -in pvtkey.pem -out
 pubkey.pem
writing RSA key
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin>
```

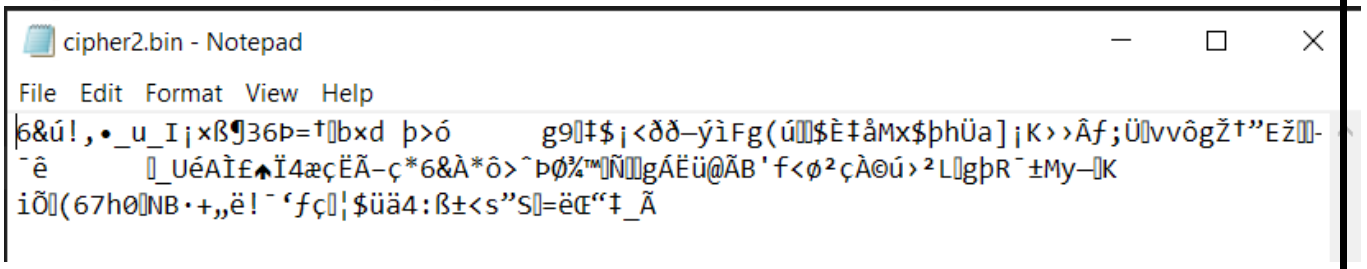
8. Display the private key in hexadecimal
   openssl rsa -text -in pvtkey.pem

```
    de:55:0e:a5:ab:0b:4f:e3
exponent2:
    00:82:43:35:3f:17:7c:5a:a6:9e:33:c9:30:98:ff:
    de:b2:74:f0:3a:02:a2:ae:3f:39:53:7d:85:12:fc:
    19:a0:e2:82:b6:82:0d:c3:3f:ca:dc:c3:aa:22:98:
    a6:ec:d4:ad:34:2d:10:a6:84:3a:3e:84:21:66:f0:
    2b:cb:24:48:79:48:5e:7a:13:5e:d0:7c:23:82:62:
    81:c3:81:30:d0:44:12:f9:51:07:cf:63:a8:66:a8:
    c2:c6:fb:80:63:32:f0:a2:a1:1f:80:3e:4a:02:44:
    83:43:c2:14:6d:30:59:73:cc:81:9e:b0:a0:10:93:
    5c:85:94:22:e0:aa:29:34:9d
coefficient:
    79:f7:1b:20:d9:ba:de:24:a6:31:2b:24:1c:24:c0:
    14:5a:d1:2f:7c:cb:4c:90:df:73:6c:a5:3f:0d:94:
    68:11:cd:72:a4:11:77:e5:71:72:4b:1e:5a:9e:60:
    7b:b9:4a:81:93:4e:fe:5a:7e:d2:86:04:17:5a:62:
    81:e9:1c:13:4f:9f:97:cb:98:79:24:56:03:db:a9:
    54:ee:90:69:8e:15:7d:a8:7b:5a:93:38:74:18:d1:
    f4:f2:79:ac:d6:11:16:cb:27:0c:ad:b8:28:c8:ca:
    40:2c:99:93:19:91:84:5b:3e:f0:d3:76:f4:29:71:
    ee:7b:33:7e:d6:68:f9:30
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAnR6uy7MFjh+U0vnzx0P5KBce2FARWAOtxu5CRxzi0iwDFBqc
eNU/rAaDqu8DS1GRnB/Sg1rOR4mApkD9dzc5KB9vWcLGWS3K4IuyNqeLhh6O+UIo
r+jyVjKkwHdT4Qjj8aG+8UefX9cUTiqeweu6qSExSuFbMwcxEFBvdXw3BCxxxmtE
LGbp8ChuSRWddC1Wf1HkOa5fducvL5afdsGXq1MtPjw2u/Sicmv66/ajPz+lWCtC
NGlA6hfdBYnpxFgC3u2/tZBt+ADGsP5reAW5NQm7I799k0o2H6UwJk9nooFJAL13
fORV3ALchHSrXgAUG7/PizekZwz5TFc12R91IQIDAQABAoIBAD0ETGeI93BRLW+d
sCGLcTgzemQQ/BtNGnB12MFbI27xbqShDQET960rvnhkZ8gZPk3+QbeHTTKU4KQv
+v8QMB83zfty26qIURBQz/3PpSyJs4ZzPfYBVhNJiZfA/48V78fHw6pZ4TfMA/Oz
M4IltJC39jeZOtyfP1uRkSdz/L2elSeE3d6Wh7FBUuruxYQ4qdqin7xVF61CEBJg
l35Y6eC4yjFIC7zsLZWC7BDCwgCF8hBrvwuZt6OfskjWXwDPwCkdVcdasGdzqbB5
YoOdRXsw6rx5gcOPxtIx5ZFMqseJB+BpG4RJ/lqR4RhxKVZcKroKiUiJWGWwWEYe
odM/8N0CgYEAzHjuuwL4P5l6/xD0unYINszyC+CiAoEOp3O+31hU6bXwMy7Gtrtc
tYmqShFDLRXIvgSmYmlqjGm2zzb72Tdvj7bf9FPAwcc4+azWquQjBGTcVEz9+Sd9
idIKXbvXX/eddgDqjNBtxMkqj3x0IOR4dGZyImpLIEjgUVZRGtu2r/MCgYEAxLbo
7OFPipslVJ8zlLzDPk1KkUtzbEF/rgWwmprihe3gUTJj0R/y9HcCbo/G1jK4Btsp
EMsdc57VQEudxIQb+FAU959nsJhDGNihtlQYsIYKuHhYD6QF9bIShWAi6ewhrhw1
kennfMzLfVv9Bcwa+HkHrPT2NABQLulu/IaZn5sCgYBkamuLagzyZvSReQlG3Rpx
izqzfJDzu9+vetddJislT5Vz/V/Xc7HuQoZfs5SS+qTINM5lfDG4XV5K60zJw0DF
mD3w33hbG5dp3URHdLYoJPXzqvXhX4RqZYvrEU8OwWYYyDNILsXn1d5efwp/yns
DcLEbptU3zeVQ6lqwtP4wKBgQCCQzU/F3xapp4zyTCY/96ydPA6AqKuPzlTfYUS
Bmg4oK2gg3DP8rcw6oimKbs1K00LRCmhDo+hCFm8CvLJEh5SF56E17QfCOCYoHD
TDQRBL5UQfPY6hmqMLG+4BjMvCioR+APkoCRINDwhRtMFlzzIGesKAQk1yFlCLg
ik0nQKBgHn3GyDZut4kpjErJBwkwBRa0S98y0yQ33NspT8NlGgRzXKkEXflcXJL
lqeYHu5SoGTTv5aftKGBBdaYoHpHBNPn5fLmHkkVgPbqVTukGmOFX2oe1qTOHQY
fTyeazWERbLJwytuCjIykAsmZMZkYRbPvDTdvQpce57M37WaPkw
-----END RSA PRIVATE KEY-----
```

9. Perform encryption using RSA public key
   openssl rsautl -encrypt -in plain.txt -pubin -inkey pubkey.pem -out c1.bin

```
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin> openssl rsautl -encrypt -in plain.txt
pubin -inkey pubkey.pem -out .\cipher2.bin
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin>
```

10. Display the contents of encrypted file

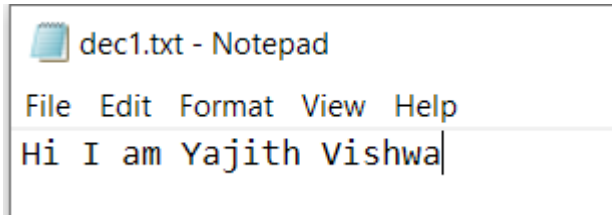


11. Decrypt the result using RSA private key

```
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin> openssl rsautl -decrypt -in cipher2.bi
  -inkey pvtkey.pem -out dec1.txt
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin>
```

12. Display the contents of decrypted file

openssl rsautl -decrypt -in c1.bin -inkey pvtkey.pem -out dec1.txt



13. Generate the hash of a file using MD5

openssl md5 plain.txt

```
PS D:\5th Semester\Information Security\Lab\OpenSsl> openssl md5 plain.txt
MD5(plain.txt)= 77ef2e5802958ba1208ba64e78c253b2
```

14. Generate the hash of a file using SHA256

Openssl SHA256 plain.txt

```
PS D:\5th Semester\Information Security\Lab\OpenSsl> Openssl SHA256 plain.txt
SHA256(plain.txt)= 620405756976f5c14b2d9c038e47261dffa4a0567e9f5f2c684c9a81be042163
PS D:\5th Semester\Information Security\Lab\OpenSsl>
```

15. *Generate Signature using SHA and RSA*

openssl dgst -sha1 -sign pvtkey.pem -out s.bin plain.txt

```
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin> openssl dgst -sha1 -sign pvtkey.pem -o
t s.bin plain.txt
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin>
```

16. *Verify the signature*

openssl dgst -sha1 -verify pubkey.pem -signature s.bin plain.txt

```
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin> openssl dgst -sha1 -verify pubkey.pem
signature s.bin plain.txt
Verified OK
PS D:\5th Semester\Information Security\Lab\OpenSsl\openssl\openssl-1.1\x64\bin>
```

RESULT:

Thus the libraries for

- Symmetric and Asymmetric encryption,
- Message digest and Hash,
- Digital signature – generation and verification

are utilized for securing the communication

Evaluation

Parameter	Max Marks	Marks Obtained
Originality of the work	30	
Completion of experiment on time	10	
Documentation	10	
Total	50	
Signature of the faculty with Date		

Ex.No.10	SQL Injection – Damn Vulnerable Web Application

AIM:

To demonstrate SQL injection attack using Damn Vulnerable Web **Application** (DVWA).

THEORY:

About DVWA:-

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a class room environment.

DVWA is a PHP/MySQL web application, whose main goal is to be an aid for security professionals to test their skills and tools in a legal environment.

SQL injection Attack:-

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior.

Ways to prevent SQL injection:

***a*)Input validation:-** The validation process is aimed at verifying whether or not the type of input submitted by a user is allowed. Input validation makes sure it is the accepted type, length, format, and so on. Only the value which passes the validation can be processed. It helps counteract any commands inserted in the input string. In a way, it is similar to looking to see who is knocking before opening the door.

b) Parametrized queries

Parameterized queries are a means of pre-compiling an SQL statement so that you can then supply the parameters in order for the statement to be executed. This method makes it possible for the database to recognize the code and distinguish it from input data.

The user input is automatically quoted and the supplied input will not cause the change of the intent, so this coding style helps mitigate an SQL injection attack.

c) Stored procedures

Stored procedures (SP) require the developer to group one or more SQL statements into a logical unit to create an execution plan. Subsequent executions allow statements to be automatically parameterized. Simply put, it is a type of code that can be stored for later and used many times.

So, whenever you need to execute the query, instead of writing it over and over, you can just call the stored procedure.

d) Escaping

Always use character-escaping functions for user-supplied input provided by each database management system (DBMS). This is done to make sure the DBMS never confuses it with the SQL statement provided by the developer.

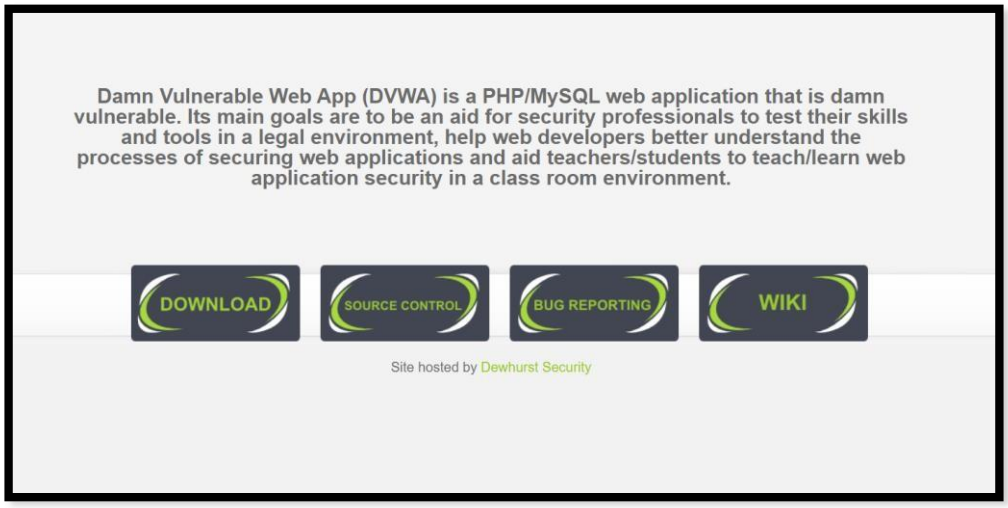
For example, use the mysql_real_escape_string() in PHP to avoid characters that could lead to an unintended SQL command.

Worksheet (Answer the following ques and paste the relevant outputs)

1) Installation procedure for WAMP and DVWA:-

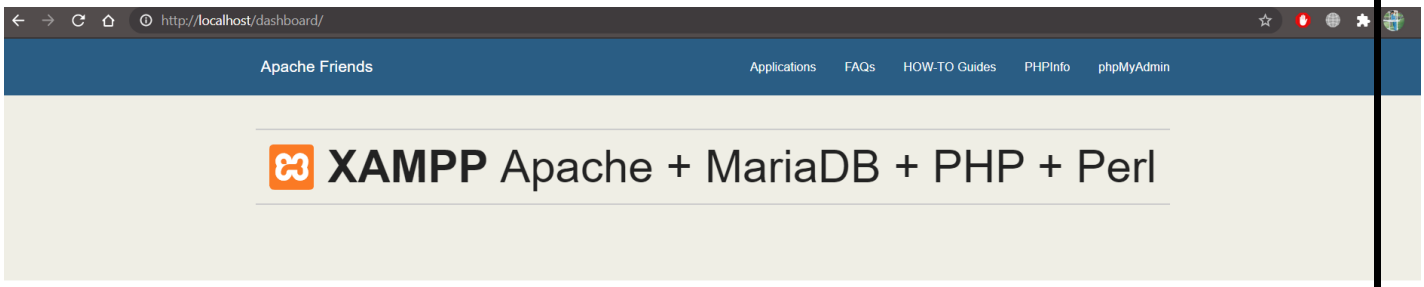
Download Wamp and click next and install it.

2) Download DVWA and extract it:-



3) Copy DVWA Folder to WAMP/www.

4) Open Wamp-Localhost.



Welcome to XAMPP for Windows 7.4.9

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.

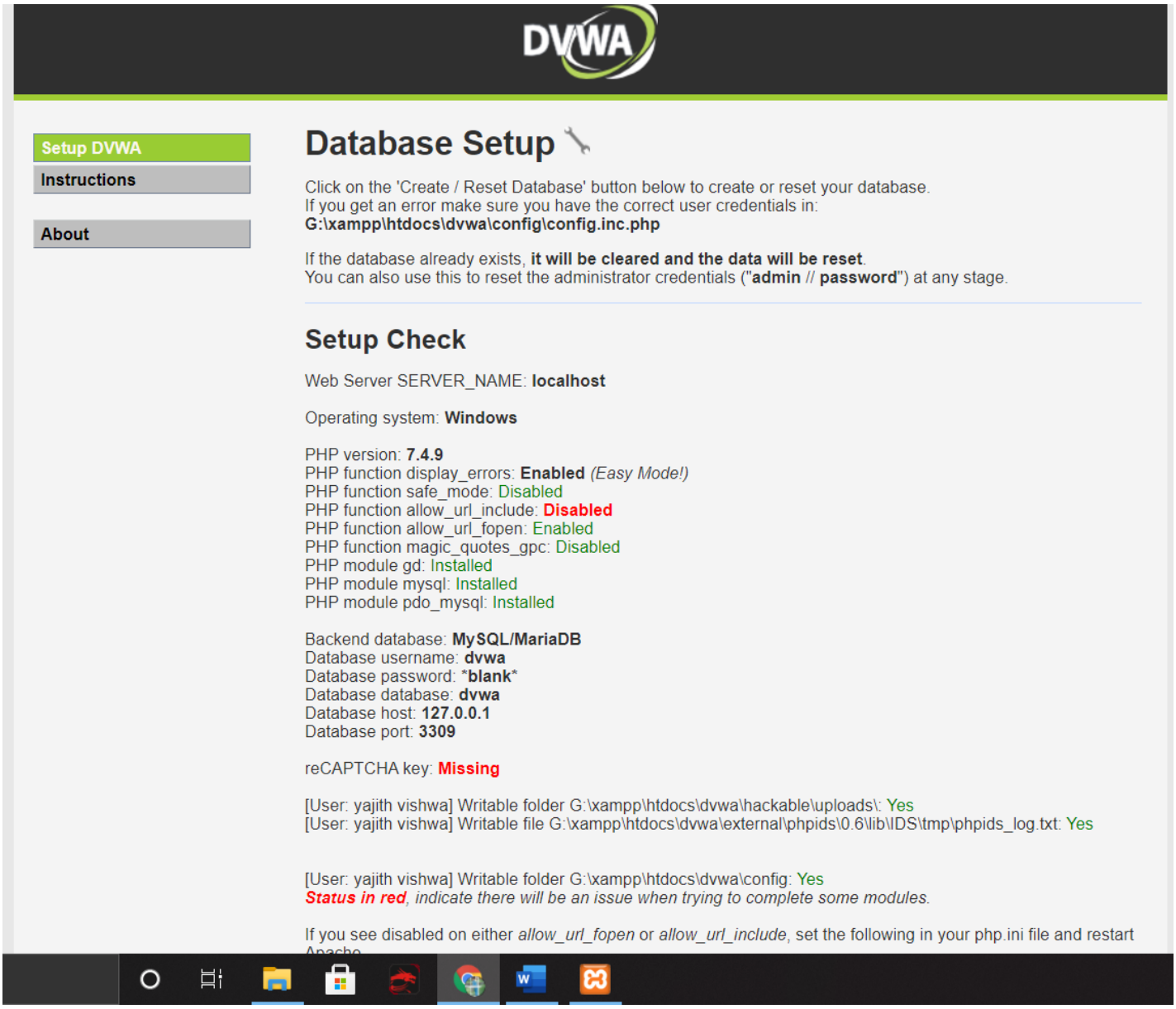
Start the XAMPP Control Panel to check the server status.

Community

XAMPP has been around for more than 10 years – there is a huge community behind it. You can get involved by joining our Forums, adding yourself to the Mailing List, and liking us on Facebook, following our exploits on Twitter, or adding us to your Google+ circles.

Contribute to XAMPP translation at translate.apachefriends.org.

5) In any browser type localhost/dvwa.


DVWA
Setup DVWA
Instructions
About
Database Setup
Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in:
G:\xampp\htdocs\dvwa\config\config.inc.php
If the database already exists, it will be cleared and the data will be reset.
You can also use this to reset the administrator credentials ("admin // password") at any stage.
Setup Check
Web Server SERVER_NAME: localhost
Operating system: Windows
PHP version: 7.4.9
PHP function display_errors: Enabled (Easy Mode!)
PHP function safe_mode: Disabled
PHP function allow_url_include: Disabled
PHP function allow_url_fopen: Enabled
PHP function magic_quotes_gpc: Disabled
PHP module gd: Installed
PHP module mysql: Installed
PHP module pdo_mysql: Installed
Backend database: MySQL/MariaDB
Database username: dvwa
Database password: *blank*
Database database: dvwa
Database host: 127.0.0.1
Database port: 3309
reCAPTCHA key: Missing
[User: yajith vishwa] Writable folder G:\xampp\htdocs\dvwa\hackable\uploads\: Yes
[User: yajith vishwa] Writable file G:\xampp\htdocs\dvwa\external\phpids\0.6\lib\IDS\tmp\phpids_log.txt: Yes
[User: yajith vishwa] Writable folder G:\xampp\htdocs\dvwa\config: Yes
Status in red, indicate there will be an issue when trying to complete some modules.
If you see disabled on either allow_url_fopen or allow_url_include, set the following in your php.ini file and restart

Instructions:

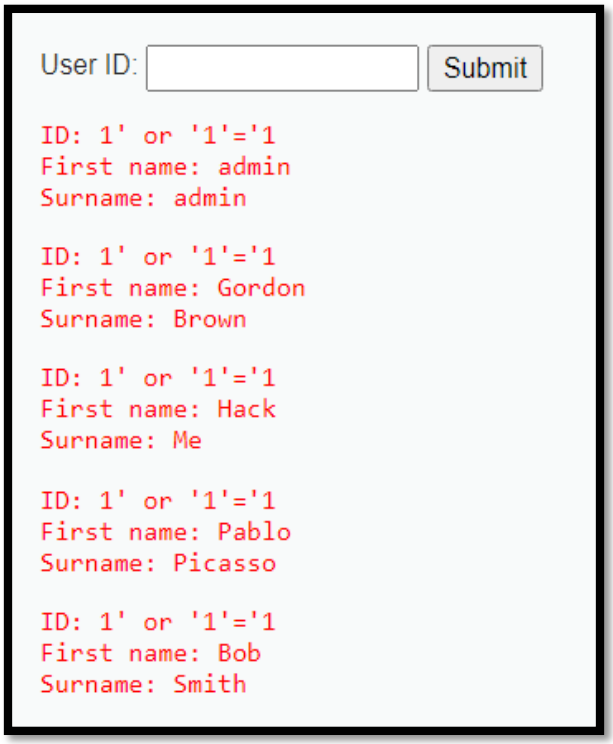
- Open Mozilla Firefox in you system.
- In URL bar, type localhost/dvwa/setup.php and then click on "Create/Reset Database"
- Again type localhost/dvwa . Press enter key.
- Now you should see DVWA login page. Credentials for login page are:
  - **Username : admin**
  - **Password : password**
- After logging in, go to "DVWA security" section. There is a drop down box on that page.
- You can set the security level of this web application accordingly. Set it "low" which is the least secure mode.
- Now move to SQL injection section, by clicking "sql injection" button on page. And try SQL injection there.

1. Check expected results:

SELECT first_name, last_name FROM users WHERE user_id = '1'

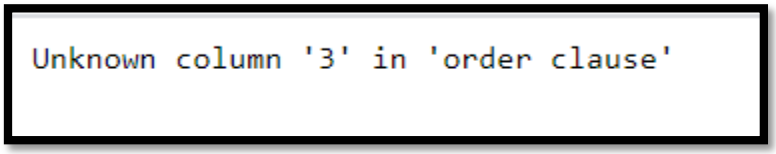
2. Check the results of an OR True statement

SELECT first_name, last_name FROM users WHERE user_id = '1' or '1'='1'



3. Find the number of columns in table:

SELECT first_name, last_name FROM users WHERE user_id = 'a' ORDER BY 3;#'



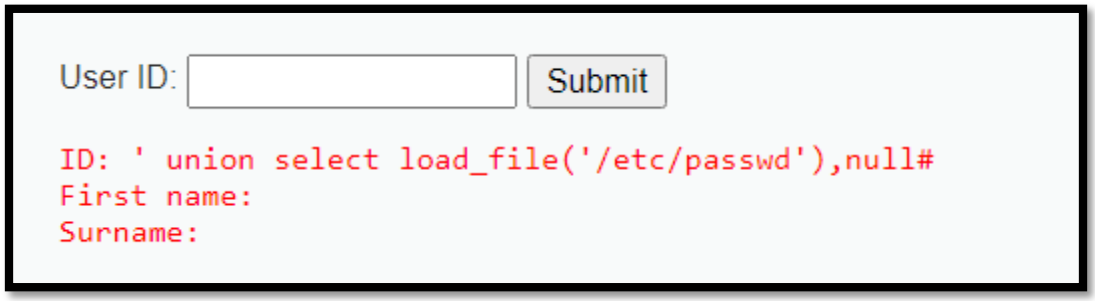
So No of column is 2.

4. Find Hostname:

SELECT first_name, last_name FROM users WHERE user_id = ' ' union select null,@@hostname#'

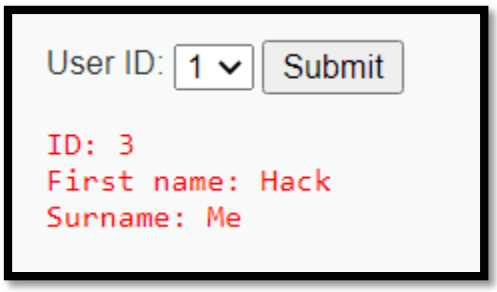
5. Display File:

SELECT first_name, last_name FROM users WHERE user_id = ' ' union select load_file('/etc/passwd'),null#'



6. Try to do SQL Injection on DVWA with security level set to "low/Medium" and find out all schema name from database.

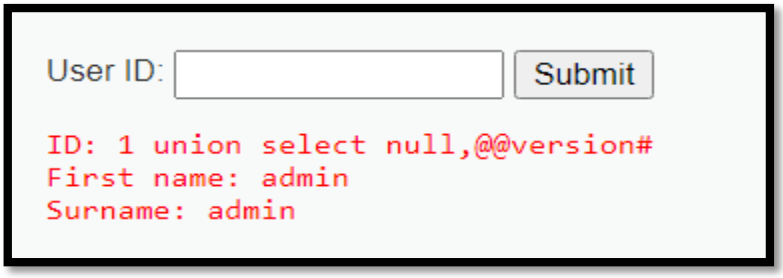
1 or 1=1 union select null, table_name from information_schema.tables#



For medium it is restricted to select only the User ID.

7. Try to do SQL Injection on DVWA with security level set to "low/Medium" and find out mysql version.

1 union select null,@@version#



For medium it is restricted to select only the User ID.

8. Analyze the source code for security on "low" and "medium" level. Comment on the ways by which SQL injection is prevented.

Ans:-

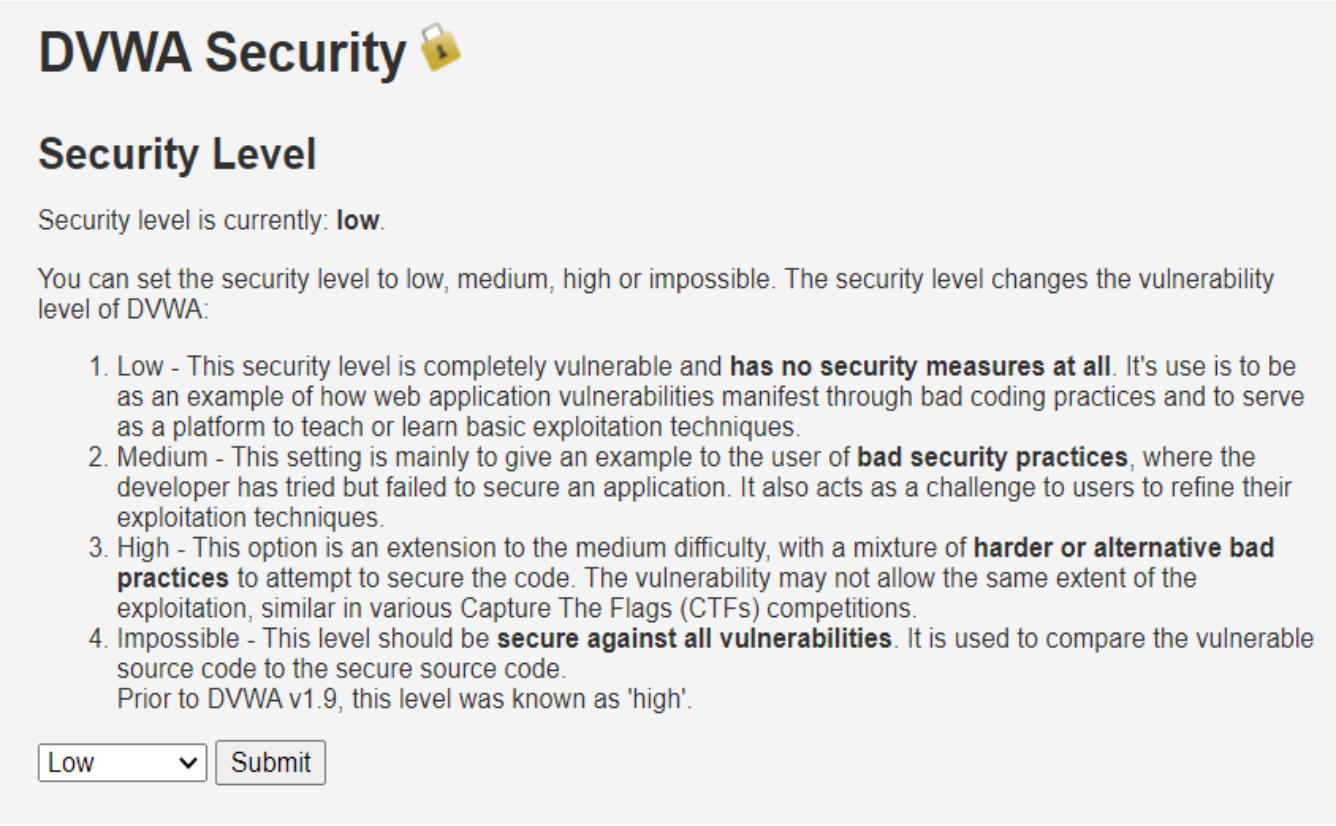
Every vulnerability has four different security levels, low, medium, high and impossible. The security levels give a challenge to the 'attacker' and also shows how each vulnerability can be counter measured by secure coding.

Medium:-

This security level's purpose is to give the 'attacker' a challenge in exploitation and also serve as an example of bad coding/security practices.This security level is basically to a case for the user having awful coding practices, where the developer is attempt but neglect to secure an application.It is also used to test the skills of client to refine their vulnerable techniques.

Low:-

This security level is meant to simulate a website with no security at all implemented in their coding. It gives the 'attacker' the chance to refine their exploitation skills.
This level contain no security i.e most vulnerable level. Programmer gives bad coding practice.



Compared to low; **medium level security** protects against various vulnerabilities.

RESULT:

Thus SQL injection attack has been demonstrated using Damn Vulnerable Web Application (DVWA).

Evaluation

Parameter	Max Marks	Marks Obtained
Originality of the work	30	
Completion of experiment on time	10	
Documentation	10	
Total	50	
Signature of the faculty with Date		

Ex.No.11	Buffer Overflow

AIM:

To demonstrate Buffer flow attack

THEORY:

Buffer overflow Attack

A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code.

Stack-based buffer overflows, which are more common among attackers, exploit applications and programs by using what is known as a stack: memory space used to store user input.

Heap-based attacks are harder to carry out and involve flooding the memory space allocated for a program beyond memory used for current runtime operations.

Ways to prevent Buffer overflow

- **Address space randomization** (ASLR) randomly moves around the address space locations of data regions. Typically, buffer overflow attacks need to know the locality of executable code, and randomizing address spaces makes this virtually impossible.
- **Data execution prevention** flags certain areas of memory as non-executable or executable, which stops an attack from running code in a non-executable region.
- **Structured exception handler overwrite protection (SEHOP)** helps stop malicious code from attacking Structured Exception Handling (SEH), a built-in system for managing hardware and software exceptions. It thus prevents an attacker from being able to make use of the SEH overwrite exploitation technique.

Demonstration

Normal Case:

```
#include <stdio.h>
#include <stdlib.h>
void main()
```

```
{
char *name;
char *command;
name=(char * ) malloc (10);
command=(char *) malloc (128);
printf("Ente r your name:");
gets(name );
printf("Hello %s\n",name);
system(command);
}
```

Input and Output:
Input: yajith
Output: Hello yajith

```
main.c:10:1: warning: 'gets' is deprecated [-Wdeprecated-declarations]
/usr/include/stdio.h:638:14: note: declared here
main.c:(.text+0x3b): warning: the `gets' function is dangerous and should not be used.
Ente r your name:yajith
Hello yajith
```

Buffer Overflow :
Input:

```
Ente r your name:11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111yajith
```

Output:

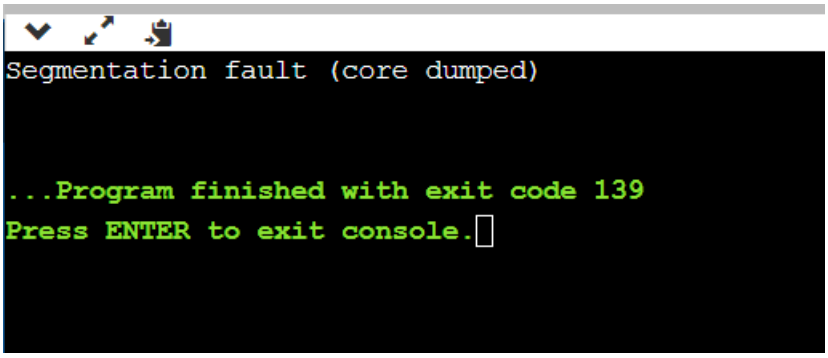
```
/usr/include/stdio.h:638:14: note: declared here
main.c:(.text+0x3b): warning: the `gets' function is dangerous and should not be used.
Ente r your name:11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111yajith
Hello 1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111yajith
sh: 1: 111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111yajith: File name too long
```

Inferences:
The input string executed the linux command in a shell because the input exceeded the size and overflows the buffer.

Stack based Overflow:

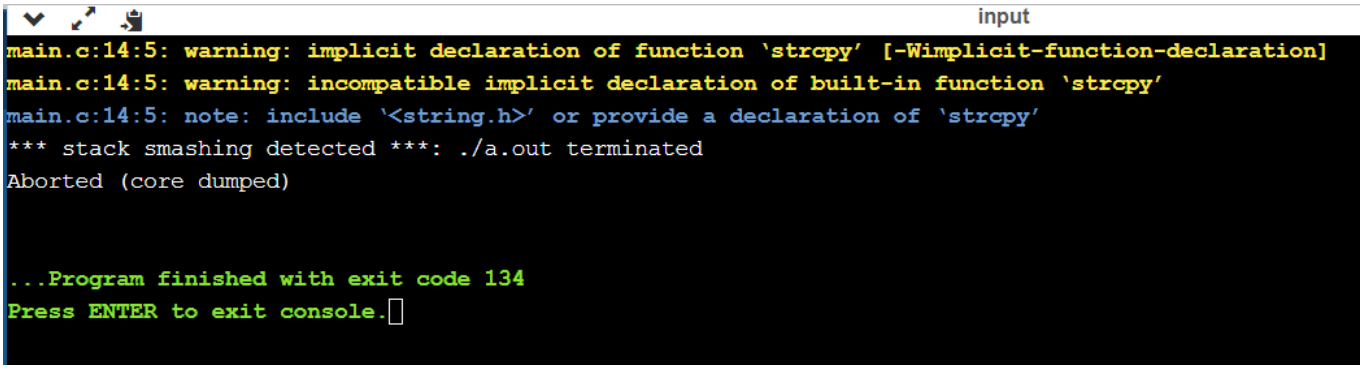
```
#include <stdio.h>
```

```
#include <stdlib.h>
void fun(int x)
{
   if (x == 1)
     return;
   x = 2;
   fun(x);
}
int main()
{
   fun(25);
   return 0;
}
```



Heap based Overflow:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   char buff[10] = {0};
   strcpy(buff, "IS Lab buffer overflow attack example");
   return 0;
}
```



RESULT:

Thus, Bufferoverflow has been demonstrated with a sample program

Evaluation

Parameter	Max Marks	Marks Obtained
Originality of the work	30	
Completion of experiment on time	10	
Documentation	10	
Total	50	
Signature of the faculty with Date		

Ex.No.12	**Configuration of Firewalls in system environment**

AIM:

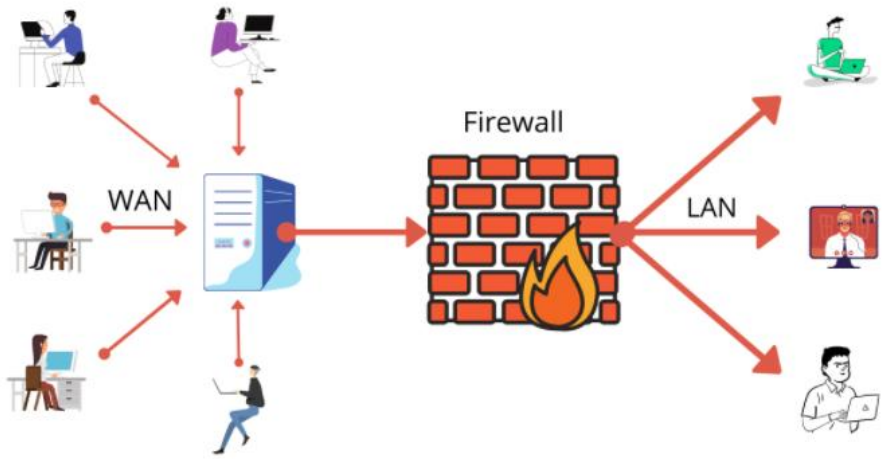
To configure the firewall in Windows operating system environment.

THEORY:

Firewall

A **firewall** is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules.

A **firewall** typically establishes a barrier between a trusted network and an untrusted network, such as the Internet.



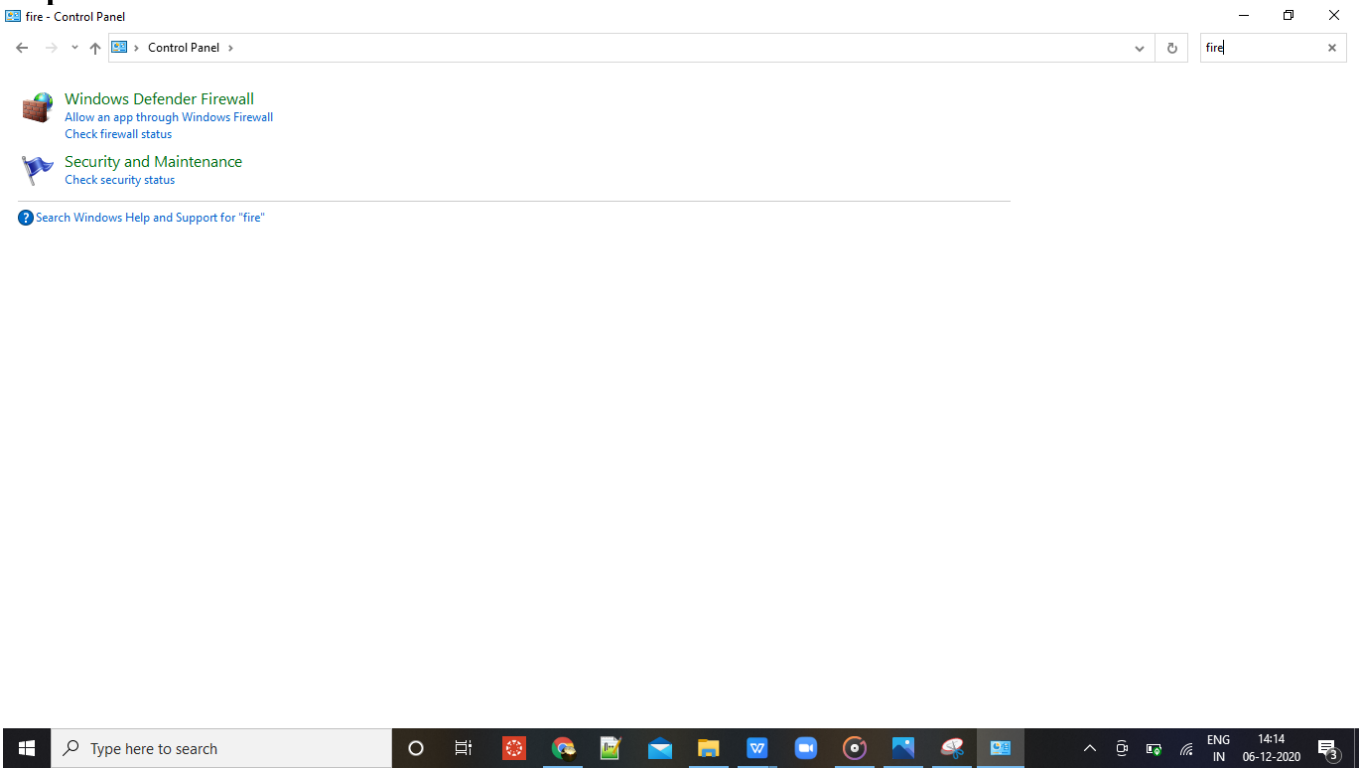
Types of Firewall

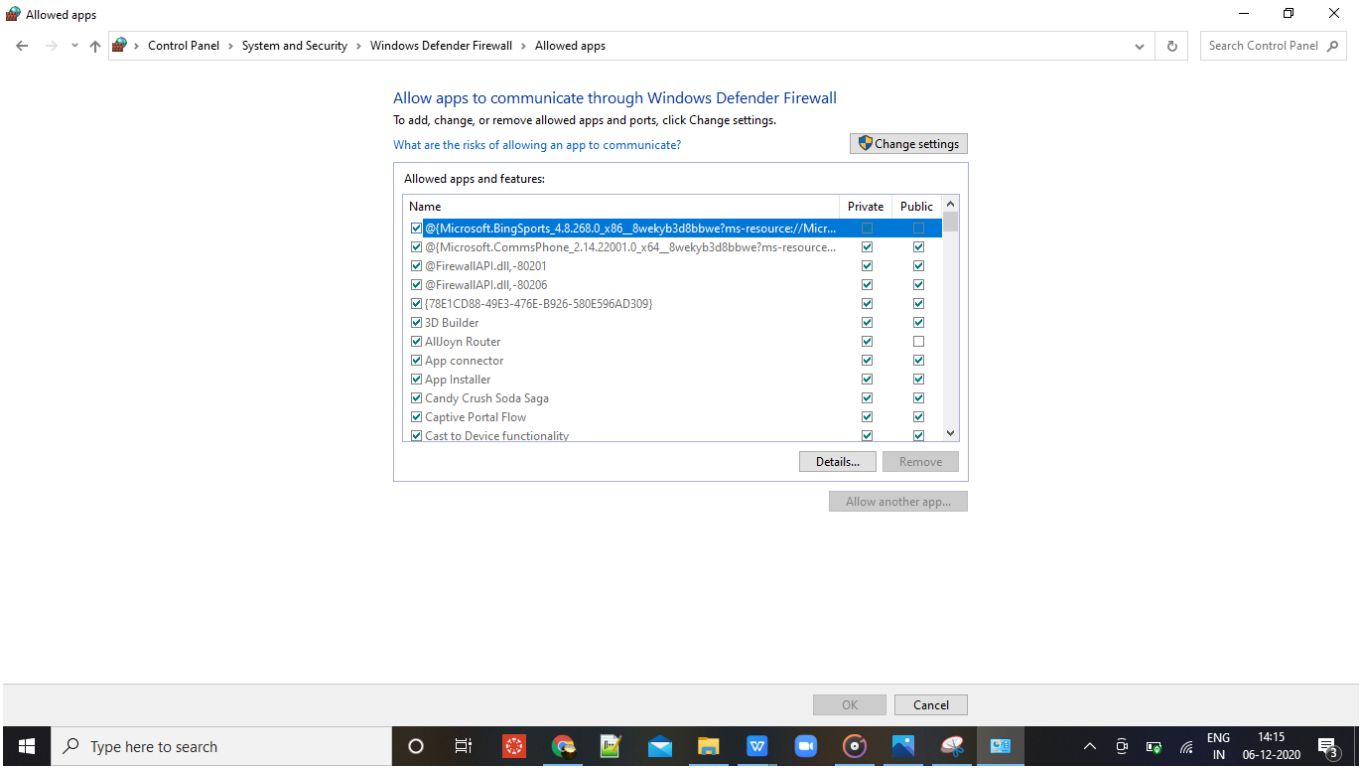
- Packet-filtering firewalls.
- Circuit-level gateways.
- Stateful inspection firewalls.
- Application-level gateways (a.k.a. proxy firewalls)
- Next-gen firewalls.
- Software firewalls.
- Hardware firewalls.
- Cloud firewalls.

Screenshots of Configuration Steps

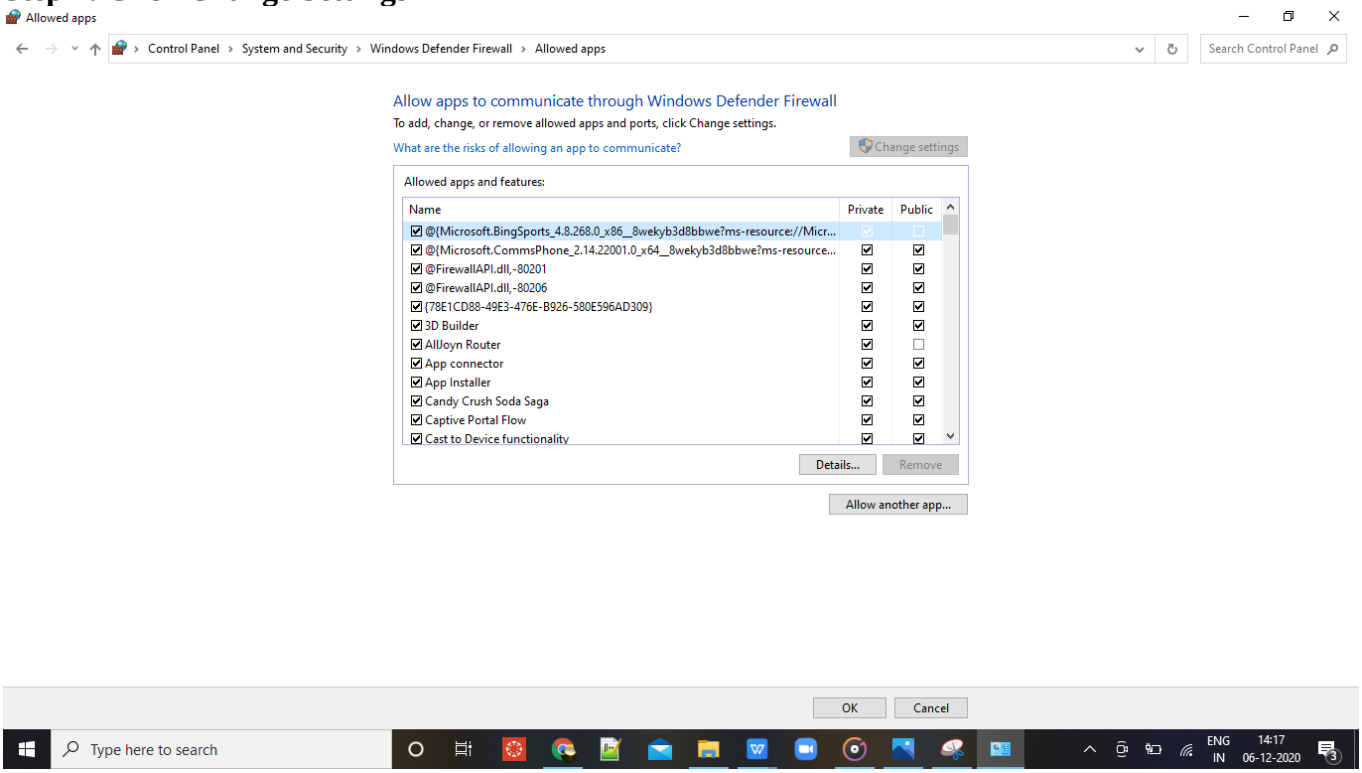
To open Windows Firewall we can go to Start > Control Panel > Windows Firewall.

Step 1: Control Panel

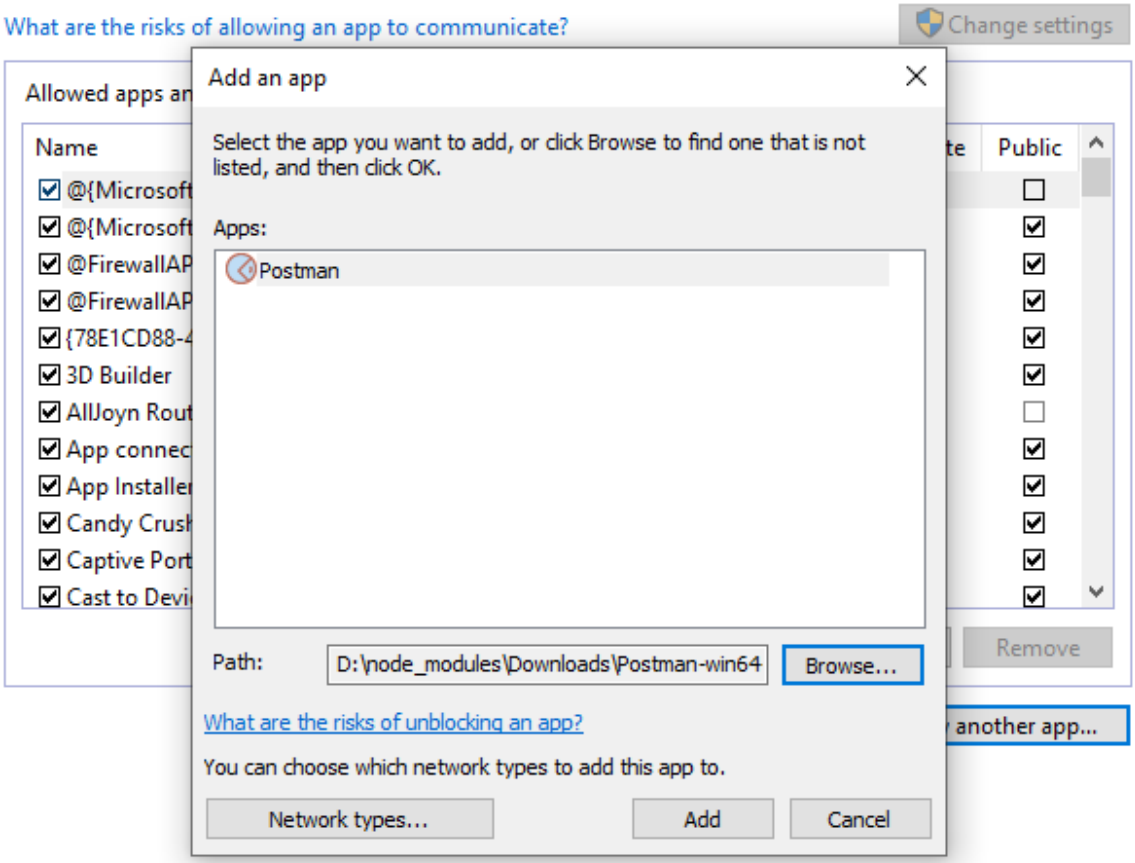




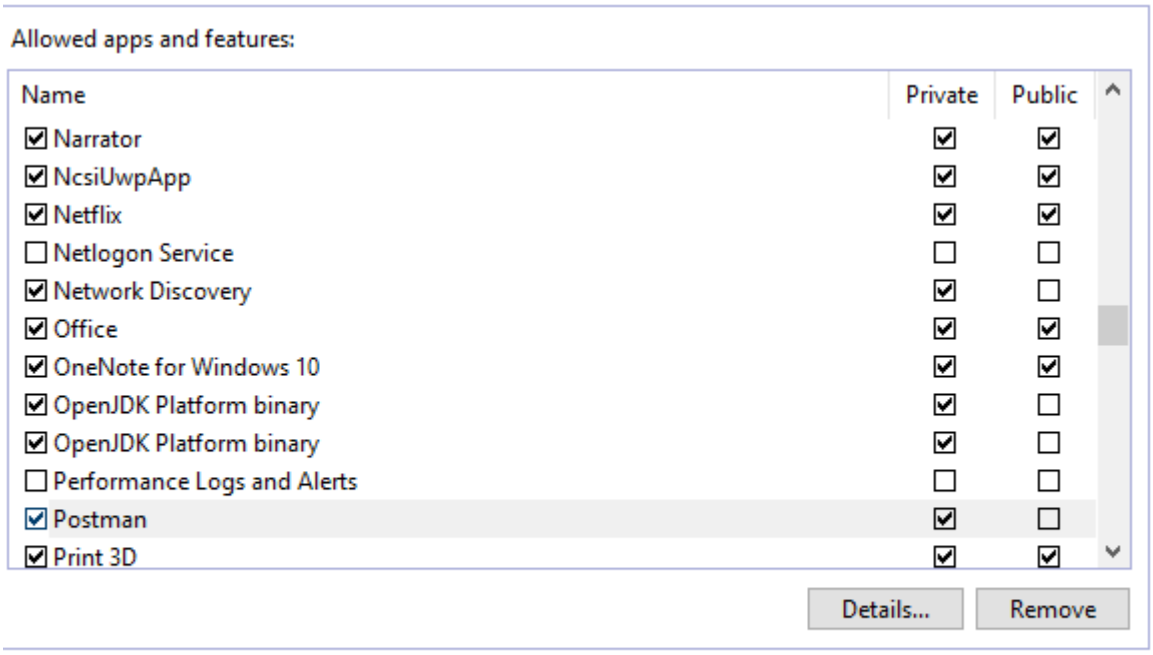
Step 2: Click Change Settings

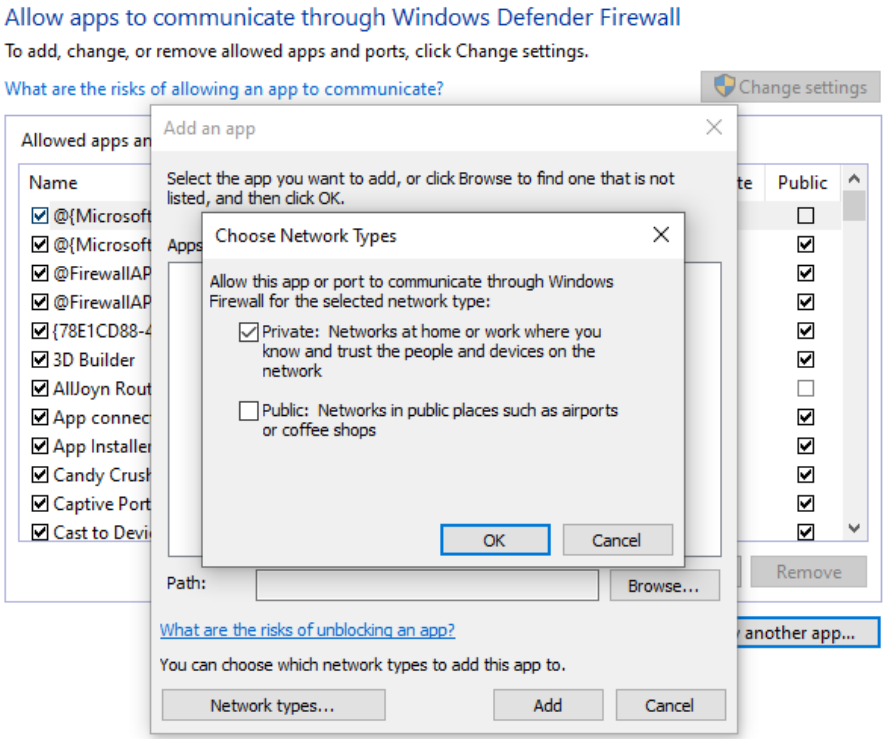


Step 3: add programs to configure

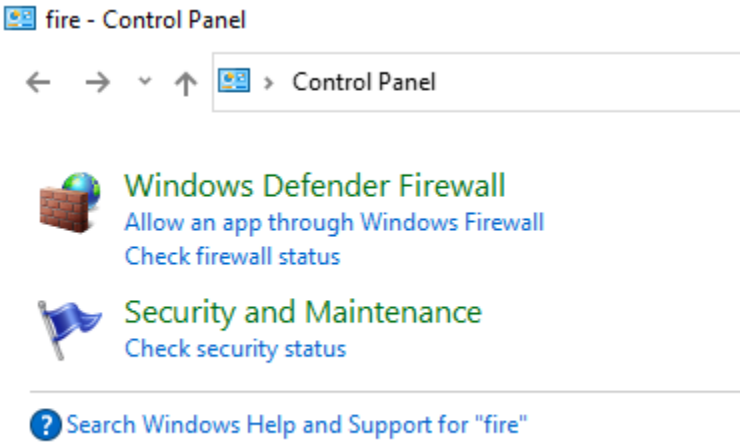


Postman Is Added


Allowed apps and features:
Name
Private
Public
Narrator
NcsiUwpApp
Netflix
Netlogon Service
Network Discovery
Office
OneNote for Windows 10
OpenJDK Platform binary
OpenJDK Platform binary
Performance Logs and Alerts
Postman
Print 3D
Details...
Remove
Allow another app...


Allow apps to communicate through Windows Defender Firewall
To add, change, or remove allowed apps and ports, click Change settings.
What are the risks of allowing an app to communicate?
Change settings
Add an app
Select the app you want to add, or click Browse to find one that is not listed, and then click OK.
Choose Network Types
Allow this app or port to communicate through Windows Firewall for the selected network type:
Private: Networks at home or work where you know and trust the people and devices on the network
Public: Networks in public places such as airports or coffee shops
OK
Cancel
Path:
Browse...
What are the risks of unblocking an app?
You can choose which network types to add this app to.
Network types...
Add
Cancel
Name
Public
Remove
another app...

Step 4: Click the Status of firewall



Step 5:

RESULT:

Thus Firewall has been configured for Windows operating systems environment.

Evaluation

Parameter	Max Marks	Marks Obtained
Originality of the work	30	
Completion of experiment on time	10	
Documentation	10	
Total	50	
Signature of the faculty with Date		