SIES COLLEGE OF ARTS, SCIENCE AND COMMERCE (AUTONOMOUS)

Sion (W), Mumbai 400022.

INS Practical

CLASS: TYBSC	**SUBJECT: COMPUTER SCIENCE**
PAPER: INS	**SUBJECT CODE: SIUSCS54**
YEAR: 2022-2023	**SEMESTER: V**
NAME: PRATHAMESH DEVKATE	**ROLL NO.: TCS2223017**

INDEX

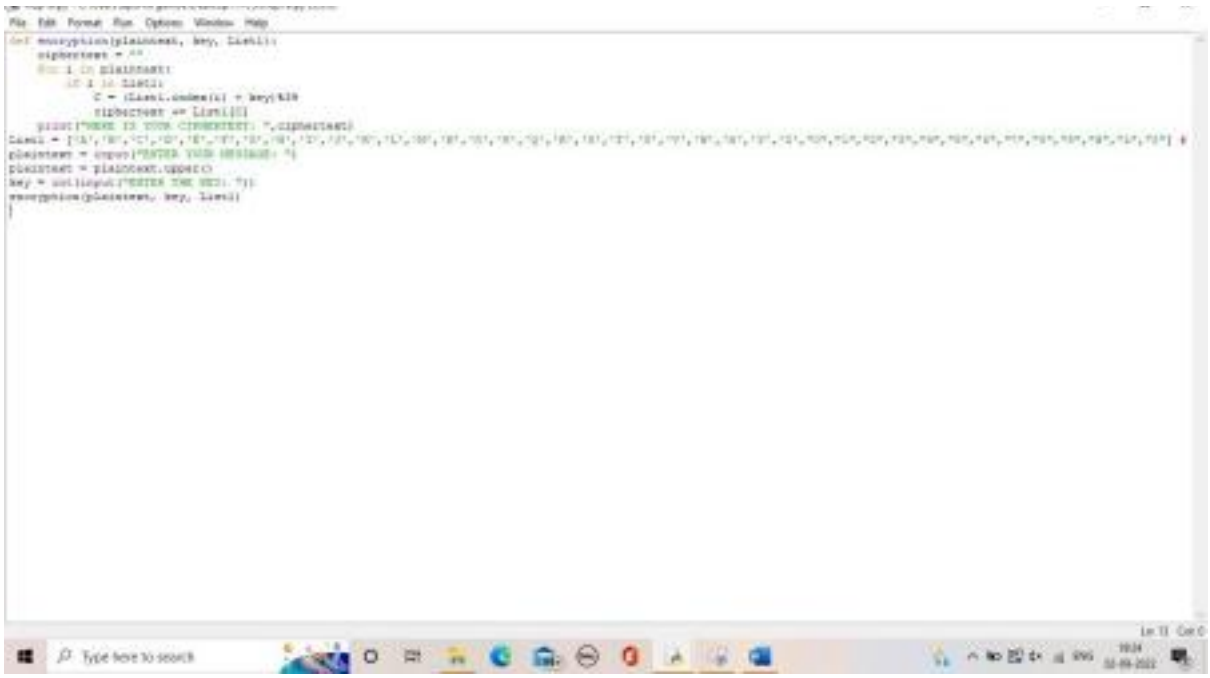
SR NO.	TOPIC	SIGN
1	Write programs to implement the following Substitution Cipher Techniques: 1)Caesar Cipher 2)Monoalphabetic Cipher	
2	Write programs to implement the following Substitution Cipher Techniques: 1)Vernam Cipher 2)Playfair Cipher	
3	Write programs to implement the following Transposition Cipher Techniques: 1) Rail Fence Cipher 2)Simple Columnar Technique	
4	Write program to encrypt and decrypt strings using: 1)DES Algorithm 2)AES Algorithm	
5	Write a program to implement RSA algorithm to perform encryption / decryption of a given string.	
6	Write a program to implement the Diffie Hellman Key Agreement algorithm to generate symmetric keys.	
7	Write a program to implement the MD5 algorithm compute the message digest	
8	Write a program to calculate HMAC-SHA1 Signature 9 Write a program to implement SSL.	
9	Configure Windows Firewall to block: 1)A port 2)An Program 3)A website	

INS PRACTICAL NO 1A

AIM: WRITE A PRAGRAM TO IMPLEMENT THE FOLLOWING SUBSTITUTION CIPHER TECHNIQUES.

1) CAESER CIPHER

CODE:



OUTPUT:

```
ENTER YOUR MESSAGE: hellol2#
ENTER THE KEY: 2
HERE IS YOUR CIPHERTEXT:  JGNNQ34$
>>>
```

b) MONOALPHABETIC CIPHER

CODE:

```
// Java Program to Implement the Monoalphabetic Cypher


import java.io.*;
class GFG {

public static char normalChar[]

                = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i',

                        'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',

                        's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };
```

```
public static char codedChar[]
                = { 'Q', 'W', 'E', 'R', 'T', 'Y', 'U', 'I', 'O',
                    'P', 'A', 'S', 'D', 'F', 'G', 'H', 'J', 'K',
                    'L', 'Z', 'X', 'C', 'V', 'B', 'N', 'M' };

// Function which returns encrypted string public static
String stringEncryption(String s)
{
        // initializing an empty String
        String encryptedString = "";

        // comparing each character of the string and
        // encoding each character using the indicesfor
        (int i = 0; i < s.length(); i++) {
            for (int j = 0; j < 26; j++) {

                // comparing the character and
                // adding the corresponding char
                // to the encryptedString
                if (s.charAt(i) == normalChar[j])
                {
                    encryptedString += codedChar[j];
                    break;
                }

                // if there are any special characters
                // add them directly to the string
                if (s.charAt(i) < 'a' || s.charAt(i) > 'z')
                {
                    encryptedString += s.charAt(i);
                    break;
                }
            }
        }
```

```
                // return encryptedString
                return encryptedString;
}

// Function which returns descryptedString public static
String stringDecryption(String s)
{

                // Initializing the string String
                decryptedString = "";

                // Run the for loop for total stringfor
                (int i = 0; i < s.length(); i++)
                {
                        for (int j = 0; j < 26; j++) {

                                // compare each characters and decode them
                                // using indices
                                if (s.charAt(i) == codedChar[j])
                                {
                                        decryptedString += normalChar[j];
                                        break;
                                }

                                // Add the special characters directly to
                                // the String
                                if (s.charAt(i) < 'A' || s.charAt(i) > 'Z')
                                {
                                        decryptedString += s.charAt(i);
                                        break;
                                }
                        }
                }
```

```
                    // return the decryptedString
                    return decryptedString;
}
public static void main(String args[])
{
                    String str = "Welcome to geeksforgeeks";

                    // print plain text System.out.println("Plain
                    text: " + str);

                    // Changing whole string to lower case
                    // function call to stringEncryption and storing in
                    // encryptedString
                    String encryptedString = stringEncryption(str.toLowerCase());

                    // printing encryptedString
                    System.out.println("Encrypted message: "
                                                + encryptedString);

                    // function call to stringDecryption and printing
                    // the decryptedString
                    System.out.println("Decrypted message: "
                            + stringDecryption(encryptedString));

}
 }
```

OUTPUT:

```
java -cp /tmp/PTQp0DAlvZ GFG
Plain text: Hello
Encrypted message: ITSSG
Decrypted message: hello
```

INS PRACTICAL NO 2A

AIM: WRITE A PRAGRAM TO IMPLEMENT THE FOLLOWING SUBSTITUTION CIPHER TECHNIQUES.

1) VERNAM CIPHER

CODE:

OUTPUT:

```
hellohellohellohe
edqjssxjo sga ast mountains are bae
None
>>>
```

INS PRACTICAL NO 2B

PLAYFAIR CIPHER

CODE:

```
    import java.awt.Point;
    import java.util.Scanner;
public class PlayfairCipher4. {
    //length of digraph array
    private int length = 0;
    //creates a matrix for Playfair cipher
    private String [][] table;
    //main() method to test Playfair method
    public static void main(String args[])
{
. PlayfairCipher pf = new PlayfairCipher();
```

```
    }
        //main run of the program, Playfair method
        //constructor of the class
        private PlayfairCipher()
    {
        //prompts user for the keyword to use for encoding & creates
        tables System.out.print("Enter the key for playfair cipher: ");
        Scanner sc = new Scanner(System.in);
String key = parseString(sc);
        while(key.equals(""))
        key = parseString(sc);
        table = this.cipherTable(key);
        //prompts user for message to be encoded
        System.out.print("Enter the plaintext to be encipher: ");
        //System.out.println("using the previously given
        keyword"); String input = parseString(sc);
        while(input.equals(""))
        input = parseString(sc);
        //encodes and then decodes the encoded message
        String output = cipher(input);
        String decodedOutput = decode(output);
        //output the results to user
        this.keyTable(table);
        this.printResults(output,decodedOutput);
     }
        //parses an input string to remove numbers,
        punctuation, //replaces any J's with I's and makes string
        all caps
        private String parseString(Scanner sc)
     {
        String parse = sc.nextLine();
        //converts all the letters in upper case
        parse = parse.toUpperCase();
        //the string to be substituted by space for each match (A to
        Z) parse = parse.replaceAll("[^A-Z]", "");
        //replace the letter J by I
        parse = parse.replace("J", "I");
        return parse;
     }
        //creates the cipher table based on some input string (already
        parsed) private String[][] cipherTable(String key)
    {
        //creates a matrix of 5*5
        String[][] playfairTable = new String[5][5];
```

```
        String keyString = key +
        "ABCDEFGHIKLMNOPQRSTUVWXYZ"; //fill string array with
        empty string
    .for(int i = 0; i < 5; i++)
    for(int j = 0; j < 5; j++)
        playfairTable[i][j] = "";
        for(int k = 0; k < keyString.length(); k++)
    {
        boolean repeat = false;
        boolean used = false;
    for(int i = 0; i < 5; i++){
    for(int j = 0; j < 5; j++)
    {
    if(playfairTable[i][j].equals("" +
    keyString.charAt(k))) {
    repeat = true;
    }
    else if(playfairTable[i][j].equals("") && !repeat &&
    !used) {
playfairTable[i][j] = "" + keyString.charAt(k);
used = true;
    }
    }
    }
    . }
    return playfairTable;
    }
        //cipher: takes input (all upper-case), encodes it, and returns the
        output private String cipher(String in)
    {
        length = (int) in.length() / 2 + in.length() % 2;
//insert x between double-letter digraphs & redefines "length"88.
    for(int i = 0; i < (length - 1); i++)
    {
    . if(in.charAt(2 * i) == in.charAt(2 * i + 1))
    {
in = new StringBuffer(in).insert(2 * i + 1,
        'X').toString(); length = (int) in.length() / 2 +
        in.length() % 2;
    }
    }
    //------------makes plaintext of even length-------------
        - //creates an array of digraphs
        String[] digraph = new String[length];
```

```
. //loop iterates over the plaintext
            for(int j = 0; j < length ; j++)
   {
   //checks the plaintext is of even length or not
   if(j == (length - 1) && in.length() / 2 == (length - 1))
   //if not addends X at the end of the plaintext
   in = in + "X";
   digraph[j] = in.charAt(2 * j) +""+ in.charAt(2 * j + 1);
   }
   //encodes the digraphs and returns the output
   String out = "";
   String[] encDigraphs = new String[length];
   encDigraphs = encodeDigraph(digraph);
   for(int k = 0; k < length; k++)
   out = out + encDigraphs[k];
   return out;
   }
   // encryption logic
   //encodes the digraph input with the cipher's specifications
   private String[] encodeDigraph(String di[])
   {
   String[] encipher = new String[length];
   for(int i = 0; i < length; i++)
   {
   char a = di[i].charAt(0);
   char b = di[i].charAt(1);
   int r1 = (int) getPoint(a).getX();
   int r2 = (int) getPoint(b).getX();
   int c1 = (int) getPoint(a).getY();
   int c2 = (int) getPoint(b).getY();
   //executes if the letters of digraph appear in the same row
   //in such case shift columns to right
   if(r1 == r2)
   {
   c1 = (c1 + 1) % 5;
   c2 = (c2 + 1) % 5;
   }
   //executes if the letters of digraph appear in the same column
   //in such case shift rows down
   else if(c1 == c2)
   {
   r1 = (r1 + 1) % 5;
```

```
r2 = (r2 + 1) % 5;
}
//executes if the letters of digraph appear in the different row and diffe
//in such case swap the first column with the second column else
{
int temp = c1;
c1 = c2;
c2 = temp;
}
//performs the table look
encipher[i] = table[r1][c1] + "" + table[r2][c2];
}
return encipher;
}
// decryption logic
158. // decodes the output given from the cipher and decode methods
(opp
. of encoding process)
private String decode(String out)
{
String decoded = "";
for(int i = 0; i < out.length() / 2; i++)
{
char a = out.charAt(2*i);
char b = out.charAt(2*i+1);
int r1 = (int) getPoint(a).getX();
int r2 = (int) getPoint(b).getX();
int c1 = (int) getPoint(a).getY();
int c2 = (int) getPoint(b).getY();
if(r1 == r2)
{
c1 = (c1 + 4) % 5;
c2 = (c2 + 4) % 5;
}
else if(c1 == c2)
{
r1 = (r1 + 4) % 5;
r2 = (r2 + 4) % 5;
}
else
{
//swapping logic
int temp = c1;
```

```
c1 = c2;
c2 = temp;
}
decoded = decoded + table[r1][c1] + table[r2][c2];
}
//returns the decoded message
return decoded;
}
// returns a point containing the row and column of the letter
private Point getPoint(char c)
{
Point pt = new Point(0,0);
for(int i = 0; i < 5; i++)
for(int j = 0; j < 5; j++)
if(c == table[i][j].charAt(0))
pt = new Point(i,j);
return pt;
}
//function prints the key-table in matrix form for playfair cipher
private void keyTable(String[][] printTable)
{
System.out.println("Playfair Cipher Key Matrix:
"); System.out.println();
//loop iterates for rows
for(int i = 0; i < 5; i++)
{
//loop iterates for column
for(int j = 0; j < 5; j++)
{
//prints the key-table in matrix form
System.out.print(printTable[i][j]+" ");
}
System.out.println();
}
System.out.println();
}
//method that prints all the results
private void printResults(String encipher, String
dec) {
System.out.print("Encrypted Message: ");
//prints the encrypted message
System.out.println(encipher);
```

```
        System.out.println();
        System.out.print("Decrypted Message: ");
        //prints the decryted message
        System.out.println(dec);
        }
                }
```

OUTPUT:

```
java -cp /tmp/urB83ID6il PlayfairCipher
Enter the key for playfair cipher: PLAYFAIR
Enter the plaintext to be encipher: FAIRPLAY
Playfair Cipher Key Matrix:

P L A Y F
I R B C D
E G H K M
N O Q S T
U V W X Z

Encrypted Message: PYRBLAYF

Decrypted Message: FAIRPLAY
```

Practical No: 3

Aim: Write programs to implement the following Transposition Cipher Techniques:

a) Rail Fence Cipher

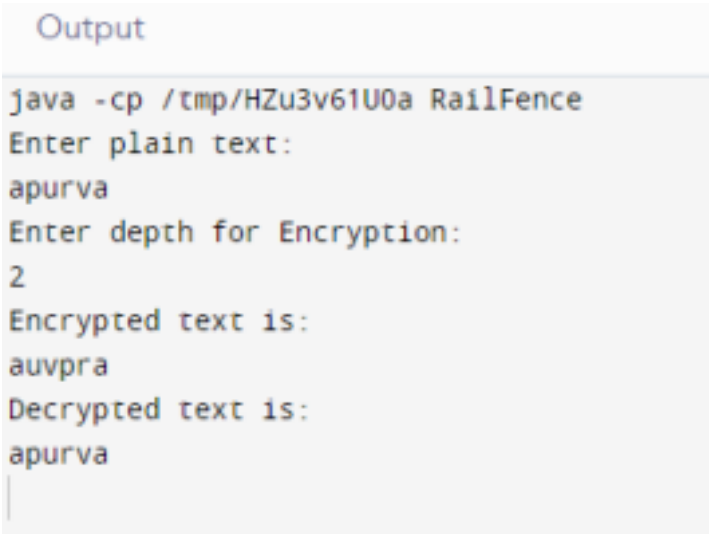
Code:

```
import java.util.*;
class RailFenceBasic{
 int depth;
 String Encryption(String plainText,int depth)throws
 Exception {
 int r=depth,len=plainText.length();
 int c=len/depth;
 char mat[][]=new char[r][c];
 int k=0;
String cipherText="";
 for(int i=0;i< c;i++)
 {
 for(int j=0;j< r;j++)
 {
 if(k!=len)
 mat[j][i]=plainText.charAt(k++);
 else
 mat[j][i]='X';
 }
 }
 for(int i=0;i< r;i++)
 {
 for(int j=0;j< c;j++)
 {
```

```
cipherText+=mat[i][j];
}
}
return cipherText;
}

String Decryption(String cipherText,int depth)throws
Exception {
int r=depth,len=cipherText.length();
int c=len/depth;
char mat[][]=new char[r][c];
int k=0;

String plainText="";

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

```
 }
 }

 return plainText;
 }
}

class RailFence{
 public static void main(String args[])throws
 Exception {
 RailFenceBasic rf=new RailFenceBasic();
 Scanner scn=new Scanner(System.in);  int
depth;

 String plainText,cipherText,decryptedText;
 System.out.println("Enter plain text:");
plainText=scn.nextLine();

 System.out.println("Enter depth for Encryption:");
depth=scn.nextInt();

 cipherText=rf.Encryption(plainText,depth);
 System.out.println("Encrypted text is:\n"+cipherText);
decryptedText=rf.Decryption(cipherText, depth);
System.out.println("Decrypted text is:\n"+decryptedText);
}
}
```

Output:



b) Simple Columnar Technique

Code:

```
import java.io.*;
class SCT
{
public static void main(String args[])throws
Exception {
BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Enter your plain text");
String accept=br.readLine();
System.out.println("Enter the no of rows ");
int r=Integer.parseInt(br.readLine());
System.out.println("Enter the no of cols");
int c=Integer.parseInt(br.readLine());
int count=0;
char cont[][]=new char[r][c];
for(int i=0;i<r;i++)
{
for(int j=0;j<c;j++)
{
if(count>=accept.length())
{
cont[i][j]=' ';
count++;
```

```
}
else
{
cont[i][j]=accept.charAt(count);
count++;
}
}
}
System.out.println("\nEnter the order of cols you want to view them
in");
int choice[]=new int[c];
for(int k=0;k<c;k++)
{
System.out.println("Choice "+k+"-> ");
choice[k]=Integer.parseInt(br.readLine())
; }
System.out.println("\nCipher text in matrix is -
>"); String cipher="";
for(int j=0;j<c;j++)
{
int k=choice[j];
for(int i=0;i<r;i++)
{
cipher+=cont[i][k];
}
}
cipher=cipher.trim();
System.out.println(cipher);
}
}
```

Output:

```
run:
Enter your plain text
apurva
Enter the no of rows
2
Enter the no of cols
4

Enter the order of cols you want to view them in
Choice 0->
0
Choice 1->
1
Choice 2->
2
Choice 3->
3

Cipher text in matrix is ->
avpau r
BUILD SUCCESSFUL (total time: 21 seconds)
```

Practical No: 4

Aim: Write program to encrypt and decrypt strings using
a) DES Algorithm
Code:

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import
java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
```

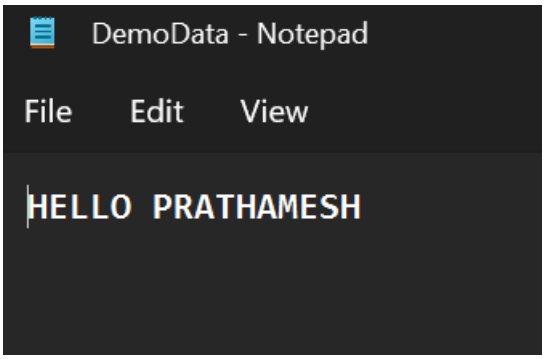
```
import javax.crypto.spec.IvParameterSpec;
public class DesProgram
{
private static Cipher encrypt;
private static Cipher decrypt;
private static final byte[] initialization_vector = { 22, 33, 11, 44, 55,
99, 66, 77 };
public static void main(String[] args)
{
String textFile =
"C:/Users/LEGION/Desktop/TYCS/Practicals/INS/DemoData.txt"
; String encryptedData =
"C:/Users/LEGION/Desktop/TYCS/Practicals/INS/encrypteddata.txt"
; String decryptedData =
"C:/Users/LEGION/Desktop/TYCS/Practicals/INS/decrypteddata.txt"
; try
{
SecretKey scrtkey =
KeyGenerator.getInstance("DES").generateKey()
; AlgorithmParameterSpec aps = new
IvParameterSpec(initialization_vector);
encrypt =
Cipher.getInstance("DES/CBC/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, scrtkey, aps);

decrypt =
Cipher.getInstance("DES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, scrtkey, aps);
encryption(new FileInputStream(textFile), new
FileOutputStream(encryptedData));

decryption(new FileInputStream(encryptedData), new
FileOutputStream(decryptedData));

System.out.println("The encrypted and decrypted files have been
created successfully.");
}
catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException | InvalidAlgorithmParameterException |
IOException e)
{
```

```
e.printStackTrace();
}
}
private static void encryption(InputStream input, OutputStream
output)
throws IOException
{
output = new CipherOutputStream(output, encrypt);

writeBytes(input, output);
}
private static void decryption(InputStream input, OutputStream
output)
throws IOException
{
input = new CipherInputStream(input, decrypt);
writeBytes(input, output);
}
private static void writeBytes(InputStream input, OutputStream
output)
throws IOException
{
byte[] writeBuffer = new byte[512];

int readBytes = 0;
while ((readBytes = input.read(writeBuffer)) >=
0) {
output.write(writeBuffer, 0, readBytes);
}
output.close();

input.close();
}

}
```

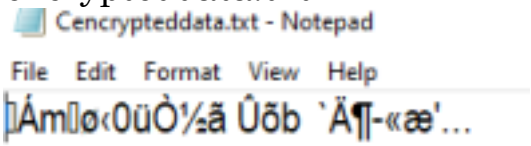
Output:

```
run:
The encrypted and decrypted files have been created successfully.
BUILD SUCCESSFUL (total time: 4 seconds)
```

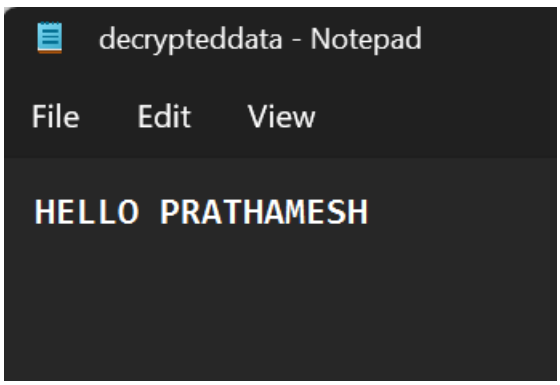
DemoData.txt



encrypteddata.txt



decrypted.txt



b) AES Algorithm

Code:

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.xml.bind.DatatypeConverter;
public class AESEncryption {
public static void main(String[] args) throws Exception
{ String plainText = "Hello World";
SecretKey secKey = getSecretEncryptionKey(); byte[]
cipherText = encryptText(plainText, secKey); String
decryptedText = decryptText(cipherText, secKey);
```

```
System.out.println("Original Text:" +
plainText); System.out.println("AES Key (Hex
Form):"+bytesToHex(secKey.getEncoded()));

System.out.println("Encrypted Text (Hex
Form):"+bytesToHex(cipherText));
System.out.println("Descrypted

Text:"+decryptedText); }

public static SecretKey getSecretEncryptionKey() throws
Exception{
KeyGenerator generator =
KeyGenerator.getInstance("AES"); generator.init(128); // The
AES key size in number of bits SecretKey secKey =
generator.generateKey();
return secKey;
}
public static byte[] encryptText(String plainText,SecretKey
secKey) throws Exception{
Cipher aesCipher = Cipher.getInstance("AES");
aesCipher.init(Cipher.ENCRYPT_MODE, secKey);
byte[] byteCipherText =
aesCipher.doFinal(plainText.getBytes()); return byteCipherText;
}
public static String decryptText(byte[] byteCipherText, SecretKey
secKey) throws Exception {
Cipher aesCipher = Cipher.getInstance("AES");
aesCipher.init(Cipher.DECRYPT_MODE, secKey);
byte[] bytePlainText =
aesCipher.doFinal(byteCipherText); return new
String(bytePlainText);
}
private static String bytesToHex(byte[] hash) {
return DatatypeConverter.printHexBinary(hash);
}
}
```

Output:

```
run:
Original Text:Hello World
AES Key (Hex Form):4C60EF76233E29AE29406F7D759DC43B
Encrypted Text (Hex Form):EF882A069EEA225B6692DF357838B184
Descrypted Text:Hello World
BUILD SUCCESSFUL (total time: 5 seconds)
```

Practical No: 5

Aim: Write a program to implement RSA algorithm to perform encryption / decryption of a given string.

Code:

```
#RSA ALGORITHM


import math


print("RSA ENCRYPTOR/DECRYPTOR")

print("********************")


#Input Prime Numbers

print("PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:")

p = int(input("Enter a prime number for p: "))

q = int(input("Enter a prime number for q: "))

print("********************")


#Check if Input's are Prime

'"THIS FUNCTION AND THE CODE IMMEDIATELY BELOW THE FUNCTION CHECKS WHETHER THE INPUTS ARE PRIME OR NOT."'

def prime_check(a):

 if(a==2):

 return True

 elif((a<2) or ((a%2)==0)):

 return False

 elif(a>2):

 for i in range(2,a):

 if not(a%i):
```

```
 return false
 return True


check_p = prime_check(p)
check_q = prime_check(q)
while(((check_p==False)or(check_q==False))):



p = int(input("Enter a prime number for p: "))
q = int(input("Enter a prime number for q: "))
check_p = prime_check(p)
 check_q = prime_check(q)


#RSA Modulus
'"CALCULATION OF RSA MODULUS
'n'.'" n = p * q
print("RSA Modulus(n) is:",n)


#Eulers Toitent
'"CALCULATION OF EULERS TOITENT
'r'.'" r= (p-1)*(q-1)
print("Eulers Toitent(r) is:",r)
print("********************")


#GCD
'"CALCULATION OF GCD FOR 'e'
CALCULATION.'" def egcd(e,r):
 while(r!=0):
 e,r=r,e%r
 return e


#Euclid's Algorithm
def eugcd(e,r):
 for i in range(1,r):
```

```
 while(e!=0):
 a,b=r//e,r%e
 if(b!=0):
 print("%d = %d*(%d) + %d"%(r,a,e,b))
 r=e
 e=b

#Extended Euclidean Algorithm

 def eea(a,b):
 if(a%b==0):

return(b,0,1)
 else:
 gcd,s,t = eea(b,a%b)
 s = s-((a//b) * t)
 print("%d = %d*(%d) + (%d)*(%d)"%(gcd,a,t,s,b))
 return(gcd,t,s)

#Multiplicative Inverse
 def mult_inv(e,r):
 gcd,s,_=eea(e,r)
 if(gcd!=1):
 return None
 else:
 if(s<0):
 print("s=%d. Since %d is less than 0, s = s(modr), i.e., s=%d."%(s,s,s%r))
elif(s>0):
 print("s=%d."%(s))
 return s%r

#e Value Calculation
 '''FINDS THE HIGHEST POSSIBLE VALUE OF 'e' BETWEEN 1 and 1000 THAT
MAKES (e,r) COPRIME.'''
```

```
for i in range(1,1000):
 if(egcd(i,r)==1):
 e=i
print("The value of e is:",e)
print("********************")
#d, Private and Public Keys
'''CALCULATION OF 'd', PRIVATE KEY, AND PUBLIC
KEY.''' print("EUCLID'S ALGORITHM:")
eugcd(e,r)
print("END OF THE STEPS USED TO ACHIEVE EUCLID'S
ALGORITHM.") print("********************")
print("EUCLID'S EXTENDED ALGORITHM:")
d = mult_inv(e,r)
print("END OF THE STEPS USED TO ACHIEVE THE VALUE OF '
d'.") print("The value of d is:",d)
print("********************")
public = (e,n)
private = (d,n)
print("Private Key is:",private)
print("Public Key is:",public)
print("********************")

#Encryption
'''ENCRYPTION ALGORITHM.'''
def encrypt(pub_key,n_text):
 e,n=pub_key
 x=[]
 m=0
 for i in n_text:
 if(i.isupper()):
 m = ord(i)-65
 c=(m**e)%n
 x.append(c)
 elif(i.islower()):
```

```
        m= ord(i)-97
        c=(m**e)%n
        x.append(c)
        elif(i.isspace()):
        spc=400
        x.append(400)
        return x
         #Decryption
       '''DECRYPTION ALGORITHM'''
       def decrypt(priv_key,c_text):
        d,n=priv_key
        txt=c_text.split(',')
        x=''

        m=0
        for i in txt:
        if(i=='400'):
        x+=' '
        else:
        m=(int(i)**d)%n
        m+=65
        c=chr(m)
        x+=c
        return x

       #Message
        message = input("What would you like encrypted or decrypted?(Separate numbers with
',' for decryption):")
        print("Your message is:",message)

       #Choose Encrypt or Decrypt and Print
        choose = input("Type '1' for encryption and '2' for
        decrytion.") if(choose=='1'):
         enc_msg=encrypt(public,message)
```

```
 print("Your encrypted message is:",enc_msg)
 print("Thank you for using the RSA Encryptor. Goodbye!")
elif(choose=='2'):
 print("Your decrypted message is:",decrypt(private,message))
print("Thank you for using the RSA Encryptor. Goodbye!") else:
 print("You entered the wrong option.")
 print("Thank you for using the RSA Encryptor. Goodbye!")
```

Output:

RSA ENCRYPTOR/DECRYPTOR

****

PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:

Enter a prime number for p: 2

Enter a prime number for q: 3

****

RSA Modulus(n) is: 6

Eulers Toitent(r) is: 2

****

The value of e is: 999

****

EUCLID'S ALGORITHM:

2 = 0*(999) + 2

999 = 499*(2) + 1

END OF THE STEPS USED TO ACHIEVE EUCLID'S ALGORITHM.

****

EUCLID'S EXTENDED ALGORITHM:

1 = 999*(1) + (-499)*(2)

s=1.

END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.

The value of d is: 1

****

Private Key is: (1, 6)

Public Key is: (999, 6)

****

What would you like encrypted or decrypted?(Separate numbers with ',' for decryption):apurva

Your message is: apurva

Type '1' for encryption and '2' for decrytion.1

Your encrypted message is: [0, 3, 2, 5, 3, 0]

Thank you for using the RSA Encryptor. Goodbye!

>

Practical No: 6

Aim: Write a program to implement the Diffie-Hellman Key Agreement algorithm to generate symmetric keys.

Code:

```
import java.util.*;
class DiffieHellman
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter the value of Xa & Xb");
int Xa=sc.nextInt();
int Xb=sc.nextInt();
System.out.println("Enter a Prime no. p");
int p=sc.nextInt();
System.out.println("Enter Primitive Root a, with condition that
a<p");
int a=sc.nextInt();
int Ya=(int)((Math.pow(a,Xa))%p);
int Yb=(int)((Math.pow(a,Xb))%p);
int Ka=(int)((Math.pow(Yb,Xa))%p);
```

```
int Kb=(int)((Math.pow(Ya,Xb))%p);
if(Ka==Kb)
{
System.out.println("Transmission successful");
}
else
{ System.out.println("Transmission failed"); }
}
}
```

Output:

```
Output

java -cp /tmp/Oof3MaiDVL DiffieHellman

Enter the value of Xa & Xb
12
13
Enter a Prime no. p
2
Enter Primitive Root a, with condition that a<p
2
Transmission successful
```

Practical No: 7

Aim: Write a program to implement the MD5 algorithm compute the message digest.

Code:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class JavaMD5Hash {

public static void main(String[] args) {

System.out.println("For null " + generateHash(""));
```

```
System.out.println("For simple text "+ generateHash("sies college."));

System.out.println("For simple numbers " +
generateHash("12345")); }

public static String generateHash(String input) {

String md5 = null;

if(null == input) return null;

try {

//Create MessageDigest object for MD5 or pass SHA-1
MessageDigest digest =
MessageDigest.getInstance("MD5"); //Update input string
in message digest
digest.update(input.getBytes(), 0, input.length());
//Converts message digest value in base 16 (hex)
md5 = new BigInteger(1,
digest.digest()).toString(16); } catch
(NoSuchAlgorithmException e) {
e.printStackTrace();
}
return md5;
}
}
```

Output:

```
Output
java -cp /tmp/0of3MaiDVL JavaMD5Hash
For null d41d8cd98f00b204e9800998ecf8427eFor simple text
    42624aa159071f71c9ec1699541a9f17
For simple numbers 827ccb0eea8a706c4c34a16891f84e7b
```

Practical No: 8

Aim: Write a program to calculate HMAC-SHA1 Signature.
Code:

```
import java.security.InvalidKeyException;
import
java.security.NoSuchAlgorithmException;
import java.security.SignatureException;
import java.util.Formatter;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;


public class HmacSha1Signature {
private static final String HMAC_SHA1_ALGORITHM =
"HmacSHA1";

private static String toHexString(byte[] bytes)
{ Formatter formatter = new Formatter();

for (byte b : bytes) {
formatter.format("%02x", b);
}
return formatter.toString();
}

public static String calculateRFC2104HMAC(String data, String
key) throws SignatureException, NoSuchAlgorithmException,
```

```
InvalidKeyException
{
SecretKeySpec signingKey = new SecretKeySpec(key.getBytes(),
HMAC_SHA1_ALGORITHM);
Mac mac =
Mac.getInstance(HMAC_SHA1_ALGORITHM);
mac.init(signingKey);
return toHexString(mac.doFinal(data.getBytes()));
}

Public



 static void main(String[] args) throws Exception {
String hmac = calculateRFC2104HMAC("data", "key");

System.out.println(hmac);
assert
hmac.equals("104152c5bfdca07bc633eebd46199f0255c9f49d"); }
}
```

Output:

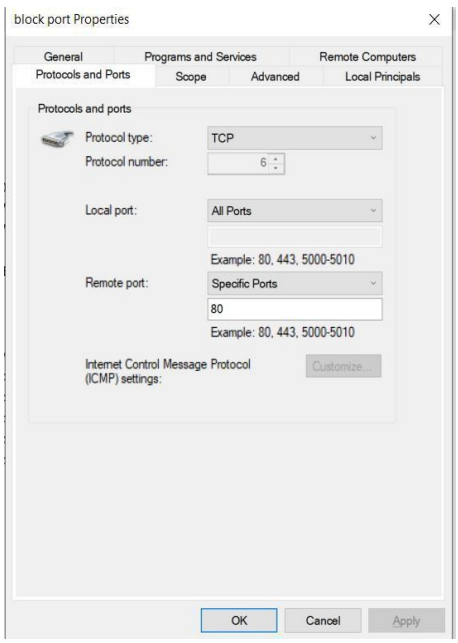
Practical No: 9

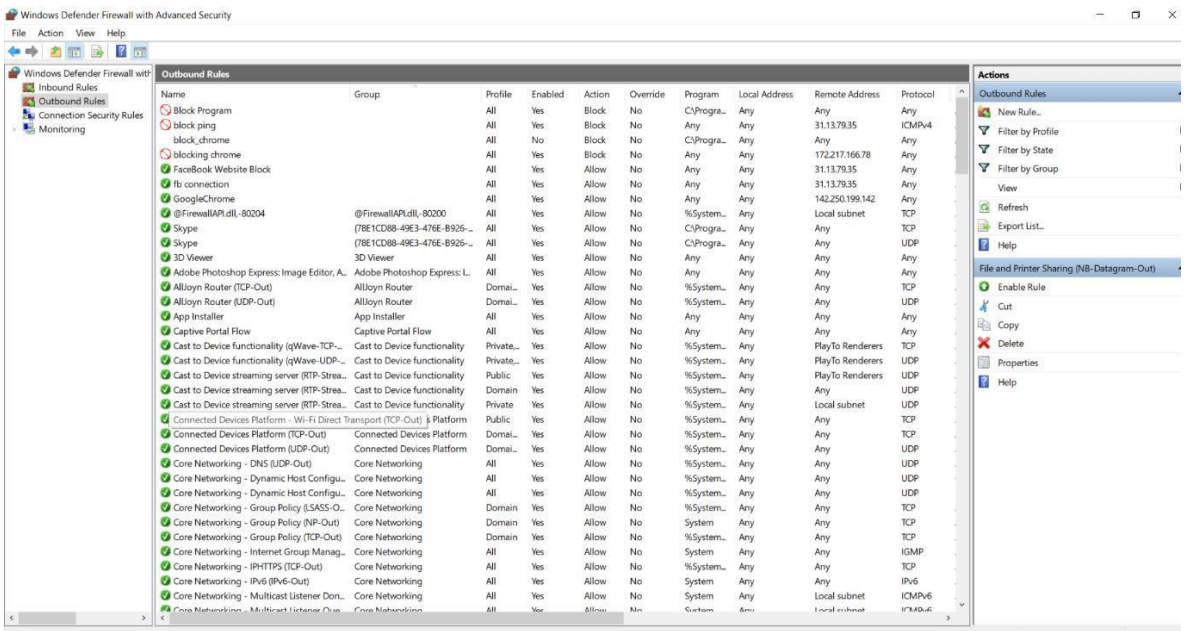
Aim: Configure Windows Firewall to block:

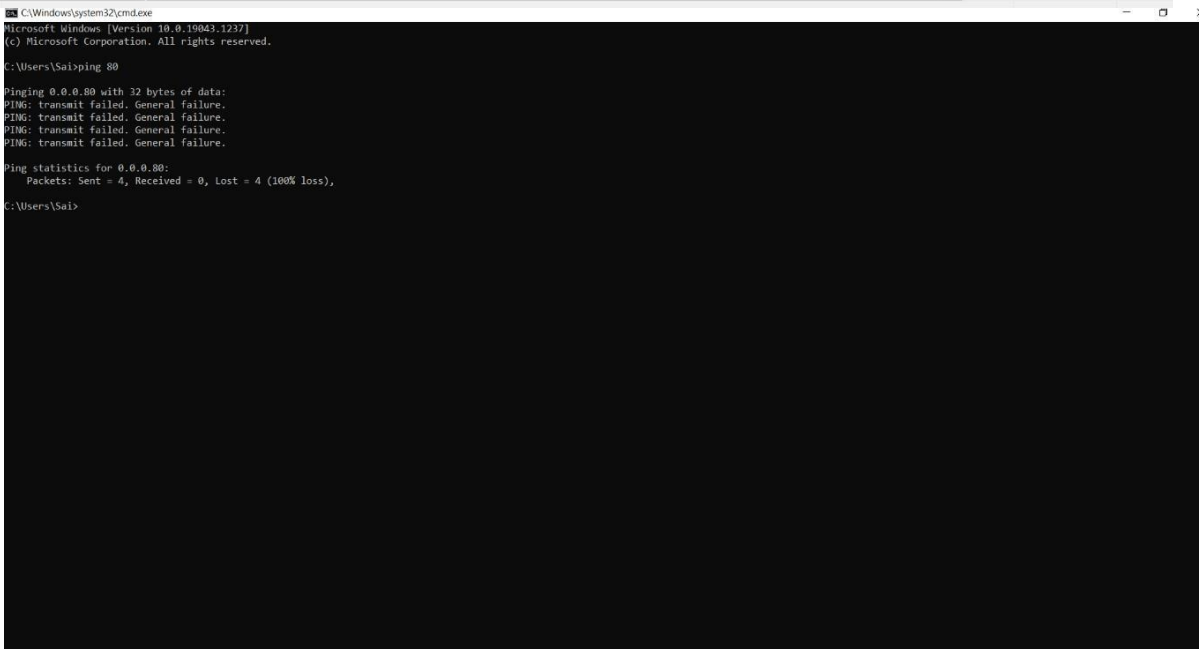
- A port
- An Program
- A website

Output:

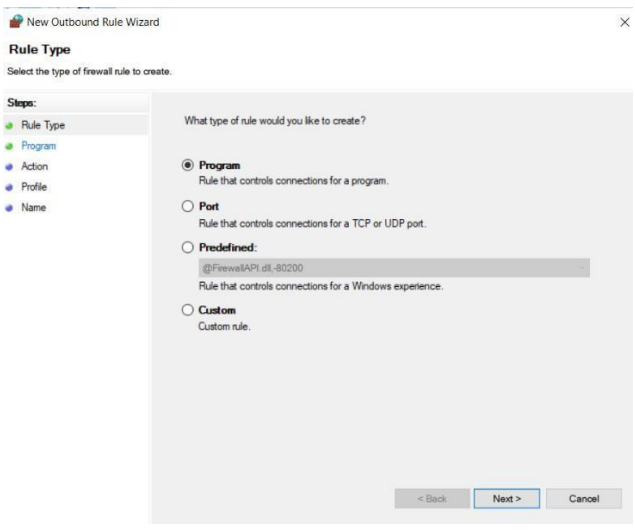
For Port:

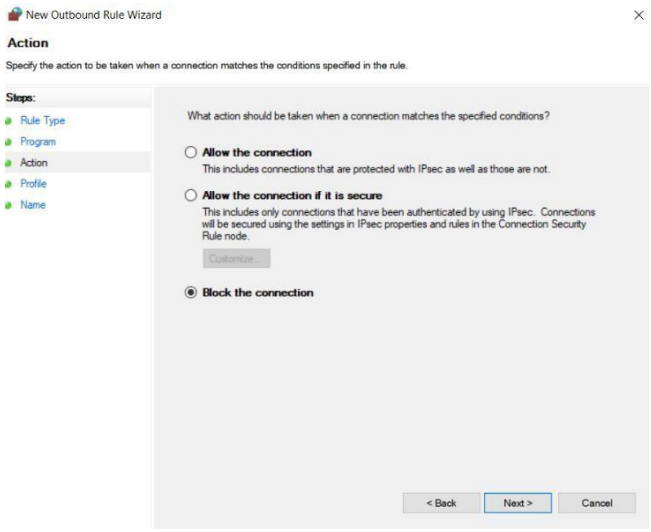




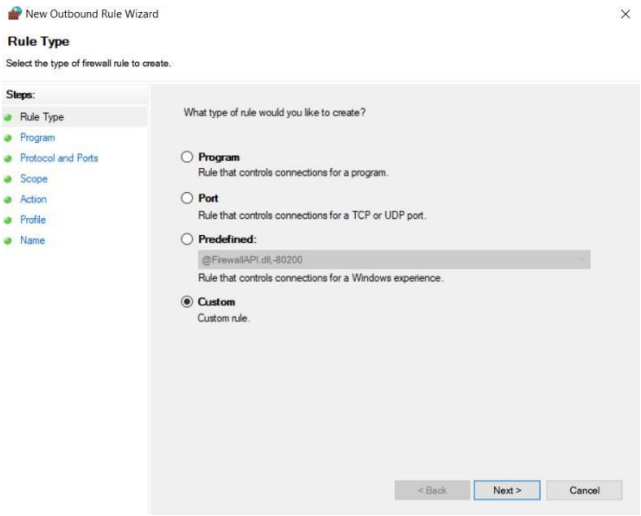


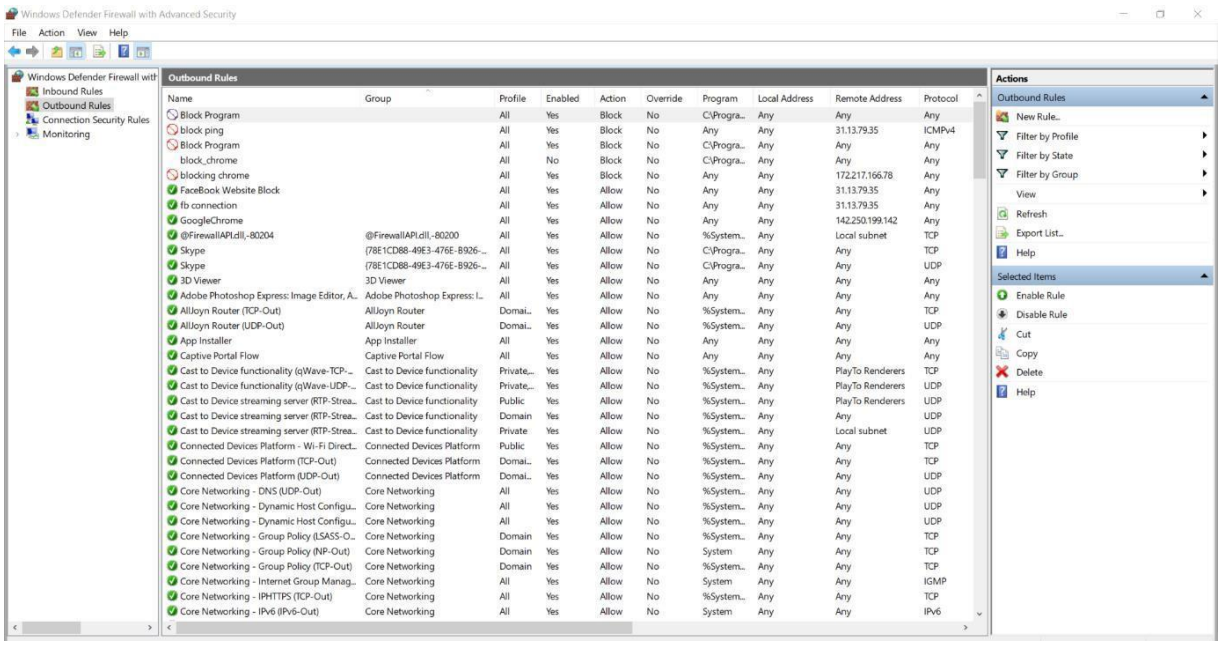
For a Program:

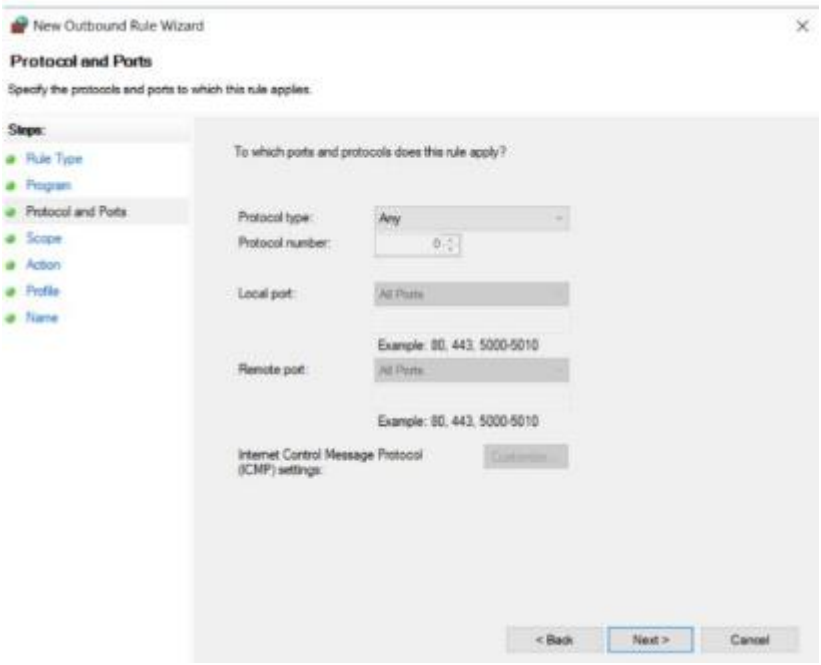


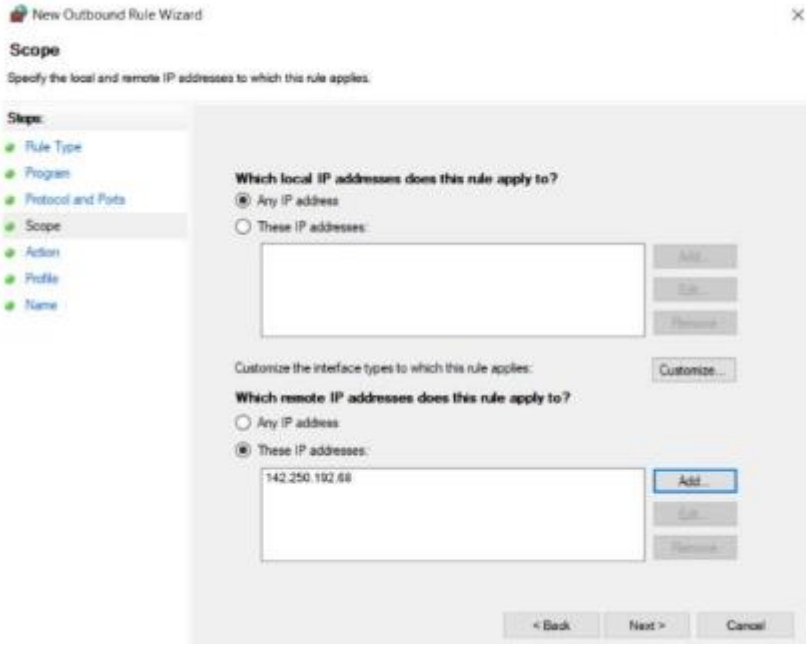


For a Website:






New Outbound Rule Wizard
Protocol and Ports
Specify the protocols and ports to which this rule applies.
Steps:
Rule Type
Program
Protocol and Ports
Scope
Action
Profile
Name
To which ports and protocols does this rule apply?
Protocol type:
Any
Protocol number:
0
Local port:
All Ports
Example: 80, 443, 5000-5010
Remote port:
All Ports
Example: 80, 443, 5000-5010
Internet Control Message Protocol (ICMP) settings:
Customize...
< Back
Next >
Cancel


New Outbound Rule Wizard
Scope
Specify the local and remote IP addresses to which this rule applies.
Steps:
Rule Type
Program
Protocol and Ports
Scope
Action
Profile
Name
Which local IP addresses does this rule apply to?
Any IP address
These IP addresses:
Add...
Edit...
Remove
Customize the interface types to which this rule applies:
Customize...
Which remote IP addresses does this rule apply to?
Any IP address
These IP addresses:
142.250.192.68
Add...
Edit...
Remove
< Back
Next >
Cancel


Windows Defender Firewall with Advanced Security
File Action View Help
Windows Defender Firewall with
Inbound Rules
Outbound Rules
Connection Security Rules
Monitoring
Outbound Rules
Name Group Profile Enabled Action Override Program Local Address Remote Address Protocol
Block Website All Yes Block No Any Any 142.250.192.68 Any
Block Program All Yes Block No C:\Progra... Any Any Any
block ping All Yes Block No Any Any 31.13.79.35 ICMPv4
Block Program All Yes Block No C:\Progra... Any Any Any
block_chrome All No Block No C:\Progra... Any Any Any
blocking chrome All Yes Block No Any Any 172.217.166.78 Any
FaceBook Website Block All Yes Allow No Any Any 31.13.79.35 Any
fb connection All Yes Allow No Any Any 31.13.79.35 Any
GoogleChrome All Yes Allow No Any Any 142.250.199.142 Any
@FirewallAPI.dll,-80200 @FirewallAPI.dll,-80200 All Yes Allow No %System... Any Local subnet TCP
Skype {78E1CD88-49E3-476E-B926-... All Yes Allow No C:\Progra... Any Any TCP
Skype {78E1CD88-49E3-476E-B926-... All Yes Allow No C:\Progra... Any Any UDP
3D Viewer 3D Viewer All Yes Allow No Any Any Any Any
Adobe Photoshop Express: Image Editor, A... Adobe Photoshop Express: I... All Yes Allow No Any Any Any Any
Allkoyn Router (TCP-Out) Allkoyn Router Domai... Yes Allow No %System... Any Any TCP
Allkoyn Router (UDP-Out) Allkoyn Router Domai... Yes Allow No %System... Any Any UDP
App Installer App Installer All Yes Allow No Any Any Any Any
Captive Portal Flow Captive Portal Flow All Yes Allow No Any Any Any Any
Cast to Device functionality (qWave-TCP-... Cast to Device functionality Private,... Yes Allow No %System... Any PlayTo Renderers TCP
Cast to Device functionality (qWave-UDP-... Cast to Device functionality Private,... Yes Allow No %System... Any PlayTo Renderers UDP
Cast to Device streaming server (RTP-Strea... Cast to Device functionality Public Yes Allow No %System... Any PlayTo Renderers UDP
Cast to Device streaming server (RTP-Strea... Cast to Device functionality Domain Yes Allow No %System... Any Any UDP
Cast to Device streaming server (RTP-Strea... Cast to Device functionality Private Yes Allow No %System... Any Local subnet UDP
Connected Devices Platform - Wi-Fi Direct... Connected Devices Platform Public Yes Allow No %System... Any Any TCP
Connected Devices Platform (TCP-Out) Connected Devices Platform Domai... Yes Allow No %System... Any Any TCP
Connected Devices Platform (UDP-Out) Connected Devices Platform Domai... Yes Allow No %System... Any Any UDP
Core Networking - DNS (UDP-Out) Core Networking All Yes Allow No %System... Any Any UDP
Core Networking - Dynamic Host Configu... Core Networking All Yes Allow No %System... Any Any UDP
Core Networking - Dynamic Host Configu... Core Networking All Yes Allow No %System... Any Any UDP
Core Networking - Group Policy (LSASS-O... Core Networking Domain Yes Allow No %System... Any Any TCP
Core Networking - Group Policy (NP-Out) Core Networking Domain Yes Allow No System Any Any TCP
Core Networking - Group Policy (TCP-Out) Core Networking Domain Yes Allow No %System... Any Any TCP
Core Networking - Internet Group Manag... Core Networking All Yes Allow No System Any Any IGMP
Core Networking - IPHTTPS (TCP-Out) Core Networking All Yes Allow No %System... Any Any TCP
Actions
Outbound Rules
New Rule...
Filter by Profile
Filter by State
Filter by Group
View
Refresh
Export List...
Help
Core Networking - Router Solicitation (ICMPv6...
Disable Rule
Cut
Copy
Delete
Properties
Help

SIES College of Arts, Science & Commerce
RISE WITH EDUCATION
NAAC REACCREDITED - 'A' GRADE
ISO 9001 : 2008

S.I.E.S College of Arts, Science and Commerce
Sion(W), Mumbai – 400 022.

CERTIFICATE

This is to certify that **Mr. Prathamesh Babu Devkate** Rollno**TYCS2223017** Has successfully completed the necessary course of experiments in the subject of **Information and Network Security** during the academic year **2022 – 2023** complying with the requirements of **University of Mumbai**,

for the course of **T. Y. BSc. Computer Science [Semester-5]**

Prof. In-Charge

Mr. Abuzar Ansari

Examination Date:

Examiner's Signature & Date:

Head of the Department

Prof. Manoj Singh

College Seal

And

Date