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 Ms. **Dipali Gupta** Roll No. **TCS2223026** 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 (INS)

Examination Date:
Examiner's Signature & Date:

Head of the Department
Prof. Manoj Singh

College Seal

&

Date

Sr.no	Practical
1.	write a program 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 a program to implement the followingTransposition Cipher Techniques 1. Rail Fence Cipher 2. Simple Columnar Technique
4.	Write a program to encrypt and decrypt strings using 1. DES Algorithm2. AES Algorithm
5.	Write a program to implement RSA algorithmtoperform encryption/decryption of a give string
6.	Write a program to implement the Diffie-HellmanKey Agreement algorithm to generate symmetrickeys
7.	Write a program to implement the MD5 algorithmcompute the message digest
8.	Write a program to calculate HMAC-SHA1Signature
9.	Configure Windows Firewall to block 1.A port 2.A program 3.A website

INS PRACTICAL 1

Dipali Gupta Roll no.: TCS2223026

Aim:

Write programs to implement the following Substitution Cipher Techniques:

1)Caesar Cipher

Code:

```
dictionary = ['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','0','1',
              '2','3','4','5',
              '6','7','8','9',
              '!','@','#','$',
              '%','^','*']
plain_text = input("Enter Plain Text: ").replace(" ",
"").lower() key = int(input("Key: ")) cipher_text = ""
print("Plain Text: ", plain_text)
# Cipher for letter in
plain_text:
   cipher_text += dictionary[(dictionary.index(letter)+key)%len(dictionary)]
print("Cipher Text: ", cipher_text)
```

Output:

```
print("Cipher Text: ", cipher_text)

Enter Plain Text: SIES400*2
Key: 2
Plain Text:  sies400*2
Cipher Text:  ukgu622b4
```

2)Monoalphabetic Cipher

Code:

```
dictionary = ['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']

plain_text = input("Enter Plain Text: ").replace(" ", "").lower()

cipher_text = "" # Cipher for letter in plain_text:
cipher_text += dictionary[-
dictionary.index(letter)-1] print(cipher_text)
```

```
print(cipher_text)

Enter Plain Text: sies
hrvh

]:
```

PRACTICAL 2

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

1)Vernam Cipher

Code:

```
# Vernam Cipher
d = ['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']
plain_text = input("Enter Plain Text: ").replace(" ", "").lower()
# Taking input (until right) key = "" while True:  key =
input("Enter key (text) (same length as plain text): ").replace("
","").lower()  if len(key) == len(plain_text):
break print("Plain Text: ", plain_text)
cipher_text = "" # Ex-OR operation function
def performXOR(binary_letter1,
binary_letter2):
 binary_output = ""  for i in
range(len(binary_letter1)):
 if binary_letter1[i] == binary_letter2[i]:
 binary_output += "0"
 else:
 binary_output += "1"
 return binary_output # Cipher for i in
range(len(plain_text)):  # Converting plain
text to binary  binary_letter1 =
bin(d.index(plain_text[i]))  binary_letter2 =
bin(d.index(key[i]))  # Converting binary
numbers to similar length  temp =
binary_letter1  binary_letter1 = "0"*(8-
len(temp[2:]))+temp[2:]  temp =
binary_letter2  binary_letter2 = "0"*(8-
len(temp[2:]))+temp[2:]
 # Converting binary to letter  binary_output =
performXOR(binary_letter1, binary_letter2)  cipher_text
+= d[int(binary_output, 2)%len(d)] print("Cipher Text:
",cipher_text)
```

```
Enter Plain Text: sies
Enter key (text) (same length as plain text): sies
Plain Text:  sies
Cipher Text:  aaaa
```

2)Playfair Cipher

Code:

```
#Playfair Cipher
#key matrix
key = [['l','g','d','b','a'],
 ['q','m','h','e','c'],
 ['u','r','n','i','f'],
 ['x','v','s','o','k'],
 ['z','y','w','t','p']]

# Taking input and making pairs plain_text = input("Enter
Plain Text: ").replace(" ", "").replace("j",
"i").lower() print("Plain
Text: ", plain_text)
bogus_letter = 'x'
seperated_letters = []
i = 1 while
True:
 try:
 if plain_text[i-1] != plain_text[i]:

seperated_letters.append(plain_text[i-1]+plain_text[i
])  i+=2  else:
 seperated_letters.append(plain_text[i]+bogus_letter)
 i += 1
 except:
```

```
 if i == len(plain_text):
 seperated_letters.append(plain_text[-1]+bogus_letter)
break

print("Letter pairs: ",
seperated_letters) cipher_text = "" #
search element in key: def
indexOfLetter(key, letter):
 for i in range(len(key)):
for j in range(len(key)):
if key[i][j] == letter:
 return (i,j)
# Cipher
for pair in seperated_letters:
 letter1_index = indexOfLetter(key, pair[0])
letter2_index = indexOfLetter(key, pair[1])
 # rule 1: same column  if
letter1_index[1] == letter2_index[1]:
 cipher_text += key[(letter1_index[0]+1)%len(key)][letter1_index[1]]
+key[(letter2_index[0]+
1)%len(key)][letter2_index[1]]  # rule
2: same row  elif letter1_index[0] ==
letter2_index[0]:
 cipher_text += key[letter1_index[0]][(letter1_index[1]+1)%len(key)]
+key[letter2_index[0]][(l etter2_index[1]+1)%len(key)]

 # rule 3: different column and row
 else:
 cipher_text += key[letter1_index[0]][letter2_index[1]]+key[letter2_index[0]]
[letter1_index[
1]]
print("Cipher Text: ",cipher_text)
```

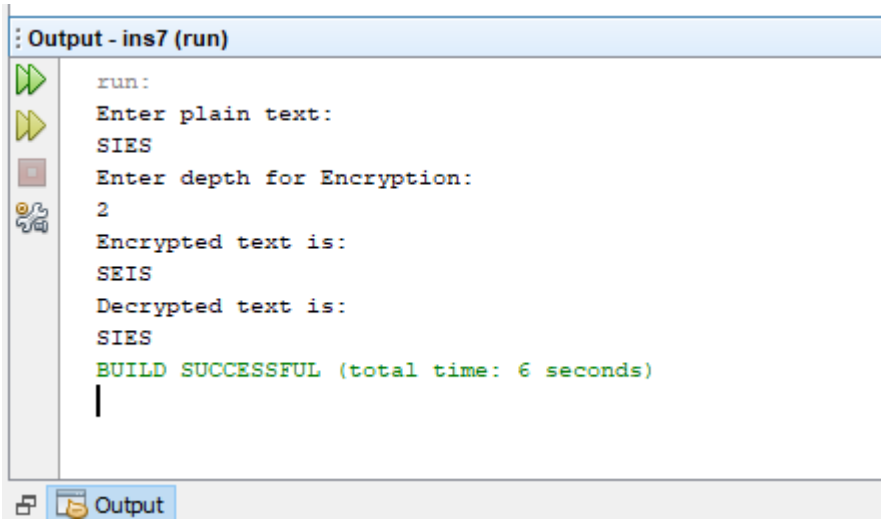
```
Enter Plain Text: sies
Plain Text:  sies
Letter pairs:  ['si', 'es']
Cipher Text:  onho
```

<u>Practical 3</u>
Aim: Write a program to implement the followingTransposition Cipher Techniques
1. Rail Fence Cipher
2. Simple Columnar Technique

Code:
Rail Fence Cipher

```
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];
```

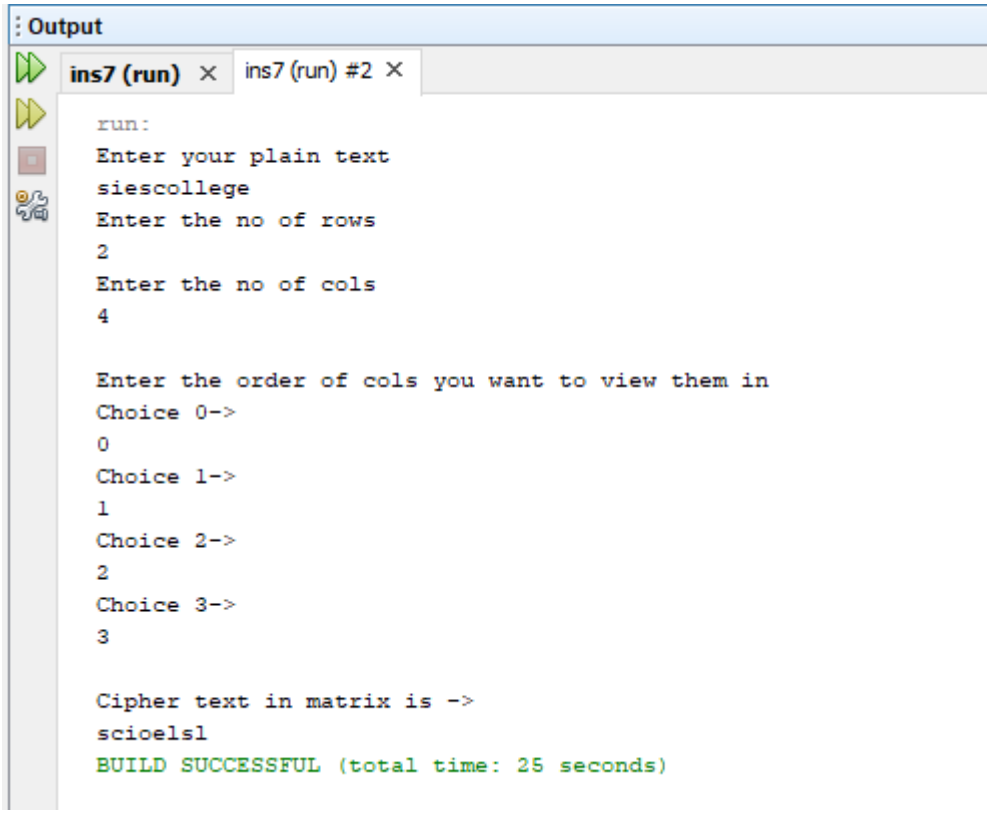
```
  int k=0;

  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);
 }
}
```



2. Simple columnar technique

```
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);
}
}
```



<u>Practical 4</u>

AIM: Write a program to encrypt and decrypt strings using

Output:

A) DES Algorithm Code:

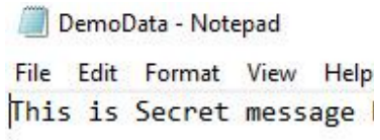
```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
```

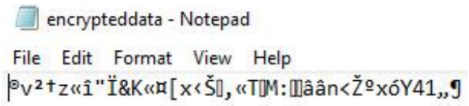
```
import java.io.OutputStream;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import 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/AGENT47/Desktop/DemoData.txt";
String encryptedData = "C:/Users/AGENT47/Desktop/encrypteddata.txt";
String decryptedData = "C:/Users/AGENT47/Desktop/decrypteddata.txt";
try
{
SecretKey scrtkey = KeyGenerator.getInstance("DES").generateKey();
AlgorithmParameterSpec aps = new IvParameterSpec(initialization_vector);
encrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, scrtkey, aps);
decrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, scrtkey, aps);
encryption(new FileInputStream(textFile), new
FileOutputStream(encryptedData));
decryption(new FileInputStream(encryptedData), new
FileOutputStream(decryptedData));
System.out.println("The encrypted and decrypted files have been created
successfully.");
}
catch (NoSuchAlgorithmException | NoSuchPaddingException |
InvalidKeyException | InvalidAlgorithmParameterException|IOException e)
{
e.printStackTrace();
}
}
private static void encryption(InputStream input, OutputStream output)
throws IOException
{
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();
}
}
```

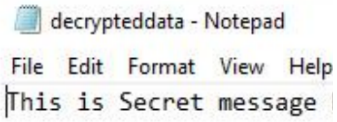
Output:

```
de\User\workspaceStorage\85e4b655d753e5a71b68a1424c9d8ab1\redhat.ja
The encrypted and decrypted files have been created successfully.
```





B) AES Algorithm




```
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.KeySpec;
import java.util.Base64;
import javax.crypto.BadPaddingException;
```

```
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
public class AESExample
{
/* Private variable declaration */
private static final String SECRET_KEY = "123456789";
private static final String SALTVALUE = "abcdefg";
/* Encryption Method */
public static String encrypt(String strToEncrypt)
{
try
{
/* Declare a byte array. */
byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
IvParameterSpec ivspec = new IvParameterSpec(iv);
/* Create factory for secret keys. */
SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
/* PBEKeySpec class implements KeySpec interface. */
KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(),
SALTVALUE.getBytes(), 65536, 256);
SecretKey tmp = factory.generateSecret(spec);
SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivspec);
/* Retruns encrypted value. */
return Base64.getEncoder()
.encodeToString(cipher.doFinal(strToEncrypt.getBytes(StandardCharsets.UTF_
8)));
}
catch (InvalidAlgorithmParameterException | InvalidKeyException |
NoSuchAlgorithmException | InvalidKeySpecException|BadPaddingException |
IllegalBlockSizeException | NoSuchPaddingException e)
{
System.out.println("Error occured during encryption: " + e.toString());
}
return null;
}
/* Decryption Method */
public static String decrypt(String strToDecrypt)
{
try
{
/* Declare a byte array. */
byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
IvParameterSpec ivspec = new IvParameterSpec(iv);
/* Create factory for secret keys. */
SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");
/* PBEKeySpec class implements KeySpec interface. */
KeySpec spec = new PBEKeySpec(SECRET_KEY.toCharArray(),
SALTVALUE.getBytes(), 65536, 256);
SecretKey tmp = factory.generateSecret(spec);
```

```
SecretKeySpec secretKey = new SecretKeySpec(tmp.getEncoded(), "AES");
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5PADDING");
cipher.init(Cipher.DECRYPT_MODE, secretKey, ivspec);
/* Retruns decrypted value. */
return new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)));
}
catch (InvalidAlgorithmParameterException | InvalidKeyException |
NoSuchAlgorithmException | InvalidKeySpecException|BadPaddingException |
IllegalBlockSizeException | NoSuchPaddingException e)
{
System.out.println("Error occured during decryption: " + e.toString());
}
return null;
}
/* Driver Code */
public static void main(String[] args)
{
/* Message to be encrypted. */
String originalval = "AES Encryption";
/* Call the encrypt() method and store result of encryption. */
String encryptedval = encrypt(originalval);
/* Call the decrypt() method and store result of decryption. */
String decryptedval = decrypt(encryptedval);
/* Display the original message, encrypted message and decrypted message on
the console. */
System.out.println("Original value: " + originalval);
System.out.println("Encrypted value: " + encryptedval);
System.out.println("Decrypted value: " + decryptedval);
}
}
```

Output:

```
Original value: AES Encryption
Encrypted value: V5E9I52IxhMaW4+hJhl56g==
Decrypted value: AES Encryption
```

Practical 5

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

Code:

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package ins_rsa;
import java.math.BigInteger;

import java.util.Random;

import java.io.*;
```

```
/**
 *
 * @author sies
 */
public class Ins_rsa {
private BigInteger p;

private BigInteger q;

private BigInteger N;

private BigInteger phi;

private BigInteger e;

private BigInteger d;

private int bitlength = 1024;

private int blocksize = 256;

    /**
     * @param args the command line arguments
     */
private Random r;

public Ins_rsa() {
        // TODO code application logic here
        r = new Random();
System.out.println("r");
System.out.println(r);

p = BigInteger.probablePrime(bitlength, r);
System.out.println("p");
System.out.println(p);

q = BigInteger.probablePrime(bitlength, r);
System.out.println("q");
System.out.println(q);

N = p.multiply(q);

System.out.println("N");
System.out.println(N);

phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
System.out.println("phi");
System.out.println(phi);

e = BigInteger.probablePrime(bitlength/2, r);
System.out.println("e");
System.out.println(e);
```

```
while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0 ) {

e.add(BigInteger.ONE);

}

d = e.modInverse(phi);
System.out.println("d");
System.out.println(d);

}

public Ins_rsa(BigInteger e, BigInteger d, BigInteger N) {

this.e = e;

this.d = d;

this.N = N;

}

public static void main (String[] args) throws IOException {

Ins_rsa rsa = new Ins_rsa();

DataInputStream in=new DataInputStream(System.in);

String teststring ;

System.out.println("Enter the plain text:");

teststring=in.readLine();

System.out.println("Encrypting String: " + teststring);

System.out.println("String in Bytes: " + bytesToString(teststring.getBytes()));

// encrypt

byte[] encrypted = rsa.encrypt(teststring.getBytes());

System.out.println("Encrypted String in Bytes: " + bytesToString(encrypted));

// decrypt

byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypted String in Bytes: " +  bytesToString(decrypted));

System.out.println("Decrypted String: " + new String(decrypted));
```

```
}

private static String bytesToString(byte[] encrypted) {

String test = "";

for (byte b : encrypted) {

test += Byte.toString(b);

}

return test;

}

//Encrypt message

public byte[] encrypt(byte[] message) {

return (new BigInteger(message)).modPow(e, N).toByteArray();

}

// Decrypt message

public byte[] decrypt(byte[] message) {

return (new BigInteger(message)).modPow(d, N).toByteArray();

}

}
```

Output:

```
Output - ins_rsa (run)
run:
r
java.util.Random@6d06d69c
p
128418511463728151058355961098417665022888637119911575571641409535033030411147686302545697795802069547095
q
969312483200421768347808846433907086019712325660631518696971971159397393594859750294525986667486655813604
N
124477666235808163695527852789604168328618296833093701987440853858056316344141003817263441750231678428732
phi
124477666235808163695527852789604168328618296833093701987440853858056316344141003817263441750231678428732
e
867360538858463156145803853817653426411636338718502466867361614056018802420417135979501702743047140243067
d
721066210424196053105148166290797775335744511731308794854346336187166672629631634930719634905608156328535
Enter the plain text:
barcelona
Encrypting String: barcelona
String in Bytes: 9897114991011081111097
Encrypted String in Bytes: 6589-1186886-5373-872577-29-5012387-64105-79-1116797-83480697512483-114-1080-5
Decrypted String in Bytes: 9897114991011081111097
Decrypted String: barcelona
BUILD SUCCESSFUL (total time: 14 seconds)
```

<u>Practical 6</u>

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

Code:

```
import java.util.*;
// create class DiffieHellmanAlgorithmExample to calculate the key for two
persons
class DiffieHellmanAlgorithmExample {
// main() method start
public static void main(String[] args)
{
long P, G, x, a, y, b, ka, kb;
// create Scanner class object to take input from user
Scanner sc = new Scanner(System.in);
System.out.println("Both the users should be agreed upon the public keys G
and P");
// take inputs for public keys from the user
System.out.println("Enter value for public key G:");
G = sc.nextLong();
System.out.println("Enter value for public key P:");
P = sc.nextLong();
// get input from user for private keys a and b selected by User1 and User2
System.out.println("Enter value for private key a selected by user1:");
a = sc.nextLong();
System.out.println("Enter value for private key b selected by user2:");
b = sc.nextLong();
// call calculatePower() method to generate x and y keys
x = calculatePower(G, a, P);
y = calculatePower(G, b, P);
// call calculatePower() method to generate ka and kb secret keys after the
exchange of x and y keys
// calculate secret key for User1
ka = calculatePower(y, a, P);
// calculate secret key for User2
kb = calculatePower(x, b, P);
// print secret keys of user1 and user2
System.out.println("Secret key for User1 is:" + ka);
System.out.println("Secret key for User2 is:" + kb);
}
// create calculatePower() method to find the value of x ^ y mod P
private static long calculatePower(long x, long y, long P)
{
long result = 0;
if (y == 1){
return x;
}
else{
result = ((long)Math.pow(x, y)) % P;
```

```
return result;
}
}
}
```

Output:

```
Both the users should be agreed upon the public keys G and P
Enter value for public key G:
8
Enter value for public key P:
33
Enter value for private key a selected by user1:
2
Enter value for private key b selected by user2:
3
Secret key for User1 is:25
Secret key for User2 is:25
```

<u>Practical 7</u>

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

Code:

```
package ins7;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class Ins7 {
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 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 - ins7 (run)
run:
For null d41d8cd98f00b204e9800998ecf8427e
For simple text 42624aa159071f71c9ec1699541a9f17
For simple text lec9bcd5fe29c481af7636071e6b09ef
For simple numbers 827ccb0eea8a706c4c34a16891f84e7b
BUILD SUCCESSFUL (total time: 0 seconds)
```

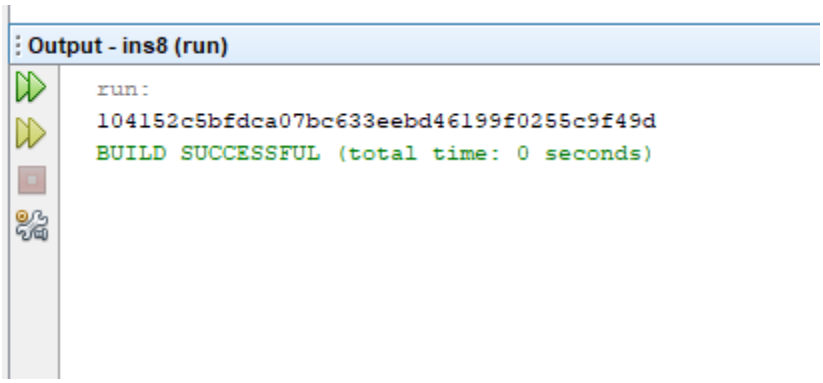
<u>Practical 8</u>

Aim: Write a program to calculate HMAC-SHA1Signature

Code:

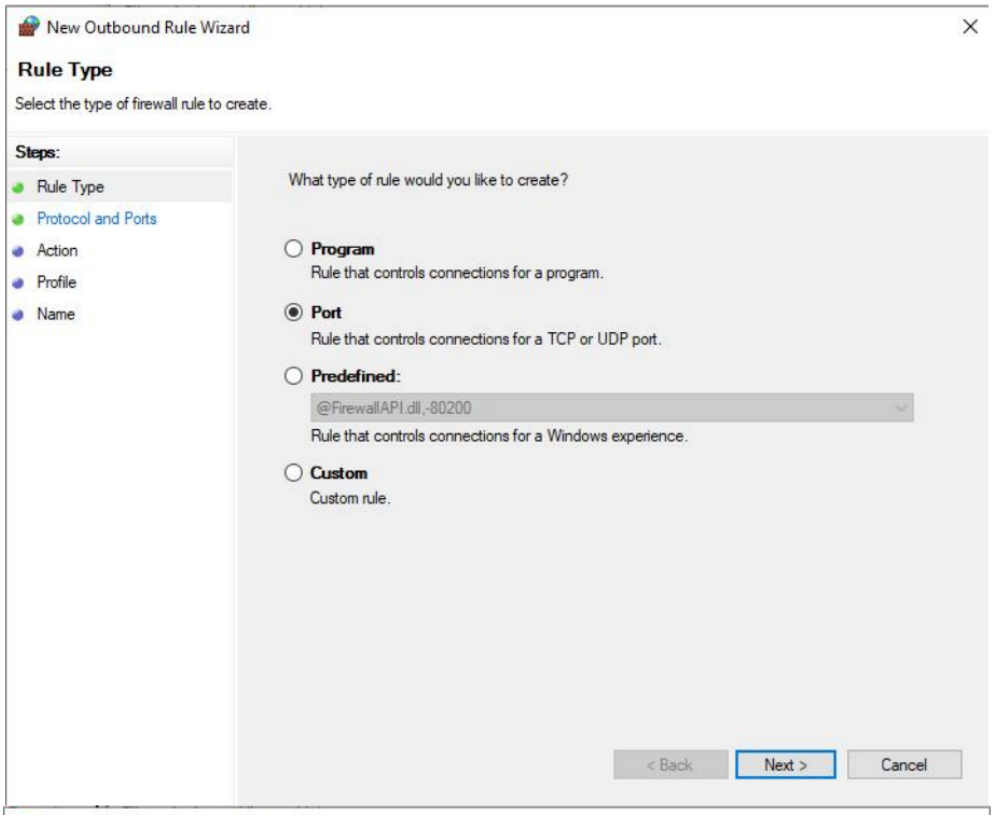
```
package ins8;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.SignatureException;
import java.util.Formatter;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class Ins8 {
   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");
```

}
}



<u>Practical 9</u>
Aim: Configure Windows Firewall to block
1. A port
2. A program
3. A website

Output:
A Port



New Outbound Rule Wizard

New Outbound Rule Wizard

Action

Specify the action to be taken when a connection matches the conditions specified in the rule.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

What action should be taken when a connection matches the specified conditions?

○ **Allow the connection**
This includes connections that are protected with IPsec as well as those are not.

○ **Allow the connection if it is secure**
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.

Customize...

◉ **Block the connection**

< Back | Next > | Cancel

New Outbound Rule Wizard

New Outbound Rule Wizard

Profile

Specify the profiles for which this rule applies.

Steps:

- Rule Type
- Protocol and Ports
- Action
- Profile
- Name

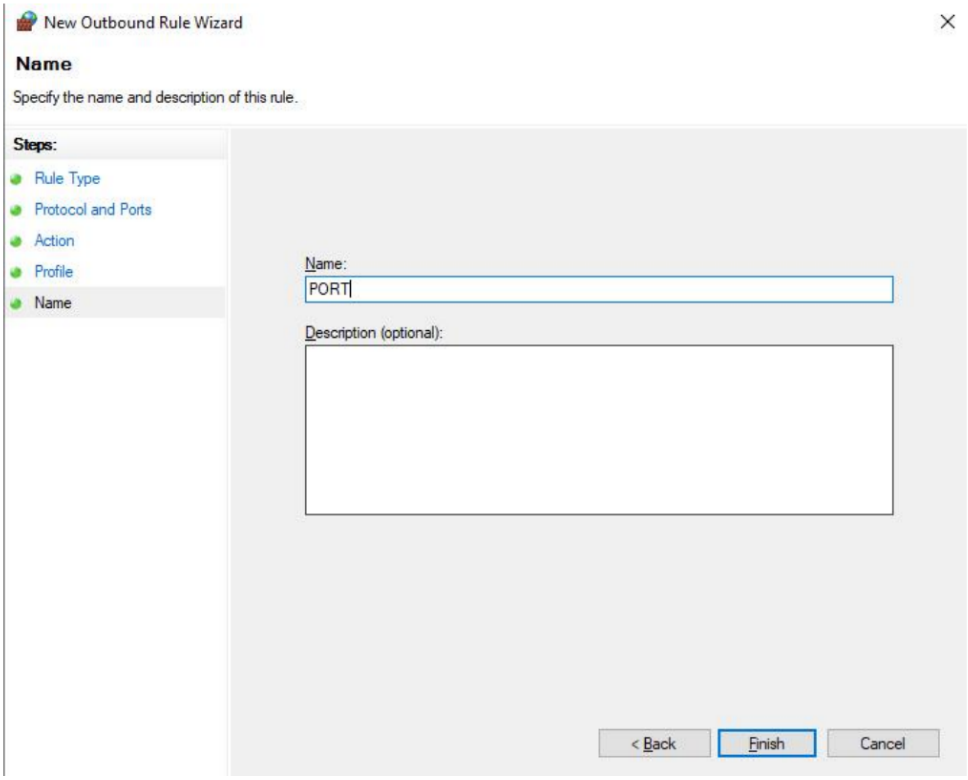
When does this rule apply?

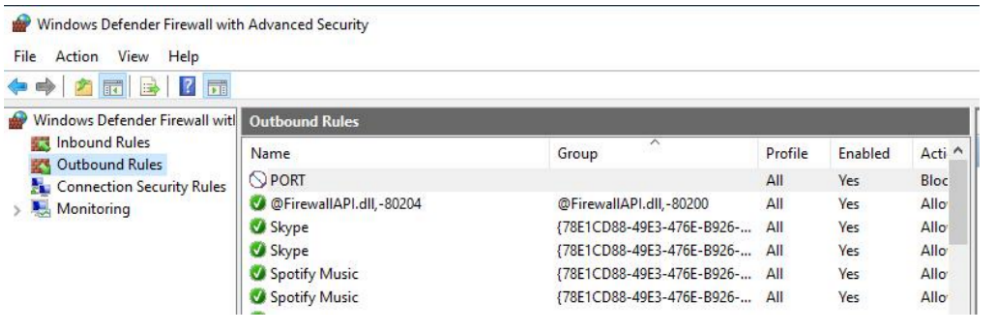
☑ **Domain**
Applies when a computer is connected to its corporate domain.

☑ **Private**
Applies when a computer is connected to a private network location, such as a home or work place.

☑ **Public**
Applies when a computer is connected to a public network location.

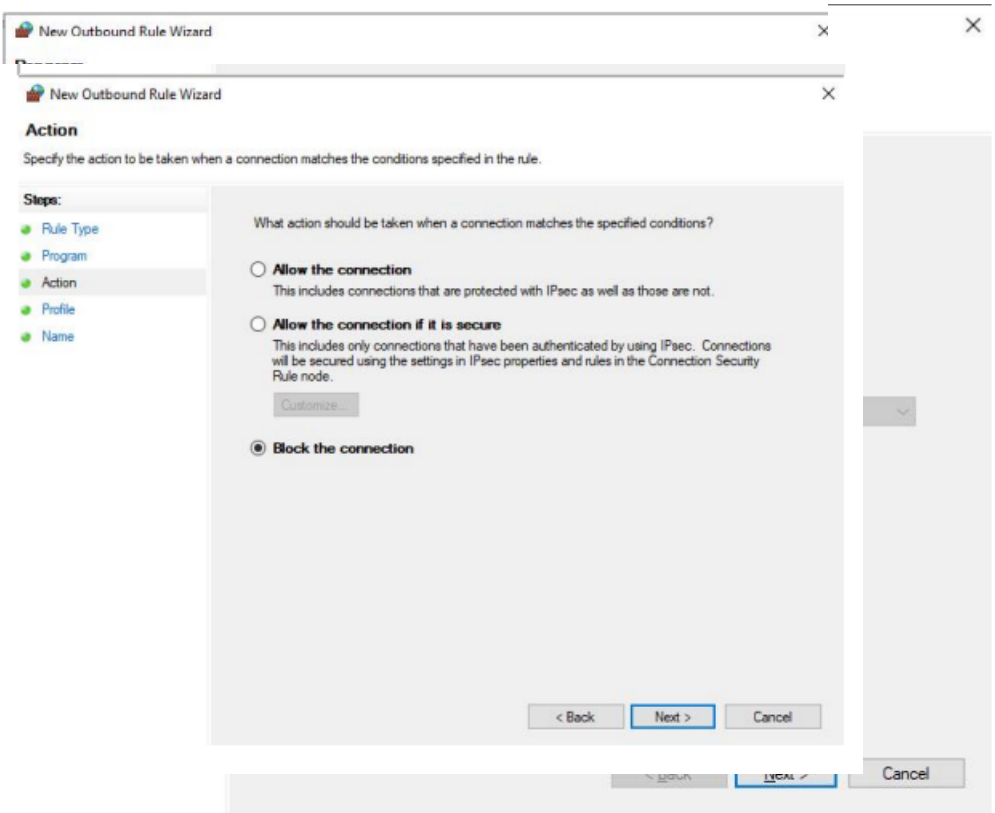
< Back | Next > | Cancel

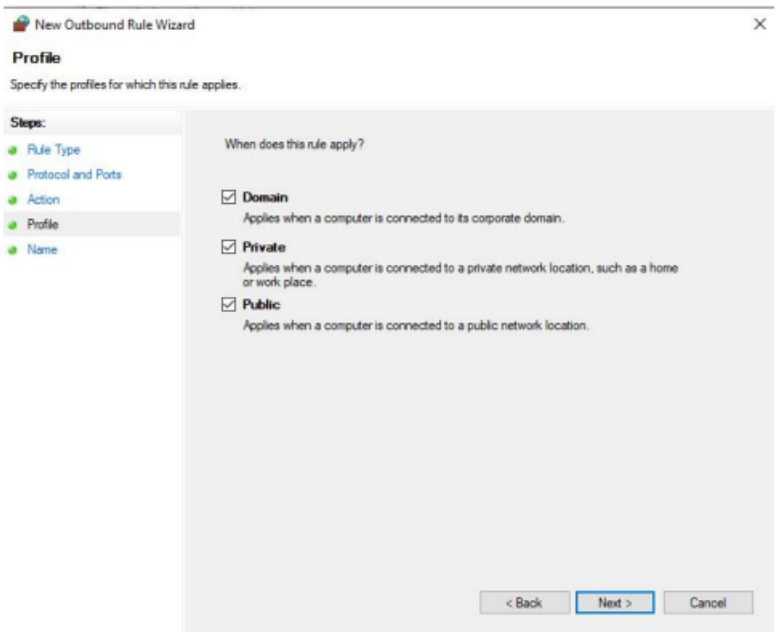


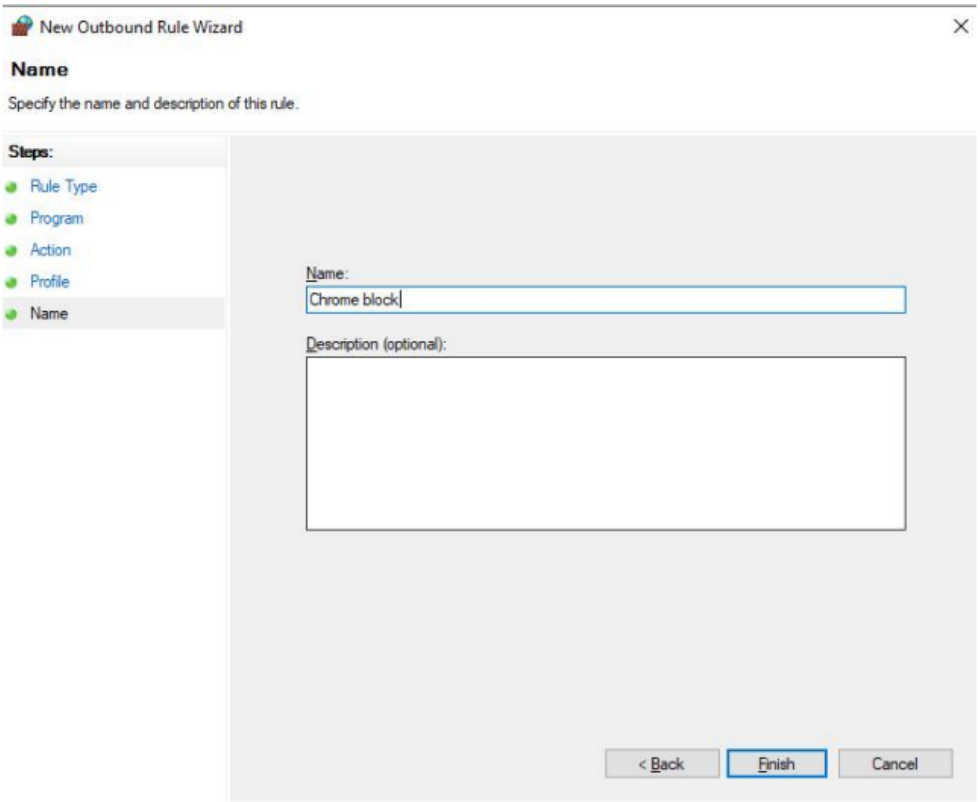


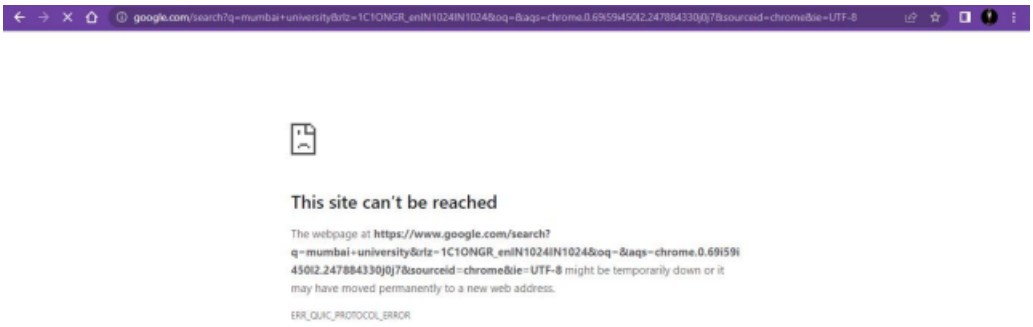
A Program:

```
Pinging 0.0.0.80 with 32 bytes of data:
PING: transmit failed. General failure.
PING: transmit failed. General failure.
PING: transmit failed. General failure.
PING: transmit failed. General failure.

Ping statistics for 0.0.0.80:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

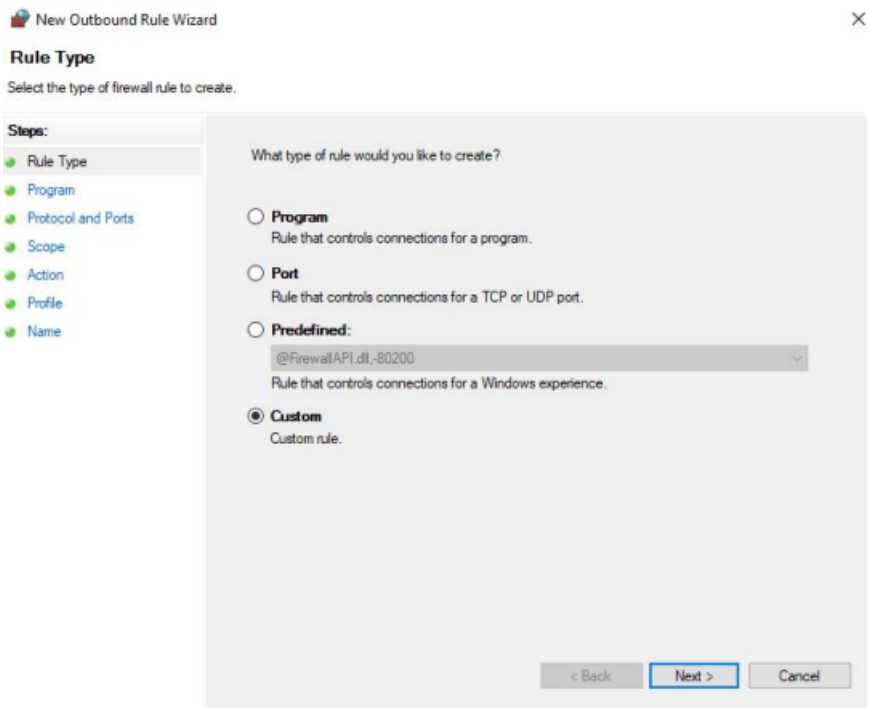

New Outbound Rule Wizard
New Outbound Rule Wizard
Action
Specify the action to be taken when a connection matches the conditions specified in the rule.
Steps:
Rule Type
Program
Action
Profile
Name
What action should be taken when a connection matches the specified conditions?
Allow the connection
This includes connections that are protected with IPsec as well as those are not.
Allow the connection if it is secure
This includes only connections that have been authenticated by using IPsec. Connections will be secured using the settings in IPsec properties and rules in the Connection Security Rule node.
Customize...
Block the connection
< Back
Next >
Cancel
Cancel

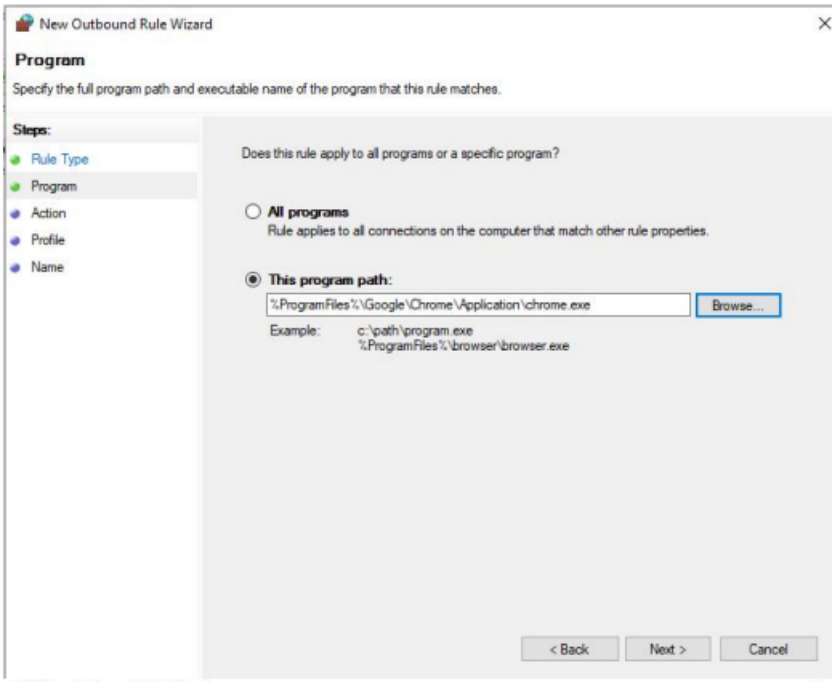

New Outbound Rule Wizard
Profile
Specify the profiles for which this rule applies.
Steps:
Rule Type
Protocol and Ports
Action
Profile
Name
When does this rule apply?
Domain
Applies when a computer is connected to its corporate domain.
Private
Applies when a computer is connected to a private network location, such as a home or work place.
Public
Applies when a computer is connected to a public network location.
< Back
Next >
Cancel

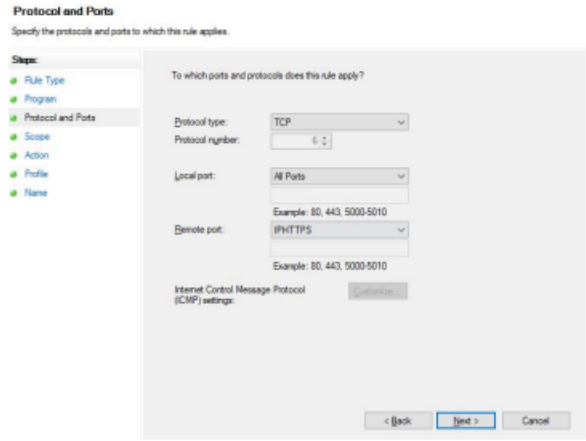

New Outbound Rule Wizard
Name
Specify the name and description of this rule.
Steps:
Rule Type
Program
Action
Profile
Name
Name:
Chrome block
Description (optional):
< Back
Finish
Cancel

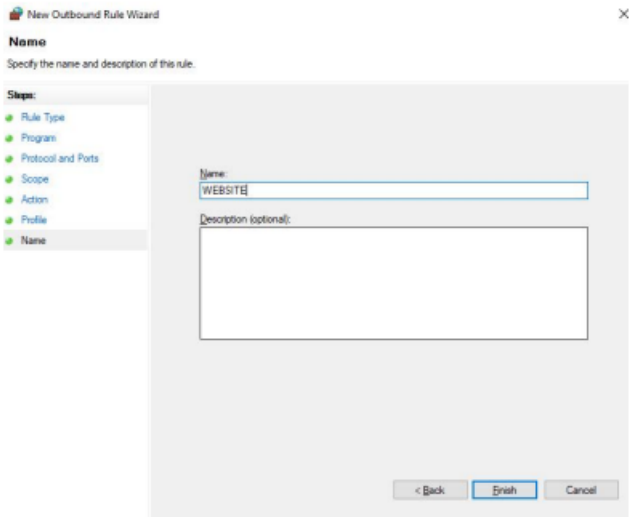

This site can't be reached
The webpage at https://www.google.com/search?q=mumbai+university&rlz=1C1ONGR_enIN1024IN1024&oq=&aqs=chrome.0.69i59i450l2.247884330j0j7&sourceid=chrome&ie=UTF-8 might be temporarily down or it may have moved permanently to a new web address.
ERR_QUIC_PROTOCOL_ERROR

A Website:


New Outbound Rule Wizard
Rule Type
Select the type of firewall rule to create.
Steps:
Rule Type
Program
Protocol and Ports
Scope
Action
Profile
Name
What type of rule would you like to create?
Program
Rule that controls connections for a program.
Port
Rule that controls connections for a TCP or UDP port.
Predefined:
@FirewallAPI.dll,-80200
Rule that controls connections for a Windows experience.
Custom
Custom rule.
< Back
Next >
Cancel


New Outbound Rule Wizard
Program
Specify the full program path and executable name of the program that this rule matches.
Steps:
Rule Type
Program
Action
Profile
Name
Does this rule apply to all programs or a specific program?
All programs
Rule applies to all connections on the computer that match other rule properties.
This program path:
%ProgramFiles%\Google\Chrome\Application\chrome.exe
Browse...
Example:
c:\path\program.exe
%ProgramFiles%\browser\browser.exe
< Back
Next >
Cancel


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:
TCP
Protocol number:
6
Local port:
All Ports
Example: 80, 443, 5000-5010
Remote port:
IPHTTPS
Example: 80, 443, 5000-5010
Internet Control Message Protocol (ICMP) settings:
Customize...
< Back
Next >
Cancel


New Outbound Rule Wizard
Name
Specify the name and description of this rule.
Steps:
Rule Type
Program
Protocol and Ports
Scope
Action
Profile
Name
Name:
WEBSITE
Description (optional):
< Back
Finish
Cancel