Date : 21/08/2020

**Practial No 1**

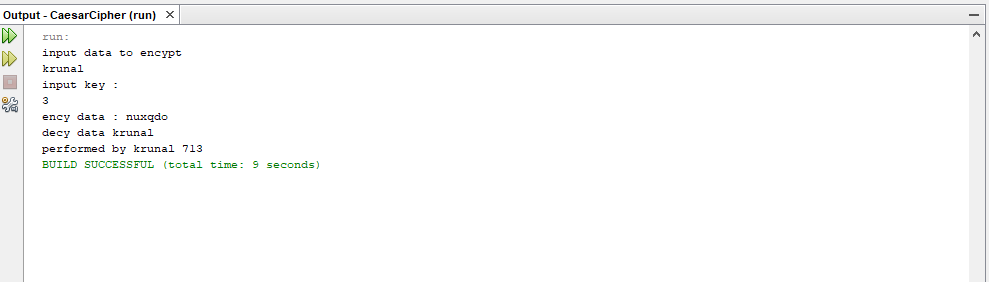
**AIM:** WAP in Java to implement the following Substitution Cipher Techniques.

**CODE**

1. **Caesar cipher :-**

|  |
| --- |
| package javaapplicationins; import java.io.\*;  import java.util.Scanner; public class CaesarCipher  {  public static void main(String[] args)  {  CaesarCipher c=new CaesarCipher(); Scanner s= new Scanner(System.in); System.out.println("Performed by krunal 713"); System.out.println("Input Data to encrypt"); String str=s.nextLine(); System.out.println("Input the key");  int key=s.nextInt();  String encrypted=c.encrypt(str,key); System.out.println("Encrypted Data:"+encrypted); String decrypted=c.decrypt(encrypted,key); System.out.println("Decrypted Data:"+decrypted);  }  String encrypt(String str,int key)  {  String encrypted="";  for(int i=0;i<str.length();i++)  {  int c=str.charAt(i); if(Character.isUpperCase(c))  {  c=c+key; if(c>'Z'){  c=c-26;  }  }  if(Character.isLowerCase(c))  {  c=c+key; if(c>'z'){ c=c-26;  }  }  encrypted +=(char) c;  }  return encrypted;  }  String decrypt(String str,int key)  {  String decrypted="";  for(int i=0;i<str.length();i++)  {  int c=str.charAt(i); if(Character.isUpperCase(c))  {  c=c-key; if(c <'A'){  c=c+26;  }  }  if(Character.isLowerCase(c))  {  c=c- key; if(c <'a'){  c = c + 26;  }  }  decrypted += (char) c;  }  return decrypted;  }  } |

**Output:**

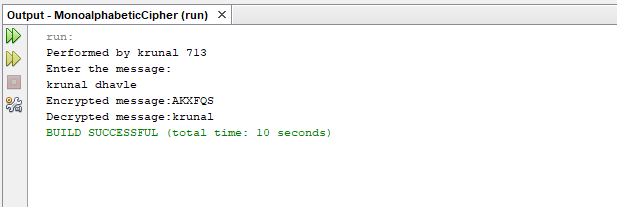
****

**B)Monoalphabetic Cipher**

**Program code:**

|  |
| --- |
| package javaapplicationins; import java.io.\*;  import java.util.Scanner;  public class MonoalphabeticCipher {  public static char p[]={'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 ch[]={'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'};  public static String doEncryption(String s)  {  char c[]=new char[(s.length())]; for (int i=0;i<s.length();i++)  {  for(int j=0;j<26;j++)  {  if(p[j]==s.charAt(i))  {  c[i]=ch[j]; break; }  }  }  return(new String(c));  }  public static String doDecryption(String s)  {  char pt[]=new char[(s.length())]; for (int i=0;i<s.length();i++)  {  for(int j=0;j<26;j++)  {  if(ch[j]==s.charAt(i))  {  pt[i]=p[j]; break; }  }  }  return(new String(pt));  }  public static void main(String args[])  {  Scanner sc=new Scanner(System.in); System.out.println("Performed by krunal 713"); System.out.println("Enter the message:");  String en=doEncryption(sc.next().toLowerCase());  System.out.println("Encrypted message:"+en); System.out.println("Decrypted message:"+doDecryption(en));  sc.close();  }  } |

**Output:**

****

Date: 02/09/2020

**Practical no 2**

**AIM:** Write program to implement the following Substitution Cipher Techniques

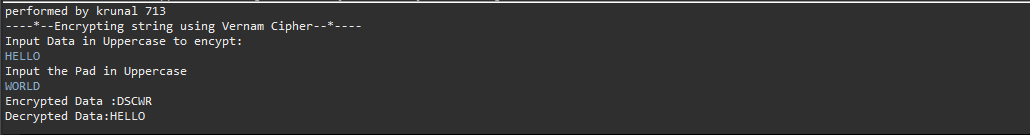
a)Vernam Cipher b)Playfair Cipher

**Code:**

**a)Vernam Cipher**

|  |
| --- |
| import java.util.Scanner;  public class Vernam {  String encrypt(String str, String pad) {  String encrypted = "";  for (int i = 0; i < str.length(); i++) {  int c = str.charAt(i);  int p = pad.charAt(i);  c = (c + p);  if (c > 'Z') {  c = c%26;  c = c+65;  }  encrypted += (char) c;  }  return encrypted;  }  String decrypt(String str, String pad) {  String decrypted = "";  for (int i = 0; i < str.length(); i++) {  int c = str.charAt(i);  int p = pad.charAt(i);  c = (c - p)+26;  if (c < 'A') {  c = (c%26);  c = c+65;  }  decrypted += (char) c;  }  return decrypted;  }  public static void main(String[] args) {  System.***out***.println("performed by krunal 713");  System.***out***.println("----\*--Encrypting string using Vernam Cipher--\*----");  Vernam v = new Vernam();  Scanner s = new Scanner(System.***in***);  System.***out***.println("Input Data in Uppercase to encypt:");  String str = s.nextLine();  System.***out***.println("Input the Pad in Uppercase");  String pad = s.nextLine();  String encrypted = v.encrypt(str, pad);  System.***out***.println("Encrypted Data :" + encrypted);  String decrypted = v.decrypt(encrypted, pad);  System.***out***.println("Decrypted Data:" + decrypted);  }  } |

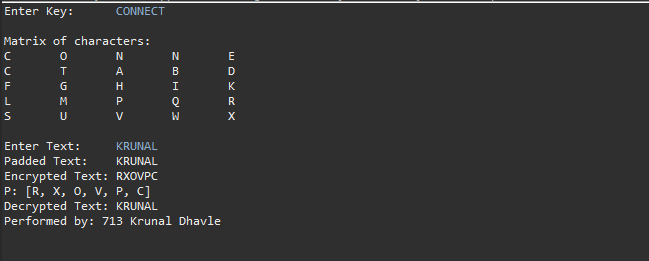
**Output:**



**b)Playfair Cipher**

|  |
| --- |
| import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.util.Arrays;  public class PlayFair {  public static char *keymat*[][] = new char[5][5];  public static String *trans* = "J";  public static char *subs* = 'X';  private static int decrem(int pos) {  if (pos < 0) {  return pos + 5;  } else {  return pos;  }  }  private static int[] srch(char c) {  int i, j;  int[] pos = new int[2];  for (i = 0; i < 5; i++) {  for (j = 0; j < 5; j++) {  if (*keymat*[i][j] == c) {  pos[0] = i;  pos[1] = j;  break;  }  }  }  return pos;  }  private static String encrypt(char c1, char c2) {  int[] pos1 = new int[2];  int[] pos2 = new int[2];  String frag = "";  pos1 = *srch*(c1);  pos2 = *srch*(c2);  if (pos1[0] == pos2[0]) { //condition for same row  c1 = *keymat*[pos1[0]][(pos1[1] + 1) % 5];  c2 = *keymat*[pos2[0]][(pos2[1] + 1) % 5];  frag = ("" + c1 + c2 + "");  } else if (pos1[1] == pos2[1]) { //condition for same column  c1 = *keymat*[(pos1[0] + 1) % 5][pos1[1]];  c2 = *keymat*[(pos2[0] + 1) % 5][pos2[1]];  frag = ("" + c1 + c2 + "");  } else { //condition for different row & column  c1 = *keymat*[pos2[0]][pos1[1]];  c2 = *keymat*[pos1[0]][pos2[1]];  frag = ("" + c1 + c2 + "");  }  return frag;  }  private static String decrypt(char c1, char c2) {  int[] pos1 = new int[2];  int[] pos2 = new int[2];  String frag = "";  pos1 = *srch*(c1);  pos2 = *srch*(c2);  if (pos1[0] == pos2[0]) { //condition for same row  c1 = *keymat*[pos1[0]][*decrem*(pos1[1] - 1) % 5];  c2 = *keymat*[pos2[0]][*decrem*(pos2[1] - 1) % 5];  frag = ("" + c1 + c2 + "");  } else if (pos1[1] == pos2[1]) { //condition for same column  c1 = *keymat*[*decrem*(pos1[0] - 1) % 5][pos1[1]];  c2 = *keymat*[*decrem*(pos2[0] - 1) % 5][pos2[1]];  frag = ("" + c1 + c2 + "");  } else { //condition for different row & column  c1 = *keymat*[pos2[0]][pos1[1]];  c2 = *keymat*[pos1[0]][pos2[1]];  frag = ("" + c1 + c2 + "");  }  return frag;  }  public static void main(String[] args) throws IOException {  BufferedReader br = new BufferedReader(new InputStreamReader(System.***in***));  String key;  int p = 0, k = 0, c = 65;  System.***out***.print("Enter Key:\t");  key = br.readLine();  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 5; j++) {  if (p < key.length()) {  *keymat*[i][j] = key.charAt(p);  p++;  } else {  if ((char) c == 'J') {  c++;  }  for (; k < key.length();) {  if ((char) c == key.charAt(k)) {  k = 0;  c++;  }  k++;  }  *keymat*[i][j] = (char) c;  c++;  k = 0;  }  }  }  System.***out***.println("\nMatrix of characters:");  for (int i = 0; i < 5; i++) {  for (int j = 0; j < 5; j++) {  System.***out***.print(*keymat*[i][j] + "\t");  }  System.***out***.println();  }  String etext = "", dtext = "";  System.***out***.print("\nEnter Text: \t");  String text = br.readLine();  text = text.toUpperCase();  text = text.replaceAll("\\s", ""); //removes whitespaces  text = text.replace(*trans*, "I"); //replaces J with I  text = text.replaceAll("([A-Z])\\1+", "$1" + *subs* + "$1");  if (text.length() % 2 != 0) {  text += *subs*;  }  char[] PTC = text.toCharArray();  System.***out***.println("Padded Text:\t" + text);  for (int i = 0; i < text.length(); i += 2) {  etext += *encrypt*(PTC[i], PTC[i + 1]);  }  System.***out***.println("Encrypted Text:\t" + etext);  char[] OTC = etext.toCharArray();  System.***out***.println("P: " + Arrays.*toString*(OTC));  for (int i = 0; i < etext.length(); i += 2) {  dtext += *decrypt*(OTC[i], OTC[i + 1]);  }  System.***out***.println("Decrypted Text:\t" + dtext);  System.***out***.println("Performed by: 713 Krunal Dhavle");  }} |

**Output**



Date: 28/08/2020

**Practical no 3**

**AIM:** Write program to implement the following Transposition Cipher Techniques

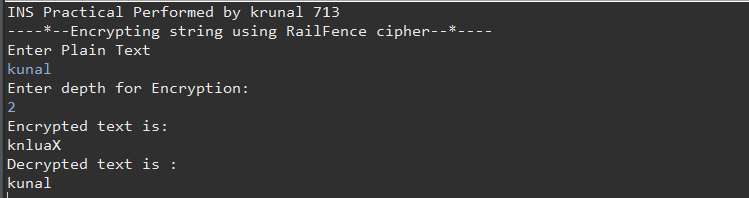
a)Rail Fence Cipher b)Simple Columnar Technique

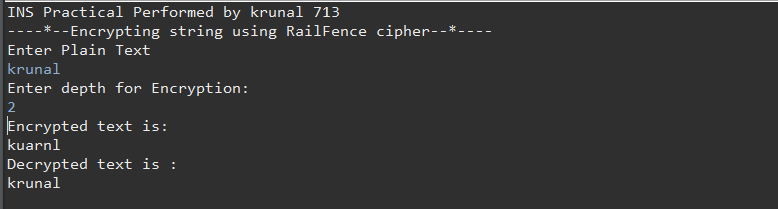
**Code:**

**a)Rail Fence Cipher**

|  |
| --- |
| import java.util.Scanner;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Rails {  String Encrypytion(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++);  }  }  }  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 ;  }  public static void main(String[] args) {  try {  System.***out***.println("INS Practical Performed by krunal 713");  System.***out***.println("----\*--Encrypting string using RailFence cipher--\*----");  Rails rf = new Rails();  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();  while(plainText.length()%depth!=0){  plainText+='X';  }  cipherText=rf.Encrypytion(plainText, depth);  System.***out***.println("Encrypted text is:\n" + cipherText);  decryptedText=rf.Decryption(cipherText, depth);  decryptedText=decryptedText.replace("X","");  System.***out***.println("Decrypted text is :\n"+decryptedText);  }catch (Exception ex){  Logger.*getLogger*(Rails.class.getName()).log(Level.***SEVERE***,null,ex);  }  }  } |

**Output:**

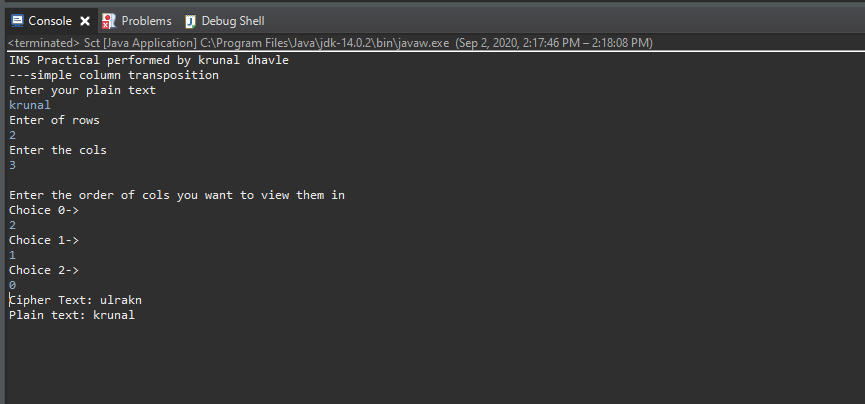




**b) Simple Columnar Technique**

|  |
| --- |
| package prac3b;  import java.io.BufferedReader;  import java.io.\*;  import java.io.InputStreamReader;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Sct {  public static void main(String[] args) {    try {  System.***out***.println("INS Practical performed by krunal dhavle ");  System.***out***.println("---simple column transposition ");  BufferedReader br = new BufferedReader(new InputStreamReader(System.***in***));  System.***out***.println("Enter your plain text");  String accept = br.readLine();  System.***out***.println("Enter of rows ");  int r = Integer.*parseInt*(br.readLine());  System.***out***.println("Enter the cols");  int c = Integer.*parseInt*(br.readLine());  int count = 0;  char table[][] = new char[r][c];  for (int i = 0; i < r; i++)  {  for (int j = 0; j < c; j++)  {  table[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());  }  String cipher = "", plain = "";  for (int j = 0; j < c; j++)  {  int k = choice[j];  for (int i = 0; i < r; i++)  {  cipher += table[i][k];  }  }  cipher = cipher.trim();  System.***out***.println("Cipher Text: "+cipher);  char mat[][] = new char[r][c];  int t = 0;  for (int j = 0; j < c; j++)  {  int k = choice[j];  for (int i = 0; i < r; i++)  {  mat[i][k] = cipher.charAt(t++);  }  }  for (int i = 0; i < r; i++)  {  for (int j = 0; j < c; j++)  {  plain += mat[i][j];  }  }  plain = plain.trim();  System.***out***.println("Plain text: "+plain);  }  catch (IOException ex)  {  Logger.*getLogger*(Sct.class.getName()).log(Level.***SEVERE***, null, ex); }}} |

**Output:**



Date:11/09/2020

**Practical no 4**

**AIM:** Write program to encrypt and decrypt strings using

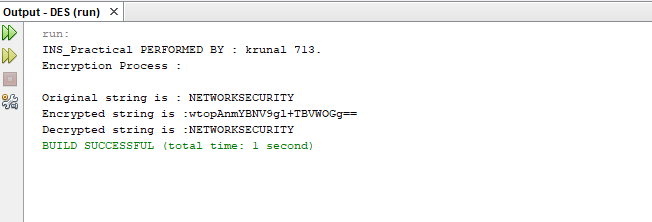
1) DES Algorithm 2) AES Algorithm

**CODE**

**1) DES Algorithm**

|  |
| --- |
| import java.util.logging.Level;  import java.util.logging.Logger;  import java.util.Base64;  import javax.crypto.Cipher;  import javax.crypto.KeyGenerator;  import javax.crypto.SecretKey;  public class DES {  public static SecretKey getSecretEncryptionKey() throws Exception{  KeyGenerator generator=KeyGenerator.getInstance("DES");  SecretKey secKey=generator.generateKey();  return secKey;  }  public String encrypt(SecretKey key,String Plaintext) throws Exception{  byte[] utf8=Plaintext.getBytes();  Cipher ecipher=Cipher.getInstance("DES");  ecipher.init(Cipher.ENCRYPT\_MODE, key);  byte[] enc=ecipher.doFinal(utf8);  Base64.Encoder encoder=Base64.getEncoder();  String et=encoder.encodeToString(enc);  return et;  }  public String decrypt(SecretKey key,String Ciphertext) throws Exception{  Base64.Decoder decoder = Base64.getDecoder();  byte[] dec=decoder.decode(Ciphertext);  Cipher dcipher=Cipher.getInstance("DES");  dcipher.init(Cipher.DECRYPT\_MODE, key);  byte[] utf8=dcipher.doFinal(dec);  return new String(utf8,"UTF8");    }  public static void main(String[] args){  try{  System.out.println("INS\_Practical PERFORMED BY : krunal 713.");  System.out.println("----'--Encrypting string using DES--'----");  System.out.println();  String message ="NETWORKSECURITY";  DES d=new DES();  SecretKey key=getSecretEncryptionKey();  String Encrypted=d.encrypt(key, message);  String Decrypted=d.decrypt(key, Encrypted);  System.out.println("Original String is : "+ message);  System.out.println("Encrypted String is : "+ Encrypted);  System.out.println("Decrypted String is : "+ Decrypted);  }catch (Exception ex){  Logger.getLogger(DES.class.getName()).log(Level.SEVERE,null,ex);  }  }  } |

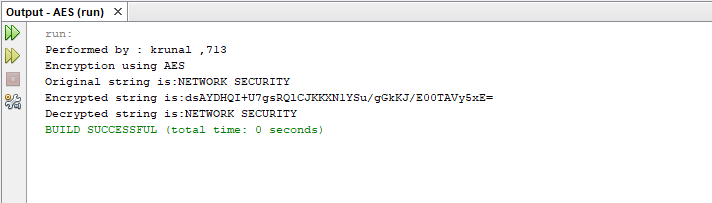
**Output:**

****

**b) AES CODE**

|  |
| --- |
| package aes;  import java.util.logging.Logger;  import java.util.logging.Level;  import javax.crypto.Cipher;  import javax.crypto.KeyGenerator;  import javax.crypto.SecretKey;  public class AES {  public static SecretKey getSecretEncryptionKey() throws Exception{  KeyGenerator generator = KeyGenerator.getInstance("AES");  generator.init(128);  SecretKey secKey= generator.generateKey();  return secKey;  }    public String encrypt(SecretKey key,String Plaintext)throws Exception{  byte[] utf8= Plaintext.getBytes("UTF8");  Cipher ecipher= Cipher.getInstance("AES");  ecipher.init(Cipher.ENCRYPT\_MODE,key);  byte[] enc= ecipher.doFinal(utf8);  return new sun.misc.BASE64Encoder().encode(enc);  }    public String decrypt(SecretKey key,String Ciphertext) throws Exception{  byte[] dec= new sun.misc.BASE64Decoder().decodeBuffer(Ciphertext);  Cipher dcipher= Cipher.getInstance("AES");  dcipher.init(Cipher.DECRYPT\_MODE,key);  byte[] utf8= dcipher.doFinal(dec);  return new String(utf8, "UTF8");  }    public static void main (String[]args) throws Exception  {  try{  System.out.println("Performed by : krunal ,713");  System.out.println("Encryption using AES");  String message="NETWORK SECURITY";  AES d= new AES();  SecretKey key= getSecretEncryptionKey();  String Encrypted= d.encrypt(key, message);  String decrypted = d.decrypt(key,Encrypted);  System.out.println("Original string is:" +message);  System.out.println("Encrypted string is:" + Encrypted);  System.out.println("Decrypted string is:" +decrypted);  }  catch(Exception ex){  Logger.getLogger(AES.class.getName()).log(Level.SEVERE,null,ex) ;  }  }  } |

**Output:**



Date:29/09/2020

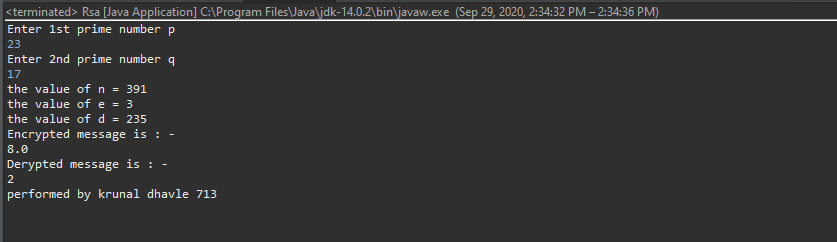
**Practical no 5**

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

**CODE**

|  |
| --- |
| package prac5;  import java.util.\*;  import java.math.\*;  public class Rsa {  public static void main(String[] args) {  // TODO Auto-generated method stub  Scanner sc=new Scanner(System.in);  int p,q,n,z,d=0,e,i;  double c;  BigInteger msgback;  System.out.println("Enter 1st prime number p");  p=sc.nextInt();  System.out.println("Enter 2nd prime number q");  q=sc.nextInt();  sc.close();  n=p\*q;  z=(p-1)\*(q-1);  System.out.println("the value of n = "+n);  for(e=2;e<z;e++)  {  if(gcd(e,z)==1) // e is for public key exponent  {  break;  }  }  System.out.println("the value of e = "+e);  for(i=0;i<=9;i++)  {  int x=1+(i\*z);  if(x%e==0) //d is for private key exponent  {  d=x/e;  break;  }  }    System.out.println("the value of d = "+d);  c=(Math.pow(2,e))%n;    System.out.println("Encrypted message is : -");  System.out.println(c);    BigInteger N = BigInteger.valueOf(n);    BigInteger C = BigDecimal.valueOf(c).toBigInteger();  msgback = (C.pow(d)).mod(N);    System.out.println("Derypted message is : -");  System.out.println(msgback);  System.out.println("performed by krunal dhavle 713");  }  static int gcd(int e, int z)  {  if(e==0)  return z;  else  return gcd(z%e,e);  }  } |

**Output:**



Date:06/10/2020

**Practical no 6**

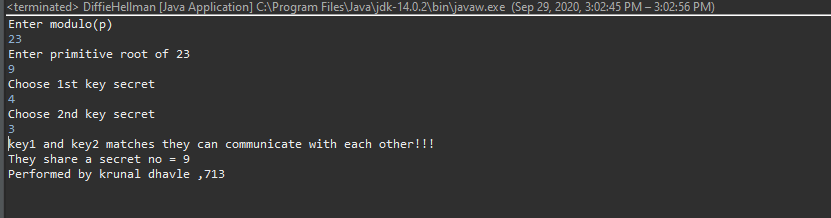
**AIM:** Write a program to implement the Diffie-Hellman Key Agreement algorithm to generate symmetric keys.

**CODE:-**

**Method 1:-**

|  |
| --- |
| package prac6;  import java.util.\*;  public class DiffieHellman {  public static void main(String[] args) {  // TODO Auto-generated method stub  Scanner sc=new Scanner(System.in);  System.out.println("Enter modulo(p)");  int p=sc.nextInt();  System.out.println("Enter primitive root of "+p);  int g=sc.nextInt();  System.out.println("Choose 1st key secret");  int a=sc.nextInt();  System.out.println("Choose 2nd key secret");  int b=sc.nextInt();  sc.close();  int A = (int)Math.pow(g,a)%p;  int B = (int)Math.pow(g,b)%p;    int S\_A = (int)Math.pow(B,a)%p;  int S\_B =(int)Math.pow(A,b)%p;    if(S\_A==S\_B)  {  System.out.println("key1 and key2 matches they can communicate with each other!!!");  System.out.println("They share a secret no = "+S\_A);  System.out.println("Performed by krunal dhavle ,713");  }    else  {  System.out.println("key1 and key2 matches they cannot communicate with each other!!!");  System.out.println("Performed by krunal dhavle ,713");  }  }} |

**Output:**



**Method 2 :-**

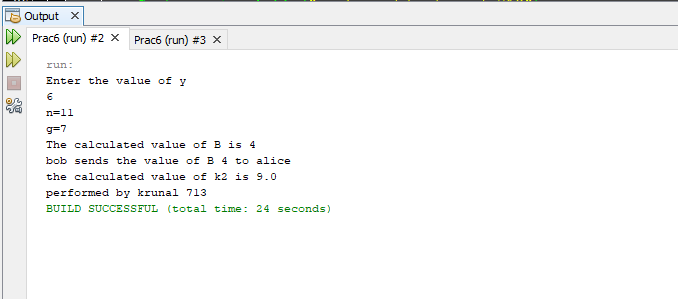
**Bob.java**

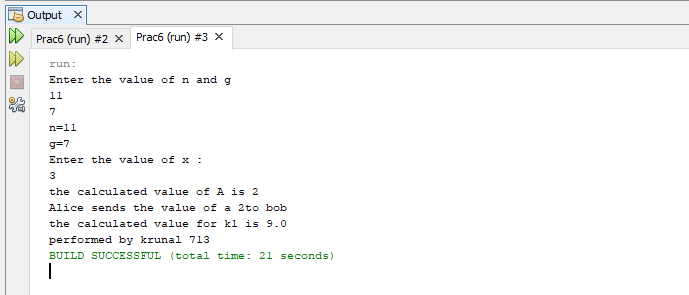
|  |
| --- |
| package prac6;  import java.io.\*;  import java.net.ServerSocket;  import java.net.Socket;  import java.util.Scanner;  public class Bob {  public static void main(String[] args) throws IOException {  ServerSocket ss = new ServerSocket(5000);  Socket s = ss.accept();  DataInputStream in = new DataInputStream(s.getInputStream());  int n = in.readInt();  int g = in.readInt();  Scanner sc = new Scanner(System.in);  System.out.println("Enter the value of y");  int y = sc.nextInt();  System.out.println("n=" +n);  System.out.println("g=" +g);  int d =(int)Math.pow(g, y);  int B =d%n;  System.out.println("The calculated value of B is " +B);  System.out.println("bob sends the value of B " +B+ " to alice");  int A = in.readInt();  int b = (int)Math.pow(A,y);  double K2 = b%n;  System.out.println("the calculated value of k2 is " +K2);  DataOutputStream out = new DataOutputStream(s.getOutputStream());  out.writeInt(B);  System.out.println("performed by krunal 713");  }  } |

**Alice.java**

|  |
| --- |
| package prac6;  import java.io.\*;  import java.net.Socket;  import java.util.Scanner;  public class Alice {  public static void main(String[] args) throws IOException {  Socket cs = new Socket("localhost" ,5000);  Scanner sc = new Scanner(System.in);  System.out.println("Enter the value of n and g ");  int n = sc.nextInt();  int g = sc.nextInt();  System.out.println("n=" +n);  System.out.println("g=" +g);  DataOutputStream out = new DataOutputStream(cs.getOutputStream());  out.writeInt(n);  out.writeInt(g);  System.out.println("Enter the value of x : ");  int x = sc.nextInt();  int c =(int)Math.pow(g,x);  int A = c%n;  System.out.println("the calculated value of A is " +A);  out.writeInt(A);  System.out.println("Alice sends the value of a " +A + "to bob");  DataInputStream in = new DataInputStream(cs.getInputStream());  int B = in.readInt();  int a = (int)Math.pow(B, x);  double K1 = a % n;  System.out.println("the calculated value for k1 is " +K1);  System.out.println("performed by krunal 713");  }  } |

**Output:**





Date:23/10/2020

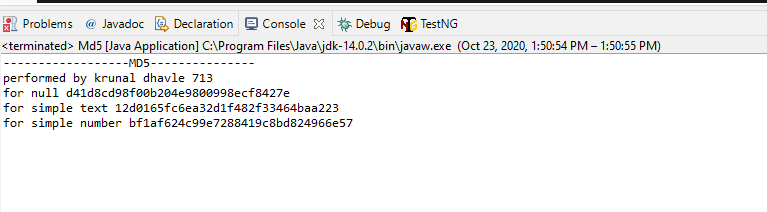
**Practical no 7**

**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** Md5 {  **public** **static** String md5(String input) **throws** NoSuchAlgorithmException {  String md5 = **null**;  **if**(**null** == input)  {  **return** **null**;  }  MessageDigest md = MessageDigest.*getInstance*("MD5");  md.update(input.getBytes());  md5 =**new** BigInteger(1 , md.digest()).toString(16);  **return** md5;  }  **public** **static** **void** main(String[] args) **throws** NoSuchAlgorithmException {  System.***out***.println("------------------MD5---------------");  System.***out***.println("performed by krunal dhavle 713");  System.***out***.println("for null " +*md5*(""));  System.***out***.println("for simple text " +*md5*(" krunal dhavle 713 "));  System.***out***.println("for simple number " +*md5*("291999"));  }  } |

**Output**

****

Date:16/10/2020

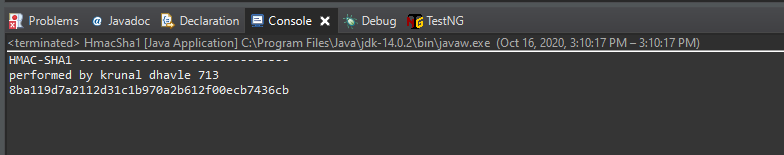
**Practical no 8**

**AIM:** Write a program to calculate HMAC-SHA1 Signature

**Code:-**

|  |
| --- |
| package prac8;  import java.util.Formatter;  import javax.crypto.\*;  import javax.crypto.spec.SecretKeySpec;  public class 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 calculateHMAC(String data , String key) throws Exception  {  SecretKeySpec signingKey = new SecretKeySpec(key.getBytes() , "HmacSHA1");  Mac mac = Mac.getInstance("HmacSHA1");  mac.init(signingKey);  return toHexString(mac.doFinal(data.getBytes()));  }    public static void main(String[] args) throws Exception  {  String hmac = calculateHMAC("krunal", "dhavle");  System.out.println("HMAC-SHA1 ------------------------------");  System.out.println("performed by krunal dhavle 713");  System.out.println(hmac);  }  } |

**Output:-**



Date:07/11/2020

**Practical no 9**

**AIM:** Configure windows firewall to block

1) A port 2) An Program 3) A Website

**Different Types of Profiles available/ When does this rule applies**

**Domain:** Applies when computer is connected to corporate domain

**Private:** Applies when computer is connected to a private network location, such as a

home or workplace.

**Public:** Applies when computer is connected to public network connection.

**Different types of actions available/What action should be taken when a connection**

**matches the specified the conditions**

**Allow the connection:** This includes connections that are protected with IPsec as well as

those are not

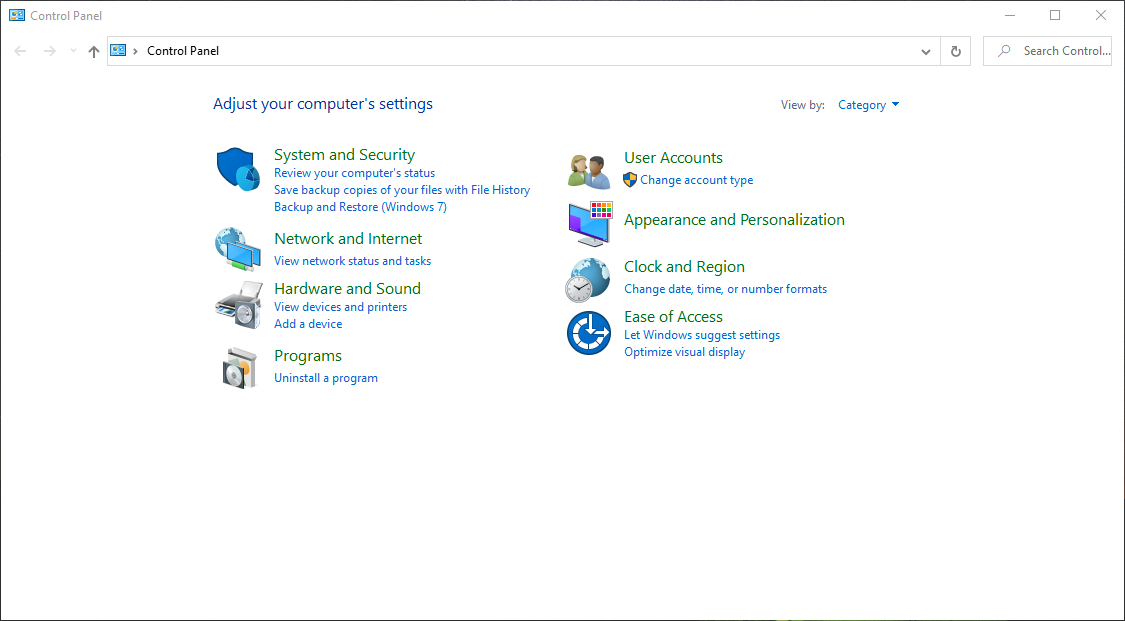
**Allow the connection if it is secure:** This includes only connections that have been

authenticated by using IPsec. Connections will be secured using the settings in IPsec

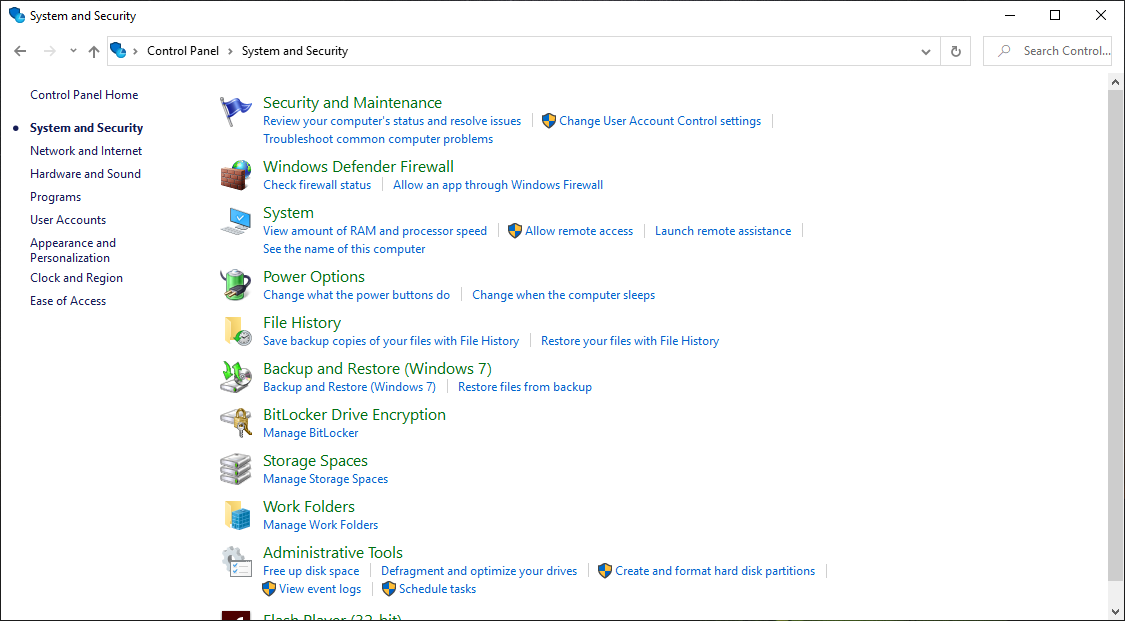
properties and rules in the Connection Security Rule node. Block the connection.

**A) Blocking a port:**

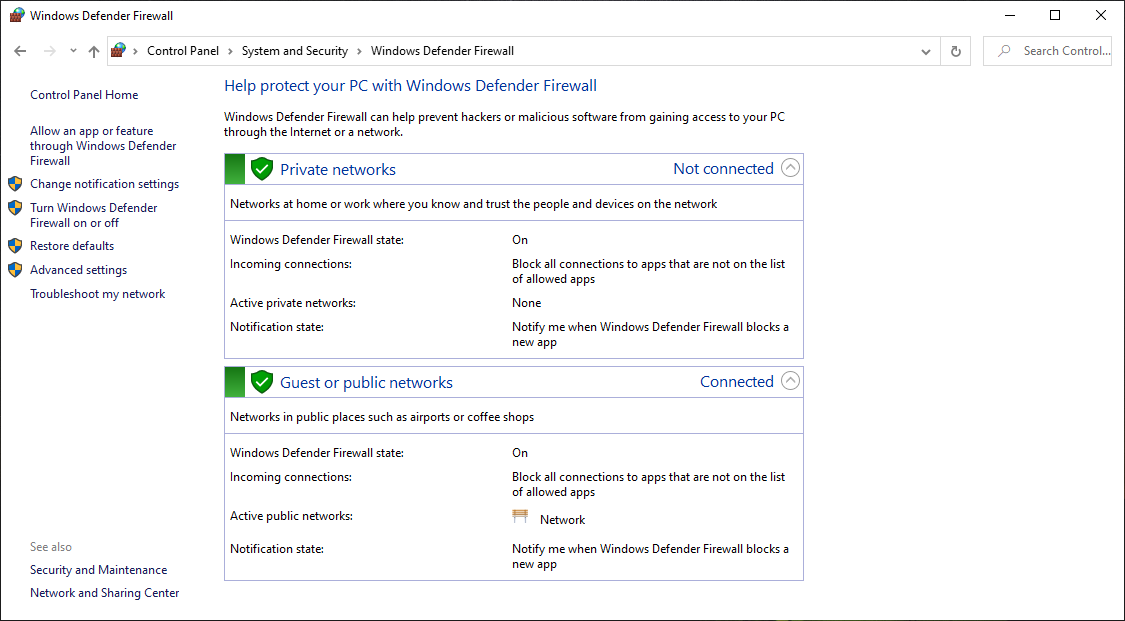
**Step 1:** Open control panel and go to System Security



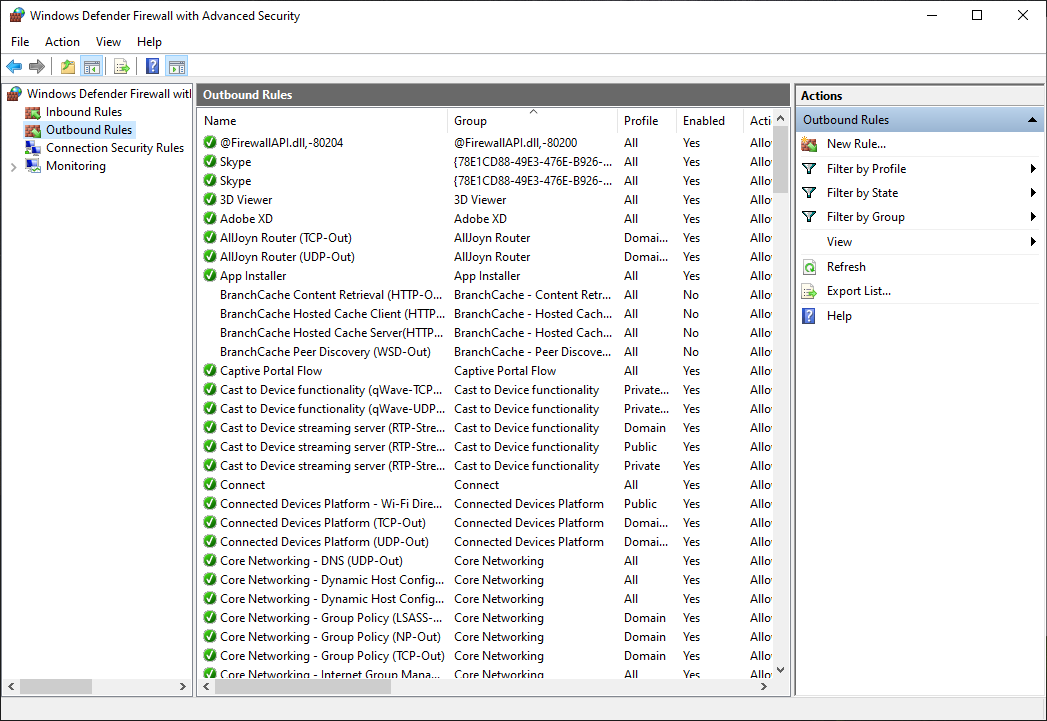
**Step 2:** Now Select Windows Defender Firewall.



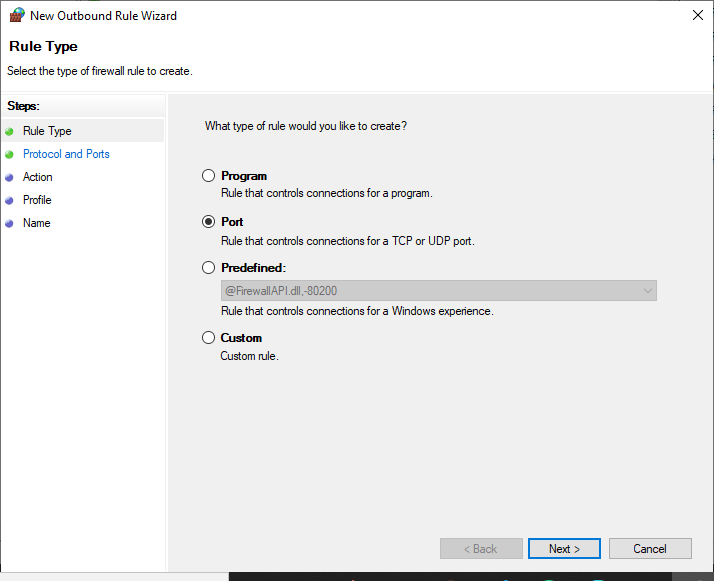
**Step 3**: Now you need to select Advanced setting.



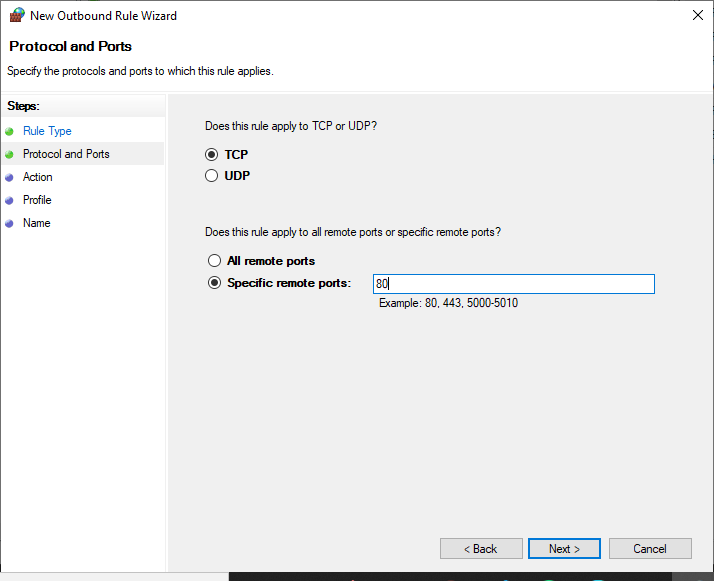
**Step 4:** Now Select Outbound Rules.



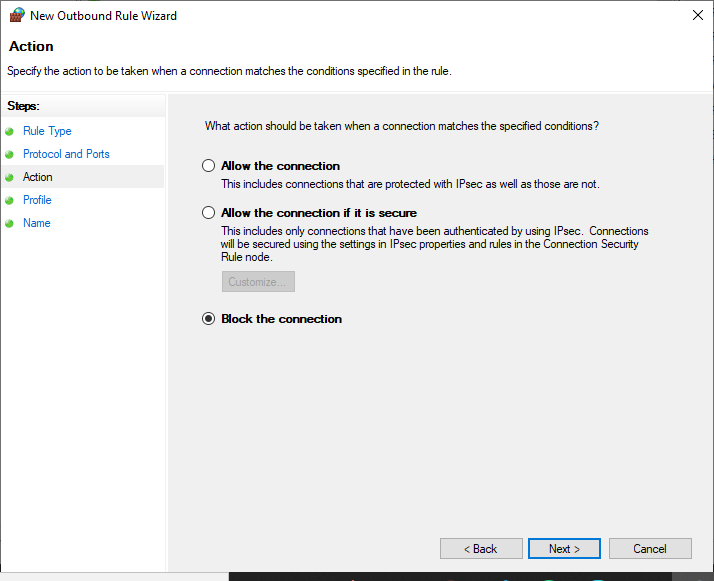
**Step 5:** Inside Outbound rules -> Select New Rules -> select Port and then click on next.



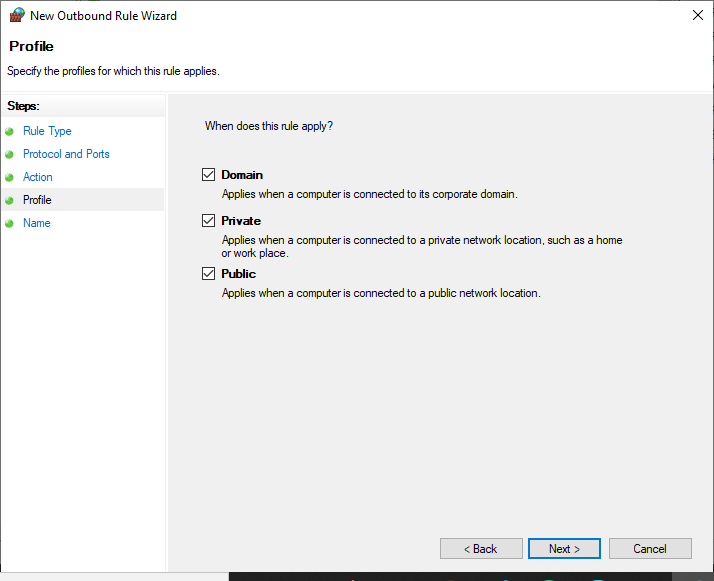
**Step 6**: Select the protocols and enter the port that you want to want to block



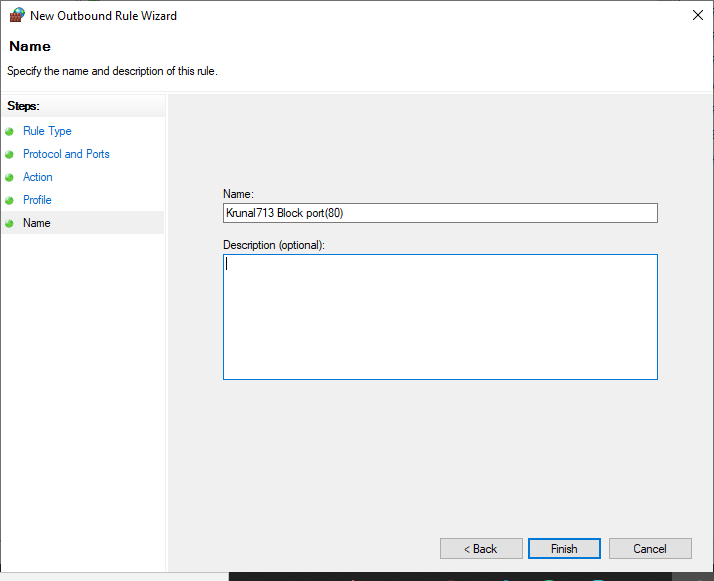
**Step 7:** Select the action block the connection for blocking a port



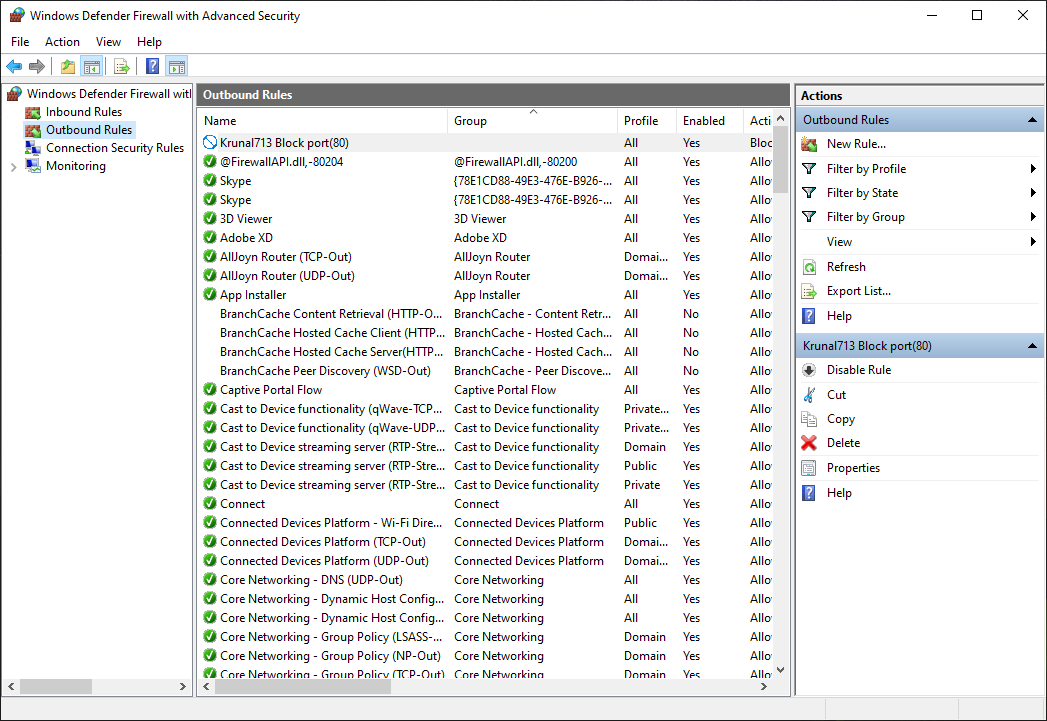
**Step 8:** Select the profiles domain private or public.



**Step 9:** Give a name to our new set rule and click on finish.



**Output:**

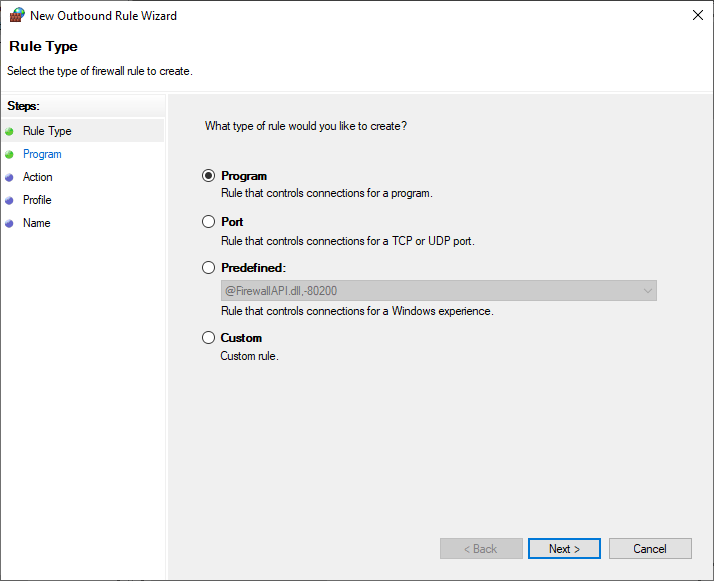
****

**B) Blocking a program:**

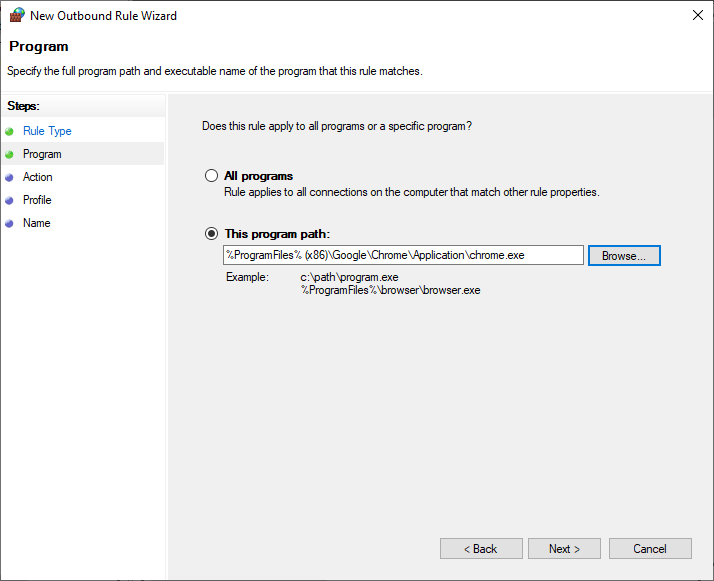
**Rule that controls the connection of a program**

**Step1: Repeat PartA Step1 to Step4.**

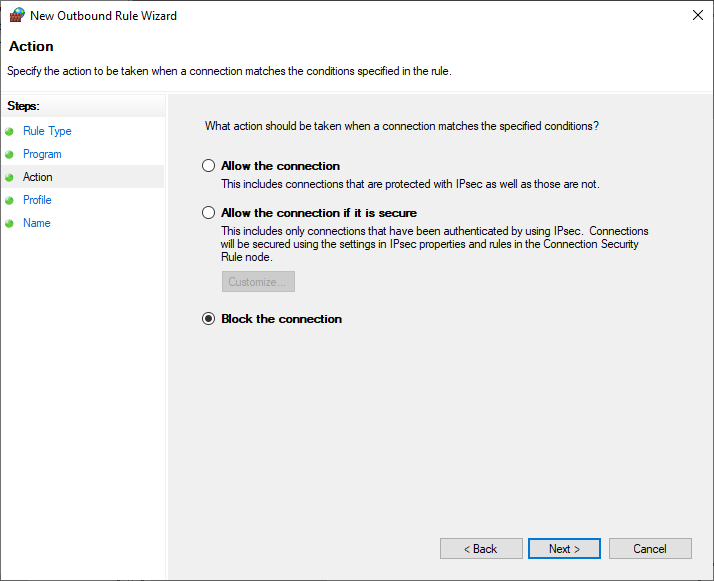
**Step 5:** Inside Outbound rules -> Select New Rules -> select a program and then click on next.



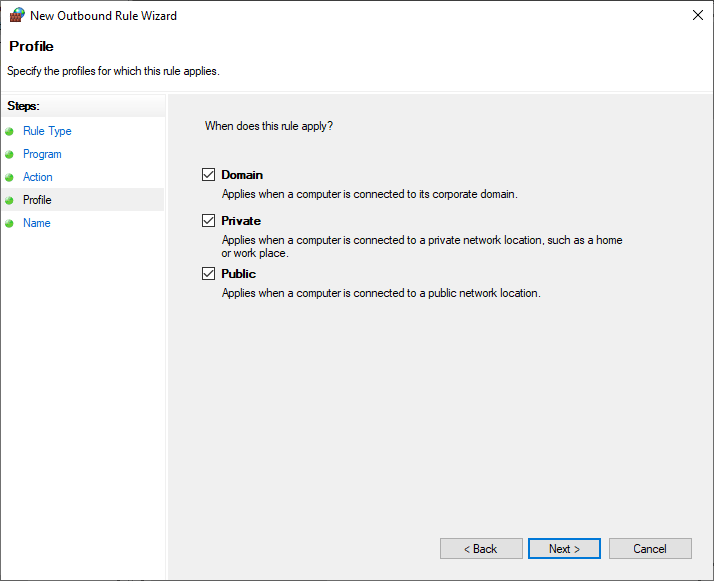
**Step 6:** Choose the path of the program from the directory.



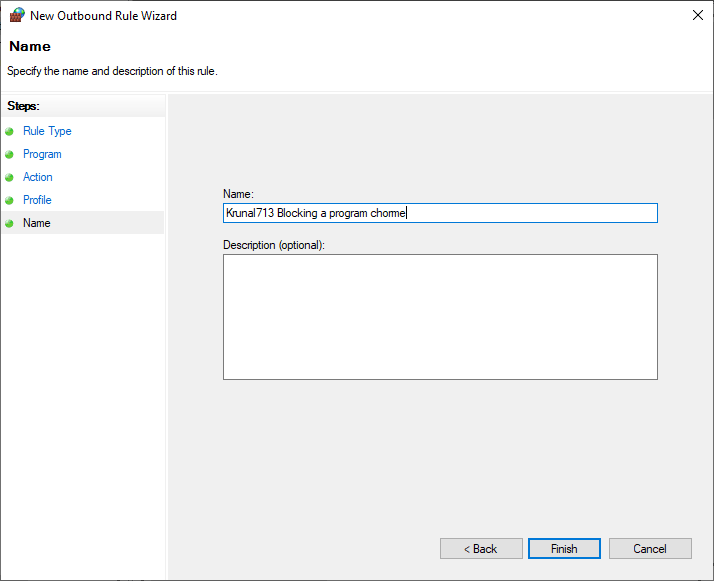
**Step 7:** Click on Block the connection.



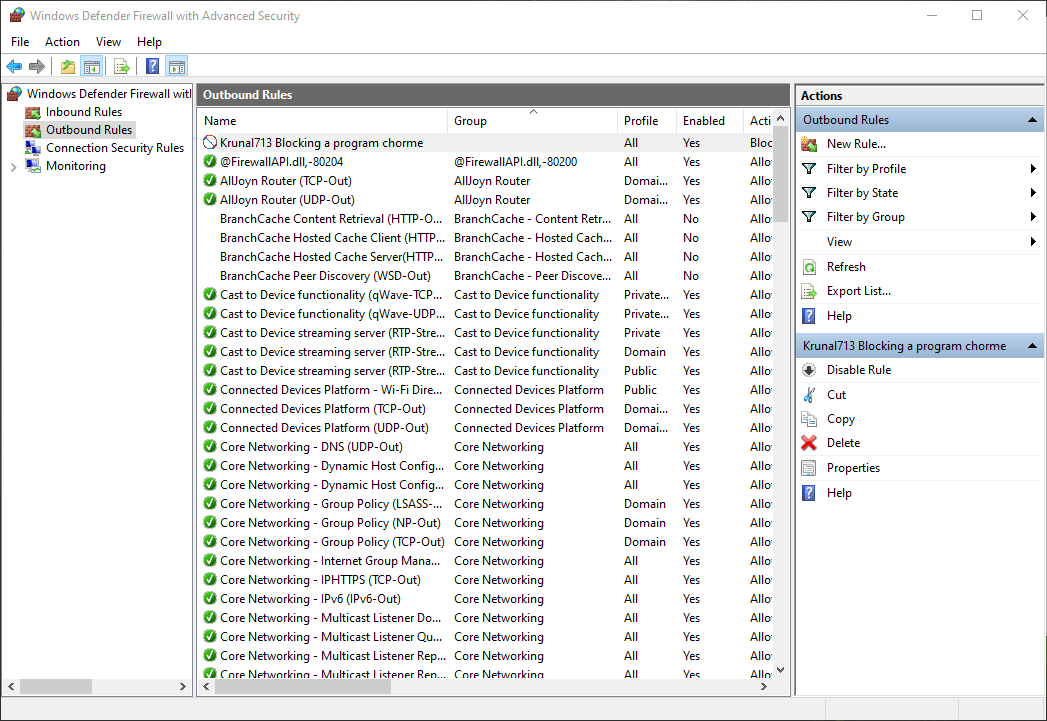
**Step 8**: Select the profiles domain private or public.



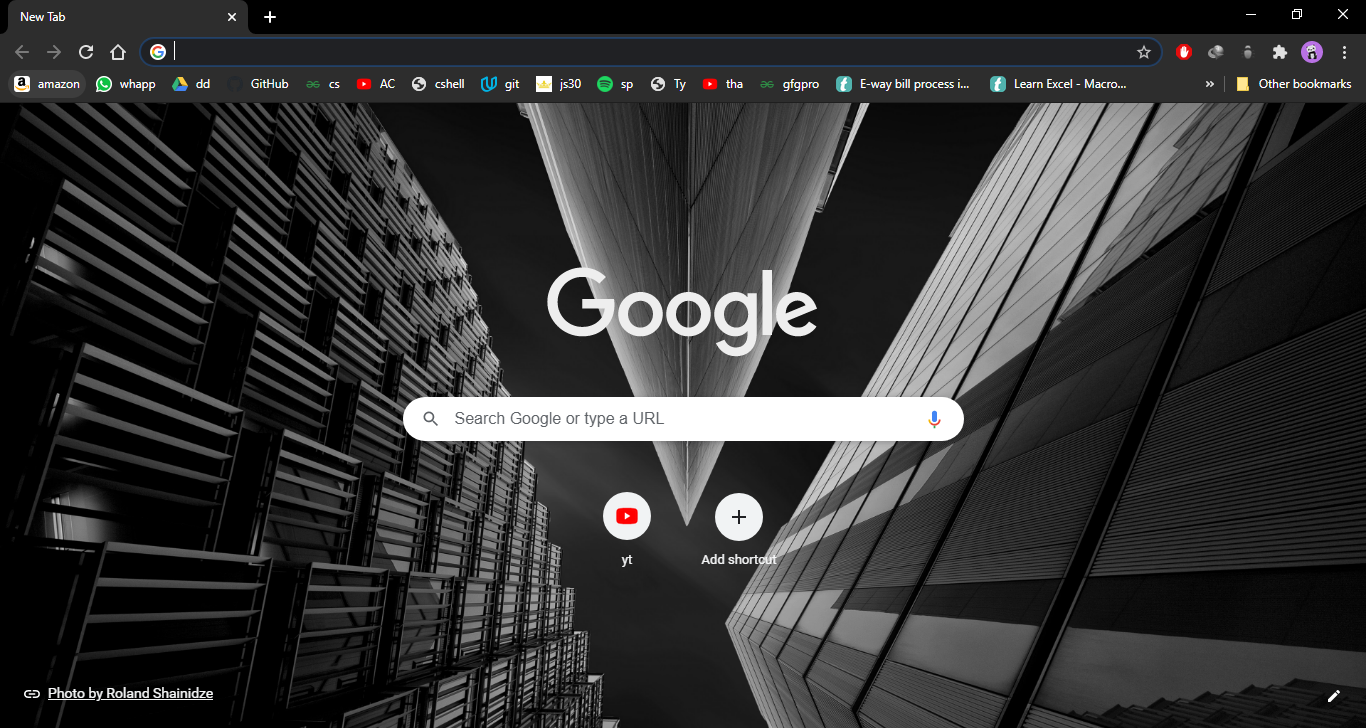
**Step 9:** Give a name to your new set rule and click on finish.



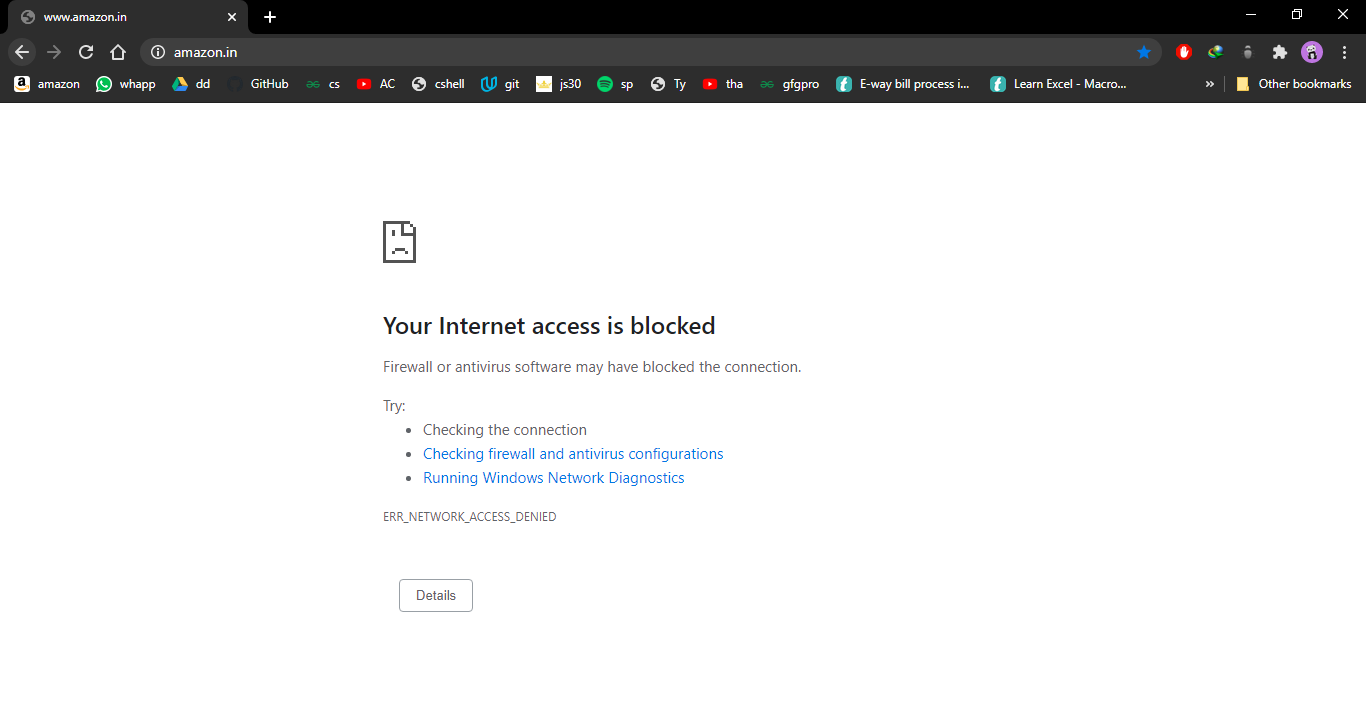
**Output:**



**Before Applying the Rule:**



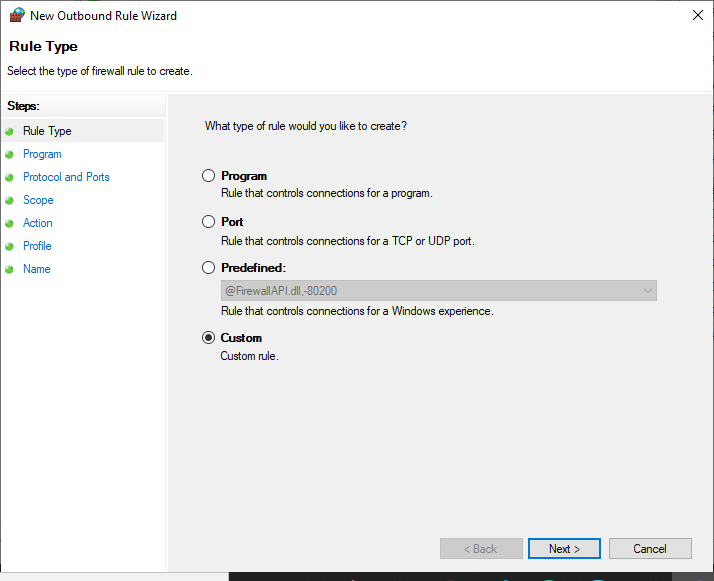
**After Applying the Rule:**

****

**C) Blocking a website:**

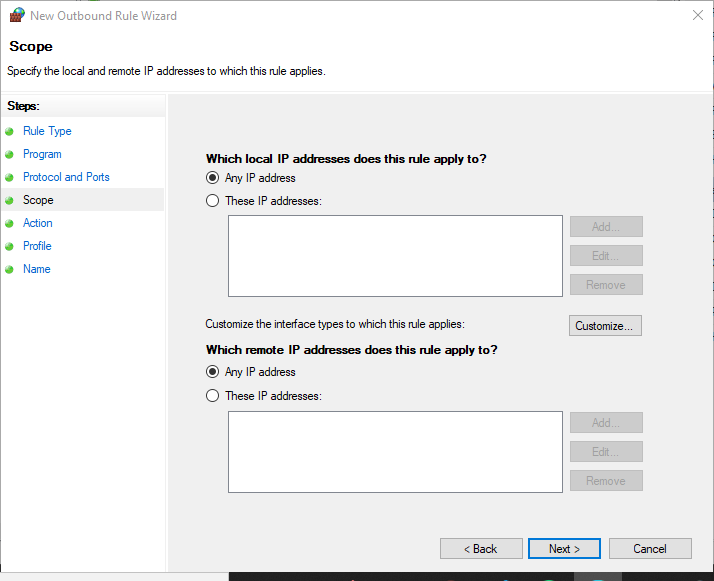
**Step1: Repeat PartA Step1 to Step4.**

**Step 5**: Inside Outbound rules -> Select New Rules -> select custom and then click on next.

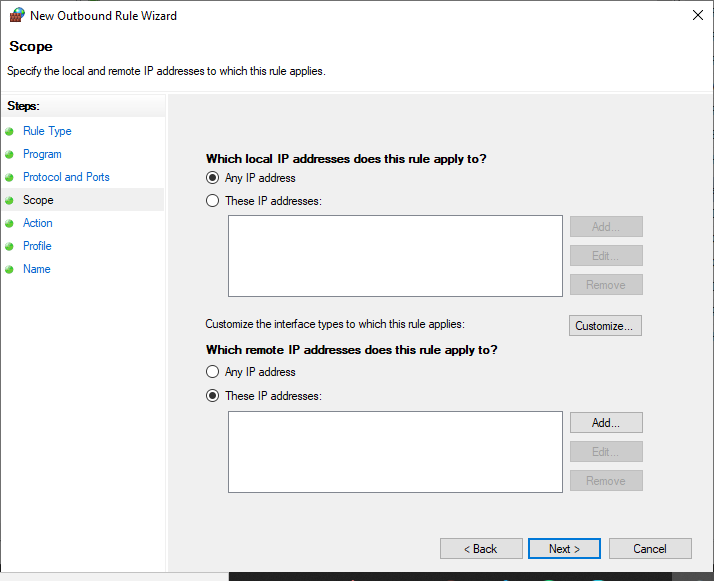


**Step 6**: When you click next you would see a window where you will see “Steps:” on left hand side of the

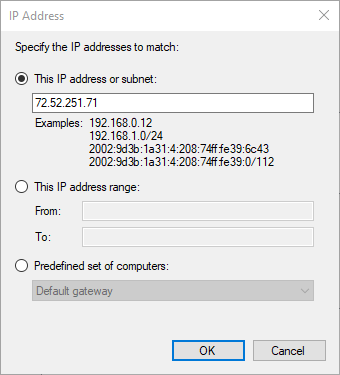
screen. From that select “Scope”.



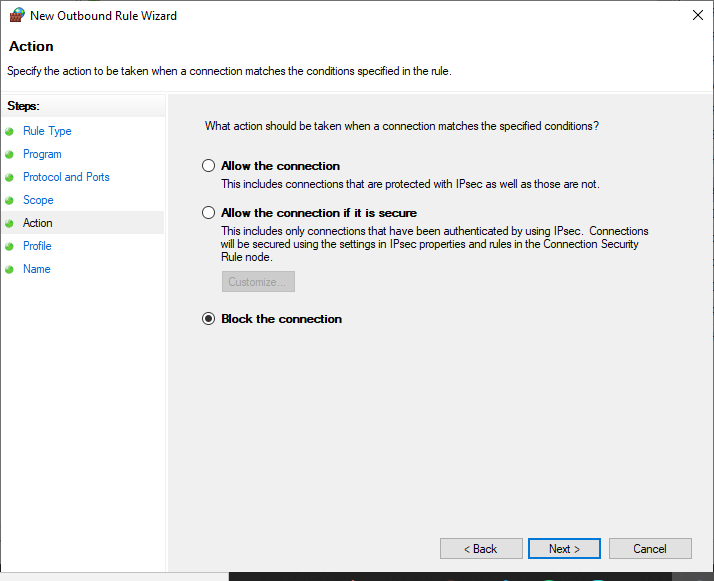
**Step 7**: In scope click on These IP addresses in remote IP



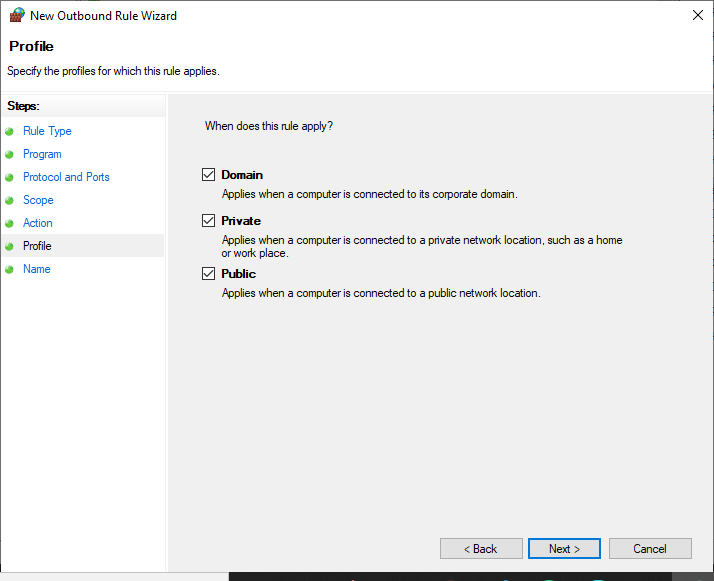
**Step 8:** Click on add and Add the IP address of the website that you want to block and click ok.



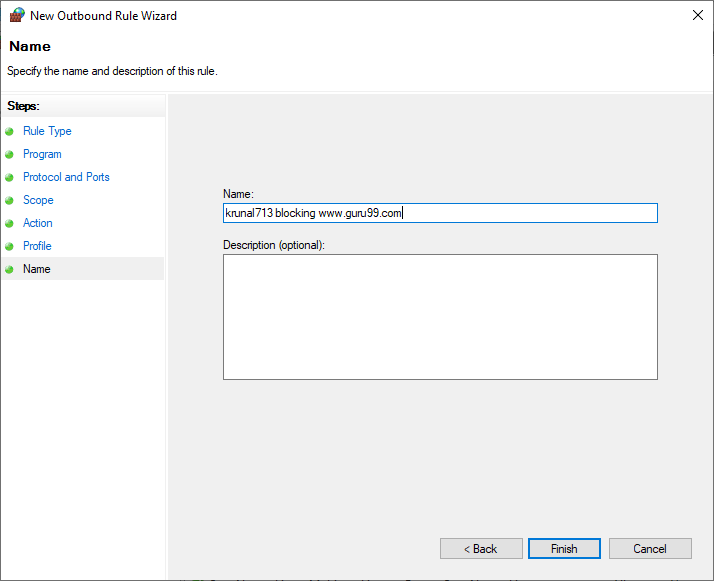
**Step 9:** Click on Block the connection in action



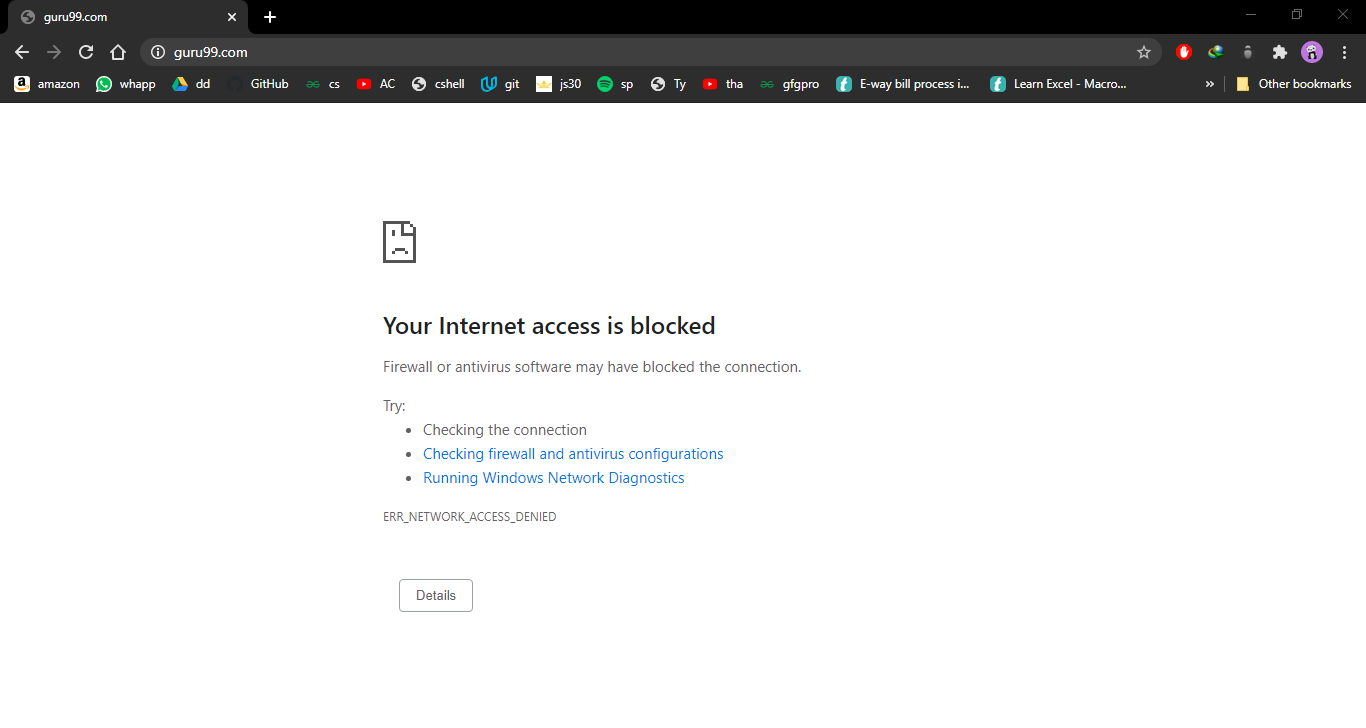
**Step 10:** Select the profiles domain private or public.

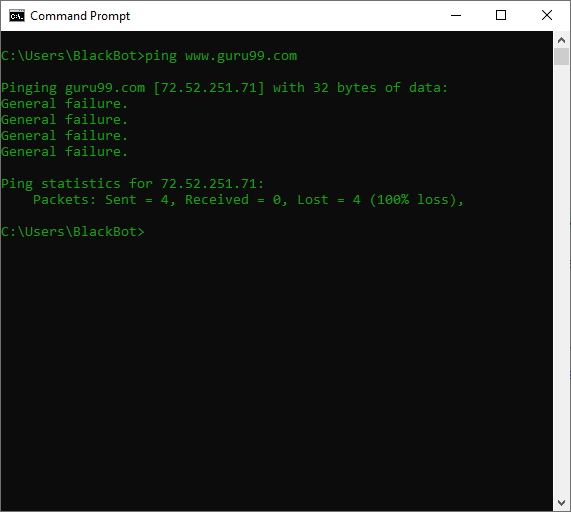


**Step 11:** Give a name to your new set rule and click on finish.



**Output:**

****

****