# EXPERIMENT – 1

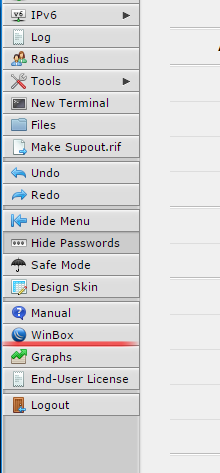
**Aim**: Make an experiment to implement WEP/WPA 2 PSK, 802.1x EAP Security Protocol.

Winbox is a small utility that allows administration of MikrotikRouterOS using a fast and simple GUI. It is a native Win32 binary, but can be run on **Linux** and **MacOS (OSX)** using Wine.

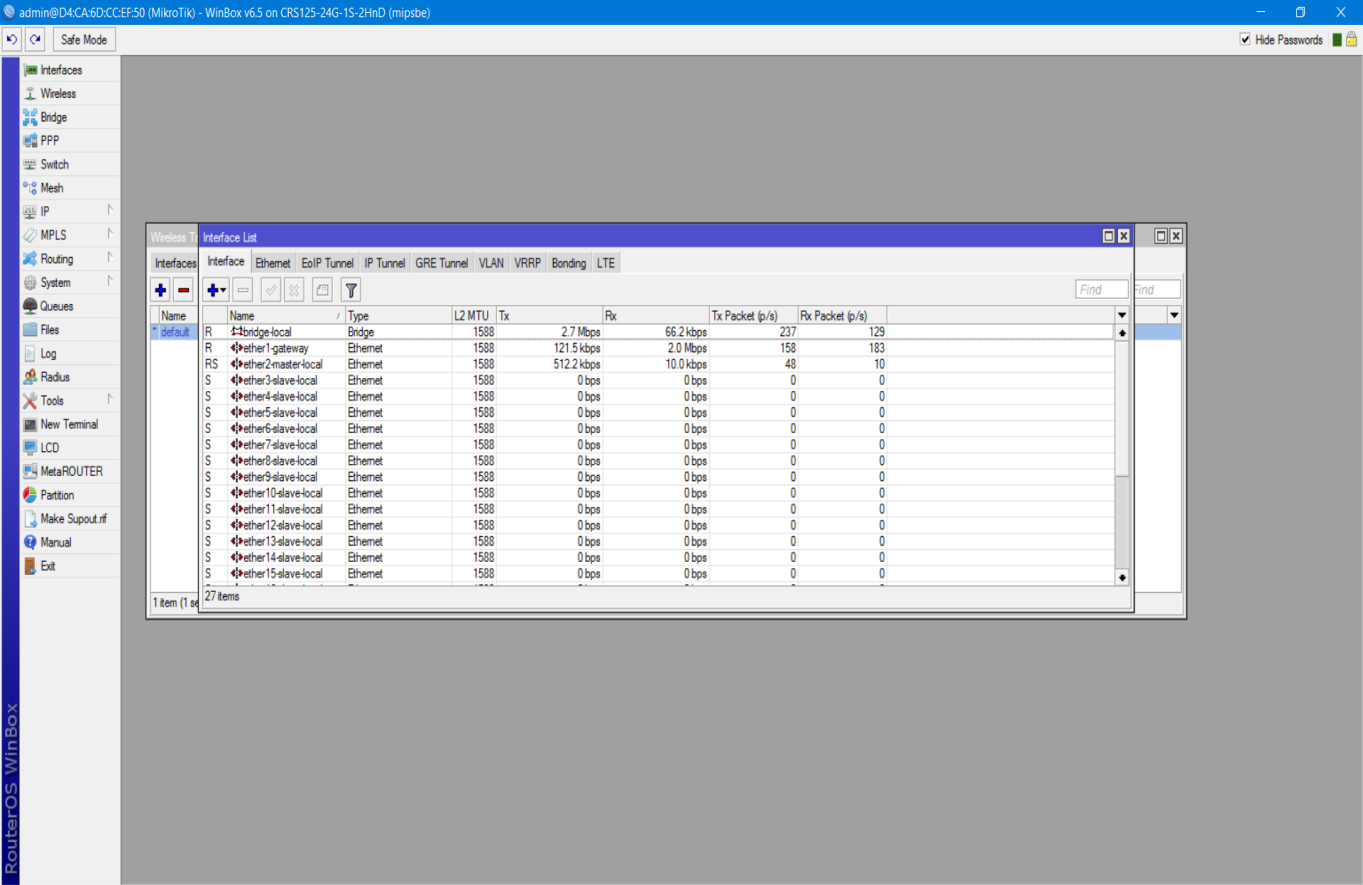
All Winbox interface functions are as close as possible to Console functions, that is why there are no Winbox sections in the manual.

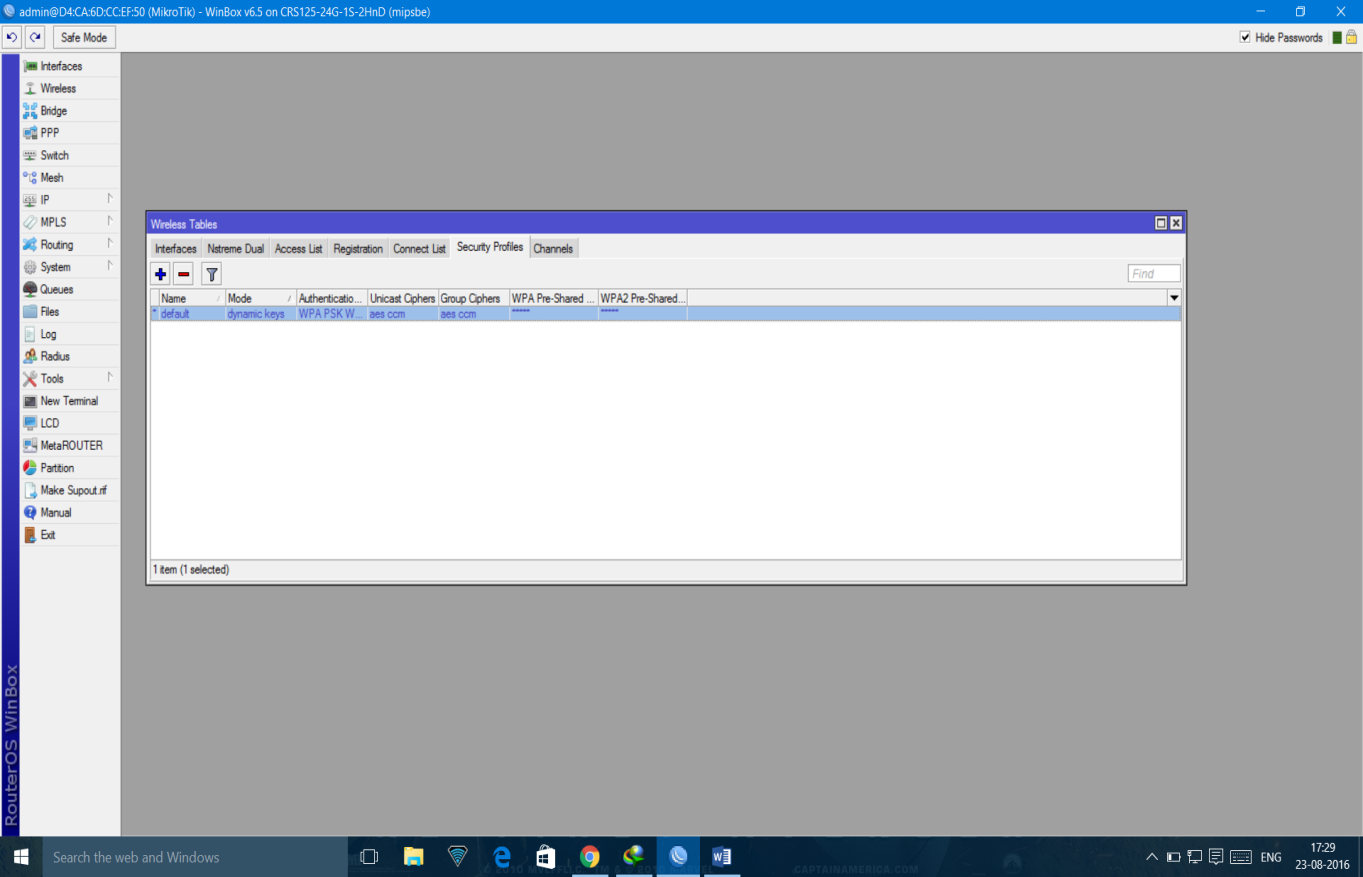
Some of advanced and system critical configurations are not possible from winbox, like MAC address change on an interface.

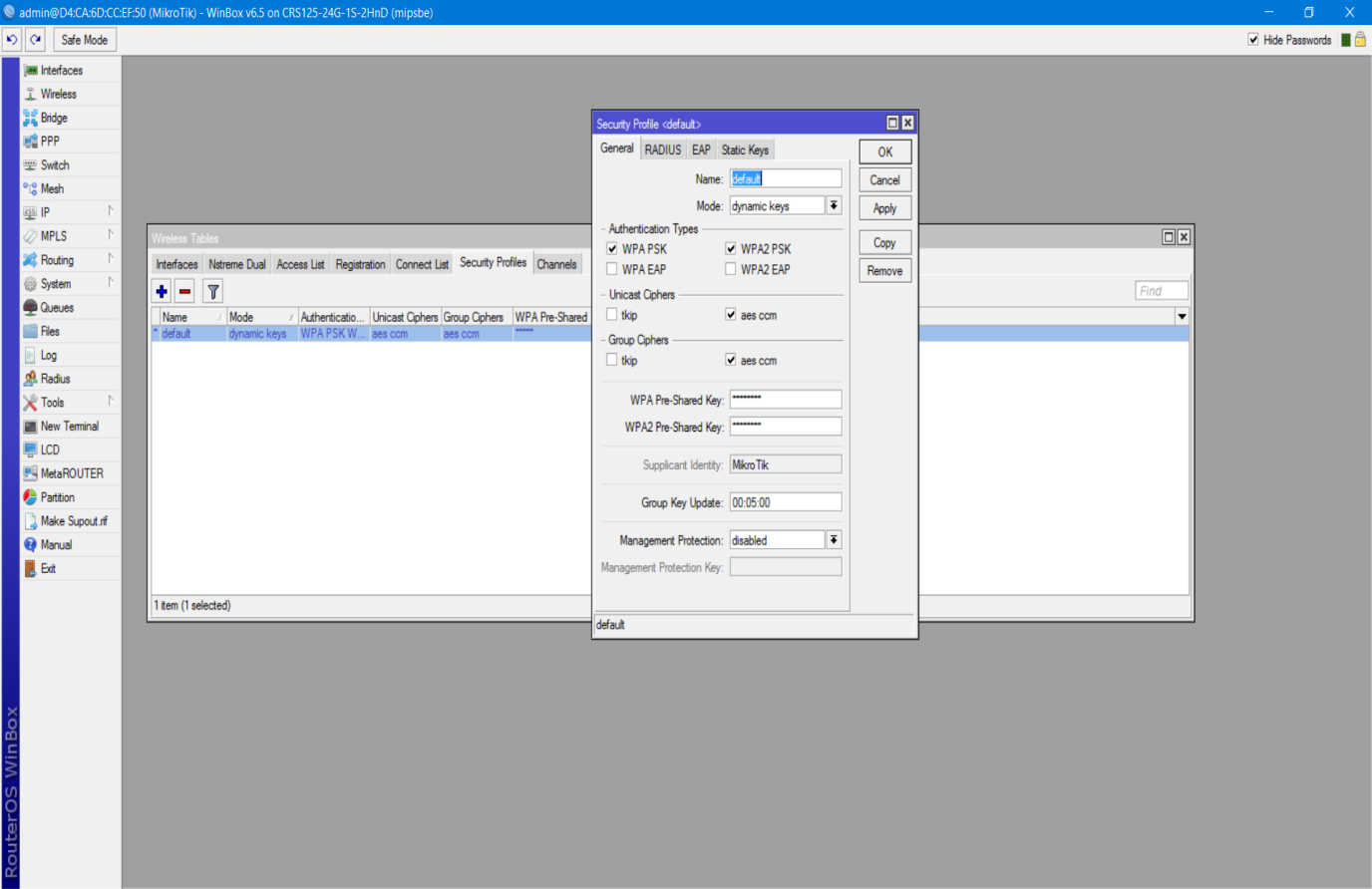
Winbox loader can be downloaded directly from the router or from the [mikrotik download page](http://www.mikrotik.com/download). When downloading from the router, open a web browser and enter router's IP address, RouterOS welcome page will be displayed. Click on the menu item that says **Winbox** to download **winbox.exe** from MikroTik download server.

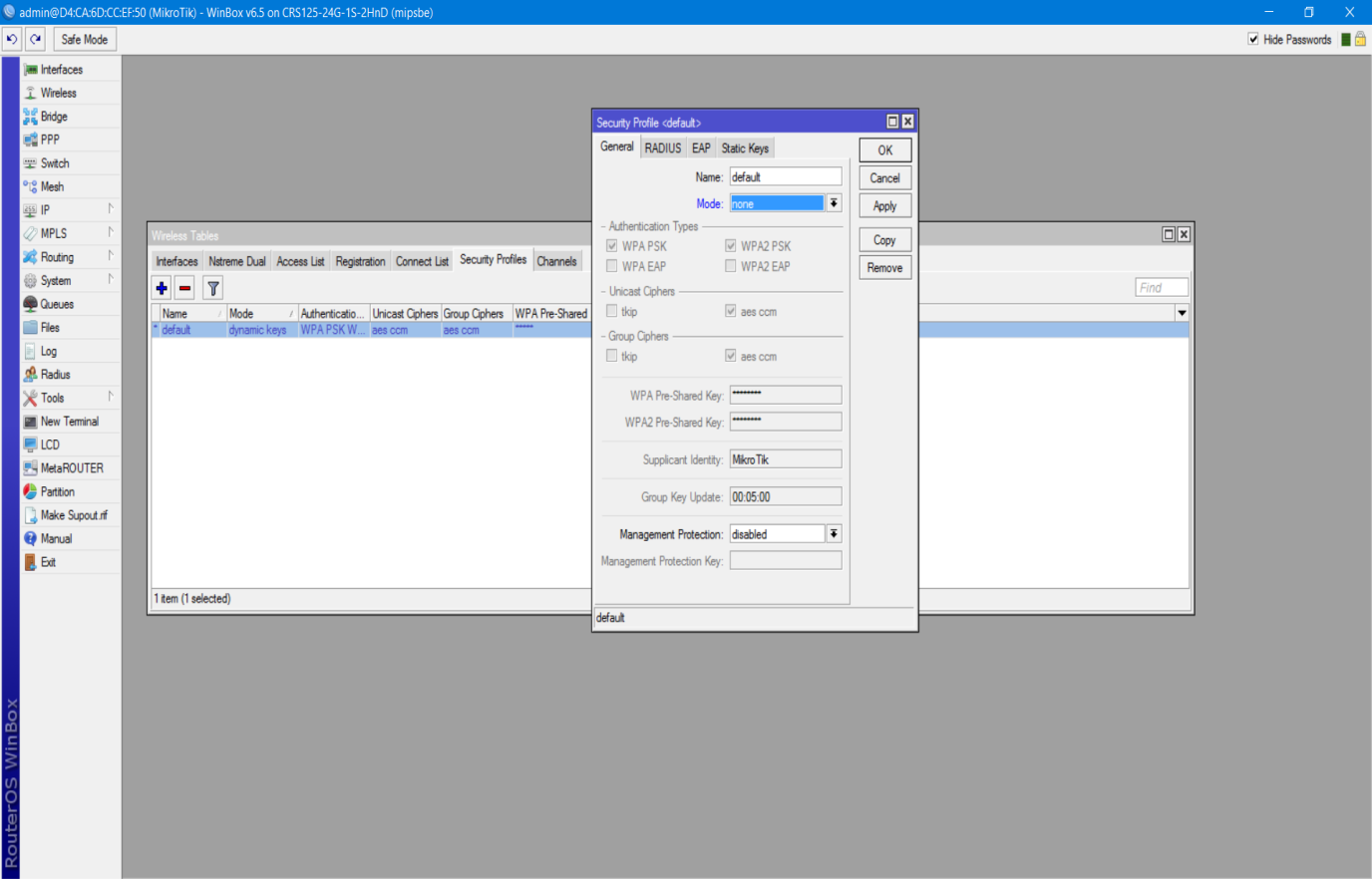


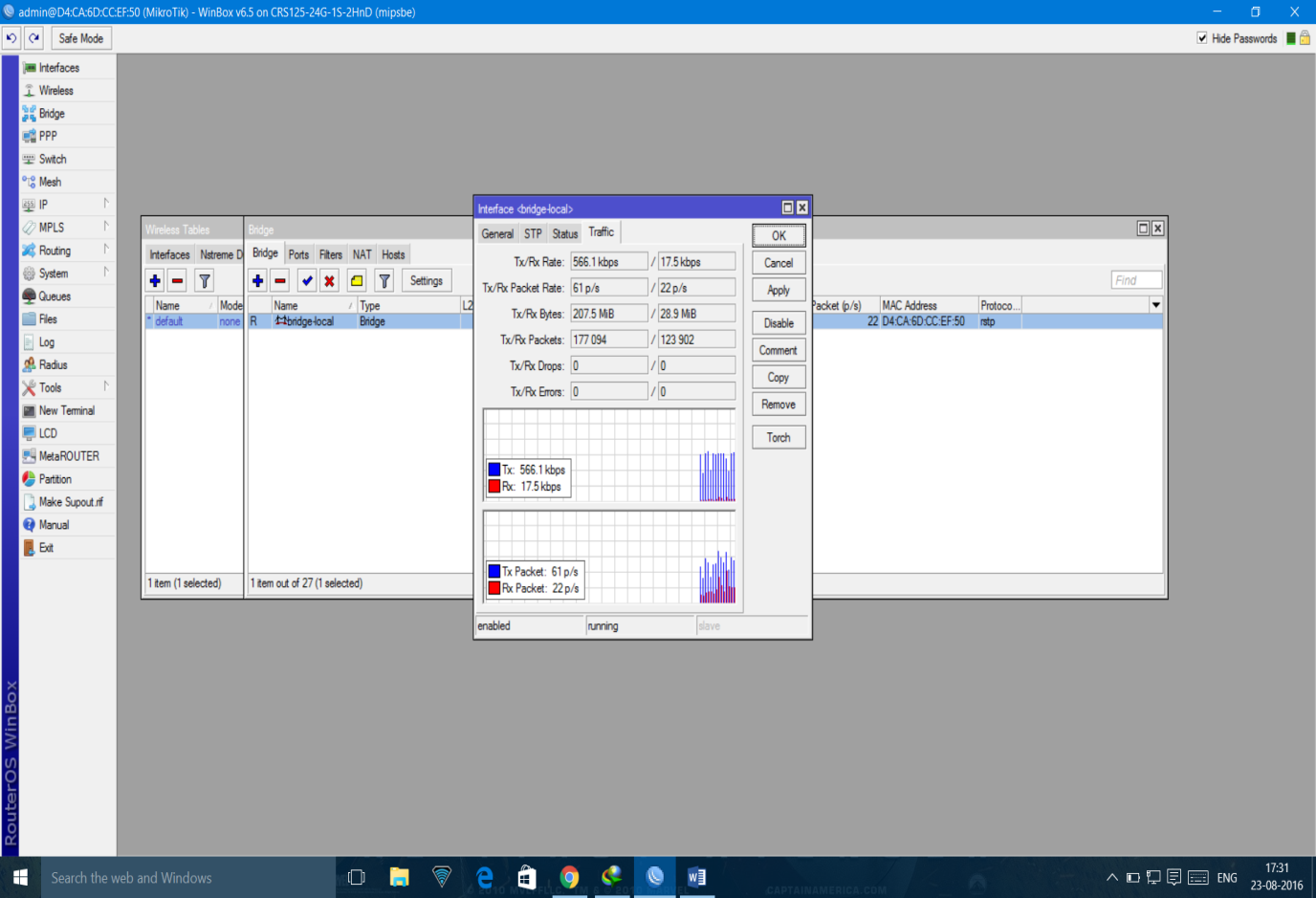
To connect to the router enter IP or MAC address of the router, specify username and password (if any) and click on **Connect** button. You can also enter the port number after the IP address, separating them with a colon, like this 192.168.88.1:9999. The port can be changed in RouterOS **services** menu.

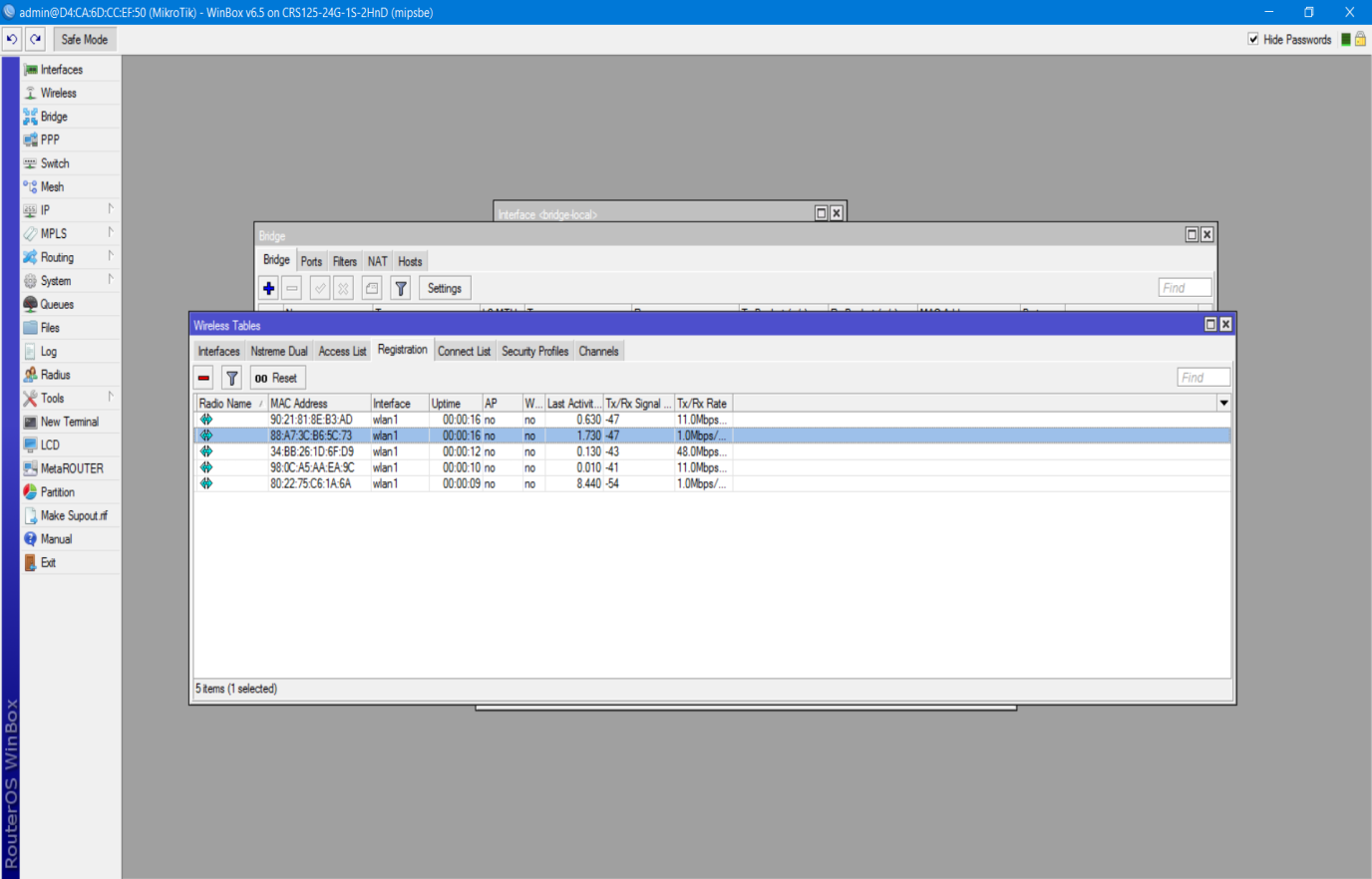










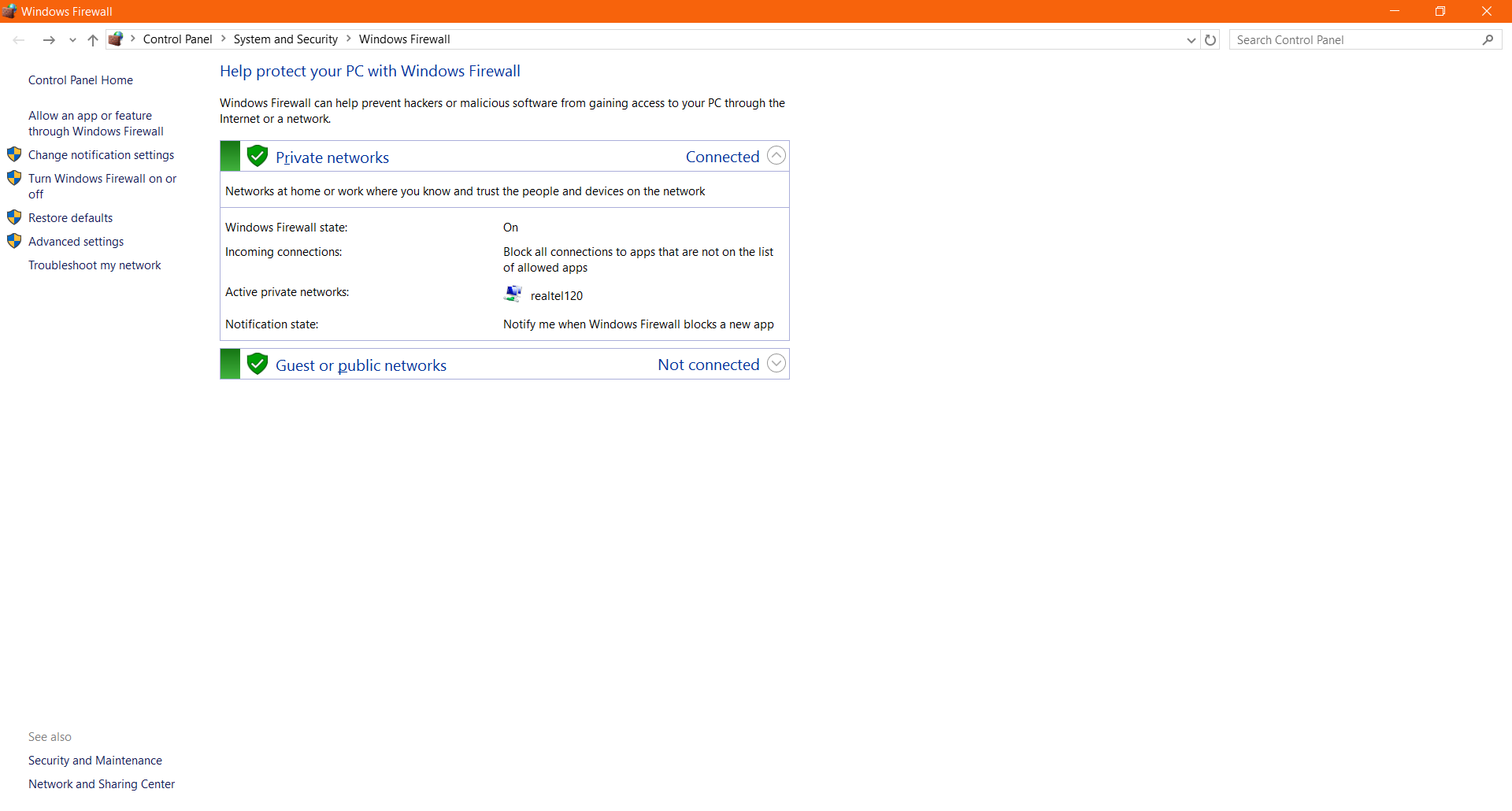


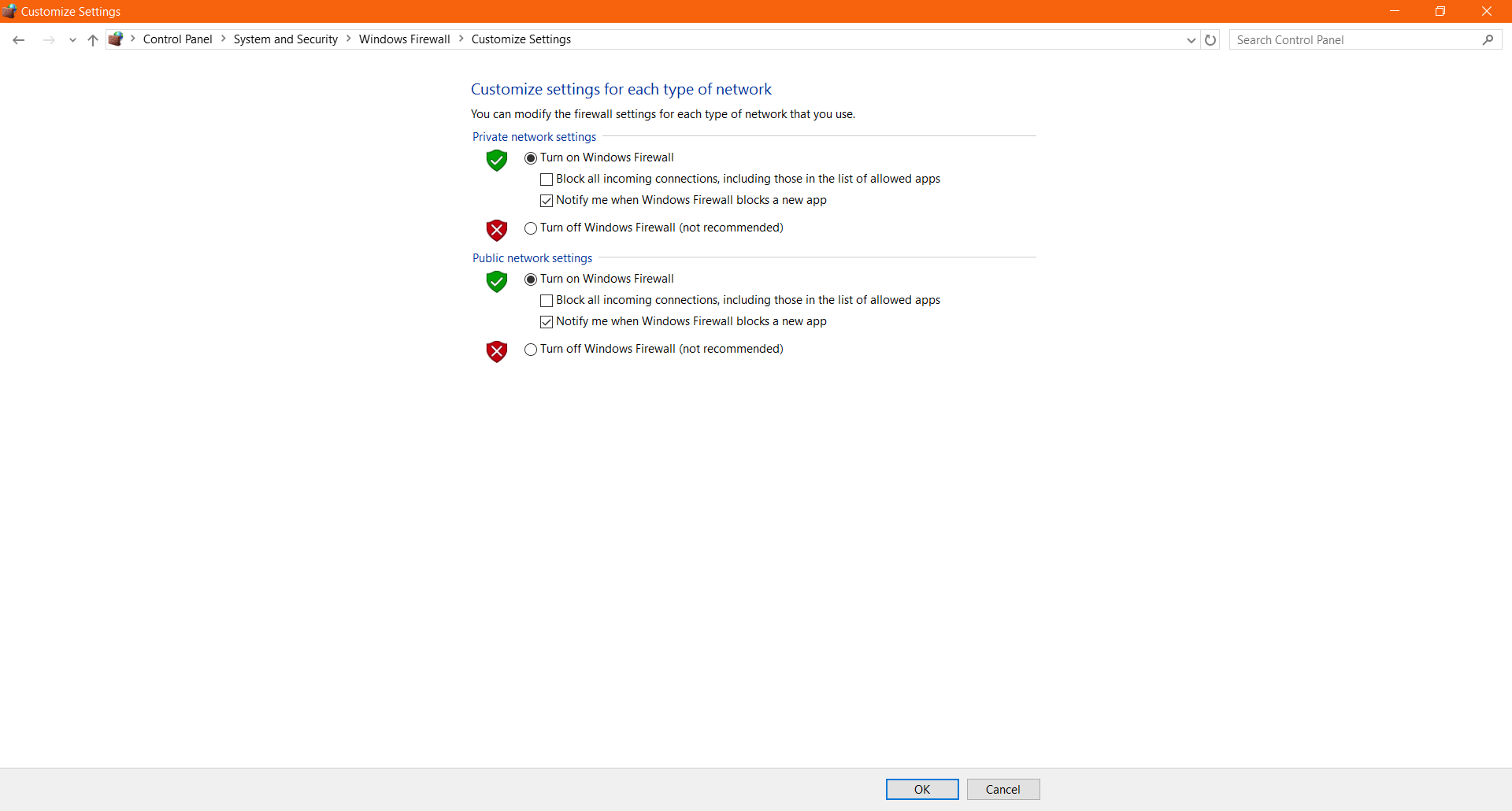
# EXPERIMENT – 2

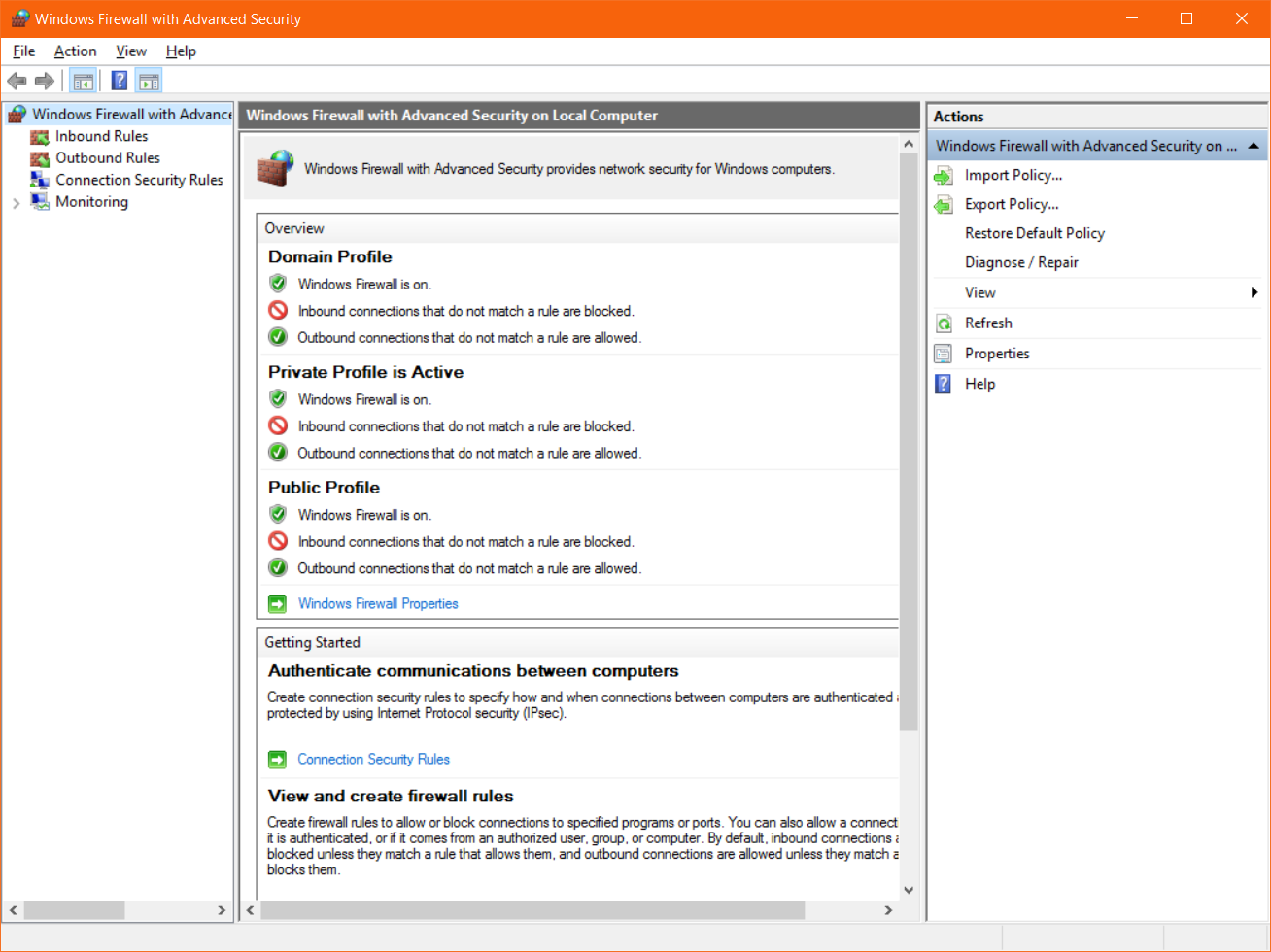
**Aim:** To implement Windows Firewall.

To enable Windows Firewall and configure the default behaviour on Windows 7, Windows Vista, Windows Server 2008, or Windows Server 2008 R2

1. [Open the Group Policy Management Console to Windows Firewall with Advanced Security](https://technet.microsoft.com/en-us/library/cc947816(v=ws.10).aspx).
2. In the details pane, in the **Overview** section, click **Windows Firewall Properties**.
3. For each network location type (Domain, Private, Public), perform the following steps.
   1. Click the tab that corresponds to the network location type.
   2. Change **Firewall state** to **On (recommended)**.
   3. Change **Inbound connections** to **Block (default)**.
   4. Change **Outbound connections** to **Allow (default)**.







**EXPERIMENT – 3**

AIM : Write a program to use monoalphabetic cipher

importjava.util.Scanner;

public class Mono

{

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 p1[] = 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))

{

p1[i] = p[j];

break;

}

}

}

return (new String(p1));

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the message: ");

String en = doEncryption(sc.next().toLowerCase());

System.out.println("Encrypted message: " + en);

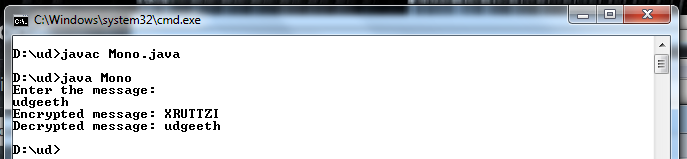
System.out.println("Decrypted message: " + doDecryption(en));

sc.close();

}

}

OUTPUT::



**EXPERIMENT – 4**

AIM: Write a program to use PlayFair Cipher

importjava.util.Scanner;

public class PlayfairCipherEncryption

{

private String KeyWord = new String();

private String Key = new String();

private char matrix\_arr[][] = new char[5][5];

public void setKey(String k)

{

String K\_adjust = new String();

boolean flag = false;

K\_adjust = K\_adjust + k.charAt(0);

for (int i = 1; i <k.length(); i++)

{

for (int j = 0; j <K\_adjust.length(); j++)

{

if (k.charAt(i) == K\_adjust.charAt(j))

{

flag = true;

}

}

if (flag == false)

K\_adjust = K\_adjust + k.charAt(i);

flag = false;

}

KeyWord = K\_adjust;

}

public void KeyGen()

{

boolean flag = true;

char current;

Key = KeyWord;

for (int i = 0; i < 26; i++)

{

current = (char) (i + 97);

if (current == 'j')

continue;

for (int j = 0; j <KeyWord.length(); j++)

{

if (current == KeyWord.charAt(j))

{

flag = false;

break;

}

}

if (flag)

Key = Key + current;

flag = true;

}

System.out.println(Key);

matrix();

}

private void matrix()

{

int counter = 0;

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

matrix\_arr[i][j] = Key.charAt(counter);

System.out.print(matrix\_arr[i][j] + " ");

counter++;

}

System.out.println();

}

}

private String format(String old\_text)

{

int i = 0;

intlen = 0;

String text = new String();

len = old\_text.length();

for (int tmp = 0; tmp<len; tmp++)

{

if (old\_text.charAt(tmp) == 'j')

{

text = text + 'i';

}

else

text = text + old\_text.charAt(tmp);

}

len = text.length();

for (i = 0; i <len; i = i + 2)

{

if (text.charAt(i + 1) == text.charAt(i))

{

text = text.substring(0, i + 1) + 'x' + text.substring(i + 1);

}

}

return text;

}

private String[] Divid2Pairs(String new\_string)

{

String Original = format(new\_string);

int size = Original.length();

if (size % 2 != 0)

{

size++;

Original = Original + 'x';

}

String x[] = new String[size / 2];

int counter = 0;

for (int i = 0; i < size / 2; i++)

{

x[i] = Original.substring(counter, counter + 2);

counter = counter + 2;

}

return x;

}

public int[] GetDiminsions(char letter)

{

int[] key = new int[2];

if (letter == 'j')

letter = 'i';

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{

if (matrix\_arr[i][j] == letter)

{

key[0] = i;

key[1] = j;

break;

}

}

}

return key;

}

public String encryptMessage(String Source)

{

String src\_arr[] = Divid2Pairs(Source);

String Code = new String();

char one;

char two;

int part1[] = new int[2];

int part2[] = new int[2];

for (int i = 0; i <src\_arr.length; i++)

{

one = src\_arr[i].charAt(0);

two = src\_arr[i].charAt(1);

part1 = GetDiminsions(one);

part2 = GetDiminsions(two);

if (part1[0] == part2[0])

{

if (part1[1] < 4)

part1[1]++;

else

part1[1] = 0;

if (part2[1] < 4)

part2[1]++;

else

part2[1] = 0;

}

else if (part1[1] == part2[1])

{

if (part1[0] < 4)

part1[0]++;

else

part1[0] = 0;

if (part2[0] < 4)

part2[0]++;

else

part2[0] = 0;

}

else

{

int temp = part1[1];

part1[1] = part2[1];

part2[1] = temp;

}

Code = Code + matrix\_arr[part1[0]][part1[1]]

+ matrix\_arr[part2[0]][part2[1]];

}

return Code;

}

public static void main(String[] args)

{

PlayfairCipherEncryption x = new PlayfairCipherEncryption();

Scanner sc = new Scanner(System.in);

System.out.println("Enter a keyword:");

String keyword = sc.next();

x.setKey(keyword);

x.KeyGen();

System.out .println("Enter word to encrypt: (Make sure length of message is even)");

String key\_input = sc.next();

if (key\_input.length() % 2 == 0)

{

System.out.println("Encryption: " + x.encryptMessage(key\_input));

}

else

{

System.out.println("Message length should be even");

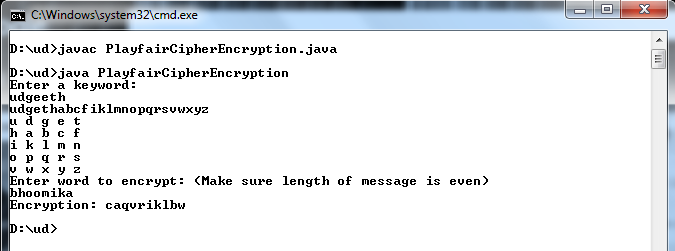
}

sc.close();

}

}

Output:



**EXPERIMENT - 5**

**AIM**: Write a program to use Custom Cipher to encrypt message

public class Custom

{

public static void main(String arg[])

{

String keys = "12";

String codes = "hello";

StringBuilder result = new StringBuilder();

char[] codeList = codes.toCharArray();

char[] keyList = keys.toCharArray();

intmaxCount = keys.length();

System.out.println("The length is "+maxCount);

int i = 0;

for (Character code : codeList) {

int key = Character.getNumericValue(keyList[i]);

if(key % 2 == 0)

{

int res = code+key;

result.append((char)res);

}

else

{

int res = code-key;

result.append((char)res);

}

i++;

if(i==maxCount)

{

i = 0;

}

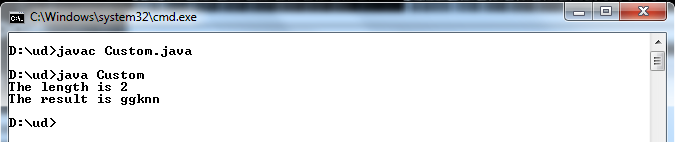
}

System.out.println("The result is "+result.toString());

}

}

Output:



**EXPERIMENT - 6**

Aim – Implement RSA Algorithm.

#include<iostream>

#include<math.h>

#include<string.h>

#include<stdlib.h>

using namespace std;

long int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100], en[100], i;

charmsg[100];

int prime(long int);

voidce();

long int cd(long int);

void encrypt();

void decrypt();

int prime(long int pr)

{

int i;

j = sqrt(pr);

for (i = 2; i <= j; i++)

{

if (pr % i == 0)

return 0;

}

return 1;

}

int main()

{

cout<< "\nENTER FIRST PRIME NUMBER\n";

cin>> p;

flag = prime(p);

if (flag == 0)

{

cout<< "\nWRONG INPUT\n";

exit(1);

}

cout<< "\nENTER ANOTHER PRIME NUMBER\n";

cin>> q;

flag = prime(q);

if (flag == 0 || p == q)

{

cout<< "\nWRONG INPUT\n";

exit(1);

}

cout<< "\nENTER MESSAGE\n";

fflush(stdin);

cin>>msg;

for (i = 0; msg[i] != NULL; i++)

m[i] = msg[i];

n = p \* q;

t = (p - 1) \* (q - 1);

ce();

cout<< "\nPOSSIBLE VALUES OF e AND d ARE\n";

for (i = 0; i < j - 1; i++)

cout<< e[i] << "\t" << d[i] << "\n";

encrypt();

decrypt();

return 0;

}

voidce()

{

int k;

k = 0;

for (i = 2; i < t; i++)

{

if (t % i == 0)

continue;

flag = prime(i);

if (flag == 1 && i != p && i != q)

{

e[k] = i;

flag = cd(e[k]);

if (flag > 0)

{

d[k] = flag;

k++;

}

if (k == 99)

break;

}

}

}

long int cd(long int x)

{

long int k = 1;

while (1)

{

k = k + t;

if (k % x == 0)

return (k / x);

}

}

void encrypt()

{

long int pt, ct, key = e[0], k, len;

i = 0;

len = strlen(msg);

while (i != len)

{

pt = m[i];

pt = pt - 96;

k = 1;

for (j = 0; j < key; j++)

{

k = k \* pt;

k = k % n;

}

temp[i] = k;

ct = k + 96;

en[i] = ct;

i++;

}

en[i] = -1;

cout<< "\nTHE ENCRYPTED MESSAGE IS\n";

for (i = 0; en[i] != -1; i++)

printf("%c", en[i]);

}

void decrypt()

{

long int pt, ct, key = d[0], k;

i = 0;

while (en[i] != -1)

{

ct = temp[i];

k = 1;

for (j = 0; j < key; j++)

{

k = k \* ct;

k = k % n;

}

pt = k + 96;

m[i] = pt;

i++;

}

m[i] = -1;

cout<< "\nTHE DECRYPTED MESSAGE IS\n";

for (i = 0; m[i] != -1; i++)

printf("%c", m[i]);

}

ENTER FIRST PRIME NUMBER

47

ENTER ANOTHER PRIME NUMBER

53

ENTER MESSAGE

Dharmendra

POSSIBLE VALUES OF e AND d ARE

3 1595

5 957

7 1367

11 435

17 985

19 1259

29 165

31 463

37 1293

41 2217

43 1947

59 1419

61 549

67 2035

71 1415

73 1409

79 1847

83 2075

89 2177

97 1233

101 1421

103 2183

THE ENCRYPTED MESSAGE IS

x`a???]??a

THE DECRYPTED MESSAGE IS

Dharmendra

**EXPERIMENT – 7**

Aim – Implement DES Algorithm.

#include<iostream>  
#include<stdio.h>  
void main()  
{  
int a[20],b[20],c[20],i,j,k,l,m,n,x1;  
int p,q,r[20],s[20],d[20],y[20],e[10],z=0;  
clrscr();  
printf(“\n Enter the plain Text number:\n”);  
scanf(“%d”,&n);  
printf(“\n Enter the key number \n”);  
scanf(“%d”,&k);  
printf(“\n Enter the bit stream \n”);  
for(i=0;i<n;i++)  
{  
scanf("%d",&a[i]);  
}  
p=n\2;  
x1=p-k;  
for(i=0;i<=x1;i++)  
{  
c[i]=0;  
z++;  
}  
z=z-1;

printf("\n Enter the Key bit Stream\n");  
for(i=0;i<k;i++)  
{  
scanf("%d",&c[z++]);  
}  
for(i=0;i<2;i++)  
printf("%d",e[i]);  
printf("Left hand data\n");  
for(i=0;i<p;i++)  
{

s[i]=a[i];  
printf("%d",a[i]);  
}  
printf("\n Right hand data \n");  
for(i=p;i<n;i++)  
{  
b[i]=a[i];  
printf("%d",a[i]);  
}  
q=p;  
for(j=0,l=p;j<z,l<n;j++,l++)  
{  
if(b[l]==1&&c[j]==1)  
d[j]=0;  
else if(b[l]==1&&c[j]==0)  
d[j]=l;  
else  
d[j]=0;  
}  
printf("\n First XOR");  
for(i=0;i<p;i++)  
{  
printf("%d",d[i]);  
}  
for(j=0,l=0;j<p,j++;j++,l++)  
{  
if(s[l]=1&&d[j]==1)  
r[j]=0;  
else if(s[l]=1&&d[j]==0)  
r[j]=l;  
else if(s[l]==0&&d[j]==1)  
r[j]=0;  
}  
printf("\n Second XOR\n ");  
for(i=0;i<p;i++)  
printf("%d",r[j]);  
getch();  
}

**EXPERIMENT – 8**

Aim – Implement Diffie-Hellman Algorithm.

#include<stdio.h>

long int power(int a,intb,int mod)

{

longlong int t;

if(b==1)

return a;

t=power(a,b/2,mod);

if(b%2==0)

return (t\*t)%mod;

else

return (((t\*t)%mod)\*a)%mod;

}

longlong int calculateKey(int a,intx,int n)

{

return power(a,x,n);

}

int main()

{

intn,g,x,a,y,b;

// both the persons will be agreed upon the common n and g

printf("Enter the value of n and g : ");

scanf("%d%d",&n,&g);

// first person will choose the x

printf("Enter the value of x for the first person : ");

scanf("%d",&x); a=power(g,x,n);

// second person will choose the y

printf("Enter the value of y for the second person : ");

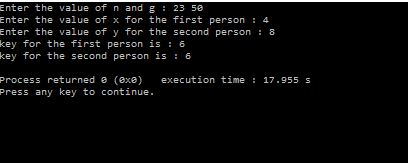
scanf("%d",&y); b=power(g,y,n);

printf("key for the first person is : %lld\n",power(b,x,n));

printf("key for the second person is : %lld\n",power(a,y,n));

return 0;

}



**EXPERIMENT – 9**

Aim – Study of Nessi2 Simulation tool based on parameters of IS.

NeSSi consists of three distinct components, the Graphical User Interface, the simulation backend and the result database. Each of these modules may be run on separate machines depending on the computational requirements; furthermore, this modular design facilitates network security researchers using NeSSi to easily exchange

#### Graphical User Interface

The graphical frontend of NeSSi allows the user to create and edit network topologies, attach runtime information, and schedule them for execution at the simulation backend. On the other hand, finished (or even currently executing, long-running) simulations can be retrieved from the database server and the corresponding simulation results are visualized in the GUI.

#### Simulation Backend

The actual simulation is performed on machine with hardware dedicated solely to this purpose, the simulation backend. At the DAI-Labor for example, the NeSSi simulation backend runs on a Sun XFire 4600 blade server (8 blades, 8 cores per blade). Once a session is submitted for execution from the GUI, the simulation backend parses the desired session parameters (which event types to log, how many runs to execute etc.), creates a corresponding simulation environment, sets up the database connection and schedules the simulation to run as soon as the necessary processing resources are available.

***Parallel Execution Model*.** Simulations in large-scale networks are very costly in terms of processing time and memory consumption. Therefore, NeSSi has been designed as a distributed simulation, allowing the subdivision of tasks to different computers and processes in a parallel-execution model.

***Discrete Event Simulation.*** NeSSi² is a discrete-event-based simulation tool, which allows to plan and to schedule time-based events such as network failures, attacks, etc.

#### Simulation Result Database Server

In NeSSi, we refer to a scenario where we generate traffic via pre-defined profiles on a single network over a certain amount of time as a *session*. The accurate reproduction of a session enables users to use different detection methods or various deployments of detection units for the same traffic data set. This allows the comparison of performance and detection efficiency of  
different security framework setups.

For these purposes, we use a ***distributed database*** in which the traffic generated during a session is stored. For each session, the agents log the traffic and detection data and send it to the database that occurs in a simulated scenario between a start and end time. The data types to be logged are specified by the user in the session parameters. The network model is saved in an XML file. This network file is stored and annotated with a version number based on its hash code in order to link a network uniquely to a session. Additionally, attack related events can be stored in the database for evaluation purposes.

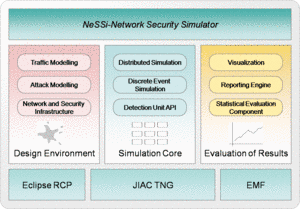
#### Standard Components and Plugin API

NeSSi² has been designed as a modularized application. Building on the Eclipse framework, it uses the inherent plugin mechanism to allow users to easily extend the functionality of NeSSi² and share it with other developers.

Often, security researchers have very specific demands regarding the protocols and features the simulation tool needs to offer. Naturally, NeSSi² provides a rich set of basic protocols and detection unit implementations; nevertheless, the special needs of various application areas (wireless networks, sensor networks, MPLS etc.) necessitates a plugin API to allow the user to adapt NeSSi² to his needs and add the functionality that is not provided by NeSSi out-of-the-box.

Hence, the NeSSi² extension API allows the creation of

* New device types with user-defined properties
* New protocols defining the behavior of the network at runtime
* Application definitions, allowing dynamic behavior to be defined, attached to a device or link, and scheduled for execution in the simulation



**EXPERIMENT – 10**

Aim – Implement VPN through Network Simulator Tool.

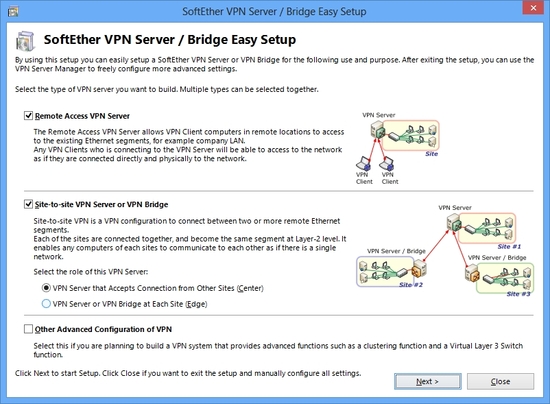
SoftEther VPN can construct distributed virtual Ethernet segment. If you can make some geologically distributed computers enable to communicate each other as if they are connected to the single Ethernet network, using SoftEther VPN is the easiest way.

First, set up a VPN Server. Next, set up VPN Clients on each member PCs. Finally start VPN connections on each VPN client. Then each clients can use any kinds of IP-based or Ethernet-based protocols via the VPN even if they are distributed around the world.

**Step 1. Set up SoftEther VPN Server**

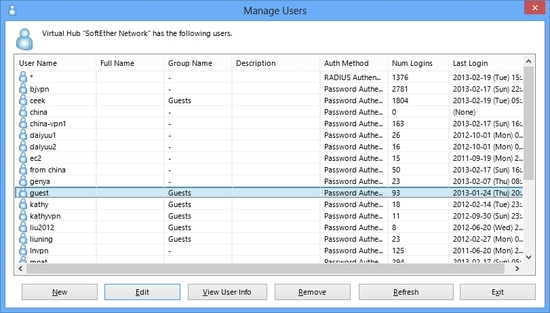
Designate a computer in the group as the VPN Server. [Set up SoftEther VPN Server](https://www.softether.org/4-docs/1-manual/7._Installing_SoftEther_VPN_Server) on that computer. It is very easy by

using Installer and Initial Setup Wizard based GUI.



**Step 2. Create Users**

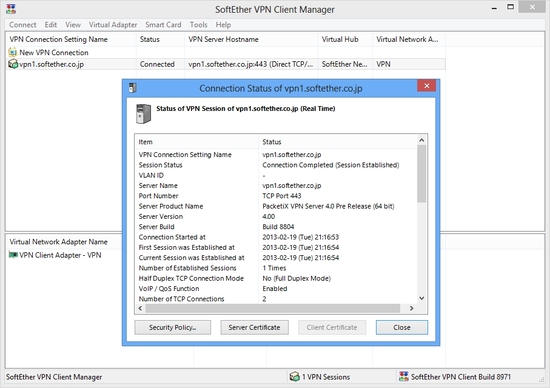
On the VPN Server [you can add several user objects](https://www.softether.org/4-docs/1-manual/2._SoftEther_VPN_Essential_Architecture/2.2_User_Authentication) on the [Virtual Hub](https://www.softether.org/4-docs/1-manual/3._SoftEther_VPN_Server_Manual/3.4_Virtual_Hub_Functions). Each user object has a password. After that, distribute pairs of username and password to each member of the VPN.



**Step 3. Set up VPN Client on Each Member's PC**

On each member's PC [**install SoftEther VPN Client.**](https://www.softether.org/4-docs/1-manual/8._Installing_SoftEther_VPN_Client) Enter the server address, username and password for each PC.

If a member of the VPN is Mac OS X, iPhone or Android, [**set up L2TP/IPsec VPN client**](https://www.softether.org/4-docs/2-howto/9.L2TPIPsec_Setup_Guide_for_SoftEther_VPN_Server) on each PC instead of SoftEther VPN. Another solution is to use OpenVPN Client on Mac OS X, iPhone or Android to connect to SoftEther VPN Server.

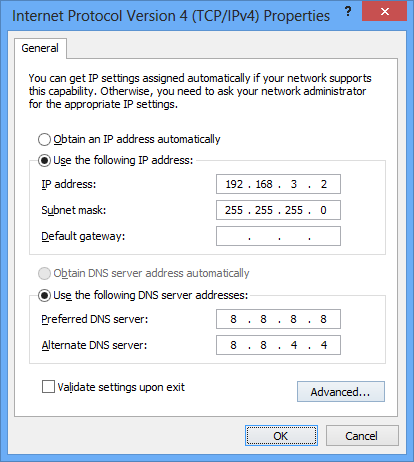


Step 4. Set up IP Addresses

The characteristics of SoftEther's virtual private network is exactly same to a physical Ethernet segment. So you should decide the IP addresses of every member PCs.

Like the physical Ethernet, the simplest way is to set up private IP addresses to each PC, for example 192.168.0.0/24. Make sure not to overlap to physical-using private IPs.

Another solution is to use DHCP server for automated IP address allocation. You can activate Virtual DHCP Server Function on the SoftEther VPN Server and it will distribute 192.168.30.0/24 by default.



Step 5. Communicate Like Physical Ethernet

Once every computers are connected to the Virtual Hub on SoftEther VPN Server, all computers and smart-phones can now communicate mutually as if they are all connected to the single Ethernet network. You can enjoy File Sharing protocols, Remote Printing applications, Remote Desktop applications, SQL Database applications and any other LAN-based applications despite the distances and differences of physical location.

