INFORMATION SECURITY LAB

ETCS - 451

Semester: 7th

Group:

Maharaja Agrasen Institute of Technology, PSP area,
Sector - 22, Rohini, New Delhi - 110085

(Affiliated to Guru Gobind Singh Indraprastha University, New Delhi)

Introduction to Information Security Lab

1.1 Objective

In the vast area of Information Technology, security is an important aspect as without this the entire systems would fall. The course helps in understating the basic concepts of the Information security from basics of wireless communication to advance VPN implementation using simulation tools.

1.2 Course outcomes

At the end of the course, a student will be able to:

C401.1: To understand the basics of Information security and common threats faced today.

C401.2: To know about basics principle of information security and various security challenges and authentication issues in Mobile devices.

C401.3: To understand advanced security issues of Internet including various protocols for security services and security acts to handle various forensics issues.

C401.4: To apply physical and bio-metric security on various economic and social frameworks

C401.5: To master fundamentals of private and public cryptography.

C401.6: To have knowledge of various security concepts and issues for network protection in VPN.

Department of Computer Science and Engineering

Rubrics for Lab Assessment

Rubrics		0	1	2	3
		Missing	Inadequate	Needs Improvement	Adequate
R1	Is able to identify the problem to be solved and define the objectives of the experiment.	No mention is made of the problem to be solved.	An attempt is made to identify the problem to be solved but it is described in a confusing manner, objectives are not relevant, objectives contain technical/ conceptual errors or objectives are not measurable.	The problem to be solved is described but there are minor omissions or vague details. Objectives are conceptually correct and measurable but may be incomplete in scope or have linguistic errors.	The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. They are written using correct technical terminology and are free from linguistic errors.
R2	Is able to design a reliable experiment that solves the problem.	The experiment does not solve the problem.	The experiment attempts to solve the problem but due to the nature of the design the data will not lead to a reliable solution.	The experiment attempts to solve the problem but due to the nature of the design there is a moderate chance the data will not lead to a reliable solution.	The experiment solves the problem and has a high likelihood of producing data that will lead to a reliable solution.
R3	Is able to communicate the details of an experimental procedure clearly and completely.	Diagrams are missing and/or experimental procedure is missing or extremely vague.	Diagrams are present but unclear and/or experimental procedure is present but important details are missing.	Diagrams and/or experimental procedure are present but with minor omissions or vague details.	Diagrams and/or experimental procedure are clear and complete.
R4	Is able to record and represent data in a meaningful way.	Data are either absent or incomprehensible.	Some important data are absent or incomprehensible.	All important data are present, but recorded in a way that requires some effort to comprehend.	All important data are present, organized and recorded clearly.
R5	Is able to make a judgment about the results of the experiment.	No discussion is presented about the results of the experiment .	A judgment is made about the results, but it is not reasonable or coherent.	An acceptable judgment is made about the result, but the reasoning is flawed or incomplete.	An acceptable judgment is made about the result, with clear reasoning. The effects of assumptions and experimental uncertainties are considered.

EXPERIMENT – 1

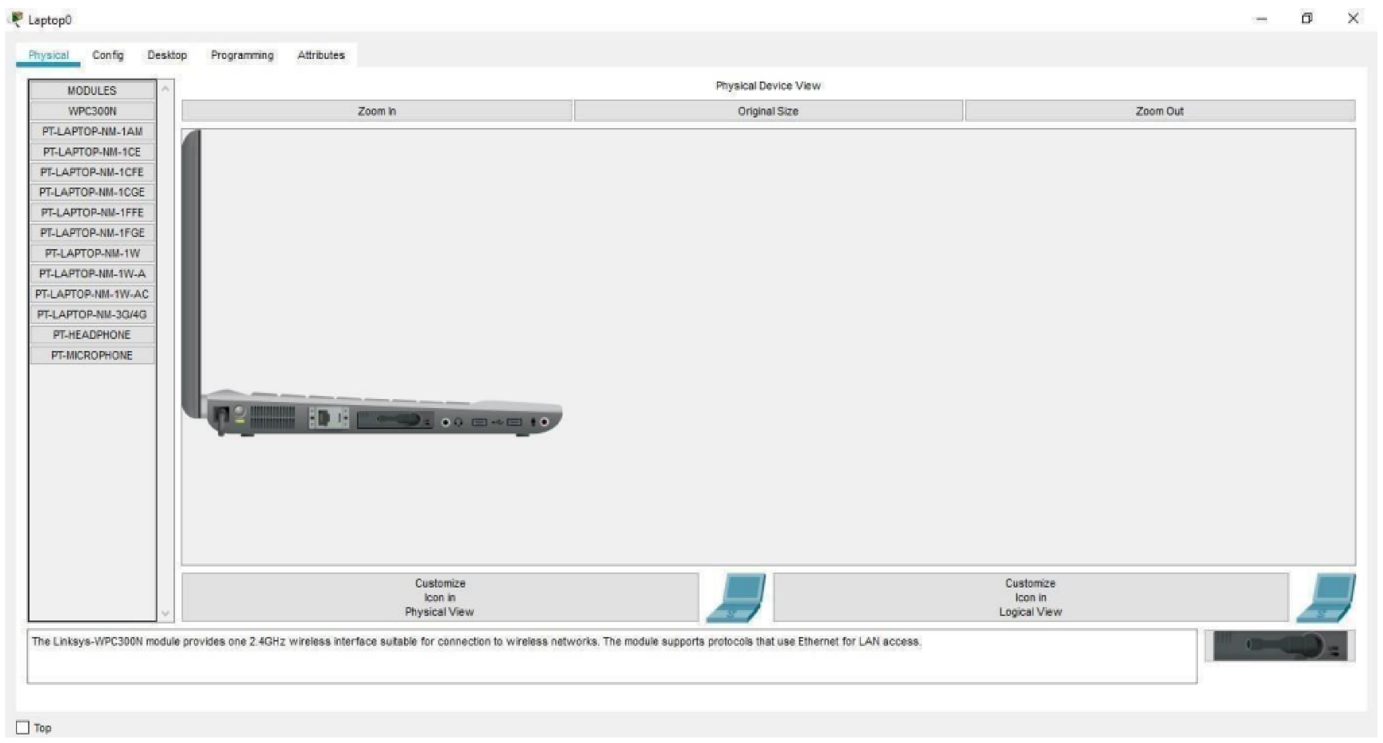
AIM: Make an experiment to implement WEP/WPA2 PSK, 802.1x EAP security protocol.

Requirement: System Loaded with Cisco Packet Tracer.

Theory:

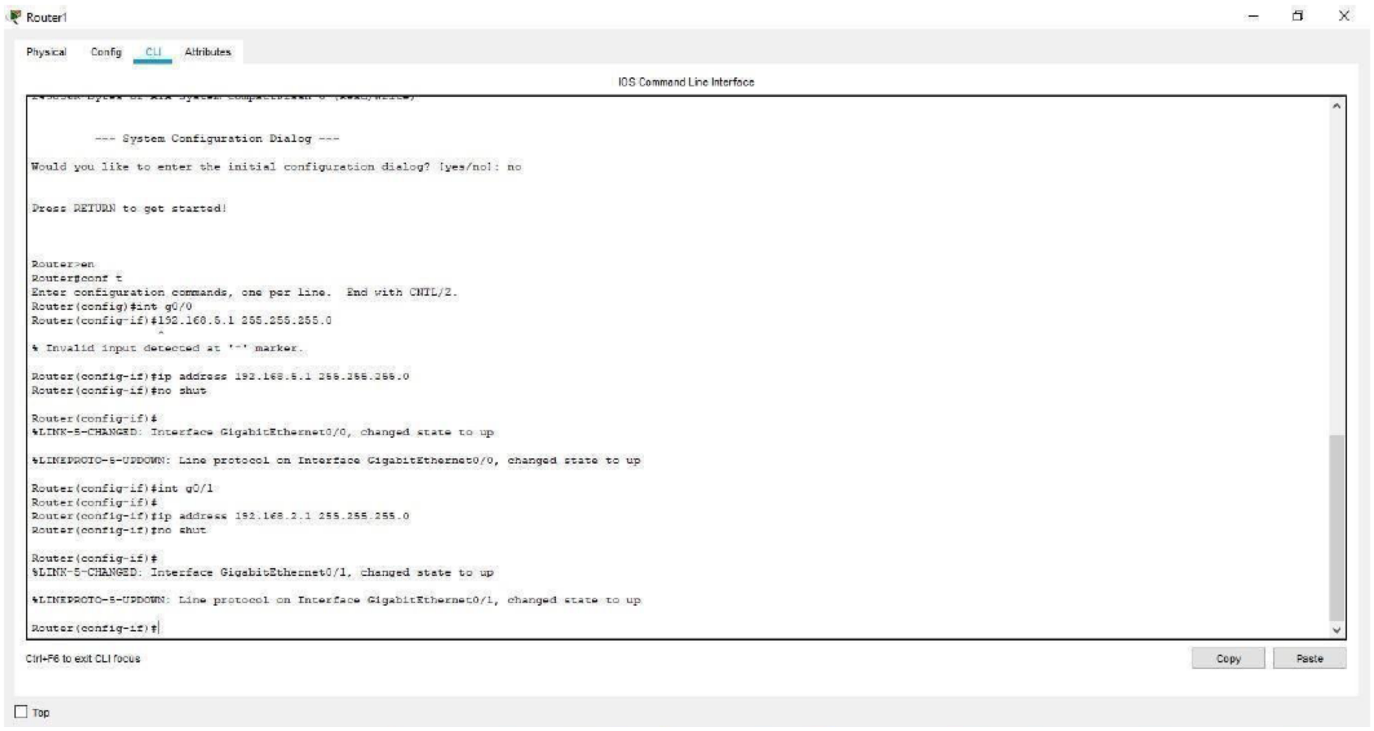
Cisco Packet Tracer is an innovative network simulation and visualization tool. This free software helps you to practice your network configuration and troubleshooting skills via your desktop computer or an Android or iOS based mobile device. Packet Tracer is available for both the Linux and Windows desktop environments.
With Packet Tracer you can choose to build a network from scratch, use a pre-built sample network, or complete classroom lab assignments. Packet Tracer allows you to easily explore how data traverses your network. Packet Tracer provides an easy way to design and build networks of varying sizes without expensive lab equipment. While this software is not a replacement for practicing on physical routers, switches, firewalls, and servers, it provides too many benefits

Output:



The Linksys-WPC300N module provides one 2.4GHz wireless interface suitable for connection to wireless networks. The module supports protocols that use Ethernet for LAN access.

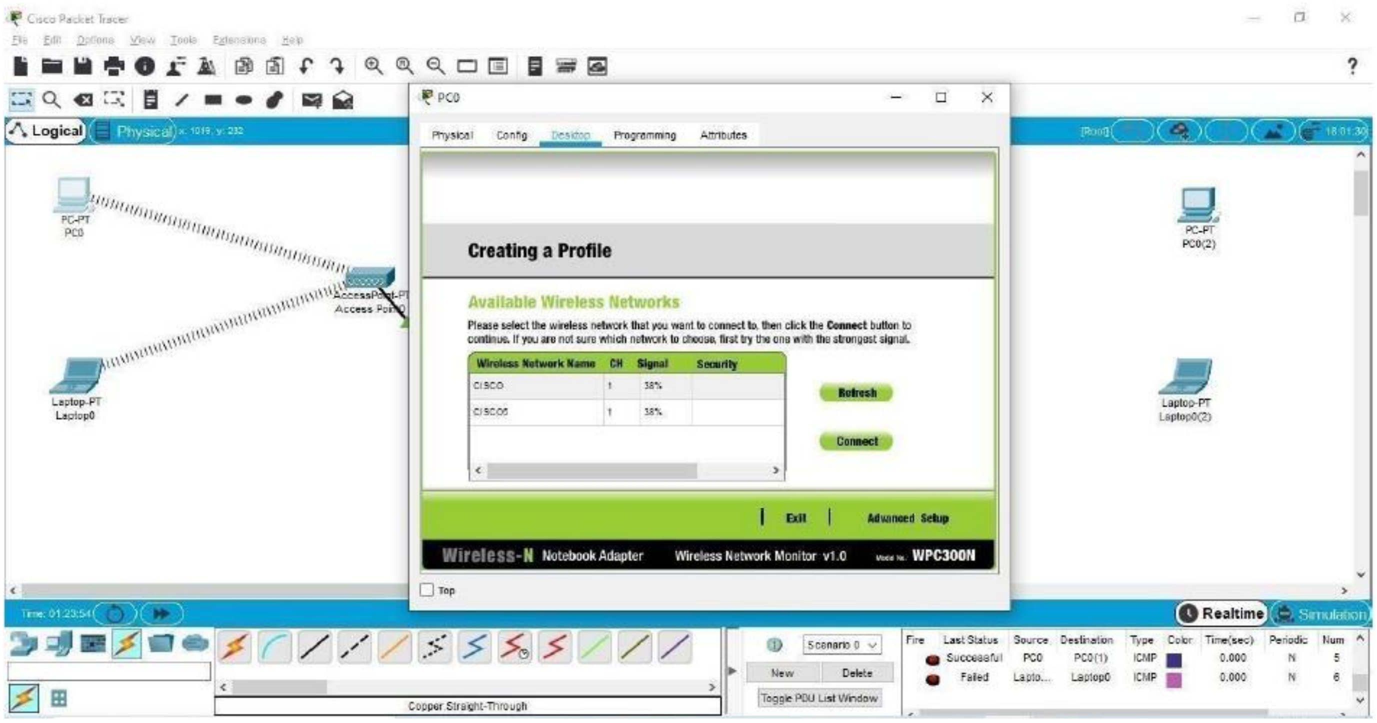
Physical Device View



```


        --- System Configuration Dialog ---

Would you like to enter the initial configuration dialog? [yes/no]: no


Press RETURN to get started!



Router>en
Router#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#int g0/0
Router(config-if)#192.168.5.1 255.255.255.0
                   ^
% Invalid input detected at '^' marker.

Router(config-if)#ip address 192.168.5.1 255.255.255.0
Router(config-if)#no shut

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0, changed state to up

Router(config-if)#int g0/1
Router(config-if)#
Router(config-if)#ip address 192.168.2.1 255.255.255.0
Router(config-if)#no shut

Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/1, changed state to up

Router(config-if)#
```

Ctrl+F6 to exit CLI focus

IOS Command Line Interface



Creating a Profile

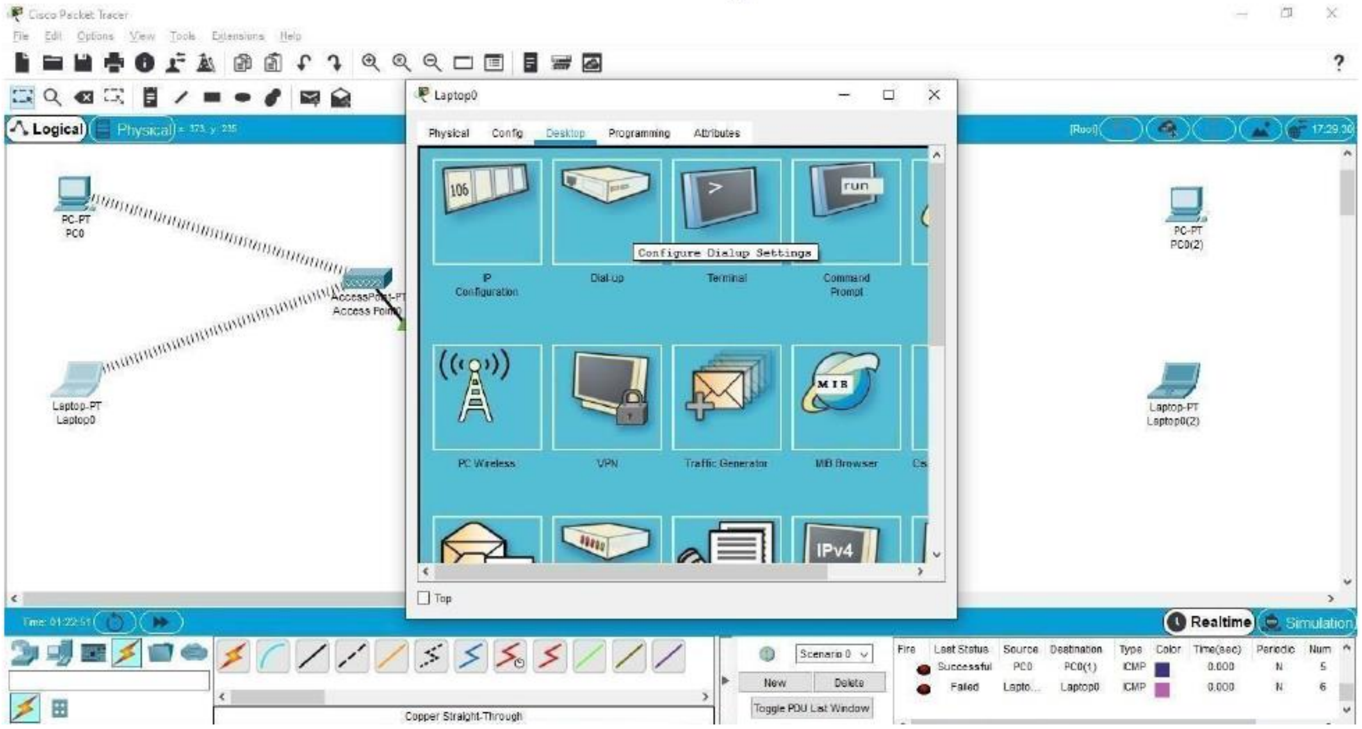
Available Wireless Networks

Please select the wireless network that you want to connect to, then click the **Connect** button to continue. If you are not sure which network to choose, first try the one with the strongest signal.

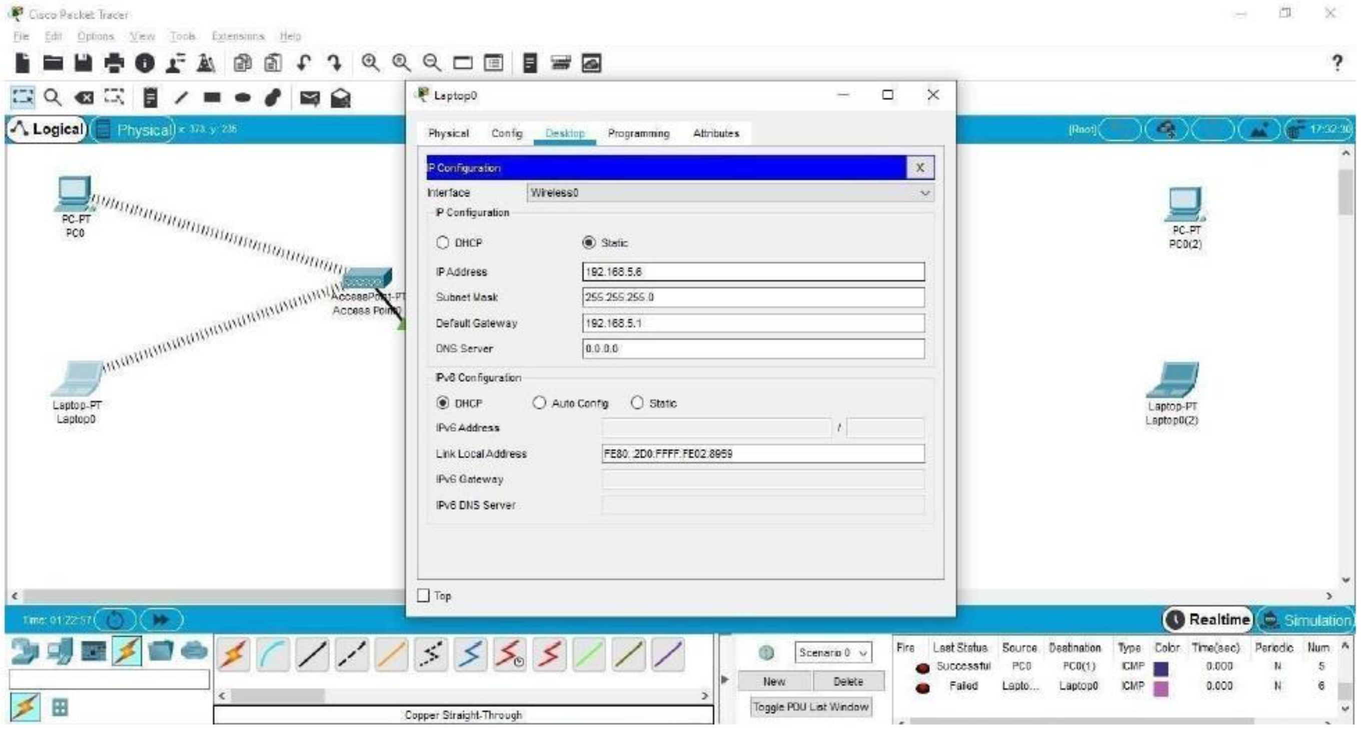
Wireless Network Name	CH	Signal	Security
CISCO	1	38%	
CISCO5	1	38%	

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
●	Successful	PC0	PC0(1)	ICMP	🟦	0.000	N	5
●	Failed	Lapto...	Laptop0	ICMP	🟪	0.000	N	6

Creating the Profile



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
●	Successful	PC0	PC0(1)	ICMP	🟦	0.000	N	5
●	Failed	Lapto...	Laptop0	ICMP	🟪	0.000	N	6



IP Configuration

Interface: Wireless0

IP Configuration

DHCP [✓ Static]

IP Address	192.168.5.6
Subnet Mask	255.255.255.0
Default Gateway	192.168.5.1
DNS Server	0.0.0.0

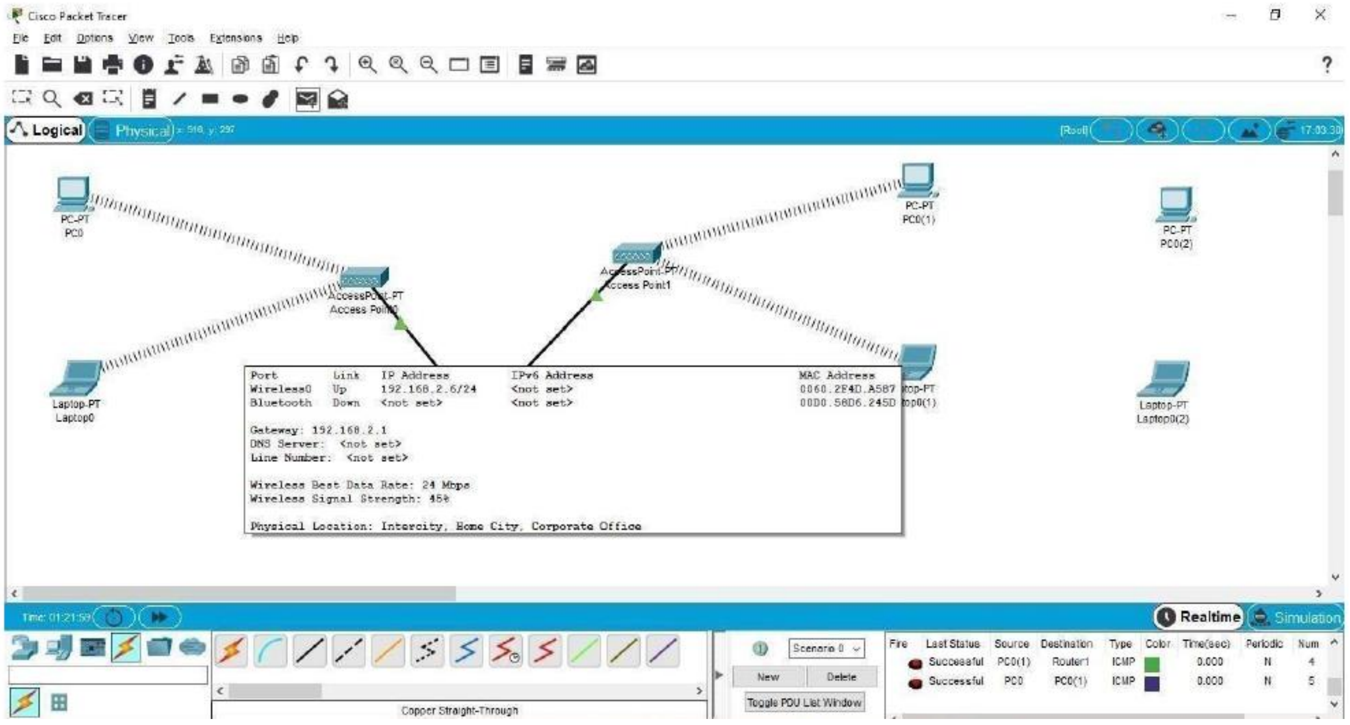
IPv6 Configuration

[✓ DHCP] Auto Config Static

IPv6 Address	/
Link Local Address	FE80::2D0:FFFF:FE02:8959
IPv6 Gateway	
IPv6 DNS Server	

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
●	Successful	PC0	PC0(1)	ICMP	🟦	0.000	N	5
●	Failed	Lapto...	Laptop0	ICMP	🟪	0.000	N	6

Making Configuration



```
Port       Link   IP Address        IPv6 Address                          MAC Address
Wireless0  Up     192.168.2.6/24    <not set>                             0060.2F4D.A587
Bluetooth  Down   <not set>         <not set>                             00D0.58D6.245D

Gateway: 192.168.2.1
DNS Server:  <not set>
Line Number:  <not set>

Wireless Best Data Rate: 24 Mbps
Wireless Signal Strength: 45%

Physical Location: Intercity, Home City, Corporate Office
```

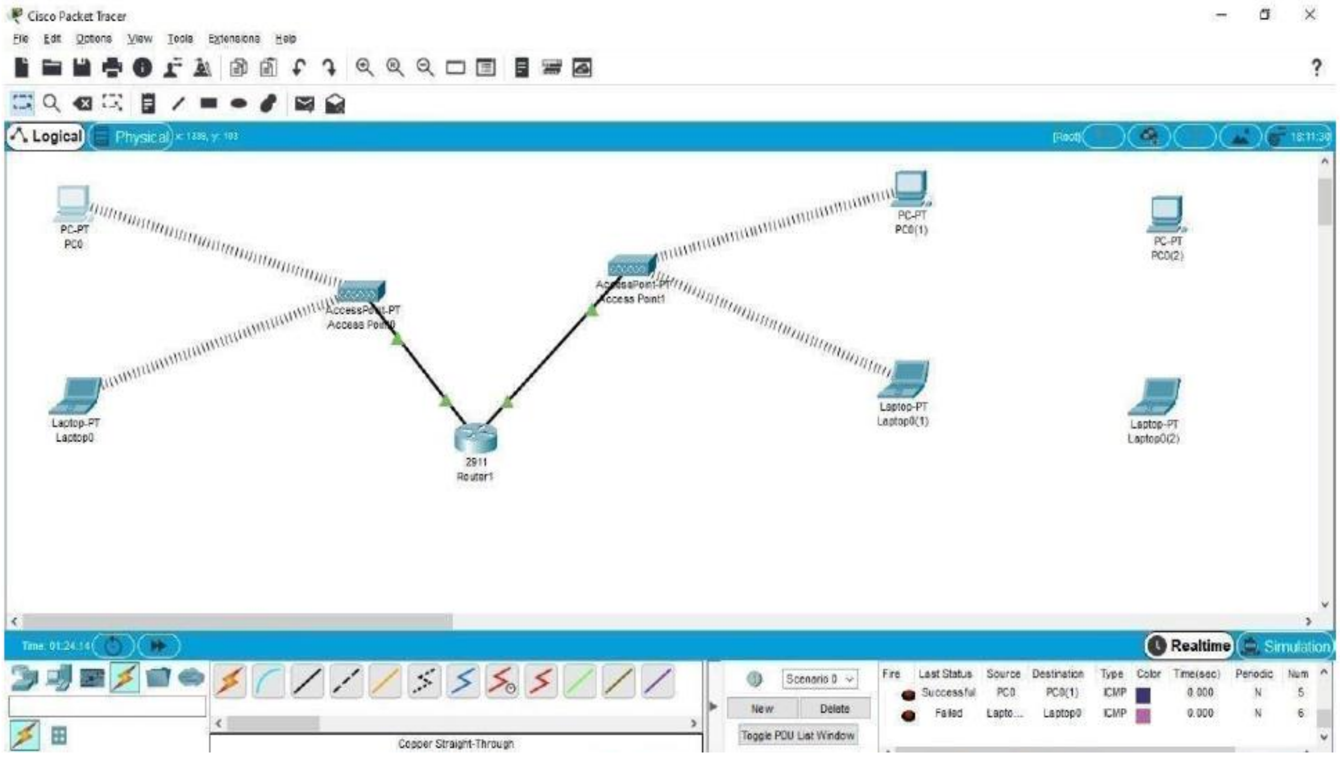
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
●	Successful	PC0(1)	Router1	ICMP	🟩	0.000	N	4
●	Successful	PC0	PC0(1)	ICMP	🟦	0.000	N	5

<u>Setting Up the WEP Network</u>

WEP (Wired Equivalent Privacy):

- It is a Security algo for wireless network.
- It is designed to provide Wireless Local Area Network(WLAN) with a level of security and privacy comparable to what is usually expected of a wired LAN.
- To provide data confidentiality comparable to that of a traditional wired network.
- A 64-bit WEP key is usually entered as a string of 10 hexadecimal (base 16) characters (0–9 and AF). Each character represents 4 bits, 10 digits of 4 bits each gives 40 bits; adding the 24-bit IV produces the complete 64- bit WEP key (4 bits × 10 + 24 bits IV = 64 bits of WEP key).

WEP is a security algorithm for IEEE 802.11 wireless networks, ratified in 1997, its intention was to provide data confidentiality comparable to that of a traditional wired network. WEP, recognizable by its key of 10 or 26 hexadecimal digits (40 or 104 bits), was at one time widely in use and was often the first security choice presented to users by router configuration tools.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num
●	Successful	PC0	PC0(1)	ICMP	🟦	0.000	N	5
●	Failed	Lapto...	Laptop0	ICMP	🟪	0.000	N	6

<u>Setting Up The WPA2 Network</u>

WPA stands for Wi-Fi Protected Access is a security standard for users of computing devices equipped with wireless internet connections. WPA was developed by the Wi-Fi Alliance to provide more sophisticated data encryption and better user authentication than Wired Equivalent Privacy (WEP), the original Wi-Fi security standard. The new standard, which was ratified by the IEEE in 2004 as 802.11i , was designed to be backward-compatible with WEP to encourage quick, easy adoption. Network security professionals were able to support WPA on many WEP-based devices with a simple firmware update.

<u>Result</u>: The working of WEP and WPA2 was successfully implemented and understood.

VIVA – VOCE

1. **Do WIFI Certificates Replace Wireless Security Protocols Like Wpa2?**

⇨ No. WIFI certificates are only used to encrypt data during the signup process. They are not used to encrypt data that is passed while an end-user is browsing the Internet.

2. **If My Wireless Network Doesn't Have a lot of Traffic, Is It Okay to Use WEP Because the Ivs Required to Crack the WEP Key Won't Be Generated?**

⇨ No. Automated tools are available that allow attackers to capture an ARP packet and reinject it to the access point very rapidly. This generates a significant amount of traffic and allows the attacker to capture enough unique initialization vectors to quickly crack the key.

3. **What is the difference between Active and Passive WLAN detection?**

⇨ Active WLAN detection requires that the SSID be broadcast in the beacon frame. Passive WLAN detection listens to all traffic in range of the device and determines what WLANs are in range.

4. **How many types of extensible Authentication Protocols (eaps) are supported by Wpa/wpa2 and What Are They?**

⇨ There are six fully supported EAP types for WPA/WPA2: EAP-TLS; EAP-TLS/MSCHAPv2; PEAPv0/EAP-MSCHAPv2; PEAPv1/EAP-GTC; EAP-SIM; and EAP-LEAP.

5. **What is the primary difference between 802.11g and 802.11a?**

⇨ 802.11g operates in the 2.4 GHz frequency range, as do 802.11b and 802.11i, whereas 802.11a operates in the 5 GHz frequency range.

6. **What is the difference between the hostap drivers and The Wlan-ng Drivers for linux?**

⇨ Both of these drivers work with a variety of cards; however, only the HostAP drivers allow you to place your card in monitor mode.

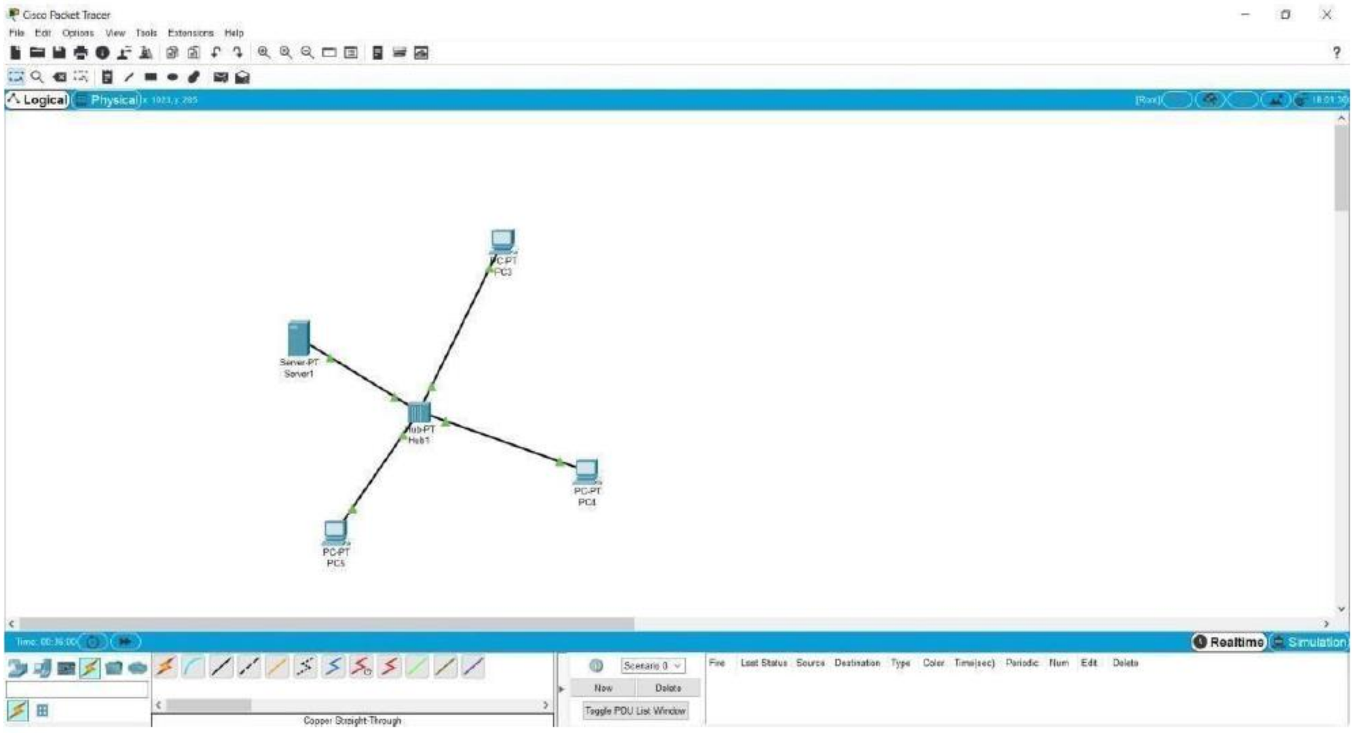
EXPERIMENT - 2

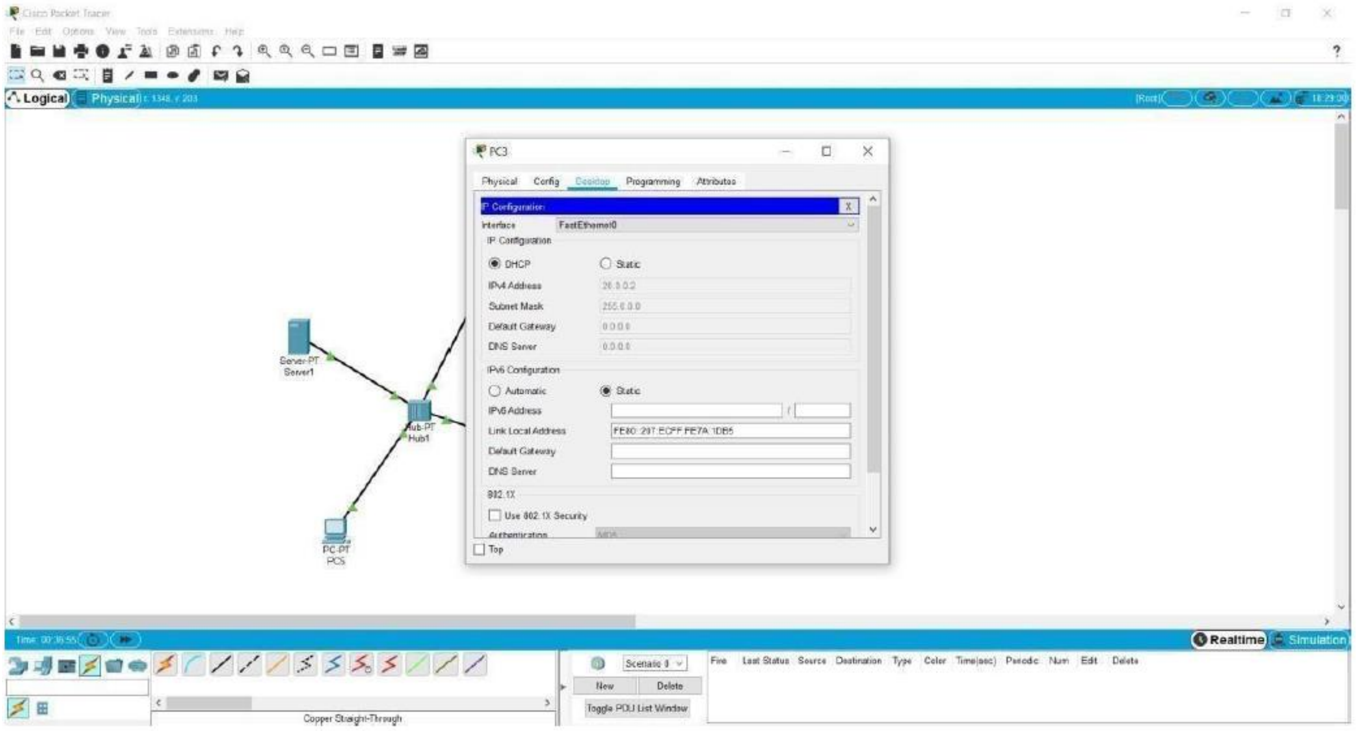
AIM: Implement a firewall on Cisco Packet Tracer.

Requirement: System Loaded with Cisco Packet Tracer.

Theory:
A firewall is a network security device that monitors incoming and outgoing network traffic and permits or blocks data packets based on a set of security rules. Its purpose is to establish a barrier between your internal network and incoming traffic from external sources (such as the internet) in order to block malicious traffic like viruses and hackers.
Firewalls carefully analyze incoming traffic based on pre-established rules and filter traffic coming from unsecured or suspicious sources to prevent attacks. Firewalls guard traffic at a computer's entry point, called ports, which is where information is exchanged with external devices. For example, - Source address 172.18.1.1 is allowed to reach destination 172.18.2.1 over port 22."

Output:





PC3

Physical | Config | Desktop | Programming | Attributes

IP Configuration

Interface: FastEthernet0

IP Configuration

(● DHCP) ○ Static

IPv4 Address	20.0.0.2
Subnet Mask	255.0.0.0
Default Gateway	0.0.0.0
DNS Server	0.0.0.0

IPv6 Configuration

○ Automatic (● Static)

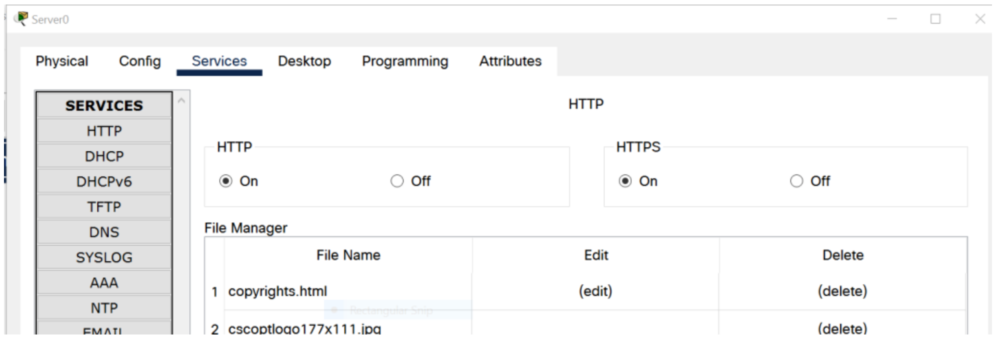
IPv6 Address	/
Link Local Address	FE80::207:ECFF:FE7A:1DB5
Default Gateway	
DNS Server	

802.1X

☐ Use 802.1X Security

Authentication MD5

☐ Top



Server0

Physical | Config | Services | Desktop | Programming | Attributes

SERVICES: HTTP, DHCP, DHCPv6, TFTP, DNS, SYSLOG, AAA, NTP, EMAIL

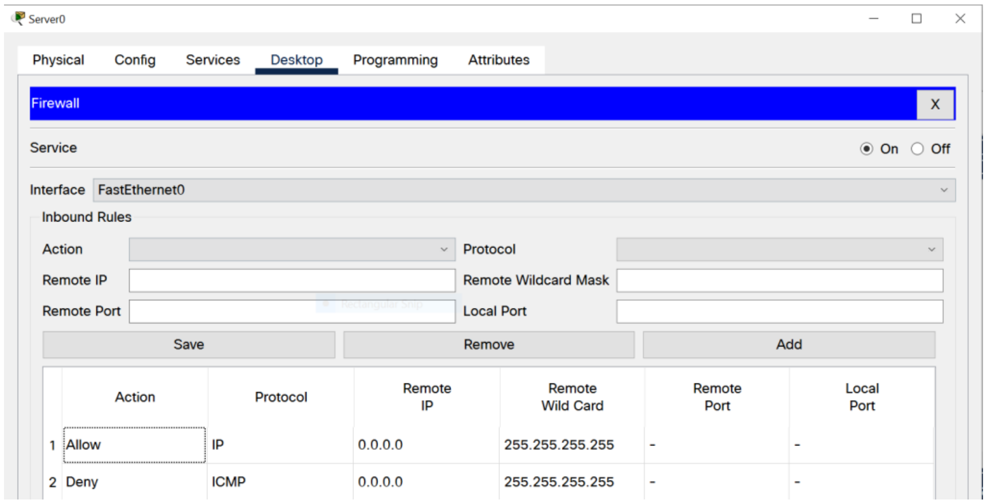
HTTP

HTTP: (● On) ○ Off

HTTPS: (● On) ○ Off

File Manager

	File Name	Edit	Delete
1	copyrights.html	(edit)	(delete)
2	cscoptlogo177x111.jpg		(delete)



Server0

Physical | Config | Services | Desktop | Programming | Attributes

Firewall

Service (● On) ○ Off

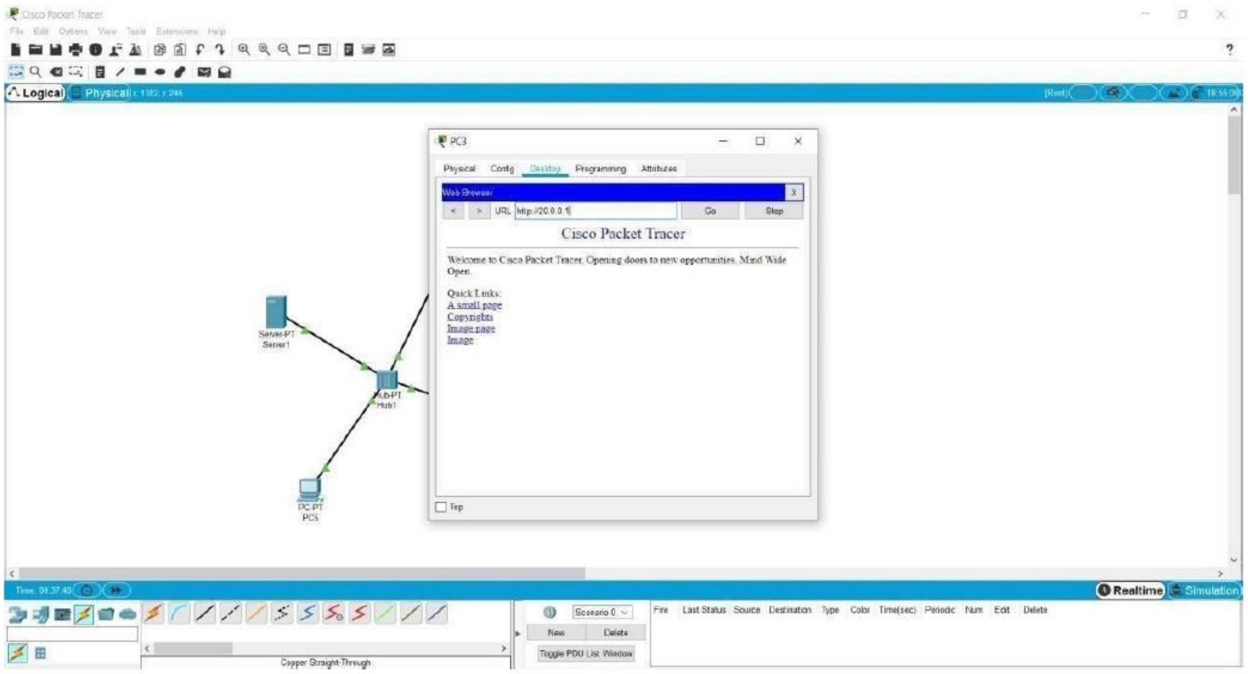
Interface: FastEthernet0

Inbound Rules

Action		Protocol	
Remote IP		Remote Wildcard Mask	
Remote Port		Local Port	

Save | Remove | Add

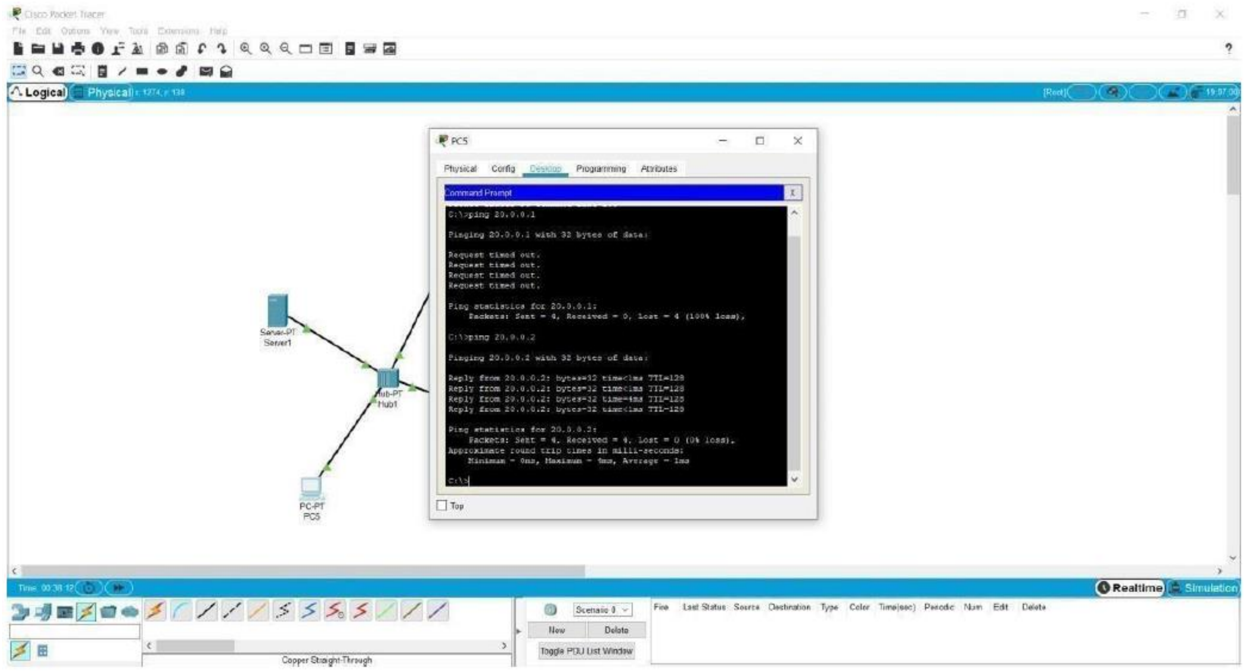
	Action	Protocol	Remote IP	Remote Wild Card	Remote Port	Local Port
1	Allow	IP	0.0.0.0	255.255.255.255	-	-
2	Deny	ICMP	0.0.0.0	255.255.255.255	-	-



Cisco Packet Tracer

Welcome to Cisco Packet Tracer. Opening doors to new opportunities. Mind Wide Open.

Quick Links:
A small page
Copyrights
Image page
Image



```
C:\>ping 20.0.0.1

Pinging 20.0.0.1 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 20.0.0.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\>ping 20.0.0.2

Pinging 20.0.0.2 with 32 bytes of data:

Reply from 20.0.0.2: bytes=32 time<1ms TTL=128
Reply from 20.0.0.2: bytes=32 time<1ms TTL=128
Reply from 20.0.0.2: bytes=32 time=4ms TTL=128
Reply from 20.0.0.2: bytes=32 time<1ms TTL=128

Ping statistics for 20.0.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 4ms, Average = 1ms

C:\>
```

Positive Response In Browser

The positive response on 3rd PC as it was configured.

Result: Working of a Firewall is Understood and Implemented.

VIVA – VOCE

1. What is a network firewall?

⇨ A firewall is a system or group of systems that enforces an access control policy between two networks. The actual means by which this is accomplished varies widely, but in principle, the firewall can be thought of as a pair of mechanisms: one which exists to block traffic, and the other permitting traffic.

2. What is synchronization and why is it important?

⇨ With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchronization, it is possible for one thread to modify a shared object while another thread is in the process of using or updating that object's value. This often leads to significant errors.

3. What are the critical resources in a firewall?

⇨ 1. Service Critical Resource
   2. Email
   3. Disk I/O
   4. Netnews Disk I/O
   5. Web Host

4. What are some common attacks, and how can I protect my system against them?

⇨ Each site is a little different from every other in terms of what attacks are likely to be used against it. Some recurring themes do arise, though.

5. Who determines the wireless standards?

⇨ IEEE develops and determines the wireless standards (802.11a, b, g, and so on). The WiFi Alliance, the group that owns the WiFi trademark, then certifies the interoperability of these devices.

EXPERIMENT – 3

AIM: Classical Cryptography: Using Classical Ciphers with pycipher.

Theory:

To learn some of the concepts and approaches used by current encryption algorithms (ciphers), it can be useful to first study how some of the original, simpler ciphers work (e.g. Caesar cipher, Play fair, Vigenere). With these simpler ciphers, often referred to as *classical ciphers*, it is quite easy to understand the algorithm and even perform encryption/decryption by hand. Although most are trivial to break with computers today, the concepts they use are often applied in current day ciphers. To learn some of the concepts and approaches used by current encryption algorithms (ciphers), it can be useful to first study how some of the original, simpler ciphers work (e.g. Caesar cipher, Play fair, Vigenere). With these simpler ciphers, often referred to as *classical ciphers*, it is quite easy to understand the algorithm and even perform encryption/decryption by hand. Although most are trivial to break with computers today, the concepts they use are often applied in current day ciphers.

ADFGX Cipher: This cipher uses a keysquare as part of its key. The ADFGX Cipher has a key consisting of a 5x5 key square and a word e.g. 'GERMAN'. The key square consists of the letters A-Z with J omitted (25 characters total).
Parameters:

- Key: The keysquare, as a 25 character string
- Keyword: The keyword, any word or phrase will do

ADFGVX Cipher: This cipher uses a keysquare as part of its key. The ADFGVX Cipher has a key consisting of a 6x6 key square and a word. The key square consists of the letters A-Z and the numbers 0-9 (36 characters total).
Parameters:

- key: The keysquare, as a 36 character string
- keyword: The keyword, any word or phrase will do

Affine Cipher: The Affine Cipher has two components to the key, numbers a and b. This cipher encrypts a letter according to the following equation:
$c = (a*p + b)\%26$

where c is the ciphertext letter, p the plaintext letter. b is an integer 0-25, a is an integer that has an inverse (mod 26). Allowable values for a are: 1,3,5,7,9,11,15,17,19,21,23,25
Parameters:

- a: The multiplicative part of the key. Allowable values are: 1,3,5,7,9,11,15,17,19,21,23,25
- b: The additive part of the key. Allowable values are integers 0-25

Caesar Cipher: The Caesar Cipher has a key consisting of an integer 1-25. This cipher encrypts a letter according to the following equation:

c = (p + key)%26

where c is the ciphertext letter, p the plaintext letter.
Parameters:

- Key: The additive key. Allowable values are integers 0-25.

Playfair Cipher: This cipher uses a keysquare as part of its key. The Playfair Cipher enciphers pairs of characters, the key consists of a keysquare 25 characters in length.
Parameters:

- Key: The keysquare, as a 25 character string.

Simple Substitution Cipher: The Simple Substitution Cipher has a key consisting of the letters A-Z jumbled up. e.g. 'AJPCZWRLFBDKOTYUQGENHXMIVS' This cipher encrypts a letter according to the following equation:

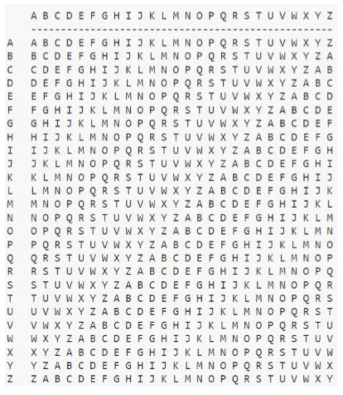
plaintext = ABCDEFGHIJKLMNOPQRSTUVWXYZ
ciphertext = AJPCZWRLFBDKOTYUQGENHXMIVS

To convert a plaintext letter into ciphertext, read along the plaintext row until the desired letter is found, then substitute it with the letter below it.
Parameters:

- Key: The key, a permutation of the 26 characters of the alphabet

Vigenere Cipher: The Vigenere Cipher has a key consisting of a word. His cipher encrypts a letter according to the Vigenere tableau.
Parameters:

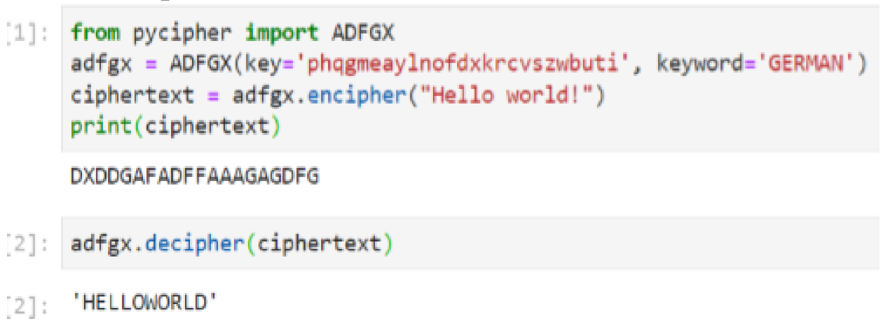
- Key: The keyword, any word or phrase will do. Must consist of alphabetical characters only, no punctuation of numbers



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

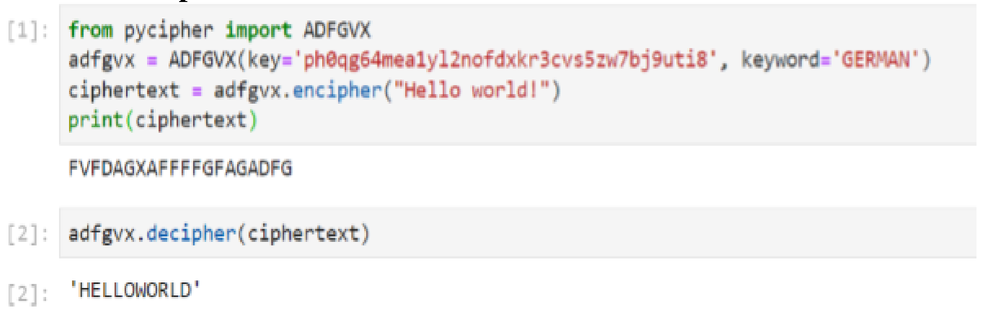
OUTPUT:

- **ADFGX Cipher:**



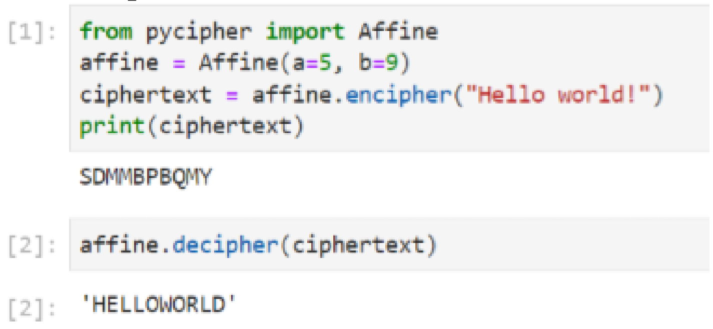
```
[1]: from pycipher import ADFGX
     adfgx = ADFGX(key='phqgmeaylnofdxkrcvszwbuti', keyword='GERMAN')
     ciphertext = adfgx.encipher("Hello world!")
     print(ciphertext)

     DXDDGAFADFFAAAGAGDFG

[2]: adfgx.decipher(ciphertext)

[2]: 'HELLOWORLD'
```

- **ADFGVX Cipher:**



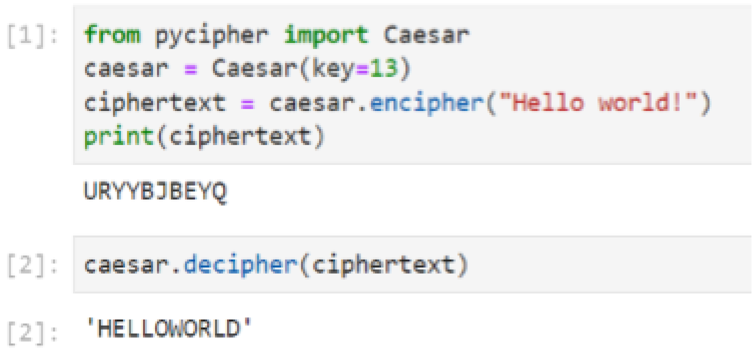
```
[1]: from pycipher import ADFGVX
     adfgvx = ADFGVX(key='ph0qg64mea1yl2nofdxkr3cvs5zw7bj9uti8', keyword='GERMAN')
     ciphertext = adfgvx.encipher("Hello world!")
     print(ciphertext)

     FVFDAGXAFFFFGFAGADFG

[2]: adfgvx.decipher(ciphertext)

[2]: 'HELLOWORLD'
```

- **Affine Cipher:**



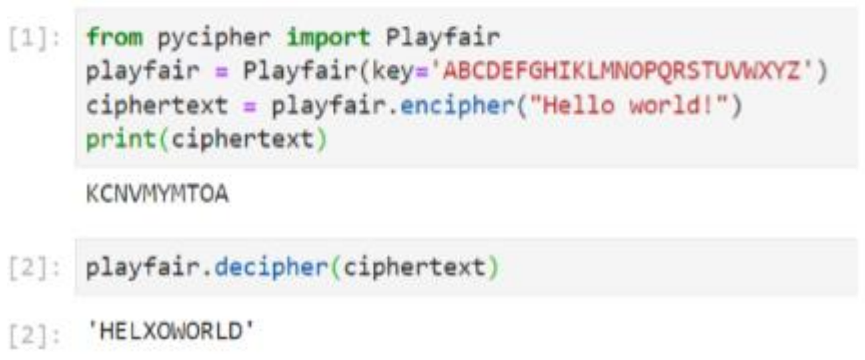
```
[1]: from pycipher import Affine
     affine = Affine(a=5, b=9)
     ciphertext = affine.encipher("Hello world!")
     print(ciphertext)

     SDMMBPBQMY

[2]: affine.decipher(ciphertext)

[2]: 'HELLOWORLD'
```

- **Caesar Cipher:**



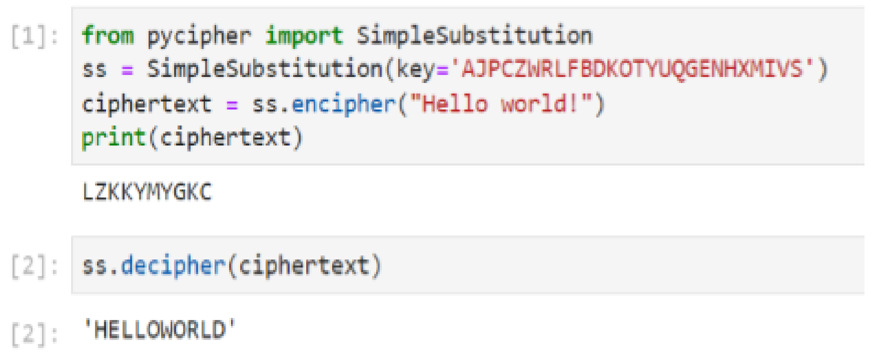
```
[1]: from pycipher import Caesar
     caesar = Caesar(key=13)
     ciphertext = caesar.encipher("Hello world!")
     print(ciphertext)

     URYYBJBEYQ

[2]: caesar.decipher(ciphertext)

[2]: 'HELLOWORLD'
```

- **Playfair Cipher:**



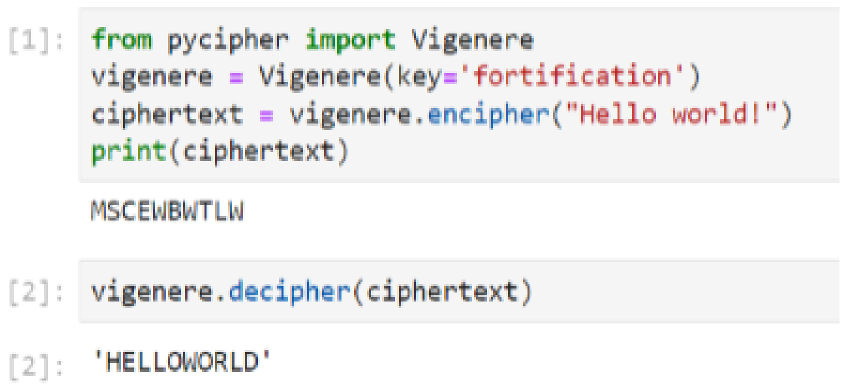
```
[1]: from pycipher import Playfair
     playfair = Playfair(key='ABCDEFGHIKLMNOPQRSTUVWXYZ')
     ciphertext = playfair.encipher("Hello world!")
     print(ciphertext)

     KCNVMYMTOA

[2]: playfair.decipher(ciphertext)

[2]: 'HELXOWORLD'
```

- **Simple Substitution Cipher:**



```
[1]: from pycipher import SimpleSubstitution
     ss = SimpleSubstitution(key='AJPCZWRLFBDKOTYUQGENHXMIVS')
     ciphertext = ss.encipher("Hello world!")
     print(ciphertext)

     LZKKYMYGKC

[2]: ss.decipher(ciphertext)

[2]: 'HELLOWORLD'
```

- **Vigenere Cipher:**



```
[1]: from pycipher import Vigenere
     vigenere = Vigenere(key='fortification')
     ciphertext = vigenere.encipher("Hello world!")
     print(ciphertext)

     MSCEWBWTLW

[2]: vigenere.decipher(ciphertext)

[2]: 'HELLOWORLD'
```

Result: Classical cryptography techniques are implemented and understood.

EXPERIMENT – 4

AIM: To implement RSA algorithm

Theory: RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and Private Key is kept private.
An example of asymmetric cryptography:

- A client (for example browser) sends its public key to the server and requests for some data.
- The server encrypts the data using client's public key and sends the encrypted data.
- Client receives this data and decrypts it.

The idea! The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024 bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Algorithm

Generating Public Key:

1. Select two prime numbers Suppose P = 53 and Q = 59.
2. Now First part of the Public key: n = P*Q = 3127.
3. We also need a small exponent say e:
4. We need to calculate Φ(n):
   a). Such that Φ(n) = (P-1) (Q-1)
   b). so, Φ(n) = 3016
5. Now calculate Private Key, d:
   a). d = (k*Φ(n) + 1) / e for some integer k
   b). For k = 2, value of d is 2011.

Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key (d = 2011)

Now we will encrypt "HI":

Convert letters to numbers: H = 8 and I = 9Thus, Encrypted Data c = 89e mod n.

Thus, our Encrypted Data comes out to be 1394

Now we will decrypt 1349:

Decrypted Data = cd mod n.

Thus, our Encrypted Data comes out to be 89 8 = H and I = 9 i.e. "HI".

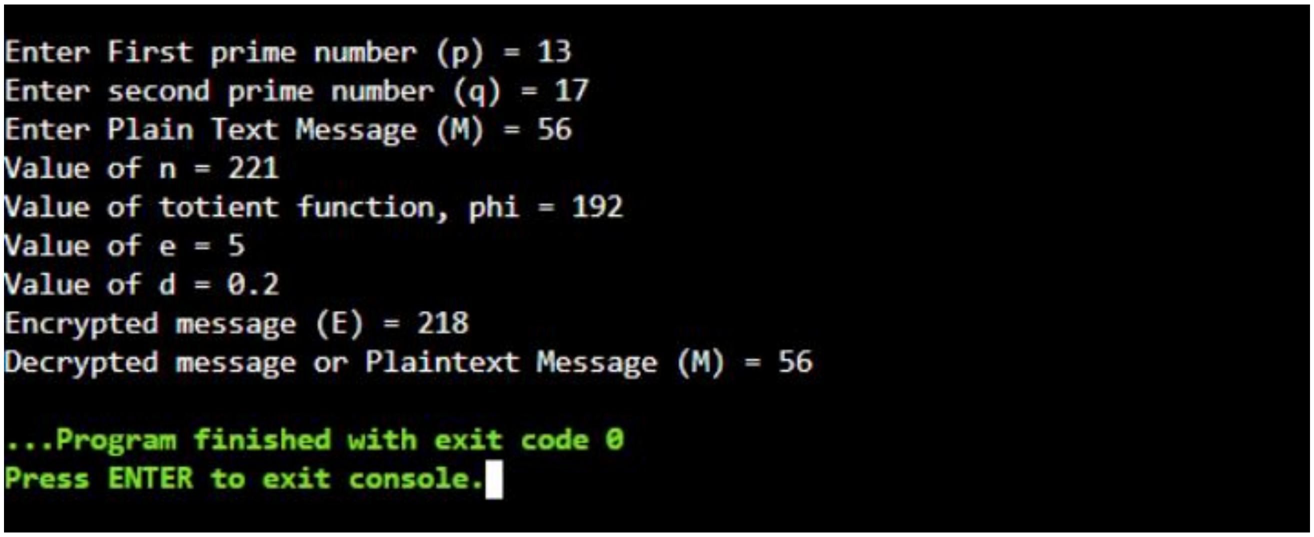
<u>Source Code</u>:

```
#include<bits/stdc++.h>
using namespace std;

int gcd(int a, int b) {
  while (true) {
    int remainder = a % b;
    if (remainder == 0) // b is a factor of a
      return b; // therefore, b is the gcd of a & b
    a = b;
    b = remainder;
  }
}

int main() {
  // 1. two prime numbers p & q are selected
  double p;
  cout << "Enter First prime number (p) = ";
  cin>>p;
  double q;
  cout << "Enter second prime number (q) = ";
  cin>>q;

  // 2. n is calculated
```

```
double n = p * q;

// 3. totient function value, phi, is calculated
double phi = (p - 1) * (q - 1);

// 4. value e is selected such that is > 1 and is coprime to phi we start searching from value 2
double e = 2;
while (e < phi) {
  double gcdOfEAndPhi = gcd(e, phi);
  // if gcd of e and phi is 1, then they are coprime
  if (gcdOfEAndPhi == 1) {
    break; // the desired value of e is found
  } else {
    e++; // keep the search for e on
  }
}

//5. value of d is calculated d should satisfy the equation d = (1 / e) * mod(phi)
double eInverse = 1 / e;
double d = fmod(eInverse, phi);

// plaintext message
double plainTextMessage;    // M = 97
cout << "Enter Plain Text Message (M) = ";
cin>>plainTextMessage;

double encryptedMessage = pow(plainTextMessage, e);
double decryptedMessage = pow(encryptedMessage, d);
```

```
  // 6. encrypting the plaintext message (M) to E
  encryptedMessage = fmod(encryptedMessage, n); // encryption complete

  // 7. decrypting the encrypted message (E) to get plaintext message (M)
  decryptedMessage = fmod(decryptedMessage, n); // decryption complete

  cout << "Value of n = " << p * q << endl;
  cout << "Value of totient function, phi = " << phi << endl;
  cout << "Value of e = " << e << endl;
  cout << "Value of d = " << d << endl;
  cout << "Encrypted message (E) = " << encryptedMessage << endl;
  cout << "Decrypted message or Plaintext Message (M) = " << decryptedMessage;
  return 0;
}
```

OUTPUT:



```
Enter First prime number (p) = 13
Enter second prime number (q) = 17
Enter Plain Text Message (M) = 56
Value of n = 221
Value of totient function, phi = 192
Value of e = 5
Value of d = 0.2
Encrypted message (E) = 218
Decrypted message or Plaintext Message (M) = 56

...Program finished with exit code 0
Press ENTER to exit console.
```

VIVA – VOCE

1. **How Fast Is RSA?**

⇨ RSA is considerably slow due to the calculation with large numbers. In particular the decryption where d is used in the exponent is slow. There are ways to speed it up by remembering p and q, but it is still slow in comparison to symmetric encryption algorithms. Asymptotic time complexity for an implementation of RSA using elementary algorithms, commonly used in practice, is O(n3) for private key use (signature generation, and decryption) and O(n2) for public-key use (signature verification, and encryption), where the public modulus N has n bits and public exponent e has a fixed size.

2. **What Would It Take to Break RSA?**

⇨ It would take a classical computer around 300 trillion years to break a RSA-2048 bit encryption key.

3. **Are Strong Primes Necessary In RSA?**

⇨ In the literature pertaining to RSA, it has often been suggested that in choosing a key pair, one should use so-called "strong" prime's p and q to generate the modulus n. Strong primes have certain properties that make the product n hard to factor by specific factoring methods. However, advances in factoring over the last ten years appear to have obviated the advantage of strong primes; the elliptic curve factoring algorithm is one such advance. The new factoring methods have as good a chance of success on strong primes as on "weak" primes. Therefore, choosing traditional "strong" primes alone does not significantly increase security. Choosing large enough primes is what matters.

4. **How large a modulus (key) should be used in RSA?**

⇨ The best size for an RSA modulus depends on one's security needs. The larger the modulus, the greater the security, but also the slower the RSA operations. One should choose a modulus length upon consideration, first, of one's security needs, such as the value of the protected data and how long it needs to be protected, and, second, of how powerful one's potential enemies are. Key sizes are now 768 bits for personal use, 1024 bits for corporate use, and 2048 bits for extremely valuable keys like the key pair of a certifying authority.

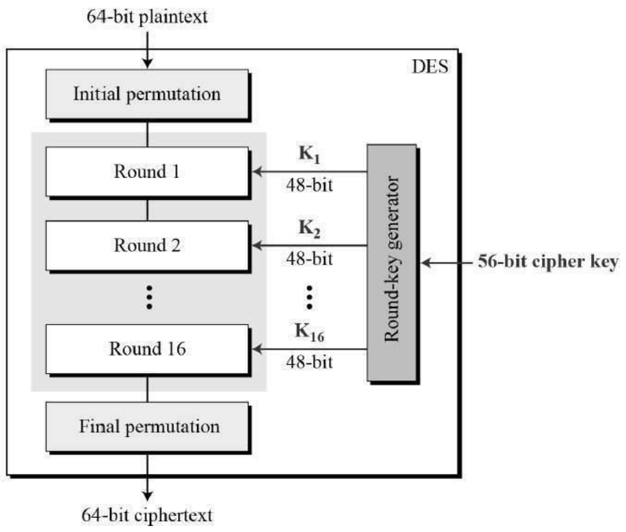
5. **How Large Should the Primes Be?**

⇨ As for whether collisions are possible- modern key sizes (depending on your desired security) range from 1024 to 4096, which means the prime numbers range from 512 to 2048 bits. That means that your prime numbers are on the order of 2^512: over 150 digits long.

EXPERIMENT – 5

AIM: To implement DES algorithm.

Theory: The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration –



Since DES is based on the Feistel Cipher, all that is required to specify DES is:

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

DES Analysis
The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

Avalanche effect − A small change in plaintext results in the very great change in the ciphertext.

Completeness − Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.
DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

Source Code:

```
#include <bits/stdc++.h>
using namespace std;
string hex2bin(string s){   // hexadecimal to binary conversion
        unordered_map<char, string> mp;
        mp['0'] = "0000";
        mp['1'] = "0001";
        mp['2'] = "0010";
        mp['3'] = "0011";
        mp['4'] = "0100";
        mp['5'] = "0101";
        mp['6'] = "0110";
        mp['7'] = "0111";
        mp['8'] = "1000";
        mp['9'] = "1001";
        mp['A'] = "1010";
        mp['B'] = "1011";
        mp['C'] = "1100";
        mp['D'] = "1101";
        mp['E'] = "1110";
        mp['F'] = "1111";

        string bin = "";
        for (int i = 0; i < s.size(); i++) {
                bin += mp[s[i]];
        }
        return bin;
}
```

```
string bin2hex(string s){     // binary to hexadecimal conversion
        unordered_map<string, string> mp;
        mp["0000"] = "0";
        mp["0001"] = "1";
        mp["0010"] = "2";
        mp["0011"] = "3";
        mp["0100"] = "4";
        mp["0101"] = "5";
        mp["0110"] = "6";
        mp["0111"] = "7";
        mp["1000"] = "8";
        mp["1001"] = "9";
        mp["1010"] = "A";
        mp["1011"] = "B";
        mp["1100"] = "C";
        mp["1101"] = "D";
        mp["1110"] = "E";
        mp["1111"] = "F";
        string hex = "";
        for (int i = 0; i < s.length(); i += 4) {
                string ch = "";
                ch += s[i];
                ch += s[i + 1];
                ch += s[i + 2];
                ch += s[i + 3];
                hex += mp[ch];
        }
        return hex;
```

```
}

string permute(string k, int* arr, int n){
	string per = "";
	for (int i = 0; i < n; i++) {
		per += k[arr[i] - 1];
	}
	return per;
}

string shift_left(string k, int shifts){
	string s = "";
	for (int i = 0; i < shifts; i++) {
		for (int j = 1; j < 28; j++) {
			s += k[j];
		}
		s += k[0];
		k = s;
		s = "";
	}
	return k;
}

string xor_(string a, string b){
	string ans = "";
	for (int i = 0; i < a.size(); i++) {
		if (a[i] == b[i])
			ans += "0";
```

```
                else
                        ans += "1";
        }
        return ans;
}

string encrypt(string pt, vector<string> rkb, vector<string> rk){     // Hexadecimal to binary
        pt = hex2bin(pt);

        // Initial Permutation Table
        int initial_perm[64] = { 58, 50, 42, 34, 26, 18, 10, 2, 60, 52, 44,36, 28, 20, 12, 4, 62, 54,
                                 46, 38, 30, 22,14, 6, 64, 56, 48, 40, 32, 24, 16, 8, 57,49, 41, 33, 25,
                                 17, 9, 1, 59, 51, 43, 35, 27, 19, 11, 3, 61, 53, 45, 37, 29, 21, 13,
                                  5, 63, 55, 47, 39, 31, 23, 15, 7 };
        // Initial Permutation
        pt = permute(pt, initial_perm, 64);
        cout << "After initial permutation: " << bin2hex(pt)<< endl;

        // Splitting
        string left = pt.substr(0, 32);
        string right = pt.substr(32, 32);
        cout << "After splitting: L0=" << bin2hex(left)<< " R0=" << bin2hex(right) << endl;

        // Expansion D-box Table
        int exp_d[48] = { 32, 1, 2, 3, 4, 5, 4, 5, 6, 7, 8, 9, 8, 9, 10, 11, 12, 13, 12, 13, 14, 15, 16, 17,
                          16, 17, 18, 19, 20, 21, 20, 21, 22, 23, 24, 25,
                           24, 25, 26, 27, 28, 29, 28, 29, 30, 31, 32, 1 };
```

```
// S-box Table

int s[8][4][16] = {

        { 14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5,

        9, 0, 7, 0, 15, 7, 4, 14, 2, 13, 1, 10, 6,

        12, 11, 9, 5, 3, 8, 4, 1, 14, 8, 13, 6, 2,

        11, 15, 12, 9, 7, 3, 10, 5, 0, 15, 12, 8, 2,

        4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13 },

        { 15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12,

        0, 5, 10, 3, 13, 4, 7, 15, 2, 8, 14, 12, 0,

        1, 10, 6, 9, 11, 5, 0, 14, 7, 11, 10, 4, 13,

        1, 5, 8, 12, 6, 9, 3, 2, 15, 13, 8, 10, 1,

        3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9 },


        { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12,

        7, 11, 4, 2, 8, 13, 7, 0, 9, 3, 4,

        6, 10, 2, 8, 5, 14, 12, 11, 15, 1, 13,

        6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12,

        5, 10, 14, 7, 1, 10, 13, 0, 6, 9, 8,

        7, 4, 15, 14, 3, 11, 5, 2, 12 },

        { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11,

        12, 4, 15, 13, 8, 11, 5, 6, 15, 0, 3, 4, 7,

        2, 12, 1, 10, 14, 9, 10, 6, 9, 0, 12, 11, 7,

        13, 15, 1, 3, 14, 5, 2, 8, 4, 3, 15, 0, 6,

        10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 },

        { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13,

        0, 14, 9, 14, 11, 2, 12, 4, 7, 13, 1, 5, 0,

        15, 10, 3, 9, 8, 6, 4, 2, 1, 11, 10, 13, 7,

        8, 15, 9, 12, 5, 6, 3, 0, 14, 11, 8, 12, 7,
        1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 },
        { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14,
        7, 5, 11, 10, 15, 4, 2, 7, 12, 9, 5, 6, 1,
        13, 14, 0, 11, 3, 8, 9, 14, 15, 5, 2, 8, 12,
        3, 7, 0, 4, 10, 1, 13, 11, 6, 4, 3, 2, 12,
        9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 },
        { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5,
        10, 6, 1, 13, 0, 11, 7, 4, 9, 1, 10, 14, 3,
        5, 12, 2, 15, 8, 6, 1, 4, 11, 13, 12, 3, 7,
        14, 10, 15, 6, 8, 0, 5, 9, 2, 6, 11, 13, 8,
        1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 },
        { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5,
        0, 12, 7, 1, 15, 13, 8, 10, 3, 7, 4, 12, 5,
        6, 11, 0, 14, 9, 2, 7, 11, 4, 1, 9, 12, 14,
        2, 0, 6, 10, 13, 15, 3, 5, 8, 2, 1, 14, 7,
        4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 }
};

// Straight Permutation Table
int per[32] = { 16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23,26, 5, 18, 31, 10, 2, 8, 24, 14, 32, 27,
                3, 9, 19, 13, 30, 6, 22, 11, 4, 25 };
cout << endl;
for (int i = 0; i < 16; i++) {
        // Expansion D-box
        string right_expanded = permute(right, exp_d, 48);

        // XOR RoundKey[i] and right_expanded
        string x = xor_(rkb[i], right_expanded);
```

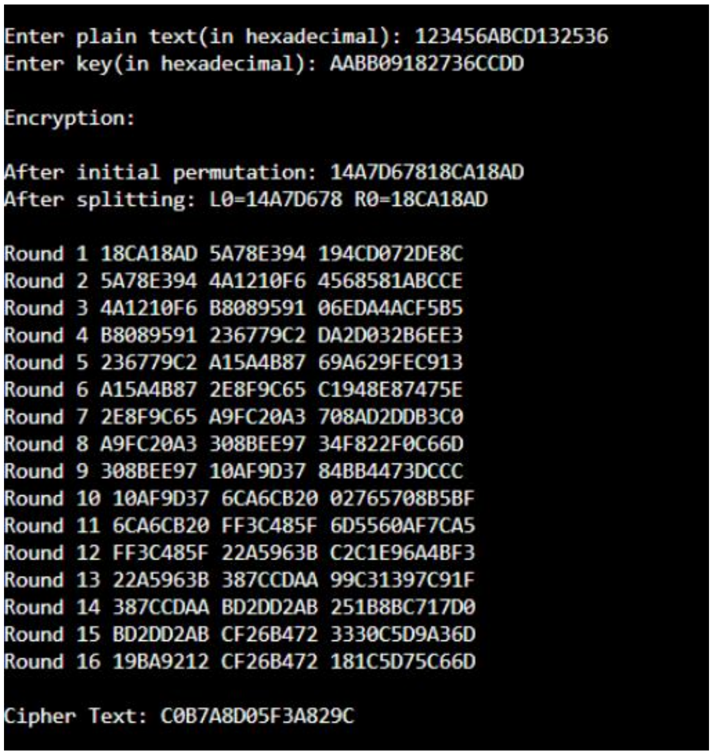
```
// S-boxes
string op = "";
for (int i = 0; i < 8; i++) {
        int row = 2 * int(x[i * 6] - '0') + int(x[i * 6 + 5] - '0');
        int col = 8 * int(x[i * 6 + 1] - '0') + 4 * int(x[i * 6 + 2] - '0')
                        + 2 * int(x[i * 6 + 3] - '0') + int(x[i * 6 + 4] - '0');
        int val = s[i][row][col];
        op += char(val / 8 + '0');
        val = val % 8;
        op += char(val / 4 + '0');
        val = val % 4;
        op += char(val / 2 + '0');
        val = val % 2;
        op += char(val + '0');
}
// Straight D-box
op = permute(op, per, 32);

// XOR left and op
x = xor_(op, left);

left = x;

// Swapper
if (i != 15)
        swap(left, right);
```

```
                cout << "Round " << i + 1 << " " << bin2hex(left)<< " " << bin2hex(right) << " "
                            << rk[i]<< endl;
        }

        string combine = left + right;              // Combination

        // Final Permutation Table
        int final_perm[64] = { 40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47,15, 55, 23, 63, 31, 38, 6, 46,
                                14, 54, 22,62, 30, 37, 5, 45, 13, 53, 21, 61, 29, 36,4, 44, 12, 52, 20,
                                60, 28, 35, 3, 43, 11,51, 19, 59, 27, 34, 2, 42, 10, 50, 18, 58,
                                26, 33, 1, 41, 9, 49, 17, 57, 25 };

        // Final Permutation
        string cipher = bin2hex(permute(combine, final_perm, 64));
        return cipher;
}

int main(){
        cout<<"AMAN CHAUHAN : 14614802719 : 7C7"<<endl<<endl;
        string pt, key;   // pt is plain text
        cout<<"Enter plain text(in hexadecimal): ";
        cin>>pt;
        cout<<"Enter key(in hexadecimal): ";
        cin>>key;

//      pt = "123456ABCD132536";     key = "AABB09182736CCDD";
        // Key Generation
        key = hex2bin(key);                 // Hex to binary
```

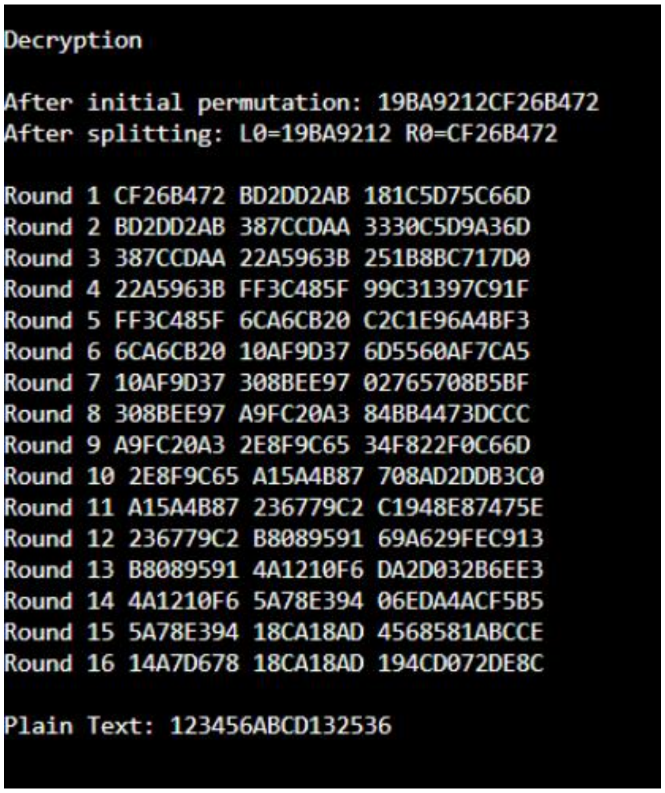
```
// Parity bit drop table
int keyp[56] = { 57, 49, 41, 33, 25, 17, 9, 1, 58, 50, 42, 34,26, 18, 10, 2, 59, 51, 43, 35, 27,
                 19, 11, 3,60, 52, 44, 36, 63, 55, 47, 39, 31, 23, 15, 7,62, 54, 46, 38, 30, 22,
                 14, 6, 61, 53, 45, 37,29, 21, 13, 5, 28, 20, 12, 4 };

// getting 56 bit key from 64 bit using the parity bits
key = permute(key, keyp, 56); // key without parity

// Number of bit shifts
int shift_table[16] = { 1, 1, 2, 2, 2, 2, 2, 2,1, 2, 2, 2, 2, 2, 2, 1 };

// Key- Compression Table
int key_comp[48] = { 14, 17, 11, 24, 1, 5, 3, 28,15, 6, 21, 10, 23, 19, 12, 4,
                     26, 8, 16, 7, 27, 20, 13, 2,41, 52, 31, 37, 47, 55, 30, 40,
                     51, 45, 33, 48, 44, 49, 39, 56,34, 53, 46, 42, 50, 36, 29, 32 };
// Splitting
string left = key.substr(0, 28);
string right = key.substr(28, 28);

vector<string> rkb; // rkb for RoundKeys in binary
vector<string> rk; // rk for RoundKeys in hexadecimal
for (int i = 0; i < 16; i++) {
        left = shift_left(left, shift_table[i]);   // Shifting
        right = shift_left(right, shift_table[i]);
        string combine = left + right;   // Combining
        string RoundKey = permute(combine, key_comp, 48);  // Key Compression
        rkb.push_back(RoundKey);
```

```
                rk.push_back(bin2hex(RoundKey));
        }

        cout << "\nEncryption:\n\n";
        string cipher = encrypt(pt, rkb, rk);
        cout << "\nCipher Text: " << cipher << endl;
        cout << "\nDecryption\n\n";
        reverse(rkb.begin(), rkb.end());
        reverse(rk.begin(), rk.end());
        string text = encrypt(cipher, rkb, rk);
        cout << "\nPlain Text: " << text << endl;
    }
```

Output:



```
Enter plain text(in hexadecimal): 123456ABCD132536
Enter key(in hexadecimal): AABB09182736CCDD

Encryption:

After initial permutation: 14A7D67818CA18AD
After splitting: L0=14A7D678 R0=18CA18AD

Round 1 18CA18AD 5A78E394 194CD072DE8C
Round 2 5A78E394 4A1210F6 4568581ABCCE
Round 3 4A1210F6 B8089591 06EDA4ACF5B5
Round 4 B8089591 236779C2 DA2D032B6EE3
Round 5 236779C2 A15A4B87 69A629FEC913
Round 6 A15A4B87 2E8F9C65 C1948E87475E
Round 7 2E8F9C65 A9FC20A3 708AD2DDB3C0
Round 8 A9FC20A3 308BEE97 34F822F0C66D
Round 9 308BEE97 10AF9D37 84BB4473DCCC
Round 10 10AF9D37 6CA6CB20 02765708B5BF
Round 11 6CA6CB20 FF3C485F 6D5560AF7CA5
Round 12 FF3C485F 22A5963B C2C1E96A4BF3
Round 13 22A5963B 387CCDAA 99C31397C91F
Round 14 387CCDAA BD2DD2AB 251B8BC717D0
Round 15 BD2DD2AB CF26B472 3330C5D9A36D
Round 16 19BA9212 CF26B472 181C5D75C66D

Cipher Text: C0B7A8D05F3A829C
```



```
Decryption

After initial permutation: 19BA9212CF26B472
After splitting: L0=19BA9212 R0=CF26B472

Round 1 CF26B472 BD2DD2AB 181C5D75C66D
Round 2 BD2DD2AB 387CCDAA 3330C5D9A36D
Round 3 387CCDAA 22A5963B 251B8BC717D0
Round 4 22A5963B FF3C485F 99C31397C91F
Round 5 FF3C485F 6CA6CB20 C2C1E96A4BF3
Round 6 6CA6CB20 10AF9D37 6D5560AF7CA5
Round 7 10AF9D37 308BEE97 02765708B5BF
Round 8 308BEE97 A9FC20A3 84BB4473DCCC
Round 9 A9FC20A3 2E8F9C65 34F822F0C66D
Round 10 2E8F9C65 A15A4B87 708AD2DDB3C0
Round 11 A15A4B87 236779C2 C1948E87475E
Round 12 236779C2 B8089591 69A629FEC913
Round 13 B8089591 4A1210F6 DA2D032B6EE3
Round 14 4A1210F6 5A78E394 06EDA4ACF5B5
Round 15 5A78E394 18CA18AD 4568581ABCCE
Round 16 14A7D678 18CA18AD 194CD072DE8C

Plain Text: 123456ABCD132536
```

VIVA – VOCE

1. **Give a one- or two-sentence definition of Symmetric key cryptography:**

⇨ Encryption:
cyphertext = plaintext + encryption key

⇨ Decryption:
ciphertext + decryption key = plaintext
such that decryption only succeeds if the encryption key and decryption key are the same key.

2. **Give a one- or two-sentence definition Asymmetric key cryptography.**

⇨ As for symmetric key cryptography, except that the encryption key and decryption key do not have to be equal, they have to be inverses of each other. When an asymmetric key pair is generated, an encryption key and its inverse decryption key are generated.

3. **How can a message between 2 people be authenticated using symmetric encryption?**

⇨ 1. Create a hash / digest of the message.
2. Encrypt the digest it with the shared key
3. Send the message and the encrypted digest to the recipient
4. Recipient creates a new digest of just the message
5. Recipient decrypts the received digest and compares it to his digest

EXPERIMENT – 6

AIM: To implement Diffie-Hellman algorithm.

Theory: Diffie-Hellman algorithm

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.
For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Step by Step Explanation:

Alice	**Bob**
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Exchange of generated keys takes place	
Key received = y	key received = x

Users now have a symmetric secret key to encrypt

Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice: x = (9^4 mod 23) = (6561 mod 23) = 6
Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and Bob receives public key x = 6

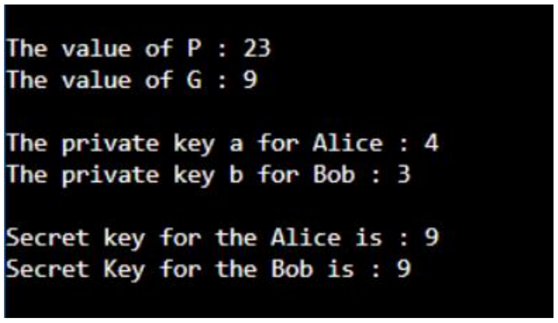
Step 6: Alice and Bob compute symmetric keys
Alice: ka = y^a mod p = 65536 mod 23 = 9
Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

Source Code:

```
#include<bits/stdc++.h>
using namespace std;

long long int power(long long int a, long long int b, long long int P){
   if (b == 1)    return a;
   else
      return (((long long int)pow(a, b)) % P);        // return value of a ^ b mod P
}

int main(){
   cout<<"AMAN CHAUHAN : 14614802719 : 7C7"<<endl<<endl;
   long long int P, G, x, a, y, b, ka, kb;
   // Both the persons will be agreed upon the public keys G and P
   P = 23; // A prime number P is taken
   cout<<"The value of P : "<<P<<endl;
   G = 9; // A primitive root for P, G is taken
   cout<<"The value of G : "<<G<<endl<<endl;

   a = 4; // a is the chosen private key by ALice
   cout<<"The private key a for Alice : "<<a<<endl;
   x = power(G, a, P); // gets the generated key

   b = 3; // b is the chosen private key by Bob
   cout<<"The private key b for Bob : "<<b<<endl<<endl;
   y = power(G, b, P); // gets the generated key

   ka = power(y, a, P); // Generating Secret key for Alice
   kb = power(x, b, P); // Generating Secret key for Bob
   cout<<"Secret key for the Alice is : "<<ka<<endl;
   cout<<"Secret Key for the Bob is : "<<kb<<endl;

   return 0;
}
```

OUTPUT:



```
The value of P : 23
The value of G : 9

The private key a for Alice : 4
The private key b for Bob : 3

Secret key for the Alice is : 9
Secret Key for the Bob is : 9
```

VIVA – VOCE

1. **How can a message between 2 people be authenticated using asymmetric encryption?**

⇨ 1. Create a hash / digest of the message.
2. Encrypt the digest with the sender's private key, which can only be decrypted with the sender's matching public key
3. Send the message and the encrypted digest to the recipient
4. Recipient creates a new digest of just the message
5. Recipient decrypts the received digest with the sender's public and compares it to his digest

2. **The exercise is to calculate what the key would be for Alice and Bob using these known values:**
Prime number selected is p = 337
Random number selected is g = 3 **Generator
Alice random private number selected is xa = 7
Bob's random private number selected is xb = 11. What is the key that Alice and Bob will use?

⇨ Alice: gamod p
37 mod 337
2187 mod 337= 165 – Public Value
gab= (gb)amod p
37*11= (311)7mod 337
5.47 x 1036 = 1771477mod 337
5.47 x 1036 = 5.47 x 1036 mod 337= 5 – Shared Secret Key
Bob: gb mod p = 3
311 mod 337
177147 mod 337
= 222 – Public Value
311*7= (37)11mod 337
5.474e+36 = 5.474e+36 mod 337 = 5 -Shared Key

EXPERIMENT – 7

AIM: Make a study of anyone simulation tool based on parameters of information security.

Theory:

NeSSi
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.

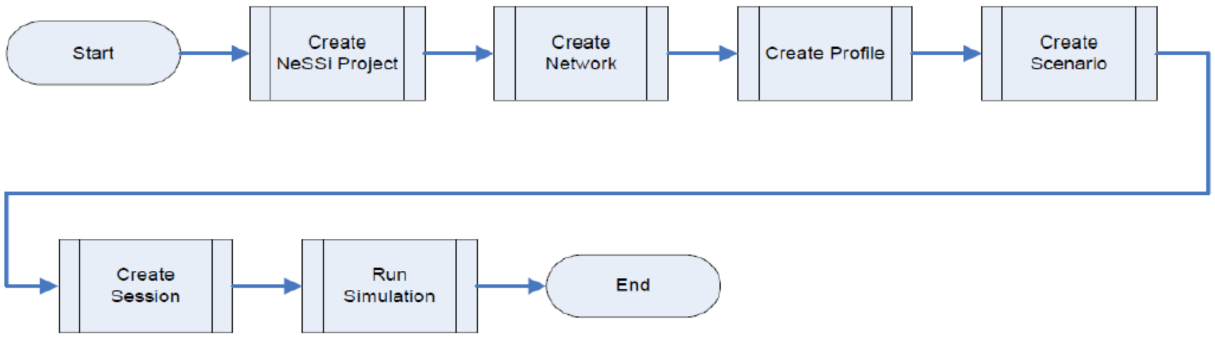
Hence, the NeSSi² extension API allows the creation of

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

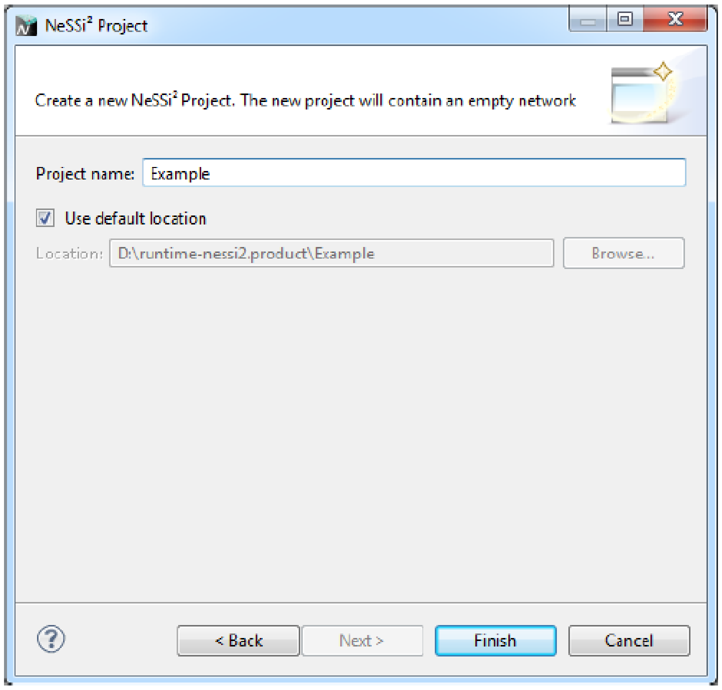
.

Sample Instructions

Figure shows the steps necessary for creating a simulation. Each step will be introduced in own section respectively.



1. As a first step a *NeSSi2* project has to be created using the *NeSSi2* project creation wizard. This wizard can be accessed in two ways in the *NeSSi2* GUI. One way is by the following menu sequence: *File -> New -> IP Network (Project).*


NeSSi² Project

Create a new NeSSi² Project. The new project will contain an empty network

Project name: Example

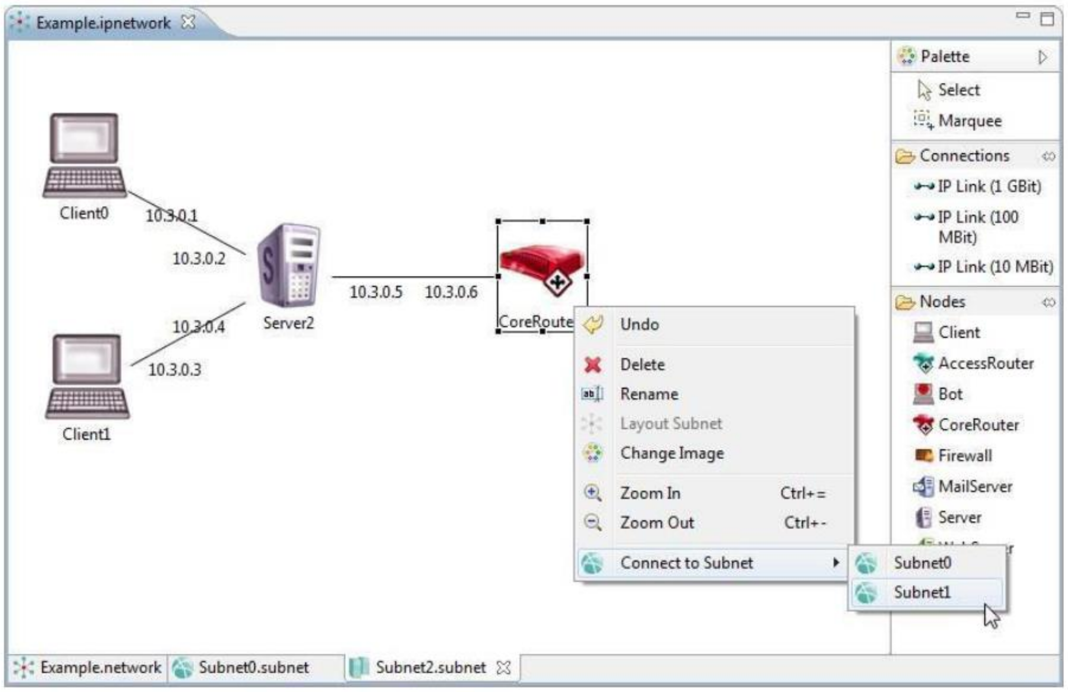
☑ Use default location

Location: D:\runtime-nessi2.product\Example

2. **Create network**

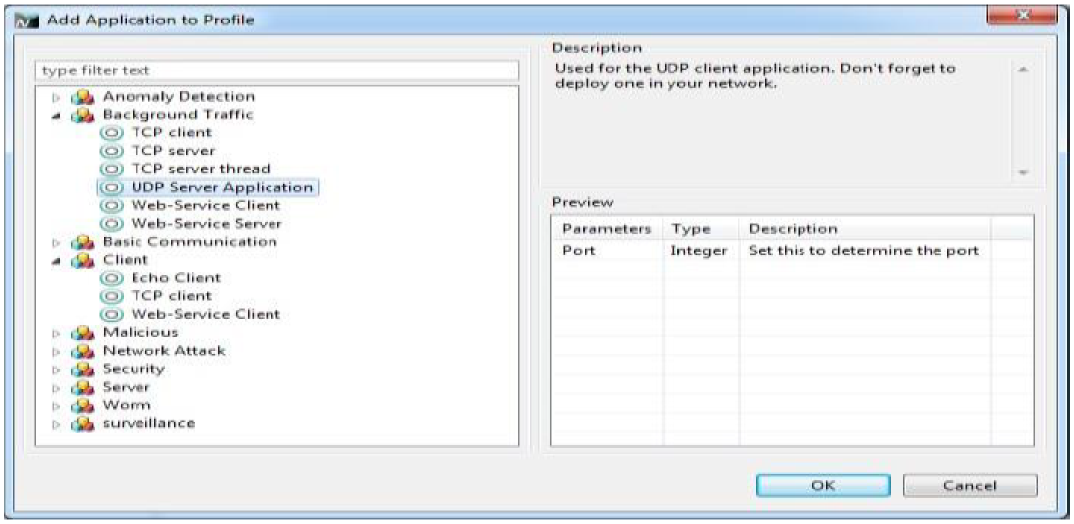
   a. For adding content to the created network, double click on the network. This will cause for the network to be opened in the Network Editor. The nodes and edges of each network are grouped in subnets. Hence, the first step of creating a network is defining some subnets for the network. The available subnet types are listed in the palette located on the right side of the Network Editor. Via drag and drop you can add the subnets to the network. The location of the subnets can be changed anytime. For selecting a single network, choose the *Select* tool from the palette. For simultaneous selection of multiple subnets, use the *Marquee* tool from the palette. The names of the subnets will be automatically generated.

   b. The next step after adding subnets to the network is to add content to the subnets. In order to do so, double click on a subnet. This will cause for a new tab to be opened in the Network Editor. Same as with the subnets, the palette will display a list of the available nodes and edges. By dragging and dropping nodes, they will be added to the network. For connecting nodes, you will have to select one of the links found in the Connections part of the palette and then clicking on the nodes that you want to connect.



3. **Create Profile**

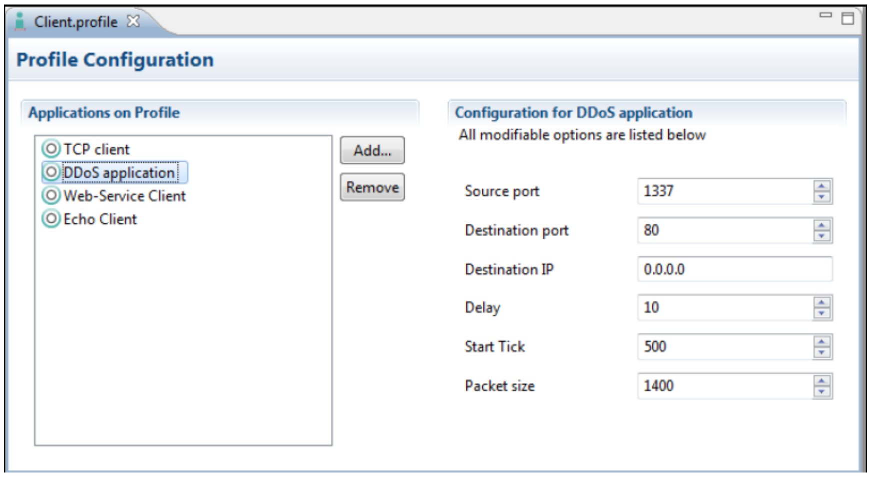
A profile is basically a container for a set of applications. A profile can be created in two ways. One is by accessing the context menu in the *NeSSi2* Project, Explorer and selecting *New->Profile*. The other way is by the using the menu sequence *File->New->Profile* in the main menu. This will open a dialog where a name has to be entered for the profile. By pressing "Finish", the dialog will close and a file with the *.profile* extension will be created in the *Profiles* folder of the project. Now, applications may be added to the profile.



Used for the UDP client application. Don't forget to deploy one in your network.

Parameters	Type	Description
Port	Integer	Set this to determine the port

By pressing the Add button in the newly created profile, a dialog containing a list of the available applications will open. The applications are sorted into categories. Note that an application may belong to several categories, so it may appear in the list several times. The search field on top of the application list can be used to filter applications according to their name. When selecting an application, if available, a description of the application and its configurable parameters will appear on the right side of the dialog. By pressing OK, the dialog will close and the selected application will be added to the profile.



Client.profile

Profile Configuration

Applications on Profile

- TCP client
- DDoS application
- Web-Service Client
- Echo Client

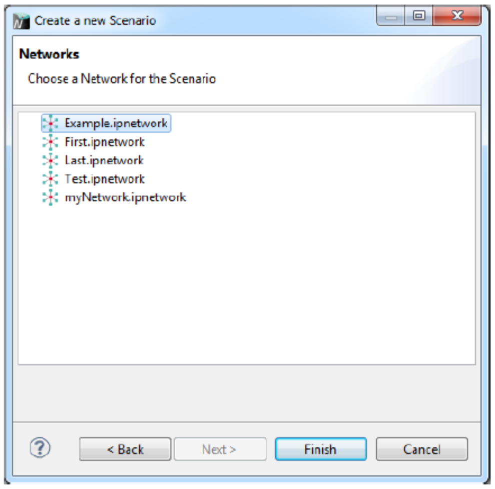
Configuration for DDoS application

All modifiable options are listed below

Source port	1337
Destination port	80
Destination IP	0.0.0.0
Delay	10
Start Tick	500
Packet size	1400

In the profile editor, you can configure the application that you have added to the profile. When clicking on an application name, the parameters will be become visible. Depending on the application, different configuration parameter will be displayed. Once the profiles have been created, they need to be mapped into the network nodes.

4. **Create Scenario**
   A scenario is the mapping of profiles onto the nodes of a specific network. Same as with creating profiles, the Scenario Creation wizard may be accessed through *NeSSi2* Project Explorer's context menu (*New! Scenario*) or through the main menu (*File->New->Scenario*). A name has to be entered for the scenario.

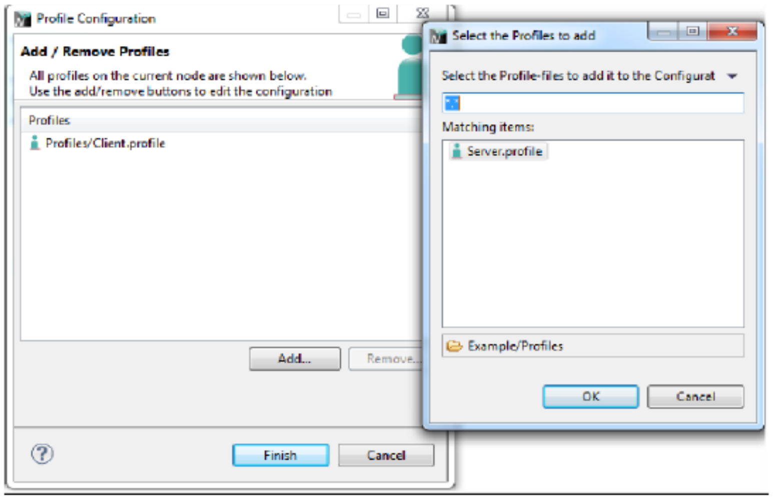


Create a new Scenario

Networks

Choose a Network for the Scenario

- Example.ipnetwork
- First.ipnetwork
- Last.ipnetwork
- Test.ipnetwork
- myNetwork.ipnetwork


Profile Configuration

Add / Remove Profiles

All profiles on the current node are shown below.
Use the add/remove buttons to edit the configuration

Profiles
Profiles/Client.profile

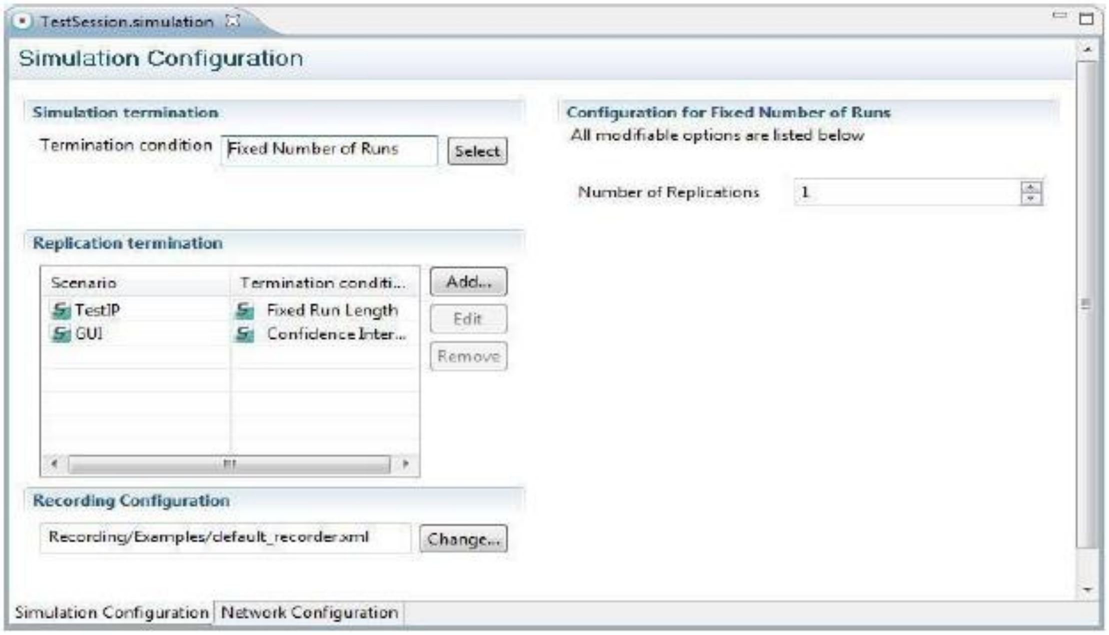
Select the Profiles to add

Select the Profile-files to add it to the Configurat

Matching items:
Server.profile

Example/Profiles

5. **Create Simulation**
   A simulation specifies the scenarios that are to be simulated and the duration of a simulation. Analogous to the creation of profiles and scenarios, a simulation can be created in *NeSSi2* Project Explorer by selecting *New->Session* or by selecting *File->New->Session* the main menu.


TestSession.simulation

Simulation Configuration

Simulation termination
Termination condition: Fixed Number of Runs

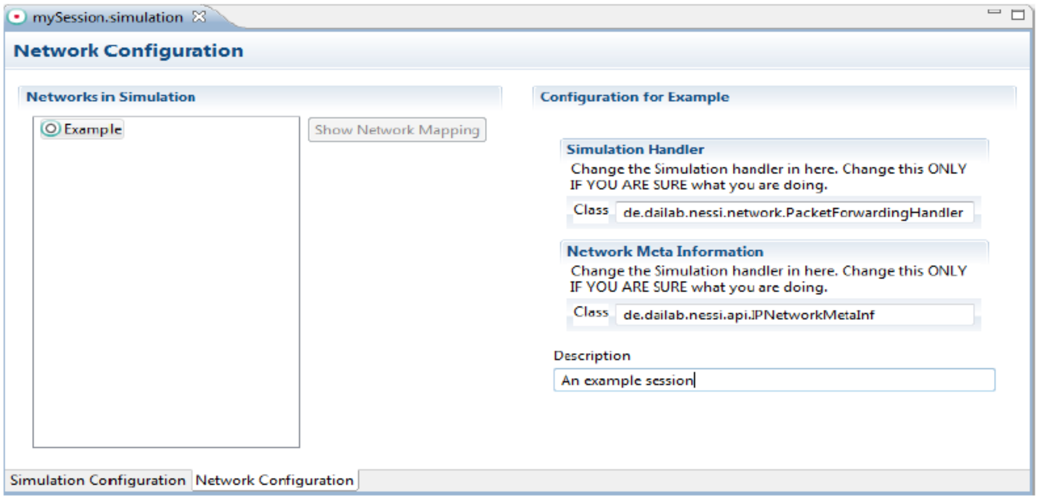
Configuration for Fixed Number of Runs
All modifiable options are listed below
Number of Replications: 1

Replication termination

Scenario	Termination conditi...
TestIP	Fixed Run Length
GUI	Confidence Inter...

Recording Configuration
Recording/Examples/default_recorder.xml

In the Simulation Configuration tab, the options for the simulation can be configured. In the top left of the editor, in the "Simulation Termination" section, the number of replications can be specified. By pressing the select button, you can choose a termination condition from the list of the available ones. Once a termination condition has been chosen, its configuration parameters will appear on the right side of the dialog. In this example, the options for the "Fixed Number of Runs" is an integer that explicitly specifies the number of times that the scenarios will be simulated.



mySession.simulation

Network Configuration

Networks in Simulation

Example

Show Network Mapping

Configuration for Example

Simulation Handler

Change the Simulation handler in here. Change this ONLY IF YOU ARE SURE what you are doing.

Class: de.dailab.nessi.network.PacketForwardingHandler

Network Meta Information

Change the Simulation handler in here. Change this ONLY IF YOU ARE SURE what you are doing.

Class: de.dailab.nessi.api.IPNetworkMetaInf

Description: An example session

Simulation Configuration | Network Configuration

6. **Run Simulation**
   For executing a simulation, select the simulation file in the *NeSSi2* Project Explorer and select the "Launch Simulation" action from the simulation's context menu. This will send the simulation to the *NeSSi2* Backend for execution. Outputs from the *NeSSi2* Backend can be viewed in the console where the backend is running. In the GUI, a dialog will appear, which proposes switching to the Simulation Perspective. By confirming, the Simulation Perspective will open, where the results of the simulation execution can be viewed.
   You will have to select the executed simulation replication from the list displayed in the Recorded Simulations view, which is located by default on the left side of the Simulation perspective. This will cause for an editor to open where the network may be viewed. By using the play buttons in the icons bar the simulation can be reviewed. The speed of the simulation execution can be adjusted using the playback buttons. When a packet is sent through a network link, the link will become green. Depending on the amount of the packets sent, the link my become wider. When a network device or link is selected, the values logged for that network element can be viewed in the Statistics view.

7. **Evaluating Simulations**
   NeSSi2 allows the simulation of various security scenarios. Additionally, there is a huge diversity in network security evaluation metrics Here, the developer of a detection algorithm respectively of a special security infrastructure set-up may not only be interested in detection rates, but also in the economical assessment of a scenario. Hence, the gathering of simulation results and the evaluation needs to be very flexible. Here, we apply an event-based approach, the so-called Meta Attack Events. Already included events incorporate dropped packets, infected flows, compromised machines, unavailable services etc. Those events are stored in the database at runtime. Events belonging to the same attack refer to a global identifier to differentiate between the impacts of different attacks. The database associates those events with a time stamp in the simulation as well as a device and/or transmitted packets related to that specific event.

EXPERIMENT – 8

AIM: Implement VPN through Packet-Tracer or any other network simulator tool

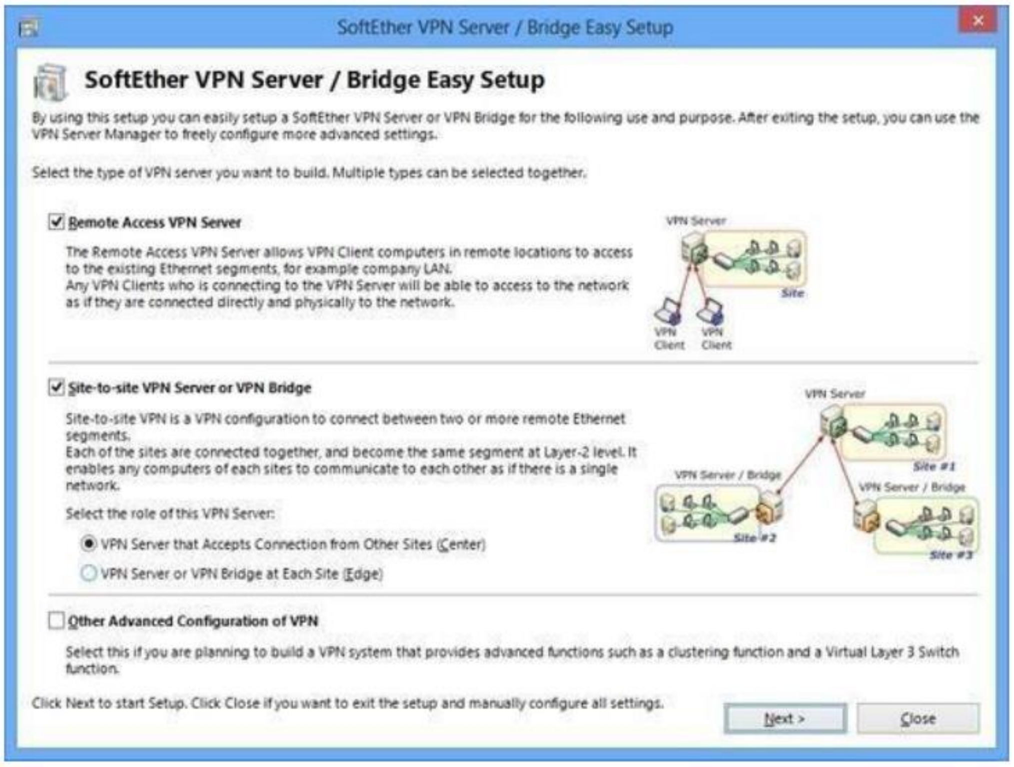
Theory:

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 client can use any kinds of IP-based or Ethernet- based protocols via the VPN even if they are distributed around the world.

Sample Instructions

Step 1. Set up SoftEther VPN Server

Designate a computer in the group as the VPN Server. Set up SoftEther VPN Server on that computer. It is very easy by using Installer and Initial Setup Wizard based GUI.



SoftEther VPN Server / Bridge Easy Setup

SoftEther VPN Server / Bridge Easy Setup

By using this setup you can easily setup a SoftEther VPN Server or VPN Bridge for the following use and purpose. After exiting the setup, you can use the VPN Server Manager to freely configure more advanced settings.

Select the type of VPN server you want to build. Multiple types can be selected together.

[✓] Remote Access VPN Server

The Remote Access VPN Server allows VPN Client computers in remote locations to access to the existing Ethernet segments, for example company LAN.
Any VPN Clients who is connecting to the VPN Server will be able to access to the network as if they are connected directly and physically to the network.

[✓] Site-to-site VPN Server or VPN Bridge

Site-to-site VPN is a VPN configuration to connect between two or more remote Ethernet segments.
Each of the sites are connected together, and become the same segment at Layer-2 level. It enables any computers of each sites to communicate to each other as if there is a single network.

Select the role of this VPN Server:

● VPN Server that Accepts Connection from Other Sites (Center)

○ VPN Server or VPN Bridge at Each Site (Edge)

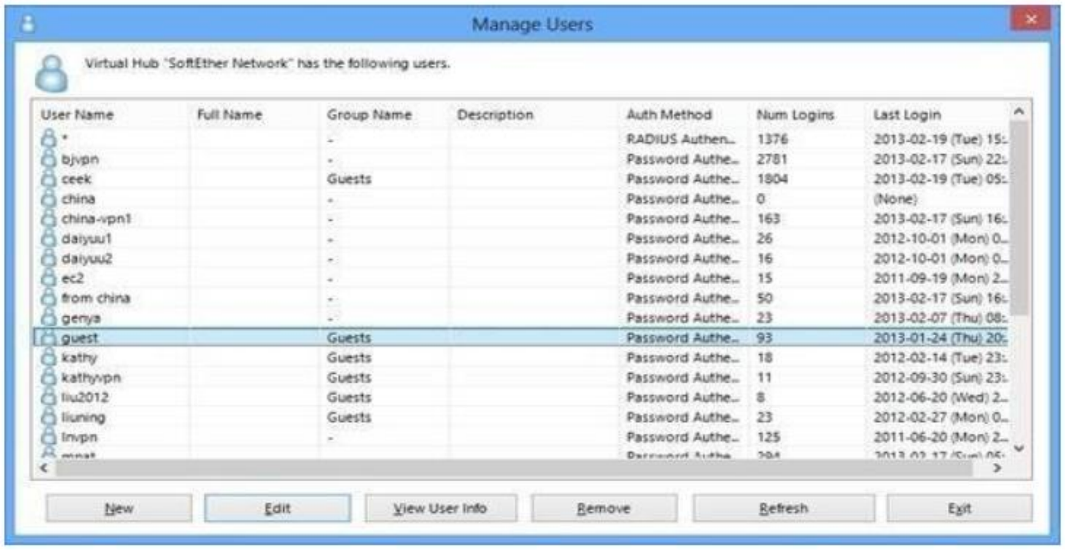
[ ] Other Advanced Configuration of VPN

Select this if you are planning to build a VPN system that provides advanced functions such as a clustering function and a Virtual Layer 3 Switch function.

Click Next to start Setup. Click Close if you want to exit the setup and manually configure all settings.

Step 2. Create Users

On the VPN Server you can add several user objects on the Virtual Hub. Each user object has a password. After that, distribute pairs of username and password to each member of the VPN.



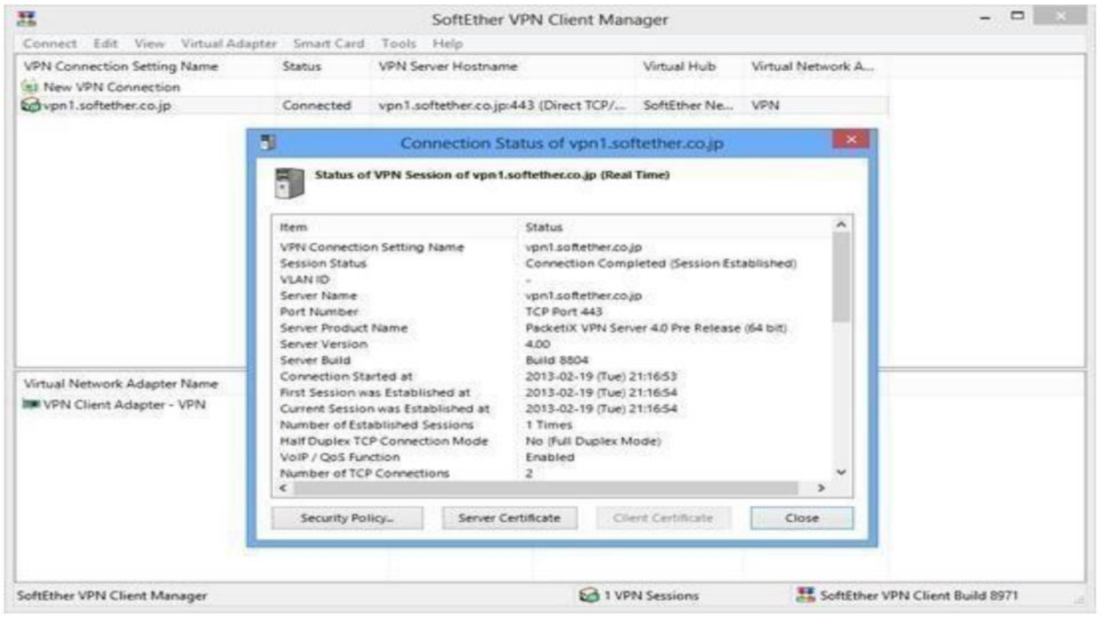
Manage Users

Virtual Hub "SoftEther Network" has the following users.

User Name	Full Name	Group Name	Description	Auth Method	Num Logins	Last Login
*		-		RADIUS Authen...	1376	2013-02-19 (Tue) 15:..
bjvpn		-		Password Authe...	2781	2013-02-17 (Sun) 22:..
ceek		Guests		Password Authe...	1804	2013-02-19 (Tue) 05:..
china		-		Password Authe...	0	(None)
china-vpn1		-		Password Authe...	163	2013-02-17 (Sun) 16:..
daiyuu1		-		Password Authe...	26	2012-10-01 (Mon) 0...
daiyuu2		-		Password Authe...	16	2012-10-01 (Mon) 0...
ec2		-		Password Authe...	15	2011-09-19 (Mon) 2...
from china		-		Password Authe...	50	2013-02-17 (Sun) 16:..
genya		-		Password Authe...	23	2013-02-07 (Thu) 08:..
guest		Guests		Password Authe...	93	2013-01-24 (Thu) 20:..
kathy		Guests		Password Authe...	18	2012-02-14 (Tue) 23:..
kathyvpn		Guests		Password Authe...	11	2012-09-30 (Sun) 23:..
liu2012		Guests		Password Authe...	8	2012-06-20 (Wed) 2...
liuning		Guests		Password Authe...	23	2012-02-27 (Mon) 0...
lnvpn		-		Password Authe...	125	2011-06-20 (Mon) 2...

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

On each member's PC install 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 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.



SoftEther VPN Client Manager

VPN Connection Setting Name	Status	VPN Server Hostname	Virtual Hub	Virtual Network A...
New VPN Connection				
vpn1.softether.co.jp	Connected	vpn1.softether.co.jp:443 (Direct TCP/...	SoftEther Ne...	VPN

Virtual Network Adapter Name
VPN Client Adapter - VPN

Connection Status of vpn1.softether.co.jp

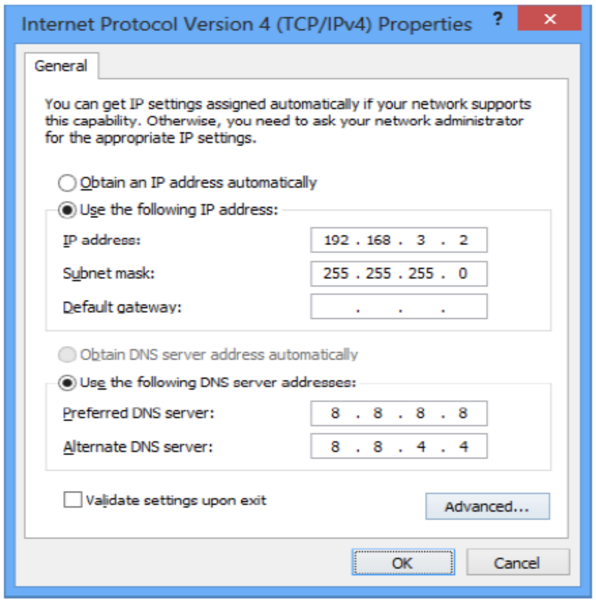
Status of VPN Session of vpn1.softether.co.jp (Real Time)

Item	Status
VPN Connection Setting Name	vpn1.softether.co.jp
Session Status	Connection Completed (Session Established)
VLAN ID	-
Server Name	vpn1.softether.co.jp
Port Number	TCP Port 443
Server Product Name	PacketiX VPN Server 4.0 Pre Release (64 bit)
Server Version	4.00
Server Build	Build 8804
Connection Started at	2013-02-19 (Tue) 21:16:53
First Session was Established at	2013-02-19 (Tue) 21:16:54
Current Session was Established at	2013-02-19 (Tue) 21:16:54
Number of Established Sessions	1 Times
Half Duplex TCP Connection Mode	No (Full Duplex Mode)
VoIP / QoS Function	Enabled
Number of TCP Connections	2

SoftEther VPN Client Manager   1 VPN Sessions   SoftEther VPN Client Build 8971

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.


Internet Protocol Version 4 (TCP/IPv4) Properties

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

○ Obtain an IP address automatically
● Use the following IP address:

IP address:	192 . 168 . 3 . 2
Subnet mask:	255 . 255 . 255 . 0
Default gateway:	. . .

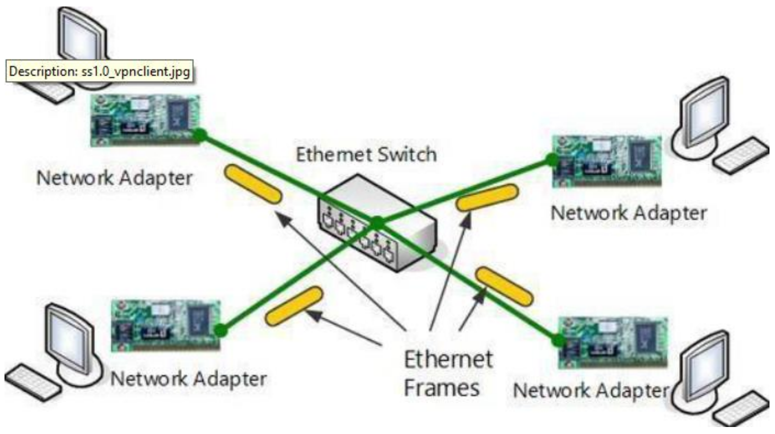
○ Obtain DNS server address automatically
● Use the following DNS server addresses:

Preferred DNS server:	8 . 8 . 8 . 8
Alternate DNS server:	8 . 8 . 4 . 4

☐ Validate settings upon exit

Step 5. Communicate Like Physical Ethernet

Once every computer 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.



VIVA – VOCE

1. **If you receive an error related to Protocol Error is occurring, what all troubleshooting measures can be taken up?**

⇨ The first thing to do when you get a Protocol Driver Error/Error 1046 is to verify Citrix Receiver/Workspace is upgraded to the latest version if possible or the latest version that has been validated in the destination environment. If upgrading receiver doesn’t help, or if the latest version is already running on the affected workstation, next recommendation would be to perform a full uninstall of receiver utilizing the Citrix Receiver Cleanup Utility.

2. **There is a large number of broadcast packets constantly being sent over the network. What should I check?**

⇨ 1. Storm control and equivalent protocols allow you to rate-limit broadcast packets. If your switch has such a mechanism, turn it on.

2. Ensure IP-directed broadcasts are disabled on your Layer 3 devices. There’s little to no reason why you’d want broadcast packets coming in from the internet going to a private address space. If a storm is originating from the WAN, disabling IP-directed broadcasts will shut it down.

3. Split up your broadcast domain. Creating a new VLAN and migrating hosts into it will load balance the broadcast traffic to a more acceptable level. Broadcast traffic is necessary and useful, but too much of it eventually leads to a poor network experience.

4. Check how often ARP tables are emptied.

3. **Which protocol is suited for communication over insecure channel?**

⇨ A secure communication over an insecure channel without any prior exchanged key can be established with the help of an authentication step to exchange a public key and then using public-key cryptography such as RSA.

4. **How many firewalls should be there in the network?**

⇨ If your network is entirely client-protecting, or is client-protecting with just a few incoming services, such as email, then one firewall (or a pair of firewalls configured as a high-availability pair) is probably all you need.