Table of Contents

Practical 1

Encryption in Networks using High-level Elliptic Curve Cryptography using a Client/Server Program.

Requires two different systems constituting the server and the client systems respectively.

Server.py

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
import socket
def generate_key_pair():
private_key = ec.generate_private_key(ec.SECP256R1(), default_backend())
public_key = private_key.public_key()
return private_key, public_key
def perform_key_exchange(client_public_key, private_key):
shared_key = private_key.exchange(ec.ECDH(), client_public_key)
derived_key = HKDF(
algorithm=hashes.SHA256(),
length=32,
salt=None,
info=b'key derivation',
backend=default_backend()
).derive(shared_key)
return derived_key
def encrypt(message, key):
cipher = Cipher(algorithms.AES(key), modes.CFB(b'\0' * 16),
backend=default_backend())
encryptor = cipher.encryptor()
return encryptor.update(message) + encryptor.finalize()
def decrypt(ciphertext, key):
cipher = Cipher(algorithms.AES(key), modes.CFB(b'\0' * 16),
backend=default_backend())
```

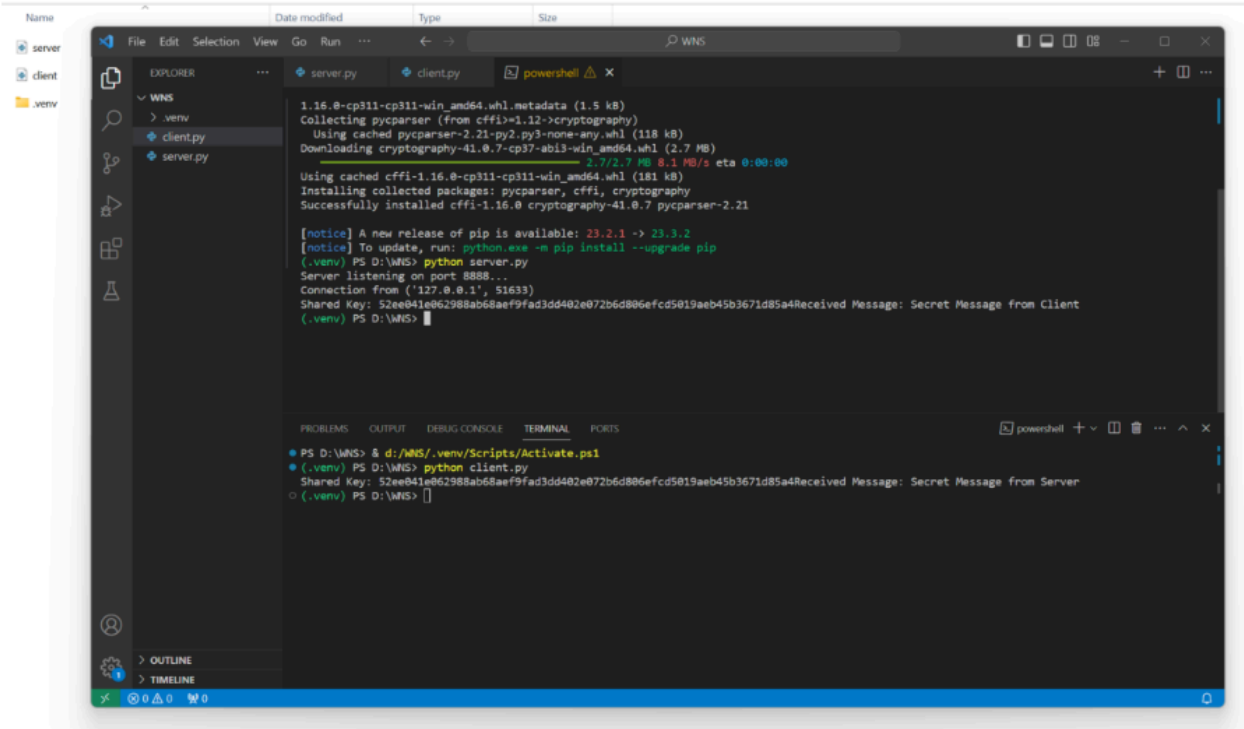
```
decryptor = cipher.decryptor()
return decryptor.update(ciphertext) + decryptor.finalize()
def start_server():
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 8888))
server_socket.listen(1)
print("Server listening on port 8888...")

client_socket, address = server_socket.accept()
print("Connection from", address)
private_key, public_key = generate_key_pair()
client_socket.sendall(public_key.public_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PublicFormat.SubjectPublicKeyInfo
))
client_public_key_bytes = client_socket.recv(1024)
client_public_key =
serialization.load_pem_public_key(client_public_key_bytes,
default_backend())
shared_key = perform_key_exchange(client_public_key, private_key)
print("Shared Key:", shared_key.hex())
message = b"Secret Message from Server"
ciphertext = encrypt(message, shared_key)
client_socket.sendall(ciphertext)
received_ciphertext = client_socket.recv(1024)
decrypted_message = decrypt(received_ciphertext, shared_key)
print("Received Message:", decrypted_message.decode())
client_socket.close()
server_socket.close()
if __name__ == "__main__":
start_server()
```

Client.py

```
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import ec
from cryptography.hazmat.primitives.kdf.hkdf import HKDF
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
```

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
import socket
def generate_key_pair():
private_key = ec.generate_private_key(ec.SECP256R1(), default_backend())
public_key = private_key.public_key()
return private_key, public_key

def perform_key_exchange(server_public_key, private_key):
shared_key = private_key.exchange(ec.ECDH(), server_public_key)
derived_key = HKDF(
algorithm=hashes.SHA256(),
length=32,
salt=None,
info=b'key derivation',
backend=default_backend()
).derive(shared_key)
return derived_key
def encrypt(message, key):
cipher = Cipher(algorithms.AES(key), modes.CFB(b'\0' * 16),
backend=default_backend())
encryptor = cipher.encryptor()
return encryptor.update(message) + encryptor.finalize()
def decrypt(ciphertext, key):
cipher = Cipher(algorithms.AES(key), modes.CFB(b'\0' * 16),
backend=default_backend())
decryptor = cipher.decryptor()
return decryptor.update(ciphertext) + decryptor.finalize()
def start_client():
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 8888))
private_key, public_key = generate_key_pair()
client_socket.sendall(public_key.public_bytes(
encoding=serialization.Encoding.PEM,
format=serialization.PublicFormat.SubjectPublicKeyInfo
))
server_public_key_bytes = client_socket.recv(1024)
server_public_key =
serialization.load_pem_public_key(server_public_key_bytes,
default_backend())
shared_key = perform_key_exchange(server_public_key, private_key)
```

```
print("Shared Key:", shared_key.hex())
received_ciphertext = client_socket.recv(1024)
decrypted_message = decrypt(received_ciphertext, shared_key)
print("Received Message:", decrypted_message.decode())
message = b"Secret Message from Client"

ciphertext = encrypt(message, shared_key)
client_socket.sendall(ciphertext)
client_socket.close()
if __name__ == "__main__":
start_client()
```

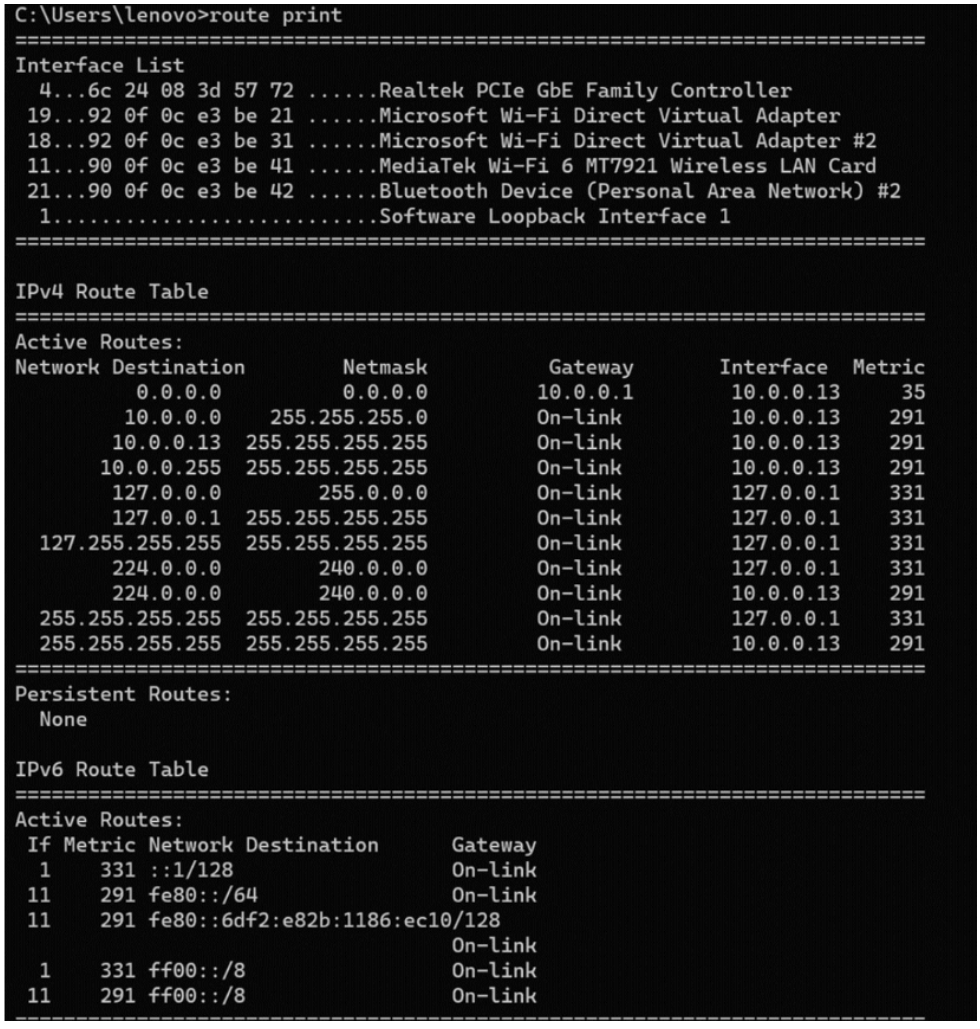


Practical 2

Write and execute commands:
a. To view routing Table
b. To view network statistics of a network
c. To view all routes
d. To update/modify/add/delete routes in a routing Table.

a.) To view routing Table:

The "route print" command is a networking utility available in the Windows operating system. It is used to display the current IPv4 and Ipv6 routing table on a Windows machine. This command provides a detailed listing of network routes, including destination addresses, subnet masks, gateway addresses, interface indexes, and associated metrics.



```
C:\Users\lenovo>route print
===========================================================================
Interface List
  4...6c 24 08 3d 57 72 ......Realtek PCIe GbE Family Controller
 19...92 0f 0c e3 be 21 ......Microsoft Wi-Fi Direct Virtual Adapter
 18...92 0f 0c e3 be 31 ......Microsoft Wi-Fi Direct Virtual Adapter #2
 11...90 0f 0c e3 be 41 ......MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
 21...90 0f 0c e3 be 42 ......Bluetooth Device (Personal Area Network) #2
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.0.0.1        10.0.0.13     35
         10.0.0.0    255.255.255.0         On-link         10.0.0.13    291
        10.0.0.13  255.255.255.255         On-link         10.0.0.13    291
       10.0.0.255  255.255.255.255         On-link         10.0.0.13    291
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link         10.0.0.13    291
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link         10.0.0.13    291
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
 11    291 fe80::/64                On-link
 11    291 fe80::6df2:e82b:1186:ec10/128
                                    On-link
  1    331 ff00::/8                 On-link
 11    291 ff00::/8                 On-link
===========================================================================
```

b.) To view network statistics of a network:
Netstat command can be used to view the network statistics. The primary function is to display network-related information, both at the transport layer (TCP/UDP) and the network layer (IP). The command provides a snapshot of active network connections, routing tables, interface statistics, masquerade connections, and various network-related metrics.

Commonly used options with netstat include:
-a (all): Display all connections and listening ports.
-n (numeric): Show numerical addresses instead of resolving hostnames.
-p (program): Display the process ID and name associated with each connection.
-r (route): Show the routing table.

c.) To view all routes:
The following commands can be used to view all routes:

i) route print:
This command displays the IPv4 routing table, including destination network addresses, subnet masks, gateways, interface indexes, and metric values.

ii) netstat -r:
This command shows the routing table in a more summarized form, providing information about destination networks, gateways, and interface indexes.

```
C:\Users\lenovo>netstat -r
===========================================================================
Interface List
  4...6c 24 08 3d 57 72 ......Realtek PCIe GbE Family Controller
 19...92 0f 0c e3 be 21 ......Microsoft Wi-Fi Direct Virtual Adapter
 18...92 0f 0c e3 be 31 ......Microsoft Wi-Fi Direct Virtual Adapter #2
 11...90 0f 0c e3 be 41 ......MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
 21...90 0f 0c e3 be 42 ......Bluetooth Device (Personal Area Network) #2
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.0.0.1        10.0.0.13     35
         10.0.0.0    255.255.255.0         On-link         10.0.0.13    291
        10.0.0.13  255.255.255.255         On-link         10.0.0.13    291
       10.0.0.255  255.255.255.255         On-link         10.0.0.13    291
        127.0.0.0        255.0.0.0         On-link         127.0.0.1    331
        127.0.0.1  255.255.255.255         On-link         127.0.0.1    331
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link         127.0.0.1    331
        224.0.0.0        240.0.0.0         On-link         10.0.0.13    291
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    331
  255.255.255.255  255.255.255.255         On-link         10.0.0.13    291
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
 11    291 fe80::/64                On-link
 11    291 fe80::6df2:e82b:1186:ec10/128
                                    On-link
  1    331 ff00::/8                 On-link
 11    291 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

d.) To update/modify/add/delete routes in a routing Table:
i) “route add” command to add a route in a routing table.

```
C:\Windows\System32>route add 192.168.39.0 MASK 255.255.255.0 192.168.221.246
 OK!
```

ii) “route change” command to change/modify a route in routing table, In the below example we have modified the gateway for the existing route.

```
C:\Windows\System32>route change 192.168.39.0 MASK 255.255.255.0 192.168.221.0
 OK!
```

iii) “route delete” command to delete a route from the routing table.

```
C:\Windows\System32>route delete 192.168.39.0
 OK!
```

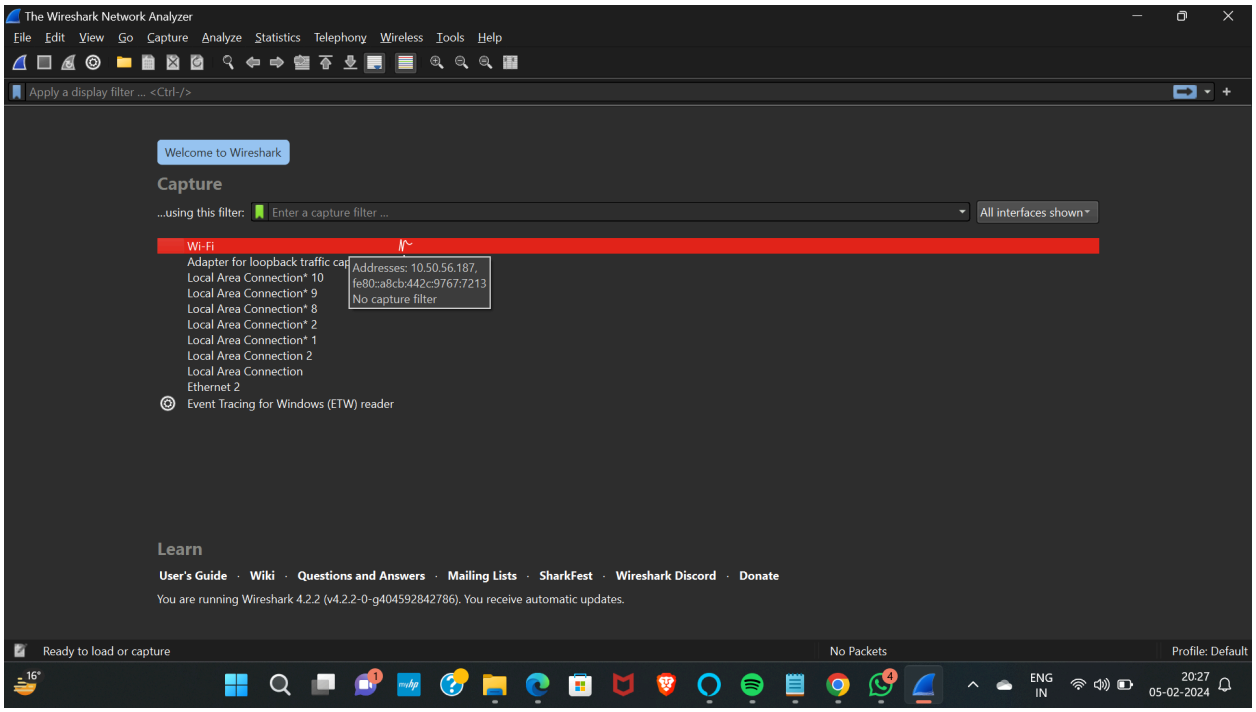
Practical 3

Decryption SSL/TLS Traffic using Wireshark.

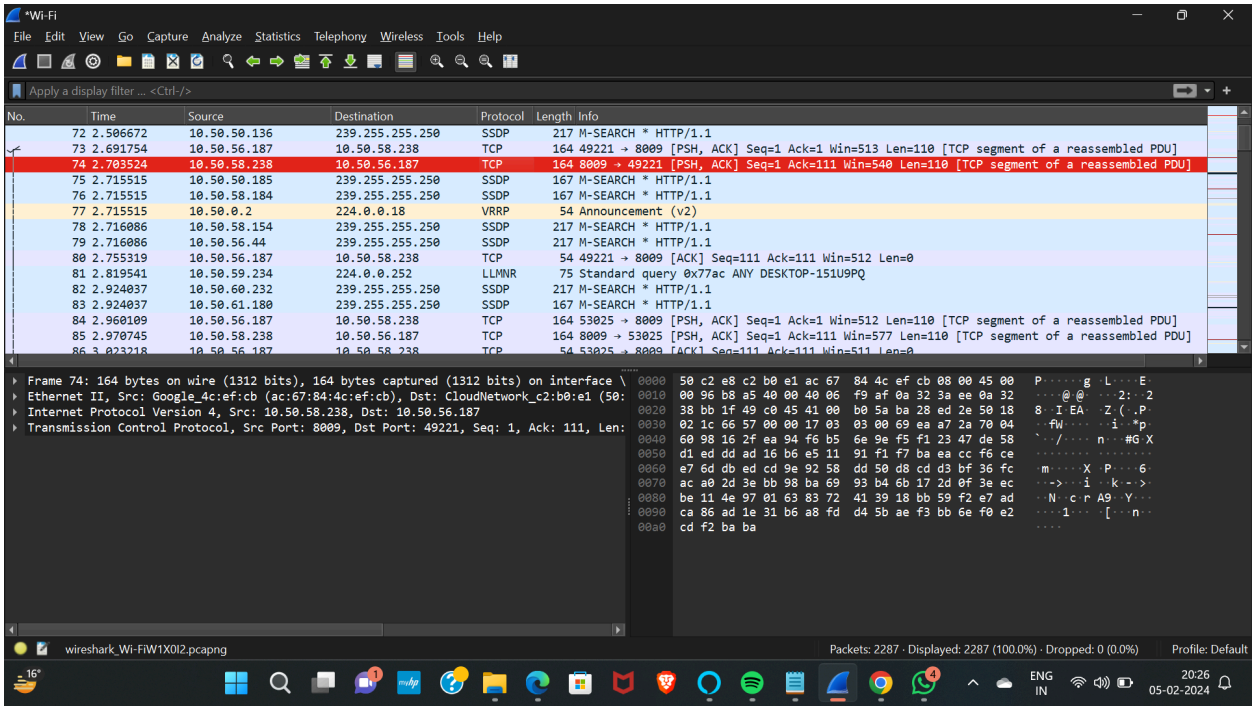
To decrypt SSL/TLS data in Wireshark, we typically need access to the private key used for the SSL/TLS connection. When a secure connection is established, the data is encrypted to ensure confidentiality. Wireshark alone cannot decrypt this encrypted data without the private key.

Steps:

1. Use Wireshark or another network capture tool to capture the SSL/TLS encrypted traffic.
    - Go to Wireshark, select the wifi or ethernet or the relevant Network access point. Wireshark will start capturing the browser traffic requests.

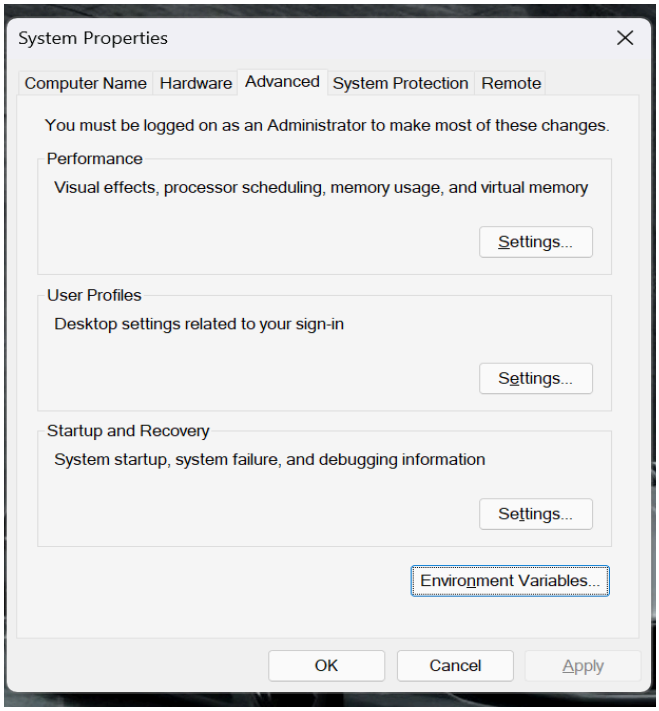


2. Locate the SSL/TLS session we are interested in within Wireshark. We can use the filter option to search for a specific request type, or from a specific port address.

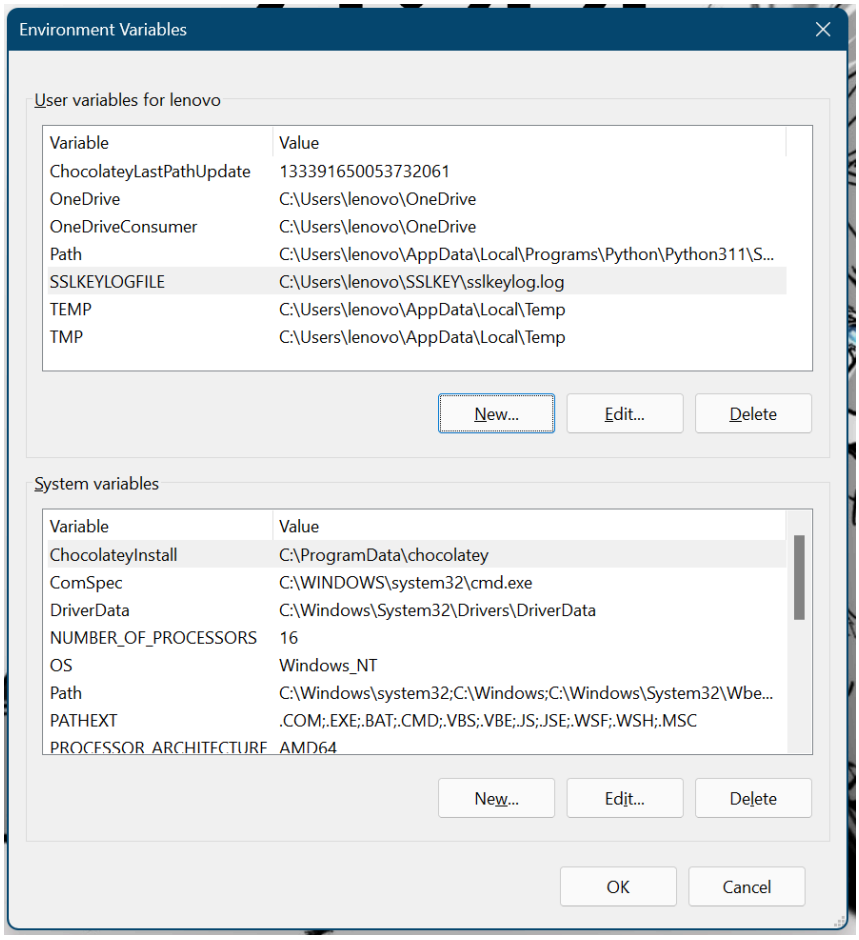


As it is evident, the TCP data is encrypted in hexadecimal. We will require the key to decrypt this.

3. Obtaining the Private key log.
   For this, we need to navigate to the **Edit Environment Variables** dialog box.

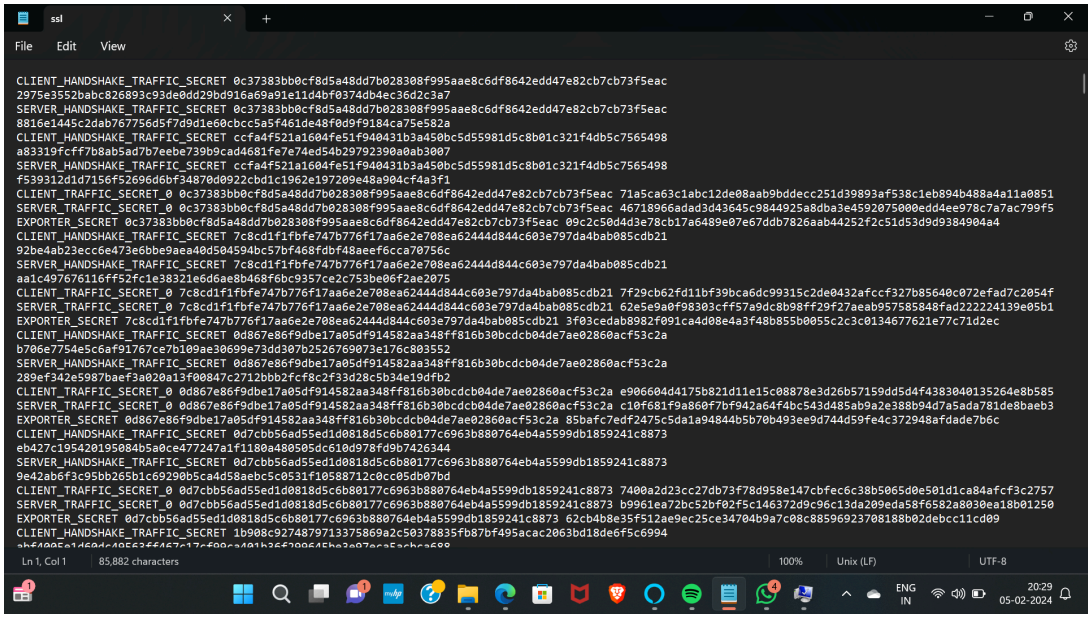


Click on **Environment Variables** then **New** and then create a Variable with name **SSLKEYLOGFILE** and the Variable Path to be anywhere with useraccess.

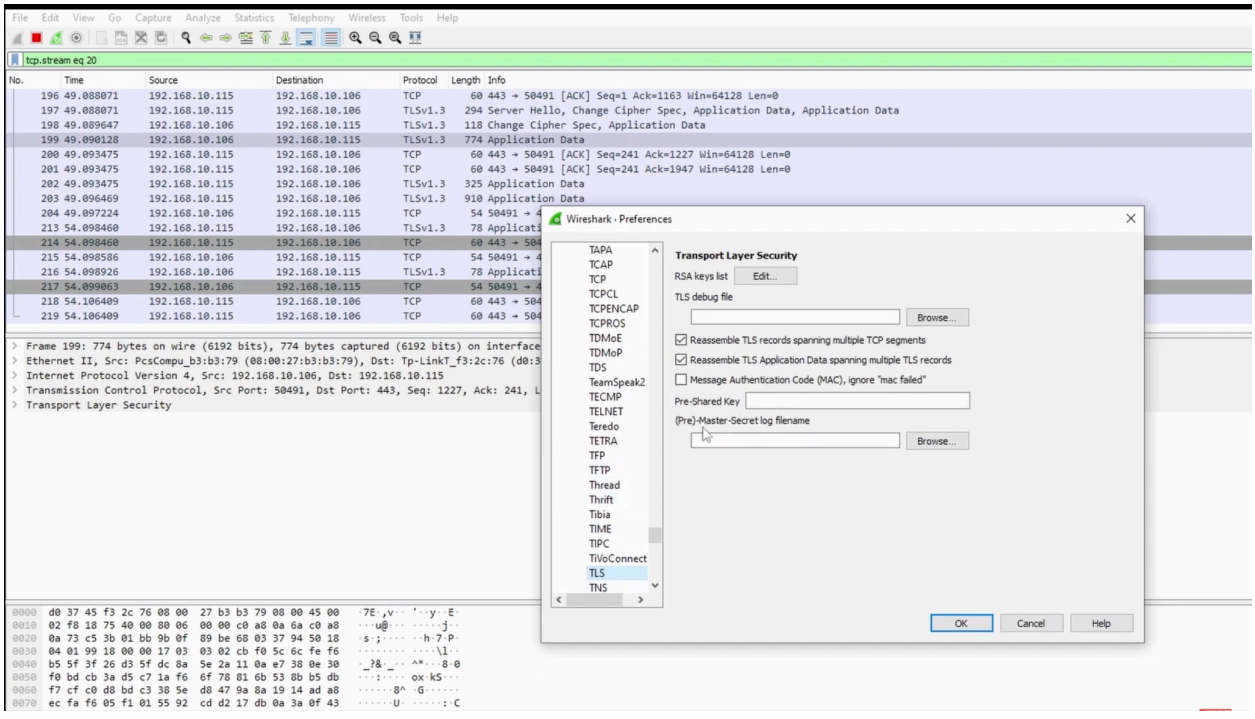


Restart the PC.
Re-open any browser. Let the system catch and store the key log.

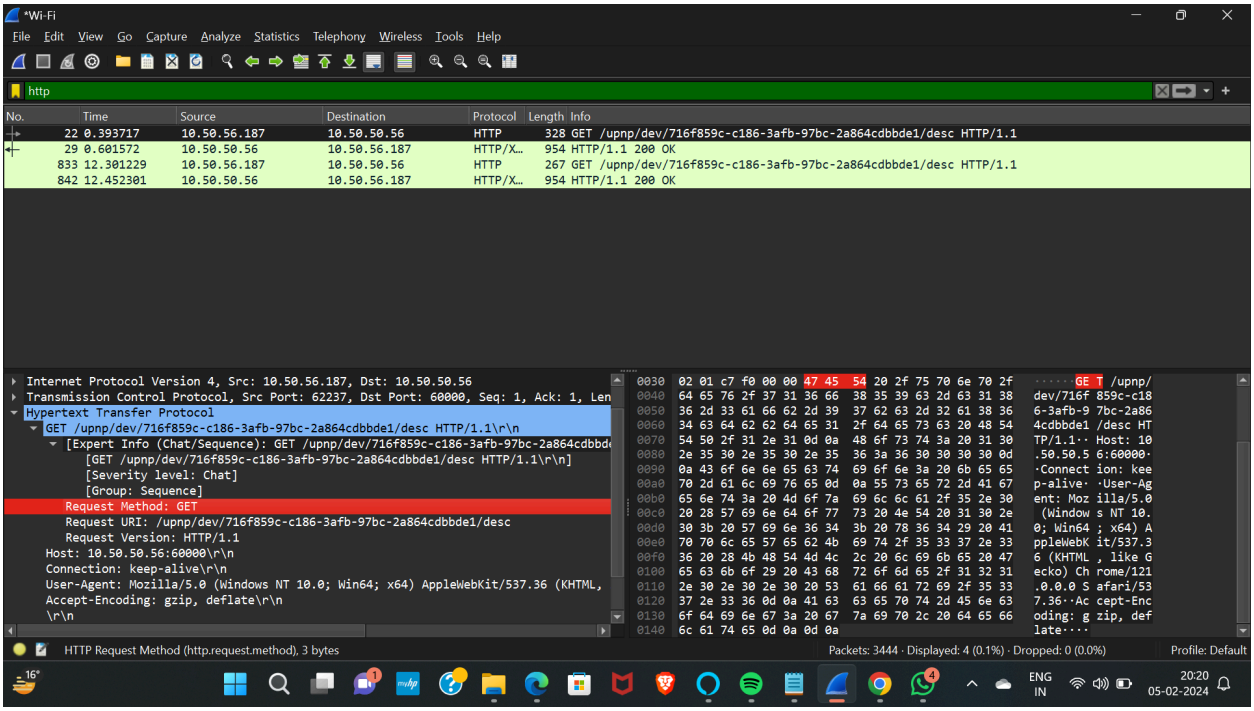
Next, to verify, check the ssl.log file created at the variable path location.



4. Configure Wireshark with the Key:
   Now, in the wireshark window, go to **EDIT->Preferences->Protocol**
   Search for **TLS,** then in the **Master-Secret Log File** option, attach the ssl.log file created.



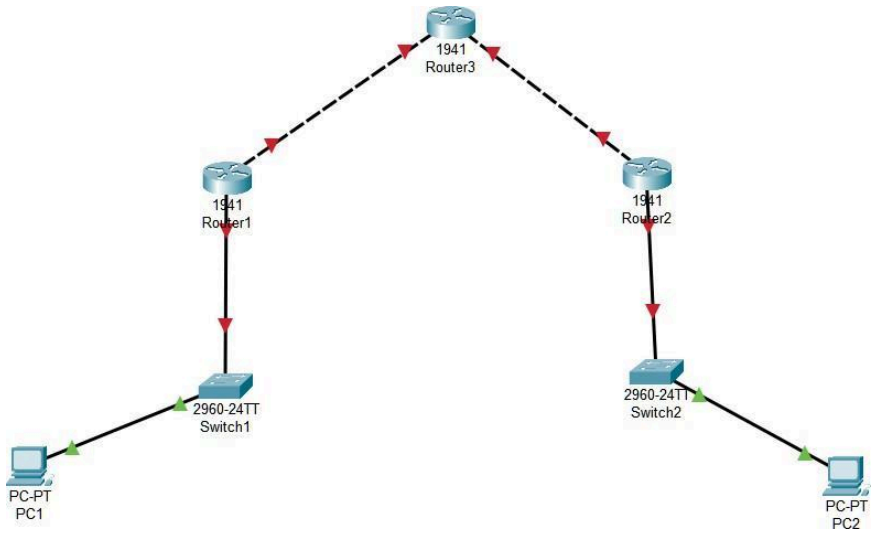
5. Now, search for the HTTP tag and we will find the decrypted HTTP requests next to the encrypted Hexadecimal Text.



Practical 4

Configuring IPSec VPN Tunnel Mode using Packet Tracer

1. Open Cisco Packet Tracer.
2. Create a basic topology with 3 routers, 2 switches and 2 PC's.



3. Configure the routers R1, R2 and R3 as shown in given images: Router1:

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R1
R1(config)#interface g0/1
R1(config-if)#ip address 172.15.1.1 255.255.255.0
R1(config-if)#no shut

R1(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

R1(config-if)#exit
R1(config)#interface g0/0
R1(config-if)#ip address 100.100.100.1 255.255.255.0
R1(config-if)#no shut

R1(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

R1(config-if)#exit
R1(config)#ip route 0.0.0.0 0.0.0.0 100.100.100.2
```

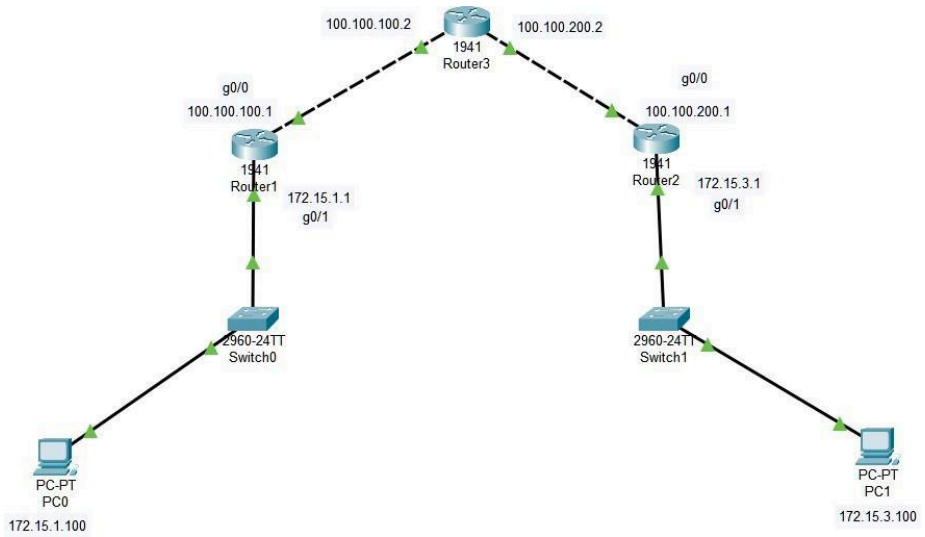
Router2:

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R2
R2(config)#interface g0/1
R2(config-if)#ip address 172.15.3.1 255.255.255.0
R2(config-if)#no shut

R2(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

R2(config-if)#exit
R2(config)#interface g0/0
R2(config-if)#ip address 100.100.200.1 255.255.255.0
R2(config-if)#no shut

R2(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0, changed state to up

R2(config-if)#exit
R2(config)#ip route 0.0.0.0 0.0.0.0 100.100.200.2
```

Router3:

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname R3
R3(config)#interface g0/1
R3(config-if)#ip address 100.100.200.2 255.255.255.0
R3(config-if)#no shut

R3(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

R3(config-if)#exit
R3(config)#interface g0/0
R3(config-if)#ip address 100.100.100.2 255.255.255.0
R3(config-if)#no shut
```

After configuring the routers, the final topology looks like this:



4. Make sure routers R1 and R2 have the security license enabled:

   license boot module c1900 technology-package securityk9

```
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#license boot module c1900 technology-package securityk9
```

Save the configuration using “copy run start” command and reload the router using “reload” command.

Check whether the license is enabled or not using “show version” command.

```
Technology Package License Information for Module:'c1900'

-----------------------------------------------------------------
Technology    Technology-package          Technology-package
              Current       Type          Next reboot
-----------------------------------------------------------------
ipbase        ipbasek9      Permanent     ipbasek9
security      securityk9    Evaluation    securityk9
data          disable       None          None

Configuration register is 0x2102
```

5. The process of setting up IPSec VPN tunnel takes place in following steps:
   i. **ACL (Access list):** To permit the matching traffic (data packets) that will go across the tunnel.
   ii. **ISAKMP policy and ISAKMP key:** These are Phase 1 of IPSec VPN tunneling.
   iii. **IPSec transform set:** This is the Phase 2 of IPSec VPN tunneling.
   iv. Crypto map
   v. Applying the crypto map to the interface.

Step-(i): Creating Access list:

Router1:

```
R1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#access-list 100 permit ip 172.15.1.0 0.0.0.255 172.15.3.0 0.0.0.255
```

Router2:

```
R2#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
R2(config)#access-list 100 permit ip 172.15.3.0 0.0.0.255 172.15.1.0 0.0.0.255
```

Step-(ii): Setting up ISAKMP policy and ISAKMP key:

Router1:

```
R1(config)#crypto isakmp policy 10
R1(config-isakmp)#    encryption aes 256
R1(config-isakmp)#    authentication pre-share
R1(config-isakmp)#    group 5
R1(config-isakmp)#exit
R1(config)#crypto isakmp key secretkey address 100.100.200.1
```

Router2:

```
R2(config)#crypto isakmp policy 10
R2(config-isakmp)#    encryption aes 256
R2(config-isakmp)#    authentication pre-share
R2(config-isakmp)#    group 5
R2(config-isakmp)#exit
R2(config)#crypto isakmp key secretkey address 100.100.100.1
```

Step-(iii): Setting up IPSec transform-set:

Router 1:

```
R1(config)#crypto ipsec transform-set R1->R2 esp-aes 256 esp-sha-hmac
```

Router 2:

```
R2(config)#crypto ipsec transform-set R2->R1 esp-aes 256 esp-sha-hmac
```

Step-(iv) Setting up Crypto map:

Router1:

```
R1(config)#crypto map IPSEC-CRYPTOMAP 100 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
        and a valid access list have been configured.
R1(config-crypto-map)# set peer 100.100.200.1
R1(config-crypto-map)# set pfs group5
R1(config-crypto-map)# set security-association lifetime seconds 86400
R1(config-crypto-map)# set transform-set R1-R2
R1(config-crypto-map)# match address 100
```

Router2:

```
R2(config)#crypto map IPSEC-CRYPTOMAP 100 ipsec-isakmp
% NOTE: This new crypto map will remain disabled until a peer
        and a valid access list have been configured.
R2(config-crypto-map)# set peer 100.100.100.1
R2(config-crypto-map)# set pfs group5
R2(config-crypto-map)# set security-association lifetime seconds 86400
R2(config-crypto-map)# set transform-set R2-R1
R2(config-crypto-map)# match address 100
```

Step-(v) Applying crypto map to the interface:

Router1: (We need to choose the interface going towards the Router3 i.e. going out from the network)

```
R1(config)#interface GigabitEthernet0/0
R1(config-if)# crypto map IPSEC-CRYPTOMAP
*Jan  3 07:16:26.785: %CRYPTO-6-ISAKMP ON OFF: ISAKMP is ON
```

Router2: (We need to choose the interface going towards the Router3 i.e. going out from the network)

```
R2(config)#interface GigabitEthernet0/0
R2(config-if)# crypto map IPSEC-CRYPTOMAP
*Jan  3 07:16:26.785: %CRYPTO-6-ISAKMP ON OFF: ISAKMP is ON
```

6. Try sending data packets from PC-1 to PC-2. The pings may initially fail, but if all configuration is accurate, the pings should succeed after a couple of tries.

```
C:\>ping 172.15.3.100

Pinging 172.15.3.100 with 32 bytes of data:

Request timed out.
Reply from 172.15.3.100: bytes=32 time=1ms TTL=126
Reply from 172.15.3.100: bytes=32 time<1ms TTL=126
Reply from 172.15.3.100: bytes=32 time=15ms TTL=126

Ping statistics for 172.15.3.100:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 15ms, Average = 5ms
```

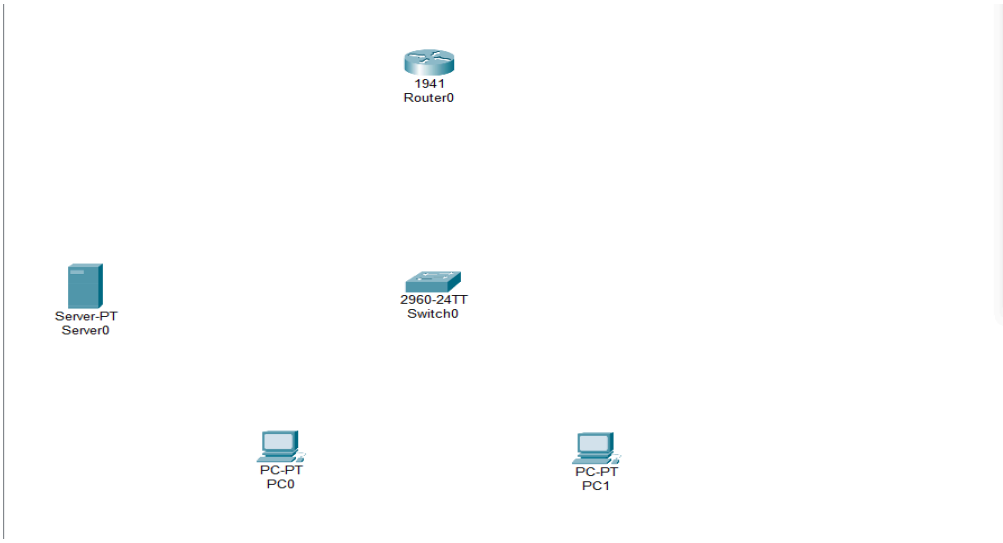
7. Finally, verify that the tunnel is up and running using the following command: “show crypto ipsec sa”.

```
R1#show crypto ipsec sa

interface: GigabitEthernet0/0
    Crypto map tag: IPSEC-CRYPTOMAP, local addr 100.100.100.1

   protected vrf: (none)
   local  ident (addr/mask/prot/port): (172.15.1.0/255.255.255.0/0/0)
   remote  ident (addr/mask/prot/port): (172.15.3.0/255.255.255.0/0/0)
   current_peer 100.100.200.1 port 500
    PERMIT, flags={origin_is_acl,}
   #pkts encaps: 7, #pkts encrypt: 7, #pkts digest: 0
   #pkts decaps: 7, #pkts decrypt: 7, #pkts verify: 0
   #pkts compressed: 0, #pkts decompressed: 0
   #pkts not compressed: 0, #pkts compr. failed: 0
   #pkts not decompressed: 0, #pkts decompress failed: 0
   #send errors 1, #recv errors 0

     local crypto endpt.: 100.100.100.1, remote crypto endpt.:100.100.200.1
     path mtu 1500, ip mtu 1500, ip mtu idb GigabitEthernet0/0
     current outbound spi: 0xFBAA9DB5(4222262709)

     inbound esp sas:
      spi: 0x112B4B17(288049943)
        transform: esp-aes 256 esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2009, flow_id: FPGA:1, crypto map: IPSEC-CRYPTOMAP
        sa timing: remaining key lifetime (k/sec): (4525504/86284)
        IV size: 16 bytes
        replay detection support: N
        Status: ACTIVE

     inbound ah sas:

     inbound pcp sas:

     outbound esp sas:
      spi: 0xFBAA9DB5(4222262709)
        transform: esp-aes 256 esp-sha-hmac ,
        in use settings ={Tunnel, }
        conn id: 2010, flow_id: FPGA:1, crypto map: IPSEC-CRYPTOMAP
        sa timing: remaining key lifetime (k/sec): (4525504/86284)
        IV size: 16 bytes
        replay detection support: N
        Status: ACTIVE

     outbound ah sas:

     outbound pcp sas:
```

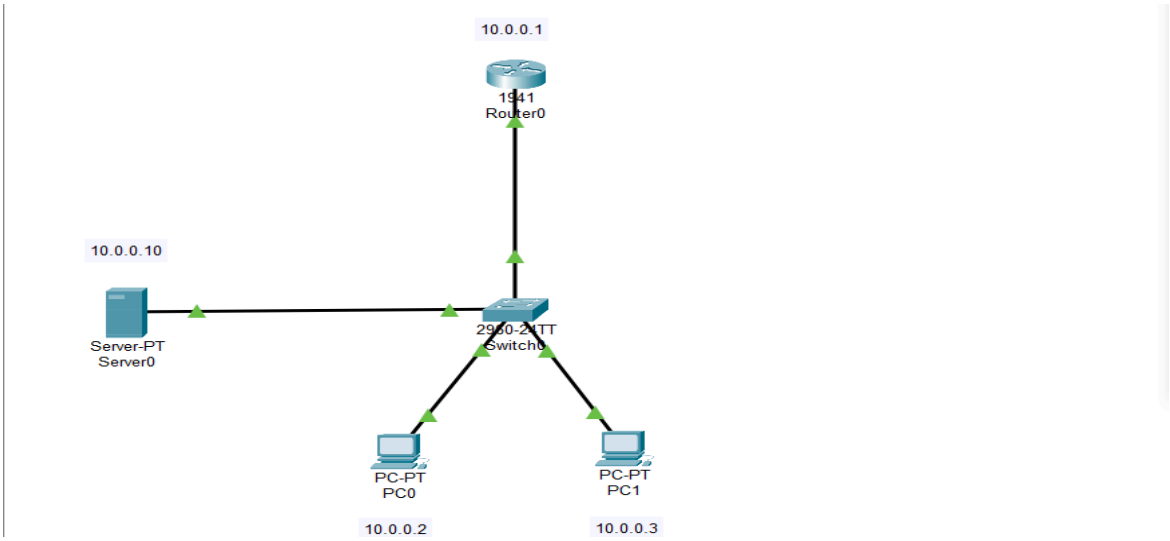
Practical 5

To Configure AAA (TACACS+) on Packet Tracer for User Authentication.

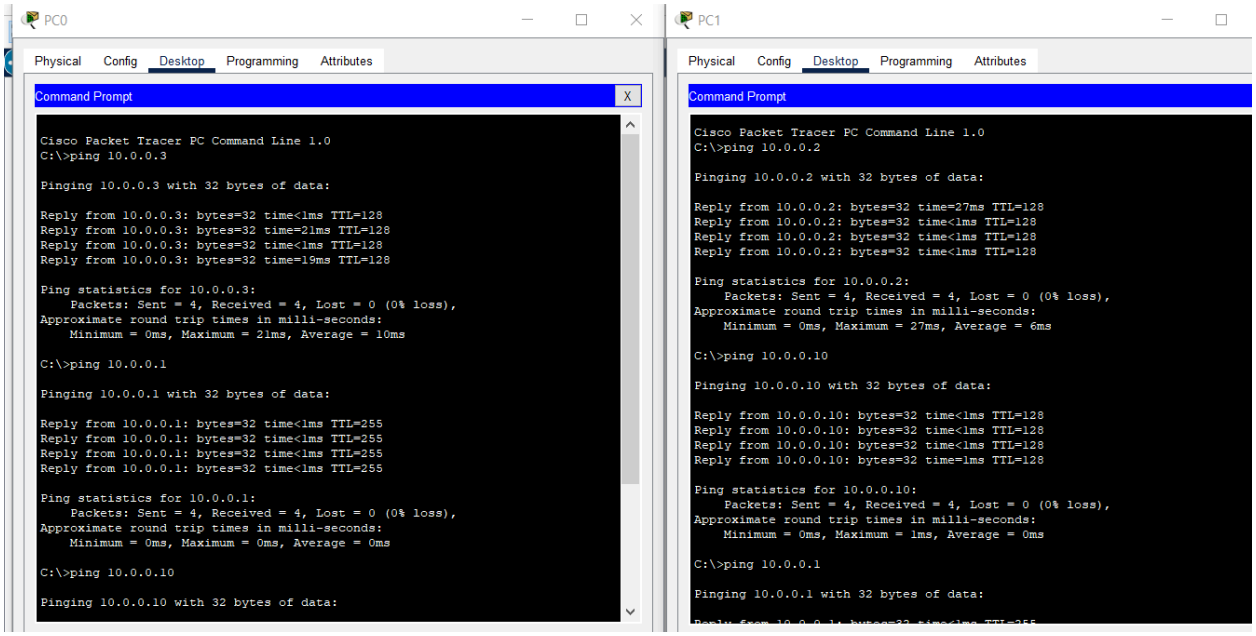
1) Open Cisco Packet Tracer and add the following devices to form a basic network: a server, a router, a switch and a few PCs



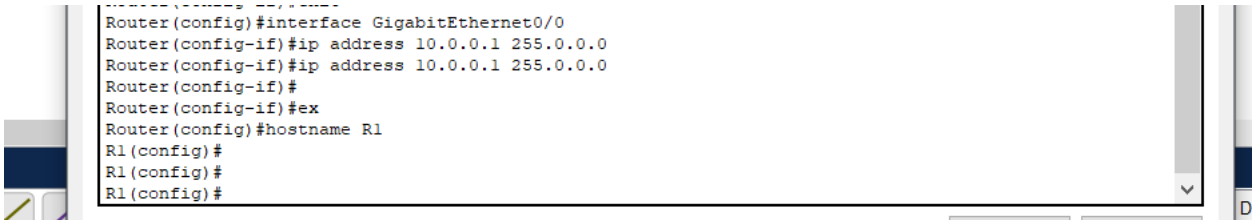
2) Connect these devices with connecting wires and also give them appropriate IP Addresses



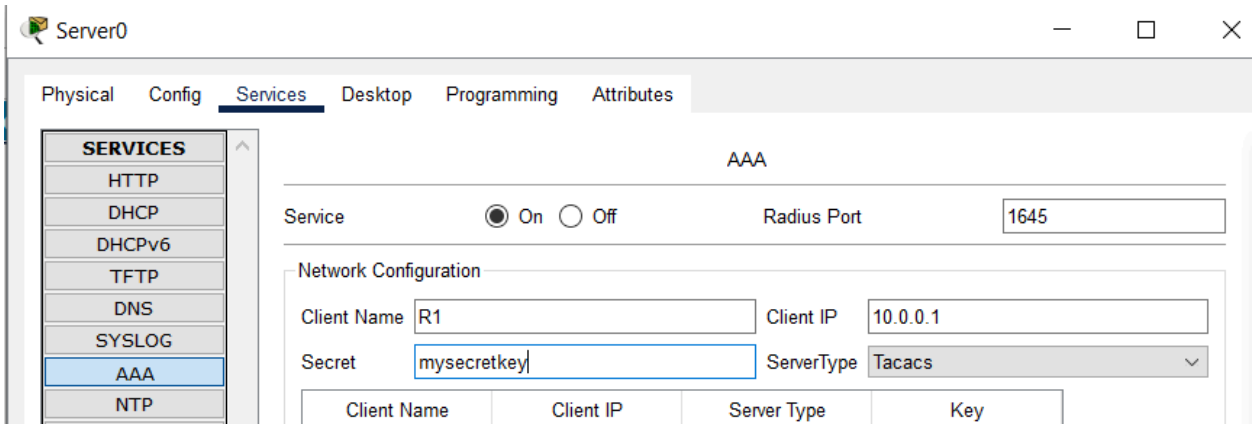
3) Check if the network is working properly, and devices can communicate with other devices in the network



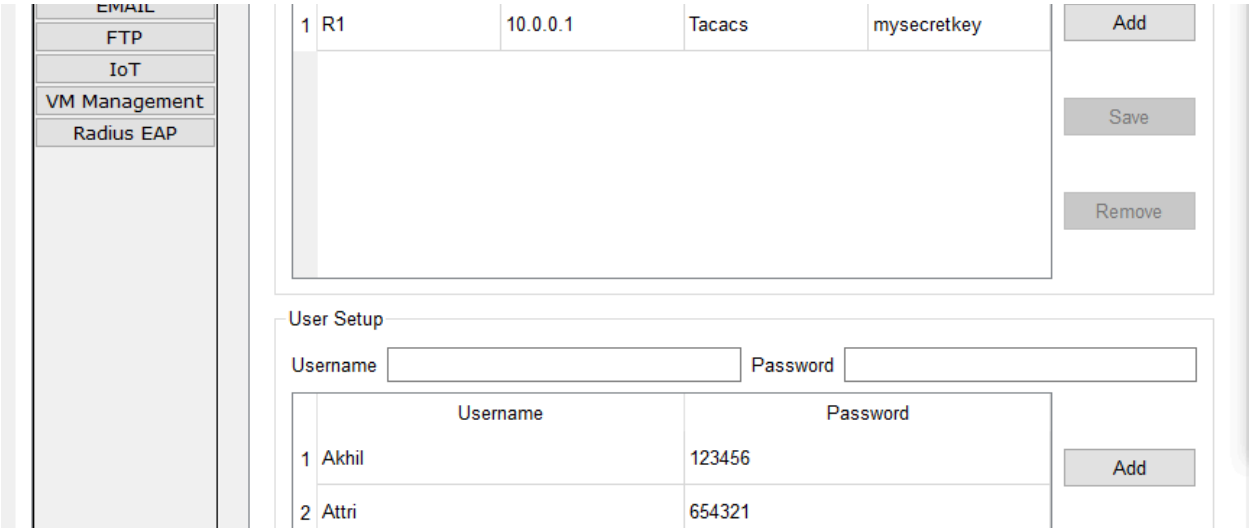
4) Set a hostname for the router, say R1



5) Enter the Services menu on the server. Select the AAA option and turn the service on. In Network Configuration, enter the router details, like the name, a passkey, and IP address



6) In the User Setup section, create user logins, to communicate from the remote PCs



7) Enable aaa authentication in the router's shell, setup a password for enabling the router shell via the remote network

```
Rl(config)#
Rl(config)#aaa new-model
Rl(config)#aaa authentication login  ?
  WORD     Named authentication list.
  default  The default authentication list.
Rl(config)#aaa authentication login myauth group ta
Rl(config)#aaa authentication login myauth group tacacs+ local
Rl(config)#tac
Rl(config)#tacacs-server host ?
  A.B.C.D  IP address of TACACS server
Rl(config)#tacacs-server host
% Incomplete command.
Rl(config)#tacacs-server host 10.0.0.10 key mysecretkey
Rl(config)#enable pass
Rl(config)#enable password c
Rl(config)#enable password cisco
Rl(config)#
```

8) Save the configuration

```
Rl(config)#
Rl(config)#
Rl(config)#line vty 04
Rl(config-line)#login
AAA is enabled. Command not supported. Use an aaa authentication methodlist
Rl(config-line)#login ?
  authentication  authenticate using aaa method list
  local           Local password checking
  <cr>
Rl(config-line)#login
AAA is enabled. Command not supported. Use an aaa authentication methodlist
Rl(config-line)#login aut
Rl(config-line)#login authentication myauth
Rl(config-line)#
Rl(config-line)#
Rl(config-line)#do wr
Building configuration...
[OK]
```

9) Use show aaa sessions command to check for number of sessions that have existed

```
R1#show aaa s
R1#show aaa sessions
Total sessions since last reload: 0
R1#
```

Run the show running-config (sh ru) command to view current operational settings and confirm that AAA is running

```
R1#sh ru
Building configuration...

Current configuration : 793 bytes
!
version 15.1
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
!
hostname R1
!
!
!
enable password cisco
!
!
!
!
!
aaa new-model
!
aaa authentication login myauth group tacacs+ local
```

```
tacacs-server host 10.0.0.10 key mysecretkey
!
!
!
line con 0
!
line aux 0
!
line vty 0 3
line vty 4
 login authentication myauth
!
!
!
end


R1#
R1#
R1#
R1#
R1#
R1#
R1#
R1#
R1#
R1#
R1#
R1#
R1#
R1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R1(config)#line vty 0 4
R1(config-line)#login authentication myauth
R1(config-line)#exit
R1(config)#
```

Copy Paste

10) To start using AAA, enter one of the PCs cm, and try to connect using the telnet command. Enter the Username and Password to login. Enter the enable password.

```
C:\>telnet 10.0.0.1
Trying 10.0.0.1 ...Open


User Access Verification

Username: Akhil
Password:
R1>en
Password:
Password:
Password:
R1#
R1#
R1#sh ru
```

11) Run the show running-config (also sh ru) command to view current operational settings and confirm that AAA is running

PC0

Physical | Config | Desktop | Programming | Attributes

Command Prompt

```
R1#sh ru
Building configuration...

Current configuration : 782 bytes
!
version 15.1
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
!
hostname R1
!
!
!
enable password cisco
!
!
!
!
!
aaa new-model
!
aaa authentication login myauth group tacacs+ local
!
!
!
tacacs-server host 10.0.0.10 key mysecretkey
!
!
!
line con 0
!
line aux 0
!
line vty 0 4
 login authentication myauth
!
!
!
end
```

12) In the router terminal, use show aaa sessions to confirm if the connection shows up

```
%SYS-5-CONFIG_I: Configured from console by console

R1#show aaa sess
R1#show aaa sessions
Total sessions since last reload: 1
Session Id:4
           Unique Id:2
           User Name:Akhil
           IP Address:10.0.0.2
           Idle Time: 0
           CT Call Handle: 0
R1#
```

Copy Paste

13) Connect using other PCs in the network in the same manner and run sh ru command

PC1

Physical Config Desktop Programming Attributes

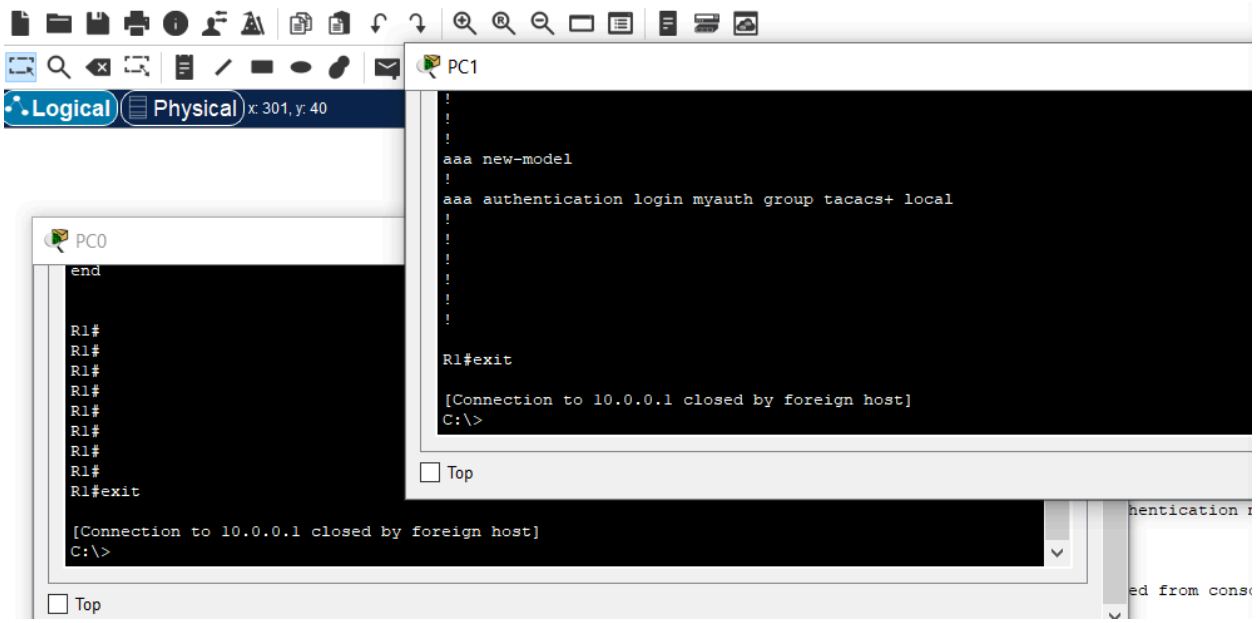
Command Prompt X

```
C:\>
C:\>telnet 10.0.0.1
Trying 10.0.0.1 ...Open


User Access Verification

Username: Attri
Password:
R1>
R1>
R1>en
Password:
R1#
R1#
R1#sh ru
Building configuration...

Current configuration : 782 bytes
!
version 15.1
no service timestamps log datetime msec
no service timestamps debug datetime msec
no service password-encryption
!
hostname R1
!
!
!
enable password cisco
!
!
!
!
!
aaa new-model
!
aaa authentication login myauth group tacacs+ local
!
!
```

14) The Router terminal confirms that two connections have been made

```
R1#show aaa sessions
Total sessions since last reload: 2
Session Id:4
           Unique Id:2
           User Name:Akhil
           IP Address:10.0.0.2
           Idle Time: 0
           CT Call Handle: 0
Session Id:4
           Unique Id:4
           User Name:Attri
           IP Address:10.0.0.2
           Idle Time: 0
           CT Call Handle: 0
R1#
```

15) Now terminate the connection in both devices



16) As Expected, the router's terminal reflects the same, no active connection is shown

```
R1#show aaa sessions
Total sessions since last reload: 2
R1#
R1#
R1#
R1#
```