LAB MANUAL

Course: Network Security and Cryptography Lab (MCS122)

Institute: Amity Institute of Information Technology

SUBMITTED BY:

AJINKYA KOLHE
A217131524023
MSc. Cyber security
Semester - 1
Batch A

SUBMITTED TO:

Dr. Vishal Sharma

TABLE OF CONTENTS

Experiment 1

Aim:

Common Netstat command parameters and outputs.

Theory:

Netstat (Network Statistics) is a command-line tool used to display active network connections, routing tables, and various network interface statistics. It provides insights into which ports and IP addresses are actively communicating with the system, helping network administrators and security professionals monitor network activity and diagnose connectivity issues. The command can show information such as current TCP connections, listening ports, and the status of these connections, which is valuable for troubleshooting and network analysis.

Procedure

Netstat

Open Command Prompt (CMD)

Type “Netstat”

```
Microsoft Windows [Version 10.0.25179.1000]
(c) Microsoft Corporation. All rights reserved.

C:\Users\seth_>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:9843         THEnc:49671            ESTABLISHED
  TCP    127.0.0.1:9843         THEnc:49678            ESTABLISHED
  TCP    127.0.0.1:9843         THEnc:49680            ESTABLISHED
  TCP    127.0.0.1:9843         THEnc:49681            ESTABLISHED
  TCP    127.0.0.1:49671        THEnc:9843             ESTABLISHED
  TCP    127.0.0.1:49678        THEnc:9843             ESTABLISHED
  TCP    127.0.0.1:49680        THEnc:9843             ESTABLISHED
  TCP    127.0.0.1:49681        THEnc:9843             ESTABLISHED
  TCP    192.168.64.3:49674     8.241.154.126:http     ESTABLISHED
  TCP    192.168.64.3:49675     117.18.237.29:http     ESTABLISHED
  TCP    192.168.64.3:49676     52.191.219.104:https   TIME_WAIT
  TCP    192.168.64.3:49682     52.137.102.105:https   TIME_WAIT
  TCP    192.168.64.3:49683     20.189.173.4:https     TIME_WAIT
^C
```

Type “ Netstat /?” For viewing help to see various parameters/ Flags

```
C:\Users\seth_>netstat /?

Displays protocol statistics and current TCP/IP network connections.

NETSTAT [-a] [-b] [-e] [-f] [-i] [-n] [-o] [-p proto] [-r] [-s] [-t] [-x] [-y] [interval]

  -a            Displays all connections and listening ports.
  -b            Displays the executable involved in creating each connection or
                listening port. In some cases well-known executables host
                multiple independent components, and in these cases the
                sequence of components involved in creating the connection
                or listening port is displayed. In this case the executable
                name is in [] at the bottom, on top is the component it called,
                and so forth until TCP/IP was reached. Note that this option
                can be time-consuming and will fail unless you have sufficient
                permissions.
  -e            Displays Ethernet statistics. This may be combined with the -s
                option.
  -f            Displays Fully Qualified Domain Names (FQDN) for foreign
                addresses.
  -i            Displays the time spent by a TCP connection in its current state.
  -n            Displays addresses and port numbers in numerical form.
  -o            Displays the owning process ID associated with each connection.
  -p proto      Shows connections for the protocol specified by proto; proto
                may be any of: TCP, UDP, TCPv6, or UDPv6.  If used with the -s
                option to display per-protocol statistics, proto may be any of:
```

Some of the Parameters

a : All Connections

```
C:\Users\seth_>netstat -a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            THEnc:0                LISTENING
  TCP    0.0.0.0:445            THEnc:0                LISTENING
  TCP    0.0.0.0:5040           THEnc:0                LISTENING
  TCP    0.0.0.0:7680           THEnc:0                LISTENING
  TCP    0.0.0.0:49664          THEnc:0                LISTENING
  TCP    0.0.0.0:49665          THEnc:0                LISTENING
  TCP    0.0.0.0:49666          THEnc:0                LISTENING
  TCP    0.0.0.0:49667          THEnc:0                LISTENING
  TCP    0.0.0.0:49668          THEnc:0                LISTENING
  TCP    0.0.0.0:49669          THEnc:0                LISTENING
  TCP    127.0.0.1:9843         THEnc:0                LISTENING
  TCP    127.0.0.1:49672        THEnc:9843             TIME_WAIT
  TCP    127.0.0.1:49679        THEnc:9843             TIME_WAIT
  TCP    127.0.0.1:49681        THEnc:9843             TIME_WAIT
  TCP    127.0.0.1:49682        THEnc:9843             TIME_WAIT
  TCP    192.168.64.3:139       THEnc:0                LISTENING
  TCP    192.168.64.3:49675     8.241.154.254:http     TIME_WAIT
  TCP    192.168.64.3:49676     117.18.237.29:http     TIME_WAIT
  TCP    192.168.64.3:49695     20.198.119.143:https   ESTABLISHED
  TCP    192.168.64.3:49701     a-0001:https           ESTABLISHED
  TCP    192.168.64.3:49702     117.18.237.29:http     ESTABLISHED
```

e : Ethernet Stats

```
C:\Users\seth_>netstat -e
Interface Statistics

                           Received            Sent

Bytes                     539303292        16811252
Unicast packets              384500          144884
Non-unicast packets              44             492
Discards                          0               0
Errors                            0               0
Unknown protocols                 0
```

i : Time spent by TCP Connection

```
C:\Users\seth_>netstat -i

Active Connections

  Proto  Local Address          Foreign Address        State           Time in State (ms)
TCP    192.168.64.3:49695     20.198.119.143:https   ESTABLISHED         359017
TCP    192.168.64.3:49715     40.99.31.178:https     ESTABLISHED         353163
TCP    192.168.64.3:49789     20.198.119.143:https   ESTABLISHED         299247
TCP    192.168.64.3:49846     20.190.146.36:https    TIME_WAIT           118919
TCP    192.168.64.3:49847     20.44.10.122:https     TIME_WAIT           118475
TCP    192.168.64.3:49850     20.190.145.171:https   TIME_WAIT           118909
TCP    192.168.64.3:49856     52.109.56.83:https     TIME_WAIT           116747
TCP    192.168.64.3:49857     52.139.176.199:https   TIME_WAIT           116669
TCP    192.168.64.3:49858     20.198.119.143:https   ESTABLISHED         226330
```

Conclusion

We have understood thecommon netstat command parameters and outputs.

Experiment 2

Aim:

Use netstat to examine protocol information on a pod host computer

Theory:

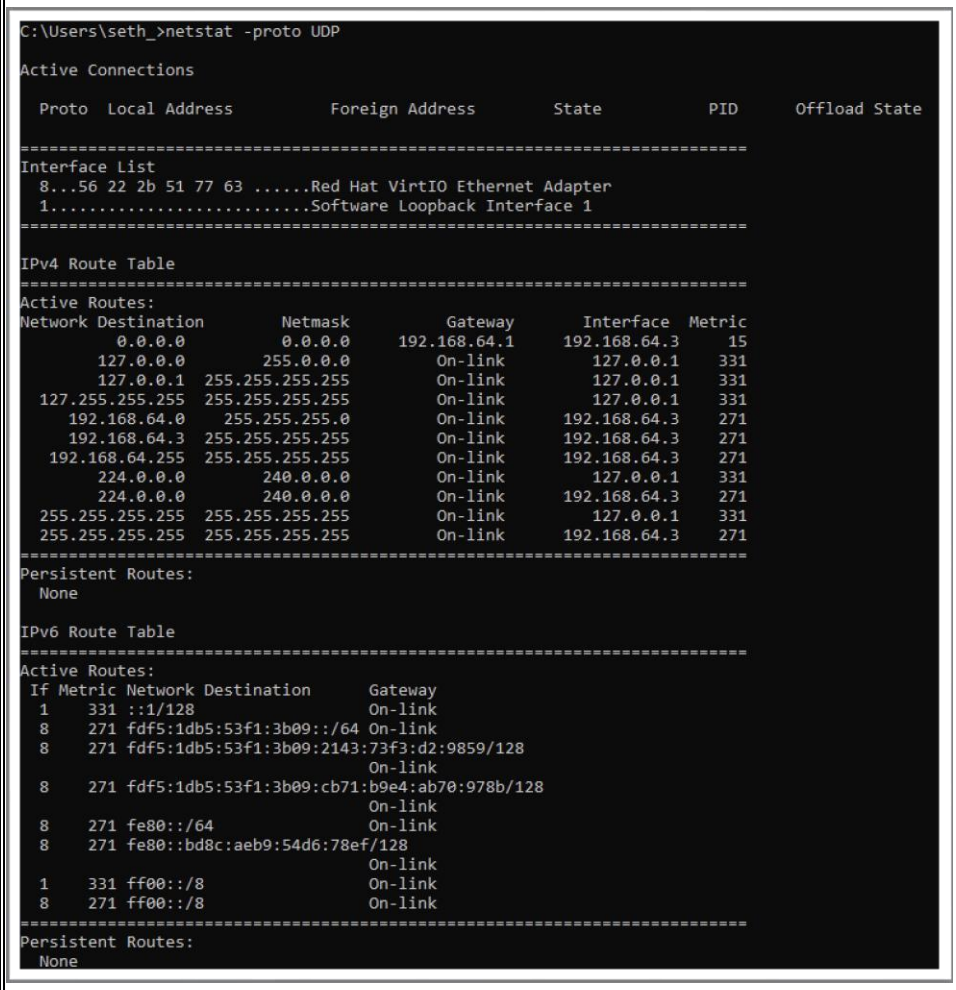
Using **Netstat** to examine protocol information on a pod host computer allows administrators to monitor active network connections, listening ports, and protocol-specific statistics (such as TCP, UDP, or ICMP). This helps identify communication patterns, troubleshoot connectivity issues, and detect unusual activity. It is particularly useful in containerized environments to analyze how pods interact with other network components.

Procedure:

Examine protocol Information of POD Host Computer:

UDP Protocol

“netstat -proto UDP”



```
C:\Users\seth_>netstat -proto UDP

Active Connections

  Proto  Local Address          Foreign Address        State           PID      Offload State

===========================================================================
Interface List
  8...56 22 2b 51 77 63 ......Red Hat VirtIO Ethernet Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     192.168.64.1     192.168.64.3     15
        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
     192.168.64.0    255.255.255.0         On-link      192.168.64.3    271
     192.168.64.3  255.255.255.255         On-link      192.168.64.3    271
   192.168.64.255  255.255.255.255         On-link      192.168.64.3    271
        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      192.168.64.3    271
  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      192.168.64.3    271
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
  8    271 fdf5:1db5:53f1:3b09::/64 On-link
  8    271 fdf5:1db5:53f1:3b09:2143:73f3:d2:9859/128
                                    On-link
  8    271 fdf5:1db5:53f1:3b09:cb71:b9e4:ab70:978b/128
                                    On-link
  8    271 fe80::/64                On-link
  8    271 fe80::bd8c:aeb9:54d6:78ef/128
                                    On-link
  1    331 ff00::/8                 On-link
  8    271 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

TCP Protocol

“netstat -proto TCP”

```
C:\Users\seth_>netstat -proto TCP
Active Connections

  Proto  Local Address          Foreign Address        State           PID      Offload State

  TCP    127.0.0.1:49921        THEnc:9843             TIME_WAIT       0        InHost
  TCP    192.168.64.3:49695     20.198.119.143:https   ESTABLISHED     3156     InHost
  TCP    192.168.64.3:49715     40.99.31.178:https     ESTABLISHED     5964     InHost
  TCP    192.168.64.3:49789     20.198.119.143:https   ESTABLISHED     3156     InHost
  TCP    192.168.64.3:49858     20.198.119.143:https   ESTABLISHED     3960     InHost
  TCP    192.168.64.3:49895     117.18.237.29:http     CLOSE_WAIT      5964     InHost
  TCP    192.168.64.3:49915     20.44.10.122:https     TIME_WAIT       0        InHost
  TCP    192.168.64.3:49935     52.231.199.126:https   TIME_WAIT       0        InHost
  TCP    192.168.64.3:49943     8.241.154.126:http     TIME_WAIT       0        InHost
  TCP    192.168.64.3:49944     20.227.52.206:https    TIME_WAIT       0        InHost
  TCP    192.168.64.3:49946     8.241.154.126:http     TIME_WAIT       0        InHost
===========================================================================
Interface List
  8...56 22 2b 51 77 63 ......Red Hat VirtIO Ethernet Adapter
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     192.168.64.1    192.168.64.3     15
        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
     192.168.64.0    255.255.255.0         On-link      192.168.64.3    271
     192.168.64.3  255.255.255.255         On-link      192.168.64.3    271
   192.168.64.255  255.255.255.255         On-link      192.168.64.3    271
        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      192.168.64.3    271
  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      192.168.64.3    271
===========================================================================
Persistent Routes:
  None
```

```
IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    331 ::1/128                  On-link
  8    271 fdf5:1db5:53f1:3b09::/64 On-link
  8    271 fdf5:1db5:53f1:3b09:2143:73f3:d2:9859/128
                                    On-link
  8    271 fdf5:1db5:53f1:3b09:cb71:b9e4:ab70:978b/128
                                    On-link
  8    271 fe80::/64                On-link
  8    271 fe80::bd8c:aeb9:54d6:78ef/128
                                    On-link
  1    331 ff00::/8                 On-link
  8    271 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

Experiment 3 :

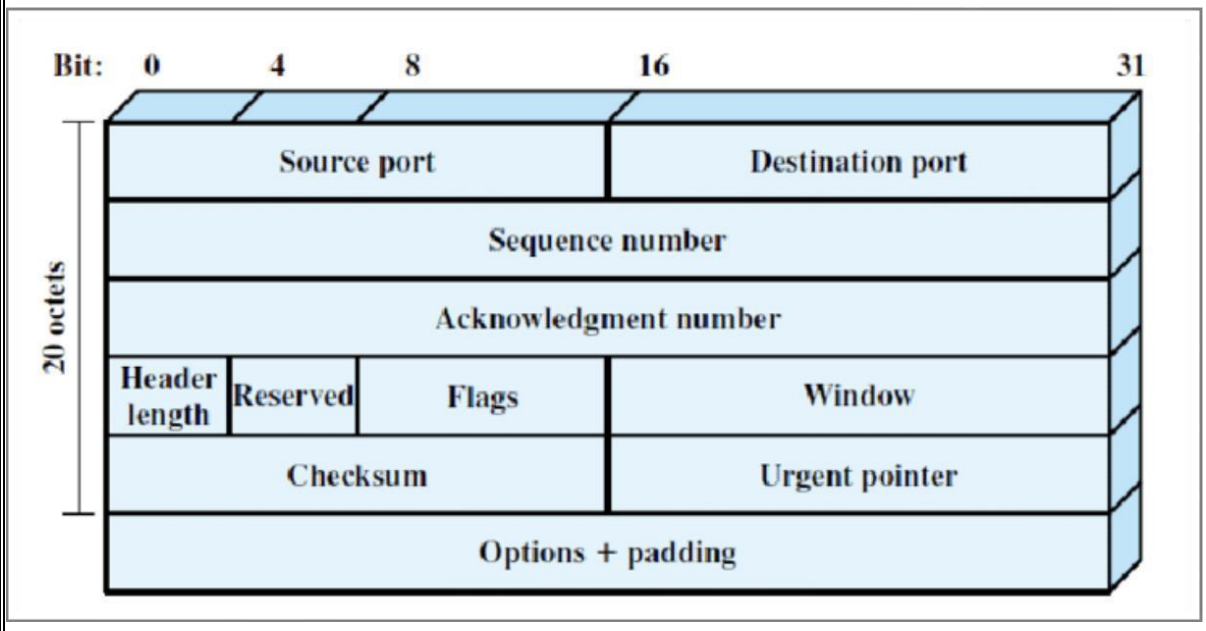
Aim :

Identify TCP header fields and operation

Theory

Transmission control Protocol

- It is connection oriented
- It's slow as it connects using " 3-Way Handshake "
- Uses advance error checking
- Protocols include HTTP , HTTPs, FTP, SMTP etc.
- Header Size : 20 Bytes



Procedure

- Download and Install Wireshark from " *https://www.wireshark.org/* " .
- Open Wireshark (Click on Icon or type on terminal)
- Select Interface " en 0 "
- Press the " Red Block " to stop capturing Packets
  Type the protocol (TCP) to filter packets Click on any packet with " Protocol : *TCP* "

- On the Left side corner/ footer, select " *Open capture file properties …* "

File

Name:	/var/folders/ft/rvkn9hb937gcd51bgcjf0dvm0000gn/T/wireshark_Wi-FiEM9AW1.pcapng
Length:	5714 kB
Hash (SHA256):	8c1db5b02139a47a899396d0e665c9bbd99b430bdc4b08c8651cc39d97d441c9
Hash (RIPEMD160):	2f21911454279d8df96206d891998dca8ebb391f
Hash (SHA1):	22ea658e893eb46f59e129308d4f4a1f8db5f153
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2022-12-04 01:14:44
Last packet:	2022-12-04 01:14:52
Elapsed:	00:00:07

Capture

Hardware:	Unknown
OS:	macOS 13.0, build 22A380 (Darwin 22.1.0)
Application:	Dumpcap (Wireshark) 3.6.7 (v3.6.7-0-g4a304d7ec222)

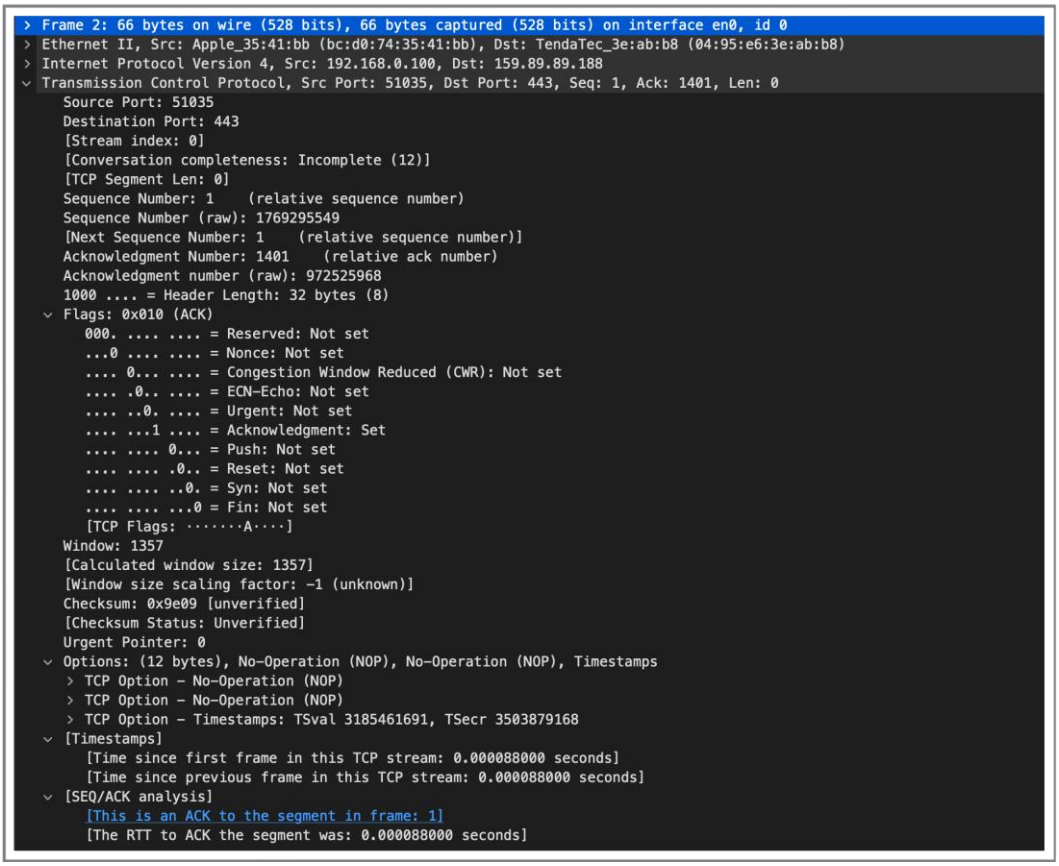
Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Wi-Fi	0 (0.0%)	none	Ethernet	524288 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	6292	6288 (99.9%)	—
Time span, s	7.546	7.546	—
Average pps	833.8	833.3	—
Average packet size, B	874	875	—
Bytes	5500101	5499799 (100.0%)	0
Average bytes/s	728 k	728 k	—
Average bits/s	5831 k	5830 k	—

Contents of TCP Packets



```
> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface en0, id 0
> Ethernet II, Src: Apple_35:41:bb (bc:d0:74:35:41:bb), Dst: TendaTec_3e:ab:b8 (04:95:e6:3e:ab:b8)
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 159.89.89.188
v Transmission Control Protocol, Src Port: 51035, Dst Port: 443, Seq: 1, Ack: 1401, Len: 0
    Source Port: 51035
    Destination Port: 443
    [Stream index: 0]
    [Conversation completeness: Incomplete (12)]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 1769295549
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 1401    (relative ack number)
    Acknowledgment number (raw): 972525968
    1000 .... = Header Length: 32 bytes (8)
  v Flags: 0x010 (ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A····]
    Window: 1357
    [Calculated window size: 1357]
    [Window size scaling factor: -1 (unknown)]
    Checksum: 0x9e09 [unverified]
    [Checksum Status: Unverified]
    Urgent Pointer: 0
  v Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    > TCP Option - No-Operation (NOP)
    > TCP Option - No-Operation (NOP)
    > TCP Option - Timestamps: TSval 3185461691, TSecr 3503879168
  v [Timestamps]
      [Time since first frame in this TCP stream: 0.000088000 seconds]
      [Time since previous frame in this TCP stream: 0.000088000 seconds]
  v [SEQ/ACK analysis]
      [This is an ACK to the segment in frame: 1]
      [The RTT to ACK the segment was: 0.000088000 seconds]
```

Conclusion

We have identified TCP header fields and its operation.

Experiment 4

Aim :

Identify UDP header fields and operation

Procedure:

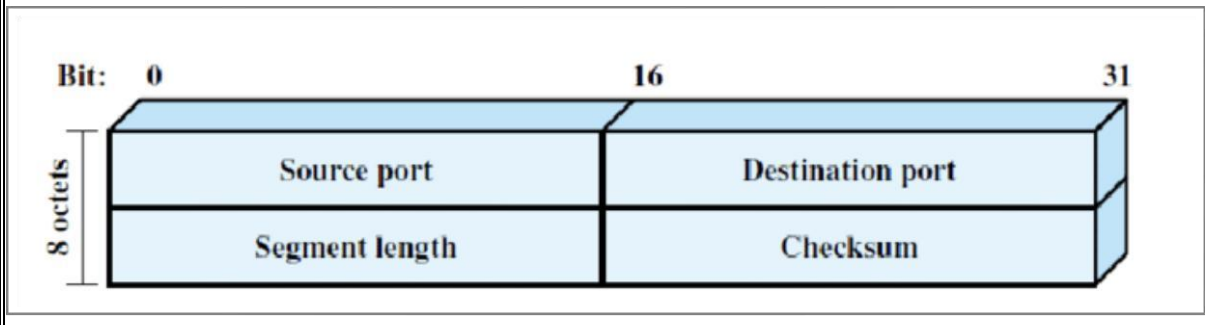
User Datagram Protocol:

It’s connection-less protocols

It’s faster than TCP as it lacks “ Feedback Mechanism”

Uses basic error checking Protocols includes DNS, DHCP, TFTP etc.

Header Size: 8 Bytes’



Procedure:

- Download and Install Wireshark from “ *https://www.wireshark.org/ “* .
- Open Wireshark (Click on Icon or type on
- terminal)Select Interface “ en 0 “
- Press the “ Red Block ” to stop capturing
- PacketsType the protocol (UPD) to filter
- packets Click on any packet with “ Protocol :

  UDP ”
- On the Left side corner/ footer, select “ *Open capture file properties …* ”

File

Name:	/var/folders/ft/rvkn9hb937gcd51bgcjf0dvm0000gn/T/wireshark_Wi-FiEM9AW1.pcapng
Length:	5714 kB
Hash (SHA256):	8c1db5b02139a47a899396d0e665c9bbd99b430bdc4b08c8651cc39d97d441c9
Hash (RIPEMD160):	2f21911454279d8df96206d891998dca8ebb391f
Hash (SHA1):	22ea658e893eb46f59e129308d4f4a1f8db5f153
Format:	Wireshark/... - pcapng
Encapsulation:	Ethernet

Time

First packet:	2022-12-04 01:14:44
Last packet:	2022-12-04 01:14:52
Elapsed:	00:00:07

Capture

Hardware:	Unknown
OS:	macOS 13.0, build 22A380 (Darwin 22.1.0)
Application:	Dumpcap (Wireshark) 3.6.7 (v3.6.7-0-g4a304d7ec222)

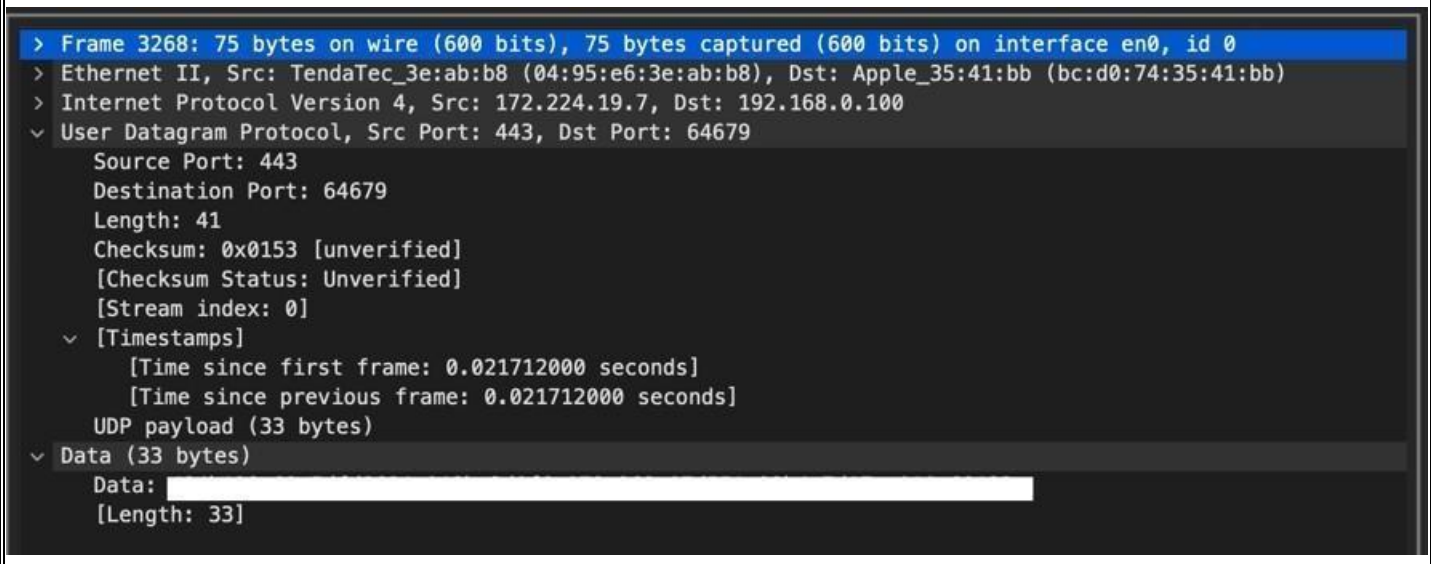
Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
Wi-Fi	0 (0.0%)	none	Ethernet	524288 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	6292	6288 (99.9%)	–
Time span, s	7.546	7.546	–
Average pps	833.8	833.3	–
Average packet size, B	874	875	–
Bytes	5500101	5499799 (100.0%)	0
Average bytes/s	728 k	728 k	–
Average bits/s	5831 k	5830 k	–

Contents of TCP / UDP Packets



Conclusion

We have successfully identified UDP header fields and its operation.

Experiment 5

Aim :

Understand and explain the purpose of a gateway address

Theory:

A **gateway address** is a critical component in computer networks, acting as a bridge between devices within a local network (LAN) and external networks, such as the internet. Typically, this is the IP address of a router or modem within a network.

1. **Facilitating Communication Outside the LAN**:
   Devices on the same LAN communicate directly using their IP addresses. However, when a device needs to communicate with a device on a different network (e.g., accessing a website), it sends the request to the gateway address. The gateway routes this request to the appropriate destination, enabling seamless communication between networks.
2. **Traffic Routing**:
   The gateway serves as the default path for outgoing network traffic. When a computer or device cannot determine the specific route for a data packet, it sends the packet to the gateway. The gateway determines the best path for the packet to reach its destination.
3. **Network Security and Control**:
   Gateways often act as checkpoints, enforcing security policies like firewalls and NAT (Network Address Translation). They help protect the local network by masking internal IP addresses and blocking unauthorized access.
4. **Connectivity to the Internet**:
   In most home or business networks, the gateway address is assigned to the router. The router connects the LAN to the wider internet, enabling users to browse websites, access cloud services, and more.

Example:

In a typical home network, if your computer has an IP address of 192.168.1.5 and your router's IP is 192.168.1.1, the router's IP becomes the gateway address. Any requests to external servers (like visiting a website) are sent to 192.168.1.1, which forwards the data to its destination. The gateway ensures proper connectivity, communication, and security in a network.

Conclusion

We have successfully understood the purpose of a gateway address.

Experiment 6

Aim :

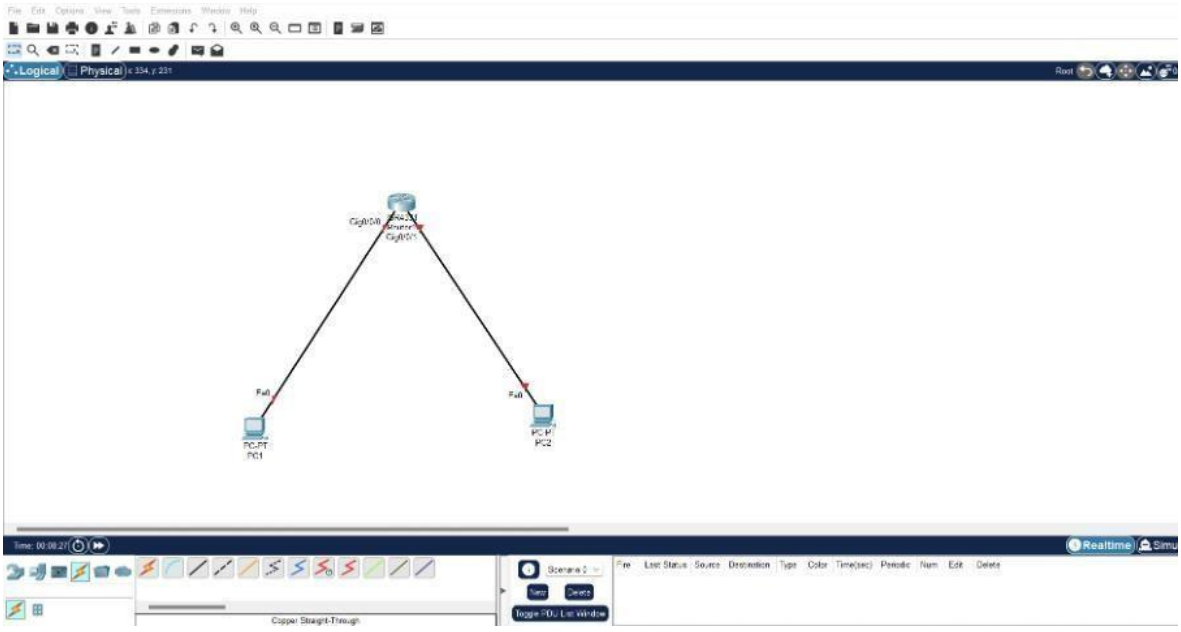
Understand how network information is configured on a Windows computer.

Theory:

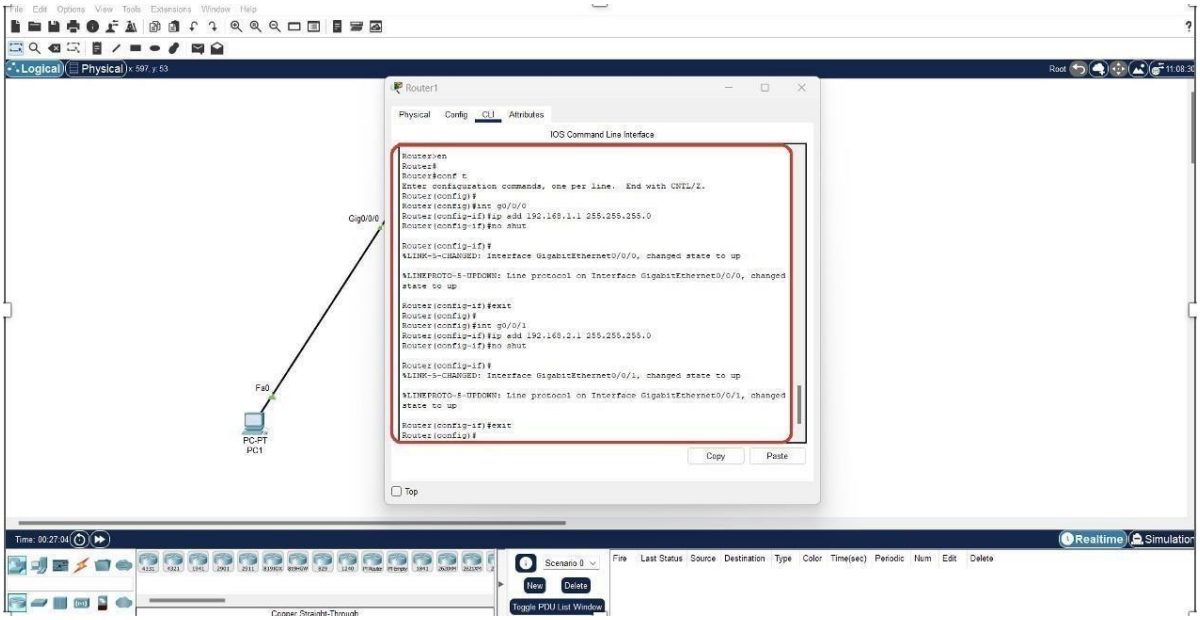
Understanding how network information is configured on a Windows computer involves examining settings such as IP addresses, subnet masks, default gateways, and DNS servers. These configurations determine how the computer communicates within a network and accesses external systems. Tools like ipconfig are commonly used to view and manage these settings, enabling troubleshooting and optimization of network connectivity.

Procedure

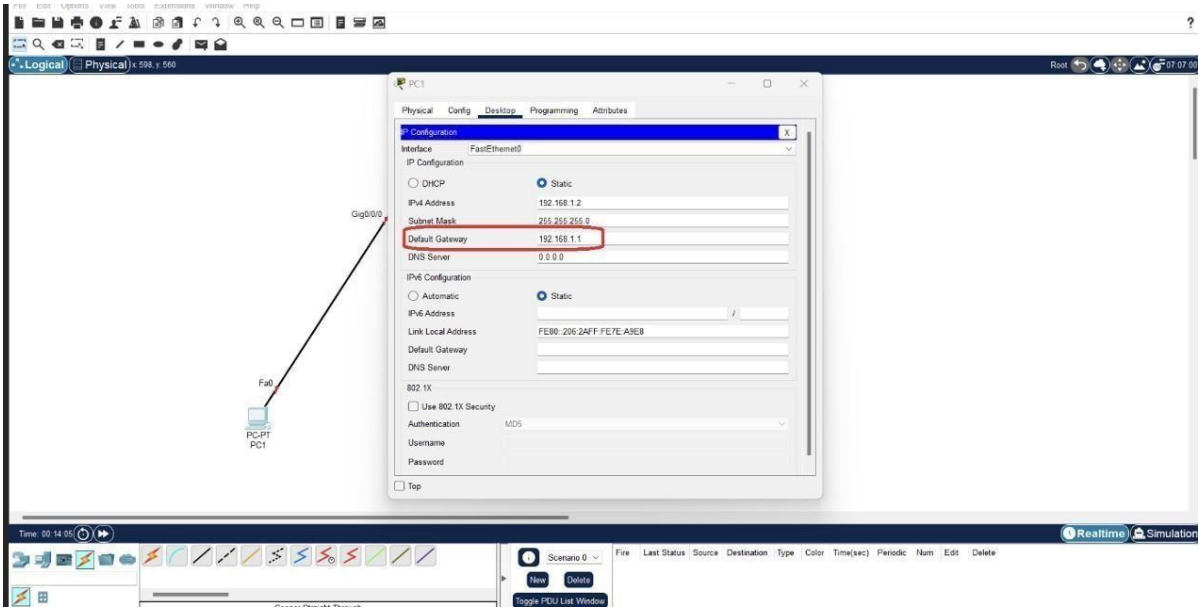
Drag and Drop the Devices according to the Layout.



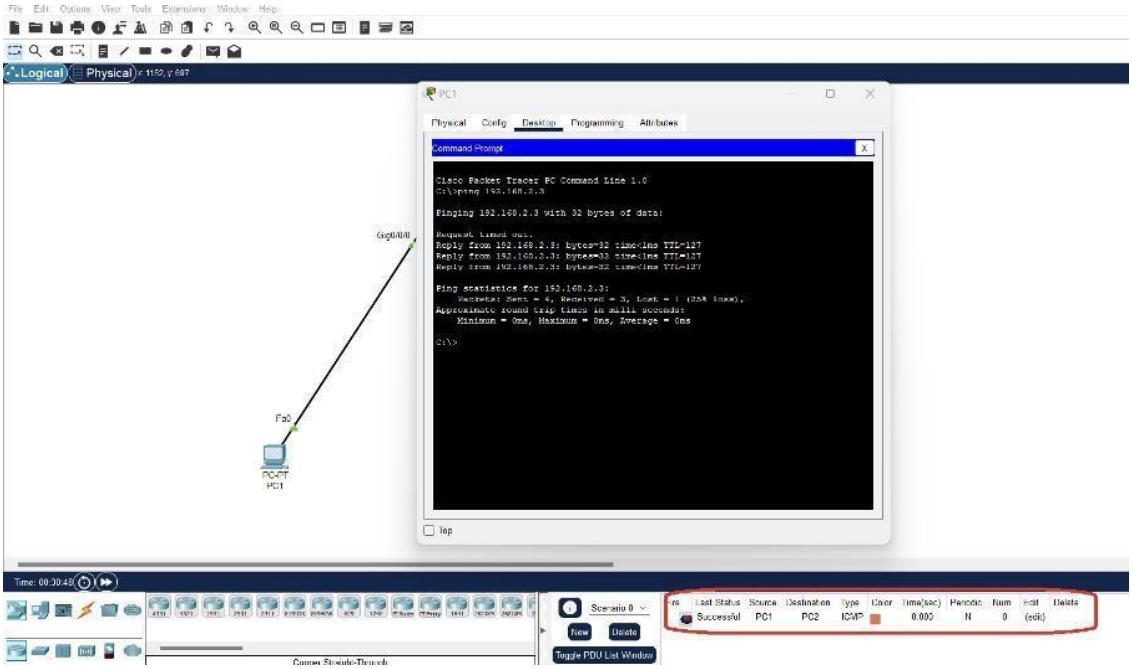
Configure the routers



Now configuring pc



Do the same on other PC



Testing using Ping command

Conclusion

We have understood how network information is configured on a Windows computer.

Experiment 7

Aim:

Understanding and Troubleshooting Hidden Gateway Problem in Cisco Packet Tracer

Theory:

The "hidden gateway" problem arises when devices in a network cannot communicate with devices in another subnet, even though the default gateway is correctly configured. This often occurs due to issues in routing or misconfigured interfaces on the routers.

Procedure:

1. Network Topology Setup
   a) Create the Topology:
      - Place two routers (R1 and R2).
      - Connect R1 and R2 using a serial or Ethernet cable.
      - Connect two PCs (PC1 and PC2) to R1 and R2 respectively via switches (optional).
      - Ensure each router has at least two active interfaces.

   b) Configure IP Addressing:
      - Assign IP addresses to all router interfaces and PCs.
      - Example:
        - R1 (Interface to R2): 192.168.1.1/30
        - R2 (Interface to R1): 192.168.1.2/30
        - R1 (Interface to PC1): 192.168.10.1/24
        - PC1: 192.168.10.2/24 (Default Gateway: 192.168.10.1)
        - R2 (Interface to PC2): 192.168.20.1/24
        - PC2: 192.168.20.2/24 (Default Gateway: 192.168.20.1)

   c) Verify Connectivity:
      - Use the ping command to check connectivity between devices within the same subnet.

2. Identify the Hidden Gateway Problem
   a) Test Inter-Subnet Communication:
      - Attempt to ping PC2 from PC1.
      - Observe that the ping fails despite correct default gateway configuration.

   b) Analyse the Issue:

- Check the routing table on R1 and R2 using the show ip route command.
- Verify if there are static routes or dynamic routing protocols configured.

3. Troubleshooting Steps

a) Step 1: Verify Routing Configuration
   - Ensure that static routes or a routing protocol (like RIP, OSPF, or EIGRP) is properly configured to enable communication between the subnets.
   - Example:
     - On R1: ip route 192.168.20.0 255.255.255.0 192.168.1.2
     - On R2: ip route 192.168.10.0 255.255.255.0 192.168.1.1

b) Step 2: Check Interface Status
   - Use the show ip interface brief command to ensure all interfaces are up and operational.

c) Step 3: Debug Packet Flow
   - Use Packet Tracer's simulation mode to observe packet flow between devices.
   - Look for dropped packets and identify where the issue occurs.

d) Step 4: Correct Misconfigurations
   - Correct any interface IP address mismatches or subnet errors.
   - Enable any disabled interfaces using no shutdown.

e) Step 5: Retest Communication
   - Use ping and traceroute to test connectivity between PC1 and PC2.
   - Verify successful communication.

Conclusion:

In this practical, we identified and resolved the hidden gateway problem by configuring routing, verifying interface statuses, and debugging packet flows. Proper routing and configuration are essential for seamless inter-subnet communication.

Experiment 8

Aim:

Use the **route** command to modify a Windows computer routing table

Theory:

The route command is used to manage the routing table of a computer. The routing table determines the path taken by packets to reach a specific destination network.
Key operations include:

- **Viewing** the routing table.
- **Adding** a route to direct traffic to a specific network.
- **Modifying** an existing route.

Procedure:

1. View the Routing Table

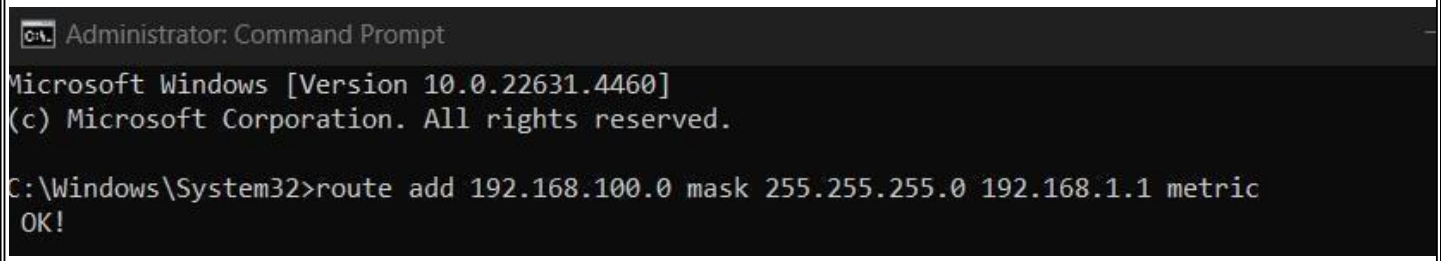
- Open Command Prompt with administrative privileges.
- Type the following command to view the current routing table:

  route print

2. Add a Route

- Add a static route to the routing table.
- Example:

  route add 192.168.50.0 mask 255.255.255.0 192.168.1.1 metric 1
  - 192.168.50.0 is the destination network.
  - 255.255.255.0 is the subnet mask.
  - 192.168.1.1 is the gateway.
  - metric 1 specifies the cost of the route.



3. Verify the Added Route

- Type the following to check if the route is added:

  route print

```
C:\Windows\System32>route print
===========================================================================
Interface List
 20...0a 00 27 00 00 14 ......VirtualBox Host-Only Ethernet Adapter
 17...ce 47 40 eb 62 29 ......Microsoft Wi-Fi Direct Virtual Adapter
  9...ce 47 40 eb 62 39 ......Microsoft Wi-Fi Direct Virtual Adapter #2
 19...cc 47 40 eb 62 69 ......MediaTek Wi-Fi 6 MT7921 Wireless LAN Card
 16...08 bf b8 da c2 89 ......Realtek PCIe GbE Family Controller
  1...........................Software Loopback Interface 1
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0      10.2.55.254      10.2.52.191     45
        10.2.48.0    255.255.248.0         On-link      10.2.52.191    301
      10.2.52.191  255.255.255.255         On-link      10.2.52.191    301
      10.2.55.255  255.255.255.255         On-link      10.2.52.191    301
        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
     192.168.56.0    255.255.255.0         On-link     192.168.56.1    281
     192.168.56.1  255.255.255.255         On-link     192.168.56.1    281
   192.168.56.255  255.255.255.255         On-link     192.168.56.1    281
    192.168.100.0    255.255.255.0     192.168.1.1      10.2.52.191     46
        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     192.168.56.1    281
        224.0.0.0        240.0.0.0         On-link      10.2.52.191    301
  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     192.168.56.1    281
  255.255.255.255  255.255.255.255         On-link      10.2.52.191    301
===========================================================================
Persistent Routes:
  None
```

4. Modify an Existing Route

- Modify the gateway for the route:
  route change 192.168.50.0 mask 255.255.255.0 192.168.2.1
  - Here, the gateway is changed to 192.168.2.1.

```
C:\Windows\System32>route change 192.168.100.0 mask 255.255.255.0 192.168.2.1
 OK!
```

Conclusion:

This practical demonstrates how to manipulate the routing table using the route command.

Experiment 9

Aim:

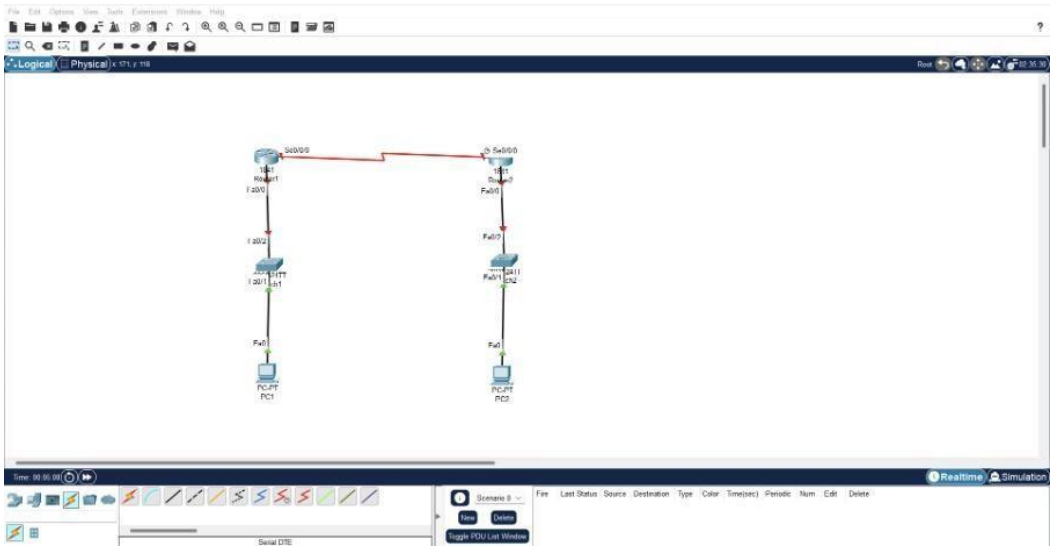
Use a Windows Telnet client command telnet to connect to a Cisco router.
Examine router routes using basic Cisco IOS commands.

Theory:

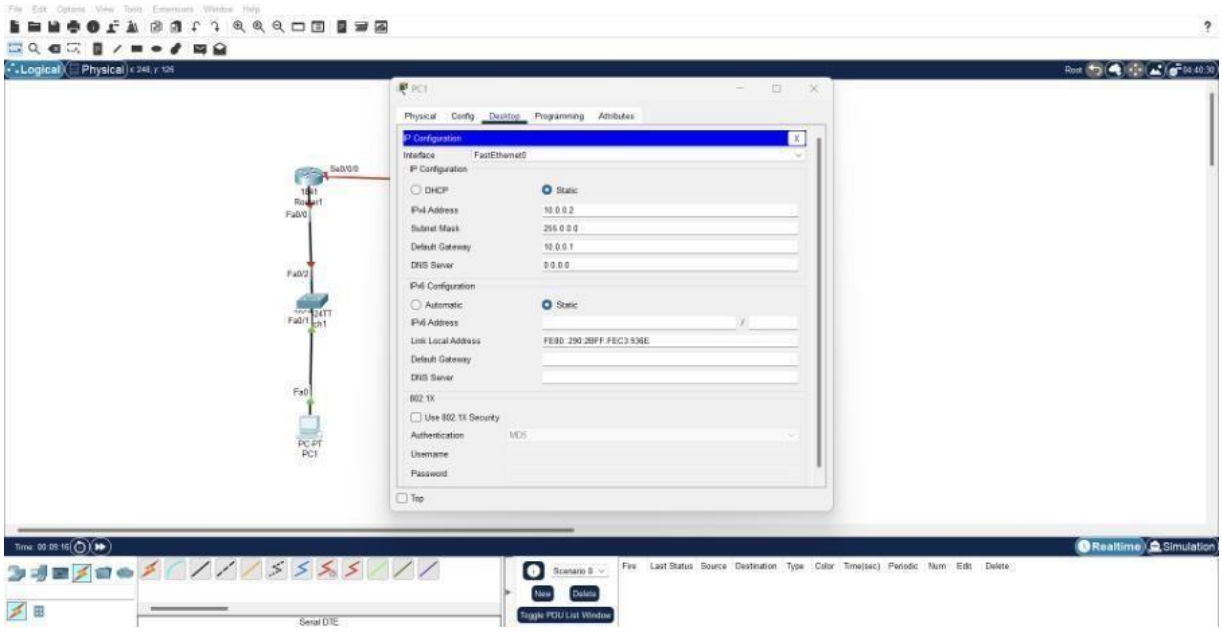
Telnet is a protocol used for remote access to network devices like Cisco routers, allowing users to manage and configure them through a command-line interface. It operates over TCP port 23 and provides a straightforward way to communicate with network equipment.

Procedure

Drag and Drop Devices to create the Layout.



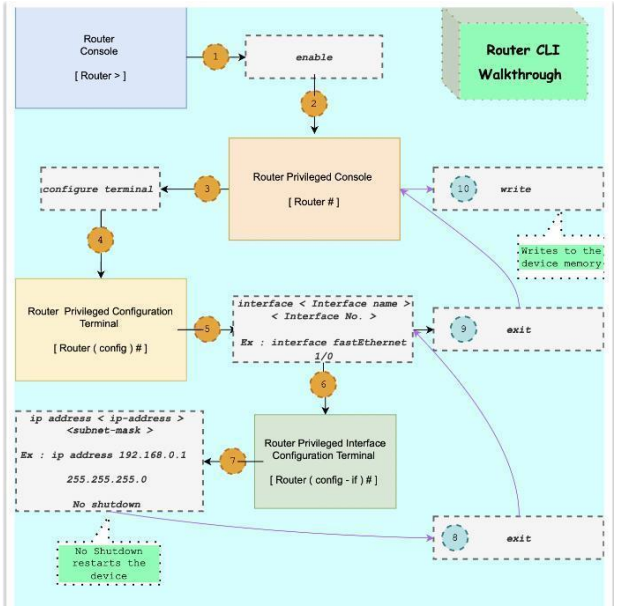
Layout



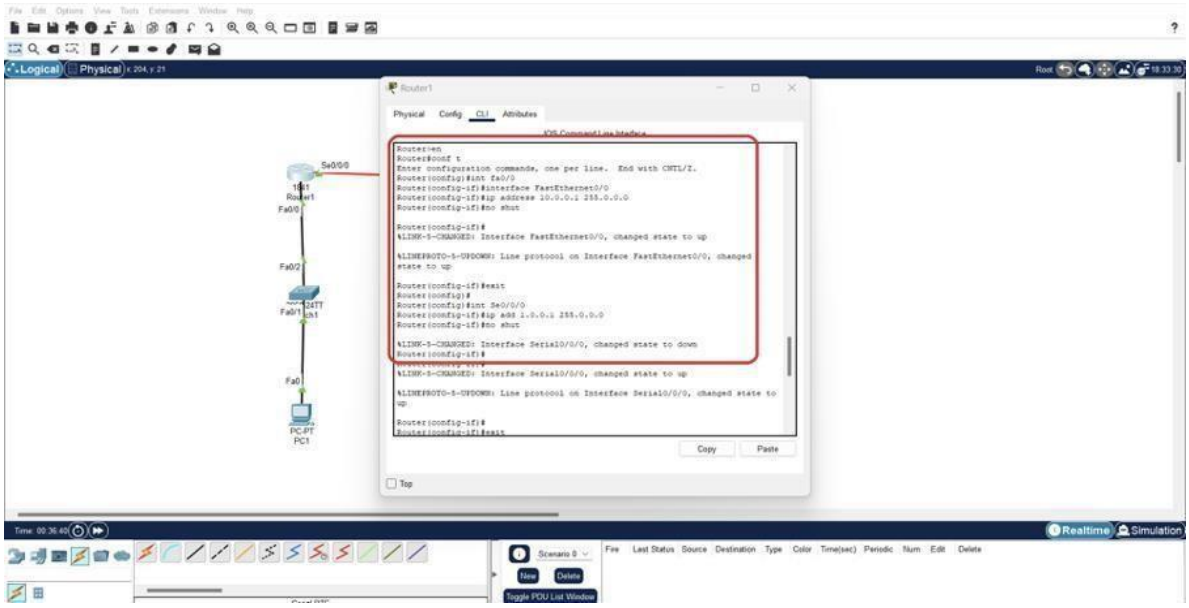
Configure PC1: Double Click on PC1 → Desktop → IP Configuration

Do the same for other PC.

Now Configure the Router



Steps to Configure Router. (Overview)

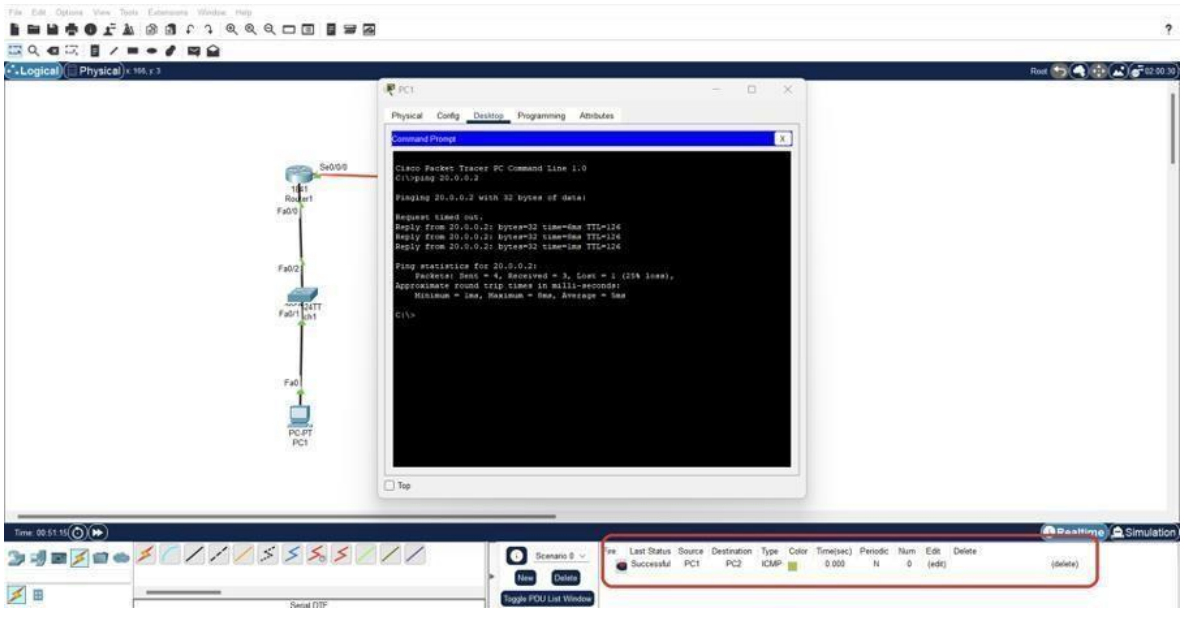


Configuring Router.

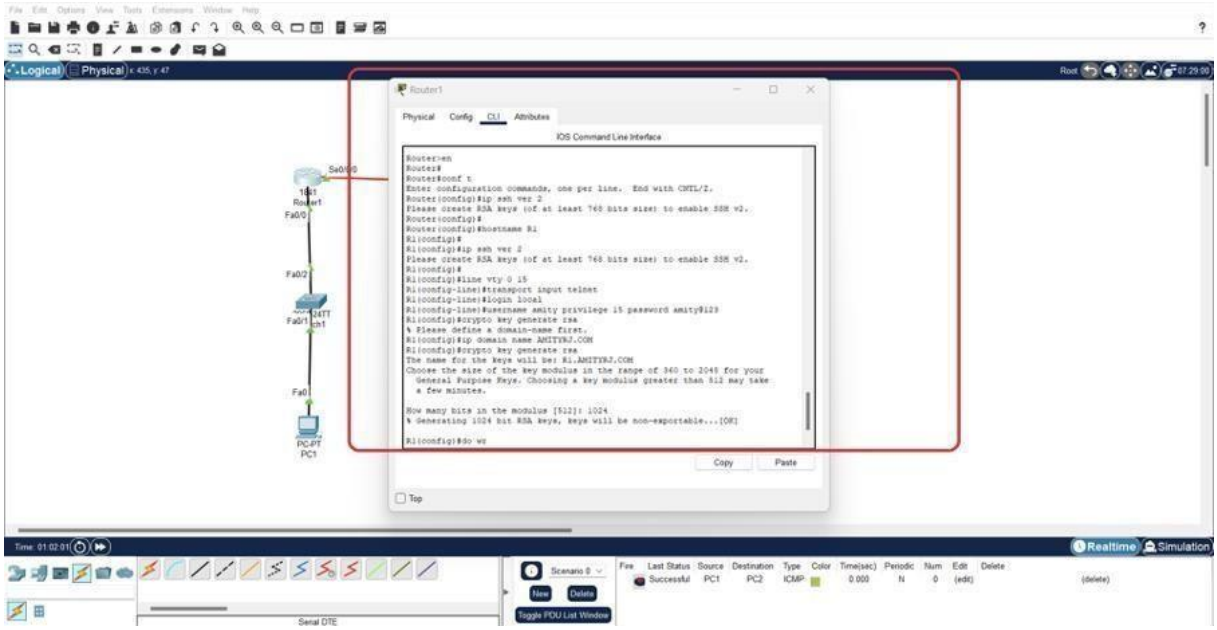
Do the same for the Other Router

Set Static route using Router → Config → Routing Static

Check Reachability Using Ping Command

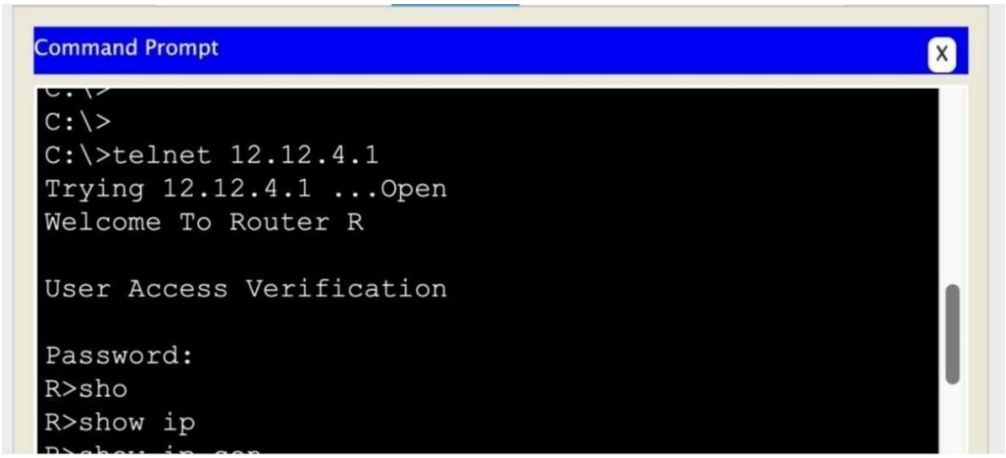


Configure Telnet

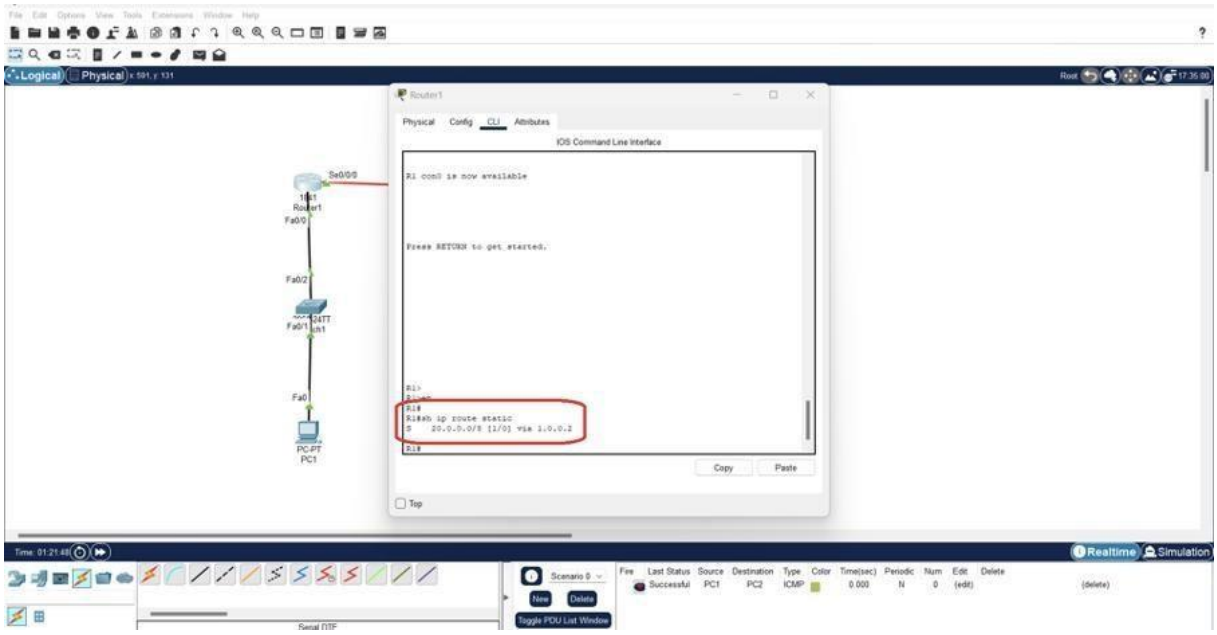


```
# Creating Virtual Terminal
R(config)# line vty < start > < end >

# Setting Password  for the terminal
R(config-line)# password asdf

# Setting config for login
R(config-line)# login

R(config)# Banner " Welcome to Router R "

R(config)# exit
R# write memory
```

Do the same on the other Router



Route:



Type “sh ip route” on CLI of Router

Do the same with the other Router

Conclusion

In this Experiment we have:

- Implemented routes using static routing and examined them.
- Implemented telnet on both routers and accessed them via PC.

Experiment 10

Aim:

Use the ping command to verify simple TCP/IP network connectivity.

Theory:

The ping command is a simple network diagnostic tool used to test connectivity between devices on a TCP/IP network. It sends ICMP Echo Request packets to a specified IP address and waits for an Echo Reply to confirm the connection. By analyzing the response, it verifies if the target device is reachable and provides details about network latency and packet loss, helping identify connectivity issues.

Procedure:

- Open CMD in Windows

  Type Ping (Shows List of Flags)

```
ping [ Flag ] < URL / IP>
-t : Send request until stopped
-n : No. of requests to be sent
-i : Specify Time to Live
-l : Specify Packet Buffer Size
```

```
C:\Users\seth_>ping google.com

Pinging google.com [142.250.194.78] with 32 bytes of data:
Reply from 142.250.194.78: bytes=32 time=127ms TTL=55
Reply from 142.250.194.78: bytes=32 time=83ms TTL=55
Reply from 142.250.194.78: bytes=32 time=63ms TTL=55
Reply from 142.250.194.78: bytes=32 time=59ms TTL=55

Ping statistics for 142.250.194.78:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 59ms, Maximum = 127ms, Average = 83ms
```

Basic ping

```
C:\Users\seth_>ping -t google.com

Pinging google.com [142.250.194.78] with 32 bytes of data:
Reply from 142.250.194.78: bytes=32 time=169ms TTL=55
Reply from 142.250.194.78: bytes=32 time=79ms TTL=55
Reply from 142.250.194.78: bytes=32 time=56ms TTL=55
Reply from 142.250.194.78: bytes=32 time=53ms TTL=55
Reply from 142.250.194.78: bytes=32 time=53ms TTL=55
Reply from 142.250.194.78: bytes=32 time=64ms TTL=55
Reply from 142.250.194.78: bytes=32 time=50ms TTL=55
Reply from 142.250.194.78: bytes=32 time=84ms TTL=55
Reply from 142.250.194.78: bytes=32 time=46ms TTL=55
Reply from 142.250.194.78: bytes=32 time=57ms TTL=55

Ping statistics for 142.250.194.78:
    Packets: Sent = 10, Received = 10, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 46ms, Maximum = 169ms, Average = 71ms
Control-C
^C
```

Using -t Flag

```
C:\Users\seth_>ping -n 5 google.com

Pinging google.com [142.250.193.46] with 32 bytes of data:
Reply from 142.250.193.46: bytes=32 time=59ms TTL=55
Reply from 142.250.193.46: bytes=32 time=152ms TTL=55
Reply from 142.250.193.46: bytes=32 time=75ms TTL=55
Reply from 142.250.193.46: bytes=32 time=71ms TTL=55
Reply from 142.250.193.46: bytes=32 time=81ms TTL=55

Ping statistics for 142.250.193.46:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 59ms, Maximum = 152ms, Average = 87ms
```

Using -n Flag

```
C:\Users\seth_>ping -i 58 google.com

Pinging google.com [142.250.193.46] with 32 bytes of data:
Reply from 142.250.193.46: bytes=32 time=76ms TTL=55
Reply from 142.250.193.46: bytes=32 time=75ms TTL=55
Reply from 142.250.193.46: bytes=32 time=62ms TTL=55
Reply from 142.250.193.46: bytes=32 time=135ms TTL=55

Ping statistics for 142.250.193.46:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 62ms, Maximum = 135ms, Average = 87ms
```

Using -i Flag

```
C:\Users\seth_>ping -l 64 google.com

Pinging google.com [142.250.193.46] with 64 bytes of data:
Reply from 142.250.193.46: bytes=64 time=120ms TTL=55
Reply from 142.250.193.46: bytes=64 time=74ms TTL=55
Reply from 142.250.193.46: bytes=64 time=86ms TTL=55
Reply from 142.250.193.46: bytes=64 time=152ms TTL=55

Ping statistics for 142.250.193.46:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 74ms, Maximum = 152ms, Average = 108ms
```

Using -l flag

Conclusion

Hence, we now understand how ping works.

Experiment 11

Aim:

Use the tracert/traceroute command to verify TCP/IP connectivity.

Theory:

The tracert (Windows) or traceroute (Linux/Unix) command traces the path packets take to reach a destination across a TCP/IP network. It identifies each hop (router or device) along the route and measures the time taken to reach each hop. This tool helps diagnose connectivity issues by pinpointing delays or failures in the network path.

Procedure:

In command prompt:

Type tracert

```
tracert [ Flag ] < IP/ URL >
-d : do not resolve addresses to host
-h : Max no. of Hops
-w : wait timeout
-4 : Force IPv4
```

```
C:\Users\seth_>tracert www.google.com

Tracing route to www.google.com [142.250.193.196]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.64.1
  2    73 ms    29 ms    11 ms  192.168.23.12
  3   264 ms   206 ms   203 ms  192.168.27.237
  4   110 ms    34 ms   480 ms  192.168.27.49
  5     *        *        *     Request timed out.
  6   121 ms    43 ms   260 ms  192.168.31.201
  7   104 ms    92 ms    41 ms  182.78.210.9
  8   117 ms   106 ms    95 ms  116.119.61.253
  9   109 ms   114 ms   115 ms  142.250.161.56
 10   593 ms   126 ms    95 ms  142.251.66.173
 11    94 ms    79 ms    66 ms  142.251.54.95
 12   107 ms    48 ms    46 ms  del11s17-in-f4.1e100.net [142.250.193.196]

Trace complete.
```

Basic Tracert

```
C:\Users\seth_>tracert -d  www.google.com

Tracing route to www.google.com [142.250.193.196]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.64.1
  2    78 ms     9 ms     6 ms  192.168.23.12
  3   182 ms   275 ms   139 ms  192.168.27.237
  4    50 ms    29 ms    37 ms  192.168.27.49
  5     *        *        *     Request timed out.
  6    75 ms    45 ms    27 ms  192.168.31.201
  7    45 ms    57 ms    44 ms  182.78.210.9
  8    74 ms    54 ms    51 ms  116.119.61.253
  9   105 ms    92 ms   101 ms  142.250.161.56
 10   152 ms   100 ms    98 ms  142.251.66.173
 11    76 ms    63 ms    60 ms  142.251.54.95
 12    44 ms    46 ms    48 ms  142.250.193.196

Trace complete.
```

Using -d Flag

```
C:\Users\seth_>tracert -h 5  www.google.com

Tracing route to www.google.com [142.250.193.196]
over a maximum of 5 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.64.1
  2   130 ms    10 ms    11 ms  192.168.23.12
  3    91 ms   207 ms   171 ms  192.168.27.237
  4   152 ms    97 ms    47 ms  192.168.27.49
  5     *        *        *     Request timed out.

Trace complete.
```

Using -h Flag

```
C:\Users\seth_>tracert -w 60  www.google.com

Tracing route to www.google.com [142.250.193.196]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.64.1
  2    52 ms    10 ms    23 ms  192.168.23.12
  3     *      191 ms   208 ms  192.168.27.237
  4   106 ms    39 ms    83 ms  192.168.27.49
  5     *        *        *     Request timed out.
  6   100 ms    47 ms    35 ms  192.168.31.201
  7    45 ms    53 ms   119 ms  182.78.210.9
  8    89 ms    51 ms    70 ms  116.119.61.253
  9   128 ms   107 ms     *     142.250.161.56
 10   138 ms    97 ms   112 ms  142.251.66.173
 11    79 ms    66 ms     *     142.251.54.95
 12    86 ms    55 ms    46 ms  del11s17-in-f4.1e100.net [142.250.193.196]

Trace complete.
```

Using -w Flag

```
C:\Users\seth_>tracert -4  www.google.com

Tracing route to www.google.com [142.250.193.196]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.64.1
  2    10 ms    27 ms    10 ms  192.168.23.12
  3   214 ms   213 ms   190 ms  192.168.27.237
  4    42 ms    42 ms    39 ms  192.168.27.49
  5     *        *        *     Request timed out.
  6    31 ms    53 ms    36 ms  192.168.31.201
  7    43 ms    50 ms    62 ms  182.78.210.9
  8    65 ms    58 ms    38 ms  116.119.61.253
  9   110 ms    88 ms   112 ms  142.250.161.56
 10   107 ms    88 ms   103 ms  142.251.66.173
 11   120 ms    67 ms    83 ms  142.251.54.95
 12    60 ms    77 ms    56 ms  del11s17-in-f4.1e100.net [142.250.193.196]
```

Using -4 Flag

‘*’ represents Response Timeout

Conclusion

Hence, we now understand how tracert works.

Experiment 12

Aim:

Implement encryption and decryption through Cryp Tool

- Caesar cipher
- Shift cipher

Theory:

Caesar Cipher

The Caesar cipher is a substitution cipher where each letter in the plaintext is shifted by a fixed number of positions in the alphabet. For example, with a shift of 3, 'A' becomes 'D,' 'B' becomes 'E,' and so on. It is one of the simplest encryption techniques.

- **Encryption**:
  Formula: $E(x)=(x+n) \mod 26$ $E(x) = (x + n) \mod 26$ $E(x)=(x+n) \mod 26$
  Where xxx is the position of the letter in the alphabet, and nnn is the shift value.

- **Decryption**:
  Formula: $D(x)=(x-n) \mod 26$ $D(x) = (x - n) \mod 26$ $D(x)=(x-n) \mod 26$
  This reverses the shift applied during encryption.

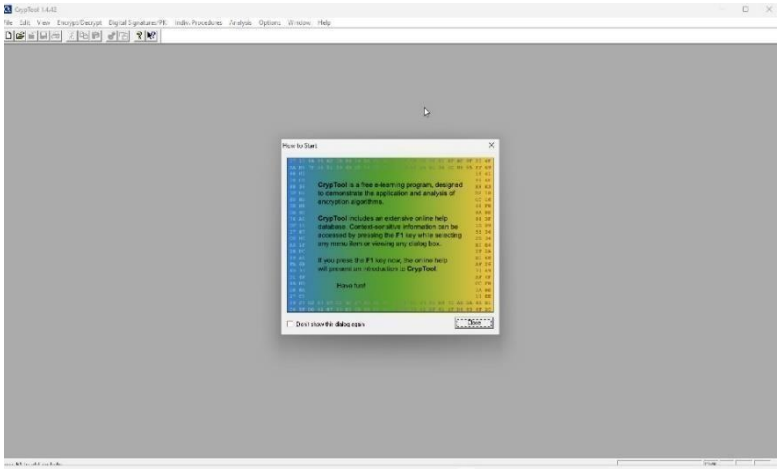
Shift Cipher

The Shift cipher is a generalized form of the Caesar cipher. While the Caesar cipher traditionally uses a shift of 3, the Shift cipher allows for any integer shift value, including negative shifts for reverse order.

- **Encryption** and **Decryption** follow the same principles as the Caesar cipher, with the shift value (nnn) being user-defined.

Procedure:

- Download Crypt-tool CT1: www.cryptool.org/en/ct1/downloads
- Complete the Installation
- Post Installation



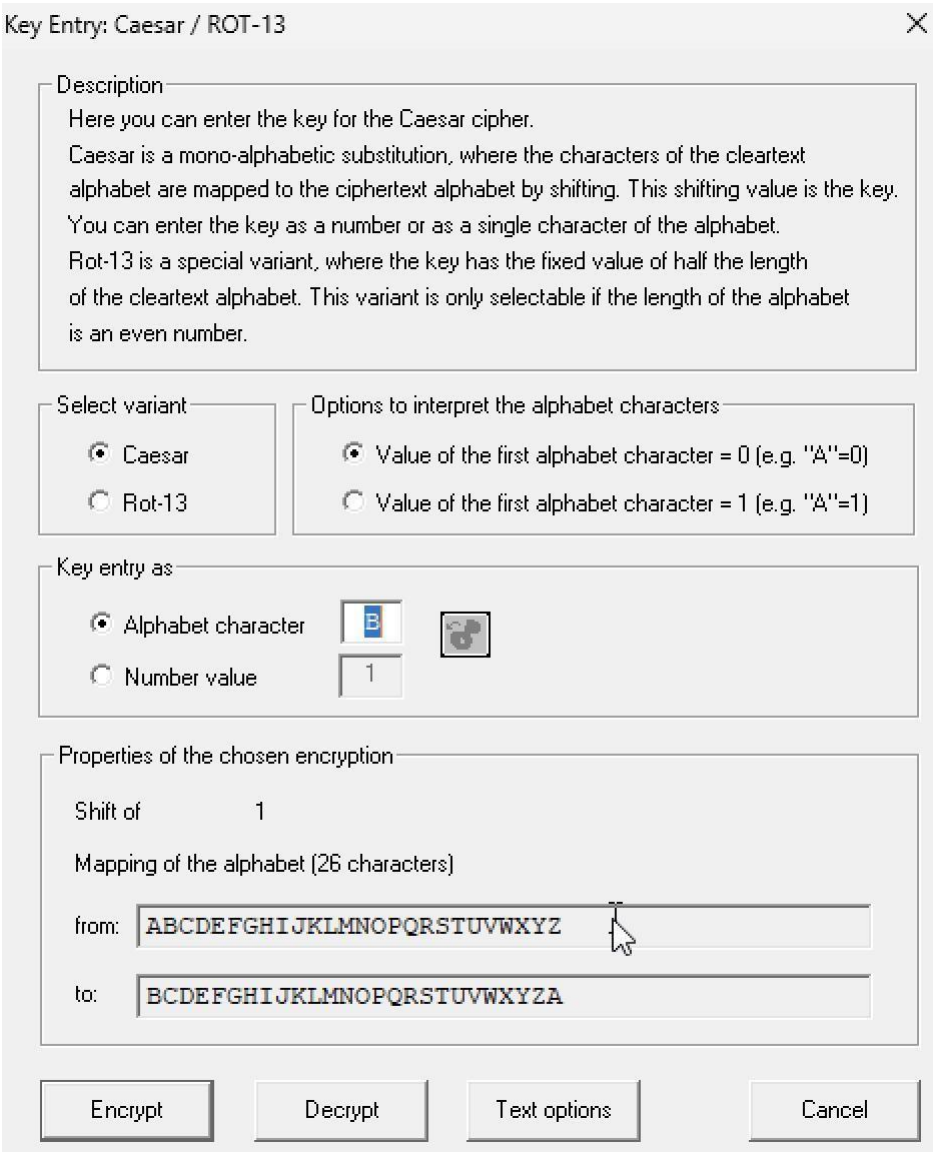
Opening Screen of the Application

For Encryption

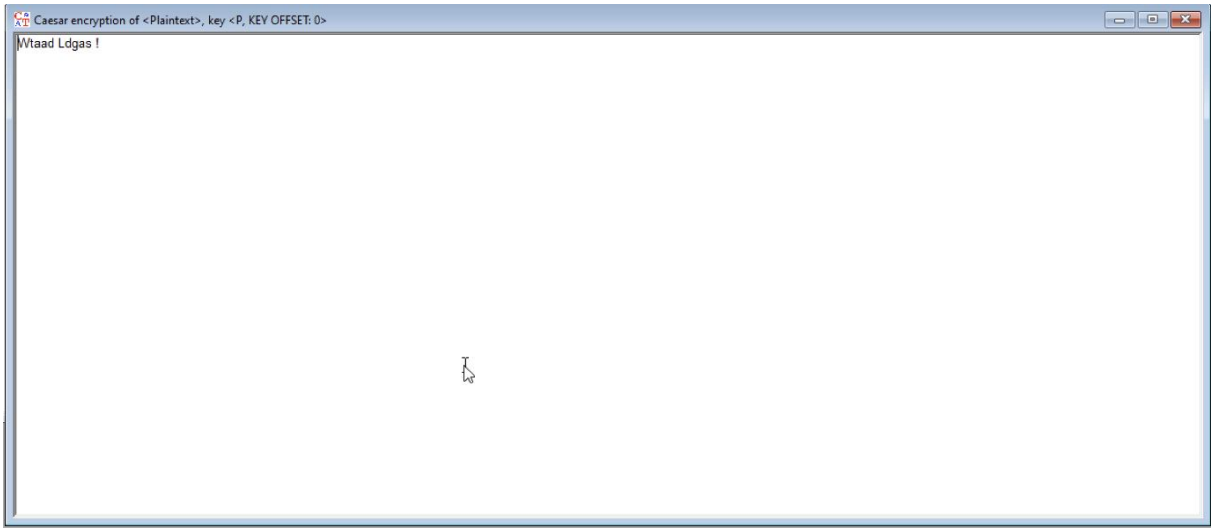
Tool Bar → File → New

```
Hello World
```

Command + S → Save as “Clear Text”
Now,
Go to Toolbar → Encrypt/Decrypt → Symmetric (Classic)→ Caesar / ROT 13 ….

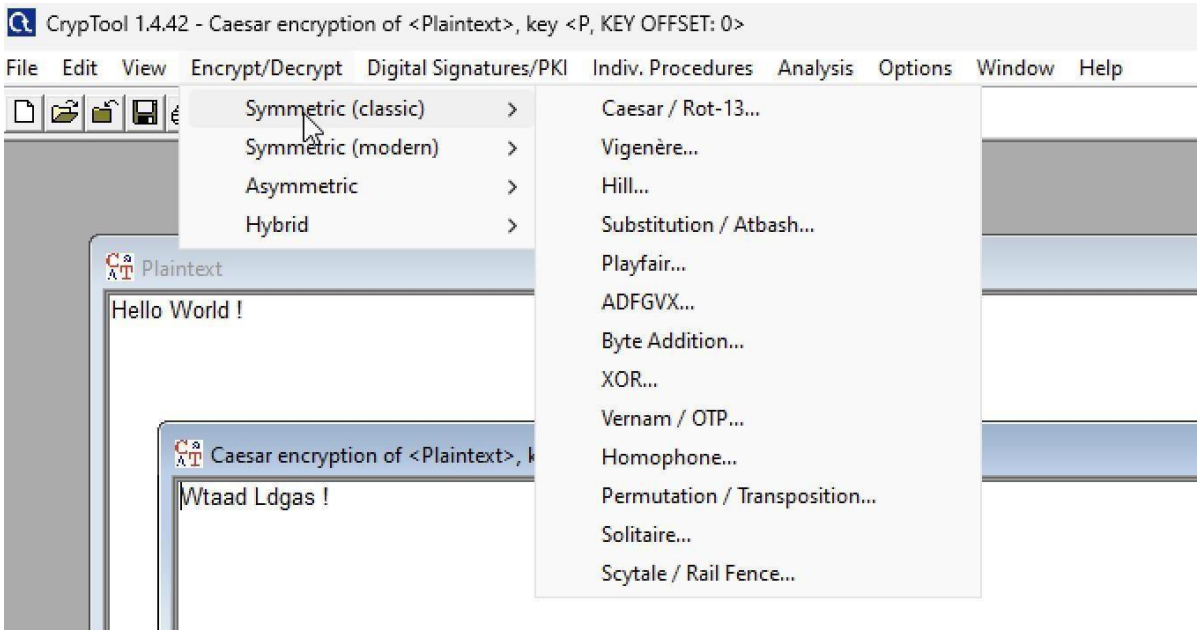


Select Number Value of the Key: 15
Then Click Encrypt.



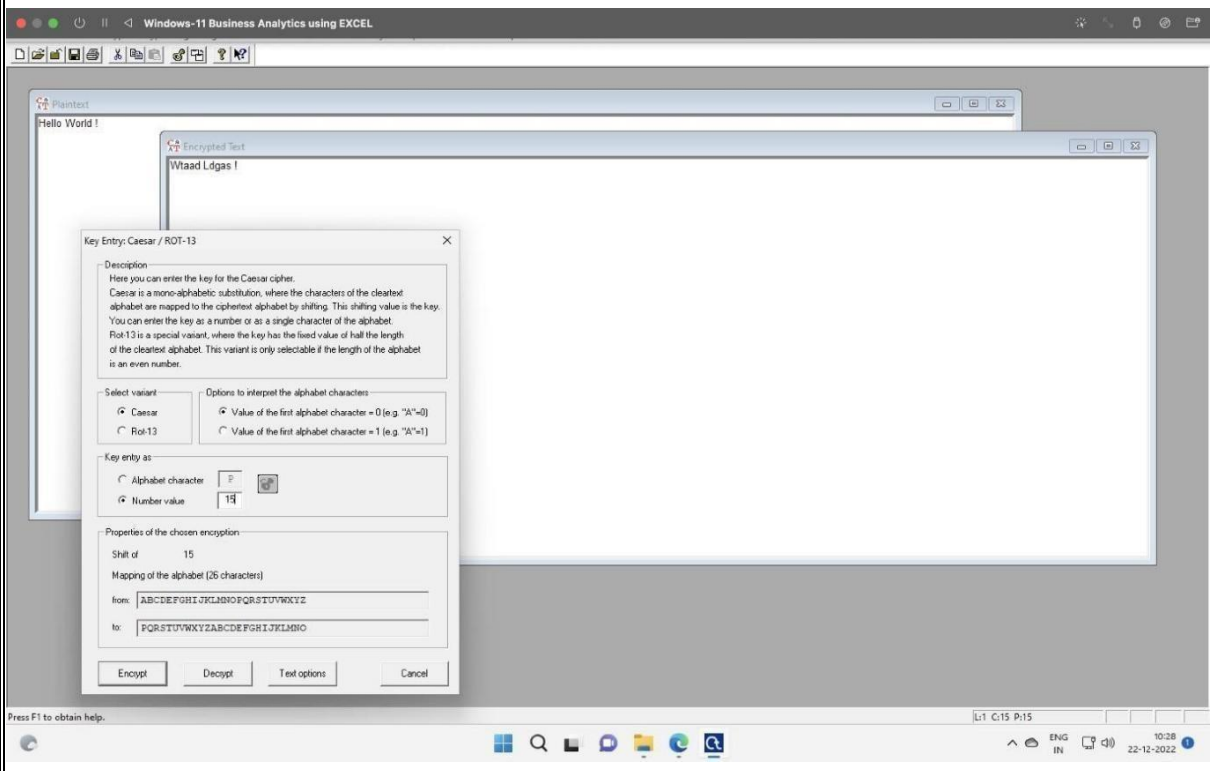
Command + S : To save it as “ EncryptedText”.

For Decryption



Go to Toolbar → Encrypt/Decrypt → Symmetric (Classic)→ Caesar / ROT 13 ….

Set Number Value of the Key : 15 Then Click Decrypt.



Caeser Cipher

Conclusion

Now we understand how crypt-tool works and have successfully understood how to implement Caesar and shift cipher.

Experiment 13

Aim:

Implement encryption and decryption through Cryp Tool and Python programming.

- Affine cipher
- Substitution with symbols

Theory:

The Affine Cipher is a substitution cipher that uses a mathematical formula to encrypt and decrypt text. For encryption, each letter x is transformed using the formula $E(x)=(a \cdot x+b) \bmod 26$, where a and b are keys. Decryption uses the modular inverse of a to reverse the transformation: $D(y)= a^{-1}(y-b) \bmod 26$. The cipher ensures simple yet secure text encoding based on modular arithmetic.

Procedure:

Affine Cipher:

The program is as follows:

```python
# Function to encrypt a message
def affine_encrypt(text, a, b):
    encrypted = ""
    for char in text.upper():
        if char.isalpha():  # Check if the character is a letter
            x = ord(char) - 65  # Convert letter to a number (A=0, B=1, ..., Z=25)
            new_char = (a * x + b) % 26  # Apply the encryption formula
            encrypted += chr(new_char + 65)  # Convert the number back to a letter
        else:
            encrypted += char  # Keep non-letters as they are
    return encrypted

# Function to decrypt a message
def affine_decrypt(cipher, a, b):
    decrypted = ""
    a_inverse = pow(a, -1, 26)  # Find the modular inverse of 'a'
    for char in cipher.upper():
        if char.isalpha():
            y = ord(char) - 65
            original_char = (a_inverse * (y - b)) % 26  # Apply the decryption formula
            decrypted += chr(original_char + 65)
        else:
            decrypted += char
    return decrypted

# Example usage
a, b = 5, 8  # Keys
message = "HELLO"
cipher_text = affine_encrypt(message, a, b)
print("Encrypted:", cipher_text)
plain_text = affine_decrypt(cipher_text, a, b)
print("Decrypted:", plain_text)
```

Output:

```
[Running] python -u "C:\Users\sneha\AppData\Local\Temp\tempCodeRunnerFile.python"
Encrypted: RCLLA
Decrypted: HELLO

[Done] exited with code=0 in 0.063 seconds
```

Substitution with symbols

The program is as follows:

```python
SUBSTITUTION_LIST = [
    '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '-', '=', '+', '{', '}',
    '[', ']', ':', ';', '"', '<', '>', '?', '/', '~'
]

# Function to encrypt a message
def affine_encrypt_with_symbols(text, a, b):
    encrypted = ""
    for char in text.upper():
        if char.isalpha():  # Check if the character is a letter
            x = ord(char) - 65  # Convert letter to a number (A=0, B=1, ..., Z=25)
            new_char = (a * x + b) % 26  # Apply the encryption formula
            encrypted += SUBSTITUTION_LIST[new_char]  # Use the substitution list
        else:
            encrypted += char  # Keep non-letters as they are
    return encrypted

# Function to decrypt a message
def affine_decrypt_with_symbols(cipher, a, b):
    decrypted = ""
    a_inverse = pow(a, -1, 26)  # Find the modular inverse of 'a'
    for char in cipher:
        if char in SUBSTITUTION_LIST:  # Check if the character is in the substitution list
            y = SUBSTITUTION_LIST.index(char)  # Find the index of the symbol
            original_char = (a_inverse * (y - b)) % 26  # Apply the decryption formula
            decrypted += chr(original_char + 65)  # Convert back to a letter
        else:
            decrypted += char  # Keep non-substituted characters as they are
    return decrypted
# Example usage
a, b = 5, 8  # Keys
message = "HELLO"

# Encrypt the message
cipher_text = affine_encrypt_with_symbols(message, a, b)
print("Encrypted with Symbols:", cipher_text)

# Decrypt the message
plain_text = affine_decrypt_with_symbols(cipher_text, a, b)
print("Decrypted:", plain_text)

```

Output:

```
[Running] python -u "C:\Users\sneha\AppData\Local\Temp\tempCodeRunnerFile.python"
Encrypted with Symbols: :#==!
Decrypted: HELLO
```

Conclusion:

Successfully implemented affine cipher and substitution with characters.

Experiment 14

Aim:

Implement encryption and decryption through Cryp Tool and Python programming.

- Vigenère cipher
- Hill cipher

Theory:

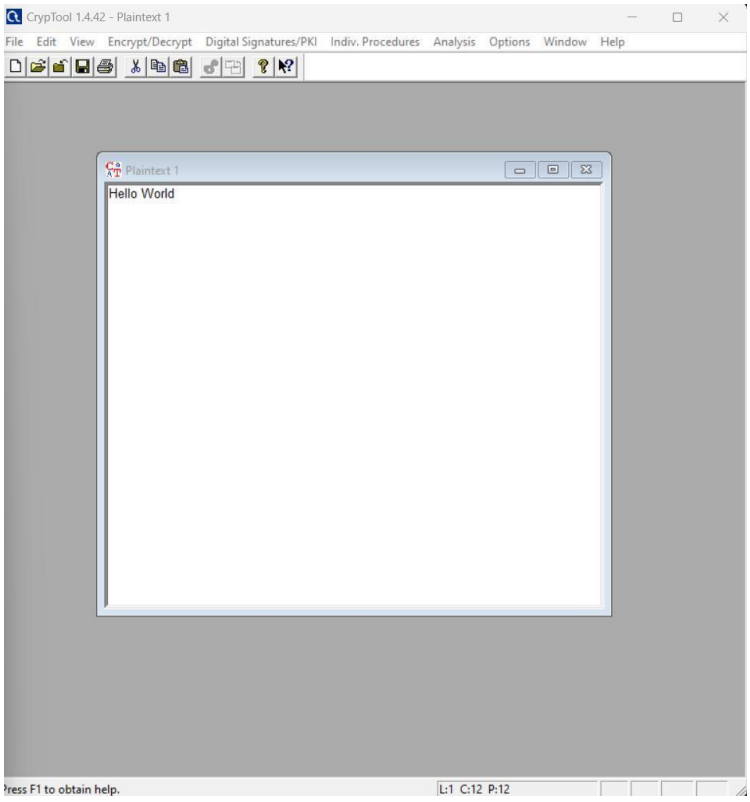
The Vigenère Cipher is a polyalphabetic substitution cipher that uses a keyword to encrypt and decrypt text. Each letter of the keyword determines a shift for the corresponding plaintext letter using the formula E(x)=(x+k) mod, where k is the key letter. This cipher provides more security than simple substitution by varying the shifts.

The Hill Cipher is a polygraphic substitution cipher that uses linear algebra. It encrypts text in blocks of letters, transforming them into numerical vectors and applying matrix multiplication with a key matrix modulo 26. Decryption requires the inverse of the key matrix, ensuring a strong encryption mechanism based on mathematics.

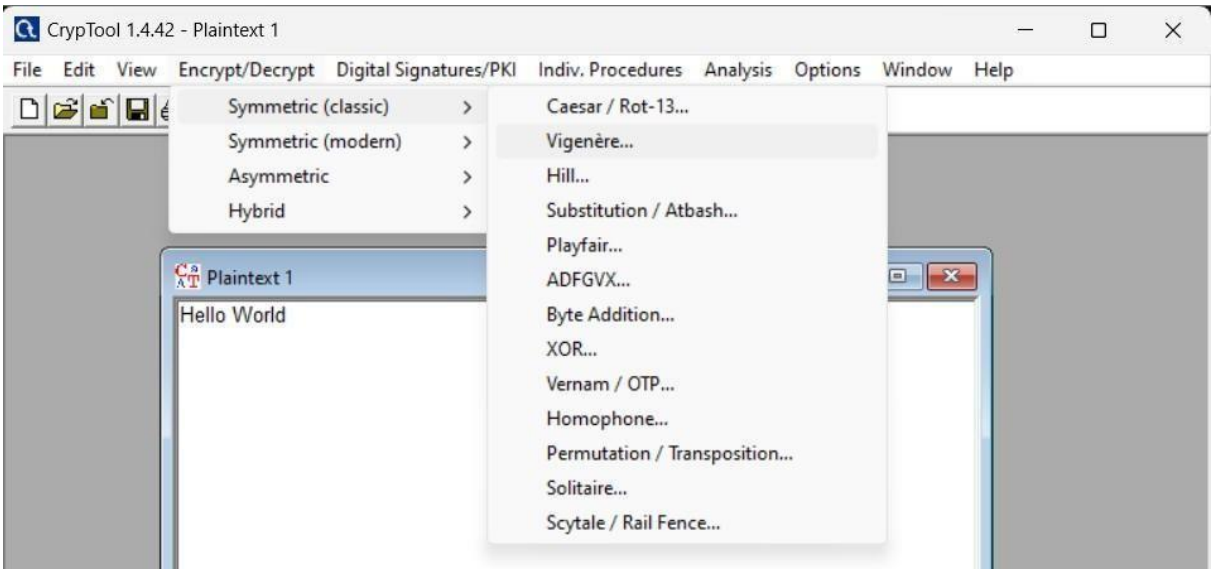
Procedure:

Vigenère Cipher

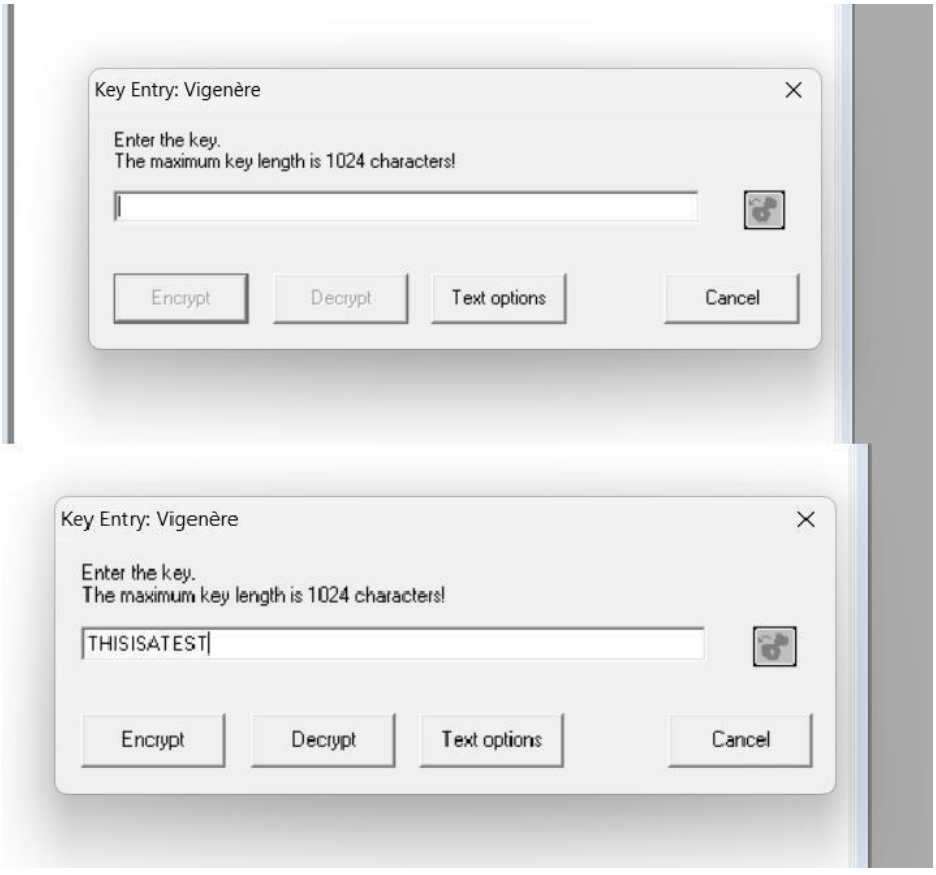
Create a new text file, enter a plain text to be encrypted and save the file.



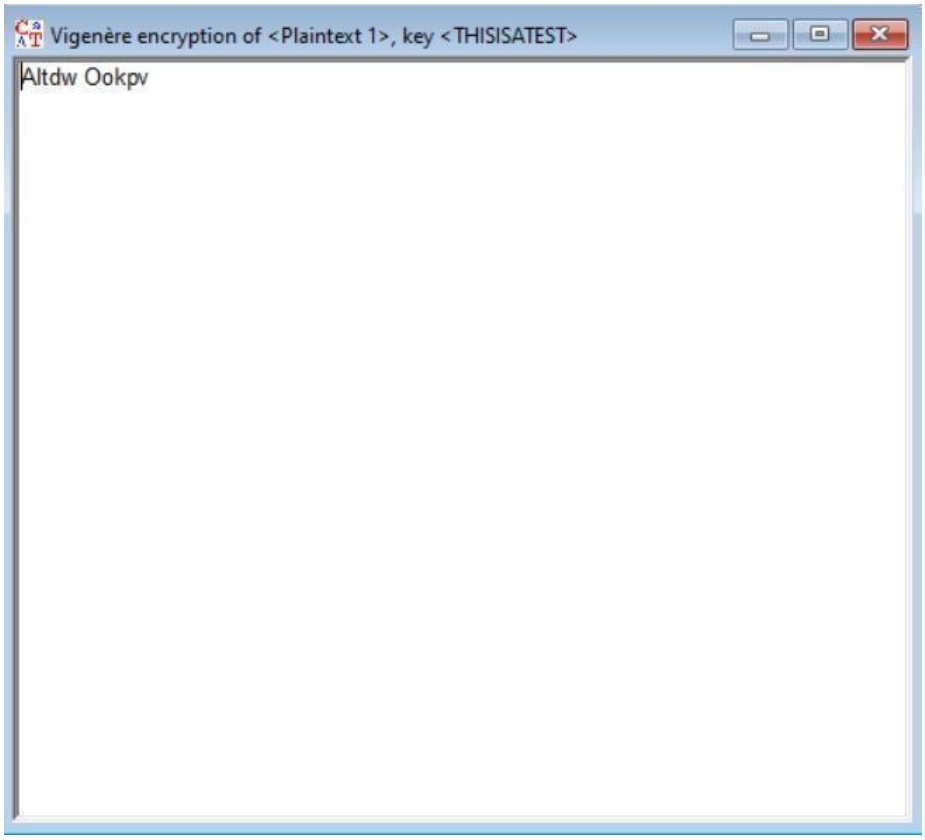
Go to the toolbar > Encrypt/Decrypt > Symmetric(classic) > Vigenère…



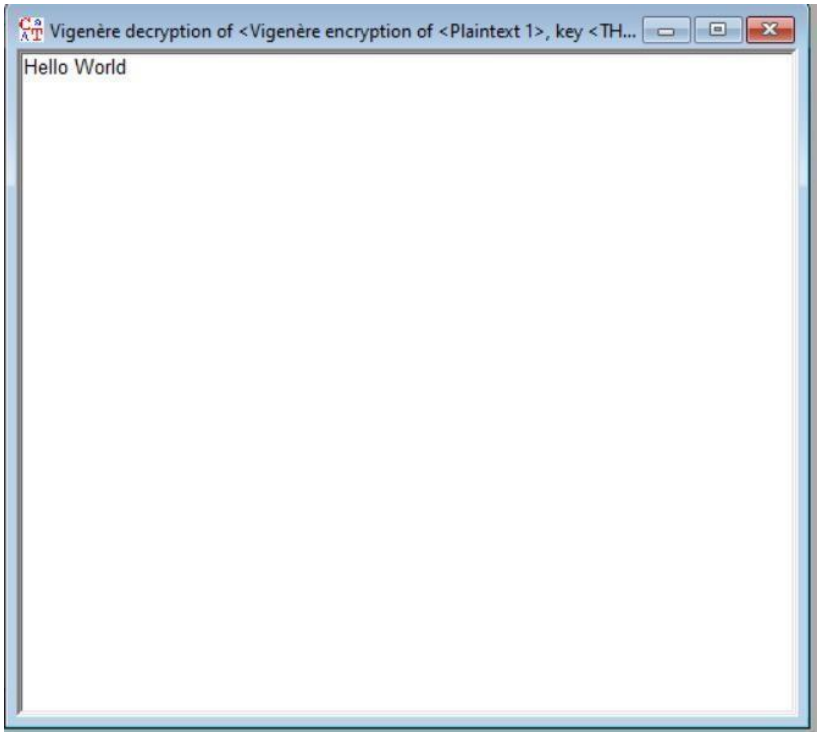
Enter the key and click on encrypt.



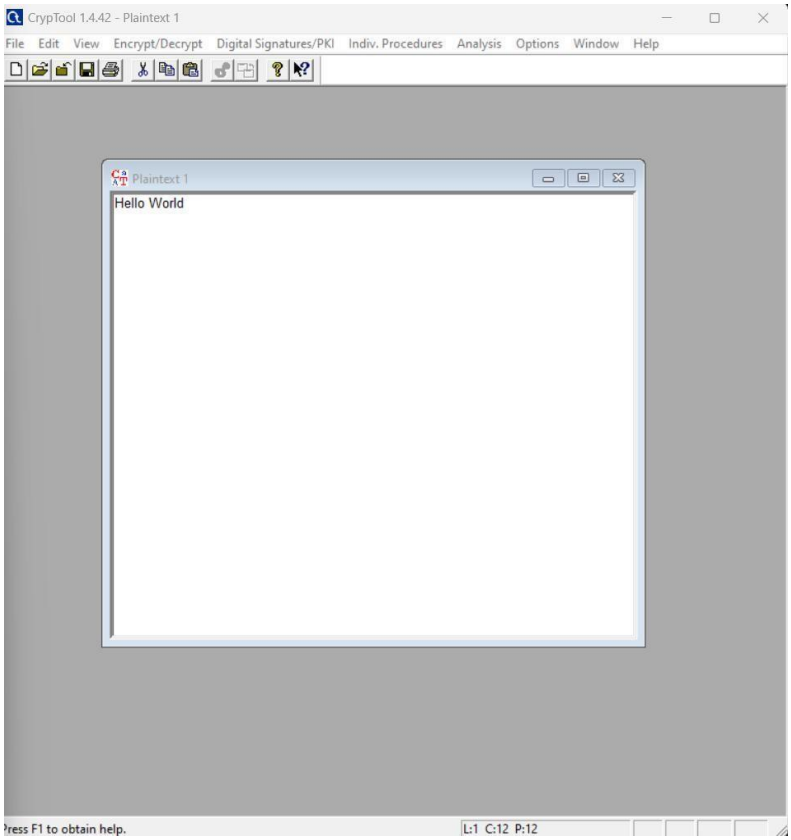
Save the encrypted text as a text file.



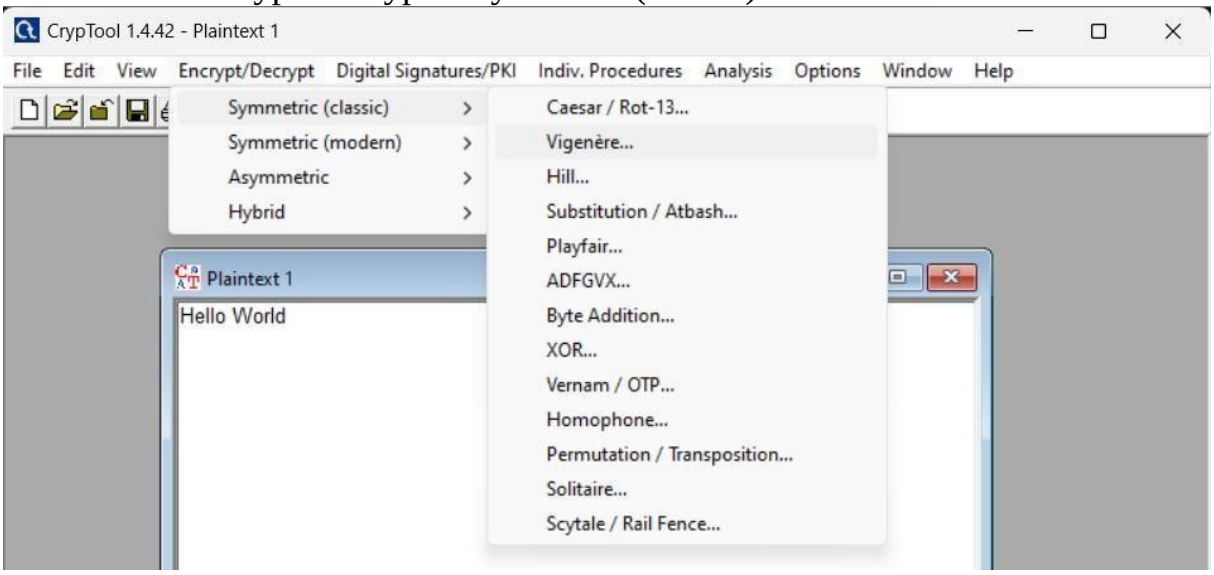
To decrypt it go to the toolbar > Encrypt/Decrypt > Symmetric(classic) > Vigenère…
Enter the key again and click on decrypt.
The ciphertext is decrypted successfully.



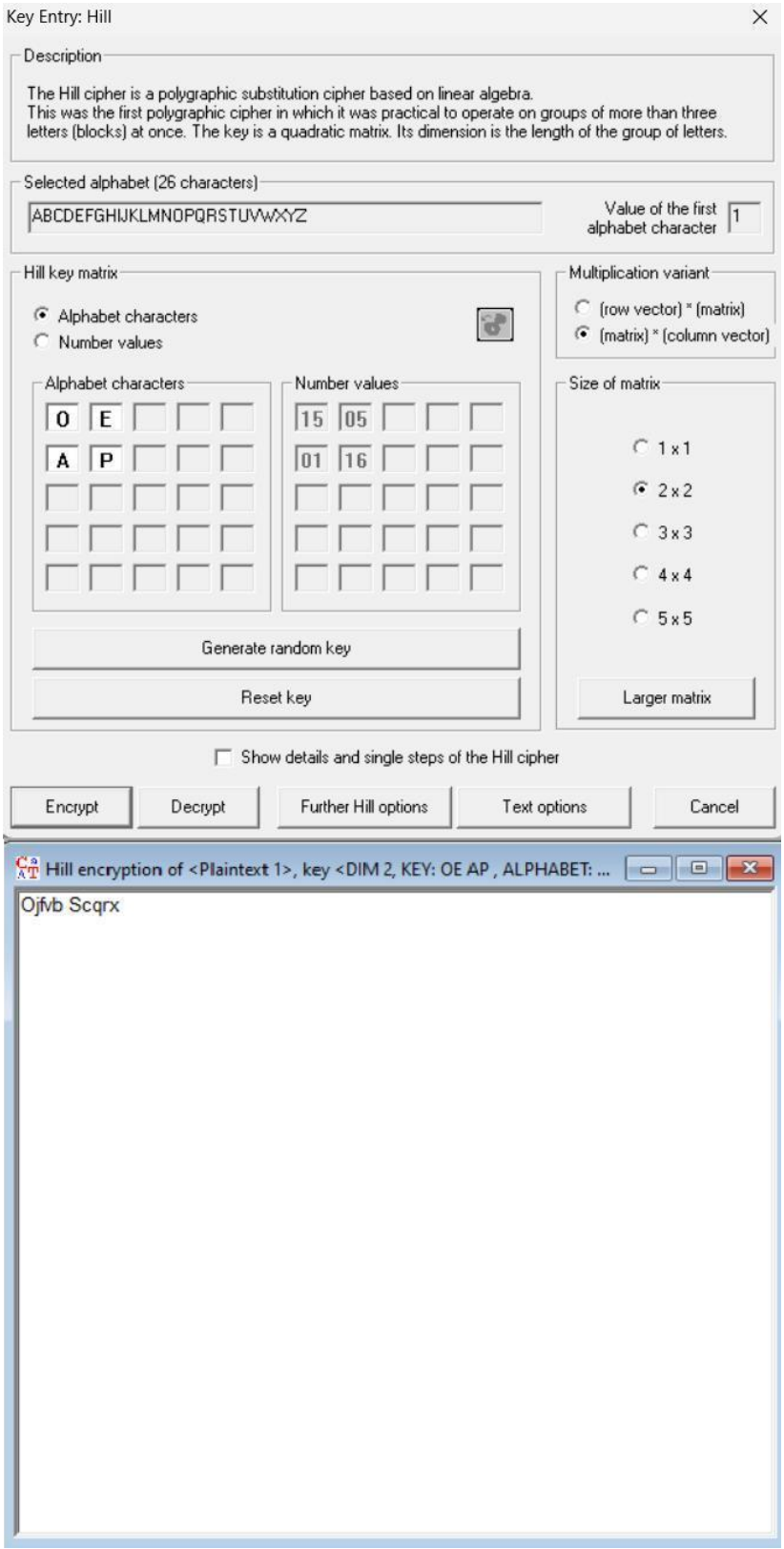
Hill Cipher
Create a new text file, enter a plain text to be encrypted and save the file.



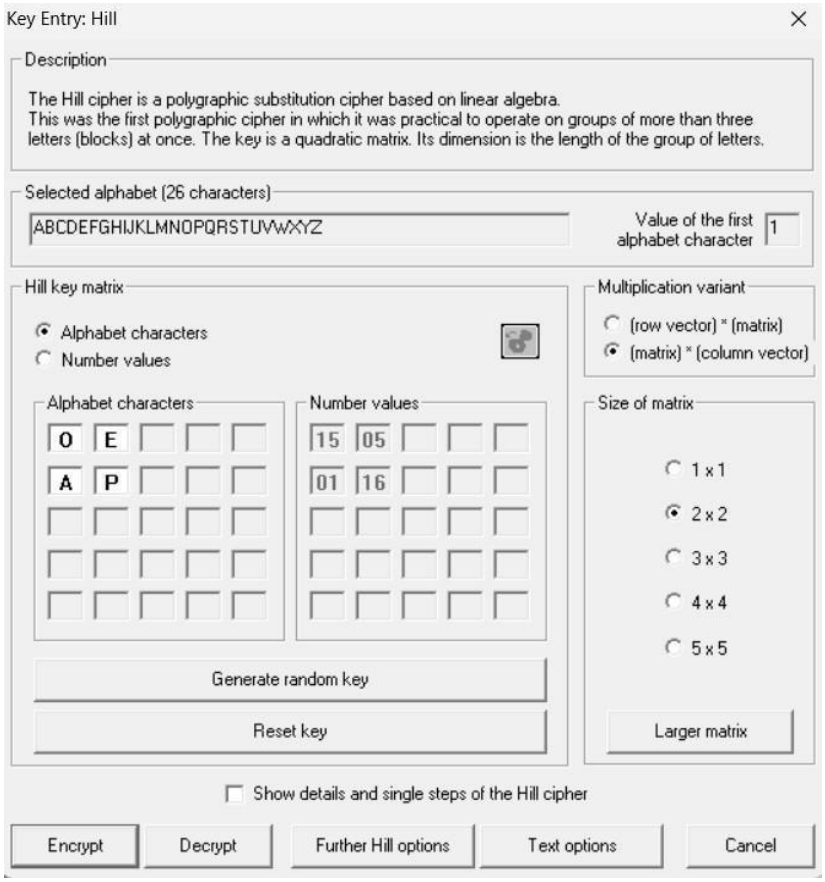
Go to the toolbar > Encrypt/Decrypt > Symmetric(classic) > Hill…



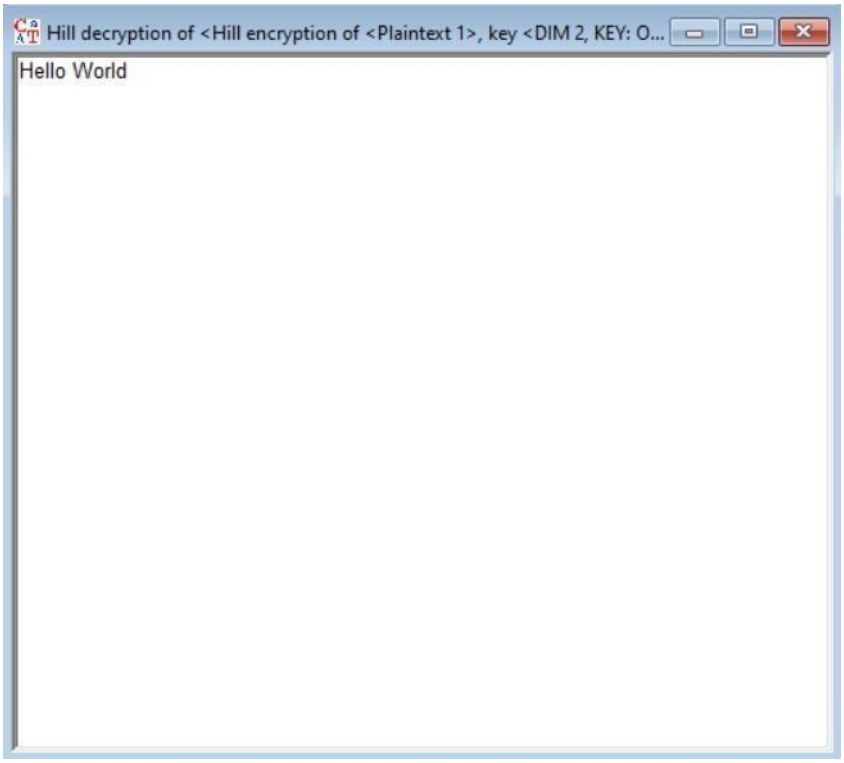
Enter the size of the matrix accordingly, enter the key in the matrix, and click on encrypt.



The cipher text is generated. To decrypt it back go to toolbar > Encrypt/Decrypt > Symmetric(classic) > Hill…



Enter the key and click on decrypt.



Conclusion

We have successfully implemented Vigenère and Hill ciphers.

Experiment 15

Aim:

To understand the applications of asymmetric cryptography

- One-way functions
- The Diffie-Hellman key exchange protocol

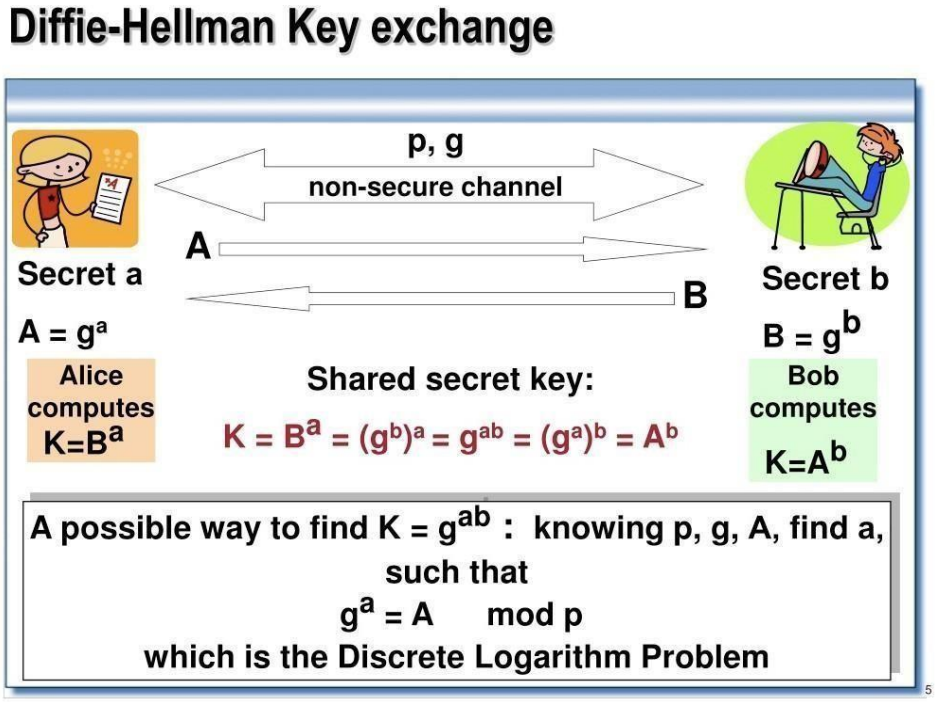
Theory:

One-Way Hash Functions:

- One-way hash functions (there are a lot of other names of functions of this type) transform input messages of various length into output sequences of fixed length (usually shorter).
- The output sequence is often called a hash value. Hash values are often used to mark input sequences, that is to assign to them some unique values that characterize them.
- One-way hash functions fulfil all conditions of one-way functions. It is easy
- to compute their values based on input data but having only a hash value one can't determine the original input sequence.
- A one-way hash function should be collision-free. This means that it should be very difficult to find two different sequences that produce the same hash value.
- Hash functions are used for storing data efficiently, in so-called hash tables. Data can be accessed by finding hash values, which are stored in computer memory.

Diffie-Hellman Key Exchange Protocol:

- It is an asymmetric cryptographic method used for key exchange or key agreement.
- It ensures that two or more communication partners agree on a common session key that everyone can use for encryption and decryption.
- With the typical key exchange methods of modern cryptography, the secret session key must be exchanged between two communication partners during the negotiation of the encryption.
- Only then can both sides encrypt and decrypt the data.



Conclusion

We have understood about One-way functions and Diffie-Hellman key exchange protocol.

Experiment 16

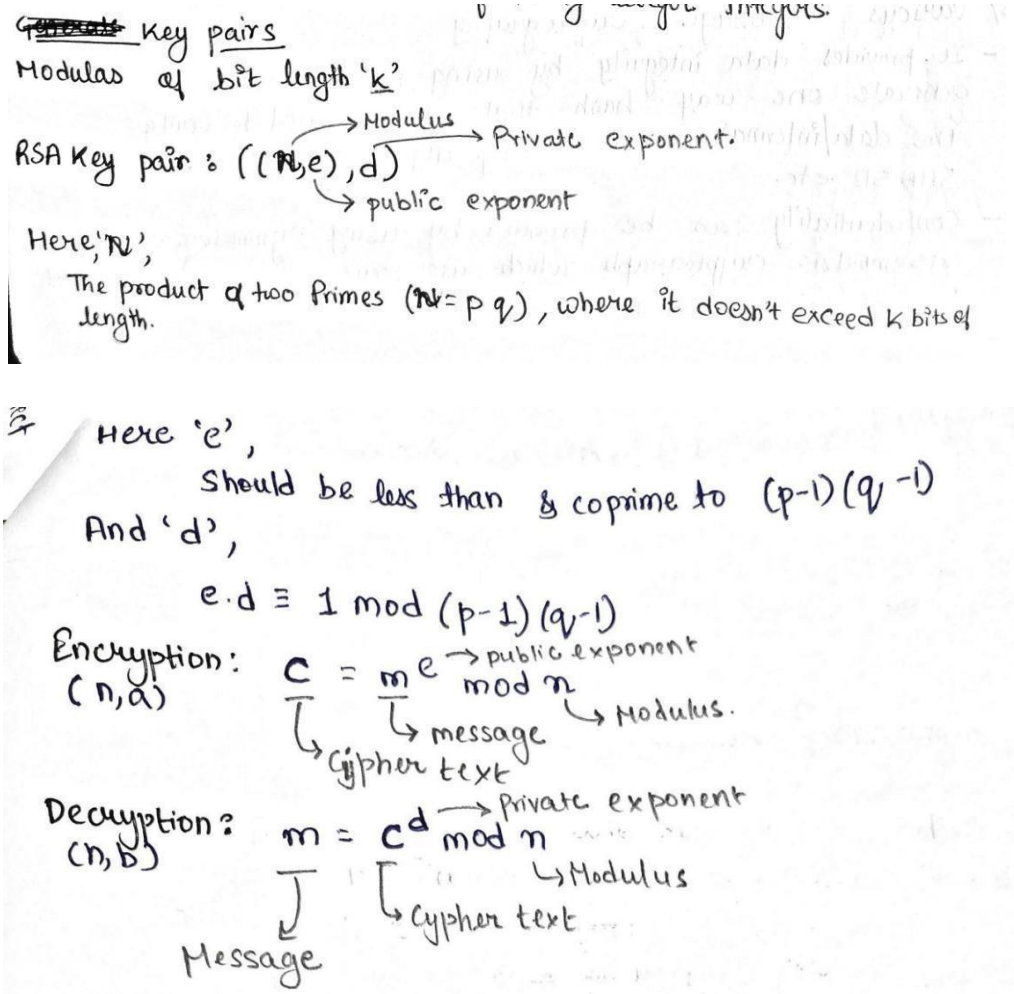
Aim:

Applications of asymmetric cryptography

- The RSA procedure

Theory:

- It is an asymmetric cryptography algorithm, this means that it uses a *public* key and a *private* key (i.e two different, mathematically linked keys).
- As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.
- The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.



Conclusion

Hence, we have understood RSA.