Practical: 1

Aim: Perform a practical to demonstrate ping of death (Denial of Service) attack in Ubuntu machine.

What Is a Ping of Death Attack?

The ping of death is a form of denial-of-service (DoS) attack that occurs when an attacker crashes, destabilizes, or freezes computers or services by targeting them with oversized data packets. This form of DoS attack typically targets and exploits legacy weaknesses that organizations may have patched.

How Does the Ping of Death Work?

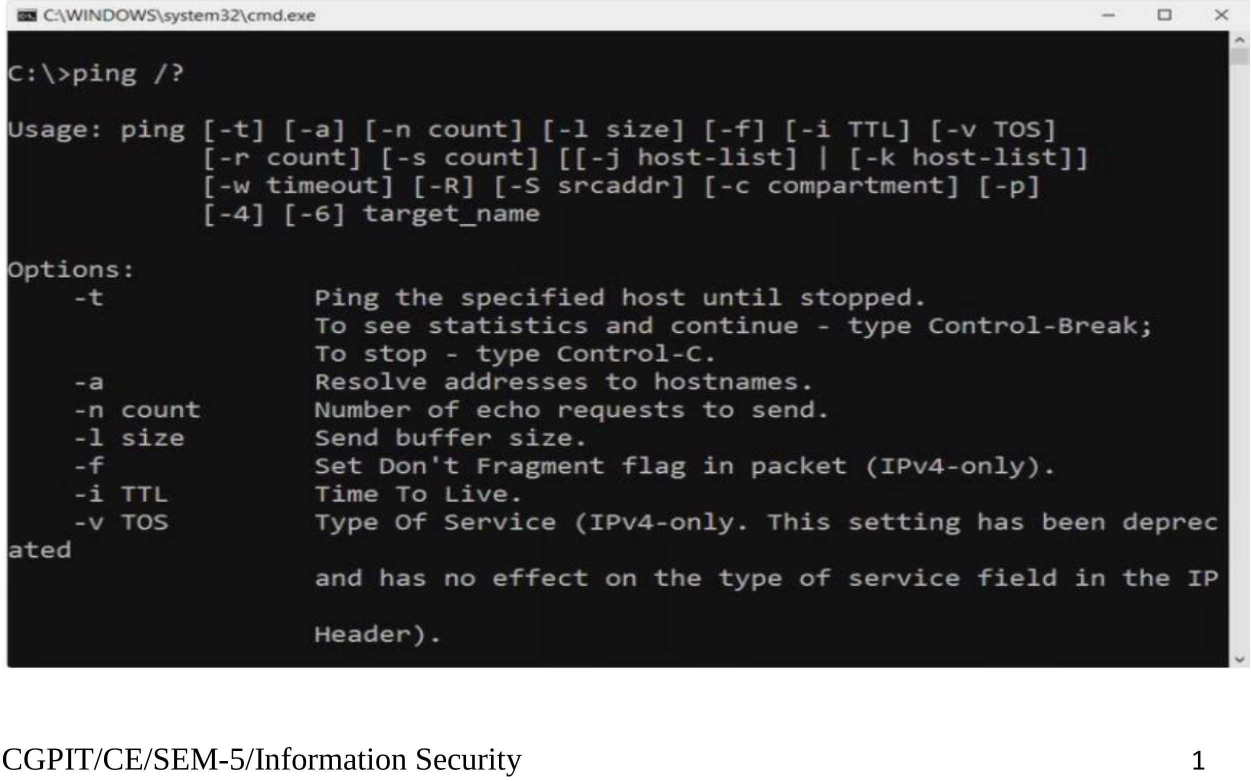
A correct Internet Protocol version 4 (IPv4) packet is formed of 65,535 bytes, and most legacy computers cannot handle larger packets. Sending a ping larger than this violates the IP, so attackers send packets in fragments which, when the targeted system attempts to reassemble, results in an oversized packet that can cause the system to crash, freeze, or reboot.

The vulnerability can be exploited by any source that sends IP datagrams, which include an ICMP echo, the Internetwork Packet Exchange (IPX), Transmission Control Protocol (TCP), and User Datagram Protocol (UDP).

Ping Command

The ping command is a Command Prompt command used to test the ability of the source computer to reach a specified destination computer. It's usually used as a simple way to verify that a computer can communicate over the network with another computer or network device.

Ping Command Availability

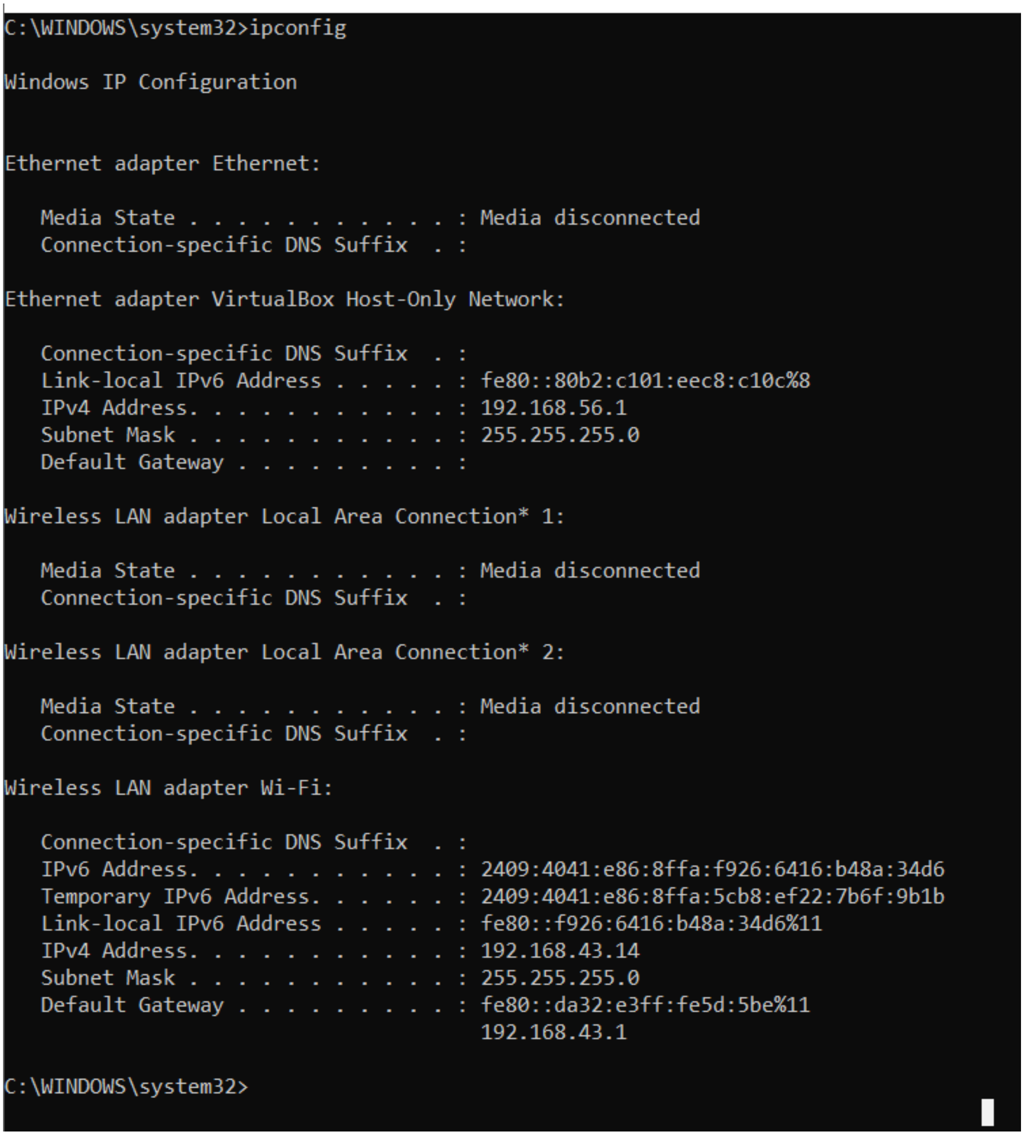


```
C:\>ping /?

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
            [-r count] [-s count] [[-j host-list] | [-k host-list]]
            [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
            [-4] [-6] target_name

Options:
    -t             Ping the specified host until stopped.
                   To see statistics and continue - type Control-Break;
                   To stop - type Control-C.
    -a             Resolve addresses to hostnames.
    -n count       Number of echo requests to send.
    -l size        Send buffer size.
    -f             Set Don't Fragment flag in packet (IPv4-only).
    -i TTL         Time To Live.
    -v TOS         Type Of Service (IPv4-only. This setting has been deprec
ated
                   and has no effect on the type of service field in the IP

                   Header).
```

Syntax:

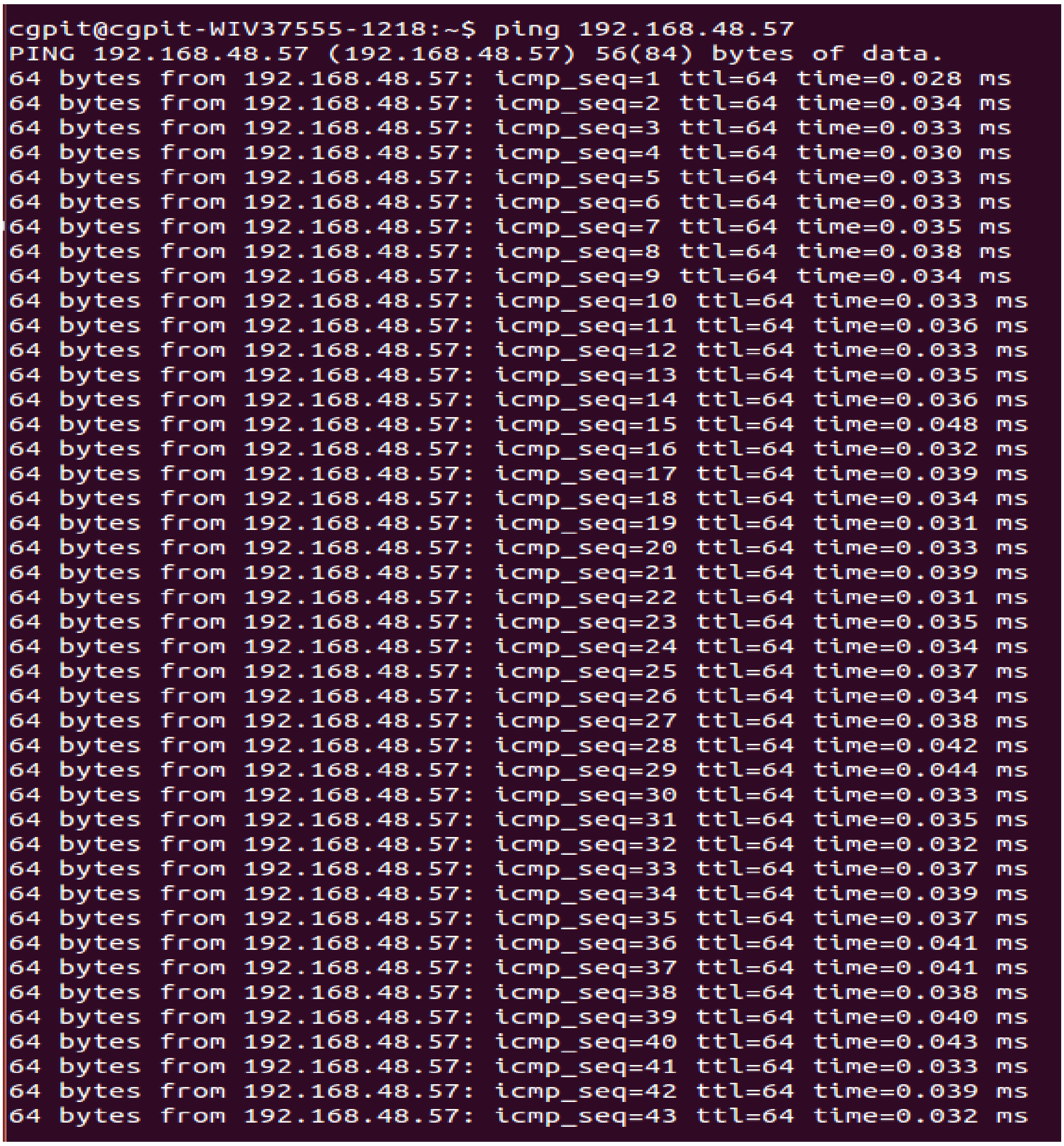
ping [**-t**] [**-a**] [**-n** *count*] [**-l** *size*] [**-f**] [**-i** *TTL*] [**-v** *TOS*] [**-r** *count*] [**-s** *count*] [**-w** *timeout*] [**-R**] [**-S** *srcaddr*] [**-p**] [**-4**] [**-6**] *target* [**/?**]

How to perform Dos attack?



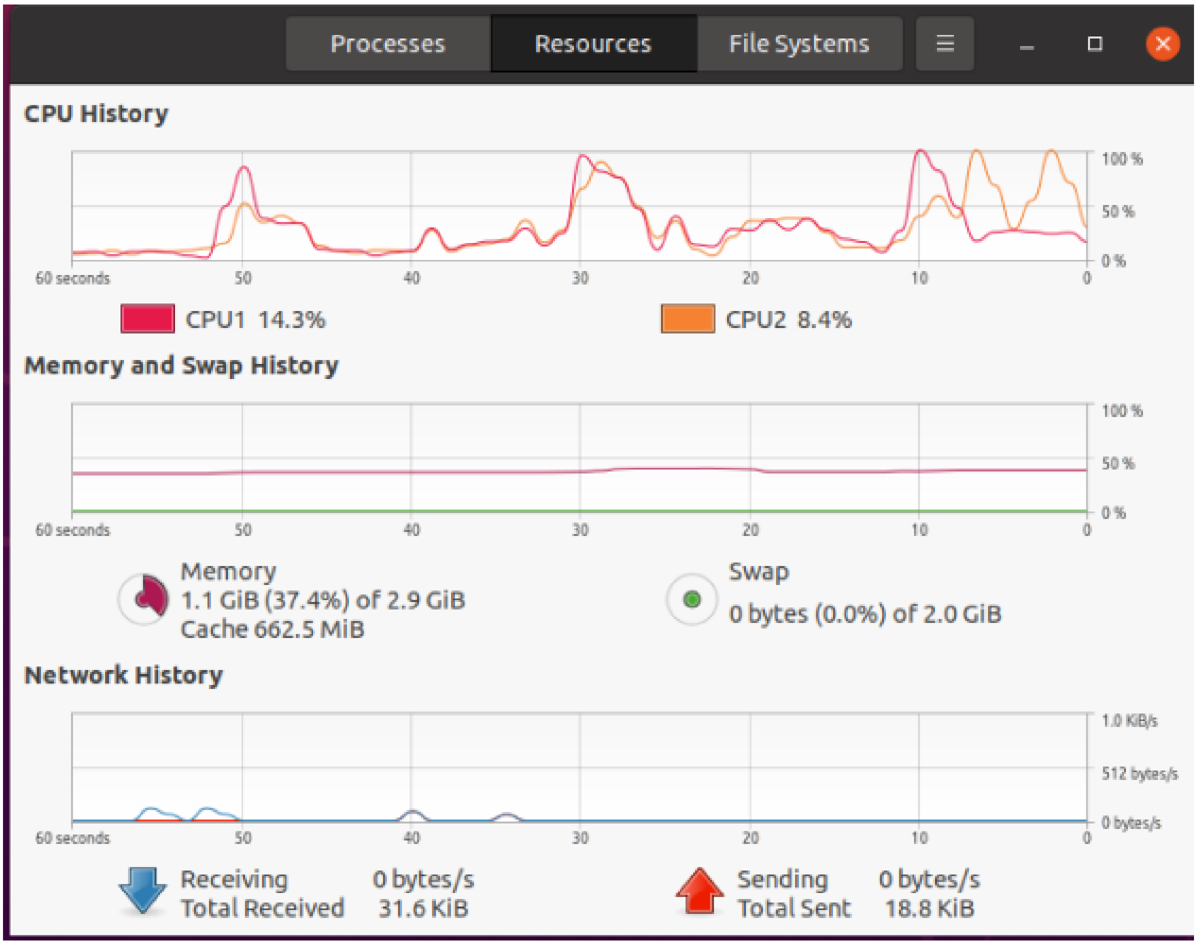
```
C:\WINDOWS\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter VirtualBox Host-Only Network:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::80b2:c101:eec8:c10c%8
   IPv4 Address. . . . . . . . . . . : 192.168.56.1
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2409:4041:e86:8ffa:f926:6416:b48a:34d6
   Temporary IPv6 Address. . . . . . : 2409:4041:e86:8ffa:5cb8:ef22:7b6f:9b1b
   Link-local IPv6 Address . . . . . : fe80::f926:6416:b48a:34d6%11
   IPv4 Address. . . . . . . . . . . : 192.168.43.14
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::da32:e3ff:fe5d:5be%11
                                       192.168.43.1

C:\WINDOWS\system32>
```

Send Ping:



```
cgpit@cgpit-WIV37555-1218:~$ ping 192.168.48.57
PING 192.168.48.57 (192.168.48.57) 56(84) bytes of data.
64 bytes from 192.168.48.57: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 192.168.48.57: icmp_seq=2 ttl=64 time=0.034 ms
64 bytes from 192.168.48.57: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=4 ttl=64 time=0.030 ms
64 bytes from 192.168.48.57: icmp_seq=5 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=6 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=7 ttl=64 time=0.035 ms
64 bytes from 192.168.48.57: icmp_seq=8 ttl=64 time=0.038 ms
64 bytes from 192.168.48.57: icmp_seq=9 ttl=64 time=0.034 ms
64 bytes from 192.168.48.57: icmp_seq=10 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=11 ttl=64 time=0.036 ms
64 bytes from 192.168.48.57: icmp_seq=12 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=13 ttl=64 time=0.035 ms
64 bytes from 192.168.48.57: icmp_seq=14 ttl=64 time=0.036 ms
64 bytes from 192.168.48.57: icmp_seq=15 ttl=64 time=0.048 ms
64 bytes from 192.168.48.57: icmp_seq=16 ttl=64 time=0.032 ms
64 bytes from 192.168.48.57: icmp_seq=17 ttl=64 time=0.039 ms
64 bytes from 192.168.48.57: icmp_seq=18 ttl=64 time=0.034 ms
64 bytes from 192.168.48.57: icmp_seq=19 ttl=64 time=0.031 ms
64 bytes from 192.168.48.57: icmp_seq=20 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=21 ttl=64 time=0.039 ms
64 bytes from 192.168.48.57: icmp_seq=22 ttl=64 time=0.031 ms
64 bytes from 192.168.48.57: icmp_seq=23 ttl=64 time=0.035 ms
64 bytes from 192.168.48.57: icmp_seq=24 ttl=64 time=0.034 ms
64 bytes from 192.168.48.57: icmp_seq=25 ttl=64 time=0.037 ms
64 bytes from 192.168.48.57: icmp_seq=26 ttl=64 time=0.034 ms
64 bytes from 192.168.48.57: icmp_seq=27 ttl=64 time=0.038 ms
64 bytes from 192.168.48.57: icmp_seq=28 ttl=64 time=0.042 ms
64 bytes from 192.168.48.57: icmp_seq=29 ttl=64 time=0.044 ms
64 bytes from 192.168.48.57: icmp_seq=30 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=31 ttl=64 time=0.035 ms
64 bytes from 192.168.48.57: icmp_seq=32 ttl=64 time=0.032 ms
64 bytes from 192.168.48.57: icmp_seq=33 ttl=64 time=0.037 ms
64 bytes from 192.168.48.57: icmp_seq=34 ttl=64 time=0.039 ms
64 bytes from 192.168.48.57: icmp_seq=35 ttl=64 time=0.037 ms
64 bytes from 192.168.48.57: icmp_seq=36 ttl=64 time=0.041 ms
64 bytes from 192.168.48.57: icmp_seq=37 ttl=64 time=0.041 ms
64 bytes from 192.168.48.57: icmp_seq=38 ttl=64 time=0.038 ms
64 bytes from 192.168.48.57: icmp_seq=39 ttl=64 time=0.040 ms
64 bytes from 192.168.48.57: icmp_seq=40 ttl=64 time=0.043 ms
64 bytes from 192.168.48.57: icmp_seq=41 ttl=64 time=0.033 ms
64 bytes from 192.168.48.57: icmp_seq=42 ttl=64 time=0.039 ms
64 bytes from 192.168.48.57: icmp_seq=43 ttl=64 time=0.032 ms
```

System monitor in Ubuntu:



🟥 CPU1 14.3%

🟧 CPU2 8.4%

Memory
1.1 GiB (37.4%) of 2.9 GiB
Cache 662.5 MiB

Swap
0 bytes (0.0%) of 2.0 GiB

Receiving	0 bytes/s		Sending	0 bytes/s
Total Received	31.6 KiB		Total Sent	18.8 KiB

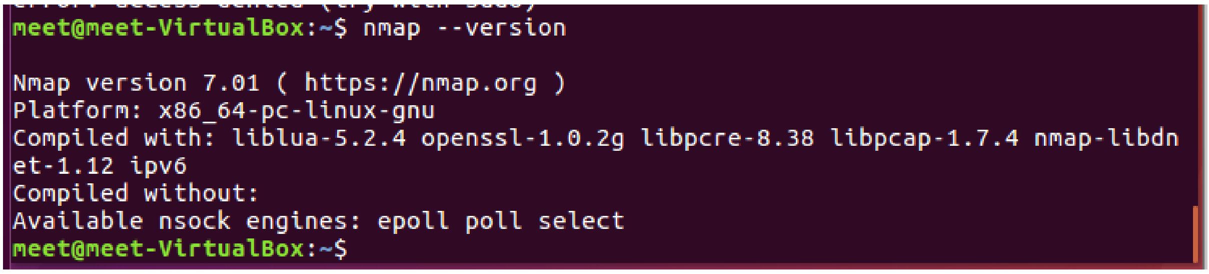
Practical: 2

Aim: Perform a practical to install network mapper tool and analyze the open ports in your Ubuntu machine.

Run a script to close all the insecure port, reopen and demonstrate

Install n-map tool:

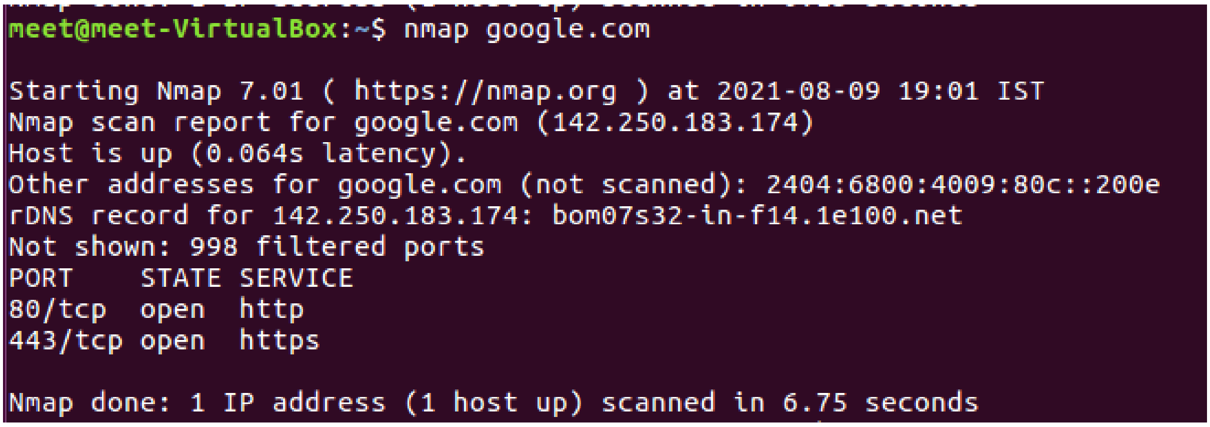
Nmap is a powerful network discovery and security auditing utility that is free, open-source, and easy to install. Nmap scans for vulnerabilities on your network, performs inventory checks, and monitors host or service uptime, alongside many other useful features



```
meet@meet-VirtualBox:~$ nmap --version

Nmap version 7.01 ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.2.4 openssl-1.0.2g libpcre-8.38 libpcap-1.7.4 nmap-libdn
et-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
meet@meet-VirtualBox:~$
```

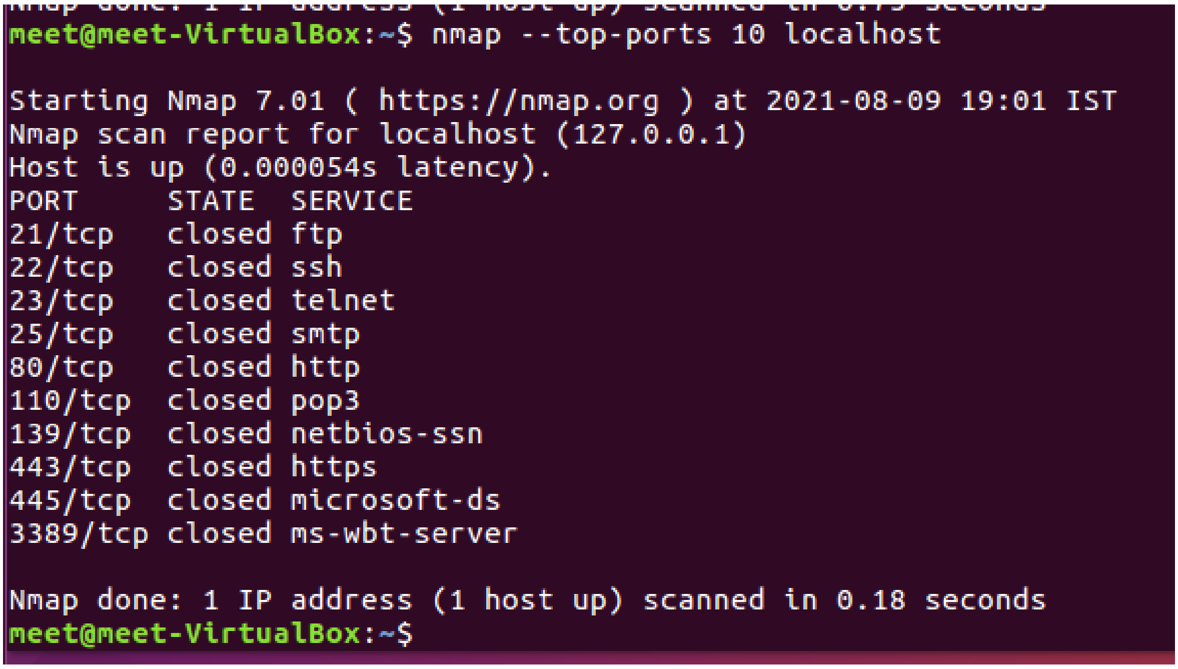
Analysis of the open port:

Nmap is a powerful and popular network exploration tool and port scanner. To install nmap on your system, use your default package manager as shown. To scan all open/listening ports in your Linux system, run the following command (which should take a long time to complete)



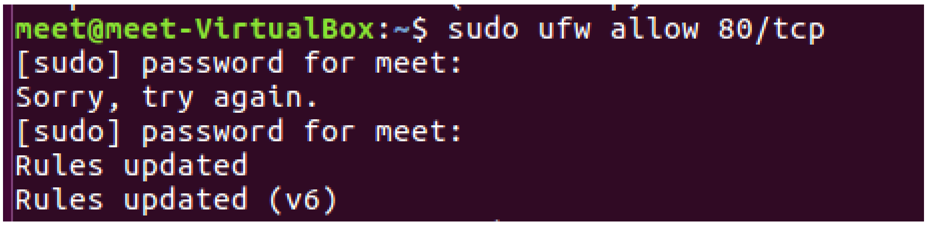
```
meet@meet-VirtualBox:~$ nmap google.com

Starting Nmap 7.01 ( https://nmap.org ) at 2021-08-09 19:01 IST
Nmap scan report for google.com (142.250.183.174)
Host is up (0.064s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:80c::200e
rDNS record for 142.250.183.174: bom07s32-in-f14.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 6.75 seconds
```

Close all the insecure port:

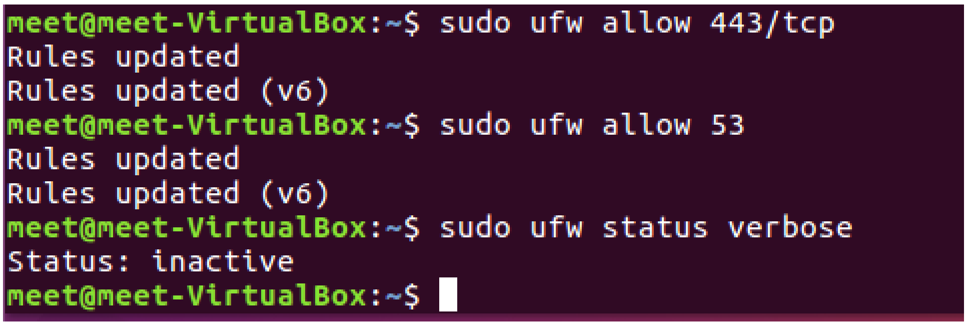


```
meet@meet-VirtualBox:~$ nmap --top-ports 10 localhost

Starting Nmap 7.01 ( https://nmap.org ) at 2021-08-09 19:01 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000054s latency).
PORT     STATE  SERVICE
21/tcp   closed ftp
22/tcp   closed ssh
23/tcp   closed telnet
25/tcp   closed smtp
80/tcp   closed http
110/tcp  closed pop3
139/tcp  closed netbios-ssn
443/tcp  closed https
445/tcp  closed microsoft-ds
3389/tcp closed ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
meet@meet-VirtualBox:~$
```

Re-Open ports:

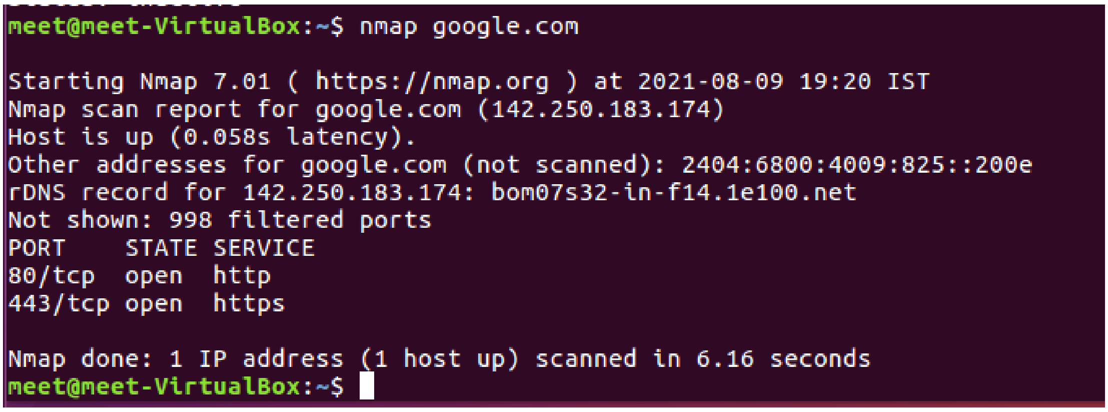


```
meet@meet-VirtualBox:~$ sudo ufw allow 80/tcp
[sudo] password for meet:
Sorry, try again.
[sudo] password for meet:
Rules updated
Rules updated (v6)
```



```
meet@meet-VirtualBox:~$ sudo ufw allow 443/tcp
Rules updated
Rules updated (v6)
meet@meet-VirtualBox:~$ sudo ufw allow 53
Rules updated
Rules updated (v6)
meet@meet-VirtualBox:~$ sudo ufw status verbose
Status: inactive
meet@meet-VirtualBox:~$
```

Reopen ports and demonstrate:



```
meet@meet-VirtualBox:~$ nmap google.com

Starting Nmap 7.01 ( https://nmap.org ) at 2021-08-09 19:20 IST
Nmap scan report for google.com (142.250.183.174)
Host is up (0.058s latency).
Other addresses for google.com (not scanned): 2404:6800:4009:825::200e
rDNS record for 142.250.183.174: bom07s32-in-f14.1e100.net
Not shown: 998 filtered ports
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 6.16 seconds
meet@meet-VirtualBox:~$
```

Practical: 3

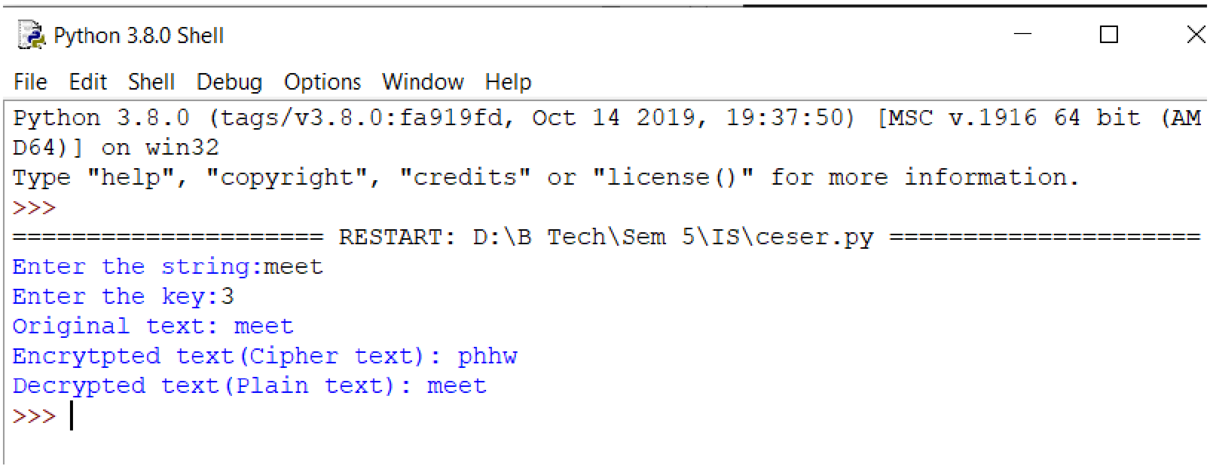
Aim: Perform a practical to implement Caesar cipher and play fair cipher.

Caesar cipher:

Code:

```
pt = str(input("Enter the string:"))
key = int(input("Enter the key:"))
ct = ""
dt = ""
print("Original text:", pt)

for letter in pt:
    l = (ord(letter)+key % 26)
    ct += chr(l)
print("Encrytpted text(Cipher text):", ct)

for letter in ct:
    l = (ord(letter)-key % 26)
    dt += chr(l)
print("Decrypted text(Plain text):", dt)
```

Output:



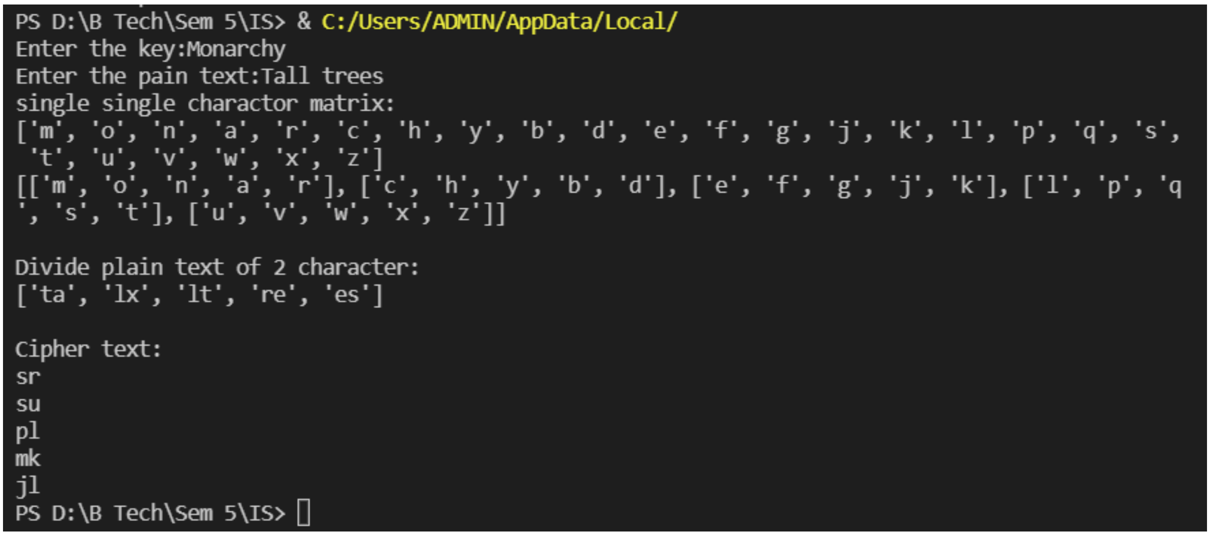
```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
===================== RESTART: D:\B Tech\Sem 5\IS\ceser.py =====================
Enter the string:meet
Enter the key:3
Original text: meet
Encrytpted text(Cipher text): phhw
Decrypted text(Plain text): meet
>>> 
```

Play fair cipher:

Code:

```
key = input("Enter the key:")
plain_text = input("Enter the pain text:")
if len(key) > 0 and len(plain_text) > 0:
    plain_text = plain_text.replace(" ", "").lower()
    group = list()

    matrix = list()
    original_5_5_matrix = list()

    list1 = list()

    # check key for value plain text
    for a in key.lower():
        if a not in matrix:
            matrix.append(a)

    # check character i and j an pattern set and check all alphabet insert but
    # check plain text value key same value
    for char in key:
        if char == 'i':
            alphabet = "abcdefghiklmnopqrstuvwxyz"
        else:
            alphabet = "abcdefghjklmnopqrstuvwxyz"
    for a in alphabet:
        if a not in matrix:
            matrix.append(a)
    print('single single charactor matrix:')
    print(matrix)
    original_5_5_matrix = [matrix[i:i+5] for i in range(0, len(matrix), 5)]
    print(original_5_5_matrix)
    print()

    # same alphabet insert x

    def change_pt(i, pt):
        pt = pt[:i]+"x"+pt[i:]
        return pt

    # divide pain text 2 character an list all 2 charator inserted for 2 part
    for i in range(0, len(plain_text), 2):
        if i == len(plain_text):
            pass
        else:
            if plain_text[i] == plain_text[i+1]:
                plain_text = change_pt(i+1, plain_text)
        group.append(plain_text[i]+plain_text[i+1])
    print("Divide plain text of 2 character:")
    print(group)
    print()

    print("Cipher text:")
    index_of_w1 = None
    index_of_w2 = None
    for word in group:
        for i in range(5):
            for j in range(5):
                if word[0] in original_5_5_matrix[i][j]:
                    # first letter index for column and row
                    index_of_w1 = [i, j]
                if word[1] in original_5_5_matrix[i][j]:
                    # second letter index for column and row
                    index_of_w2 = [i, j]
        if index_of_w1 != None and index_of_w2 != None:
            # for same row
            if index_of_w1[0] == index_of_w2[0]:

                if index_of_w1[1] == 4:  # first letter : last position of row index
                    print(original_5_5_matrix[index_of_w1[0]][0] +
                          original_5_5_matrix[index_of_w2[0]][index_of_w2[1]+1])

                elif index_of_w2[1] == 4:  # second letter : last position of row index
                    print(original_5_5_matrix[index_of_w1[0]]
                          [index_of_w1[1]+1] + original_5_5_matrix[index_of_w2[0]][0])

                else:
                    print(original_5_5_matrix[index_of_w1[0]][index_of_w1[1]+1] +
                          original_5_5_matrix[index_of_w2[0]][index_of_w2[1]+1])

            # for same column
            # first letter : last position of column index
            elif index_of_w1[1] == index_of_w2[1]:
                if index_of_w1[0] == 4:
                    print(original_5_5_matrix[0][index_of_w1[1]] +
                          original_5_5_matrix[index_of_w2[0]+1][index_of_w2[1]])

                # second letter : last position of column index
                elif index_of_w2[0] == 4:
                    print(original_5_5_matrix[index_of_w1[0]+1]
                          [index_of_w1[1]] + original_5_5_matrix[0][index_of_w2[1]])
                else:
                    print(original_5_5_matrix[index_of_w1[0]+1][index_of_w1[1]] +
                          original_5_5_matrix[index_of_w2[0]+1][index_of_w2[1]])

            # otherwise
            else:
                print(original_5_5_matrix[index_of_w1[0]][index_of_w2[1]] +
                      original_5_5_matrix[index_of_w2[0]][index_of_w1[1]])
else:
    print("Key and Pain text are required!")
```

Output:

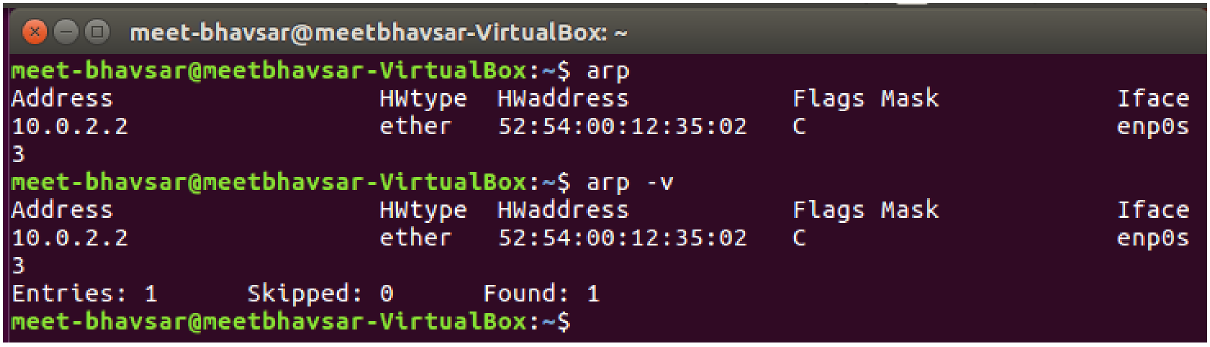


```
PS D:\B Tech\Sem 5\IS> & C:/Users/ADMIN/AppData/Local/
Enter the key:Monarchy
Enter the pain text:Tall trees
single single charactor matrix:
['m', 'o', 'n', 'a', 'r', 'c', 'h', 'y', 'b', 'd', 'e', 'f', 'g', 'j', 'k', 'l', 'p', 'q', 's',
 't', 'u', 'v', 'w', 'x', 'z']
[['m', 'o', 'n', 'a', 'r'], ['c', 'h', 'y', 'b', 'd'], ['e', 'f', 'g', 'j', 'k'], ['l', 'p', 'q
', 's', 't'], ['u', 'v', 'w', 'x', 'z']]

Divide plain text of 2 character:
['ta', 'lx', 'lt', 're', 'es']

Cipher text:
sr
su
pl
mk
jl
PS D:\B Tech\Sem 5\IS> 
```

Practical: 4

Aim: Perform a practical to demonstrate important Linux command of information security.

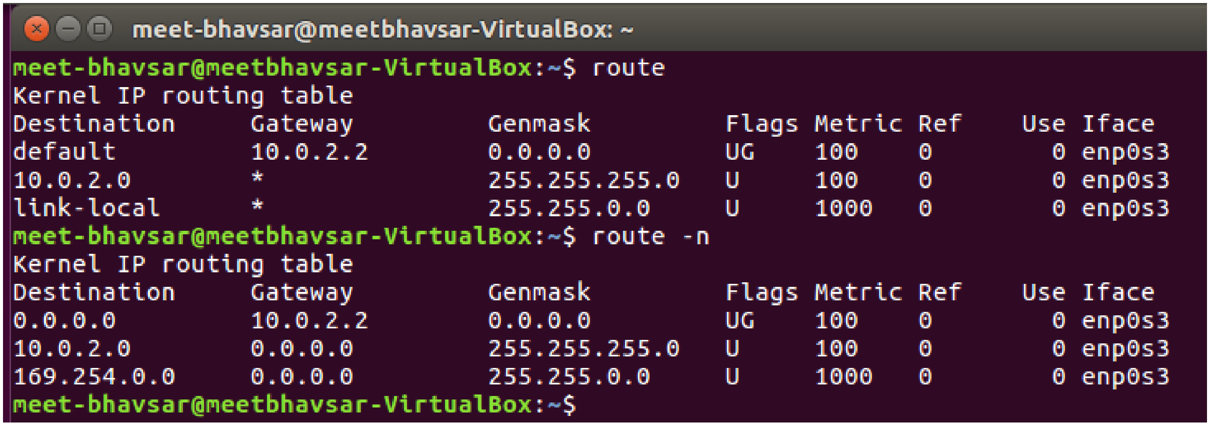
arp: arp command manipulates the System's ARP cache. It also allows a complete dump of the ARP cache. ARP stands for Address Resolution Protocol. The primary function of this protocol is to resolve the IP address of a system to its mac address, and hence it works between level 2(Data link layer) and level 3(Network layer).



meet-bhavsar@meetbhavsar-VirtualBox: ~

```
meet-bhavsar@meetbhavsar-VirtualBox:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.2                 ether   52:54:00:12:35:02   C                     enp0s
3
meet-bhavsar@meetbhavsar-VirtualBox:~$ arp -v
Address                  HWtype  HWaddress           Flags Mask            Iface
10.0.2.2                 ether   52:54:00:12:35:02   C                     enp0s
3
Entries: 1      Skipped: 0      Found: 1
meet-bhavsar@meetbhavsar-VirtualBox:~$
```

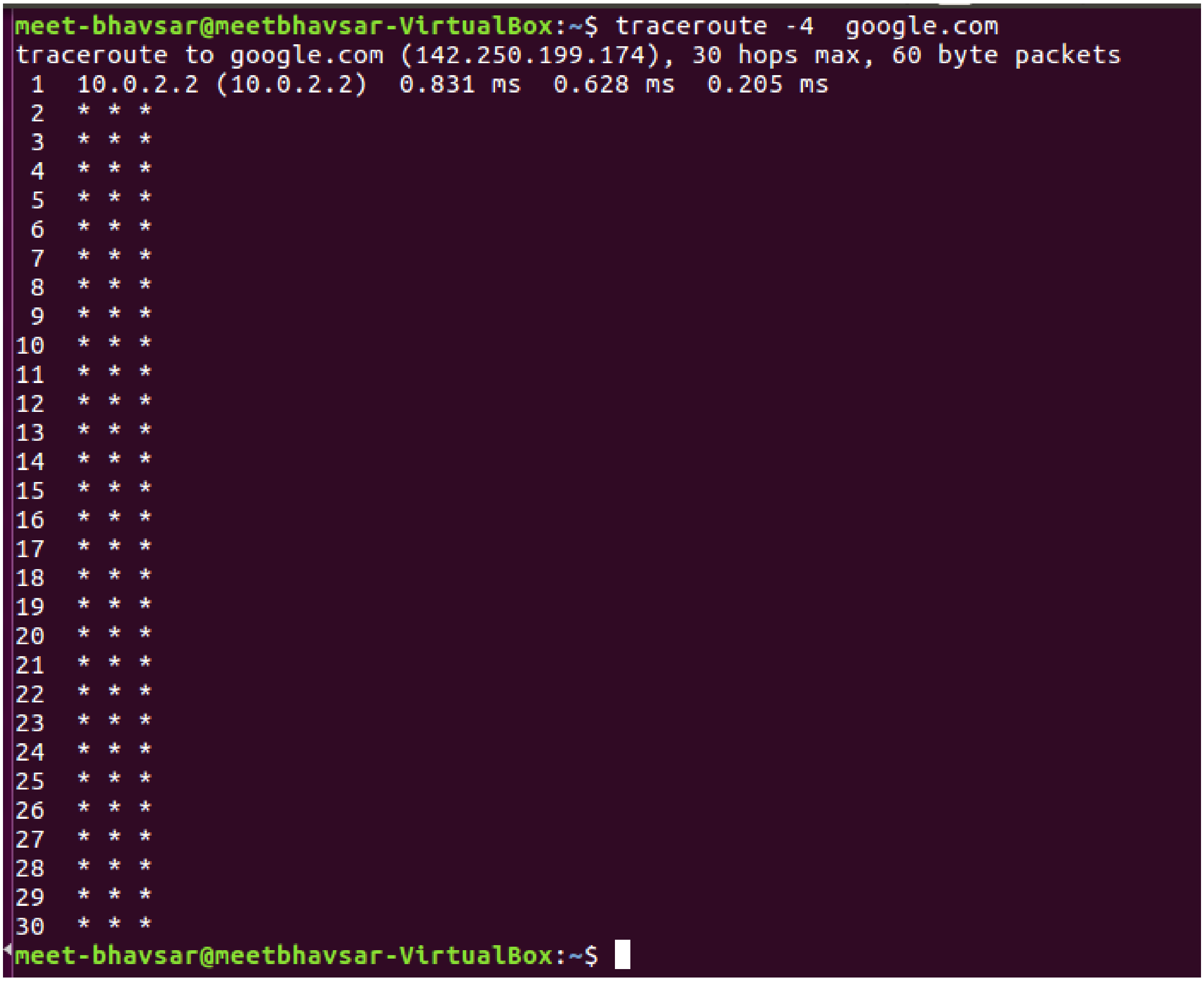
route: The route command allows you to make manual entries into the network routing tables. The route command distinguishes between routes to hosts and routes to networks by interpreting the network address of the Destination variable, which can be specified either by symbolic name or numeric address.



meet-bhavsar@meetbhavsar-VirtualBox: ~

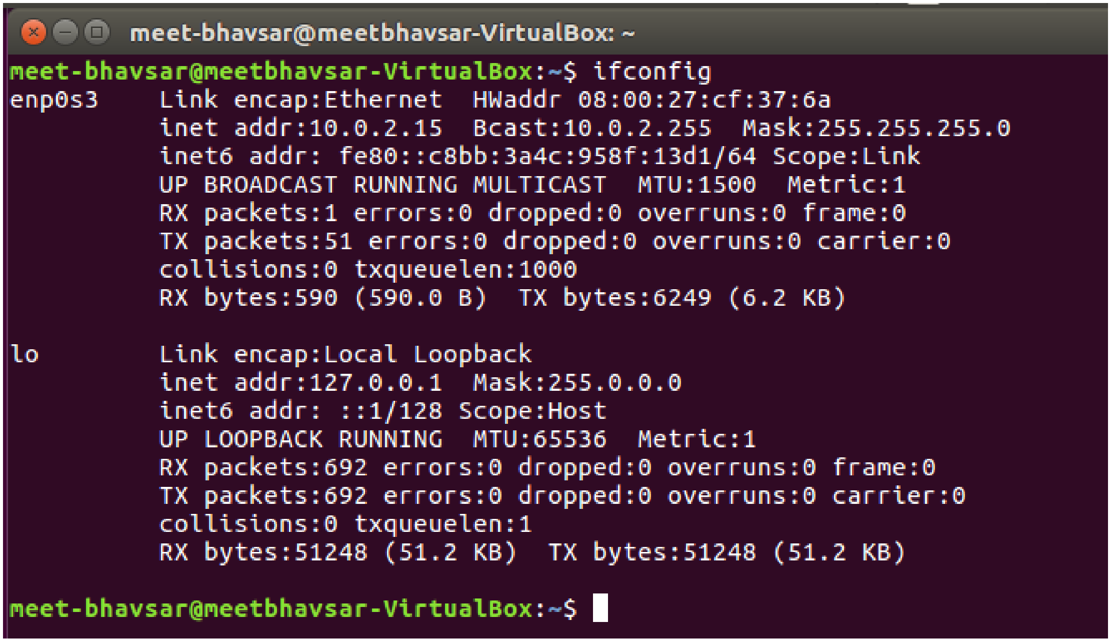
```
meet-bhavsar@meetbhavsar-VirtualBox:~$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        *               255.255.255.0   U     100    0        0 enp0s3
link-local      *               255.255.0.0     U     1000   0        0 enp0s3
meet-bhavsar@meetbhavsar-VirtualBox:~$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.0.2.2        0.0.0.0         UG    100    0        0 enp0s3
10.0.2.0        0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 enp0s3
meet-bhavsar@meetbhavsar-VirtualBox:~$
```

traceroute: traceroute command in Linux prints the route that a packet takes to reach the host. This command is useful when you want to know about the route and about all the hops that a packet takes.



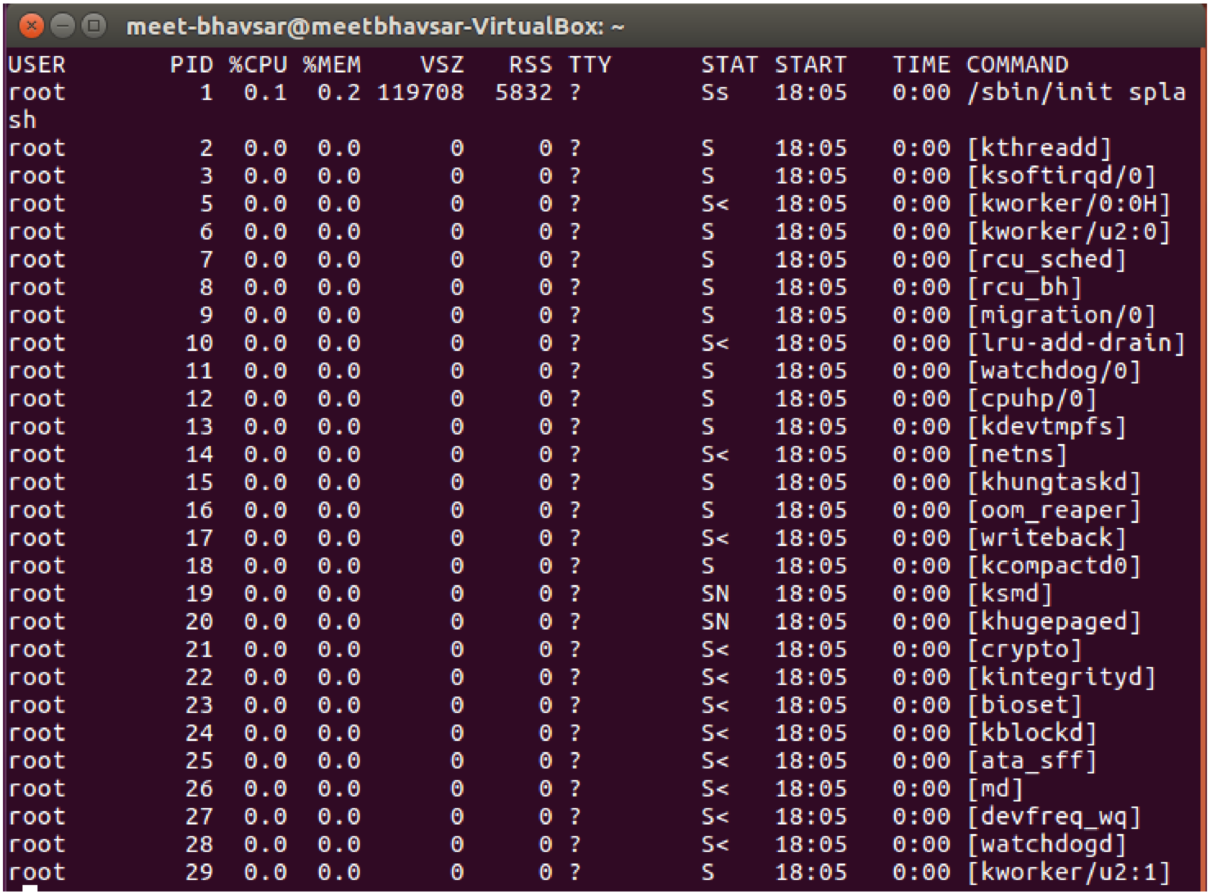
```
meet-bhavsar@meetbhavsar-VirtualBox:~$ traceroute -4  google.com
traceroute to google.com (142.250.199.174), 30 hops max, 60 byte packets
 1  10.0.2.2 (10.0.2.2)  0.831 ms  0.628 ms  0.205 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
meet-bhavsar@meetbhavsar-VirtualBox:~$
```

ifconfig: You can use the ifconfig command to assign an address to a network interface and to configure or display the current network interface configuration information.



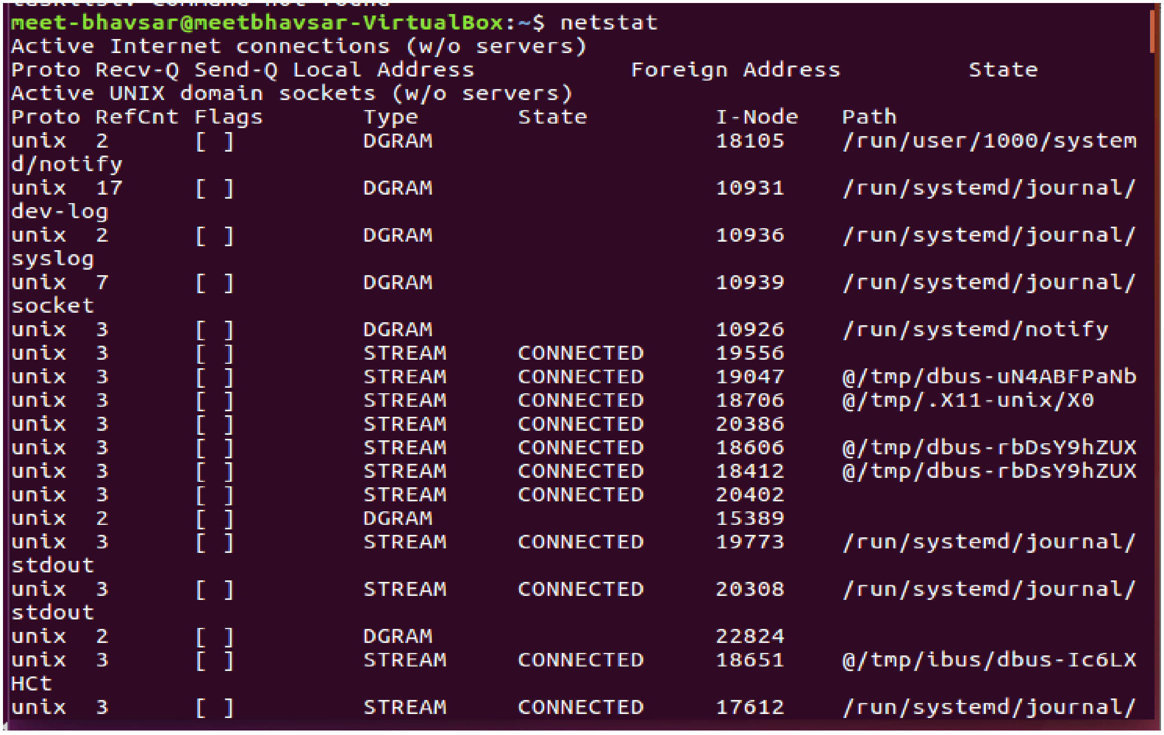
```
meet-bhavsar@meetbhavsar-VirtualBox:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:cf:37:6a
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::c8bb:3a4c:958f:13d1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:51 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:590 (590.0 B)  TX bytes:6249 (6.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:692 errors:0 dropped:0 overruns:0 frame:0
          TX packets:692 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:51248 (51.2 KB)  TX bytes:51248 (51.2 KB)

meet-bhavsar@meetbhavsar-VirtualBox:~$
```

tasklist: You can use the tasklist command to display a list of currently-running tasks. tasklist displays the process ID number for each running task, the name of the executable program that started the task, and, when available, the window title.



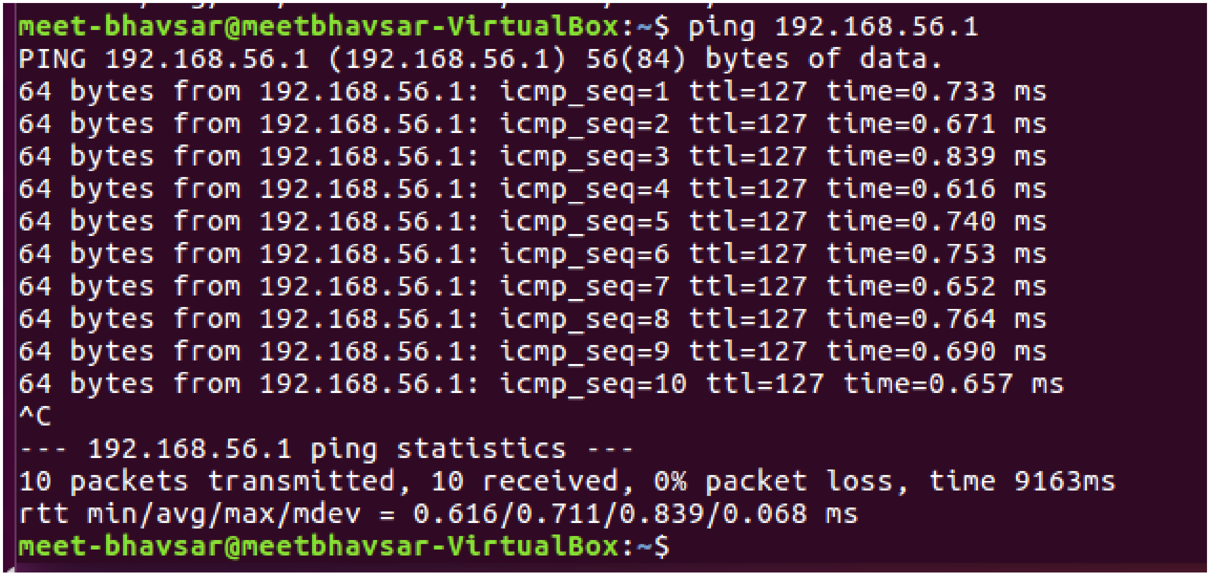
```
meet-bhavsar@meetbhavsar-VirtualBox: ~
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.2 119708  5832 ?        Ss   18:05   0:00 /sbin/init spla
sh
root           2  0.0  0.0      0     0 ?        S    18:05   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    18:05   0:00 [ksoftirqd/0]
root           5  0.0  0.0      0     0 ?        S<   18:05   0:00 [kworker/0:0H]
root           6  0.0  0.0      0     0 ?        S    18:05   0:00 [kworker/u2:0]
root           7  0.0  0.0      0     0 ?        S    18:05   0:00 [rcu_sched]
root           8  0.0  0.0      0     0 ?        S    18:05   0:00 [rcu_bh]
root           9  0.0  0.0      0     0 ?        S    18:05   0:00 [migration/0]
root          10  0.0  0.0      0     0 ?        S<   18:05   0:00 [lru-add-drain]
root          11  0.0  0.0      0     0 ?        S    18:05   0:00 [watchdog/0]
root          12  0.0  0.0      0     0 ?        S    18:05   0:00 [cpuhp/0]
root          13  0.0  0.0      0     0 ?        S    18:05   0:00 [kdevtmpfs]
root          14  0.0  0.0      0     0 ?        S<   18:05   0:00 [netns]
root          15  0.0  0.0      0     0 ?        S    18:05   0:00 [khungtaskd]
root          16  0.0  0.0      0     0 ?        S    18:05   0:00 [oom_reaper]
root          17  0.0  0.0      0     0 ?        S<   18:05   0:00 [writeback]
root          18  0.0  0.0      0     0 ?        S    18:05   0:00 [kcompactd0]
root          19  0.0  0.0      0     0 ?        SN   18:05   0:00 [ksmd]
root          20  0.0  0.0      0     0 ?        SN   18:05   0:00 [khugepaged]
root          21  0.0  0.0      0     0 ?        S<   18:05   0:00 [crypto]
root          22  0.0  0.0      0     0 ?        S<   18:05   0:00 [kintegrityd]
root          23  0.0  0.0      0     0 ?        S<   18:05   0:00 [bioset]
root          24  0.0  0.0      0     0 ?        S<   18:05   0:00 [kblockd]
root          25  0.0  0.0      0     0 ?        S<   18:05   0:00 [ata_sff]
root          26  0.0  0.0      0     0 ?        S<   18:05   0:00 [md]
root          27  0.0  0.0      0     0 ?        S<   18:05   0:00 [devfreq_wq]
root          28  0.0  0.0      0     0 ?        S<   18:05   0:00 [watchdogd]
root          29  0.0  0.0      0     0 ?        S    18:05   0:00 [kworker/u2:1]
```

netstat: Netstat command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships.



```
meet-bhavsar@meetbhavsar-VirtualBox:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ]         DGRAM                    18105    /run/user/1000/system
d/notify
unix  17     [ ]         DGRAM                    10931    /run/systemd/journal/
dev-log
unix  2      [ ]         DGRAM                    10936    /run/systemd/journal/
syslog
unix  7      [ ]         DGRAM                    10939    /run/systemd/journal/
socket
unix  3      [ ]         DGRAM                    10926    /run/systemd/notify
unix  3      [ ]         STREAM     CONNECTED     19556
unix  3      [ ]         STREAM     CONNECTED     19047    @/tmp/dbus-uN4ABFPaNb
unix  3      [ ]         STREAM     CONNECTED     18706    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     20386
unix  3      [ ]         STREAM     CONNECTED     18606    @/tmp/dbus-rbDsY9hZUX
unix  3      [ ]         STREAM     CONNECTED     18412    @/tmp/dbus-rbDsY9hZUX
unix  3      [ ]         STREAM     CONNECTED     20402
unix  2      [ ]         DGRAM                    15389
unix  3      [ ]         STREAM     CONNECTED     19773    /run/systemd/journal/
stdout
unix  3      [ ]         STREAM     CONNECTED     20308    /run/systemd/journal/
stdout
unix  2      [ ]         DGRAM                    22824
unix  3      [ ]         STREAM     CONNECTED     18651    @/tmp/ibus/dbus-Ic6LX
HCt
unix  3      [ ]         STREAM     CONNECTED     17612    /run/systemd/journal/
```

ping: PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host. This command takes as input the IP address or the URL and sends a data packet to the specified address with the message “PING” and get a response from the server/host this time is recorded which is called latency.



```
meet-bhavsar@meetbhavsar-VirtualBox:~$ ping 192.168.56.1
PING 192.168.56.1 (192.168.56.1) 56(84) bytes of data.
64 bytes from 192.168.56.1: icmp_seq=1 ttl=127 time=0.733 ms
64 bytes from 192.168.56.1: icmp_seq=2 ttl=127 time=0.671 ms
64 bytes from 192.168.56.1: icmp_seq=3 ttl=127 time=0.839 ms
64 bytes from 192.168.56.1: icmp_seq=4 ttl=127 time=0.616 ms
64 bytes from 192.168.56.1: icmp_seq=5 ttl=127 time=0.740 ms
64 bytes from 192.168.56.1: icmp_seq=6 ttl=127 time=0.753 ms
64 bytes from 192.168.56.1: icmp_seq=7 ttl=127 time=0.652 ms
64 bytes from 192.168.56.1: icmp_seq=8 ttl=127 time=0.764 ms
64 bytes from 192.168.56.1: icmp_seq=9 ttl=127 time=0.690 ms
64 bytes from 192.168.56.1: icmp_seq=10 ttl=127 time=0.657 ms
^C
--- 192.168.56.1 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9163ms
rtt min/avg/max/mdev = 0.616/0.711/0.839/0.068 ms
meet-bhavsar@meetbhavsar-VirtualBox:~$
```

Practical: 7

Aim: Implement S-DES Key generation and Encryption.

Code:

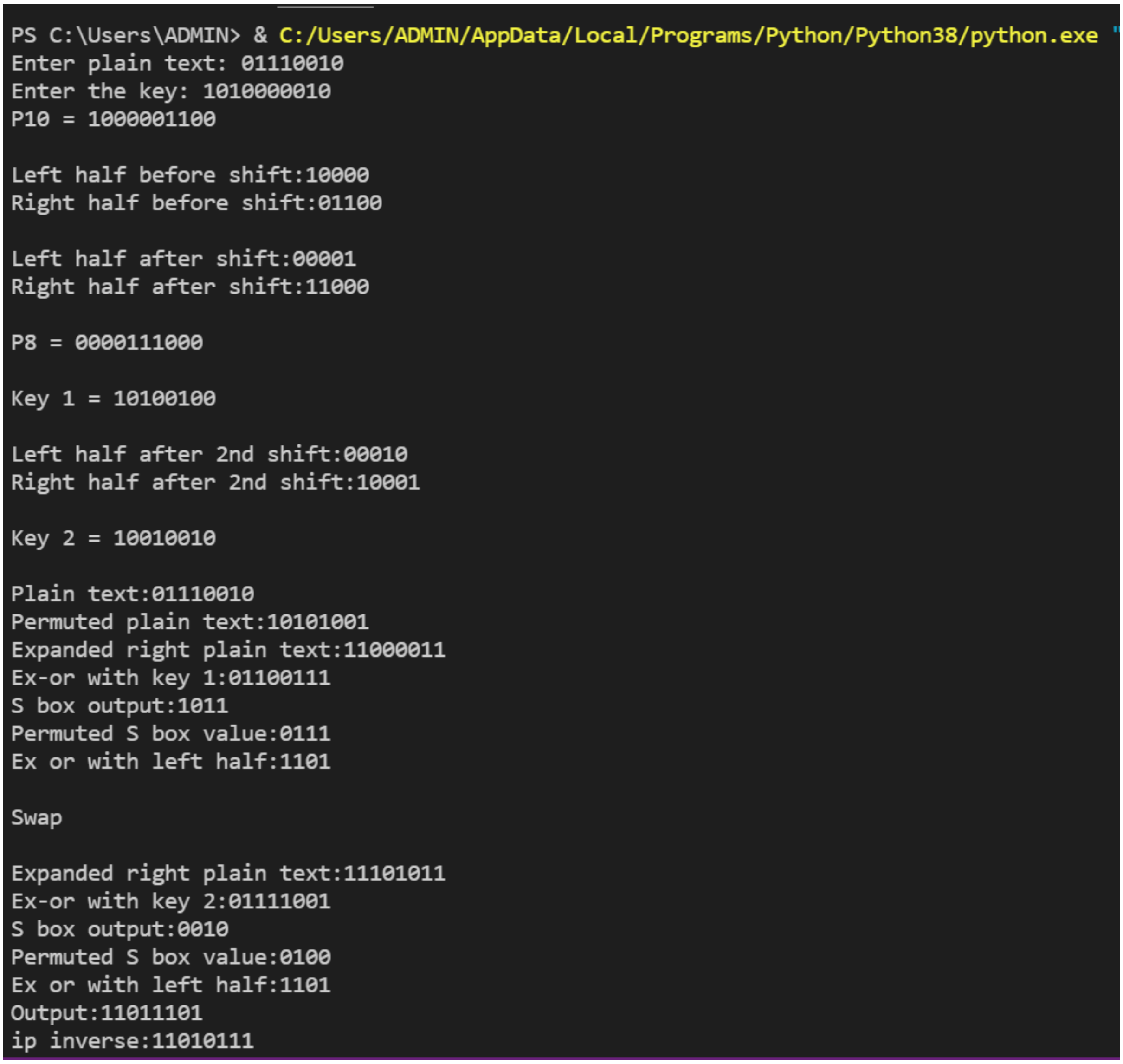
```
# plain_text = "01110010"
# key = "1010000010"
plain_text = str(input("Enter plain text: "))
key = str(input("Enter the key: "))
p10 = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]
p8 = [6, 3, 7, 4, 8, 5, 10, 9]
p4 = [2, 3, 4, 1]
ip = [2, 6, 3, 1, 4, 8, 5, 7]
ip_inv = [4, 1, 3, 5, 7, 2, 8, 6]
ep = [4, 1, 2, 3, 2, 3, 4, 1]

s0_box = [['01', '00', '11', '10'], ['11', '10', '01', '00'],
          ['00', '10', '01', '11'], ['11', '01', '11', '10']]
s1_box = [['00', '01', '10', '11'], ['10', '00', '01', '11'],
          ['11', '00', '01', '00'], ['10', '01', '00', '11']]

p10_per, key1, key2, per_pt, exp_right_pt = "", "", "", "", ""
left_s0, right_s1, per_sbox_output, ip_inverse = "", "", "", ""

def permutation(per_table, key):
    a = ""
    for i in per_table:
        a = a+(key[i-1])
    return a

p10_per = permutation(p10, key)
print("P10 = "+p10_per+"\n")

left_half = p10_per[:5]
right_half = p10_per[5:]

print("Left half before shift:"+left_half)
print("Right half before shift:"+right_half+"\n")

def shift(left_half, d):
    Lfirst = left_half[0: d]
    Lsecond = left_half[d:]
    return Lsecond + Lfirst

left_half = shift(left_half, 1)
right_half = shift(right_half, 1)
```

```
print("Left half after shift:"+left_half)
print("Right half after shift:"+right_half+"\n")

p8_key = left_half+right_half
print("P8 = "+p8_key+"\n")

key1 = permutation(p8, p8_key)

print("Key 1 = "+key1+"\n")

left_half = shift(left_half, 1)
right_half = shift(right_half, 1)

print("Left half after 2nd shift:"+left_half)
print("Right half after 2nd shift:"+right_half+"\n")

p8_key2 = left_half+right_half

key2 = permutation(p8, p8_key2)

print("Key 2 = "+key2+"\n")

# Encryption
print("Plain text:"+str(plain_text))

per_pt = permutation(ip, plain_text)

print("Permuted plain text:"+per_pt)

left_pt = per_pt[:4]
right_pt = per_pt[4:]

exp_right_pt = permutation(ep, right_pt)
print("Expanded right plain text:"+exp_right_pt)

def ex_or_operation(length, first_word, second_word):
    ex_or = ""
    for i in range(0, len(length)):
        if str(first_word[i]) == "1" and str(second_word[i]) == "1" or str(first_word[i]) == "0"
and str(second_word[i]) == "0":
            ex_or += "0"
        else:
            ex_or += "1"
    return ex_or
```

```
ex_or = ex_or_operation(ip, key1, exp_right_pt)
print("Ex-or with key 1:"+ex_or)

left_exor = ex_or[:4]
right_exor = ex_or[4:]

a = int((left_exor[0]+left_exor[3]), 2)
b = int((left_exor[1]+left_exor[2]), 2)
left_s0 = s0_box[a][b]

c = int((right_exor[0]+right_exor[3]), 2)
d = int((right_exor[1]+right_exor[2]), 2)
right_s1 = s1_box[c][d]
s_box_output = str(left_s0)+str(right_s1)
print("S box output:"+str(s_box_output))

per_sbox_output = permutation(p4, s_box_output)
print("Permuted S box value:"+str(per_sbox_output))

ex_or2 = ex_or_operation(left_pt, left_pt, per_sbox_output)

print("Ex or with left half:"+str(ex_or2)+"\n")

# swap
print("Swap\n")
left_pt2 = right_pt
right_pt2 = ex_or2
exp_right_pt, per_sbox_output = "", ""
exp_right_pt = permutation(ep, right_pt2)

print("Expanded right plain text:"+exp_right_pt)

ex_or = ex_or_operation(key2, key2, exp_right_pt)
print("Ex-or with key 2:"+ex_or)
left_exor = ex_or[:4]
right_exor = ex_or[4:]

a = int((left_exor[0]+left_exor[3]), 2)
b = int((left_exor[1]+left_exor[2]), 2)
left_s0 = s0_box[a][b]

c = int((right_exor[0]+right_exor[3]), 2)
d = int((right_exor[1]+right_exor[2]), 2)
right_s1 = s1_box[c][d]

s_box_output = str(left_s0)+str(right_s1)
```

```
print("S box output:"+str(s_box_output))

per_sbox_output = permutation(p4, s_box_output)
print("Permuted S box value:"+str(per_sbox_output))

ex_or2 = ex_or_operation(per_sbox_output, per_sbox_output, left_pt2)
print("Ex or with left half:"+str(ex_or2))

str1 = ex_or2+right_pt2

print("Output:"+str(str1))

ip_inverse = permutation(ip_inv, str1)
print("ip inverse:"+str(ip_inverse))
```

Output:



```
PS C:\Users\ADMIN> & C:/Users/ADMIN/AppData/Local/Programs/Python/Python38/python.exe "
Enter plain text: 01110010
Enter the key: 1010000010
P10 = 1000001100

Left half before shift:10000
Right half before shift:01100

Left half after shift:00001
Right half after shift:11000

P8 = 0000111000

Key 1 = 10100100

Left half after 2nd shift:00010
Right half after 2nd shift:10001

Key 2 = 10010010

Plain text:01110010
Permuted plain text:10101001
Expanded right plain text:11000011
Ex-or with key 1:01100111
S box output:1011
Permuted S box value:0111
Ex or with left half:1101

Swap

Expanded right plain text:11101011
Ex-or with key 2:01111001
S box output:0010
Permuted S box value:0100
Ex or with left half:1101
Output:11011101
ip inverse:11010111
```

Practical: 8

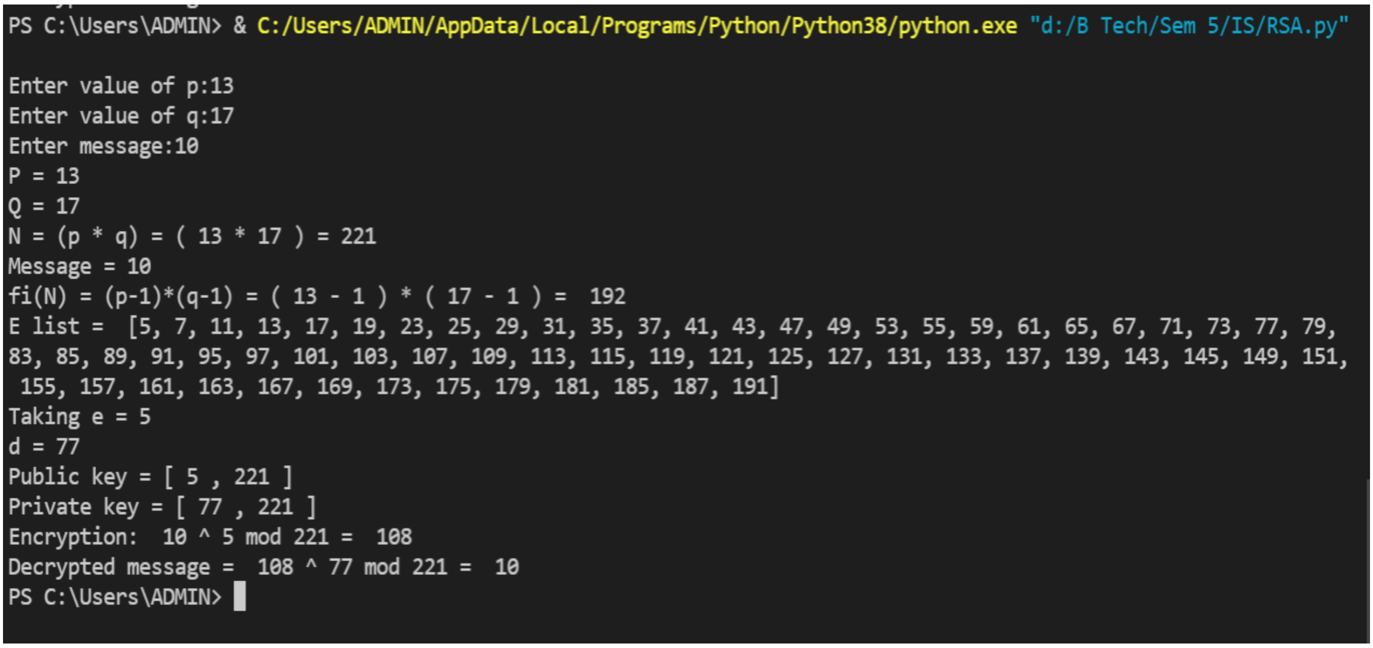
Aim: Perform a practical to implement RSA algorithm.

Code:

```
def check_prime(num):
    for i in range(2, num):
        if (num % i) == 0:
            return True

def gcd(a, b):
    if a < b:
        a, b = b, a
    if(b == 0):
        return a
    else:
        return gcd(b, a % b)

p = int(input("Enter value of p:"))
q = int(input("Enter value of q:"))
M = int(input("Enter message:"))
e_list = list()

if p < 1 or q < 1 or M < 1:
    print("sorry,Invalid number")
elif check_prime(p) == True:
    print(p, "is not a prime number")
elif check_prime(q) == True:
    print(q, "is not a prime number")
else:
    n = p*q
    print("P =", p)
    print("Q =", q)
    print("N = (p * q) = (", p, "*", q, ") =", n)
    print("Message =", M)
    fin_n = (p-1)*(q-1)
    print("fi(N) = (p-1)*(q-1) = (", p, "- 1 ) * (", q, "- 1 ) = ", fin_n)

    for i in range(2, fin_n):
        final_gcd = gcd(i, fin_n)
        if final_gcd == 1:
            e_list.append(i)
    print("E list = ", str(e_list))
    e = e_list[0]
```

```
print("Taking e =", e)
for i in range(1, 100):
    if (i*e) % fin_n == 1:
        d = i
        break

print("d =", d)
print("Public key = [", e, ",", n, "]")
print("Private key = [", d, ",", n, "]")
# Encryption
c_t = (M**e) % n
print("Encryption: ", M, "^", e, "mod", n, "= ", c_t)
# decryption
decrypt = (c_t**d) % n
print("Decrypted message = ", c_t, "^", d, "mod", n, "= ", decrypt)
```

Output:



```
PS C:\Users\ADMIN> & C:/Users/ADMIN/AppData/Local/Programs/Python/Python38/python.exe "d:/B Tech/Sem 5/IS/RSA.py"

Enter value of p:13
Enter value of q:17
Enter message:10
P = 13
Q = 17
N = (p * q) = ( 13 * 17 ) = 221
Message = 10
fi(N) = (p-1)*(q-1) = ( 13 - 1 ) * ( 17 - 1 ) =  192
E list =  [5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, 41, 43, 47, 49, 53, 55, 59, 61, 65, 67, 71, 73, 77, 79,
83, 85, 89, 91, 95, 97, 101, 103, 107, 109, 113, 115, 119, 121, 125, 127, 131, 133, 137, 139, 143, 145, 149, 151,
 155, 157, 161, 163, 167, 169, 173, 175, 179, 181, 185, 187, 191]
Taking e = 5
d = 77
Public key = [ 5 , 221 ]
Private key = [ 77 , 221 ]
Encryption:  10 ^ 5 mod 221 =  108
Decrypted message =  108 ^ 77 mod 221 =  10
PS C:\Users\ADMIN> 
```

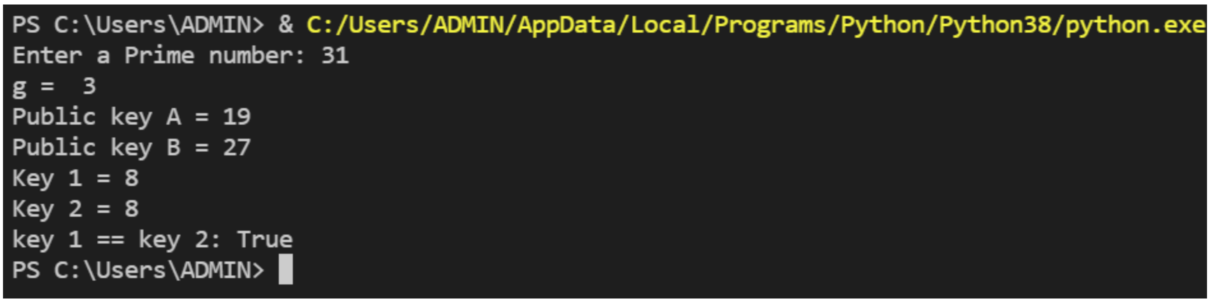
Practical: 9

Aim: Implement Diffie-Hellman Key exchange algorithm.

Code:

```
def primitive_root(q):
   c = [i for i in range(1, q)]
   for i in range(2, q):
      actual_set = [((i**power) % q) for power in range(1, q)]
      if c == sorted(actual_set):
         return i

def check_prime(num):
   for i in range(2, num):
      if (num % i) == 0:
         return True

q = int(input("Enter a Prime number: "))
if q < 1:
   print("sorry,Invalid number")
elif check_prime(q) == True:
   print(q, "is not a prime number")
else:
   xA = 4
   yB = 3
   g = primitive_root(q)
   print("g = ", g)
   A = (g**xA) % q
   print("Public key A =", A)
   B = (g**yB) % q
   print("Public key B =", B)
   k1 = (B**xA) % q
   print("Key 1 =", k1)
   k2 = (A**yB) % q
   print("Key 2 =", k2)
   print("key 1 == key 2:", k1 == k2)
```

Output:



```
PS C:\Users\ADMIN> & C:/Users/ADMIN/AppData/Local/Programs/Python/Python38/python.exe
Enter a Prime number: 31
g =  3
Public key A = 19
Public key B = 27
Key 1 = 8
Key 2 = 8
key 1 == key 2: True
PS C:\Users\ADMIN>
```

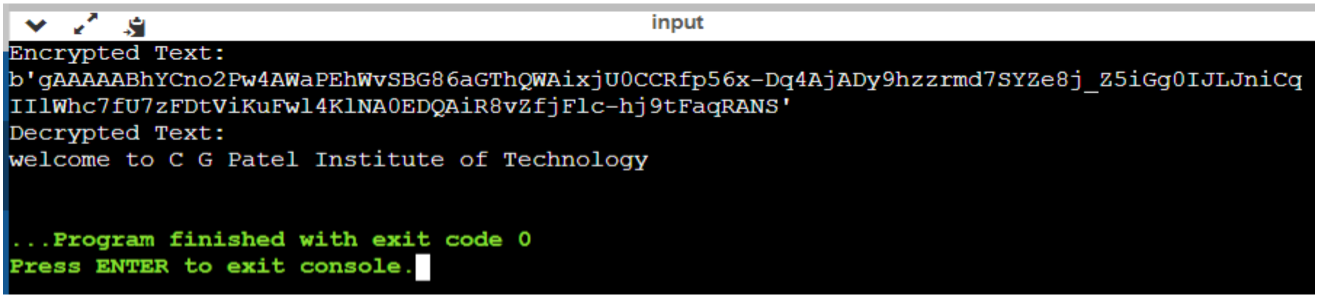
Practical: 10

Aim: Demonstration of python cryptography package to perform symmetric encryption algorithm.

Code:

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()
f = Fernet(key)
token = f.encrypt(b"welcome to C G Patel Institute of Technology")

print("Encrypted Text: ")
print(token)

print("Decrypted Text: ")
d = f.decrypt(token)

print(d.decode())
```

Output:



```
Encrypted Text:
b'gAAAAABhYCno2Pw4AWaPEhWvSBG86aGThQWAixjU0CCRfp56x-Dq4AjADy9hzzrmd7SYZe8j_Z5iGg0IJLJniCq
IIlWhc7fU7zFDtViKuFwl4KlNA0EDQAiR8vZfjFlc-hj9tFaqRANS'
Decrypted Text:
welcome to C G Patel Institute of Technology


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

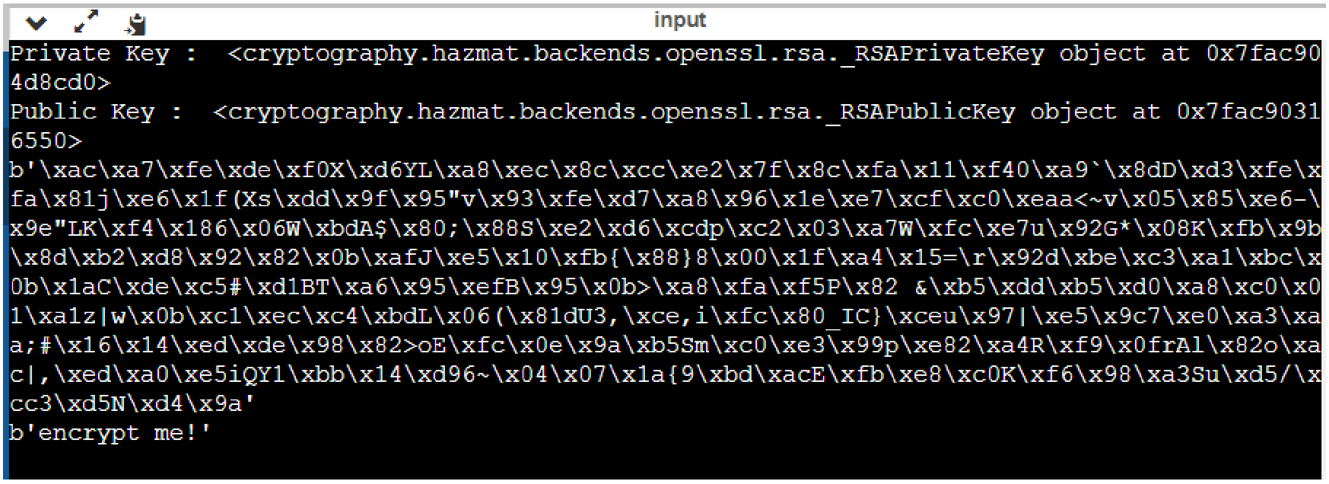
Practical: 11

Aim: Demonstration of python cryptography package to perform asymmetric encryption algorithm.

Code:

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding

private_key = rsa.generate_private_key(
    public_exponent=65537, key_size=2048, backend=default_backend())

public_key = private_key.public_key()
print("Private Key : ", private_key)
print("Public Key : ", public_key)

pem_pr = private_key.private_bytes(encoding=serialization.Encoding.PEM,
format=serialization.PrivateFormat.PKCS8,encryption_algorithm=serialization.NoEncryptio
n()
)

with open('private_key.pem', 'wb') as f:
    f.write(pem_pr)

pem_pu = public_key.public_bytes(encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo)

with open('public_key.pem', 'wb') as f:
    f.write(pem_pu)

with open("private_key.pem", "rb") as key_file:
    private_key = serialization.load_pem_private_key(
        key_file.read(),password=None,backend=default_backend())

with open("public_key.pem", "rb") as key_file:
    public_key = serialization.load_pem_public_key(
        key_file.read(),backend=default_backend())

message = b'encrypt me!'

encrypted = public_key.encrypt(
    message,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

print(encrypted)
original_message = private_key.decrypt(
    encrypted,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)

print(original_message)
```

Output:



```
Private Key :  <cryptography.hazmat.backends.openssl.rsa._RSAPrivateKey object at 0x7fac90
4d8cd0>
Public Key :  <cryptography.hazmat.backends.openssl.rsa._RSAPublicKey object at 0x7fac9031
6550>
b'\xac\xa7\xfe\xde\xf0X\xd6YL\xa8\xec\x8c\xcc\xe2\x7f\x8c\xfa\x11\xf40\xa9`\x8dD\xd3\xfe\x
fa\x81j\xe6\x1f(Xs\xdd\x9f\x95"v\x93\xfe\xd7\xa8\x96\x1e\xe7\xcf\xc0\xeaa<~v\x05\x85\xe6-\
x9e"LK\xf4\x186\x06W\xbdA$\x80;\x88S\xe2\xd6\xcdp\xc2\x03\xa7W\xfc\xe7u\x92G*\x08K\xfb\x9b
\x8d\xb2\xd8\x92\x82\x0b\xafJ\xe5\x10\xfb{\x88}8\x00\x1f\xa4\x15=\r\x92d\xbe\xc3\xa1\xbc\x
0b\x1aC\xde\xc5#\xd1BT\xa6\x95\xefB\x95\x0b>\xa8\xfa\xf5P\x82 &\xb5\xdd\xb5\xd0\xa8\xc0\x0
1\xa1z|w\x0b\xc1\xec\xc4\xbdL\x06(\x81dU3,\xce,i\xfc\x80_IC}\xceu\x97|\xe5\x9c7\xe0\xa3\xa
a;#\x16\x14\xed\xde\x98\x82>oE\xfc\x0e\x9a\xb5Sm\xc0\xe3\x99p\xe82\xa4R\xf9\x0frAl\x82o\xa
c|,\xed\xa0\xe5iQY1\xbb\x14\xd96~\x04\x07\x1a{9\xbd\xacE\xfb\xe8\xc0K\xf6\x98\xa3Su\xd5/\x
cc3\xd5N\xd4\x9a'
b'encrypt me!'
```