DEBAGNIK KAR

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)
Deemed to be University U/S 3 of the UGC Act, 1956

KIIT School of Electronics Engineering

1804373
ETC-06

WCN Lab Report (EC-3094)

Index Page

Experiment Number	09
Date of Experiment	09/03/2021
Date of Submission	16/03/2021
Name of student	Debagnik Kar
Roll Number	1804373
Section	ETC-06

Aim of the Experiment :-

To create a web server on a local area network using a raspberry Pi running on ubuntu server.

Software Requirement :-

1. Raspberry Pi 4

2. Laptop

3. Ubuntu server 20.10 4.

4. Nginx

5. SSH client (CMD Used)

6. Sample static HTML5 and CSS files

7. Internet router

8. MicroSD card (for flashing Ubuntu server)

9. Any standard web browser

Theory :-

SSH

The **Secure Shell Protocol** (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

SSH provides a secure channel over an unsecured network by using a client–server architecture, connecting an SSH client application with an SSH server. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2. The standard TCP port for SSH is 22. SSH is generally used to access Unix-like operating systems, but it can also be used on Microsoft Windows. Windows 10 uses OpenSSH as its default SSH client and SSH server.

HTTP

Hypertext Transfer Protocol (HTTP) is an application-layer protocol for transmitting hypermedia documents, such as HTML. It was designed for communication between web browsers and web servers, but it can also be used for other purposes. HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response. HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests. Though often based on a TCP/IP layer, it can be used on any reliable transport layer, that is, a protocol that doesn't lose messages silently like UDP does. RUDP — the reliable update of UDP — is a suitable alternative.

NGINX

Nginx, stylized as **NGINX**, nginx or NginX, is a web server that can also be used as a reverse proxy, load balancer, mail proxy and HTTP cache. The software was created by Igor Sysoev and publicly released in 2004. Nginx is free and open-source software, released under the terms of the 2-clause BSD license.

Nginx can be deployed to serve dynamic HTTP content on the network using Fast CGI, SCGI handlers for scripts, WSGI application servers or Phusion Passenger modules, and it can serve as a software load balancer. Nginx uses an asynchronous event-driven approach, rather than threads, to handle requests.

UBUNTU Server

Ubuntu Server is a server operating system, developed by Canonical and open-source programmers around the world, that works with nearly any hardware or virtualization platform. It can serve up websites, file shares, and containers, as well as expand your company offerings with an incredible cloud presence.

Raspberry Pi :-

Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom. The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. The Raspberry Pi 4 has the following specifications:

- Quad Core 1.8GHz Broadcom BCM2837 64bit CPU
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- Full size HDMI Display (for head start)

- Micro SD port for loading your operating system and storing data

- Micro USB-C power source up to 3A

LAN :-

A **local area network** (**LAN**) is a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school. LAN has the following advantages:

- Central back-up can take place automatically at regular intervals. A user will usually be able to retrieve work that has been deleted by mistake.

- Data can be shared across the network. For example, this would allow several people to work on the same project. The setting up of hierarchical system passwords to allow different users different access is the key to controlling access to most company databases.

- Costly resources such as printers can be shared by all of the computers. This means that better quality printing is available to everyone because one or two expensive, high specification printers can be bought instead of several cheaper, lower specification models. A line (bus) topology is the cheapest in terms of cabling costs.

Output :-

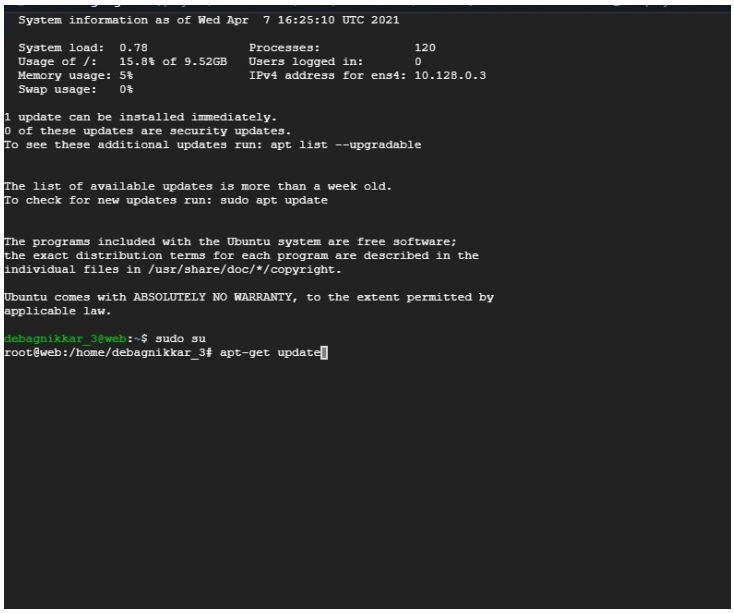


Fig 9.1 getting remote ssh connection to raspberry pi and updating it.

```
root@web:/home/debagnikkar_3# ifconfig
ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1460
        inet 10.128.0.3  netmask 255.255.255.255  broadcast 0.0.0.0
        inet6 fe80::4001:aff:fe80:3  prefixlen 64  scopeid 0x20<link>
        ether 42:01:0a:80:00:03  txqueuelen 1000  (Ethernet)
        RX packets 2209  bytes 28120706 (28.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1873  bytes 242957 (242.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 152  bytes 14940 (14.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 152  bytes 14940 (14.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@web:/home/debagnikkar_3# apt-get install nginx
```

Fig 9.2: getting IP Address and installing nginx

Fig 9.3: nginx install successful html page.

```
root@web:/home/debagnikkar_3# cd /var/www/html
root@web:/var/www/html# ls
index.nginx-debian.html
root@web:/var/www/html# rm *
root@web:/var/www/html# git clone https://github.com/abhijeetdey361/survey-form.gi
Cloning into 'survey-form'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), 2.07 KiB | 705.00 KiB/s, done.
root@web:/var/www/html# mv survey-form/* .
root@web:/var/www/html# ls
Form.html  formstyle.css  survey-form
root@web:/var/www/html# rm survey-form/
rm: cannot remove 'survey-form/': Is a directory
root@web:/var/www/html# rm -f survey-form/
rm: cannot remove 'survey-form/': Is a directory
root@web:/var/www/html# rm -fr survey-form/
root@web:/var/www/html# ls
Form.html  formstyle.css
root@web:/var/www/html# mv ./Form.html ./index.html
root@web:/var/www/html# ls
formstyle.css  index.html
root@web:/var/www/html#
```

Fig 9.4: removing all the previous data in the nginx directory and cloning the sample webpage.

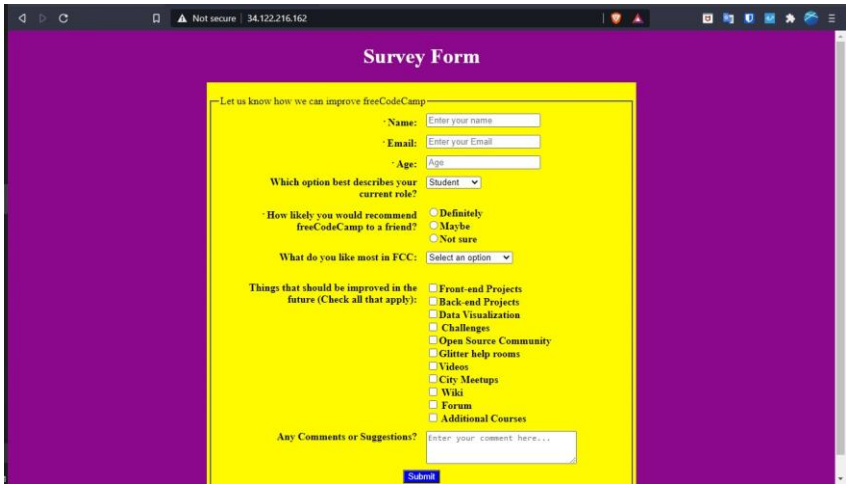


Fig 9.5: website deployed successfully on a local network using a RPi Server

Conclusion :-

In this experiment we have successfully created a web server on a local area network using a raspberry Pi running on ubuntu server.

Experiment Number	10
Date of Experiment	09/03/2021
Date of Submission	16/03/2021
Name of student	Debagnik Kar
Roll Number	1804373
Section	ETC-06

Aim of the Experiment :-

To create a real time RSA-encrypted Chatting application using Python and TCP/UDP protocols

Software Requirement :-

- Python
- Cloud Platform (Google Cloud Platform)
- Ubuntu server 20.04
- OpenSSH console (CMD Used)
- Any standard web browser

Theory :-

Transmission Control Protocol:

Transmission control protocol or TCP is a connection oriented, reliable protocol. It uses a combination of GBN and SR protocols to provide reliability, it clearly defines connection establishment, data transfer, a connection teardown phases to provide a connection-oriented service. TCP uses checksum for error detection and retransmit lost or corrupted packets by the use of selective acknowledgements and timers, this results in the lengthening of latency[10]. TCP also provides process to process communication using port numbers like *Secure.ly*.

TCP was first described by Vint Cerf, Bob Kahn and Carl Sunshine describing the internetworking protocol for sharing resources using packet tracing among network nodes in their publication

Specification of Internet Control Program in Dec 1974 [11], they were awarded the prestigious Turing award for their foundational work on TCP/IP in 2004 [12].

2.3.1 A TCP Connection: TCP is connection oriented; it provides a communication service at an intermediate level between an application program and the Internet Protocol. It provides host-to-host connectivity at the transport layer in the OSI Model. TCP protocol operation maybe divided into three parts:

2.3.1.1 Connection establishment: The connection establishment in TCP is called the *Three-way handshake.* Before a client attempts to connect to a server, it must bind and listen at a port (passive open). A client may establish a connection by starting an active open using the 3-way handshake:

1. **SYN:** the active open is performed by the client-side system sending a SYN to the server-side system. The client send a random sequence of number 'N'
2. **SYN-ACK:** In reply, the server sends a SYN-ACK, which is one more than SYN, 'N+1'. And the server chooses another random number 'M' to the client
3. **ACK:** Finally, the client, on receiving SYS-ACK, responds with ACK which is set to one more than the receiving sequence 'M+1'.

Step 1 and 2 establishes and acknowledges the sequence number for client to server connection and step 2 and 3 establishes and acknowledges the number of server to client connection, therefore ensures a full-duplex communication is established. This sequence is visualized in fig 4 [13]

2.3.1.2 Data Transfer: After the connection is established the data transfer is can take place in both the direction, the acknowledgment is sent along with the data. TCP has many features in data transmission, such as

- **Ordered Data transfer:** The destination host rearranges segments according to a sequence number[14]
- **Retransmission of lost data:** Any sent data not acknowledged back to the sender in a specific duration of time is resent again.[14]
- **Flow Control:** If the buffer of the receiving end fills it ask the sender to stop to get the received data processed and free the buffer space.[14]
- **Error free data transfer:** Uses checksums to detect errors or corrupted data[15]

- **Congestion control:** uses timestamps and Karn's algorithm to prevent congestion.

2.3.1.3 Connection Termination: The connection termination phase uses a four-way handshake, with both the sides ending the connection independently. Any of the side sends a FIN packet, to which the other side acknowledges with a ACK. After the side who sent the first FIN has responded with the final ACK, It waits for a time out before finally ending the connection.[13]. A sample TCP connection is visualized in fig 2.4. and the TCP packet is visualized in figure 2.5

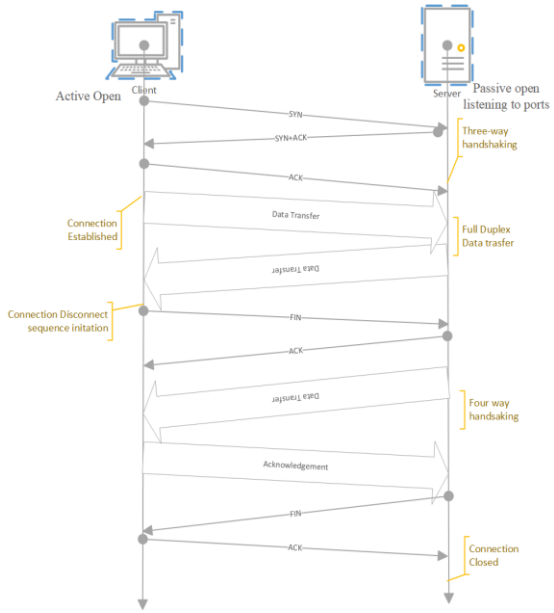


Fig 2.4: Timing diagram of TCP Connection

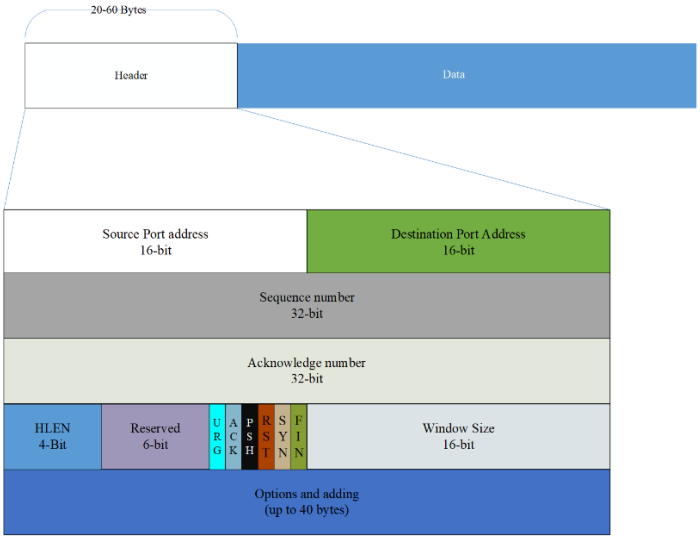


Fig 2.5: a TCP segment

Riverman-Shamir-Adleman Algorithm:

RSA is an asymmetric, non-secret cryptographic algorithm widely used to encrypt or decrypt data. It basically uses the difficulty of factoring a very large integers. It is generally used to encrypt data and sign digital documents. RSA works by generating two very long (usually 512-bit but 1024-bit is considered secure) integers and publishing one number (or key) while keeping the other a secret. The algorithm was invented in 1977 in MIT by a group of three researchers named Ron Rivest, Adi Shamir and Len Adleman. The algorithm is named after the researchers initials R(Riverman)-S(Shamir)-A(Adleman) [3]. The highly secure algorithm was actually based on the ideas and techniques of Clifford Cocks who first stated in 1973 that a secure cryptographic technique should be based on two factor one which encrypts the data which is available to the public and another that is kept a secret and will decrypt the contents of cypher text [4]. RSA algorithm replaced the previous Diffie-Hellman algorithm which was a symmetric cryptography technique it showed that encryption should be a one-way trapdoor, if the data is once encrypted which should be easy to compute by the sender using a key it will be very difficult to decrypt without that particular key [5]. Although Diffie-Hellman algorithm was highly secure but it posed two problems for the users, the users of the algorithm need to share the private key to encrypt or decrypt the data this was achieved by either physically meeting the person before sending data, or sending the keys via a network. Both the ways were pretty inefficient. The other problem was the sender or the receiver have to keep track of all the keys of different receiver or sender respectively which was a headache for users who communicate with large groups like a corporate. RSA addressed both the problems faced by the Diffie-Hellman algorithm by generating a set of private key and public key so that the receiving user could first send the public key to all the senders to encrypt the data keeping only the private key as a secret to decrypt the cypher text [6]. Thus, allowing the receiving user of RSA to keep only a keys secret and publishing the public key to anyone who wishes to send data to the receiving user.

Output:-

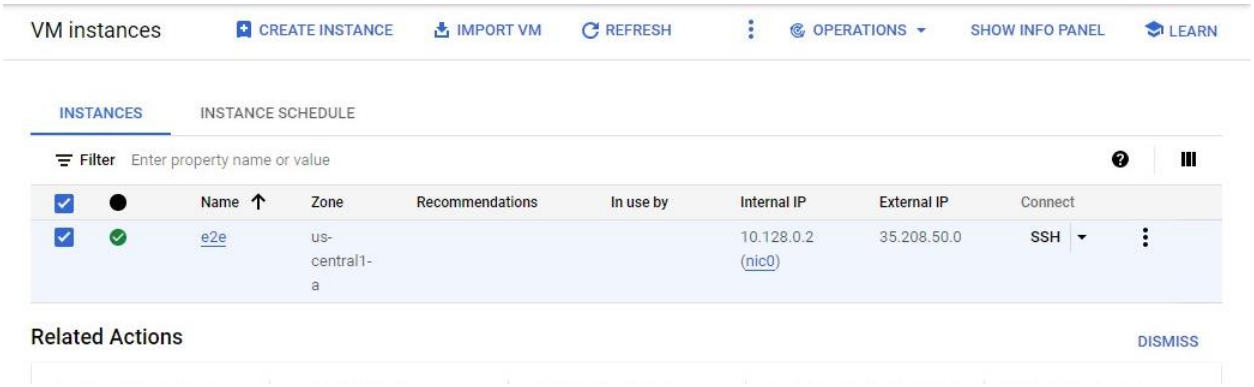


Fig 10.3: virtual compute instance running on GCP

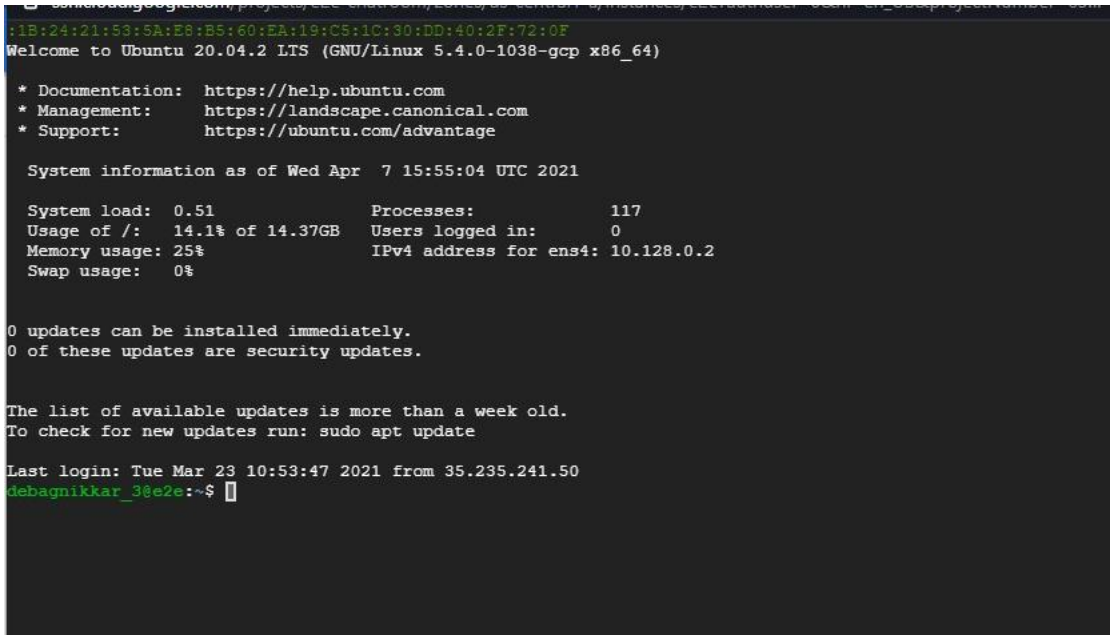


Fig 10.4: Getting remote connection using ssh

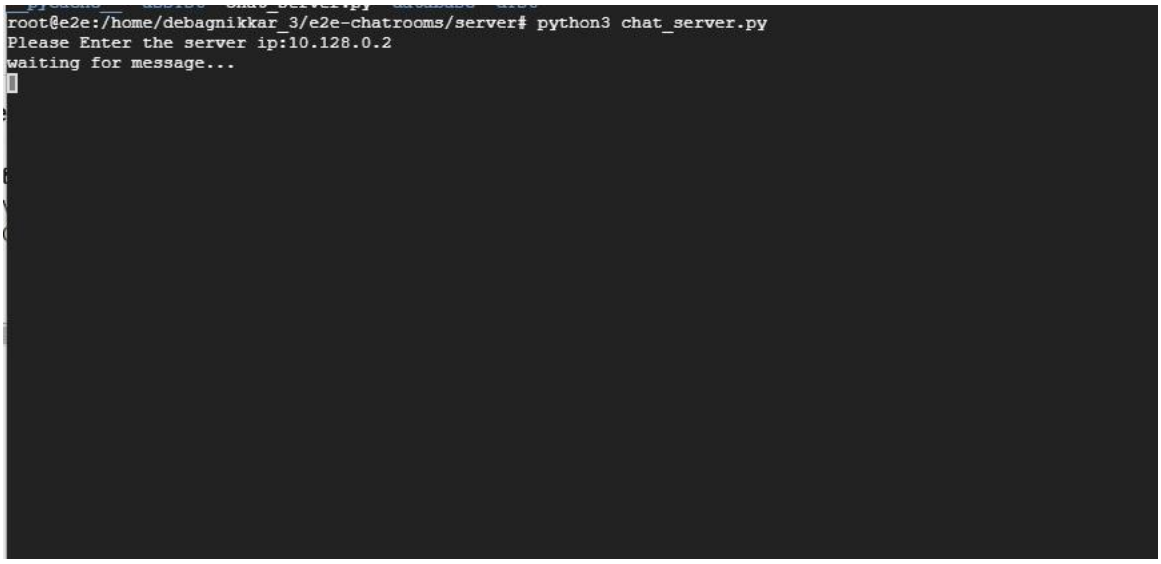


Fig 10.5: Setting up Server Service

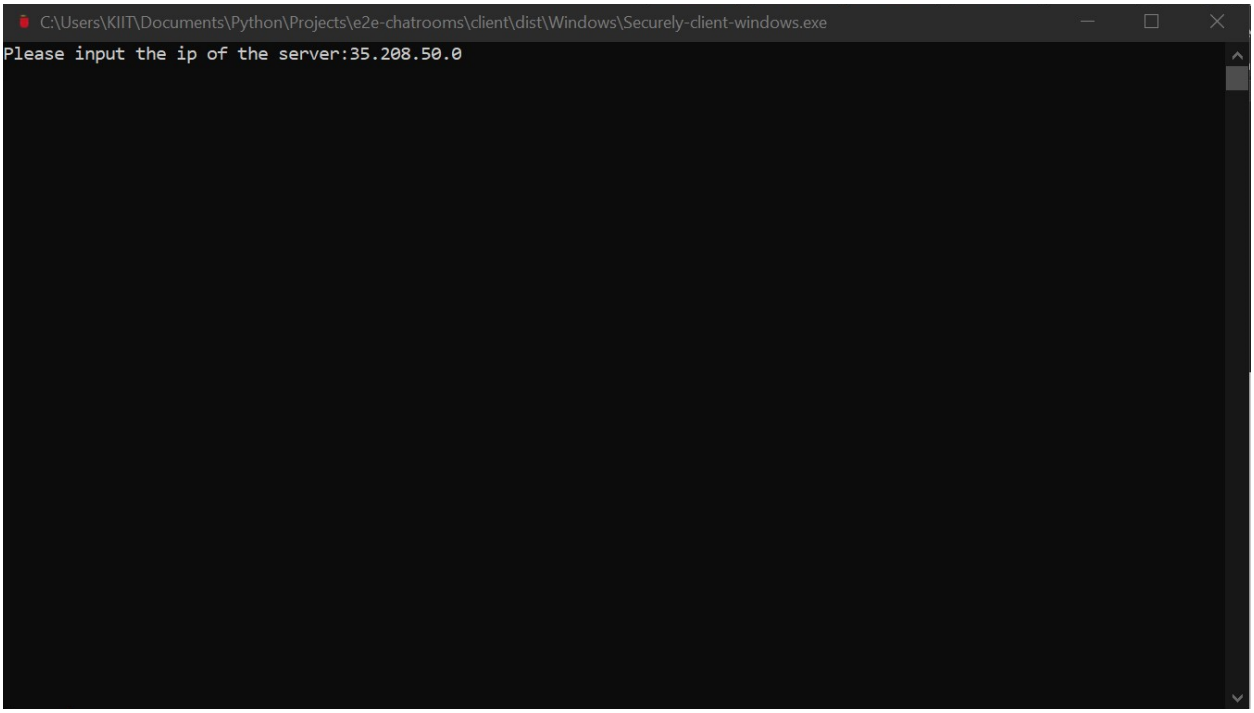
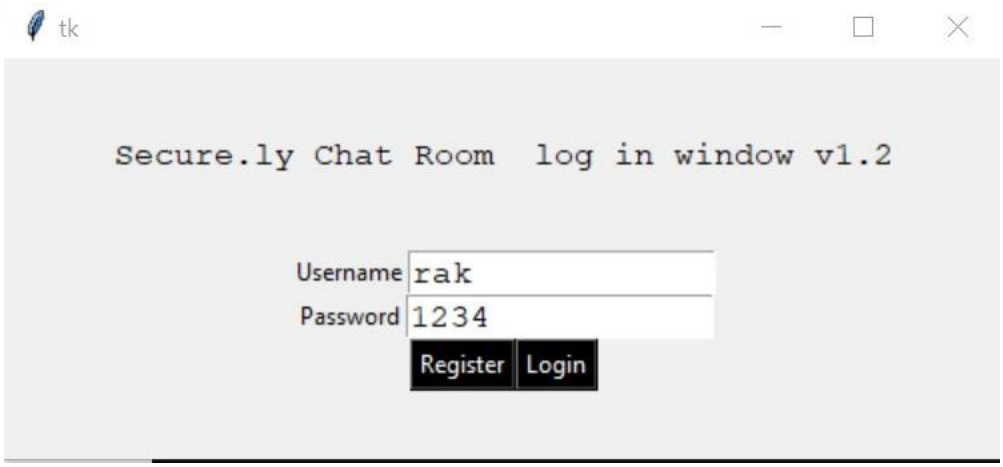


Fig 10.6: Entering IP of the server on the client-side



10.7: Log in/Sign up Window prompt

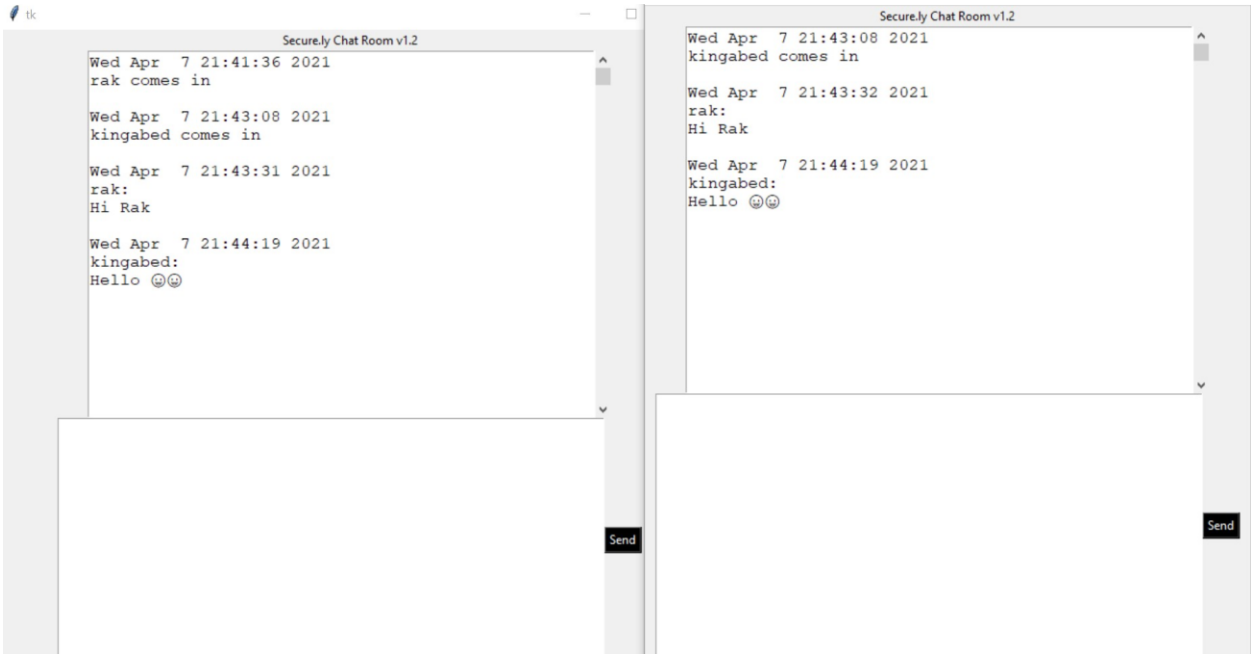


Fig 10.8: two chat threads running on the same IP

Fig 10.9: While the chat happens, the server receives garbage value as cypher text.

Conclusion :-

We successfully created a chatting application using python3, TCP/UDP protocols and set encryption to RSA-1024 bits