

SECURE FILE TRANSFER APPLICATION

GROUP 2

Deepak Sahu - 20535011

Pranav Mehta - 20535020

Sarthak Sharma - 20535025

Shivam - 20535026

Shweta - 20535027

Taj Mohammad - 20535030

Karthi Kannan M - 20535037

| <u>S.No</u> | <u>Name</u> | <u>Contribution</u> | <u>Page No</u> |
|--------------------|--------------------|---|-----------------------|
| 1. | Sarthak Sharma | Connection establishment using socket programming for Server. | 6,7,8,10-15 |
| 2. | Karthi Kannan M | Connection establishment using socket programming for Client. | 5,8,16-19 |
| 3. | Shweta | Sending file in form of Strings. | 5,18,19 |
| 4. | Pranav Mehta | Diffie Hellman. | 5,6 |
| 5. | Taj Mohammad | Encryption | 8,19,20 |
| 6. | Deepak Sahu | Decryption | 8,19,20 |
| 7. | Shivam | Wireshark | 8,24,26 |

CONTENTS

1. Problem Statement and Description
2. Discussion, implementation and Analysis
3. Source Code
4. Output
5. References

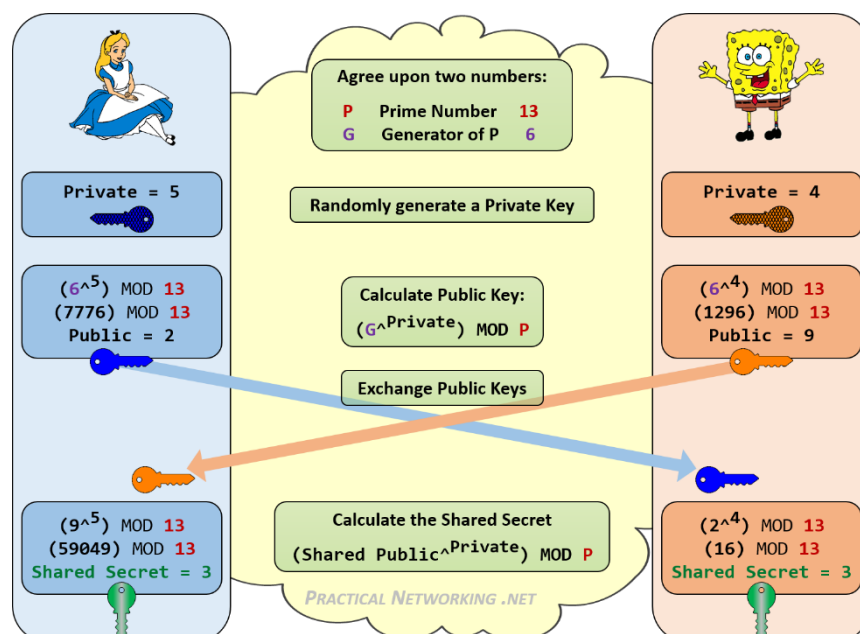
Problem Statement and Description

1. An organization needs an application which can help their employees to transfer files between them securely on the same network.
2. Develop an application using Socket programming to send files between two machines.
3. Secure the data transfer using a strong Encryption Algorithm.
4. To Capture these packets using Wireshark and to show that data transfer is secure.

Discussion, Implementation and Analysis

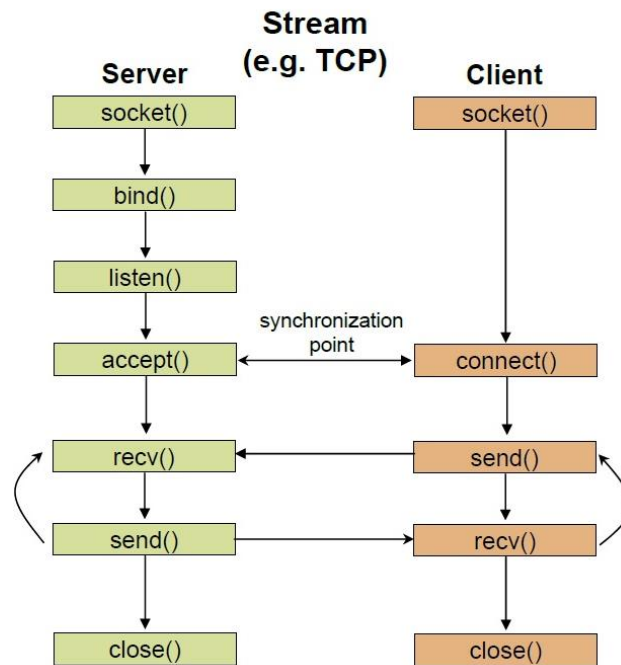
1. Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while the client reaches out to the server.

2. Diffie hellman is a key exchange algorithm which enables 2 entities over a network to securely produce and share keys. The keys finally generated at the end of the computation are same on both the systems and hence the key so generated can be used for any symmetric encryption algorithm. The Diffie hellman needs to calculate a prime number followed by a primitive root (generator) of that prime number, this could be done at one of the server, then following some mathematical computations the resultant private key is obtained as explained in the figure below:



3. **AES** stands for **Advanced Encryption Standard** and is a majorly used symmetric encryption algorithm. It is mainly used for encryption and protection of electronic data. It was used as the replacement of DES(Data encryption standard) as it is much faster and better than DES. AES consists of three block ciphers and these ciphers are used to provide encryption of data.
4. Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible. You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable. In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, that has changed. Wireshark is available for free, is open source, and is one of the best packet analyzers available today

5. Implementation.



Stages for Server

a) **Socket creation:**

```
int sockfd = socket(domain, type, protocol)
```

sockfd: socket descriptor, an integer (like a file-handle)

domain: integer, communication domain e.g., AF_INET (IPv4 protocol) , AF_INET6 (IPv6 protocol)

type: communication type

SOCK_STREAM: TCP(reliable, connection oriented)

SOCK_DGRAM: UDP(unreliable, connectionless)

protocol: Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

b) **Setsockopt:**

```
int setsockopt(int sockfd, int level, int optname,
               const void *optval, socklen_t optlen);
```

This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as: “address already in use”.

c) **Bind:**

```
int bind(int sockfd, const struct sockaddr *addr,
```

```
socklen_t addrlen);
```

After creation of the socket, bind function binds the socket to the address and port number specified in addr(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR_ANY to specify the IP address.

d) **Listen:**

```
int listen(int sockfd, int backlog);
```

It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

e) **Accept:**

```
int new_socket= accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.

Stages for Client

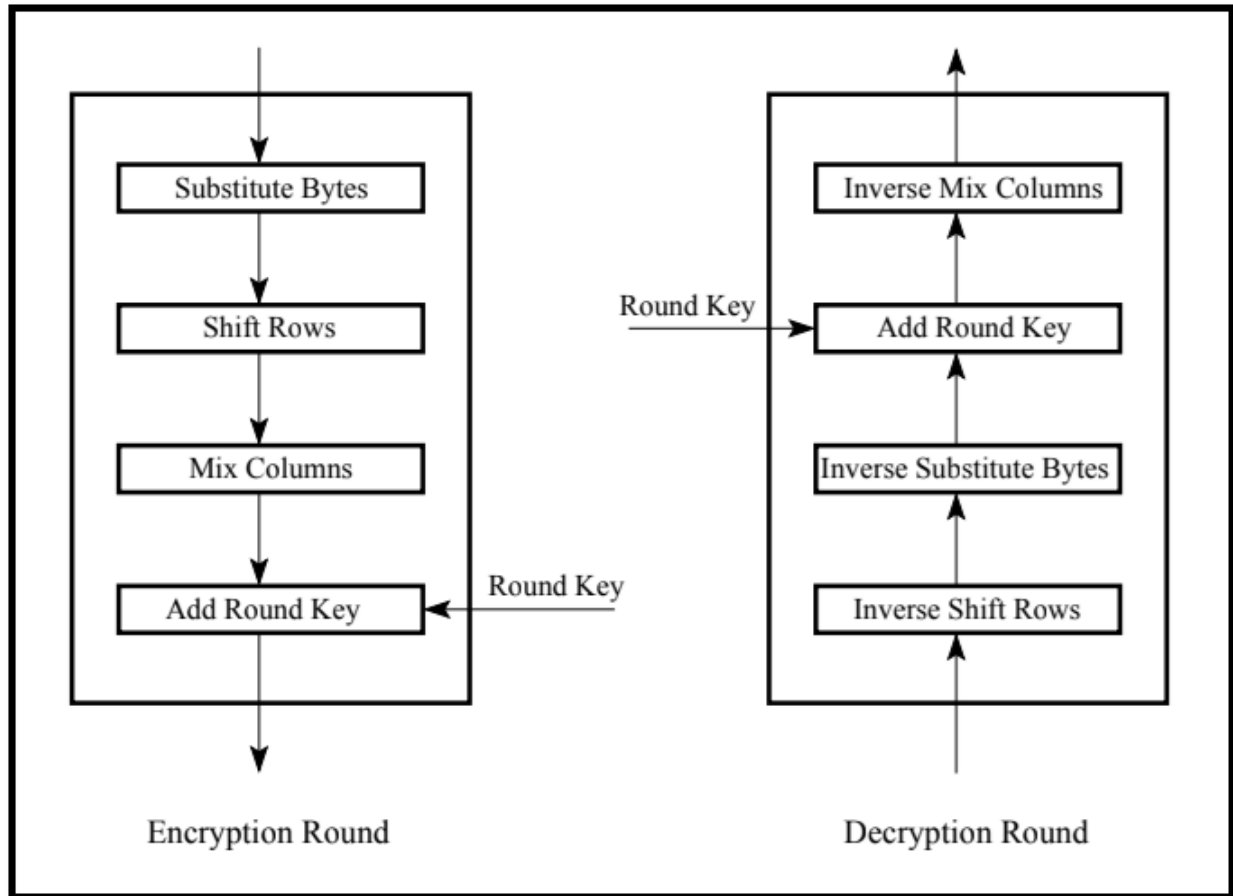
a) **Socket connection:** Exactly same as that of server's socket creation

b) **Connect:**

```
int connect(int sockfd, const struct sockaddr *addr,  
            socklen_t addrlen);
```

The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server's address and port is specified in addr.

AES Steps



6. Wireshark (3.2.3):

- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and many other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.

7. Analysis of Wireshark has been attached in the output section.

Source Code**Server End:**

```
#include <stdio.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <unistd.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <string.h>

#include <stdint.h>

#include <openssl/aes.h>

#define CBC 1

#define PORT 8080


/* AES key for Encryption and Decryption */

const static unsigned char

aes_key[]={0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0x

EE,0xFF};

void print_data(const char *tittle, const void* data, int len)

{

    printf("%s : ",tittle);
```

```

const unsigned char * p = (const unsigned char*)data;

int i = 0;

printf("Decrypted Data : \n");

for (; i<len; ++i)

    printf("%02X ", *p++);

printf("\n");
}

int main()
{
    struct sockaddr_in address;

    int option = 1,server_fd, sock;

    int address_length = sizeof(address);

    char msg[1024] = {0};

    char *msg_from_server = "Hello from server";

    // Creating socket file descriptor

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)

    {

        printf("Server Creation Failed\n");

        return 0;

    }

```

```

address.sin_family = AF_INET;

address.sin_addr.s_addr = INADDR_ANY;

address.sin_port = htons( PORT );


// Forcefully attaching socket to the port 8080
if (bind(server_fd, (struct sockaddr *)&address, sizeof(address))<0)
{
    printf("Binding Failed\n");
    return 0;
}


while(1)
{
    int in;

    printf("Press 1 for Exit 2 for Sending File : ");

    scanf("%d",&in);

    if(in==1)
        break;

    printf("Going in listening mode...\n");


    //Server now in listening with maximum backlog requests upto 5
    if (listen(server_fd, 3) < 0)
    {

```

```

        printf("Listening Failed\n");

        return 0;
    }

```

```

        if ((sock = accept(server_fd, (struct sockaddr *)&address,
(socklen_t*)&address_length))<0)

```

```

        {
            printf("Connection can't be accepted\n");

            return 0;
        }

```

```

        int value = read( sock , msg, 1024);

```

```

        printf("Recieving File %s\n",msg);

```

```

        int n;

```

```

        FILE *fp;

```

```

        unsigned char enc_out[100];

```

```

        fp=fopen("recv.txt","w");

```

```

        while(1)

```

```

        {

```

```

        unsigned char iv[AES_BLOCK_SIZE];

memset(iv, 0x00, AES_BLOCK_SIZE);

AES_KEY dec_key;

/* Buffers for Encryption and Decryption */

```

```

        char a[4];

        if(recv(sock,a,4,0)<=0)

        break;

```

```

        int b;

        b=atoi(a);

        unsigned char enc_out[b];

        unsigned char dec_out[b];

```

```

        if(recv(sock,enc_out,100,0)<=0)

        break;

```

```

        memset(iv, 0x00, AES_BLOCK_SIZE); // don't forget to set iv vector

```

again, else you can't decrypt data properly

```

        AES_set_decrypt_key(aes_key, sizeof(aes_key)*8, &dec_key); // Size of

```

key is in bits

```

        AES_cbc_encrypt(enc_out, dec_out, b, &dec_key, iv, AES_DECRYPT);

```

```

int i=0;

char temp[b];

for(i=0;i<b;i++)
{

    if(((char)dec_out[i]>='a'&&(char)dec_out[i]<='z')||((char)dec_out[i]>='A'&&(char)dec_o
ut[i]<='Z')||((char)dec_out[i]>='0'|(char)dec_out[i]<='9'))

        temp[i]=(char)dec_out[i];

    else

        break;

}

for(i=0;i<b;i++)
{

    printf("%c",temp[i]);

}

fprintf(fp,"%s",temp);

bzero(enc_out,b);

}

fclose(fp);

}

return 0;

}

```

Client End:

```
#include <stdio.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <unistd.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <string.h>

#include <stdint.h>


#include <openssl/aes.h>

#define CBC 1

#define PORT 8080


/* AES key for Encryption and Decryption */

const static unsigned char

aes_key[]={0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xAA,0xBB,0xCC,0xDD,0x

EE,0xFF};


void print_data(const char *tittle, const void* data, int len)

{
```



```

printf("%s : ",tittle);

const unsigned char * p = (const unsigned char*)data;

int i = 0;

printf("Ecryptd Data : \n");

for (; i<len; ++i)

    printf("%02X ", *p++);


printf("\n");
}

char * encrypt(char*line)
{
    //cout<<"hey";

    for(int i=0;line[i]!='\n';i++)
    {
        line[i]=(line[i]+14)%26;
    }

    printf("In e %s\n",line);

    return line;
}

int main()
{
    int sock1 = 0, value;

    struct sockaddr_in serv_addr;

```

```
char *msg_from_client = "file.txt";

char msg[1024] = {0};

if ((sock1 = socket(AF_INET, SOCK_STREAM, 0)) < 0)

{

    printf("\n Socket creation error \n");

    return -1;

}


serv_addr.sin_family = AF_INET;

serv_addr.sin_port = htons(PORT);


// Convert IPv4 and IPv6 addresses from text to binary form

if(inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr)<=0)

{

    printf("Invalid Address\n");

    return 0;

}


if (connect(sock1, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)

{

    printf("Connection Failed\n");

    return 0;

}
```

```
send(sock1 , msg_from_client , strlen(msg_from_client) , 0 );
```

```
FILE *fp;
```

```
fp=fopen("file.txt", "r");
```

```
unsigned char line[100];
```

```
while(fgets(line,100,fp)!=NULL)
```

```
{
```

```
char a[2];
```

```
sprintf(a, "%lu", strlen(line));
```

```
if(send(sock1,a,sizeof(a),0)<0)
```

```
{
```

```
    printf("Error in sending...\n");
```

```
    return 0;
```

```
}
```

```
unsigned char iv[AES_BLOCK_SIZE];
```

```
memset(iv, 0x00, AES_BLOCK_SIZE);
```

```
/* Buffers for Encryption and Decryption */
```

```
unsigned char aes_input[100];
```

```
strcpy(aes_input,line);
```

```

unsigned char enc_out[sizeof(aes_input)];

unsigned char dec_out[sizeof(aes_input)];


/* AES-128 bit CBC Encryption */

AES_KEY enc_key, dec_key;

AES_set_encrypt_key(aes_key, sizeof(aes_key)*8, &enc_key);

AES_cbc_encrypt(aes_input, enc_out, sizeof(aes_input), &enc_key, iv,
AES_ENCRYPT);


print_data("\n Encrypted",enc_out, sizeof(enc_out));


if(send(sock1,enc_out,sizeof(enc_out),0)<0)
{
    printf("Error in sending...\n");
    return 0;
}

bzero(line,100);


}

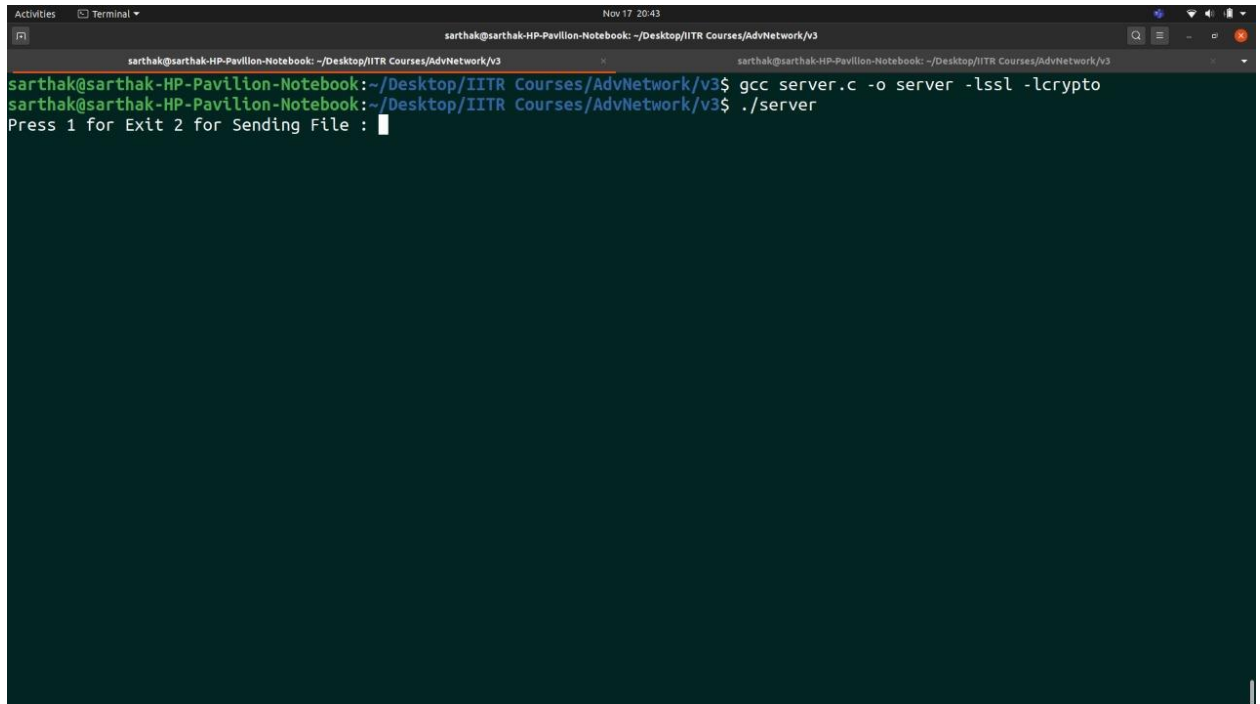
fclose(fp);


return 0; }

```

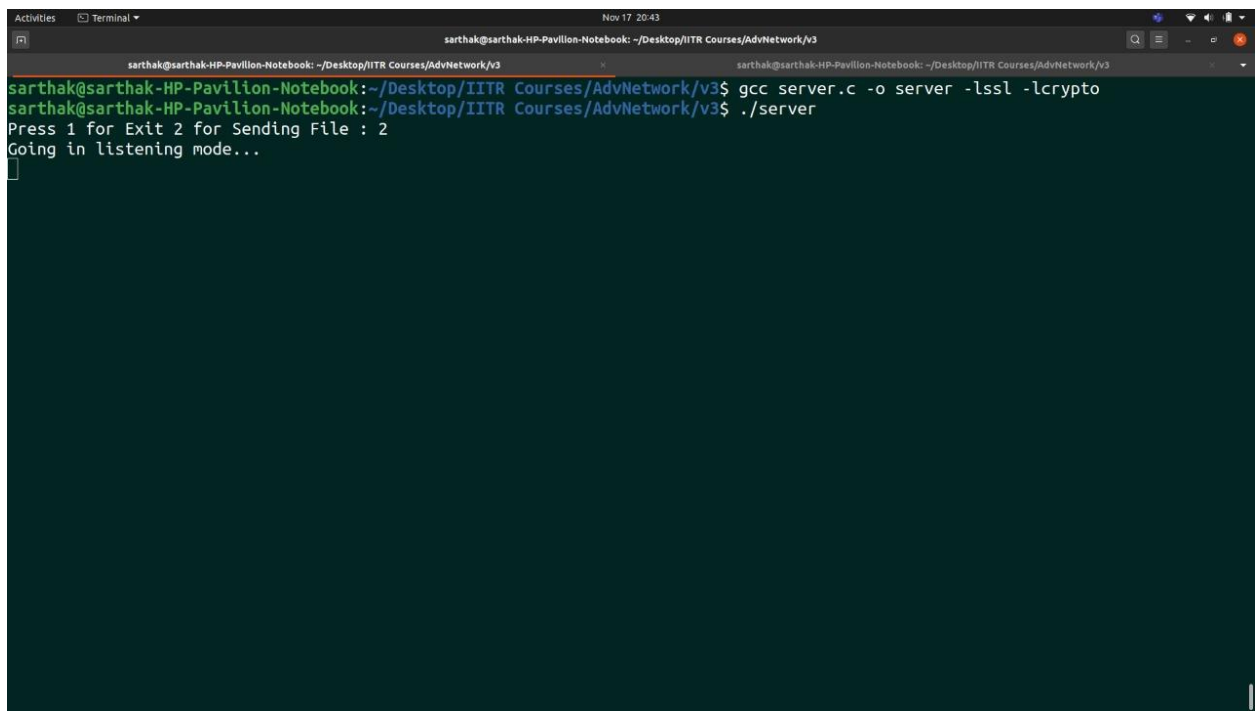
Output

1. Running Server



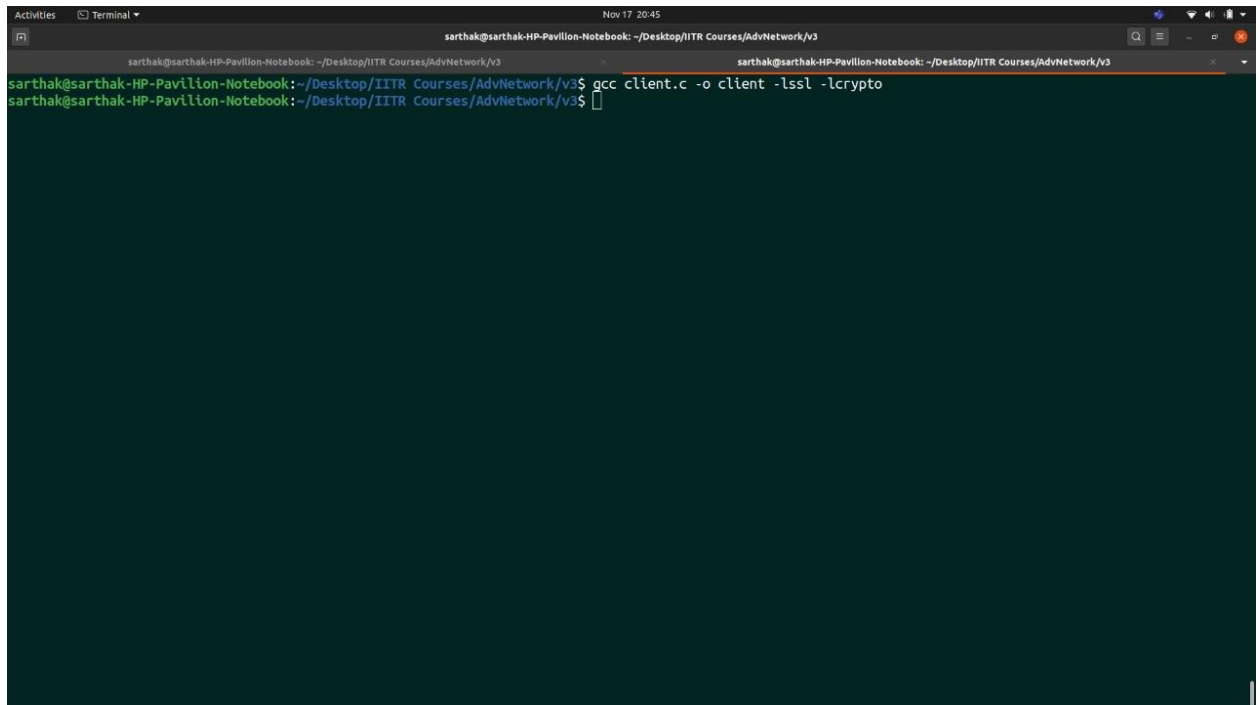
```
Activities Terminal Nov 17 20:43
sarthak@sarthak-HP-Pavilion-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3
sarthak@sarthak-HP-Pavilion-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3$ gcc server.c -o server -lssl -lcrypto
sarthak@sarthak-HP-Pavilion-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3$ ./server
Press 1 for Exit 2 for Sending File : █
```

2. Server Listening Mode



```
Activities Terminal Nov 17 20:43
sarthak@sarthak-HP-Pavilion-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3
sarthak@sarthak-HP-Pavilion-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3$ gcc server.c -o server -lssl -lcrypto
sarthak@sarthak-HP-Pavilion-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3$ ./server
Press 1 for Exit 2 for Sending File : 2
Going in listening mode...
█
```

3. Running Client

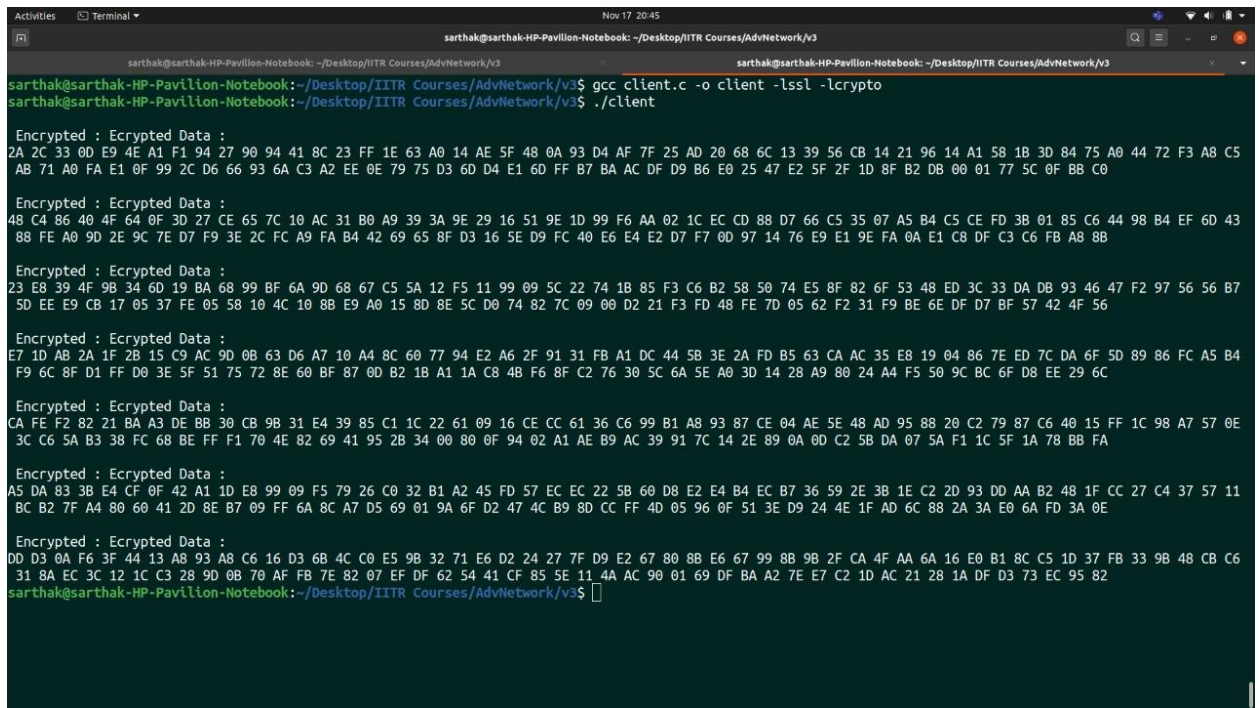


```

sarthak@sarthak-HP-Pavillon-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3
sarthak@sarthak-HP-Pavillon-Notebook:~/Desktop/IITR Courses/AdvNetwork/v3$ gcc client.c -o client -lssl -lcrypto
sarthak@sarthak-HP-Pavillon-Notebook:~/Desktop/IITR Courses/AdvNetwork/v3$

```

4. Encrypted Client Data



```

sarthak@sarthak-HP-Pavillon-Notebook:~/Desktop/IITR Courses/AdvNetwork/v3$ gcc client.c -o client -lssl -lcrypto
sarthak@sarthak-HP-Pavillon-Notebook:~/Desktop/IITR Courses/AdvNetwork/v3$ ./client

Encrypted : Ecrypted Data :
2A 2C 33 0D E9 4E A1 F1 94 27 90 94 41 8C 23 FF 1E 63 A0 14 AE 5F 48 0A 93 D4 AF 7F 25 AD 20 68 6C 13 39 56 CB 14 21 96 14 A1 58 1B 3D 84 75 A0 44 72 F3 A8 C5
AB 71 A0 FA E1 0F 99 2C D6 66 93 6A C3 A2 EE 0E 79 75 D3 6D D4 E1 6D FF B7 BA AC DF D9 B6 E0 25 47 E2 5F 2F 1D 8F B2 DB 00 01 77 5C 0F BB C0

Encrypted : Ecrypted Data :
48 C4 86 40 4F 64 0F 3D 27 CE 65 7C 10 AC 31 B0 A9 39 3A 9E 29 16 51 9E 1D 99 F6 AA 02 1C EC CD 88 D7 66 C5 35 07 A5 B4 C5 CE FD 3B 01 B5 C6 44 98 B4 EF 6D 43
88 FE A0 9D 2E 9C 7E D7 F9 3E 2C FC A9 FA B4 42 69 65 8F D3 16 5E D9 FC 40 E6 E4 E2 D7 F7 0D 97 14 76 E9 E1 9E FA 0A E1 C8 DF C3 C6 FB A8 8B

Encrypted : Ecrypted Data :
23 E8 39 4F 9B 34 6D 19 BA 68 99 BF 6A 9D 68 67 C5 SA 12 F5 11 99 09 5C 22 74 1B 85 F3 C6 B2 58 50 74 E5 8F 82 6F 53 48 ED 3C 33 DA DB 93 46 47 F2 97 56 56 B7
5D EE E9 CB 17 05 37 FE 05 58 10 4C 10 8B E9 A0 15 8D 8E 5C D0 74 82 7C 09 00 D2 21 F3 FD 48 FE 7D 05 62 F2 31 F9 BE 6E DF D7 BF 57 42 4F 56

Encrypted : Ecrypted Data :
E7 1D 0A 2A 1F 2B 15 C9 AC 9D 0B 63 D6 A7 10 A4 8C 60 77 94 E2 A6 2F 91 31 FB A1 DC 44 5B 3E 2A FD B5 63 CA AC 35 E8 19 04 86 7E ED 7C DA 6F 5D 89 86 FC A5 B4
F9 6C 8F D1 FF D0 3E 5F 51 75 72 8E 60 BF 87 0D B2 1B A1 1A C8 4B F6 8F C2 76 30 5C 6A 5E A0 3D 14 28 A9 80 24 A4 F5 50 9C BC 6F D8 EE 29 6C

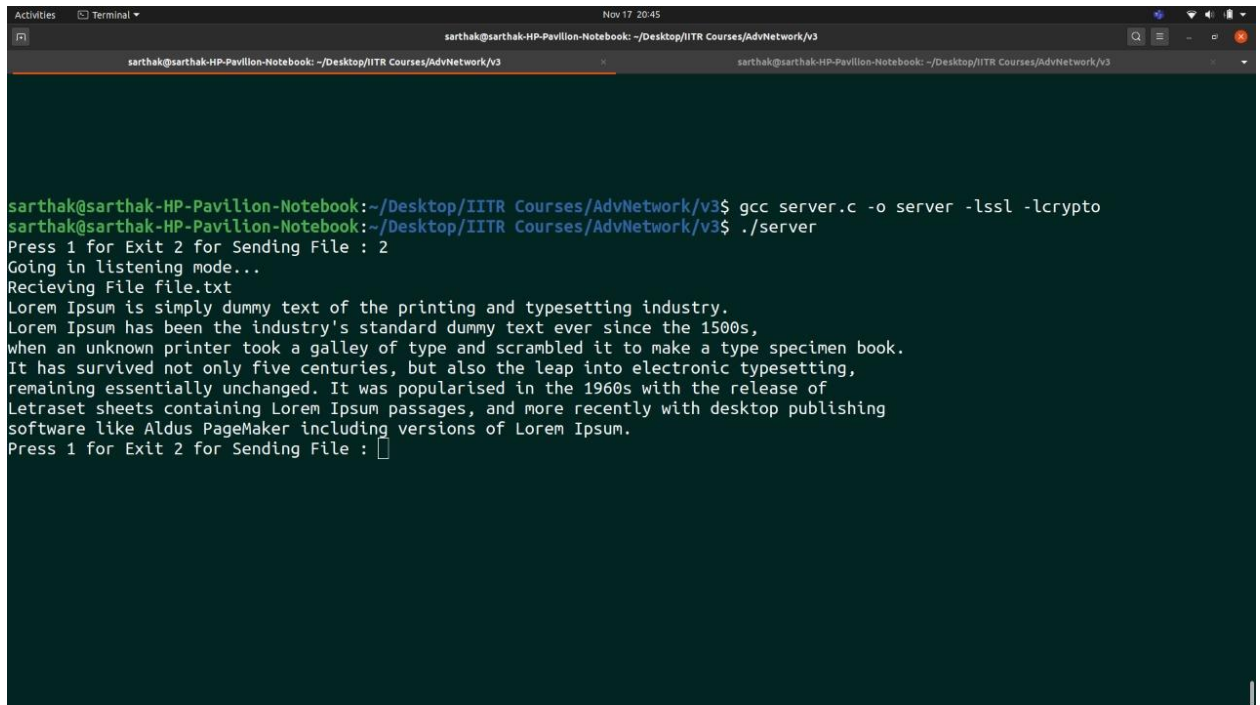
Encrypted : Ecrypted Data :
CA FE F2 82 21 BA A3 DE BB 30 CB 9B 31 E4 39 85 C1 1C 22 61 09 16 CE CC 61 36 C6 99 B1 A8 93 87 CE 04 AE 5E 48 AD 95 88 20 C2 79 87 C6 40 15 FF 1C 98 A7 57 0E
3C C6 5A B3 38 FC 68 BE FF F1 70 4E 82 69 41 95 2B 34 00 80 0F 94 02 A1 AE B9 AC 39 91 7C 14 2E 89 0A 0D C2 5B DA 07 5A F1 1C 5F 1A 78 BB FA

Encrypted : Ecrypted Data :
A5 DA 83 3B E4 CF 0F 42 A1 1D E8 99 09 F5 79 26 C0 32 B1 A2 45 FD 57 EC EC 22 5B 60 D8 E2 E4 B4 EC B7 36 59 2E 3B 1E C2 2D 93 DD AA B2 48 1F CC 27 C4 37 57 11
BC B2 7F A4 80 60 41 2D 8E B7 09 FF 6A 8C A7 D5 69 01 9A 6F D2 47 4C B9 8D CC FF 4D 05 96 0F 51 3E D9 24 4E 1F AD 6C 88 2A 3A E0 6A FD 3A 0E

Encrypted : Ecrypted Data :
DD D3 0A F6 3F 44 13 A8 93 A8 C6 16 D3 6B 4C C0 E5 9B 32 71 E6 D2 24 27 7F D9 E2 67 80 8B E6 67 99 8B 9B 2F CA 4F AA 6A 16 E0 B1 8C C5 1D 37 FB 33 9B 48 CB C6
31 8A EC 3C 12 1C C3 28 9D 0B 70 AF FB 7E 82 07 EF DF 62 54 41 CF 85 5E 11 4A AC 90 01 69 DF BA A2 7E E7 C2 1D AC 21 28 1A DF D3 73 EC 95 82
sarthak@sarthak-HP-Pavillon-Notebook:~/Desktop/IITR Courses/AdvNetwork/v3$

```

5. Decrypted Data at Server end

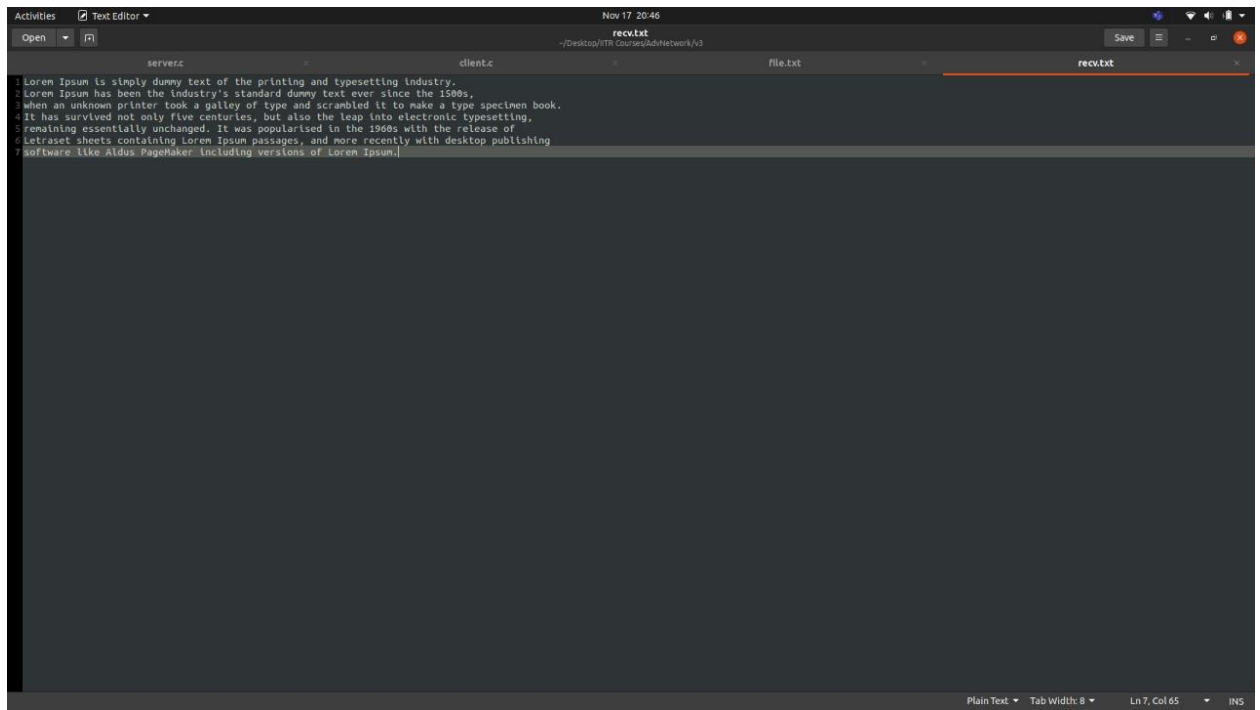


```

sarthak@sarthak-HP-Pavilion-Notebook: ~/Desktop/IITR Courses/AdvNetwork/v3
sarthak@sarthak-HP-Pavilion-Notebook:~/Desktop/IITR Courses/AdvNetwork/v3$ gcc server.c -o server -lssl -lcrypto
sarthak@sarthak-HP-Pavilion-Notebook:~/Desktop/IITR Courses/AdvNetwork/v3$ ./server
Press 1 for Exit 2 for Sending File : 2
Going in listening mode...
Receiving File file.txt
Lorem Ipsum is simply dummy text of the printing and typesetting industry.
Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
when an unknown printer took a galley of type and scrambled it to make a type specimen book.
It has survived not only five centuries, but also the leap into electronic typesetting,
remaining essentially unchanged. It was popularised in the 1960s with the release of
Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing
software like Aldus PageMaker including versions of Lorem Ipsum.
Press 1 for Exit 2 for Sending File : 

```

6. Final File Received



```

1 Lorem Ipsum is simply dummy text of the printing and typesetting industry.
2 Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,
3 when an unknown printer took a galley of type and scrambled it to make a type specimen book.
4 It has survived not only five centuries, but also the leap into electronic typesetting,
5 remaining essentially unchanged. It was popularised in the 1960s with the release of
6 Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing
7 software like Aldus PageMaker including versions of Lorem Ipsum.

```

7. WireShark analysis

Activities Wireshark Nov 17 20:54
Capturing from Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl>/>

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|------------|-------------|----------|--------|---|
| 30 | 16.848893808 | 127.0.0.1 | 127.0.0.1 | TCP | 166 | 37512 → 8080 [PSH, ACK] Seq=419 Ack=1 Win=65536 Len=100 TSval=... |
| 31 | 16.848897099 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=519 Win=65536 Len=0 TSval=305313... |
| 32 | 16.848907328 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 37512 → 8080 [PSH, ACK] Seq=519 Ack=1 Win=65536 Len=2 TSval=3... |
| 33 | 16.848911384 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=521 Win=65536 Len=0 TSval=305313... |
| 34 | 16.848945175 | 127.0.0.1 | 127.0.0.1 | TCP | 166 | 37512 → 8080 [PSH, ACK] Seq=521 Ack=1 Win=65536 Len=100 TSval=... |
| 35 | 16.848948417 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=621 Win=65536 Len=0 TSval=305313... |
| 36 | 16.848958000 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 37512 → 8080 [PSH, ACK] Seq=621 Ack=1 Win=65536 Len=2 TSval=3... |
| 37 | 16.848962988 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=623 Win=65536 Len=0 TSval=305313... |
| 38 | 16.848966044 | 127.0.0.1 | 127.0.0.1 | TCP | 166 | 37512 → 8080 [PSH, ACK] Seq=623 Ack=1 Win=65536 Len=100 TSval=... |
| 39 | 16.848999909 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=723 Win=65536 Len=0 TSval=305313... |
| 40 | 16.849154244 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 37512 → 8080 [FIN, ACK] Seq=723 Ack=1 Win=65536 Len=0 TSval=3... |
| 41 | 16.890758585 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=724 Win=65536 Len=0 TSval=305313... |
| 42 | 38.874765978 | 127.0.0.1 | 127.0.0.53 | DNS | 86 | Standard query 0xf6ce A ssl.gstatic.com OPT |
| 43 | 38.886549076 | 127.0.0.53 | 127.0.0.1 | DNS | 102 | Standard query response 0xf6ce A ssl.gstatic.com A 172.217.16... |
| 44 | 50.243262113 | 127.0.0.1 | 127.0.0.53 | DNS | 87 | Standard query 0xf6e3 A daisy.ubuntu.com OPT |

[Next sequence number: 723 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 1611094376
1000 ... = Header length: 32 bytes (8)
+ Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xf6fc [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
+ TCP Option - No-Operation (NOP)
+ TCP Option - No-Operation (NOP)
+ TCP Option - Timestamps: TSval 305313109, Tsecr 305313109
Kind: Time Stamp Option (8)

Frame (166 bytes) Reassembled TCP (204 bytes)
Loopback: lo - [live capture in progress]

Packets: 110 - Displayed: 110 (100.0%) Profile: Default

Activities Wireshark Nov 17 20:54
Capturing from Loopback: lo

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl>/>

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|--------------|------------|-------------|----------|--------|---|
| 30 | 16.848893808 | 127.0.0.1 | 127.0.0.1 | TCP | 166 | 37512 → 8080 [PSH, ACK] Seq=419 Ack=1 Win=65536 Len=100 TSval=... |
| 31 | 16.848897099 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=519 Win=65536 Len=0 TSval=305313... |
| 32 | 16.848907328 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 37512 → 8080 [PSH, ACK] Seq=519 Ack=1 Win=65536 Len=2 TSval=3... |
| 33 | 16.848911384 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=521 Win=65536 Len=0 TSval=305313... |
| 34 | 16.848945175 | 127.0.0.1 | 127.0.0.1 | TCP | 166 | 37512 → 8080 [PSH, ACK] Seq=521 Ack=1 Win=65536 Len=100 TSval=... |
| 35 | 16.848948417 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=621 Win=65536 Len=0 TSval=305313... |
| 36 | 16.848958000 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 37512 → 8080 [PSH, ACK] Seq=621 Ack=1 Win=65536 Len=2 TSval=3... |
| 37 | 16.848962988 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=623 Win=65536 Len=0 TSval=305313... |
| 38 | 16.848966044 | 127.0.0.1 | 127.0.0.1 | TCP | 166 | 37512 → 8080 [PSH, ACK] Seq=623 Ack=1 Win=65536 Len=100 TSval=... |
| 39 | 16.848999909 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=723 Win=65536 Len=0 TSval=305313... |
| 40 | 16.849154244 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 37512 → 8080 [FIN, ACK] Seq=723 Ack=1 Win=65536 Len=0 TSval=3... |
| 41 | 16.890758585 | 127.0.0.1 | 127.0.0.1 | TCP | 66 | 8080 → 37512 [ACK] Seq=1 Ack=724 Win=65536 Len=0 TSval=305313... |
| 42 | 38.874765978 | 127.0.0.1 | 127.0.0.53 | DNS | 86 | Standard query 0xf6ce A ssl.gstatic.com OPT |
| 43 | 38.886549076 | 127.0.0.53 | 127.0.0.1 | DNS | 102 | Standard query response 0xf6ce A ssl.gstatic.com A 172.217.16... |
| 44 | 50.243262113 | 127.0.0.1 | 127.0.0.53 | DNS | 87 | Standard query 0xf6e3 A daisy.ubuntu.com OPT |

[Next sequence number: 621 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Acknowledgment number (raw): 1611094376
1000 ... = Header length: 32 bytes (8)
+ Flags: 0x018 (PSH, ACK)
Window size value: 512
[Calculated window size: 65536]
[Window size scaling factor: 128]
Checksum: 0xf6fc [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
+ TCP Option - No-Operation (NOP)
+ TCP Option - No-Operation (NOP)
+ TCP Option - Timestamps: TSval 305313109, Tsecr 305313109
Kind: Time Stamp Option (8)

Frame (166 bytes) Reassembled TCP (204 bytes)
Loopback: lo - [live capture in progress]

Packets: 110 - Displayed: 110 (100.0%) Profile: Default



Packet size analysis

Activities Wireshark Nov 17 22:40

Wireshark - Packet Lengths - Loopback: lo

| Topic / Item | Count | Average | Min val | Max val | Rate (ms) | Percent | Burst rate | Burst start |
|------------------|-------|---------|---------|---------|-----------|---------|------------|-------------|
| Packet Lengths | 46 | 90.80 | 54 | 234 | 0.0004 | 100% | 0.3500 | 66.129 |
| 0-19 | 0 | - | - | - | 0.0000 | 0.00% | - | - |
| 20-39 | 0 | - | - | - | 0.0000 | 0.00% | - | - |
| 40-79 | 31 | 66.84 | 54 | 74 | 0.0003 | 67.39% | 0.2800 | 66.129 |
| 80-159 | 7 | 101.29 | 87 | 130 | 0.0001 | 15.22% | 0.0200 | 0.000 |
| 160-319 | 8 | 174.50 | 166 | 234 | 0.0001 | 17.39% | 0.0700 | 66.146 |
| 320-639 | 0 | - | - | - | 0.0000 | 0.00% | - | - |
| 640-1279 | 0 | - | - | - | 0.0000 | 0.00% | - | - |
| 1280-2559 | 0 | - | - | - | 0.0000 | 0.00% | - | - |
| 2560-5119 | 0 | - | - | - | 0.0000 | 0.00% | - | - |
| 5120 and greater | 0 | - | - | - | 0.0000 | 0.00% | - | - |

Display filter: Apply

Copy Save as... Close

References

1. Hands on Network programming with C – Lewi Van Winkle.
2. Geeks for Geeks.