Text

Description automatically generated

**T.Y.B.Tech (CSE)**

Information Security

**Lab Assignment No – B2**

**Name: Aniruddha Shende**

**Roll number: PE04**

**Batch: E1**

**Panel: E**

Letter

Description automatically generated

A picture containing text, document, receipt

Description automatically generated

A sheet of music

Description automatically generated with low confidence

Diagram

Description automatically generated

**SHA Code :**

import hashlib

def to(file,write):

for each in file:

input\_to\_hash = hashlib.sha256(each.encode())

write.write(input\_to\_hash.hexdigest())

write.write("\n")

write=open('SHA.txt','w')

file=open('raw\_text.txt','r')

to(file,write)

write.close()

Graphical user interface, text, application

Description automatically generated

**File which contains the raw text**

Table

Description automatically generated with low confidence

**Resultant text file**

Graphical user interface

Description automatically generated with medium confidence

**Client server authentication using socket programming :**

**Server code :**

//Name : Aniruddha Shende

//Roll no : PE04

//Batch : E1

//Panel : E

#include <unistd.h>

#include <stdio.h>

#include <sys/socket.h>

#include <stdlib.h>

#include <netinet/in.h>

#include <string.h>

int main()

{

int server\_fd, new\_socket;

struct sockaddr\_in address;

int addrlen = sizeof(address);

char server\_buffer[1024] = {0};

char message[1024] = {0};

if ((server\_fd = socket(AF\_INET, SOCK\_STREAM, 0)) == 0)

{

perror("\n\nsocket failed to create");

exit(0);

}

address.sin\_family = AF\_INET;

address.sin\_addr.s\_addr = INADDR\_ANY;

address.sin\_port = htons(6542);

bind(server\_fd, (struct sockaddr \*)&address, sizeof(address));

listen(server\_fd, 3);

new\_socket = accept(server\_fd, (struct sockaddr \*)&address, (socklen\_t \*)&addrlen);

printf("\n\nClient Connected!\n");

int z = 0;

while (z!=1)

{

memset(&server\_buffer, '\0', 1024);

memset(&message, '\0', 1024);

if (read(new\_socket, server\_buffer, 1024) < 0)

{

printf("\n\nCan't Listen...");

}

printf("\n\nChallenge number received from the client is : %s", server\_buffer);

if (strcmp(server\_buffer, "exit") == 0)

break;

printf("\n\nCipher reply sent to client is : ");

gets(message);

if (send(new\_socket, message, strlen(message), 0) < 0)

{

printf("\n\nMessage not sent");

}

if (strcmp(message, "exit") == 0)

break;

z++;

}

close(server\_fd);

close(new\_socket);

printf("\n");

return 0;

}

**Client code :**

//Name : Aniruddha Shende

//Roll no : PE04

//Batch : E1

//Panel : E

#include <stdio.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

int main()

{

int sock = 0;

struct sockaddr\_in serv\_addr;

char message[1024] = {0};

char client\_buffer[1024] = {0};

if ((sock = socket(AF\_INET, SOCK\_STREAM, 0)) < 0)

{

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

exit(0);

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(6542);

inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr);

if (connect(sock, (struct sockaddr \*)&serv\_addr, sizeof(serv\_addr)) < 0)

{

printf("\nConnection Failed \n");

exit(0);

}

int z = 0;

while (z!=1)

{

memset(&client\_buffer, '\0', 1024);

memset(&message, '\0', 1024);

printf("\n\nEnter the number to be sent to the server : ");

gets(message);

printf("\n\nChallenge number sent to the server is : %s",message);

if (send(sock, message, strlen(message), 0) < 0)

{

printf("\n\nMessage not sent");

}

if (strcmp(message, "exit") == 0)

break;

if (read(sock, client\_buffer, 1024) < 0)

{

printf("\n\nCan't Listen...");

}

printf("\n\nCipher reply received from the server is : %s", client\_buffer);

if (strcmp(client\_buffer, "exit") == 0)

break;

z++;

}

printf("\n\nREcovered number is : %s",message);

printf("\n\nClient has successfully verified the server\n");

close(sock);

printf("\n");

return 0;

}

**Output Screenshot :**

On server side On client Side

Graphical user interface, text

Description automatically generated