PROJECT TITLE: TRIPLE CRYPT – HIGH DELIVERABLE PASS-CODE SYSTEM

ABSTRACT

Triple crypt as the title suggests is a cryptography system which is used to protect the data specifically passwords by using three layers of security algorithms. The entire system is built in C programming language using various functions and libraries of the C. The main idea of creating this system is to provide a high level security to the weak passwords, as such weak passwords are easy to crack and their hashes are easily available on the internet. In this system the password will undergo three layers of protection- encryption, salting and hashing. We will be using AES for encryption and for hashing we will be using SHA256. After hashing we will be doing salting on it. A salt is simply added to make a password hash output unique even for users adopting common passwords. AES and SHA256 are the best combination according to the study for security.

Keywords

Hashing, salting, AES encryption, Triple layer, SHA256, C language, Security

TABLE OF CONTENT

Introduction

C is a procedural programming language which was developed by Dennis Ritchie in 1972.It was mainly developed as a system programming language to write an operating system. The main features of C language include low-level access to memory, simple set of keywords and clean style which makes it suitable for system programming.

Cryptography is the study and practice of techniques for secure communication in the presence of third parties called adversaries. It deals with developing and analyzing protocols which prevents malicious third parties from retrieving information being shared between two entities thereby following the various aspects of information security. Cryptography holds a very special thing for the security purpose which is commonly known as Encryption. . Encryption is the conversion of data from a readable form, called plaintext, into a form, called cipher text that cannot be easily understood by unauthorized people.

Nowadays , security of data , passwords, files etc. From being hacked has become a very crucial step before accessing anything online because there are millions of hackers present all over the world. This project will use above mentioned methodologies to create a system providing security of personal data of users/students/employees. The main purpose fulfilled by this project would be to provide triple security to the passwords of any signing account.

Literature Review

Encryption is a process that uses digital keys to encode various components like text , files, databases, passwords, application or network packet, so that appropriate user, system process, and so on can decode the item and view , modify, or add the data. Cloudera provides encryption mechanisms to protect data persisted to disk or other storage media and as it moves over the network.

Hashing is a method of cryptography that converts any form of data into a unique string of text. Any piece of data can be hashed, no matter its size or type. Regardless of the data's size, type, or length, the hash that any data produces is always the same length. A hash is designed to act as a one-way function. Hash Function is a function which has a huge role in making a system secure as it converts normal data given to it as an irregular value of fixed length.
One of the most successful and secure algorithms is SHA256. In simple words, SHA-256 (Secure Hash Algorithm, FIPS 182-2), is one of the cryptographic hash functions which has digest length of 256 bits. It's a keyless hash function, meaning an MDC (Manipulation Detection Code). The SHA-256 algorithm is one flavor of SHA-2 (Secure Hash Algorithm 2), which was created by the National Security Agency in 2001 as a successor to SHA-1.

Salting is an additional technique of hashing. Salting is a concept that typically pertains to password hashing. Essentially, it's a unique value that can be added to the end of the password to create a different hash value. This adds a layer of security to the hashing process, specifically against brute force attacks. A brute force attack is where a computer or botnet attempts every possible combination of letters and numbers until the password is found.

Problem Statement

People use simple passwords to protect their data, Which are very easy to crack as their hashes are available on the internet and by using rainbow attacks they can be cracked. There is a need to create a system which will protect these weak and simple passwords from getting cracked. So that no information could be leaked.

Objectives

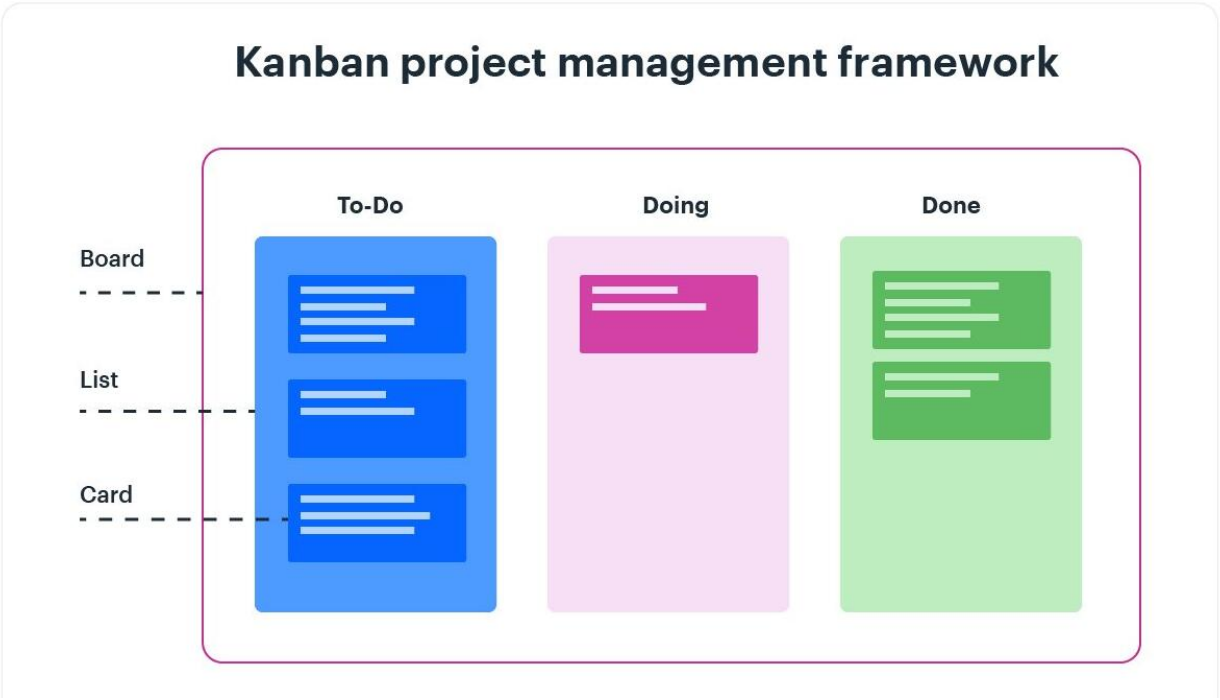
To provide Triple layer security to the passwords-

a) The opportunity of using simple passwords
b) Protecting the easy to crack passwords
c) Protecting passwords from rainbow attacks
d) Storing keys, salts in unreadable form

Methodology

The project will be developed using the Agile methodology implemented via Kanban in github

The Triple crypt system will make use of “sha256.h” and “crypto.h” library for hashing and “aes.h” library for AES encryption. This system will use the concept of file handling and memory allocation.



Kanban Approach-

1) To-do list describes the work to be done or changed. It increases when the requirements increases
2) Cards are taken from to-do list to the doing list in the kanban board to start working or developing them.
3) When they are built then they are tested. If test passes then they are shifted to the done list otherwise goes back to the to-do list
4) Using Kanban the entire development process is flexible and provides the stats of the work in progress

System Requirements

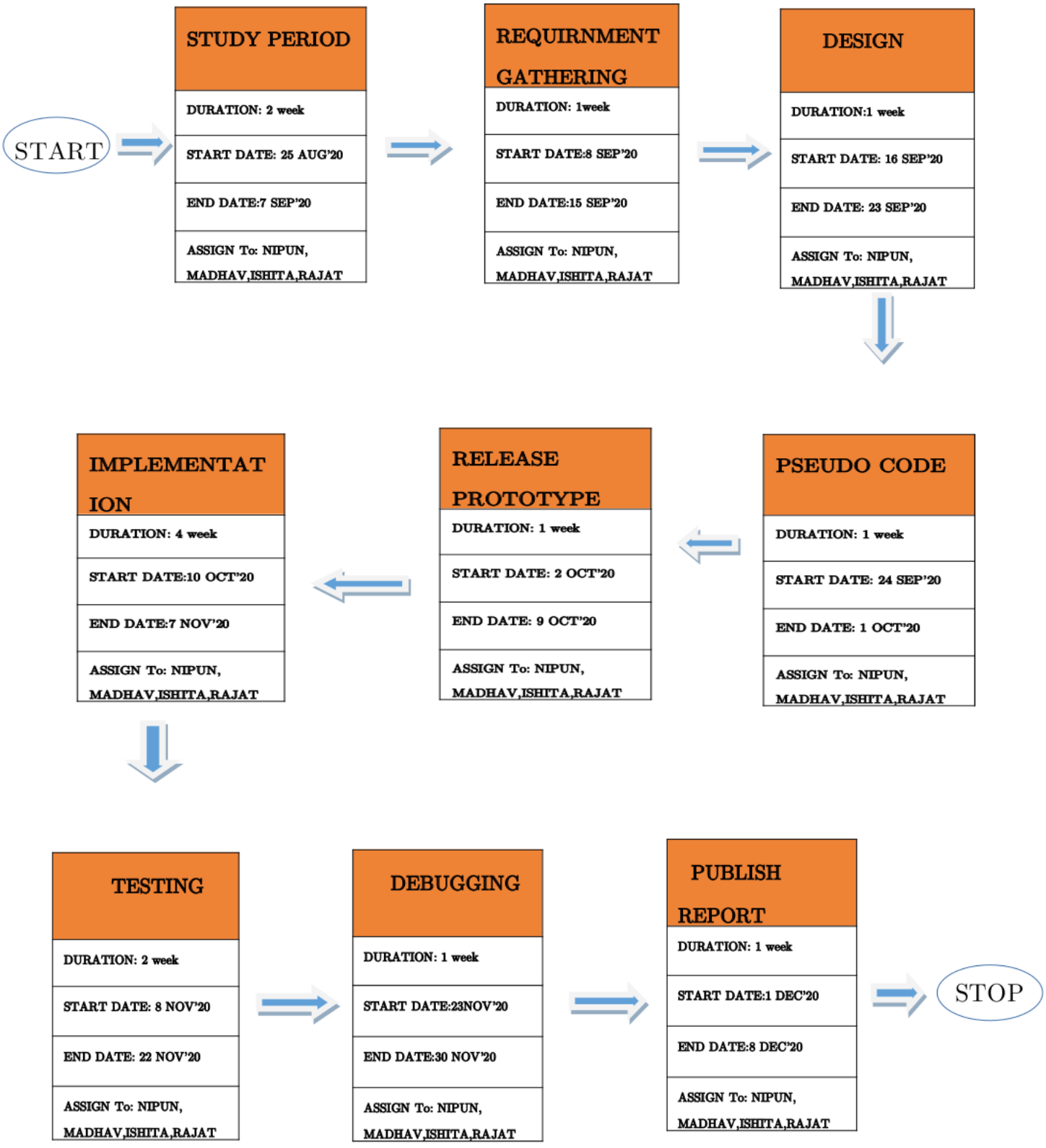
For windows 7/8/10--

CPU: Intel Core i3 @ 2.30GHz processor
RAM: 1 GB
Video card: 256 or 512 MB
Compiler: C language

For Linux/Mac--

CPU: Intel Core i3 @ 2.30GHz processor
RAM: 1 GB
Video card: 256 or 512 MB
Compiler: C language

Schedule


START
STUDY PERIOD
DURATION: 2 week
START DATE: 25 AUG'20
END DATE:7 SEP'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
REQUIRNMENT GATHERING
DURATION: 1week
START DATE:8 SEP'20
END DATE:15 SEP'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
DESIGN
DURATION:1 week
START DATE: 16 SEP'20
END DATE: 23 SEP'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
PSEUDO CODE
DURATION: 1 week
START DATE: 24 SEP'20
END DATE: 1 OCT'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
RELEASE PROTOTYPE
DURATION: 1 week
START DATE: 2 OCT'20
END DATE: 9 OCT'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
IMPLEMENTATION
DURATION: 4 week
START DATE:10 OCT'20
END DATE:7 NOV'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
TESTING
DURATION: 2 week
START DATE: 8 NOV'20
END DATE: 22 NOV'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
DEBUGGING
DURATION: 1 week
START DATE:23NOV'20
END DATE:30 NOV'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
PUBLISH REPORT
DURATION: 1 week
START DATE:1 DEC'20
END DATE:8 DEC'20
ASSIGN To: NIPUN, MADHAV,ISHITA,RAJAT
STOP

Algorithm

1) start

2) MAIN file starts

3) Include aes.c , salting.c, hashing.c, login.c files

4) Define variable choice and a character array cipher[16]

5) Make display() function of the starting page.

6) Make display2() function to get the password and choice of the user.

7) Make display3() function for login and signup

8)Make a check() function to check the already saved username and
Avoid duplication.

9) Make a main function and call display() and display3() functions in it.

10) MAIN file ends

11) AES file starts

12) Define unsigned char key[16] array.

13) Define unsigned char s[256] lookup table, unsigned char m2[256] multiple of 2, unsigned char m3[256] multiple of 3 table and unsigned char rcon[10] table.

14) Define function roundkey

```
for (int i=0; i<16;i++)

text[i]=text[i] ^ key1[i]
```

15) Define function subbytes

```
for(int g=0;g<16;g++)
```

16) Change text to ASCII value and store it in integer value.

17) See the lookup table at respective ASCII value s[value].

18) Define function shiftrow

19) Define an unsigned char temp[16] array.

20) For 1st row shift nothing
    For 2nd row shift one element to the right
    For 3rd row shift two element to the right
    For 4th row shift three element to the right

21) for(int f=0;f<16;f++) put all the elements from temp to cipher.

    text2[f]=temp[f]

22) Define function mixcolumns

23) Define an unsigned char temp[16] array.

24) Do matrix multiplication and perform xor instead of addition

  temp[0]= m2[(int)text3[0]] ^ m3[(int)text3[1]] ^ text3[2] ^ text3[3]

25) for(int l=0;l<16;l++) put all the elements from temp to cipher.

    text3[l]=temp[l]

26) Define function keyexpansion

27) Define an unsigned char temp[16] array.

28) For first column of new key

    temp[0]=s[(int)key0[13]]^key0[0]^rcon[z]

29) For rest of the columns of new key

temp[4]=temp[0]^key0[4]

30) for(int a=0;a<16;a++) put all the elements from temp to key.

key0[a]=temp[a]

31) Define main function.

32) call roundkey function for 1st round.

33) for(int y=0;y<9;y++) for 9 rounds call

subbyte(cipher)
shiftrow(cipher)
mixcolumns(cipher)
keyexpansion(key,y)
roundkey(cipher,key)

34) For final round call

subbyte(cipher)
shiftrow(cipher)
keyexpansion(key,9)
roundkey(cipher,key)

35) AES file ends

36) HASHING file starts

37) Define rightrotate for rotating the bits.

38) Define arr[64],value[64] and h[8],k[64]

39) Make a bitchager() function which will convert 8bits elements to one 32 bit element.

for(int j=0,l=0;j<16;j++,l+=3)

value[j]=(arr[j+l]<<24) | (arr[j+l+1]<<16) | (arr[j+l+2]<<8) | (arr[j+l+3])

40)Make newbytes() functions to generate the remaining 48 words.

```
    for(int i=16;i<64;i++)

 uint32_t s0=rightrotate(value[i-15],7) ^ rightrotate(value[i-15],18) ^
(value[i-15]>>3)

uint32_t s1= rightrotate(value[i-2],17) ^ rightrotate(value[i-2],19) ^   (value[i-
2]>>10)

   value[i]=value[i-16]+s0+value[i-7]+s1
```

41) Make compression() function to calculate the hash value. for that we will be using the formulas.

```
    for(int j=0;j<64;j++)

 uint32_t s0=rightrotate(a[4],6) ^ rightrotate(a[4],11) ^ rightrotate(a[4],25)

   uint32_t ch=(a[4] & a[5]) ^ (~a[4] & a[6])

   uint32_t temp1=a[7]+s0+ch+k[j]+value[j]

   uint32_t s1=rightrotate(a[0],2) ^ rightrotate(a[0],13) ^ rightrotate(a[0],22)

   uint32_t maj=(a[0] & a[1]) ^ (a[0] & a[2]) ^ (a[1] & a[2])

   uint32_t temp2=maj+s1

   a[7]=a[6]
   a[6]=a[5]
   a[5]=a[4]
   a[4]=a[3]+temp1
   a[3]=a[2]
   a[2]=a[1]
   a[1]=a[0]
   a[0]=temp1+temp2
```

42) Make a hashing() function to call all these functions and to perform the padding process

```
for(int i=0;i<64;i++)

  if(i<16)
 arr[i]=(int) cipher[i]


  if(i>=17 && i<63)
  arr[i]=0

 if(i==63 || i==16)
 arr[i]=128
```

43) HASHING file ends

44) SALTING file starts

45) Define the variables and array arr1,binary.

46) Make a function salt() to calculate the salt value

47) To generate salt use the formula

```
arr1[i]= (rand() % (lower-upper+1)) + lower

 value1=(int) arr1[i]
```

48) Now convert the salt in binary form and store in binary array

```
while(value1>0)

   rem=value1%2
   result=result+(a*rem)
   value1=value1/2
   a=a*10
```

49) Then check if the salt generated already exists in the file or not. If it exists then do recursion.

```
for(int j=0;j<32;j++)

fscanf(fp,"%d",&test)

  if(test==binary[j])
   count++

  if(count>29)
   salt()
   break
```

50) Make a salting() function to call all the above functions and to store the salting values,hash values to the files

```
for(int q=0;q<8;q++)
fprintf(hfp,"%x",h[q])
 printf("%x",h[q])

for(int f=0;f<32;f++)
number=(int)arr1[f]
fprintf(hfp,"%x",number)
 printf("%x",number)
```

51)SALTING file ends

52) LOGIN file starts

53) Declare the variables

54) Make a decimal() function to convert the binary number to decimal number

```
while(data)
```

```
ldigit=data%10
data=data/10
result2+=ldigit*base
 base=base*2
```

55) Make a login function in which Inputted name will be matched in the file.

```
 while(!feof(fp1))
memset(test, '\0',sizeof(test))

  fscanf(fp1,"%s",&test)

   count+=1

 if(strcasecmp(test,name1)==0)
       flag=1
       break
```

b) Inputted password will go through aes,hashing and salting and then matched with the password in the file

56) If both name and password match in the file. Successful login will be shown otherwise incorrect username and password will be shown.

57) LOGIN file ends

58) Stop

Results

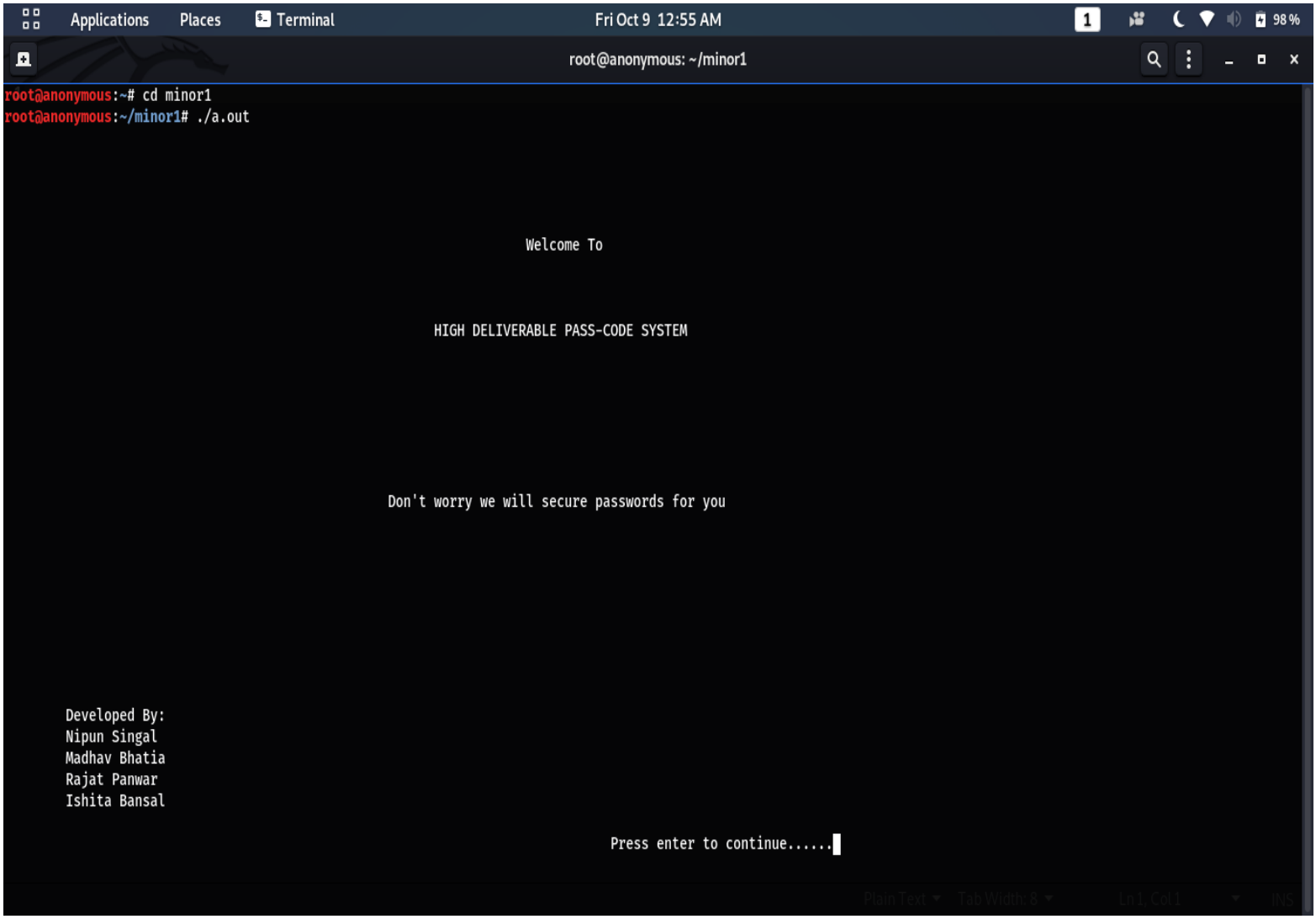


Fig-1.0 Starting page of the program

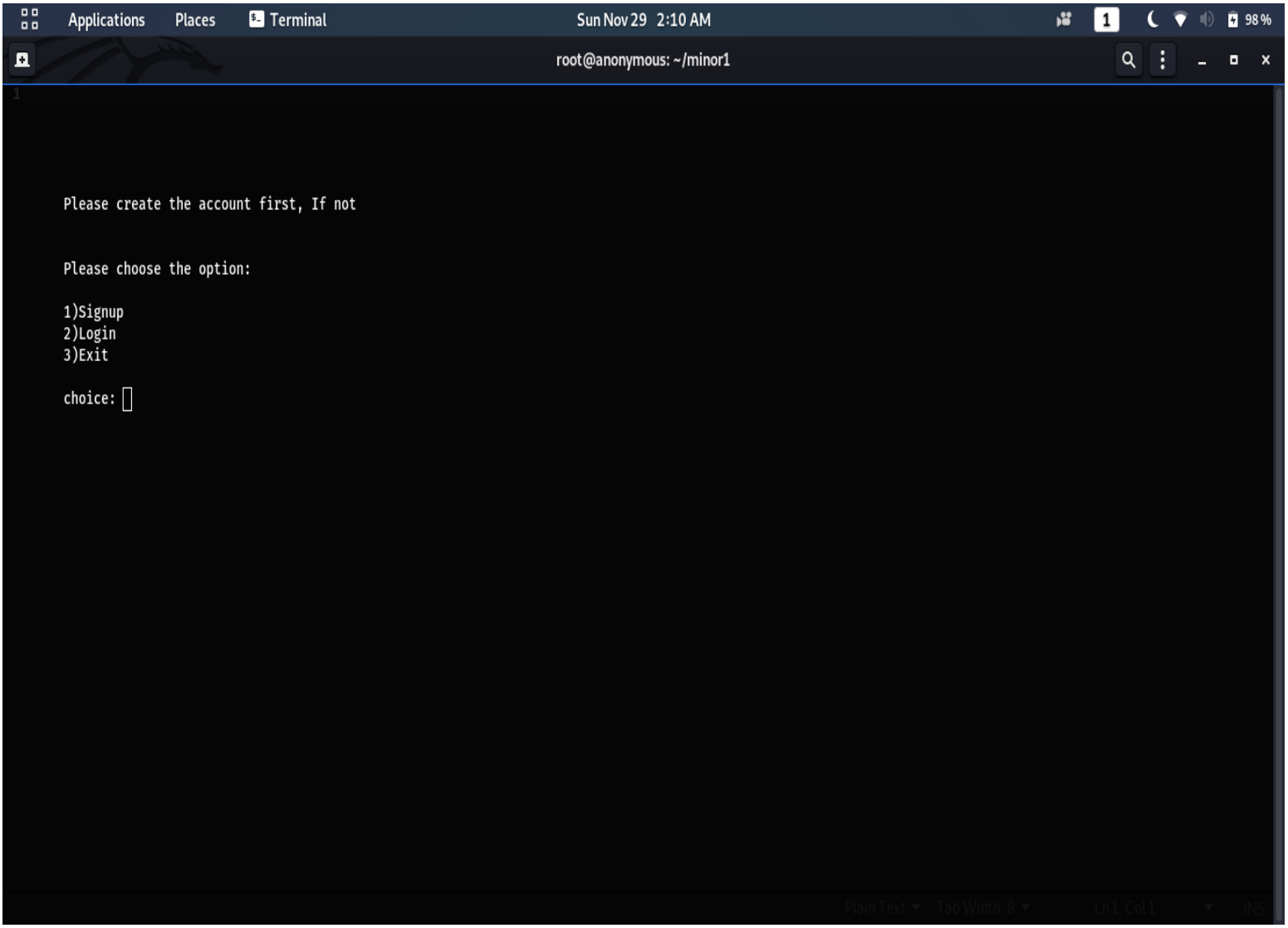


Fig-1.1 Page showing choose to login or signup or exit the program

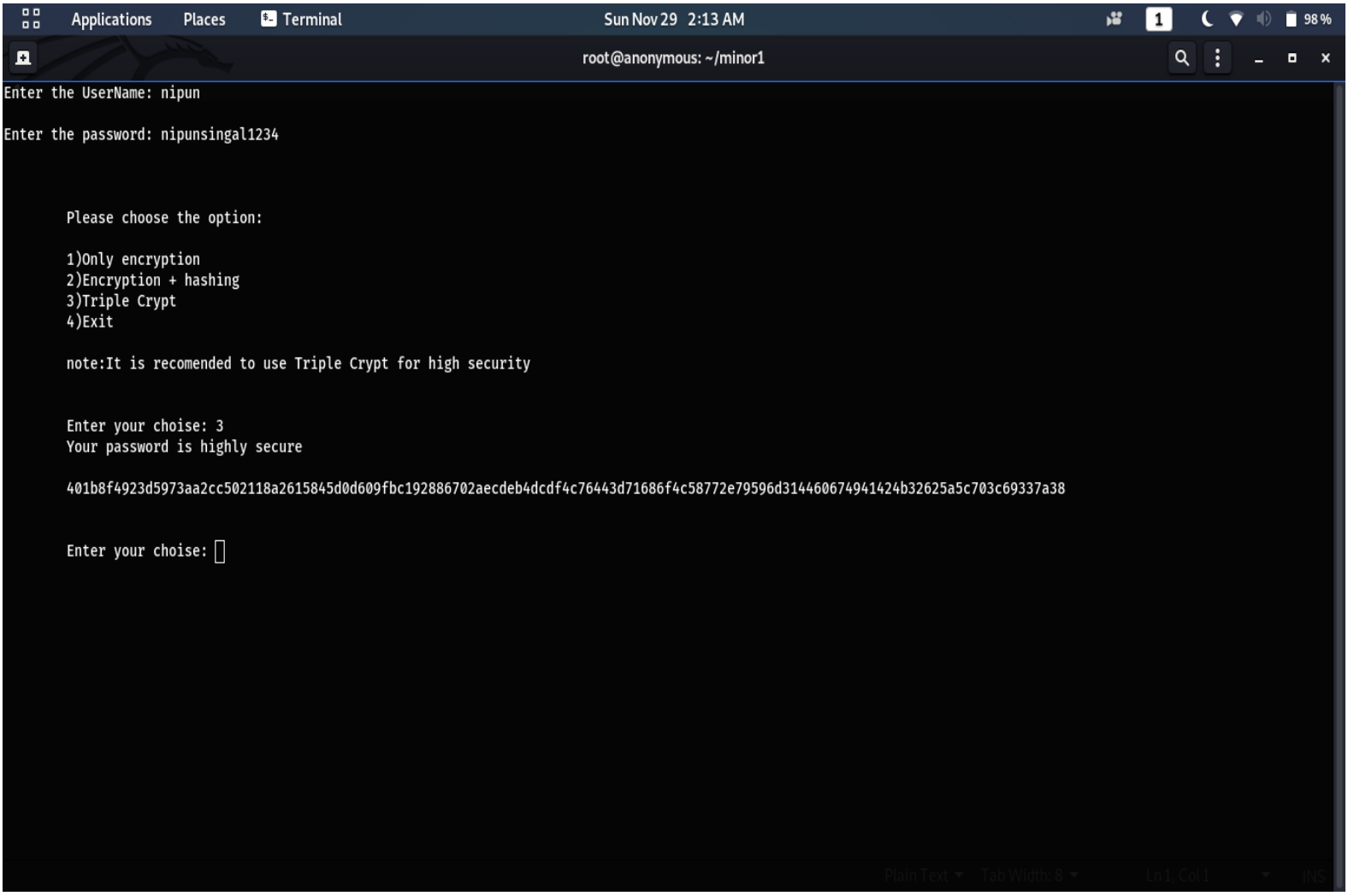


Fig-1.2 Signup page for creating new user

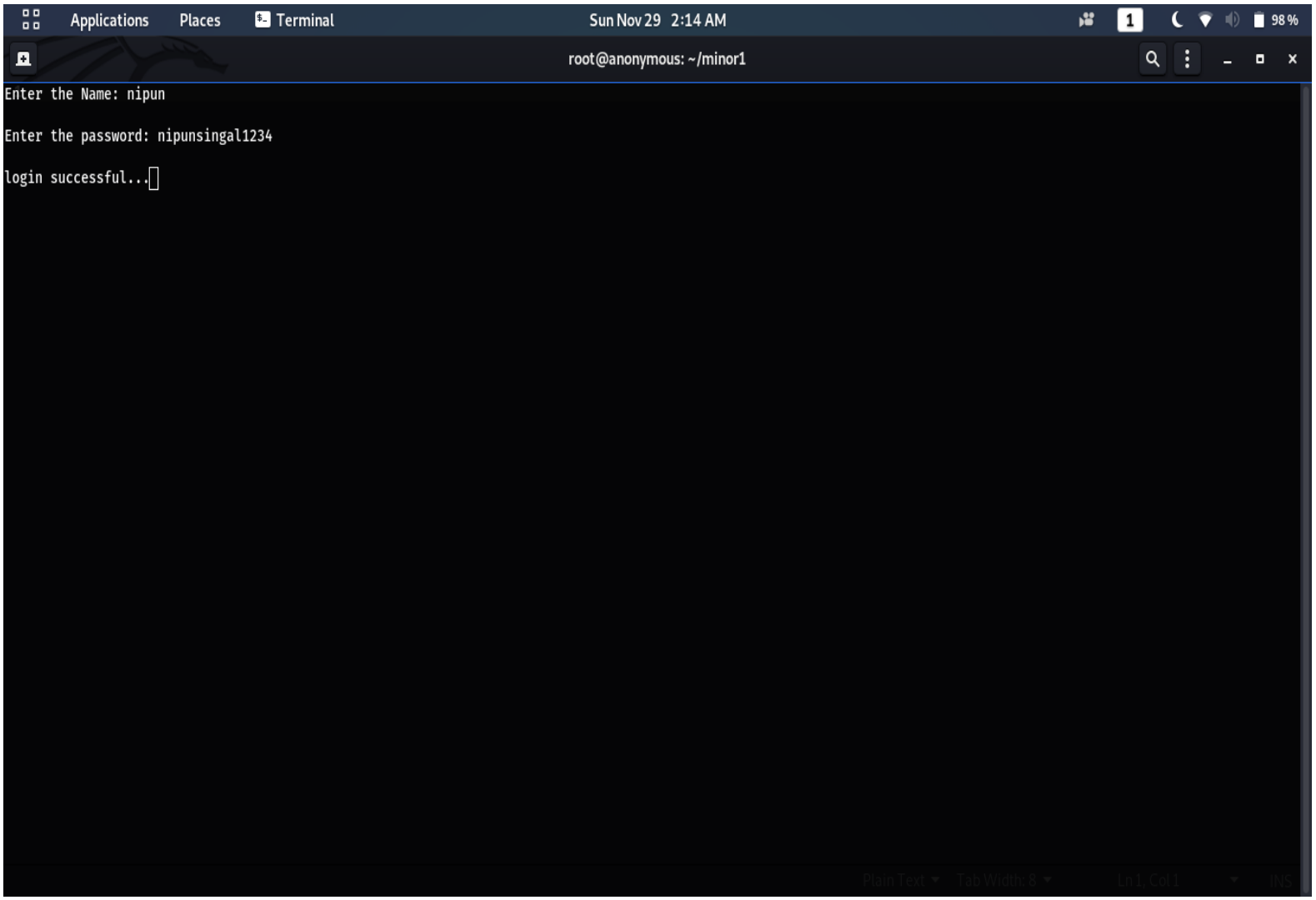


Fig-1.3 Login page

Graphs

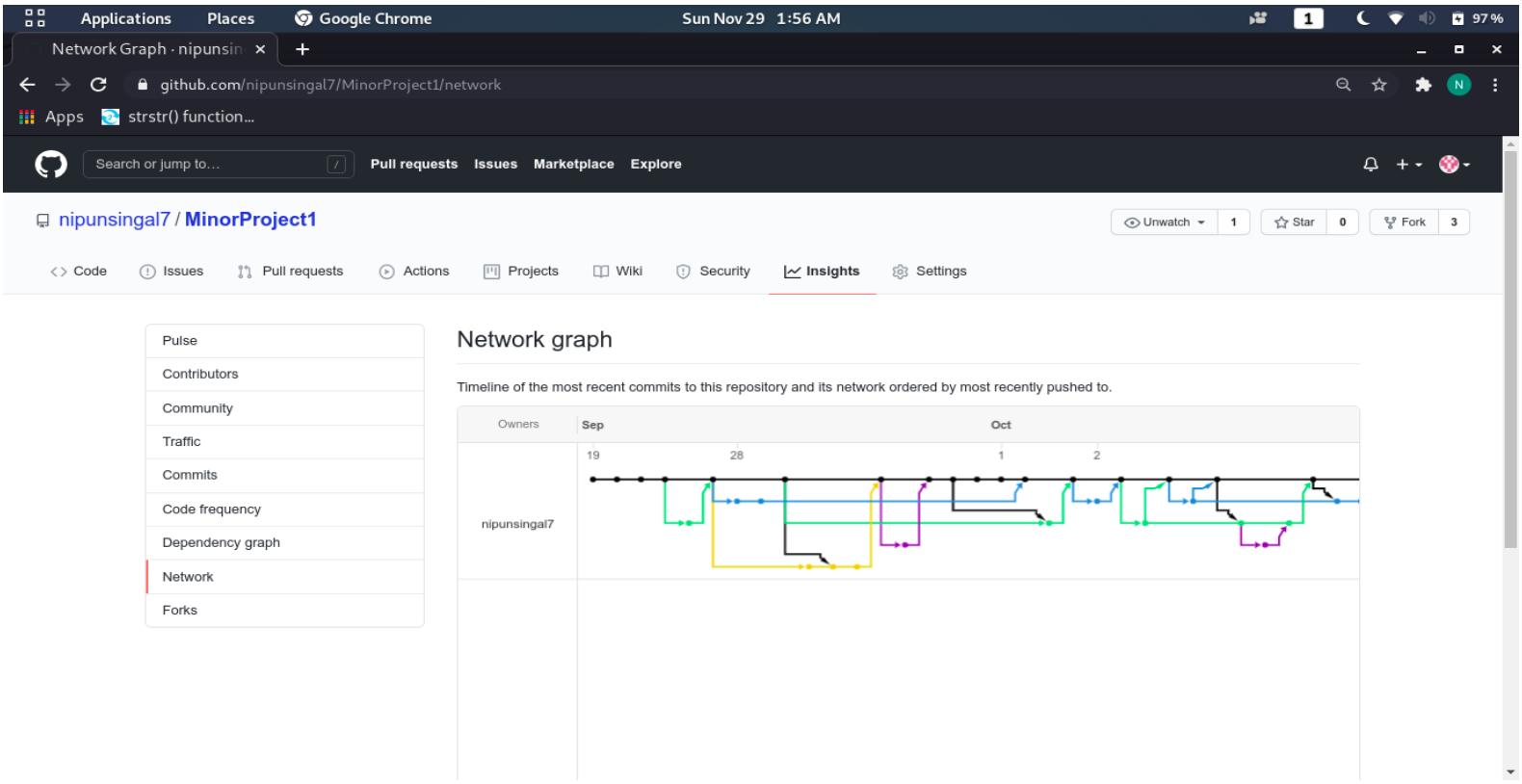


Fig-1.4 Project network graph1

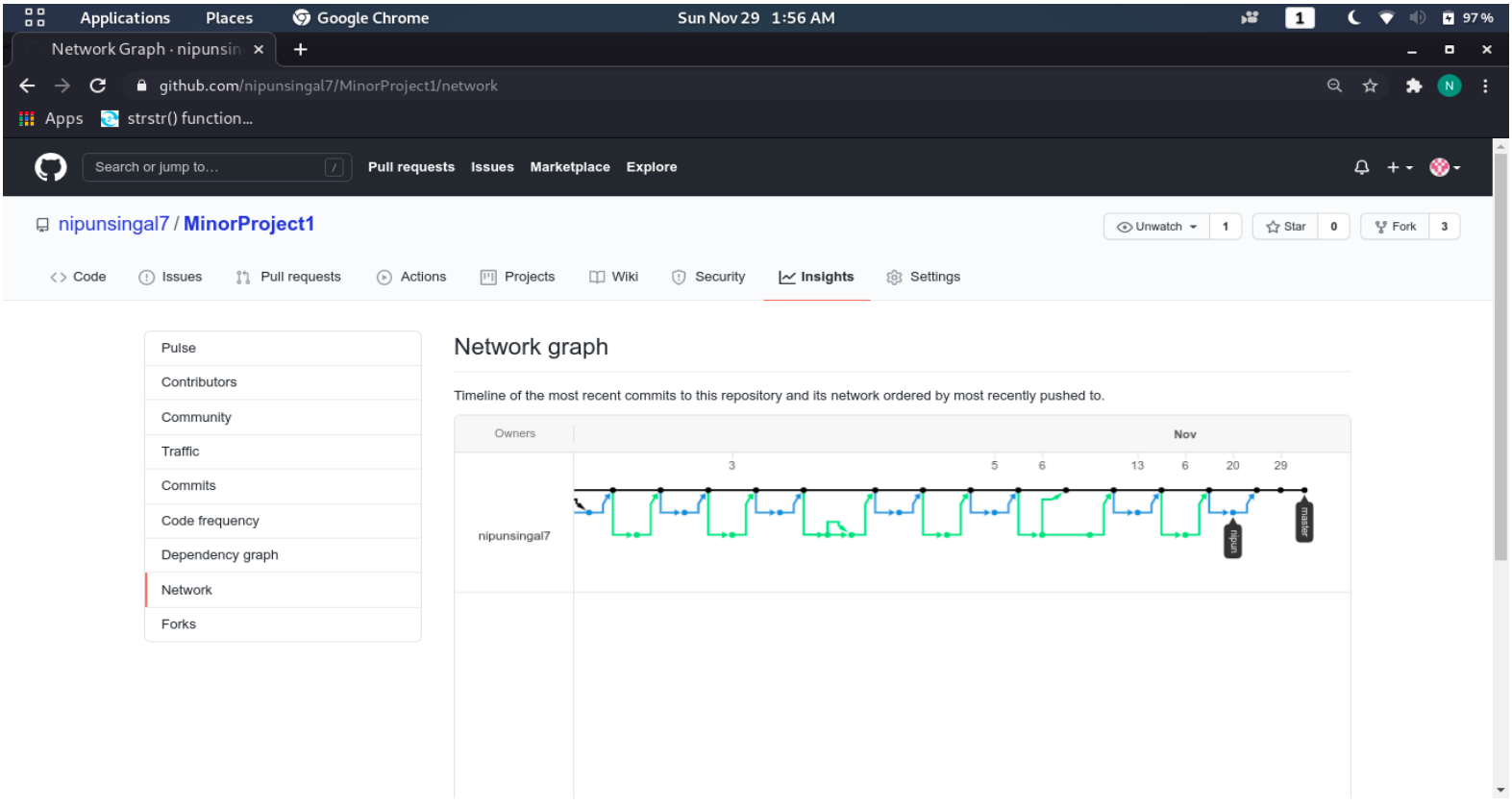


Fig-1.5 Project network graph2

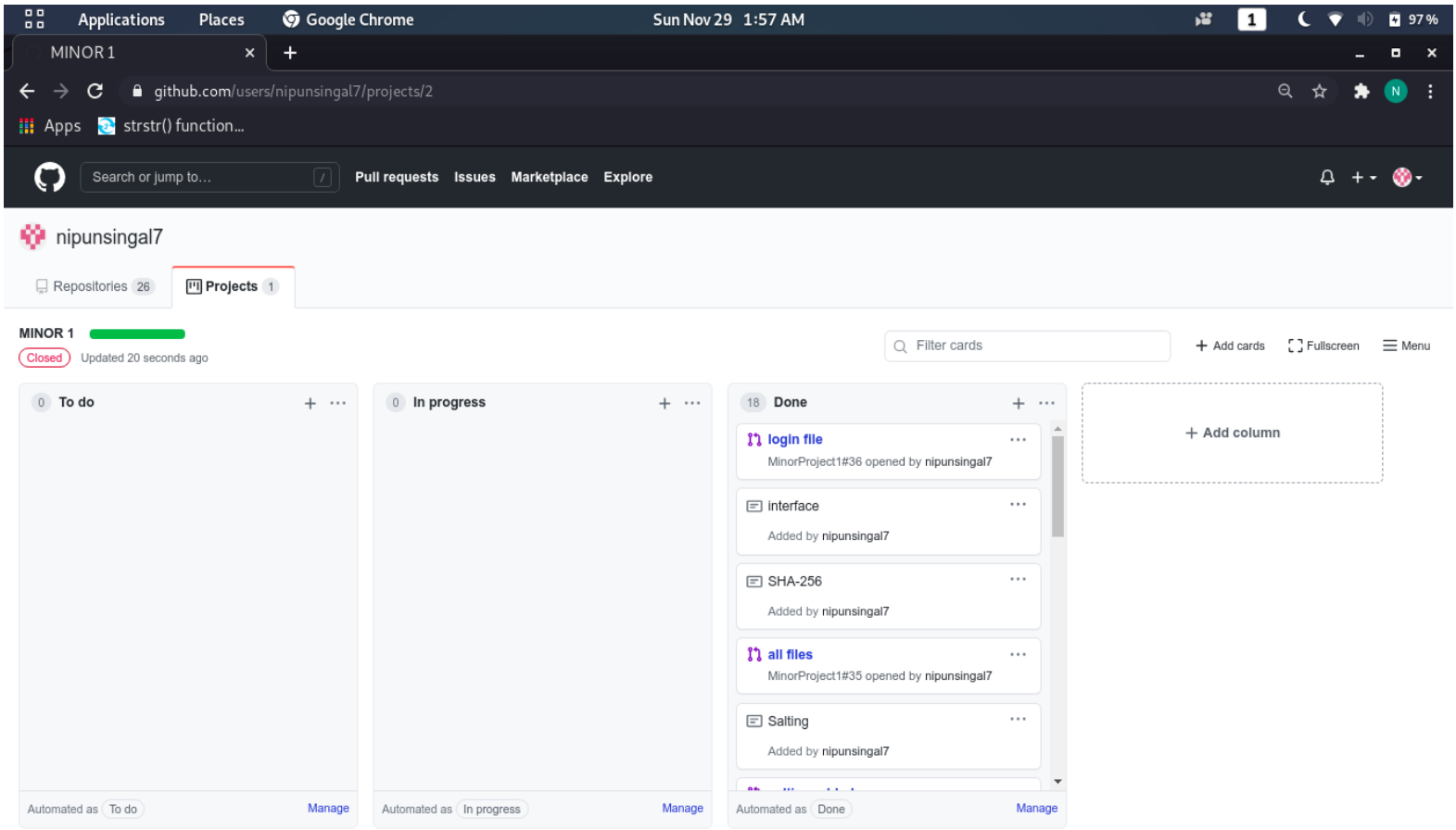


Fig-1.6 Project management page

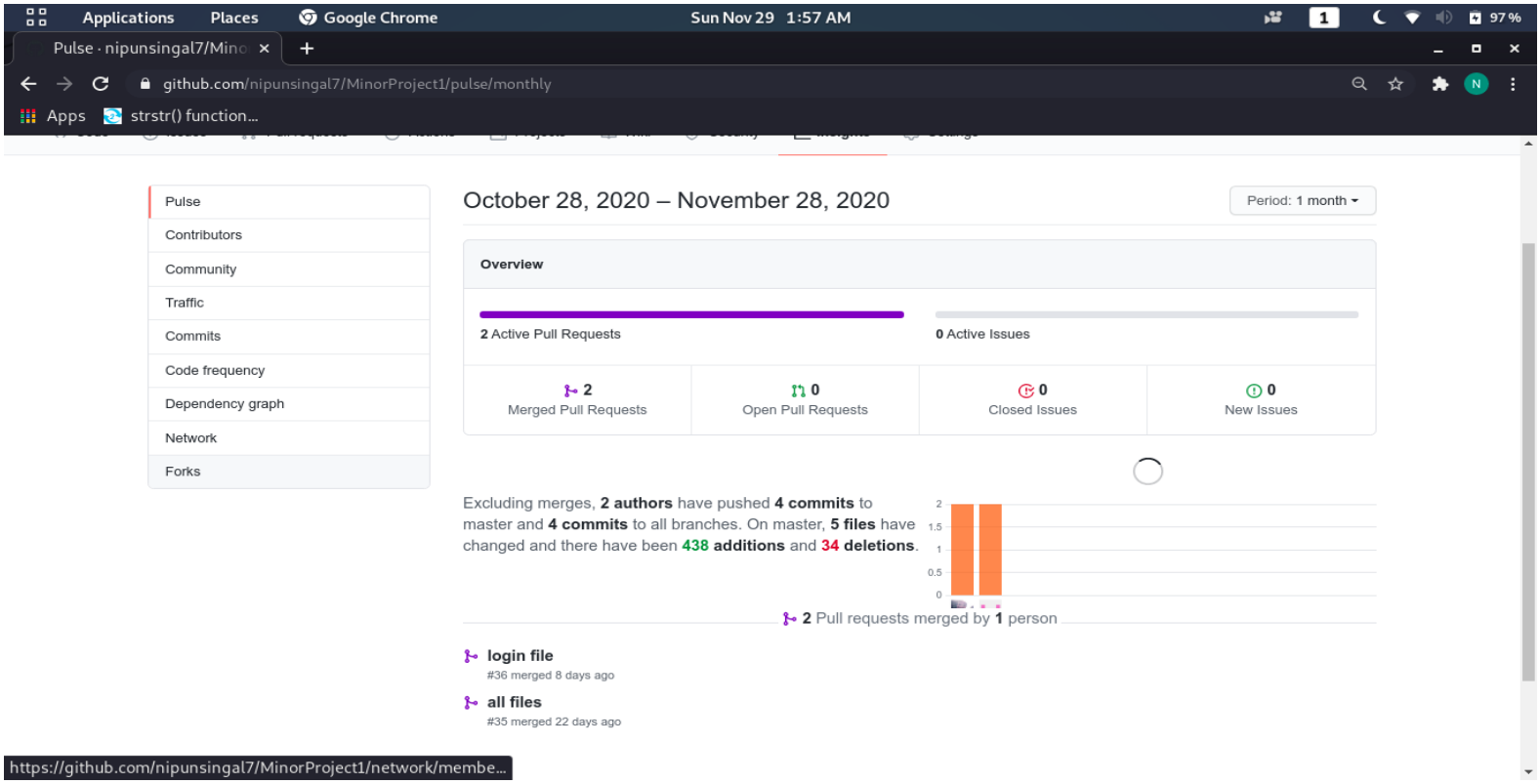


Fig-1.7 Pull and push statistics

Diagrams

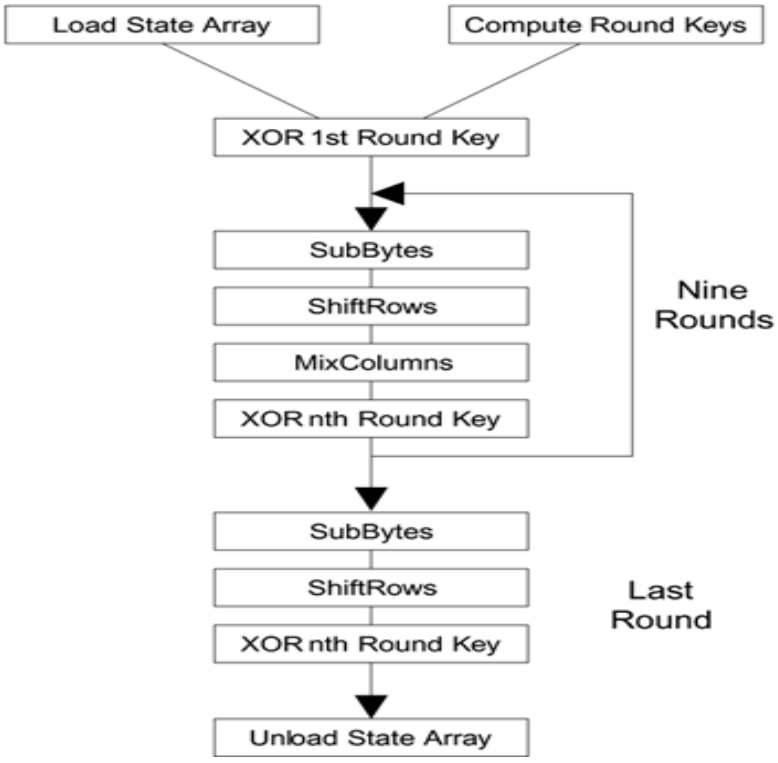


Fig-1.8 AES working model

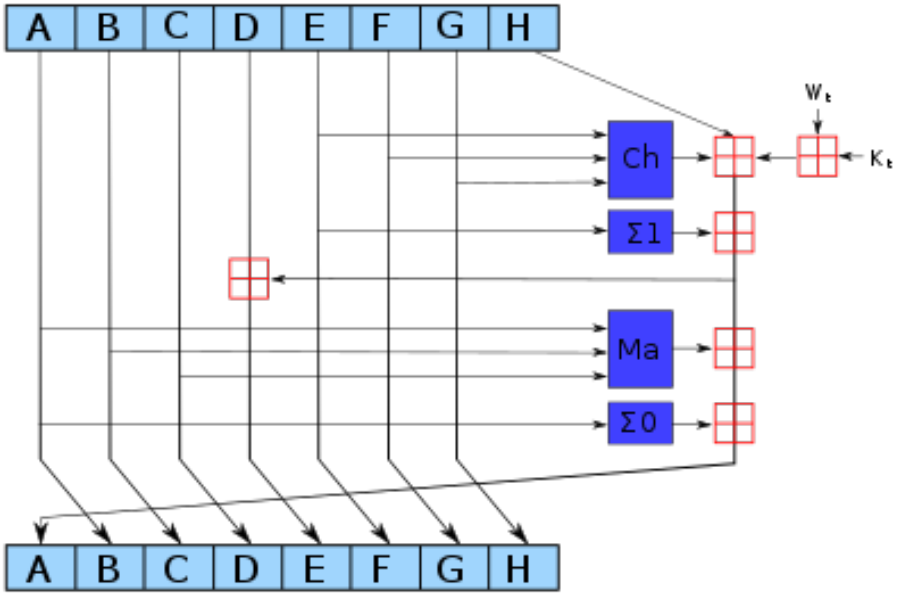


Fig-1.9 HASHING working model

Appendix

Code--

MAIN FILE

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include "aes.c"
#include "hashing.c"
#include "salting.c"
#include "login.c"

unsigned char cipher1[16];
unsigned char cipher2[16];
int q=0;
unsigned char testname[20];
unsigned char name[20];


void display()
{printf("\n\n\n\n\n\t\t\t\t\t\t\t\t    Welcome To\t\t\t\t\t\t\t\t\t");
 printf("\n\n\n\n\t\t\t\t\t\t\tHIGH DELIVERABLE PASS-CODE SYSTEM\t\t\t\t\
t\t\t");
 printf("\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t  Don't worry we will secure passwords for
you\t\t\t\t\t\t");
 printf("\n\n\n\n\n\n\n\n\n\n\tDeveloped By:\n\tNipun Singal\n\tMadhav
Bhatia\n\tRajat Panwar\n\tIshita Bansal");
 printf("\n\n\t\t\t\t\t\t\t\t\t\t\t\t\tPress enter to continue.....");

}

void check()
{ FILE *fp0=fopen("name.txt","r");
  memset(name,'\0',sizeof(name));
  printf("Enter the UserName: ");
```

```
  scanf("%s",&name);

   while(!feof(fp0))
  {
     fscanf(fp0,"%s",&testname);
     if(strcasecmp(testname,name)==0)
     {printf("username already taken\n\n");
       fclose(fp0);
       check();
      break;}


   }


}


void display2()
{
 int choice=0;

check();

printf("\nEnter the password: ");
 scanf("%s",&cipher1);
 printf("\n\n\n\tPlease choose the option:");
 printf("\n\n\t1)Only encryption\n\t2)Encryption + hashing\n\t3)Triple Crypt\
n\t4)Exit\n\n\tnote:It is recomended to use Triple Crypt for high security");



 while(choice!=4)
{memset(cipher2, '\0',sizeof(cipher2));
 strcpy(cipher2,cipher1);

 printf("\n\n\n\tEnter your choise: ");
 scanf("%d",&choice);
 switch (choice)
{case 1:printf("\n\n\tYour password is very less secure");
```

```
        aes(cipher2);
        printf("\n\n\tEncrypted message is: ");
        for(int l=0;l<16;l++)
        {
          printf("%x",cipher2[l]);

        }

        break;

case 2:printf("\tYour password is somewhat secure\n\n");
         aes(cipher2);
         hashing(cipher2);
         for(int m=0;m<8;m++)
         {printf("%x",h[m]);  }
         break;

case 3:printf("\tYour password is highly secure\n\n");

          FILE *kp=fopen("name.txt","a+");

            fseek(kp,0,SEEK_END);

             if(ftell(kp)!=0)
            {fprintf(kp,"\n");}

           for(int c=0;c<strlen(name);c++)
           {fprintf(kp,"%c",name[c]);}


            fclose(kp);

         aes(cipher2);
         hashing(cipher2);
         salting(h);
         break;


case 4: break;
```

```
 default: printf("invalid choice, select again");
          break;

}
}
}



void display3()
{

  while(q!=3)
  {printf("\n\n\n\n\n");
  printf("\tPlease create the account first, If not");
  printf("\n\n\n\tPlease choose the option:");
  printf("\n\n\t1)Signup\n\t2)Login\n\t3)Exit");

  printf("\n\n\tchoice: ");
  scanf("%d",&q);
  switch (q)
  {
    case 1: system("clear");
            display2();
            system("clear");
            break;

  case 2:  system("clear");
            login();
            system("clear");
            break;



  case 3: system("exit");
           break;

  default: printf("invalid choice, select again");
          break;
```

```
  }

}



}

int main()
{char a;
 display();
scanf("%c",&a);
system("clear");
display3();
 return(0);}
```

AES FILE

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>


void subbytes(unsigned char *text1);
void roundkey(unsigned char *text,unsigned char *key1);
void mixcolumns(unsigned char *text3);
void keyexpansion(unsigned char *key0, int z);
void shiftrow(unsigned char *text2);



void roundkey(unsigned char *text,unsigned char *key1)
{
        for (int i=0; i<16;i++)
        {       text[i]=text[i] ^ key1[i];



        }



}


void subbytes(unsigned char *text1)
{  for(int g=0;g<16;g++)
   {int value= (int) text1[g];
      text1[g]=s[value];


   }


}
```

```
void shiftrow(unsigned char *text2)
{ unsigned char temp[16];


 temp[0]=text2[0];                          //1st row, nothing happens
 temp[1]=text2[5];                           //2nd row, 1 shift
 temp[2]=text2[10];                          //3rd row, 2 shift
 temp[3]=text2[15];                          //4th row, 3 shift



  temp[4]=text2[4];
  temp[5]=text2[9];
  temp[6]=text2[14];
  temp[7]=text2[3];



  temp[8]=text2[8];
  temp[9]=text2[13];
  temp[10]=text2[2];
  temp[11]=text2[7];



  temp[12]=text2[12];
  temp[13]=text2[1];
  temp[14]=text2[6];
  temp[15]=text2[11];


  for(int f=0;f<16;f++)
  { text2[f]=temp[f];

  }
}
```

```
void mixcolumns(unsigned char *text3)
{ unsigned char temp[16];

    //1st column
   temp[0]= m2[(int)text3[0]] ^ m3[(int)text3[1]] ^ text3[2] ^ text3[3];
   temp[4]= m2[(int)text3[4]] ^ m3[(int)text3[5]] ^ text3[6] ^ text3[7];
   temp[8]= m2[(int)text3[8]] ^ m3[(int)text3[9]] ^ text3[10] ^ text3[11];
   temp[12]= m2[(int)text3[12]] ^ m3[(int)text3[13]] ^ text3[14] ^ text3[15];

   //2nd column
    temp[1]= text3[0] ^ m2[(int)text3[1]] ^ m3[(int)text3[2]] ^ text3[3];
    temp[5]= text3[4] ^ m2[(int)text3[5]] ^ m3[(int)text3[6]] ^ text3[7];
    temp[9]= text3[8] ^ m2[(int)text3[9]] ^ m3[(int)text3[10]] ^ text3[11];
    temp[13]= text3[12] ^ m2[(int)text3[13]] ^ m3[(int)text3[14]] ^ text3[15];

    //3rd column
    temp[2]= text3[0] ^ text3[1] ^ m2[(int)text3[2]] ^ m3[(int)text3[3]];
    temp[6]= text3[4] ^ text3[5] ^ m2[(int)text3[6]] ^ m3[(int)text3[7]];
    temp[10]= text3[8] ^ text3[9] ^ m2[(int)text3[10]] ^ m3[(int)text3[11]];
    temp[14]= text3[12] ^ text3[13] ^ m2[(int)text3[14]] ^ m3[(int)text3[15]];

    //4th column
    temp[3]= m3[(int)text3[0]] ^ text3[1] ^ text3[2] ^ m2[(int)text3[3]];
    temp[7]= m3[(int)text3[4]] ^ text3[5] ^ text3[6] ^ m2[(int)text3[7]];
    temp[11]= m3[(int)text3[8]] ^ text3[9] ^ text3[10] ^ m2[(int)text3[11]];
    temp[15]= m3[(int)text3[12]] ^ text3[13] ^ text3[14] ^ m2[(int)text3[15]];

    for(int l=0;l<16;l++)
    { text3[l]=temp[l];

    }


}



void keyexpansion(unsigned char *key0, int z)
{ unsigned char temp[16];
```

```
//1st column

temp[0]=s[(int)key0[13]]^key0[0]^rcon[z];
temp[1]=s[(int)key0[14]]^key0[1];
temp[2]=s[(int)key0[15]]^key0[2];
temp[3]=s[(int)key0[12]]^key0[3];

//2nd column
temp[4]=temp[0]^key0[4];
temp[5]=temp[1]^key0[5];
temp[6]=temp[2]^key0[6];
temp[7]=temp[3]^key0[7];

//3rd column
temp[8]=temp[4]^key0[8];
temp[9]=temp[5]^key0[9];
temp[10]=temp[6]^key0[10];
temp[11]=temp[7]^key0[11];

//4th column
temp[12]=temp[8]^key0[12];
temp[13]=temp[9]^key0[13];
temp[14]=temp[10]^key0[14];
temp[15]=temp[11]^key0[15];

for(int a=0;a<16;a++)
{
  key0[a]=temp[a];
}

}



void aes(unsigned char *cipher)
{

    roundkey(cipher,key);              //1st round
```

```
  for(int y=0;y<9;y++)              //9 rounds
   {
      subbytes(cipher);
      shiftrow(cipher);
      mixcolumns(cipher);
      keyexpansion(key,y);
      roundkey(cipher,key);
   }

   subbytes(cipher);                //final round
   shiftrow(cipher);
   keyexpansion(key,9);
   roundkey(cipher,key);

}
```

HASHING FILE

```
#include<string.h>
#include<inttypes.h>
#include<stdio.h>

#define rightrotate(n,m) ((n>>m) | (n<<(32-m)))
uint8_t arr[64];
uint32_t value[64];


void newbytes();
void bitchanger();
void compression();


uint32_t h[8];


//this function converts four 8bits elements to one 32bit element
void bitchanger()
{
  for(int j=0,l=0;j<16;j++,l+=3)
  {
    value[j]=(arr[j+l]<<24) | (arr[j+l+1]<<16) | (arr[j+l+2]<<8) | (arr[j+l+3]) ;
  }

}


//this function generates the remaining 48words
void newbytes()
{
  for(int i=16;i<64;i++)
  {
    uint32_t s0=rightrotate(value[i-15],7) ^ rightrotate(value[i-15],18) ^
(value[i-15]>>3);
    uint32_t s1= rightrotate(value[i-2],17) ^ rightrotate(value[i-2],19) ^
(value[i-2]>>10);
    value[i]=value[i-16]+s0+value[i-7]+s1;
```

```
  }

}


//this is the main function which generates the hash
void compression(uint32_t *h1)
{uint32_t a[8]={h1[0],h1[1],h1[2],h1[3],h1[4],h1[5],h1[6],h1[7]};


 for(int j=0;j<64;j++)
 {
   uint32_t s0=rightrotate(a[4],6) ^ rightrotate(a[4],11) ^ rightrotate(a[4],25);
   uint32_t ch=(a[4] & a[5]) ^ (~a[4] & a[6]);
   uint32_t temp1=a[7]+s0+ch+k[j]+value[j];
   uint32_t s1=rightrotate(a[0],2) ^ rightrotate(a[0],13) ^ rightrotate(a[0],22);
   uint32_t maj=(a[0] & a[1]) ^ (a[0] & a[2]) ^ (a[1] & a[2]);
   uint32_t temp2=maj+s1;

   a[7]=a[6];
   a[6]=a[5];
   a[5]=a[4];
   a[4]=a[3]+temp1;
   a[3]=a[2];
   a[2]=a[1];
   a[1]=a[0];
   a[0]=temp1+temp2;

 }

 h[0]=h1[0]+a[0];
 h[1]=h1[1]+a[1];
 h[2]=h1[2]+a[2];
 h[3]=h1[3]+a[3];
 h[4]=h1[4]+a[4];
 h[5]=h1[5]+a[5];
 h[6]=h1[6]+a[6];
 h[7]=h1[7]+a[7];
```

```
}

void hashing(unsigned char *cipher)
{

//First 32 bits of square root of first 8 prime numbers
//formula= hex(2^32(squareroot(number)))
uint32_t
h1[8]={0x6a09e667,0xbb67ae85,0x3c6ef372,0xa54ff53a,0x510e527f,0x9b
05688c,0x1f83d9ab,0x5be0cd19};

   for(int i=0;i<64;i++)
    {   if(i<16)
      {arr[i]=(int) cipher[i];}

       if(i>=17 && i<63)
       {arr[i]=0;}

       if(i==63 || i==16)
       {arr[i]=128;}

    }

    bitchanger();
    newbytes();
    compression(h1);


}
```

SALTING FILE

```
#include<time.h>
#include<inttypes.h>
#include<stdio.h>


char arr1[32];
int binary[32];
int test,value1;
int lower=35,upper=126;
FILE *fp;


void salt()
{
    rewind(fp);
   for(int i=0;i<32;i++)

  {
     arr1[i]= (rand() % (lower-upper+1)) + lower;
     value1=(int) arr1[i];

      int result=0,a=1,rem;
      while(value1>0)
       {
          rem=value1%2;
          result=result+(a*rem);
          value1=value1/2;
          a=a*10;

       }

       binary[i]=result;

  }


    while(!feof(fp))
  {  int count=0;
```

```
    for(int j=0;j<32;j++)
    {
      fscanf(fp,"%d",&test);


     if(test==binary[j])
        { count++; }

    }

    if(count>29)
    {salt();
      break;
    }

  }


}

void salting(uint32_t *h)
{int number;

fp=fopen("salt.bin","ab+");

FILE *hfp=fopen("password.txt","a");

  if(hfp==NULL)
  {printf("\n error..file cannot be open");}


  if(fp==NULL)
  {printf("\n error..file cannot be open");}

 srand(time(0));
  salt();

for(int k=0;k<32;k++)
  { fprintf(fp," %d",binary[k]); }
```

```
  fseek(hfp,0,SEEK_END);

  if(ftell(hfp)!=0)
  {fprintf(hfp,"\n");}

 printf("\t");
  for(int q=0;q<8;q++)
  {fprintf(hfp,"%x",h[q]);
   printf("%x",h[q]);}

  for(int f=0;f<32;f++)
  {number=(int)arr1[f];
  fprintf(hfp,"%x",number);
   printf("%x",number);}


  fclose(hfp);

  fclose(fp);

}
```

LOGIN FILE

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>




char u;


int decimal(int data)
{  int ldigit=0,base=1,result2=0;

  while(data)
  {
     ldigit=data%10;
     data=data/10;
       result2+=ldigit*base;
       base=base*2;
  }


  return result2;
}




void login()
{
unsigned char passw[16];
unsigned char test[20],name1[20];;
 int flag=0,value2,count=0;
 char testing1[130],testing2[130],c;


printf("Enter the Name: ");
scanf("%s",&name1);
```

```
printf("\nEnter the password: ");
scanf("%s",&passw);



FILE *fp1=fopen("name.txt","r");
if(fp1==NULL)
  {printf("\n error..file cannot be open");}


while(!feof(fp1))
{ memset(test, '\0',sizeof(test));

  fscanf(fp1,"%s",&test);

   count+=1;

   if(strcasecmp(test,name1)==0)
   {flag=1;
    break;}


}

fclose(fp1);




if(flag!=1)
{printf("\nincorrect username or password..");
scanf("%c",&u);
scanf("%c",&u);
return;}


aes(passw);
hashing(passw);
```

```
FILE *fp2=fopen("salt.bin","rb");
if(fp2==NULL)
  {printf("\n error..file cannot be open");}

FILE *fp3=fopen("temp.txt","w+");
if(fp3==NULL)
  {printf("\n error..file cannot be open");}


  for(int m=0;m<8;m++)
  {fprintf(fp3,"%x",h[m]); }

  for(int i=0;i<(count-1)*32;i++)
  {fscanf(fp2,"%d",&value2);}

  for(int j=0;j<32;j++)
  {fscanf(fp2,"%d",&value2);
   fprintf(fp3,"%x",decimal(value2));
  }
 fclose(fp2);

FILE *fp4=fopen("password.txt","r");
if(fp4==NULL)
  {printf("\n error..file cannot be open");}

for(int h=0;h<count;h++)
{fscanf(fp4,"%s",&testing1);}


rewind(fp3);
fscanf(fp3,"%s",&testing2);


if(strcmp(testing1,testing2)==0)
{printf("\nlogin successful...");
scanf("%c",&u);

scanf("%c",&u);
}
```

```
else
{printf("\n incorrect username or password...");
scanf("%c",&u);
scanf("%c",&u);
}


fclose(fp3);
fclose(fp4);

}
```

References

[1] https://www.thesslstore.com/blog/difference-encryption-hashing-salting/

[2] https://www.solarwindsmsp.com/blog/sha-256-encryption#:~:text=SHA%2D256%20is%20a%20patented,that%20is%20256%20bits%20long.&text=In%20hashing%2C%20by%20contrast%2C%20data,string%20through%20SHA%2D256%20hashing

[3] https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard#:~:text=The%20Advanced%20Encryption%20Standard%20(AES,cybersecurity%20and%20electronic%20data%20protection.

[4] https://en.wikipedia.org/wiki/Salt_(cryptography)#:~:text=In%20cryptography%2C%20a%20salt%20is,to%20safeguard%20passwords%20in%20storage.&text=Salts%20defend%20against%20a%20pre,hash%20attack%2C%20e.g.%20rainbow%20tables.

* Whole Documents should not be more than 7 pages excluding Front Page
* The Front should contain Project Name, Partial Submission for Minor, Students name, Enrollment No, SAP Id no, Mentor Name

Approved By

(Name & Sign) (Name & Sign)

Project Guide Head of Department