DIGILOCKER

JAIDITYA BEERAKA – 21CSB0B23
JASWANTH YERRAMSETTI – 21CSB0B68

INTRODUCTION

In today's digital age, ensuring the integrity, authentication, and confidentiality of sensitive documents has become paramount. The Digilocker project addresses the pressing need for a secure and efficient digital document storage and retrieval system. This system allows users to securely sign in or sign up, gaining access to their personal repository of documents, including vital records like Aadhar.

The significance of this project lies in safeguarding the authenticity and privacy of user data. With increasing reliance on digital documents, the risk of unauthorized access and tampering has escalated. Existing solutions have made strides in digital storage but often fall short in ensuring the holistic security of documents.

Our project's key contributions revolve around implementing robust encryption and decryption mechanisms using RSA and AES algorithms. These algorithms not only guarantee data confidentiality but also provide strong authentication measures, ensuring that only authorized users can access their documents.

By prioritizing integrity, authentication, and confidentiality, the Digilocker project aims to set a new standard in digital document management, empowering users with a trusted and secure platform for their sensitive information.

OBJECTIVES

The objectives of our project can be summarized as follows:

- Implementing a secure sign-in/sign-up process for users.
- Facilitating secure document retrieval and storage with encryption and decryption using RSA and AES.

IMPLEMENTATION AND RESULT ANALYSIS

- Each citizen have their unique **citizen pin** provided by government. If client enters his citizen pin correctly, he can get his documents from respective servers to digiLocker server

docServer.cpp

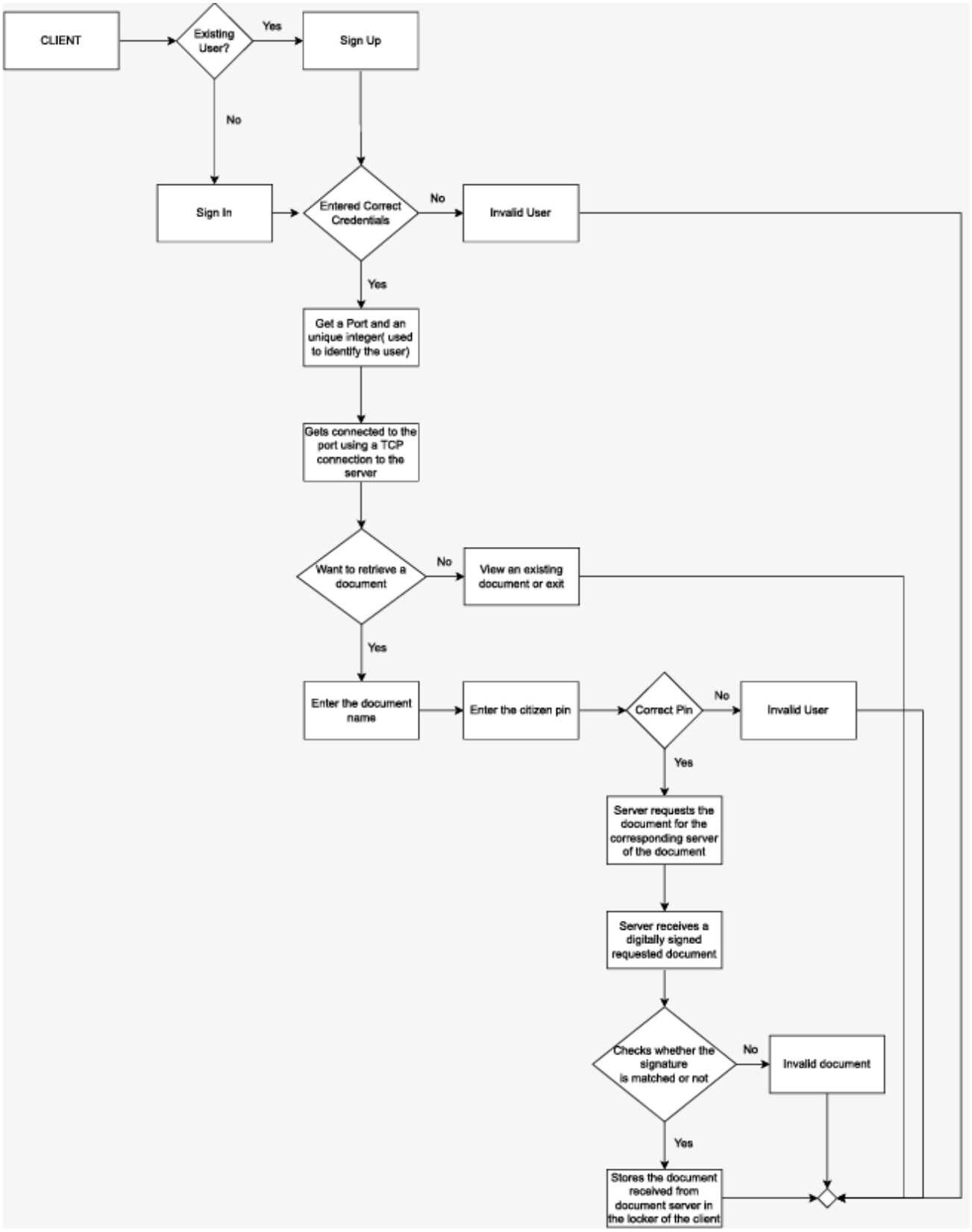
```
RSA* rsa = generateRSAKeyPair(2048);
const BIGNUM *n = NULL, *e = NULL, *d = NULL;
RSA_get0_key(rsa, &n, &e, &d);

char* pubN = printHex(n, "Public key (n)");
char* pubE = printHex(e, "Public key (e)");
```

Generate Public Key of respective services

```
send(nsfd, pubN, strlen(pubN), 0);
sleep(1);
send(nsfd, pubE, strlen(pubE), 0);
```

Send public key of service to digiLocker server

FLOW CHART


CLIENT
Existing User?
Yes
Sign Up
No
Sign In
Entered Correct Credentials
No
Invalid User
Yes
Get a Port and an unique integer(used to identify the user)
Gets connected to the port using a TCP connection to the server
Want to retrieve a document
No
View an existing document or exit
Yes
Enter the document name
Enter the citizen pin
Correct Pin
No
Invalid User
Yes
Server requests the document for the corresponding server of the document
Server receives a digitally signed requested document
Checks whether the signature is matched or not
No
Invalid document
Yes
Stores the document received from document server in the locker of the client

```
char encryptedUsername[2048] = {0}; // Adjust size as needed
int len = recv(nsfd, encryptedUsername, sizeof(encryptedUsername), 0);
encryptedUsername[len] = '\0';

sleep(1);

cout << "Received encrypted Username: " << encryptedUsername <<" "<<len<< endl;

unsigned char decryptedUsername[2048] = {0};

int usernameLen = len; // Use the actual length received
int decryptedUsernameLen = rsaDecrypt((const unsigned char*)encryptedUsername, usernameLen, decryptedUsername, rsa);

if (decryptedUsernameLen == -1) {
    cerr << "Decryption failed" << endl;
    RSA_free(rsa);
    close(nsfd);

}
sleep(1);
```

Receiving encrypted username and decrypting it

```
int pinlen = recv(nsfd, encryptedUserPin, sizeof(encryptedUserPin), 0);
int decryptedUserPinLen = rsaDecrypt((const unsigned char*)encryptedUserPin, pinlen, decryptedUserPin, rsa);

if (decryptedUserPinLen == -1) {
    cerr << "Decryption failed" << endl;
    RSA_free(rsa);
    close(nsfd);

}
sleep(1);
```

Receiving encrypted citizen pin and decrypting it

```
if(strcmp(or_pin,(const char*)decryptedUserPin)==0)
{
     // Corrected string concatenation
    string folder_loc = name + "/" + string((const char*)decryptedUsername) + ".txt";

    cout << folder_loc << endl;

    // Corrected folder_loc usage and added .c_str()
    int rfd = open(folder_loc.c_str(), O_RDONLY);

    char details[1000];

    int n=read(rfd,details,sizeof(details));
    details[n-1]='\0';

    string hash = sha256Hash((string)details);

    cout<<hash<<endl;

    string h1 = rsaPrivateEncrypt((const unsigned char*)hash.c_str(),hash.length(),rsa);

    string toSend=string(details)+","+h1;
    cout<<"to send is "<<toSend<<endl;

    send(nsfd, toSend.c_str(),toSend.length(),0);

}
else
{
    string details="wrong pin";

    send(nsfd, details.c_str(),details.length(),0);
}
```

If correct citizen pin, sending the document with digital signature, Else reporting error

Server.cpp

- Each user is assigned a port and unique ID for every session. User should communicate with that port.

```
char pdf[100],details[1000];
int n=recv(ncsfd,pdf,sizeof(pdf),0);
pdf[n]='\0';
cout<<pdf<<endl;

string filePath=cd.username+"/"+string(pdf)+".txt";
```

Receiving document request from user

```
// Receive and decrypt port
unsigned char encryptedPin[AES_BLOCK_SIZE];
int n = recv(ncsfd, encryptedPin, 10000, 0);
cout<<"Size recieved is : "<<n<<endl;
if (n < 0) {
    cerr << "Error receiving decryptedPort\n";
    return NULL;
}
encryptedPin[n] = '\0'; // Ensure null-termination

cout<<"EncryptedPin is : "<<encryptedPin<<endl;
sleep(1);
cout<<"Decryption of Pin from client is going on ... "<<endl;
sleep(1);
string pin;
pin = aesDecrypt(decryptedKey, (unsigned char*)encryptedPin);
```

Receiving and encrypting citizen pin entered by user

```
char pin_flag[2100];
n=recv(dsfd, pin_flag,sizeof(pin_flag),0);
cout<<pin_flag<<endl;
pin_flag[n]='\0';
```

Receiving document from docServer

```
string hash = sha256Hash(d1);
string h1 = rsaPublicDecrypt((const unsigned char*)eh1.c_str(),eh1.length(),rsaDoc);
```

Verifying digital signature

```
string fileCreate="touch "+filePath;
system(fileCreate.c_str());
const char* buff = d1.c_str();

cout<<strlen(buff)<<" "<<buff<<endl;
int fd=open(filePath.c_str(),O_WRONLY);
write(fd,buff,d1.length());
```

Saving the file

Client.cpp

```
string username, password;
cout << "Enter the username" << endl;
cin >> username;
cout << "Enter the password" << endl;
cin >> password;
```

User enters credentials

```
string pdfName;
cout<<"Enter the name of the pdf you want to store"<<endl;
cin>>pdfName;
sleep(1);
send(csfd,pdfName.c_str(),pdfName.length(),0);
```

User enters required document

```
cout<<"enter citizen pin"<<endl;
string pin;
cin>>pin;

unsigned char ciphertextPin[AES_BLOCK_SIZE];

aesEncrypt(key, (unsigned char*)pin.c_str(), ciphertextPin);

sleep(1);

cout<<"Encrypted Pin is : "<<ciphertextPin<<endl;

send(csfd, ciphertextPin,strlen((char*)ciphertextPin),0);
```

If correct citizen pin, document is loaded into locker

Implementation Overview:

User Authentication: Users could sign in or sign up to access their digital documents securely.

Document Retrieval: Upon authentication, users could request specific documents like Aadhar.

Server Communication: The client-server architecture facilitated communication, with the server interacting with a main server dedicated to each document.

Encryption/Decryption: RSA and AES algorithms were implemented for securing data during transmission and storage.

Unique ID Handling: Users received a unique ID from the server to access their documents securely.

Tools Used:

Programming Languages : c++

Libraries : openSSL

OUTPUT:

Client signs up and enters credentials

```
Type 1 for 'SignIn' Else Type 2 for 'SignUp'
2
Enter the username
jaswanth
Enter the password
12345
```

Client's password hash is stored at server side

```
Stored in map: 5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5
User successfully registered
```

Server send port and identity to client

```
Decrypted key: 7506288f5b50ce719bce05d5adbdce31c0c460a08474cd40f7696523a47a026b
port assigned : 8000
Encrypted Port is : /B◆l        ◆◆◆◆vB◆o◆C
Identity assigned : 237673457
```

User enters required document name and citizen pin

```
Enter the name of the pdf you want to store
aadhar
msg is : no
enter citizen pin
987654321010
```

File not exists so docServer connection is established

```
file not exists
D000616B43EDE5E71B6A08430367ECFB135C7CB73544ED50BC638E4407F4A8023A5F0242DF2E3A562A7B9557B0C1
A81DED172825BA4CF7C17421B1ECAEE9CE34381549B28577638470693FC7D9055F9F0FF6A4F0BFBA13073FFF0B76
CB4B4175E2FEB1DA6E5C9140BB1A67BBCD15386E101902E093F9D105A73438305EB33A75C5FE82BF734C04C14005
029F0752093BCED742364727EA88CBF90F6F6D00F9FD5A7B4103C2B739491DC2E459200AC656A582BCC42D665A9C
928C4DE0308A4AD4D3861285D5C1FBA849260FB3D21720D8143FB1B345C745A16FEECD4CDEA86AA5FDBFBAFD5756
281208D7F291B3A3E71FE0C47967990BAD1328A15985A78080A7

010001

Encrypting username and sending to docServer...
```

docServer sends back details

```
Decrypted Username: 6a617377616e7468
Decrypted Username (String): jaswanth
Received encrypted UserPin: q1"6yt◆◆1◆8U◆Gr◆'◆_Gr◆◆*◆◆;I◆"◆p4,o◆C◆J 256
Decrypted UserPin: 393837363534333231303130
Decrypted UserPin (String): 987654321010
12
aadhar/jaswanth.txt
5642aafe7d23afc5fdf52e6ae23ca0cf7aff16627e224602f5b955f4c0b25b19
```

User tries to signin

```
Type 1 for 'SignIn' Else Type 2 for 'SignUp'
1
Enter the username
jaswanth
Enter the password
12345
```

If requested file already exists

```
Enter the name of the pdf you want to store
aadhar
msg is : yes
```

If incorrect password is entered

```
Decrypted password: 03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4
  File: .
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 803h/2051d      Inode: 729193      Links: 4
Access: (0775/drwxrwxr-x)  Uid: ( 1000/jaidityabeeraka)   Gid: ( 1000/jaidityabeeraka)
Access: 2024-04-23 22:49:36.990701130 +0530
Modify: 2024-04-23 22:54:04.552415237 +0530
Change: 2024-04-23 22:54:04.552415237 +0530
 Birth: 2024-03-14 14:56:47.480703282 +0530
Invalid User
```

If incorrect citizen pin is entered

```
Decryption of Pin from client is going on ...
Decrypted pin: 98765432
Encrypting userPin and sending to docServer...
Encrypted userPin is ...
256   G}u<A◆8K~◆uyBV
wrong pin
wrong pin
```

CONCLUSION

In conclusion, the Digilocker system presented in this project embodies a significant advancement in secure digital document management. By integrating robust encryption techniques such as RSA and AES, we have ensured the utmost integrity, authentication, and confidentiality of user data. The system's ability to securely store and retrieve documents, coupled with the seamless user experience of signing in, requesting documents, and accessing them through unique identifiers, marks a substantial step forward in modern document management solutions