Assignment-4

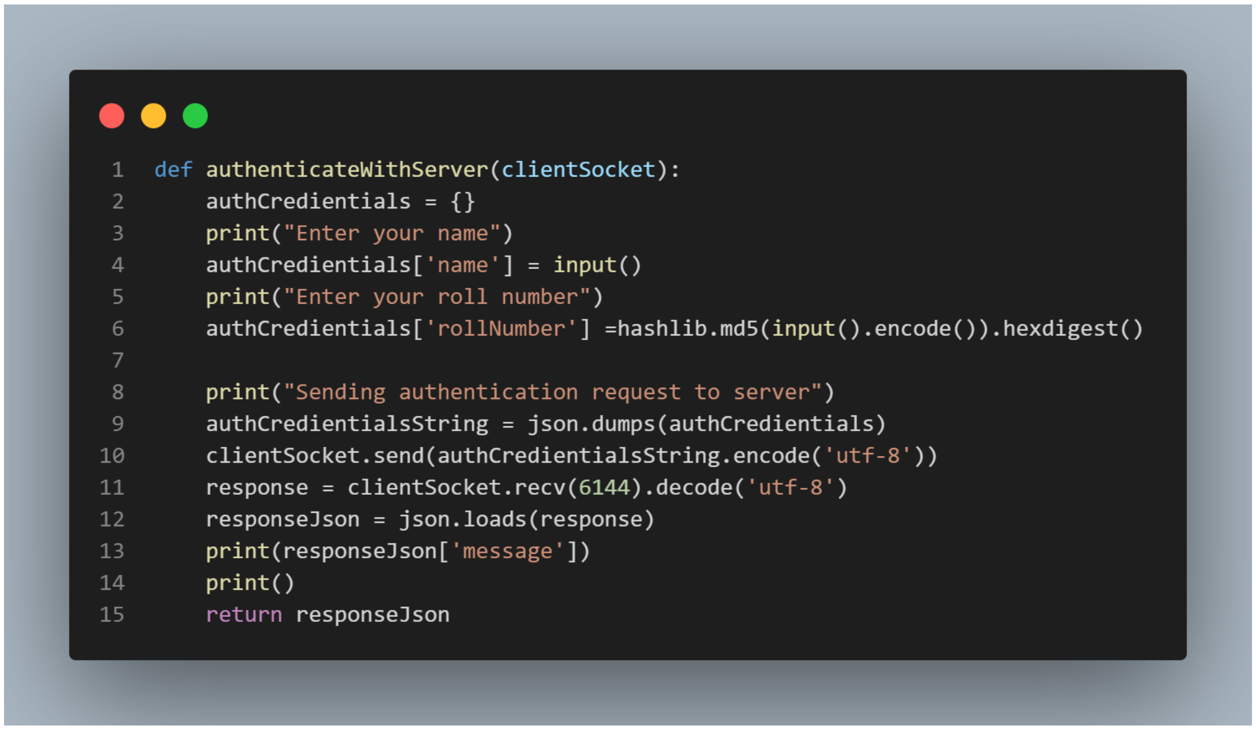
Keshav Gambhir 2019249
Tanmay Rajore 2019118

Introduction
In this assignment the objective was to implement a server which serves the client in a secure manner. The client requests for certificate and marksheet by providing the suitable name and roll number which serves as a credential for the server. Once the server authenticates the client It then encrypts the certificate and marksheet and sends it to the client along with HMAC signed by the private key of director and registrar

Authentication to server
To authenticate the client to the server, the client sends the name and roll number in an encrypted manner to the server which first decodes and then authenticates the client. The name and the roll number serves as username and password for the server.

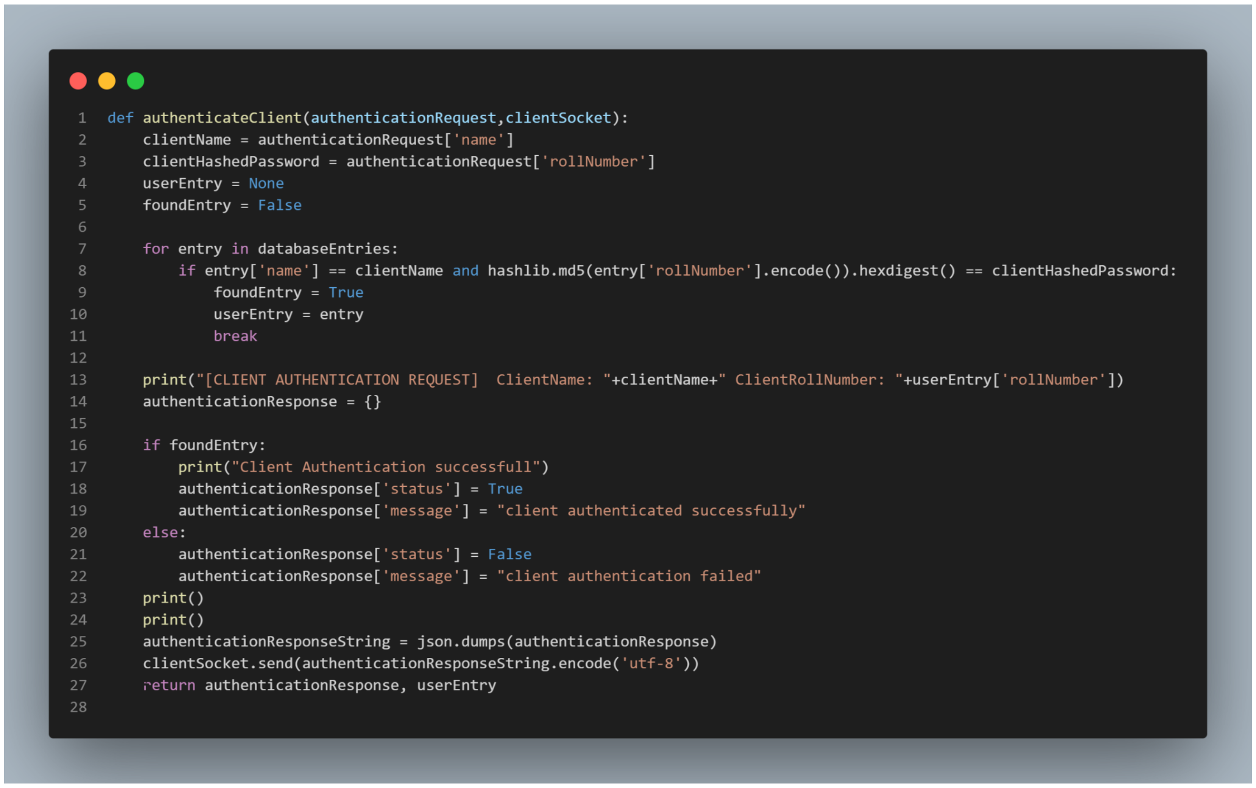
Design Choice justification: Unlike Kerberos based authentication this is a mutual authentication between client and server in an encrypted manner. If we are using TLS handshake between the server and client this method would become more secure and would ensure that there is no tampering with the message

Client Code:



```python
def authenticateWithServer(clientSocket):
    authCredientials = {}
    print("Enter your name")
    authCredientials['name'] = input()
    print("Enter your roll number")
    authCredientials['rollNumber'] =hashlib.md5(input().encode()).hexdigest()

    print("Sending authentication request to server")
    authCredientialsString = json.dumps(authCredientials)
    clientSocket.send(authCredientialsString.encode('utf-8'))
    response = clientSocket.recv(6144).decode('utf-8')
    responseJson = json.loads(response)
    print(responseJson['message'])
    print()
    return responseJson
```

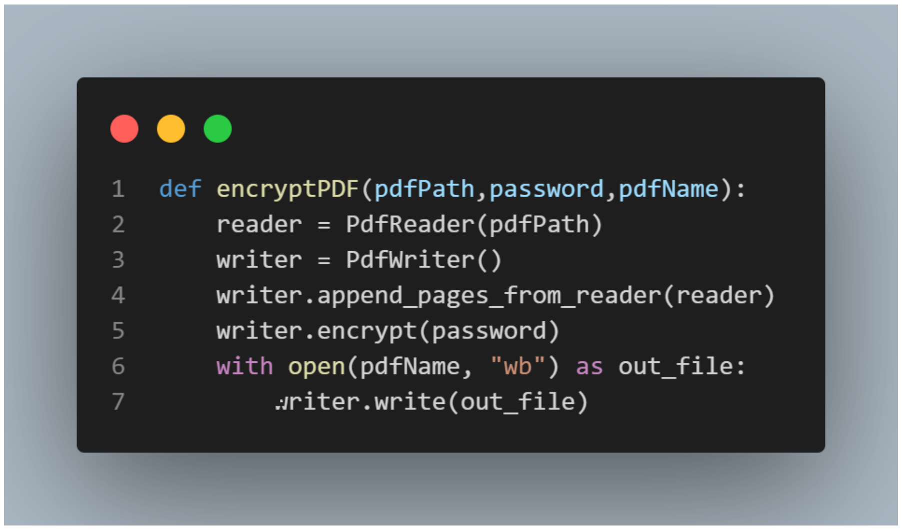
Server Code:



```python
def authenticateClient(authenticationRequest,clientSocket):
    clientName = authenticationRequest['name']
    clientHashedPassword = authenticationRequest['rollNumber']
    userEntry = None
    foundEntry = False

    for entry in databaseEntries:
        if entry['name'] == clientName and hashlib.md5(entry['rollNumber'].encode()).hexdigest() == clientHashedPassword:
            foundEntry = True
            userEntry = entry
            break

    print("[CLIENT AUTHENTICATION REQUEST]  ClientName: "+clientName+" ClientRollNumber: "+userEntry['rollNumber'])
    authenticationResponse = {}

    if foundEntry:
        print("Client Authentication successfull")
        authenticationResponse['status'] = True
        authenticationResponse['message'] = "client authenticated successfully"
    else:
        authenticationResponse['status'] = False
        authenticationResponse['message'] = "client authentication failed"
    print()
    print()
    authenticationResponseString = json.dumps(authenticationResponse)
    clientSocket.send(authenticationResponseString.encode('utf-8'))
    return authenticationResponse, userEntry

```

Encrypting the Data with Password

The pdf which is returned by the server is encrypted by password which is the DOB of the student. This is similar to the way Aadhar Cards are secured in PDF formats. The DOB password can be changed with any other password which is provided by the client. This can also be encrypted by the public key of the client and sent to the client which then can only be decrypted by the client's private key. The code snippet for the same is shown below



```python
def encryptPDF(pdfPath,password,pdfName):
    reader = PdfReader(pdfPath)
    writer = PdfWriter()
    writer.append_pages_from_reader(reader)
    writer.encrypt(password)
    with open(pdfName, "wb") as out_file:
        writer.write(out_file)
```

Providing Message Authentication

To provide message authentication a HMAC which is signed by first the director and then by the registrar is sent along with the message. This is similar to certificate chaining and the message integrity is checked only after both the unsigned HMACs verify the original message HMAC.

Getting Timestamp Secruly

Timestamps are obtained using the NTP server. Since it is a global server the chance of it being corrupt is less. Moreover the request HTTPs can be used to obtain the timestamp

Sharing with others

To share the certificate and marksheet with others, the signed HMAC of the director, registrar and client is sent along with the message. Hence the message integrity is only verified once all the 3 unsigned HMACs matches the original message HMAC