Assignment 4

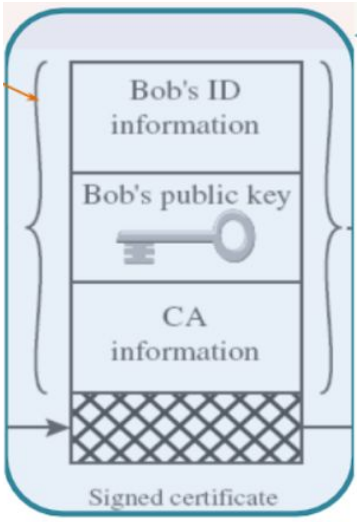
Digitally Signed Degree Certificates

By: Dheeraj - 2017044, Sehaj Singh - 2017099

Details of Implementation

- First of all, the student sends his/her credentials to the server where the server verifies if the credentials of the student are valid or not, that is, if the data of this student matches in the database or not.
- If the credentials don't match, an error is sent back to the student and no certificate is generated.
- If the credentials are a match, the certificate is generated and the certificate of degree completion consists of all the student's data, the correct GMT time when the certificate was being processed and the signatures of the Director and Registrar of the college.
- Now, the generation of signatures consists of 2 steps. Firstly, the certificate data is taken and encrypted using the Director's/ Registrar's private key and then the result is passed through the SHA-256 algorithm to generate the final Director/Registrar signature.
- The above step is done separately for both, the Director and Registrar and the signatures are appended in the certificate.
- Another crucial step is the generation of the digital watermark. The watermark is used essentially for the tracing back of the certificate's origin. The student details are taken together with the GMT time of issuing the certificate and then is passing through a hashing function (SHA-256) to create a digital watermark.

Certificate Diagram



How to Run?

- Firstly, launch a terminal and run server.py.
- Next, launch another terminal and run verifier.py
- Finally, launch the third terminal and run student.py
- You will be prompted to enter a student's credentials. Enter the credentials the code asks for and you will end up with a successful run of our code.

Final Results

- The results obtained were as expected. The student's certificate was verified successfully when there was no tampering made. If there was tampering made, the verifier threw an error that the certificate is not verified. Please find the screenshots of a sample run at the end of this report.

Some Assumptions

- We have assumed that every entity has public keys of all other entities. In summary, we have assumed that the key distribution using a public key distribution authority has already been performed.
- We have also assumed that the Director and Registrar have provided their private keys to the college server so that the certificates can be suitably signed at the server to ensure the authenticity of the certificates.

Assignment Questions:

1. How and where do you get the correct GMT date and time? Is the source reliable and the GMT date and time obtained in a secure manner?
   Answer: We have utilized a python function named 'time.gmtime()' which is present in the 'time' library of python. It provides the correct GMT time as well as in a secure manner.

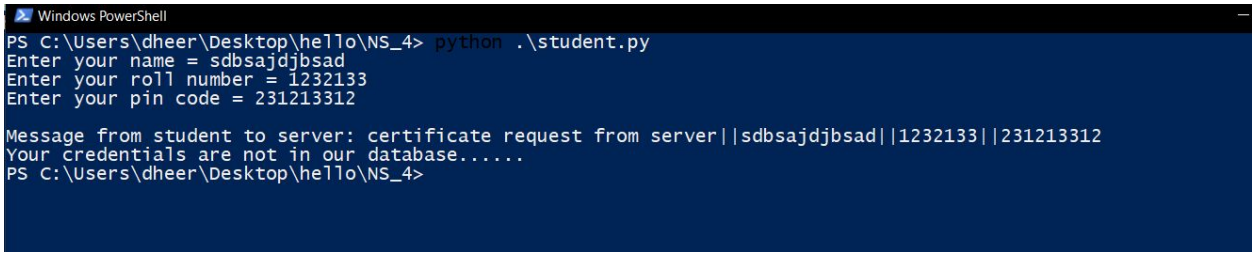
2. How do you get the document to be digitally signed by two persons (say the Registrar and the Director?
   Answer: We have Director's and Registrar's private keys present at the college server and therefore, the certificate generated by the server is digitally signed by Registrar's and Director's private key.

3. How do you ensure that only the graduate is able to download it?
   Answer: We have asked for the Name (which is registered on the Server), Roll Number and the PinCode associated with the student's permanent address so as to *authenticate the student*. Moreover, the certificate sent to the student is suitably encrypted using the student's public key which can be only decrypted by the student (using his private key).

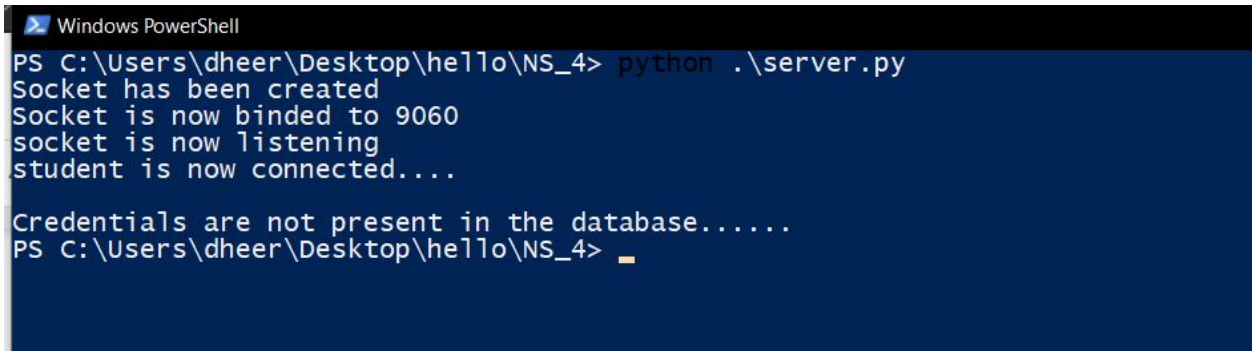
4. Should the graduate decide to share the document with others, how can one trace the origin of the document (could watermarks be useful?)?
   Answer: A digital watermark is used which is essentially an SHA hash of the string formed by the concatenation of student's information and the GMT time (time at which the request was made). This hash can be used to trace back the origin of the document (can be verified as the GMT time and the student's information is present in the certificate).

5. Do we need to have access to public-keys, and if so how?
   Answer: Yes. For the matter of confidentiality, the various entities must have public keys of each other so as to suitably encrypt the messages sent over the channel in order to avoid eavesdropping and maintain the integrity of the message. For authentication, the public keys of Director and Registrar are used by the verifier to authenticate the certificate sent by the student. The public keys are distributed using the Public Key Distribution Authority (PKDA) that we implemented in the previous project. For this project, we have assumed that the public keys of the 3 entities are present with each of them.

Sample Run 1 (Wrong credentials)

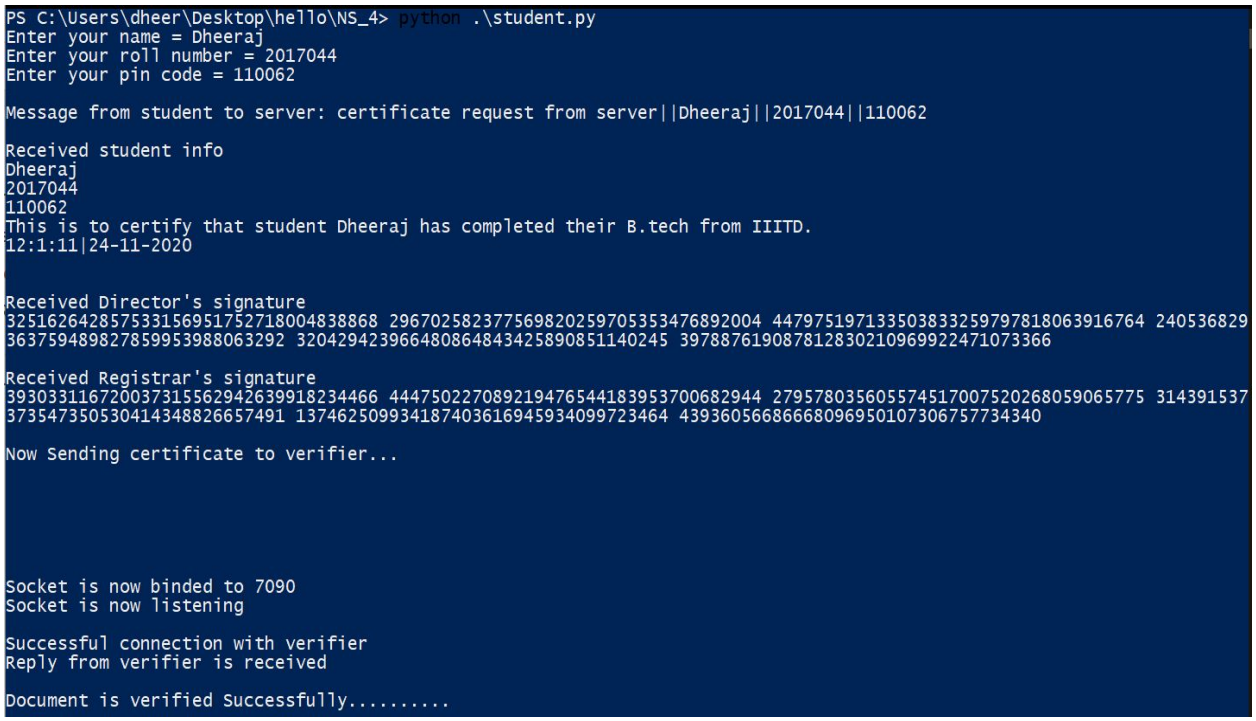


```
Windows PowerShell
PS C:\Users\dheer\Desktop\hello\NS_4> python .\student.py
Enter your name = sdbsajdjbsad
Enter your roll number = 1232133
Enter your pin code = 231213312

Message from student to server: certificate request from server||sdbsajdjbsad||1232133||231213312
Your credentials are not in our database......
PS C:\Users\dheer\Desktop\hello\NS_4>
```



```
Windows PowerShell
PS C:\Users\dheer\Desktop\hello\NS_4> python .\server.py
Socket has been created
Socket is now binded to 9060
socket is now listening
student is now connected....

Credentials are not present in the database......
PS C:\Users\dheer\Desktop\hello\NS_4>
```

Sample Run 2 (Correct credentials)



```
PS C:\Users\dheer\Desktop\hello\NS_4> python .\student.py
Enter your name = Dheeraj
Enter your roll number = 2017044
Enter your pin code = 110062

Message from student to server: certificate request from server||Dheeraj||2017044||110062

Received student info
Dheeraj
2017044
110062
This is to certify that student Dheeraj has completed their B.tech from IIITD.
12:1:11|24-11-2020


Received Director's signature
3251626428575331569517527180048388 29670258237756982025970535347689200 4479751971335038332597978180639167 64 240536829
3637594898278599539880632 92 320429423966480864843425890851140245 39788761908781283021096992247107336 6

Received Registrar's signature
3930331167200373155629426399182344 66 4447502270892194765441839537006829 44 279578035605574517007520268059065775 314391537
3735473505304143488266574 91 13746250993418740361694593409972346 4 439360566866680969501073067577343 40

Now Sending certificate to verifier...




Socket is now binded to 7090
Socket is now listening

Successful connection with verifier
Reply from verifier is received

Document is verified Successfully.....
```

```
PS C:\Users\dheer\Desktop\hello\NS_4> python .\server.py
Socket has been created
Socket is now binded to 9060
socket is now listening
student is now connected....

student info with gmt being sent from server to student:  Dheeraj||2017044||110062||This is to certify that student Dhee
raj has completed their B.tech from IIITD.||12:1:11|24-11-2020


watermark being sent from server to student:  0e5662ea9bc907ab278901e9802ec05d7e883a01f5eb07b48e12bff4a92bf292


director signature being sent from server to student:  325162642857533156951752718004838868 296702582377569820259705353
76892004 447975197133503833259797818063916764 240536829363759489827859953988063292 320429423966480864843425890851140245
397887619087812830210969922471073366

ragistrar signature being sent from server to student:  393033116720037315562942639918234466 444750227089219476544183953
700682944 279578035605574517007520268059065775 314391537373547350530414348826657491 137462509934187403616945934099723464
 439360566866680969501073067577343 40
```

```
Windows PowerShell
PS C:\Users\dheer\Desktop\hello\NS_4> python .\verifier.py
Socket is now listening

Successful connection with student

student info received from student :  ['Dheeraj', '2017044', '110062', 'This is to certify that student Dheeraj has comp
leted their B.tech from IIITD.', '12:1:11|24-11-2020']


watermark received from student :  0e5662ea9bc907ab278901e9802ec05d7e883a01f5eb07b48e12bff4a92bf292


Received Director's signature :  325162642857533156951752718004838868 296702582377569820259705353476892004 4479751971335
038332597978180639167 64 240536829363759489827859953988063292 320429423966480864843425890851140245 397887619087812830210969922471073366


Received Registrar's signature :  393033116720037315562942639918234466 444750227089219476544183953700682944 279578035605
574517007520268059065775 314391537373547350530414348826657491 137462509934187403616945934099723464 439360566866809695010730675773434 0

generating the hash value for the student certificate info to verify the signatures
hash message at verifier side 0e5662ea9bc907ab278901e9802ec05d7e883a01f5eb07b48e12bff4a92bf292
```