ASSIGNMENT - 4

NSC

We build the web server using Flask in which client provides their name, Roll Number, D.O.B and password and if user is registered in out database he will be provided the grade card and degree Certificate. And before providing degree certificate and grade card it will be signed by Director and Registrar and also verified in the backend using public keys. If verification results in success then user can download both the documents.

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?

For this part we are using NTP in which UDP socket is created that fetched current date and time in seconds from a secure server in our case which is “time.google.com” and then it gets converted into human readable format.

```
NTP_SERVER = "time.google.com"
```

```
#-------------------1--------------------
def get_ntp_time():
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    data = b'\x1b' + 47 * b'\0'
    client.sendto(data, (NTP_SERVER, 123))
    data, address = client.recvfrom(1024)
    if data:
        response = struct.unpack('!12I', data)[10]
        response -= 2208988800
        return time.ctime(response)
```

2. How do you ensure that only the graduate is able to download it (by providing information beyond the roll no, such as date of birth, home pin code, etc.?

For above query we are using database which stores D.O.B, password for authentication.

All the data is stored in a csv file.

3. Should the graduate decide to share the document with others, how can one trace the origin of the document (could watermarks be useful?)?

For this we used watermarks.

```python
# Add watermark to the PDF
watermark_text = f"Issued to {roll_number} on {timestamp_str} by ABC University"
watermark_file = "watermark.pdf"
watermark_canvas = canvas.Canvas(watermark_file, pagesize=letter)
watermark_canvas.setFont('Helvetica-Bold', 15)
watermark_canvas.setFillColorRGB(0.5, 0.5, 0.5, 0.2)
watermark_canvas.rotate(45)
text_width = watermark_canvas.stringWidth(watermark_text)

x = -6.5 * inch
y = 0.5 * inch
while x < 8.5 * inch:
    watermark_canvas.drawString(x, y, watermark_text)
    x += text_width + 20

watermark_canvas.save()
```

4. Do we need to have access to public-keys, and if so how?

Access to public keys is necessary for verifying the digital signature of the document. The public keys can be obtained from a trusted key authority, such as a certificate authority (CA), or through a public key infrastructure (PKI) that is used to manage digital certificates and keys. The web server can retrieve the public keys of the university authorities and use them to verify the digital signature of the document. In our keys we saved the keys in file.

Bonus points if you address this issue: How do you get the document to be digitally signed by two persons (say the Registrar and the Director?

For Bonus part we generated two pair of keys and signed the documents one by i.e. first by director and then by registrar.

```python
# Sign the PDF hash with the private key
signature = private_key_registrar.sign(
    pdf_hash,
    padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),
        salt_length=padding.PSS.MAX_LENGTH
    ),
    hashes.SHA256()
)

director_signature = private_key_director.sign(
    pdf_hash,
    padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),
        salt_length=padding.PSS.MAX_LENGTH
    ),
    hashes.SHA256()
)
```

And when downloading the documents we verified those using the respective public keys

```python
# Verify the signature of Registrar
is_verified_registrar = verify_signature(public_key_registrar, signature_registrar, pdf_hash)
# Verify the signature of Director
is_verified_director = verify_signature(public_key_director, signature_director, pdf_hash)


if is_verified_registrar and is_verified_director:
    print("Document is digitally signed by both Registrar and Director")
    return render_template('download_files.html', degree_name=f"{graduate_name}.pdf", grade_name=f"{graduate_name}_gradecard.pdf")

else:
    print("Document signature is invalid")
    return "Document signature is invalid"
```

```python
# Function to verify a digital signature using a public key
def verify_signature(public_key, signature, document):
    try:
        public_key.verify(
            signature,
            document,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
            hashes.SHA256()
        )
        return True
    except:
        return False
```