Network Security Assignment - 04
Project - Zero

Ketan Mohan Garg (2022248)
Keshav Bindlish (2022246)

Introduction

This project focuses on securely time-stamping a document to prove its existence at a specific moment without revealing its contents. A client application first hashes the document and encrypts the hash using the server's public key. The server then obtains the accurate GMT time from a trusted NTP server, digitally signs the hash along with the timestamp, and returns a timestamp certificate. This ensures confidentiality, integrity, authentication, and non-repudiation, allowing third parties to verify that the document existed at the stated time without any modifications.

CODE WORKFLOW-

1. **Document Input and Hashing (Client Side)**
   The user provides a document file through the client application.
   The client does not send the full document to the server.
   Instead, the client calculates the SHA-256 hash of the document, producing a unique, fixed-size fingerprint representing the document’s content.

2. **Hash Encryption**
   To ensure that even the document's hash is not visible to the server, the client encrypts the hash using the server's public key (RSA encryption with OAEP padding). This guarantees confidentiality — only the server can decrypt it using its private key.

3. **Sending Encrypted Hash to Server**
   The encrypted document hash is sent to the server over HTTP POST.

4. **Server Decryption and GMT Time Retrieval**
   Upon receiving the encrypted hash:
   - The server decrypts it using its private key.
   - The server fetches the accurate GMT time using NTP (Network Time Protocol) from trusted sources like `pool.ntp.org`.
   - This prevents dependency on the client's potentially incorrect local system clock.

5. **Timestamp Creation and Signing**
   The server creates a payload containing:
   - The original document hash
   - The current GMT time

It then digitally signs this payload using its private key. This signature ensures that:

- The document hash and time cannot be tampered with.
- Anyone can verify later that this exact data was signed by the server.

6. **Returning the Timestamp Certificate**
   The server returns a JSON certificate to the client, containing:
   - Document hash
   - GMT time
   - Digital signature

7. **Saving the Timestamp**
   The client saves this certificate alongside the original document, preserving proof that the document existed at the recorded time.

8. **Verification**
   Anyone in the future can verify:
   - The document's hash matches
   - The GMT time is authentic
   - The server's signature is valid using its public key

ANSWERS to GIVEN QUESTIONS

1. How and where do you get the correct GMT date and time?
The server takes the GMT time from an external, trusted NTP server like *pool.ntp.org*, and is not relevant on its local system clock to avoid any manipulation in timestamp.

2. When is the correct GMT date/time obtained?
The GMT time is obtained immediately after getting the encrypted document hash from the client that would be just before generating the timestamp certificate.

3. Is the source reliable? Is the GMT date and time obtained in a secure manner?
Yes, NTP servers like `pool.ntp.org` are the most trusted among all. However, standard NTP communication is not encrypted, so while the time is mostly accurate, the communication itself is not fully secure against sophisticated attacks.

4. How do you ensure privacy, in that the server does not see/keep the original document?

Only the encrypted document hash is sent by the client, not the document itself. Thus, the server can never look up or save the actual document content with itself or somewhere.

5. How do you share the document with third parties in a secure manner with the GMT date/time preserved, and its integrity undisturbed?

The original document is shared along with its timestamp certificate. The server's key grants the facility to verify the document's integrity and timestamp without making any changes in the content. If there is any kind of alteration the certificate will fail.

6. How does one ensure that the user (both the owner and anyone else verifying the date/time) uses the correct "public-key" of the server stamping/signing the "GMT date/time"?

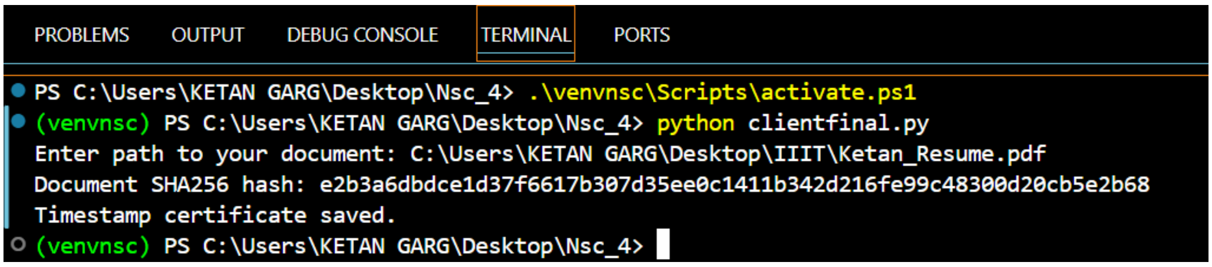
The client directly uses the server's public key, The public key is securely shared or distributed beforehand through trusted means like publishing on some official website like HTTPS . When verifying the timestamp, users use this known public key to verify the server's signature on the document hash and GMT time. Since the public key is before hand given to client and trusted manually, the integrity and authenticity of the GMT timestamp

7. Which of these, viz. confidentiality, authentication, integrity and non-repudiation is/are relevant?

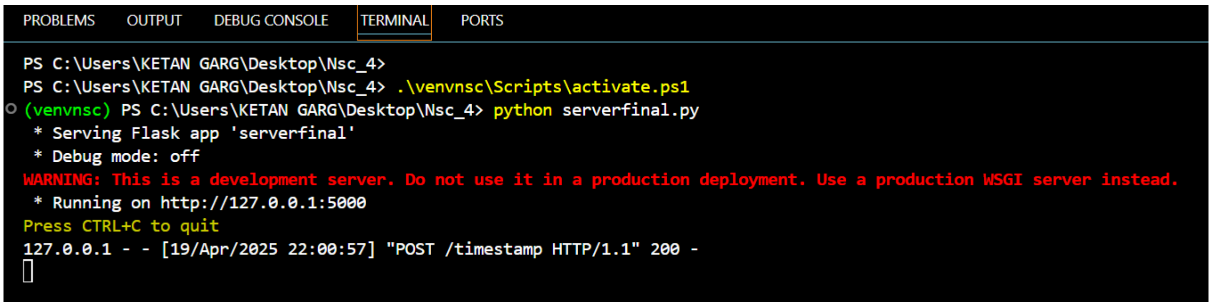
All four are relevant:

- Confidentiality: Only encrypted hash is sent; original document remains private with the client.
- Authentication: Timestamp certificate is digitally signed by the server's private key.
- Integrity: Any changes with the document or certificate would cancel the verification.
- Non-repudiation: The server cannot deny having issued a timestamp once its signature is verified.

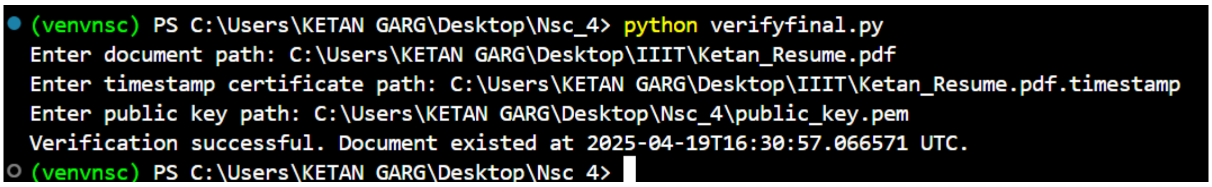
OUTPUTS-



```
PS C:\Users\KETAN GARG\Desktop\Nsc_4> .\venvnsc\Scripts\activate.ps1
(venvnsc) PS C:\Users\KETAN GARG\Desktop\Nsc_4> python clientfinal.py
Enter path to your document: C:\Users\KETAN GARG\Desktop\IIIT\Ketan_Resume.pdf
Document SHA256 hash: e2b3a6dbdce1d37f6617b307d35ee0c1411b342d216fe99c48300d20cb5e2b68
Timestamp certificate saved.
(venvnsc) PS C:\Users\KETAN GARG\Desktop\Nsc_4>
```

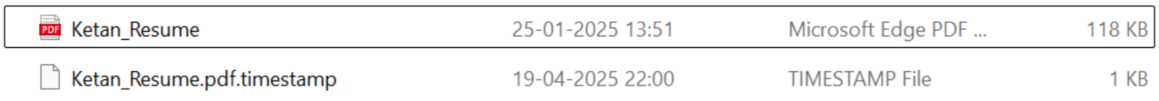


```
PS C:\Users\KETAN GARG\Desktop\Nsc_4>
PS C:\Users\KETAN GARG\Desktop\Nsc_4> .\venvnsc\Scripts\activate.ps1
(venvnsc) PS C:\Users\KETAN GARG\Desktop\Nsc_4> python serverfinal.py
 * Serving Flask app 'serverfinal'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [19/Apr/2025 22:00:57] "POST /timestamp HTTP/1.1" 200 -
```



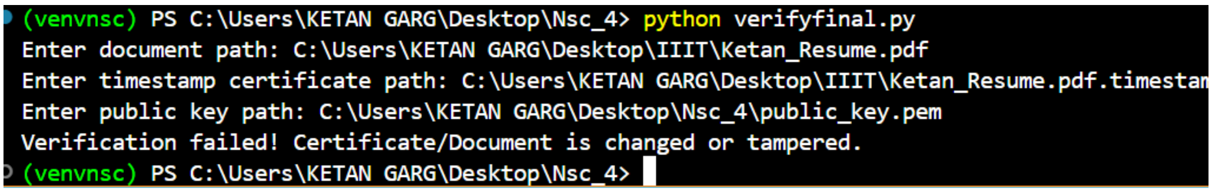
```
(venvnsc) PS C:\Users\KETAN GARG\Desktop\Nsc_4> python verifyfinal.py
Enter document path: C:\Users\KETAN GARG\Desktop\IIIT\Ketan_Resume.pdf
Enter timestamp certificate path: C:\Users\KETAN GARG\Desktop\IIIT\Ketan_Resume.pdf.timestamp
Enter public key path: C:\Users\KETAN GARG\Desktop\Nsc_4\public_key.pem
Verification successful. Document existed at 2025-04-19T16:30:57.066571 UTC.
(venvnsc) PS C:\Users\KETAN GARG\Desktop\Nsc_4>
```

.timestamp file saved in the same directory as of the original document.



Ketan_Resume	25-01-2025 13:51	Microsoft Edge PDF ...	118 KB
Ketan_Resume.pdf.timestamp	19-04-2025 22:00	TIMESTAMP File	1 KB

When document or certificate is tampered, verification fails -



```
(venvnsc) PS C:\Users\KETAN GARG\Desktop\Nsc_4> python verifyfinal.py
Enter document path: C:\Users\KETAN GARG\Desktop\IIIT\Ketan_Resume.pdf
Enter timestamp certificate path: C:\Users\KETAN GARG\Desktop\IIIT\Ketan_Resume.pdf.timestan
Enter public key path: C:\Users\KETAN GARG\Desktop\Nsc_4\public_key.pem
Verification failed! Certificate/Document is changed or tampered.
(venvnsc) PS C:\Users\KETAN GARG\Desktop\Nsc_4>
```