Project 0

CSE350: Programming Assignment 4

PRANJAL BHARTI 2021080S
DAKSH BHASIN 2021035

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Description

This Assignment investigates the establishment of a secure GMT-based Timestamping Authority, enabling clients to timestamp and digitally sign their documents through the Authority. Subsequently, any user can verify the authenticity of these documents. By employing cryptographic methods and adhering to standardized formats, the time stamped documents are safeguarded against tampering, instilling trust in their authenticity and integrity.

Tools & Algorithms Used

- Python RSA library is used for securely sending and receiving the files between user and the GMT Server application.
- Python hashlib library is used for generating the reliable sha-256 hash value.
- Internet Based GMT API is used for getting reliable & secure timestamp.
- gRPC is used as secured communication channel.

Server Working

- Key Generation
  - Generates a public-private key pair upon server initialization.
- Document Decryption
  - Decrypts the received document using the server's private key to access its content.
- Signature Generation
  - Creates a signature by hashing the document content along with the current timestamp.
- Response Generation
  - Sends back the signature and timestamp as a response to the client.
- Security Measures
  - Utilizes RSA encryption for secure communication.
  - Ensures confidentiality of document content through private key decryption.

Client Side Application: Request Sending

- At client side application, parse the content of the input file/string.
- Calculate the Hash value using hashlib.sha256() function. Store the hash value H1. Use the Authority Server public key to encrypt H1, using RSA 1024 based encryption. Send the encrypted Hash to Authority Server using gRPC, with request to timestamp the document. Wait for Server response.

```
if document.decode('utf8').endswith(".pdf"):
    response = stub.GetDocumentTimeStamp(GMT_pb2.TimeStampRequest
                                         (document=rsa.encrypt(
        generate_PDF_hash(document.decode('utf8')).encode(), server_public_key)))

    generate_PDF(document, response.signature, response.timestamp)

elif document.decode('utf8').endswith(".txt"):
    response = stub.GetDocumentTimeStamp(GMT_pb2.TimeStampRequest
                                         (document=rsa.encrypt(
        generate_TXT_hash(document.decode('utf8')).encode(), server_public_key)))

    generate_TXT(document, response.signature, response.timestamp)
```

Client Side Application: Response

Response Recording:

- At client side application, parse the content of the response received from server. Write the received
- encrypted Hash Response from server along with timestamp received into the document.

Verification:

- Recalculate the hash (H1) using same algorithm
- used initially for original content of file.
- Then append the timestamp extracted from
- document to H1 and recalculate the hash H2.
- Now decrypt the extracted message from the
- document sent by the server as hash H3.
  Use server public key for decryption.
  Now verify that calculated hash H2 is equal to decrypted hash H3.
  If equals then document is verified to be correctly time stamped & signed.

```
def verify_document(document):

    if document.decode('utf8').endswith(".pdf"):

        all_pages = ""

        with open(document, "rb") as file:
            reader = PyPDF2.PdfReader(file)

            for page in range(len(reader.pages) - 1):
                all_pages += reader.pages[page].extract_text()

            signature = RSA.decrypt(
                server_public_key.e, server_public_key.n,
                reader.pages[-1].extract_text()[-174:][:-1])

            hash = hashlib.sha256((hashlib.sha256(all_pages.encode()).hexdigest(
            ) + reader.pages[-1].extract_text()[-208:-175][1:]).encode()).hexdigest()

            if hash == signature:
                print("\n\nDocument Timestamped and VERIFIED Successfully.\n\n")

            else:
                print("\n\nDocument Verification Failed.\n\n")

            file.close()
```

Important Questions

Q1) How and where do you get the correct GMT date and time? Your laptop or the local Linux server is not good enough.

Ans: By utilizing an internet-based API call from a trusted World Time API service , we securely obtain the accurate GMT time. This method ensures reliability by fetching the most up-to-date GMT time directly from the server, thereby maintaining high accuracy and precision. Additionally, it eliminates any potential discrepancies caused by drift in local clocks, further enhancing its dependability.

Q2) When is the correct GMT date/time obtained?

Ans: The server application initially retrieves the accurate GMT date/time at the time of document signing, ensuring the document is signed reliably immediately upon receiving the request from the client side. Subsequently, this timestamp is transmitted to the client along with the response containing the signed document, maintaining the integrity and reliability of the signing process.

Important Questions

Q3) Is the source reliable? Is the GMT date and time obtained in a secure manner? The term ‘obtained’ refers to security of communication.

Ans: The server utilizes a reliable and secure HTTPS GET request to access the World Time API service, ensuring both integrity and security in obtaining the GMT time. Subsequently, the client securely receives the GMT time from the server application in an encrypted format, further enhancing the security of the data exchange process.

Q4) How do you ensure privacy, in that the server does not see/keep the original document?

Ans: By solely transmitting or utilizing the hash value of file contents instead of directly sending the entire file or original content to the server application, this ensures that other applications or parties cannot view or access the file content. The hash value, calculated using the SHA-256 algorithm, instills confidence in its reliability. Additionally, communication occurs via RSA-based encryption, further reducing the likelihood of privacy breaches.

Important Questions

Q5) How do you share the document with third parties in a secure manner with the GMT date/time preserved, and its integrity un-disturbed?

Ans: The document is shared using the public key cryptography-based RSA algorithm. This ensures that the document remains secure during transmission to third parties. The document is encrypted with the recipient's public key, ensuring that only the intended recipient, who possesses the corresponding private key, can decrypt and access the document. By incorporating the GMT date/time within the encrypted document, the integrity of the document is preserved, and the timestamp remains unaltered. This approach guarantees that the document's authenticity and the timing of its creation or transmission are verifiable by third parties. As a result, sensitive information can be securely shared with confidence, knowing that the document's integrity remains intact throughout the communication process.

Q6) 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”.

Ans: To ensure that users, including both the owner and anyone verifying the date/time, utilize the correct public key of the server stamping or signing the GMT date/time, a multi-step verification process is employed. Initially, the user identifies the authority responsible for the signature by recognizing its organization name appended within the file content at the time of signature creation. Once the authority responsible for the signature is identified, third parties can obtain its public key from any key distribution or certification authority. This public key is then utilized to decrypt the signature, ensuring the integrity and authenticity of the GMT date/time stamp. By following this process, any potential issues with the public key are mitigated, as the correct key associated with the signing authority is utilized for verification.

Q7) Which of these, viz. confidentiality, authentication, integrity and non-repudiation is/are relevant?

All four are relevant. Here’s a one-line summary you can drop into your slides:

- Confidentiality – Client hashes the document locally so the server never sees its contents.
- Authentication – Verifiers use the server’s published public key to confirm the stamp really came from your service.
- Integrity – Signing SHA256(document)||timestamp ensures any change to either breaks the signature.
- Non-Repudiation – Because only the server’s private key can produce the signature, it can’t later deny having stamped the document.

Results

```
/usr/local/bin/python3 /Users/greasyfinger/Desktop/NSC_A4_2021080_2021035/Client.py
greasyfinger@greasy-MacBook-2 NSC_A4_2021080_2021035 % /usr/local/bin/python3 /Users/greasyfinger/Desktop/NSC_A4_2021080_2021035/Cli
ent.py



<---------------GMT Authentication Service--------------->



Enter the Document Path or Content: Report.pdf


Document Timestamped and VERIFIED Successfully.


Enter the Document Path or Content: Presentation.pdf


Document Timestamped and VERIFIED Successfully.


Enter the Document Path or Content: I AM VERY HAPPY

I AM VERY HAPPY
GMT Authentication Service
2025-04-19 18:54:26.312540+00:00
NUubov3bolMPhkrAb4Gt3W7uUihj0aLcIsc4Wsj9yBXHrDeFn56XIbLLXBuAzHWS0Mid/6WGDPSgaMvkqGG93CcciTzJiR2zipWpiBGYpUyT7BQS5BVerGjp4Pqe5Me7i9Lb
gAN8ghHPQ2TBPaYoO0IjByE1efi4bfa9CL7Wets=


Document Timestamped and VERIFIED Successfully.


Enter the Document Path or Content: NETWORK SECURITY IS VERY INFORMATIVE COURSE AND PROFESSOR B.N JAIN IS VERY HELPFUL

NETWORK SECURITY IS VERY INFORMATIVE COURSE AND PROFESSOR B.N JAIN IS VERY HELPFUL
GMT Authentication Service
2025-04-19 18:55:27.573530+00:00
RuUy3MXWiPYGVBGY+zPdS+2JdSx9o0i8nKsdV2n8Fj8FG4wR3ag9hwjmyuSh3XZDWBkxIpOjR2Lkrcd8vVoEwB8KXYyImCcWElC61uhbbe6NuHiaqYmA3eZk0O9C6aX6Ovoc
rOoewtY1GYGLYqo5SoTmiMR6nlClfXz7qbAVFlE=


Document Timestamped and VERIFIED Successfully.


Enter the Document Path or Content:
```

```
greasyfinger@greasy-MacBook-2 NSC_A4_2021080_2021035 % python -u "/Users/greasyfinger/Desktop/NSC_A4_2021080_2021035/GMT Server.py"

GMT Time Stamping Server Running...



Document Signed and Timestamped Successfully.

Document Signed and Timestamped Successfully.

Document Signed and Timestamped Successfully.

Document Signed and Timestamped Successfully.
```

GMT Authentication Service
2025-04-19 19:17:50.186093+00:00
VHZmprDaI+9Q4/aWl37vp+SKbcaDsdkuQajt2yDqhVyIfGXe0YdT1pKZ/KVauhKOqpHMigYco7m