NS ASSIGNMENT 3 REPORT

Apoorv Singh (2017027)
Rohan Dev Verma (2017088)

Problem Statement

Project 0 time-stamping a document: This application relates to time-stamping a document that one may have prepared some moments ago. The process envisaged is: upload the document to the server (or perhaps some version of it) and expect to receive the same but with the current date and time stamped onto the document. Thus, there must exist a "GMT date & time-stamping server" which has the correct GMT date and time. It uses that to time-stamp documents (in some standard format) with the current GMT date/time and a digital signature. At any time, it should be possible to establish the fact that the document existed at the date/time stamped, and that the document has not been modified. Further:

1) How and where do you get the correct GMT date and time? And how often?
2) Is the source reliable and the GMT date and time obtained in a secure manner?
3) How do you ensure privacy, in that the server does not see/keep the original document?
4) How do you share the document with others in a secure manner with the date/time preserved, and its integrity un-disturbed?
5) Also ensure that the user has (and uses) the correct "public-key" of the "GMT date/time stamping server".

Solution

Suppose the client who wants to send the document is Alice, the GMT Server is TSA (or Time Stamping Authority) and the receiver of the document is Bob. Here, we have to assume TSA is a trusted source, and as it has a capability to stamp digital certificates, it performs the role of a Certification Authority as well. Hence, it contains the IDs and public keys of all clients in its

realm. Also, the clients contain the public keys of each other and TSA as well. The encryption used in this scheme is RSA throughout and the hashing algorithm used is SHA-256.

The procedure is as follows:

Alice first checks if she already has the certificate or not. If she doesn't have the certificate or the certificate has expired, she sends a request to the TSA for the document to be timestamped.

Alice derives a hash of the document she needs to send. She also adds who she wants to send this document, in this case, Bob. She sends this request to the TSA. TSA takes the hashed document, the current timestamp and the expiry time and constructs a package. The TSA derives the hash of this package and encrypts this hash with its own private key. This is the Digital Certificate. The TSA then sends this package along with the encrypted hash of this package to Alice.

Alice has the public key of the TSA. She decrypts the encrypted package using this key, and checks whether the hash of P matches the hashed package or not. Then, it checks whether the document has not been tampered with by checking that the hashed value of the document she sent (and which has been stamped) matches with the hash of the original document.

Now, she has to send the document to Bob. She takes the original document and encrypts with Bob's public key. She constructs a message containing the plaintext document, P as well as the certificate. Bob first checks the validity of the certificate by comparing the hash of P and the decrypted version of the certificate. Bob can do this as he has the public key of TSA. Then he checks whether the certificate has been expired or not. Finally he checks the integrity of the document by comparing the plaintext with the hash provided by Alice.

This scheme solves all the given problems mentioned by the problem statement. Due to the presence of the certificate which contains the hash of the document as well as the date and the time, one can check the integrity of the document as well as the validity of the certificate. Further:

1. The correct GMT date and time is derived by the TSA. Alice decides to get the certificate if the validity of the document has expired or if the document is new.
2. As the certificate is encrypted by the private key of TSA, no one else can tamper with the certificate.
3. As Alice sends the hash of the file, the TSA can't read the contents of the document
4. Alice sends the document to Bob encrypted with Bob's public key, hence only Bob can read it. Further, the message also contains the certificate which in turn contains the hash of the document, and hence Bob can check the integrity of the file.
5. All entities have each other's public keys. In case Alice has an incorrect public key of TSA, she won't be able to decrypt the hash correctly, and hence would be notified that the public key is incorrect.

Implementation Details

The language used is Python 3. The scheme is implemented by the following files:

1. methods.py: Contains the algorithms for encryption, decryption and hashing.
2. A.py: Contains the communication side of Alice.
3. B.py: Contains the communication side of Bob.
4. TSA.py: Contains the communication side of TSA.
5. Main.py: This is called the first time when the system has to be set up. The file sets up TSA, Alice and Bob, generating their public keys, private keys, IDs etc.

The certificate received by TSA is stored using the pickle module. pickle is also used to store the data generated by Main.py. SHA-256 from the library hashlib. Communication is done using the python socket library. The time and date is generated from the datetime module.

Example

- Request from Alice to TSA for a valid time-stamping

  Plaintext - “Hi this is a file to test RSA encrytion (Let's test-hashing while we're at it).”

```
From A to TSA
FOLLOWING IS THE MESSAGE SENT FROM ALICE TO TSA. THE MESSAGE CONTAINS-:
1. HASH OF THE ORIGINAL FILE
2. ID of Bob
36bc98bce460221c628b33121a179e086238b23f52fc3c87d0fb641e552834b5||527877009
```

- ```
  Connected to A
  FOLLOWING IS THE MESSAGE SENT BY TSA TO 'A'. THE MESSAGE CONTAINS-:
  1. HASH OF THE WHOLE MESSAGE, WHICH IS ENCRYPTED USING TSA'S PRIVATE KEY
  2. ID'S OF 'B' AND 'A'
  3. TIME AT WHICH THE MESSAGE WAS TIMESTAMPED
  4. EXPIRY OF THE TIMESTAMP
  908615864 18482485 1025700282 962417093 198654733 416104624 770672810 1326237804 1511780745 599168773 924774818
  801900914 538005970 464700011 1548260986 1878774250 120705396 1107494731 ||36bc98bce460221c628b33121a179e086238b
  23f52fc3c87d0fb641e552834b5||527877009||612886874||2020-03-23 19:53:49.235975||2020-03-23 19:58:49.235975
  ```

- ```
  FOLLOWING IS THE MESSAGE SENT FROM ALICE TO BOB. THE MESSAGE CONTAINS-:
  1. ORIGINAL FILE ENCRYPTED USING B'S PUBLIC KEY
  2. ALL THE DETAILS SENT BY TSA TO 'A'
  300893755 1596300181 1197685217 430518540 161399445 1877952752 1073730441 27753548
  2 1372249584 442212292 1331058122 214469694 105206825 174944949 1053756725 1769403
  209 2042614010 1888066009 960256969 799909233 401375177 341871715 ||908615864 1848
  2485 1025700282 962417093 198654733 416104624 770672810 1326237804 1511780745 5991
  68773 924774818 801900914 538005970 464700011 1548260986 1878774250 120705396 1107
  494731 ||36bc98bce460221c628b33121a179e086238b23f52fc3c87d0fb641e552834b5||5278770
  09||612886874||2020-03-23 19:53:49.235975||2020-03-23 19:58:49.235975
  ```

- ```
  Connected to A
  INFORMATIN WITH BOB
  Expiry Time 2020-03-23 19:58:49.235975
  Current Time 2020-03-23 19:53:49.237786
  PLAINTEXT
  Hi this is a file to test RSA encrytion (Let's test-hashing while we're at it).
  ```
  ```