# SIL-765 Network & System Security
# Assignment 3

-By Bipul Kumar & Ayush Gupta

**Project Statement**: This application relates to time-stamping a document that one may have prepared some moments ago. The process envisaged is: upload the document to server (or some version thereof) 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 document (in some standard format) with the current GMT data/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.

**1. How and where do you get the correct GMT date and time? And how often?**
→ We use the "time.ctime" module to get the correct GMT date and time. It's used in Server to add to the contents of the file before hashing it.
**2. Is the source reliable and the GMT date and time obtained in a secure manner?**
→ Yes, the module fetches time from the OS of the device on which its run, and this fetching is done at the server. We assume that Server has not become malicious and is secure.
**3. How do you ensure privacy, in that the server does not see/keep the original document?**
→ The hash of the message is sent to server. This ensures that the information is not altered during the communication. Server only received the hash of data so this ensure message integrity.
**4. How do you share the document with others in a secure manner with the date/time preserved, and integrity un-disturbed?**
→ During sharing with other client ,we encrypt the data by rsa so this will ensure that data received by only client2,i.e only client2 can decrypt the received data.
**5. Also ensure that the user has (and uses) the correct "public-key" of the "GMT date/time stamping server".**
→ For this I have create a separate public directory and any client who want to use it can access from it.

## Implementation

### Key Generation

For this use two different script one for clients key generation and other for server's key generation,and create two file for private and public key for each client and server.

- **Generating a private/public key pair using RSA algorithm with pycrypto**
- **Specify the key size as 4096 bits for client key generation and key size 1024 in case of server key generation. Larger is more secure.**

- **Use Random module of pycrypto to specify a random number generator function.**

## Public Key Directory

Also make public key directory which contains public key for each client and server ,along with its id.

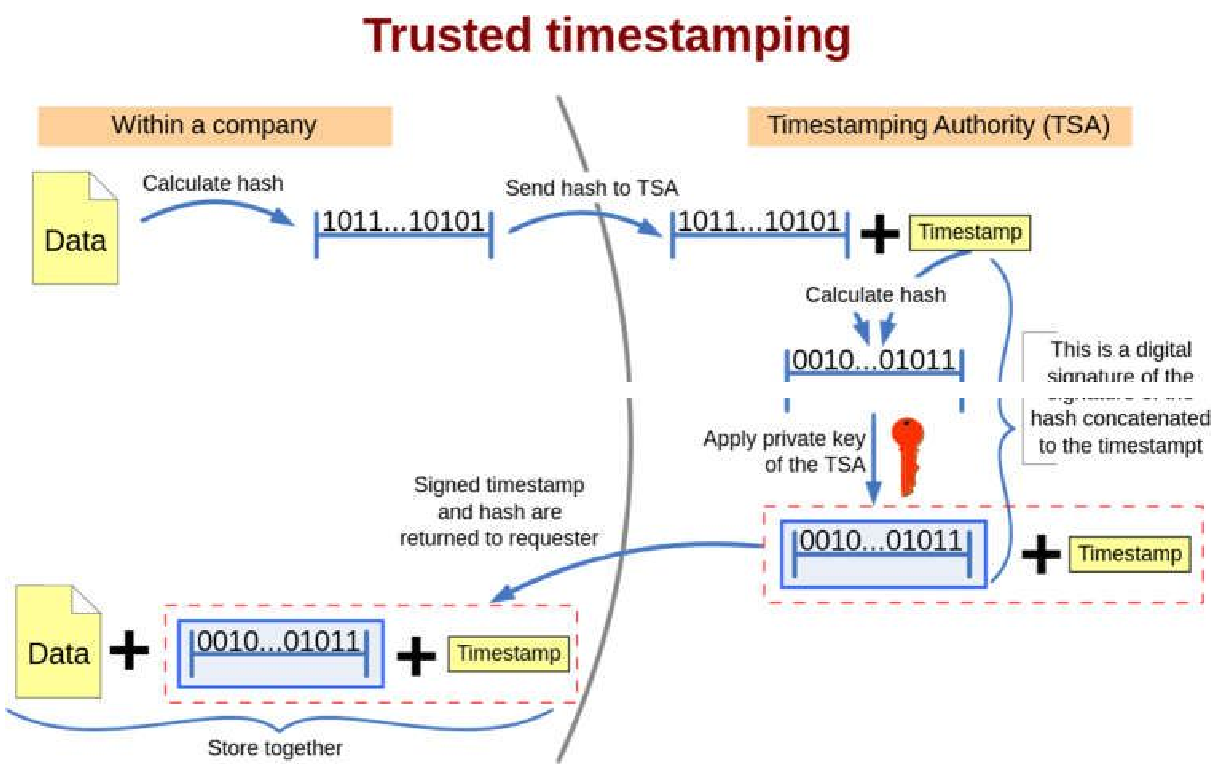
## Client/Server Communication

1. First run script serverkeyGenerate.py and clientkeyGenerate.py script to generate public/private key for server, client1, client2.
2. The client1 hash of the message and encrypt with public key of server, and send to server.
    a. The cryptographic hash is used to ensure Integrity of the message, essentially to ensure that the information has not been altered in the communication.
    b. Encrypting by server ‘public key ensure that this message only decrypt by server by its private key.
3. Now server add the timestamp and hash the whole content, then sign it with its private key. This result in Digital Signature which ensure that in future this data signed by server, this sign and above received hashed doc along with timestamp sent back to client1 by encrypting client1 private key.
4. Client1 decrypted the received data and get timestamp, signature and hashed document.
    a. Now client1 can verify hash document by comparing it with sent hash data.
    b. It can verify the signature by hashing the hash doc combine with timestamp then verify with signature received.

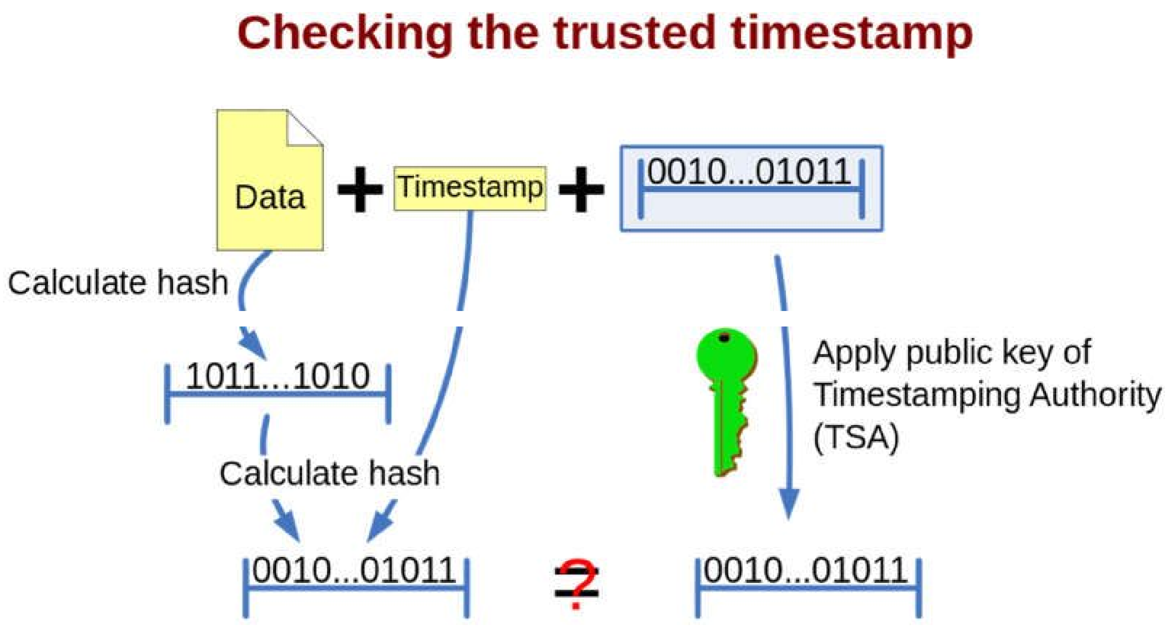
**Sharing Documents with other client2**

5. Now client1 make another message by combine original data ,timestamp with digital signature received and time stamp received from server, then encrypts whole message by public key of client2.
    a. This ensure confidentiality that, this message is only seen or decrypt by client2’s private key.
6. At client2’s site, it received message from client1 then it decrypts it by using its own private key.
    a. From there it retrieves original message, digital signature and timestamp.
    b. It makes hash of original data and concatenate with time stamp and again do hash of all.
    c. Also, do decrypt digital signature by server public key.
    d. Now it verifies above two, if both same then verified else not verified.

**Flow Chart**



Trusted timestamping
Within a company
Timestamping Authority (TSA)
Data
Calculate hash
1011...10101
Send hash to TSA
1011...10101
Timestamp
Calculate hash
0010...01011
This is a digital signature of the hash concatenated to the timestampt
Apply private key of the TSA
0010...01011
Timestamp
Signed timestamp and hash are returned to requester
Data
0010...01011
Timestamp
Store together



Checking the trusted timestamp
Data
Timestamp
0010...01011
Calculate hash
1011...1010
Calculate hash
Apply public key of Timestamping Authority (TSA)
0010...01011
≟
0010...01011