ASSIGNMENT - 2

Network and System Security(SIL-765)

GMT Date and Time Server

RUCHIR DHIMAN
(2016MCS2685)

SHANTANU AGARWAL
(2016MCS2661)

Problem Statement

This application relates to keeping and providing a certified copy of the current time and date in a secure manner. In particular a client should be able to request and obtain a digitally signed copy of the current GMT date and time.

For this, used, we accessed a web based “GMT date and time server” which then returns a file which contains the current GMT date and time, suitably signed digitally with the server’s RSA-based private key.

Introduction

RSA Algorithm

RSA (**Rivest–Shamir–Adleman**) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and it is different from the decryption key which is kept secret (private).

We have used the RSA algorithm to generate the public private key pairs for all the entity in the current system. These keys are further used in encryption and decryption process.

Digital Signature

A **digital signature** is a mathematical scheme for demonstrating the authenticity of digital messages or documents. A valid digital signature gives a recipient reason to believe that the message was created by a known sender (authentication), that the sender cannot deny having sent the message (non-repudiation), and that the message was not altered in transit (integrity).

We have used the digital signatures in all the communications. A sender sends the data and a digital signature over that data to the receiver and receiver verifies the authenticity and integrity of the message before accepting it in the system.

Cryptographic Hash Functions

They are also useful in cryptography. A cryptographic hash function allows one to easily verify that some input data maps to a given hash value, but if the input data is unknown, it is deliberately difficult to reconstruct it (or equivalent alternatives) by knowing the stored hash value. This is used for assuring integrity of transmitted data, and is the building block for HMACs, which provide message authentication.

We have used the “SHA1” hashing algorithm everywhere for signing and verifying the sign during communication.

Technical Details

1. Key Distribution Centre and the GMT time server are implemented in PHP 5.4.
2. In-built RSA algorithm and the signing and verification algorithms are used.
3. Client is implement in Python 2.7.
4. Communication between the three entities are implemented using HTTP and HTTPS connections.
5. PHP files are hosted on “www.shantanu.pro” domain while client python file is run on localhost.

Assumptions

1. The public key of the key distribution centre is already available with the client and the GMT server.
2. The public key of client and GMT server are already there with the key distribution centre.
3. The GMT server is hosted on the online servers which are in turn synced with the UTC servers, which are reliable source of current date and time.[Question 1 asked in assignment detail].
4. Message latency is assumed to be zero. So, the time received from the server won't be offset due to transmission time and network delays.

Algorithm

Our mechanism uses three entities: a client, running as a python script, a key distribution centre and the time server, both of which run online. The KDC has the public keys of the participating entities, and the client gets the public key of the server using this KDC, using the mechanism described below (steps 1 - 5).

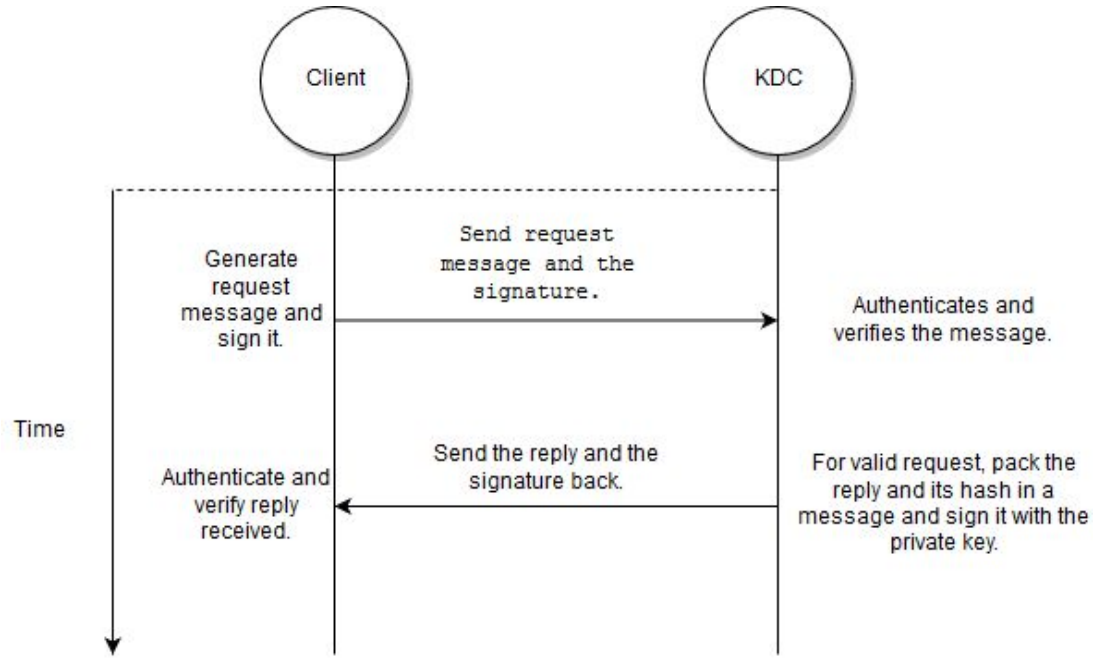


Figure 1. Obtaining the public key from the KDC.

The actual time request is then made to the time server. If the request is valid the server replies with the date and time. Both, the request to the server and the reply from it, are digitally signed using RSA to ensure integrity as well as to allow for message authentication.

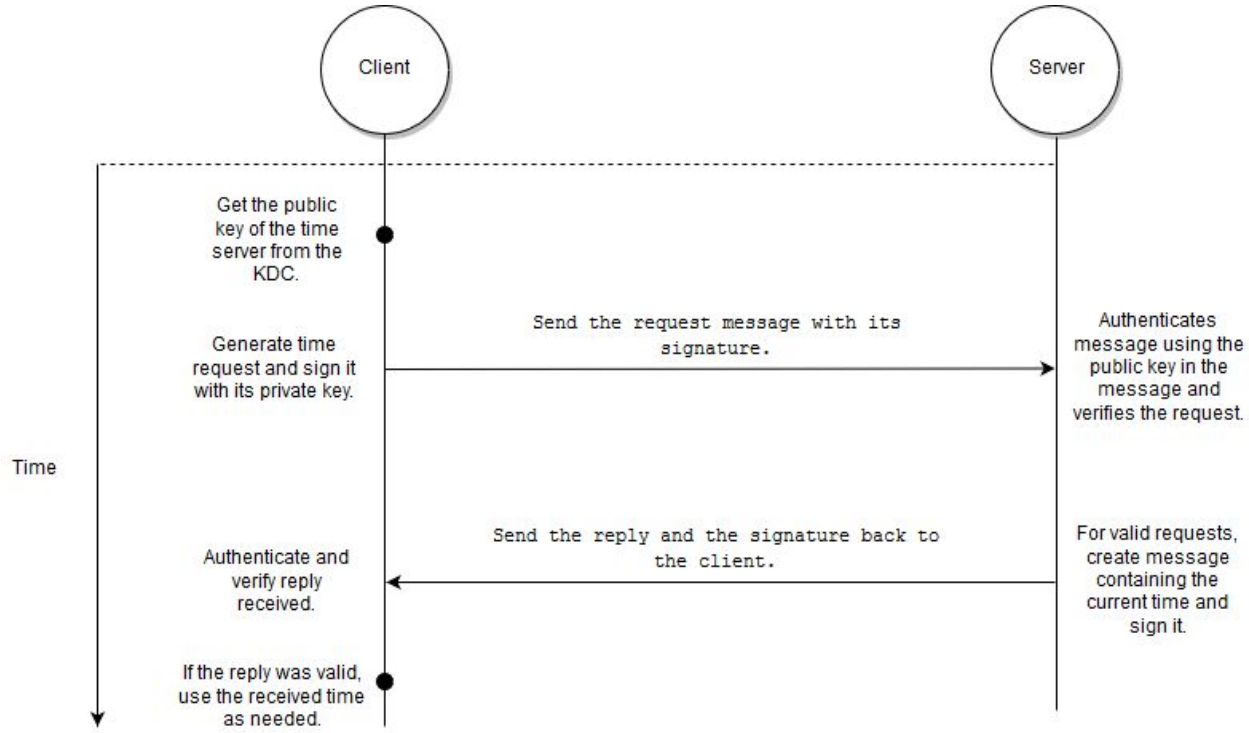
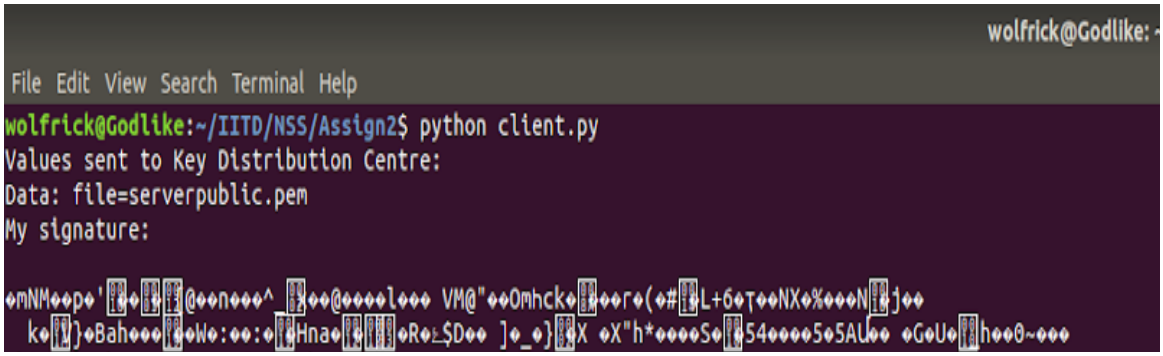


Figure 2. Procedure of requesting time.

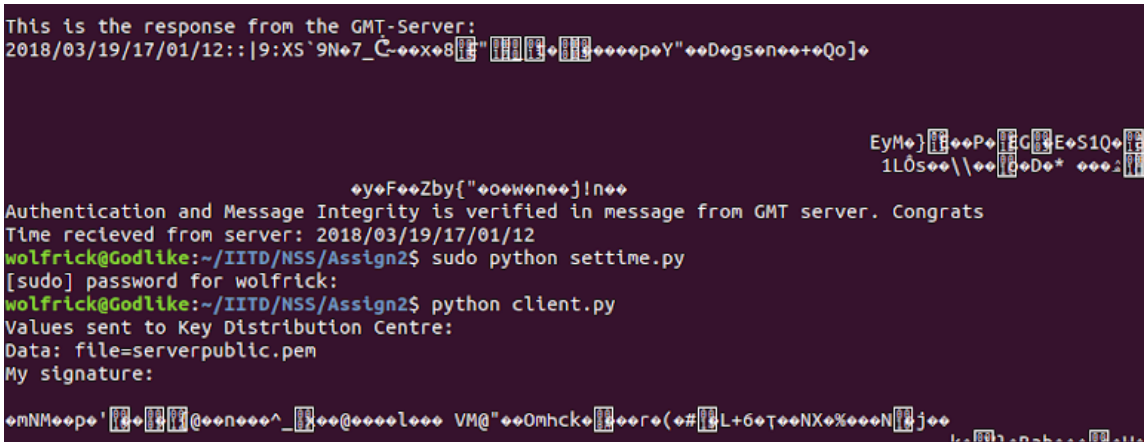
The detailed functioning is illustrated and described, step by step, below.

1. The client generates a message to request the server's public key and signs it using its own RSA-based private key.
2. It sends the message with the signature to the KDC. The signature is sent along with the message to thwart tampering of the message by some intermediate entity (man-in-the-middle).



3. The KDC authenticates the message using the public key of the client and verifies it (both done using the same library function).
4. If the message is validated, the KDC generates a reply (the public key of server), and calculates the RSA-based signature. The client already has the public key of the KDC, and so can authenticate if the message was indeed sent by the KDC. The signature also prevents MITM attacks.
5. The client authenticates the message and unpacks it to get the server's public key. This will be later used to authenticate the message received from the time server.

6. The client, then, makes a request to the time server. It finds the signature for the message, and then sends the request and the signature to the time server. The message has its public key in the message, so that the server can authenticate the message (verification also done alongside).
7. The server authenticates the message using the client's public key retrieved from the request message. This step also includes verification of the message, so as to avoid tampered messages.
8. It then replies by generating a reply (containing the date and time), signs it using its private key, and sends the reply back to the client.



9. The client authenticates the message using the public key received in step 5. If authentication is successful, it can then use the data received and set the current system date and time; otherwise, it displays an error.

Files Used

client.py:

This file contains all the code used by the client for sending and receiving the authenticated and integrity maintained messages.

clientrsa.py

This file is used to generate new public private key pairs for client.

KDCrsa.php

This file is used to generate new public private key pairs for KDC.

serverrsa.php

This file is used to generate new public private key pairs for server.

settime.py

This file is used to set the systems date and time to the one received from GMT server.

keydistributor.php

This file includes all the code needed by the KDC to communicate with client and server and providing them the public keys of each other in secure manner.

gmtserver.php

This file contains code needed by gmt server to communicate with client to provide it with current date and time.