

Foundations of Cybersecurity

Project documentation

Irene Cantini

Federica Perrone

Content

[Introduction 3](#_Toc108705903)

[Authentication phase 4](#_Toc108705904)

[Operations’ implementation 7](#_Toc108705905)

[Upload 7](#_Toc108705906)

[Download 7](#_Toc108705907)

[Delete 7](#_Toc108705908)

[List 7](#_Toc108705909)

[Rename 7](#_Toc108705910)

[Logout 7](#_Toc108705911)

[Meeting the security requirements 8](#_Toc108705912)

[User manual 9](#_Toc108705913)

# Introduction

This project is about the implementation of a Client-Server application that resembles Cloud Storage.

Each user has a dedicated storage on the server, and each of them can only access their own dedicated storage.

After the authentication phase, in which the client must authenticate with the server, the client can upload, download, rename or delete data to/from the Cloud Storage in a safe manner.

Users are pre-registered on the server, specifically there are four users. Each user has a long-term RSA key-pair, and the long-term private key is password-protected. The server knows the username of every registered user and the RSA public key of every user. For each registered user, the server has already allocated the user dedicated storage. In the following table (Table 1) are show the four pre-registered users with their username and password.

|  |  |  |
| --- | --- | --- |
| User | Username | Password |
| Alice | alice | alice |
| Bob | bob | bobb |
| Carol | carol | carol |
| Dave | dave | dave |

Table 1 - Pre-registered users

# Authentication phase

When the client application starts, Server and Client must authenticate.

The Server must authenticate with the public key certified by the certification authority, while the Client must authenticate with the public key pre-shared with the server.

The communication between the Client and the Server for the authentication and the establishment of the session key is shown in the image below (Figure 1).

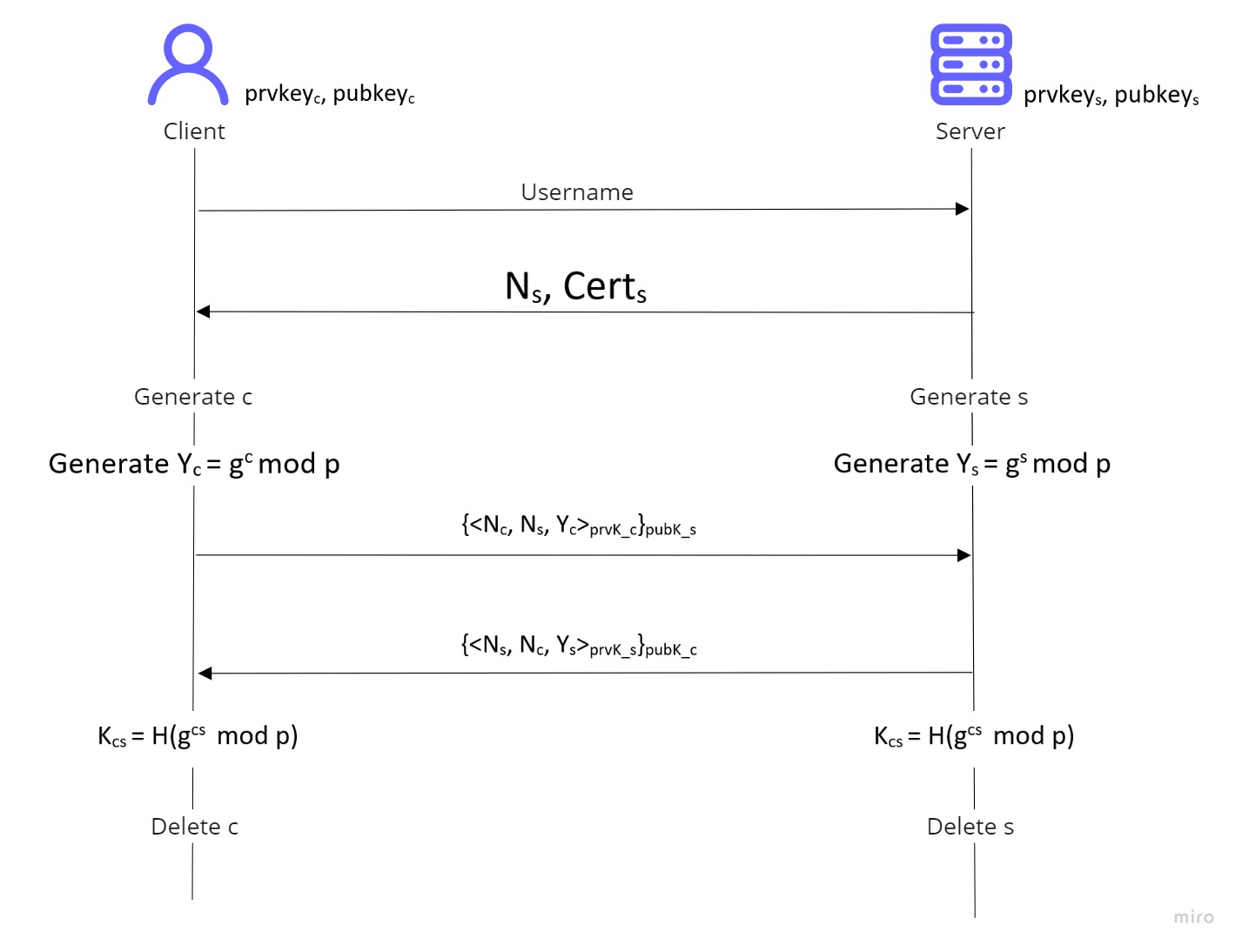
As you can see from the sequence diagram, the first message the client transmits is its username. This message is sent in clear (Figure 2).

Figure - Authentication phase sequence diagram

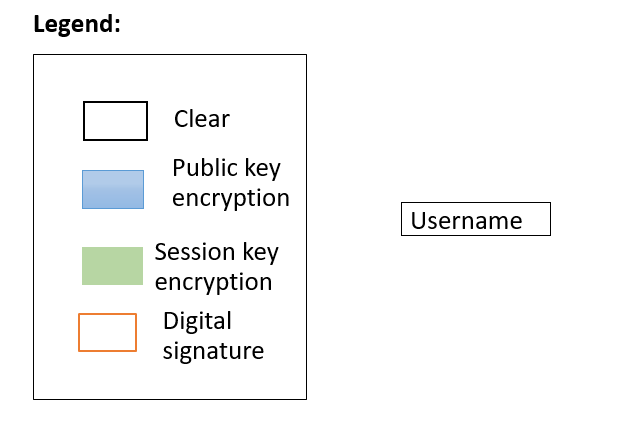


Figure 2 - Client's Username message format

After receiving the Client’s username, the Server verifies that the username belongs to a pre-registered user. If the check is not successful, the server closes the connection and the authentication phase fails.

If instead the verification is successful, the server sends to the client its nonce and its certificate. This message is sent in clear (Figure 3).

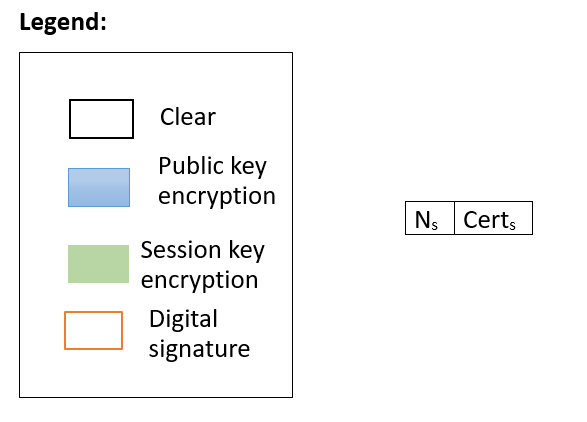
After this message, Client and Server generates their public key following *Diffie-Hellman* key generation protocol.

Figure 3 - Server's nonce and certificate message format

In the next step, the Client concatenates its public key with its nonce and the server nonce. (S)He signs this message with its private key to prove the server his identity, and encrypts the concatenation between the message and the signature by means of the server’s public key. Finally (s)he sends the message to the server (Figure 4).

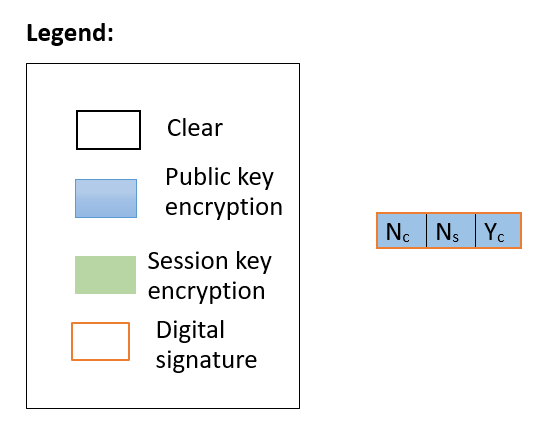
The server follows the same procedure that the Client has followed in the previous message. It takes Client’s nonce, its nonce and its public key, signs the whole message and encrypts it with client’s public key. Then the Server sends it to the Client (Figure 5).

Figure 4 - Client's DH key message format

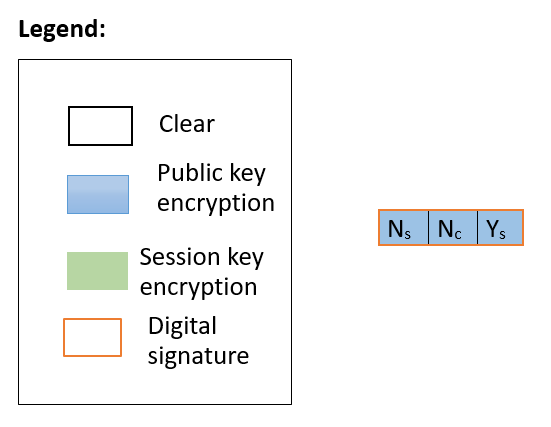
Finally, client and server are both able to generate the session key using the other part public key and their private key following the *Diffie-Hellman* protocol. The shared key is obtained first by deriving the shared secret using the two *Diffie-Hellman* keys and then it is hashed by using SHA-256 algorithm in order to obtain the symmetric key.

Figure 5 - Server's DH key message format

# Operations’ implementation

After the authentication phase, in which the client must authenticate with the server, the client can upload, download, rename or delete data to/from the Cloud Storage in a safe manner.

All the message in the session are encrypted with the session key established in the authentication phase and authenticated using the *authenticated encryption*. In particular, all the messages are been encrypted using AES with a key length of 128 bits as a block cipher and *Galois-counter mode* as the encryption mode. For each symmetric encryption, a random generated IV is used, while the AAD is constituted by a counter that counts the number of messages sent by a certain party. It is useful to avoid reply attacks during a single session.

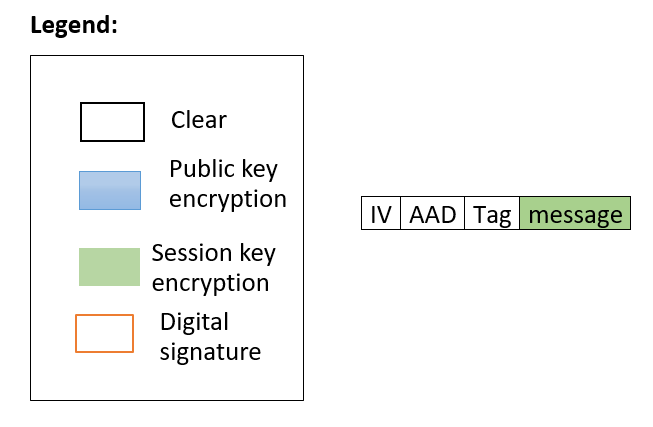
In the following picture is shown the general message format exchanged during the session (Figure 6).

Figure 6 - General message format during session

## Upload

## Download

## Delete

## List

## Rename

## Logout

# Meeting the security requirements

The security requirements where **Perfect Forward Secrecy**, **encryption and authentication** of the entire session and **resistance to reply attacks**.

**Perfect Forward Secrecy** is guaranteed by the *Ephemeral Diffie-Hellman key exchange*.

**Encryption and authentication** is achieved using *authenticated encryption*. In particular, all the session messages are been encrypted using *AES* with a key length of 128 bits as a block cipher and *Galois-counter mode* as the encryption mode. For each symmetric encryption a random generated IV is used to avoid predictability.

**Resistance to reply attacks** in the authentication phase is given by the use of *Ephemeral Diffie-Hellman*, while during data exchange is given by an incremental counter that is sent in the AAD field of every session message.

# User manual