



UNIVERSITÀ DI PISA

Foundations of Cybersecurity  
Project documentation

Irene Cantini  
Federica Perrone



## Content

Introduction.....	3
Authentication phase .....	4
Operations' implementation .....	6
Upload .....	6
Download .....	7
Delete .....	8
List.....	9
Rename.....	9
Logout.....	10
Meeting the security requirements.....	11
User manual.....	12
Starting the server .....	12
Starting the client .....	12
Upload .....	13
Download .....	14
Delete .....	14
List.....	15
Rename.....	16
Logout.....	16

# 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).

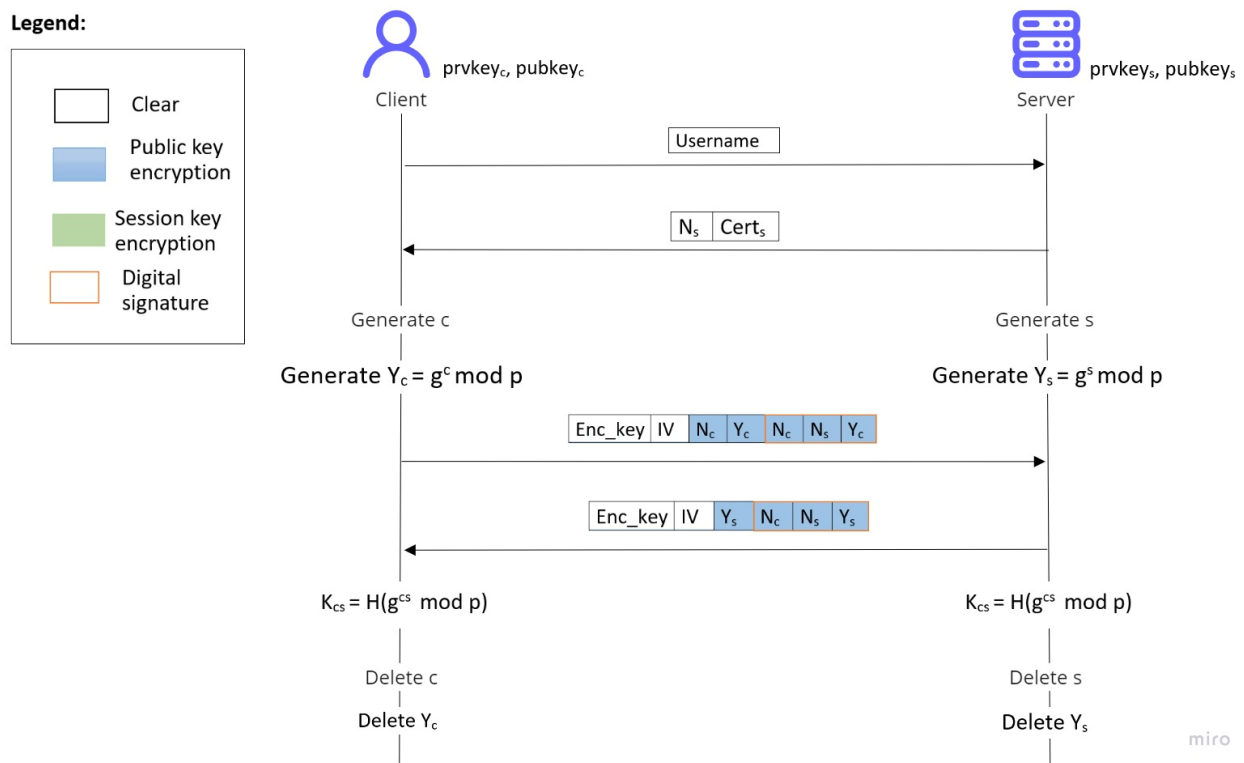


Figure 1 - Authentication phase sequence diagram

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

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.

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

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. Then, (s)he concatenates this signature with its nonce and its public key and encrypts the concatenation by means of the server's public key. Finally, (s)he concatenates this message with a random IV and additional information that are necessary for the asymmetric encryption and then (s)he sends the message to the server.

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 and signs the whole message. Then, it concatenates the signature with its public key and encrypts it with client's public key. Finally, it concatenates this message with a random IV and additional information that are necessary for the asymmetric encryption and then it sends it to the Client.

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 digest. The symmetric key is obtained extracting the first 128 bit from the digest.

## Operations' implementation

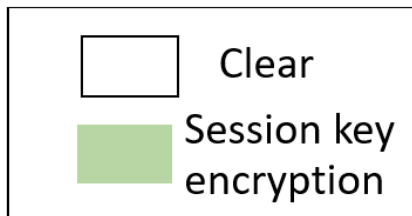
After the authentication phase, in which the client must authenticate with the server, the client can upload, download, list, rename or delete data to/from the Cloud Storage and logout in a safe manner.

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 2).

The tag is computed only on the encrypted part.

### Legend:



12 Byte	12 Byte	16 Byte	4 Byte	Variable
IV	AAD	Tag	Type	Payload

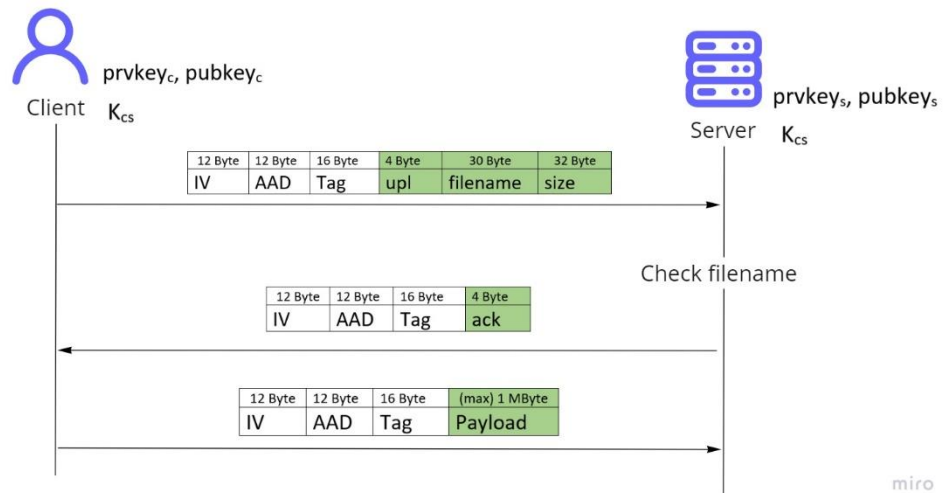
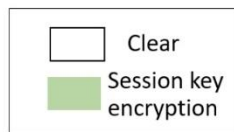
Figure 2 - General message format

The Type of the message can assume one of the following values:

Update_request	"upl"
Acknowledgement	"ack"
Not_acknowledgement	"not"
Download_request	"dow"
Size	"siz"
Delete_request	"del"
Ask_confirmation	"ask"
List_request	"lre"
List_file	"lfi"
Rename_request	"ren"
Logout_request	"log"

### Upload

The communication between the Client and the Server for the upload operation is shown in the image below (Figure 3).

**Legend:***Figure 3 - Upload sequence diagram*

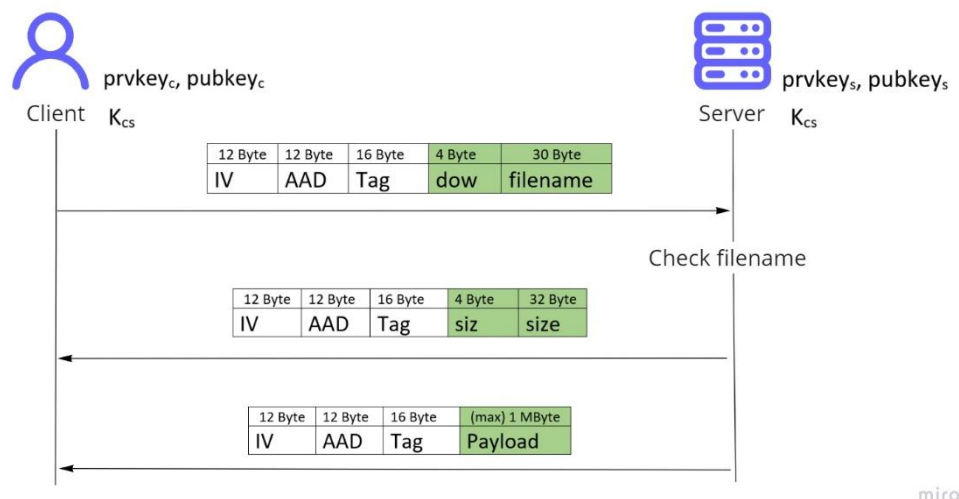
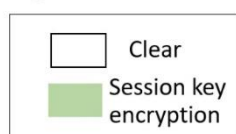
In the first message, the client sends the upload request, along with the filename and the size<sup>1</sup> of the file to upload. The filename can not contain special characters, for this reason it is checked using a white list, and if the check fails the application asks for a valid filename, in addition the file to upload must exist.

When the server receive the upload request, it checks if the filename is correct using a white list. If the file already exists it will be overwritten. Then, the server sends an acknowledgement to the client.

After these steps, the client sends the file content to the server. It could be necessary encrypt and send the file content more than once if the file size is bigger than 1MByte.

**Download**

The communication between the Client and the Server for the download operation is shown in the image below (Figure 4).

**Legend:***Figure 4 - Download sequence diagram*

<sup>1</sup> The size field is on 32 byte because it is a string representing a long long int.



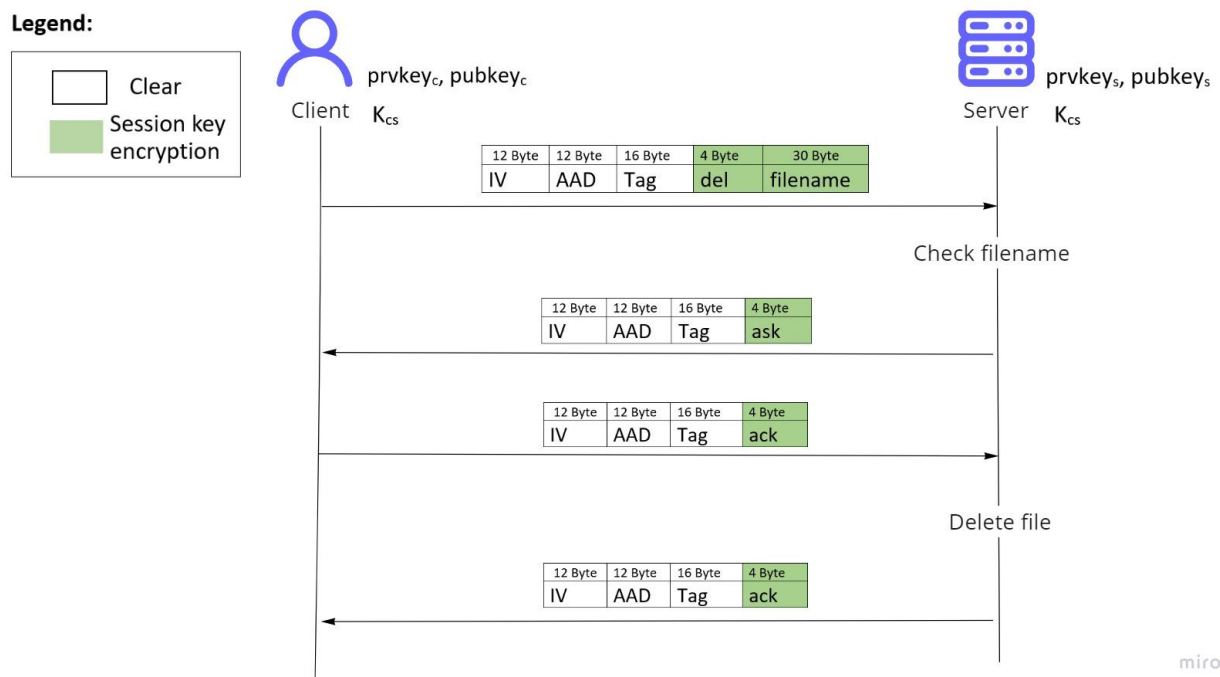
In the first message, the client sends the download request, along with the filename of the file to download. The filename can not contain special characters, for this reason it is checked using a white list, and if the check fails the application asks for a valid filename.

When the server receive the download request, it checks if the filename is correct using a white list and it checks that the file exists. Then, the server sends the size<sup>2</sup> of the file to the client.

After these steps, the server sends the file content to the client. It could be necessary encrypt and send the file content more than once if the file size is bigger than 1MByte.

## Delete

The communication between the Client and the Server for the delete operation is shown in the image below (Figure 5).



In the first message, the client sends the delete request, along with the filename of the file to delete. The filename can not contain special characters, for this reason it is checked using a white list, and if the check fails the application asks for a valid filename.

When the server receive the delete request, it checks if the filename is correct using a white list and it checks that the file exists. Then, the server sends a confirmation request to the client.

If the client want to delete the file, it sends an acknowledgement to the server, otherwise a not acknowledgement.

Finally, the server deletes the file and send an acknowledgement to the client.

<sup>2</sup> The size field is on 32 byte because it is a string representing a long long int.

## List

The communication between the Client and the Server for the list operation is shown in the image below (Figure 6).

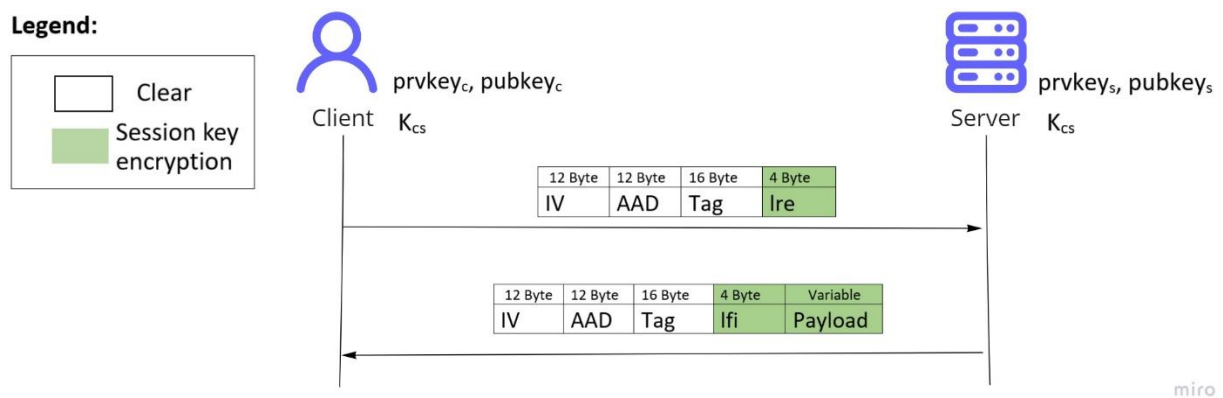


Figure 6 - List sequence diagram

In the first message, the client sends the list request to the server.

The server sends the user cloud storage content to the client. The different filenames are separated by comma.

## Rename

The communication between the Client and the Server for the rename operation is shown in the image below (Figure 7).

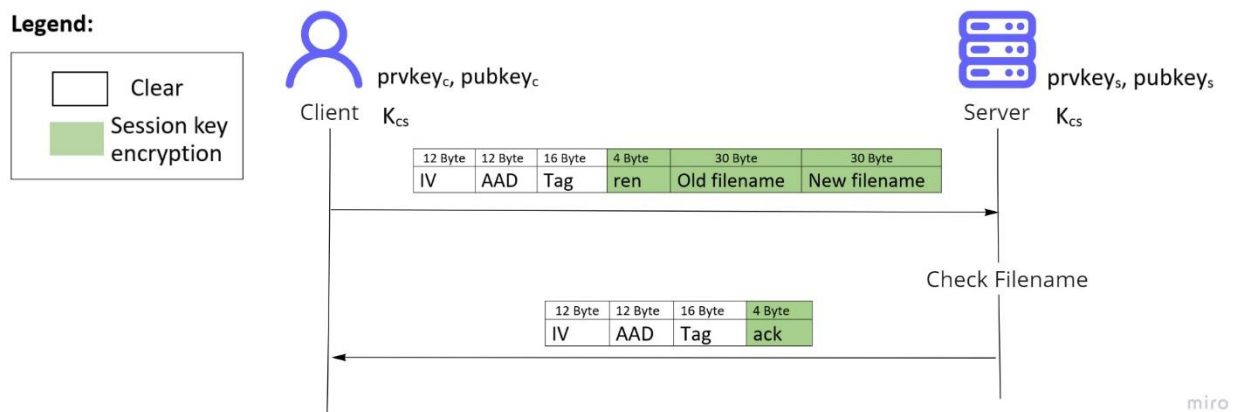


Figure 7 - Rename sequence diagram

In the first message, the client sends the rename request, along with the old filename and the new filename of the file to rename. The two filenames can not contain special characters, for this reason they are checked using a white list, and if the check fails the application asks for valid filenames.

When the server receive the rename request, it checks if the filenames are correct using a white list and it checks that the old file exists.

Finally, the server renames the file and send an acknowledgement to the client.

## Logout

The communication between the Client and the Server for the logout operation is shown in the image below (Figure 8).

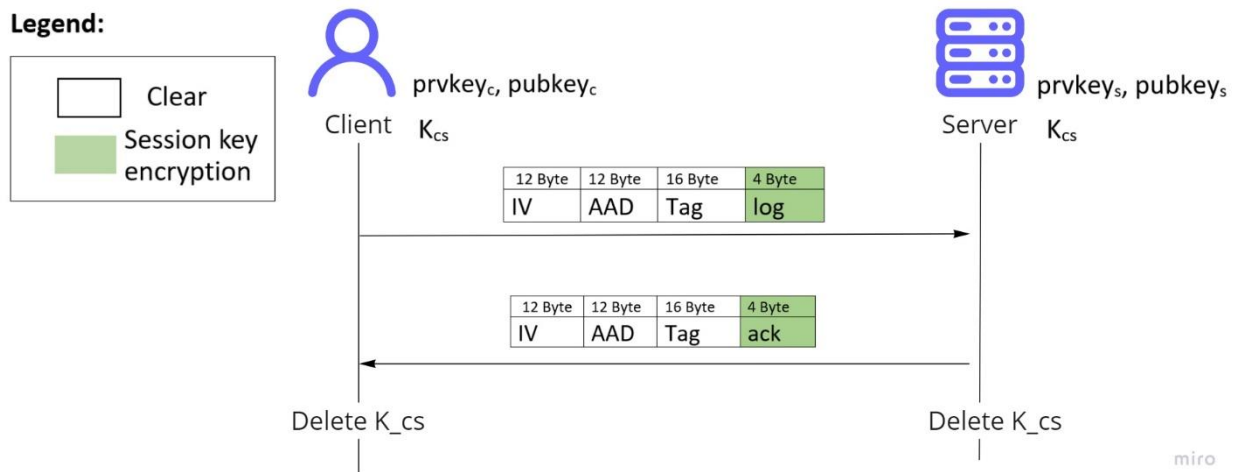


Figure 8 - Logout sequence diagram

In the first message, the client sends the logout request to the server.

The server sends the acknowledgement.

Finally, both the client and the server delete the session key.

## 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.

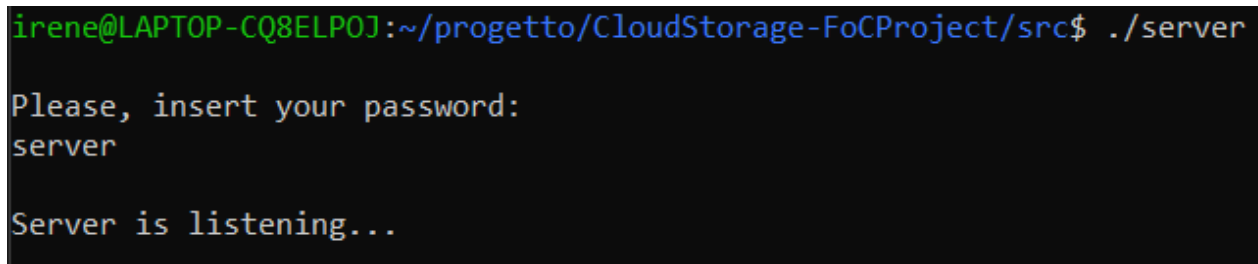
**Resistance to reply attacks** in the authentication phase is given by the use of *Ephemeral Diffie-Hellman*, while during data exchange is given by an incremental counter that is sent in the AAD field of every session message.

## User manual

Before starting executing the server and the client, the source files have to be compiled. To simplify this task, a makefile has been created.

### Starting the server

The server can be run using the command `./server`. After starting, the server asks you to insert a password. The password requested is the one used to protect the server's private key. If the password entered is wrong the server will not be able to access its private key and the execution will stop with an error.

A terminal window with a black background and green text. The prompt is 'irene@LAPTOP-CQ8ELP0J:~/progetto/CloudStorage-FoCProject/src\$'. The user has entered './server'. The output shows 'Please, insert your password:' followed by 'server' on the next line, and 'Server is listening...' on the third line.

```
irene@LAPTOP-CQ8ELP0J:~/progetto/CloudStorage-FoCProject/src$ ./server
Please, insert your password:
server
Server is listening...
```

*Figure 9 - Server starts*

### Starting the client

The client can be run using the command `./client <port number>`. After starting, the user is asked to insert his/her password. The password here requested is the one used to protect the user's private key. If the password entered is wrong the client will not be able to access his/her private key and the execution will stop with an error.

After the password insert, the authentication phase starts and at the end the menu of the possible operations is shown.

```

irene@LAPTOP-CQ8ELP0J:~/progetto/CloudStorage-FoCProject/src$ ./client 5001

Please, insert your username:
alice

Please, insert your password:
alice

Connection established

Start the AUTHENTICATION PHASE...

Send the username to the server
Received nonce and certificate from server
received valid cert
Sended message: <EncKey | IV | Nc | Yc | sign>
Received message: <EncKey | IV | Ys | sign>
Authenticated server
Session key generation: success

Finish AUTHENTICATION PHASE

Start SESSION...

*****MENU*****
1) Upload
2) Download
3) Delete
4) List
5) Rename
6) Logout
*****
Insert the chosen code operation

```

Figure 10 - Client starts

## Upload

When the user enter the input 1 it means he/she wants upload a file.

```

*****MENU*****
1) Upload
2) Download
3) Delete
4) List
5) Rename
6) Logout
*****
Insert the chosen code operation
1

Please, type the file to upload:
testUpload.txt
Sended message: <IV | AAD | tag | upload_request | filename | size>
Received message <IV | AAD | tag | Acknowledgement_type>
Sendend message <IV | AAD | tag | file content>
Upload request: success

```

Figure 11 - Upload operation

## Download

When the user enter the input 2 it means he/she wants download a file.

```

*****MENU*****
1) Upload
2) Download
3) Delete
4) List
5) Rename
6) Logout
*****
Insert the chosen code operation
2

Please, type the file to download:
testDownload.txt
Sended message: <IV | AAD | tag | download_request | filename >
Received message <IV | AAD | tag | Size_type | size file>
Received message <IV | AAD | tag | file content>
Download: success

```

Figure 12 - Download operation

## Delete

When the user enter the input 3 it means he/she wants delete a file.

```

*****MENU*****
1) Upload
2) Download
3) Delete
4) List
5) Rename
6) Logout
*****
Insert the chosen code operation
3

Please, type the file to delete:
testDelete.txt
Sended message: <IV | AAD | tag | delete_request | filename >
Received message <IV | AAD | tag | Ask_confirmation_type>
Are you sure?(Yes/No)
Yes
Sended message <IV | AAD | tag | Acknowledgement_type>
Delete request: success

```

Figure 13 - Delete operation

## List

When the user enter the input 4 it means he/she wants to list the content of its clod storage.

```

*****MENU*****
1) Upload
2) Download
3) Delete
4) List
5) Rename
6) Logout
*****
Insert the chosen code operation
4
Sended message: <IV | AAD | tag | List_request >

List:
data.tsv
testDownload.txt
file1.txt
file2.txt

list request: success

```

Figure 14 - List operation



## Rename

When the user enter the input 5 it means he/she wants rename a file.

```
*****MENU*****
1) Upload
2) Download
3) Delete
4) List
5) Rename
6) Logout
*****
Insert the chosen code operation
5

Please, type the file to rename:
file1.txt

Please, type the new filename:
file3.txt
Sended message: <IV | AAD | tag | rename_request | old_filename | new_filename >
Received message <IV | AAD | tag | Acknowledgement_type>
Rename request: success
```

Figure 15 - Rename operation

## Logout

When the user enter the input 6 it means he/she wants to logout.

```
*****MENU*****
1) Upload
2) Download
3) Delete
4) List
5) Rename
6) Logout
*****
Insert the chosen code operation
6

Init Logout...
Sended message <IV | AAD | tag | Logout_request_type>
Received message <IV | AAD | tag | Acknowledgement_type>
Logout: success
```

Figure 16 - Logout operation