README.md - Grip

Secure-online-messaging-service

Group project for the Foundations of Cybersecurity course of the Artificial Intelligence and Data Engineering Msc at University of Pisa

Project by Lorenzo Barigliano, Matilde Mazzini, Giulio Silvestri

Introduction

This is a secure client server application: the messages exchanged are signed and encrypted using the OpenSSL library.

The multi threaded server can handle multiple clients connected at the same time.

For every new connection the server creates a new thread which will comunicate with its associated client.

After a preliminar authentication phase whith the incoming clients, the server main thread forwards the messages received from the sending client to the destination client.

The client is multi threaded too in order to receive messages while sending them.

Opcodes:

These are the opcodes that help identifing the messages different structure.

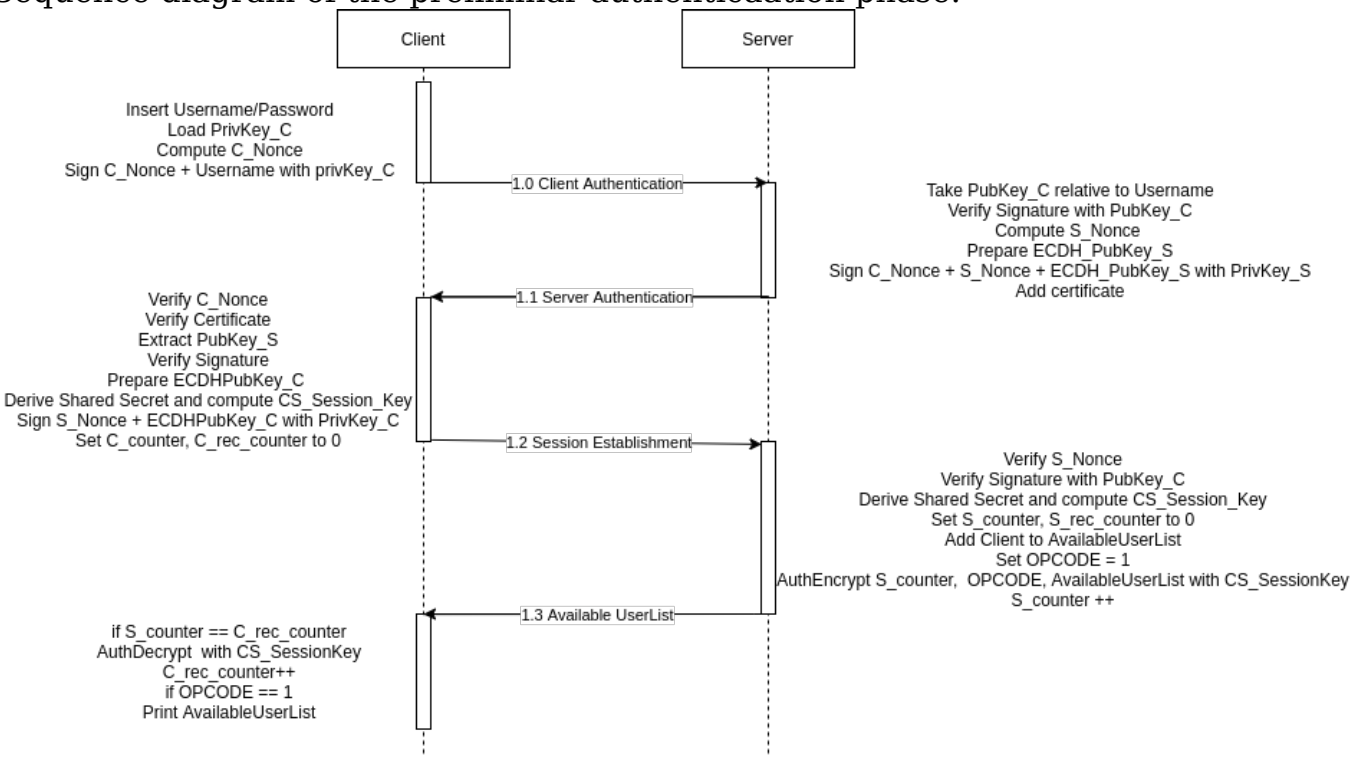
OPCODE#	Name	Meaning
0	QUIT	the user exits the application
1	LIST	the user requests the list of the updated online users
2	RTT	the user requests to talk with the user specified in the payload
3	ACCEPT	the user receives an RTT and accept it
4	REFUSE	the user refuses to talk with the user who sent the RTT
5	MESSAGE	the messages with this opcode are the messages exchanged during the chat
6	CC_SESSION_ESTABLISHMENT	send ECDH_PUB_KEY and PUB_KEY to user who accepted the chat request
7	USER_NOT_FOUND	request to unavailable or unknown client

OPCODE#	Name	Meaning
8	NOTIFY	the peer closed the chat

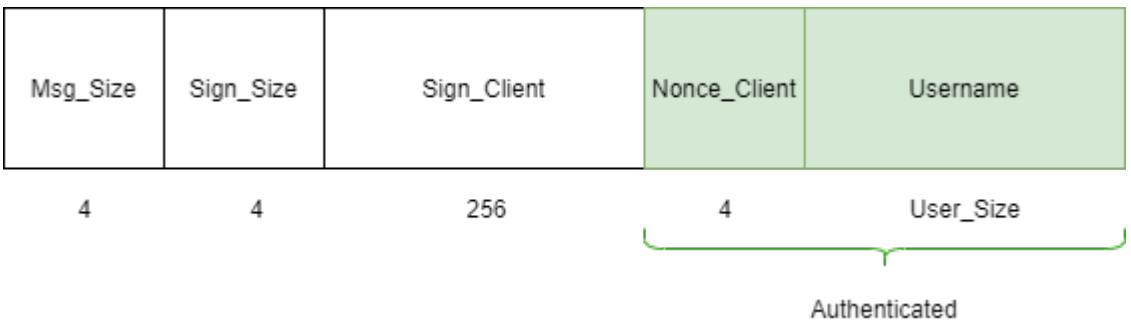
Authentication phase:

Sequence diagram:

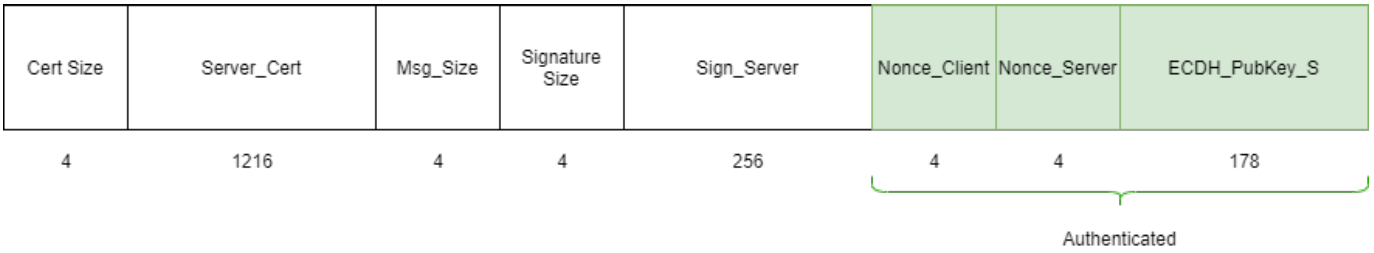
Sequence diagram of the preliminar authenticaation phase:



Message 1.0:



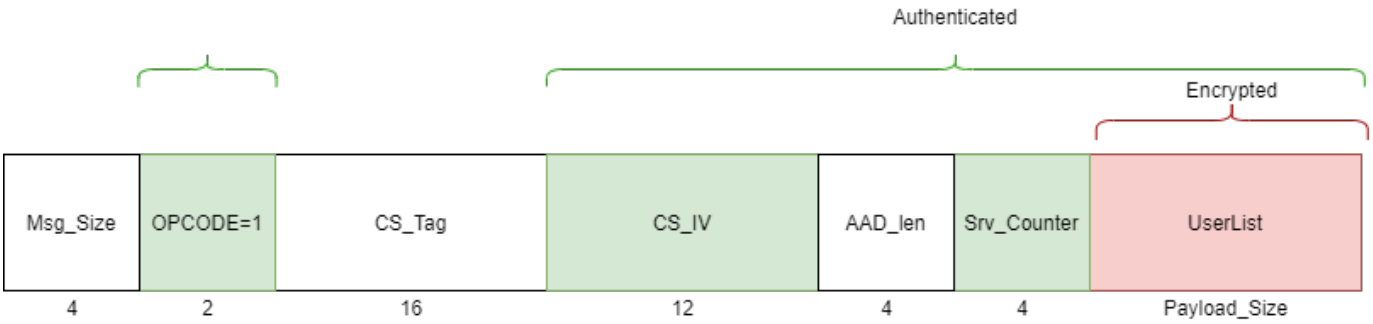
Message 1.1:



Message 1.2:

Message 1.3 OPCODE 1:

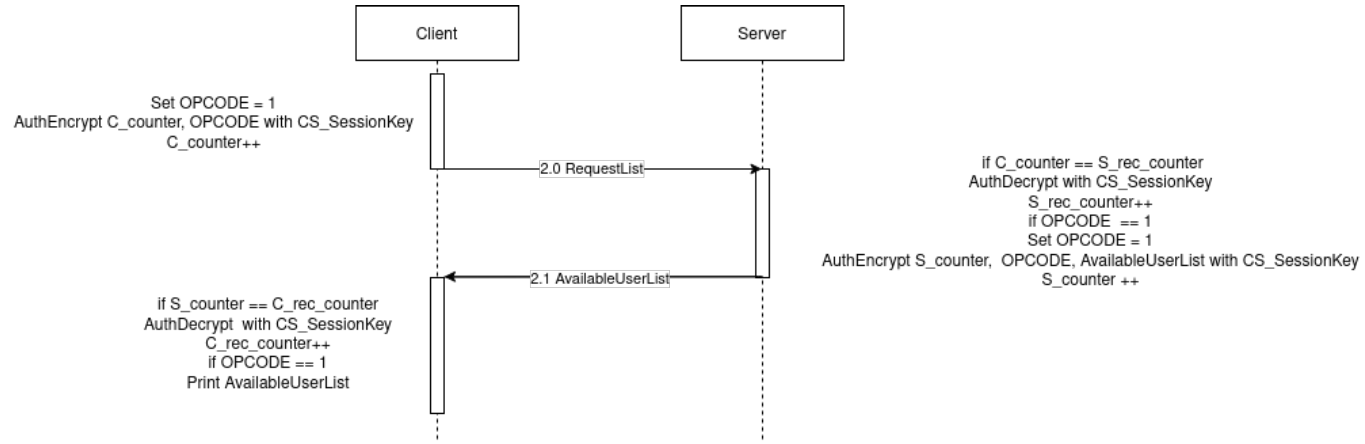
Send available user list



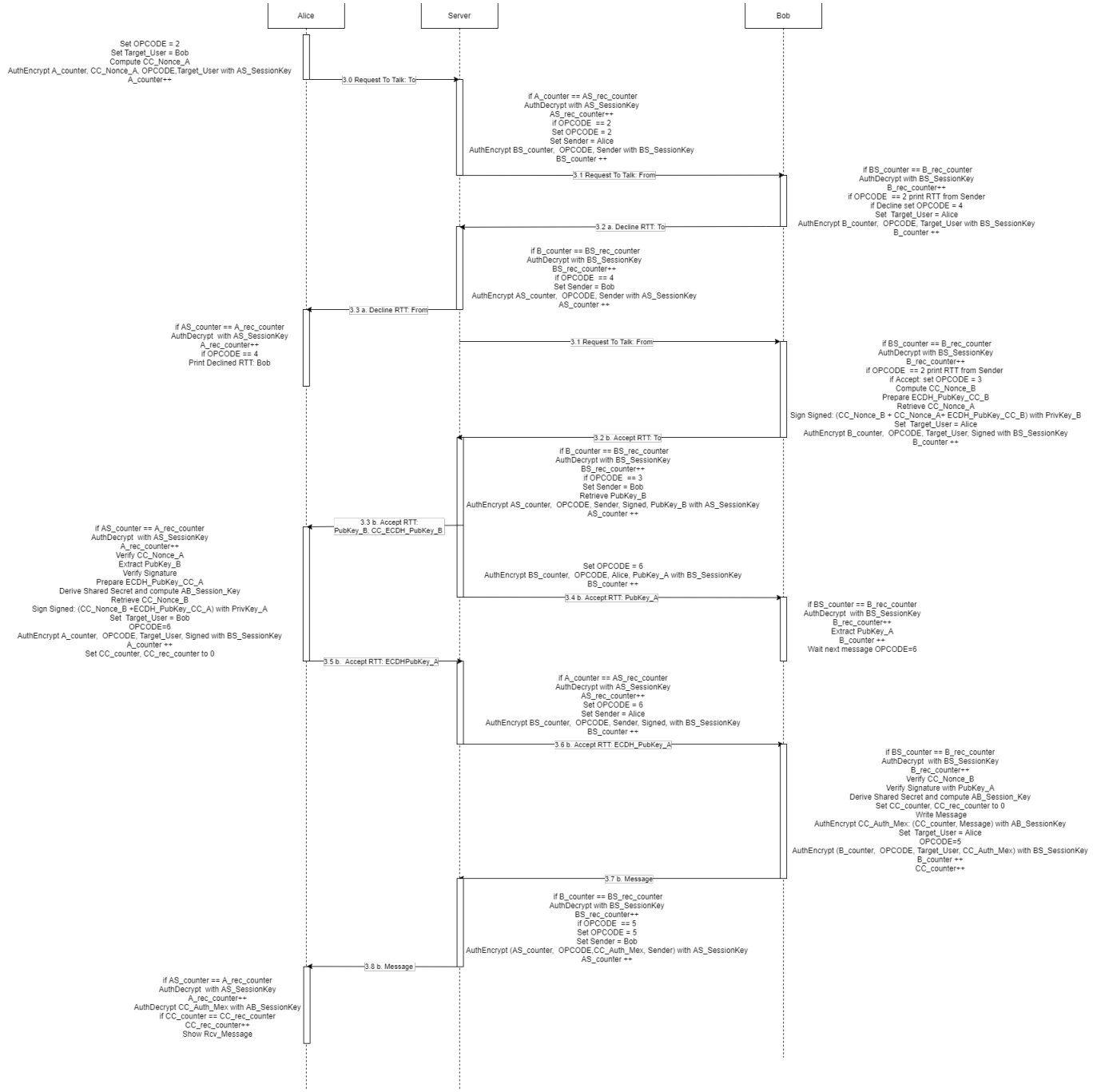
Comunication phase:

Sequence diagram:

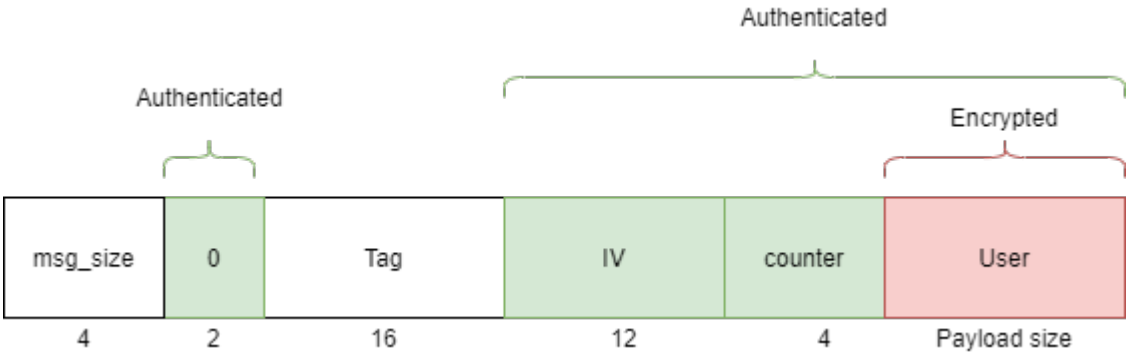
Sequence diagram to request the user list:



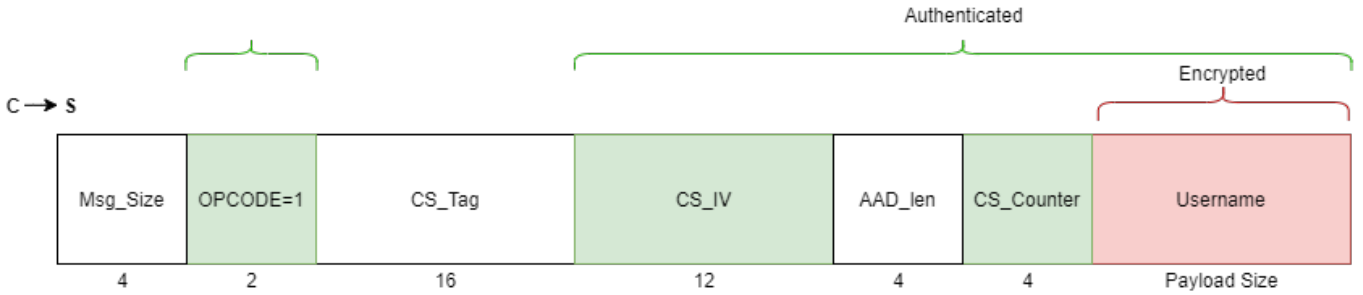
Sequence diagram to send an RTT, accept it or refuse it, then starting the comunication with the other user:

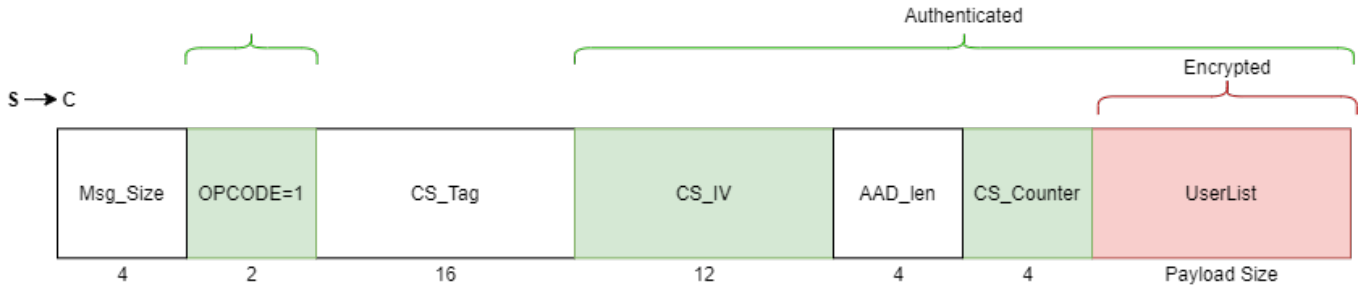


OPCODE 0 QUIT:

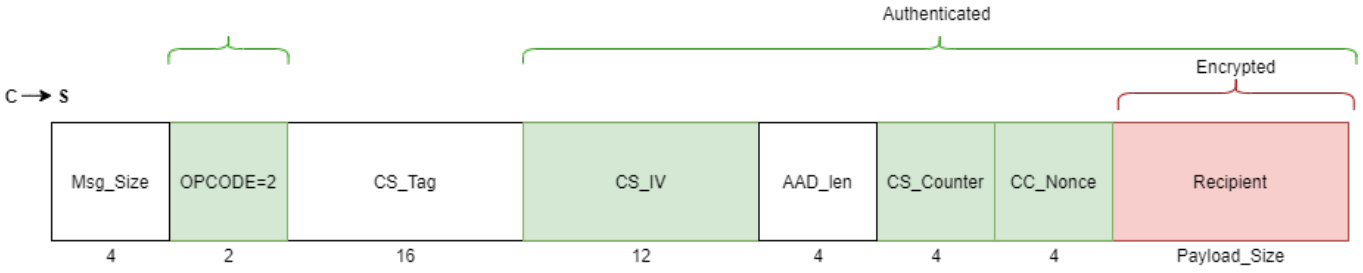


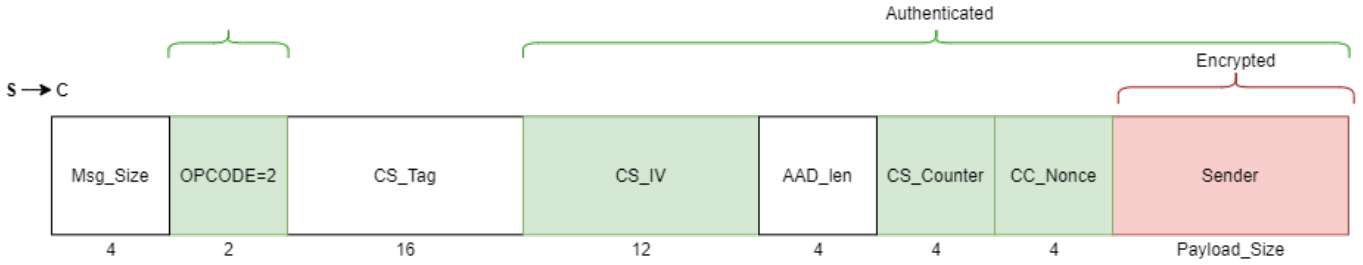
OPCODE 1 LIST:



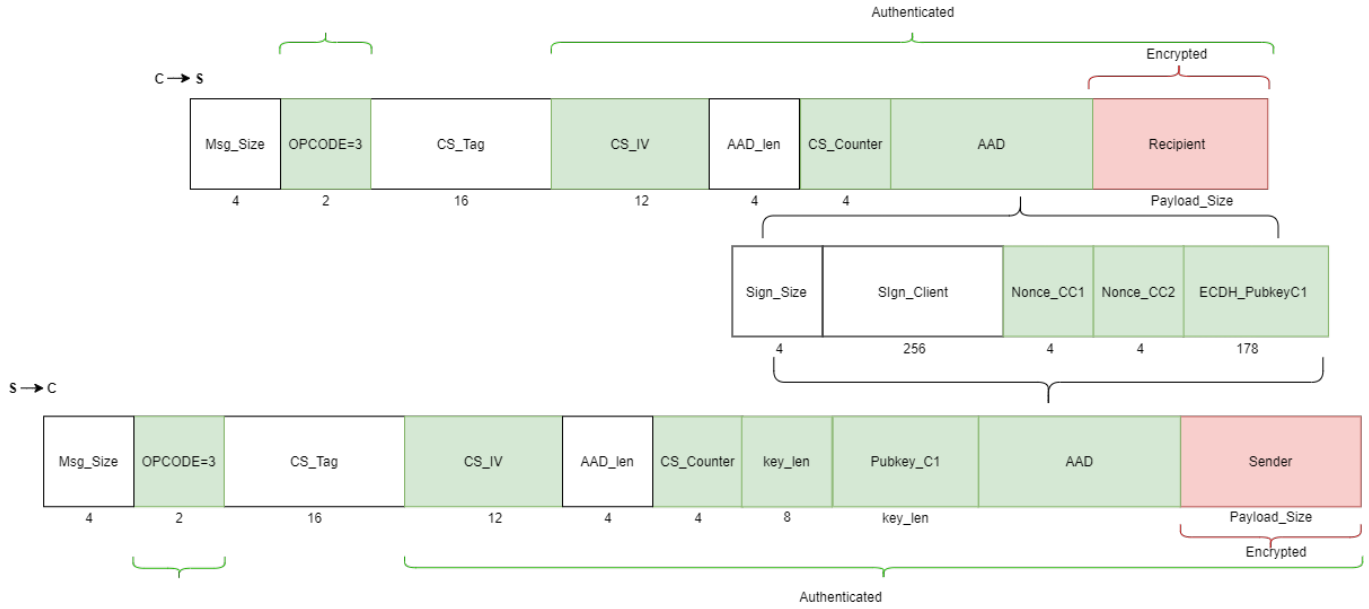


OPCODE 2 RTT:

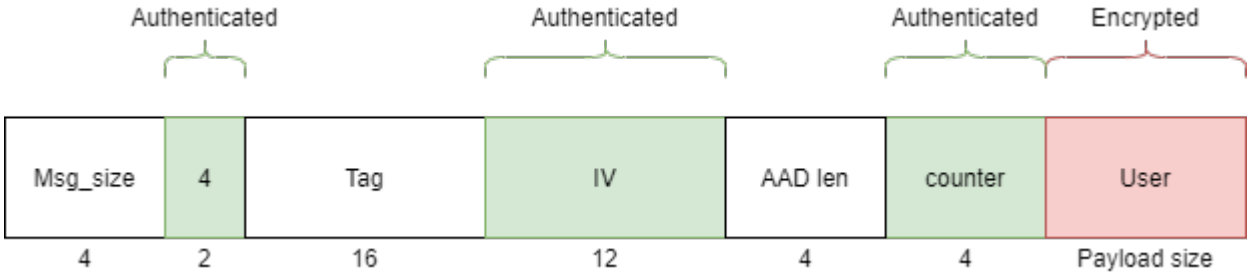




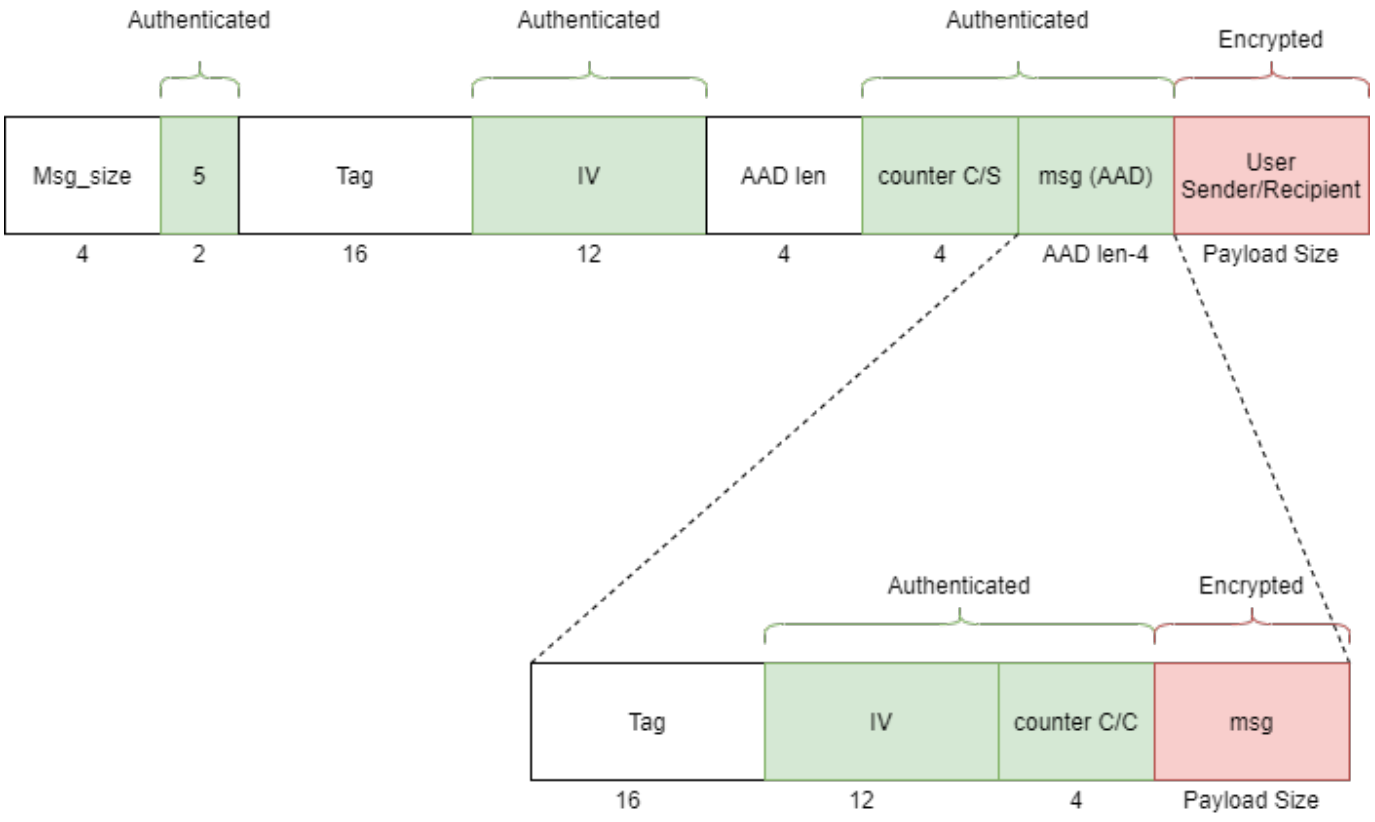
OPCODE 3 ACCEPT:



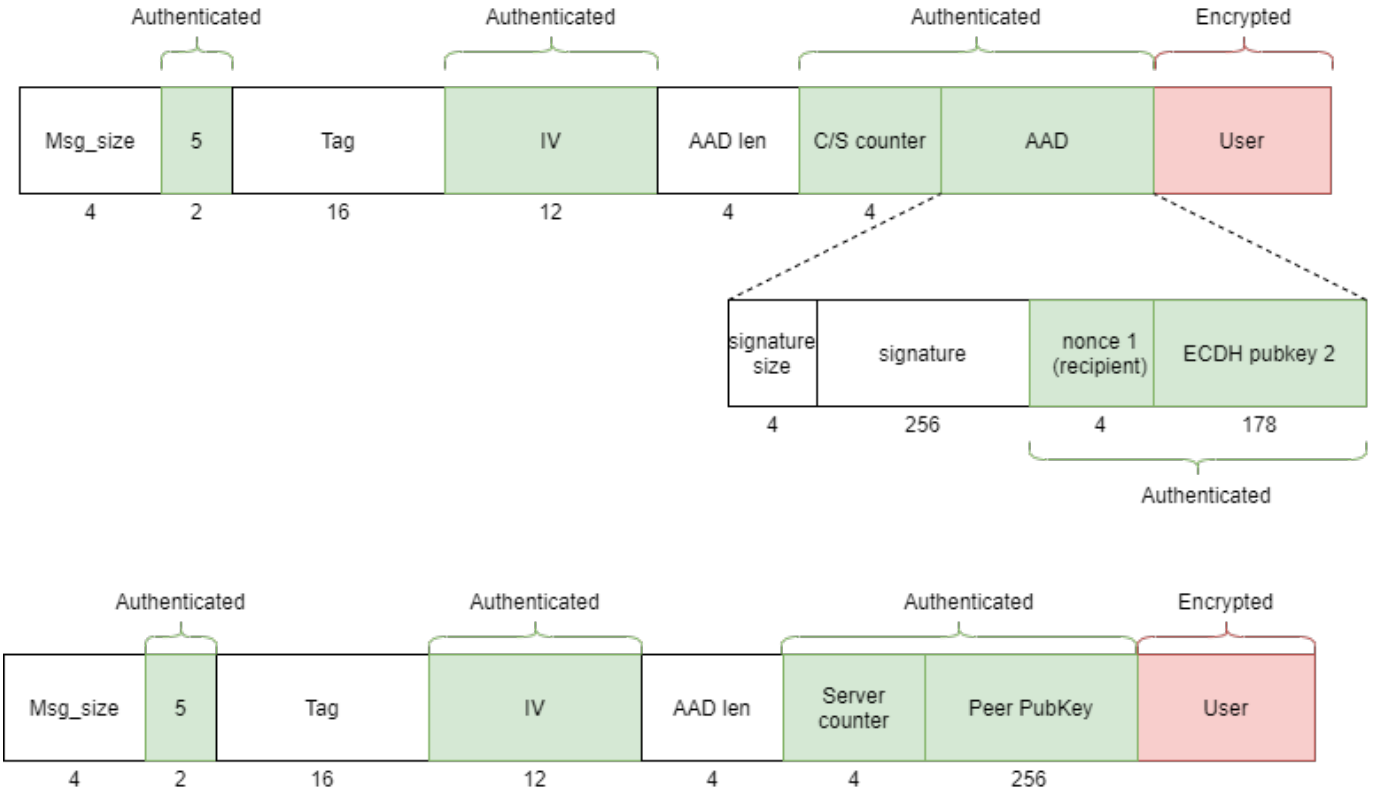
OPCODE 4 REFUSE:



OPCODE 5 MESSAGE:



OPCODE 6 CC_SESSION_ESTABLISHMENT:



Implementation choices

After the connection establishment with the server, the list of the available users and each recipient or sender usernames are encrypted to address privacy concerns of the users.

The opcodes are authenticated to verify the user intentions, but it's not necessary to encrypt the operation type.

Also the counters are authenticated because we need to verify that the message source is the one we are expecting, but it's not considered a sensitive data.

Being public the keys are not encrypted, but only authenticated.

The messages exchanged between two clients are always encrypted.

Run it

To run this project:

- Open a terminal and run the server.

```
g++ -o server server.cpp -lcrypto -pthread
./server 10000
```

- Open more terminals and run the clients.

```
g++ -o client client.cpp -lcrypto -pthread
./client localhost 10000 alice
```