UNIVERSITÀ DI PISA

Foundations of Cybersecurity

The SafeCloud Service

Table of Contents

Introduction and Requirements

The SafeCloud service consists of a simple yet secure cloud-based storage service in which, by connecting to a remote *SafeCloud server*, *authenticated users* are offered functionalities such as uploading or downloading arbitrary files from their reserved space, or *storage pool*, on the server, deleting or renaming existing files as well as listing the contents of their *storage pool* and of their local *download directory*.

Functional Requirements

The service's functional requirements are outlined below:

- The service shall be comprised of two separate modules exchanging information via socket communication:
    - A *SafeCloud Server* application, to be deployed on a system holding the users' storage pool and credentials, which by accepting incoming client connections on a specified port implements functionalities such as establishing a secure communication with multiple *SafeCloud Client* applications, authenticating users within the service and carrying out their desired operations on their private *storage pools*.
    - A *SafeCloud Client* application, by which users can authenticate and establish a secure communication with the specified *SafeCloud Server* endpoint as well as issuing and viewing the results of *application commands* relative to their private *storage pools*.
- Users in the service shall be divided into *unauthenticated users* (or *guests*) and *authenticated users*.
- Each *authenticated user* is assigned, within the *SafeCloud Server* system they are registered in, a unique and private *storage pool* holding their files and not accessible by other users.
- *Unauthenticated users* shall be allowed only to log in by authenticating within the service.
- *Authenticated users* shall be allowed the following *application operations* on the *SafeCloud Server* system they are registered in:
    - Upload arbitrary local files of a size up to 4GB to their storage pool.
    - Download files from their storage pool into their local predefined *download folder*.
    - Delete files from their storage pool after asking for confirmation.
    - Rename files in their storage pool.
    - List the contents of their storage pool.
    - List the contents of their download folder.
    - Log out from the *SafeCloud Server* and terminate the *SafeCloud Client* application.

Non-Functional Requirements

The service's main non-functional requirements are discussed below:

Security

1) The service shall enforce *Confidentiality* and *Origin Authentication* on all the application-level messages associated with the user's operations without relying on a pre-shared secret between the *SafeCloud Client* and *SafeCloud Server* application.
2) Each user of the service shall be associated with the following cryptographic quantities:
    - A long-term RSA key pair on 2048-bit in PEM format encrypted with a user-defined password.
    - The certificate of a root *Certification Authority* (CA)
    - The root CA Certificate Revocation List (CRL).
3) Each *SafeCloud Server* system shall be associated with the following cryptographic quantities:
    - The server's long-term RSA key pair on 2048-bit.
    - The server's certificate signed by the root *Certification Authority* (CA).
    - The RSA public key on 2048-bit of each user registered within the system.
4) In establishing a secure communication, users and the *SafeCloud Server* application shall authenticate with one another by:
    - The *SafeCloud Server* shall provide its certificate released by the root CA, which shall be automatically validated by the *SafeCloud Client* application using the root CA's certificate and certificate revocation list.
    - The *SafeCloud Client* shall provide the server the user's name (*username*) and a challenge's response signed with the user's private RSA key, which shall be verified by the server through the public RSA key associated with such username.
5) A custom *Key Exchange Protocol* aimed at exchanging a shared symmetric *session key* between the *SafeCloud Client* and *Server* applications shall be devised.
6) The secure communication shall provide protection against replay attacks.
7) The secure communication shall provide *Perfect Forward Secrecy (PFS)*, meaning that the compromising of the user's or server's long-term private RSA keys must not compromise the confidentiality of past sessions.
8) Secure coding principles shall be enforced to prevent vulnerabilities allowing for undesired behaviours, such as but not limited to accessing or manipulating the private storage pool of other users or uploading files of size greater than 4GB.

Reliability

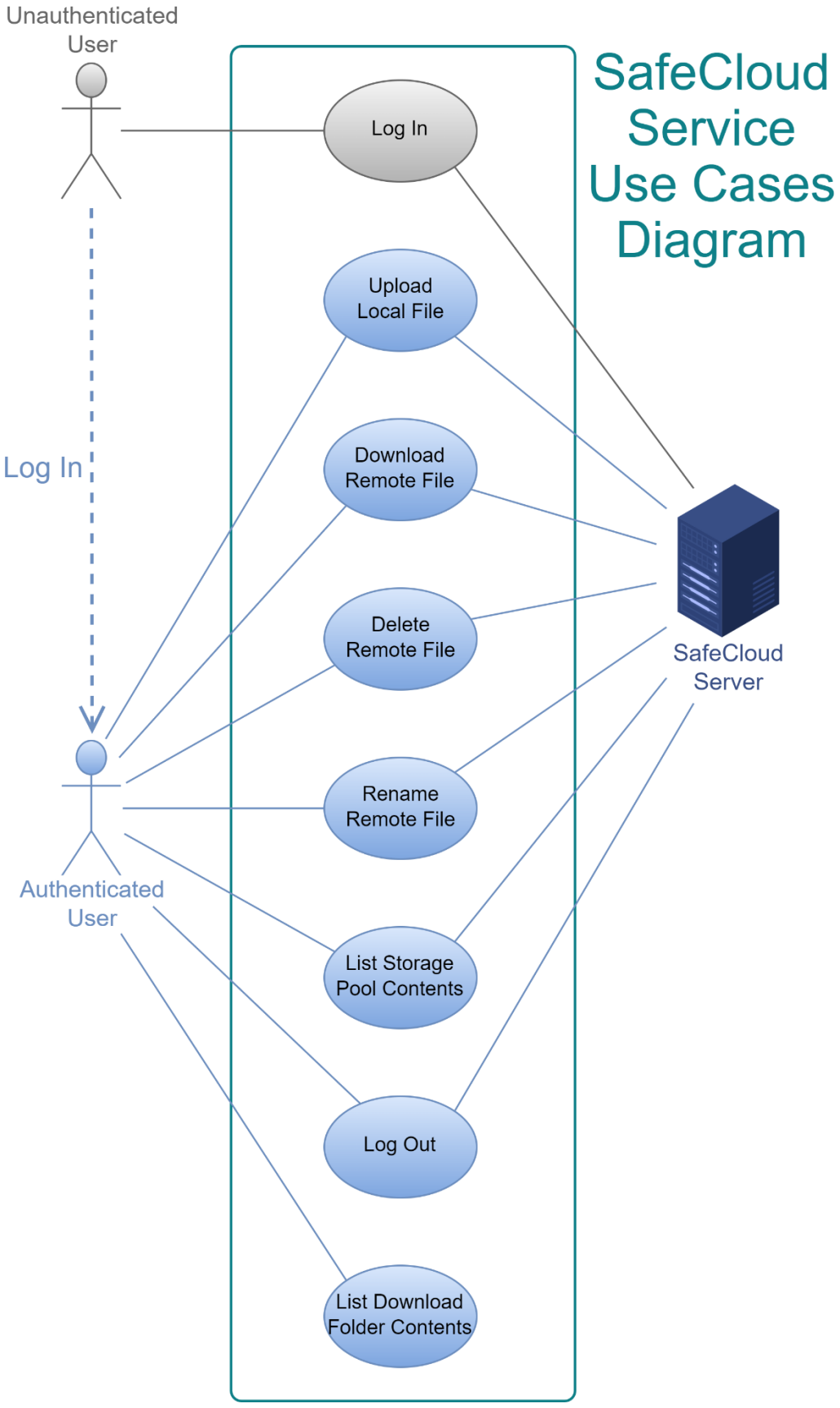
9) All erroneous conditions in the service execution should be handled so as to enforce a consistent state between the *SafeCloud Client* and *Server* application, allowing if possible the transparent resumption of the application's functionalities.

Usability

10) The *SafeCloud Client* application shall offer its functionalities via an intuitive interface requiring no IT expertise in its use.

Use Cases Diagram

From the analysis of its functional requirements the service's actor and use cases have been outlined in the diagram below:



General Design Choices

The general design choices and underlying hypotheses of the service that have been developed are outlined below:

1) The service was developed in the C++14 language to enable for an object-oriented design as well as for the versatile error handling technique represented by the exceptions mechanism.
2) Being the service innately I/O bound and lacking specific performance requirements, in their first release both the *SafeCloud Client* and *Server* were developed as single-threaded applications, with requests from multiple clients being handled on the latter via I/O multiplexing mechanisms.
3) In view of the considerable number of OS-specific parameters in accessing the file system (e.g. path traversal modes, maximum file names length, metadata format, etc.) in their first release both the *SafeCloud Client* and *Server* application have been developed to be executed in Linux-based environments only.
4) From a security standpoint:
    - The system the *SafeCloud Server* is deployed in is considered a *trusted security domain*, with security mechanisms for preventing the access or manipulation of the server or user files not being taken into account, while both the user input and messages sent by the *SafeCloud Client* application are considered as *tainted*, with sanitization techniques being employed by both the *SafeCloud Client* and *Server* applications.
    - Mechanisms for preventing the access or manipulation of the user's private files from outside the *SafeCloud Server* and *Client* applications are not taken into account.
5) While actually implemented through a challenge-response mechanism based on their long-term RSA key pair, from the standpoint of users of the *SafeCloud Client* applications authentication is carried out by providing:
    - Their case-insensitive *username*, which must match a registered username with its associated public RSA key within the *SafeCloud Server* instance to connect to, and, in terms of constraints, it:
        - Cannot be empty.
        - Must be at most 30 characters long.
        - Can contain only letters (a-z, A-Z), numbers (0-9) and the '_' character.
        - Must start with a letter (a-z, A-Z).
    - Their case-sensitive *password*, corresponding to the password allowing the decryption of their private RSA key file encrypted using the AES_128_CBC cipher, whose characters should be hidden during inputs and, in terms of constraints:
        - Cannot be empty.
        - Must be at most 30 characters long.
6) A user registration mechanism conceptually based on retrieving their public RSA key via a certificate and initializing their storage pool has been postponed to future application versions.
7) In addition to their cryptographic quantities outlined in the application's requirements, once authenticated within the service users are associated with:
    - Their *main directory*, corresponding to the local *download directory* on the *SafeCloud Client* and their *storage pool* on the *SafeCloud Server* application.
    - A *temporary directory*, which is used to store the partial files being uploaded or downloaded, which are renamed and moved to the associated *main directory* once the file transfer has completed.

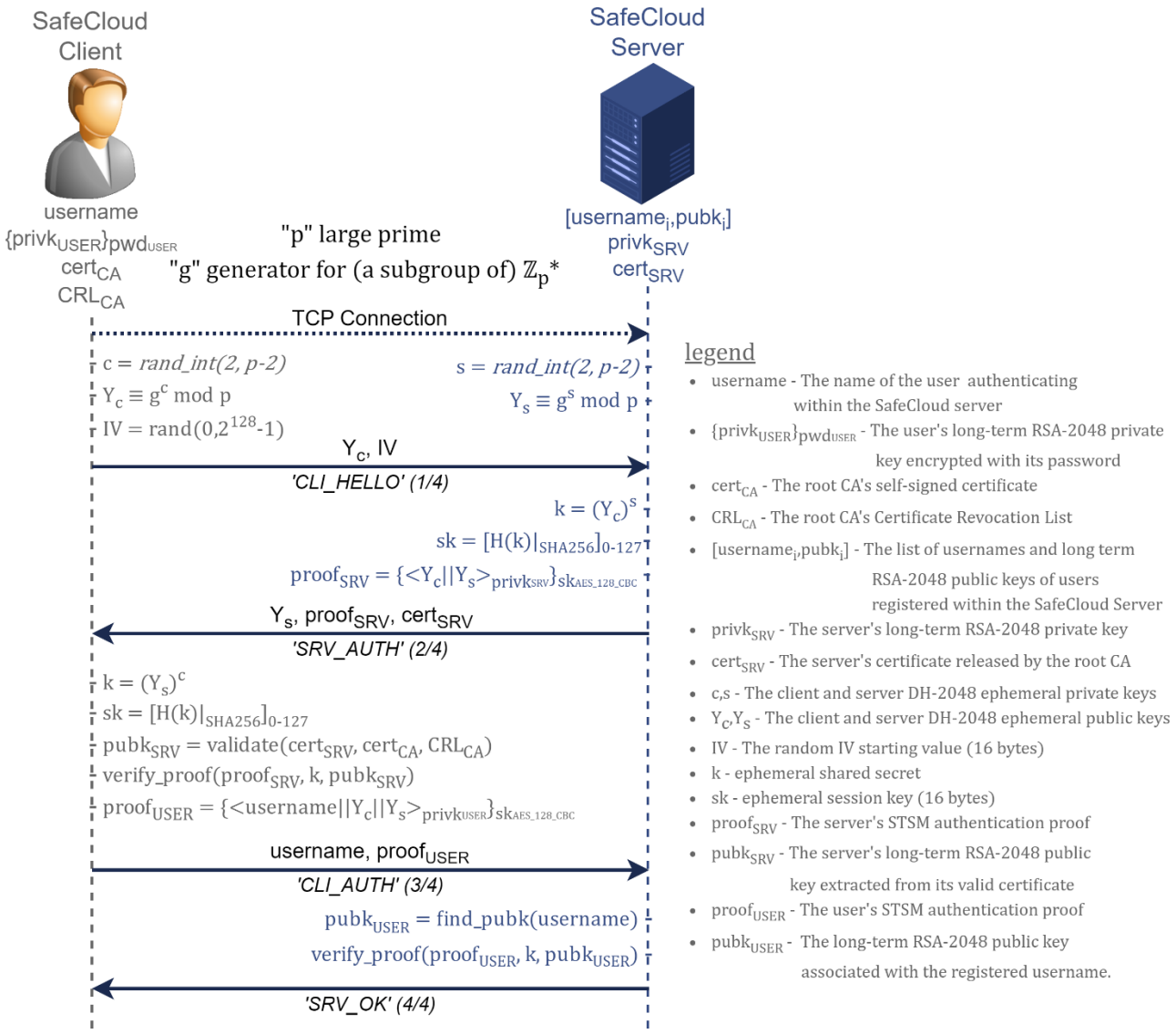
8) Once a TCP/IP connection has been established, the service execution and communication between the *SafeCloud Client* and *Server* application has been divided into two main phases:
    - A *Key Establishment and Authentication Phase*, which is carried out by the two applications performing an ad-hoc protocol termed *Station-to-Station Modified (STSM)*, which is thoroughly described later.
    - A *Session Phase*, in which the *authenticated user* can issue and view the result of *application commands* regarding their storage pool, with the associated messages exchanged by the *Client* and *Server* application being both encrypted and authenticated.
9) *Confidentiality* in the data exchanged between the *SafeCloud Client* and *Server* is obtained by encrypting and decrypting it via the *AES_128* cipher, which is used in different *encryption modes* in the two service phases:
    - In *CBC mode* in the *Key Establishment and Authentication Phase*.
    - In *GCM mode* in the *Session Phase* so as to also enforce the *integrity* of the exchanged data.
10) The *AES_128 encryption modes* used to exchange encrypted data between the *SafeCloud Client* and *Server* require a 16-byte (128 bit) shared *Initialization Vector (IV)*, whose initial random value is established at the start of the *Key Establishment and Authentication Phase* and is kept synchronized by the two parties by autonomously incrementing its value for each encryption or decryption operation.
11) The users' storage pools on the *SafeCloud Server* system consist of a single directory, with files being uploaded or downloaded via their *name*, with the capability of defining sub-directories within a storage pool being postponed to future service developments.
12) In addition to their names and possibly contents, all provided *operations* also manage the files' *metadata*, including:
    - Their *size* in bytes.
    - Their *last modified time*, which is mirrored from their source into uploaded or downloaded files.
    - Their *creation time*, which unlike the *last modified* time cannot be mirrored from their source into uploaded or downloaded files as such an API is not provided by the Linux kernel.
13) Checks to prevent transferring the same version or overwriting a more recent version of a file have been provided when uploading or downloading a file.
14) In addition to *authenticated users* explicitly logging out from the *SafeCloud Server* they are logged in, an action triggering the shutdown of their *SafeCloud Client* application, both the *SafeCloud Server* and *SafeCloud Client* applications are capable of handling the subset of Linux inter-process signals aimed at interrupting or terminating a process (*SIGINT*, *SIGTERM*, *SIGQUIT*), whose reception triggers an orderly shutdown procedure which is described later.

Station-to-Station Modified (STSM) Protocol

As previously introduced, once a TCP/IP connection has been established, the *SafeCloud Client* and *Server* applications execute via message exchange a *Key Establishment and Authentication* phase, whose purposes include:

1) Establishing the initial random value of the 16-byte (128 bit) *Initialization Vector (IV)* to be used for encrypting and decrypting data.
2) Establishing a shared ephemeral *Session Key* so as to meet the *Perfect Forward Secrecy (PFS)* requirement.
3) Authenticating the user and the server with one another, which is accomplished by exchanging and verifying *proofs* based on their long-term RSA key pairs, preventing on the one hand *Man-in-the-Middle (MIM)* attacks and on the other acting as an implicit *Log In* mechanism.

Such operations are carried out by the two applications by executing an ad-hoc protocol termed *Station-to-Station-Modified (STSM)* protocol, consisting in a custom variant of the *Station-to-Station (STS)* key agreement protocol that, in addition to establishing a shared DH-2048 ephemeral *Session Key* and providing the *Key Confirmation* property, has been extended so as to also establish the initial IV value as well as incorporate the user's *Log In* functionalities, a protocol whose nominal execution and message exchange is outlined below:



The execution of the STSM protocol can be logically divided into the following steps associated with the messages being exchanged:

0) Preliminaries

Once a TCP/IP connection has been established, both the *SafeCloud Client* and *Server* application generate an ephemeral DH-2048 key pair $\{c, Y_C\}, \{s, Y_S\}$, with the client also randomly generating the 16-byte "IV" starting value.

1) 'CLI_HELLO' message

The client sends the server the 'CLI_HELLO' STSM message, containing:

1- Its ephemeral DH-2048 public key "Y_C".

2- The starting random "IV" value.

With the server using the ephemeral public key "Y_C" to derive the ephemeral shared secret "k" and the resulting shared ephemeral key "sk" as the first 16 bytes of its SHA-256 hash.

2) 'SRV_AUTH' message

The server prepares its *STSM authentication proof* consisting of the concatenation of the two actors' ephemeral DH-2048 public keys "$Y_C \,||\, Y_S$":

1. Signed with the server's long-term RSA-2048 private key "$privk_{SRV}$" so as to authenticate the *SafeCloud Server* with respect to the client.
2. Encrypted with the AES-128 cipher in CBC mode using the resulting ephemeral session key "sk" and the "IV" starting value so as to provide the *key confirmation* property.

$$proof_{SRV} = \{< Y_C \,||\, Y_S >_{privk_{SRV}}\}_{sk_{AES\_128\_CBC}}$$

The server then sends the client the 'SRV_AUTH' STSM message, containing:

1- Its ephemeral DH-2048 public key "Y_S".

2- Its STSM authentication proof "$proof_{SRV}$".

3- Its certificate signed by the root CA "$cert_{SRV}$".

Once received, such message is validated by the *SafeCloud Client* by:

1. Verifying the server's certificate to belong to the *SafeCloud Server*, to have been signed by the root CA and to have not been revoked.
2. Extracting from the server's certificate its public key "$pubk_{SRV}$"
3. Using the ephemeral public key "Y_S" to derive the ephemeral shared secret "k" and the resulting shared ephemeral key "sk" as the first 16 bytes of its SHA-256 hash.
4. Verifying the server's STSM proof by decrypting it using the ephemeral key "sk" and the starting "IV" value and verifying the signature via the server's long-term RSA-2048 public key "$pubk_{SRV}$".

3) 'CLI_AUTH' message

The client prepares its *STSM authentication proof* consisting of the concatenation of its username and the two actors' ephemeral DH-2048 public keys "$username \,||\, Y_C \,||\, Y_S$":

1. Signed with the client's long-term RSA-2048 private key "$privk_{USER}$" so as to authenticate and *Log In* the user with respect to the *SafeCloud Server*.
2. Encrypted with the AES-128 cipher in CBC mode using the resulting ephemeral session key "sk" and the incremented "IV" value so as to provide the *key confirmation* property.

$$proof_{USER} = \{< username \,||\, Y_C \,||\, Y_S >_{privk_{USER}}\}_{sk_{AES\_128\_CBC}}$$

The client then sends the server the 'CLI_AUTH' STSM message, containing:

1- The user's name "$username$".

2- The user's authentication proof "$proof_{USER}$".

Once received, such message is validated by the *SafeCloud Server* by:

1. Asserting that a user with such "*username*" is registered within the server and retrieving its associated long-term RSA-2048 public key "$pubk_{USER}$".
2. Verifying the user's STSM proof by decrypting it using the ephemeral key "sk" and the starting "IV" value and verifying the signature via the client's long-term RSA-2048 public key "$pubk_{SRV}$".

   It should also be noted that the user's STSM proof verification:

   - Prevents an adversary from impersonating a user as, even by correctly carrying out the previous protocol steps, without knowing the user's private key "$privk_{USER}$" it is impossible to forge a valid user STSM proof signature.
   - Protects against replay attacks, as the server's ephemeral DH-2048 public key Y_S is randomly generated for every new client connection, preventing the reuse of a previous valid user STSM proof signature.

4) 'SRV_OK' message

In the case of a successful user authentication the *SafeCloud Server* notifies the *SafeCloud* Client of the STSM protocol completion by sending a 'SRV_OK' STSM message with no payload.

While computationally secure regardless of the timing of its execution, the STSM protocol was provided with the additional security measure of the *SafeCloud Server* enforcing the *SafeCloud Client* messages ('CLI_HELLO', 'CLI_AUTH') to be provided in a timely manner, i.e. within a maximum delay from the moment the TCP connection was established or the 'SRV_AUTH' message was sent, a constraint further precluding online attacks based on traffic analysis techniques carrying no performance penalty.

STSM Messages Structure

STSM messages exchanged in the protocol are divided into:

- A *header*, comprised of:
  - A *"msgLen"* field of 2 bytes storing the message's total length in bytes, information that is used on the receiving side to wait for the full message to have been received before parsing it as well as a means to assert the validity of received STSM message types of constant length.
  - A *"msgType"* field of 1 byte storing the STSM message type.
- *(optional)* A *payload,* whose contents and length depend on the STSM message type and its total length in bytes.

The STSM message types are logically divided into:

- *STSM handshake messages*, which include the previously described four messages exchanged in a successful STSM execution.

STSM Handshake Message Types					
Type	**Description**	**Direction**	**Payload**		
			Attribute	**Description**	**Size (B)**
CLI_HELLO	STSM protocol start	*Client → Server*	Y_C	Client's ephemeral DH-2048 public key	1194
			IV	Initialization Vector initial random value	16
SRV_AUTH	Server STSM authentication message	*Client ← Server*	Y_S	Server's ephemeral DH-2048 public key	1194
			$proof_{SRV}$	Server's STSM authentication proof	272
			$cert_{SRV}$	Server's certificate	*variable*
CLI_AUTH	User STSM authentication message	*Client → Server*	*username*	User's name	*variable*
			$proof_{USER}$	User's STSM authentication proof	272
SRV_OK	STSM protocol completion	*Client ← Server*	—		

- STSM *error messages*, which have no payload and are sent in case of errors in the protocol's execution, causing its abortion and the disconnection between the *SafeCloud Client* and *Server* applications.

STSM Error Message Types		
Type	**Description**	**Direction**
ERR_CLI_TIMEOUT	The server received a STSM message after the maximum delay	*Client ← Server*
ERR_INVALID_PUBKEY	The party received an invalid DH-2048 ephemeral public key	*Client ↔ Server*

ERR_SRV_AUTH_FAILED	The server failed the STSM authentication	*Client → Server*
ERR_SRV_CERT_REJECTED	The client rejected the Server's certificate	*Client → Server*
ERR_CLIENT_LOGIN_FAILED	The server has no registered user with such *username*	*Client ← Server*
ERR_CLI_AUTH_FAILED	The client failed the STSM authentication	*Client ← Server*
ERR_UNEXPECTED_MESSAGE	The party received an out-of-order STSM handshake message	*Client ↔ Server*
ERR_MALFORMED_MESSAGE	The party received a malformed STSM message	*Client ↔ Server*
ERR_UNKNOWN_STSMMSG_TYPE	The party received a STSM message of unknown type	*Client ↔ Server*

Service Session Phase

Should the *Key Establishment and Authentication Phase* complete successfully, the service's execution switches to the *Session Phase*, in which, as previously introduced, users can issue and view the results of the provided *application operations*, which, apart from listing the contents of the user's download directory, are carried out through the exchange of *Session Messages* and *Session Raw Data* between the *SafeCloud Client* and *Server* applications.

Session Messages Structure

In a similar fashion to STSM messages, *session* messages consist of:

- A *header*, comprised of:
  - A *"msgLen"* field of 2 bytes storing the message's total length in bytes, information that is used on the receiving side to wait for the full message to have been received before parsing it as well as a means to assert the validity of received session message types of constant length.
  - A "msgType" field of 1 byte storing the session message type.
- *(optional)* A *payload,* whose contents and length depend on the session message type and its total length in bytes.

The session message types are logically divided into:

- *Payload Session Message Types*, which are session messages carrying a payload:

Payload Session Message Types					
Type	**Description**	**Direction**	**Payload**		
			Attribute	**Description**	**Size (B)**
FILE_UPLOAD_REQ	*UPLOAD* operation start	*Client → Server*	*fileName*	The name of the file to be uploaded	*variable*
			fileMeta	The metadata of the file to be uploaded	*24*
FILE_DOWNLOAD_REQ	*DOWNLOAD* operation start	*Client → Server*	*fileName*	The name of the file to be downloaded	*variable*
FILE_DELETE_REQ	*DELETE* operation start	*Client → Server*	*fileName*	The name of the file to be deleted	*variable*
FILE_RENAME_REQ	*RENAME* operation start	*Client → Server*	*oldFileNameLen*	The old file name length (B)	8
			oldFilename	The old file name	*variable*
			newFilename	The new file name	*variable*
FILE_EXISTS	The storage pool contains a file of such *name*	*Client ← Server*	*fileMeta*	The *metadata* of the file with such *name* in the storage pool	*24*
POOL_SIZE	Storage pool serialized size	*Client ← Server*	*poolInfoSize*	Storage pool information size	*32*

- *Non-error Signaling Session Message Types*, representing session messages without a payload that are not used for notifying errors:

Non-error Signaling Session Message Types		
Type	**Description**	**Direction**
FILE_LIST_REQ	*LIST* operation start	*Client* → *Server*
FILE_NOT_EXISTS	The storage pool does not contain a file of such *name*	*Client* ← *Server*
CONFIRM	Operation confirmation notification	*Client* → *Server*
CANCEL	Operation cancellation notification	*Client* → *Server*
COMPLETED	Operation completion notification	*Client* ↔ *Server*
BYE	Graceful disconnection notification	*Client* ↔ *Server*

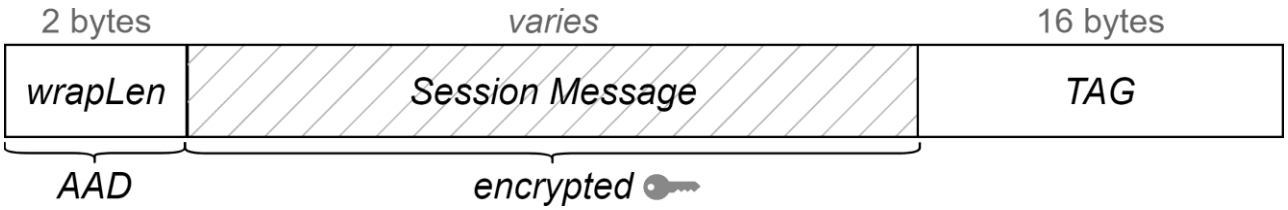
- *Error Signaling Session Message Types*, representing session messages without a payload that are used for notifying errors:

Error Signaling Session Message Types		
Type	**Description**	**Direction**
ERR_INTERNAL_ERROR	The party experienced an internal error	*Client* ↔ *Server*
ERR_UNEXPECTED_SESS_MESSAGE	The party received a session message type invalid for its current *operation* and *step*	*Client* ↔ *Server*
ERR_MALFORMED_SESS_MESSAGE	The party received a malformed session message	*Client* ↔ *Server*
ERR_UNKNOWN_SESSMSG_TYPE	The party received a session message of unknown type	*Client* ↔ *Server*

Unlike *STSM error messages*, sending or receiving *Error Signaling Session Messages* does not trigger the disconnection between the *SafeCloud Client* and *Server* but just the abortion of the current *operation*, with the only exception of the "*ERR_UNKNOWN_SESSMSG_TYPE*", whose error is to be attributed to a desynchronization between the *Client* and *Server*'s IVs requiring in turn their connection to be reset.

Session Message Wrappers

To enforce their *confidentiality* and *integrity*, session messages are exchanged between the *SafeCloud Client* and *Server* by encrypting and decrypting them, using the *AES_128_GCM* cipher with the *session key* and the current *IV* value, to and from their associated *Session Message Wrappers*, which are comprised of:



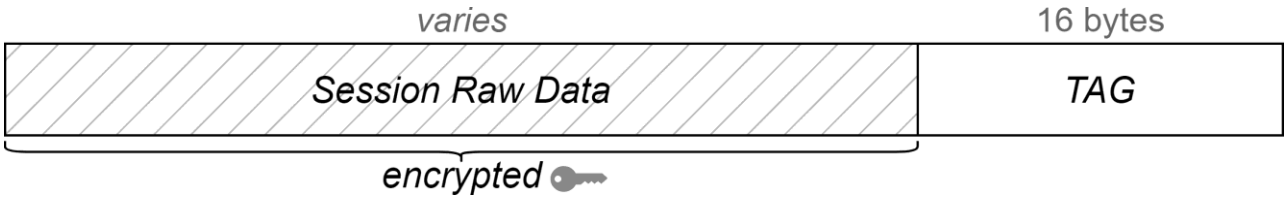
- A *"wrapLen"* field of 2 bytes storing the wrapper's total length in bytes, which again is used to enable the recipient to wait for the full session message wrapper to have been received before parsing it, information that is used as *Associated Authenticated Data (AAD)* for the purposes of generating the wrapper's *TAG*.
- The encrypted *Session Message* to be sent.
- The wrapper's *TAG*, which is used by the receiver to verify its integrity.

Session Raw Data

In addition to session messages in some *operations* the *SafeCloud Client* and *Server* also exchange *raw data*, which may consist of:

- For the *UPLOAD* and *DOWNLOAD* operations, the contents of the file being uploaded or downloaded.
- For the *LIST* operation, the information on the contents of the user's *storage pool*, comprising the *name* and *metadata* of all its files.

Raw data that again is exchanged by encrypting and decrypting it using the *AES_128_GCM* cipher with *session key* and current *IV* value and that is appended with the resulting *TAG* for verifying its integrity on the recipient.



Session Operations

Apart from listing the contents of the user's *download directory*, the execution of all other *session operations* can be modelled via two *Finite State Machines (FSM)*, one on the *SafeCloud Client* and the other on the *SafeCloud Server*, which are kept synchronized through the exchange of *Session Messages* and *Raw Data*, and are characterized by:

- The current *operation* in progress, if any:

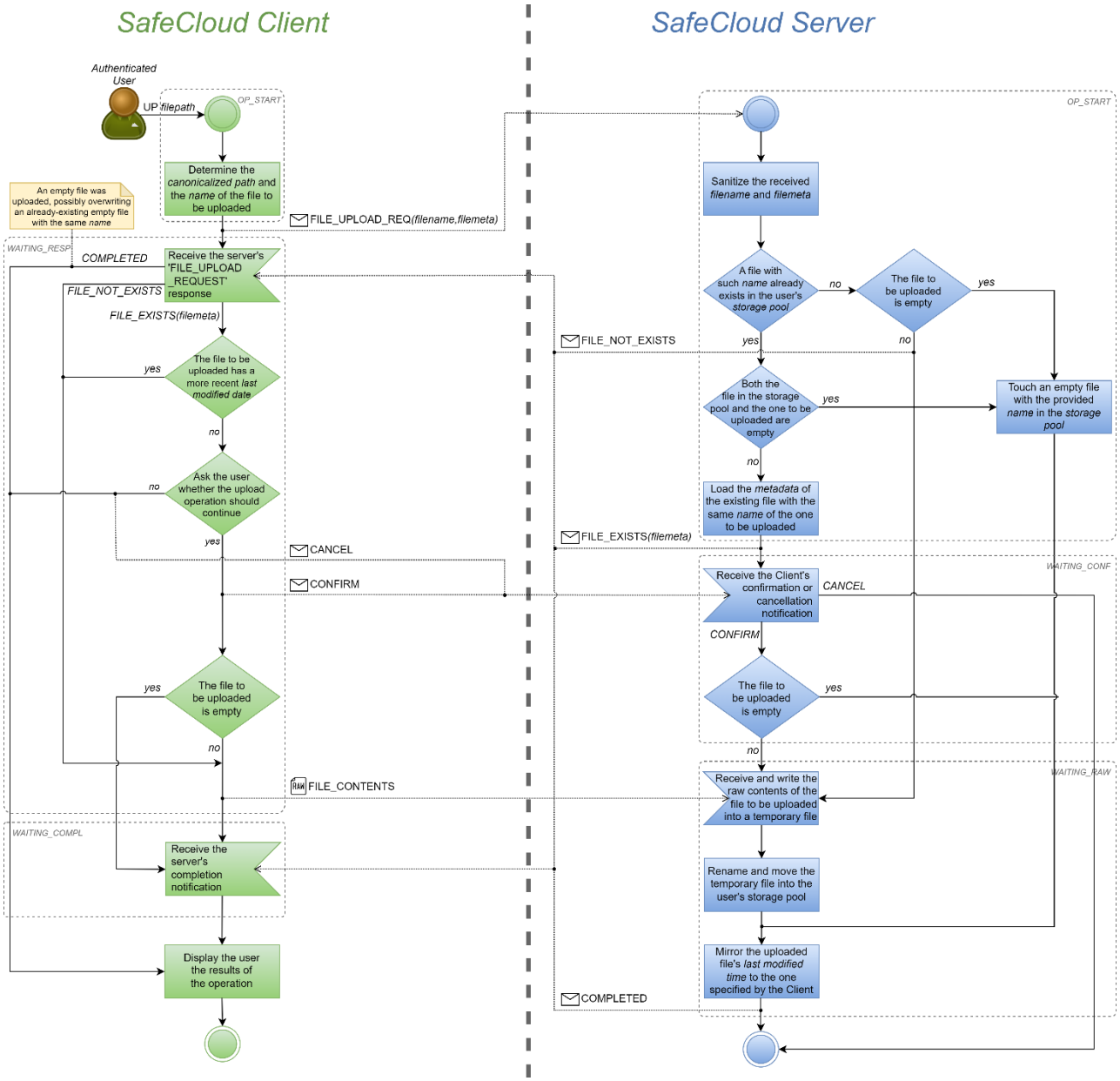
Session Operations	
Operation	**Description**
IDLE	No operation
UPLOAD	Uploading a file from the *SafeCloud Client* to the user's *storage pool* in the *SafeCloud Server*
DOWNLOAD	Download a file from the user's *storage pool* in the *SafeCloud Server* to the user's *download directory* in the *SafeCloud Client*
DELETE	Deleting a file from the user's *storage pool* in the *SafeCloud Server*
RENAME	Renaming a file in the user's *storage pool* in the *SafeCloud Server*
LIST	Listing the contents of the user's *storage pool* in the *SafeCloud Server*

- The current operation *step*, or sub-state within the operation:

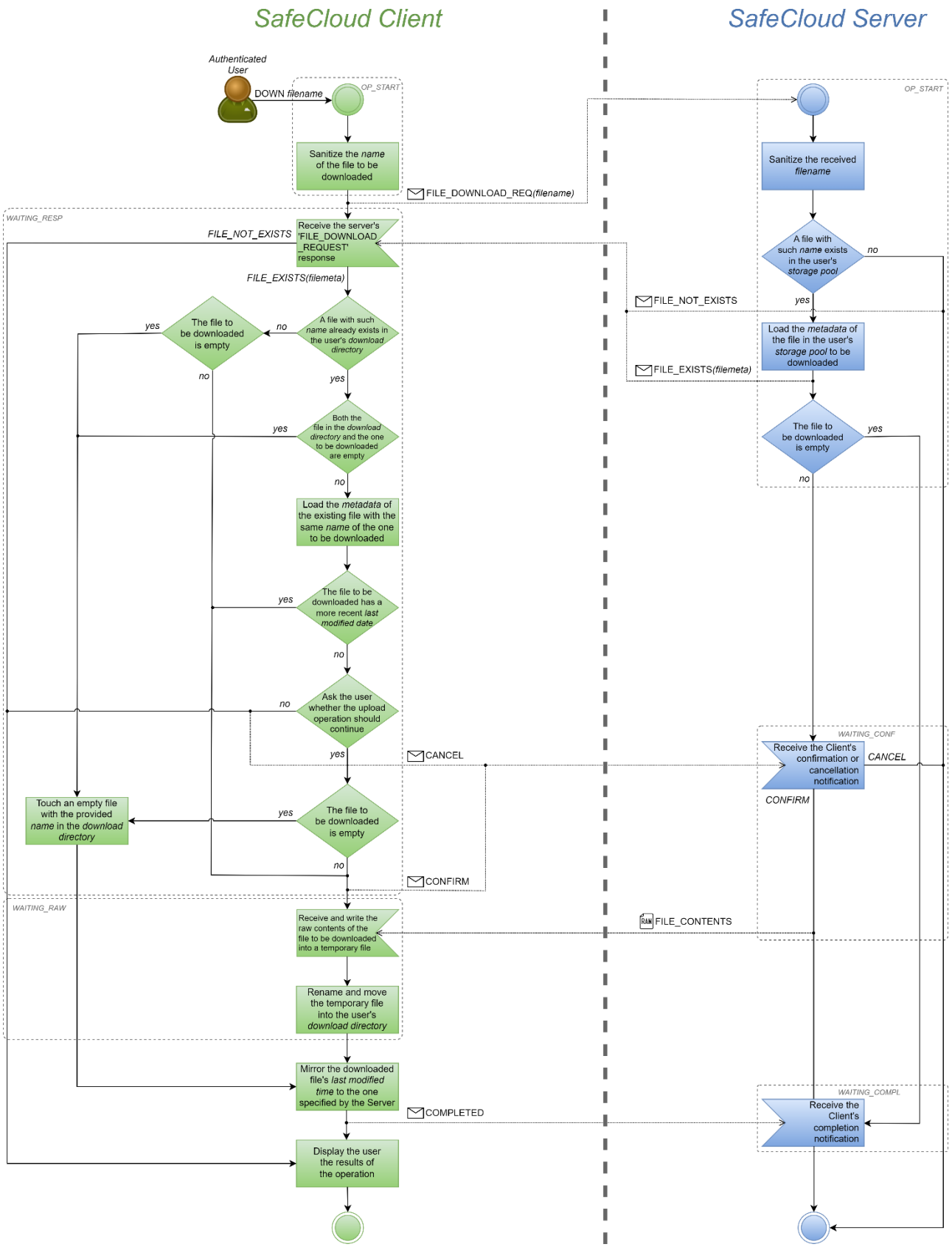
Session Operations Steps		
Step	**Description**	**Used By**
OP_START	Operation Start	*Client, Server*
WAITING_RESP	Waiting for the *SafeCloud Server* first response	*Client*
WAITING_CONF	Waiting for the *SafeCloud Client* confirmation or cancellation notification	*Server*
WAITING_RAW	Waiting for the other party's raw data	*Client, Server*
WAITING_COMPL	Waiting for the other party's completion notification	*Client, Server*

The simplified finite state machines and data exchange associated with successful executions of each *session operation* is presented below:

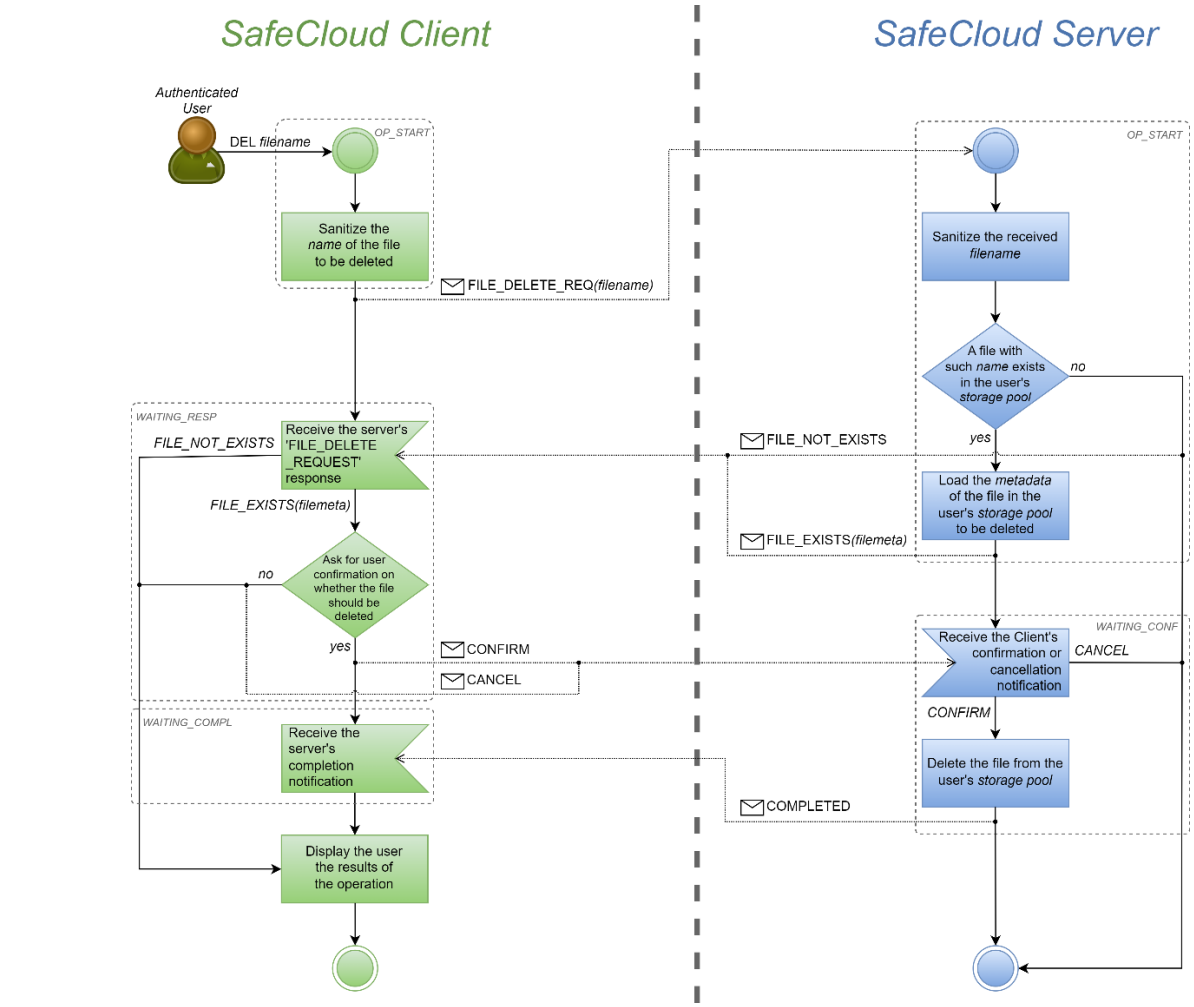
UPLOAD operation



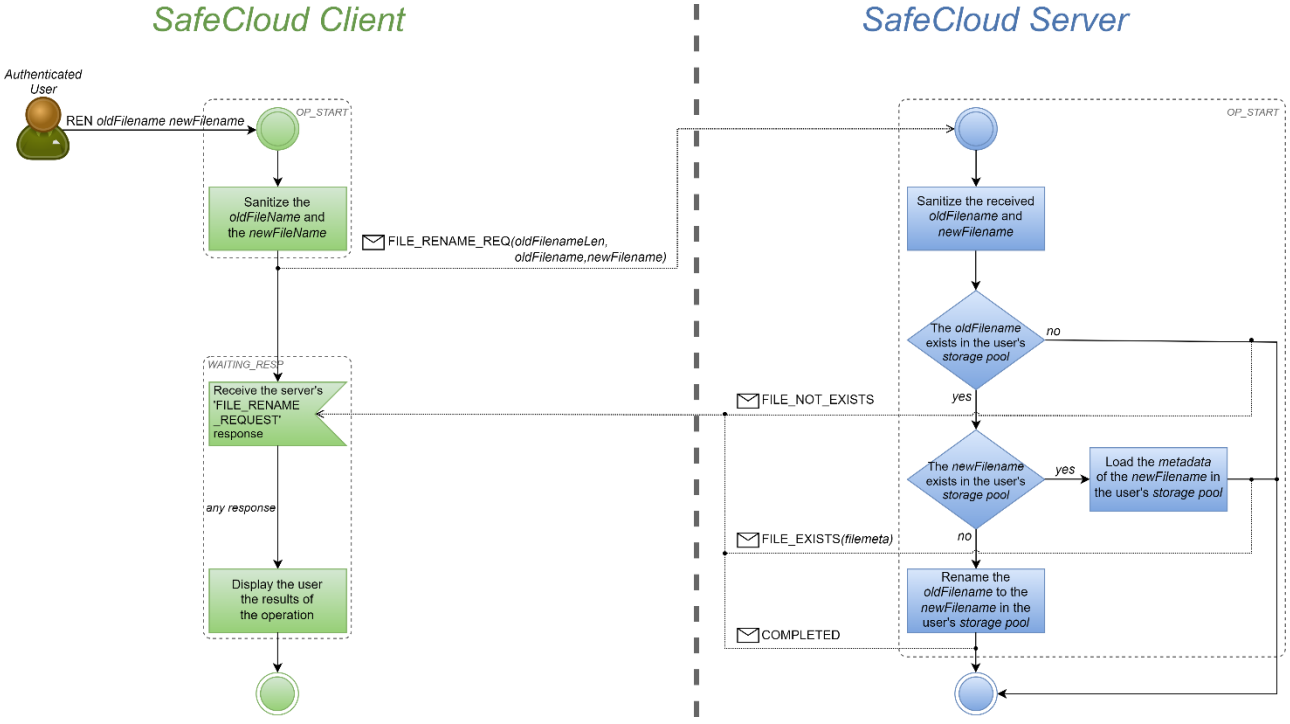
DOWNLOAD operation



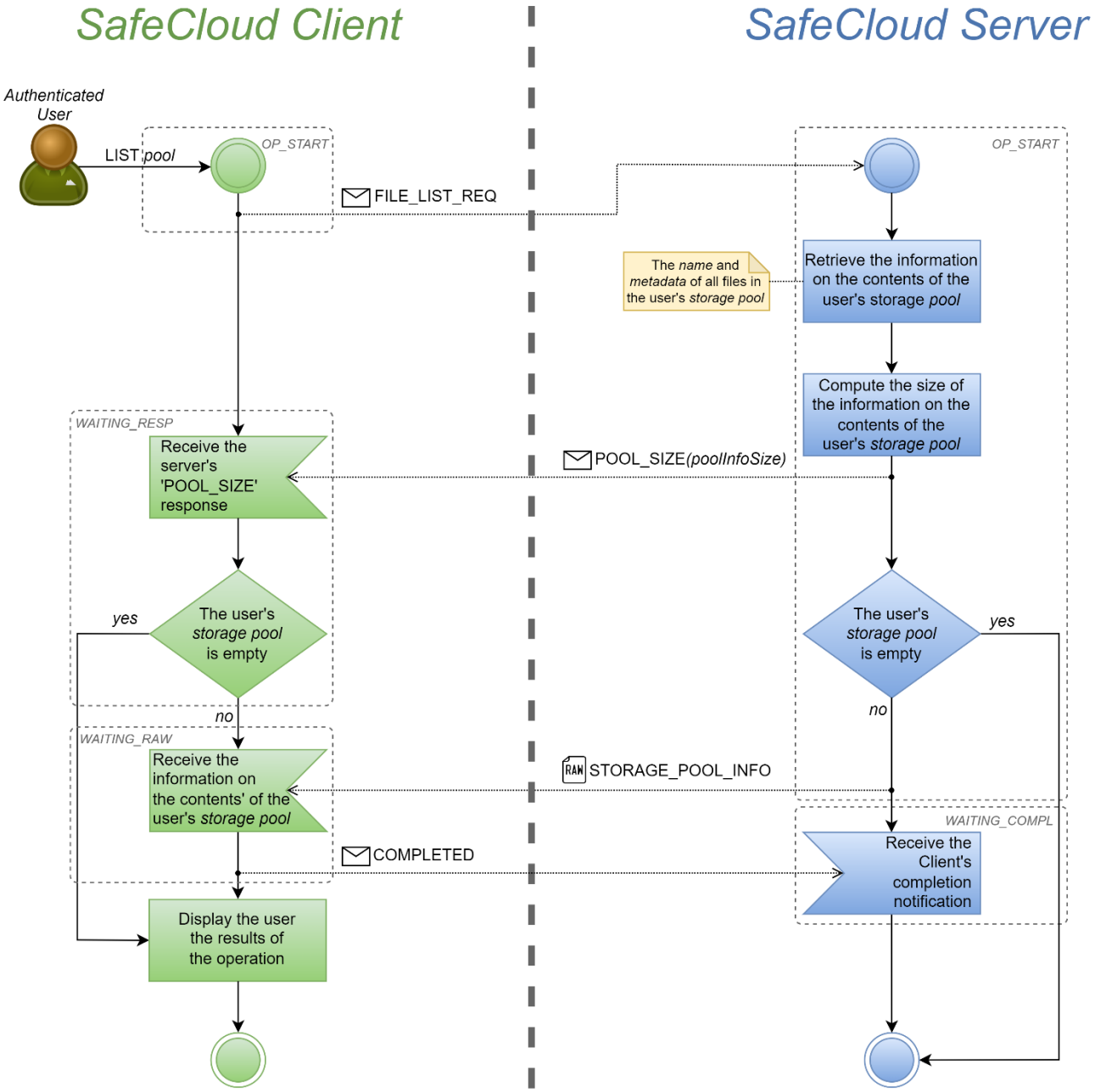
DELETE operation



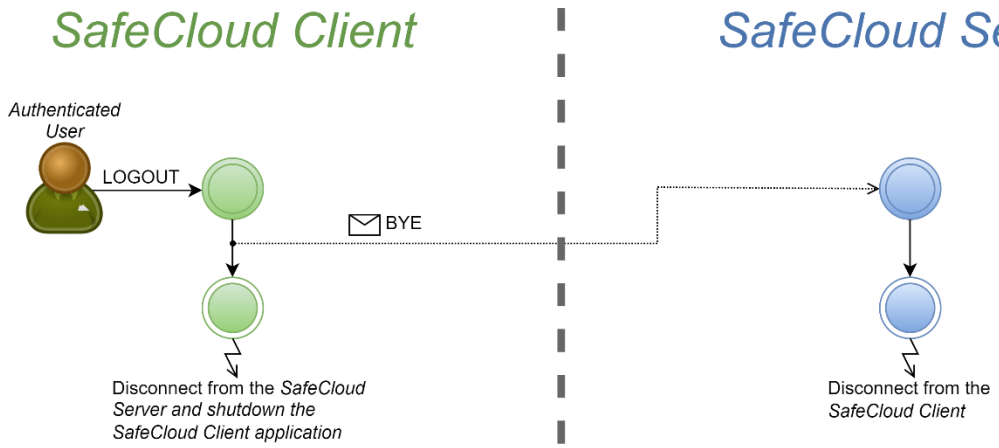
RENAME operation



LIST pool operation



LOGOUT operation



Other Design Details

IV Management

Since on the one hand the *AES_128 CBC* and *GCM encryption modes* employed in the *STSM Protocol* and the *Session Phase* use an IV of different size (16 and 12 bytes respectively), and on the other no 128-bit integer type is defined in the C++14 standard, to provide a compiler-independent implementation, the 16 bytes IV used in the service is divided into:

- A *variable* lower half (8 bytes), whose value as previously discussed is kept synchronized by the two parties by autonomously incrementing it for each encryption or decryption operation.
- A *constant* upper half (8 bytes), whose value is fixed and is used in its entirety by the *AES_128_CBC encryption mode* and only its lower half (4 bytes) by the *AES_128_GCM encryption mode*.

Even if variable in 64 bits only, since performing the 2^{64} encryption or decryption operations that would lead to the same IV being reused would require over 50 years performing an operation every 100ms, no rekeying or other IV refreshing mechanisms have been deemed necessary to prevent its reuse.

Applications Shutdown Management

In addition to *authenticated users* explicitly logging out from the *SafeCloud Server*, an action terminating their *SafeCloud Client* application, both the *SafeCloud Client* and *Server* can be shut down by receiving the subset of Linux inter-process signals aimed at interrupting or terminating a process (*SIGINT*, *SIGTERM*, *SIGQUIT*), which are handled by performing the following orderly shutdown procedures on the two applications:

SafeCloud Client

- If connection with a *SafeCloud Server* has not yet been established, the application is terminated directly in the shutdown signal's handler.
- If connected with the *SafeCloud Server* and in the *IDLE* session *operation*, a '*BYE*' *Session Signaling Message* is sent to the server and the application is terminated directly in the shutdown signal's handler
- Otherwise a *shutdown* flag is set for the application to autonomously terminate as soon as its pending session *operation* has completed.

SafeCloud Server

- If no client is connected, the application is terminated directly in the shutdown signal's handler.
- If there are connected clients, all clients in the *IDLE* session *operation* are disconnected by sending them a '*BYE*' *Session Signaling Message*, after which, if no client connection remains, the application is terminated directly in the shutdown signal's handler.
- Otherwise the server's listening socket is closed to prevent further client connections and a *shutdown* flag is set for the application to autonomously terminate as soon as no client connection remains, with clients being disconnected by sending them a '*BYE*' *Session Signaling Message* when their sessions return in the *IDLE* session *operation*.

Errors Management

All error conditions in the *SafeCloud Client* and *Server* applications have been associated with an *error code*, a *severity level* (*FATAL, CRITICAL, ERROR, INFO, DEBUG*) and a *human-readable description* for debugging purposes, and have been divided into:

- *Execution Errors*, which, if of *FATAL* severity, require the application to be terminated and, in any case, an existing connection between the *SafeCloud Client* and *Server* to be aborted.
- *Session Errors*, which are of non-*FATAL* severity and may occur during the service's *Session Phase* only, causing the current session *operation* to be aborted while maintaining the connection between the *SafeCloud Client* and *Server*.

The service's complete list of *execution* and *session error codes*, along with their *severity level* and *human-readable description*, can be found in the appendix.

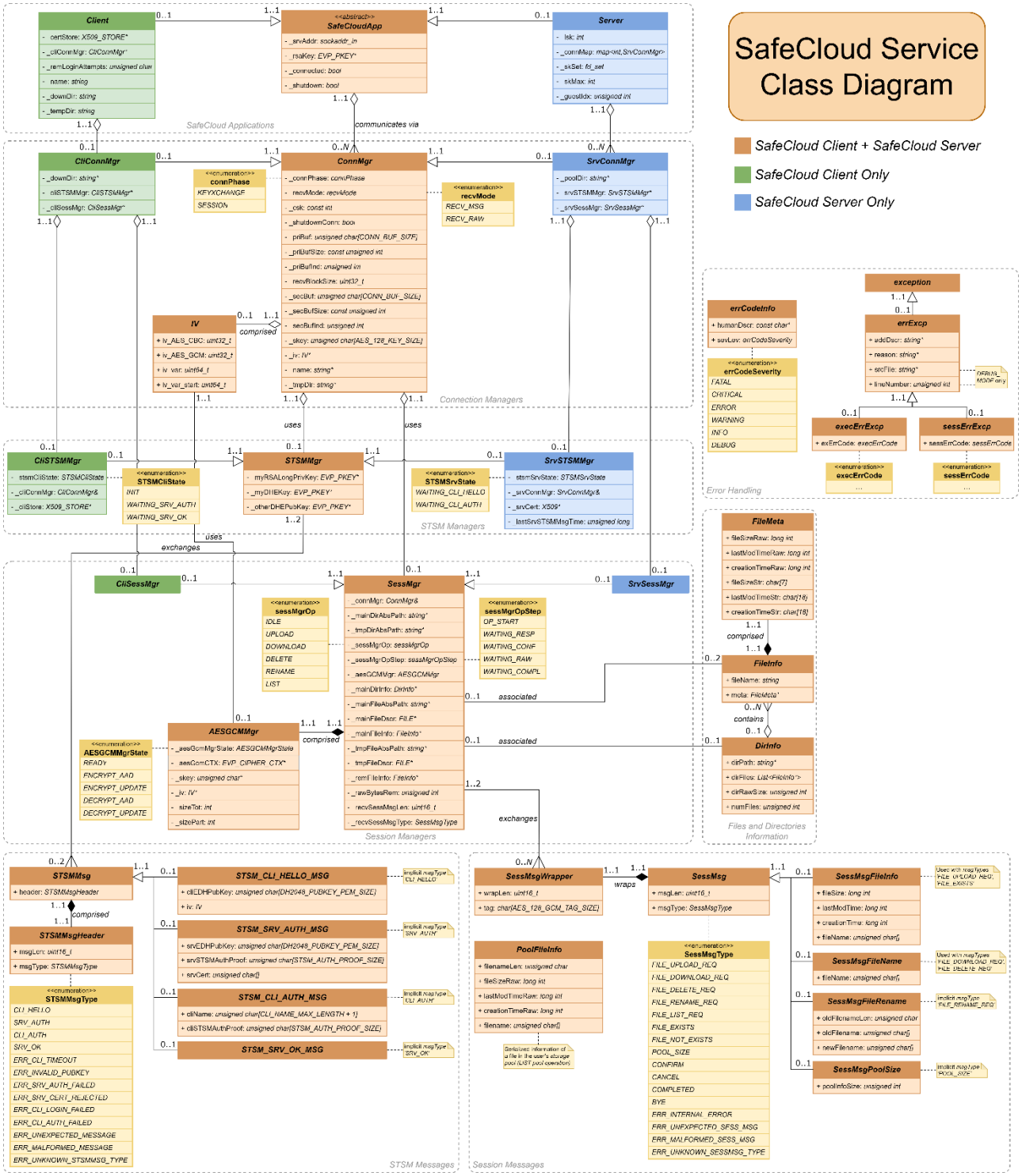
Service Object-Oriented Design

The service's object-oriented design can be summarized as follows:

- Both the *SafeCloud Client* and *Server* are *SafeCloud Applications (SafeCloudApp)* sharing a subset of attributes such as the *shutdown* flag and the TCP/IP endpoint of the *SafeCloud Server* to connect to or bind on.
- Each *SafeCloud Application* is associated with a *Connection Manager (ConnMgr)*, specialized for the two applications in the *Client Connection Manager (CliConnMgr)* and the *Server Connection Manager (SrvConnMgr)*, which implement the low-level data structure and functionalities required for securely exchanging messages and raw data via sockets.
- The *Key Establishment and Authentication Phase* is carried out by an *STSM Manager (STSMMgr)*, again specialized for the two applications in a *Client STSM Manager (CliSTSMMgr)* and a *Server STSM Manager (SrvSTSMMgr)*, which perform the *STSM* protocol by exchanging *STSM Messages (STSMMsg)*.
- The *Session Phase* is executed by a *Session Manager (SessMgr)*, again specialized for the two applications in a *Client Session Manager (CliSessMgr)* and a *Server Session Manager (SrvSessMgr)*, which is characterized by a state that is reset across the different session *operations* which are carried out by exchanging *Session Messages (SessMsg)*.

The complete set of classes comprising the *SafeCloud Service* is presented in the following *detailed class diagram* and listed in the appendix, with the signatures and descriptions of their methods, omitted for clarity purposes, to be found in the associated *header files* in the service's project repository.

Service Detailed Class Diagram



Service Implementation and Testing

The service has been implemented in the C++14 language according to the presented specification using the *cmake* toolchain based on the *ninja* build system and the *clang* compiler, with two versions having been developed for both the *SafeCloud Client* and *Server* applications:

- A *Debug Version* intended for development and testing purposes, which is compiled by defining the *DEBUG_MODE* project-wide symbol at compile time (*-DDEBUG_MODE* option), in which most of the applications' intermediate results, secret quantities included, as well as the *source file* and *line number* at which errors occurs, are logged to *stdout* so to allow for an easier troubleshooting.
- A *Release Version* intended to be deployed on the *SafeCloud Server* system(s) and to be distributed to the service's users, in which only the applications' main results are reported and the underlying reason of *login errors* is concealed in the *SafeCloud Client* in order to preclude malicious users from gaining any insights in attempting to impersonate other *authenticated users*.

To customize the host OS port the *SafeCloud Server* should bind to or the TCP/IP endpoint of the *SafeCloud Server* to connect to, the two applications can be started by specifying any of the following command-line parameters:

- *SafeCloud Server:* "-p [PORT]" → The port the application should bind to on the host OS.
- *SafeCloud Client:* "-a [IPv4]" → The IP address of the *SafeCloud Server* instance to connect to.
  "-p [PORT]" → The port of the *SafeCloud Server* instance to connect to.

With the allowed values of the [PORT] parameter falling into the IANA standard port range for dynamic applications (49152-65535), and with the SafeCloud Server application binding by default on port 51234 on all available IP interfaces on the host OS.

To test the service, the following entities along with their required cryptographic quantities were defined:

- A stub *"BertCA"* root certification authority, which was created by using the *simpleauthority* tool.
- A *SafeCloud Server* instance.
- The credentials of the following stub *authenticated users*, which have been pre-registered within the *SafeCloud Server* instance:

username	*password*
alice	alicepassword
bob	bobpassword
carol	carolpassword

The service has been thoroughly tested by deploying a *SafeCloud Server* and multiple *SafeCloud Client* instances in a *Local Area Network (LAN)* environment using *Ubuntu 18.04.6 LTS* as the host operating system, with the *Valgrind memory checker* tool that has also been used to ascertain the absence of *memory leaks* and other memory-related vulnerabilities.

Appendix I: Service Error Codes

Execution Errors

Server-Specific Execution Errors

Server Private Key Retrieval Execution Errors		
Error	**Severity**	**Description**
ERR_SRV_PRIVKFILE_NOT_FOUND	*FATAL*	The server's RSA private key file was not found
ERR_SRV_PRIVKFILE_OPEN_FAILED	*FATAL*	Error in opening the server's RSA private key file
ERR_SRV_PRIVK_INVALID	*FATAL*	The contents of the server's RSA private key file could not be interpreted as a valid RSA key pair

Server Certificate Retrieval Execution Errors		
Error	**Severity**	**Description**
ERR_SRV_CERT_OPEN_FAILED	*FATAL*	The server's certificate file could not be opened
ERR_SRV_CERT_INVALID	*FATAL*	The server's certificate file does not contain a valid X.509 certificate

Server Listening Socket Execution Errors		
Error	**Severity**	**Description**
ERR_LSK_INIT_FAILED	*FATAL*	Failed to initialize the server's listening socket
ERR_LSK_SO_REUSEADDR_FAILED	*FATAL*	Failed to set the listening socket's SO_REUSEADDR option
ERR_LSK_BIND_FAILED	*FATAL*	Failed to bind the listening socket on the specified OS port
ERR_LSK_LISTEN_FAILED	*FATAL*	Failed to listen on the listening socket
ERR_LSK_CLOSE_FAILED	*FATAL*	Failed to close the listening socket

Server Connection Sockets Execution Errors		
Error	**Severity**	**Description**
ERR_SRV_SELECT_FAILED	*FATAL*	Server *select()* failed
ERR_CSK_ACCEPT_FAILED	*CRITICAL*	Failed to accept an incoming client's connection
ERR_CSK_MAX_CONN	*WARNING*	Maximum number of clients connections reached, an incoming client connection has been rejected
RR_CSK_MISSING_MAP	*CRITICAL*	Connection socket with available data is missing from the connected clients' map
ERR_CLI_DISCONNECTED	*WARNING*	Abrupt client disconnection

Server STSM Execution Errors		
Error	**Severity**	**Description**
ERR_STSM_SRV_TIMEOUT	*ERROR*	Client STSM timeout
ERR_STSM_SRV_CLI_INVALID_PUBKEY	*CRITICAL*	The client has provided an invalid ephemeral public key
ERR_STSM_SRV_SRV_INVALID_PUBKEY	*CRITICAL*	The client reported that the server has provided an invalid ephemeral public key
ERR_STSM_SRV_SRV_AUTH_FAILED	*ERROR*	The client reported that the server has failed the STSM authentication
ERR_STSM_SRV_SRV_CERT_REJECTED	*ERROR*	The client rejected the server's X.509 certificate
ERR_STSM_SRV_CLIENT_LOGIN_FAILED	*ERROR*	The client has provided a username not registered within the *SafeCloud Server*
ERR_STSM_SRV_CLI_AUTH_FAILED	*ERROR*	The client has failed the STSM authentication
ERR_STSM_SRV_UNEXPECTED_MESSAGE	*CRITICAL*	The client reported to have received an out-of-order STSM message
ERR_STSM_SRV_MALFORMED_MESSAGE	*ERROR*	The client reported to have received a malformed STSM message
ERR_STSM_SRV_UNKNOWN_STSMMSG_TYPE	*ERROR*	The client reported to have received a STSM message of unknown type

Server Client Login Execution Errors		
Error	**Severity**	**Description**
ERR_LOGIN_PUBKEYFILE_NOT_FOUND	*ERROR*	The RSA private key file associated with the client-provided *username* was not found
ERR_LOGIN_PUBKEYFILE_OPEN_FAILED	*CRITICAL*	Failed to open the user's RSA public key file
ERR_LOGIN_PUBKEY_INVALID	*CRITICAL*	The contents of the user's RSA public key file could not be interpreted as a valid RSA public key

Server Session-Aborting Execution Errors		
Error	**Severity**	**Description**
ERR_SESSABORT_UNEXPECTED_POOL_SIZE	*CRITICAL*	The size of the information on the contents of the user's *storage pool* differ from its expected value
ERR_SESSABORT_SRV_CLI_UNKNOWN_SESSMSG_TYPE	*CRITICAL*	The client reported to have received a session message of unknown type

Client-Specific Execution Errors

Client X.509 Certificates Store Initialization Execution Errors		
Error	**Severity**	**Description**
ERR_CA_CERT_OPEN_FAILED	*FATAL*	The CA certificate file could not be opened
ERR_CA_CERT_INVALID	*FATAL*	The contents of the CA certificate file could not be interpreted as a valid X.509 certificate
ERR_CA_CRL_OPEN_FAILED	*FATAL*	The CA CRL file could not be opened
ERR_CA_CRL_INVALID	*FATAL*	The contents of the CA CRL file could not be interpreted as a X.509 certificate revocation list
ERR_STORE_INIT_FAILED	*FATAL*	Failed to initialize the X.509 certificates store
ERR_STORE_ADD_CACERT_FAILED	*FATAL*	Failed to add the CA certificate to the X.509 certificates store
ERR_STORE_ADD_CACRL_FAILED	*FATAL*	Failed to add the CA CRL to the X.509 certificates store
ERR_STORE_REJECT_SET_FAILED	*FATAL*	Error in setting the X.509 certificates store so as to reject revoked certificates

Client Login Execution Errors		
Error	**Severity**	**Description**
ERR_LOGIN_PWD_EMPTY	*ERROR*	The user-provided password is empty
ERR_LOGIN_PWD_TOO_LONG	*ERROR*	The user-provided password is too long
ERR_LOGIN_PRIVKFILE_NOT_FOUND	*ERROR*	The RSA private key file associated with the user-provided *username* was not found
ERR_LOGIN_PRIVKFILE_OPEN_FAILED	*ERROR*	Failed to open the user's RSA private key file
ERR_LOGIN_PRIVK_INVALID	*ERROR*	The contents of the user's private key file could not be interpreted as a valid RSA key pair
ERR_DOWNDIR_NOT_FOUND	*CRITICAL*	The user's *download directory* was not found
ERR_CLI_LOGIN_FAILED	*CRITICAL*	Maximum number of login attempts reached

Client Connection Socket Errors		
Error	**Severity**	**Description**
ERR_CSK_INIT_FAILED	*FATAL*	Failed to create the connection socket
ERR_SRV_UNREACHABLE	*WARNING*	Failed to connect with the *SafeCloud Server*
ERR_CSK_CONN_FAILED	*FATAL*	Fatal error in connecting with the *SafeCloud Server*
ERR_SRV_DISCONNECTED	*WARNING*	The *SafeCloud Server* has abruptly disconnected

Client STSM Execution Errors		
Error	**Severity**	**Description**
ERR_STSM_CLI_ALREADY_STARTED	*CRITICAL*	The client has already started the STSM protocol
ERR_STSM_CLI_TIMEOUT	*CRITICAL*	The server reported the client's timeout in the STSM protocol
ERR_STSM_CLI_CLI_INVALID_PUBKEY	*CRITICAL*	The server reported that the client has provided an invalid ephemeral public key
ERR_STSM_CLI_SRV_INVALID_PUBKEY	*CRITICAL*	The server has provided an invalid ephemeral public key
ERR_STSM_CLI_SRV_AUTH_FAILED	*CRITICAL*	The server has failed the STSM authentication
ERR_STSM_CLI_SRV_CERT_REJECTED	*ERROR*	The server has provided an invalid X.509 certificate
ERR_STSM_CLI_CLIENT_LOGIN_FAILED	*ERROR*	A user with such *username* is not registered within the server
ERR_STSM_CLI_CLI_AUTH_FAILED	*CRITICAL*	The server reported that the server has failed the STSM authentication
ERR_STSM_CLI_UNEXPECTED_MESSAGE	*FATAL*	The server reported to have received an out-of-order STSM message
ERR_STSM_CLI_MALFORMED_MESSAGE	*FATAL*	The server reported to have received a malformed STSM message
ERR_STSM_CLI_UNKNOWN_STSMMSG_TYPE	*FATAL*	The server reported to have received a STSM message of unknown type

Client Session-Aborting Execution Errors		
Error	**Severity**	**Description**
ERR_SESSABORT_CLI_SRV_UNKNOWN_SESSMSG_TYPE	*CRITICAL*	The server reported to have received a session message of unknown type
ERR_SESSABORT_SRV_GRACEFUL_DISCONNECT	*WARNING*	The server has gracefully disconnected
ERR_SESSABORT_INTERNAL_ERROR	*CRITICAL*	Unrecoverable session internal error

Client-Server Common Execution Errors

Common Server TCP Endpoint Execution Errors		
Error	**Severity**	**Description**
ERR_SRV_ADDR_INVALID	*ERROR*	The custom *SafeCloud Server* IP address is invalid
ERR_SRV_PORT_INVALID	*ERROR*	The custom *SafeCloud Server* port is invalid

Common Connection Sockets Execution Errors		
Error	**Severity**	**Description**
ERR_CSK_CLOSE_FAILED	*CRITICAL*	Failed to close the connection socket
ERR_CSK_RECV_FAILED	*CRITICAL*	Error in receiving data from the connection socket
ERR_PEER_DISCONNECTED	*WARNING*	Abrupt peer disconnection
ERR_SEND_FAILED	*FATAL*	Failed to send data via the connection socket
ERR_SEND_OVERFLOW	*FATAL*	Attempting to send a number of bytes greater than the size of the primary connection buffer
ERR_MSG_LENGTH_INVALID	*FATAL*	A message of invalid length was received

Common Files and Directories Execution Errors		
Error	**Severity**	**Description**
ERR_DIR_OPEN_FAILED	*CRITICAL*	The directory was not found
ERR_DIR_CLOSE_FAILED	*CRITICAL*	Failed to close the directory
ERR_FILE_OPEN_FAILED	*CRITICAL*	The file was not found
ERR_FILE_READ_FAILED	*CRITICAL*	Failed to read from the file
ERR_FILE_WRITE_FAILED	*CRITICAL*	Failed to write to the file
ERR_FILE_DELETE_FAILED	*CRITICAL*	Failed to delete the file
ERR_FILE_TOO_LARGE	*CRITICAL*	The file is too large
ERR_FILE_CLOSE_FAILED	*CRITICAL*	Failed to close the file

Common Client Login Execution Errors		
Error	**Severity**	**Description**
ERR_LOGIN_NAME_EMPTY	*CRITICAL*	The provided *username* is empty
ERR_LOGIN_NAME_TOO_LONG	*CRITICAL*	The provided *username* is too long
ERR_LOGIN_NAME_WRONG_FORMAT	*CRITICAL*	The provided *username* has an invalid format
ERR_LOGIN_NAME_INVALID_CHARS	*CRITICAL*	The provided *username* contains invalid characters
ERR_LOGIN_WRONG_NAME_OR_PWD	*CRITICAL*	Wrong *username* or *password*

Common OpenSSL Execution Errors		
Error	**Severity**	**Description**
ERR_OSSL_EVP_PKEY_NEW	*FATAL*	*EVP_PKEY* struct creation failed
ERR_OSSL_EVP_PKEY_ASSIGN	*FATAL*	*EVP_PKEY* struct assignment failed
ERR_OSSL_EVP_PKEY_CTX_NEW	*FATAL*	*EVP_PKEY* context creation failed
ERR_OSSL_EVP_PKEY_KEYGEN_INIT	*FATAL*	*EVP_PKEY* key generation initialization failed
ERR_OSSL_EVP_PKEY_KEYGEN	*FATAL*	*EVP_PKEY* key generation failed
ERR_OSSL_EVP_PKEY_DERIVE_INIT	*FATAL*	Failed to initialize the key derivation context
ERR_OSSL_EVP_PKEY_DERIVE_SET_PEER	*FATAL*	Failed to set the party's public key in the key derivation context
ERR_OSSL_EVP_PKEY_DERIVE	*FATAL*	Failed to derive the shared secret in a key derivation context
ERR_OSSL_RAND_POLL_FAILED	*FATAL*	*RAND_poll()* seed generation failed
ERR_OSSL_RAND_BYTES_FAILED	*FATAL*	*RAND_pool()* random bytes generation failed
ERR_OSSL_BIO_NEW_FAILED	*FATAL*	Failed to initialize the memory BIO
ERR_OSSL_BIO_NEW_FP_FAILED	*CRITICAL*	Failed to initialize the file BIO
ERR_OSSL_PEM_WRITE_BIO_PUBKEY_FAILED	*FATAL*	Failed to write the ephemeral DH public key to the memory BIO
ERR_OSSL_EVP_PKEY_PRINT_PUBLIC_FAILED	*CRITICAL*	Failed to write the ephemeral DH public key to the file BIO
ERR_OSSL_BIO_READ_FAILED	*FATAL*	Failed to read from the BIO
ERR_OSSL_BIO_FREE_FAILED	*CRITICAL*	Failed to free the BIO
ERR_OSSL_EVP_MD_CTX_NEW	*FATAL*	*EVP_MD* context creation failed
ERR_OSSL_EVP_DIGEST_INIT	*FATAL*	*EVP_MD* digest initialization failed
ERR_OSSL_EVP_DIGEST_UPDATE	*FATAL*	*EVP_MD* digest update failed
ERR_OSSL_EVP_DIGEST_FINAL	*FATAL*	*EVP_MD* digest final failed
ERR_OSSL_EVP_SIGN_INIT	*FATAL*	*EVP_MD* signing initialization failed
ERR_OSSL_EVP_SIGN_UPDATE	*FATAL*	*EVP_MD* signing update failed
ERR_OSSL_EVP_SIGN_FINAL	*FATAL*	*EVP_MD* signing final failed
ERR_OSSL_AES_128_CBC_PT_TOO_LARGE	*FATAL*	The plaintext to be encrypted via the *AES_128_CBC* cipher is too large
ERR_OSSL_EVP_CIPHER_CTX_NEW	*FATAL*	*EVP_CIPHER* context creation failed
ERR_OSSL_EVP_ENCRYPT_INIT	*FATAL*	*EVP_CIPHER* encrypt initialization failed
ERR_OSSL_EVP_ENCRYPT_UPDATE	*FATAL*	*EVP_CIPHER* encrypt update failed
ERR_OSSL_EVP_ENCRYPT_FINAL	*FATAL*	*EVP_CIPHER* encrypt final failed
ERR_OSSL_PEM_WRITE_BIO_X509	*FATAL*	Failed to write the X.509 certificate to the memory BIO
ERR_OSSL_X509_STORE_CTX_NEW	*FATAL*	*X509_STORE* context creation failed
ERR_OSSL_X509_STORE_CTX_INIT	*FATAL*	*X509_STORE* context initialization failed
ERR_OSSL_EVP_VERIFY_INIT	*FATAL*	*EVP_MD* verification initialization failed

ERR_OSSL_EVP_VERIFY_UPDATE	*FATAL*	*EVP_MD* verification update failed
ERR_OSSL_EVP_VERIFY_FINAL	*FATAL*	*EVP_MD* verification final failed
ERR_OSSL_SIG_VERIFY_FAILED	*CRITICAL*	Signature verification failed
ERR_OSSL_EVP_DECRYPT_INIT	*FATAL*	*EVP_CIPHER* decrypt initialization failed
ERR_OSSL_EVP_DECRYPT_UPDATE	*FATAL*	*EVP_CIPHER* decrypt update failed
ERR_OSSL_EVP_DECRYPT_FINAL	*FATAL*	*EVP_CIPHER* decrypt final failed
ERR_OSSL_GET_TAG_FAILED	*FATAL*	Failed to retrieve the encryption operation's *AES_128_GCM* tag
ERR_OSSL_SET_TAG_FAILED	*FATAL*	Failed to set the decryption operation's expected *AES_128_GCM* tag

Common STSM Execution Errors		
Error	**Severity**	**Description**
ERR_STSM_UNEXPECTED_MESSAGE	*CRITICAL*	An out-of-order STSM message was received
ERR_STSM_MALFORMED_MESSAGE	*CRITICAL*	A malformed STSM message was received
ERR_STSM_UNKNOWN_STSMMSG_TYPE	*CRITICAL*	A STSM message of unknown type was received
ERR_STSM_UNKNOWN_STSMMSG_ERROR	*FATAL*	Attempting to send a STSM error message of unknown type
ERR_STSM_MY_PUBKEY_MISSING	*FATAL*	The owner's ephemeral DH public key is missing
ERR_STSM_OTHER_PUBKEY_MISSING	*FATAL*	The party's ephemeral DH public key is missing

Common SafeCloud Object States Execution Errors		
Error	**Severity**	**Description**
ERR_CONNMGR_INVALID_STATE	*CRITICAL*	Invalid *ConnMgr state*
ERR_AESGCMMGR_INVALID_STATE	*CRITICAL*	Invalid *AES_128_GCM* manager state

Common Session-Aborting Execution Errors		
Error	**Severity**	**Description**
ERR_SESSABORT_UNEXPECTED_FILE_SIZE	*CRITICAL*	The file contents that were read differ from their expected size
ERR_SESSABORT_UNKNOWN_SESSMSG_TYPE	*CRITICAL*	A session message of unknown type was received

Common Other Execution Errors		
Error	**Severity**	**Description**
ERR_MALLOC_FAILED	*FATAL*	*malloc()* failed
ERR_NON_POSITIVE_BUFFER_SIZE	*FATAL*	A non-positive buffer size was passed (probable overflow)
ERR_EXEC_UNKNOWN	*CRITICAL*	Unknown *execution error*

Session Errors

Server-Specific Session Errors

Server Session Messages Errors		
Error	**Severity**	**Description**
ERR_SESS_SRV_CLI_INTERNAL_ERROR	*WARNING*	The client reported to have experienced an internal error
ERR_SESS_SRV_CLI_UNEXPECTED_MESSAGE	*ERROR*	The client reported to have received an unexpected session message
ERR_SESS_SRV_CLI_MALFORMED_MESSAGE	*ERROR*	The client reported to have received a malformed session message

Client-Specific Session Errors

Client Unsupported Session Command Error		
Error	**Severity**	**Description**
ERR_UNSUPPORTED_CMD	*INFO*	The user submitted an unsupported session command

Client Files Session Errors		
Error	**Severity**	**Description**
ERR_SESS_FILE_NOT_FOUND	*WARNING*	The user-specified file was not found
ERR_SESS_FILE_READ_FAILED	*ERROR*	Failed to read from the user-specified file
ERR_SESS_FILE_IS_DIR	*WARNING*	The user-specified file is a directory
ERR_SESS_FILE_TOO_BIG	*WARNING*	The user-specified file is too big (≥4GB)
ERR_SESS_UPLOAD_DIR	*WARNING*	The file the user is attempting to upload is a directory
ERR_SESS_UPLOAD_TOO_BIG	*WARNING*	The file the user is attempting to upload is too big
ERR_SESS_RENAME_SAME_NAME	*WARNING*	The user is attempting to rename a file to itself

Client Session Messages Errors		
Error	**Severity**	**Description**
ERR_SESS_CLI_SRV_INTERNAL_ERROR	*ERROR*	The server reported to have experienced an internal error
ERR_SESS_CLI_SRV_UNEXPECTED_MESSAGE	*CRITICAL*	The server reported to have received an unexpected session message
ERR_SESS_CLI_SRV_MALFORMED_MESSAGE	*CRITICAL*	The server reported to have received a malformed session message

Client-Server Common Session Errors

Common Files Session Errors		
Error	**Severity**	**Description**
ERR_SESS_DIR_INFO_OVERFLOW	*ERROR*	Directory contents information size overflow (≥4GB)
ERR_SESS_MAIN_FILE_IS_DIR	*CRITICAL*	The *main file* was found as a sub-directory of the session's *main directory*
ERR_SESS_FILE_INVALID_NAME	*ERROR*	Invalid file *name*
ERR_SESS_FILE_META_NEGATIVE	*CRITICAL*	Attempting to initialize a file's metadata to negative values
ERR_SESS_FILE_INFO_COMP_NULL	*CRITICAL*	Attempting to compare the metadata of a NULL *FileInfo* pointer
ERR_SESS_FILE_INFO_COMP_DIFF_NAMES	*CRITICAL*	Attempting to compare the metadata of two files of different *names*
ERR_SESS_FILE_OPEN_FAILED	*ERROR*	Failed to open the file
ERR_SESS_FILE_DELETE_FAILED	*CRITICAL*	Failed to delete the file
ERR_SESS_FILE_META_SET_FAILED	*CRITICAL*	Failed to set the file's *metadata*
ERR_SESS_FILE_CLOSE_FAILED	*CRITICAL*	Failed to close the file
ERR_SESS_FILE_RENAME_FAILED	*CRITICAL*	Failed to rename the file

Common Session Messages Errors		
Error	**Severity**	**Description**
ERR_SESS_INTERNAL_ERROR	*CRITICAL*	An internal error has occurred
ERR_SESS_UNEXPECTED_MESSAGE	*ERROR*	An unexpected session message was received
ERR_SESS_MALFORMED_MESSAGE	*ERROR*	A malformed session message was received

Common Other Session Errors		
Error	**Severity**	**Description**
ERR_OSSL_DECRYPT_VERIFY_FAILED	*ERROR*	*AES_128_GCM* tag verification failed
ERR_SESS_UNKNOWN	*CRITICAL*	Unknown *session error*

Appendix II: Service Classes Definitions

SafeCloud Application Classes

SafeCloud App		
SafeCloud application abstract base class		
Attribute	**Description**	**Type**
_srvaddr	The TCP/IP endpoint of the *SafeCloud Server* to connect to or bind on	*sockaddr_in*
_rsaKey	The long-term RSA key pair associated with the application	*EVP_PKEY**
_connected	Whether the application has established a TCP connection	*bool*
_shutdown	Whether the application should shut down as soon as possible	*bool*

Client		
SafeCloud Client main class		
Attribute	**Description**	**Type**
_certStore	The client's X.509 certificates store	*X509_STORE**
_cliConnMgr	A reference to the associated *Client Connection Manager*	*CliConnMgr**
_name	The user's *name*	*string*
_downDir	The absolute path of the user's *download directory*	*string*
_tempDir	The absolute path of the user's *temporary directory*	*string*

Server		
SafeCloud Server main class		
Attribute	**Description**	**Type**
_lsk	The server listening socket's file descriptor	*int*
_srvCert	The server's X.509 certificate	*X509**
_connMap	A map associating the file descriptors of open connection sockets to their associated *srvConnMgr* objects (one per client)	*map<int,SrvConnMgr*>*
_skSet	The set of file descriptors of open sockets (listening socket + connection sockets)	*fd_set*
_skMax	The maximum socket file descriptor value in the *SafeCloud Server* execution (*select()* optimization purposes)	*int*
_guestIdx	Temporary identifier of the last *guest* connected with the *SafeCloud Server*	*unsigned int*

Connection Managers Classes

ConnMgr		
Connection Manager base class		
Attribute	**Description**	**Type**
_connPhase	The connection's current *phase* (*Key Establishment and Authentication* or *Session*)	*connPhase*
_recvMode	The connection manager current *reception mode* (*message* or *raw*)	*recvMode*
_csk	The file descriptor of the connection socket associated with this connection manager	*const int*
_shutdownConn	Whether the connection should be terminated	*bool*
_priBuf	Primary connection buffer	*unsigned char [CONN_BUF_SIZE]*
_priBufSize	Primary connection buffer size	*const unsigned int*
_priBufInd	Index of the first available byte (or number of significant bytes) in the primary connection buffer	*unsigned int*
_recvBlockSize	Expected size in bytes of the data block (message or raw) to be received	*uint32_t*
_secBuf	Secondary connection buffer	*unsigned char [CONN_BUF_SIZE]*
_secBufSize	Secondary connection buffer size	*const unsigned int*
_secBufInd	Index of the first available byte (or number of significant bytes in the secondary connection buffer	*unsigned int*
_skey	The connection's ephemeral session key	*unsigned char [AES_128_KEY_SIZE]*
_iv	The connection's current IV value	*IV**
_name	The name of the user associated with this connection manager	*string**
_tmpDir	The absolute path of the temporary directory associated with this connection manager	*string**

CliConnMgr		
Client Connection Manager main class		
Attribute	**Description**	**Type**
_downDir	The absolute path of the *authenticated user*'s *download directory*	*string**
_cliSTSMMgr	A reference to the associated *Client STSM Manager*	*CliSTSMMgr**
_cliSessMgr	A reference to the associated *Client Session Manager*	*CliSessMgr**

SrvConnMgr		
Server Connection Manager main class		
Attribute	**Description**	**Type**
_poolDir	The absolute path of the *storage pool* of the *authenticated user* associated with this connection manager	*string**
_srvSTSMMgr	A reference to the associated *Server STSM Manager*	*SrvSTSMMgr**
_srvSessMgr	A reference to the associated *Server Session Manager*	*SrvSessMgr**

IV		
A connection manager's IV value		
Attribute	**Description**	**Type**
iv_AES_CBC	The starting index of the IV used by the *AES_128_CBC* cipher in the IV's constant upper half	*uint32_t*
Iv_AES_GCM	The starting index of the IV used by the *AES_128_GCM* cipher in the IV's constant upper half	*uint32_t*
iv_var	The IV variable lower half that is incremented at every encryption or decryption operation	*uint64_t*
iv_var_start	The initial value of the IV variable lower half *(unused)*	*uint64_t*

Enumeration	**Description**	**Possible Values**
connPhase	Connection Phases (*Key Establishment and Authentication* or *Session*)	*KEYXCHANGE, SESSION*
recvMode	Connection Manager *reception modes* (*message* or *raw*)	*RECV_MSG, RECV_RAW*

STSM Managers Classes

STSMMgr		
STSM Manager base class		
Attribute	**Description**	**Type**
_myRSALongPrivKey	The owner's long-term RSA-2048 private key	*EVP_PKEY**
_myDHEKey	The owner's ephemeral DH-2048 key pair	*EVP_PKEY**
_otherDHEPubKey	The other connection party's ephemeral DH-2048 public key	*EVP_PKEY**

CliSTSMMgr		
Client STSM Manager main class		
Attribute	**Description**	**Type**
_stsmCliState	The client current state in the STSM protocol	*STSMCliState*
_cliConnMgr	A reference to the associated *Client Connection Manager*	*CliConnMgr&*
_cliStore	A reference to the client's X.509 certificates store	*X509_STORE**

SrvSTSMMgr		
Server STSM Manager main class		
Attribute	**Description**	**Type**
_stsmSrvState	The server current state in the STSM protocol	*STSMSrvState*
_srvConnMgr	A reference to the associated *Server Connection Manager*	*SrvConnMgr&*
_srvCert	A reference to the server's X.509 certificate	*X509 **
_lastSrvSTSMMsgTime	The time in Unix epochs at which the server sent its last STSM message to the client (STSM timeout purposes)	*unsigned long*

Enumeration	**Description**	**Possible Values**
STSMCliState	STSM protocol client states	*INIT, WAITING_SRV_AUTH, WAITING_SRV_OK*
STSMSrvState	STSM protocol server states	*WAITING_CLI_HELLO, WAITING_CLI_AUTH*

STSM Messages Classes

STSMMsg		
STSM Message base class		
Attribute	**Description**	**Type**
header	The STSM message's header	*STSMMsgHeader*

STSMMsgHeader		
STSM Message Header		
Attribute	**Description**	**Type**
msgLen	The STSM message total bytes length	*uint16_t*
msgType	The STSM message type	*STSMMsgType*

STSM_CLI_HELLO_MSG		
STSM *'CLI_HELLO'* Message (implicit *msgType = CLI_HELLO*)		
Attribute	**Description**	**Type**
cliEDHPubKey	The client's ephemeral DH-2048 public key	*unsigned char [DH2048_PUBKEY_PEM_SIZE]*
iv	The initialization vector's initial random value	*IV*

STSM_SRV_AUTH_MSG		
STSM *'SRV_AUTH'* Message (implicit *msgType = SRV_AUTH*)		
Attribute	**Description**	**Type**
srvEDHPubKey	The server's ephemeral DH-2048 public key	*unsigned char [DH2048_PUBKEY_PEM_SIZE]*
srvSTSMAuthProof	The server's STSM authentication proof	*unsigned char [STSM_AUTH_PROOF_SIZE]*
srvCert	The server's X.509 certificate	*unsigned char[]*

STSM_CLI_AUTH_MSG		
STSM *'CLI_AUTH'* Message (implicit *msgType = CLI_AUTH*)		
Attribute	**Description**	**Type**
cliName	The user's name	*unsigned char [CLI_NAME_MAX_LENGTH + 1]*
cliSTSMAuthProof	The client's STSM authentication proof	*unsigned char [STSM_AUTH_PROOF_SIZE]*

STSM_SRV_OK_MSG		
STSM *'SRV_OK* Message (implicit *msgType = SRV_OK*)		
Attribute	**Description**	**Type**
(same attributes of the *STSMMsg* base class)		

Enumeration	Description	Possible Values
STSMMsgType	STSM Message Types	*CLI_HELLO,* *SRV_AUTH,* *SRV_OK,* *ERR_CLI_TIMEOUT,* *ERR_INVALID_PUBKEY,* *ERR_SRV_AUTH_FAILED,* *ERR_SRV_CERT_REJECTED,* *ERR_CLI_LOGIN_FAILED,* *ERR_CLI_AUTH_FAILED,* *ERR_UNEXPECTED_MESSAGE,* *ERR_MALFORMED_MESSAGE,* *ERR_UNKNOWN_STSMMSG_TYPE*

Session Managers Classes

SessMgr		
Session Manager base class		
Attribute	**Description**	**Type**
_connMgr	A reference to the associated *Connection Manager*	*ConnMgr&*
_mainDirAbsPath	The absolute path of the session's *main directory*	*string**
_tmpDirAbsPath	The absolute path of the session's *temporary directory*	*string**
_sessMgrOp	The current session manager *operation*	*sessMgrOp*
_sessMgrOpStep	The current session manager *operation step*	*sessMgrOpStep*
_aesGCMMgr	The associated *AES_128_GCM* Manager	*AESGCMMgr*
_mainDirInfo	Information on the contents of the session's *main directory*	*DirInfo**
_mainFileAbsPath	The absolute path of a file in the session's *main directory*	*string**
_mainFileDscr	The descriptor of a file in the session's *main directory*	*FILE**
_mainFileInfo	Information on a file in the session's *main directory*	*FileInfo**
_tmpFileAbsPath	The absolute path of a file in the session's *temporary directory*	*string**
_tmpFileDscr	The descriptor of a file in the session's *temporary directory*	*FILE**
_remFileInfo	Information on remote file	*FileInfo**
_rawBytesRem	The remaining number of bytes to be sent or received in a *Raw Session Data* exchange	*unsigned int*
_recvSessMsgLen	The length of the last received *Session Message*	*uint16_t*
_recvSessMsgType	The type of the last received *Session Message*	*SessMsgType*

CliSessMgr		
Client Session Manager main class		
Attribute	**Description**	**Type**
(same attributes of the *SessMgr* base class)		

SrvSessMgr		
Server Session Manager main class		
Attribute	**Description**	**Type**
(same attributes of the *SessMgr* base class)		

AESGCMMgr		
Session AES_128_GCM Manager		
Attribute	**Description**	**Type**
_aesGcmMgrState	The current AES_128_GCM manager state	*AESGCMMgrState*
_aesGcmCTX	The cipher context used in the current or next AES_128_GCM encryption or decryption operation	*EVP_CIPHER_CTX**
_skey	A reference to the connection's ephemeral session key	*unsigned char**
_iv	A reference to the connection's IV	*IV**
_sizeTot	The total number of bytes encrypted or decrypted in the current operation, eventually representing the resulting ciphertext or plaintext size, AAD included	*int*
_sizePart	The number of bytes encrypted or decrypted in the last *OpenSSL* API call	*int*

Enumeration	**Description**	**Possible Values**
sessMgrOp	Session managers *operations*	*IDLE, UPLOAD, DOWNLOAD, DELETE, RENAME, LIST*
sessMgrOpStep	Session managers *operations steps*	*OP_START, WAITING_RESP, WAITING_CONF, WAITING_RAW, WAITING_COMPL*
AESGCMMgrState	AES_128_GCM Manager states	*READY, ENCRYPT_AAD ENCRYPT_UPDATE, DECRYPT_AAD DECRYPT_UPDATE*

Session Messages Classes

SessMsgWrapper		
Session Messages Wrapper		
Attribute	**Description**	**Type**
wrapLen	The session message wrapper total bytes length	*uint16_t*
tag	The session message wrapper's *AES_128_GCM* integrity tag (16 bytes)	*char[AES_128_GCM_TAG_SIZE]*

SessMsg		
Session Message base class		
Attribute	**Description**	**Type**
msgLen	The session message total bytes length	*uint16_t*
msgType	The session message type	*SessMsgType*

SessMsgFileInfo		
A session message carrying the *name* and *metadata* of a file *(used with msgTypes FILE_UPLOAD_REQ, FILE_EXISTS)*		
Attribute	**Description**	**Type**
fileSize	The file's size in bytes	*long int*
lastModTime	The file's *last modified time* in Unix epochs	*long int*
creationTime	The file's *creation time* in Unix epochs	*long int*
fileName	The file's *name*	*unsigned char[]*

SessMsgFileName		
A session message carrying a file's *name* *(used with msgTypes FILE_DOWNLOAD_REQ, FILE_DELETE_REQ)*		
Attribute	**Description**	**Type**
fileName	The file's *name*	*unsigned char[]*

SessMsgFileRename		
A session message specifying a file's desired new name *(implicit msgType = FILE_RENAME_REQ)*		
Attribute	**Description**	**Type**
oldFilenameLen	The current file *name* length in bytes	*unsigned char*
oldFilename	The current file *name*	*unsigned char[]*
newFilename	The desired new file *name*	*unsigned char[]*

SessMsgPoolSize		
A session message carrying the size of the information on the contents of a user's *storage pool* *(implicit msgType = POOL_SIZE)*		
Attribute	**Description**	**Type**
poolInfoSize	The size of the information on the contents of the user's *storage pool*	*unsigned int*

PoolFileInfo		
Serialized information of a file in a user's *storage pool*		
Attribute	**Description**	**Type**
filenameLen	The file name *length* in bytes	*unsigned char*
fileSizeRaw	The file's size in bytes	*long int*
lastModTimeRaw	The file's *last modified time* in Unix epochs	*long int*
creationTimeRaw	The file's *creation time* in Unix epochs	*long int*
filename	The file's *name*	*unsigned char[]*

Enumeration	**Description**	**Possible Values**
SessMsgType	Session message types	*FILE_UPLOAD_REQ,* *FILE_DOWNLOAD_REQ,* *FILE_DELETE_REQ,* *FILE_RENAME_REQ,* *FILE_LIST_REQ,* *FILE_EXISTS,* *FILE_NOT_EXISTS,* *POOL_SIZE,* *CONFIRM,* *CANCEL,* *COMPLETED,* *BYTE* *ERR_INTERNAL_ERROR,* *ERR_UNEXPECTED_SESS_MSG,* *ERR_MALFORMED_SESS_MSG,* *ERR_UNKNOWN_SESSMSG_TYPE*

Files and Directories Information Classes

DirInfo		
Information on a directory's contents		
Attribute	**Description**	**Type**
dirPath	The directory's absolute path	*string**
dirFiles	The list of information (*name* + *metadata*) of files in the directory	*List<FileInfo*>*
dirRawSize	The directory's information raw size, consisting in the sum of the sizes of its files' *names* and *metadata*	*unsigned int*
numFiles	The number of files in the directory	*unsigned int*

FileInfo		
Information on a file consisting in its *name* and *metadata*		
Attribute	**Description**	**Type**
fileName	The file's *name*	*string*
meta	The file's *metadata*	*FileMeta**

FileMeta		
File *metadata* in *raw* and *stringyfied* form		
Attribute	**Description**	**Type**
fileSizeRaw	The file's size in bytes	*long int*
lastModTimeRaw	The file's *last modified time* in Unix epochs	*long int*
creationTimeRaw	The file's *creation time* in Unix epochs	*long int*
fileSizeStr	The stringyfied file size	*char[7]*
lastModTimeStr	The stringyfied file *last modified time*	*char[18]*
creationTimeStr	The stringyfied file *creation time*	*char[18]*

Error Handling Classes

errCodeInfo		
A service's error severity level and human-readable description		
Attribute	**Description**	**Type**
sevLev	The error's severity level	*errCodeSeverity*
humanDscr	The error's human readable description	*const char**

errExcp		
General information on a service's exception		
Attribute	**Description**	**Type**
addDscr	An optional additional description associated with the exception's error	*string**
reason	An optional reason associated with the exception's error	*string**
srcFile	The source file at which the exception was raised (*DEBUG_MODE* only)	*string**
lineNumber	The line number at which the exception was raised (*DEBUG_MODE* only)	*unsigned int*

execErrExcp		
Information on a service's *execution exception*		
Attribute	**Description**	**Type**
exErrCode	The *execution error* associated with this exception	*execErrCode*

sessErrExcp		
Information on a service's *session exception*		
Attribute	**Description**	**Type**
sessErrCode	The *session error* associated with this exception	*sessErrCode*

Enumeration	**Description**	**Possible Values**
errCodeSeverity	Errors severity levels	*FATAL, CRITICAL, ERROR, WARNING, INFO, DEBUG*
execErrCode	*Execution errors* codes	*Execution Error Codes List*
sessErrCode	*Session errors* codes	*Session Error Codes List*