UNIVERSITÀ DI PISA

Università degli Studi di Pisa

Dipartimento di Ingegneria dell'Informazione

Applied Cryptography

Cloud Storage

Vito Chiumiento
Giuseppe Pericone
Eugenio Scaramozza

Anno Accademico 2022/2023

Indice

1 Linee guida e scelte implementative **3**

2 Autenticazione **4**

2.1 Messaggio 1 5

2.2 Messaggio 2 5

2.3 Messaggio 3 6

2.4 Messaggio 4 6

3 Trasferimento dati **8**

3.1 Funzioni Send() e Receive() 8

3.1.1 Gestione dei counter 9

3.2 Operazioni 9

3.2.1 Command Request 9

3.2.2 Upload 10

3.2.3 Download 10

3.2.4 Rename 11

3.2.5 Delete 12

3.2.6 List e Logout 14

Capitolo 1

Linee guida e scelte implementative

Le seguenti specifiche progettuali descrivono l'implementazione di un'applicazione client-server che simuli un cloud storage in cui ogni utente possiede uno spazio dedicato di archiviazione, non accessibile agli altri utenti. Quanto segue è l'esposizione delle scelte progettuali e implementative adottate al fine di garantire una comunicazione sicura tra il Client e il Server, sia in fase di autenticazione che in fase di traferimento dati.

Con "comunicazione sicura" si intende un tipo di comunicazione cifrata, integra, autenticata, immune agli attacchi replay e che garantisce la perfect forward secrecy nella negoziazione delle chiavi. Per la gestione delle operazioni effettuabili dal Client (descritte in dettaglio nel Capitolo 3) è stata costruita una mappa 'stringa-oggetto', dove a ogni stringa è collegata l'esecuzione di uno dei sei comandi. Questa gestione permette di avere un codcie più semplice e efficiente.

Capitolo 2

Autenticazione

La Figura 2.1 rappresenta lo scambio di messaggi tra Client e Server, necessario per l'autenticazione, basato su RSA effimero. Nei seguenti diagrammi verranno indicati in verde le parti del messaggio trasmesse in chiaro, mentre in arancione e in rosso troveremo rispettivamente gli elementi firmati e quelli cifrati con le apposite chiavi.

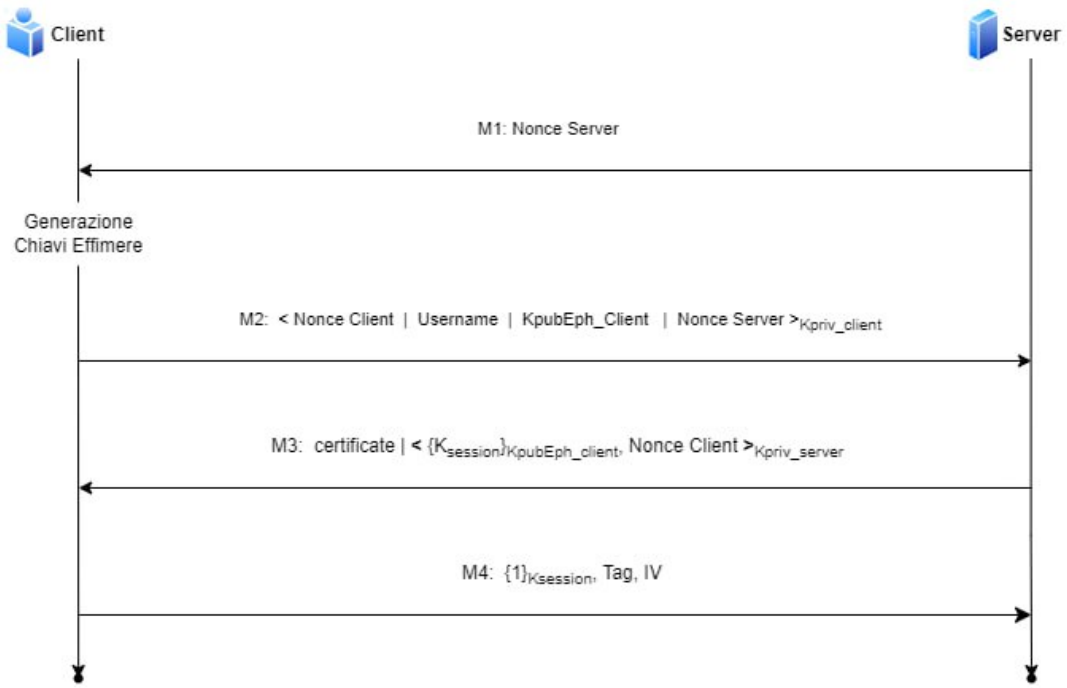

Figura 2.1: Handshake per la condivisione di una chiave di sistema

2.1 Messaggio 1

L'inizio della comunicazione tra Client e Server si ha nel momento in cui il Client crea il socket per la comunicazione. Per gestire le richieste in maniera efficiente, il Server genera un numero finito di thread (uno per ogni Client che crea un socket) che mette in una coda la cui gestione e sincronizzazione avviene con l'utilizzo del meccanismo di blocco basato su mutex. Una volta collegati tra loro tramite socket, ha inizio lo scambio di messaggi dell'handshake.

La Figura 2.2 rappresenta il primo messaggio della fase di autenticazione, inviato dal Server al Client. Il messaggio contiene al suo interno un nonce generato dal Server, così da garantire la freshness della comunicazione.

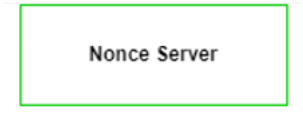

Figura 2.2: Messaggio 1

2.2 Messaggio 2

Figura 2.3: Messaggio 2

Una volta ricevuto il Nonce dal Server, il Client provvede alla generazione di una coppia di chiavi effimere (come mostrato in Figura 2.1), con cui si cifrerà la chiave di sessione. L'uso di chiavi effimere permette di garantire la perfect forward secrecy, in quanto la compromissione di una delle chiavi a lungo termine non causerà la perdità di confidenzialità di tutti i precedenti messaggi. La chiave effimera pubblica verrà utilizzata dal Server per cifrare la chiave di sessione. Il Client decifrerà la chiave di sessione, una volta ricevuta, con la chiave effimera privata.

Oltre alla chiave pubblica effimera vengono inviati al Server il Nonce Client per garantire la freshness del messaggio e il Nonce Server per confermare la ricezione del messaggio precedente. Infine troviamo l'username con cui il Client intende identificarsi. Il tutto è firmato dal Client con la sua chiave privata a lungo termine per garantire l'autenticazione. (Figura 2.3).

2.3 Messaggio 3

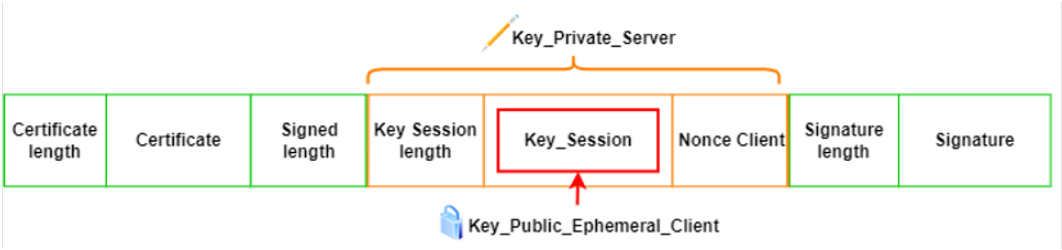

Figura 2.4: Messaggio 3

A questo punto della comunicazione il Server invia il proprio certificato (fornito dalla Certificate Autority)[1], la chiave di sessione, cifrata con la chaive pubblica effimera del Client, e il Nonce ricevuto dal Client (Figura 2.4). Inoltre, il ciphertext della chiave di sessione e Nonce vengo firmati con la chiave privata del Server, per garantire l'autenticità. All'interno del certificato si trova la chiave pubblica del Server che permette al Client di verificare la validità della firma.

2.4 Messaggio 4

Figura 2.5: Messaggio 4

[1]Per simulare il lavoro della CA (Certificate Authority) è stato utilizzato il programma 'Simple Authority'.

Una volta ricevuto il ciphertext contenente la chiave di sessione, il Client usa la propria chiave privata effimera per decifrarla. Grazie all'uso delle chiavi effimere per inviare in maniera sicura la chiave di sessione è possibile garantire la Perfect Forward Secrecy.

Adesso Client e Server si sono autenticati correttamente e possono comunicare cifrando ogni messaggio con AES-GCM 256 utilizzando la chiave di sessione. Finita la fase di handshake, le chiavi effimere vengono distrutte.

Capitolo 3

Trasferimento dati

3.1 Funzioni Send() e Receive()

La Figura 3.1 mostra la struttura dei messaggi inviati tramite la funzione 'send' e quindi ricevuti tramite la 'receive'. Nel dettaglio:

1. **Payload length**: contiene la lunghezza del ciphertext.
2. **AAD**: "Additional Authenticated Data" contiene il counter necessario per garantire protezione da replay attack. Inoltre, può contenere anche la variabile "not_last_message" che avrà valore 0 se il messaggio è l'ultimo della sequenza di send() , 1 altrimenti.
3. **Ciphertext**: è il payload del messaggio ed è cifrato con AES-GCM 256 tramite la chiave di sessione scambiata in precedenza.
4. **Tag e IV**: Il Tag è il parametro che permette di verificare l'integrità e l'autenticità dei dati e viene calcolato direttamente dall'algoritmo di cifratura AES-GCM sull'AAD e

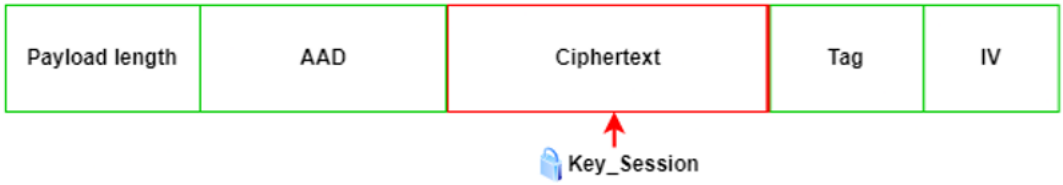

Figura 3.1: Struttura del messaggio inviato dalla send()

sul ciphertext. L'Initialization Vector(IV) è un valore casuale generato dalla funzione RAND_bytes() di OpenSSL e utilizzato poi dall'algoritmo di cifratura per ottenere ciphertext diversi anche in caso di plaintext identici.

3.1.1 Gestione dei counter

Il counter contenuto nell'AAD dei messaggi inviati tramite 'send' serve per permettere a Client e Server di constatare, per ogni messaggio, di non essere soggetti a replay attack. Per fare questo, sia Client che Server hanno un counter che viene incrementato tramite 'send' o 'receive'. Più precisamente, nella 'send' il counter del mittente viene incrementato una volta inviato il messaggio e quindi alla fine della funzione, mentre nella 'receive' viene incrementato all'inizio, subito dopo aver ricevuto il messaggio. Infine è stato realizzato un meccanismo che verifica che il valore di counter non venga incrementato dopo aver raggiunto il valore massimo di un unsigned int per evitare overflow. Nel momento in cui i counter raggiungono il valore massimo, viene eseguito il logout e deve essere eseguito di nuovo l'handshake.

3.2 Operazioni

Nei paragrafi seguenti vengono descritti i messaggi con cui sono state implementate le operazioni e quindi le possibili richieste da parte del Client di Upload, Download, Rename, Delete, List e Logout. I diversi diagrammi rappresentati per ogni operazione hanno tutti, come base comune, la richiesta di comando esposta in Figura 3.2.

3.2.1 Command Request

In Figura 3.2 è rappresentato il primo scambio di messaggi tra Client e Server per l'esecuzione di una delle sei operazioni possibili, presentate nei paragrafi successivi. Il Client invia al Server un comando e il Server ne controlla la validità e risponde tramite variabile "not_last_message": 1 se il comando è valido e può continuare la comunicazione, 0 altrimenti.

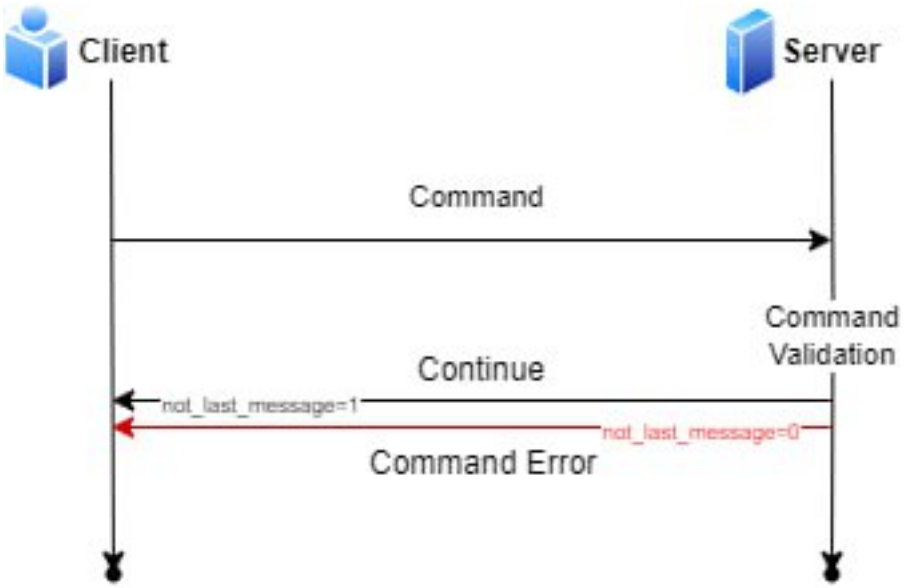

Figura 3.2: Command request

3.2.2 Upload

La prima operazione che andiamo ad analizzare è quella di Upload di un file. Dopo aver inserito il comando apposito e aver ricevuto risposta positiva dal Server, il Client provvede a inviare la dimensione del file che intende caricare, non prima, però, di aver controllato, all'interno del file system, che esista e che abbia dimensione inferiore a 4GB.

Inviata la dimensione del file al Server, vengono effettuati i controlli, anche qui, sulla dimensione e viene creato un file vuoto delle dimensioni richieste. A questo punto il Server invia una risposta al Client sempre tramite variabile "not_last_message" contenuta nell'AAD richiedendo il file o segnalando un errore e quindi la fine della comunicazione. In caso di risposta positiva dal Server, il Client frammenta il file che intende caricare (Chop File) in N chunk da (al più) 1MB l'uno e provvede poi a inviarli al Server.

3.2.3 Download

L'operazione di Download parte con la richiesta di un file da parte del Client. Il Server controlla che il file richiesto esista e che possa essere inviato. Se il check va a buon fine il Server specifica la dimensione del file al Client che a questo punto risponde con "s" se intende scaricare il file,

con “n” altrimenti. In caso di risposta affermativa il file viene frammentato e inviato in N chunk di al massimo 1MB.

3.2.4 Rename

L’operazione mostrata in Figura 3.5 è quella di rinomina di un File. Dopo la richiesta di comando “Rename” il Client invia al Server il nome del File che intende modificare e il nuovo nome da associare a questo. Se il file esiste il Server provvede a effettuare la modifica richiesta e invia il responso dell’operazione al Client che può avere anche esito negativo se il file non esiste o non è modificabile.

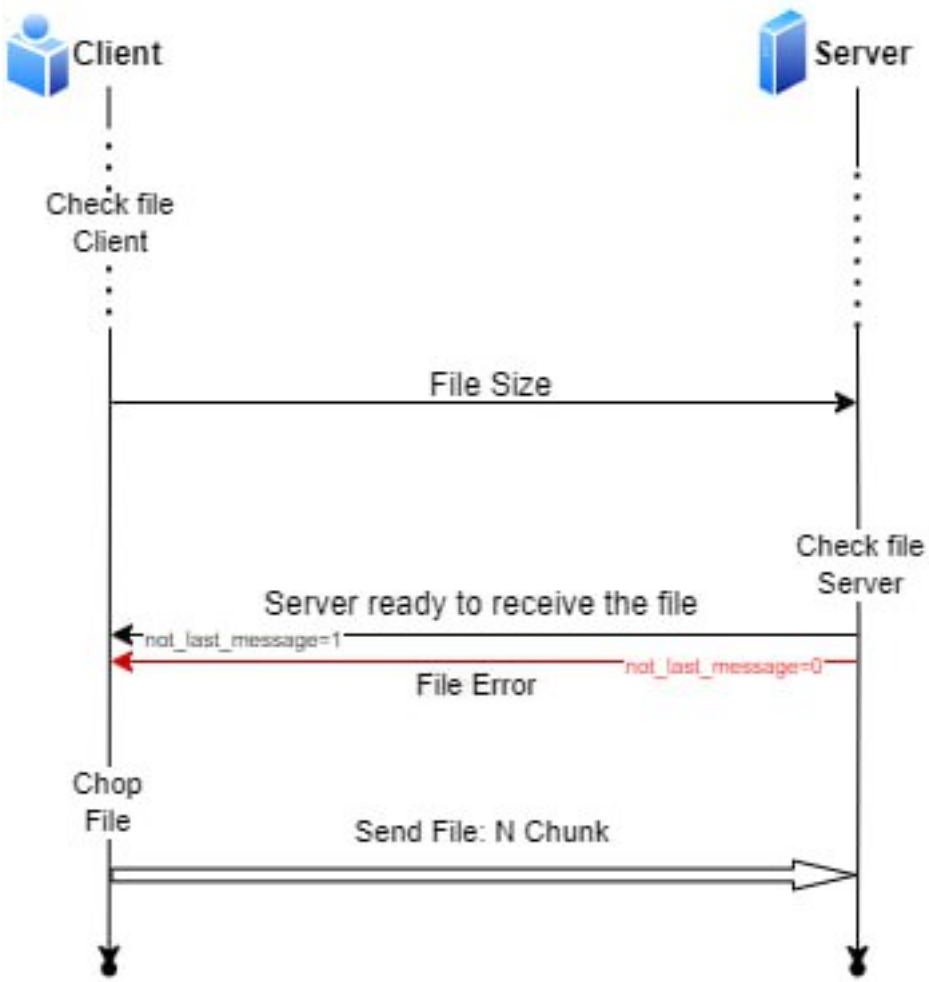

Figura 3.3: Upload

3.2.5 Delete

In Figura 3.6 è rappresentata la richiesta di eliminazione di un file. Il Client invia il nome del file che intende eliminare e il Server effettua i dovuti controlli di sicurezza, ovvero controlla se il file esiste e se si hanno i diritti per eliminarlo. Se il check va a buon fine il Server chiede al Client un'ulteriore conferma per l'eliminazione per poi inviare un responso, affermativo o negativo, ad operazione terminata.

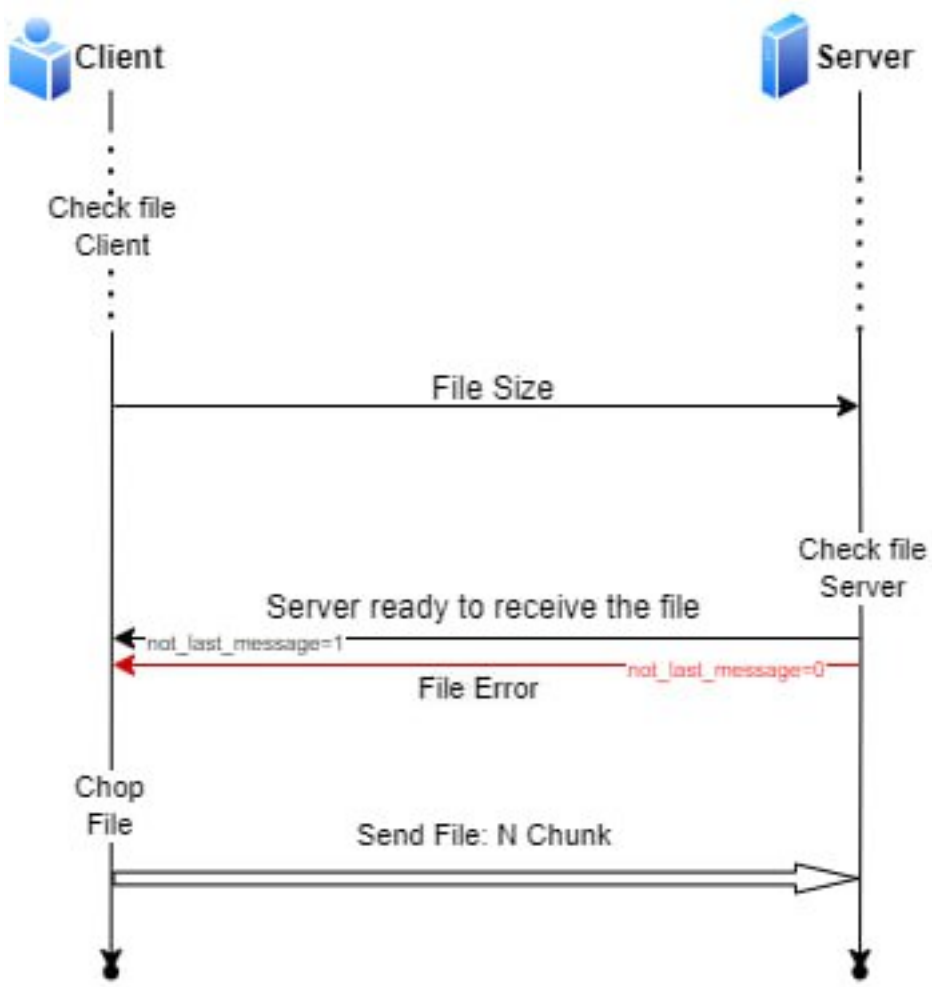

Figura 3.4: Download

Figura 3.5: Rename

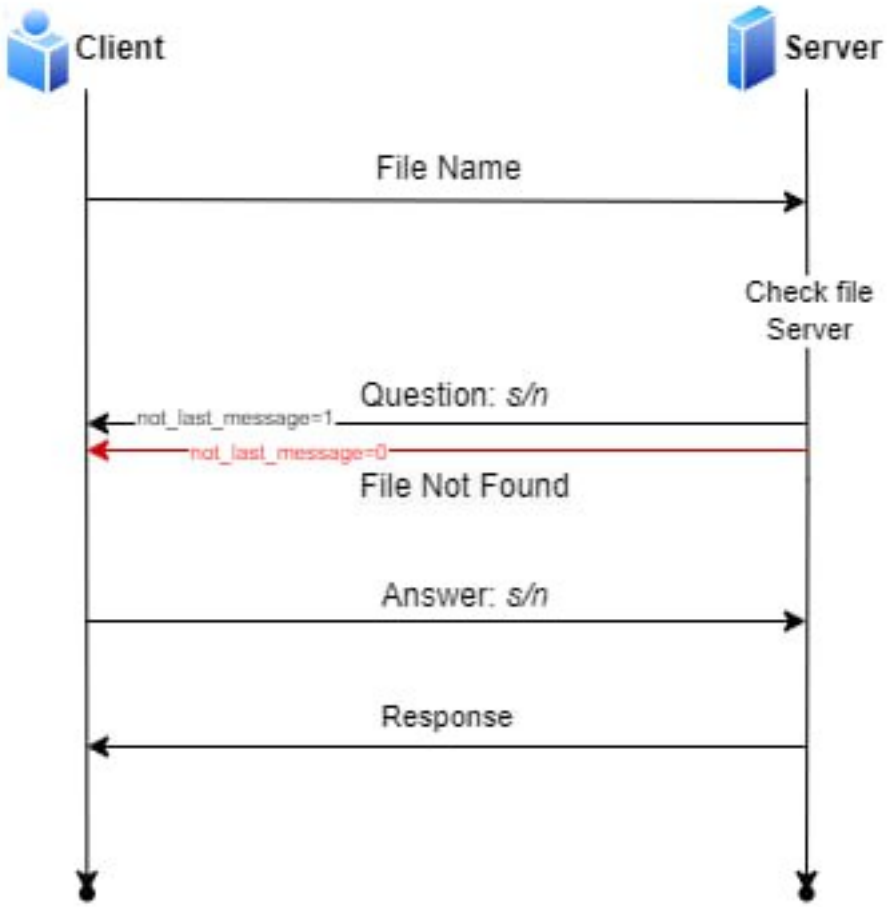

Figura 3.6: Delete

3.2.6 List e Logout

Infine, se i comandi inseriti dal Client sono “List” o “Logout” il Server provvede a inviare rispettivamente una lista di tutti i file nella cartella associata al Client. Nel primo caso, al termine, il Server si mette in attesa di un nuovo comando, mentre nel secondo caso viene chiuso il socket e vengono eliminati tutti i dati della sessione.