

# Lecture TLS

Riccardo Torre

4 dicembre 2025

**TLS** È un protocollo di sicurezza con l'obiettivo di rendere una connessione TCP sicura, autentica. Kerberos lavora al livello applicativo, mentre TLS al livello di trasporto dello stack TCP/IP. Gli attacchi interessano i Web server. Se viene fornito come parte di TCP, tutti i servizi che lavorano con TCP ne fanno uso, o fornito come una dipendenza delle applicazioni. È il discendente di SSL.

**Architettura dello stack protocollare** Record protocol stabilisce come i pacchetti devono essere spediti criptati e autenticati. L'handshake protocol permette di comunicare i parametri di crittografia per creare la connessione sicura, l'alert protocol chiude una connessione quando avviene qualcosa di anomalo (MAC scorretto, certificato non supportato, fallimento dell'handshake), Change cipher protocol per cambiare la suite crittografica.

**Record protocol** Il dato viene tagliato in frammenti, che vengono compressi, a cui vengono aggiunti un MAC, criptati e a cui si aggiunge un TLS record header.

**Sessione TLS** Associazione di criteri su come client e server devono comunicare. Viene creata con l'handshake protocol. Definisce un insieme di parametri crittografici di sicurezza che possono essere condivise tra connessioni multiple. Le sessioni TLS vengono create per evitare la negoziazione dei parametri di sicurezza per ogni connessione perché è dispendioso.

**Connessione TLS** Sfruttando le informazioni crittografiche scambiate durante l'handshake. Le connessioni sono transienti; ciascuna connessione è associata ad un'unica sessione.

**Stato di una sessione** È caratterizzata da un identificatore di sessione, certificati x509.v3, metodi di compressione, specifiche degli algoritmi crittografici, un **master secret**<sup>1</sup>, is resumable è un flag per riattivare una connessione se viene interrotta.

**Stato di una connessione** È caratterizzata dalle chiavi per calcolare i MAC del server e del client, e le relative chiavi per fare l'encryption, gli initialization vectors (stabiliti nell'handshake) che vengono utilizzati nel CBC (cipher block chain), e i sequence numbers per non perdere traccia della sequenza della comunicazione.

**Servizi del Record Protocol** Confidenzialità (AES e chiave segreta handshake protocol) e integrità (stabilendo durante l'handshake protocol una chiave condivisa per formare i MAC).

---

<sup>1</sup>chiamato così perché da esso vengono derivati diversi secret. È lungo 48 byte.

## L'handshake protocol

Serve a creare la sessione TLS. Il client e il server stabiliscono attraverso i rispettivi messaggi di hello, i parametri di sicurezza, tra cui la versione del protocollo TLS (l'ultima è la 1.3, la 1.2 è ancora utilizzata), l'ID di sessione, la suite di crittografia, il metodo di compressione e gli initial random numbers. Il client propone il suo livello di crittografia e il server se riesce a supportarlo continua a seguire il protocollo, altrimenti abortisce. Questo serve per verificare che il client abbia un livello di crittografia che rispetta le policy di sicurezza del server. Nella seconda fase, il server spedisce il proprio certificato e il client verifica se è fidato (e aggiornato). Il certificato (X509.v3) contiene la chiave pubblica del server. Il messaggio **server\_key\_exchange** viene utilizzato solo se si sceglie di utilizzare **DH effimero** (durante l'handshake si inventano dei nuovi parametri, si generano le mezze chiavi). Se si sceglie di utilizzare RSA il messaggio suddetto non viene utilizzato. RSA nell'ultima versione di TLS (la 1.3) ha la **perfect forward secrecy** mentre nella versione di TLS 1.2 è stato deprecato. Nella terza fase, il server può richiedere al client di inviargli un certificato.

Se il server aveva scelto di usare DH effimero, aveva inviato al client **in maniera autenticata** la sua mezza chiave e i parametri fondamentali del protocollo ( $G$  e  $p$ ). In maniera autenticata vuol dire che il server ha codificato la sua mezza chiave e i parametri fondamentali di DH con la sua chiave privata. Il client può recuperare dal messaggio codificato tali informazioni perché è in possesso del certificato del server ricevuto nella seconda fase, che contiene la chiave pubblica del server. Quindi il client deve inviare il **client\_key\_exchange** per formare il pre-master-secret da cui poi si deriva il master-secret. Nella quarta fase, client e server, una volta che hanno il master-secret creano una serie di chiavi seguendo il protocollo scambiandosi in fase di hello (prima fase). In questa fase client e server si stanno parando il [...] da un possibile attacco man-in-the-middle. Il client fa un hash di tutti i messaggi scambiati nelle fasi precedenti e lo firma con la chiave per l'integrity che deriva da quella scambiata con il server. Il server fa la stessa cosa comprendendo l'ultimo hash.

Questo serve ad entrambi per verificare che la comunicazione sia avvenuta in maniera integra.