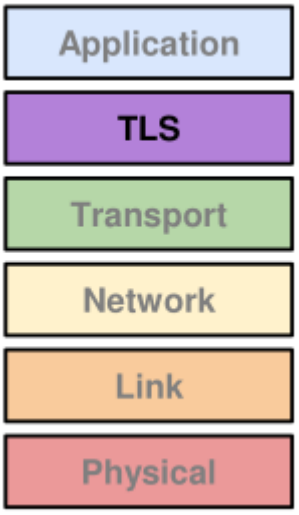
Indice generale

TLS....1
TLS security properties....2
TLS architecture....3
TLS record protocol (authenticate-then-encrypt)....4
TLS 1.2 data protection (authenticate-then-encrypt)....5
TLS – calcolo del MAC....6
TLS operation phases (high level)....6
TLS handshake protocol - summary....7
TLS session-id....8
TLS – security services....9
TLS – authentication and integrity....9
TLS – confidentiality....10
TLS – ciphersuits....11
TLS Handshake (v1.2), server authentication only: steps and messages (high-level)....11
TLS handshake step 1: exchange hellos....11
TLS Handshake Step 2: X.509v3 certificate....12
TLS Handshake Step 3: Premaster Secret....13
TLS Handshake Step 3: Premaster Secret (RSA)....13
TLS Handshake Step 3: Premaster Secret (DH)....14
TLS Handshake Step 4: Derive Symmetric Keys....15
Quando il protocollo di handshake è in esecuzione, vengono stabilite due chiavi pubbliche, chiamate pre-master secret (segreto a lungo termine) e master secret. Quando la connessione è la prima della sessione, viene generato il segreto pre-master....16
TLS Handshake Step 5: Exchange MACs....16
TLS Handshake Step 6: Send Messages....17
TLS – Come un utente può essere sicuro di parlare con un server legittimo?....17
TLS: Secure messages....18
TLS – Reply Attack....19
TLS: Replay/Filtering Attacks....20
TLS sequence numbers and TCP sequence numbers....20
Forward Secrecy in TLS....21
Perfect forward secrecy....21
Forward secrecy: with “ephemeral” mechanisms....21
TLS and virtual server: the problem....21
TLS and virtual server: the solution....22
When is TLS effective?....22

TLS

Transport Layer Security (TLS) è un protocollo crittografico utilizzato per garantire la sicurezza delle comunicazioni su una rete, in particolare su Internet. TLS opera a livello di trasporto e fornisce un canale sicuro attraverso il quale i dati possono essere trasmessi in modo crittografato tra un client e un server. L'obiettivo principale di TLS è garantire la riservatezza e l'integrità delle informazioni durante il trasferimento attraverso la rete.

- **Sostituzione di SSL:**
  - TLS sostituisce SSL (Secure Sockets Layer), che è la versione originale del protocollo. Questa sostituzione è avvenuta in quanto TLS rappresenta una versione migliorata e più sicura del protocollo di sicurezza.
- **Relazione con TCP:**
  - TLS è implementato sopra il protocollo TCP (Transmission Control Protocol) a livello di trasporto. TCP fornisce un flusso di byte astratto tra client e server, ma i segmenti TCP non sono crittograficamente protetti.
- **Obiettivo di TLS:**
  - TLS fornisce un'ulteriore astrazione del flusso di byte tra client e server. L'aspetto crittografico di TLS si concentra sulla protezione dei dati applicativi, fornendo un canale sicuro che previene accessi non autorizzati e manipolazioni durante la trasmissione.



- **Sicurezza e Obsolescenza:**
  - Le versioni precedenti di SSL/TLS (sotto TLS v1.2) sono considerate non sicure e deprecate a causa di vulnerabilità rilevate nel corso del tempo. Attualmente, le versioni più comunemente utilizzate sono TLS 1.2, ma c'è una tendenza generale a migrare verso TLS 1.3, che è una versione più recente e avanzata dal punto di vista della sicurezza.

TLS security properties

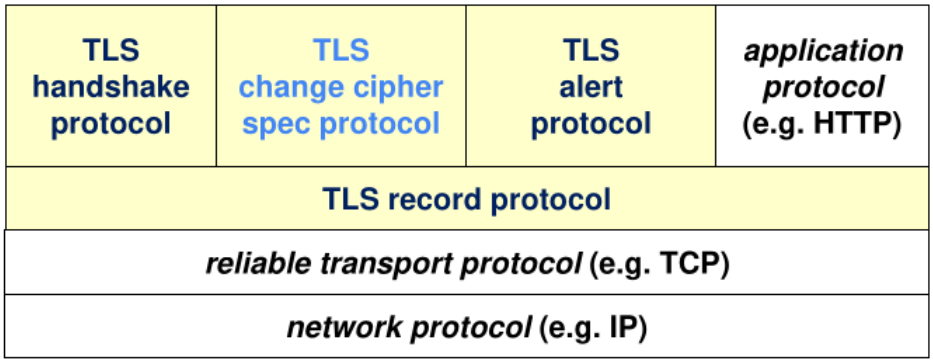
TLS crea un canale di trasporto sicuro tramite TCP (Transmission Control Protocol), collegando in modo sicuro client e server. Questo canale di comunicazione sicuro è caratterizzato da diverse proprietà di sicurezza.

- **peer authentication**, permettendo al client di verificare che la connessione avvenga con un server legittimo. Ciò è cruciale per prevenire potenziali attacchi in cui un aggressore si finge essere il server, proteggendo così la connessione.
- **client authentication** durante l'instaurazione del canale è **opzionale**, questa opzione consente al server di verificare l'identità del client, aggiungendo un livello aggiuntivo di sicurezza.
- **data confidentiality** è garantita da TLS, impedendo agli attaccanti di leggere il traffico scambiato tra client e server.

- assicura **authentication** e **integrity of exchanged messages**, prevenendo attacchi che potrebbero alterare o modificare il traffico, o inserire nuovi dati durante la trasmissione.
- **protection against replay attacks**, riordinamento e filtraggio/cancellazione del traffico scambiato. Ciò garantisce che gli aggressori non possano riprodurre, riordinare o filtrare le comunicazioni, preservando l'integrità e la coerenza delle informazioni scambiate.
- **versatility**, che consente di applicare facilmente le sue proprietà di sicurezza a vari protocolli basati su TCP, come HTTP, SMTP, NNTP, FTP, TELNET, e molti altri. Un esempio notevole è il protocollo HTTPS (HTTP sicuro), che opera su TLS sulla porta 443/TCP, garantendo la sicurezza delle comunicazioni web.

TLS architecture

L'architettura di TLS (Transport Layer Security) è composta da diversi protocolli che collaborano per fornire un canale di comunicazione sicuro.

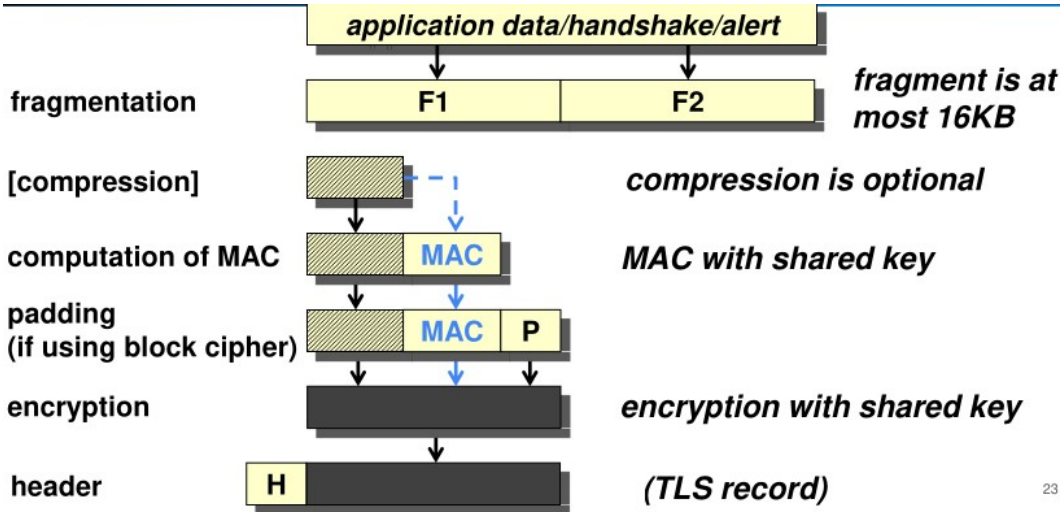


- **TLS Handshake Protocol:**
  - Questo protocollo gestisce la fase iniziale della comunicazione, durante la quale client e server stabiliscono una connessione sicura. Comprende una serie di messaggi di handshake specifici per negoziare parametri e chiavi crittografiche.
- **TLS Change Cipher Spec Protocol:**
  - Utilizzato per innescare il cambio di algoritmi e chiavi utilizzati per la protezione dei messaggi. Questo protocollo è coinvolto nella definizione dei parametri crittografici che saranno applicati durante la sessione.
- **TLS Alert (Teardown) Protocol:**
  - Questo protocollo utilizza specifici messaggi di avviso per chiudere il canale in modo controllato o segnalare errori critici. Ad esempio, può essere utilizzato per notificare una terminazione normale della connessione o segnalare un problema di sicurezza.
- **(Application) Data Protocol:**
  - Questo protocollo gestisce il flusso di dati applicativi, come ad esempio il traffico HTTP, FTP, SMTP, POP3, e altri. I dati vengono inviati attraverso il canale di comunicazione sicuro stabilito durante la fase di handshake.
- **TLS Record Protocol:**

- Questo protocollo specifica il formato per proteggere criptograficamente i dati dell'applicazione, i messaggi di handshake, gli avvisi o i messaggi di cambio di specifica di cifratura. Definisce come i dati debbano essere suddivisi in record, crittografati e trasmessi attraverso il canale sicuro.

L'interazione tra questi protocolli è fondamentale per la sicurezza complessiva della comunicazione in TLS. Durante la fase di handshake, vengono negoziati parametri crittografici e autenticazione reciproca. Il protocollo Change Cipher Spec segnala il passaggio ai parametri concordati. Successivamente, il protocollo di dati applicativi gestisce il flusso di informazioni, mentre il protocollo di avviso può essere attivato in caso di problemi o alla chiusura della connessione.

TLS record protocol (authenticate-then-encrypt)

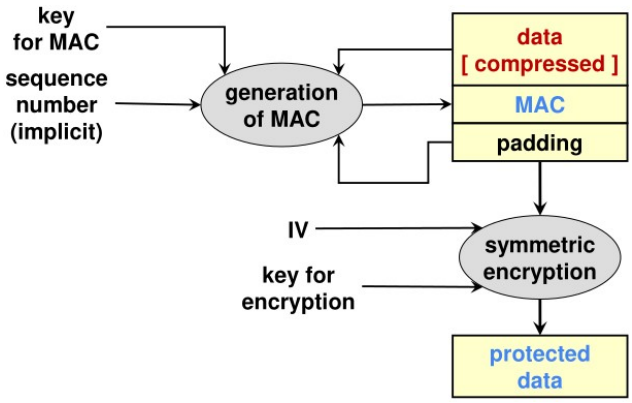


l protocollo di record di TLS (Transport Layer Security) è responsabile della trasmissione sicura dei dati attraverso il canale di comunicazione stabilito durante la fase di handshake.

1. **Fragmentation:**
   - Quando il protocollo di record è utilizzato per inviare dati applicativi, possono essere coinvolti file di grandi dimensioni, ad esempio, inviati dal server al client. Per affrontare questo scenario, il protocollo di record frammenta i dati in blocchi gestibili, o pacchetti di una lunghezza specifica.
2. **Compression (**in versioni più vecchie**):**
   - Nelle versioni più vecchie di TLS, è possibile che i dati siano stati compressi per risparmiare spazio durante la trasmissione. Tuttavia, questa pratica è stata abbandonata in alcune versioni di TLS a causa di potenziali vulnerabilità legate agli attacchi di compressione.
3. **Computation of MAC (Message Authentication Code):**
   - Viene calcolato un MAC per garantire l'autenticazione e l'integrità dei dati. Il MAC è un codice crittografico associato ai dati, che può essere verificato dalla parte ricevente per garantire che i dati non siano stati alterati durante la trasmissione.

4. **Padding (**se necessario**):**
    - Se è richiesta anche la riservatezza dei dati, può essere aggiunto del padding ai dati. Il padding è una tecnica che aggiunge dati casuali o non significativi per rendere più difficile l'analisi di eventuali modelli nei dati criptati.
5. **Encryption:**
    - Il nuovo pacchetto, composto da dati compressi (se applicabile), il MAC e il padding, viene quindi crittografato. Questo processo garantisce la confidenzialità dei dati durante la trasmissione.
6. **Header:**
    - Infine, viene aggiunta un'intestazione al pacchetto crittografato per renderlo riconoscibile come un segmento TLS all'interno del protocollo TCP. Questo consente alle entità di comunicare in modo sicuro all'interno del canale di trasporto.

TLS 1.2 data protection (authenticate-then-encrypt)



1. **Accordo sugli Algoritmi e Scambio di Numeri Casuali:** durante il protocollo di handshake, client e server concordano su un insieme di algoritmi per garantire riservatezza e integrità. Inoltre, avviene lo scambio di numeri casuali tra le due parti, che verranno utilizzati per le successive operazioni di generazione delle chiavi.

1. **Stabilire una Chiave Simmetrica:** utilizzando operazioni a chiave pubblica come RSA o DH (Diffie-Hellman), client e server stabiliscono una chiave simmetrica condivisa. Questa chiave sarà utilizzata per la crittografia simmetrica dei dati durante la comunicazione.

2. **Negoziazione dell'ID di Sessione e Scambio dei Certificati:** l'ID di sessione viene negoziato per evitare la rinegoziazione dei parametri di sicurezza quando si contatta più volte lo stesso server. Inoltre, avviene lo scambio dei certificati necessari per garantire l'autenticazione reciproca e stabilire l'identità delle parti coinvolte nella comunicazione.

3. **Protezione dei Dati:** ora che la chiave simmetrica è stata stabilita, vengono protetti i dati (eventualmente compressi) scambiati tra client e server. Per fare ciò, viene calcolato un MAC (Message Authentication Code) prendendo in considerazione i dati, la chiave, il numero di sequenza implicito e l'eventuale padding. Il MAC viene quindi inserito nel pacchetto.

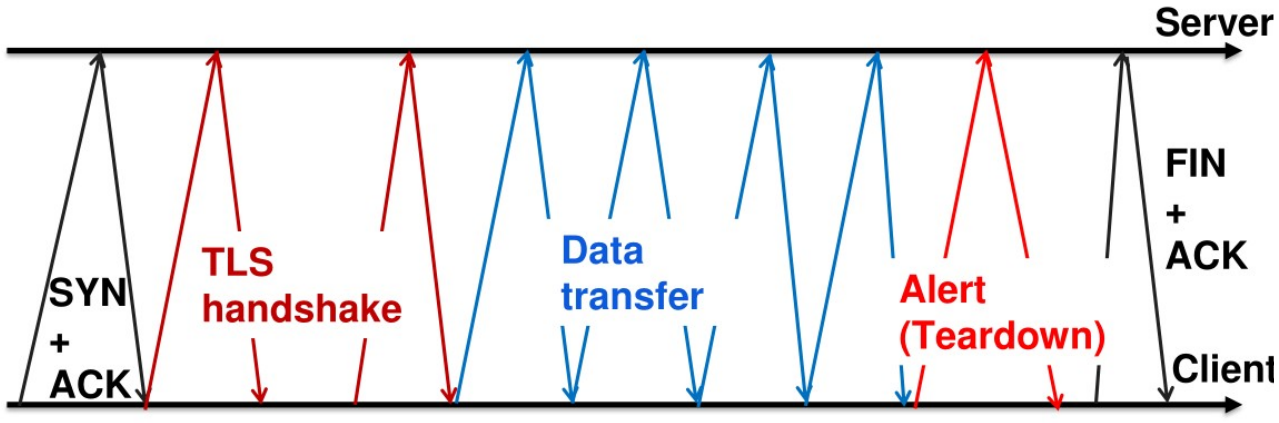
4. **Crittografia Simmetrica:** il pacchetto, contenente il MAC, viene crittografato simmetricamente utilizzando una specifica chiave e un IV (Initialization Vector), con particolare attenzione al modo in cui avviene la crittografia, ad esempio, nel modo di funzionamento a blocchi come CBC (Cipher Block Chaining).
5. **Header SSL:** infine, l'intestazione SSL viene aggiunta al pacchetto crittografato. Questa intestazione fornisce informazioni necessarie per la corretta interpretazione del pacchetto all'interno del flusso di dati.

TLS – calcolo del MAC

MAC = message_digest (key, seq_number || type || version || length || fragment)

- **message_digest**: Tipicamente, HMAC-SHA256 o superiore
- **key**: La chiave utilizzata per generare il MAC. Questa chiave può essere la client_write_MAC_key o la server_write_MAC_key, a seconda di chi sta generando il MAC (client o server).
- **seq_number**: Un numero di sequenza a 64 bit. Questo valore inizia da 0 per una nuova connessione e viene implicitamente calcolato sia dal client che dal server (non può eccedere 2**64-1). Non viene mai trasmesso sulla rete ma è utilizzato internamente per garantire l'unicità di ciascun MAC.
- **type**: Indica il tipo di record o messaggio TLS al quale il MAC viene applicato. Può essere "application data", "handshake", "alert", o "change_cipher_spec".
- **version**: La versione del protocollo TLS in uso.
- **length**: La lunghezza del frammento o del messaggio a cui viene applicato il MAC.

TLS operation phases (high level)



1. **TCP Connection (3-way Handshake):**

- La connessione TCP inizia con un processo noto come "3-way handshake". Questo è il modo in cui due entità, ad esempio un client e un server, stabiliscono una connessione affidabile.

- **SYN (Synchronize):** Il client invia un pacchetto SYN al server per indicare l'intenzione di stabilire una connessione.
- **SYN-ACK (Synchronize-Acknowledge):** Il server risponde con un pacchetto SYN-ACK per indicare che è disposto a stabilire la connessione.
- **ACK (Acknowledge):** Infine, il client risponde con un pacchetto ACK, confermando la volontà di stabilire la connessione. Ora la connessione TCP è stabilita.

2. **TLS Handshake:**

- Una volta che la connessione TCP è stabilita, inizia il processo di handshake TLS.
  - **Autenticazione:** Durante questa fase, il server si autentica verso il client e, opzionalmente, anche il client può autenticarsi. Questo passo assicura che le parti coinvolte siano chi affermano di essere.
  - **Negoziazione degli Algoritmi Crittografici:** Le entità concordano sugli algoritmi crittografici da utilizzare per lo scambio/chiarimento delle chiavi e la protezione dei dati (integrità tramite MAC e crittografia).
  - **Stabilire Chiavi:** Viene stabilita una chiave condivisa tra il client e il server, che verrà utilizzata per la crittografia simmetrica durante la comunicazione successiva.

3. **(Application) Data Transfer:**

- Dopo il completamento dell’handshake, inizia il trasferimento dei dati applicativi. I dati scambiati tra client e server sono ora protetti utilizzando le chiavi concordate durante l’handshake TLS.

4. **TLS Teardown:**

- La fase di "teardown" si verifica quando la connessione TLS deve essere chiusa. Ciò può essere causato dalla volontà dell'applicazione di chiudere la connessione o da errori durante l’handshake TLS o il trasferimento dei dati.
  - **Chiusura Normale:** Se la connessione viene chiusa normalmente, l'applicazione invia un segnale di chiusura, e le due parti procedono a chiudere la connessione in modo ordinato.
  - **Chiusura a Causa di Errori:** In caso di errori durante l’handshake o il trasferimento dei dati, può essere innescata una chiusura della connessione.

TLS handshake protocol - summary

1. **Exchange random numbers:**

   Client e server si scambiano numeri casuali durante il handshake per generare successivamente le chiavi crittografiche necessarie. Questi numeri casuali includono il "client random" e il "server random", e vengono generati in modo casuale per ogni connessione.

2. **Negotiate the session-id:**

   In versioni precedenti di TLS (prima della 1.2), veniva negoziato un identificatore di sessione (session-ID) di 32 byte. Questo identificatore veniva selezionato dal server per gestire in modo efficiente la rinegoziazione dei parametri di sicurezza quando si contattava più volte lo stesso server.

3. **Agree on set of algorithms (Cipher Suites):**

Le parti concordano su un insieme di algoritmi, noti come "cipher suites", per proteggere i messaggi e scambiare le chiavi. Questi algoritmi determinano come verrà gestita la riservatezza, l'autenticazione dei dati e l'integrità durante la comunicazione.

4. **Exchange of X.509 certificates:**

   Il server invia obbligatoriamente il suo certificato X.509 al client per autenticare la sua identità. Il client può opzionalmente rispondere con il proprio certificato per autenticare a sua volta la sua identità.

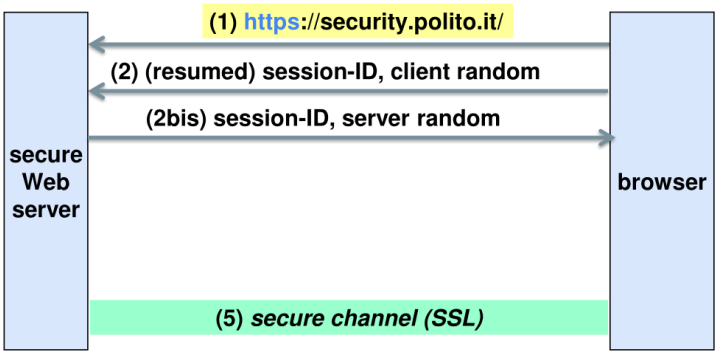
5. **Exchange of the keys (Key Exchange/Agreement):**

   Viene eseguito uno scambio di chiavi per stabilire un segreto, chiamato "**master secret**". Questo segreto deriva da un altro segreto chiamato "**pre-master secret**", che viene scambiato utilizzando operazioni di chiave pubblica, come DH (Diffie-Hellman) o RSA. La chiave segreta condivisa sarà la base per la generazione di altre chiavi crittografiche.

6. **Derivation of Symmetric keys and IVs:**

   Dal "master secret", insieme al "client random" e al "server random", client e server derivano attraverso PRF le chiavi crittografiche simmetriche (per MAC e crittografia) e vettori di inizializzazione per la protezione dei dati. Questi valori sono **distinti per ogni connessione** e vengono utilizzati per garantire la confidenzialità, l'autenticazione e l'integrità dei dati scambiati.

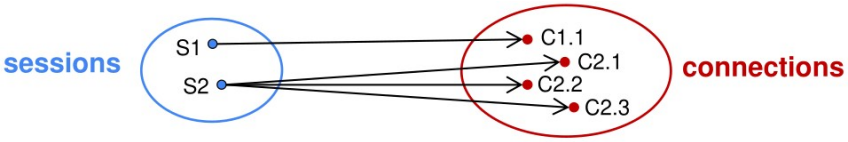
TLS session-id



I canali vengono continuamente aperti e chiusi! Questo è particolarmente stressante quando si usa TLS per aprire i canali, perché se i parametri crittografici di TLS devono essere negoziati ogni volta, il carico computazionale (operazioni a chiave pubblica) diventa molto elevato. Per evitare di rinegoziare tutti i parametri crittografici per ogni connessione TLS, il server TLS può inviare un identificatore permanente chiamato **identificatore di sessione** (più connessioni possono far parte della stessa sessione logica): si tratta di un modo per rappresentare un insieme di algoritmi e chiavi che sono stati negoziati in un certo momento.

Se il client, quando apre la connessione TLS, prima di iniziare l'handshake, invia un session-id valido (già negoziato in precedenza), la parte di negoziazione viene saltata e i dati vengono immediatamente scambiati sul canale sicuro. Si noti che il server può rifiutare l'uso del session-id (sempre o dopo un certo tempo dalla sua emissione).

Una sessione TLS rappresenta un'associazione logica tra un client e un server, definita durante il protocollo di handshake iniziale. Questa sessione stabilisce parametri crittografici, come algoritmi e chiavi, condivisi da una o più connessioni TLS.



D'altro canto, una connessione TLS è un canale transitorio tra client e server, associato a una specifica sessione TLS. Per ottimizzare l'efficienza, il server TLS può inviare un identificatore di sessione, rappresentando un insieme di parametri precedentemente negoziati. Quando un client apre una connessione, può inviare un session-id valido, saltando la rinegoziazione e consentendo uno scambio immediato di dati su un canale sicuro. Tuttavia, la gestione accurata del tempo di conservazione delle sessioni è essenziale per evitare sovraccarichi del server. Questa dinamica offre flessibilità nell'apertura di canali sicuri, migliorando l'efficienza senza compromettere la sicurezza.

TLS – security services

I servizi offerti da TLS mirano a garantire una comunicazione sicura su Internet, fornendo diversi livelli di protezione.

- **Server Authentication (Mandatory):**

  Nel contesto di TLS, l'autenticazione del server è obbligatoria. Il server autentica la sua identità nei confronti del client attraverso l'uso di certificati a chiave pubblica e una sfida/risposta asimmetrica. Ciò garantisce al client che sta comunicando con il server legittimo, impedendo a un attaccante di impersonare il server.

- **Client Authentication (Optional, if required by server):**

  L'autenticazione del client è un servizio opzionale e può essere richiesta dal server. In questo caso, il client si autentica nei confronti del server utilizzando un certificato a chiave pubblica e un processo di asymmetric challenge response. Questo livello aggiuntivo di autenticazione può essere utile in scenari in cui è necessario verificare l'identità del client.

- **Data protection:**
  - **Confidentiality:** TLS garantisce la confidenzialità dei dati scambiati tra client e server. Questo significa che anche se un aggressore riesce ad intercettare la comunicazione, non sarà in grado di comprendere il contenuto a causa della crittografia.
  - **Message (data) Integrity and Authentication:** I dati scambiati sono protetti da meccanismi che garantiscono l'integrità e l'autenticazione. Ciò impedisce a un attaccante di modificare il traffico o di inserire dati non autorizzati durante la trasmissione.
  - **Reliability:** TLS offre protezione contro eventi come riordinamento, replay o cancellazione di messaggi. Questo garantisce che il flusso di dati sia coerente e sicuro da manipolazioni.
- **Efficiency:**

  TLS consente la ripresa di sessioni in nuove connessioni senza la necessità di ripetere l'intero processo di handshake. Questo è possibile grazie all'uso di identificatori di sessione, che

permettono al client di riutilizzare parametri crittografici precedentemente negoziati, migliorando l'efficienza e riducendo il carico computazionale.

TLS – authentication and integrity

Il processo di autenticazione durante la configurazione del canale coinvolge la presentazione dei certificati X.509v3 da parte di entrambi i peer e risposte a sfide asimmetriche. Durante la comunicazione, l'autenticazione e l'integrità dei dati sono mantenute attraverso l'uso di codici di autenticazione dei messaggi (MAC), numeri di identificazione dei messaggi (MID) e l'uso di algoritmi e chiavi simmetriche precedentemente concordate durante il protocollo di handshake di TLS.

- **Peer Authentication (at channel setup):** Durante la configurazione del canale, si verifica l'autenticazione reciproca dei peer:
  - **Server:** Il server si autentica inviando il suo certificato X.509v3, che contiene la sua chiave pubblica. Risponde anche a una sfida/risposta asimmetrica implicita.
  - **Client:** L'autenticazione del client avviene mediante la presentazione del suo certificato X.509v3 e rispondendo esplicitamente a una asymmetric challenge response.
- **Authentication and Integrity of the data exchanged (during the comunication):** Per garantire l'autenticazione e l'integrità dei dati durante lo scambio sul canale sicuro, il protocollo TLS record utilizza diverse tecniche:
  - **MAC (Message Authentication Code):** Viene utilizzato un codice di autenticazione del messaggio (MAC) per proteggere i dati. Questo è un digest basato su una chiave, comunemente implementato con HMAC-SHA256 o una versione più avanzata.
  - **MID (Message ID - seq_number):** Un numero di identificazione del messaggio (MID o seq_number) viene utilizzato per evitare attacchi di replay e cancellazione. Il MID è un numero di sequenza implicito che inizia da 0 per una nuova connessione e viene implicitamente calcolato da client e server.
  - **Algorithms and Symmetric keys for HMAC calculation:** Gli algoritmi e le chiavi simmetriche utilizzate per il calcolo di HMAC sono negoziati durante la fase di handshake di TLS. Ciò assicura che entrambe le parti siano d'accordo sugli algoritmi e sulle chiavi da utilizzare per proteggere l'autenticazione e l'integrità dei dati scambiati.

TLS – confidentiality

Il concetto di confidenzialità riguarda la protezione dei dati durante la trasmissione attraverso un canale sicuro. Il processo di confidenzialità in TLS comporta la selezione di un algoritmo di cifratura simmetrica durante la negoziazione, il successivo scambio di chiavi simmetriche attraverso la crittografia asimmetrica, e l'uso di queste chiavi per cifrare e decifrare i dati durante la trasmissione. L'introduzione di TLS v1.3 ha contribuito a migliorare la sicurezza abbandonando algoritmi meno sicuri a favore di ciphers AEAD più robusti.

1. **Client and server negotiate symmetric algorithm to be used for data encryption:** Client e server negoziano un algoritmo simmetrico da utilizzare per la cifratura dei dati scambiati. Questo algoritmo è responsabile della conversione dei dati in una forma crittografata durante la trasmissione e della loro decifrazione alla ricezione. Gli algoritmi possono essere di tipo blocco,

come AES (Advanced Encryption Standard) o 3DES (Triple Data Encryption Standard), o, a partire da TLS v1.2, ciphers AEAD (Authenticated Encryption with Associated Data), come AES in GCM (Galois/Counter Mode) o CCM (Counter with CBC-MAC).

2. **Client and server perform** key exchange/agreement**:** Prima di iniziare la cifratura dei dati, client e server eseguono uno scambio o concordano sulle chiavi simmetriche da utilizzare. Questo processo avviene attraverso la crittografia asimmetrica, in particolare con protocolli come il Diffie-Hellman. Le chiavi simmetriche risultanti da questo scambio verranno utilizzate per la cifratura e la decifrazione dei dati durante la comunicazione.

Con l'introduzione di TLS v1.3, sono stati apportati miglioramenti significativi alla sicurezza e all'efficienza. In questa versione, vengono supportati esclusivamente i ciphers AEAD, rinunciando a opzioni meno sicure come RC4 (Rivest Cipher 4), che era ancora possibile in TLS v1.2.

TLS – ciphersuits

Le "ciphersuites" sono specifiche di stringhe che definiscono l'insieme di algoritmi utilizzati per stabilire una connessione sicura tra client e server. Ogni "ciphersuite" specifica i seguenti componenti:

1. **Key Exchange Algorithm:** indica l'algoritmo utilizzato per scambiare le chiavi simmetriche tra il client e il server (ad esempio RSA).
2. **Symmetric Encryption Algorithm:** specifica l'algoritmo utilizzato per cifrare e decifrare i dati durante la comunicazione (ad esempio AES).
3. **Hash Algorithm for MAC (Message Authentication Code):** indica l'algoritmo utilizzato per generare il codice di autenticazione dei messaggi (MAC), che viene utilizzato per garantire l'integrità dei dati scambiati (ad esempio SHA, Secure Hash Algorithm).

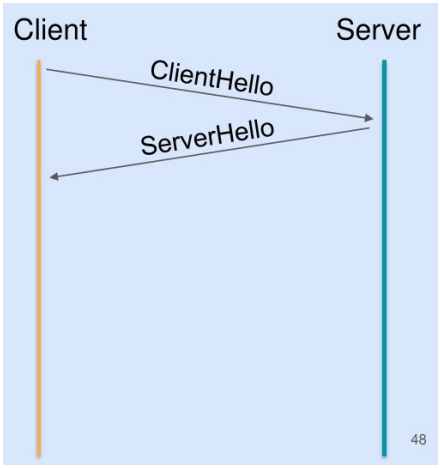
Ecco alcuni esempi di "ciphersuites" e la loro interpretazione:

1. **SSL_NULL_WITH_NULL_NULL:** questa ciphersuite indica l'assenza di cifratura, scambio chiavi e algoritmo di hash. È una ciphersuite non sicura e obsoleta.
2. **SSL_RSA_WITH_NULL_SHA:** utilizza RSA per lo scambio chiavi e NULL per la cifratura simmetrica. SHA è utilizzato come algoritmo di hash.
3. **SSL_RSA_WITH_3DES_EDE_CBC_SHA:** utilizza RSA per lo scambio chiavi, 3DES (Triple Data Encryption Standard) per la cifratura simmetrica e SHA per l'algoritmo di hash.
4. **TLS_RSA_WITH_AES_128_CBC_SHA:** utilizza RSA per lo scambio chiavi, AES con una chiave di lunghezza 128 bit per la cifratura simmetrica e SHA per l'algoritmo di hash.
5. **TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384:** utilizza ECDHE_RSA per lo scambio chiavi, AES con una chiave di lunghezza 256 bit in modalità GCM (Galois/Counter Mode) per la cifratura simmetrica e SHA-384 per l'algoritmo di hash.
6. **TLS_AES_128_GCM_SHA256:** utilizza AES con una chiave di lunghezza 128 bit in modalità GCM per la cifratura simmetrica e SHA-256 per l'algoritmo di hash.

TLS Handshake (v1.2), server authentication only: steps and messages (high-level)

TLS handshake step 1: exchange hellos

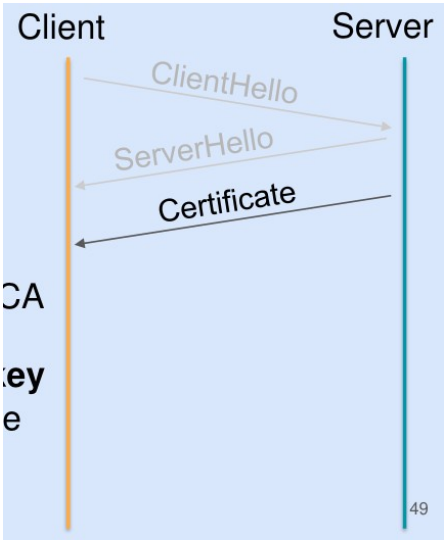
1. **Il Client invia il messaggio ClientHello:** Il processo inizia quando il client invia un messaggio chiamato "ClientHello" al server. Questo messaggio contiene due componenti principali:
   - **Numero Casuale del Cliente (Rc):** Un numero casuale di 256 bit noto come "client random".
   - **Lista di Algoritmi Supportati:** Il client invia al server una lista di algoritmi crittografici, noti come "cipher suites", che sono supportati dal client per la comunicazione sicura. Questi algoritmi includono quelli per lo scambio di chiavi, la cifratura simmetrica e l'algoritmo di hash.



2. **Il Server risponde con il messaggio ServerHello:** In risposta al messaggio ClientHello, il server invia il messaggio "ServerHello" al client. Questo messaggio contiene:
   - **Numero Casuale del Server (Rs):** Un numero casuale di 256 bit noto come "server random".
   - **Selezione della Cipher Suite:** Il server seleziona un insieme di algoritmi crittografici dalla lista inviata dal client (cipher suites) e informa il client della scelta.
3. **Prevenzione degli Attacchi di Ripetizione:** I numeri casuali del client (Rc) e del server (Rs) svolgono un ruolo importante nella prevenzione degli attacchi di ripetizione. Poiché sono generati casualmente per ogni sessione TLS, impediscono a un attaccante di registrare e riprodurre una sequenza di handshake per tentare un attacco di ripetizione.

TLS Handshake Step 2: X.509v3 certificate

4. **Il Server invia il suo certificato X.509v3:** Dopo il passaggio del ServerHello, il server invia al client il suo certificato X.509v3. Questo certificato contiene due componenti principali:
   - **Identità del Server:** Il certificato contiene informazioni sull'identità del server, come il nome del dominio associato al server.
   - **Chiave Pubblica del Server:** Il certificato include la chiave pubblica del server, che sarà utilizzata per stabilire la connessione sicura.



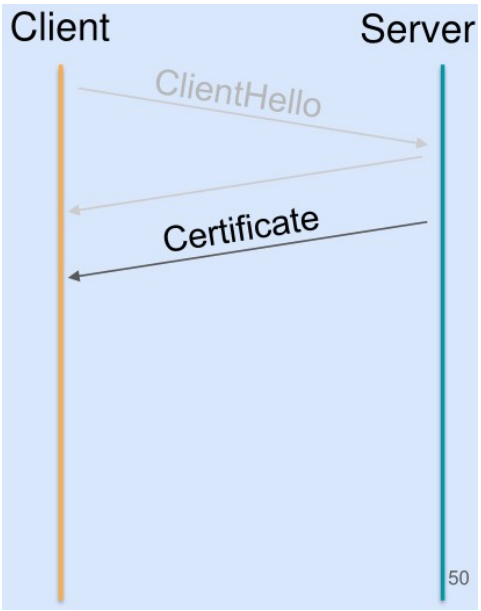
5. **Validazione del Certificato da parte del Client:** Il client è responsabile di validare il certificato del server. Questa validazione include diversi aspetti:
   - **Verifica della Firma:** Il client verifica la firma digitale presente nel certificato utilizzando la chiave pubblica della Certificate Authority (CA) che ha emesso il certificato del server.
   - **Revoca e Scadenza:** Il client verifica che il certificato non sia stato revocato e che non sia scaduto.

- **Catena di Certificati:** Tipicamente, viene inviata l'intera catena di certificati, dal certificato del server al certificato della CA radice. Il client verifica la validità di tutta la catena.

6. **Conseguenze della Validazione:** Se la validazione ha successo, il client acquisisce la chiave pubblica del server dal certificato. Tuttavia, a questo punto, il client non può essere completamente sicuro che stia comunicando con il server legittimo. Questo perché i certificati sono di dominio pubblico, e chiunque può ottenere un certificato per qualsiasi entità, inclusi attaccanti malevoli.
7. **Necessità della CA Radice/Attendibile:** Affinché la validazione abbia successo, il client deve avere già la chiave pubblica della CA radice nel suo insieme di certificati attendibili. Questa chiave pubblica è utilizzata per verificare la firma della CA che ha emesso il certificato del server. La fiducia nella CA radice è fondamentale per garantire che il certificato del server sia autentico.

TLS Handshake Step 3: Premaster Secret

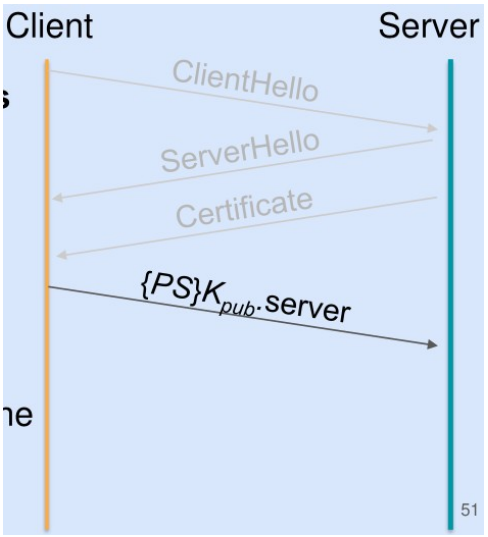
8. **Autenticazione del Server:** durante questa fase, il server deve dimostrare al client che possiede la chiave privata corrispondente alla chiave pubblica contenuta nel certificato inviato al client nella fase precedente. Questo passaggio è fondamentale per garantire che il client stia comunicando con il server legittimo e non con un server malevolo. La dimostrazione di possesso della chiave privata è spesso effettuata attraverso un processo crittografico basato sulla firma digitale.



9. **Scambio/Accordo del Segreto (Premaster Secret):** parallelamente all'autenticazione del server, il client e il server si impegnano nello scambio o nell'accordo di un segreto noto come "premaster secret". Questo segreto sarà alla base della generazione delle chiavi simmetriche che saranno utilizzate per cifrare e decifrare i dati durante la sessione di comunicazione. Due approcci principali sono utilizzati per scambiare il premaster secret:
   - **RSA (Rivest-Shamir-Adleman):** In questa opzione, il client genera casualmente un premaster secret, lo cifra con la chiave pubblica del server e lo invia al server. Solo il server, possedendo la chiave privata corrispondente, può decifrare il premaster secret.
   - **Diffie-Hellman (DH):** In alternativa, il client e il server possono utilizzare il protocollo Diffie-Hellman per scambiare pubblicamente informazioni e derivare in modo indipendente lo stesso segreto condiviso. Questo processo consente alle entità di stabilire un segreto senza mai scambiare direttamente il premaster secret attraverso la rete.

TLS Handshake Step 3: Premaster Secret (RSA)

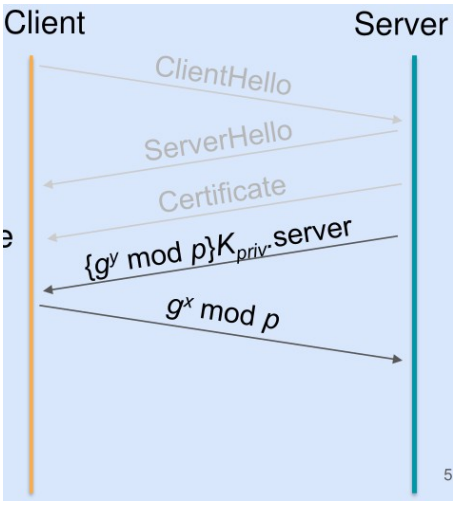
10. **Generazione del Premaster Secret (PS) da Parte del Client:** il client genera casualmente un segreto noto come "Premaster Secret" (PS). Questo premaster secret sarà alla base della creazione delle chiavi simmetriche utilizzate per cifrare la comunicazione tra il client e il server.
11. **Cifratura del Premaster Secret con la Chiave Pubblica del Server:** il client utilizza la chiave pubblica del server (Kpub.server), precedentemente ottenuta dal certificato inviato dal server, per cifrare il premaster secret (PS). La chiave pubblica è nota a tutti e può essere utilizzata per cifrare i dati in modo sicuro, ma solo la chiave privata corrispondente può decifrare tali dati.



12. **Invio del Premaster Secret Cifrato al Server:** il client invia il premaster secret cifrato al server tramite il canale di comunicazione sicuro. Questo passaggio è fondamentale per stabilire un segreto condiviso tra il client e il server.
13. **Decifratura del Premaster Secret da Parte del Server:** il server, possedendo la chiave privata corrispondente (Kpriv.server) alla chiave pubblica utilizzata dal client per la cifratura, può decifrare il premaster secret (PS). Solo il server legittimo, che possiede la chiave privata corrispondente alla chiave pubblica utilizzata per la cifratura, può effettuare questa operazione con successo.
14. **Condivisione del Premaster Secret tra Client e Server:** una volta che il server ha decifrato con successo il premaster secret, entrambe le parti (client e server) condividono ora lo stesso segreto (PS). Questo dimostra che il server è legittimo, in quanto solo il server corretto, possedendo la chiave privata, sarebbe in grado di decifrare il premaster secret correttamente.

TLS Handshake Step 3: Premaster Secret (DH)

15. **Generazione del Segreto DH da Parte del Server:** il server genera un segreto Diffie-Hellman, indicato come y, e calcola il valore g^y mod p. In questa espressione, g rappresenta un generatore, y è il segreto generato dal server e p è un numero primo noto a entrambe le parti.
16. **Firma e Invio del Messaggio al Client:** il server firma il valore g^y mod p con la propria chiave privata e invia sia il valore g^y mod p che la firma al client.
17. **Verifica della Firma da Parte del Client:** il client riceve il messaggio dal server contenente g^y mod p e la firma. Il client verifica la firma utilizzando la chiave pubblica del server, dimostrando che il messaggio proviene effettivamente dal server e che il server possiede la chiave privata corrispondente.



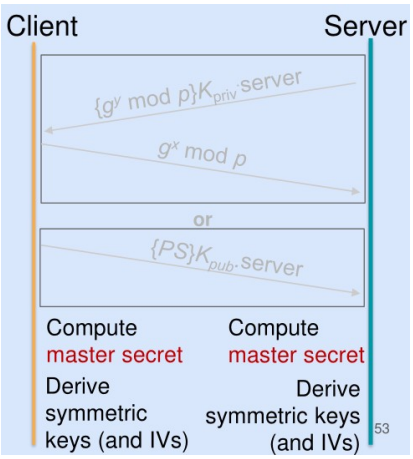
18. **Generazione del Segreto DH da Parte del Client:** il client genera il proprio segreto Diffie-Hellman, indicato come x, e calcola il valore g^x mod p.

19. **Calcolo del Premaster Secret Condiviso:** entrambe le parti, il client e il server, utilizzano il segreto generato da ciascuna parte per calcolare un premaster secret condiviso. Il premaster secret condiviso è calcolato come g^xy mod p, dove g è il generatore, x è il segreto del client e y è il segreto del server.

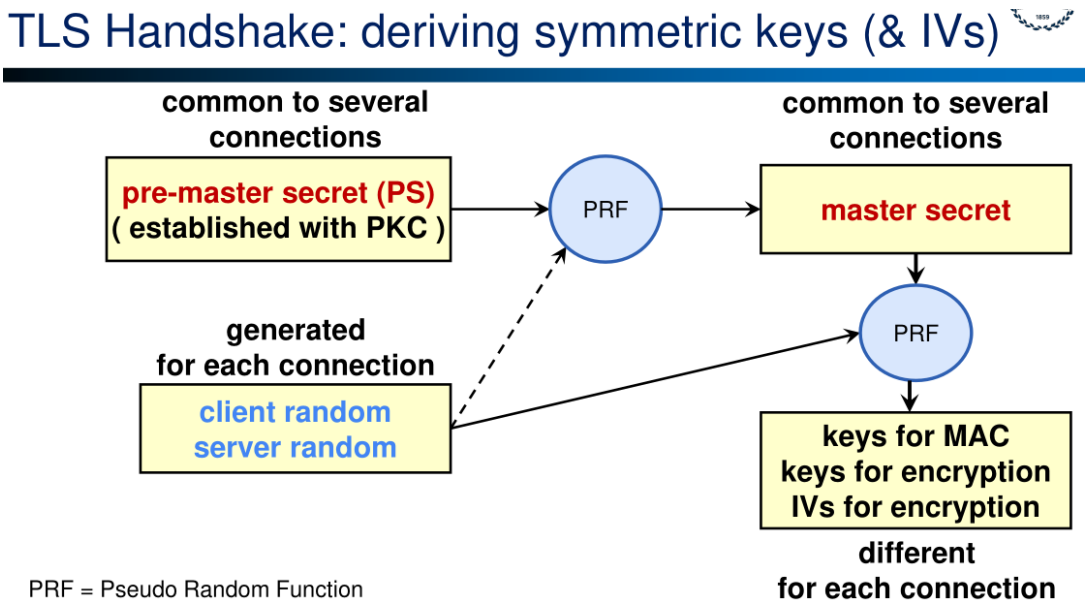
TLS Handshake Step 4: Derive Symmetric Keys

20. **Derivazione del Master Secret:** entrambe le parti, il client e il server, utilizzano un algoritmo di PRF (Pseudo-Random Function) per derivare un "master secret" dal "premaster secret" (PS) precedentemente scambiato, insieme ai valori casuali del client (Rc) e del server (Rs). Cambiando anche uno solo di questi valori, si otterrà un "master secret" diverso.

21. **Derivazione delle Chiavi Simmetriche:** dal "master secret" e dai valori casuali del client (Rc) e del server (Rs), il client e il server derivano:
    - **2 chiavi di cifratura** (Kenc_c e Kenc_s): utilizzate per cifrare i messaggi dal client al server e viceversa.
    - **2 chiavi di autenticazione del messaggio** (KIA_c e KIA_s): utilizzate per calcolare il codice di autenticazione del messaggio (MAC) per i messaggi dal client al server e viceversa.
    - **2 Vettori di Inizializzazione** (IV): uno per ogni lato della comunicazione. Gli IV sono utilizzati insieme alle chiavi di cifratura per garantire una cifratura più robusta.



22. **Condivisione delle Chiavi Simmetriche:** entrambe le parti coinvolte nella comunicazione (client e server) conoscono ora tutte e quattro le chiavi simmetriche (due per la cifratura e due per l'autenticazione dei messaggi) e gli IV associati.

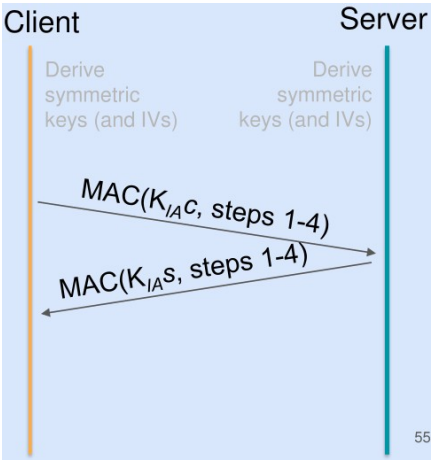


Quando il protocollo di handshake è in esecuzione, vengono stabilite due chiavi pubbliche, chiamate pre-master secret (segreto a lungo termine) e master secret. Quando la connessione è la prima della sessione, viene generato il segreto pre-master.

Quindi, per ogni connessione, compresa la prima, il client e il server generano un numero casuale che viene combinato con il pre-master secret per creare il master secret (una sorta di operazione simmetrica, come la funzione di derivazione della chiave). Per ogni connessione, le chiavi per il MAC, le chiavi per la crittografia e per il MAC, le chiavi per la crittografia e l'IV per la crittografia. Il master secret con i valori casuali che sono diversi per ogni connessione e vengono scambiati in chiaro. L'unica cosa persistente nella stessa sessione è il master secret: la prima volta sarà generato a partire dal pre-master secret e dai numeri casuali che vengono generati ogni volta che c'è una nuova connessione.

TLS Handshake Step 5: Exchange MACs

23. **Scambio dei Codici di Autenticazione del Messaggio (MACs):**
    - Il client e il server calcolano e scambiano i codici di autenticazione del messaggio (MACs) per tutti i messaggi scambiati durante la procedura di handshake di TLS fino a questo momento.
    - I codici di autenticazione del messaggio sono calcolati utilizzando le chiavi di autenticazione del messaggio (KIA_c e KIA_s) precedentemente derivate durante la fase di "Derive Symmetric Keys". Queste chiavi vengono utilizzate per generare i MAC associati ai messaggi.



24. **Protezione della Procedura di Handshake di TLS:**
    - Questo passo è estremamente importante per proteggere la procedura di handshake di TLS. Qualsiasi tentativo di alterare uno qualsiasi dei messaggi della procedura di handshake sarà rilevato attraverso il controllo dei MACs scambiati. In caso di qualsiasi manipolazione o errore, la connessione viene immediatamente interrotta, senza salvare alcuna chiave simmetrica o informazione di sessione.

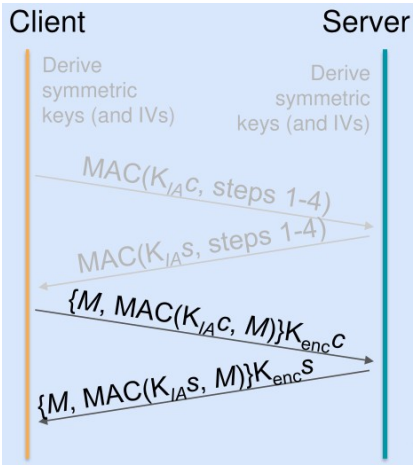
25. **Autenticazione del Server:**
    - Questo passo svolge anche un ruolo critico nell'autenticazione del server. Se il server fosse un impostore (un falso), non sarebbe in grado di derivare correttamente la chiave (KIA_s) utilizzata per calcolare il MAC. Pertanto, il tentativo di manipolare i MACs fallirebbe, poiché il server non possiede la chiave segreta necessaria per produrre un MAC valido.

TLS Handshake Step 6: Send Messages

26. **Invio Sicuro dei Messaggi (Application Data):**
    - Con le chiavi simmetriche e i codici di autenticazione del messaggio (MACs) condivisi, sia il client che il server possono inviare dati applicativi (messaggi) in modo sicuro.
    - L'ordine delle operazioni per garantire sicurezza è importante. In TLS 1.2, la pratica utilizzata è "MAC-then-encrypt", il che significa che il messaggio viene prima autenticato

tramite il calcolo del codice di autenticazione del messaggio (MAC) e successivamente cifrato per garantire la riservatezza.

27. **Operazione MAC-then-Encrypt:**
    - Prima di cifrare il messaggio, viene calcolato il codice di autenticazione del messaggio (MAC) utilizzando la chiave di autenticazione del messaggio (KIA). Questo MAC serve a garantire l'integrità del messaggio e a verificare che non sia stato alterato durante la trasmissione.
    - Dopo il calcolo del MAC, il messaggio viene cifrato utilizzando la chiave di cifratura appropriata (Kenc). Questo processo aggiuntivo di cifratura serve a proteggere la riservatezza del messaggio.



Considerazioni su Authenticate-then-Encrypt (A&E)
Sebbene TLS 1.2 utilizzi la pratica A&E, esistono dibattiti nella comunità crittografica sulla sequenza preferita. E&A è generalmente considerato un approccio più sicuro, ma TLS 1.2 ha mantenuto A&E per compatibilità con implementazioni esistenti.

TLS – Come un utente può essere sicuro di parlare con un server legittimo?

La sicurezza del protocollo TLS si basa su diverse misure per garantire che il client stia effettivamente comunicando con il server legittimo:

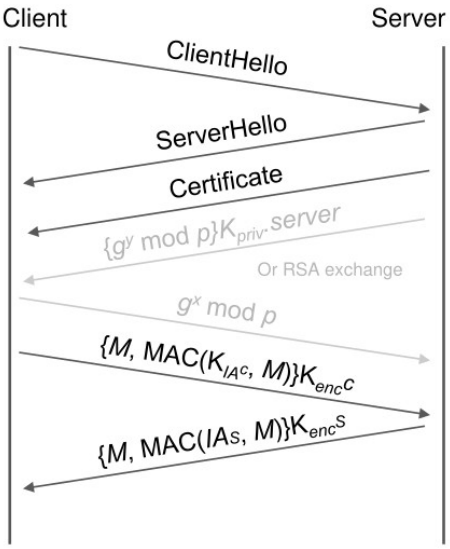
- **Certificato del Server:**
    - Il server invia al client il suo certificato X.509v3, che contiene la chiave pubblica del server e altre informazioni relative all'identità del server.
    - Il client può verificare la validità del certificato del server verificando la firma digitale del certificato, controllando la sua validità temporale, verificando che sia stato emesso da un'autorità di certificazione (CA) attendibile, e così via.
- **Prova del Possesso della Chiave Privata:**
    - Nel caso dell'utilizzo di RSA per lo scambio della chiave segreta, il server dimostra di possedere la chiave privata corrispondente alla chiave pubblica nel certificato. Il server riesce a fare ciò durante la decifrazione del segreto premaster (PS) inviato dal client, dimostrando di poter utilizzare con successo la sua chiave privata associata.
- **Prova del Possesso della Chiave Privata con DH:**
    - Se viene utilizzato lo scambio di chiavi Diffie-Hellman (DH), il server dimostra di possedere la chiave privata firmando la sua metà dello scambio DH e calcolando il corretto MAC durante la fase di "Exchange MACs" (passo 5) utilizzando la chiave segreta derivata.
- **Autenticazione del Server con RSA:**
    - Nel caso di uno scambio basato su RSA, l'attaccante non avrebbe la chiave privata del server (supponendo che non abbia compromesso il server), quindi non sarebbe in grado di dimostrare di possedere la chiave privata del server.

TLS: Secure messages

Come possiamo essere sicuri che un attaccante non possa leggere o manomettere i messaggi scambiati tra client e server?

La sicurezza delle comunicazioni tra client e server è garantita attraverso l'utilizzo di cifratura e codici di autenticazione del messaggio (MACs). Questi meccanismi operano sulla fase di trasferimento dei dati, assicurando che le informazioni siano confidenziali, autenticate e intatte durante il processo di scambio.

- **Cifratura dei Messaggi:**
  - La cifratura viene utilizzata per garantire la confidenzialità dei messaggi scambiati tra il client e il server. TLS utilizza diverse modalità di cifratura, come ad esempio i cifrari a blocchi (block ciphers) o i cifrari a flusso (stream ciphers).
  - I messaggi inviati tra client e server sono cifrati in modo che solo le parti legittime, dotate delle chiavi appropriate, possano decifrare e comprendere il contenuto effettivo del messaggio.

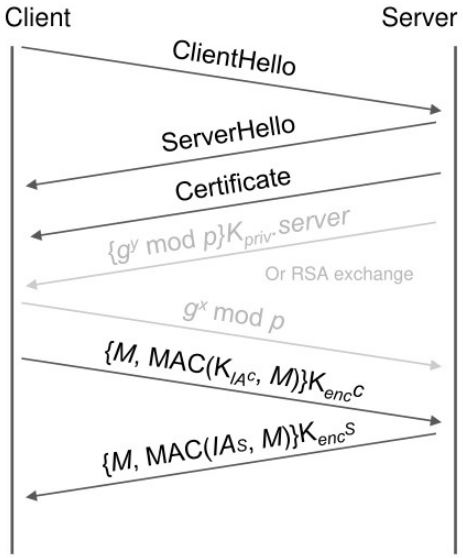


- **Codici di Autenticazione del Messaggio (MACs):**
  - I codici di autenticazione del messaggio (MACs) vengono utilizzati per garantire l'autenticità e l'integrità dei messaggi scambiati. Il MAC viene calcolato utilizzando una chiave condivisa tra il client e il server.
  - Nel contesto di TLS, il MAC viene calcolato sulla base di informazioni specifiche del messaggio, come il tipo di messaggio, la versione, la lunghezza e i dati effettivi del messaggio.
  - L'inclusione del MAC nel messaggio consente alla parte ricevente di verificare l'autenticità del mittente e garantire che il messaggio non sia stato alterato durante la trasmissione.
- **Applicabilità a diversi Tipi di Messaggi:**
  - Questi meccanismi di sicurezza sono applicati a diversi tipi di messaggi scambiati tra client e server, inclusi i dati applicativi effettivi e i messaggi specifici di TLS come quelli associati alle fasi di handshake, alert e cambio di specifica di cifratura.
- **L’attaccante non conosce il Premaster Secret**
  - Nel caso RSA, l'attaccante non può conoscere il premaster secret perché è stato cifrato con la chiave pubblica del server, che è teoricamente sicura.
  - Nel caso DH, l'attaccante non può conoscere il segreto DH condiviso, poiché DH è basato su operazioni matematiche difficili da invertire senza conoscere gli elementi segreti (x e y).
- **Derivazione delle Chiavi Simmetriche:**
  - Il premaster secret è utilizzato come input in una funzione di derivazione delle chiavi, che genera il master secret. Da questo master secret, le chiavi simmetriche (usate per la cifratura e il MAC) e i vettori di inizializzazione sono derivati in modo che solo le parti legittime conoscano questi valori.

TLS – Reply Attack

Come possiamo essere sicuri che un attaccante non ha reinserito nella comunicazione dati precedentemente trasmessi da una vecchia connessione TLS?

- **Utilizzo di Numeri Casuali Diversi:**
  - Durante ogni fase di handshake, sia il client che il server generano un numero casuale. Il client random (Rc) e il server random (Rs) sono unici per ogni sessione TLS. Questi numeri casuali vengono utilizzati nella derivazione delle chiavi simmetriche durante il processo di handshake.

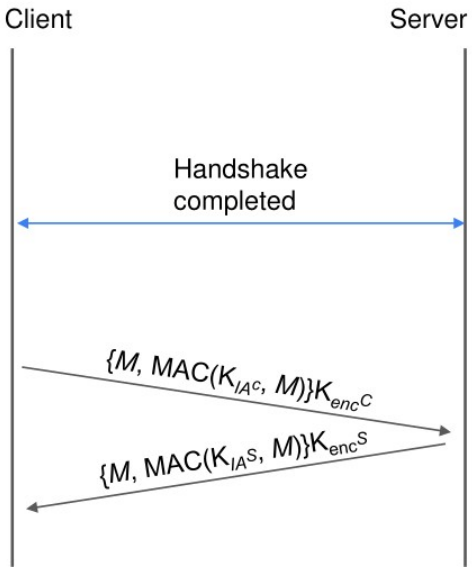


- **Derivazione delle Chiavi Differente per Ogni Connessione:**
  - Poiché le chiavi simmetriche sono derivate dal premaster secret, che include Rc e Rs, esse saranno diverse per ogni connessione TLS.
- **Sicurezza attraverso Chiavi Diverse:**
  - Poiché le chiavi simmetriche sono derivate utilizzando numeri casuali differenti, anche se un attaccante cattura e cerca di ripetere una sequenza cifrata da una sessione precedente, le chiavi utilizzate saranno diverse.
- **Protezione Aggiuntiva nella Derivazione delle Chiavi:**
  - La derivazione delle chiavi è influenzata non solo dai numeri casuali ma anche dal master secret, che è segreto condiviso tra client e server e unico per ogni sessione.

TLS: Replay/Filtering Attacks

Come possiamo essere sicuri che un attaccante non abbia reinserito nella comunicazione dati precedentemente trasmessi dalla connessione corrente TLS o filtrati parti di essi?

- **Numeri di Sequenza in TLS:**
  - Durante la fase di handshake e successivamente nella fase di trasferimento dati, ogni record TLS (un'unità di dati trasferita tra client e server) è associato a un numero di sequenza (seq_number).
  - Il seq_number è un valore univoco e incrementale, assegnato a ogni record. Ad esempio, il primo record avrà seq_number 1, il secondo 2, e così via.



- **Prevenzione tramite Sequenza Unica:**

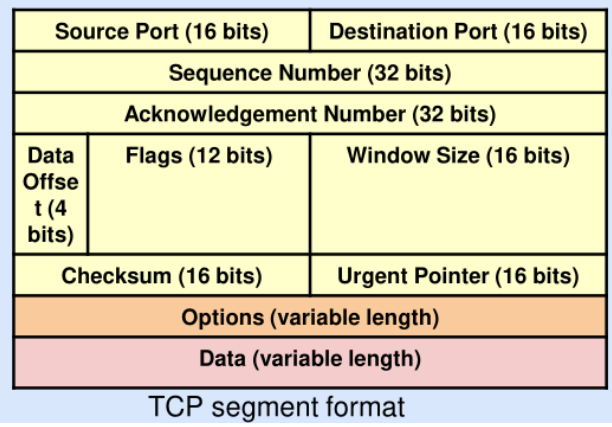
  Ogni record inviato durante la connessione TLS utilizza il proprio numero di sequenza. Se un attaccante tenta di ripetere un record, filtrare parti della comunicazione o riordinare i record, ciò comporterà l'uso di numeri di sequenza non validi o fuori sequenza.

- **Verifica durante la Verifica MAC:**

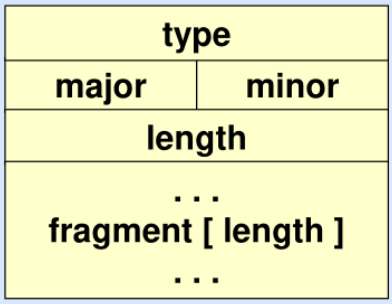
La sicurezza aggiuntiva viene garantita durante la verifica del codice di autenticazione del messaggio (MAC). Poiché il numero di sequenza è incorporato nei dati utilizzati per calcolare il MAC, qualsiasi manipolazione da parte di un attaccante influenzerà il numero di sequenza. Quando il server (o il destinatario) riceve un record, verifica il MAC e, durante questo processo, controlla anche che il numero di sequenza sia corretto.

In caso di errore durante la verifica del MAC, porta a un rifiuto del record manipolato e, di conseguenza, a una protezione contro gli attacchi di ripetizione o filtraggio.

TLS sequence numbers and TCP sequence numbers



TCP segment format



TLS record format

- Il TLS sequence number è cifrato ed è usato per la sicurezza
- Il TCP sequence number non è cifrato ed è usato per l’affidabilità in TCP

Forward Secrecy in TLS

Forward Secrecy è un concetto di sicurezza che affronta il rischio associato all'utilizzo di una chiave privata a lungo termine per entrambe le firme e la crittografia. Se un aggressore compromette la chiave privata a lungo termine in seguito, può decifrare tutto il traffico, incluso quello passato (perché all’inizio le informazioni passano tutte in chiaro).

Forward Secrecy mitiga questo rischio generando chiavi di sessione temporanee, in modo che anche se la chiave privata a lungo termine viene scoperta, non si possa accedere alle chiavi di sessione passate, garantendo la sicurezza delle comunicazioni precedenti.

Perfect forward secrecy

Perfect Forward Secrecy (PFS) è un concetto di sicurezza che garantisce che la compromissione della chiave privata asimmetrica di un server non influisca sulle comunicazioni passate.

In questo scenario la compromissione di una chiave privata compromette solo il traffico attuale (ed eventualmente quello futuro) ma non quello passato. Ciò implica che la chiave utilizzata per l'autenticazione deve essere diversa da quella utilizzata per la crittografia, ma non è sufficiente. La chiave utilizzata per la crittografia deve essere una chiave unica.

Tuttavia, con RSA nel protocollo TLS, se il premaster secret viene scambiato senza precauzioni aggiuntive, non si ottiene la Forward Secrecy. In questo caso, un avversario potrebbe registrare informazioni durante uno scambio crittografato e, in caso di compromissione della chiave privata del server, ottenere accesso alle chiavi di crittografia e autenticazione del MAC, mettendo a rischio la sicurezza delle comunicazioni passate.

Forward secrecy: with “ephemeral” mechanisms

La Perfect Forward Secrecy (PFS) viene implementata attraverso meccanismi "effimeri", come nell'utilizzo di Ephemeral Diffie-Hellman (DHE). Con DHE, una chiave temporanea DH viene generata dinamicamente per ogni sessione, e per garantire l'autenticità, l'esponente pubblico DH del server viene firmato utilizzando la sua chiave privata a lungo termine solo per la firma. Questo approccio assicura una PFS: se la chiave privata temporanea del server viene compromessa, l'attaccante può decifrare solo il traffico associato a quella specifica sessione. La compromissione della chiave privata a lungo termine del server rappresenta una minaccia per l'autenticazione, ma non compromette la confidenzialità del traffico. Esempi pratici di questo concetto includono l'utilizzo diffuso di Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) nei server TLS moderni.

TLS and virtual server: the problem

I server web utilizzano porte diverse per HTTP e HTTPS, comunemente 80 per HTTP e 443 per HTTPS. Quando si tratta di server virtuali, che possono avere nomi logici diversi ma con lo stesso indirizzo IP, sorgono sfide con l'attivazione di TLS, poiché quest'ultimo avviene prima del livello di applicazione. Nel contesto di HTTPS, la questione principale riguarda l'indicazione di quale server virtuale il client desidera connettersi. Mentre in HTTP/1.1 il client utilizza l'intestazione Host per specificare il server di destinazione, in HTTPS la situazione è più complessa poiché TLS si attiva prima di HTTP. Durante l'apertura del canale TCP, il server deve fornire un certificato, e la sfida è garantire che il nome nel certificato corrisponda al nome del server virtuale desiderato.

TLS and virtual server: the solution

Per gestire il problema dell'attivazione di TLS con server virtuali, esistono diverse soluzioni.

- Una opzione è utilizzare un certificato collettivo (wildcard) con un esempio come CN=*.myweb.it, ma questa approccio richiede la condivisione della chiave privata tra tutti i server e può comportare comportamenti diversi da parte dei browser.
- Un'altra possibilità è utilizzare un certificato del server con un elenco di nomi di server virtuali nel campo subjectAltName, tuttavia, richiede la riemissione del certificato per ogni modifica nella lista dei server.
- La soluzione preferita è l'utilizzo dell'estensione SNI (Server Name Indication) nel messaggio ClientHello, standardizzata da RFC-4366, che consente al client di specificare il nome del server a cui si intende connettere. Tuttavia, va notato che il supporto per SNI può essere limitato da parte di browser e server.

When is TLS effective?

TLS è efficace per garantire sicurezza end-to-end in una comunicazione, proteggendo:

- l'autenticazione
- l'autenticazione dei dati
- l'integrità
- la riservatezza durante il transito all'interno del canale di comunicazione TLS tra i due endpoint.

Questo significa che anche se ci fossero intermediari malintenzionati tra il client e il server, TLS fornirebbe comunque un canale di comunicazione sicuro. Ad esempio, un attaccante in una rete locale potrebbe tentare un attacco Man-in-the-Middle (MITM) con ARP poisoning, ma non sarebbe in grado di leggere i messaggi TLS né di modificarli o cancellarli senza che il MAC risulti corretto alla destinazione. Allo stesso modo, un MITM nel router o sul backbone che cerca di iniettare pacchetti TCP vedrebbe i pacchetti respinti perché il MAC non sarebbe corretto.

Tuttavia, è importante notare che la sicurezza end-to-end di TLS non è utile se uno degli endpoint è malintenzionato. Ad esempio, se un client comunica con un server falso che presenta un certificato "valido", come uno emesso da una CA compromessa, oppure se una delle parti (client/server) è stata infettata da malware. In questi casi, TLS potrebbe non essere in grado di prevenire attacchi provenienti da uno degli endpoint compromessi.