Analisi Dettagliata del Protocollo Transport Layer Security (TLS)

Sintesi e approfondimento

1 Introduzione a Transport Layer Security (TLS)

Il Transport Layer Security (TLS), e il suo predecessore Secure Sockets Layer (SSL), è uno dei protocolli di sicurezza più noti e diffusi, fondamentale per garantire la confidenzialità e l'integrità delle comunicazioni su reti insicure come Internet. Lo scopo di questa analisi è duplice:

1. **Dissezionare il protocollo:** Analizzare in profondità i dettagli tecnici del TLS per comprendere come i meccanismi crittografici vengono combinati per raggiungere gli obiettivi di sicurezza. Come spesso accade in questo campo, "il diavolo è nei dettagli".
2. **Comprendere il design di un protocollo longevo:** Studiare le scelte di progettazione, sia quelle vincenti che quelle rivelatesi problematiche, che hanno permesso al protocollo di evolvere e adattarsi a nuove minacce per oltre due decenni.

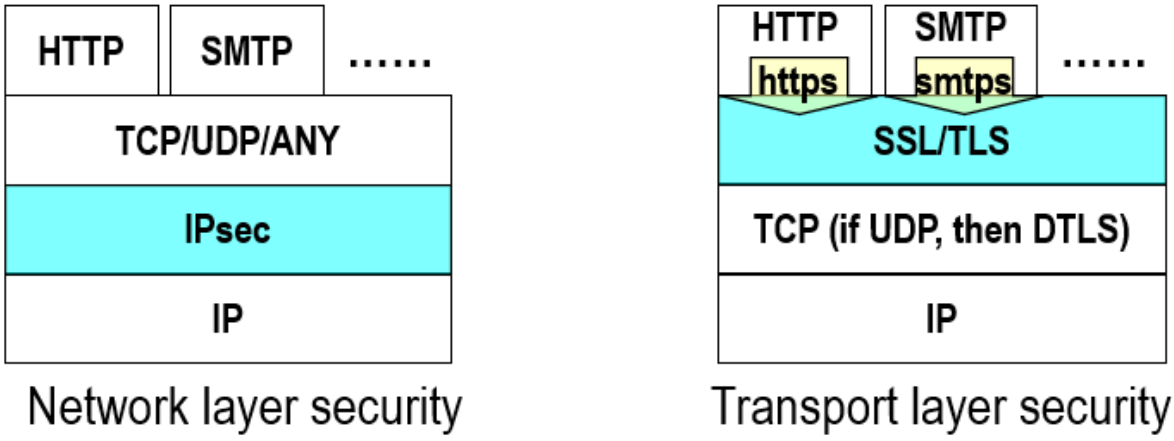
1.1 Storia ed Evoluzione di SSL/TLS

La storia del protocollo è un'interessante cronistoria di vulnerabilità scoperte e di continui miglioramenti:

- **1994, SSL v1:** Sviluppato da Netscape, non fu mai rilasciato pubblicamente a causa di gravi falle di sicurezza scoperte internamente.
- **1995, SSL v2:** Integrato nel browser Netscape 1.1, divenne rapidamente popolare ma si rivelò anch'esso gravemente vulnerabile a diversi attacchi (es. messaggi di handshake non protetti, uso della stessa chiave per MAC e cifratura).
- **1996, SSL v3:** Una riprogettazione quasi completa da parte di Netscape che risolse molte delle vulnerabilità della versione 2 e che ha costituito la base per il futuro TLS.
- **1999, TLS v1.0 (RFC 2246):** Il primo standard sviluppato dall'IETF (Internet Engineering Task Force), di fatto una versione leggermente migliorata e standardizzata di SSLv3, tanto da essere talvolta definito come SSLv3.1.
- **2006, TLS v1.1 (RFC 4346):** Introdusse alcune migliorie per mitigare attacchi noti, come quelli legati all'uso della modalità di cifratura CBC. Nello stesso anno fu standardizzato anche **DTLS (Datagram TLS)**, una variante per il trasporto su protocolli inaffidabili come UDP.
- **2008, TLS v1.2 (RFC 5246):** Una versione molto importante che ha rimosso gli algoritmi crittografici più deboli (come MD5 e SHA-1) e ha introdotto il supporto per suite crittografiche più moderne e sicure, come quelle basate su AES-GCM (Authenticated Encryption with Associated Data) e l'uso di SHA-256 come default.
- **2018, TLS v1.3 (RFC 8446):** Rappresenta una revisione radicale del protocollo. Ha rimosso algoritmi e meccanismi obsoleti, ridotto la latenza dell'handshake (passando da due round-trip a uno solo) e migliorato notevolmente la sicurezza, ad esempio cifrando una porzione maggiore dell'handshake e supportando esclusivamente cifrari AEAD.

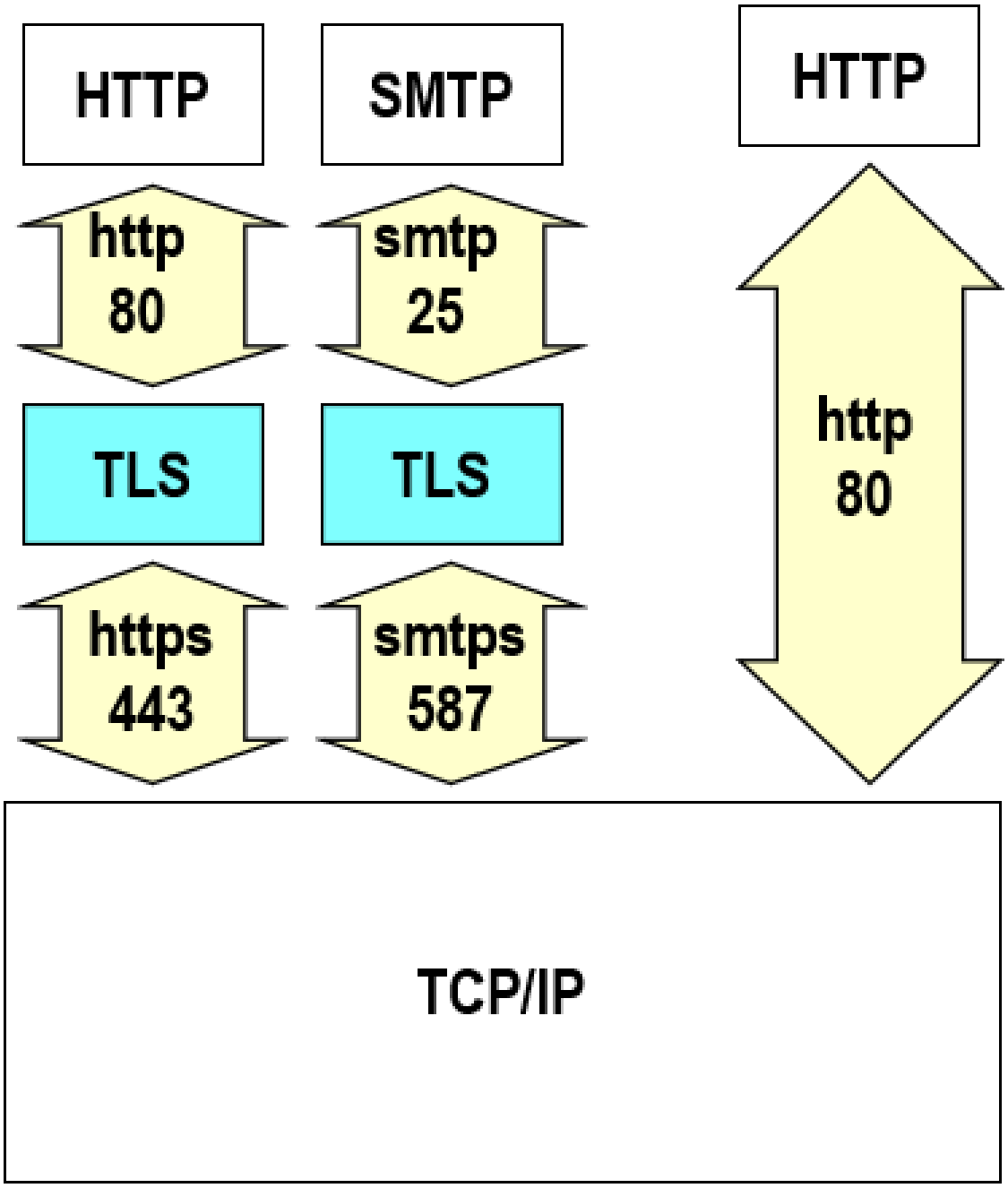
2 Posizionamento e Integrazione Applicativa

2.1 Collocazione nello Stack ISO/OSI



Il TLS si posiziona tra il livello di trasporto (tipicamente TCP) e il livello applicativo (HTTP, SMTP, etc.). È importante sottolineare che **non è un'estensione di sicurezza del TCP**; piuttosto, è un livello a sé stante che offre un canale di comunicazione sicuro ai protocolli di livello superiore. Sebbene sia nato per il trasporto su TCP, il suo design è sufficientemente generico da poter essere utilizzato su qualsiasi protocollo punto-punto affidabile. Esistono infatti implementazioni come EAP-TLS che lo utilizzano al livello 2.

2.2 Integrazione con le Applicazioni



Storicamente, si sono affermati due approcci principali per far utilizzare TLS a un'applicazione:

1. **Porte Dedicate:** Si assegna un numero di porta TCP distinto per la versione sicura del protocollo. Ad esempio, HTTP utilizza la porta 80, mentre HTTPS (HTTP su TLS) utilizza la porta 443. Questo approccio, sebbene semplice da implementare e diventato uno standard de-facto, è stato deprecato dall'IETF perché porta a uno spreco di porte riservate (es. SMTPS porta 465/587, IMAPS porta 993, etc.).

2. **Meccanismo di Upgrade:** Il protocollo applicativo inizia la comunicazione in chiaro sulla sua porta standard e, successivamente, negozia il passaggio a una comunicazione cifrata tramite un comando specifico. Un esempio è il comando 'Upgrade: TLS/1.0' introdotto in HTTP/1.1 (RFC 2817). Questo approccio è più flessibile e non richiede porte aggiuntive.

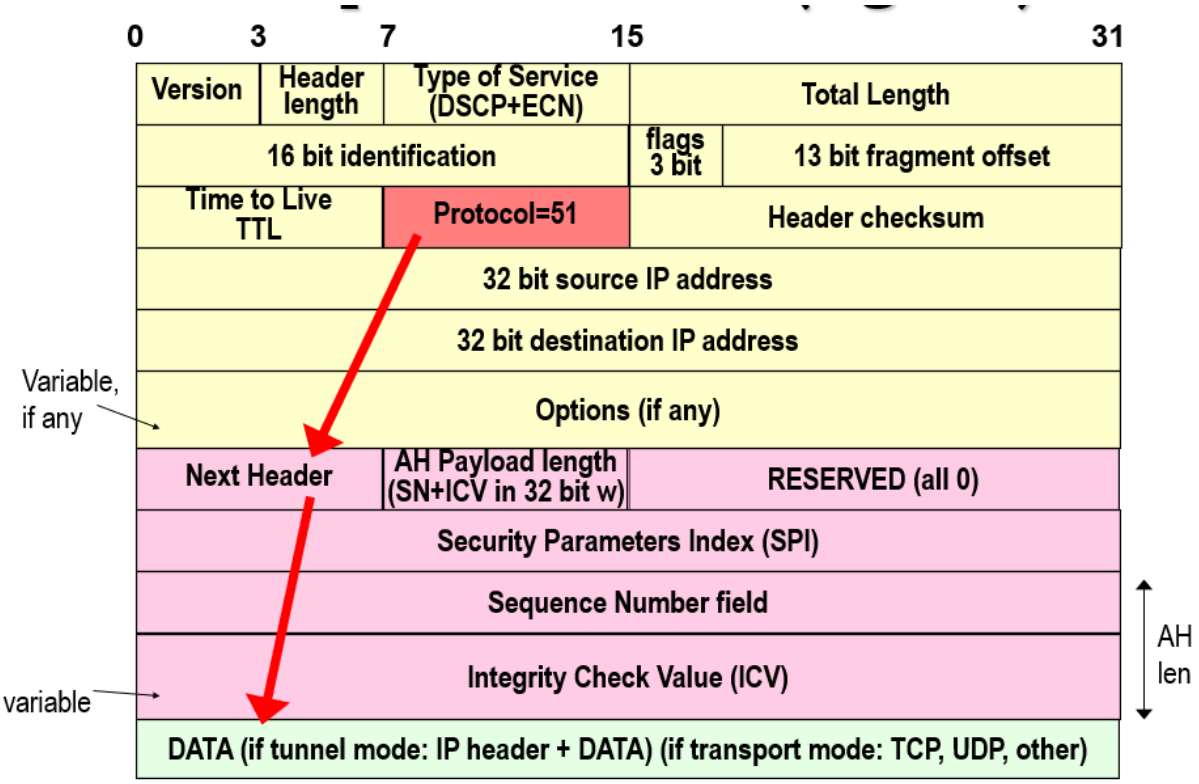
3 Obiettivi e Architettura del Protocollo TLS

Il funzionamento del TLS si articola in due fasi principali con obiettivi distinti:

1. **Fase di Handshake:** È la fase iniziale di negoziazione, durante la quale client e server:
   - **Concordano gli algoritmi** crittografici da utilizzare (suite crittografica).
   - **Condividono le chiavi segrete** che verranno usate per la sessione.
   - **Eseguono l'autenticazione** (tipicamente del server, ma opzionalmente anche del client) tramite certificati digitali.
2. **Fase di Trasferimento Dati:** Una volta stabilita la sessione, il protocollo garantisce:
   - **Privacy della comunicazione** tramite cifratura simmetrica.
   - **Integrità dei dati** tramite un Message Authentication Code (MAC) basato su hash con chiave (HMAC).

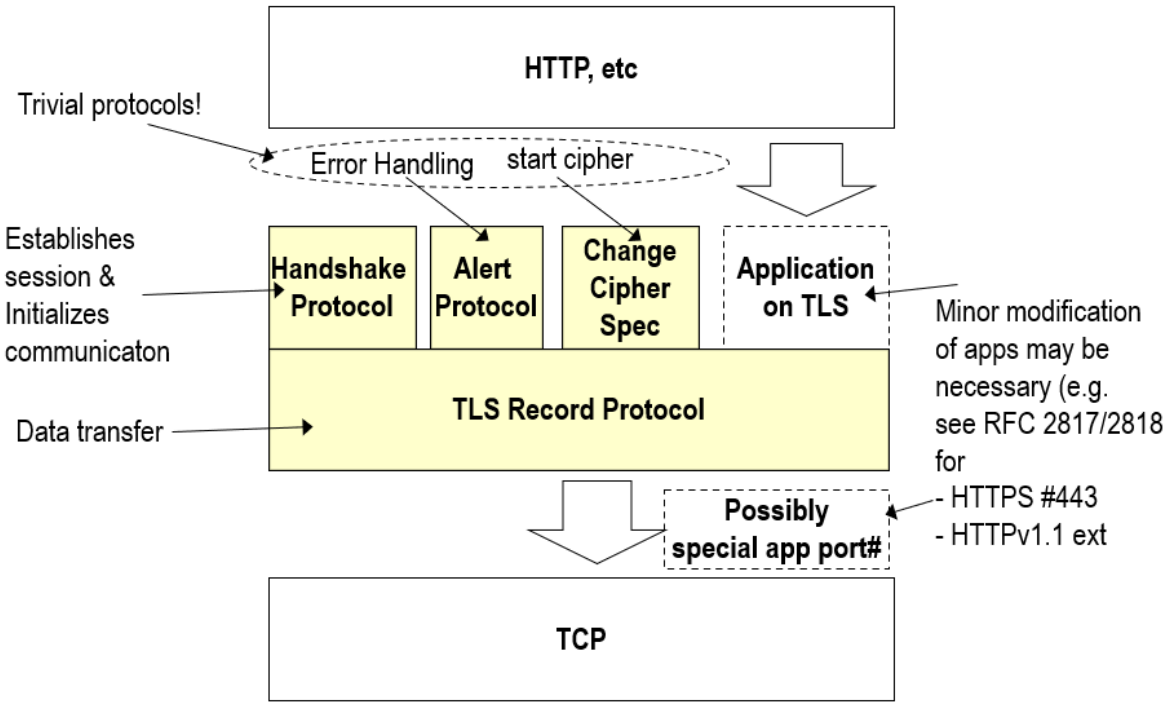
3.1 Architettura Interna

A differenza di altri protocolli come IPsec, che separano nettamente il protocollo di gestione delle chiavi (IKE) da quello di protezione dei dati (ESP/AH), TLS integra queste funzionalità in un'unica suite di protocolli.



L'architettura interna di TLS è stratificata:

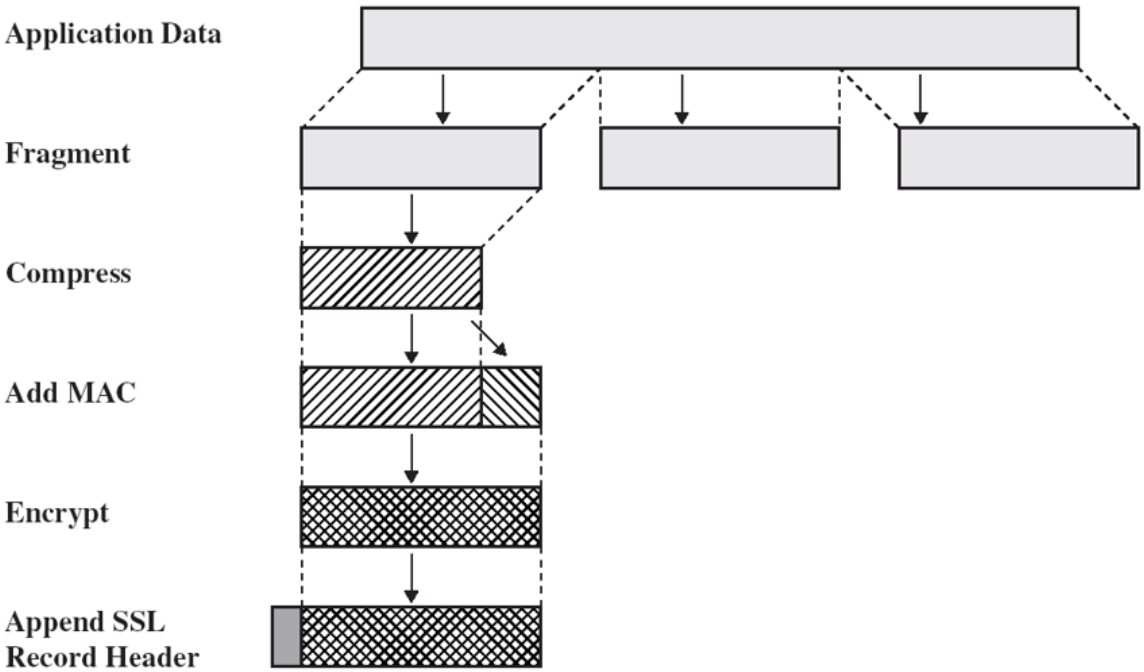
- **TLS Record Protocol:** È il livello più basso, responsabile di frammentare, comprimere (opzionale e oggi deprecato), proteggere (con MAC e cifratura) e incapsulare i dati provenienti dai protocolli superiori.
- **Handshake Protocol:** Gestisce la complessa negoziazione della sessione sicura.
- **Alert Protocol:** Utilizzato per segnalare errori o condizioni anomale (es. certificato non valido, MAC errato).
- **Change Cipher Spec Protocol:** Un protocollo molto semplice il cui unico scopo è segnalare il momento in cui i parametri di sicurezza negoziati diventano attivi.



4 Il TLS Record Protocol (fino a TLS v1.2)

Il Record Protocol è il cavallo di battaglia del TLS per il trasferimento dei dati. Il suo funzionamento (fino a TLS v1.2) prevede una sequenza di passaggi applicati ai dati provenienti dal livello applicativo:

1. **Frammentazione:** I dati vengono suddivisi in blocchi di dimensione massima di 2^{14} byte (16 KB).
2. **Compressione:** Originariamente prevista, questa fase è stata di fatto abbandonata e impostata su "null" a causa della scoperta di vulnerabilità come l'attacco **CRIME** (2012), che sfruttava la variazione della lunghezza dei dati compressi per dedurre informazioni sensibili (es. cookie di sessione).
3. **Aggiunta del MAC:** Viene calcolato un Message Authentication Code (tipicamente un HMAC) sui dati (compressi) per garantirne l'integrità. Nel calcolo del MAC viene inclusa anche l'intestazione del record e un numero di sequenza implicito per proteggersi da attacchi di replay e riordino.
4. **Cifratura:** Il blocco risultante (dati + MAC) viene cifrato utilizzando l'algoritmo di cifratura simmetrica (es. AES) e le chiavi negoziate durante l'handshake.
5. **Aggiunta dell'Header:** Infine, viene anteposto un header di 5 byte contenente il tipo di contenuto (es. dati applicativi, handshake), la versione del protocollo e la lunghezza del payload cifrato.

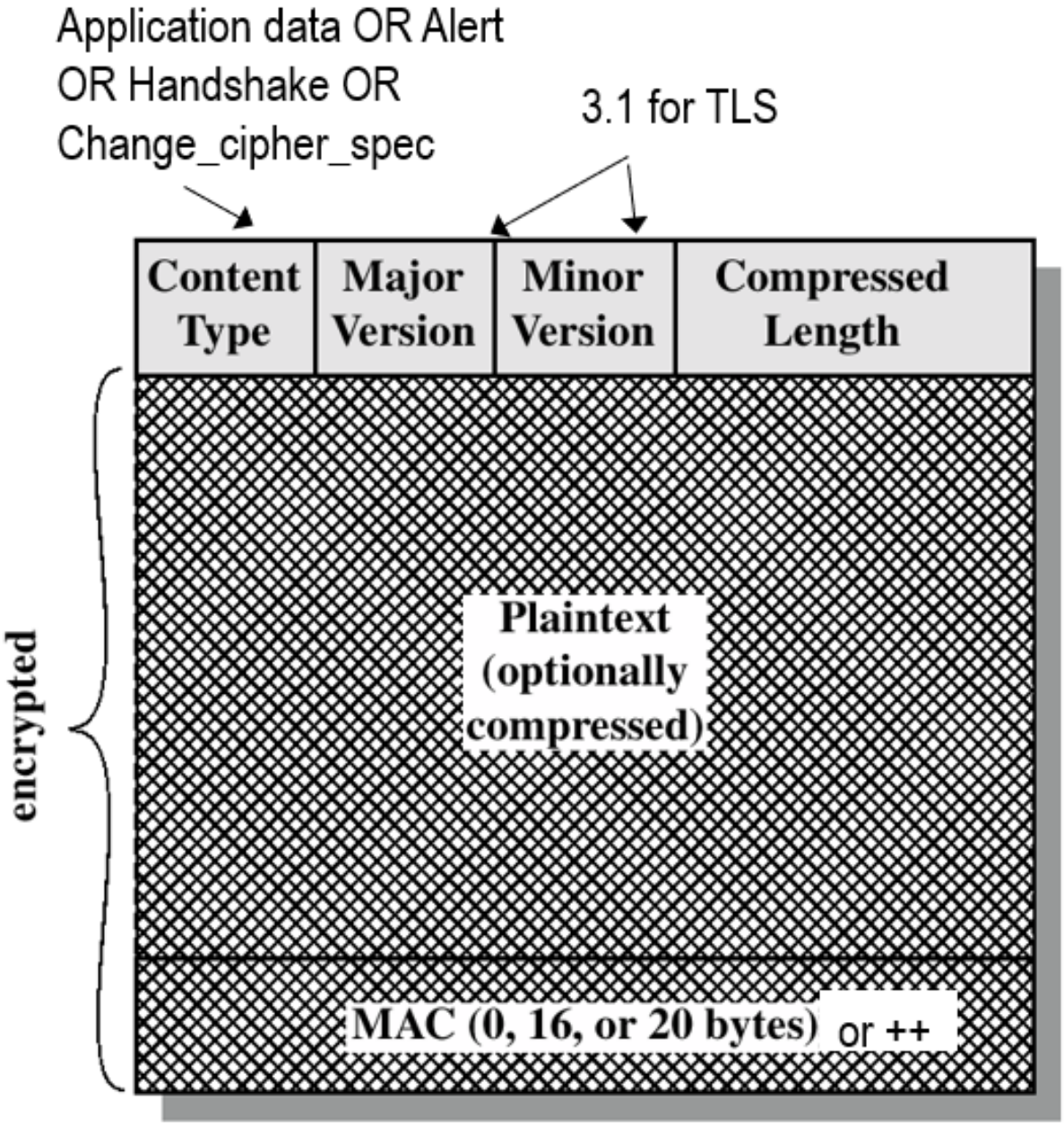


4.1 Dettagli e Implicazioni di Sicurezza del Record Protocol

Errori Comuni nei Diagrammi La slide 11 della presentazione originale avverte della presenza di due errori significativi nel diagramma del Record Protocol. Questi errori, comuni in molte rappresentazioni semplificate, sono:

1. **Ordine MAC-then-Encrypt:** Il diagramma mostra che prima si calcola il MAC e poi si cifra il risultato. Sebbene questo approccio sia stato utilizzato da TLS fino alla versione 1.2, la crittografia moderna considera più sicuro l'approccio inverso (**Encrypt-then-MAC**). L'approccio MAC-then-Encrypt si è rivelato vulnerabile ad attacchi di tipo "padding oracle" in alcune configurazioni. TLS 1.3 ha risolto questo problema adottando esclusivamente cifrari AEAD, che integrano autenticazione e cifratura in un unico passaggio sicuro.

2. **Dati inclusi nel MAC:** Il diagramma suggerisce che il MAC sia calcolato solo sui dati. In realtà, per prevenire attacchi più sofisticati, l'HMAC in TLS viene calcolato su una concatenazione di: un numero di sequenza, l'header del record (tipo, versione, lunghezza) e i dati stessi. Questo lega il MAC non solo al contenuto, ma anche al suo contesto all'interno della comunicazione.



Protezione dagli Attacchi di Replay Un attacco di replay consiste nel catturare un messaggio valido e reinviarlo in un momento successivo per ottenere un effetto indesiderato (es. ripetere un'operazione di pagamento). Il TLS si protegge da questo tipo di attacco includendo un **numero di sequenza** implicito nel calcolo del MAC.

- Client e server mantengono due contatori separati, uno per i messaggi inviati e uno per quelli ricevuti.
- Questi contatori vengono inizializzati a 0 all'inizio della sessione e incrementati per ogni record inviato.
- Il numero di sequenza non viene trasmesso in chiaro nell'header del record, ma viene utilizzato come input per il calcolo dell'HMAC.
- Il ricevente, che mantiene il proprio contatore sincronizzato, calcola l'HMAC sul messaggio ricevuto usando il numero di sequenza che si aspetta. Se il MAC calcolato non corrisponde a quello ricevuto, il record viene scartato.

Questo meccanismo è efficace perché si basa su un trasporto affidabile come il TCP, che garantisce la consegna ordinata e senza "buchi" dei pacchetti. Nel caso di DTLS (su UDP), il numero di sequenza deve essere invece trasmesso esplicitamente nell'header.