Secure Socket Layer

Sicurezza del livello Trasporto

Rendere sicure le connessioni TCP con SSL

- Ad una applicazione, le tecniche di criptografia:
  - forniscono la riservatezza delle comunicazioni,
  - garantiscono l'integrità dei dati
  - richiedono l'autenticazione del client e del server.
- La libreria **Secure Sockets Layer (SSL)** introduce la criptografia a livello trasporto per soddisfare gli stessi requisiti di sicurezza anche al protocollo TCP.
- É stata standardizzata anche una versione leggermente modificata di SSL, chiamata **Transport Layer Security (TLS).**

SSL

- SSL si è diffuso rapidamente. Tutti i browser e molti server WEB incorporano la libreria SSL.
- Tutti i siti di commercio elettronico affidano i loro server alla libreria SSL (Amazon, eBay, Yahoo!, MSN, …). Le transazioni si aggirano sulle decine di miliardi di dollari spesi ogni anno.
- Chiunque abbia fatto acquisti con la propria carta di credito si sarà accorto che la comunicazione tra il browser e il server web avviene su una connessione SSL. (nella barra dell'indirizzo di un browser che comunica su una connessione SSL il protocollo è https anziché http.)

Transazioni on line

Esempio

- Bob accede al sito Web di Alice, alla ricerca di un prodotto da acquistare.
- La pagina del sito di Alice propone un form in cui Bob deve introdurre il nome e la quantità del prodotto desiderato, il suo indirizzo e il suo numero di carta di credito. Dopo aver completato tutti i campi del form, Bob clicca sul pulsante Submit, e si aspetta di ricevere, a mezzo corriere, ciò che ha acquistato. Si aspetta anche di trovare l'addebito dell'ordine sul suo conto.

Possibili frodi in assenza di sicurezza

La transazione on line di Bob è soggetta ai seguenti rischi:

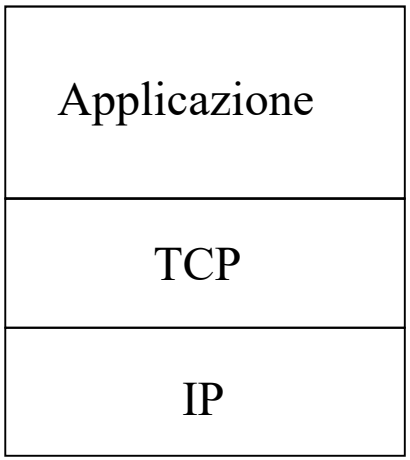
- Se manca la riservatezza (criptografia dei dati), un intruso potrebbe intercettare l'ordine di Bob e venire a conoscenza del numero di carta di credito di Bob. Con questa informazione, l'intruso potrà fare acquisti a spese di Bob.
- Se manca il controllo sull'integrità dei dati, un intruso potrebbe modificare l'ordine di Bob, ad esempio, facendogli acquistare qualcosa di indesiderato.
- Se il server non viene autenticato, un attaccante potrebbe riprodurre fedelmente il sito web di Alice, spacciandolo per quello vero. L'attaccante riceve l'ordine e l'accredito, quindi fa perdere le proprie tracce. Inoltre ha rubato le informazioni personali di Bob.

La libreria SSL risolve questi problemi.

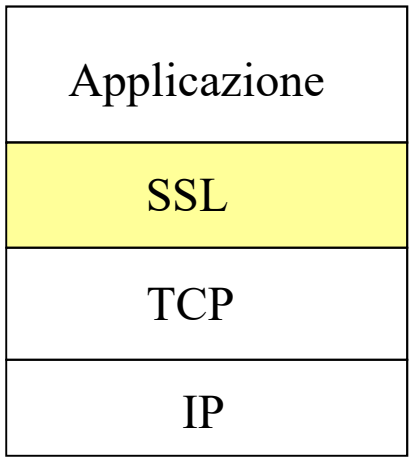
SSL: Secure Sockets Layer

- Scopi della libreria SSL:
  - Realizzare applicazioni per transazioni con siti Web specializzati in e-commerce
  - Criptare i dati (come i numeri di carta di credito)
  - Autenticare il server Web
  - Opzionalmente autenticare il client
  - Minimizzare le operazioni accessorie richieste per proteggere acquirente e venditore
- La libreria SSL è disponibile a tutte le applicazioni che si appoggiano sul TCP
  - La libreria usa un interfaccia per un socket sicuro
- Soddisfa i requisiti fondamentali
  - *Riservatezza*
  - *Integrità*
  - *Autenticazione*

SSL e il modello TCP/IP



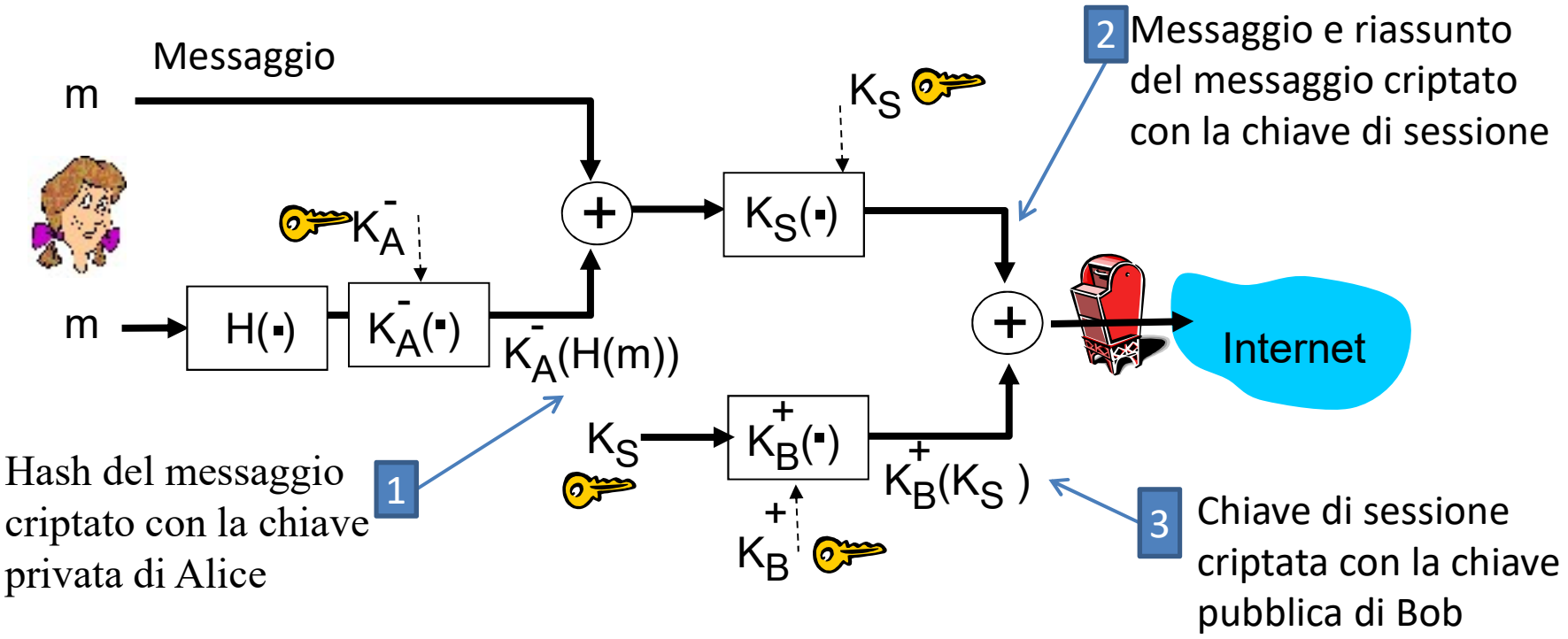
Modello base



Modello con SSL

- La libreria SSL fornisce l'interfaccia di programmazione alle applicazioni (API), in genere al protocollo http, ma siccome risiede al livello TCP, qualunque applicazione può utilizzare i servizi di sicurezza offerti da SSL.
- Sono disponibili librerie e classi SSL in C e Java

Misure di sicurezza contro i rischi:



Sono insufficienti perché:

- Si desidera interagire con il server (mantenere lo stato della sessione)
- La chiave condivisa, deve essere valida per tutta la connessione
- Il protocollo deve prevedere lo scambio dei certificati: fase di handshake

SSL: canale sicuro

Si suppone che il server possegga due chiavi e un certificato. Il certificato attesta che la sua identità è stata verificata e indica qual è la sua chiave pubblica.

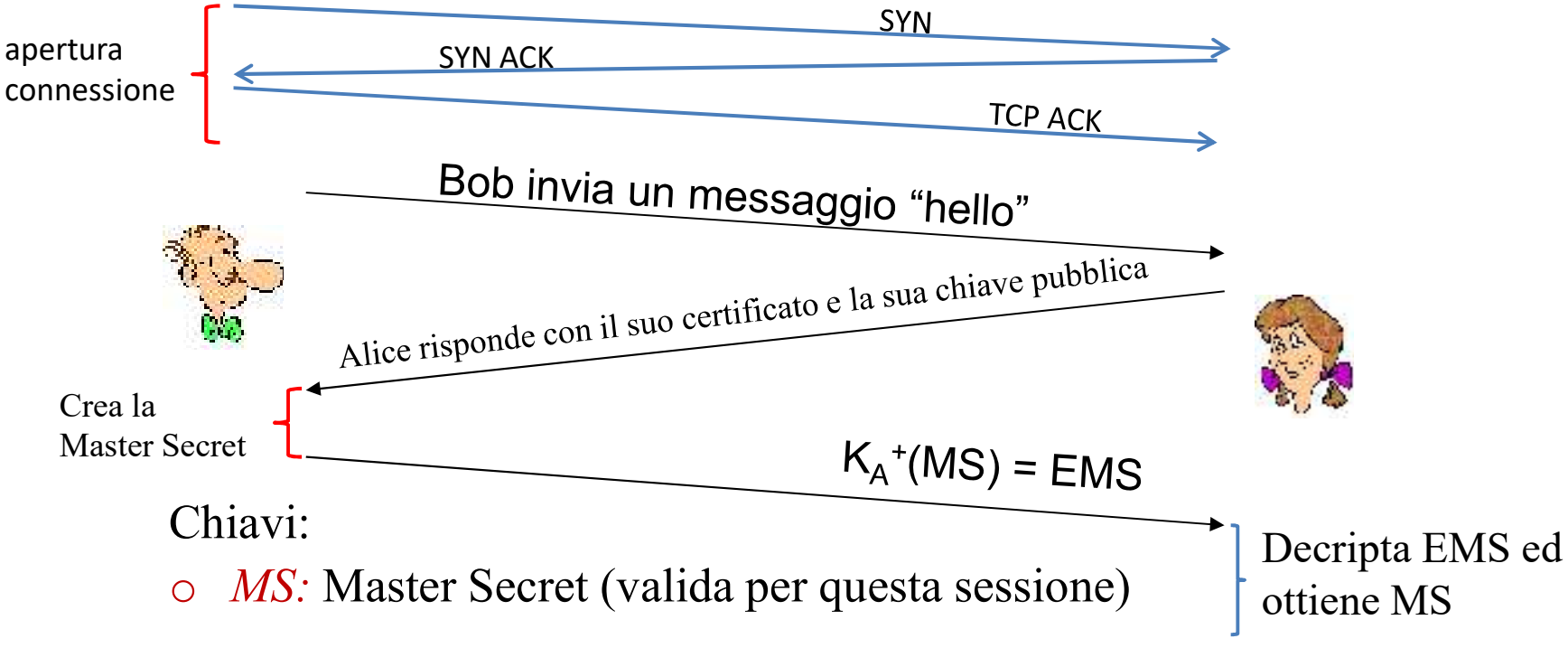
SSL è composto dalle seguenti fasi:

- ***handshake***: Alice e Bob usano i loro certificati, e le proprie chiavi private per autenticarsi e scambiare una chiave segreta condivisa
- ***derivazione chiavi***: Alice e Bob usano la chiave segreta per derivare un insieme di chiavi
- ***trasferimento dati***: i dati da trasferire vengono suddivisi in record
- ***chiusura connessione***: un messaggio speciale chiude la connessione

Handshake semplificato

Dopo aver stabilito la connessione, durante la fase di handshake, Bob deve:

a) Stabilire una connessione TCP con Alice,
b) Verificare che Alice è effettivamente *Alice,*
c) Inviare ad Alice una chiave master secret, che servirà ad Alice e Bob per generare tutte le chiavi simmetriche necessarie per quella sessione.



Chiavi:

- *MS:* Master Secret (valida per questa sessione)
- *EMS:* Encrypted Master Secret

Derivazione delle chiavi

- Si ritiene sbagliato usare la stessa chiave di criptografia per più operazioni diverse
  - Ad esempio si usa una chiave per calcolare il Message Authentication Code (MAC) e una per criptare i dati (na coppia di chiavi per ciascuna direzione)
- Sono richieste quattro chiavi, due per criptare i dati e due per calcolare o verificare il MAC (integrità dei dati):
  - K_c = chiave per criptare i dati inviati dal client al server
  - M_c = chiave per calcolare il MAC inviato dal client al server
  - K_s = chiave per criptare i dati inviati dal server al client
  - M_s = chiave per calcolare il MAC inviato dal server al client
- Le chiavi vengono derivate dalla MS con la funzione “key derivation” (KDF)
  - Per creare le chiavi, opera combinando la master secret con alcuni dati casuali

Record di dati

- Dopo la derivazione delle chiavi, Client e Server condividono le stesse 4 chiavi di sessione (K_s, M_s, K_c, M_c) e possono iniziare a scambiare dati in modo riservato sulla connessione TCP.
- Si potrebbe pensare che SSL cripti i dati ricevuti dall'applicazione, nel momento stesso in cui li riceve, e li passi già criptati al TCP. Ma nasce il problema di calcolare il MAC. Non si può attendere la fine della sessione TCP per verificare l'integrità di tutti i dati trasmessi durante l'intera sessione.
- SSL:
  - scompone il flusso dei byte in **record**,
  - aggiunge un MAC ad ogni record applicando una funzione hash ai dati che compongono il record concatenato con la chiave M_c. Ottiene MAC = H(record, M_c)
  - cripta il record + MAC, usando la sua chiave di sessione K_c. Questo blocco criptato K_c(MAC) viene consegnato al TCP per inviarlo su Internet.
- Problema: i record sono di lunghezza variabile quindi, in ogni record, il ricevitore deve poter distinguere il MAC dai dati.

Attacco man in the middle

- Un intruso in possesso degli strumenti per inserire, cancellare e sostituire segmenti nel flusso TCP scambiati tra due processi A e B, potrebbe, ad esempio, intercettare due segmenti, invertire l'ordine, modificare i numeri di sequenza TCP (che non sono criptati) e consegnare i due segmenti così scambiati al destinatario.
- Assumendo che ogni segmento TCP incapsula solo un record, il destinatario dovrebbe elaborare i segmenti nel modo seguente.

1) Il TCP in esecuzone sul destinatario non trova errori e passa i due record a SSL.
2) SSL nel destinatario decripta i due record.
3) SSL nel destinatario usa il MAC in ciascun record per verificare l'integrità dei dati dei due record.
4) SSL passa il contenuto decriptato dei due record al livello applicazione; ma l'applicazione destinataria riceve dati errati in conseguenza dell'ordine invertito dei due record.

Numeri di sequenza

- *problema:* l'intruso intercetta i record e può ripeterli o rimescolarli, prima di ritrasmetterli.
- *soluzione:* il mittente usa un contatore per numerare i record ed inserisce i numeri di sequenza all'interno del MAC:
  - MAC = HASH(chiave MAC, numero di sequenza, dati)
  - nota: il campo numero di sequenza è criptato durante il calcolo della funzione hash

- *problema:* l'intruso intercetta i record, li registra e, in un momento successivo, li ripete
- *soluzione:* usare il nonce

Informazioni di controllo

- *problema:* attacco di troncamento:
  - Il campo tipo indica se il record è un messaggio di handshake o un messaggio dati
  - L'attaccante falsifica il segmento di chiusura della connessione TCP
  - Uno o entrambi i sistemi pensano che mancano dei dati.
- *soluzione:* usare un tipo record riservato a specificare la chiusura della connessione
  - tipe 0 per i dati; tipo 1 per la chiusura
- MAC = HASH(chiave MAC, tipo, numero di sequenza, dati)

SSL: riepilogo

bob.com

hello

certificato, nonce

$K_B^+(MS) = EMS$

tipo 0, seq 1, dati

tipo 0, seq 2, dati

tipo 0, seq 1, dati

tipo 0, seq 3, dati

tipo 1, seq 4, chiusura

type 1, seq 2, chiusura

criptati

SSL non è ancora completo

- Bisogna specificare la lunghezza dei campi
- Quale algoritmo di criptografia usare
- Il client ed il server devono accordarsi sul livello di sicurezza. SSL non obbliga il client ed il server ad usare uno specifico:
  - algoritmo a chiave simmetrica,
  - algoritmo a chiave pubblica,
  - specifica funzione hash.
- SSL consente al client ed al server di accordarsi sugli algoritmi di criptografia all'inizio della sessione, durante la fase di handshake. Inoltre, durante la fase di handshake, il client ed il server si scambiano un nonce l'uno con l'altro, che serve per creare le chiavi di sessione (EB, MB, EA, e MA).

SSL: cifrari

- Cifrari
  - Algoritmo a chiave pubblica
  - Algoritmo di criptografia a chiave simmetrica
  - Funzione hash
- SSL ammette un'ampia scelta di cifrari
- negoziazione: client e server si accordano sui cifrari
  - Il client propone gli algoritmi di cui dispone
  - Il server ne sceglie uno

Cifrari simmetrici di SSL

- DES – Data Encryption Standard: cifrario a blocchi
- 3DES – Triple strength: cifrario a blocchi
- RC2 – Rivest Cipher 2: cifrario a blocchi
- RC4 – Rivest Cipher 4: cifrario a stream

Criptografia a chiave pubblica di SSL

- RSA

SSL reale: handshake (1)

Scopo dell'handshake

1. Autenticazione del server
2. Negoziazione: client e server si accordano sugli algoritmi di criptografia da usare
3. Generazione delle chiavi
4. Autenticazione del client (opzionale)

SSL reale: passi della fase di handshake

1) Il client invia
   a) l'elenco degli algoritmi di cui dispone,
   b) ed un nonce
2) Il server sceglie gli algoritmi dall'elenco; invia al client:
   a) Le scelte,
   b) Il certificato
   c) Il nonce del server
3) Il client
   a) verifica il certificato,
   b) estrae la chiave pubblica del server,
   c) genera la chiave pre_master_secret, la cripta con la chiave pubblica del server, e la invia al server

SSL reale: passi della fase di handshake

4) Il client ed il server, indipendentemente, usando la stessa funzione di derivazione delle chiavi:
   a) Calcolano la Master Secret dalla pre_master_secret e dai nonce.
   b) La MS viene scomposta per generare le due chiavi di criptografia e le due chiavi per il calcolo del MAC.
   c) Inoltre, quando viene scelto un cifrario a chiave simmetrica che impiega il metodo a blocchi (come 3DES o AES), dalla Master Secret vengono anche ottenuti due vettori di inizializzazione (uno per il client e uno per il server). In tal modo, tutti i messaggi scambiati tra client e server sono criptati e autenticati (con il MAC).
5) Il client invia il MAC di tutti i messaggi di handshake
6) Il server invia il MAC di tutti i messaggi di handshake

SSL reale: passi della fase di handshake

le ultime 2 operazioni proteggono l'handshake dalla falsificazione. Il rischio è che

- Il client propone, in chiaro, un elenco di algoritmi, alcuni resistenti, alcuni deboli
- Poichè non sono state ancora derivate le chiavi, un intruso potrebbe eliminare alcuni algoritmi dall'elenco, per forzare la scelta di un algoritmo debole.

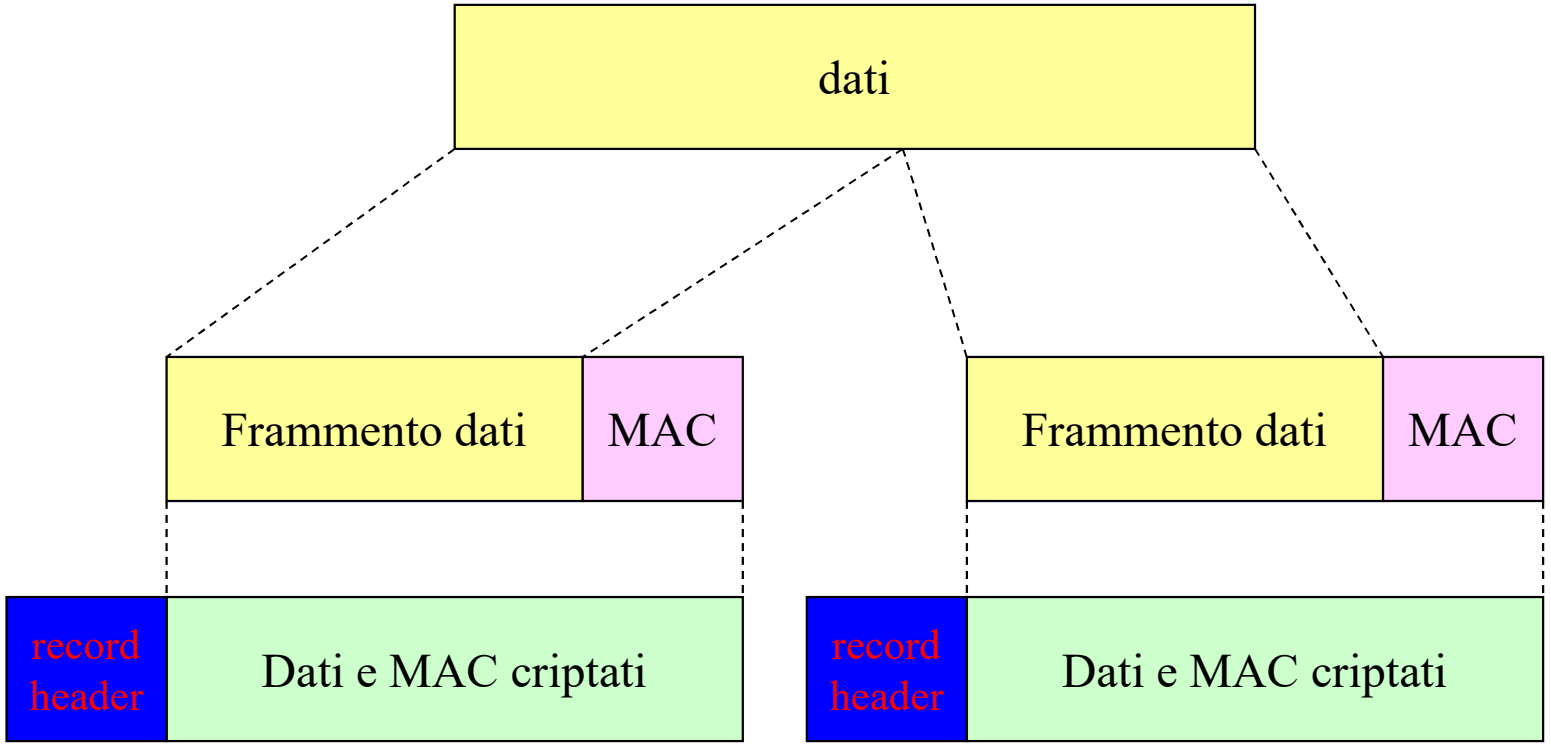
Per proteggere dall'attacco di falsificazione:

- il client invia il MAC ottenuto applicando la funzione hash alla concatenazione di tutti i messaggi di handshake inviati e ricevuti. Il server confronta questo MAC con il MAC di tutti i messaggi di handshake ricevuti e inviati. Se non c'è corrispondenza, il server termina la connessione. Allo stesso modo, il server invia il MAC di tutti i messaggi di handshake che ha ricevuto, permettendo al client di acertarsi che siano originali.
- Questi due messaggi contenenti il MAC sono criptati

SSL reale: passi della fase di handshake

- Ci si potrebbe chiedere per quale motivo si usano due nonce casuali, potrebbero bastare i numeri di sequenza.
- I numeri di sequenza non proteggono dall'attacco di ripetizione, ma proteggono dall'attacco di sostituzione dei record, infatti:
  - L'intruso intercetta e registra tutti i messaggi scambiati tra il client ed il server
  - Qualche giorno dopo, l'intruso apre una connessione TCP con il server ed invia la stessa sequenza di record
  - Il server pensa che il client abbia ripetuto lo stesso acquisto. Ogni record supera il controllo di integrità.
- soluzione: il client invia un diverso nonce casuale per ogni connessione. Di conseguenza le chiavi di criptografia cambiano ad ogni sessione.
  - L'attacco di ripetizione non supera il controllo di integrità.

SSL: formato del record

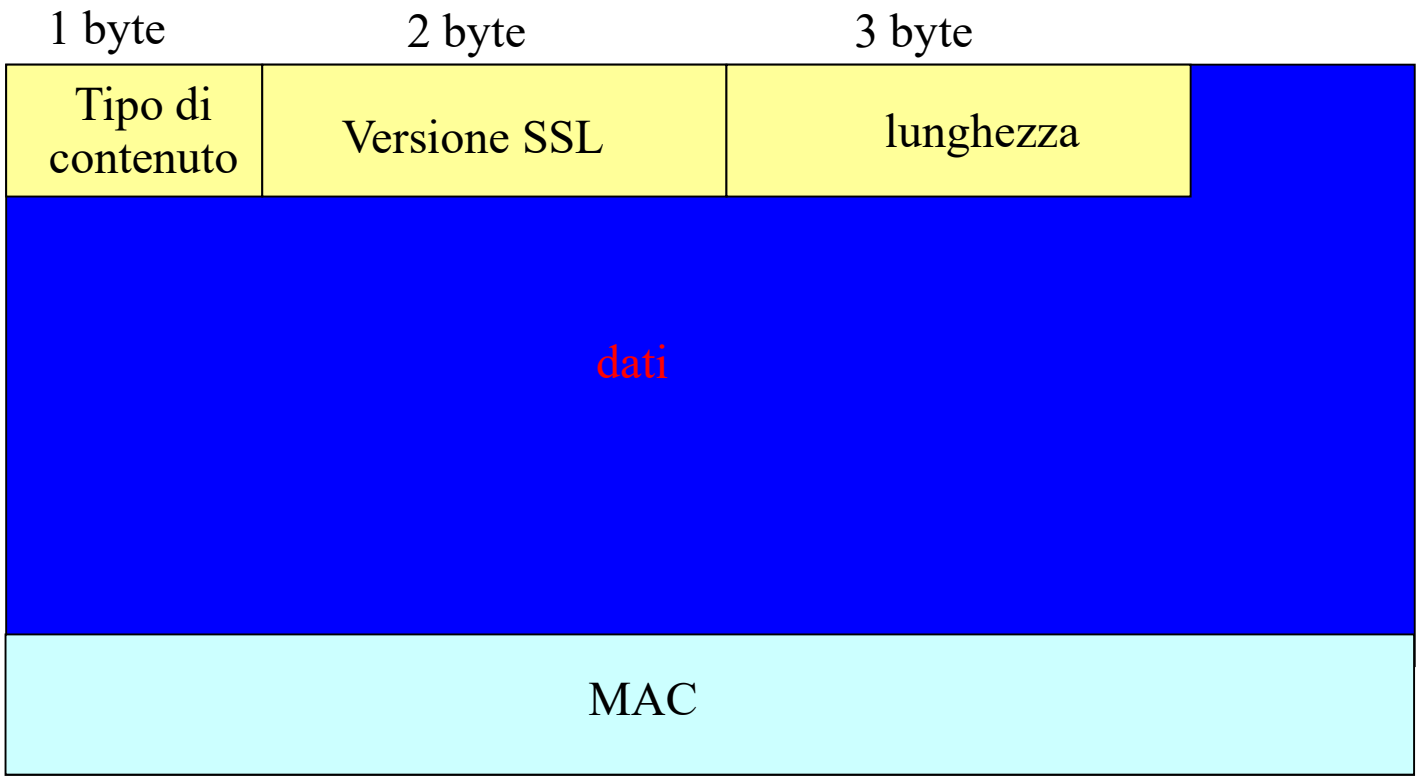


record header: tipo del contenuto; versione; lunghezza

MAC: include numero di sequenza, chiave MAC M_x

Frammento dati: ogni frammento SSL è 2^{14} byte (16 Kbyte)

SSL: formato del record



Dati e MAC sono criptati con un algoritmo a chiave simmetrica

SSL apertura connessione

handshake: ClientHello

handshake: ServerHello

handshake: Certificato

handshake: ServerHelloDone

handshake: ClientKeyExchange

ChangeCipherSpec

handshake: Finished

ChangeCipherSpec

handshake: Finished

application_data

application_data

Alert: warning, close_notify

Tutti i messaggi sono criptati

Si conclude con TCP FIN

Derivazione delle chiavi

- Il nonce del client, il nonce del server, la pre-master secret vengono date in input ad un generatore di numeri pseudo casuali.
  - Si ottiene la master secret
- La master secret e i nuovi nonce vengono dati in input ad un secondo generatore di numeri pseudo casuali: “blocco delle chiavi”
- Il blocco delle chiavi scompone la master secret e produce:
  - la chiave MAC del client
  - la chiave MAC del server
  - la chiave di criptografia del client
  - la chiave di criptografia del server
  - il vettore di inizializzazione (IV) del client
  - il vettore di inizializzazione (IV) server

Chiusura della connessione

- Ad un certo punto, il client o il server decidono di chiudere la sessione SSL. Il client potrebbe terminare la sessione SSL semplicemente terminando la connessione TCP su cui si appoggia – cioè, il client invia un segmento TCP FIN al server.
- Il rischio è che un attaccante potrebbe troncare anticipatamente la connessione, inviando un segmento TCP FIN. Il server potrebbe pensare che il client non ha più dati da trasmettere, mentre in realtà ne ha ricevuto solo una parte.
- La soluzione a questo problema è di indicare nel campo “tipo” se il record serve per terminare la sessione SSL session. (il campo “tipo” è trasmesso in chiaro, ma è autenticato al ricevitore usando il MAC del record).
- Se il server riceve un segmento TCP FIN prima di ricevere un record di chiusura SSL, capisce che è un segmento da scartare..