TLS: Transport Layer Security (version 1.3, 2018)

- basato su SSL (Netscape, mid-1990s)
- versione 1.0, 1999
- versione 1.1, 2006
- versione 1.2, 2008
- versione 1.3, 2018

- descrizione (semplificata e ad alto livello)
- Il protocollo TLS permette a un CLIENT (web browser) e a un SERVER (sito web) di stabilire un insieme di chiavi simmetriche condivise, da utilizzare per cifrare e autenticare la sessione di comunicazione.

- Come SSL, consiste di due parti

1) HANDSHAKE protocol

si esegue un protocollo di scambio di chiavi per stabilire le chiavi simmetriche condivise

2) RECORD-LAYER protocol

utilizza le chiavi condivise per cifrare e autenticare la comunicazione

- TLS permette al client e al server di autenticarsi reciprocamente, ma è tipicamente usato solo per autenticare il SERVER (l'autenticazione del CLIENT, se necessaria, avviene quando la sessione TLS è avviata, ad esempio tramite login/password.)

PROTOCOLLO di HANDSHAKE

client C:

possiede un insieme di chiavi pubbliche

$pk_1, pk_2, \dots$

di CA

server S:

possiede una coppia di chiavi pubblica/private

pk_S, sk_S

per la firma digitale

e un certificato $cert_S$ per pk_S rilasciato da una delle CA di cui C ha la chiave pubblica

PASSO 1 CLIENT

- C invia a S il messaggio iniziale del protocollo DH per lo scambio di chiavi, che include
  - la specifica del gruppo $\mathbb{G}$ usato dal client, l'ordine e il generatore g

    il gruppo $\mathbb{G}$ può essere $\mathbb{Z}_p^*$ (p primo), oppure una curva ellittica
  - il valore g^x calcolato usando un intero segreto x scelto casualmente da C
  - un nonce N_C (sequenza random di bit)
  - informazioni sulle ciphersuites che è in grado di supportare

PASSO 2 SERVER

- S completa il protocollo DH inviando un messaggio al client che contiene g^y, calcolato usando un intero segreto y scelto casualmente da S
- S invia inoltre un suo nonce N_S
- S calcola $K = g^{xy}$

  e applica una KEY-DERIVATION FUNCTION per estrarre da K le chiavi

  $$k'_S, k'_C, k_S, k_C$$

  per una cifratura autenticata.
- S invia la propria chiave pubblica pk_S, il certificato $cert_S$, la firma σ calcolata usando la chiave privata sk_S su tutti i messaggi di handshake inviati.
  Tutti i dati inviati sono cifrati con k'_S

PASSO 3 CLIENT

- C calcola K, e deriva le chiavi k'_S, k'_C, k_S, k_C
- Usa k'_S per recuperare pk_S, il certificato $cert_S$, e la firma σ
- Controlla se possiede la chiave pubblica della CA che ha rilasciato $cert_S$.
  In questo caso verifica il certificato.
- Verifica la firma σ sui messaggi di handshake usando pk_S
- Calcola il MAC dei messaggi di handshake scambiati usando k'_C e lo invia a S.

PROTOCOLLO RECORD LAYER

SE TUTTI I PASSI VANNO A BUON FINE, SI PROCEDE AL PROTOCOLLO RECORD LAYER

C usa k_C per cifrare i messaggi che invia a S

S usa k_S per cifrare i messaggi per C

DISCUSSIONE

- Alla fine del protocollo di HANDSHAKE, C e S condividono le chiavi di sessione

  k_C e k_S

  che possono usare per cifrare e autenticare la comunicazione (k_C' e k_S' sono usate solo per l'handshake).

- Dato che C verifica il certificato, sa che pk_S è la chiave pubblica corretta di S.
  Se la firma σ è valida, C conclude che sta comunicando con S (è l'unico che conosce la chiave privata sk_S associata a pk_S)

- Dato che S firma tutti i messaggi scambiati per l'esecuzione del protocollo DH, C sa che nessun valore è stato modificato (non ci sono stati attacchi man-in-the-middle)

- Il protocollo DH assicura che nessun crittoanalista passivo possa ottenere informazioni su K (e sulle chiavi derivate da K)

  ↳ Alla fine della fase di HANDSHAKE, C sa che condivide le chiavi k_c, k_s con il vero ultimo S

TLS 1.3 vs TLS 1.2 (e SSL)

- Nelle versioni precedenti (SSL, TLS 1.2) C e S potevano stabilire la chiave K usando un cifrario a chiave pubblica al posto del protocollo DH

  ↳ il client sceglieva la chiave K e la cifrava con la chiave pubblica pk_S di S

- Il client verificava il certificato $cert_S$ prima della cifratura

- In TLS 1.3 non è più possibile

  ↳ per garantire la FORWARD SECURITY, cioè la segretezza delle chiavi di sessione precedenti nel caso di un server compromesso

- DH fornisce forward secrecy dato che il valore y del server usato nel protocollo di handshake può essere cancellato alla fine del protocollo
  (e senza y, il crittoanalista non può ricostruire k)

- Invece, usando un cifrario a chiave pubblica, non si ha forward secrecy dato che la chiave privata sk_S del server non può essere cancellata

  ↳ se un avversario la ottiene, può decifrare crittogrammi scambiati nelle esecuzioni passate del protocollo di handshake e recuperare le chiavi di sessione usate dalle parti coinvolte.

Autenticated Encryption

combinazione tra cifratura e autenticazione del messaggio al fine di fornire simultaneamente
- confidenzialità
- autenticità
- e integrità del dato trasmesso

→ Encrypt-then-MAC

prima si cifra, poi si genera il MAC a partire dal cifrato

$m' = C(m, k) \,||\, MAC(C(m, k))$

due chiavi (una per C, una per MAC)

→ Encrypt-and-MAC

$m' = C(m, k) \,||\, MAC(m)$

una sola chiave, usata sia per cifrare che per il MAC

→ MAC-then-Encrypt (usato in TLS)

$m' = C(m \,||\, MAC(m), k)$

una sola chiave, usata sia per cifrare che per il MAC