Secure Socket Layer ↑ Transport Layer Security ↗

Part II: SSL / TLS

HW: Watch the lecture by David Benjamin from Real World Crypto (2018)

Information security:

Ensure integrity and confidentiality of the information from the sending point to the receiving point.

- We learned about various components:

  hash functions, symmetric encryption, MACs, public key encryption, signature schemes...

SSL/TLS: Protocol that ensures end-to-end security. This protocol is what allows us to achieve secure e-commerce.

Requires paying attention to:

Authenticity, Confidentiality, Chosing best params & key management

Very important!

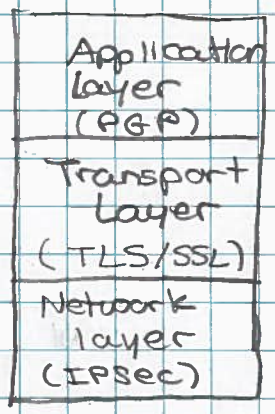
- Two endpoints of a communication link may have different computational power, may have access to different cryptographic algorithms, etc.

There are three 3 layers where end-to-end security can be provided:

1. Network layer (Eg. IPSec protocol) (Internet Protocol Security)
   - Makes it difficult to customize the security policies to specific applications.

2. Security for TCP (Transmission Control Protocol) packets:
   - This is done using Transport Layer Security (TLS) protocol

3. Security in the applications itself (Eg. PGP protocol / (Pretty Good Privacy) used for email applications



SSL / TLS for Transport Layer Security (TLS)

1995: SSL (Secure Socket Layer) was developed by Netscape

Provided end-to-end security, originally between browsers & servers

1999: SSL version 3 became Open Standard, and is called TLS (Transport Layer Security) version 1.

(TLS 1.0, 1.1, 1.2, TLS 1.3) — the secure version not widely deployed yet see lecture by David Benjamin.

Often called SSL/TLS

- The most prominent Internet security protocol.
- Every HTTPS connection is secured via SSL/TLS.

* As we will see, fundamental to the security of SSL/TLS are the certificates issued by the Certification Authority (CA).

* SSL/TLS allows for either server-only authentication (most common), or server-client authentication (less common).

SSL/TLS is composed of 4 protocols:

- Handshake Protocol:
  - Agree on which crypto primitives to use
  - Authenticate
  - Establish keys to be used in the Record Protocol

Main Ingredients

- Record Protocol:
  - Provides confidentiality & integrity/authenticity of the data using the keys from the Handshake Protocol. (and encryption algs)

- Cipher Change Protocol (used only for compatibility purposes)
- Alert Protocol (Alerts is errors occur)

} minor

Vocabulary:

- A connection is a one-time transport of information between two nodes in a communication network.
- Every connection is associated with a session

A session is an enduring association between two nodes (client & server), and can consist of multiple connections.

A session is created by the SSL Handshake Protocol

A session allows data to transfer back and forth, without having to renegotiate the secret keys & parameters for each connection.

The Handshake Protocol

Consists of 4 phases:

Phase 1: Initiated by the client, establishes the security capabilities of the client & server (or more generally, the two ends of a connection).

Specifically, the client sends a client_hello msg which consists of the following parameters:

- Version (the highest SSL version understood by the client)
- Random value N_c (to be used as nonce during the key exchange, to prevent replay attacks)
  256 bit nonce
- Session ID

- Cipher Suit : A list of cryptographic algorithms supported by the client, in decreasing order of preference.

- Compression Method : A list of compression methods the clien supports (to compress data before encrypting)

often not used (due to security vulnerabilities) Not used in TLS 1.3

- Server responds with its server_hello msg, which includes specific algorithms selected by the server from the client lists.

It includes two elements declaring :

1. key exchange method
2. CipherSpec, indicating the authentication alg selected, the encryption alg selected etc.

It also includes a 256-bit nonce Ns

Phase 2 Server sends his certificate to the client. Could be followed by a certificate request msg (if server wants to validate the client)

most clients don't have certified PK whereas servers do.

Could be followed by server_key_exchange msg (if they chose to choose a common sk using DH key Exchange)

Phase 2 ends with the server sending server_hello_done msg.

Client hello : TLS_ECDHE_AES256_CBC_SHA →

← Server_hello, Cert., Server_hello_done + server_key_exchage (?)

Check Cert.
Extract PK_S from cert.

Phase 3 : Client sends his certificate (if it was requested) ← This is done rarely.

Client sends client_key_exchange msg :

This could either be the second msg of a DH key exchange, or can be a session PMK, pre-master key encrypted w. PK_S using RSA. Today, most often DH is used.

Both parties compute

$$MK = KDF(PMK, N_C, N_S)$$

KDF ← Key Derivation Function, most commonly HMAC based

$$MK = (K_C, K_C', K_S, K_S')$$

K_C, K_C' : client's keys for enc & auth

K_S, K_S' : server's keys for enc & auth

Phase 4: Client sends $MAC_{K'_c}$(transcript)

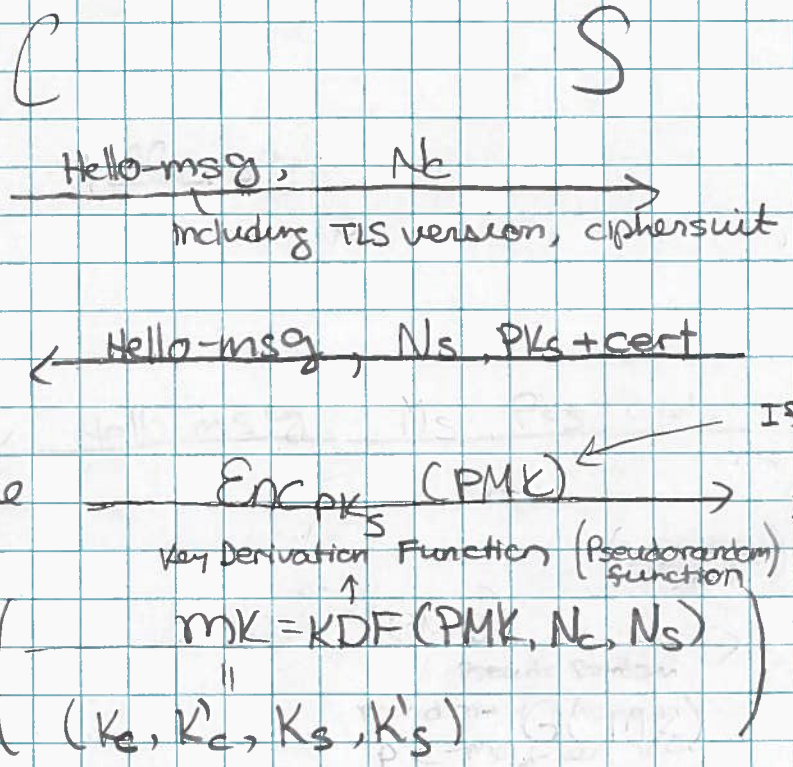
↑ checks the integrity of the Handshake protocol

Server " $MAC_{K'_s}$(transcript')

↑ includes client previous msg.

Client & Server declare this hand shake successful only if the MAC's verify.

Summary of Handshake Protocol



Important Notes:

- N_c & N_s are used to prevent replay attacks

  (prevent copying the same exact handshake).

- Different (independent) keys are later used for encrypting and authentication (this is done in the record protocol)

- Server & Client use different keys!

  To prevent reflection attacks

  (a man-in-the-middle copying msgs of server back to him)

What goes wrong?

① If key exchange is w. RSA then:

  * If later SK_s is revealed, no security!
  * Also, current RSA implementations are not CCA secure, and are vulnerable.

TLS 1.3: Only supports DH key Exchange.

② Which group is used in DH key Exchange?

Server chooses in current versions (TLS 1.0–1.2)

This is a problem since servers often make a bad choice

In TLS 1.3 server cannot choose the group. Rather, it has a few options, and must choose one of the available options

③ Downgrade attacks: A MIM can cause the client to believe that the server does not support the secure versions. Then he will be able to forge the MAC (since the downgraded version is not stuff secure)

In TLS 1.3: Server signes the transcript of the HandShake using his SK.

④ Record protocol in previous versions SSL 3.0

breaks msgs into blocks, and MACs then encrypts. (after padding)

known to be problematic! (POODLE Attacks)
Padding Oracle On Downgraded Legacy Encryption

TLS 1.3 uses authenticated encryption

The hello msg includes an authenticated enc scheme & a KDF hash function, (EC) DH group, sig alg

↑ HMAC based