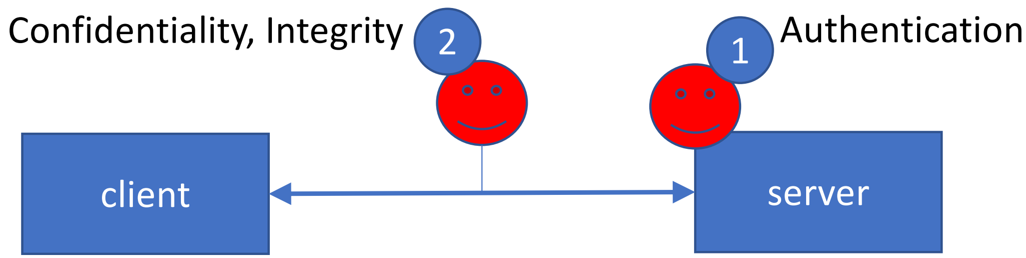
Security of Networks, Services, and Systems

TLS - Transport Layer Security

Ricardo Morla

FEUP – SSR/M.EEC, SR/M.EIC

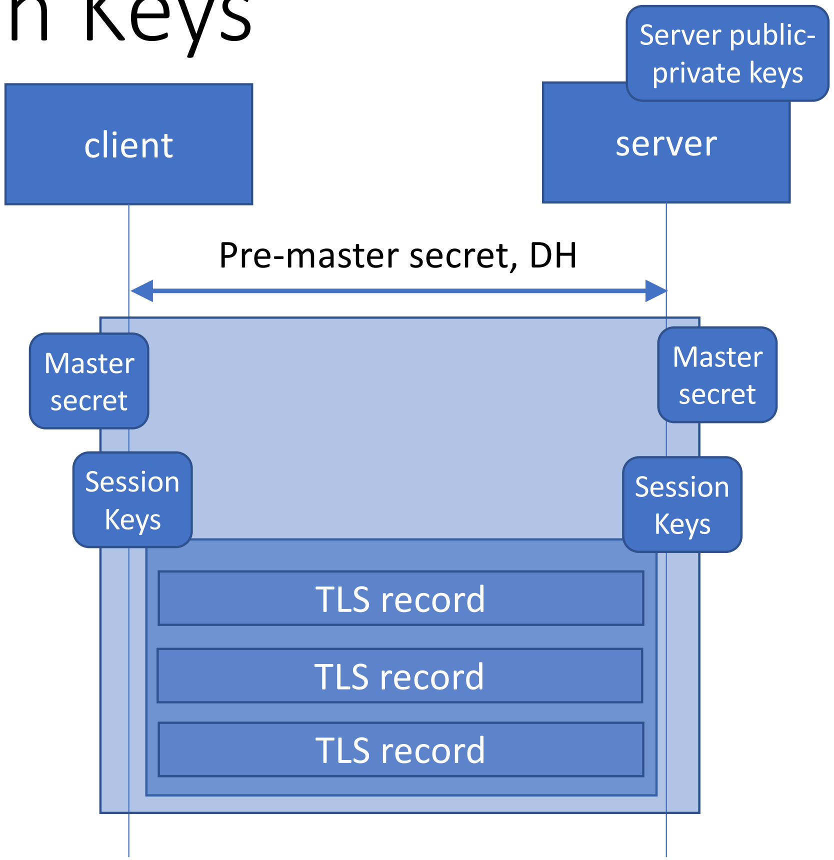
Overview



- RFC 5246 (TLS 1.2), RFC 8446 (TLS 1.3)
- Secures end-to-end communications between web client and web server on top of TCP
- Provides confidentiality, web server authentication, and data integrity
  - Authentication 1
    - The server sends certificate to client, client **validates** certificate
    - The server then can use its public key to sign any new information it wants to send and the client will be able to authenticate it
  - Session data confidentiality 2
    - The **client and server encrypt** data with shared keys for efficiency
    - They must agree on shared key for each session
  - Session data integrity 2
    - Compute **MAC** and concatenate to message

Confidentiality and Encryption Keys

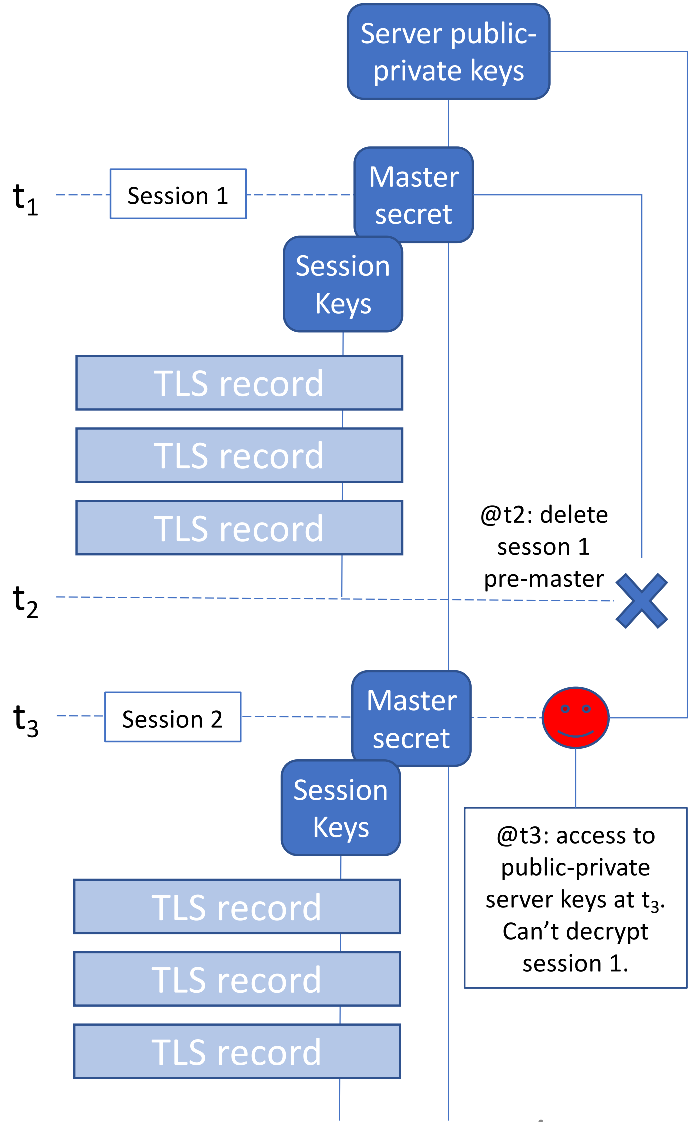
- Pre-master secret
  - randomly generated or result of DH key exchange
- Master secret
  - generated from a pseudo-random function (RFC 5246, section 5)
  - on the pre-master secret and client/server "Hello" random values (RFC 5246, section 8.1)
- Session keys
  - used to actually encrypt the data with symmetric cipher
  - generated from master key and client/server random values (RFC 5246, section 6.3)
  - generates client/server write keys for MAC, encryption, and Initialization Vectors



Perfect Forward Secrecy

https://en.wikipedia.org/wiki/Forward_secrecy

- Pre-master keys required to decrypt session data
- Vulnerable if:
  - Pre-master keys encrypted with the server's public key are vulnerable to attack in the future
  - Pre-master keys exchanged using static DH parameters
- Ephemeral (used only once) DH parameters can be used, which generate unique private/public keys for each session
- If the attacker only has access to your private key and not to the ephemeral keys, it will not be able to decrypt your data
- This is called perfect forward secrecy



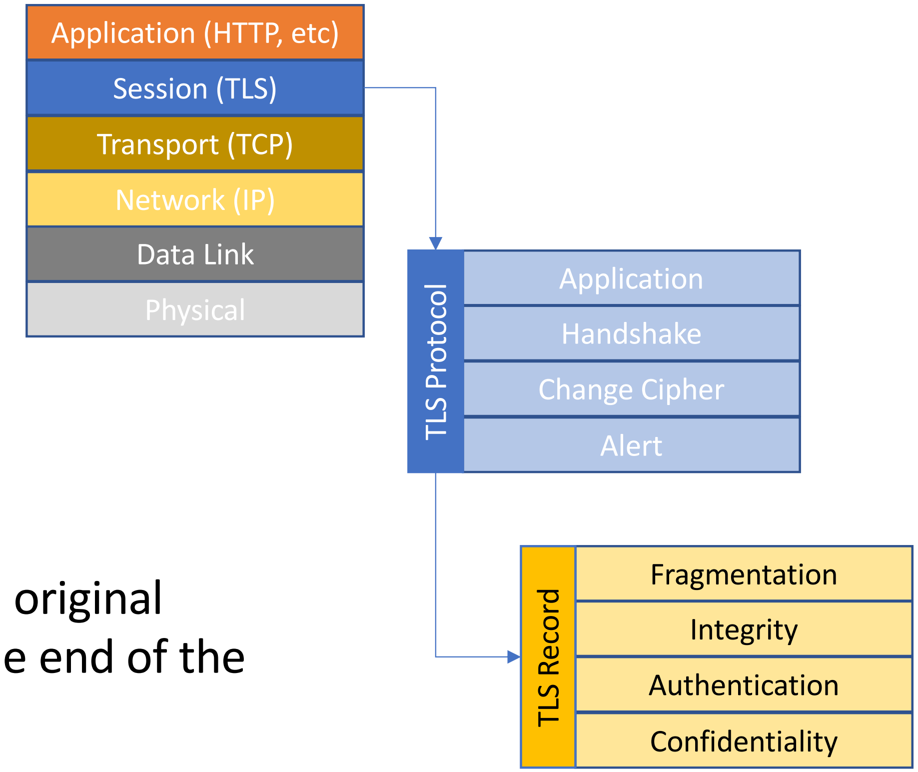
Algorithms

https: //en.wikipedia.org/wiki/Transport_Layer_Security

- TLS supports may different algorithms
- Key exchange
  - RSA, DHE-RSA, ECDHE-RSA, ...
- Ciphers (symmetric encryption)
  - AES with different modes of operation, Camellia, 3DES;
  - ChaCha20-Poly135 stream cipher

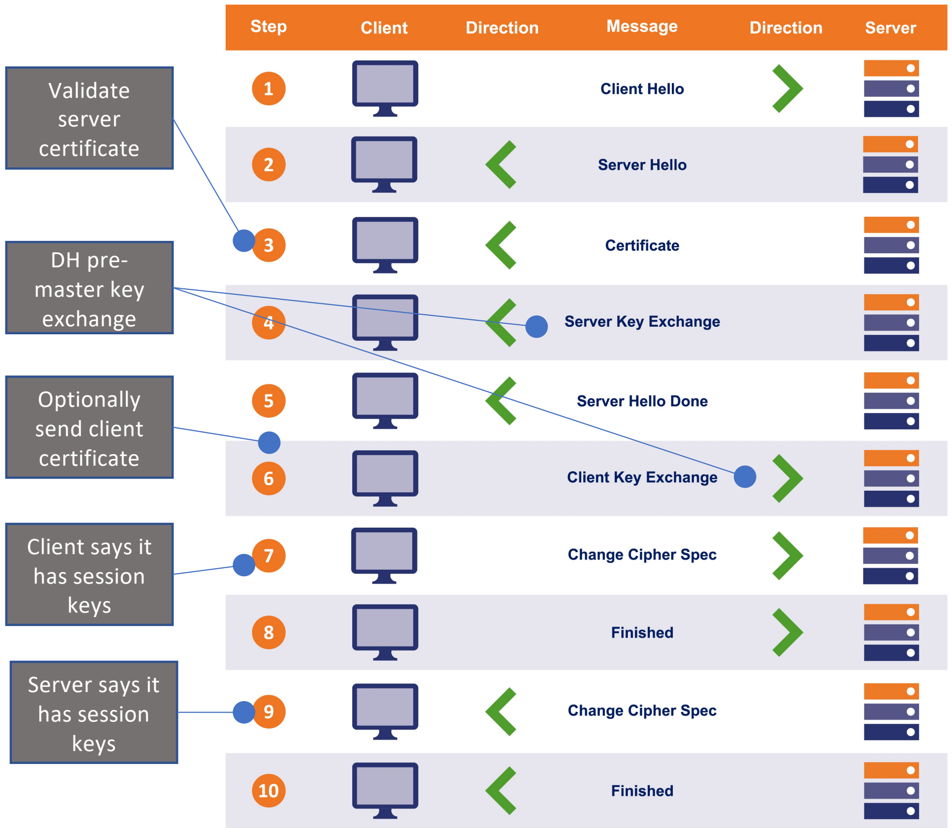
TLS Records

- TLS Record Protocol runs on top of TCP
  - Application
  - Handshake
  - Change Cipher
  - Alert
- TLS Record fragments
  - With up to 2^{14} bytes per fragment
  - Compressed, small records may be larger than original
  - A Message Authentication Code is added to the end of the compressed fragment
  - Message are encrypted with session key
  - TLS Record header and MAC added



Handshake

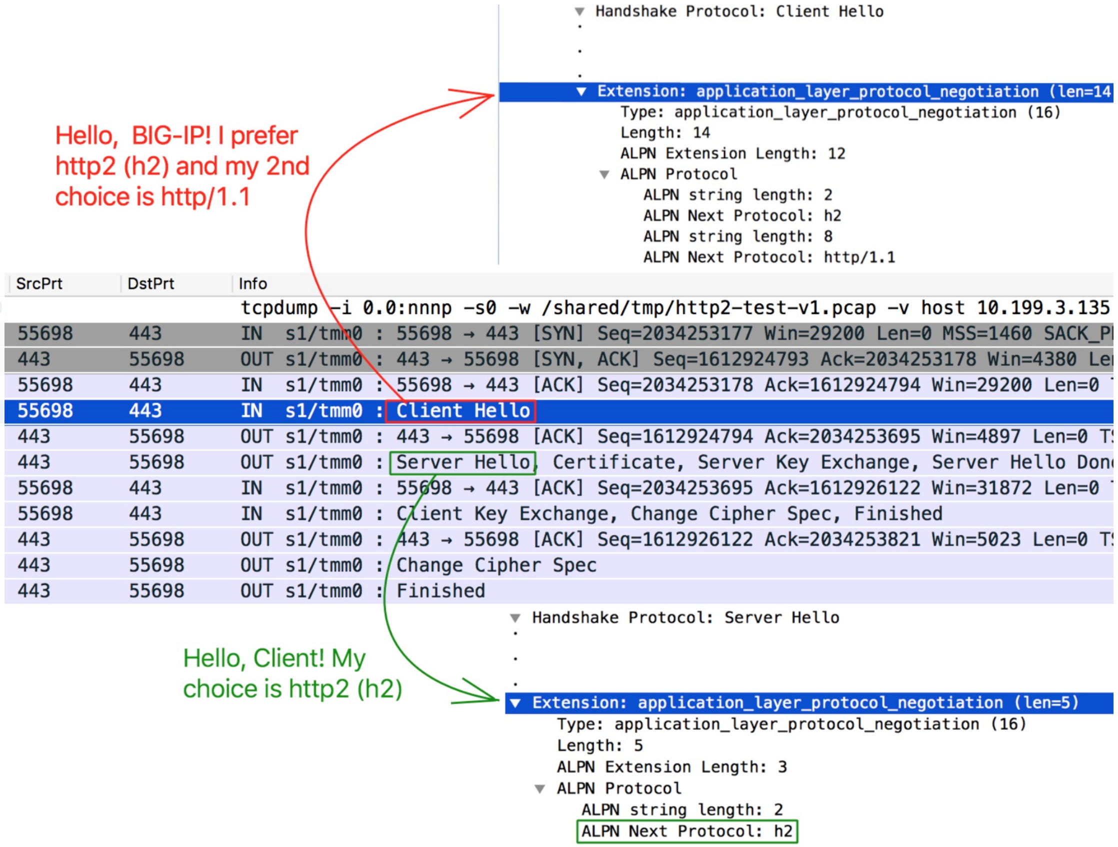
- TCP syn/ack
- ClientHello
  - Client random number, available ciphers and key exchange algorithms, SNI server name
- ServerHello
  - Server random number, chosen cipher
  - Optionally the server can request for the client to authenticate itself
- Server certificate
- KeyExchange (client/server)
  - either RSA or DH
  - pre-master secret setup at both ends
- Change Cipher Spec
  - Session key computed, ok to start sending encrypted data
- At any time Alert messages can be sent



https://www.thesslstore.com/blog/explaining-ssl-handshake/

Application Layer Protocol Negotiation

- TLS end-to-end encrypted tunnel obfuscates the type of data from proxies
- Browsers may request server capabilities regarding HTTP protocols supported and vice-versa
  - HTTP/1.0, HTTP/1.1, HTTP/2.0, SPDY/3.1
- TLS Extension to handshake allows browser to tell which protocols it supports
- Server inspects list and returns the chosen application layer protocol
- This is done in plaintext



HTTPS

- Specifies port number, URL format, URL to certificate mapping
- HTTP/1.1 (RFC 2818)
- HTTP/2.0 (RFC 7540)
- A good discussion about TLS, HTTP2, and security in general
  - https://daniel.haxx.se/blog/2015/03/06/tls-in-http2/

Security of Networks, Services, and Systems

TLS - Transport Layer Security

Ricardo Morla

FEUP – SSR/M.EEC, SR/M.EIC