Transport Layer Security (TLS) v1.3

- What is TLS?
- TLS handshake protocol
- TLS record protocol
- TLS 0-RTT mode
- Limitations

Transport Layer Security (TLS) v1.3

- TCP is not secure – neither the TCP/IP headers nor the data are encrypted or authenticated
- The **Transport Layer Security** protocol (TLS v1.3) can be used to encrypt and authenticate data carried **within** a TCP connection
  - Official specification:
    - https://tools.ietf.org/html/rfc8446 – The Transport Layer Security (TLS) Protocol Version 1.3
  - Useful blog posts:
    - https://ietf.org/blog/tls13/ – Introduction to TLS 1.3 from IETF
    - https://blog.cloudflare.com/rfc-8446-aka-tls-1-3/ – A detailed look at what's new in TLS 1.3
    - https://davidwong.fr/tls13/ – The TLS 1.3 specification, redrawn in a readable way with explanatory videos and comments
    - https://tls13.ulfheim.net – An annotated packet capture, showing details of a TLS Connection

TLS v1.3 Goals

> The primary goal of TLS is to provide a secure channel between two communicating peers; the only requirement from the underlying transport is a reliable, in-order data stream. Specifically, the secure channel should provide the following properties:
>
> - **Authentication:** The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via asymmetric cryptography (e.g., RSA [RSA], the Elliptic Curve Digital Signature Algorithm (ECDSA) [ECDSA], or the Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032]) or a symmetric pre-shared key (PSK).
>
> - **Confidentiality:** Data sent over the channel after establishment is only visible to the endpoints. TLS does not hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.
>
> - **Integrity:** Data sent over the channel after establishment cannot be modified by attackers without detection.

E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3”, IETF, August 2018, RFC 8446, https://tools.ietf.org/html/rfc8446

TLS v1.3 Overview

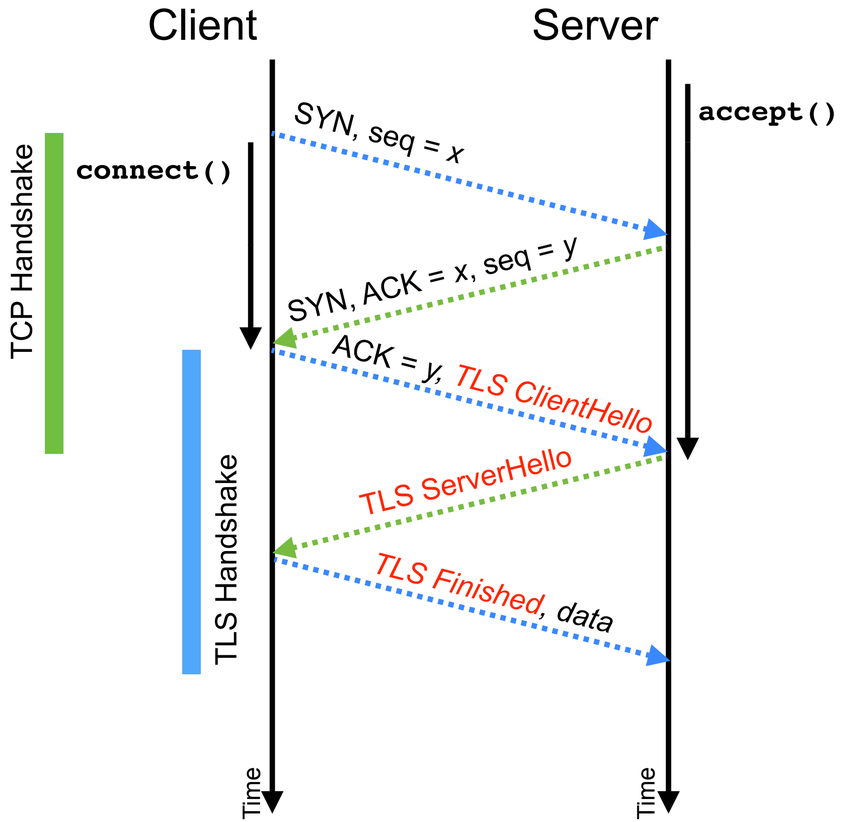
> TLS consists of two primary components:
>
> - A handshake protocol (Section 4) that authenticates the communicating parties, negotiates cryptographic modes and parameters, and establishes shared keying material. The handshake protocol is designed to resist tampering; an active attacker should not be able to force the peers to negotiate different parameters than they would if the connection were not under attack.
>
> - A record protocol (Section 5) that uses the parameters established by the handshake protocol to protect traffic between the communicating peers. The record protocol divides traffic up into a series of records, each of which is independently protected using the traffic keys.

E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3”, IETF, August 2018, RFC 8446, https://tools.ietf.org/html/rfc8446

- A TCP connection is established
- TLS handshake protocol runs within that TCP connection
  - Authenticates endpoints and agrees on the encryption keys to use
- TLS record protocol then runs over the TCP connection, lets endpoints exchange authenticated and encrypted blocks of data
  - TLS turns the TCP byte stream into a series of records → provides framing

TLS v1.3 Handshake Protocol

- TCP connection established as usual
  - **`SYN`** → **`SYN+ACK`** → **`ACK`**
- TLS handshake protocol immediately follows
  - TLS **`ClientHello`** sent with the **`ACK`**
  - TLS **`ServerHello`** sent in response
  - TLS **`Finished`** message concludes, and carries initial secure data record
- Adds 1-RTT to connection establishment



TLS v1.3 `ClientHello`

> - **Key Exchange: Establish shared keying material and select the cryptographic parameters. Everything after this phase is encrypted.**

E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3”, IETF, August 2018, RFC 8446, https://tools.ietf.org/html/rfc8446

- The **`ClientHello`** message:
  - Indicates that TLS v1.3 is to be used
    - Signals TLS **v1.2** in the main **`ClientHello`** message, with an extension header to say “I’m really TLS v1.3” because too many middleboxes break if the TLS version changes – *protocol ossification*
  - Provides the cryptographic algorithms the client supports
  - Provides the name of the server to which the client is connecting
    - If connecting to a web hosting server, it’s likely that more than one site is hosted on the same server, so need to specify which is intended
  - Does not contain any data

TLS v1.3 `ServerHello`

> - **Server Parameters: Establish other handshake parameters (whether the client is authenticated, application-layer protocol support, etc.).**
>
> **The server processes the ClientHello and determines the appropriate cryptographic parameters for the connection. It then responds with its own ServerHello (Section 4.1.3), which indicates the negotiated connection parameters. The combination of the ClientHello and the ServerHello determines the shared keys**

E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", IETF, August 2018, RFC 8446, https://tools.ietf.org/html/rfc8446

- The **`ServerHello`** message:
  - Indicates that TLS v1.3 is to be used
    - It signals TLS **v1.2** in the main **`ServerHello`** message, and includes an extension header to say "I'm really TLS v1.3" because too many middleboxes break if the TLS version changes – *protocol ossification*
  - Provides cryptographic algorithms selected by the server, from the set the client suggested
  - Provides the server's public key and digital signature used to verify its identity
  - Does not contain any data

TLS v1.3 `Finished`

> - **Authentication: Authenticate the server (and, optionally, the client) and provide key confirmation and handshake integrity.**

E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.3”, IETF, August 2018, RFC 8446, https://tools.ietf.org/html/rfc8446

- The **`Finished`** message:
  - Provides the client’s public key
  - Optionally, provides the certificate needed to authenticate the client to the server
  - May contain data sent from client to server

TLS v1.3 Cryptographic Algorithms

- Client and server use the Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) key exchange algorithm
  - Client sends its public key to server in **`ClientHello`**
  - Server sends its public key to client in **`ServerHello`**
  - Client and server both combine the two public keys to derive the key used for the symmetric cryptography – complex mathematics; will not describe
  - Server provides a certificate (digital signature) to allow the client to verify its identify in the **`ServerHello`** – client can optionally provide this in **`Finished`**
- Client proposes symmetric encryption algorithms in **`ClientHello`**, server picks from them and replies in **`ServerHello`**
  - Either AES or ChaCha20 symmetric encryption

TLS v1.3 Record Protocol

- TLS v1.3 splits the data into **records**, each containing ≤ 2^{14} bytes
  - Each record is both encrypted and authenticated; it has a sequence number
  - Can renegotiate encryption keys between records
  - TCP does not preserve record boundaries; TLS adds **framing** so that it does – reading from a TLS connection will block until a complete record is received

- Client and server exchange records – send and receive data – then close connection

TLS v1.3 0-RTT Mode

- TLS v1.3 usually takes 1-RTT to establish a connection, after TCP connection setup
- If a client and server have previously communicated, they can re-use a key:
  - Server can send additional encryption keys as part of **`ServerHello`**, that client can use the *next* time is connect to that server
  - On next connection, **`ClientHello`** message can include data to be delivered to the server, encrypted with that pre-shared key; the **`ServerHello`** can contain a reply
  - This *0-RTT data* may be delivered more than once, if the **`ClientHello`** or **`ServerHello`** messages are duplicated – TLS doesn't stop this
    - Protection against duplicate messages is provided by sequence numbers in the record layer
    - Handshake messages do not have sequence numbers, so duplicated messages can deliver data more than once
    - **Be careful writing applications that use 0-RTT mode**

Limitations of TLS

- TLS **is secure**, but has limitations:
  - Does not encrypt server name:
    - Exposes hostname of server to which connection is being made https://datatracker.ietf.org/doc/draft-ietf-tls-esni/
    - Encrypted SNI extension in development – difficult to deploy https://blog.cloudflare.com/encrypted-sni/
  - Operates within a TCP/IP connection:
    - IP addresses and TCP port numbers are not protected – exposes information about who is communicating and what application is being used
  - Relies on a PKI to validate public keys:
    - Significant concerns about trustworthiness of PKI
  - May deliver 0-RTT data more than once

Transport Layer Security (TLS) v1.3

- What is TLS?
- TLS handshake protocol
- TLS record protocol
- TLS 0-RTT mode
- Limitations