Transport Layer Security (TLS)

Daniel Zappala

CS 465 · Introduction to Security and Privacy · Fall 2026

BYU


Varno za
foto filme
SECURITY
OFFICER



We have Diffie-Hellman and certificates. What does it take to turn them into one protocol every browser and server can use?

Negotiate algorithms, detect tampering, and protect every byte that is sent

SSL and TLS history

01

01

History

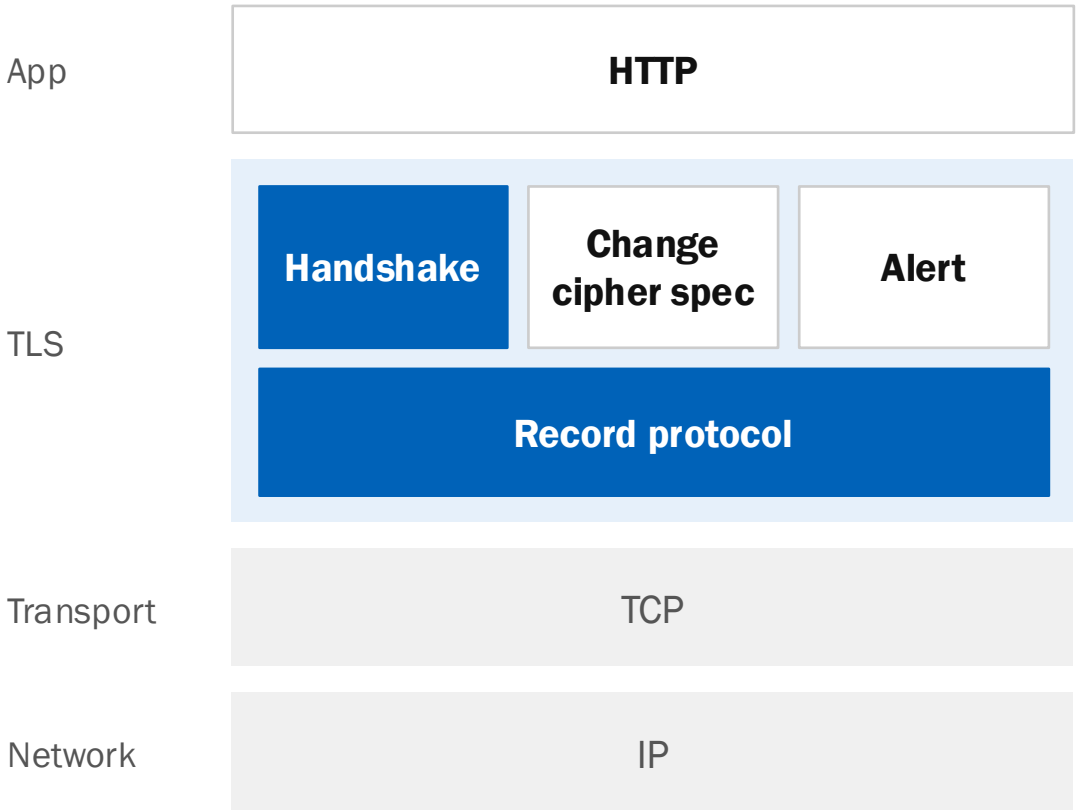
SSL became TLS, and every version before TLS 1.2 is now deprecated

1995	1996	1999	2006	2008	2018
SSL 2.0	**SSL 3.0**	**TLS 1.0**	**TLS 1.1**	**TLS 1.2**	**TLS 1.3**
Netscape's first public release; flawed design	Full redesign; broken by POODLE in 2014	IETF standardizes SSL 3.0 under a new name	Patches CBC padding and IV attacks	SHA-256 and AEAD modes; still widely used	One round trip; legacy options removed
Prohibited 2011	Prohibited 2015	Deprecated 2021	Deprecated 2021	In use	Preferred

RFC 8996 (2021) formally deprecated TLS 1.0 and 1.1; major browsers had already switched them off in 2020

Sources: E. Rescorla, SSL and TLS; RFC 6176, RFC 7568, RFC 8996, RFC 8446

TLS runs between TCP and HTTP in two layers



Handshake layer: set up the channel

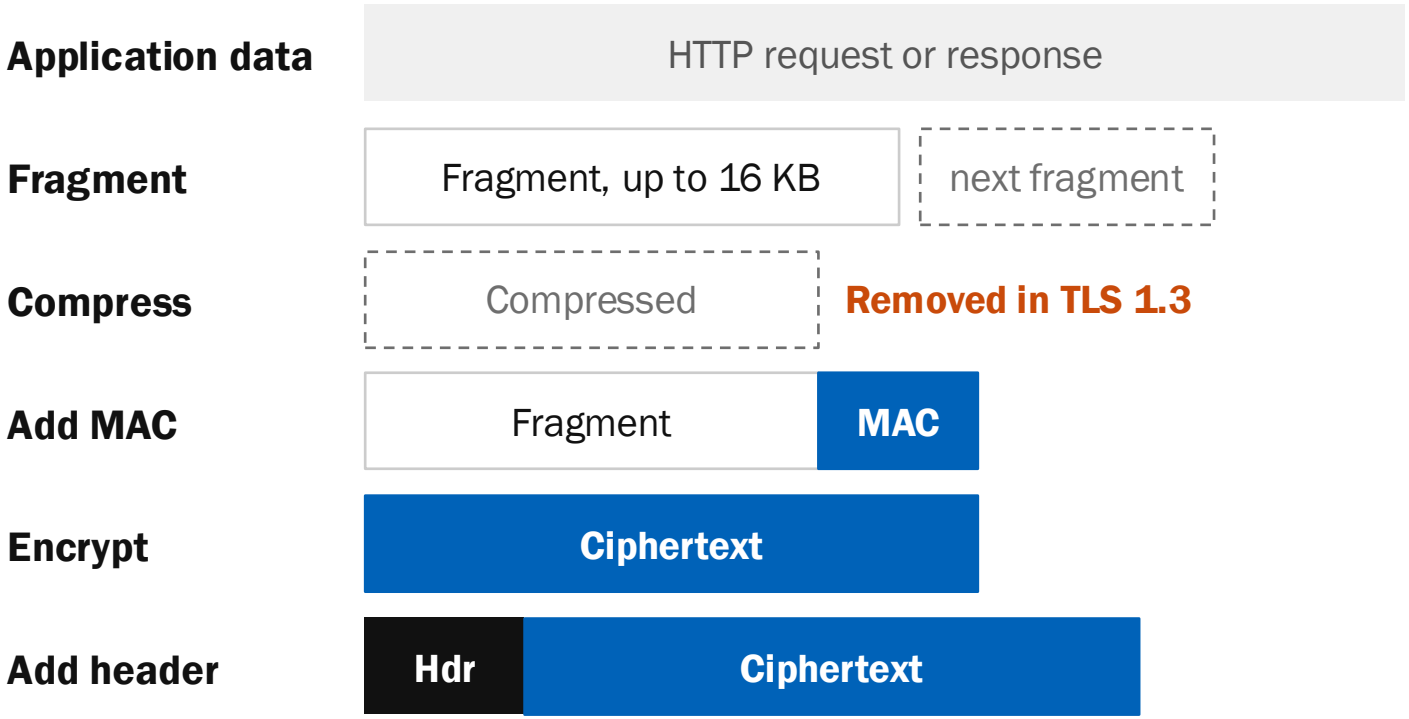
- **Key exchange:** agree on a shared master key
- **Server parameters:** settle every other option
- **Authentication:** server to client, and optionally client to server

Record layer: carry the data

- **Protects application data** with the negotiated keys
- **Each record** is labeled with its type, version and length

HTTPS is HTTP sent through a TLS channel over TCP, usually on port 443

The record layer splits, protects and labels every byte of data



Record header: 5 bytes

Content type · 1 byte
Handshake, alert, change cipher spec or application data

Version · 2 bytes
Protocol version, e.g. 3,3 for TLS 1.2

Length · 2 bytes
Size of the protected fragment

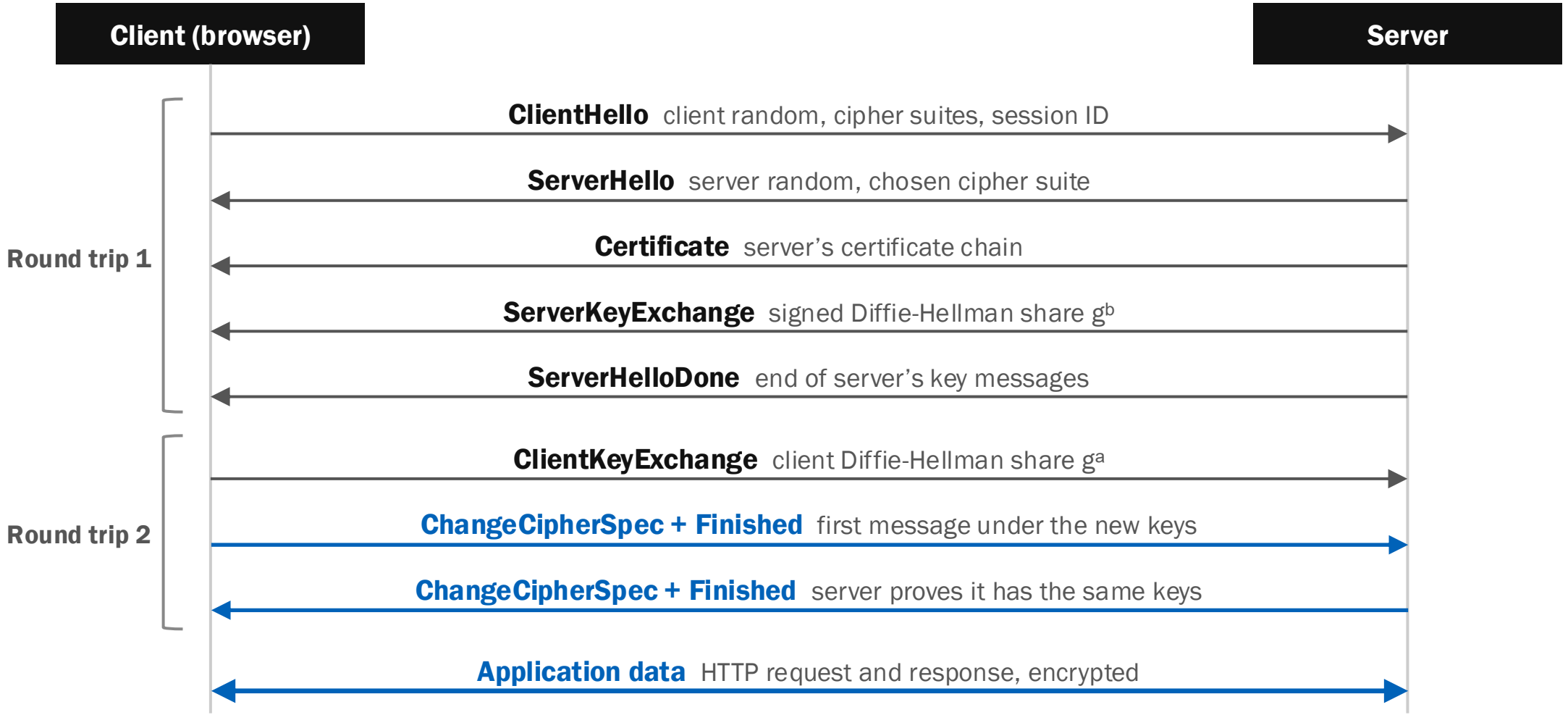
TLS 1.3: one AEAD step replaces MAC-then-encrypt, the CRIME attack ended compression, and the real content type now travels inside the encryption

Adapted from W. Stallings, Network Security Essentials; RFC 8446

The TLS handshake

02

The TLS 1.2 handshake takes two round trips before any data flows



Diffie-Hellman variant shown; RSA key transport also existed in TLS 1.2. Based on RFC 5246 and the CS 465 notes

02

Handshake

Each handshake message adds a nonce, a choice or a proof

Message	What it carries	Why it matters
ClientHello	Client random, cipher suites, session ID	Fresh randomness; a session ID can resume an earlier session
ServerHello	Server random, one chosen suite, session ID	Server picks from the client's list or closes the connection
Certificate	Chain from the server up to a trusted root	Binds the server's public key to its domain name
ServerKeyExchange	Diffie-Hellman parameters and both randoms, signed	The signature stops a middle-person from swapping in their own share
ServerHelloDone	Nothing	Marks the end, since ServerKeyExchange is optional
ClientKeyExchange	Client's Diffie-Hellman public value	Both sides can now compute the premaster secret
ChangeCipherSpec	A one-byte signal	Signals the switch to encryption; gone in 1.3

TLS 1.2 with ephemeral Diffie-Hellman; based on RFC 5246

02

Handshake

Finished messages catch any tampering with the handshake

What each side sends in its Finished message

PRF(master secret, label, **hash of all handshake messages**)

1 **Key confirmation**

Proves the other side derived the same master secret

2 **Transcript integrity**

If an attacker edits the offered cipher suites, the hashes differ and Finished fails

3 **Limit**

It cannot stop a client that voluntarily retries with an older version: POODLE exploited this fallback

TLS 1.3 fix: a 1.3 server pushed to an older version writes a fixed code into its random value, so the client detects the downgrade and hangs up

02

Handshake

Ephemeral Diffie-Hellman keeps recorded traffic safe even if the server key leaks

TLS 1.2 only · removed in TLS 1.3

RSA key transport

1. Client picks the premaster secret
2. Encrypts it with the server's long-term RSA public key
3. An attacker records the traffic and waits

Steal the key later: decrypt every recorded session

TLS 1.2 option · required in TLS 1.3

Ephemeral Diffie-Hellman

1. Each side picks a fresh secret, a and b
2. The server signs g^b with its long-term key
3. Both erase a, b and the session keys afterward

Steal the key later: impersonate from now on, but the past stays secret

Forward secrecy: leaking a long-term key does not expose session keys that were already used and erased

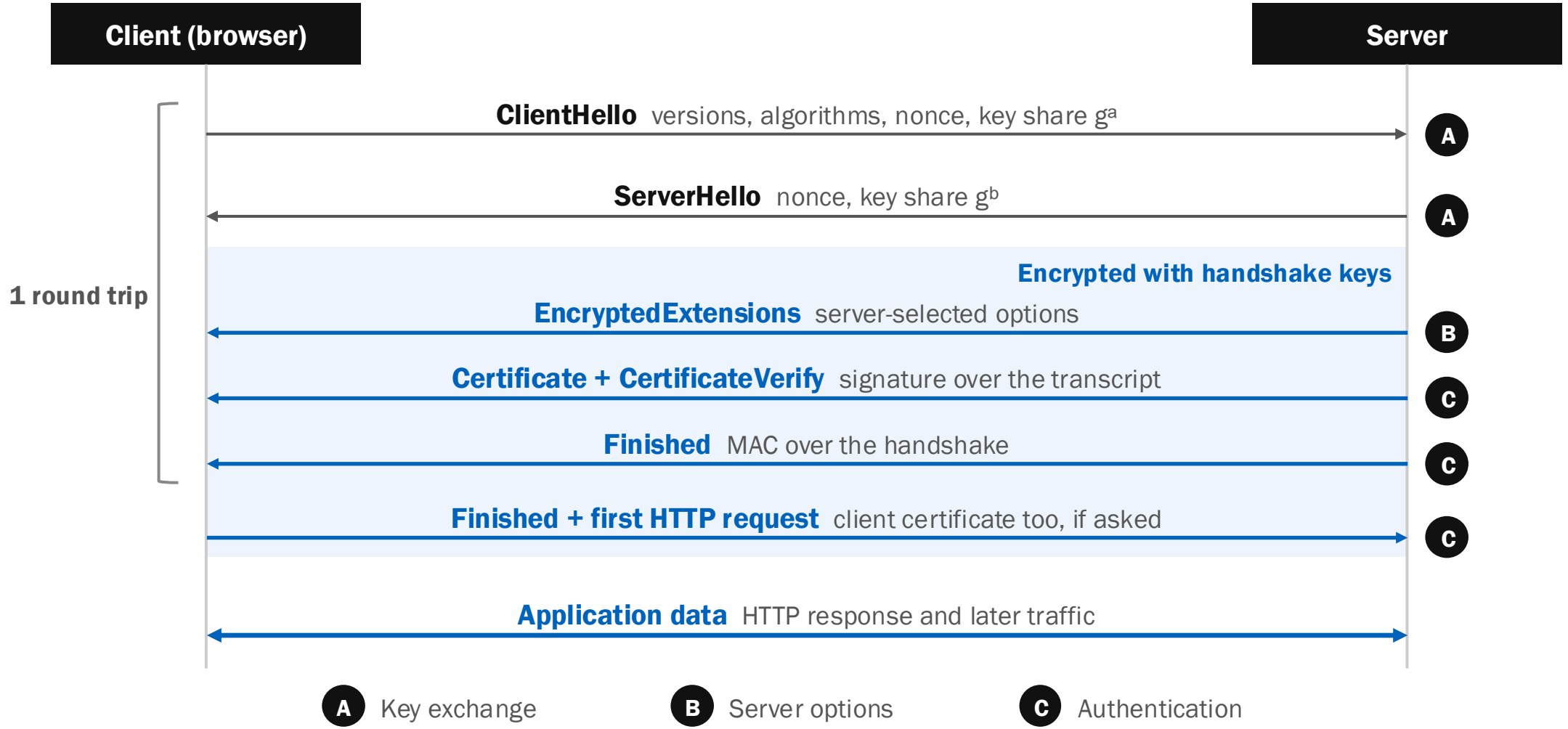
In TLS 1.2 you only got it by choosing ECDHE or DHE suites over TLS_RSA ones; TLS 1.3 removed the choice

Adapted from P. C. van Oorschot, Computer Security and the Internet, Ch. 4 and Sect. 9.2

02

Handshake

TLS 1.3 needs one round trip and encrypts everything after ServerHello



Adapted from P. C. van Oorschot, Computer Security and the Internet, Fig. 9.4; RFC 8446

02

Handshake

Each TLS 1.3 change removes a feature attackers exploited in TLS 1.2

TLS 1.3 change	Problem in TLS 1.2	Attacks it ends
One-round-trip handshake	Two round trips slowed every new connection	None: faster setup helps HTTPS spread
(EC)DHE or PSK key exchange only	RSA key transport gave no forward secrecy	Bleichenbacher, ROBOT
AEAD ciphers only	CBC padding and RC4 kept breaking	BEAST, Lucky 13, POODLE, RC4 biases
Signed transcript, fixed DH groups	Attackers forced export-grade or weak DH	FREAK, Logjam
No compression or renegotiation	Extra features leaked secrets or spliced sessions	CRIME, renegotiation attack
Encrypted certificate	Observers could see which certificate was sent	Passive tracking of sites

Source: RFC 8446 (TLS 1.3), Sect. 1.2; P. C. van Oorschot, Computer Security and the Internet, Sect. 9.2

02

Handshake

Resumption skips the full handshake, and 0-RTT trades replay safety for speed

Full handshake

1 RTT before the first request

Key share, certificate and signature every time

Server can then issue a session ticket holding a PSK

PSK resumption

1 RTT before the first request

Client returns the ticket; no certificate or signature needed

Add a fresh key share (PSK + DHE) to keep forward secrecy

0-RTT early data

0 RTT before the first request

HTTP request rides in the first flight, encrypted under the PSK

An attacker can replay it: allow only idempotent requests like GET

TLS 1.2 resumed with session IDs or tickets and took 2 RTT for a full handshake; TLS 1.3 folds both into PSKs

RTT = network round trip. Adapted from P. C. van Oorschot, Computer Security and the Internet, Sect. 9.2; RFC 8446 Sect. 8

TLS cipher suites

03

03

Cipher suites

A TLS 1.2 cipher suite names four algorithms in one string

Example from the notes · IANA name `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`

ECDHE	ECDSA	AES128-GCM	SHA256
Key exchange	**Authentication**	**Bulk encryption**	**Hash for key derivation**
Ephemeral elliptic-curve Diffie-Hellman, so sessions get forward secrecy	Elliptic-curve signature made with the certificate's key	AES with a 128-bit key in Galois/Counter Mode: one AEAD step encrypts and authenticates	Feeds the PRF that derives keys; GCM already supplies the MAC

> **Older CBC suites** such as AES_128_CBC_SHA ended in a real HMAC algorithm; with AEAD suites the last field only names the PRF hash

RFC 5246 and RFC 5289; IANA TLS Cipher Suites registry

03

Cipher suites

TLS 1.3 cut the menu to five AEAD suites and dropped every legacy option

	TLS 1.2	TLS 1.3
Suite names	Key exchange, signature, cipher and hash	Cipher and hash only; key exchange chosen separately
Key exchange	RSA, static DH, DHE or ECDHE	Ephemeral DHE or ECDHE, or a PSK
Encryption	RC4, 3DES, AES-CBC, AES-GCM, ChaCha20	AES-GCM, AES-CCM or ChaCha20-Poly1305
Integrity	HMAC (MD5, SHA-1, SHA-2) or AEAD	AEAD only
Signatures	RSA PKCS#1 v1.5, DSA, ECDSA	RSA-PSS, ECDSA or EdDSA
Also removed	Compression, renegotiation, export ciphers	None of these exist

The three suites browsers use in TLS 1.3

`TLS_AES_128_GCM_SHA256` · `TLS_AES_256_GCM_SHA384` · `TLS_CHACHA20_POLY1305_SHA256`

The two AES-CCM suites for small IoT devices, which browsers don’t offer

`TLS_AES_128_CCM_SHA256` · `TLS_AES_128_CCM_8_SHA256`

Sources: RFC 8446 Sect. 9.1 and Appendix B.4; P. C. van Oorschot, Sect. 9.2

Attacks on TLS

04

TLS 1.3 is the result of 20 years of attacks, each removing a weak feature

What attackers still target is around the protocol: certificates, redirects and code

04

Attacks

Superfish put one root key on every Lenovo laptop, so anyone could impersonate any site

Adware preinstalled on Lenovo consumer laptops, 2014–2015



Why: to put shopping ads beside products, even on HTTPS pages, so it decrypted and re-signed every page. A bad idea from the start

1. **One private key on every laptop**
   The root CA's key shipped inside the software itself
2. **Protected by the password "komodia"**
   A researcher pulled it from memory and cracked it in a few hours
3. **Anyone on café Wi-Fi could sign for any site**
   Browsers trusted the fake certificates and showed no warning

Lesson: a root added on the device skips every CA safeguard; browsers don't require CT for local roots

Sources: R. Graham, Extracting the Superfish certificate, Errata Security, Feb. 2015; CERT VU#529496

TLS stripping attacks the HTTP-to-HTTPS redirect, not TLS itself



Swallows the redirect to https:// and rewrites secure links to http://, so the user never sees an error

Defenses: make the browser refuse plain HTTP before the first request

HSTS header

Strict-Transport-Security tells the browser to use only HTTPS for this site from now on

HSTS preload list

Domains compiled into the browser, so even the first visit is protected

HTTPS-First modes

Browsers now try HTTPS for every site; this replaced the retired HTTPS Everywhere extension

Sources: M. Marlinspike, sslstrip (Black Hat 2009); RFC 6797; EFF, HTTPS Everywhere retirement (2022)

04

Attacks

Most major TLS breaks exploited legacy options that TLS 1.3 removed

Attack	What it exploited	How TLS 1.3 closes it
BEAST · 2011	Predictable CBC initialization vectors in TLS 1.0	**AEAD ciphers only**
CRIME · 2012	Compression ratios that leak secret cookies	**Compression removed**
Lucky 13 · 2013	Timing of CBC padding checks	**AEAD ciphers only**
POODLE · 2014	SSL 3.0 padding, reached by forced version fallback	**SSL 3.0 gone; downgrade marker**
FREAK, Logjam · 2015	Servers still accepting 1990s export-grade keys	**Export ciphers removed**
ROBOT · 2017	RSA padding oracle (Bleichenbacher, 1998)	**RSA key transport removed**

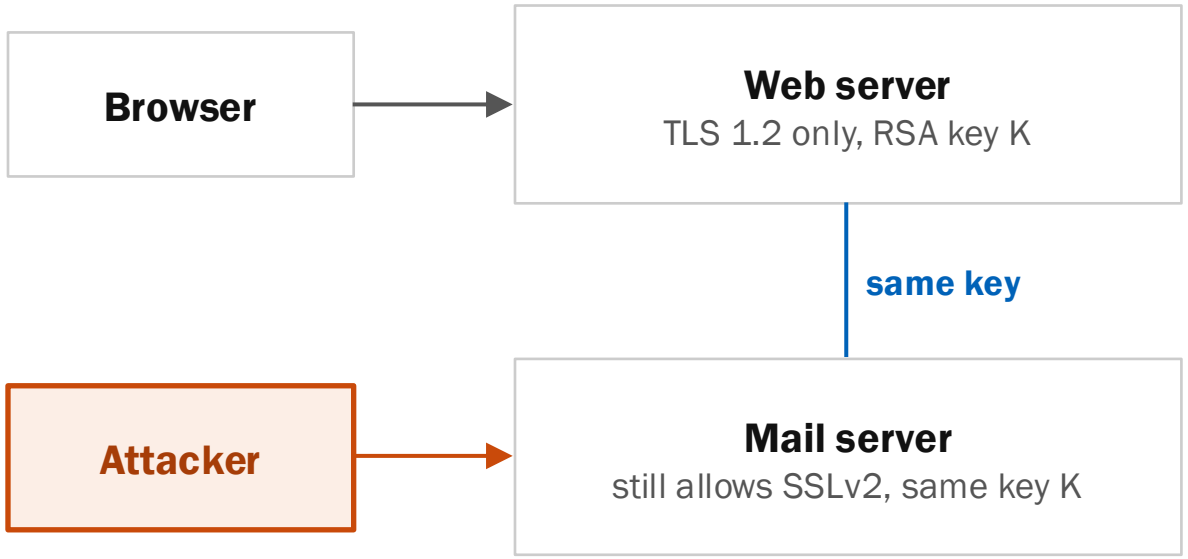
Two more need a closer look: DROWN broke TLS through an old SSLv2 server; BREACH works even on TLS 1.3

Sources: Acunetix, TLS Security 6: Examples of TLS Vulnerabilities and Attacks; RFC 7457; RFC 8446

04

Attacks

DROWN used an old SSLv2 server to decrypt TLS on every server sharing its RSA key



1. Attacker records a TLS session to the web server
2. Probes the SSLv2 server, which leaks RSA padding
3. Recovers the session key and reads the recording

33%

of HTTPS servers were vulnerable in March 2016, half through a shared key

Decrypting one session took under 8 hours and $440 on Amazon EC2

Lesson: an old protocol anywhere on the key breaks the new one. TLS 1.3 drops RSA key transport, so there is nothing to decrypt

Sources: N. Aviram et al., DROWN: Breaking TLS Using SSLv2, USENIX Security 2016; drownattack.com

04

Attacks

BREACH still works on TLS 1.3, because the leak is in HTTP compression

Black Hat USA 2013: the CRIME idea, moved up into HTTP

Page body, gzip-compressed, then encrypted	Encrypted size
`token=8f3a… q=token=7`	**1,204 bytes**
`token=8f3a… q=token=8`	**1,201 bytes** match
`token=8f3a… q=token=8f`	**1,198 bytes** match

Each match fixes one more character; repeat for the whole token (sizes illustrative)

Works when a page has all three

- HTTP compression (gzip or Brotli) on the response
- Attacker-chosen input reflected in the body
- A secret such as a CSRF token in the same body

> **Fix it in the app:** mask or rotate tokens per request and keep secrets away from reflected input

Sources: A. Prado, N. Harris, Y. Gluck, BREACH, Black Hat USA 2013; CVE-2013-3587

04

Attacks

Heartbleed leaked server memory through one missing length check

The heartbeat request in OpenSSL 1.0.1, disclosed April 2014

Honest client

Echo “bird”, length 4 → **bird**

Attacker

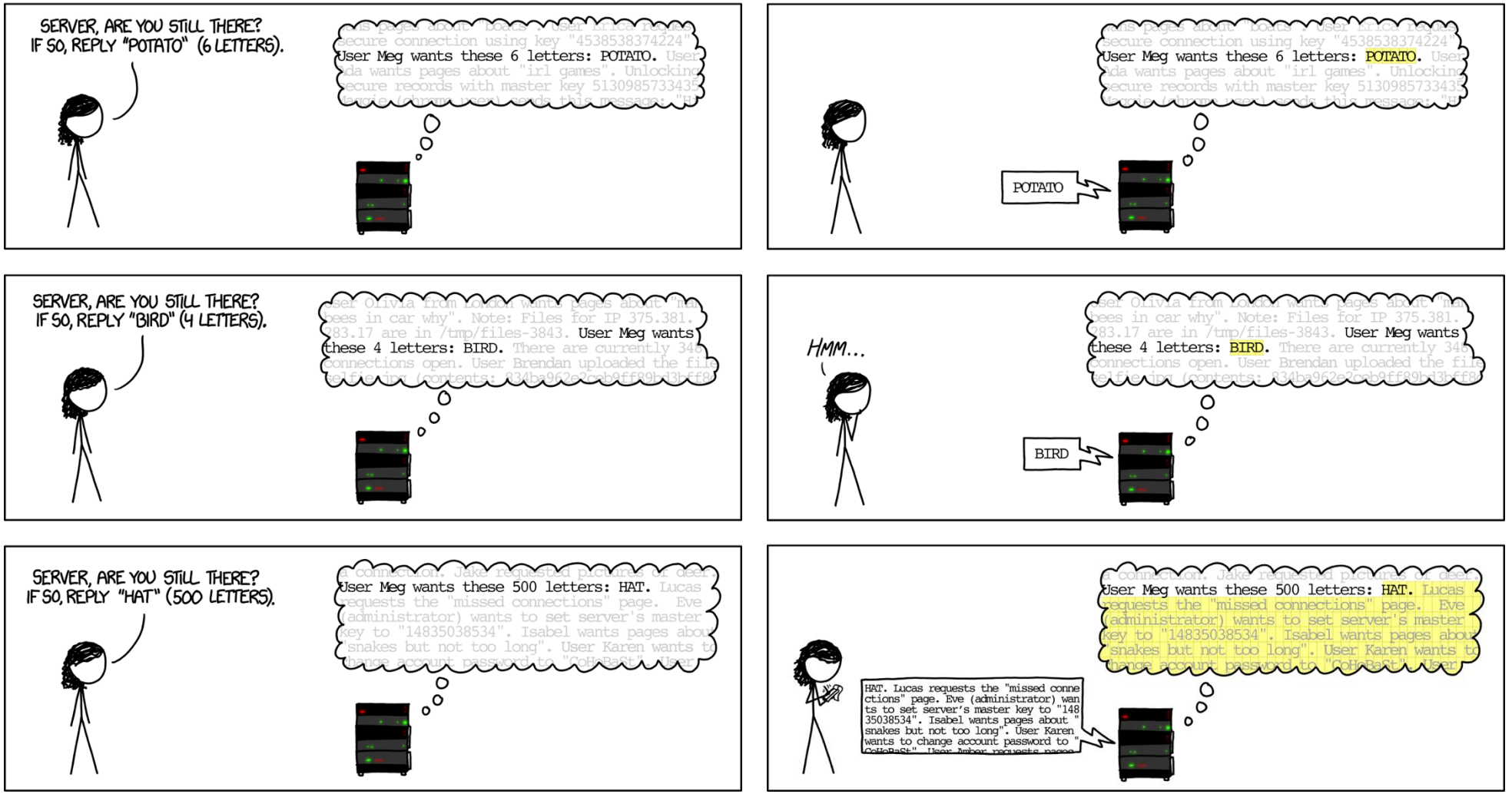
Echo “bird”, length 64 KB → **bird + 64 KB of server memory**
passwords, cookies, private keys

17% **of HTTPS servers with trusted certificates, about half a million, were vulnerable at disclosure**

Cleanup: patch, then revoke and reissue every certificate whose key might have leaked, which strained revocation systems

Sources: Heartbleed 101, IEEE Security & Privacy 2014; Netcraft, April 2014; xkcd 1354 explains it in six panels

Heartbleed even made it to xkcd



Source: R. Munroe, xkcd 1354, How the Heartbleed Bug Works, xkcd.com/1354 (CC BY-NC 2.5); panels rearranged into two columns

04

Attacks

Two small coding mistakes let forged signatures pass verification

goto fail

Apple iOS and macOS, fixed February 2014

```
if ((err = SSLHashSHA1.update(...)) != 0)
    goto fail;
    goto fail;  // err is 0 → success
```

The extra line skipped the signature check on the server's key exchange, so any key was accepted for about 17 months

Psychic Signatures

Java 15 to 18, fixed April 2022

```
ECDSA signature: r = 0, s = 0
verify(anyMessage, anyKey)  →  true
```

Java's rewritten ECDSA check never rejected zero values, so a blank signature verified for any message and any key

Lesson: the protocol was sound both times.
Tests must check that bad signatures fail, not just that good ones pass

Sources: D. A. Wheeler, The Apple goto fail vulnerability: lessons learned; CVE-2014-1266; N. Madden, Psychic Signatures in Java, Apr. 2022; CVE-2022-21449

TLS in practice today

05

05

TLS today

CDNs still hold the keys for much of the web, but no longer the site's own long-lived key



Whoever holds the edge key can pose as the site and read its traffic

Problem found (2014–2016)	Current practice (2026)
Sites uploaded their own private key to the CDN	The CDN gets its own certificate via ACME and holds that key. **Keyless SSL** and **delegated credentials** avoid this but are rare
Cruise-liner certificates: one certificate and key shared by many customers	SNI everywhere, so each site gets its own unshared certificate
Slow revocation: after Heartbleed, 66% of hosts managing customers' certificates revoked none of them	Automated 90-day renewal now, 47-day cap by 2029; backup certificates on a separate key
Unchecked CDN-to-site link: 2 of 5 CDNs used plain HTTP; 3 skipped certificate checks (2014)	Still optional: sites must turn on certificate checks; major CDNs skip revocation checks (2022)
Concentration: 10 providers held keys for 45.3% of domains	Unchanged: a few CDNs still serve a large share of the web

Sources: Cangialosi et al., CCS 2016; Liang et al., IEEE S&P 2014; CDNs' Dark Side, ACM 2022 (doi 10.1145/3499428); Cloudflare docs; RFC 9345; CA/B Forum SC-081v3

05

TLS today

SNI is universal; Keyless SSL and delegated credentials remain niche

SNI

Fixes cruise-liner certificates

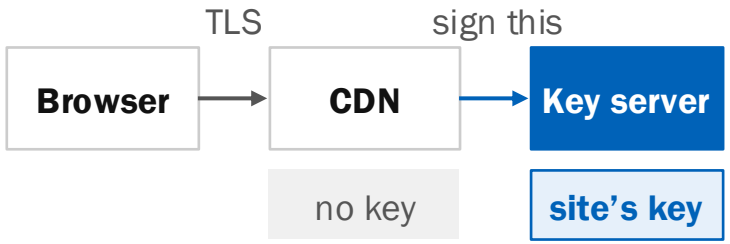


The browser names the site in its first message, so one CDN address can pick the right certificate

Universal: about 99.8% of requests send it (Akamai, 2018)

Keyless SSL

Keeps the key with the site

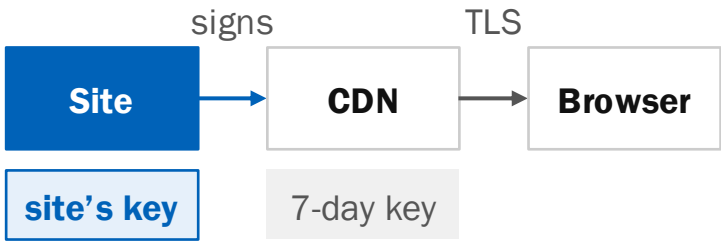


The CDN runs the handshake but asks the site's key server for the one signature it needs

Niche: an Enterprise add-on, mostly for banks, with no TLS 1.3 support

Delegated credentials

Limits a stolen key to 7 days



The site's key signs a short-lived key for the CDN; the browser checks both signatures

Niche: only Cloudflare, Facebook and Firefox support it

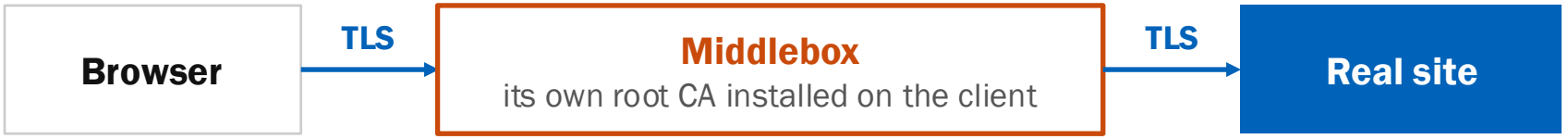
Sources: Akamai, TLS SNI adoption, IETF 101 MAPRG, 2018; Cloudflare Keyless SSL docs and Geo Key Manager blog; RFC 9345; E. Thompson et al., Fine-grained CDN Delegation, 2025

05

TLS today

TLS interception is widespread and often weakens the connection it inspects

Interception middleboxes corporate proxies and antivirus software



Many re-encrypt with weaker TLS than the browser uses

1 in 250

TLS connections proxied, mostly by firewalls and antivirus

O'Neill et al., IMC 2016

5–10%

of HTTPS connections intercepted, seen from three large services

Durumeric et al., NDSS 2017

58%

of middlebox connections had flaws that allow later interception

Durumeric et al., NDSS 2017

Practice in 2026

Cloud-based: SSE providers inspect using a company root CA on every device

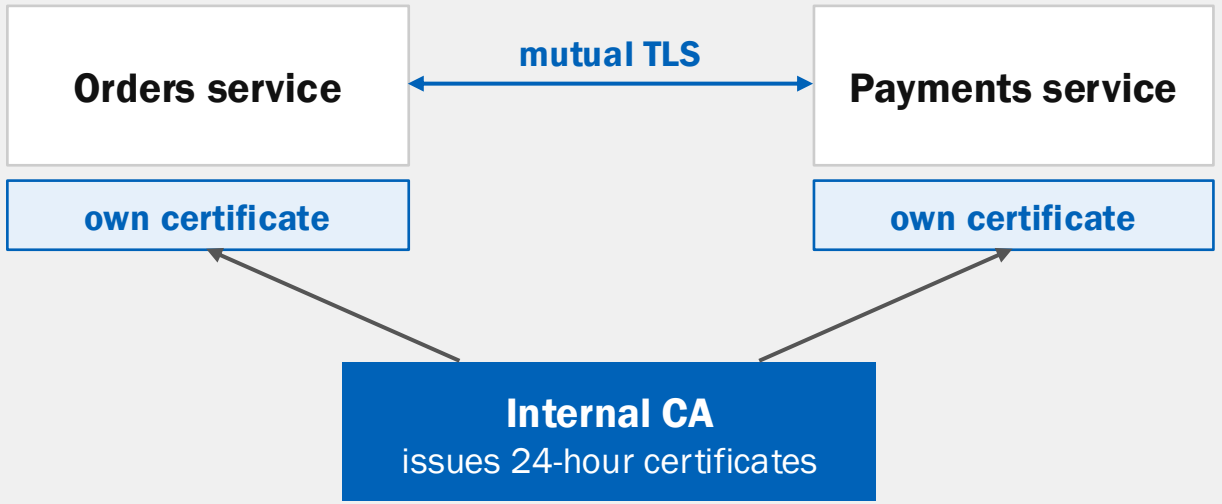
Partial: vendors report 60–70% coverage, and pinned apps are skipped

Optional: CISA says inspection is no longer universally recommended

Sources: O'Neill et al., TLS Proxies: Friend or Foe?, IMC 2016; Durumeric et al., NDSS 2017; Zscaler TLS inspection guide; CISA SASE TIC 3.0 guidance, June 2026

Inside companies, mutual TLS makes both services prove who they are

Inside the company network, no connection is trusted by default



Browsers rarely use it: web users log in with passwords or passkeys, not client certificates

The handshake already allows it

The server sends a CertificateRequest and the client answers with its own certificate

Zero trust

Every service authenticates every call, so an attacker inside the network can’t impersonate one

Automated, short-lived certificates

Service meshes like Istio issue and rotate them with no human involved

Sources: RFC 8446 §4.3.2 (CertificateRequest); NIST SP 800-207, Zero Trust Architecture, 2020; Istio security docs (workload certificates, 24-hour default)

05

TLS today

Chrome now tries HTTPS first and warns before any public HTTP site

2015	2018	2020	2022	Oct 2026
30–45%	**Not secure**	**95–99%**	**Opt-in**	**Default**
of Chrome page loads use HTTPS	label on every HTTP page	then progress stalls	HTTPS-first setting added	Chrome 154 turns it on for all

What the browser does now

- Tries https:// before http:// for every site
- Warns on the first visit to a public HTTP site
- Skips sites you visit often and private addresses

Why it matters for TLS

TLS stripping needs that first plain HTTP request; now the browser skips it or warns, even for sites without HSTS

What made this possible: free, automated certificates from Let’s Encrypt let almost every site switch

Sources: Google Security Blog, HTTPS by default, Oct 28 2025; Google HTTPS Transparency Report; Chromium blog, A secure web is here to stay, Feb 2018

05

TLS today

In 2026, certificates are shorter-lived, keys are post-quantum and site names are hidden

Certificate lifetimes

200 days

Maximum since March 2026, down from 398; 100 days in 2027 and 47 in 2029, so renewal must be automated

Post-quantum keys

ML-KEM

Hybrid X25519MLKEM768 is the browser default, blocking harvest-now, decrypt-later; certificates still use classical signatures

Encrypted Client Hello

RFC 9849

Published March 2026; encrypts the site name (SNI), the last plaintext in the handshake. Cloudflare leads deployment

TLS beyond TCP

HTTP/3

QUIC merges the TLS 1.3 handshake with connection setup, so the first HTTP request goes out a round trip sooner than over TCP

Look for these in the exercise: the protocol version, the key-exchange group and the certificate's validity period

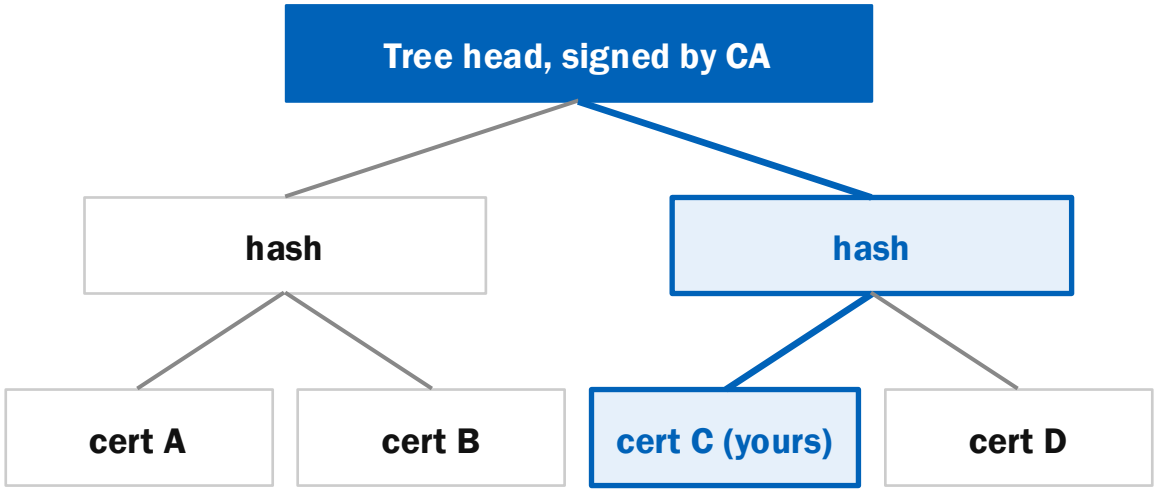
Sources: CA/Browser Forum Ballot SC-081v3 (2025); Cloudflare Radar and Encryption Consulting, 2026; RFC 9848, RFC 9849 (March 2026); RFC 9000, RFC 9001

05

TLS today

Post-quantum certificates will be short proofs in a Merkle tree, not bigger signatures

The CA adds each certificate to a public tree and signs only the root



The server sends your certificate plus the **few hashes needed to rebuild the blue path**; the browser checks them against a tree head it already trusts

Authentication data per handshake

14,700 bytes

if today's certificates used post-quantum signatures

736 bytes

with a Merkle Tree Certificate

CT is built in: a certificate cannot exist outside the public tree

Feb 2026
Chrome picks MTCs for post-quantum certificates

2026
Chrome and Cloudflare test MTCs on real traffic

2027
Let's Encrypt and Cloudflare plan production issuance

Sources: Google/Chrome MTC announcement, Feb 2026; Let's Encrypt, A Post-Quantum Future for Let's Encrypt, Jun 2026; Cloudflare CA announcement, Sep 29 2026; IETF PLANTS working group

06

Exercise

Class exercise: grade a real server with SSL Labs

- **Test:** ssllabs.com/ssltest on www2.pnwx.com
- **Versions:** which does it accept, and which should it drop?
- **Cipher suites:** which lack forward secrecy or still use CBC?
- **Certificate:** who issued it, and is it within the 200-day limit?
- **Fix:** what would you change to earn an A+?