§ TLS 1.3 SESSION RESUMPTION · AN UNDEFINED REQUIREMENT, AND WHAT IT IS ALREADY COSTING

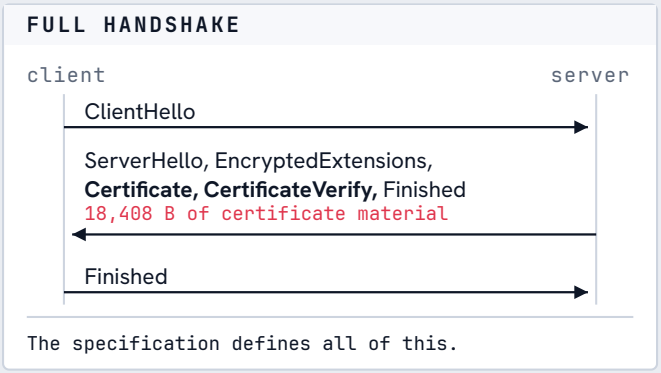
TLS 1.3 never tells the server what a resumed connection must re-check.

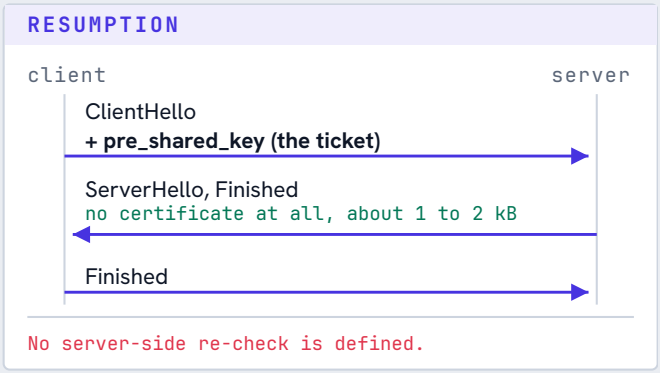
Eleven identifiers since 2014, and resumption is off under mutual TLS at four front doors, exactly when post-quantum makes that the most expensive state to be in.

01 The gap: what a resumption skips, and what nothing requires it to re-check

TLS 1.3 establishes a connection in one of two ways. A **full handshake** sends the certificate chain and validates it: expiry, trust anchors, revocation, and whether the peer is authorised for what it asked for. A **resumption** sends none of that. The client presents a ticket from an earlier session and both sides carry the earlier result forward, which is precisely why it is cheap. A full handshake gives an endpoint the material and the moment to decide all of that. **A resumption gives it neither, and nothing in TLS says what it must re-establish instead.**

The server is not idle at resumption. RFC 9846 requires it to validate the PSK binder, to select a cipher suite whose KDF hash matches the original connection, and, for early data, to check that the version, cipher suite and ALPN are unchanged. Every one of those checks the *ticket* against the session that issued it. Not one re-checks the authentication verdict that session carried, which is what this paper is about.





02 What that costs, classically and post-quantum

PROFILE	CERTIFICATE MATERIAL	SERVER'S FIRST FLIGHT	VS 14,600 B INITIAL CONGESTION WINDOW
Classical ECDSA P-256, web	3,465 B	3,763 B	fits, one round trip
Post-quantum ML-DSA, web, two SCTs	**18,408 B**	**19,762 B**	**exceeds by 5,162 B: two round trips**
Post-quantum ML-DSA-44, in-mesh mutual TLS	11,044 B per direction	12,398 B	fits, one round trip
Resumption, any of the above	**No certificate is sent at all. About 1 to 2 kB, always fits, always one round trip.**		

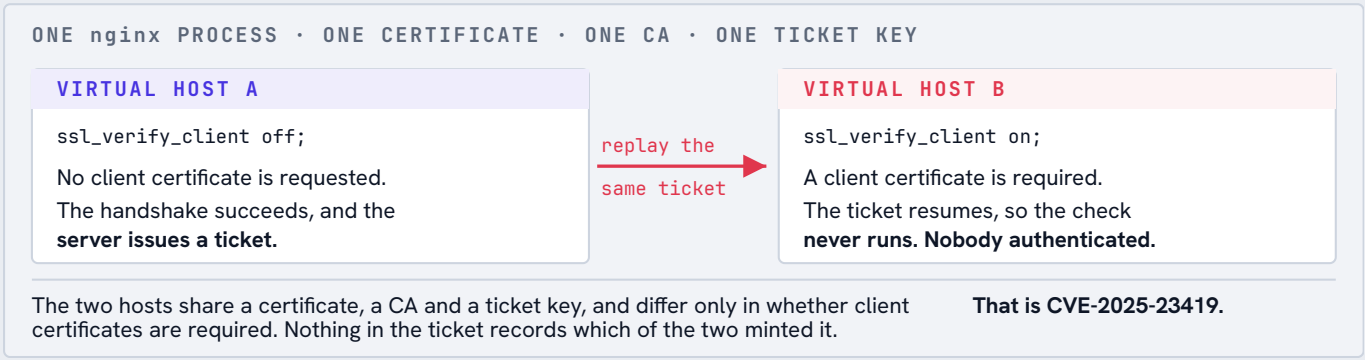
Classically, resuming saves about 3.5 kB and nobody agonises over it. Post-quantum it saves 18 kB and a round trip. The same optimisation stops being a nicety and becomes the thing that decides whether post-quantum authentication is affordable at all.

This is not a forecast. On 27 August 2026 eleven of fourteen public endpoints we measured already selected X25519MLKEM768 when it was offered alongside the classical groups, among them AWS, Google Cloud, Azure, Cloudflare and Fastly; the same eleven on a second run the following day. `ietf.org` was one: post-quantum key exchange on, resumption on, and the specification its own publisher maintains does not say what that resumption must re-check. The key exchange has migrated; authentication, which carries the bytes above, has not. Panel and per-host records in supporting document 3. Key-exchange and resumption state is measured daily over a larger public panel by the open-source worker quantakrypto/pqc-observatory.

Constants from FIPS 203 and 204; X.509 encoding measured against live letsencrypt.org and cloudflare.com chains; the classical chain measured against ietf.org on 25 August 2026, which sits inside a range of 2,465 to 5,785 B, median 3,650.5, measured across fourteen public endpoints on 27 August. Every row counts its CertificateVerify as certificate material and adds the same framing, which differs between the rows only in the key share it carries: 1,088 B of ML-KEM-768 ciphertext post-quantum against 32 B of X25519 classically. A front-door mutual TLS handshake is the web row's server flight plus a client chain in the other direction, so the four parties in section 5 sit *above* the web row, not on the in-mesh row. Every constant, its source and the sum per cell are in **supporting document 3**, the byte model.

03 What goes wrong: scope, not staleness

The obvious worry is staleness: a session outliving the certificate that authorised it. That case is real, measurable, and the weakest one. A long-lived connection is equally stale, and where operators care they bound its age rather than its resumptions. **The consequential failure is scope.** A resumption carries the authentication verdict, not just the keys, into contexts where that verdict was never valid.



> The failure is scope, not staleness. Nothing here has expired: the certificate is valid, the CA is valid, the ticket is fresh. An authentication verdict simply moved somewhere it was never valid.

Eleven identifiers, eleven years and five months, eight products. Apache, Envoy and stunnel each shipped their own version of the same thing. In five of them a party outside the current trust scope gains standing access needing no network position, and in all five the replayed verdict is a client-certificate verdict. One is literally a revocation bypass. The table itself, the inclusion test that decides what counts, and the provenance for every row are in **supporting document 1**, the vulnerability corpus.

04 How much of the internet this touches

94.8%

of reachable TLS domains issue a session ticket, so resumption is offered almost everywhere it could be.

1,108,322 of the 1,169,527 domains that completed a TLS handshake, themselves 79.9% of the 1,463,917 with :443 open, from the Tranco top 1M expanded to 2,063,760 candidates. Dec 2024 to Jan 2025, Hebrok et al., Table 3 and section 5.1.1, whose scan probed for tickets by sending an HTTP request. Corroborated on that same condition: of 14 public endpoints measured for this paper, 14 issued a ticket after one HEAD and 8 on a silent connection, 27 to 28 Aug 2026. The condition is load-bearing: measure without a request and the figure is not reproducible.

10 to 94%

of connections complete by resumption, depending on vantage. Even the low end, against the saving in section 2, is not a corner case.

Holz et al. 9.7% of TLS 1.3 connections, passive, 2019; Cloudflare 27% of QUIC connections in 2025, and 25% mobile to 70% desktop in its post-quantum experiment; Chou and Cao 94.2% CDN-terminated against 46.1% not. Rates vary by vantage and must not be averaged: supporting document 1, section 6.

24.9%

of all websites terminate TLS at Cloudflare, which disabled resumption under mutual TLS across that entire network.

W3Techs share, August 2026. The switch was thrown network-wide on 24 January 2025 and Cloudflare has published no re-enablement since, only that it is "exploring ways to bring back" the performance.

The first two bound how much is at stake. Resumption is offered nearly everywhere it can be, and at every vantage that has published a measurement it carries a substantial share of connections, from a tenth measured passively in 2019 to nine in ten behind a CDN, so the per-connection saving in section 2 is not a corner case at any of them. The third bounds only the reach of a single operator's switch, not the size of the problem: the population that has *lost* that saving is the one using client certificates, which is service meshes, API gateways, business-to-business APIs, mobile backends and zero-trust deployments, and that population has never been measured globally. What can be said precisely is who stopped, and on what basis each of them acted, which is section 5.

Prevalence, bounded. Exploitation is not what is expensive here. The internet-scale scan that found this class manually reviewed 4,370 suspicious resumptions and classified **176** as vulnerable, mostly clustered behind two DDoS-mitigation providers. None of the eleven identifiers appeared in CISA's KEV catalogue as of 26 August 2026. Issuing a ticket is also not the same as honouring one: of the fourteen endpoints measured for this paper, github.com issued a ticket and then completed a full

handshake when it was offered back to the same address and the same SNI, in three runs across two days and in a second, independent implementation. It is still a plain connection rather than a client-authenticated one, so it is not a fifth party in section 5, and one endpoint that declines is not a rate. The client-authentication half cannot be scanned at that scale at all, because it needs credentials, so its prevalence is unknown rather than low. The cost being paid today is the retreat, not the breach.

05 Who has already turned it off, and on whose authority

Faced with a requirement the specification does not state, operators land on the answer that is safe without further analysis: do not resume under client authentication. Two chose it under disclosure; two were already there. The table's third column says which is which, and this paper claims nothing beyond it. As last measured or documented, **four parties do not resume** under mutual TLS, and where the switch was thrown it was thrown network-wide rather than per customer.

PARTY	STATE	WHAT IT DOES UNDER CLIENT AUTHENTICATION	HOW THIS IS KNOWN	WHEN
Cloudflare	OFF	"We have disabled TLS session resumption for all customers that have mTLS enabled." Described by the reporters as a preliminary fix.	Vendor incident post, after a HackerOne report from the Paderborn group	23 to 24 Jan 2025
Caddy	OFF	"Caddy has disabled session tickets when using client authentication."	The reporters' wording, not Caddy's, describing a vendor change under the same disclosure	2025
Google Cloud Load Balancer	OFF	"[W]e found that session tickets are disabled when enabling client authentication."	Measured, not announced: Hebrok et al. section 5.2	reported 2025
AWS Application Load Balancer	OFF	Documented as not resuming under mutual authentication.	Vendor documentation. Explicitly *not* tested by Hebrok et al., section 5.3	documented
curl	OFF 2016–17	Disabled session reuse outright whenever a client certificate was set, across fourteen months, less a regression it had to fix midway.	Commits 247d890da8 (7.50.1) to 9d3dde37a8 (7.56.0)	Aug 2016 to Oct 2017
Azure Application Gateway	ON	**Keeps resumption on** under client authentication, though the researchers could not resume across two configured gateways, so nothing was shown exposed. The same question, the opposite default.	Measured: Hebrok et al. section 5.2	reported 2025

The last row is the point. **Three** cloud front doors were measured in the same study and gave **three** different behaviours: one had resumption off, one kept it on, and one kept it on and was bypassed. The bypassed one switched off under disclosure, Caddy followed the same disclosure, and AWS documents the off state. curl took the same exit a decade earlier. That is not a bug in any of them. There is no answer in TLS itself to converge on.

06 Why this needs a specification and not six more patches

This was once written down, on the server. RFC 6066 section 3, TLS 1.2: *"A server that implements this extension MUST NOT accept the request to resume the session if the server_name extension contains a different name."* TLS 1.3 removed it. RFC 9846 section 4.3.11: *"there is no need for the server to associate an SNI value with the ticket."* The scope requirement is now written once, normatively, and written for the client. RFC 9846 section 4.7.1: *"Clients MUST only resume if the new SNI*

The result is exactly what an unassigned requirement produces. **Seven parties responded**, not to one disclosure and not in one year, by keeping resumption and adding a check. **Six implementations invented** six different answers between them, some binding time and some binding scope, with a seventh discharging at the HTTP layer instead. Four more turned the feature off. One never bounded anything.

value is valid for the server certificate presented in the original session". On time the RFC is normative but unquantified: it is *RECOMMENDED* that implementations bound the total lifetime of resumption keying material against the peer certificate's lifetime and the likelihood of intervening revocation. On scope, server-side, there is nothing.

Every 2025 entry in this corpus is a server-side failure. A server that resumes a session into a virtual host requiring client certificates has violated no TLS-layer requirement, because nothing was asked of it.

One vendor has now done this twice, which is the argument in one history. In January 2023 Cloudflare found that revoked client certificates were being honoured on resumption. It disabled resumption for mTLS, then built a real fix, checking revocation status by serial and authority key identifier at resumption, and re-enabled the feature within weeks. That fix was correct and it was specific: it bound revocation. Two years later CVE-2025-23419 was a *scope* failure, one zone's certificate resumed at another, which the 2023 fix did not address and was never meant to. The switch went off again, network-wide, and has stayed off. A vendor that solves this class correctly still has no way to know what else the class contains, because nothing defines it. That is the difference between a patch and a specification.

One profile does ask, and it is the proof this can be written. RFC 9190, EAP-TLS, section 5.7: *"Any security policies for authorization MUST be followed also for resumption"*, and decisions taken on information that has since changed *"MUST be reevaluated."* Section 8 generalises that sentence from one profile to TLS.

07 The six answers, and the seventh

Nobody had a server-side requirement to conform to, so each of these is an invention. They fall into two families, time bounds and scope bindings, and inside the scope family the same mechanism was invented twice: Envoy in February 2022 and HAProxy in August 2026, independently, citing neither each other nor any specification. They still hash different inputs, so they cannot resume each other's sessions.

WHO	WHAT IT BOUND
nginx	the session age to the chain in 2022, and separately a scope fix after CVE-2025-23419
BoringSSL	a 7-day non-renewable authentication timeout, plus session id context partitioning
Go crypto/tls	leaf expiry, then full-chain expiry, then root-in-pool, across three releases
HAProxy	a session-lifetime clamp, plus verify mode, every CA, and a CRL *identifier* rather than its contents, hashed into the context
Envoy	the whole trust configuration digested into the session id context, in 2022
Fastly	the ticket to the issuing certificate
Apache httpd	**nothing in the session.** It resumes, then refuses the request with HTTP 421: the seventh party, discharging at a different layer rather than inventing a seventh semantics

Turn section 8's rule on this table and it adjudicates at a glance, uncomfortably. The time bounds satisfy no discharge; they only bound how long a stale verdict survives. Go fully discharges the temporal condition only; its root check re-establishes anchor membership, but nothing of policy or revocation. Fastly binds the ticket to the issuing certificate, which distinguishes nothing in CVE-2025-23419, where both hosts share one. Even Envoy, the strongest entry, leaves `notAfter` open. None of the six fully conforms. That is not an indictment of the implementers. It is what an unwritten requirement produces, and it is the argument.

Nothing in any specification says which of them conform, and two are not even internally settled: HAProxy carries three of these in a development snapshot, selected by the TLS library it is linked against, and Go ships two selected by `runtime.GOOS`. That is the cost being described. Not six bugs, six private answers to a question the standard declined to ask. Each is set out with its commits, and its qualifications, in **supporting document 2**, the implementation survey.

A mixed fleet sharing one ticket key therefore stops resuming across tiers, silently, and nothing alarms on it by default. Under post-quantum that silent degradation costs a full chain, and on a web-sized chain an extra round trip per connection.

08 The fix, and the four ways to satisfy it

> **An endpoint MUST NOT rely on a replayed authentication verdict for any decision that a full authentication, performed when the verdict is first relied upon, would decide differently.**

Completing a resumed handshake is transport, not reliance. Reliance begins at the first decision the verdict authorises, and from there the connection stands on the same footing as one that handshook in full. That is deliberate: the governing principle is parity, and a resumed connection must not be held to a stricter standard than a fresh one.

It binds the decision, not the handshake path, so there are four conforming ways to satisfy it and only one of them is expensive:

Re-establish the condition, where the cached session holds the material for it: one chain validation, tens to hundreds of microseconds, and no extra bytes where the session state is server-side; a stateless deployment that carries the chain in the ticket pays it on the client's flight instead. Where it does not, as when a ticket minted at a context that never requested a client certificate is offered at one that requires it, there is nothing to re-establish and the ticket must be declined. A ticket offered back at the context that minted it is untouched: where no verdict is being replayed across a boundary, the rule does not bite.

Refuse to resume. The state the four parties above are in, and what this paper asks the IETF to stop making anyone settle for.

Defer the decision. What Apache does, returning 421 after resuming.

Over-approximate. Bind the trust configuration and the authentication policy into the resumption scope, so any change forces a full handshake, **paired** with a lifetime cap at the status snapshot's remaining validity. Without the pairing a digest changes when a snapshot is replaced but not when it merely expires. Envoy has shipped the digest half since 2022 and HAProxy since a development snapshot in August 2026. Neither ships the pairing, which is why section 7 finds that none of the six fully conforms: the mechanism is proven, the completion is not.

Either direction. The rule binds an endpoint relying on a replayed verdict, which is the server holding a client-certificate verdict in every 2025 entry here, and equally the client holding a server-identity verdict, which is the half that RFC 9846 already binds and that was measured at scale. The conditions a verdict depends on are validity across the whole cached path, scope meaning identity and authentication policy, and continued acceptance by the current trust configuration. The full normative text, the amortisation boundary, and what it deliberately leaves open are in **supporting document 2**.

Bind only the host, as RFC 9846's client-side rule does, and CVE-2025-23419 conforms to the rule. That is why policy, not just identity, has to be in scope.

The condition idea is Hebrok et al.'s, who also proposed binding "the rules that the certificate passed initially" and rejected SNI-only binding; the lifetime idea is Chuat et al.'s. The contribution here is specification: a rule an implementation can conform to, and answers to the two questions that arise the moment it is written down. **Remaining limits:** no global measurement of the mutual-TLS population exists, so the affected share is large but unsized; and if Merkle Tree Certificates or trust-anchor negotiation shrink post-quantum chains, the per-connection saving shrinks with them.