§ RFC 9846 · THE GAP, THE SURVEY, AND THE PROPOSED TEXT

Why this needs a definition, and the text

Supporting document for *"TLS 1.3 never tells the server what a resumed connection must re-check"*, IAB Workshop on Accelerating the Deployment of Post-Quantum Authentication. León Acosta, QuantaKrypto. v1, 28 August 2026.
What the specification says and does not say, what seven implementations did instead, the prior art, and the proposed normative text with the provenance of each part.

1. What the specification requires

It is often said that the specification recommends but never requires and never defines. That is true of the time bound and false of scope, and the distinction sharpens the argument.

1.1 The time bound: RECOMMENDED and undefined

RFC 9846, Standards Track, July 2026, obsoletes RFC 8446 and retains the text verbatim at section 4.7.1:

> *"Note that in principle it is possible to continue issuing new tickets which indefinitely extend the lifetime of the keying material originally derived from an initial non-PSK handshake... It is RECOMMENDED that implementations place limits on the total lifetime of such keying material; these limits should take into account the lifetime of the peer's certificate, the likelihood of intervening revocation, and the time since the peer's online CertificateVerify signature."*

1.2 Scope: a normative MUST, but only on the client

RFC 9846 section 4.3.11 records that this was a deliberate move. Before TLS 1.3 the server was "required to enforce that the SNI value associated with the session matches the one specified in the resumption handshake"; in TLS 1.3 "there is no need for the server to associate an SNI value with the ticket". The one normative certificate-scope rule at resumption now sits on the client, by design, and every 2025 entry in the corpus is a server-side failure.

Then 4.7.1, the ticket definition:

> *"Clients MUST only resume if the new SNI value is valid for the server certificate presented in the original session, and SHOULD only resume if the SNI value matches the one used in the original session."*

So the specification does know how to write this requirement. It wrote it once, normatively, for one side. Hebrok et al. make the point directly: the requirement is "explicitly placed on the client", and "the standard completely relieves the server from any responsibility in validating session resumption". **Every entry from the 2025 disclosure cluster is a server-side failure.**

1.3 RFC 9190 section 5.7 is the strongest precedent, and this paper undersold it

RFC 9190 is usually cited only for its seven-day storage limit. It contains far more:

> *"Authorization during resumption MUST be based on such cached data. The EAP-TLS peer and EAP-TLS server MAY perform fresh revocation checks on the cached certificate data.* ***Any security policies for authorization MUST be followed also for resumption.*** *The certificates may have been revoked since the initial full handshake and the authorizations of the other party may have been reduced."*

That is item 2 of this paper's proposal, already written as normative text in a published RFC, for one profile. Note also that fresh revocation checking there is only MAY, so a notAfter floor would be new even relative to EAP-TLS.

The same section goes further, and this is the sentence section 4 generalises. RFC 9190 names the failure mode and then states the obligation: *"This change creates a 'time-of-check time-of-use' (TOCTOU) security vulnerability. A malicious or compromised*

user could supply one set of data during the initial authentication, and a different set of data during resumption, potentially allowing them to obtain access that they should not have. If any authorization, accounting, or policy decisions were made with information that has changed between the initial full handshake and resumption, and if change may lead to a different decision, such decisions MUST be reevaluated." That is a rule on the *decision* rather than on the handshake path, written for one profile, and it is the closest thing in any published RFC to the text this paper asks for.

2. Implementation survey, split into the two things that differ

An undefined requirement does not produce one wrong answer. It produces a different answer per implementation, and in two cases more than one answer inside a single implementation: HAProxy ships three, selected by the TLS library it is linked against, and Go ships two, selected by `runtime.GOOS`. Seven parties responded, and the stated figure is **six settled semantics**; four more parties stopped resuming under client authentication rather than define one; and one never bounded anything at all. Twelve parties, three disjoint groups.

Why not nine. Counting HAProxy's three library builds and Go's two `runtime.GOOS` branches as separate semantics would give nine, and that number does not survive a reader who checks it: HAProxy's three exist only in a development snapshot and no stable release, and Go's two exist because a root-membership check regressed on macOS and are documented as temporary pending `CertPool.Contains`. They are churn, and churn is evidence for the argument, but counting transient states as durable divergence is advocacy. Six is the number of distinct semantics that survives, and it is a count of session bindings; the seventh responder, Apache httpd, enforced its condition at the HTTP layer instead and is counted as a party rather than as a seventh semantics. The three extra states are described in 2.2 where they belong.

WHO	SESSION-AGE CLAMP	RE-ESTABLISHED CONDITION	DETAIL
nginx	yes, 12 Oct 2022	not at the time	1.23.2 clamps to `ssl_session_timeout`, resets creation time and subtracts elapsed, so the budget shrinks along the chain; CVE-2025-23419 for scope followed in Feb 2025
BoringSSL	yes	re-verification: clients only. sid_ctx partitioning: servers	`auth_timeout`, 7 days, non-renewable; `reverify_on_resume` is documented "only respected on clients". The server-side condition mechanism is the session id context, which is what Envoy drives below and what Cloudflare misused
Go crypto/tls	no	staged	leaf expiry from Go 1.21 (2023); full-chain expiry in 1.25.6 and 1.24.12 on 15 Jan 2026; root-in-pool in 1.25.7 and 1.24.13 on 4 Feb 2026, which reverted the `Config.Clone` change because "this fix is broader". No revocation, policy or hostname check; `VerifyPeerCertificate` still does not run on resumption
HAProxy, Aug 2026	depends on the library it is built against		a session-lifetime clamp, not a certificate check: nothing in it reads `notAfter`. OpenSSL: 9f1d5909, whose message states "This fix does not carry over to BoringSSL/AWS-LC". BoringSSL and AWS-LC: 819b73ce, a different bug where `tune.ssl.lifetime` had no effect at all on TLS 1.3, now min(lifetime, 7 days), the 7 days being a pre-existing library ceiling rather than something this commit imposes. wolfSSL: 379a0715 skips the clamp; **wolfSSL's own behaviour is not established by these commits** (see 2.2). Companion commits isolate resumption per X509 certificate (5d5f1237, 14 Aug), per crt-list filter (52bdd0d0) and per authentication policy (4a8f7622), plus a build fix on 25 Aug (6ab73c3d)
Envoy (Istio, Consul)	no	extensive	hashes into the session id context: server certificate CN and SANs, issuer name, CA certificate, pinned hashes and SPKIs, trusted-CA flag, SAN matchers, CRL contents, allow-expired, trust-chain-verification mode, max verify depth, and the filter chain's SNIs. A session minted under one trust configuration will not resume under another
Fastly	-	yes	binds the ticket to the issuing certificate; reported separately in September 2023
Apache httpd	no	fixed in 2.4.64	condition fix for CVE-2025-23048; no chain time bound found in mod_ssl
OpenSSL	no	no	resets session time on every ticket; issue 19341 open, declined as a vulnerability

Caddy, Cloudflare	stopped resuming under client authentication	Cloudflare's is described by Hebrok et al. as a preliminary fix. AWS Application Load Balancer is documented that way by AWS, and Hebrok et al. excluded it from measurement for that reason; Google Cloud Load Balancer they measured themselves

2.1 The row that contains the whole argument

nginx shipped a chain-time budget in October 2022 and still received CVE-2025-23419 in February 2025 for a scope failure. The time bound was in place, correctly implemented, and the condition bug went straight through it. That is why the proposal in section 4 does not rest on a time bound: nginx already had one, and item 2's scope condition is what would have excluded the 2025 bypass.

2.2 The row that is sharper still

HAProxy has three answers in one release. First, what the fix is *not*: it clamps `tune.ssl.lifetime` across TLS 1.3 ticket renewals, and nothing in it reads the certificate's `notAfter`. It is a time bound, not a condition bound, which is exactly this paper's split. The certificate appears only as motivation, and the commit says the thing this paper is arguing: a client resuming just before each window closes "can keep a session alive indefinitely, well past its original full handshake's deadline and past any point where a client certificate would need to be re-verified for expiration or revocation".

Then the three answers, one per library. On **OpenSSL** the clamp works (9f1d5909), and its message states that it "does not carry over to BoringSSL/AWS-LC". On **BoringSSL and AWS-LC** the problem was worse than a missing clamp: `SSL_CTX_set_timeout` covers only TLS 1.2 there, so `tune.ssl.lifetime` had **silently had no effect at all** on TLS 1.3 sessions, which instead used those libraries' 2-day PSK/DHE default. 819b73ce calls the missing `SSL_CTX_set_session_psk_dhe_timeout`, making the effective cap min(lifetime, 7 days), the 7 days being a pre-existing library authentication timeout with no public API to override. On **wolfSSL**, 379a0715 skips the clamp entirely, for a build failure plus a hedged "doesn't appear to behave correctly against wolfSSL's TLS1.3 session/ticket handling". **Note what the source does not say:** it does not say wolfSSL is left with no bound. `SSL_CTX_set_timeout` is still called on every build, and wolfSSL's own behaviour is simply not established by these commits.

The companion commits are the more interesting half. 4a8f7622 hashes the verify mode, every CA certificate and every CRL in the `X509_STORE`, and the ignore-error bitfields into `sid_ctx`. That is Envoy's mechanism, reached independently: a CRL change alters the context, every existing session fails the comparison, and the next connection is a full handshake. Two qualifications matter. HAProxy hashes an *identifier*, "issuer and CRL Number extension", not the CRL contents, so a CRL reissued with the same number and different contents leaves the context unchanged. A CRL carrying no CRL Number extension at all also never distinguishes anything, though that case is narrow: RFC 5280 section 5.2.3 says conforming CRL issuers "MUST include this extension in all CRLs". It is non-critical, so a non-conforming CRL is still accepted and then indistinguishable, but the gap is a conformance failure rather than something the profile permits. Envoy digests the raw bytes and so fails closed. The asymmetry reverses on CAs, where Envoy hashes only the first certificate in the file and HAProxy walks the whole store. And the timing is not a coincidence of months: Envoy did this in February 2022 under CVE-2022-21654, HAProxy in August 2026, citing neither.

One codebase, one release, three answers by build flag, because there is no definition to conform to. As of 26 August 2026 the series is on the **3.4, 3.3 and 3.2** maintenance branches, some of those backports landing within a day of this writing, but **not on 3.0**. No stable release contains it: 3.4.3, 3.3.13, 3.2.22 and 3.0.26 all shipped 29 July 2026, before the series. The only tag that contains it is the **3.5-dev5** development snapshot of 21 August. Six commits, dated 14 to 18 August, plus a build follow-up on 25 August; the mTLS one was reported externally.

2.3 Envoy, and what it corrects

Meshes are not uniformly exposed, and Envoy is the reason. It already binds trust anchors and scope on the server side, so **the 2025 scoping class does not apply to it.** That is not superior design, it is scar tissue: the binding dates from **CVE-2022-21654**, February 2022, which is Envoy's row in section 2. Before that fix the digest covered only the CA certificate and the two pin lists, and `match_subject_alt_names`, the CRL, `allow_expired_certificate`, `trust_chain_verification`, `only_verify_leaf_certificate_crl` and `verify_trusted_ca` were all absent, so a client-certificate verdict reached under one validation context could be resumed into a stricter one. Envoy is four and a half years ahead of HAProxy on this specific point because it took the CVE first. The hash is computed in `ServerContextImpl::generateHashForSessionContextId()`, with the default validator contributing under the comment "Hash all the settings that affect whether the server will allow/accept the client connection... even if session resumption across different listeners is enabled".

The CRL is in that hash, and that matters. The bytes are contributed by `DefaultCertValidator::updateDigestForSessionId()`, which digests `certificateRevocationList()`, the CRL *contents* rather than its path. So publishing a CRL changes the session id context; every existing session then fails the context comparison, and BoringSSL's response is not an error but a fallback to a full handshake, on which the CRL applies. Envoy therefore does re-establish revocation by CRL, crudely, by invalidating every session at once. **The residual is `notAfter` alone.** Two qualifications: only server contexts set a session id context in Envoy, so this does not partition cached upstream sessions; and the SPIFFE validator digests no CRL at all, though Istio does not select it.

The residual exposure is temporal and is precisely the paper's subject. Envoy's only verification hook is `SSL_CTX_set_custom_verify`, which BoringSSL runs on full handshakes only, and `ssl_session_is_resumable()` compares the session id context, endpoint type, session time, version, cipher, the stored client-credential form and QUIC versus TCP. It checks **nothing about the certificate's validity**. `ssl_session_is_time_valid` tests only the session's own timeout against its own creation time, entirely independent of the peer certificate's `notAfter`. BoringSSL's own header says "offering or accepting a session short-circuits certificate verification".

Concretely: Istio issues 24-hour workload certificates (`SECRET_TTL` and `DEFAULT_WORKLOAD_CERT_TTL`, both 24h). BoringSSL's own timeouts are `SSL_DEFAULT_SESSION_PSK_DHE_TIMEOUT` at two days and `SSL_DEFAULT_SESSION_AUTH_TIMEOUT` at seven, **but in a default Istio install neither is ever reached.** BoringSSL keeps its auto-generated ticket key in the `SSL_CTX` (`ticket_key_current` and `ticket_key_prev` in `ssl/internal.h`, rotated by `ssl_ctx_rotate_ticket_encryption_key` under the context's own lock, with no process-scoped state). Envoy's `ServerSslSocketFactory::onAddOrUpdateSecret` (`source/common/tls/server_ssl_socket.cc`) responds to every SDS secret update by calling `createSslServerContext`, which constructs a fresh `ServerContextImpl` and a fresh `SSL_CTX_new`, then swaps it in and drops the old one; there is no timed drain. Istio's agent rotates the workload certificate at `SECRET_GRACE_PERIOD_RATIO`, default 0.5, so roughly every twelve hours with about fifteen minutes of jitter. At each rotation the key is destroyed with the context, every outstanding ticket fails the key-name match in `ssl_decrypt_ticket_with_ticket_keys` and returns `ssl_ticket_aead_ignore_ticket`, and the connection falls back to a full handshake. **The real ceiling is one rotation period, about twelve hours, not two days or seven.** Continuous resumption cannot extend it, because renewed tickets are encrypted under the same doomed key. Envoy's `session_timeout` maps only to `SSL_CTX_set_timeout`, which is TLS 1.2 and earlier, and a search of Envoy's source finds no call to `SSL_CTX_set_session_psk_dhe_timeout` at all, so that configuration field **cannot shorten a TLS 1.3 session**. Anyone reaching for it as a mitigation gets nothing. Rotating `session_ticket_keys` over SDS would bound the window without disabling anything, and Envoy supports exactly that through `session_ticket_keys_sds_secret_config`.

The Istio open item is now closed. A full checkout of istio/istio at master was searched for `session_ticket`, `disable_stateless`, `disable_stateful` and `resumption`: **zero occurrences of each**, against 84 for `DownstreamTlsContext` as a control that the search was live. Inbound mutual-TLS contexts are built in `pilot/pkg/security/authn/utils/utils.go`, which sets only `CommonTlsContext`, `RequireClientCertificate`, `AlpnProtocols` and `TlsParams`. Envoy's defaults therefore apply unmodified.

The bound is accidental, and one deliberate check is made redundant by it. Nothing in the ticket-decrypt path examines a certificate: the key-name match is the entire gate, and BoringSSL's resumption checks are time arithmetic on the session's own timeouts. The twelve-hour ceiling is a side effect of swapping a context whose destructor takes the ticket key with it, on a timer whose purpose is certificate renewal. Separately, Envoy does call `SSL_CTX_set_reverify_on_resume`, which BoringSSL honours on clients only, so an Envoy client re-runs validation against the stored server chain on each resumed handshake; in a default install that can never fire on an expired certificate, because the client's session state also dies at twelve hours while the certificate lives twenty-four. Line numbers here are from the current main and master branches; pin them to the proxy revision named in any published version.

2.4 The fourth answer: switch it off. The provenance behind the paper's table

The paper's section 5 states this roster and the reader should take it from there. This section exists because the quotations have to live somewhere they can be checked, and because four things here are more precise than the paper has room for: **the rollout timestamps**, **that two of the four are measurements rather than vendor announcements and that the evaluation behind them is undated**, **that AWS was excluded from testing rather than tested**, and **the exact scope of the Azure result**. Everything else below is the same claim as the paper's, kept here verbatim as its source. **Four parties do not resume** under client authentication, and not one of them was following a specification when it decided that.

PARTY	WHAT IT DOES UNDER CLIENT AUTHENTICATION	BASIS	WHEN

Cloudflare	"We have disabled TLS session resumption for all customers that have mTLS enabled." Hebrok et al. describe it as a preliminary fix.	Vendor incident post, following a HackerOne report from the Paderborn group	rollout 23 Jan 2025 21:26 UTC to 24 Jan 2025 20:15 UTC
Caddy	"Caddy has disabled session tickets when using client authentication."	Vendor change in response to the same disclosure, reported in the paper's responsible-disclosure section	2025
Google Cloud Load Balancer	"We found that session tickets are disabled when enabling client authentication."	**Measured** by Hebrok et al. section 5.2, not announced by the vendor	reported 2025; the section 5.2 evaluation is undated
AWS Application Load Balancer	Documented as not resuming under mutual authentication.	**Documented** by AWS. Hebrok et al. list it in section 5.3 Limitations as explicitly *not* tested, excluded because "the documentation indicates secure behavior"	standing
curl	Disabled session reuse outright whenever a client certificate was set. Not a provider, but the same reflex, and the policy spanned fourteen months, in force for about ten of them.	Commits `247d890da8` (7.50.1) through `9d3dde37a8` (7.56.0); the timeline is in supporting document 1 section 2.5	Aug 2016 to Oct 2017
Azure Application Gateway	**Keeps resumption on** under client authentication. "The Azure App Gateway supports session resumption with client authentication. However, we could not resume tickets across two configured gateways."	**Measured** by Hebrok et al. section 5.2, in the same evaluation as Google's	reported 2025; the section 5.2 evaluation is undated

The paper's section 5 draws the conclusion from the last row; it is not repeated here. What this document adds to it is the qualification below.

Scope of the Azure result, stated precisely: Hebrok et al. report that they could not resume across two configured gateways, which is an observation about that deployment, not a guarantee about the product. It establishes that Azure does not switch resumption off under client authentication, which is the only claim made here.

2.5 The mesh cost model, with every input sourced or flagged

The paper's cost argument raises a deployment objection: what an over-approximating digest costs a fleet that shares ticket keys. The numbers answering it are derived here.

What a CRL publication actually does. Envoy digests the CRL bytes into the session id context (`DefaultCertValidator::updateDigestForSessionId`, over `certificateRevocationList()`), so a new CRL changes the context and no existing session resumes. But an SDS secret update swaps the validation context in place: Envoy drains connections on hot restart, admin drain, health-check failure and listener removal, not on a secret update, and on a listener update filter chains carried over unchanged keep their connections open. **So a CRL push disables resumption; it does not terminate anything.** The re-handshakes spread over natural connection churn, which between sidecars is slow. Istio's 45-second `drain_duration` is the default drain for *hot restarts* and is unrelated to CRLs. Caveat worth recording: Envoy has no single document enumerating the no-drain guarantee for SDS updates, and the issue asking for exactly that matrix (envoyproxy/envoy#27452) was never answered, so this rests on the drain documentation and the absence of any contrary mechanism.

The worst case, costed anyway. Assume an operator event forces every session to re-establish inside a 45-second window. Per-endpoint asymmetric cost of a mutual TLS 1.3 handshake with ML-KEM-768 and ML-DSA-44 and a two-certificate chain, using pq-crystals AVX2 figures at 3 GHz: encapsulation 67,624 cycles, one signature 333,013, three verifications at 118,412 each, totalling about 756,000 cycles or **0.25 ms**. Those figures are **not from one machine**: pq-crystals measured Kyber on Haswell and Dilithium on Skylake, and supporting document 3, section 1.2, says the two tables must not be added. They are added here deliberately and only as an upper bound. On eBACS figures from a single machine, hertz, the terms it reports stably are 25,929 for encapsulation and 69,059 per verification, which brings the total nearer 0.19 ms even keeping the pq-crystals signature; 0.25 ms is the conservative end and the one the table below uses. On portable code measured locally with OpenSSL 3.6.1 it is nearer

0.65 ms. Bytes per mutual handshake come to roughly **25 kB**: 2,272 for the key exchange, about 17,200 for two two-certificate chains, 4,840 for two CertificateVerify messages, and about a kilobyte of framing. The X.509 body overhead inside that is an estimate, not a measurement.

SESSIONS PER NODE	HANDSHAKES/S PER NODE	CPU, SHARE OF ONE CORE	BANDWIDTH PER NODE
10	0.22	0.006 to 0.014%	0.05 Mbps
40 (the paper's assumption)	0.89	0.02 to 0.06%	0.18 Mbps
100	2.2	0.06 to 0.14%	0.45 Mbps
5,000 (gateway)	111	2.8 to 7.2%	~22 Mbps
20,000 (large gateway)	444	11 to 29%	~90 Mbps

Three figures not to reintroduce. First, "40 ms of CPU per sidecar" was too high: the figure is about 10 ms on optimised x86-64 and about 26 ms on portable code, so 40 ms survives only as a loose upper bound. Second, "about 900 handshakes per second fleet-wide" and "under one per second per sidecar" cannot both be true at 40 sessions per sidecar, because one counts endpoint participations and the other counts sessions; the consistent pairing is 444 and 0.9, or 889 and 1.8. Third, and most important, **no published figure for concurrent TLS sessions per Istio sidecar exists.** Istio's own performance documentation benchmarks at 16 client connections and 1,000 RPS, which is a benchmark rather than a census. The 40 is an assumption; the table above is why that is tolerable, since the model is linear in it and the conclusion holds across three orders of magnitude.

2.6 What Hebrok et al. actually recommend, quoted at length

The identity-versus-policy point is theirs. Their section 7, Countermeasures, verbatim:

Framing: "The vulnerabilities identified in this paper stem from the inconsistent isolation of virtual hosts following TLS session resumption. Addressing these issues requires guarantees that the identities asserted during the initial handshake remain consistent throughout resumed sessions. A robust solution involves binding session tickets to all exchanged certificates and asserted identities from the initial handshake."

Server authentication: "We recommend binding the ticket to the server certificate chain." *Client authentication:* "We recommend the server to store the certificate presented by the client in the session ticket, as recommended by the standard. This way, the server can revalidate the certificate in a resumption handshake, ensuring the authentication is still valid for the resumption."

The policy-binding alternative, which is the point in question: "Alternatively, the server could bind the ticket to the **rules** that the certificate passed initially. If the server expects different rules to pass upon resumption, the server can fall back to a full handshake. This approach may be easier to implement, but has worse performance for resumptions where the rules differ." And in their implementation notes: "Upon resumption, the certificate is not checked again, even if a different list of certificates to validate against is specified. Therefore, we recommend including the list of certificates in the session context too."

They also rejected identity-only binding explicitly: "We also considered whether binding the ticket to the SNI is a viable alternative. While it can prevent some client authentication bypasses we observed, fallback mechanisms in multi-server scenarios may cause authentication to be bypassed… Further, session resumption across hostnames may be used intentionally within the same certificate."

So what remains. Both halves of the point are theirs: that identity binding is insufficient, and that the authentication policy or "rules" must be bound. What this paper adds is confined to specification. Their text is advice to implementers, expressed through OpenSSL's session-context API, and offers the rules-binding as an alternative with a performance caveat rather than as a requirement. Two implementations have since acted on the principle independently, Envoy under CVE-2022-21654 and HAProxy in 4a8f7622, hashing different input sets. What does not exist is normative text an implementation could conform to, nor answers to the two questions that arise the moment it is written down: when an amortised check expires, and what an over-approximating digest costs a fleet sharing ticket keys. Section 4 is that text.

3. Prior art and what remains

PRIOR WORK	ESTABLISHES	LEAVES OPEN

RFC 9846 section 4.7.1	Names the quantity, recommends a bound	Advisory, undefined, nothing on the wire
RFC 9190, EAP-TLS	**Normative text for item 2, in a published RFC**: authorization policies MUST be followed on resumption	One profile only, and fresh revocation checking is merely MAY
Chuat et al., EuroS&P 2020, VII-C	Proposes coordinating PSK lifetime with certificate lifetime	Judged hard to deploy; no mechanism given
Hebrok et al., USENIX Sec 2025	Finds the 2025 cluster empirically, **and proposes the countermeasure in both forms**: bind the ticket to the server certificate chain and revalidate the stored client certificate, **or** "bind the ticket to the rules that the certificate passed initially". Rejects SNI-only binding as insufficient	Advice to implementers via OpenSSL session contexts; the rules form is offered as an alternative with a performance caveat, not as a requirement. No normative text, and no answer to when an amortised check expires or what an over-approximating digest costs a shared-ticket-key fleet
nginx, BoringSSL, Go, HAProxy, Envoy, Fastly	Prove a bound is implementable and shipping *in the session*	Six different answers, with nothing in any specification saying which of them conform; plus three transient states described in 2.2 rather than counted
Apache httpd	Proves the condition can be enforced at all	Enforced at HTTP routing, not in the session: the resumption succeeds and the request is refused with 421. A seventh party, discharging at a different layer rather than a seventh semantics

What remains is a definition written for the general case and for the server side. The contribution is specification, not discovery, and the credit is divided: **Hebrok et al. proposed the condition idea** (bind the ticket to the chain, revalidate the client certificate) and **Chuat et al. proposed the lifetime idea**. RFC 9190 shows both can be written normatively. The new information since 2020 is that Go and Fastly have shipped most of the lifetime half, and that the condition half keeps being rediscovered one CVE at a time.

3.1 The comparator base rates, listed rather than asserted

The paper answers "eleven identifiers is rare" by saying it is the ordinary rate for a narrow TLS implementation class. A count is meaningless without its inclusion rule, so both are given here.

TLS state-machine flaws. Rule: CVEs in widely deployed TLS implementations whose root cause is incorrect handshake state-machine enforcement, accepting, skipping or mis-ordering handshake messages, enabling downgrade, impersonation or authentication bypass. Excludes parsing and memory-safety bugs, pure denial of service, and cryptographic weaknesses. Members: CVE-2014-0224 (OpenSSL, ChangeCipherSpec injection), CVE-2014-6593 (Oracle JSSE), CVE-2015-0204 and CVE-2015-0205 (OpenSSL, FREAK and the DH-certificate client-auth bypass), CVE-2015-1637 (Schannel), CVE-2015-1067 (Apple Secure Transport), CVE-2015-2318 and CVE-2015-2319 (Mono), CVE-2020-24613 (wolfSSL, WAIT_CERT_CR). **Nine identifiers, 2014 to 2020, eight of them inside the fourteen-month EarlyCCS and SMACK/FREAK burst.** Caveat: CVE-2014-6593's official text says only "Unspecified vulnerability", and its SKIP-TLS attribution comes from Beurdouche et al., "A Messy State of the Union", IEEE S&P 2015, not from the CVE record.

Client certificate and hostname verification bypasses. Rule: logic flaws, not memory corruption, not CA compromise, not protocol downgrade, in chain, signature or hostname verification of widely deployed TLS libraries or platform stacks, causing a client to accept an attacker-available certificate. Excludes application-level misuse, which Georgiev et al. documented in the hundreds and which would make the count meaningless. Members: CVE-2013-4238 (Python), CVE-2014-1266 (Apple, "goto fail"), CVE-2014-0092 (GnuTLS), CVE-2015-1793 (OpenSSL alternative chains), CVE-2021-3450 (OpenSSL X509_V_FLAG_X509_STRICT), CVE-2021-44531 (Node.js URI SAN). **Six under that rule, 2013 to 2022.** Under a broader reading admitting platform-validation flaws, add CVE-2014-1568 (NSS, BERserk) and CVE-2020-0601 (Windows CryptoAPI, CurveBall), giving eight. Note "goto fail" is a key-exchange signature check rather than X.509 chain logic; it is counted here because it is universally treated as a verification bypass, and the bucket is stated so a reader can move it.

So eleven identifiers over eleven years and five months sits inside the comparator range rather than above it, and the objection applied consistently would dismiss both of the most canonical TLS implementation-flaw classes. No trend is claimed, and section 3 explains why: the

2025 cluster is one research group looking systematically for the first time.

3.2 The adjudication evidence

The paper says the class is inconsistently adjudicated and cites a report to curl closed as *Informative* rather than as a vulnerability: **HackerOne report 3781305**: curl's OpenSSL backend accepts a resumed TLS 1.2 session after the server certificate has expired, while a full handshake with the same certificate fails, because the resumed path honours the cached verification result. That is the mechanism of supporting document 1, section 5.2's measurement, at TLS 1.2 rather than 1.3, reported in a product that holds four rows of the table, and declined. **It is recorded here because the paper's claim is about adjudication, not code, and a claim about how something was triaged has to point at the triage.**

4. The proposal, and the provenance of each part

This section corresponds to the paper's section 8: the paper lists four ways to satisfy the rule, this one lists the three conditions plus the over-approximation that discharges them. There is no separate age-ceiling item: item 1 is the temporal condition, and the RFC 9846 paragraph quoted in section 1.1 already recommends limits.

The governing principle is parity: resumption must not be more permissive than the full authentication it stands in for, and is not required to be stricter. Everything below follows from it.

1. **Time.** No resumption once a temporal condition that full authentication evaluated has lapsed, across every certificate on the cached path rather than the leaf alone. *Provenance:* Go has shipped the leaf form since 1.21 and the full-chain form since CL 739321 in January 2026, so this is not hypothetical. *Why parity and not strictness:* a condition full authentication did not evaluate, such as the trust anchor's own validity where RFC 5280 does not require checking it, is not added at resumption. golang/go#77359 is the live compatibility complaint that results when resumption becomes stricter than the handshake.
2. **Scope, meaning identity and policy.** Not only the authorisation identity, which is RFC 9846's client-side SNI rule extended to the server, but the peer-authentication policy in force: whether client authentication was required, optional or not requested, the validation configuration, and any error-tolerance setting whose relaxation changes the outcome. *Provenance and credit:* Envoy's fix for CVE-2022-21654 added a contract requiring its session hash to cover "all configuration used to validate a peer certificate", and HAProxy's 4a8f7622 hashes the verify mode and the ca/crt-ignore-err bitfields. The principle is theirs and it exists in shipped code. What does not exist is the requirement. **This is the item that carries the paper's contribution**. Bind only the host and CVE-2025-23419 conforms: two nginx server blocks, one certificate, one CA, differing only in `ssl_verify_client`.
3. **Continued acceptance.** The trust configuration in force must still accept the cached credential; removal of a relied-on anchor and any narrowing change invalidate, a purely widening addition need not. *The amortisation boundary, which is the load-bearing detail:* this may be discharged once per change to configuration or status data rather than per resumption, and that period ends at the status snapshot's own `nextUpdate`. Past it, a hard-fail endpoint must refresh before resuming and a soft-fail endpoint treats the snapshot as unfetchable, because an endpoint must not resume on data it would refuse in a full handshake (RFC 5280 section 6.3, RFC 6960 section 4.2.2.1). So the period is the lesser of the next configuration change and `nextUpdate`, not a free parameter, and steady state is one refresh per boundary rather than a fetch per resumption. *Acknowledged:* this goes past RFC 9190, which leaves fresh revocation checking at MAY even for full handshakes. Softening it would reopen the hole.
4. **Over-approximation is conforming.** An endpoint may invalidate more broadly but not more narrowly, so a digest over the invalidating inputs satisfies item 2 outright. *It satisfies item 3 only when paired* with either capping the resumption state's lifetime at the snapshot's remaining validity, or treating snapshot expiry as a scope change: a digest changes when a snapshot is replaced, not when it merely expires unreplaced, and without the pairing a hard-fail endpoint keeps resuming past `nextUpdate`.

The price of item 4, stated rather than left implicit. A scope digest makes a shared ticket-key fleet fail closed, but it does not make it interoperate. Two stacks that hash different inputs, or the same inputs in different encodings, never resume each other's sessions: an nginx tier and an Envoy tier behind one ticket key accept only their own tickets, and the symptom is a quietly elevated full-handshake rate rather than an alarm. Operators of mixed fleets pay that in CPU and latency and should measure it. The default is still right, because a fleet that resumes across differing trust configurations is precisely the cross-context bypass Hebrok et al. describe, and the paper this work builds on is titled *STEK Sharing is Not Caring* for that reason. Real

interoperability needs a canonical serialisation of the resumption scope: an agreed list of covered inputs, a deterministic encoding, a fixed digest. That is protocol work and this paper does not do it, which is the honest boundary of the contribution.

5. References

This is the **shared bibliography for the whole submission**, reproduced identically in each supporting document so that any one of them can be read on its own. Not every entry is cited in this particular document.

S. Hebrok, T. L. Storm, F. M. Cramer, M. Radoy, J. Somorovsky, "STEK Sharing is Not Caring: Bypassing TLS Authentication in Web Servers using Session Tickets", 34th USENIX Security Symposium, August 2025. https://www.usenix.org/conference/usenixsecurity25/presentation/hebrok

RFC 9846, TLS 1.3, section 4.7.1, July 2026. https://www.rfc-editor.org/rfc/rfc9846.txt

RFC 9190, EAP-TLS 1.3. https://datatracker.ietf.org/doc/rfc9190/

L. Chuat et al., "SoK: Delegation and Revocation, the Missing Links in the Web's Chain of Trust", IEEE EuroS&P 2020. https://arxiv.org/abs/1906.10775

R. Holz et al., "The Era of TLS 1.3", IMC 2019. https://arxiv.org/abs/1907.12762

Cloudflare, "Resolving a Mutual TLS session resumption vulnerability", 7 February 2025. https://blog.cloudflare.com/resolving-a-mutual-tls-session-resumption-vulnerability/

BoringSSL include/openssl/ssl.h; Chromium net/socket/ssl_client_socket_impl.cc; NSS lib/ssl/tls13con.c. All retrieved 25 August 2026.

nginx changeset 8086:496241338da5, 12 October 2022, shipped in 1.23.2. HAProxy commit 9f1d590999c2, 18 August 2026.

`Scripts: pdf/rows.py (the table and its derived counts), fftest/server.py, fftest/nonbrowser.sh, fftest/RESULT.md.`