§ THE UNIT OF SAVING, FIVE DEPLOYMENTS, AND THE ARITHMETIC AT FLEET SCALE

What conforming saves, at scale

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 an operator gets back by being able to resume safely, per connection and at fleet scale, with the assumptions stated so the arithmetic can be disagreed with.

1. The unit is a connection, not a request

This is the first thing to get right, because getting it wrong overstates every figure below by up to an order of magnitude. A TLS handshake is paid **per connection**. Under HTTP/2, or HTTP/1.1 with keep-alive, many requests share one connection, so ten API calls are usually one handshake and not ten.

Where the two converge is worth naming, because those are the deployments this matters most to:

- **Mobile clients.** Radio state changes, network handover and app backgrounding tear connections down constantly, so a mobile app approaches one handshake per burst of activity.
- **Serverless and autoscaled workers.** A cold instance has an empty connection pool. Scale events and deploys produce a spike of fresh handshakes.
- **Server-to-server clients that do not reuse an agent.** A new HTTP client per call, which is a common default, is a new connection per call.
- **Service meshes.** Pod-to-pod connections churn on every deploy, scale event and eviction, and every one of them is mutual TLS.

2. Two different quantities, which must not be added together

There is a saving from **resumption**, and a saving from **the rule**. They are not the same number and conflating them overstates the case badly.

Resumption saves the certificate material, because a resumed connection sends none. That saving is already being realised across most of the web: 94.8% of reachable TLS domains issue a session ticket, so for ordinary server-authenticated traffic the bytes are being saved today and **the rule changes nothing about them**. What the rule buys there is correctness, not bandwidth.

> **The rule's saving is the per-connection saving, multiplied by the connections that cannot resume today because nobody can say whether resuming is safe. That population is the one using client certificates, and it is the one that has been switching resumption off.**

PROFILE	SAVED BY RESUMING	DOES THE RULE CHANGE IT?
Classical, server authentication only	3,465 B	No. Already resuming. Correctness only.
Classical, mutual TLS where resumption was switched off	3,465 B	**Yes, all of it. This is what the four parties gave up.**
Post-quantum, server authentication only	18,408 B + one round trip	Only where the operator also stopped resuming. Otherwise correctness only.
Post-quantum, mutual TLS	**29,452 B + one round trip**	**Yes, and this is the case that grows.**

The mutual-TLS row is the server's 18,408 B plus a client chain in the other direction, 11,044 B for ML-DSA-44 throughout. The round trip appears because the post-quantum server flight is 19,762 B against a 14,600 B initial congestion window: it does not fit in the first ten packets, so the server stops and waits for an acknowledgement before sending the rest.

Why the last row is the one that matters, and why it grows. Today the population that has given up resumption is small and named: four parties, in the paper's section 04. The reason it grows is that they did not switch it off out of carelessness. They switched it off because the specification does not say what a resumed connection must re-establish, so the safe answer available to them was to stop. Every operator who deploys post-quantum authentication under mutual TLS meets the same absence and has the same safe answer available. Writing the rule is what stops that answer from being the only one.

3. Five deployments, before and after

DEPLOYMENT	TODAY	CONFORMING	SAVING
nginx front door, partner API on mTLS	`ssl_verify_client on` in one server block and `ssl_session_tickets off`, set after CVE-2025-23419 or out of caution. Every partner connection sends both chains in full.	Trust configuration and authentication policy bound into the session scope, so a ticket minted at the public virtual host provably cannot resume into the mTLS one. Tickets back on.	29,452 B per connection
Istio and Envoy mesh	mTLS on every hop, 24-hour workload certificates, and connection churn on every deploy, scale event and eviction. The ticket key outlives the identity, bounded near twelve hours by accident.	The scope digest Envoy already ships, completed with a lifetime cap at the status snapshot's remaining validity, so the bound is designed rather than incidental.	29,452 B per connection, on the highest connection count most estates run
Anything behind a CDN's mTLS product	Measured, not hypothetical: resumption under mutual TLS has been off across Cloudflare's network since 24 January 2025, and its published position is that it is exploring ways to bring the performance back.	A definition to restore it against, rather than a per-vendor judgement call each time the class resurfaces.	3,465 B today, 29,452 B post-quantum
Mobile backend with client certificates	Connection reuse genuinely breaks down, so the handshake count approaches one per activity burst, on the network where a round trip costs most.	Resumption available, so the certificate material is sent once per session rather than once per reconnection.	29,452 B and one round trip, at 80 ms or worse
Apache httpd, per-location client auth	It resumes, then refuses the request with HTTP 421. The handshake is paid, the request is rejected, and the client retries with a full handshake.	The decision deferred correctly, or the condition re-established, rather than a wasted handshake plus a retry.	one full handshake per rejected request avoided

4. Today against with resumption, at one million users

The point is not the delta on its own. It is what a deployment pushes today, and what it would push if it could resume. Both columns, so the size of the change is visible rather than asserted.

Per connection, the resumed figure is the full flight minus the certificate material, which lands exactly on the framing the model already states: 298 B classically, 1,354 B post-quantum. Nothing new is introduced here.

PROFILE	TODAY, PER CONNECTION	WITH RESUMPTION	SAVED
Classical, server authentication	3,763 B	298 B	3,465 B
Post-quantum, server authentication	19,762 B	1,354 B	18,408 B
Post-quantum, mutual TLS	**30,806 B**	**1,354 B**	**29,452 B**

4.1 The same three, at one million users

Users multiplied by connections per day, by the per-connection figures above, by 365. Three connection-reuse assumptions, because the answer depends on that far more than on the byte model. Pick the block that matches your traffic rather than the one that flatters the argument.

1,000,000 USERS	TODAY	WITH RESUMPTION	SAVED
10 connections per user per day, no connection reuse			
Classical, server authentication	13.73 TB/yr	1.09 TB/yr	12.65 TB/yr
Post-quantum, server authentication	72.13 TB/yr	4.94 TB/yr	67.19 TB/yr
Post-quantum, mutual TLS	**112.44 TB/yr**	**4.94 TB/yr**	**107.50 TB/yr**
3 connections per user per day, one app session plus reconnections			
Classical, server authentication	4.12 TB/yr	0.33 TB/yr	3.79 TB/yr
Post-quantum, server authentication	21.64 TB/yr	1.48 TB/yr	20.16 TB/yr
Post-quantum, mutual TLS	**33.73 TB/yr**	**1.48 TB/yr**	**32.25 TB/yr**
1 connection per user per day, HTTP/2 keep-alive holding			
Classical, server authentication	1.37 TB/yr	0.11 TB/yr	1.26 TB/yr
Post-quantum, server authentication	7.21 TB/yr	0.49 TB/yr	6.72 TB/yr
Post-quantum, mutual TLS	**11.24 TB/yr**	**0.49 TB/yr**	**10.75 TB/yr**

Read the rows as section 2 defines them. The mutual-TLS row is the rule's saving: it is what an operator who cannot safely resume today would get back. The classical row is context, already realised by almost everyone, and the rule does not add it. Adding the two together would be double counting. And multiply by the share of connections that actually resume, which section 5 bounds.

4.1 The round trip, separately

Bytes are not the whole cost. The extra round trip is paid by the user, in time.

CONNECTIONS PER USER PER DAY	AT 30 MS RTT	AT 80 MS RTT	AGGREGATE ACROSS 1M USERS
10	0.30 s per user per day	0.80 s per user per day	83 to 222 hours per day
3	0.09 s	0.24 s	25 to 67 hours per day
1	0.03 s	0.08 s	8 to 22 hours per day

5. What this model does not claim

It is a model, not a measurement of anyone's traffic. The byte constants are measured and audited in supporting document 3. The multipliers are assumptions, stated above so they can be replaced with your own.

It only pays off on connections that actually resume. Observed resumption rates vary enormously by vantage, from 9.7% of TLS 1.3 connections carrying a PSK in a 2019 passive measurement to 94.2% for CDN-terminated traffic. Multiply by the rate you actually observe; the paper's section 04 gives the range and why the figures must not be averaged.

It says nothing about how many deployments use client certificates. That population has never been measured globally and this paper does not estimate it. The mutual-TLS column is the saving per connection for those that do, not a claim about how many there are.

And if post-quantum chains get smaller, this shrinks with them. Merkle Tree Certificates or trust-anchor negotiation would reduce the certificate material, and the saving is exactly that material. Supporting document 3 covers what each would and would not change.

Every constant here traces to supporting document 3, and every measurement behind it is public in `github.com/quantakrypto/tls-resumption-tests`, so these figures can be recomputed rather than taken on trust.