Post-Quantum

Cryptography Conference

Lessons Learned from Testing Millions of Servers for Post-Quantum Compatibility

Protocol ossification delayed the rollout of TLS 1.3 for years, and has once again become a roadblock in the rollout of post-quantum cryptography. In a recent large-scale study of TLS servers, we assessed the deployment compatibility of post-quantum key agreements, uncovering surprising results and insights. Notably, we observed protocol ossification in areas beyond the well-known issue of fragmented ClientHello messages due to large key sizes. We believe more surprises will emerge with post-quantum certificates, making deployment far more complex than a “flip-of-a-switch” transition. In this talk, we share our findings from the study, and emphasize the importance of testing early to identify potential post-quantum migration challenges rather than making assumptions about where issues may arise. We walk through the subtle deployment complexities and operational issues that can arise when managing the complexities of post-quantum PKI implementations, particularly for end-user connection stability. By offering practical insights, we hope to contribute to a smoother shift to the post-quantum era, enhancing crypto-agility and strengthening the reliability of the Web PKI as a by-product.

Syed Suleman Ahmad
Research Engineer at Cloudflare

SSL.com PQ SHIELD HID KEYFACTOR ENTRUST

January 15 and 16, 2025 - Austin, TX (US) | Online

PKI Consortium Inc. is registered as a 501(c)(6) non-profit entity (“business league”) under Utah law (10462204-0140) | pkic.org

PKI Consortium

Lessons Learned from testing **Millions of Servers** for **Post-Quantum Compatibility**

Suleman Ahmad, Research Engineer, Cloudflare
suleman@cloudflare.com

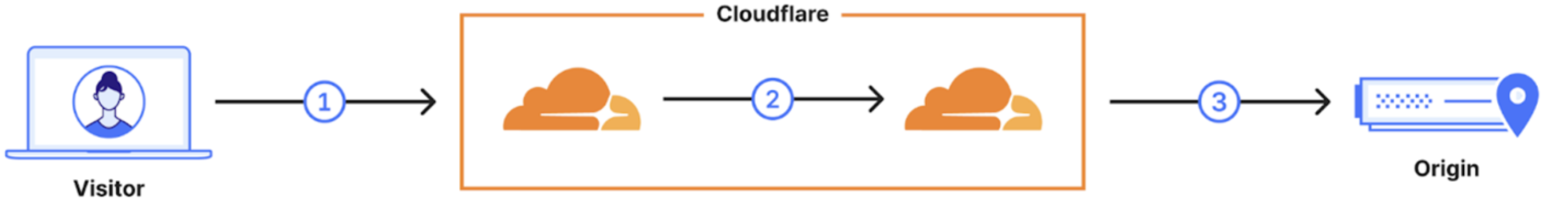
PKI Consortium Post-Quantum Cryptography Conference, January 16th, 2025

About Cloudflare

Cloudflare is used as a global reverse proxy by approximately **20%** of all websites.

- Network spans more than 330 cities in over 120 countries. Interconnects with approximately 13,000 networks globally.
- We serve over 63 million HTTP requests per second on average.
- Responsible for serving a major portion of Internet connections.

Secure Connections Involved

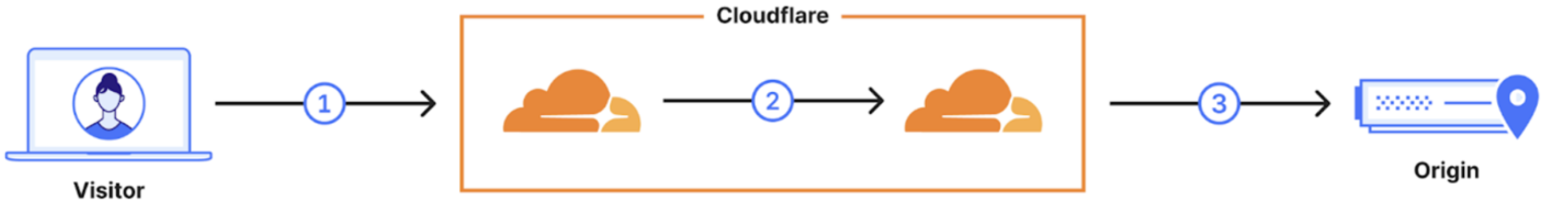


The first connection is from the visitor's browser to Cloudflare.

The visitor's request can be routed through Cloudflare's Internal network.

The upstream connection is between Cloudflare and the customer's origin server.

Secure Connections Involved



The first connection is from the visitor's browser to Cloudflare.

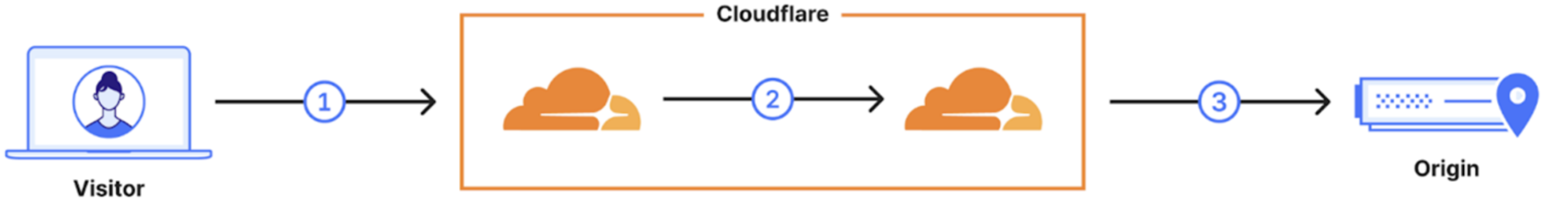
The visitor's request can be routed through Cloudflare's Internal network.

The upstream connection is between Cloudflare and the customer's origin server.

Goal: All connections <u>must</u> be post-quantum secure

But first, remember: *Two* post-quantum migrations

1) Post-Quantum Key Agreements 🤝



🔁 We enabled PQ support back in 2022, browsers ramping up PQ connections to our network.

Today, +30% connections are PQ secure (by *X25519MLKEM768)* as shown on Cloudflare Radar.

🔁 In 2023, every engineering team created inventory of cryptography used, risks, and planned/executed migration (huge effort!).

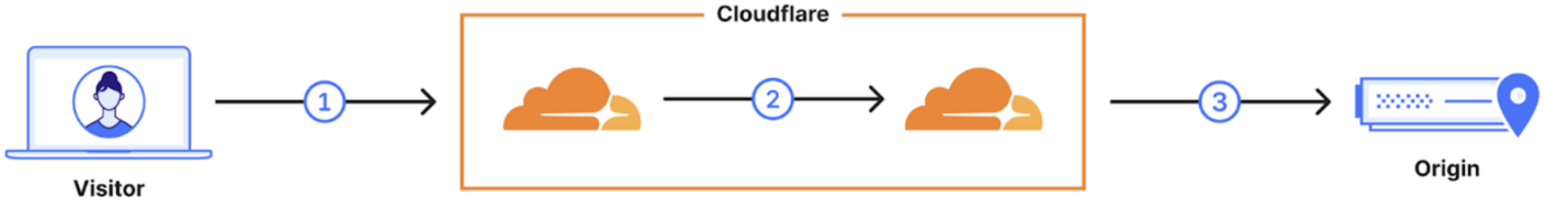
Many of our internal connections are secured by PQ hybrids (prioritizing sensitive connections), but a long fat tail remains.

🔁 PQ supported for connections from Cloudflare to origin servers.

But, origin servers must be updated to support post-quantum connections over classical ones.

0.8% of origins ‘prefer’ PQ at time of writing.

2) Post-Quantum Digital Signatures 🖋



⚠️ Less urgent, but much more challenging...

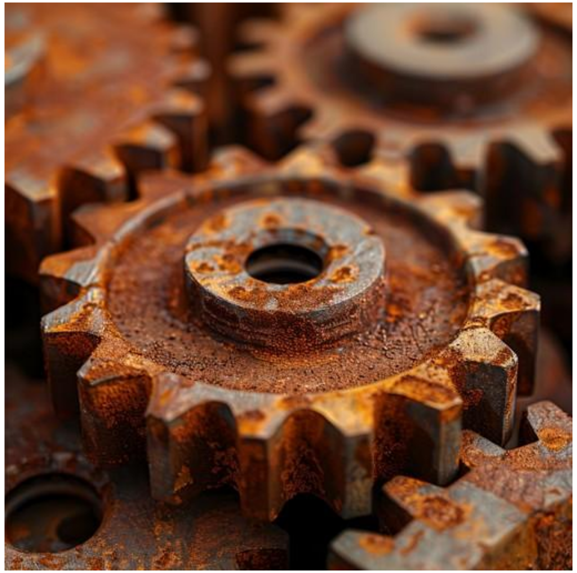
- Many more parties involved than for key agreement
  - library developers, browsers, server operators, CAs, HSM manufacturers, etc.
- There is no all-around great PQ signature (most are too large, some are too slow)
- Typically 5 signatures and 2 public keys when visiting a website
  - not even counting DNSSEC or OCSP stapling

🔄 *Ongoing experiments with ML-DSA, along with efforts for reducing the number of signatures.*

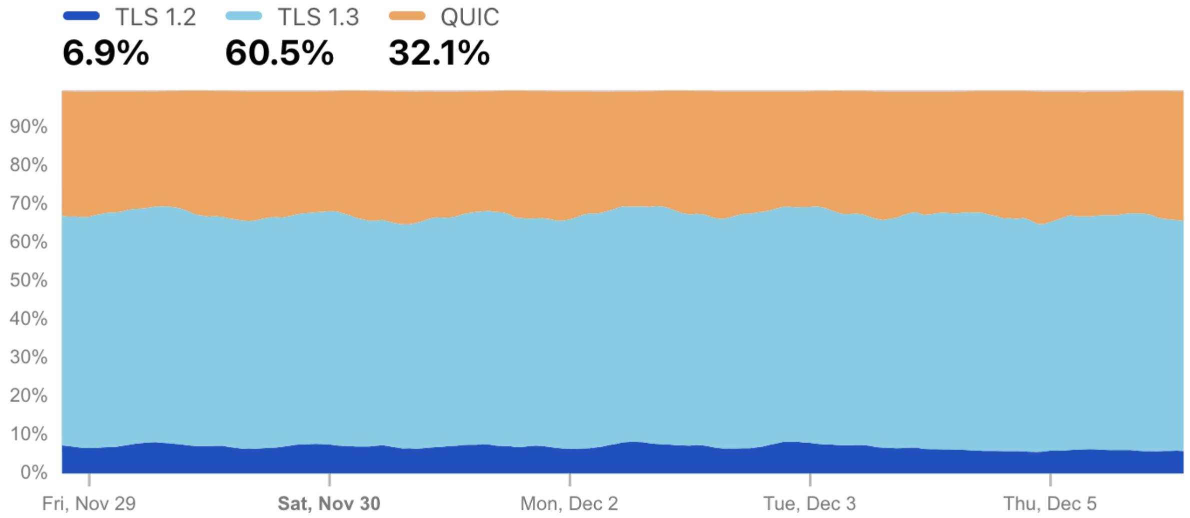
This talk: **Lessons learned** on migrating to post-quantum security for all those connections

Changing the Internet / WebPKI is hard

- **Protocol Ossification**. Even if designed for flexibility and upgrades, unused protocol features often fail due to implementation issues.
- **Highly Diverse Landscape**. Numerous users and stakeholders have different performance needs and software update cycles.
- **Varied Capabilities**. We can't assume everyone has fiber internet, modern CPUs, state storage, or the ability to update systems.



Protocol ossification delayed the rollout of TLS 1.3 for years



After years of testing and adding workarounds, the final version of TLS 1.3 is a success, used by over 93% of our visitors.

Assessing deployment compatibility for PQ TLS 1.3

From our ongoing internal migration and **active scans of million of servers** across the Internet, we've discovered protocol bugs which cause rejection of post-quantum connections.

E.g. Origin facing scans reveal **~0.34%** servers are incompatible when we send post-quantum keyshare in `ClientHello` (**a significant number at Internet-scale**).

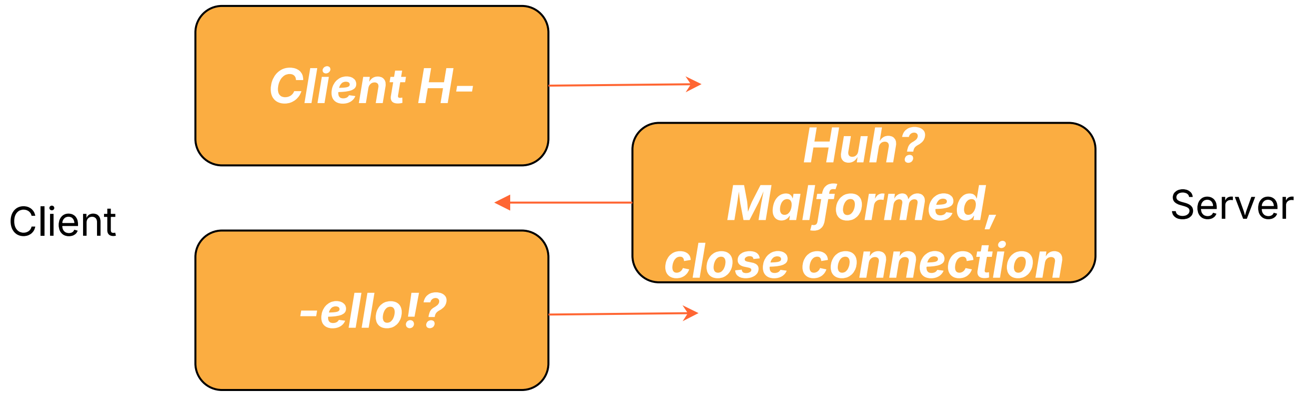
Note: It is ***not*** a 'flip-of-a-switch' transition to post-quantum. Let's have a look why...

Lesson #1: Large PQ Keys and Signatures Cause Side-effects

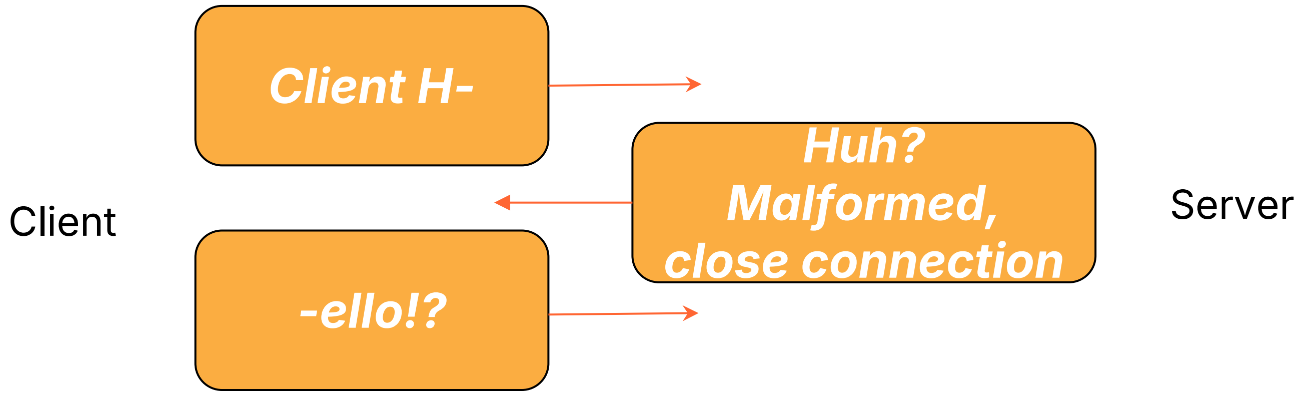
Post-quantum keyshares in TLS connections are big! Migration has revealed side-effects...

	Keyshares size (in bytes)		Ops/sec (higher is better)	
Algorithm	**Client**	**Server**	**Client**	**Server**
X25519 (classical)	32	32	19,000	19,000
ML-KEM-512	800	768	45,000	70,000
ML-KEM-768	1,184	1,088	29,000	45,000
ML-KEM-1024	1,568	1,568	20,000	30,000

Lesson #1: Large PQ Keys and Signatures Cause Side-effects



Lesson #1: Large PQ Keys and Signatures Cause Side-effects



- Used to be so rare that many software and hardware systems falsely assume it never occurs
  - e.g. buggy servers do not call *read()* more than once to read the entire `ClientHello` (tldr.fail by Google captures some known incompatibilities).
- In our scans, 0.02% of origin servers consistently failed (*connection resets*) to parse a fragmented ClientHello but always succeeded when it fit within one packet.
  - Your network's MTU limits can determine the extent of fragmentation.

Lesson #1: Large PQ Keys and Signatures Cause Side-effects

For five signatures,

Using only ML-DSA-44

+14,724 bytes

Using ML-DSA for the TLS handshake and FN-DSA for the rest

+7,293 bytes

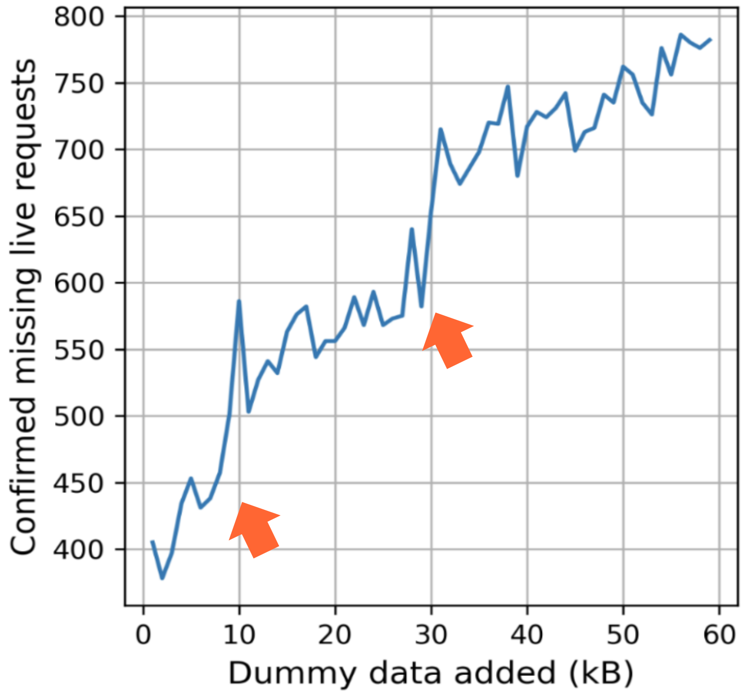
Is that too much?

Lesson #1: Large PQ Keys and Signatures Cause Side-effects

Bump in missing requests suggests some clients or middleboxes do not like certificate chains longer than 10kB and 30kB.

This is problematic for composite certificates.

Instead configuring servers with multiple certificates and letting TLS negotiate.



Lesson #2: Don't Assume Fixed Values for Extensible Fields

- Without extensible fields in TLS, migrating to post-quantum cryptography would have required major overhauls to the protocol design, significantly delaying adoption to PQC.
  - **`supported_groups`** extension allows specifying key agreement algorithms
  - **`signature_algorithms`** allows clients to advertise supported signature schemes

Lesson #2: Don’t Assume Fixed Values for Extensible Fields

- Without extensible fields in TLS, migrating to post-quantum cryptography would have required major overhauls to the protocol design, significantly delaying adoption to PQC.
    - **`supported_groups`** extension allows specifying key agreement algorithms
    - **`signature_algorithms`** allows clients to advertise supported signature schemes

- **Case Study: Ossification for Key Exchange**

  “A *handful* of external origin servers failed to successfully establish a TLS connection when **non-existent** key exchange algorithms were presented.

  Instead of renegotiating for valid parameters, they **only accepted specific hardcoded values**, causing compatibility issues with different (classical or post-quantum) algorithms.”

```
if (keyExchange == "ECDH") {
    acceptConnection();
} else {
    rejectConnection();
}
```

Lesson #2: Don't Assume Fixed Values for Extensible Fields

To help prevent ossification, protocols such as TLS 1.3 and QUIC allow using GREASE (RFC 8701), where clients can send unknown identifiers for these extensible fields on purpose, to ensure flexibility and maintain interoperability.

While *some* ossified implementations still remain, client support for GREASE has significantly reduced the impact, easing the transition to post-quantum algorithms on Internet.

TL;DR: Embrace ***Crypto-agility*** in software / protocol design.

Lesson #3: Software may not fully adhere to the standard

TLS 1.3 standard (RFC 8446) requires TLS stacks to implement *`HelloRetryRequest`*.

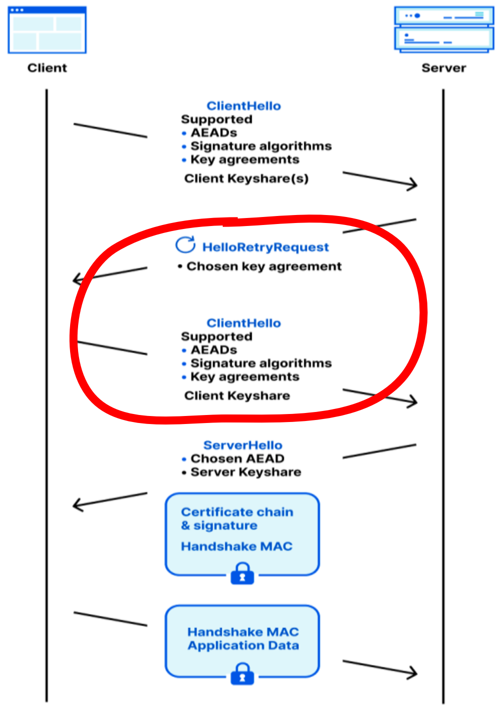
- A mechanism to allow the server to request changes to the client's handshake parameters (e.g key share group)

Migration Use-Case:

- Client sends a keyshare using a 'classical' key agreement.
- Server prefers using post-quantum key agreement, sends a *`HelloRetryRequest`* to the client to transition to post-quantum secure connection.

'Safe' way to transition to PQ key agreement.

Unsupported Key Share



Lesson #3: Software may not fully adhere to the standard

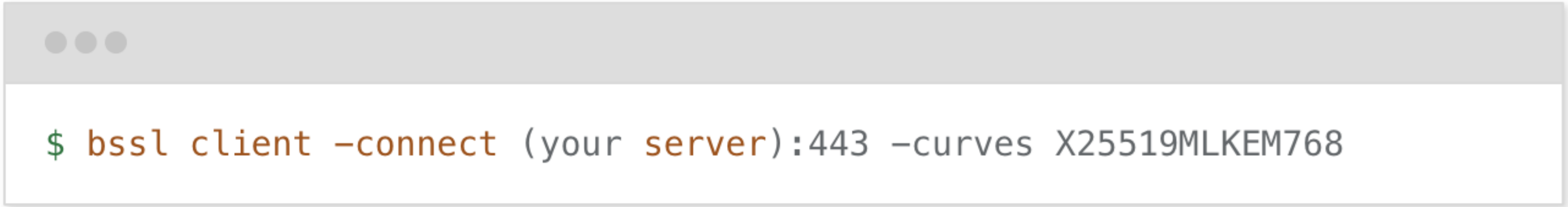
But our expectations did not align with reality...

- **Case Study: Broken Graceful Fallback Mechanism**
  - A fraction of servers we scanned (0.32%) did not implement `HelloRetryRequest` properly, rejecting a second ClientHello on the same connection.
    - And yes, broken HRR implementations were far more than Split ClientHello bug
  - Incompatibility is not related to the choice of key agreement or whether the first *ClientHello* fits in a single packet.
    - It is simply due to a lack of HRR support, or in other words, non-compliant with the standard.

* For instance, servers using the Rust TLS library rustls did not implement `HelloRetryRequest` correctly before v0.21.7. **Keep libraries up-to-date!**

How can you test your systems for compatibility?

Example of a client: *bssl* tool of BoringSSL



```
$ bssl client -connect (your server):443 -curves X25519MLKEM768
```

Verify that the `ECDHE curve` in the handshake output indicates `X25519MLKEM768`.

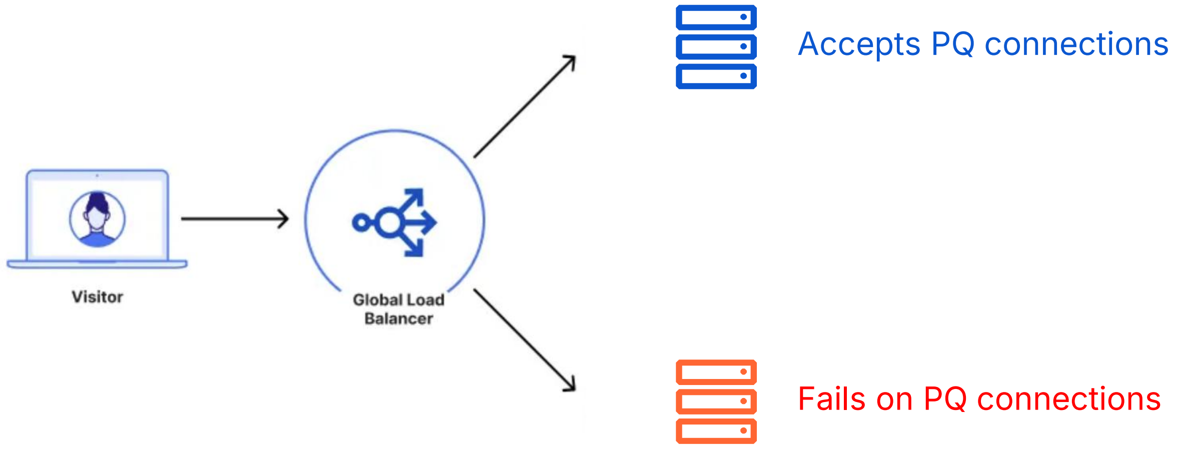
- Or, check if your server *"prefers"* PQ key agreement by specifying a classical key agreement like *X25519* (can use our custom bssl fork)
    - Which will trigger a *HelloRetryRequest* to that PQ key agreement.

- Issues may stem from server software or middleboxes, so contact your vendor as soon as possible for a patch, if needed.

When Testing Isn’t Straightforward…

The long tail is *inconsistently* long

- **Load balancers**. Backend server running heterogeneous TLS stacks.
  - More prominent on shared hosting providers



The long tail is *inconsistently* long

- **Load balancers**. Backend server running heterogeneous TLS stacks.
  - More prominent on shared hosting providers

- **Atomic Fragments**. Fragmented ClientHello on IPv6-only servers causing re-assembly issues.
  - Bug fix upstreamed to Linux Kernel

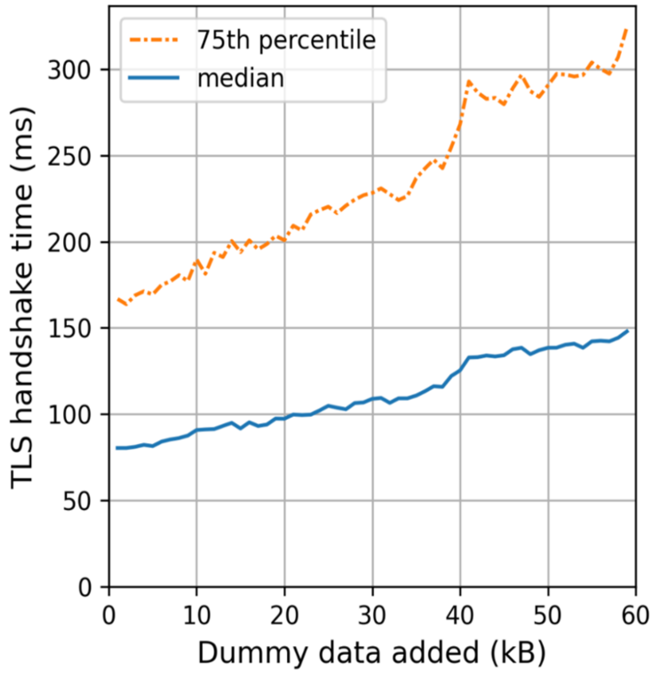
The long tail is *inconsistently* long

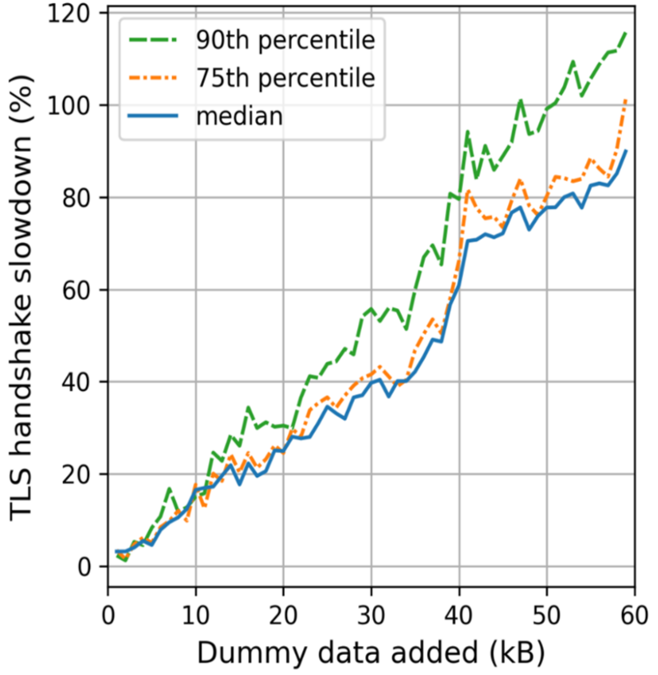
- **Load balancers**. Backend server running heterogeneous TLS stacks.
  - More prominent on shared hosting providers
- **Atomic Fragments**. Fragmented ClientHello on IPv6-only servers causing re-assembly issues.
  - Bug fix upstreamed to Linux Kernel
- **Network Boundaries.** Figure out what pieces of hardware and software are involved in the connection.
  - Varying configuration of proxies, Path-specific MTU differences, etc

The long tail is *inconsistently* long

- **Case Study: Out-of-Order Packet Assembly**
  - Server able to successfully parse fragmented ClientHello.
    - Still intermittently failing for some requests
  - Investigation revealed that due to packet loss, the second fragment reached the server before the first fragment.
    - Servers unable to handle out-of-order assembly fail the handshake in cases of packet loss
  - Example: Chromium bug tracker report for Palo Alto firewall – important to engage with your vendors.

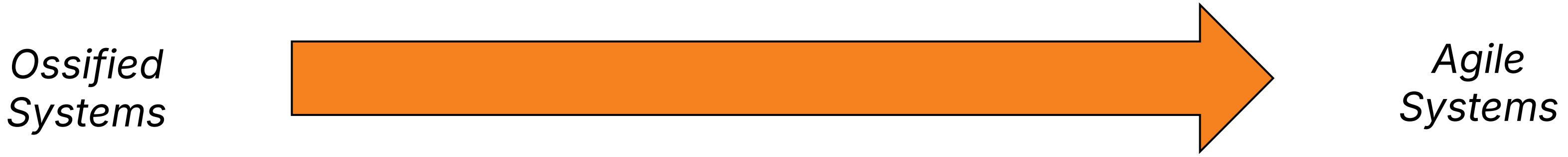
What about PQ certificates? Reducing Signatures can help





blog.cloudflare.com/sizing-up-post-quantum-signatures, 2021

Meanwhile: PQC TLS Migration Journey Continues



- Migration is acting as a stress test for revealing limitations in our systems that have been there all along.
  - Increases the agility and reliability of our services as a side-effect!
- There can be further "migrations" down the road so as engineers we should remain vigilant against the pitfalls of ossification.

There are still many 'unknown' unknowns

- Impossible to list all bugs that could interfere with a post-quantum connection but we shared what we have seen so far...
  - There will be surprises too with post-quantum certificates deployment!

- Do not make assumptions about where issues may arise
  - Without observability, mitigation is difficult.
  - Origin server facing remediation efforts have been challenging.

Takeaways: Test, Expect Surprises, and Adapt

- Lesson #1: **Large PQ Keys and Signatures Cause Side-effects**
  - ***Advice***: *Expect the Unexpected and Start Testing ~~Early~~ Now!*
- Lesson #2: **Don't Assume Fixed Values for Extensible Fields**
  - ***Advice***: *Ensure flexibility by avoiding hard-coded assumptions.*
- Lesson #3: **Software may not Fully Adhere to the Standard**
  - ***Advice***: *Test your TLS implementations thoroughly – don't assume fallback compliance by default.*

References

- Further reading: state of the post-quantum Internet (2024).
- Check out Cloudflare developers docs for details about our deployment.
- Follow adoption on Cloudflare Radar.
- tldr.fail collects known PQ incompatibilities.
- Or, reach out: *ask-research@cloudflare.com*

QUESTIONS?