# NIST PQC Standards: Final Algorithms, Open Gaps, and Timeline Reality

Bas Westerbaan, Cloudflare Research

PQ Cyber Day, ICMC 2026, Arlington, April 20th

Bill suggested the title back in December.

# NIST PQC Standards: Final Algorithms, Open Gaps, and **Timeline Reality**

Bas Westerbaan, Cloudflare Research

PQ Cyber Day, ICMC 2026, Arlington, April 20th

# Prescient indeed

Google The Keyword

## Quantum frontiers may be closer than they appear

Mar 25, 2026 · 2 min read

Share

We're setting a timeline for post-quantum cryptography migration to 2029.

Heather Adkins
VP, Security Engineering

S

Sophie Schmieg
Senior Staff Cryptography Engine

TIME SUBSCRIBE

BUSINESS AI

## AI Helped Spark a Quantum Breakthrough. The World 'Is Not Prepared'

ars TECHNICA SECTIONS FORUM SIGN IN

THOSE WHO DON'T REMEMBER THE PAST...

## Recent advances push Big Tech closer to the Q-Day danger zone

Here's which players are winning the race to transition to post-quantum crypto.

DAN GOODIN – APR 17, 2026 1:00 PM | 48

## Cloudflare targets 2029 for full post-quantum security

2026-04-07

Bas Westerbaan

8 min read

# NIST PQC Standards: **Final Algorithms**, Open Gaps, and Timeline Reality

Bas Westerbaan, Cloudflare Research

PQ Cyber Day, ICMC 2026, Arlington, April 20th

# Final NIST algorithms

| Public key encryption | Signatures |
| --- | --- |
| ML-KEM ⭐ | ML-DSA ⭐<br>SLH-DSA<br>LMS / XMSS |

What about FN-DSA, HQC, and the *signatures onramp*?

“You go to war with the algorithms you have, not the ones you wish you had”

— Eric Rescorla, *how to manage a quantum computing emergency*, *2024*

# NIST PQC Standards: Final Algorithms, **Open Gaps**, and Timeline Reality

Bas Westerbaan, Cloudflare Research

PQ Cyber Day, ICMC 2026, Arlington, April 20th

# Technical gaps

These algorithms are larger on the wire; have different performance profiles (ML-* often faster than classical); and can’t be adapted in the same way for esoteric use cases.

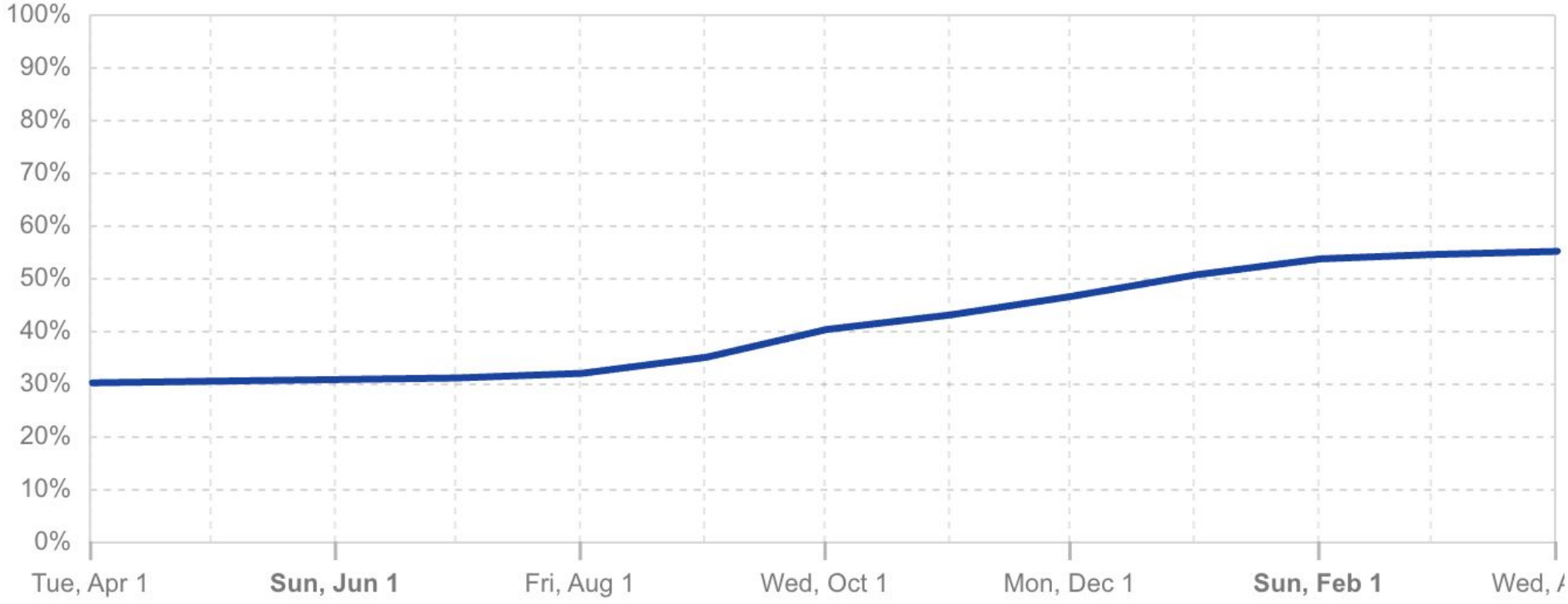
For most cases it works just fine, but the hard cases take a lot of time (90/10 rule.)

Approach is clear: try it *early* and adjust accordingly!

(That’d be my pick for a definition of crypto agility.)

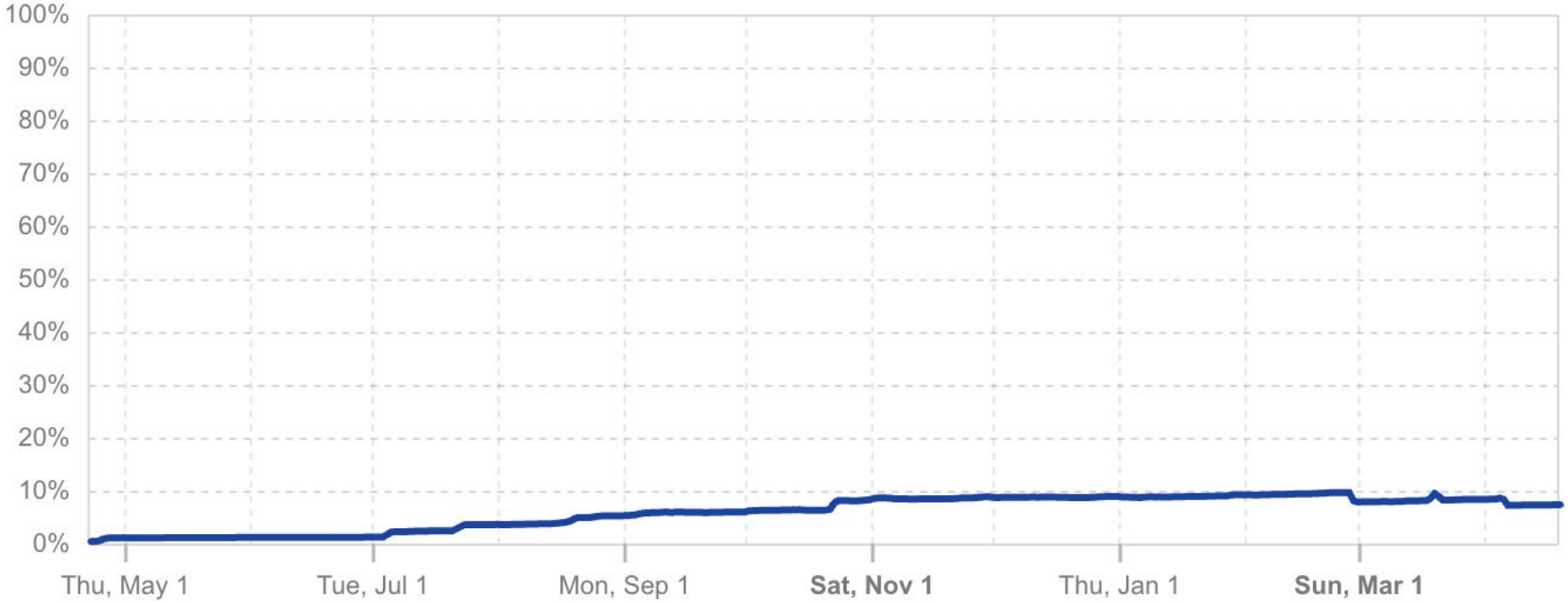
Doesn’t mean it’s easy or fast…

# Cloudflare visitor support for X25519MLKEM768




April 2025–2026 from radar.cloudflare.com/post-quantum

# Support at Cloudflare customer’s origins




April 2025–2026 from radar.cloudflare.com/post-quantum

# Post-quantum certificate deployment in WebPKI?

(nothing)

… but coming early 2027.

We’re in the experimental phase.

We’re all well aware and working on gaps in

- deployment
- hardware support
- protocol-level standards
- interoperability testing
- certification
- awareness

There are two other gaps I worry about in particular …

Most systems can’t upgrade all at once

# Protocol agility in theory

| Alice | Bob | Safe? |
|---|---|---|
| Vulnerable | Vulnerable | No |
| PQ supported<br>Alice: my job is done, just waiting for Bob! | Vulnerable | No |
| PQ supported | PQ supported<br>Bob: all good! | Yes |

# First gap: no common algorithm

| Alice | Bob | Safe? |
|---|---|---|
| Vulnerable | Vulnerable | No |
| PQ-A supported<br>Alice: my job is done, just waiting for Bob! | Vulnerable | No |
| PQ-A supported | PQ-B supported<br>Bob: my job is done, just waiting for Alice! | **No!** |

You need common PQ algorithms to be secure.

So, which?

# Which algorithm? (TLS)

- MLDSA44-RSA2048-PSS
- MLDSA44-RSA2048-PKCS15
- MLDSA44-Ed25519
- MLDSA44-ECDSA-P256
- MLDSA65-RSA3072-PSS
- MLDSA65-RSA3072-PKCS15
- MLDSA65-RSA4096-PSS
- MLDSA65-RSA4096-PKCS15
- MLDSA65-ECDSA-P256
- MLDSA65-ECDSA-P384
- MLDSA65-ECDSA-brainpoolP256r1
- MLDSA65-Ed25519
- MLDSA87-ECDSA-P384
- MLDSA87-ECDSA-brainpoolP384r1
- MLDSA87-Ed448
- MLDSA87-RSA3072-PSS
- MLDSA87-RSA4096-PSS
- MLDSA87-ECDSA-P521
- MLDSA44
- MLDSA65
- MLDSA87

- slh-dsa-sha2-128s
- slh-dsa-sha2-128f
- slh-dsa-sha2-192s
- slh-dsa-sha2-192f
- slh-dsa-sha2-256s
- slh-dsa-sha2-256f
- slh-dsa-shake-128s
- slh-dsa-shake-128f
- slh-dsa-shake-192s
- slh-dsa-shake-192f
- slh-dsa-shake-256s
- slh-dsa-shake-256f
- hash-slh-dsa-sha2-128s
- hash-slh-dsa-sha2-128f
- hash-slh-dsa-sha2-192s
- hash-slh-dsa-sha2-192f
- hash-slh-dsa-sha2-256s
- hash-slh-dsa-sha2-256f
- hash-slh-dsa-shake-128s
- hash-slh-dsa-shake-128f
- hash-slh-dsa-shake-192s
- hash-slh-dsa-shake-192f
- hash-slh-dsa-shake-256s
- hash-slh-dsa-shake-256f

Many servers struggle installing even *two* certificates.

# My thought process (1)

- MLDSA44-RSA2048-PSS
- MLDSA44-RSA2048-PKCS15
- MLDSA44-Ed25519
- MLDSA44-ECDSA-P256
- MLDSA65-RSA3072-PSS
- MLDSA65-RSA3072-PKCS15
- MLDSA65-RSA4096-PSS
- MLDSA65-RSA4096-PKCS15
- MLDSA65-ECDSA-P256
- MLDSA65-ECDSA-P384
- MLDSA65-ECDSA-brainpoolP256r1
- MLDSA65-Ed25519
- MLDSA87-ECDSA-P384
- MLDSA87-ECDSA-brainpoolP384r1
- MLDSA87-Ed448
- MLDSA87-RSA3072-PSS
- MLDSA87-RSA4096-PSS
- MLDSA87-ECDSA-P521
- MLDSA44
- MLDSA65
- MLDSA87

- ~~slh-dsa-sha2-128s~~
- ~~slh-dsa-sha2-128f~~
- ~~slh-dsa-sha2-192s~~
- ~~slh-dsa-sha2-192f~~
- ~~slh-dsa-sha2-256s~~
- ~~slh-dsa-sha2-256f~~
- ~~slh-dsa-shake-128s~~
- ~~slh-dsa-shake-128f~~
- ~~slh-dsa-shake-192s~~
- ~~slh-dsa-shake-192f~~
- ~~slh-dsa-shake-256s~~
- ~~slh-dsa-shake-256f~~
- ~~hash-slh-dsa-sha2-128s~~
- ~~hash-slh-dsa-sha2-128f~~
- ~~hash-slh-dsa-sha2-192s~~
- ~~hash-slh-dsa-sha2-192f~~
- ~~hash-slh-dsa-sha2-256s~~
- ~~hash-slh-dsa-sha2-256f~~
- ~~hash-slh-dsa-shake-128s~~
- ~~hash-slh-dsa-shake-128f~~
- ~~hash-slh-dsa-shake-192s~~
- ~~hash-slh-dsa-shake-192f~~
- ~~hash-slh-dsa-shake-256s~~
- ~~hash-slh-dsa-shake-256f~~

We need ML-KEM to be secure today, so relying on ML-DSA doesn’t add much of a security assumption.

# My thought process (2)

- MLDSA44-RSA2048-PSS
- MLDSA44-RSA2048-PKCS15
- MLDSA44-Ed25519
- MLDSA44-ECDSA-P256
- ~~**MLDSA65-RSA3072-PSS**~~
- ~~**MLDSA65-RSA3072-PKCS15**~~
- ~~**MLDSA65-RSA4096-PSS**~~
- ~~**MLDSA65-RSA4096-PKCS15**~~
- ~~**MLDSA65-ECDSA-P256**~~
- ~~**MLDSA65-ECDSA-P384**~~
- ~~**MLDSA65-ECDSA-brainpoolP256r1**~~
- ~~**MLDSA65-Ed25519**~~
- ~~**MLDSA87-ECDSA-P384**~~
- ~~**MLDSA87-ECDSA-brainpoolP384r1**~~
- ~~**MLDSA87-Ed448**~~
- ~~**MLDSA87-RSA3072-PSS**~~
- ~~**MLDSA87-RSA4096-PSS**~~
- ~~**MLDSA87-ECDSA-P521**~~
- MLDSA44
- ~~**MLDSA65**~~
- ~~**MLDSA87**~~

- ~~slh-dsa-sha2-128s~~
- ~~slh-dsa-sha2-128f~~
- ~~slh-dsa-sha2-192s~~
- ~~slh-dsa-sha2-192f~~
- ~~slh-dsa-sha2-256s~~
- ~~slh-dsa-sha2-256f~~
- ~~slh-dsa-shake-128s~~
- ~~slh-dsa-shake-128f~~
- ~~slh-dsa-shake-192s~~
- ~~slh-dsa-shake-192f~~
- ~~slh-dsa-shake-256s~~
- ~~slh-dsa-shake-256f~~
- ~~hash-slh-dsa-sha2-128s~~
- ~~hash-slh-dsa-sha2-128f~~
- ~~hash-slh-dsa-sha2-192s~~
- ~~hash-slh-dsa-sha2-192f~~
- ~~hash-slh-dsa-sha2-256s~~
- ~~hash-slh-dsa-sha2-256f~~
- ~~hash-slh-dsa-shake-128s~~
- ~~hash-slh-dsa-shake-128f~~
- ~~hash-slh-dsa-shake-192s~~
- ~~hash-slh-dsa-shake-192f~~
- ~~hash-slh-dsa-shake-256s~~
- ~~hash-slh-dsa-shake-256f~~

Ciphertext needs to be secure forever. Certificates can be rolled when they’re weakened. ML-DSA-44 has a very comfortable margin.

# My thought process (3)

- ~~**MLDSA44-RSA2048-PSS**~~
- ~~**MLDSA44-RSA2048-PKCS15**~~
- MLDSA44-Ed25519
- MLDSA44-ECDSA-P256
- ~~MLDSA65-RSA3072-PSS~~
- ~~MLDSA65-RSA3072-PKCS15~~
- ~~MLDSA65-RSA4096-PSS~~
- ~~MLDSA65-RSA4096-PKCS15~~
- ~~MLDSA65-ECDSA-P256~~
- ~~MLDSA65-ECDSA-P384~~
- ~~MLDSA65-ECDSA-brainpoolP256r1~~
- ~~MLDSA65-Ed25519~~
- ~~MLDSA87-ECDSA-P384~~
- ~~MLDSA87-ECDSA-brainpoolP384r1~~
- ~~MLDSA87-Ed448~~
- ~~MLDSA87-RSA3072-PSS~~
- ~~MLDSA87-RSA4096-PSS~~
- ~~MLDSA87-ECDSA-P521~~
- MLDSA44
- ~~MLDSA65~~
- ~~MLDSA87~~

- ~~slh-dsa-sha2-128s~~
- ~~slh-dsa-sha2-128f~~
- ~~slh-dsa-sha2-192s~~
- ~~slh-dsa-sha2-192f~~
- ~~slh-dsa-sha2-256s~~
- ~~slh-dsa-sha2-256f~~
- ~~slh-dsa-shake-128s~~
- ~~slh-dsa-shake-128f~~
- ~~slh-dsa-shake-192s~~
- ~~slh-dsa-shake-192f~~
- ~~slh-dsa-shake-256s~~
- ~~slh-dsa-shake-256f~~
- ~~hash-slh-dsa-sha2-128s~~
- ~~hash-slh-dsa-sha2-128f~~
- ~~hash-slh-dsa-sha2-192s~~
- ~~hash-slh-dsa-sha2-192f~~
- ~~hash-slh-dsa-sha2-256s~~
- ~~hash-slh-dsa-sha2-256f~~
- ~~hash-slh-dsa-shake-128s~~
- ~~hash-slh-dsa-shake-128f~~
- ~~hash-slh-dsa-shake-192s~~
- ~~hash-slh-dsa-shake-192f~~
- ~~hash-slh-dsa-shake-256s~~
- ~~hash-slh-dsa-shake-256f~~

RSA's only remaining benefit is fast verification time, which is negated by combining with ML-DSA-44's, which has verification times on the order of EC.

# My thought process (4)

- ~~MLDSA44-RSA2048-PSS~~
- ~~MLDSA44-RSA2048-PKCS15~~
- ~~**MLDSA44-Ed25519**~~
- MLDSA44-ECDSA-P256
- ~~MLDSA65-RSA3072-PSS~~
- ~~MLDSA65-RSA3072-PKCS15~~
- ~~MLDSA65-RSA4096-PSS~~
- ~~MLDSA65-RSA4096-PKCS15~~
- ~~MLDSA65-ECDSA-P256~~
- ~~MLDSA65-ECDSA-P384~~
- ~~MLDSA65-ECDSA-brainpoolP256r1~~
- ~~MLDSA65-Ed25519~~
- ~~MLDSA87-ECDSA-P384~~
- ~~MLDSA87-ECDSA-brainpoolP384r1~~
- ~~MLDSA87-Ed448~~
- ~~MLDSA87-RSA3072-PSS~~
- ~~MLDSA87-RSA4096-PSS~~
- ~~MLDSA87-ECDSA-P521~~
- MLDSA44
- ~~MLDSA65~~
- ~~MLDSA87~~

- ~~slh-dsa-sha2-128s~~
- ~~slh-dsa-sha2-128f~~
- ~~slh-dsa-sha2-192s~~
- ~~slh-dsa-sha2-192f~~
- ~~slh-dsa-sha2-256s~~
- ~~slh-dsa-sha2-256f~~
- ~~slh-dsa-shake-128s~~
- ~~slh-dsa-shake-128f~~
- ~~slh-dsa-shake-192s~~
- ~~slh-dsa-shake-192f~~
- ~~slh-dsa-shake-256s~~
- ~~slh-dsa-shake-256f~~
- ~~hash-slh-dsa-sha2-128s~~
- ~~hash-slh-dsa-sha2-128f~~
- ~~hash-slh-dsa-sha2-192s~~
- ~~hash-slh-dsa-sha2-192f~~
- ~~hash-slh-dsa-sha2-256s~~
- ~~hash-slh-dsa-sha2-256f~~
- ~~hash-slh-dsa-shake-128s~~
- ~~hash-slh-dsa-shake-128f~~
- ~~hash-slh-dsa-shake-192s~~
- ~~hash-slh-dsa-shake-192f~~
- ~~hash-slh-dsa-shake-256s~~
- ~~hash-slh-dsa-shake-256f~~

Even though Ed25519 has advantages, it's regrettably not used in the WebPKI today. Let's not impose unnecessary extra work.

# My thought process (5)

That leaves MLDSA44-ECDSA-P256 and ML-DSA-44.

I prefer the hybrid, as it can be deployed without *having to convince anyone*.

Unfortunately it is clear there is no consensus that this particular hybrid should be the go-to like X25519MLKEM768 is for key agreement in the WebPKI.

For maximum compatibility, as it stands, ML-DSA-44 is the safest bet.

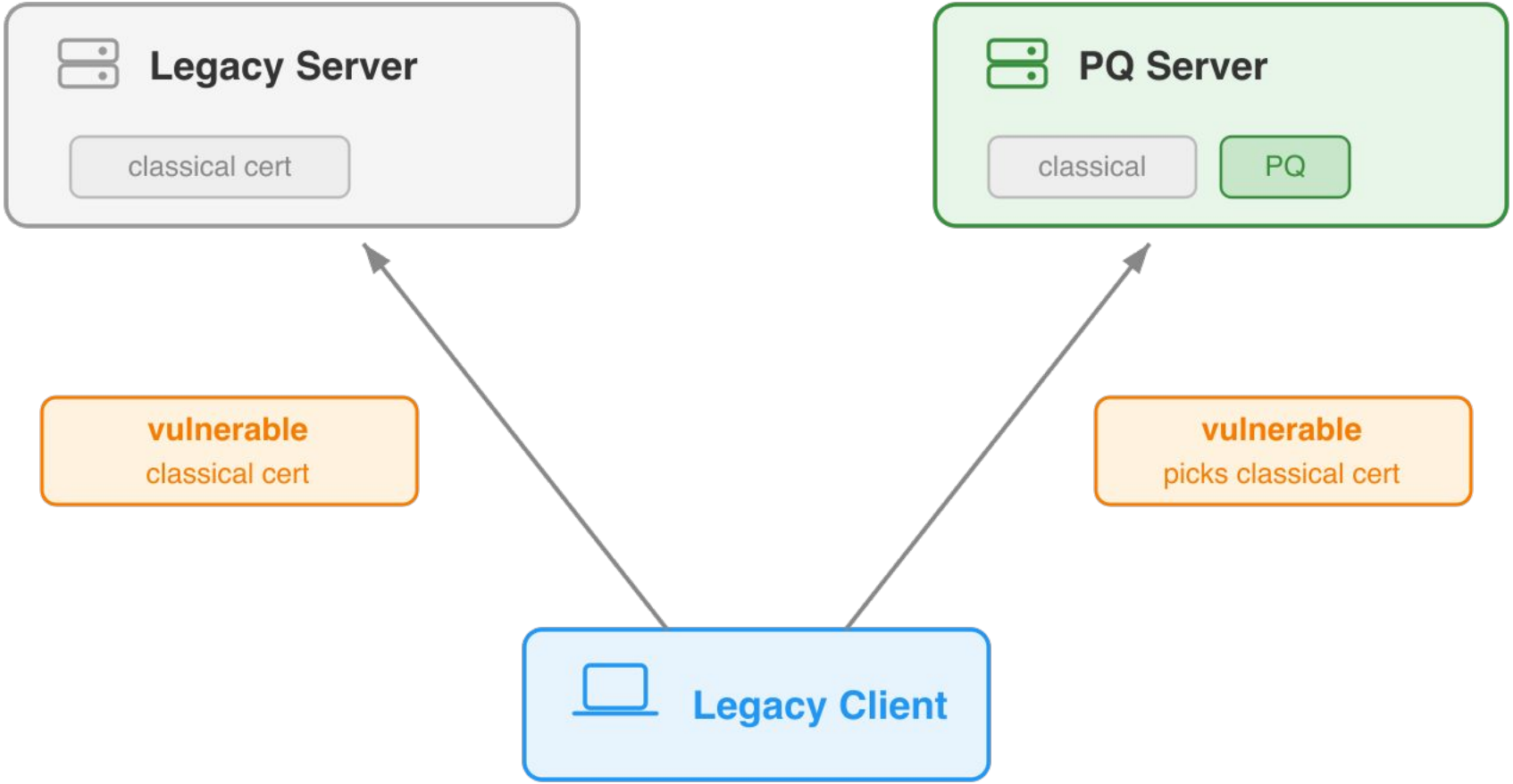
# Protocol agility in theory

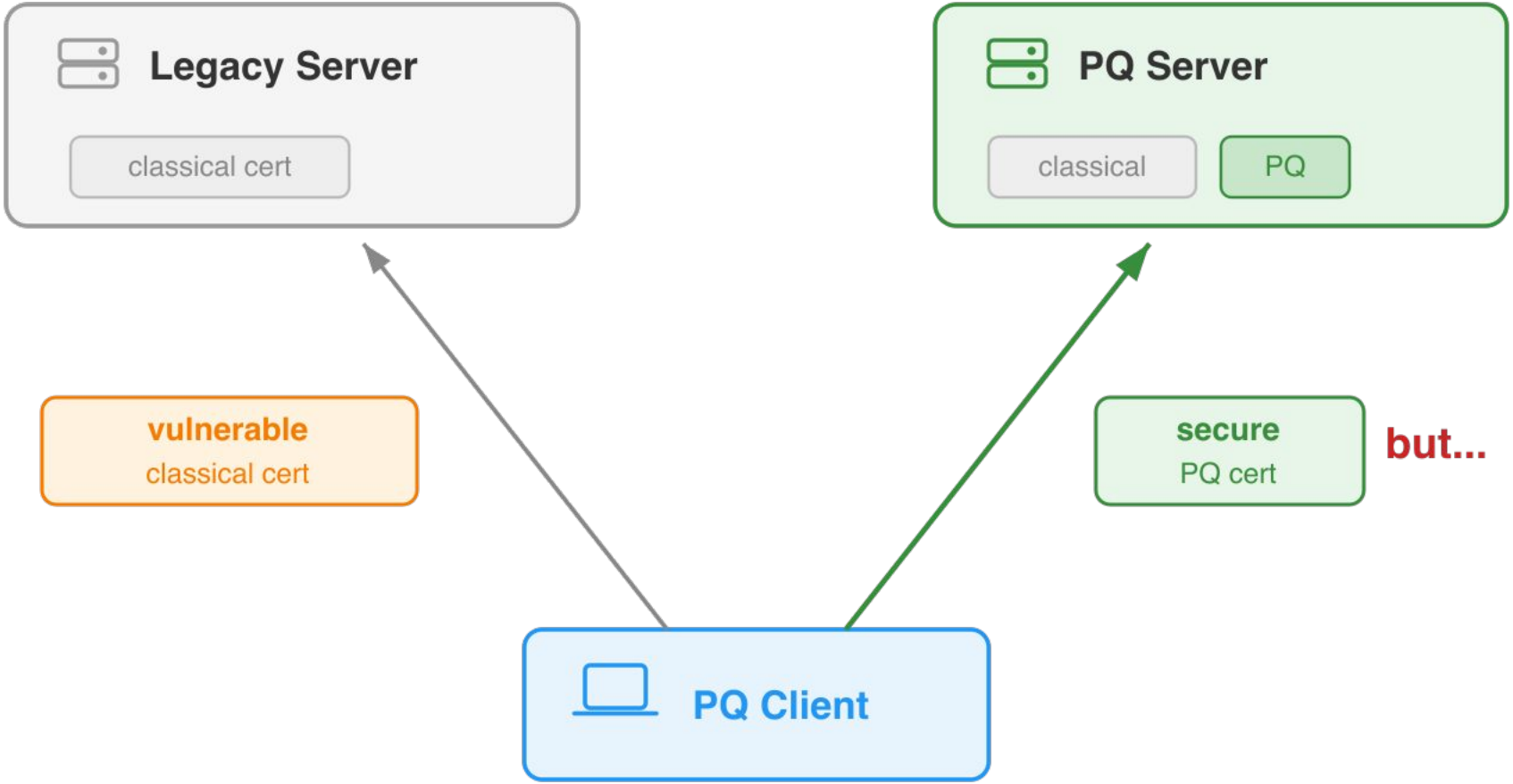
| Alice | Bob | Safe? |
|---|---|---|
| Vulnerable | Vulnerable | No |
| PQ supported<br>Alice: my job is done, just waiting for Bob! | Vulnerable | No |
| PQ supported | PQ supported<br>Bob: all good! | Yes |

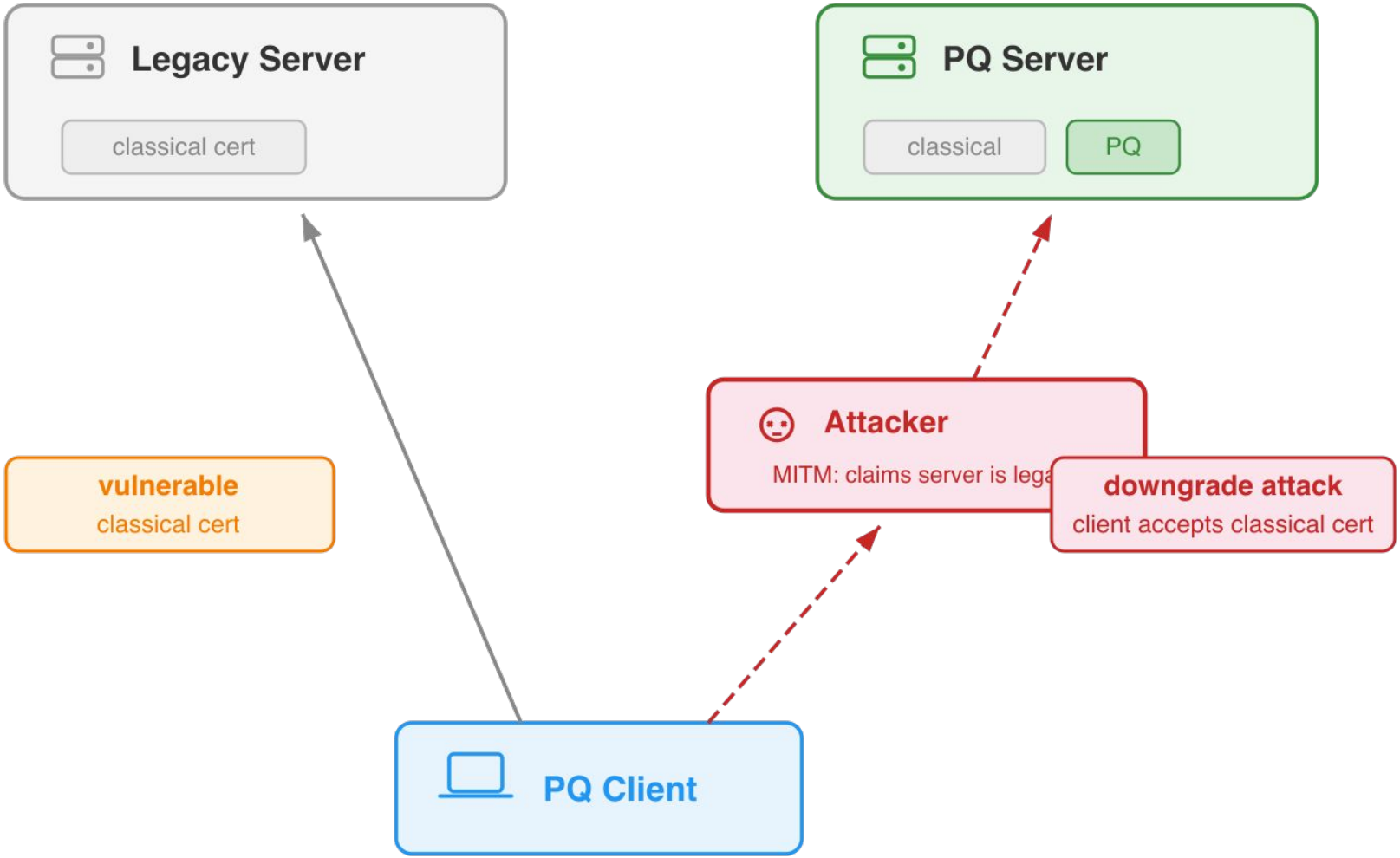
# Second gap: downgrades

| Alice | Bob | Safe? |
|---|---|---|
| Vulnerable | Vulnerable | No |
| PQ & vulnerable | Vulnerable | No |
| PQ & vulnerable | PQ & vulnerable | **No!** |

Turning off support for vulnerable cryptography on client and server prevents downgrades, but in most large systems this is simply impossible.


Legacy Server
classical cert
PQ Server
classical
PQ
vulnerable
classical cert
vulnerable
picks classical cert
Legacy Client


Legacy Server
classical cert
PQ Server
classical
PQ
vulnerable
classical cert
secure
PQ cert
but...
PQ Client


Legacy Server
classical cert
PQ Server
classical
PQ
Attacker
MITM: claims server is lega
vulnerable
classical cert
downgrade attack
client accepts classical cert
PQ Client

# Preventing downgrades with PQ-HSTS

Server can signal to a client that it will have a post-quantum certificate going forward. This is similar to how today HSTS signals that a server will support TLS.

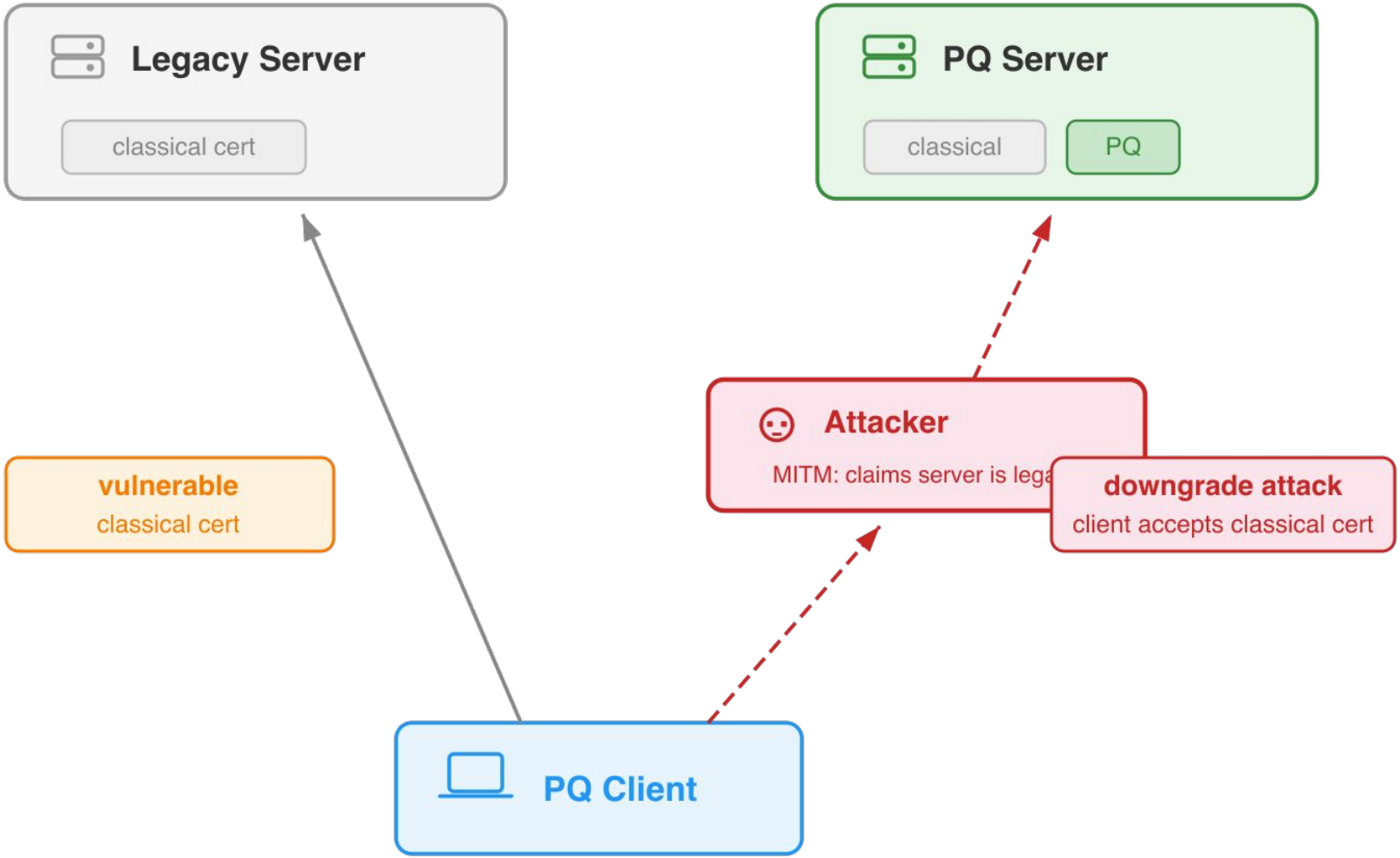
Downside is that this doesn’t work for first connections and can cause surprise breakages when switching to a provider that doesn’t support PQ certs.

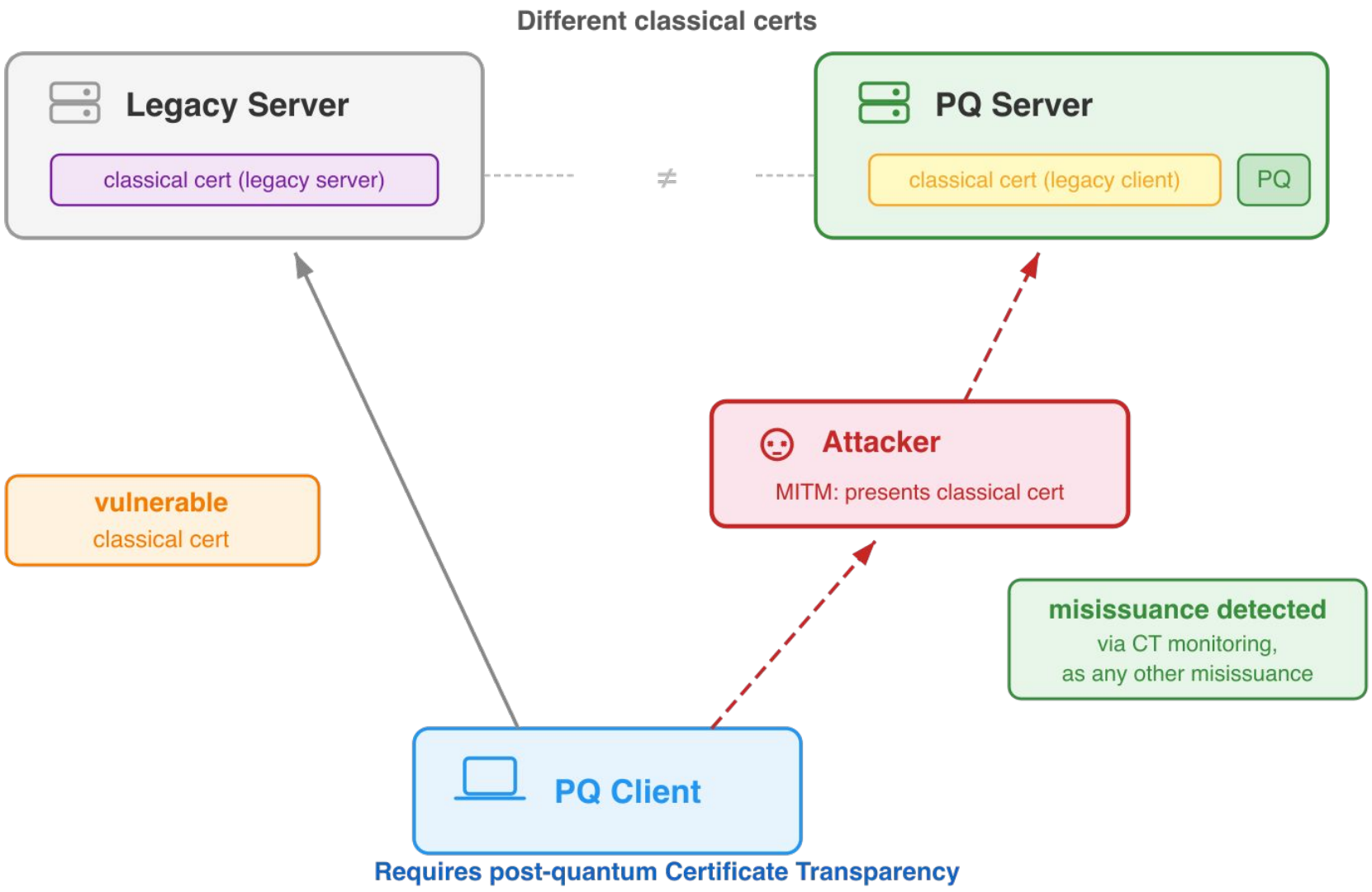
# Preventing downgrades using transparency

Key idea: distinguish between the certificate chain for the legacy server and legacy client.

Then we can detect issuance of a *downgrade certificate* for legacy servers, using Certificate Transparency.

This of course requires the transparency to be post-quantum, like with Merkle Tree Certificates. See also Chrome's authentication roadmap.


Legacy Server
classical cert
PQ Server
classical
PQ
Attacker
MITM: claims server is lega
vulnerable
classical cert
downgrade attack
client accepts classical cert
PQ Client


Different classical certs
Legacy Server
classical cert (legacy server)
≠
PQ Server
classical cert (legacy client)
PQ
Attacker
MITM: presents classical cert
vulnerable
classical cert
misissuance detected
via CT monitoring,
as any other misissuance
PQ Client
Requires post-quantum Certificate Transparency

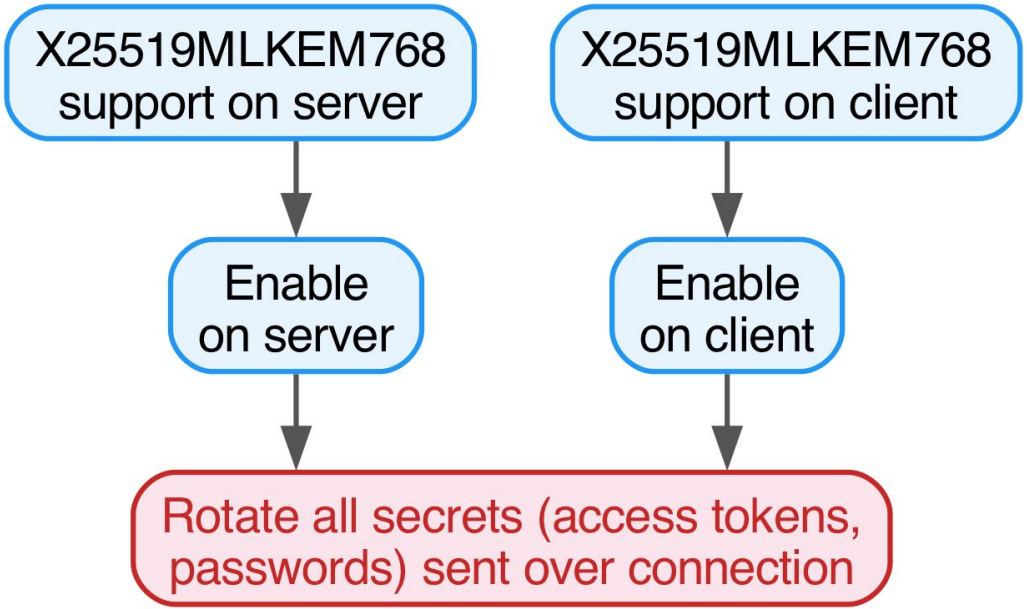
# In summary

- We have the NIST standards. They'll have to do.
- Agility: you surface technical gaps by trying.
- The more federated an ecosystem is, the more important it is to pick a small set of common algorithms.
- You're not done until you've dealt with downgrades!

# Thank you, questions?

blog.cloudflare.com/post-quantum-roadmap

# Post-quantum key agreement is “easy”



# Post-quantum certs are a tad more involved