

MTC support is only step 1

How Chrome will protect *all* users from quantum attackers

Joe DeBlasio, Google Chrome

(with thanks to David Benjamin, whose talk I've purloined)

Overview

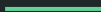
Why we're doing this

MTCs as an option

That's not good enough

What we're doing instead

Finally: Chrome's timelines



[Home](#) > [Innovation & AI](#) > [Technology](#) > [Safety & Security](#)

Quantum frontiers may be closer than they appear

Mar 25, 2026

1 min read

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

Heather Adkins
VP, Security Engineering

Sophie Schmieg
Senior Staff Cryptography
Engineer

 Share



Home > Innovation & AI > Technology > Safety & Security

Quantum frontiers may be closer than they appear

Mar 25, 2026
1 min read

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

"Google's introducing a 2029 timeline to secure the quantum era with post-quantum cryptography (PQC) migration."

Heather Adkins
VP, Security En

Sophie Schmieg
Product Lead

Share



Cultivating a robust and efficient quantum-safe HTTPS

Feb 27, 2026

4 min read

Chrome Secure Web Networking Team

 Share

Today we're announcing a new program in Chrome to make HTTPS certificates secure against quantum computers. The Internet Engineering Task Force (IETF) recently created a working group, [PKI, Logs, And Tree Signatures](#) ("PLANTS"), aiming to address the performance and bandwidth challenges that the increased size of [quantum-resistant cryptography](#) introduces into TLS connections requiring [Certificate Transparency](#) (CT). We recently [shared](#) our call to action to secure quantum computing and have written about challenges introduced by quantum-resistant cryptography and some of the steps we've taken to address them in earlier [blog posts](#).

Cultivating a robust and efficient quantum-safe HTTPS

Feb 27, 2026

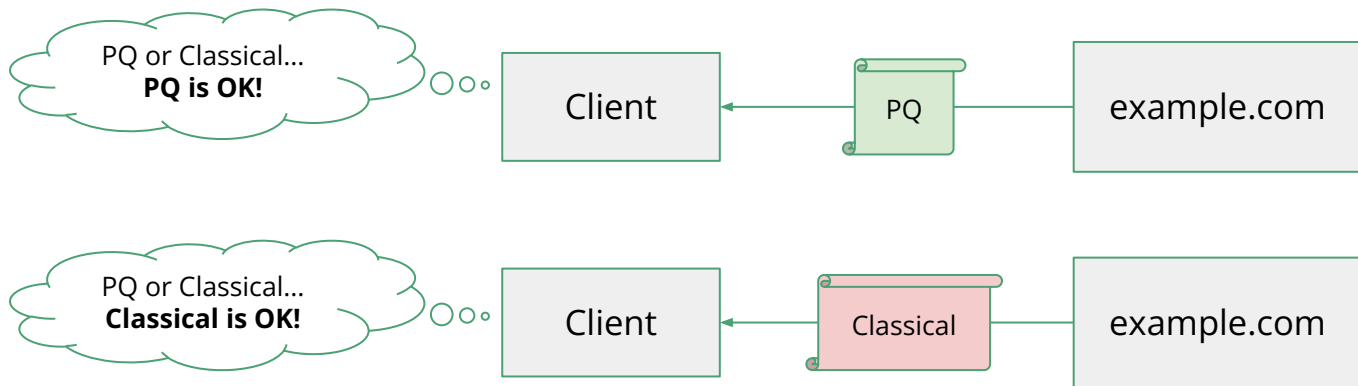
4 min read

- **Phase 2 (Q1 2027): Bootstrapping production MTCs with a few CAs.**
- **Phase 3 (Q3 2027): Production MTCs from any qualifying CA** [Share](#)

Today we're announcing a new program in Chrome to make HTTPS certificates secure against quantum computers. The Internet Engineering Task Force (IETF) recently created a working group, [PKI, Logs, And Tree Signatures](#) ("PLANTS"), aiming to address the performance and bandwidth challenges that the increased size of [quantum-resistant cryptography](#) introduces into TLS connections requiring [Certificate Transparency](#) (CT). We recently [shared](#) our call to action to secure quantum computing and have written about challenges introduced by quantum-resistant cryptography and some of the steps we've taken to address them in earlier [blog posts](#).

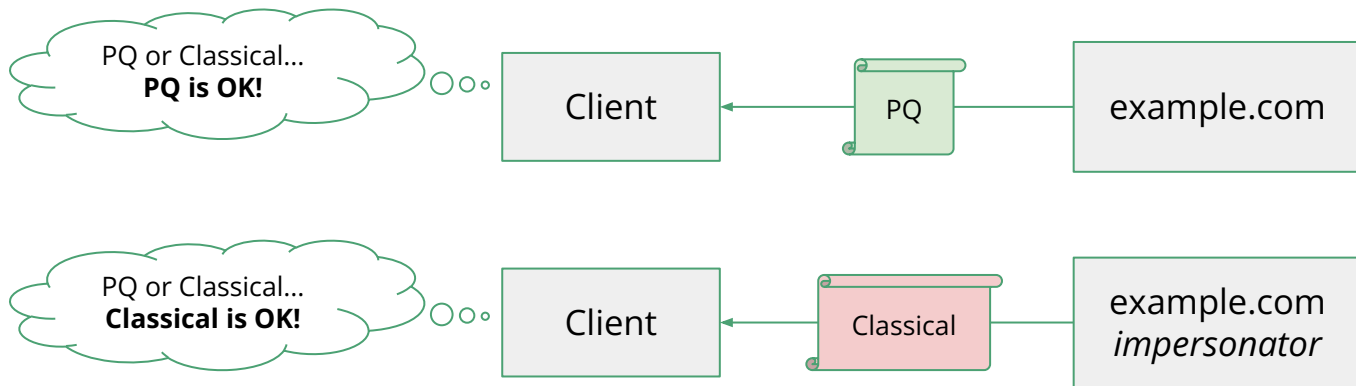
In 2027, Chrome will *support* MTCs, but not require them

i.e. Chrome will accept a TLS connection with
either a valid classical certificate *or* an MTC.



The attacker chooses the certificate

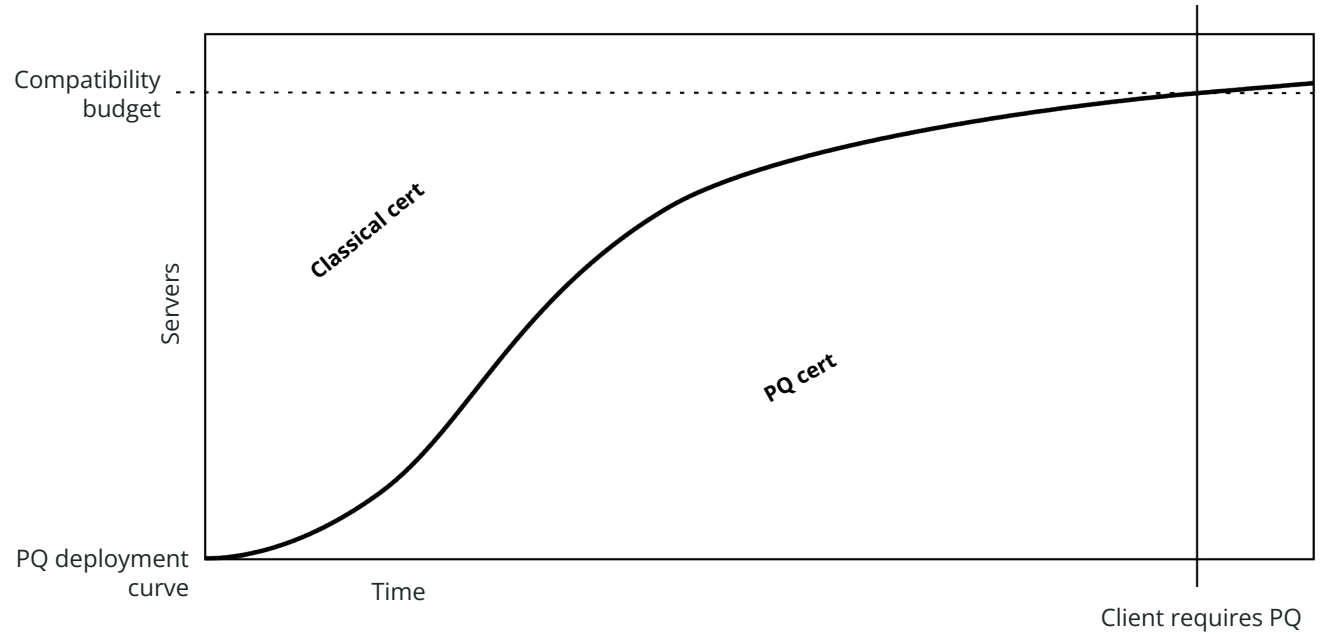
If Chrome trusts *any* classical certs for a website, the attacker just forges a classical certificate and pretends the server doesn't support PQ.



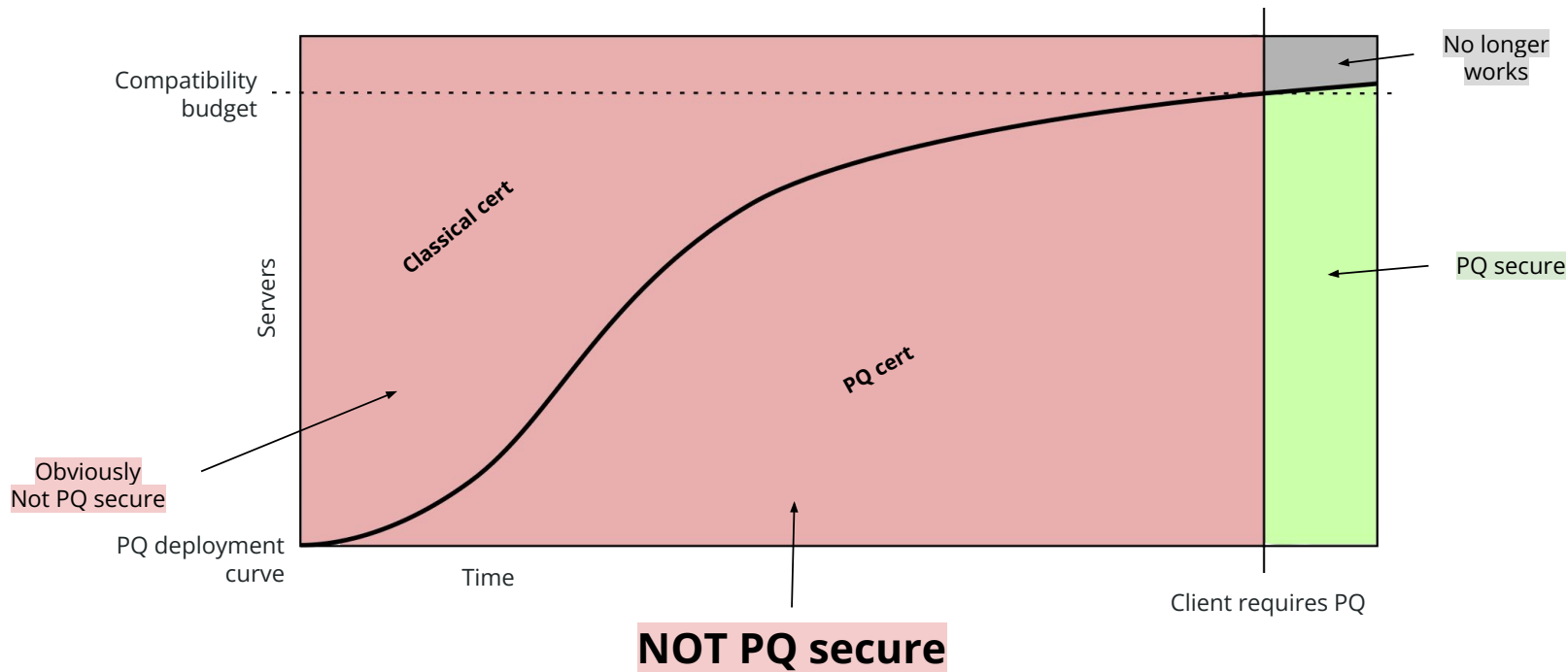
We need a strategy for ***downgrade protection***

The S-Curve of Misery

PQ deployment curve



Downgrade is possible until client *requires* PQ



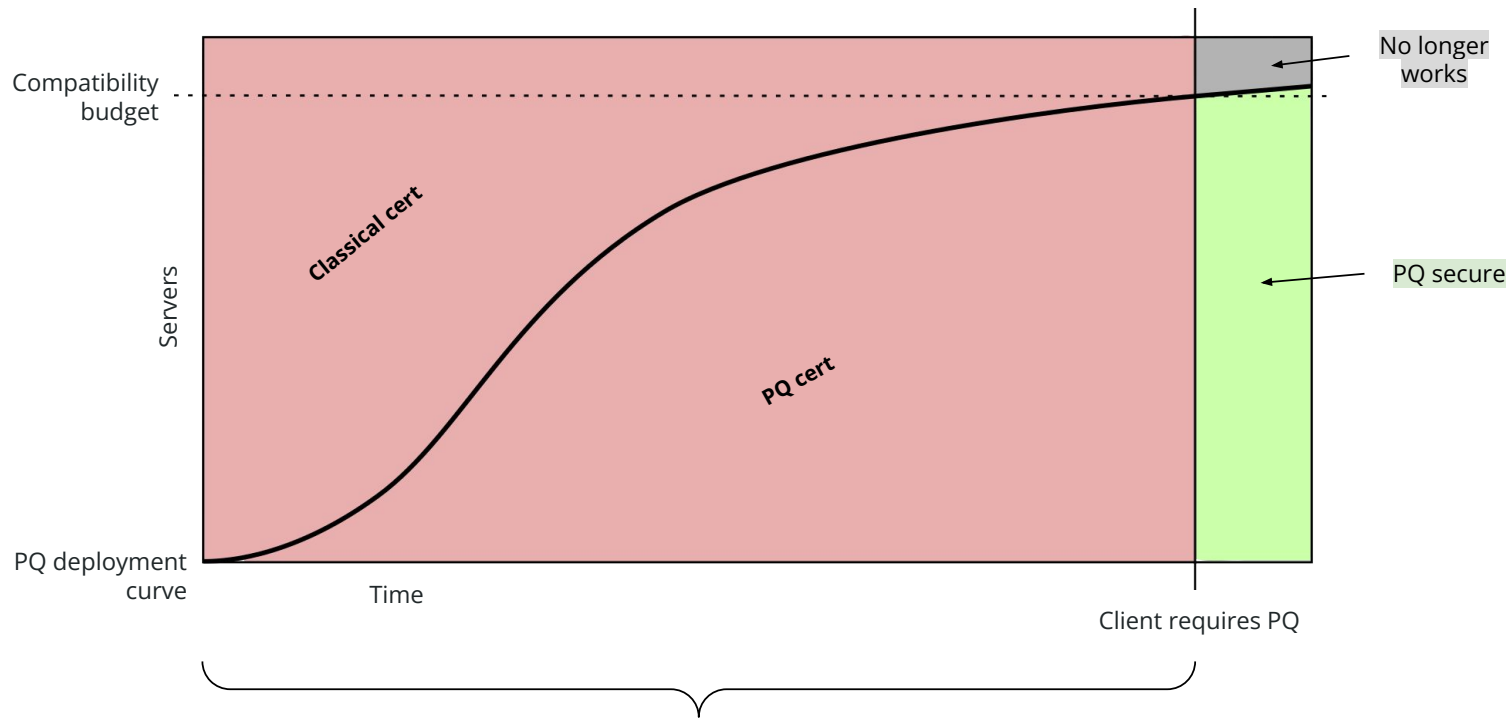
Legacy systems are important and aren't going away

Old systems *will never understand PQ crypto*. **We can't fix them.**

The internet is big. Lots of sites will not be updated promptly.

These sites are important! We can't just refuse to connect to them.

This could take a while... 🥲



This may take decades on the Web

PQ HSTS?

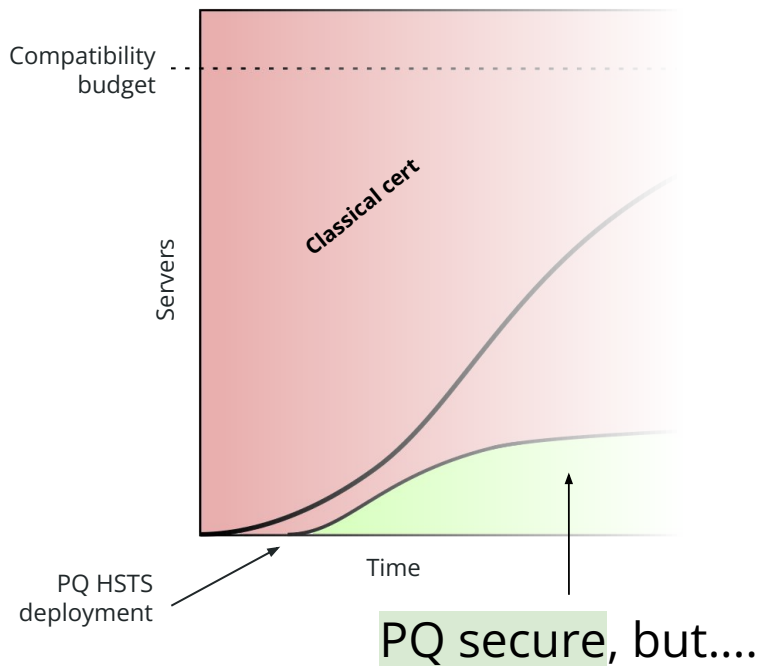
Client must *securely* know if **example.com** was *supposed to* have a PQ cert

...What if we just had a **list** of PQ-capable servers?

Mimic HTTP Strict Transport Security?

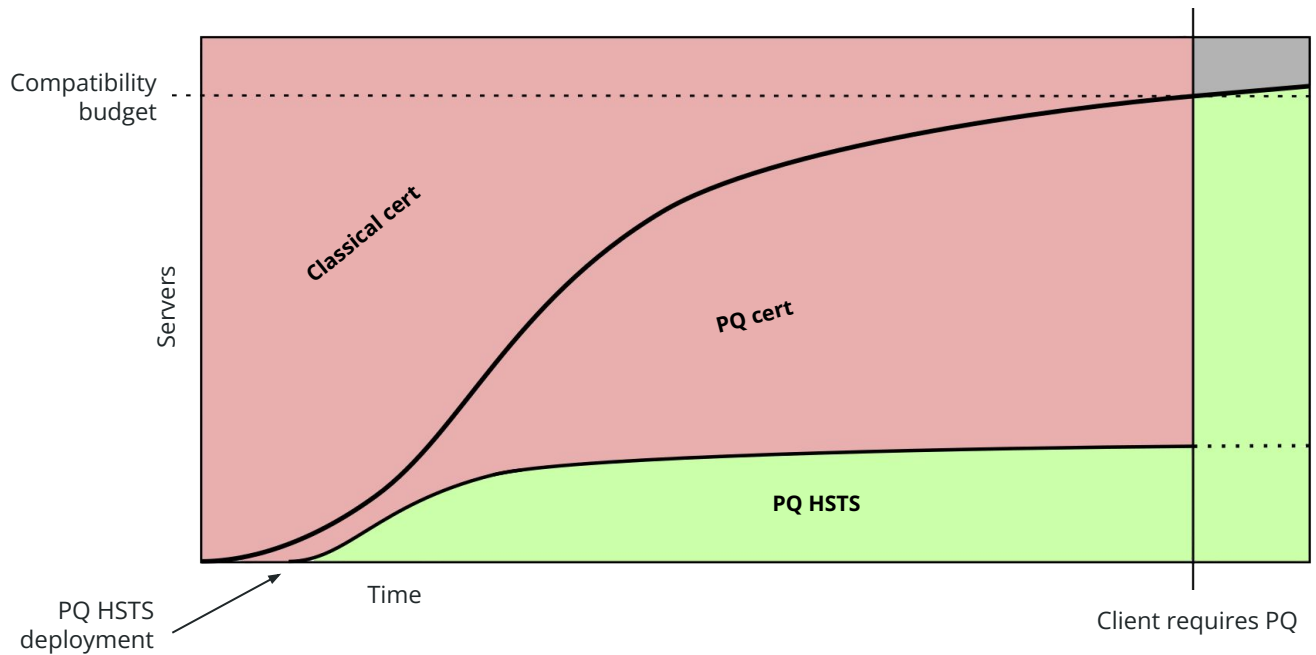
- Dynamic: **Require-Post-Quantum: max-age=31536000; includeSubDomains**
- Static: Preload a list of them somewhere

Limitations of PQ HSTS



- Dynamic requires app-specific state management, can't protect first visit...
- ...but static lists hit scaling limits
- Deployment risks for site operators
- Must be separate opt-in
- includeSubDomains even riskier
- TLS interception proxy troubles
- Difficult to deploy in non-browser clients
- Stateful mechanisms act as supercookies
- ...

Limitations of PQ HSTS



Another idea... split the transition in two

There are *two* keys involved in certificates

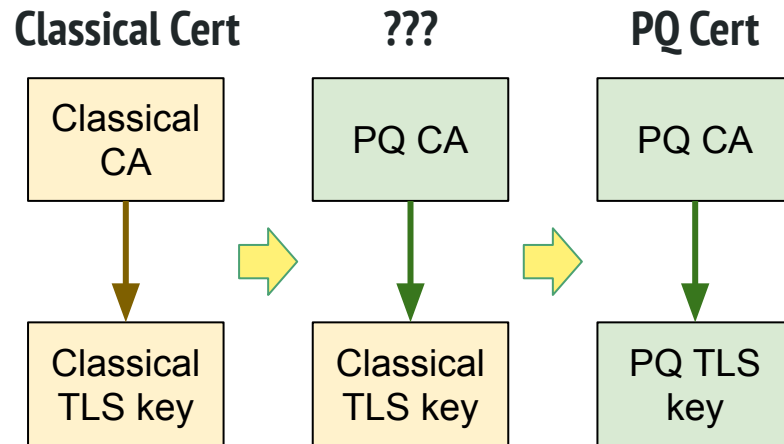
- Server TLS key
- Issuing CA's key

Switching TLS key requires **per-server** work

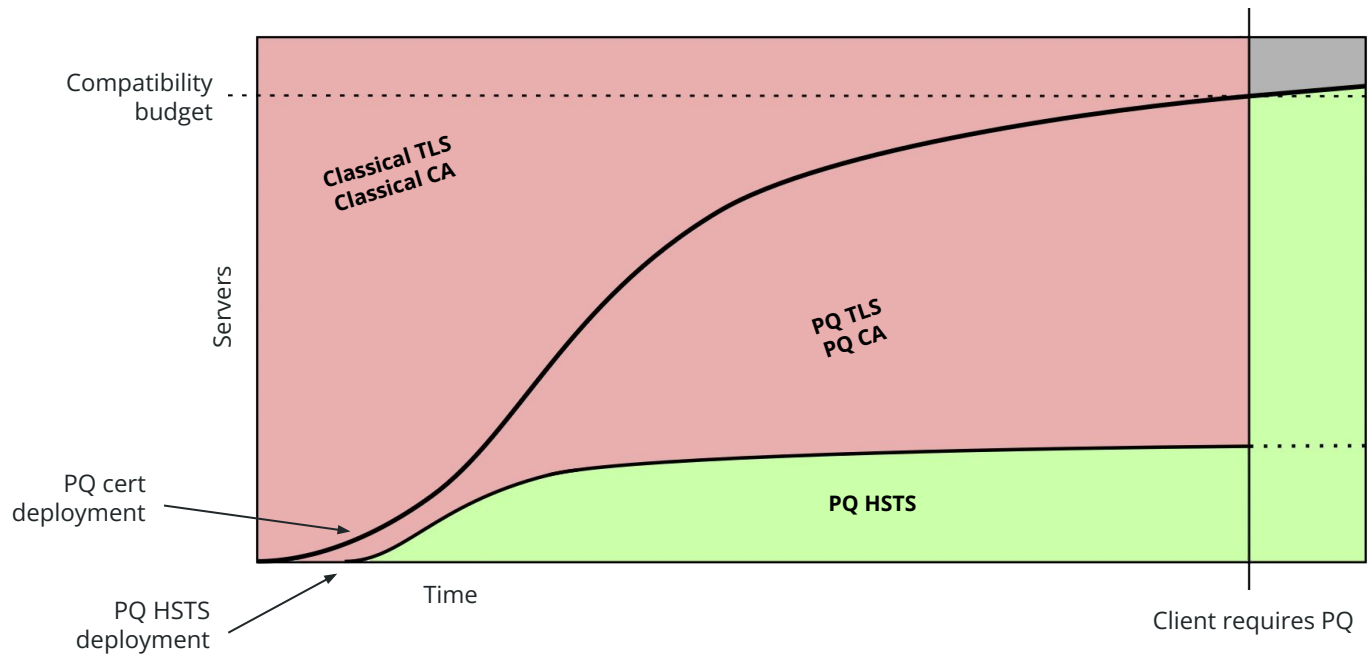
- Update software
- Maintain both traditional and PQ TLS keys

Could CA key be switched with **per-CA** work?

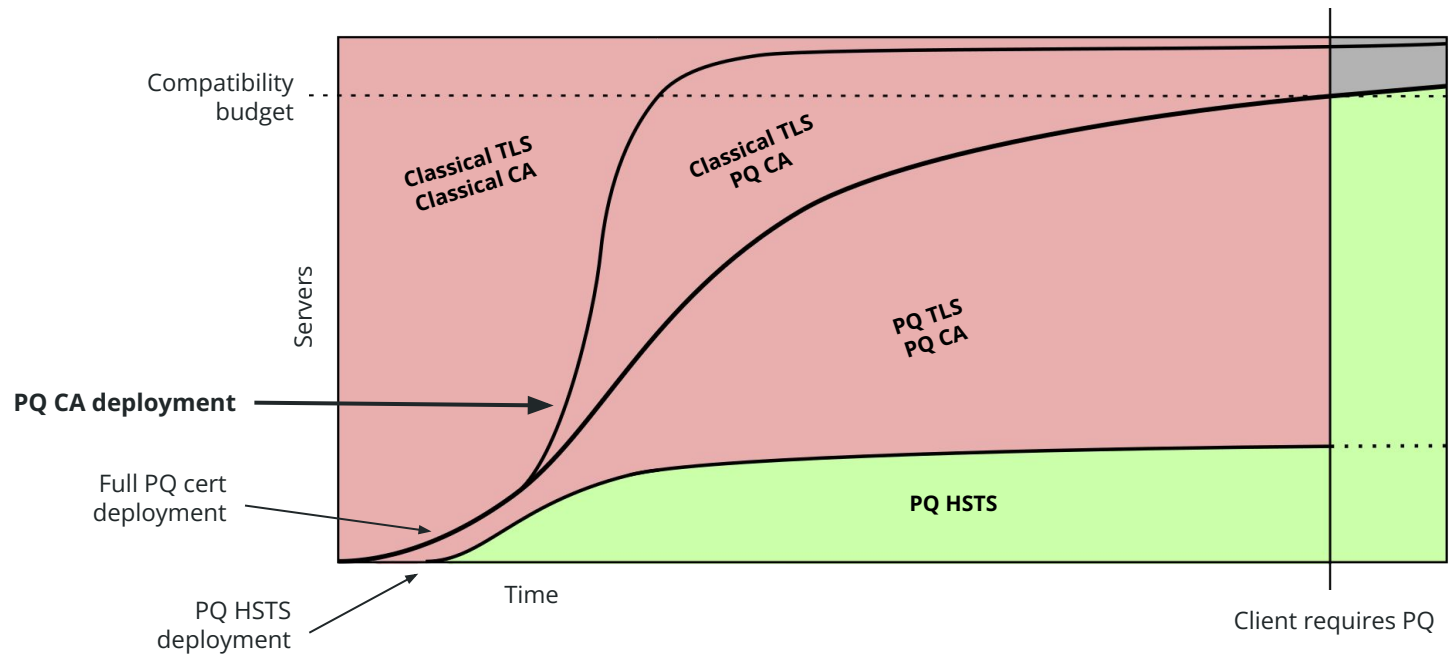
- Servers *already* renew certificates
- What if we automatically gave them certificates signed by new CAs?



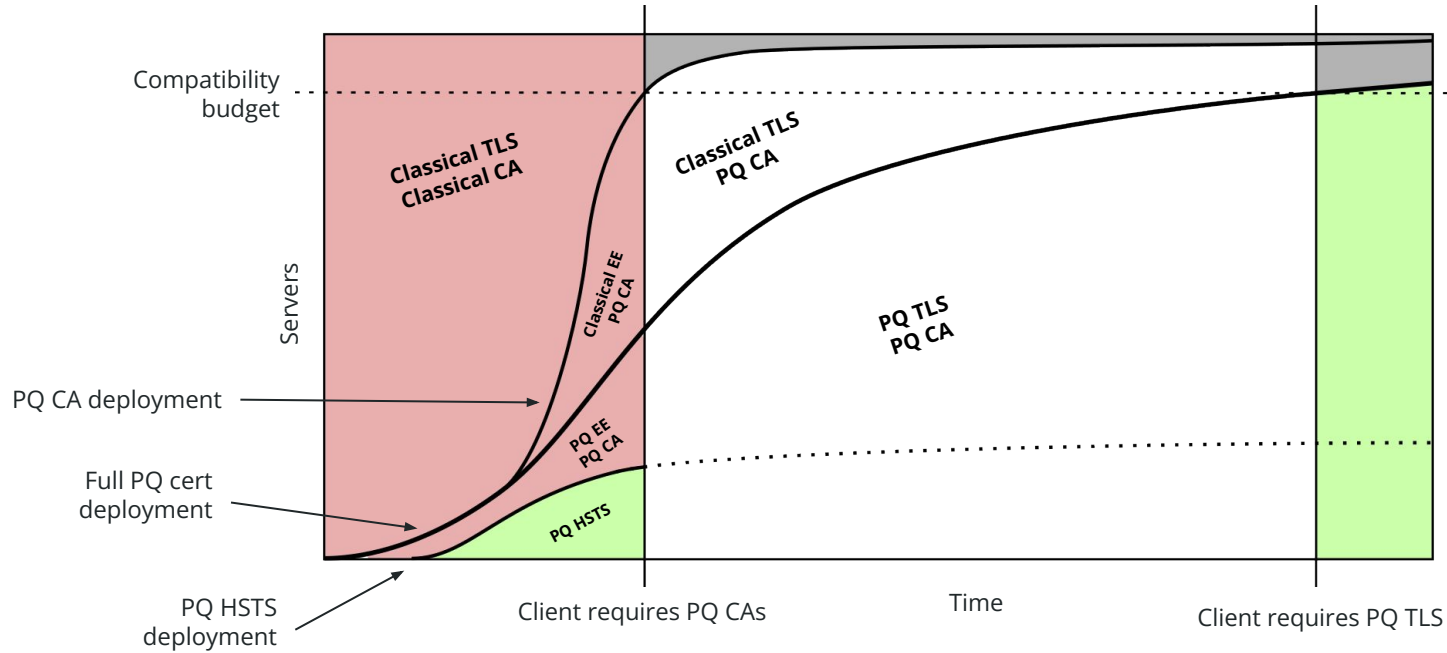
Back to our picture...



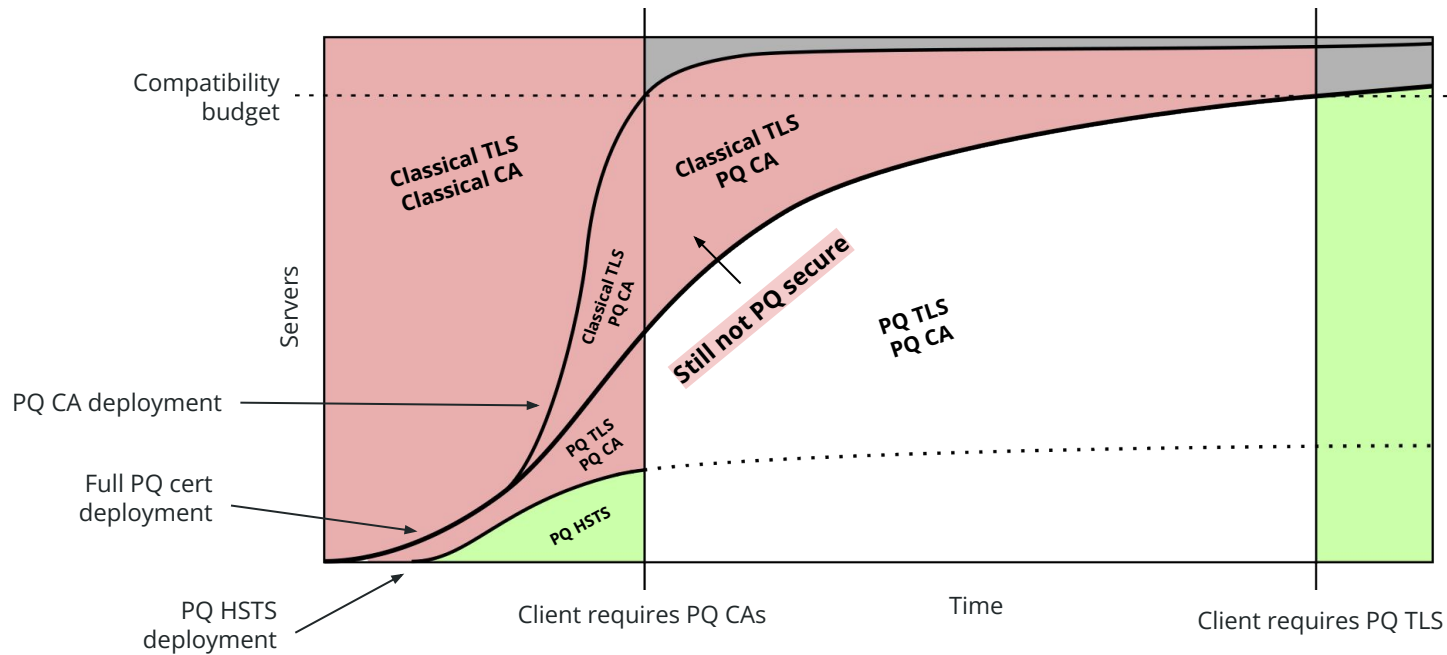
If we can automatically move to PQ CAs...



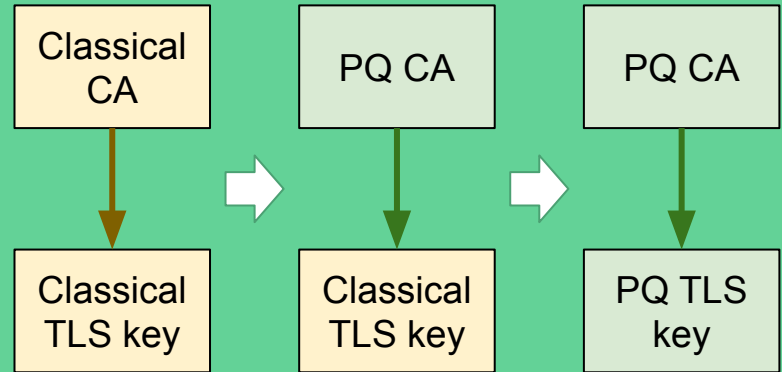
...the client can require PQ CAs sooner



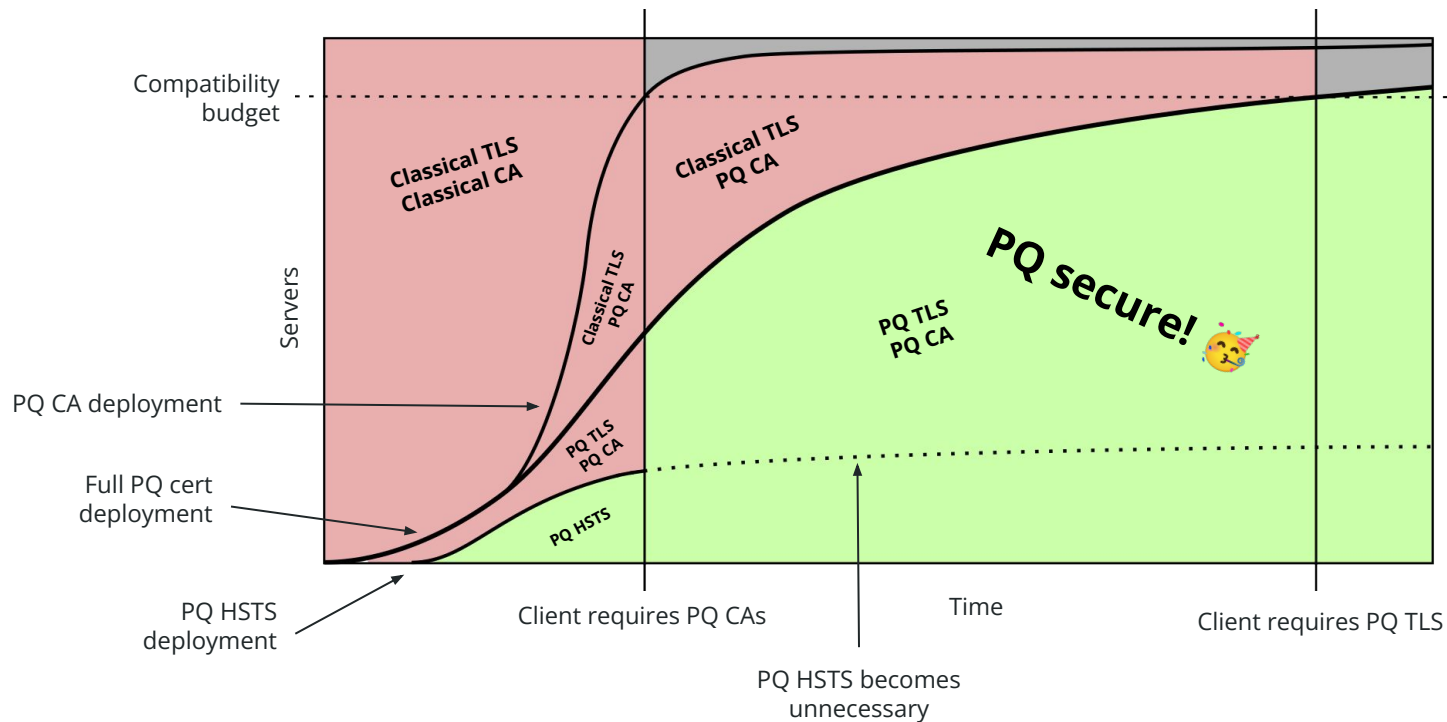
Classical TLS keys are still weak...



A PQ CA signing
a classical key is
a **PQ-secure**
signal that *this*
service is **not yet**
PQ-secure



...so PQ TLS is no longer vulnerable to classical CAs



Our plan to beat the S-curve, draft-00

1. Clients deploy PQ support
2. Servers gradually roll out PQ→PQ certs, but get *no security benefit*
3. Maybe deploy PQ HSTS as stopgap
4. CAs reconfigure ACME endpoints to issue from PQ CA
5. Classical→Classical certs become PQ→Classical on renewal
6. Now all certs are PQ→X, clients require PQ CAs
7. **All PQ→PQ servers are now secure**
8. As remaining servers move to PQ→PQ, they immediately become secure

~~Our plan to beat the S-curve, draft-00~~

OK wait a minute...

1. Clients deploy PQ support
2. Servers gradually roll out PQ→PQ certs, but get *no security benefit*
3. Maybe deploy PQ HSTS as stopgap
4. CAs reconfigure ACME endpoints to issue from PQ CA
5. Classical→Classical certs become PQ→Classical on renewal
6. Now all certs are PQ→X, clients require PQ CAs
7. **All PQ→PQ servers are now secure**
8. As remaining servers move to PQ→PQ, they immediately become secure

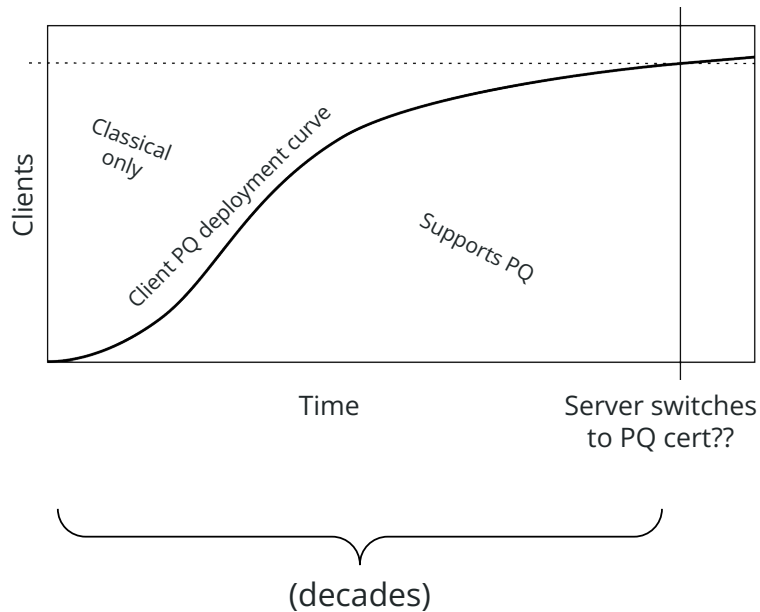
...this is not going to work!

Servers have to support old clients, too

Servers have their own S-curve problem

- Switching to a PQ CA breaks old clients
- Keeping a classical CA holds back new clients

**If switching to a PQ CA breaks stuff,
then we can't "just" move everyone to a PQ CA**



Answer: deploy multiple certificates

Let TLS negotiate what certificate the client wants

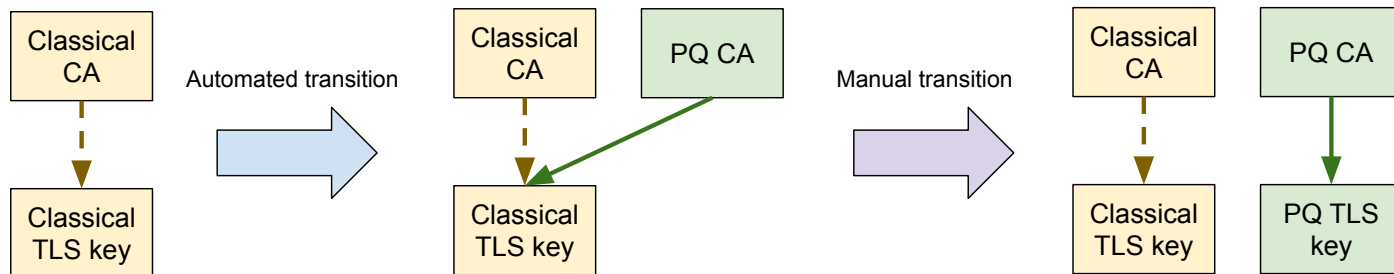
Have one certificate for legacy clients and one for PQ clients

Automatically issue both certificates with the same TLS key.

Certificates

For PQ clients 

For legacy clients 



But old servers can only handle *one* certificate!

Until now, it has sufficed for servers to only have a single certificate, and know nothing about negotiation

By definition we can't update these servers to negotiate certificates.

But! We can solve this with a one-time hack for legacy servers.

Turducken certificates



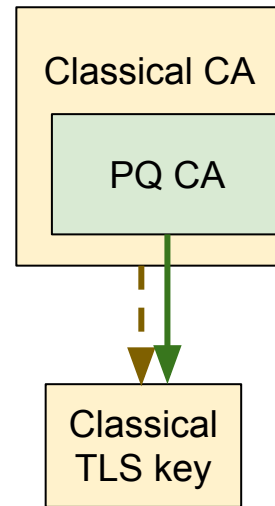
We get *one* certificate slot in the legacy server,
and it must be compatible with *unmodified* legacy clients!

So give them a legit traditional certificate...

...but **sneak a PQ cert inside a legacy cert's extension**

- Old clients won't understand, but they will ignore it.
- It's still *worthless* against a quantum attacker

A **one-time** tool to enable old servers to support multiple certificates.



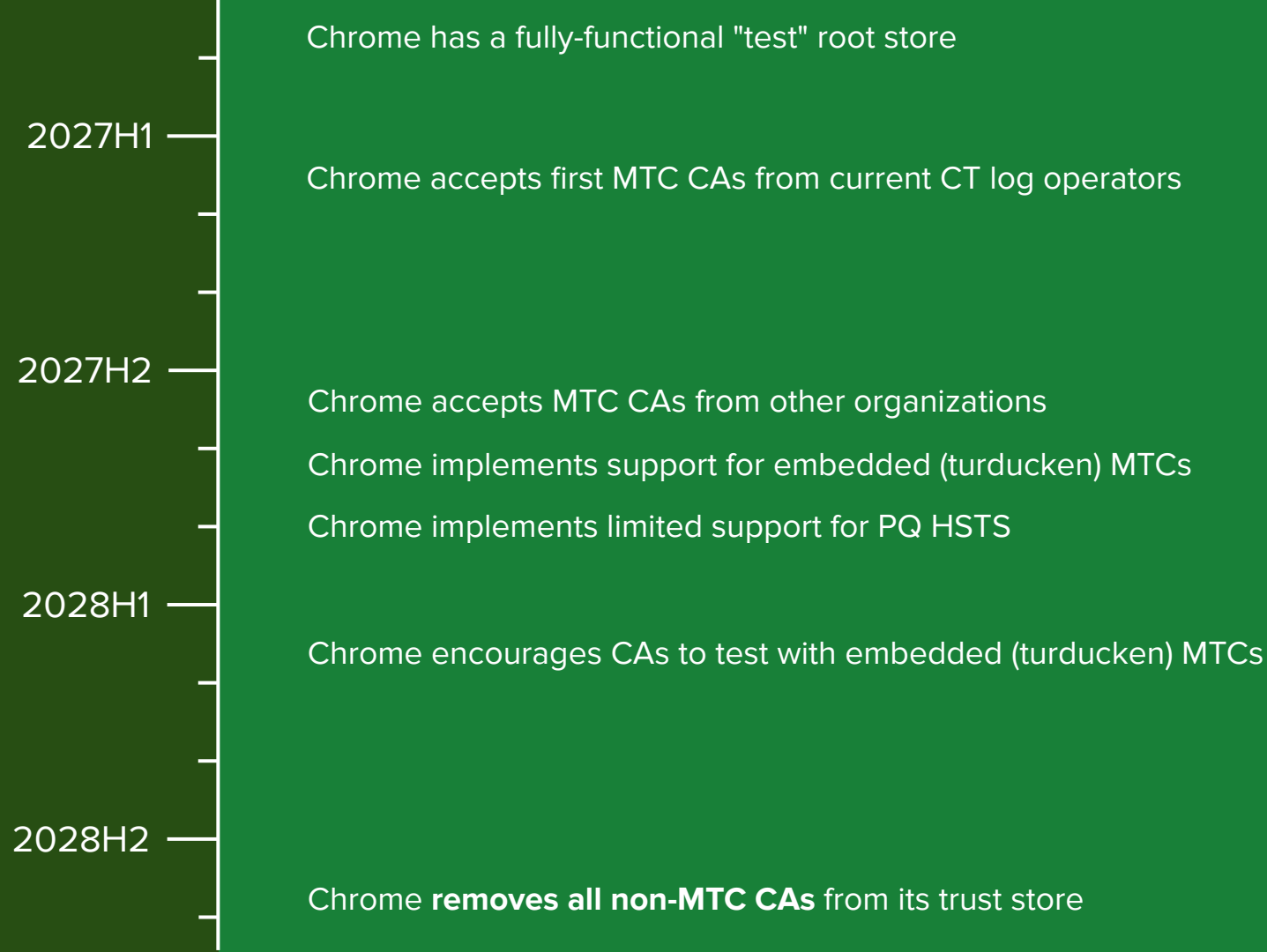
Our plan to beat the S-curve, draft-01

1. Clients deploy PQ support and “turducken certificates”
2. Servers gradually roll out PQ→PQ certs, but get *no security benefit*
3. Maybe deploy PQ HSTS as stopgap
4. CAs issue turducken certs for classical keys by default
5. Now all certs are PQ→X, clients require PQ CAs
6. **All PQ→PQ servers are now secure**
7. As remaining servers convert to PQ→PQ, they *stop using turducken* and become secure

So overall, to get all eligible sites safe as soon as possible, Chrome must...

1. Make MTCs available (in progress)
2. Let high-value sites opt-in to PQ downgrade protections with list *temporarily*
3. Remove all non-PQ roots

Turducken certs make step (3) possible.



Questions?

Joe DeBlasio
jdeblasio@chromium.org