Let’s stay encrypted—

Rethinking the WebPKI for the post-quantum age with

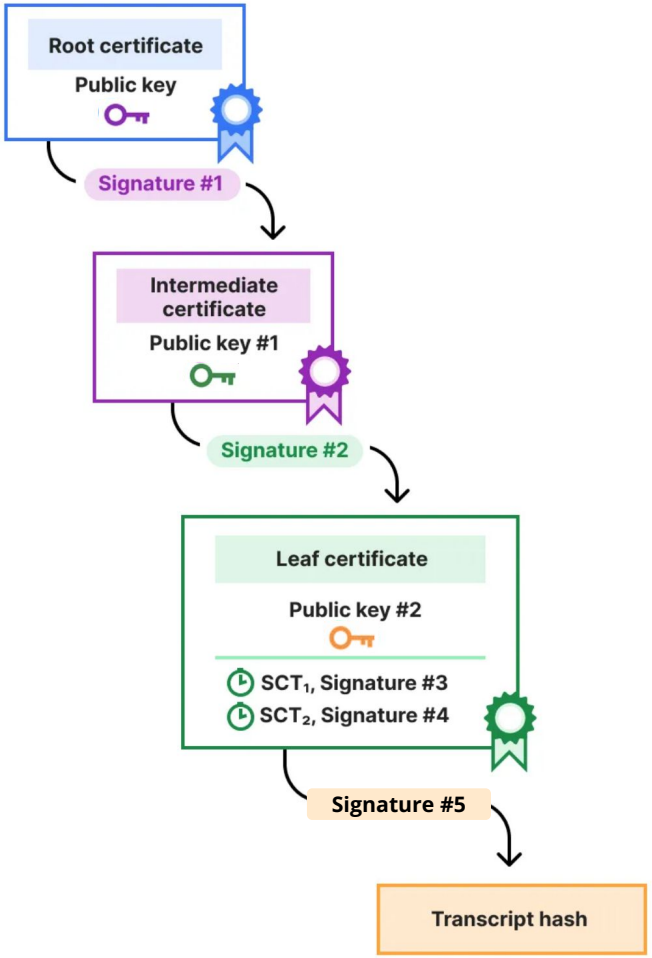
Merkle Tree Certificates

Bas Westerbaan, Cloudflare Research

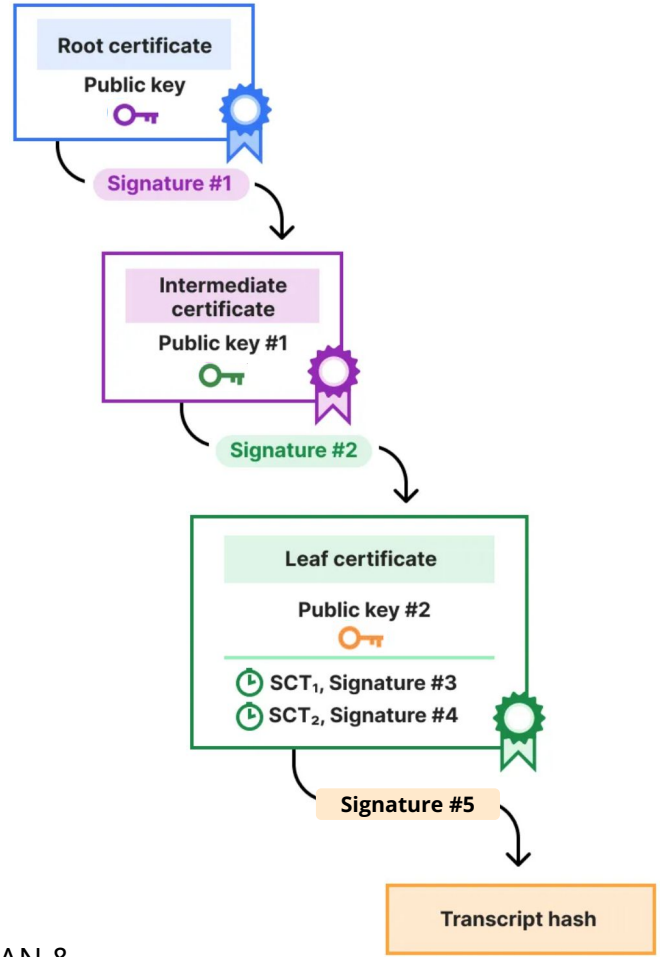
Pass the Salt 2026, July 1st, Lille

There are many signatures on the Web

Typically 5 signatures and 2 public keys when visiting a website.



Classical ❌	Algorithm	Size (bytes)
Signature #1 (root on intermediate)	RSA_{4096}	512
Public key #1 (intermediate)	RSA_{2048}	256
Signature #2 (intermediate on leaf)	RSA_{2048}	256
Public key #2 (leaf)	P-256	64
Signature #5 (leaf on transcript)	P-256	64
Signature #3 (signed certificate timestamp)	P-256	64
Signature #4 (signed certificate timestamp)	P-256	64
	Total cryptography	1,280

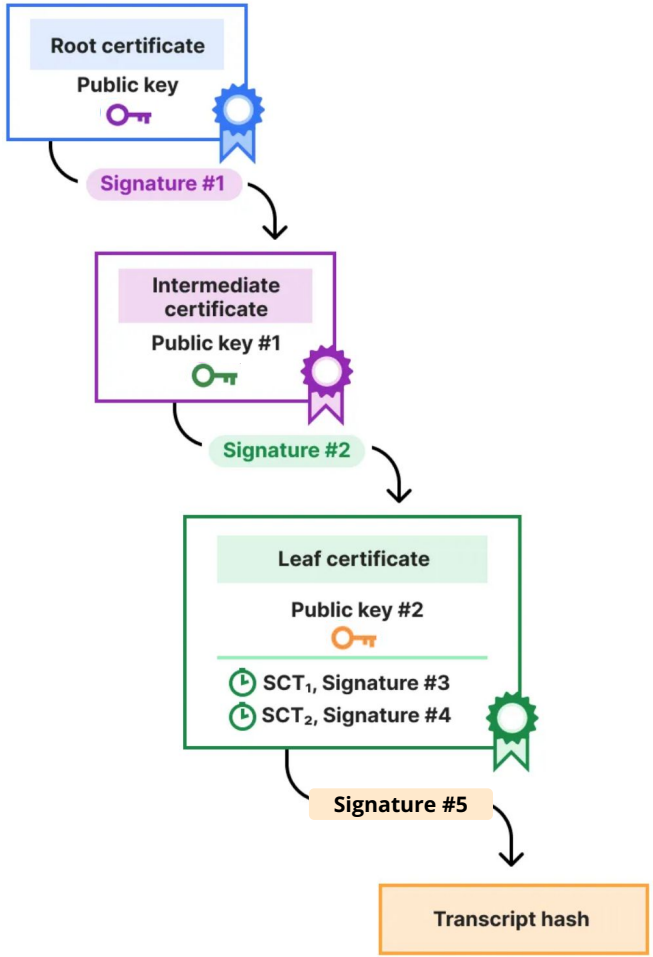


Full certificate is almost double the size! Extra data is unavoidable for SAN & validity, but quite a lot is bloat: eg. AIA, CRL DP, and certificate policy.

			Sizes (bytes)		**CPU time** (lower is better)	
Family	**Name** variant	A	Public key	Signature	Signing	Verification
Elliptic curves	Ed25519	❌	32	64	0.15	1.3
Factoring	RSA 2048	❌	272	256	80	0.4
Lattices	ML-DSA 44	✅	1,312	2,420	1 (baseline)	1 (baseline)
Symmetric	SLH-DSA 128s	✅	32	7,856	14,000	40
	SLH-DSA 128f	✅	32	17,088	720	110
	SLH-DSA 128-24	📝	32	3,856	7,000,000 ⚠️	4
	LMS M24_H20_W8	✅	48	1,112	2.9 ⚠️	8.4

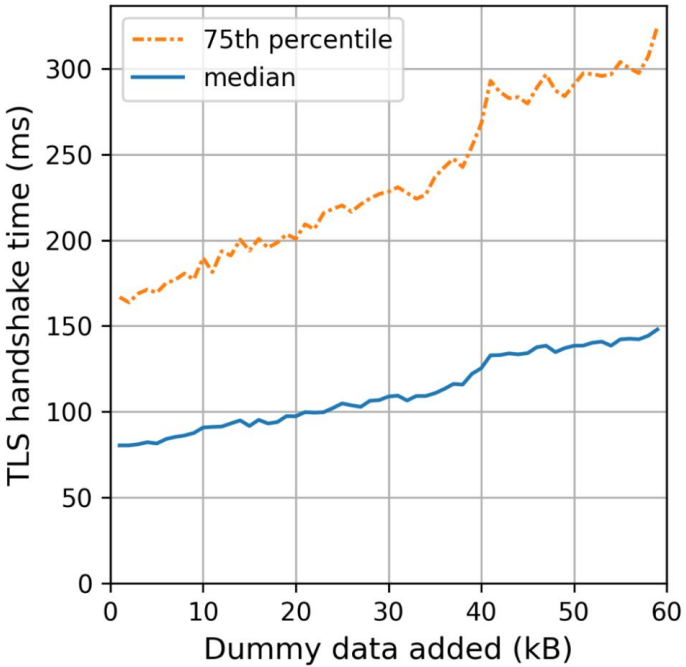
All ML-DSA ✅	Algorithm	Size (bytes)
Signature #1 (root on intermediate)	ML-DSA$_{44}$	2,420
Public key #1 (intermediate)	ML-DSA$_{44}$	1,312
Signature #2 (intermediate on leaf)	ML-DSA$_{44}$	2,420
Public key #2 (leaf)	ML-DSA$_{44}$	1,312
Signature #5 (leaf on transcript)	ML-DSA$_{44}$	2,420
Signature #3 (signed certificate timestamp)	ML-DSA$_{44}$	2,420
Signature #4 (signed certificate timestamp)	ML-DSA$_{44}$	2,420
	Total	14,724

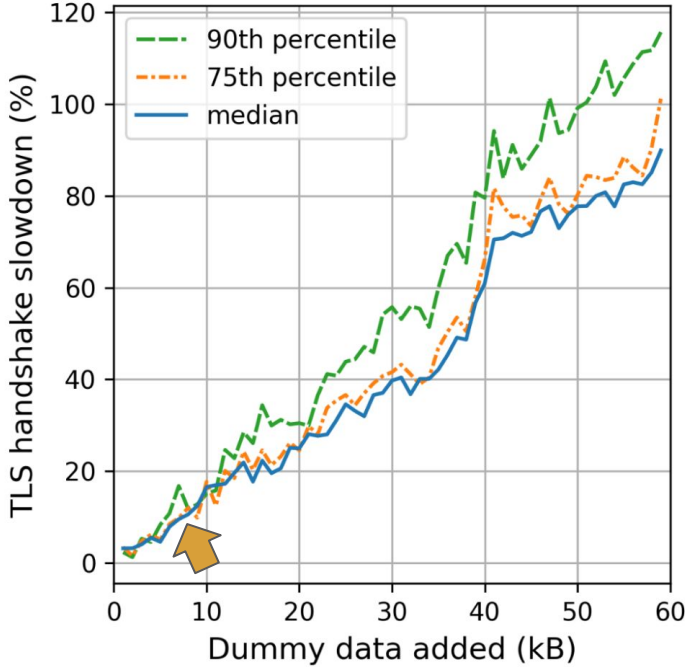
Using ML-DSA$_{65}$ instead adds 20kB.



How many (bytes) is too many?

Sizing up post-quantum signatures, 2021: We found that every 1kB added to the TLS handshake slows it down by about 1.5% at the median.





How many (bytes) is too many?

Sizing up post-quantum signatures, 2021: We found that every 1kB added to the TLS handshake slows it down by about 1.5% at the median.

Chromium Security Design Principles, 2024: “Adding ~7kB is implausible unless a cryptographically relevant quantum computer (CRQC) is tangibly imminent.”

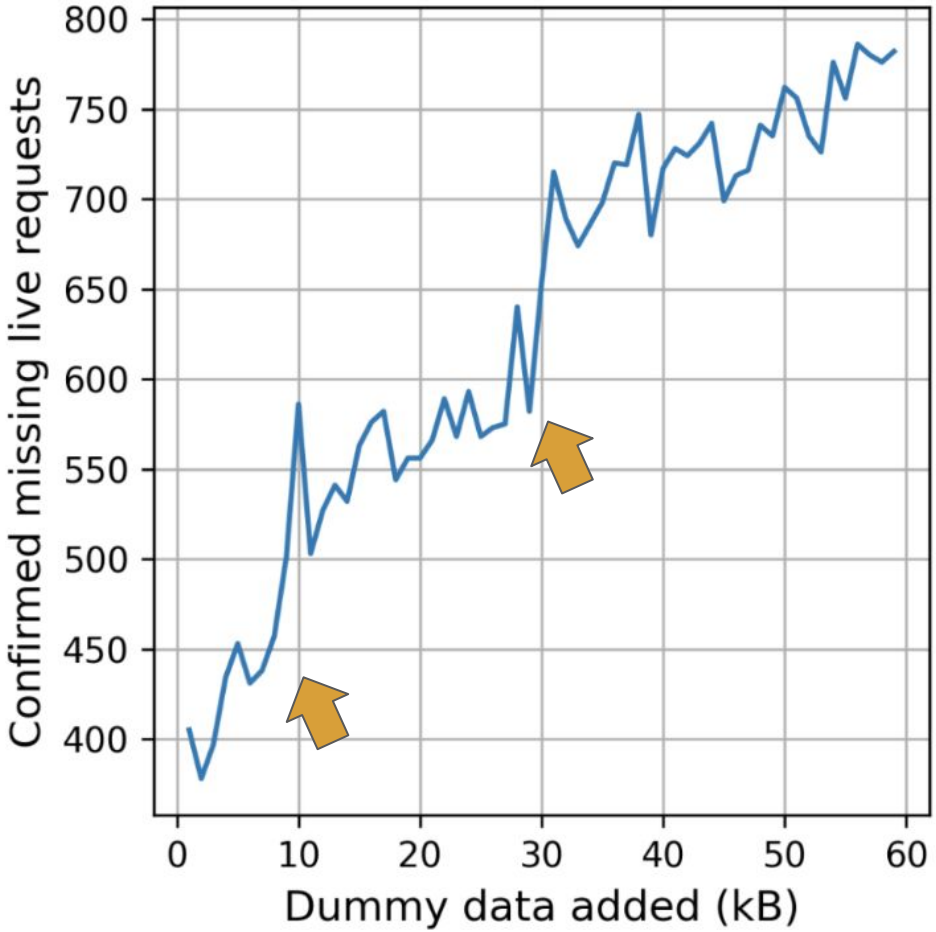
Another look at PQ signatures, 2024: Median bytes transferred from server to client for the lifetime of non-resumed QUIC connections to Cloudflare is 4.4kB.

- Classical signatures and public keys ***already*** account for about 25% of all bytes transferred on over half the connections!

If PQ signatures are too expensive, the real risk is that we deploy too late!

Protocol ossification

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



			Sizes (bytes)		CPU time (lower is better)	
Family	**Name** variant	A	Public key	Signature	Signing	Verification
Elliptic curves	Ed25519	❌	32	64	0.15	1.3
Factoring	RSA 2048	❌	272	256	80	0.4
Lattices	ML-DSA 44	✅	1,312	2,420	1 (baseline)	1 (baseline)
Symmetric	SLH-DSA 128s	✅	32	7,856	14,000	40
	SLH-DSA 128f	✅	32	17,088	720	110
	SLH-DSA 128-24	📝	32	3,856	7,000,000 ⚠️	4
	LMS M24_H20_W8	✅	48	1,112	2.9 ⚠️	8.4
Lattices	FN-DSA 512	📝	897	666	3 ⚠️	0.7
Lattices	HAWK 512	🤔	1,024	555	0.25	1.2
Proof of knowledge	MQOM L1-gf16-fast-5r	🤔	60	3,280	8	20
	SDitH SDitH2-L1-gf2-fast	🤔	70	4,484	15	40
	FAEST EM-128f	🤔	32	5,060	4.2	9
Isogeny	SQISign I	🤔	65	148	300 ⚠️	50
Multivariate	MAYO one	🤔	1,420	454	2.1	0.4
	MAYO two	🤔	4,912	186	1.1	0.8
	QR-UOV I-(127 156 54 3)	🤔	24,225	200	9.3	20
	SNOVA (24,5,4)	🤔	1,016	248	1.2	1.7
	SNOVA (25,8,3)	🤔	2,320	165	1	1.5
	SNOVA (37,17,2)	🤔	9,842	124	0.8	1.3
	UOV Is-pkc	🤔	66,576	96	0.3	2.4
	UOV Ip-pkc	🤔	43,576	128	0.3	2

… even if you wait

There are better, but still not perfect, signature algorithms on the horizon, but none of these (assuming they're not broken) will be generally available before 2031.

That includes FN-DSA.

How to manage a quantum computing emergency

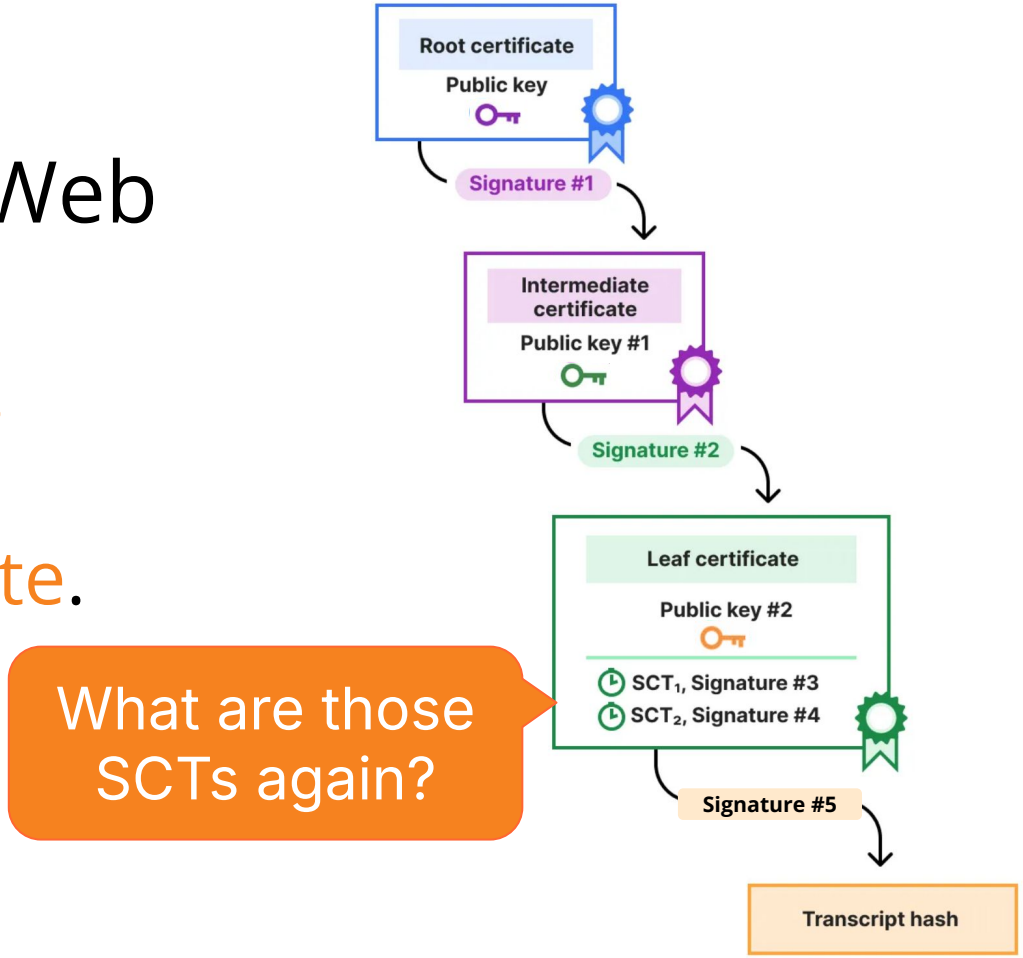
You go to war with the algorithms you have, not the ones you wish you had

Posted by ekr on 15 Apr 2024

ML-DSA will have to do.

There are many signatures on the Web

Typically 5 signatures and 2 public keys when visiting a website.



A Post Mortem on the Iranian DigiNotar Attack

SEPTEMBER 13, 2011

by Eva Galperin, Seth Schoen and Peter Eckersley

More facts have recently come to light about the compromise of the DigiNotar Certificate Authority, which appears to have enabled Iranian hackers to launch successful man-in-the-middle attacks against hundreds of thousands of Internet users inside and outside of Iran.

(full article)

After the DigiNotar hack, browsers require CAs to publish every single certificate they issue with third party *certificate transparency logs* (CT log). CAs prove they did using SCTs.

Internet Engineering Task Force (IETF)
Request for Comments: 6962
Category: Experimental
ISSN: 2070-1721

B. Laurie
A. Langley
E. Kasper
Google

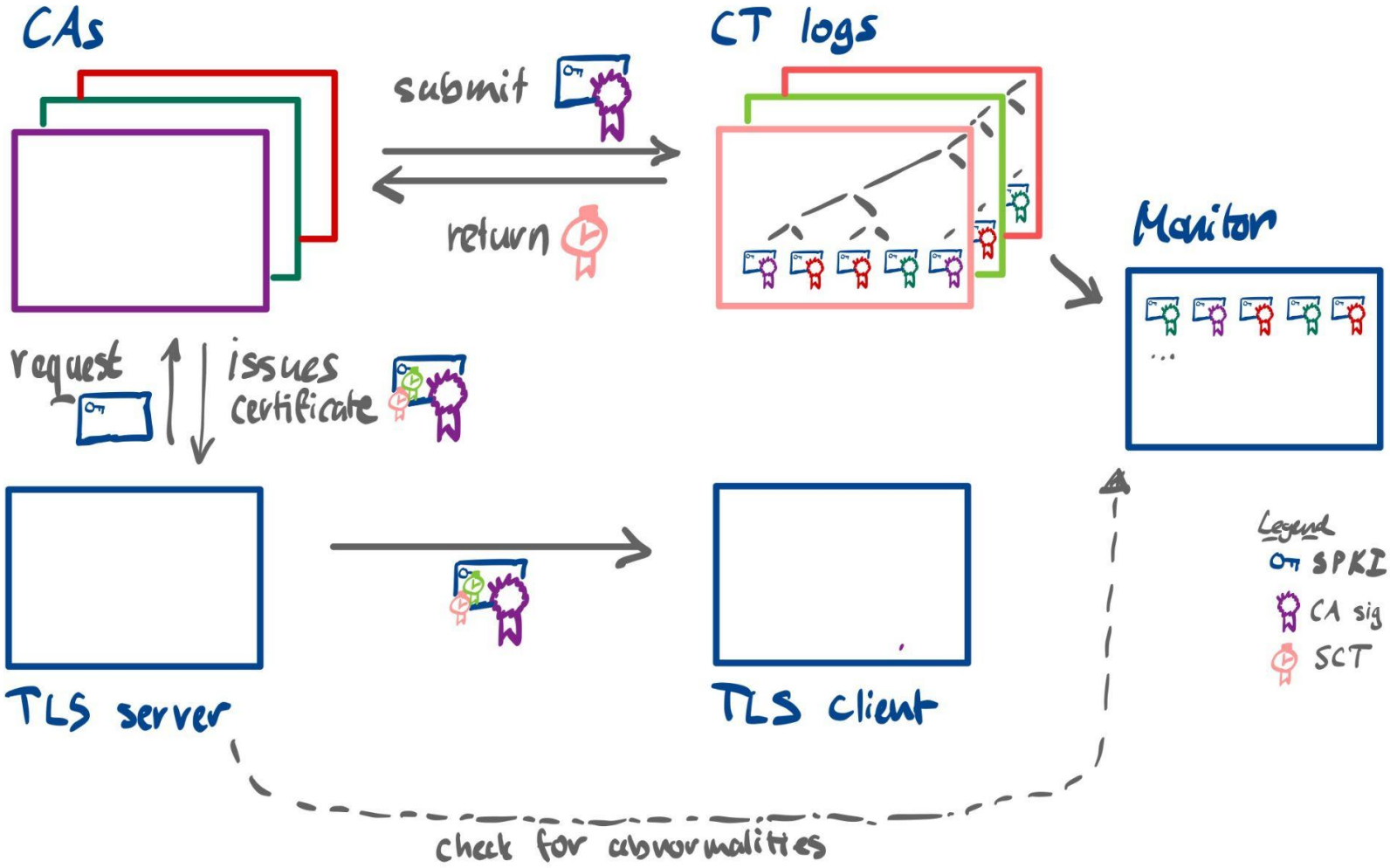
June 2013

Certificate Transparency

Abstract

This document describes an experimental protocol for publicly logging the existence of Transport Layer Security (TLS) certificates as they are issued or observed, in a manner that allows anyone to audit certificate authority (CA) activity and notice the issuance of suspect certificates as well as to audit the certificate logs themselves. The intent is that eventually clients would refuse to honor certificates that do not appear in a log, effectively forcing CAs to add all issued certificates to the logs.

(RFC 6962. By the way, CT v2 RFC 9162 is not used in practice. Instead CT logs are moving to Static CT which is easier to cache.)


CAs
submit
CT logs
return
Monitor
request
issues certificate
...
Legend
SPKI
CA sig
SCT
TLS server
TLS client
check for abnormalities

Pain points Certificate Transparency today

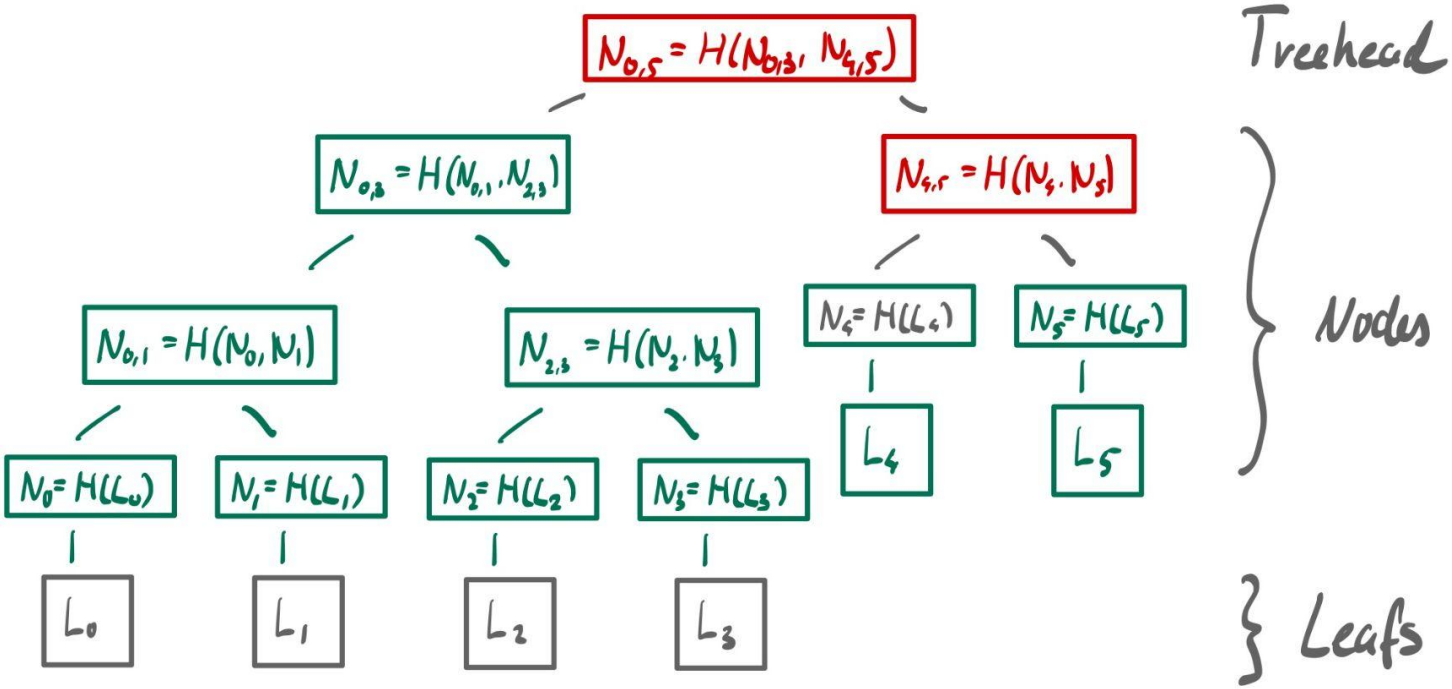
Denial of Service: attackers pull certificates from one log and use it to hammer other logs.

Monitors need to follow every single log. One broken or slow-to-monitor log might have had certificates of any CA.

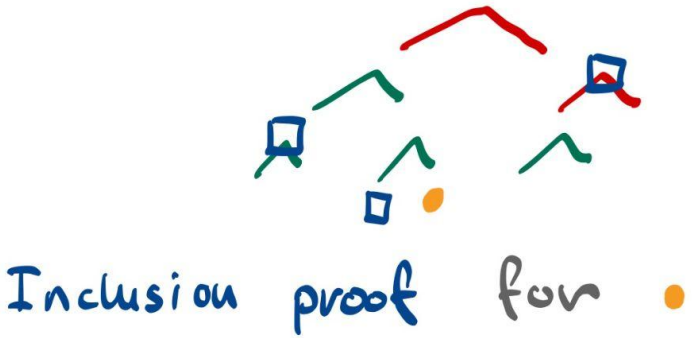
Certificates including public keys and and signatures are logged. With PQC this becomes quite a bit larger.

Tricky and costly to run. StaticCT does improve things a bit.

Before we talk about the solution, what are those Merkle Trees built by CT logs again?

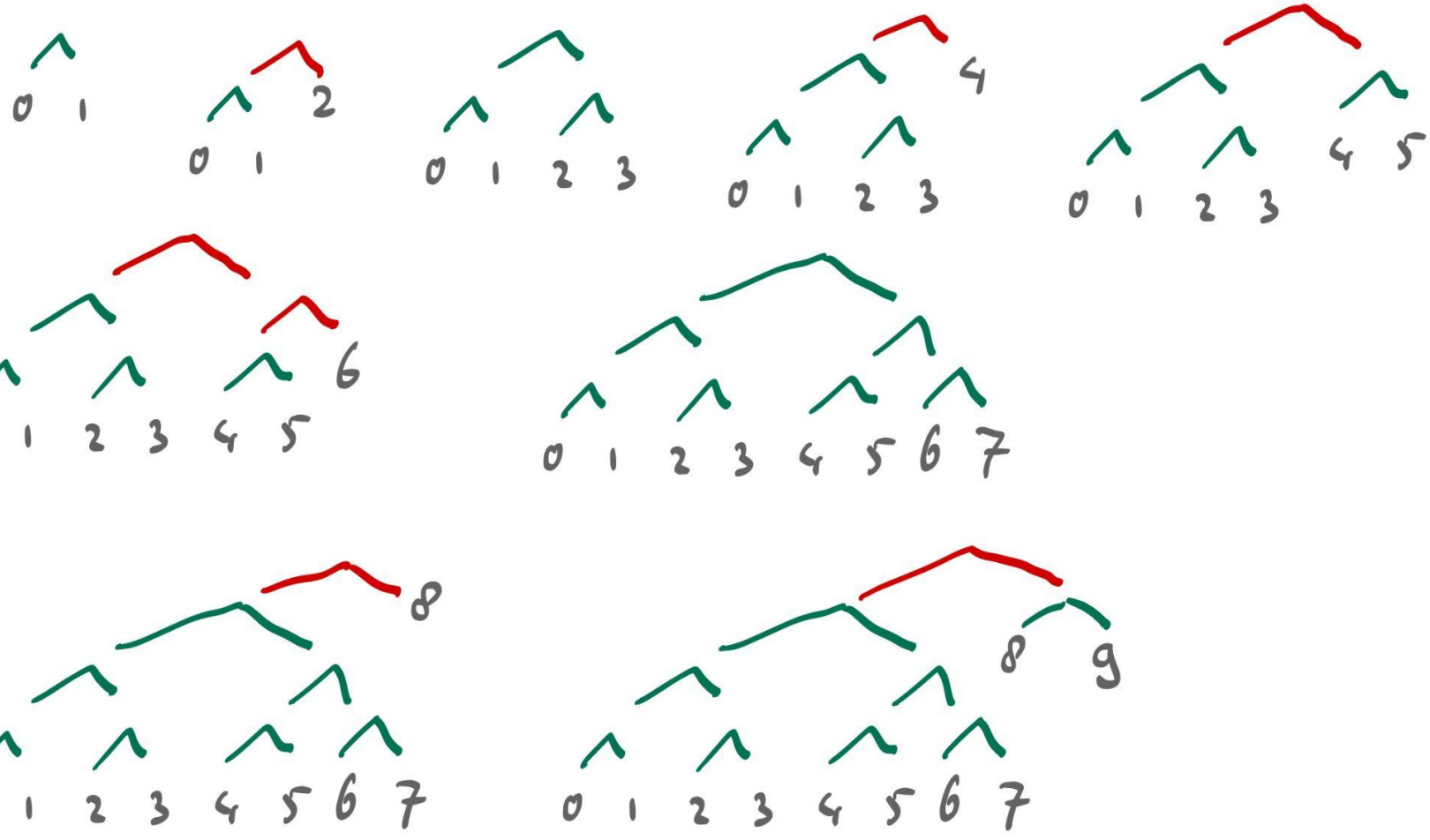


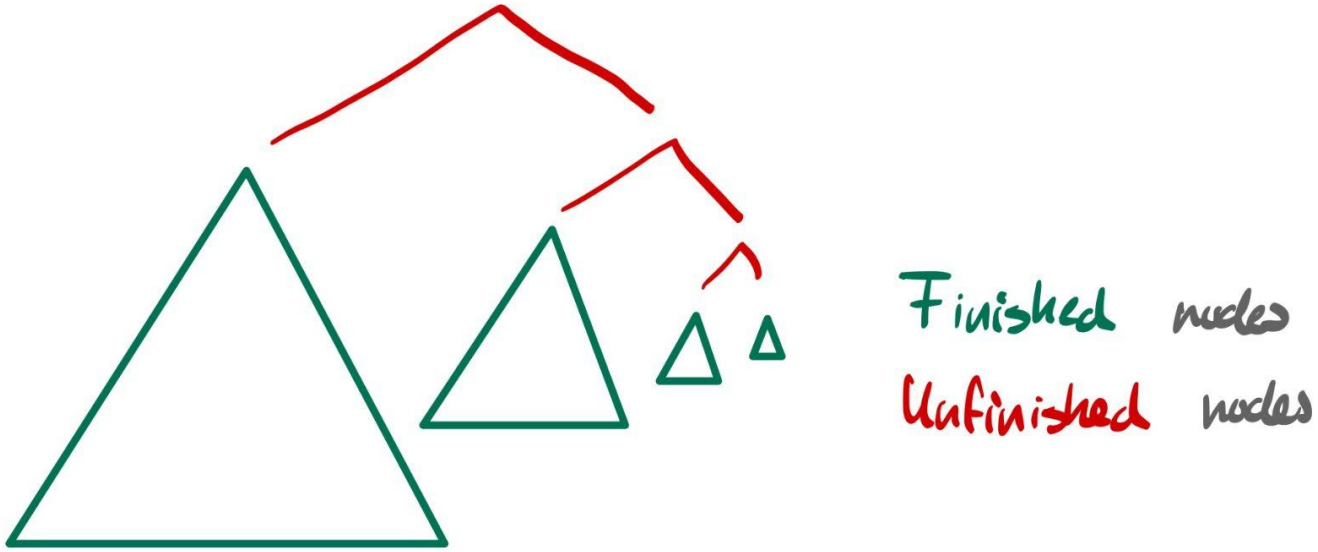
For certificate transparency, the leafs are (pre)certificates.



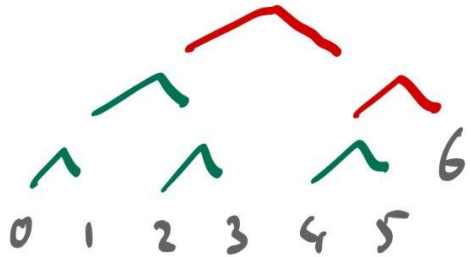
Easy to prove a certificate is in a tree with given tree head.

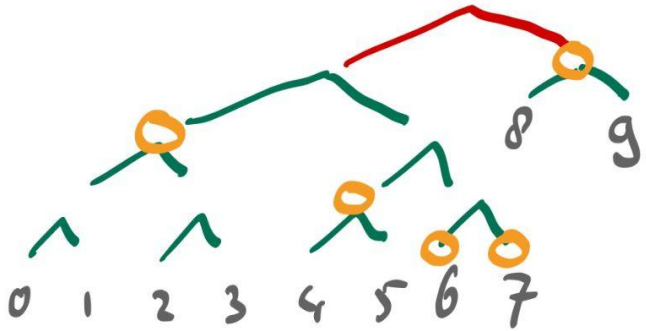
The certificate transparency tree is an ever growing tree.


0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7 8
0 1 2 3 4 5 6 7 8 9



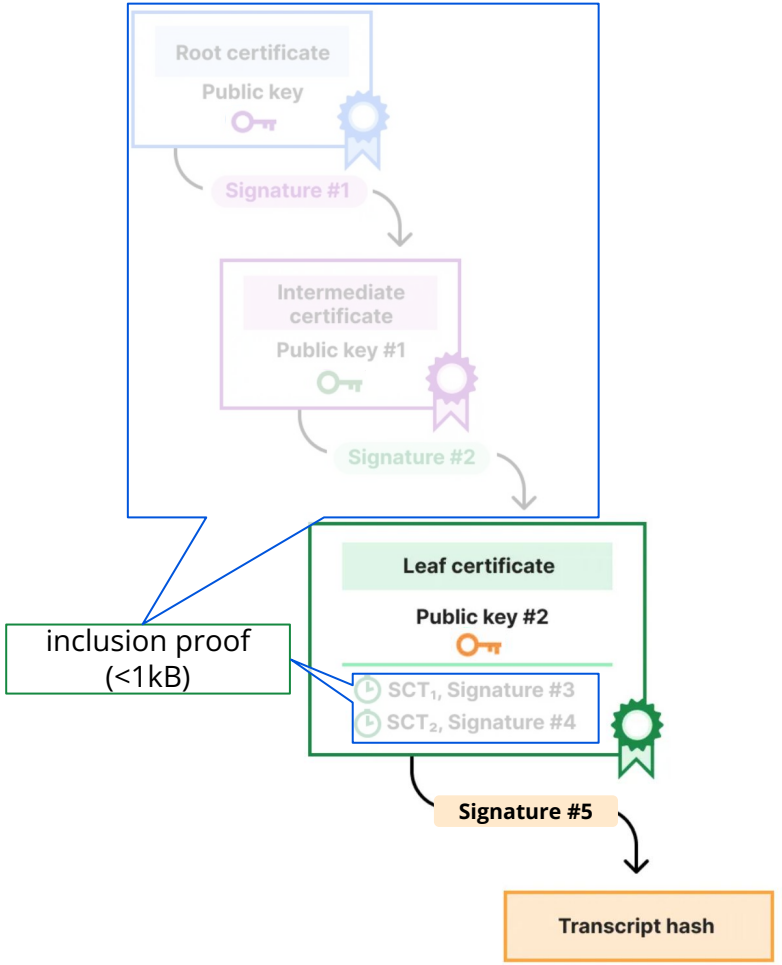
StaticCT makes CT more cacheable by using the fact that almost all nodes in a tree are finished.





Consistency proof between tree with 7 and 10 leafs

Merkle Tree Certificates (MTC)



MTC idea #1

Do not log what you issue, but issue by logging.

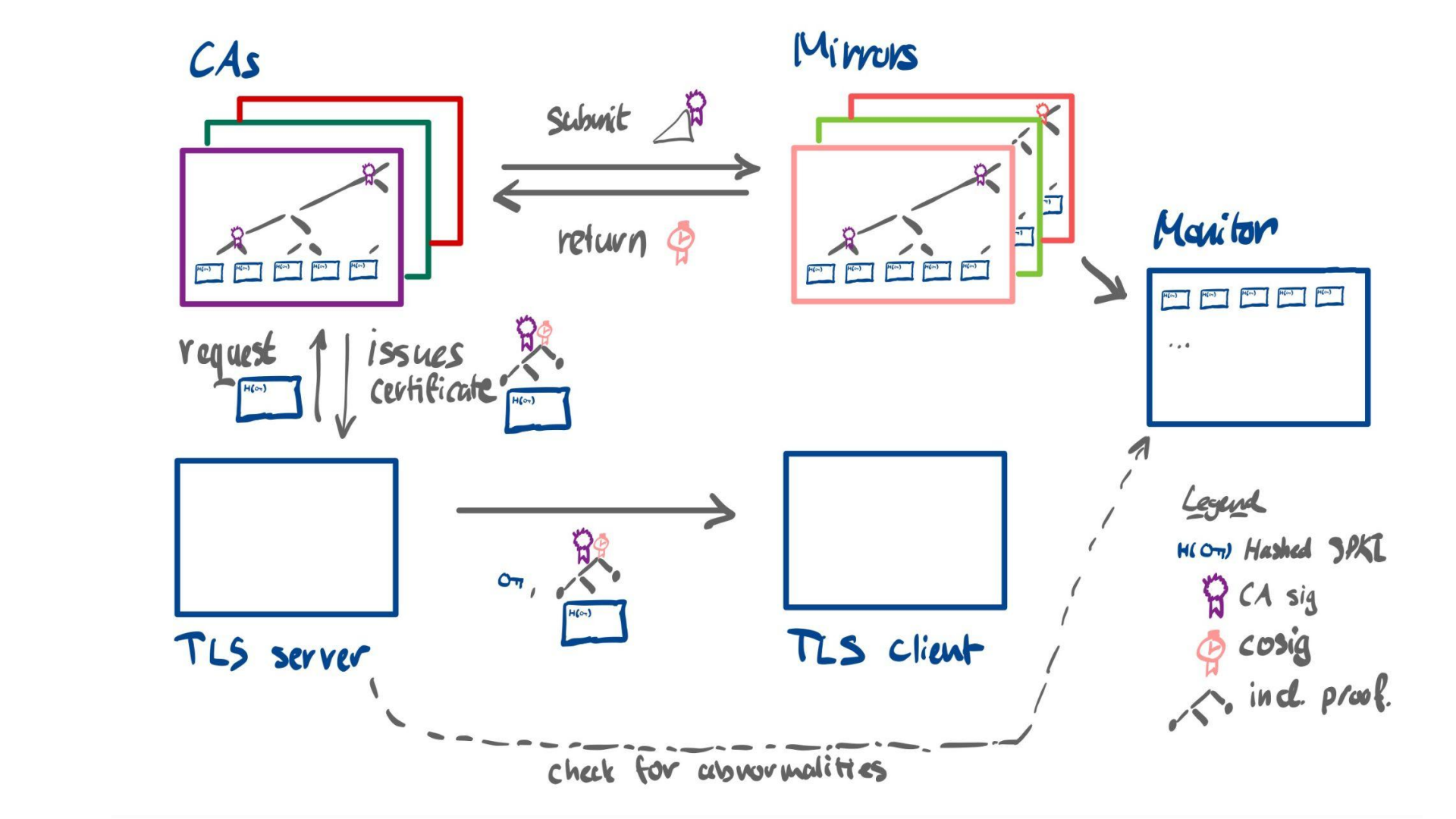
MTC idea #1: Issue by logging

CAs build and publish a log of *to-be-signed-certificates* themselves.

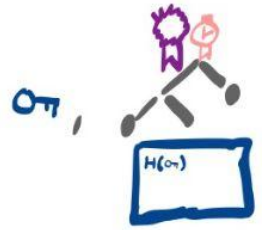
CAs issue the leafs by signing the tree head.

Replace public key in t.b.s.-certs by hash-of-public key.

Role of CT logs is replaced by mirrors. Much easier to run. Monitors only need to follow *one mirror* for each CA.


CAs
Mirrors
Submit
return
Monitor
...
request
issues certificate
TLS server
TLS client
check for abnormalities
Legend
Hashed SPKI
CA sig
cosig
incl. proof.

Standalone Merkle Tree Certificate



Like a normal X509 certificate, but:

- Certificate is not signed directly, but include Merkle Tree inclusion proof to a tree head signed by the CA and a mirror.

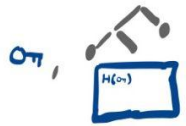
Makes CT much easier to run and monitor. Saves a little space, but not much.

To properly solve the size problem, we need a second idea.

MTC idea #2 

Sideload landmarks, hourly picked tree heads, to clients.

Landmark-relative Merkle Tree Certificate



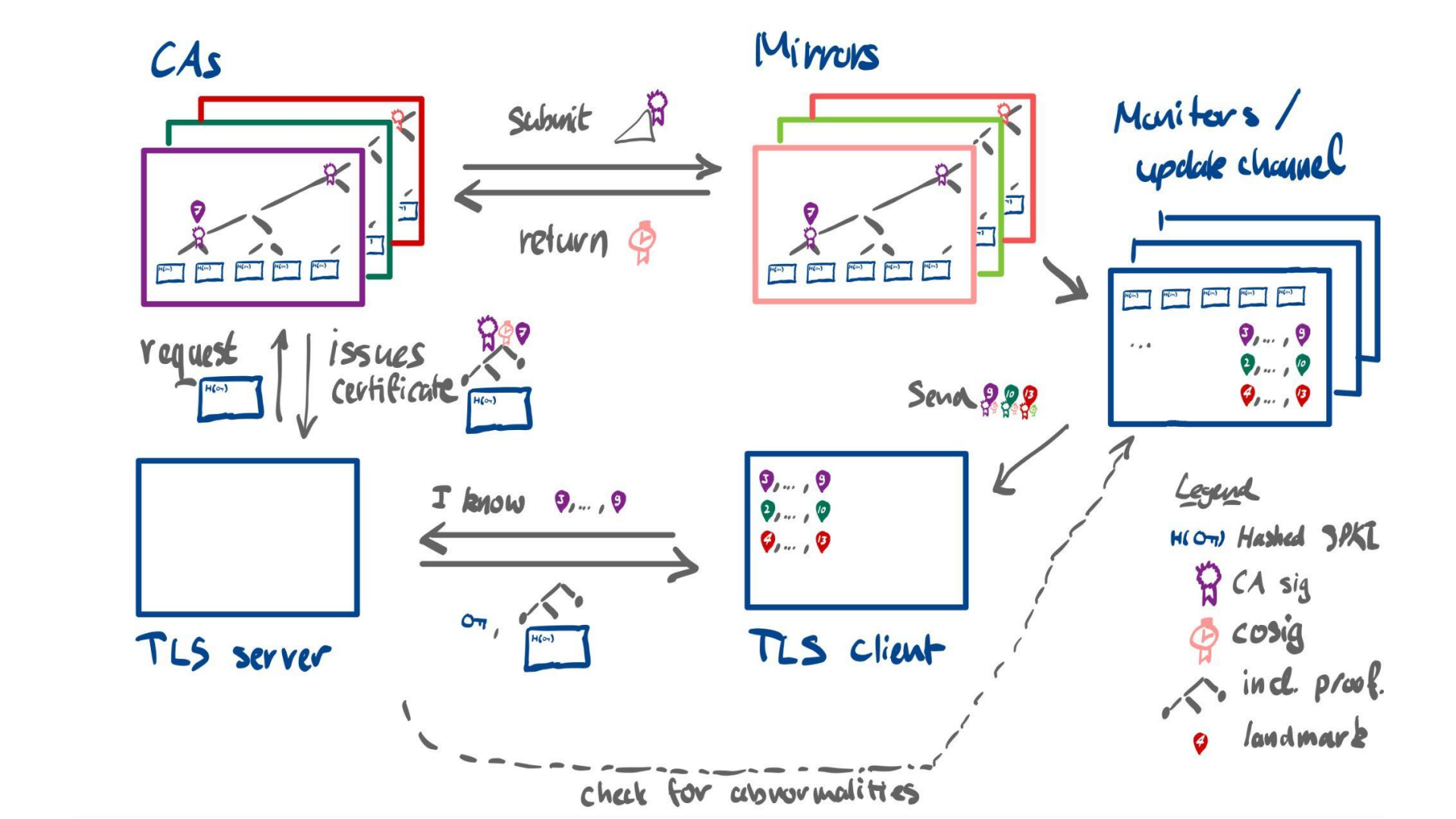
CA picks a tree head, the landmark every hour.

Third parties, for instance browser vendors or upki, monitor CAs and mirrors, and sideload these landmarks to TLS clients (browsers.)

TLS clients tells the server which landmarks they have.

TLS server can leave out (co)signatures from Merkle Tree Certificate anchored at one of those landmarks.

Only crypto left is public key (1.3kB for ML-DSA-44) and inclusion proof (<1kB). Could be smaller than classical!


CAs
Mirrors
Monitors / update channel
Submit
return
request
issues certificate
Send
I know
TLS server
TLS client
Legend
Hashed SPKI
CA sig
cosig
incl. proof
landmark
check for abnormalities

Can’t always use landmark certificates

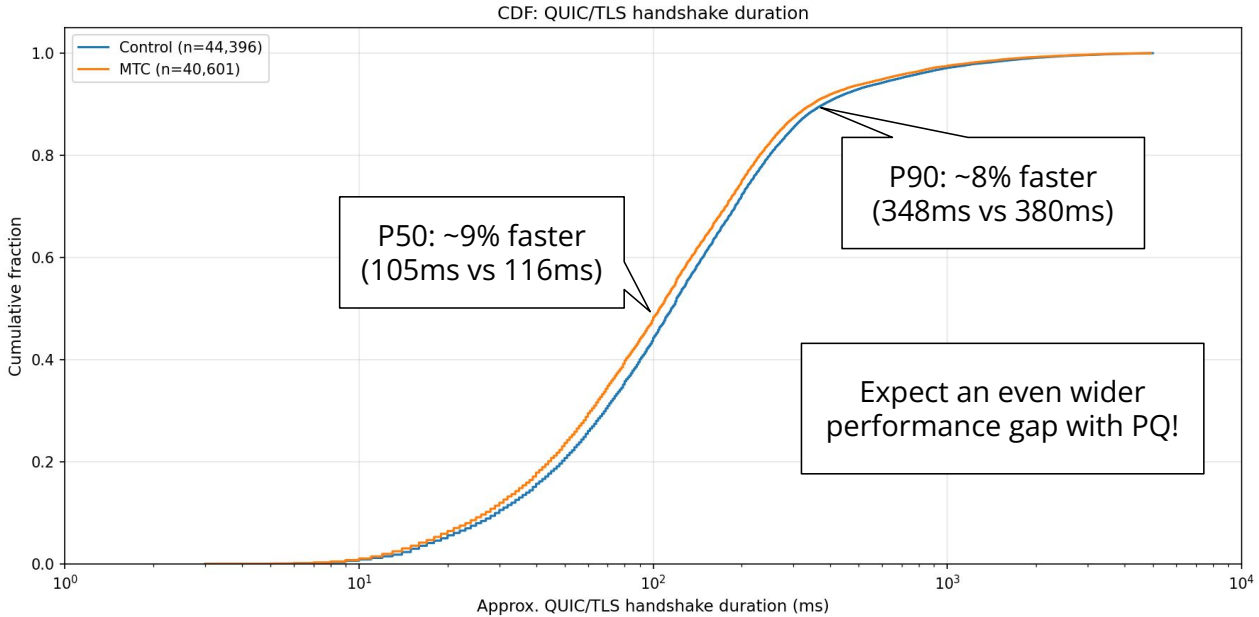
Clients need to have the latest landmarks. Think: opening laptop after a week of holiday.

When setting up a certificate for a new domain (or switching provider), it takes some time for the certificate to be included under a landmark.

In these cases we can fall back to standalone Merkle Tree Certificates.

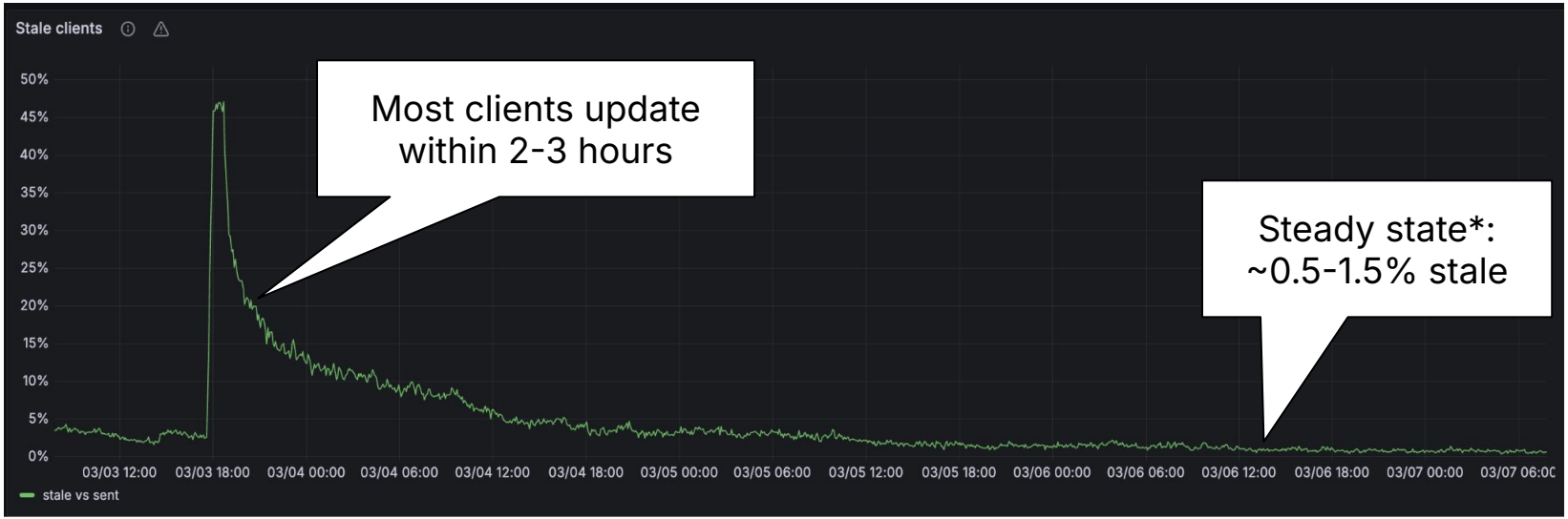
MTC testing

Since late 2025 we're experimenting MTC with classic keys with Chrome on thousands of domains, with good results so far.



MTC testing (cntd.)

For small sizes, relying parties need landmarks. How fast can Chrome pull these landmarks?



*This is an *upper bound* due to experimental quirks

MTC: looking ahead

Being standardized at the IETF, at the PLANTS working group.

Chrome sees MTC as *the* path forward for a post-quantum WebPKI. Let's Encrypt agrees. First usable production MTC CAs expected in Chrome early 2027.

Biggest lift with CAs. For clients / servers it's a simple software update to support standalone MTCs. For the small MTCs, clients need to pull landmarks.

Private PKIs that do not care for performance, transparency, or batching will likely stick to non-MTC ML-DSA certificates.

MTC: what's done and what's open?

Basic design of CA, mirrors and certs is mostly settled. Final touches on smaller details.

How to bring landmarks to non-browser clients: integration with upki? On-disk format for say /etc/ssl/mtc?

Use cases in private PKIs.

Can we improve monitoring using verifiable indices?

Come join the PLANTS working group!

Thank you, questions?

blog.cloudflare.com/post-quantum-roadmap