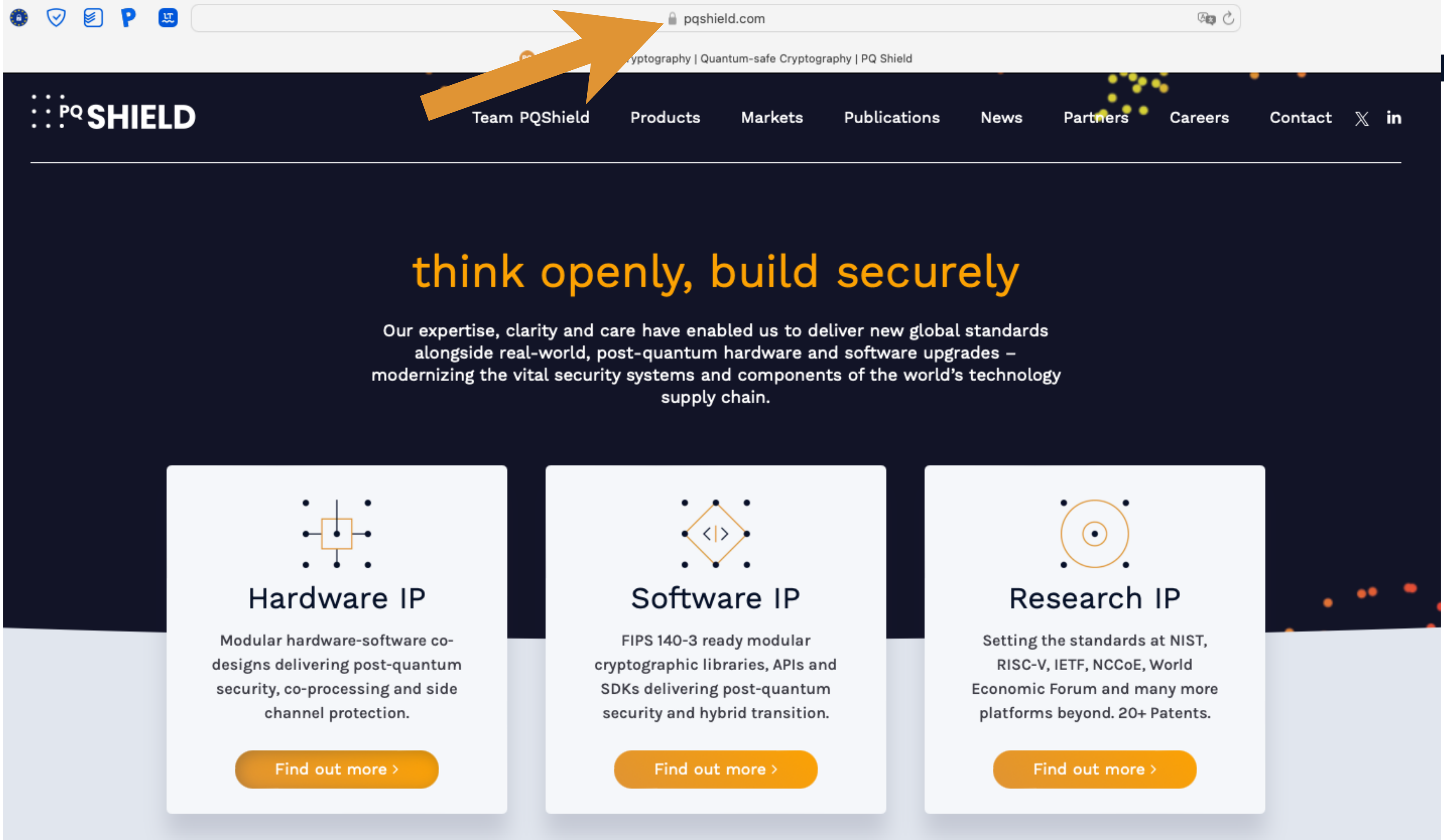
PQ TLS and WebPKI

(or: Are we PQ yet?)

Thom Wiggers

Thom Wiggers

- Cryptography researcher at PQShield
  - Oxford University spin-off
  - We develop and license PQC hardware and software IP
  - Side-channel protected hardware designs
  - FIPS 140-3 validated software
  - We also do fundamental research
- Research interest: applying PQC to real-world systems
  - Post-Quantum TLS
  - Secure messaging
- Ph.D from Radboud University (2024)
  - Dissertation: Post-Quantum TLS


pqshield.com
yptography | Quantum-safe Cryptography | PQ Shield
PQ SHIELD
Team PQShield
Products
Markets
Publications
News
Partners
Careers
Contact
in
think openly, build securely
Our expertise, clarity and care have enabled us to deliver new global standards alongside real-world, post-quantum hardware and software upgrades – modernizing the vital security systems and components of the world's technology supply chain.
Hardware IP
Modular hardware-software co-designs delivering post-quantum security, co-processing and side channel protection.
Find out more ›
Software IP
FIPS 140-3 ready modular cryptographic libraries, APIs and SDKs delivering post-quantum security and hybrid transition.
Find out more ›
Research IP
Setting the standards at NIST, RISC-V, IETF, NCCoE, World Economic Forum and many more platforms beyond. 20+ Patents.
Find out more ›

D

"TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery."

RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3

> 94,5 %

of US Firefox page loads use TLS

Firefox Telemetry, 2024-04-23

TLS

TLS 1.3 wishlist

- ✓ Secure handshake
  - ✓ More privacy
  - ✓ Only forward secret key exchanges
  - ✓ Get rid of MD5, SHA1, 3DES, EXPORT, NULL, …
- ✓ Simplify parameters
- ✓ More robust cryptography
- ✓ Faster, 1-RTT protocol
- ✓ 0-RTT resumption

TLS 1.3 wishlist

✓ Secure handshake
  ✓ More privacy
  ✓ Only forward secret key exchanges
  ✓ Get rid of MD5, SHA1, 3DES, EXPORT, NULL, …
✓ Simplify parameters
✓ More robust cryptography
✓ Faster, 1-RTT protocol
✓ 0-RTT resumption

❑ Post-quantum?

Post-Quantum TLS


Peter Shor
ECC
g^x
RSA

Post-Quantum Cryptography PQC

Overview

Public comments are available for Draft FIPS 203, Draft FIPS 204 and Draft FIPS 205, which specify algorithms derived from CRYSTALS-Dilithium, CRYSTALS-KYBER and SPHINCS⁺. The public comment period closed November 22, 2023.

PQC Seminars
Next Talk: April 23, 2024

4th Round KEMs

Additional Digital Signature Schemes - Round 1 Submissions

PQC License Summary & Excerpts

Background

NIST initiated a process to solicit, evaluate, and standardize one or more quantum-resistant public-key cryptographic algorithms. **Full details can be found in the Post-Quantum Cryptography Standardization page.**

In recent years, there has been a substantial amount of research on quantum computers – machines that exploit quantum

PROJECT LINKS
Overview
FAQs
News & Updates
Events
Publications
Presentations
ADDITIONAL PAGES
Post-Quantum Cryptography Standardization
Call for Proposals
Example Files
Round 1 Submissions
Round 2 Submissions
Round 3 Submissions
Round 3 Seminars
Round 4 Submissions
Selected Algorithms 2022

TLS 1.3

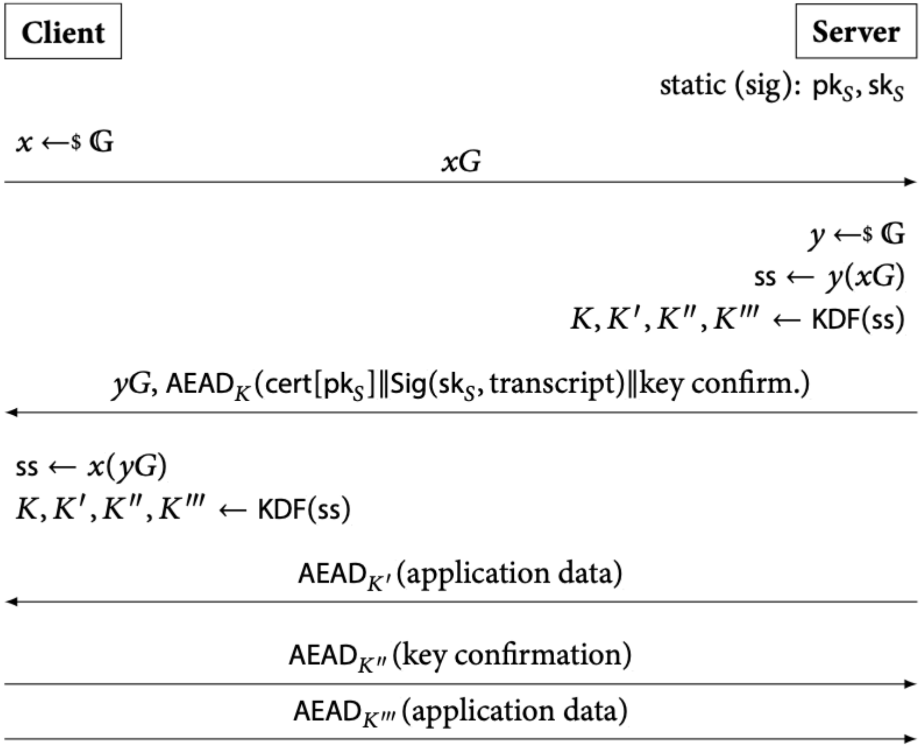


Figure 3.1: High-level overview of the TLS 1.3 handshake.

Post-Quantum

TLS 1.3

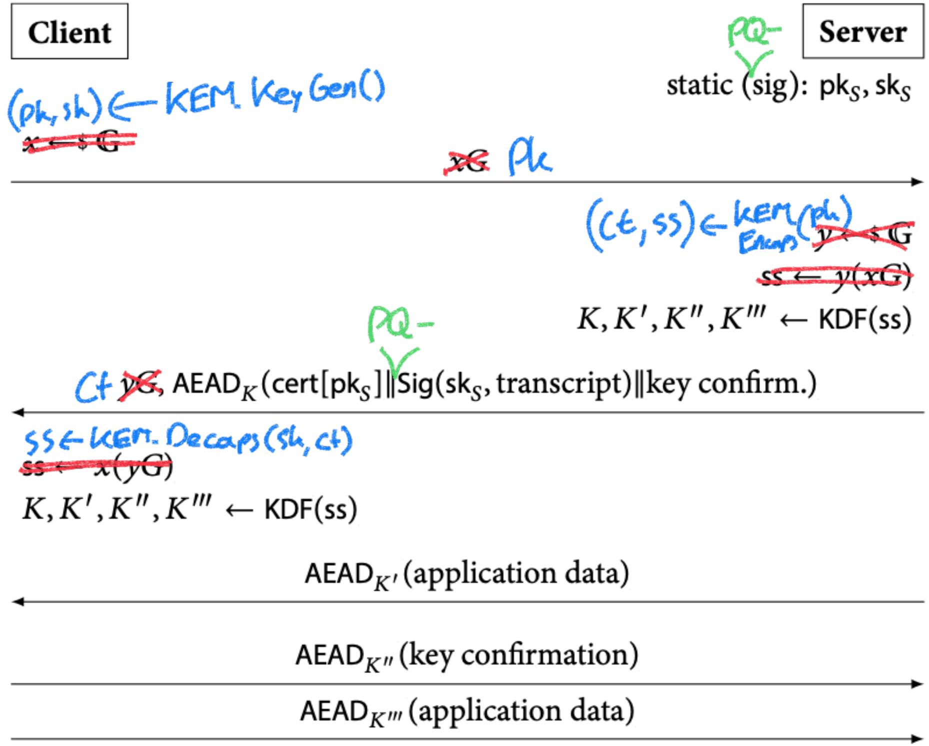


Figure 3.1: High-level overview of the TLS 1.3 handshake.

(AES-128 is fine btw)

Post-Quantum KEMs

Operation	Description
$(pk, sk) \leftarrow \text{KEM-KeyGen}()$	Generates a public/private key pair.
$(K, ct) \leftarrow \text{KEM-Encaps}(pk)$	Generates shared key K and encapsulates it to public key pk as ct.
$K \leftarrow \text{KEM-Decaps}(ct, sk)$	Decapsulates ct using sk to obtain K

	Public key	Ciphertext
ML-KEM 512	800 b	768 b
ML-KEM 768	1184 b	1088b
ML-KEM 1024	1568 b	1568 b

Post-Quantum Signatures: NIST Standards

	Public key	Signature
ML-DSA 44	1312 b	2420 b
ML-DSA 65	1952 b	3309 b
ML-DSA 87	2592 b	4627 b

Formerly Dilithium

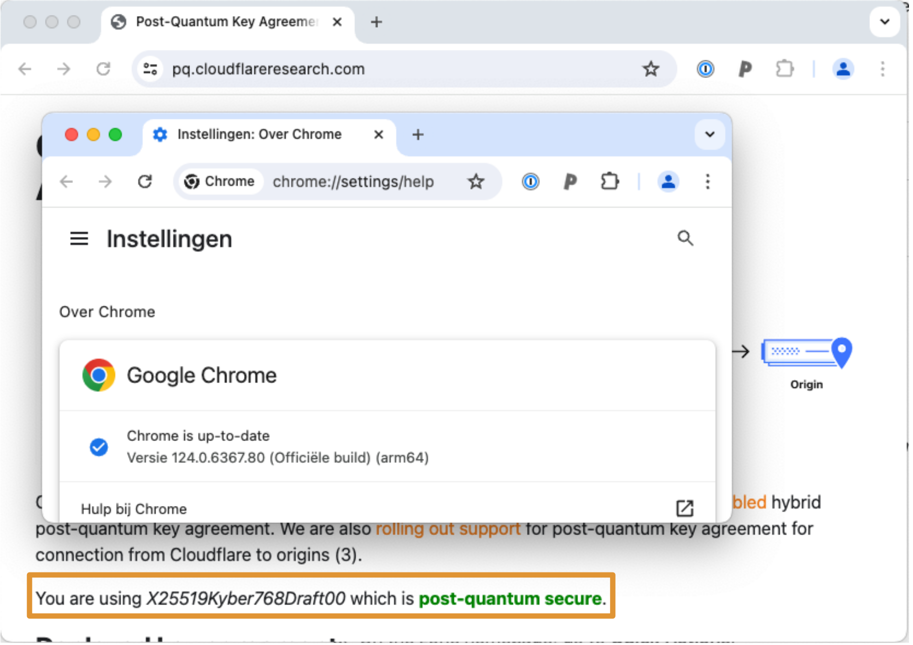
	Public key	Signature
Falcon-512	897 b	666 b
Falcon-1024	1793 b	1280 b

⚠️ Falcon *signing* uses 64-bit floats:
side-channel issues

SLH-DSA	Public Key	Signature
128s	32 b	7856 b
128f	32 b	17088 b
192s	48 b	16224 b
192f	48 b	35664 b
256s	64 b	29792 b
256f	64 b	49856 b

Formerly known as SPHINCS+

By the way: Chrome 124.0



David Adrian

Archief...omwiggers.nl 3 juni 2024 om 16:21

[TLS]Re: Curve-popularity data?

Aan: [redacted] Kopie: [redacted] <tls@ietf.org> <tls@ietf.org>

Details

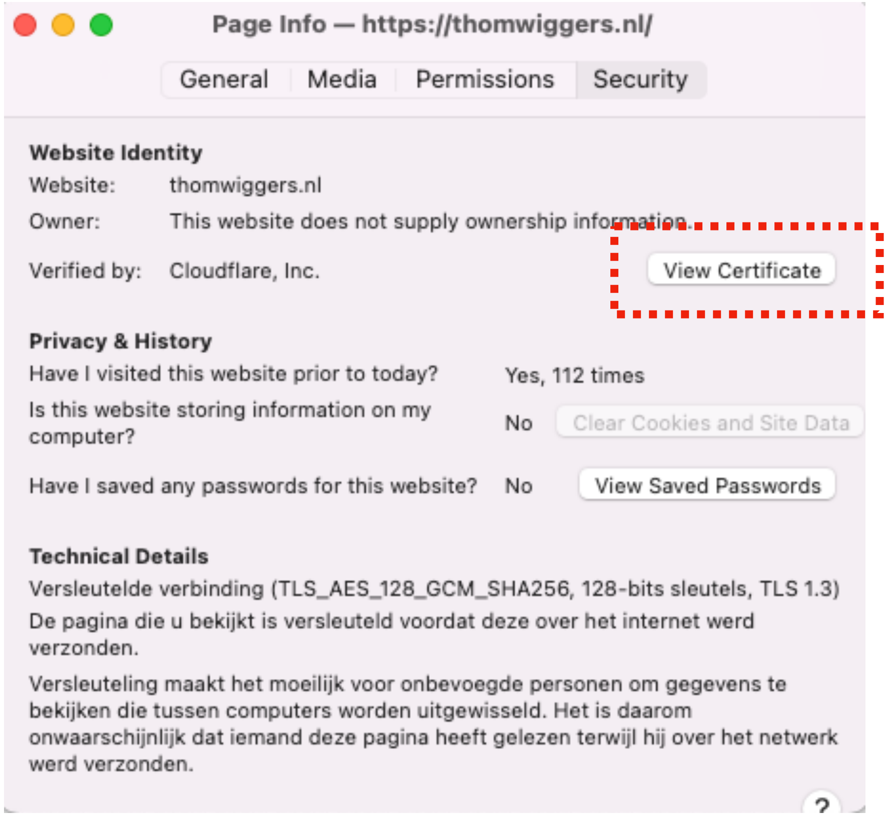
I don't really see why popularity of previous methods is relevant to picking what the necessarily new method will be is, but from the perspective of Chrome on Windows, across all ephemeral TCP TLS (1.2 and 1.3, excluding 1.2 RSA), the breakdown is roughly:

15% P256
3% P384
56% X25519
26% X25519+Kyber

We’re done!

Muhammad Mahdi Karim/www.micro2macro.net
GFDL 1.2 Source

WebPKI


Page Info — https://thomwiggers.nl/
General
Media
Permissions
Security
Website Identity
Website: thomwiggers.nl
Owner: This website does not supply ownership information.
Verified by: Cloudflare, Inc.
View Certificate
Privacy & History
Have I visited this website prior to today? Yes, 112 times
Is this website storing information on my computer? No
Clear Cookies and Site Data
Have I saved any passwords for this website? No
View Saved Passwords
Technical Details
Versleutelde verbinding (TLS_AES_128_GCM_SHA256, 128-bits sleutels, TLS 1.3)
De pagina die u bekijkt is versleuteld voordat deze over het internet werd verzonden.
Versleuteling maakt het moeilijk voor onbevoegde personen om gegevens te bekijken die tussen computers worden uitgewisseld. Het is daarom onwaarschijnlijk dat iemand deze pagina heeft gelezen terwijl hij over het netwerk werd verzonden.
?

3 Certificates

Certificate

sni.cloudflaressl.com | Cloudflare Inc ECC CA-3 | Baltimore CyberTrust Root

Pre-installed

Subject Name

Country	US
State/Province	California
Locality	San Francisco
Organization	Cloudflare, Inc.
Common Name	sni.cloudflaressl.com

Issuer Name

Country	US
Organization	Cloudflare, Inc.
Common Name	Cloudflare Inc ECC CA-3

Validity

Not Before	Wed, 16 Jun 2021 00:00:00 GMT
Not After	Wed, 15 Jun 2022 23:59:59 GMT

Subject Alt Names

DNS Name	thomwiggers.nl
DNS Name	sni.cloudflaressl.com
DNS Name	*.thomwiggers.nl

Public Key Info

Algorithm	Elliptic Curve
Key Size	256
Curve	P-256
Public Value	04:04:FF:B8:9F:66:B9:D5:CE:40:91:4B:B7:B4:8C:B4:D2:C4:17:E7:AA:75:2...

Miscellaneous

Serial Number	05:E1:B4:51:22:F8:E4:1A:9F:87:F0:61:D0:40:BD:07
Signature Algorithm	ECDSA with SHA-256
Version	3
Download	PEM (cert) PEM (chain)

Fingerprints

SHA-256	B3:D7:D5:C2:9A:ED:DE:A1:AA:7C:EA:9E:21:E9:A7:4F:6C:DA:7C:40:86:CA:8...
SHA-1	8E:D8:3E:CC:C1:95:D9:25:32:E9:97:47:30:13:6D:9D:42:93:E6:83

Basic Constraints

Certificate Authority	No

Key Usages

Purposes	Digital Signature

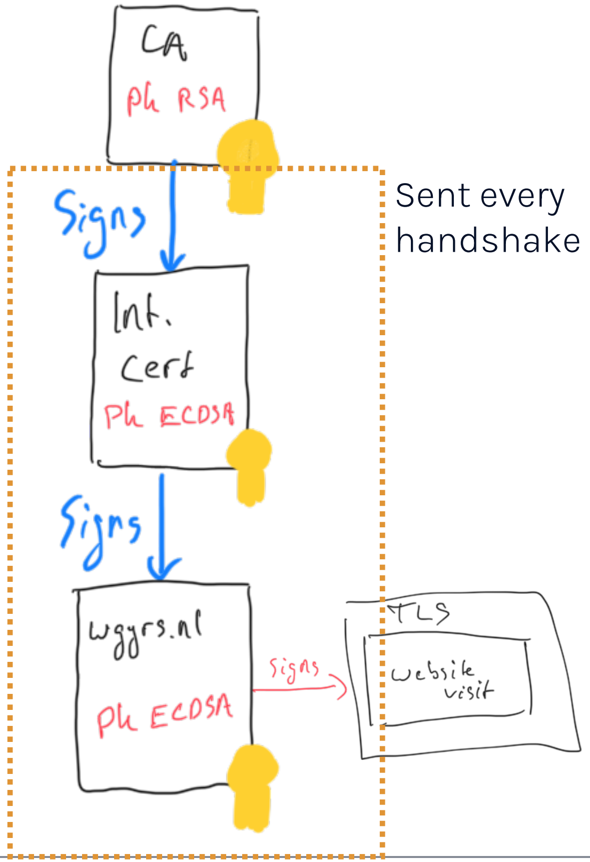
Extended Key Usages

Purposes	Server Authentication, Client Authentication

handshake signature
+ leaf certificate public key + intermediate certificate signature
+ root signature on intermediate
= 3 signatures and 2 public keys

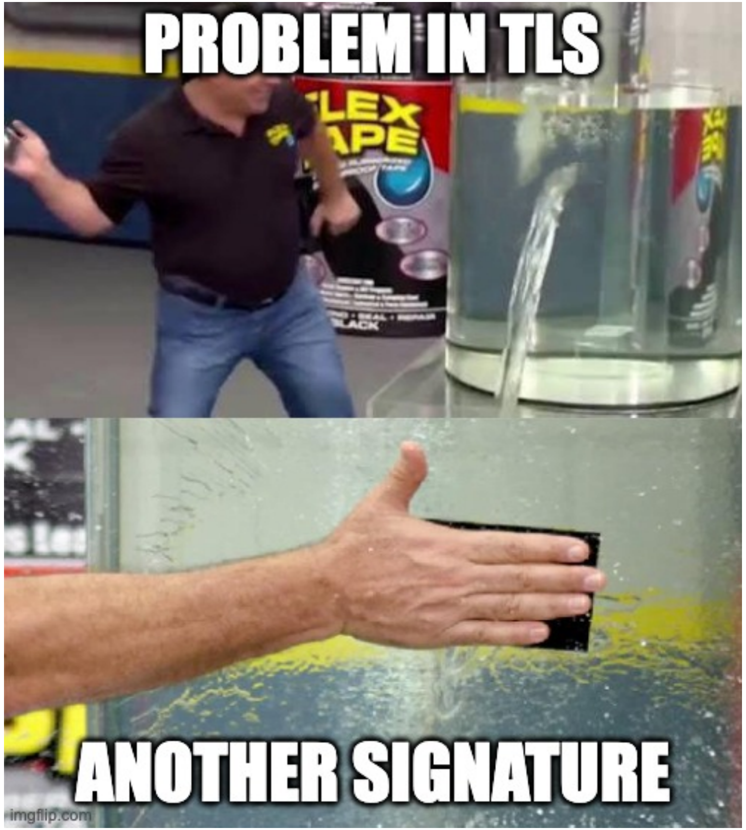
Public Key Infrastructure

- Certificate Authorities (CA)
- Become a trusted CA by:
  - spending 💰 💰 on audits
  - convince vendors to install your certificate
- Vendors trust CAs to check if I own wggrs.nl
- Intermediate CA certs make key management easier
  - (offline master signing key, etc)



Aside: PKI open problems

- Certificate issuance
- Certificate Revocation
  - Certificate Revocation Lists (CRL)
  - Online Certificate Status Protocol (OCSP)
- Any trusted CA can issue a certificate for anyone
  - Famously abused by Iran(?) to attack Gmail in DigiNotar.nl hack
  - “Certificate Transparency” (CT)


PROBLEM IN TLS
ANOTHER SIGNATURE
imgflip.com

Slap another signature on it

Online Certificate Status Protocol

Authority Info (AIA)

Location	http://ocsp.digicert.com
Method	Online Certificate Status Protocol (OCSP)
Location	http://cacerts.digicert.com/CloudflareIncECCCA-3.crt
Method	CA Issuers

+= 1 signature

Certificate Transparency

Embedded SCTs

Log ID	29:79:BE:F0:9E:39:39:21:F0:56:73:9F:63:A5:77:E5:BE:57:7D:9C:60:0A:F8:...
Name	Google "Argon2022"
Signature Algorithm	SHA-256 ECDSA
Version	1
Timestamp	Wed, 16 Jun 2021 17:11:33 GMT
Log ID	22:45:45:07:59:55:24:56:96:3F:A1:2F:F1:F7:6D:86:E0:23:26:63:AD:C0:4B...
Name	DigiCert Yeti2022
Signature Algorithm	SHA-256 ECDSA
Version	1
Timestamp	Wed, 16 Jun 2021 17:11:33 GMT
Log ID	51:A3:B0:F5:FD:01:79:9C:56:6D:B8:37:78:8F:0C:A4:7A:CC:1B:27:CB:F7:9E...
Name	DigiCert Nessie2022
Signature Algorithm	SHA-256 ECDSA
Version	1
Timestamp	Wed, 16 Jun 2021 17:11:33 GMT

+= 3 signatures

Certificate Transparency

Certificate Transparency

- Chrome, Safari require all certificates to be submitted to at least 2 certificate transparency logs

Certificate Transparency

- Chrome, Safari require all certificates to be submitted to at least 2 certificate transparency logs
- Log is a Merkle tree of hostnames and hashes of included certificates
    - No privacy! You can search this using https://crt.sh

Certificate Transparency

- Chrome, Safari require all certificates to be submitted to at least 2 certificate transparency logs
- Log is a Merkle tree of hostnames and hashes of included certificates
  - No privacy! You can search this using https://crt.sh
- Auditing, etc, are part of the design

Certificate Transparency

- Chrome, Safari require all certificates to be submitted to at least 2 certificate transparency logs
- Log is a Merkle tree of hostnames and hashes of included certificates
    - No privacy! You can search this using https://crt.sh
- Auditing, etc, are part of the design

- SCT proofs in certificates are promises of inclusion within 24 hours for deployment reasons

Certificate Transparency

- Chrome, Safari require all certificates to be submitted to at least 2 certificate transparency logs
- Log is a Merkle tree of hostnames and hashes of included certificates
  - No privacy! You can search this using https://crt.sh
- Auditing, etc, are part of the design

- SCT proofs in certificates are promises of inclusion within 24 hours for deployment reasons
- CT logs typically only accept certificates from trusted issuers

Summarising

- Typical **web** TLS handshake:
  - ephemeral key exchange
  - handshake signature
  - leaf certificate:
    pk
    + signature by intermediate CA crt
    + OCSP staple
    + 3x SCT
  - intermediate CA certificate:
    pk + signature by root CA
  - root certificate (preinstalled)

1 online keygen+key exchange

1 online signing operation

6 offline signatures

PQ Performance

Impact of PQ

- ~~Kyber~~ML-KEM key exchange: ~1.5kB
- ML-DSA-44: 18 kB of certificates!!
- Falcon-512: ~5 kB

Note: TCP congestion control

On connection establishment, TCP will allow you to send some amount of data before acknowledgement from the other side.

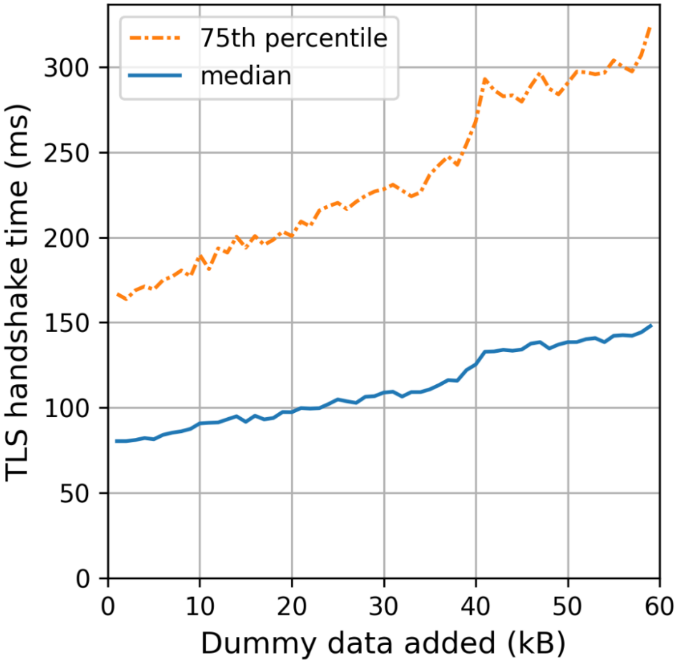
This window (and thus available connection bandwidth) scales as the connection is proven reliable when receiving TCP ACKs.

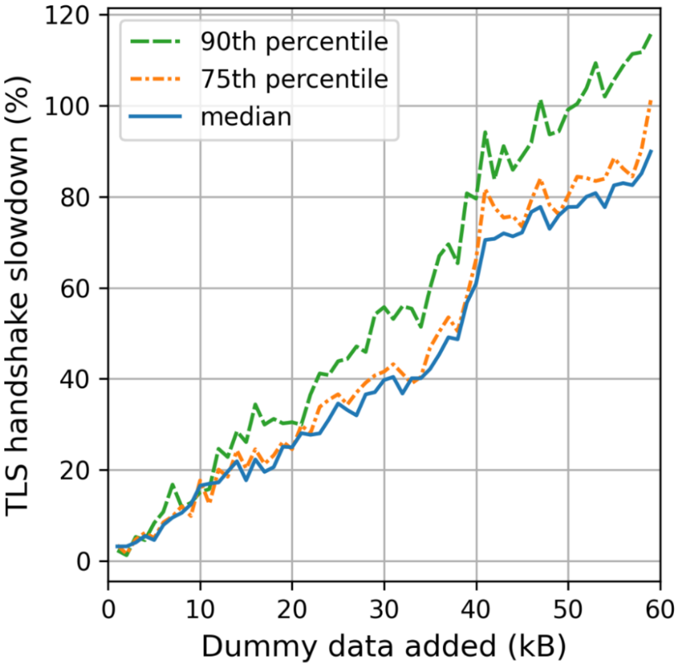
The default initial window on Linux is 10 packets, so **if you send more than ~15 kB of data, you're stuck waiting for an extra round-trip!**

Even without congestion control,
more bytes = more slowlier

sizes per https://blog.cloudflare.com/sizing-up-post-quantum-signatures/

Cloudflare live internet experiment: More data results in slowdown





Bas Westerbaan, https://blog.cloudflare.com/sizing-up-post-quantum-signatures/. **Cloudflare has a 30 MSS = ~40kb congestion window**

Severe performance impact

- Kyber-768 “only” adds 2.3 kB to the handshake
- Google notes this already slows down handshakes by 4%
- Google observes a significant impact on lower-quality internet connections
  - This is why they’re only enabling this on Chrome Desktop right now

- To stay under 10% slowdown, we seem to have a budget of at most 10kB including KEX
  - We need something better than just replacing signatures

https://dadrian.io/blog/posts/pqc-signatures-2024/

https://blog.chromium.org/2024/05/advancing-our-amazing-bet-on-asymmetric.html

https://securitycryptographywhatever.com/2024/05/25/ekr/

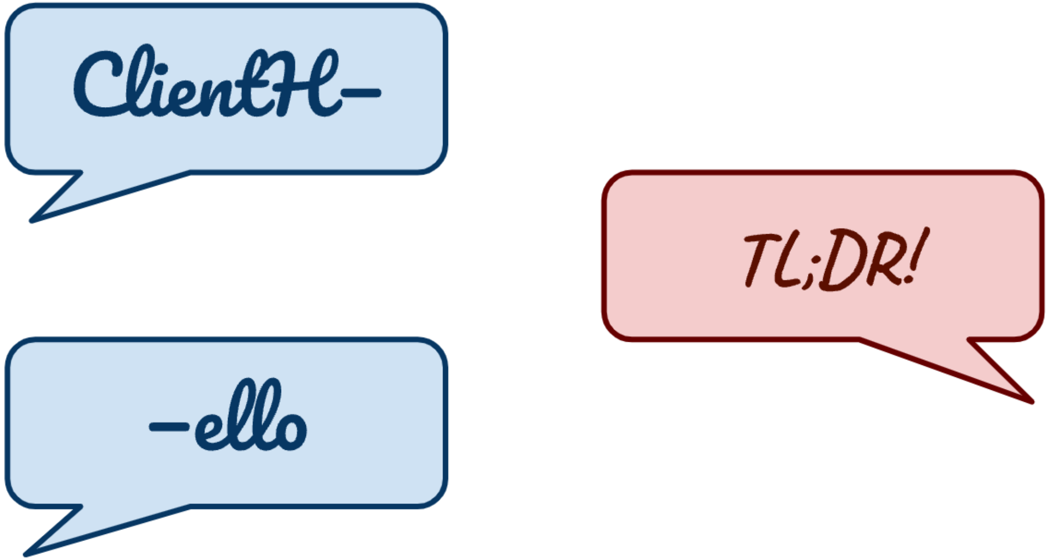
Not just speed

- Larger Hello messages can lead to fragmentation
- Not all implementations are prepared to deal with fragmented packets
- Especially middle boxes affected

Product	Status	Discovered	Via	Patched	Links
Vercel	✅	2023-08-15	Chrome Beta	2023-08-23	Twitter
ZScalar	✅	2023-08-17	Chrome Beta	2023-09-28	
Cisco		2024-04-23	Chrome 124	Unknown	Cisco Bug
Envoy	✅	2024-04-29	Chrome 124	n/a (config-only)	Github

Table last updated 2024-05-13

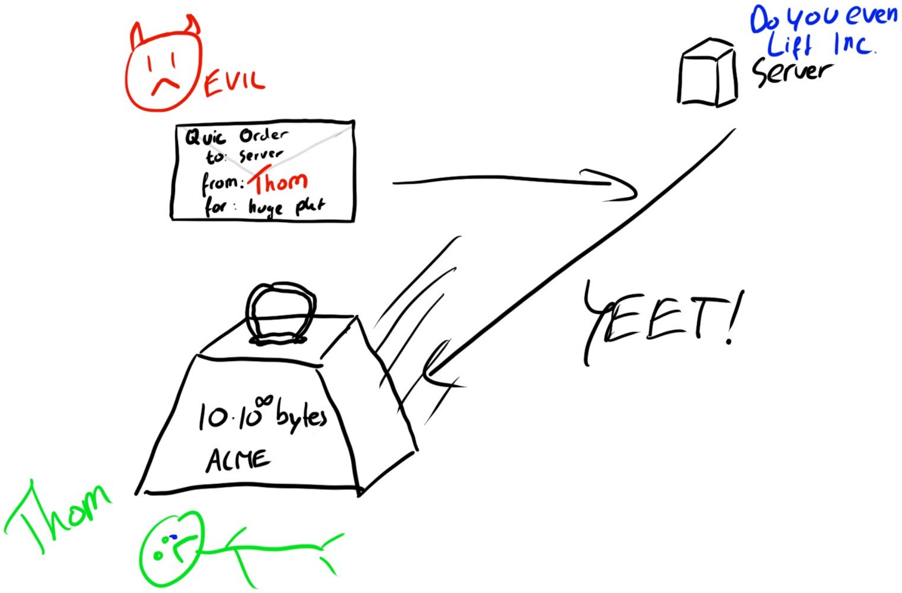
(List not exhaustive)



https://tldr.fail

More problems with sizes

- Variant protocols DTLS and QUIC are based on UDP: no TCP SYN/ACK sequence
- ClientHello message received by server could be spoofed, so QUIC allows sending back at most 3x the ClientHello size (avoids DoS amplification)
- Sending back 18kB of ML-DSA requires the client to pad its ClientHello message with ~5kB



Avoiding the costs of certificates

- Certificates are already very large, PQ makes this much worse
- We have multiple signatures that prove validity in each certificate:
  - Signature on certificate itself
  - OCSP staple that proves that certificate is currently valid
  - Certificate Transparency log inclusion proves that certificate was from a trusted issuer

Can we do things in a smarter way?

New WebPKI?

Combining different algorithms

- handshake signature
- leaf certificate:

  pk

  + signature by intermediate CA crt

  + OCSP staple

  + 3x SCT
- intermediate CA certificate:

  pk

  + signature by root CA
- root certificate (preinstalled)

Robust against side-channels, pk+sig small, fast signing

ML-DSA

Signature-verification only, pk+sig small

Falcon

Signature-verification only, signature small

UOV? (Signatures on-ramp)

Note: using multiple algorithms also has cost!

Avoiding the costs of certificates

- Certificates are very large, PQ makes this much worse
- We have multiple signatures that prove validity in each certificate:
  - Signature on certificate itself
  - OCSP staple that proves that certificate is currently valid
  - Certificate Transparency log inclusion proves that certificate was from a trusted issuer

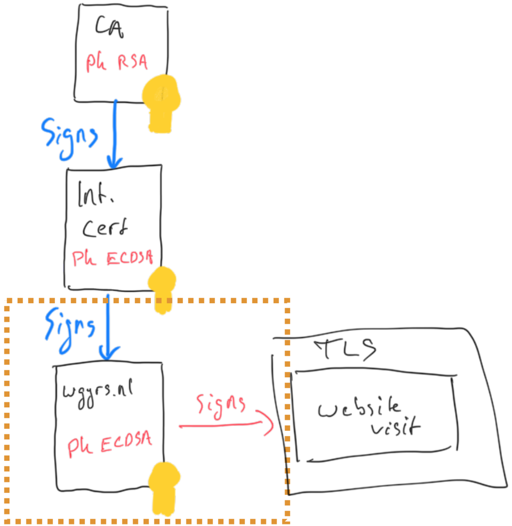
Can we do things in a smarter way?

Now is the time for redesigning the PKI

Abridged Compression for WebPKI Certificates

- Browser vendors control the root certificates that are included
- Step 1: Just ship the intermediate certificates as well
  - Client indicates to the server it has version N of the intermediate certificates list
  - Server omits intermediate certificate if present in list version N
  - Immediate savings: 1 certificate including 1 public key + 1 signature

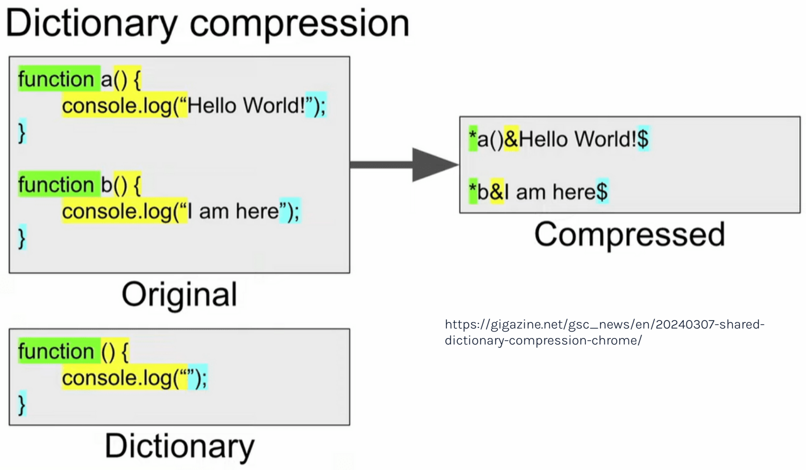
Sent every handshake



Dennis Jackson, Mozilla
https://datatracker.ietf.org/doc/draft-ietf-tls-cert-abridge/

Abridged Compression for WebPKI Certificates

- Certificates contain many common strings
  - policy urls, CA names, CT urls, extensions …
  - RFC 8879 already specifies certificate compression using zlib, brotli, zstd
- Step 2: Instead of applying compression algorithm directly, pre-train a compression dictionary based on sample certificates from all issuers
- Ship compression dictionary in browser



https://gigazine.net/gsc_news/en/20240307-shared-dictionary-compression-chrome/

https://datatracker.ietf.org/doc/draft-ietf-tls-cert-abridge/

Abridged Certificate Compression for TLS

- Step 3: compress certificates before sending using the pre-trained dictionary (if client up-to-date)

- Shipping compression dictionary out-of-band massively improves compression results
- Gain ~3000 bytes, i.e. space for 1 ML-DSA

- Remember that public keys and signatures themselves don't compress at all
- Security analysis very easy: just uncompress and you have the same TLS handshake

Scheme	Storage Footprint	p5	p50	p95
Original	0	2308	4032	5609
TLS Cert Compression	0	1619	3243	3821
Intermediate Suppression and TLS Cert Compression	0	1020	1445	3303
This Draft	65336	661	1060	1437
This Draft with opaque trained dictionary	3000	562	931	1454
Hypothetical Optimal Compression	0	377	742	1075

https://datatracker.ietf.org/doc/draft-ietf-tls-cert-abridge/

Merkle Tree Certificates

What if we build the PKI on Certificate Transparency's ideas, combined with OCSP?

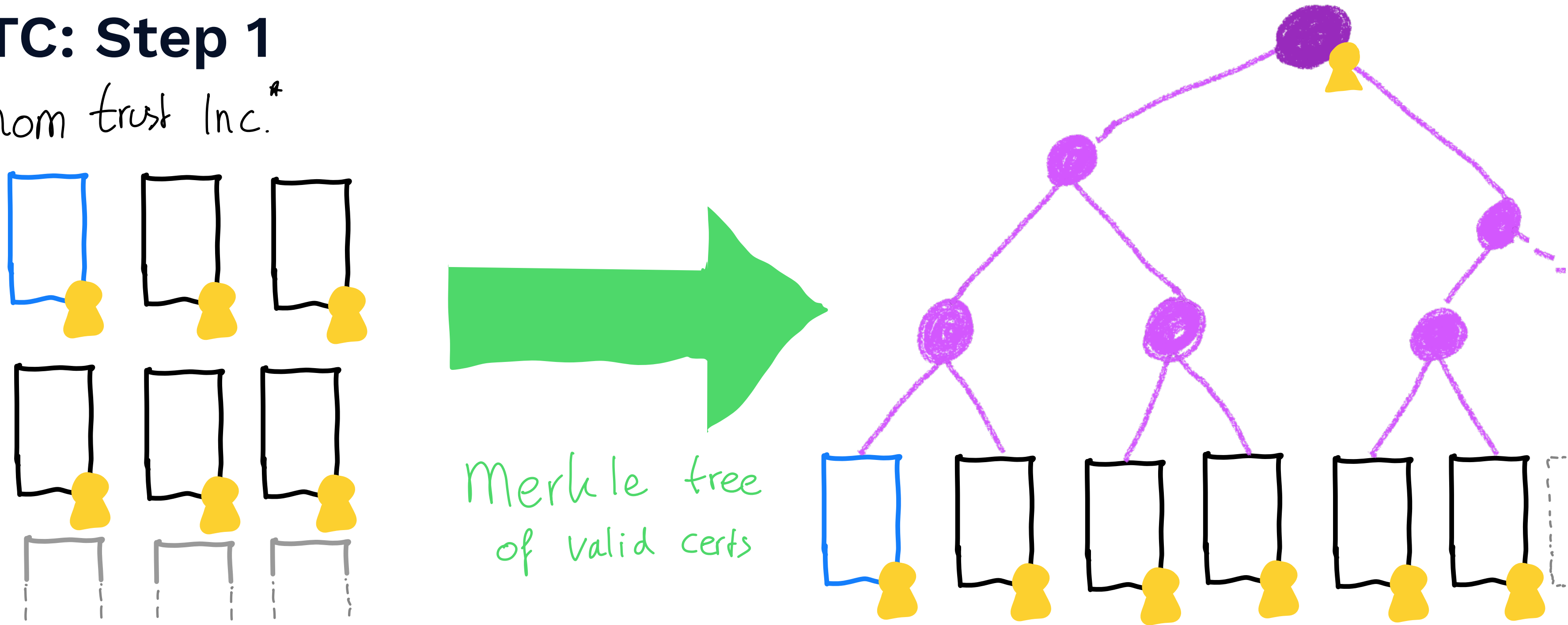
```
Transport Layer Security                                    D. Benjamin
Internet-Draft                                               D. O'Brien
Intended status: Experimental                                Google LLC
Expires: 5 September 2024                               B. E. Westerbaan
                                                             Cloudflare
                                                           4 March 2024

                  Merkle Tree Certificates for TLS
             draft-davidben-tls-merkle-tree-certs-02
```

https://datatracker.ietf.org/doc/draft-davidben-tls-merkle-tree-certs/

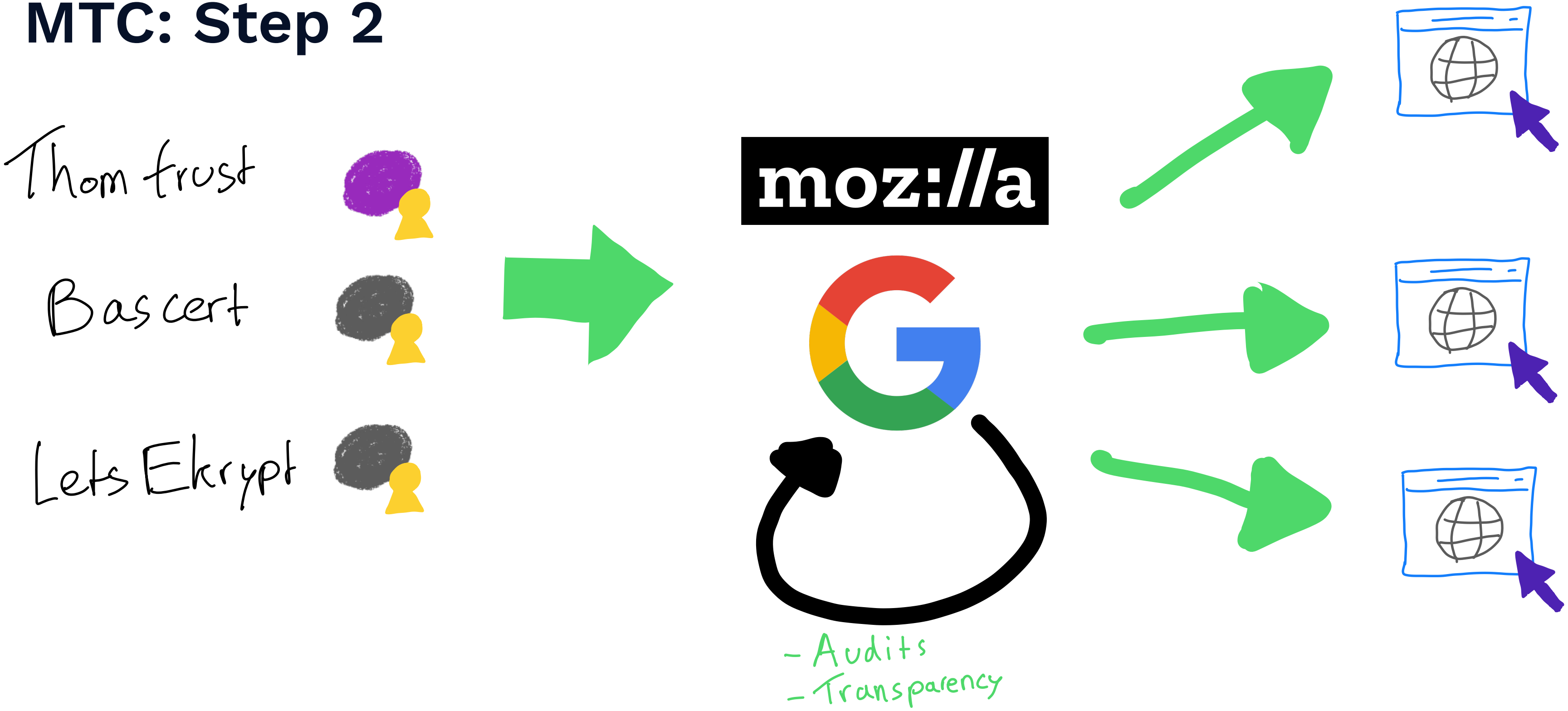
MTC: Step 1

Thom trust Inc.*

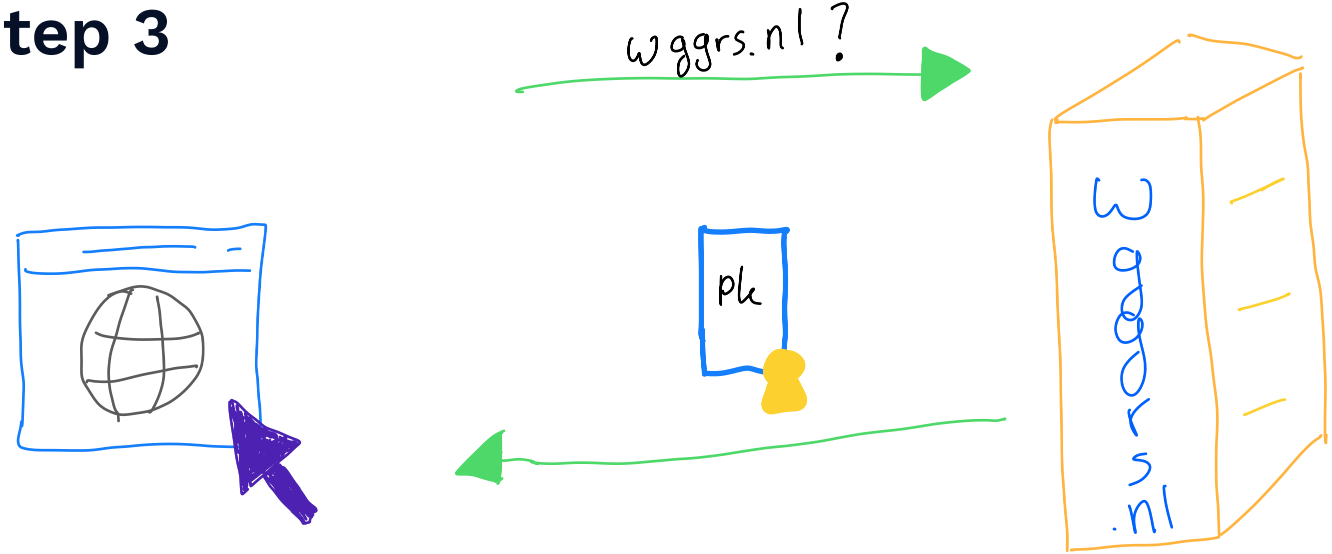


* Fund my Startup

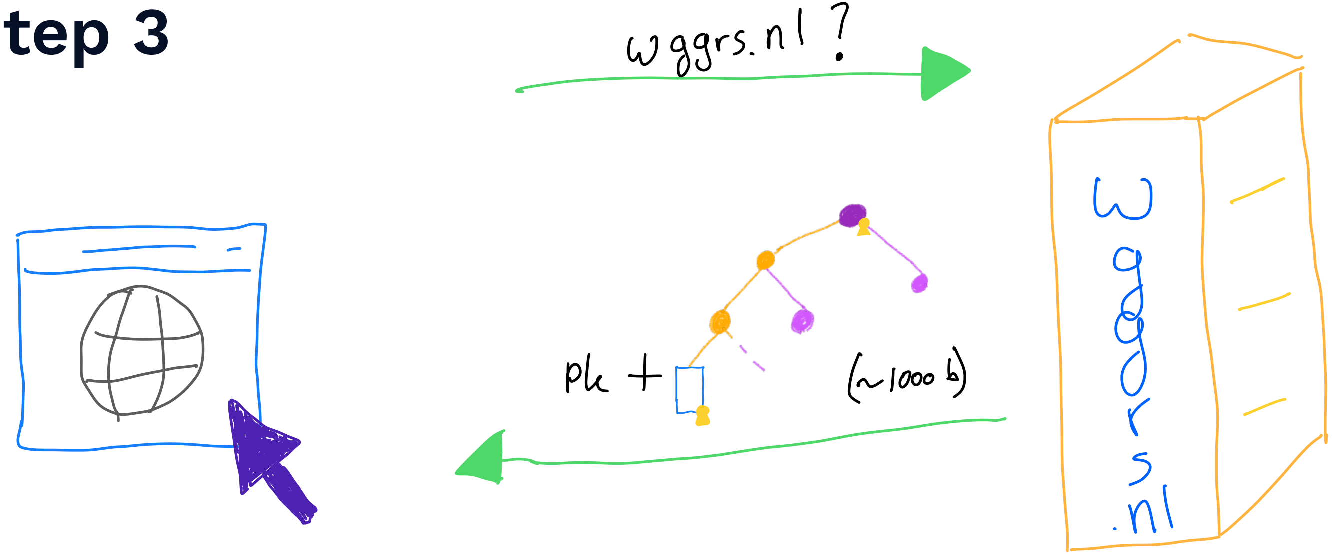
MTC: Step 2



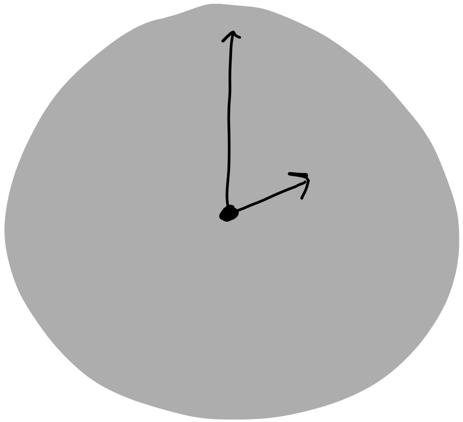
MTC: Step 3



MTC: Step 3



Merkle Tree Certificates



Repeat every hour

Merkle Tree Certificates

- Big changes necessary to every part of the ecosystem
  - Short-lived certificates
  - Webserver must continuously fetch the latest authentication paths
  - Clients must keep downloading currently valid tree heads
  - Automated certificate provisioning such as ACME [RFC8555] should help with this
- New trust model makes security analysis more complicated

- Both MTC and Abridged Compression designed for big deployments and publicly trusted CAs
  - What about IoT? What about ABN AMRO's internal stuff?

Save even more data?

- Handshake authentication still uses signatures, so ~3.5 kB (pk + sig) for Dilithium2
- KEMTLS: (implicitly) authenticate handshake by using key exchange instead
  - Put KEM public key in certificate / Merkle Tree Cert
  - Authentication in ~2 kB (ML-KEM 768)
  - BAT-KEM (non-NIST): ~1 kB (too slow keygen for general purpose)
  - Redesigns TLS handshake
  - IETF: draft-celi-wiggers-tls-authkem

- https://kemtls.org

Transitioning to PQ

- The transition to post-quantum means:
  - KEMs are less flexible than Diffie—Hellman
    - No non-interactive key exchange
  - PQ is bigger than ECC we got used to
  - Post-Quantum Signatures are big
- **Big changes to surrounding ecosystems might be necessary**
  - “Slapping another signature on it” is no longer a cheap solution
  - The WebPKI may see a big redesign
  - Even with the big redesign, we may still need **KEMTLS** (AuthKEM @ IETF) to mitigate the cost of the handshake signature to keep the slowdown under 10%