Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Freie Universität Berlin

Revisiting QUIC Handshakes and TLS Deployment: About Three Challenges

Marcin Nawrocki, Pouyan Fotouhi Tehrani, Raphael Hiesgen, Jonas Mücke, Thomas C. Schmidt, Matthias Wählisch

{marcin.nawrocki, jonas.muecke, m.waehlisch}@fu-berlin.de
pouyan.fotouhi.tehrani@fokus.fraunhofer.de
{raphael.hiesgen, t.schmidt}@haw-hamburg.de

Methods, more results, more details? Read our paper! :)

On the Interplay between TLS Certificates and QUIC Performance

Marcin Nawrocki
marcin.nawrocki@fu-berlin.de
Freie Universität Berlin
Germany

Pouyan Fotouhi Tehrani
pft@acm.org
Weizenbaum Inst., Fraunhofer FOKUS
Germany

Raphael Hiesgen
raphael.hiesgen@haw-hamburg.de
HAW Hamburg
Germany

Jonas Mücke
jonas.muecke@fu-berlin.de
Freie Universität Berlin
Germany

Thomas C. Schmidt
t.schmidt@haw-hamburg.de
HAW Hamburg
Germany

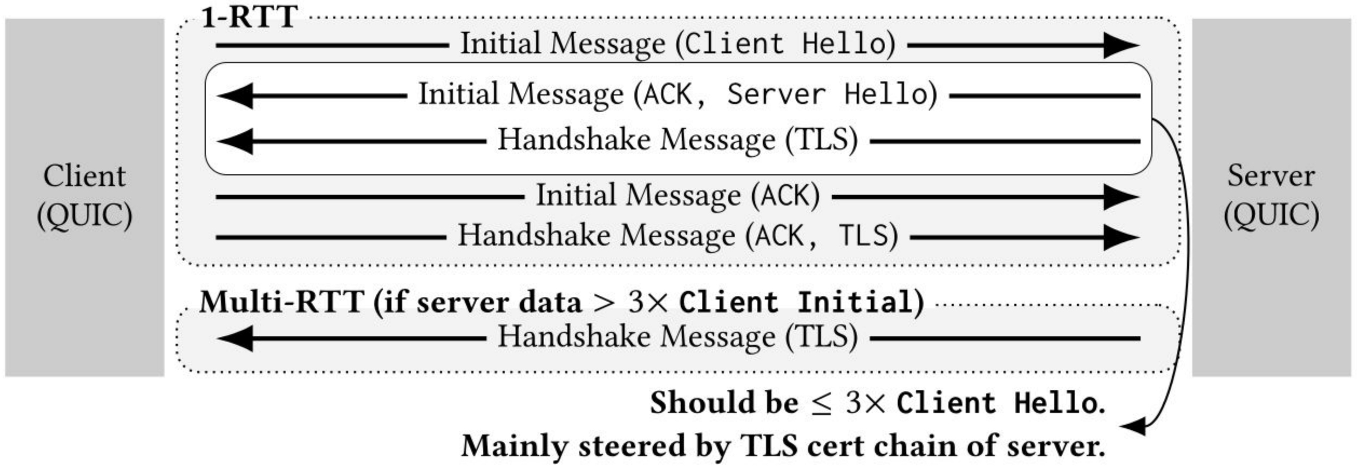
Matthias Wählisch
m.waehlisch@fu-berlin.de
Freie Universität Berlin
Germany

[https://doi.org/10.1145/3555050.3569123]

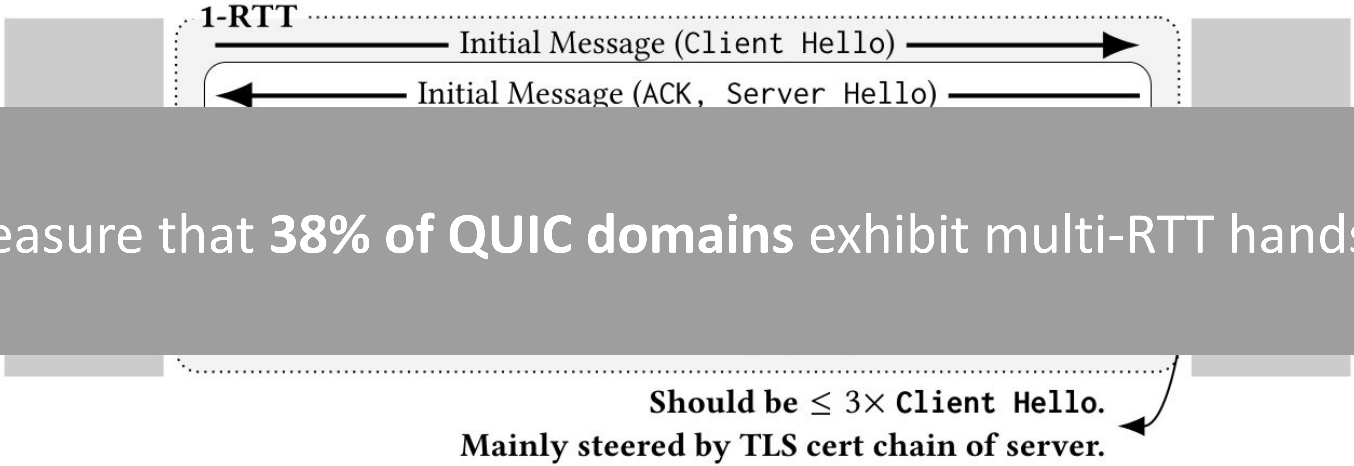
Large TLS data triggers multiple RTTs.

QUIC Handshake Challenge 1

Multi-RTT prevents amplification attacks but is inefficient.

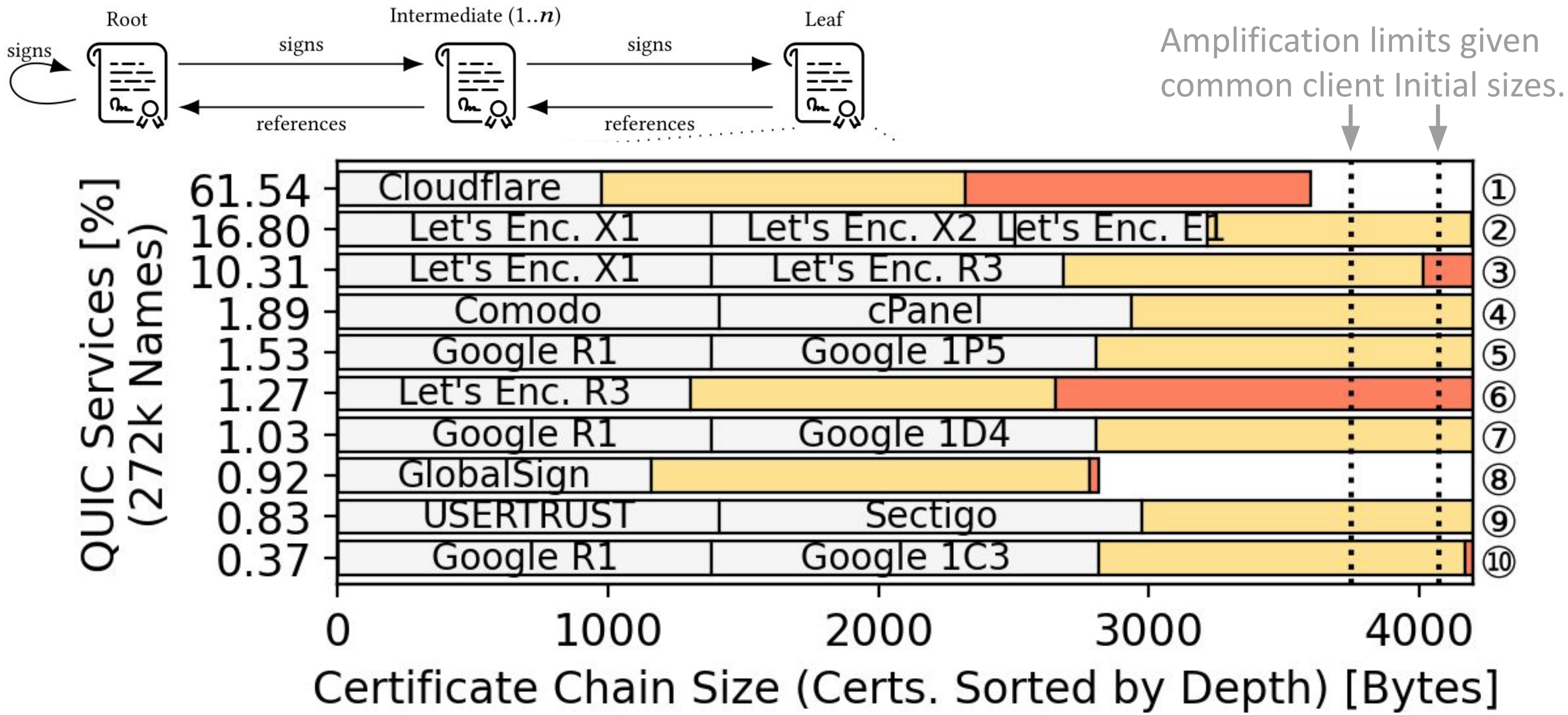


Multi-RTT prevents amplification attacks but is inefficient.



We measure that **38% of QUIC domains** exhibit multi-RTT handshakes.

Non-leaf certificates are large. Even median-sized chains are likely to exceed anti-amplification limits.



Will future QUIC extensions make the situation worse?

- draft-ietf-quic-multipath: A Multipath QUIC connection starts with a regular QUIC handshake. Adding new paths does not require additional certificate exchanges. Same challenge ☺️.

- draft-ietf-quic-version-negotiation: Compatible Version Negotiation prevents extra RTT because of `VERSION_NEG` packets. The subsequent handshake process is as usual and may still require Multi-RTT due to large TLS data. Same challenge ☺️.

- draft-ietf-tls-hybrid-design: Hybrid key exchange in TLS 1.3 makes QUIC connections quantum-proof. Recent implementations need additional 800 bytes for new secrets in `SERVER_HLO`. Even worse ☹️.

Observation 1: QUIC was designed for low latency (1-RTT), but multi-RTT are needed in the wild due to **cert sizes**.

Should we encourage TLS certificate compression?

Tackles only symptoms but is effective if available in client and server implementations.

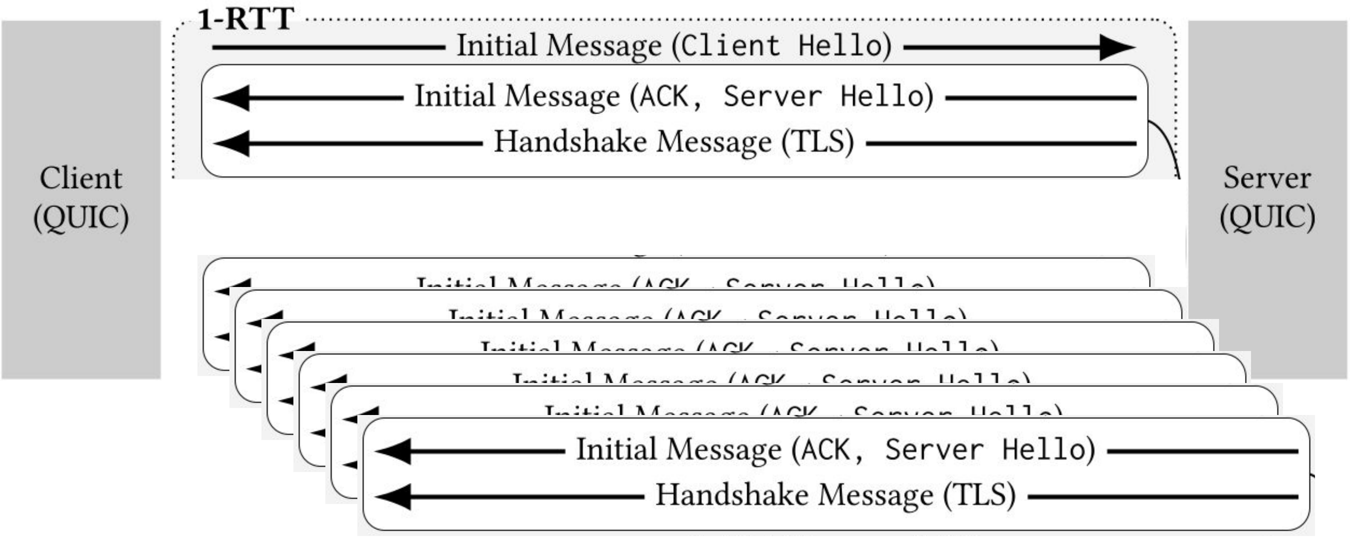
Should we propose Best Current Practices for TLS in QUIC?

Make use of elliptic curve crypto, limit max. chain depths, max. SANs, …

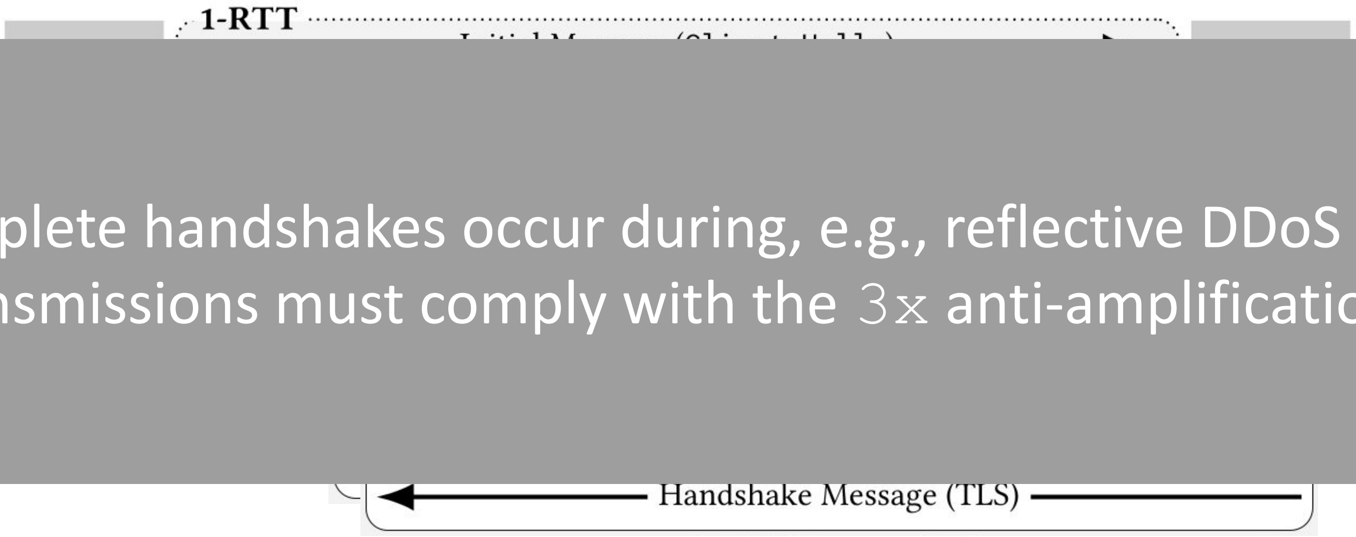
No efficient `resend` strategy exists.

QUIC Handshake Challenge 2

Servers that experience incomplete handshakes assume packet loss and resend packets, which can lead to high amplification factors.

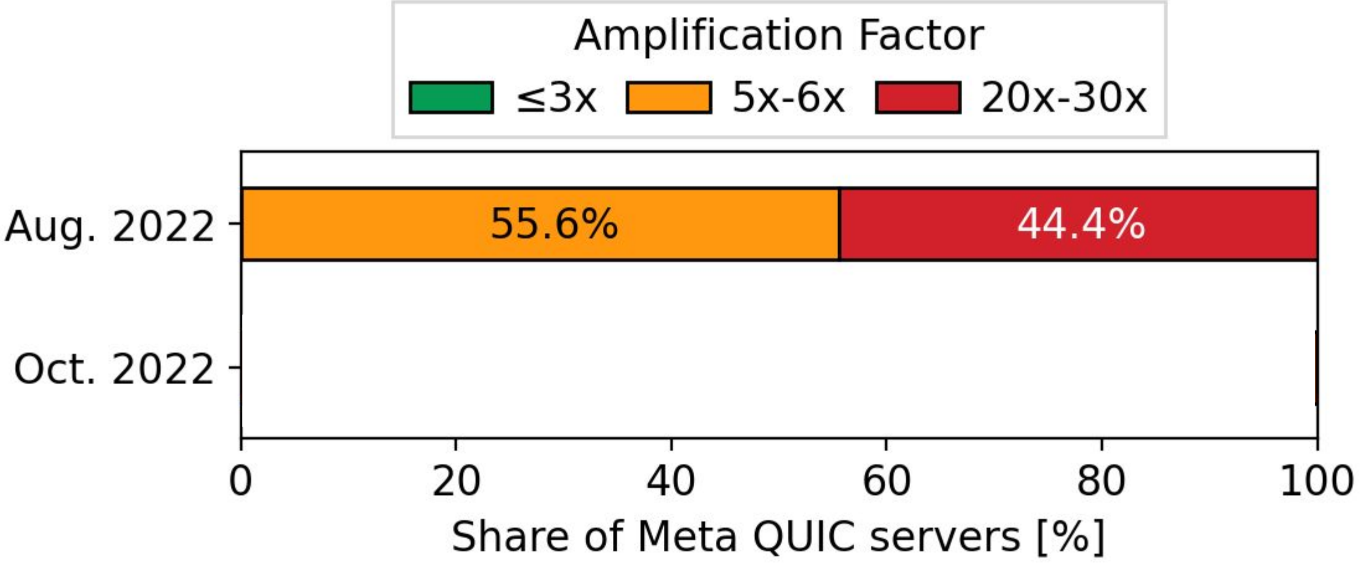


Servers that experience incomplete handshakes assume packet loss and resend packets, which can lead to high amplification factors.

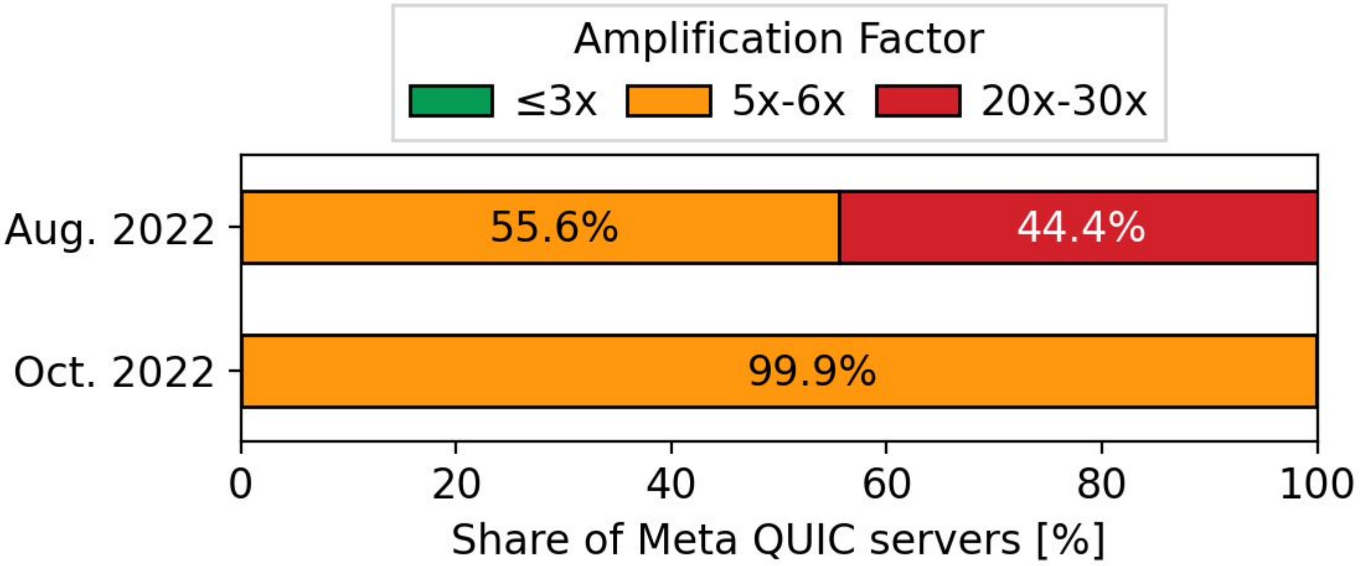


Incomplete handshakes occur during, e.g., reflective DDoS attacks.
Retransmissions must comply with the `3x` anti-amplification limit.

Triggering incomplete handshakes with Meta PoPs.



Triggering incomplete handshakes with Meta PoPs.



Observation 2: Resends easily contribute to reaching the amplification limit.

Do we need a different strategy for incomplete handshakes?

An alternative strategy for incomplete handshakes could be:

i) server resends TLS data once, if data fits below `3x` anti-amplification limit;

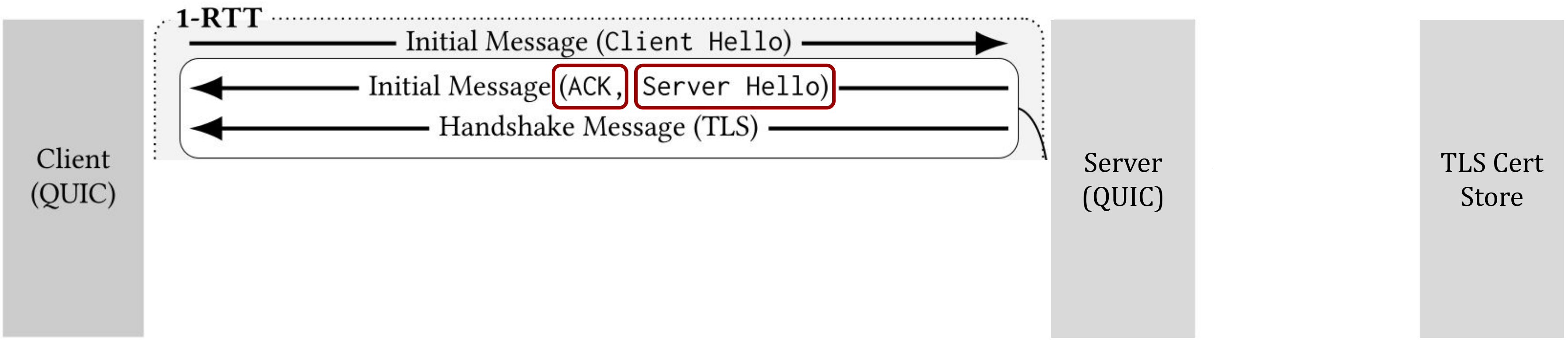
ii) then, server starts to validate the client with small probes, *e.g.*, `PING`.

CDN setups optimize for low delays but lead to larger handshake data.

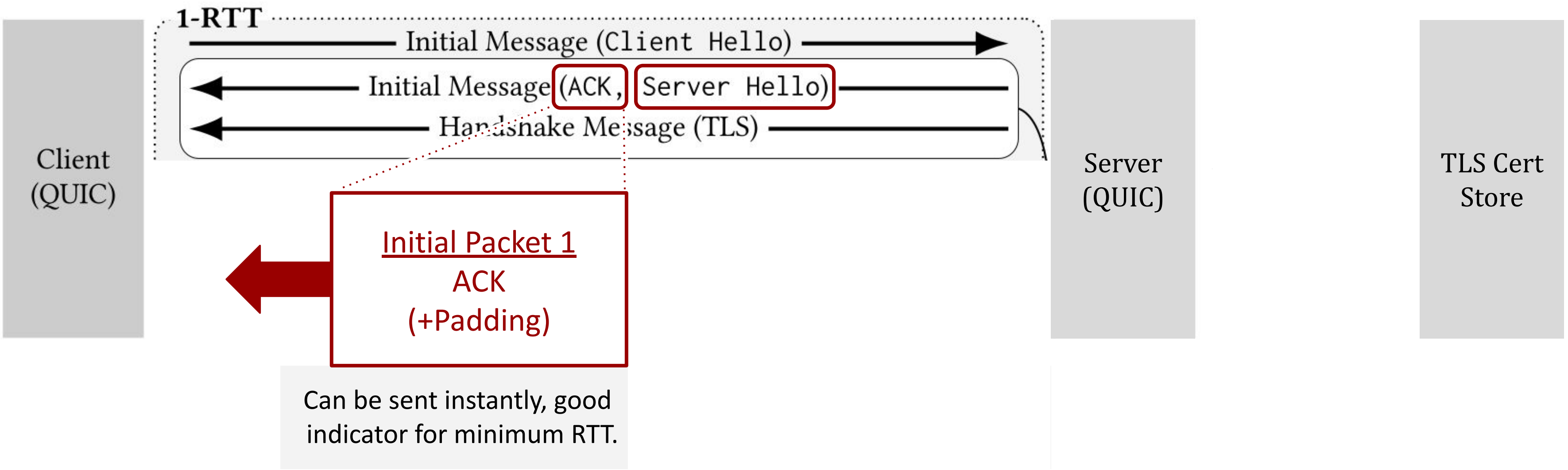
QUIC Handshake Challenge 3

In several CDN deployments, the QUIC server can be separate from the process that has access to TLS material. This may add delay and disturb the client RTT estimation.

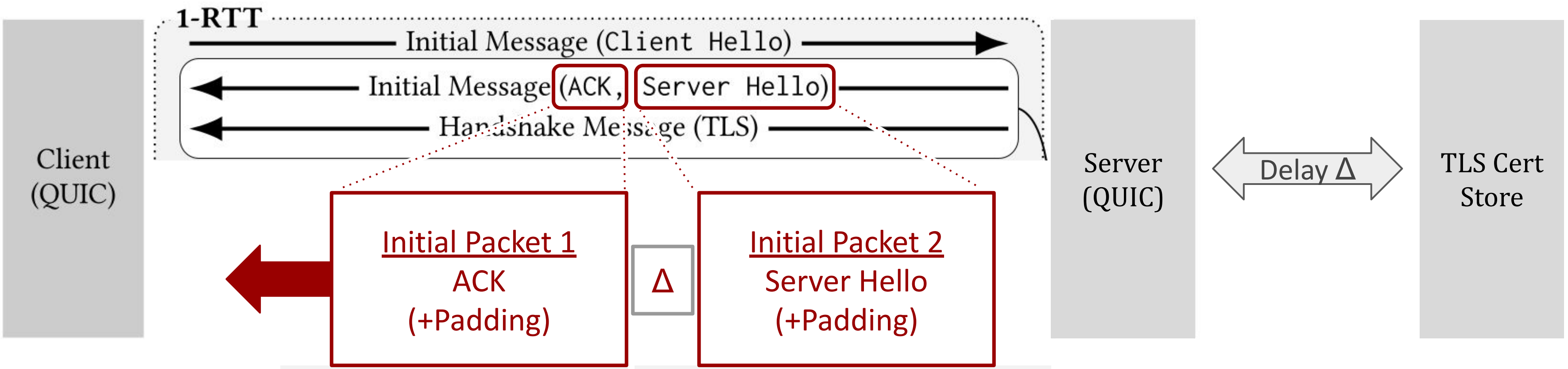
CDNs deal with this by splitting server Initials …



CDNs deal with this by splitting server Initials … and responding instantly only with the ACK …



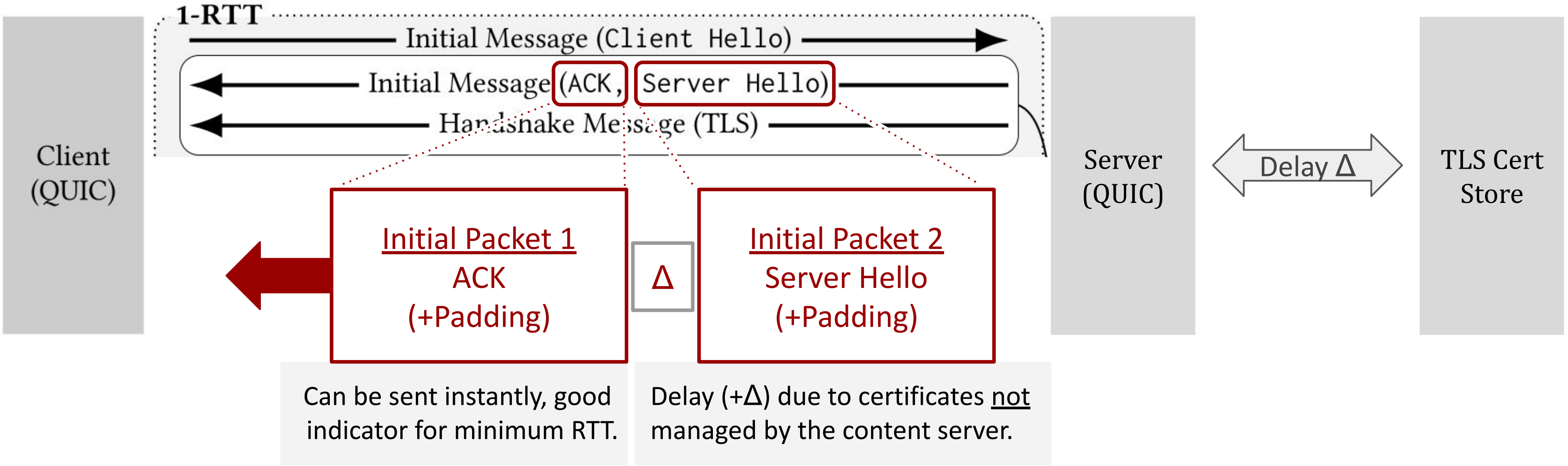
… and then retrieve and deliver the certificate.



Can be sent instantly, good indicator for minimum RTT.

Delay (+Δ) due to certificates not managed by the content server.

Pro: This keeps ProbeTimeouts for RTT estimation low.
Con: But it leads to larger handshake data (>3x).



Observation 3: QUIC design does not account for distributed certificate management, which skews minimal RTT estimations.

How to enable a precise RTT estimation for all deployments?

E.g., sending endpoints could tag delayed packets and
receiving endpoints could exclude such packets from RTT estimations.

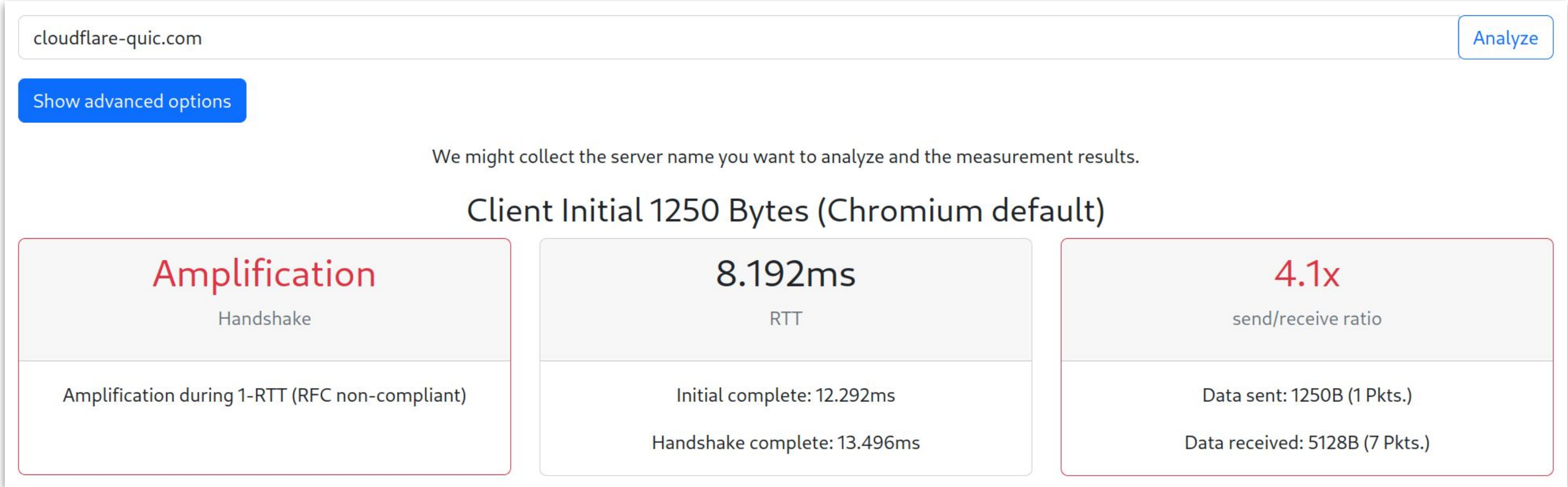
Conclusion – Where can the QUIC WG best help?

CDN setups optimize for low delays but lead to larger handshake data.
→ Would tagging of delayed packets enable a precise RTT estimation for all deployments?

No efficient resend strategy exists.
→ In case of incomplete handshakes, would small probes help instead of large resends?

Large TLS data triggers multiple RTTs.
→ Should we encourage TLS certificate compression?
→ Should we propose Best Current Practices for TLS in QUIC?

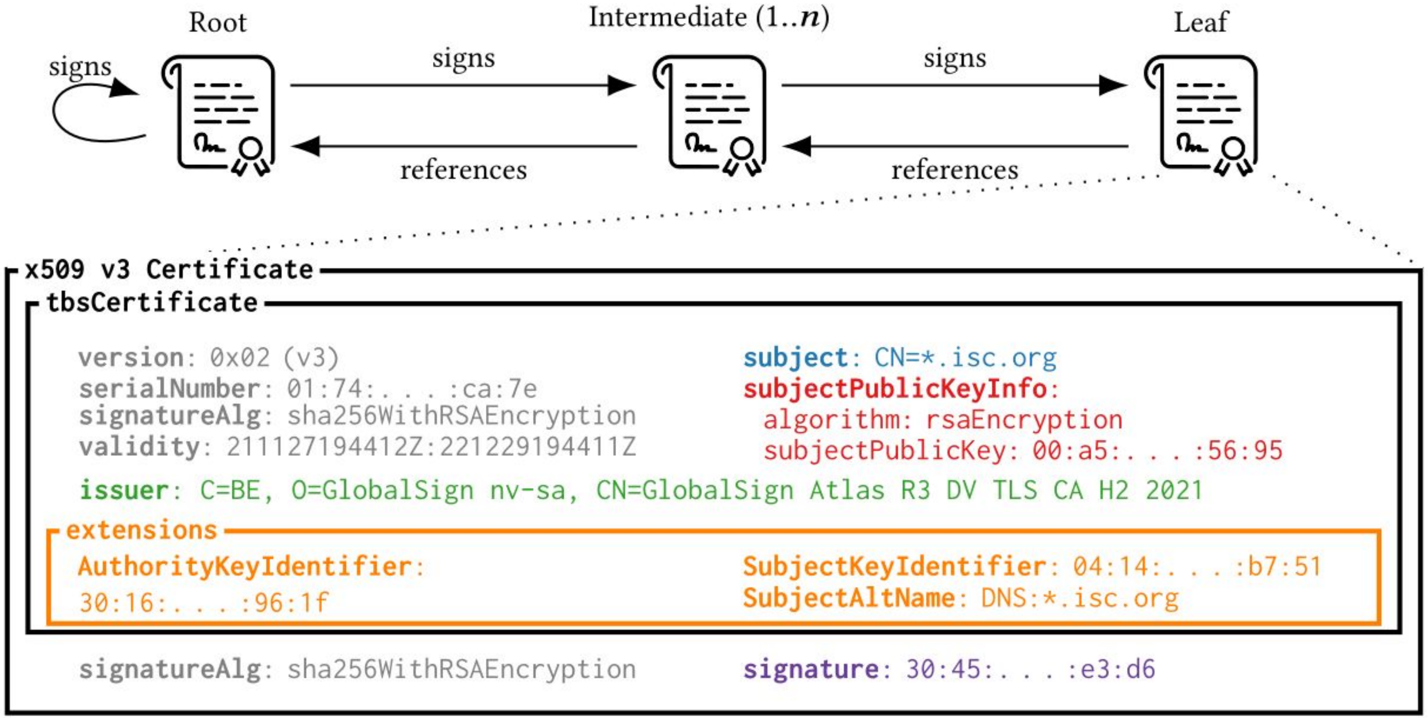
QUIC Handshake Classification API (IETF 115 Hackathon)



[https://understanding-quic.net]

Backup

TLS data matters. Chains, large keys, alternative names, …



Calculating the `RESEND` bytes

● TLS certificate chain (1 non-leaf, elliptic curve):	2200 bytes
● With TLS compression (0.73x):	1600 bytes
● TLS Server Hello and QUIC headers (300 bytes):	1900 bytes
● Resending (2x) …	3800 bytes
● … vs Client Initial (1300 bytes, 3x)	3900 bytes

PINGs during handshake will be probably padded…

QUIC MUST NOT be used if the network path cannot support a maximum datagram size of at least 1200 bytes.

A client MUST expand the payload of all UDP datagrams carrying Initial packets to at least the smallest allowed maximum datagram size of 1200 bytes [...]

[...] a server MUST expand the payload of all UDP datagrams carrying ack-eliciting Initial packets to at least the smallest allowed maximum datagram size of 1200 bytes.

Ack-eliciting packet: A QUIC packet that contains frames other than ACK, PADDING, and CONNECTION_CLOSE.

```
▸ User Datagram Protocol, Src Port: 443, Dst Port: 56062
▾ QUIC IETF
  ▸ QUIC Connection information
    [Packet Length: 70]
    1... .... = Header Form: Long Header (1)
    .1.. .... = Fixed Bit: True
    ..00 .... = Packet Type: Initial (0)
    .... 00.. = Reserved: 0
    .... ..00 = Packet Number Length: 1 bytes (0)
    Version: 1 (0x00000001)
    Destination Connection ID Length: 20
    Destination Connection ID: 6d99b1113ca9872e256318689c9a95cf76a5b259
    Source Connection ID Length: 20
    Source Connection ID: 4ec7f8591c9dcc9c25b6d7b90ae8810f522ad7af
    Token Length: 0
    Length: 21
    Packet Number: 1
    Payload: 45194f693c12f2fd6f44f4c9772b628ea13bbeaf
  ▸ PING
  ▸ PADDING Length: 3
▾ QUIC IETF
    [Packet Length: 1130]
```

Triggering incomplete handshakes with Meta PoPs.