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

ABSTRACT

In this paper, we revisit the performance of the QUIC connection setup and relate the design choices for fast and secure connections to common Web deployments. We analyze over 1M Web domains with 272k QUIC-enabled services and find two worrying results. First, current practices of creating, providing, and fetching Web certificates undermine reduced round trip times during the connection setup since sizes of 35% of server certificates exceed the amplification limit. Second, non-standard server implementations lead to larger amplification factors than QUIC permits, which increase even further in IP spoofing scenarios. We present guidance for all involved stakeholders to improve the situation.

CCS CONCEPTS

• **Networks** → **Transport protocols**; *Public Internet*; **Network protocol design**; **Network measurement**; • **Security and privacy** → **Web protocol security**.

ACM Reference Format:
Marcin Nawrocki, Pouyan Fotouhi Tehrani, Raphael Hiesgen, Jonas Mücke, Thomas C. Schmidt, and Matthias Wählisch. 2022. On the Interplay between TLS Certificates and QUIC Performance. In *The 18th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '22), December 6–9, 2022, Roma, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3555050.3569123

1 INTRODUCTION

The QUIC protocol [23] was designed to improve Web performance and reduce access latency [7, 45] while keeping communication confidential [8]. A key approach is the reduction of initial roundtrip times by integrating the QUIC handshake with the TLS 1.3 handshake and coalescing multiple QUIC packets into one UDP datagram. At the same time, security concerns about the UDP-based QUIC protocol demanded to limit the amplification potential, *i.e.,* the byte ratio of the server answer to the client request, for the initial reply to an (unauthenticated) client.

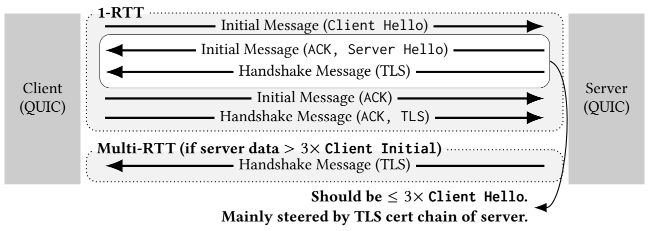

1-RTT
Initial Message (Client Hello)
Initial Message (ACK, Server Hello)
Handshake Message (TLS)
Initial Message (ACK)
Handshake Message (ACK, TLS)
Client (QUIC)
Server (QUIC)
Multi-RTT (if server data > 3× Client Initial)
Handshake Message (TLS)
Should be ≤ 3× Client Hello.
Mainly steered by TLS cert chain of server.

Figure 1: In QUIC handshakes, server replies are limited to 3× the size of the client `Initial` until the client is verified.

In QUIC, the details of a connection handshake depend on a variety of factors: version negotiation, QUIC retry option, packet coalescing, and the size and compression of TLS certificates. In this work, we focus on the latter because it has great relevance for the handshake process, see Figure 1. Using active and passive measurements we observe significant effects on latency, amplification, and protocol behavior in current deployments.

In detail, we contribute the following.

(1) Background on the interplay between the QUIC handshake and TLS certificates and prior work (§ 2).
(2) A measurement method to systematically analyze the problem space (§ 3).
(3) Analysis of QUIC server behaviors for different sizes of client `Initial` messages. The majority of QUIC servers incorrectly amplify handshakes or require multiple RTTs, even for common `Initial` sizes used by Web browsers (§ 4.1).
(4) An in-depth study of QUIC handshake behavior that shows that multi-RTT handshakes are caused by large certificates and missing packet coalescence. Furthermore, some certificates unnecessarily contain cross-signed certificates instead of self-signed versions or include their trust anchors (§ 4.2).
(5) Empirical results highlighting the benefits of certificate compression during the handshake. 99% of certificate chains would remain below the allowed amplification factor (§ 4.2).
(6) A major reason why large amplification factors may appear during connection setups in the wild. For large CDNs, we observe up to 45× amplification for spoofed handshakes (§ 4.3).
(7) Guidance to improve the situation (§ 5) and our artifacts, which are publicly available (Appendix A).

2 BACKGROUND & RELATED WORK

In this section, we briefly recap the QUIC protocol mechanics of the connection setup, introduce challenges of TLS certificates that undermine fast setups, and discuss related work.

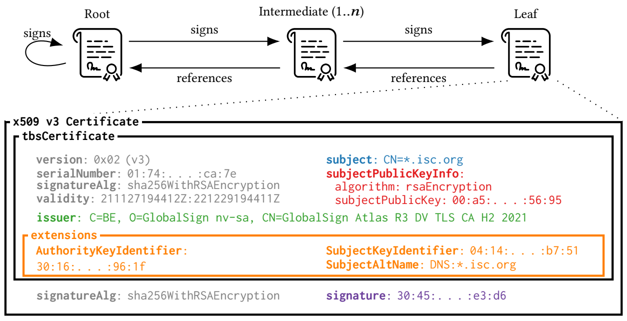
The QUIC handshake and amplification mitigation. QUIC [23] was designed to provide low latency, reliability, and security on top of UDP. A crucial part is the initial connection setup, which should be fast [51] and prevent attacks related to amplification [41, 43] or state exhaustion [39]. For this purpose, the QUIC handshake integrates TLS within the protocol handshake. A client starts with an `Initial` that is answered by an `Initial` from the server and a `Handshake` packet that can be sent in a single UDP datagram (*packet coalescence*). The client confirms receipt with an `Initial ACK` and then sends an its `Handshake` message. The server resends unconfirmed `Initial` and `Handshake` packets. Protection against state exhaustion is achieved when a server uses `RETRY` packets but such protection is rarely deployed [39].

To prevent amplification attacks, a server must not reply with more bytes than the QUIC *anti-amplification factor* allows until the client IP address is verified. RFC 9000 [23] limits the data size from the server to 3× the bytes that have been received in the client `Initial`, see Figure 1, and includes padding and resent bytes [17]. After the server validates the client by a complete roundtrip, it is free to send any amount of data. The factor of three is low compared to the amplification potential of other protocols [41, 43]. We recap the IETF design of the threshold in more detail in Appendix C.

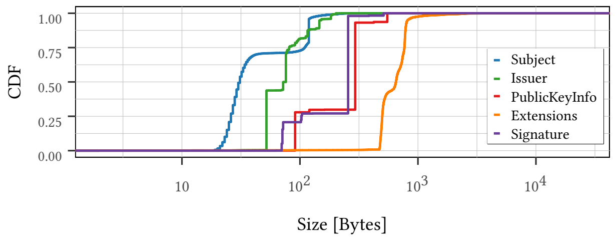
QUIC TLS connection setup. QUIC integrates TLS 1.3 [40] to cater for authenticated confidentiality and integrity [48]. A TLS 1.3 handshake is initiated by a client sending its supported cipher suites, key parameters, and other metadata in the first `Initial` message to the server, which in turn replies with its own parameters and an X.509 certificate [6] used to authenticate its identity [48, §4.4.].

In contrast to TLS over TCP, the sum of *first responses* of a QUIC server must not be larger than the anti-amplification factor. This limit reduces the amplification attack surface but poses a challenge for benign QUIC peers to achieve the goal of low latency and low connection overhead. Either the client sends an `Initial` that is large enough to allow the server to accommodate its reply within the anti-amplification limit, or the server adapts responses to be small enough. Please note that the size of a server reply is mainly determined by its certificate, *i.e.,* the complete certificate chain sent by the server. Figure 2(a) illustrates the structure of a TLS certificate.

Popular browsers use between 1250 and 1357 bytes in the `Initial` message, which can easily conflict with common sizes of server certificate (see § 4 for details). Depending on public key and signature algorithms in use, sizes of issuer and subject names, as well as extensions (*e.g.,* subject alternative names), the total size of a certificate may vary by an order of magnitude. Figure 2(b) depicts the size distribution from our data corpus; certificate extension fields followed by signature and public key fields are the most space consuming in certificates. A server can apply optimizations to its own certificate sizes, but it has no control over intermediate certificates that it delivers as part of the certificate chain of trust. To compress the entire chain, TLS 1.3 provides certificate compression [12]. To take effect, client and server must support compression. While


Root
Intermediate (1..n)
Leaf
signs
signs
signs
references
references
x509 v3 Certificate
tbsCertificate
version: 0x02 (v3)
serialNumber: 01:74:. . . :ca:7e
signatureAlg: sha256WithRSAEncryption
validity: 211127194412Z:221229194411Z
subject: CN=*.isc.org
subjectPublicKeyInfo:
algorithm: rsaEncryption
subjectPublicKey: 00:a5:. . . :56:95
issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Atlas R3 DV TLS CA H2 2021
extensions
AuthorityKeyIdentifier:
30:16:. . . :96:1f
SubjectKeyIdentifier: 04:14:. . . :b7:51
SubjectAltName: DNS:*.isc.org
signatureAlg: sha256WithRSAEncryption
signature: 30:45:. . . :e3:d6

(a) X.509 certificate



(b) Size distribution

Figure 2: Example of a TLS certificate and our observed distribution for various X.509 certificate field sizes.

the adoption of TLS 1.3 is well studied [15], analysis of certificate compression deployment is not included in prior research.

QUIC performance and adoption. QUIC provides good performance [2, 4, 5, 25] and can outperform TCP. Prior work suggests that some security trade-offs were specifically made in favor of improved latency [32]. The handshake, however, can suffer from additional latency if client and server do not agree on a version directly [11].

Prior deployment studies mainly focus on the availability of QUIC services. QUIC adoption started before the finalization of the standard [33, 46, 49] and continues since then [52], led by hypergiants [38, 42]. DNS over QUIC lacks wide adoption and exhibits inefficient handshakes due to large certificates if `Session Resumption` is not used [26, 27]. Most closely related is [35], showing that 40% of QUIC handshakes with uncompressed certificates may trigger an additional roundtrip, based on data from a specific CDN. To the best of our knowledge, this paper is the first that systematically assesses the impact of TLS certificates on QUIC performance, leveraging comprehensive measurements.

3 MEASUREMENT METHOD AND SETUP

We search for (*i*) common HTTPS services and (*ii*) QUIC-services, to collect related TLS certificates and compare performance of protocol design and deployment choices. In this paper, we use the term *service* to quantify the number of domains served via a specific protocol, irrespectively of whether these domains are delivered by the same IP host, *i.e.,* we present a domain-centric perspective. The point of departure for our scans is the Tranco list [29] generated on September 10, 2022, since the Tranco list provides a good compromise [44] between reflecting popularity and robustness. Subsequently, we scan 1M domain names to broadly capture what clients receive when contacting a web domain.

When conducting our measurements, we leverage existing tools where possible and minimize extensions to achieve maintainability and ease reproducibility. Unfortunately, there is no single tool available that implements all necessary features. We present an overview of our toolchain in Appendix A, Figure 10.

3.1 TLS Certificate Scans via HTTPS

Not all names in the Tranco list resolve to web servers that allow for TLS over TCP connections. For each name in the list, first, we try to resolve IPv4 addresses using Google public resolver `8.8.8.8`. Upon success, we then try to establish HTTP connections on ports 80 and 443 and follow any redirects using HTTP(S) (status code 3xx) and HTML (`meta` tag with `http-equiv` attribute). For every secure domain, including all redirects, we collect TLS certificates.

We were able to resolve 976k (out of 1M). For 13k names, the domain query returned `SERVFAIL`, 9k could not be resolved (`NXDOMAIN`), and the remaining either timed out (10s) or refused the answer (`REFUSED` [1, 37]). About 866k names returned an IP address (`A` record). For every domain name which resolved to an IP address we tried to establish an HTTP connection on both ports 80 (HTTP) and 443 (HTTPS). After following redirects, we collected 821k unique certificates for more than 1.1M names along the redirection path.

3.2 QUIC Scans

We analyze QUIC handshakes in two scenarios: (*i*) a complete handshake including client verification and (*ii*) an incomplete handshake imitating an unverified client, *e.g.,* when clients spoof IP addresses.

Complete handshakes. We scan all domains discovered during our certificate collection via HTTPS and assign each successful handshake to one of the four groups:

(1) **1-RTT (optimal):** Handshakes that complete within 1-RTT and comply with the anti-amplification limit.
(2) **RETRY (less efficient):** Handshakes that require multiple RTTs because the `Retry` option is used [23, §8.1.].
(3) **Multi-RTT (unnecessary):** Handshakes that do not use `Retry` but require multiple RTTs because of large certificates.
(4) **Amplification (not RFC-compliant):** Handshakes that complete within 1-RTT but exceed the anti-amplification limit.

To conduct QUIC handshakes and assign groups, we use `quicreach` [36], extended by `RETRY` support. We find 272k QUIC services (~25%). To investigate the effect of client `Initial` sizes on server handshake behavior we vary the client `Initial` size between 1200 bytes (mandated minimum [23]) and 1472 bytes (dictated by our MTU since QUIC forbids fragmentation) in steps of 10 bytes. Handshakes targeting the same domain service pause 30 minutes to avoid side effects such as DDoS mitigation.

`quicreach` does not provide access to certificates nor does the underlying stack support certificate compression. We rescan with (*i*) `QScanner` [50] to access TLS certificates sent over QUIC and (*ii*) extend `quiche` [14] to support three popular TLS compression algorithms in QUIC.

In the majority of cases (96.7%), we find that the same TLS certificates are used in both QUIC and HTTPS deployments for the same domains, which confirms prior work [52]. For the remaining 3.3% of QUIC services, certificates differ from TLS over TCP. These differences are mainly due to certificate rotations during the period of time between our HTTPS and QUIC scans, leaving only 0.47% of QUIC services with different certificates because of other reasons. To sanitize inconsistent data, we decide to base our QUIC certificate analysis on the TLS certificates gathered via HTTPS.

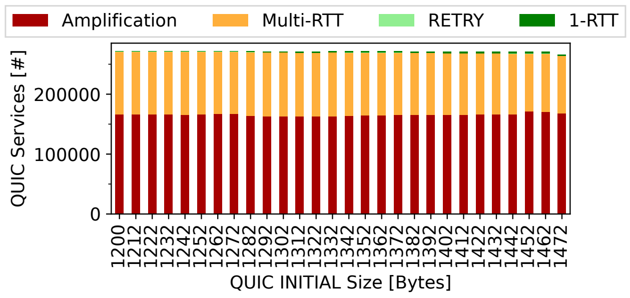


Figure 3: Influence of QUIC `Initial` sizes on the QUIC handshake. With respect to all names, we find almost no effect.

Table 1: Comparison of QUIC INITIAL packet sizes and support for TLS 1.3 certificate compression in popular browsers.

		Init. Size	Compression		
Browser	Version	[Bytes]	Algorithm[3]	Rate[4]	Services[5]
Firefox	`101.x`	1357	–	–	–
Chromium-based[1]	`105.x`	1250[2]	brotli	73%	96%
Safari (macOS)	`15.5`	no QUIC	`zlib`	74%	0.05%
			`zstd`	72%	0.05%

[1] Chrome `102.x`, Brave `V1.39`, Vivaldi `5.3.x`, Edge `102.x`, Opera `88.0.x`.
[2] Recently reduced from 1350 [13]. [3] Tested with TLS 1.3 in TCP.
[4] Mean rate observed by our Quiche client. [5] Out of 272k QUIC services.

Incomplete handshakes. To analyse the performance when a client successfully initiates but does not complete a handshake, *e.g.,* because of malicious activities, we conduct two measurements. First, we collect QUIC backscatter from a telescope during January 2022. Since telescopes do not emit any traffic, we can observe server behavior to non-responding, spoofed client IP addresses. Here, we group QUIC traffic by major content providers and source connection IDs (SCIDs). Second, we send a single `Initial` of 1252 bytes to the servers without sending `ACK` messages, using ZMap [9].

4 RESULTS

In this section, we (*i*) present our analysis of complete handshakes, (*ii*) study TLS certificates as potential reason for performance drawbacks in more detail, and (*iii*) show results that reveal QUIC amplification potentials in the wild.

4.1 Classifying QUIC Handshakes

Overview. Figure 3 shows the absolute number of handshakes types for all QUIC-reachable names, depending on the `Initial` size. For an `Initial` size of 1362 bytes, which is similar to common browser default values (see Table 1), we find that 61% of handshakes are classified as amplifying and 38% as requiring multiple RTTs.

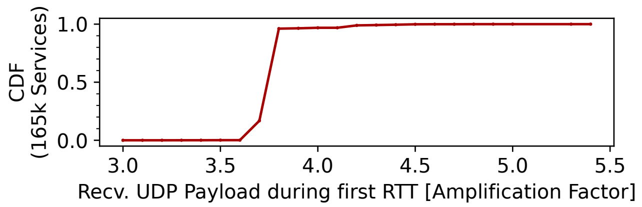


Figure 4: Amplification factor during first RTT. For complete client handshakes, the amplification is relatively small.

Worryingly, the `Retry` and 1-RTT handshakes account for only 0.07% and 0.75%, respectively. This means that a priori DoS protection and fast handshakes are rare, unveiling that the QUIC design goals have not been met in the wild, yet.

We now investigate the effect of different `Initial` sizes. We find that amplifying handshakes occur independently of the `Initial` size. However, we observe an interdependence between multi-RTT and 1-RTT handshakes. With larger `Initials`, multi-RTT handshakes are less likely and 1-RTT handshakes more likely (de- and increase by ∼1%). This nicely illustrates the performance impact of the interplay between `Initial` sizes and deployed certificate sizes.

We also observe that the reachability of QUIC services is reduced by 1.2% for large `Initials`, as indicated by the decreased height of the stacked bars. Interestingly, this effect is more pronounced for top-ranked services (not shown). The top 1k and top 10k domains are seeing a 25% and 12% drop in reachability, respectively. We argue that this corresponds to load-balancer deployments that are more likely to be used for very popular names. Load-balancers utilize packet tunneling to distribute the load across multiple, redundant server instances. Packet encapsulation used during tunneling adds bytes due to additional headers, which then exceed the local MTU. Our observations of reachability issues comply with prior measurements [28]. Other than that, we find little differences across ranks, compare Appendix D.

1-RTT exceeding anti-amplification limit. Independently of the `Initial` size, the majority of handshakes exceed the anti-amplification limit in the first RTT. We calculate the amplification factor for our default `INITIAL` scans of 1362 bytes by dividing UDP payload bytes received by the UDP payload bytes sent by the client. Figure 4 shows the amplification distribution. The amplification factor, although exceeded, remains relatively small below 6x.

Cloudflares missing coalescence explains amplification. Based on TLS information and additional IP prefix mapping, we find that 96% of the amplifying handshakes are completed with Cloudflare servers and subject to the same implementation behavior. Surprisingly, we observe exactly 2462 superfluous QUIC padding bytes for ≈157k handshakes. In these cases, although the TLS data can vary in size, the remaining QUIC bytes are constant in size. Cloudflare servers do not support packet coalescence at two levels: (*i*) `Initial` flags are sent separately, leading to two UDP datagrams. The first containing the `ACK` and the second the `ServerHello` flag, both of which are padded resulting in 2462 extra bytes, although only the latter elicits `ACKs` and thus requires padding. (*ii*) We do not observe any coalescence of `Initial` and `Handshake` messages. The extra bytes account for ≈60% of the limit but are (incorrectly) not considered during amplification limit checks. We report this implementation shortcoming to Cloudflare.

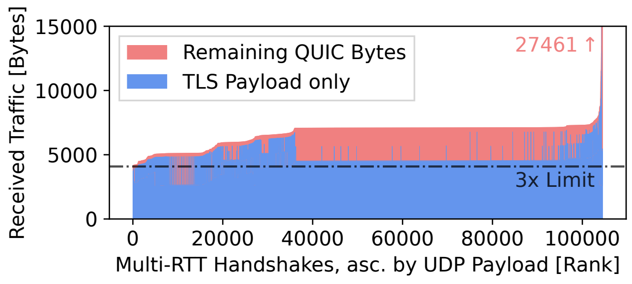


Figure 5: Payload exchanged during multi RTT handshakes. TLS bytes almost always exceed the limit but also QUIC padding can have a significant impact.

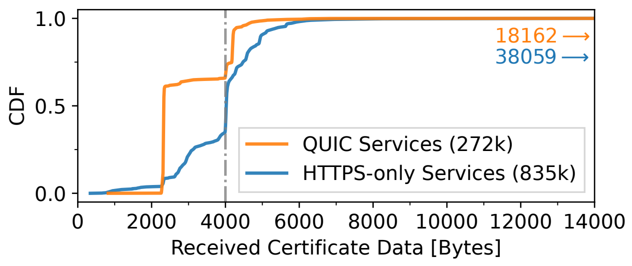


Figure 6: Distribution of certificate sizes grouped by QUIC support. QUIC domains use smaller certificates.

`Retry`. We observe ≈200 services that predominantly request a `Retry` to authenticate client addresses. We conclude that always-on DDoS mitigation is currently not widely adopted, however, `Retrys` might also be triggered adaptively based on the current server load.

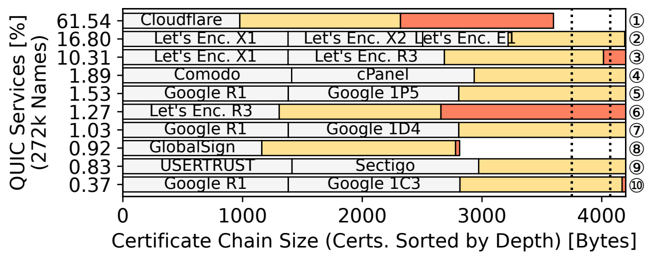
Multi-RTT (no `Retry`). Due to rare deployment of *always-on* `Retry` messages, we assume that multi-RTT handshakes are caused by other factors. We analyze these factors, *i.e.,* TLS certificates, in more detail in the next section.

4.2 Impact of TLS Certificates

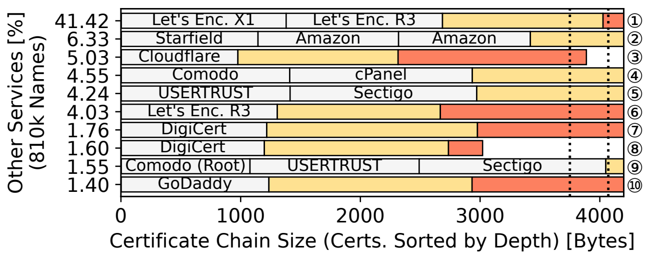
We presume that TLS certificate data causes multi-RTT handshakes. To verify our assumption, we divide the bytes exchanged during a handshake into TLS payload and QUIC-related payload, *e.g.,* QUIC header and padding.

In the majority of cases (87%), TLS payloads alone exceed the amplification limit (see Figure 5). The distribution of (uncompressed) certificate chain sizes exchanged over TLS is shown in Figure 6. Overall, we observe a median of 2329 bytes for certificate chains delivered by QUIC domains compared to 4022 bytes for other names. We find that 35% of all certificate chains exceed even the larger of the two common amplification limits (3·1357 bytes). This means that domains without QUIC support will be affected negatively when they decide to support QUIC in the future and continue to use existing certificates.

Popular parent certificates for QUIC unveil consolidation. By zooming into certificate chains, we examine how the choice



(a) QUIC services



(b) Only HTTPS services

Figure 7: Certificate chain sizes, depths and their dependency. 🟨 represents median leaf size, and 🟥 the additional bytes required for the maximum leaf size. Dotted lines represent the max allowed reply sizes of a server given common client Initial sizes. The x-axis is cut off at 4200 bytes. Average sized certificate chains are likely to exceed QUIC amplification limits.

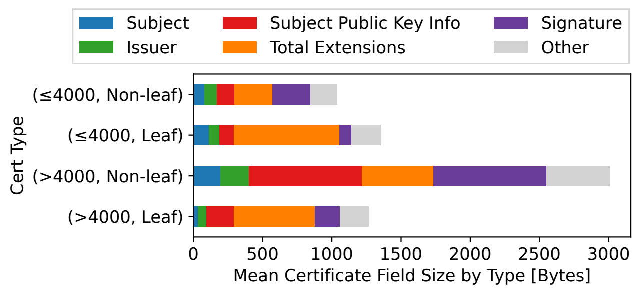


Figure 8: Mean sizes of certificate fields for QUIC domains. Non-leafs contribute most bytes to large chains.

of a specific CA can impact the size of the certificate chain that a service provider needs to deploy. For this analysis, we exclude certificate chains that are not ordered correctly. Figure 7 exhibits the top-10 certificate chains deployed. Each white box represents the sizes of the certificates in the chain (excluding leaf certificates), yellow boxes (🟨) and orange boxes (🟥) represent the median sizes and the largest leaf certificate that we observed in that chain.

Overall, we find that 7 out of 10 parent chains, together with the median leaf size, exceed common amplification limits (5 out of 10 for HTTPS-only services).

For both QUIC and non-QUIC services the shortest chains, *i.e.,* the smallest number of intermediates, are issued by Cloudflare followed by Let's Encrypt R3, GlobalSign, DigiCert, and GoDaddy. We also observe cases in which cross-signed certificates are redundantly included in chains while the self-signed version of the same public key is already included in client trust stores. For example, row ② and ③ in Figure 7(a) include the cross signed version [10, 30] of `ISRG Root X1` (signed by `DST Root CA X3`) instead of relying on the self-signed variant [10, 31], as in row ⑥. In other cases, servers superfluously include trust anchors (*i.e.,* root) certificates (*e.g.,* row ⑨ in Figure 7(b)).

Furthermore, a high consolidation trend for QUIC services is visible, as the top-10 parent chains cover 96.5% of QUIC services. For HTTPS-only services, this trend is less pronounced with only 72% of services. Consequently, to improve the deployment of QUIC services, optimizing the parent chains can have a significant, beneficial effect but only needs to involve a small number of stakeholders.

Table 2: Relative ratio of crypto algorithms and key lengths [bits] in use (limited to types with a frequency of $>$ 1%). HTTPS-only domains depend heavily on RSA.

		RSA		ECDSA	
Service	Certificate	2048	4096	256	384
QUIC	Non-leaf	15.1%	22.4%	40.4%	22.1%
	Leaf	19.2%	1.4%	78.9%	0.0%
HTTPS-only	Non-leaf	63.3%	32.1%	2.7%	1.6%
	Leaf	81.4%	8.1%	7.8%	1.9%

Certificates delivered by QUIC servers tend to use more efficient crypto algorithms, though, compared to non-QUIC Web services (see Table 2).

Non-leaf certificates bring the heavy load. We find very large certificate chains requiring transmissions between 18k and 38k bytes, indicated by the long tail above 4000 bytes in Figure 6. We proceed to use this value as a threshold to classify certificate chains.

Figure 8 depicts the mean size of various TLS certificate fields divided into leaf and non-leaf certificates. We observe that for large chains the sum of public key and signature sections on non-leaf certificates has the biggest impact on the chain size. This again shows the negative effects of selecting a large non-leaf parent chain, even if the related leaf certificate has a reasonable size.

We also find that large cruise-liner leaf certificates [3] are rarely used in QUIC deployments, details see Appendix E.

Compression helps. Compressing certificate chains can avoid exceeding anti-amplification limits and thus improve the situation in the future. Our synthetic experiment of compressing collected certificate chains shows a median compression rate of ≈65%. This keeps the size under the amplification limits for 99% of TLS chains, which in turn prevents multi-RTT QUIC handshakes.

We find that 96% of QUIC services currently support the brotli algorithm, which is used by Chromium derivatives. The support of multiple algorithms, however, is very rare with only 0.05% of QUIC services offering all three. These services relate to Meta.

The mean compression ratio in the wild is 73%, which is close to our synthetic experiments. Here, 99% of all compressed certificates fit below a common anti-amplification limit ($3 \cdot 1357$ bytes).

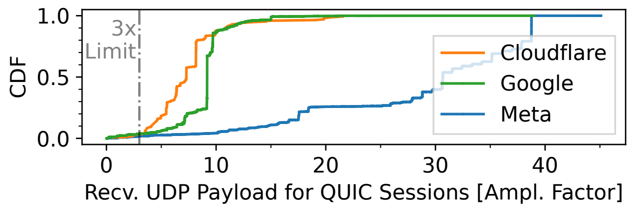


Figure 9: QUIC amplification factors including resends when clients do not respond (*e.g.,* due to spoofing). Amplification increases drastically.

4.3 Examining Amplification Potential

Our previous analysis considered the behavior of servers when handshakes complete successfully. Now, we consider the case when a client fails to send an ACK to a server response. This would cause a resend of data by the server. Since the client IP address is not verified all resends must comply in sum with the 3× amplification limit. A resend occurs, for example, when malicious actors initiate a handshake with a spoofed IP address. All resends, *i.e.,* amplified traffic, are reflected to the spoofed address belonging to a victim.

Figure 9 depicts amplification factors of handshakes collected at our telescope vantage point. We sum all bytes received from a server for a specific SCID, and divide by an assumed client `Initial` of 1362 bytes. All hypergiants exceed the amplification limit due to resends. The majority of Cloudflare and Google backscatter traffic remains below factors of 10×. Worryingly, traffic from Meta servers lead to amplification factors of up to 45×. As a crosscheck, we inspect the duration of backscatter sessions for Meta. We find a median of ~51s and a maximum of 206s. This indicates that the amplified traffic is received within a short time frame and the observed amplification factors are not biased by *e.g.,* reused, overlapping SCIDs.

To confirm that Meta servers do not comply with the current QUIC specification [23], we conduct active scans as follows. We send a single QUIC `Initial` but do not acknowledge the response. We focus on the `/24` subnet of a Meta point-of-presence and identify three groups of IP addresses:

(1) No response or ≤150 bytes, due to no QUIC HTTP3 service.
(2) Responses of ≈7k bytes, which corresponds to an amplification factor of over 5×. IP addresses that typically serve `facebook.com` (`*.35, *.36`) belong to this group.
(3) Responses of ≈35k bytes, which corresponds to an amplification factor of over 28×. This amplification factor is similar to what can be achieved using popular amplification protocols [41]. We find IP addresses that relate to Instagram and WhatsApp (`*.60, *.63`) belonging to this group.

Overall, our active scans confirm the telescope observations. Current deployments of Metas QUIC implementation `mvfst` [16] do not respect the 3× limit in case of resends. Those deployments can be misused as amplifiers in attacks.

5 DISCUSSION & GUIDANCE

Should the QUIC protocol specification be updated? Our results suggest that the QUIC anti-amplification limit specified in RFC 9000 [23] is indeed tight but large enough to achieve 1-RTT handshakes. The limit does not need to be increased to foster better deployments when network conditions are reliable. In the case of packet loss and necessary resends, the anti-amplification limit challenges performance, though. It allows for at most one single retransmission of all server `Initial` and `Handshake` messages, given current certificate deployments including small ECDSA certificate chains and certificate compression. Dealing efficiently with loss of messages during the connection setup seems an open challenge.

Next to protocol design challenges, we also find non-standard QUIC implementations that amplify during the 1-RTT handshake and increase significantly for incomplete QUIC handshakes. More comprehensive testing of QUIC implementations is clearly needed.

Does certificate compression help? We found that certificate compression is an impactful extension to allow servers staying below the amplification limit. Unfortunately, popular TLS implementations such as OpenSSL do not support certificate compression. Given that recent QUIC implementations (*e.g.,* Microsoft QUIC) depend on existing TLS libraries, compression may remain in far reach and alternate measures are required to improve the situation.

Can a QUIC client mitigate lack of compression? To be independent of certificate compression, a QUIC client could maintain a cache that includes certificate sizes of servers that the client frequently requests. For entries in the cache, the client can then adapt the size of `Initial` requests to comply with the anti-amplification limit of the servers and achieve low latency connection setup.

Guidance for certificate authorities. We argue that carefully created TLS certificates and certificate chains can positively influence the QUIC protocol performance. ECDSA certificates lead to substantially smaller certificates chains. They can, however, not unfold their potential because especially root certificates are secured by RSA algorithms. Our results show that updating these certificates can have beneficial cascading effects.

Guidance for QUIC implementations. We infer the following guidelines when implementing QUIC network stacks: First, at the server side implementation, bytes that result from padding or `Resend` must be included in anti-amplification limit checks. Second, enabling packet coalescence at the server is recommended to omit padding and thus free space for TLS certificates reducing the need for additional round trips. However, this can increase the latency when large-scale deployments deliver certificates by servers others than those providing content. Third, we recommend the integration of a TLS library that supports compression to compensate large TLS certificates, which currently trigger multi-RTT handshakes.

6 CONCLUSION AND OUTLOOK

In this paper, we measured and analyzed the QUIC handshake processes in the wild and found that the current Web certificate ecosystem challenges the QUIC design objective of a 1-RTT quick connection setup at low amplification potential. As a consequence, large portions of QUIC connection setups are either multi-RTT, do not comply to the amplification limit, or both. Future work shall closely monitor the evolution of the QUIC ecosystem and analyze the impact of measures to reduce certificate sizes effectively.

Responses from Hypergiants. We contacted Meta as well as Cloudflare. Details about our responsible disclosure policy are explained in Appendix B.

ACKNOWLEDGMENTS

We would like to thank our shepherd Olivier Bonaventure and the anonymous reviewers for their detailed and helpful feedback. We gratefully acknowledge the wisdom of Gorry Fairhurst when discussing IETF magic. The assistance provided by Nicholas Banks during early stages of QUIC handshake classification is greatly appreciated. This work was supported in parts by the German Federal Ministry of Education and Research (BMBF) within the projects Deutsches Internet-Institut (grant no. 16DII111) and PRIMEnet.

REFERENCES

[1] D. Eastlake 3rd. 2013. *Domain Name System (DNS) IANA Considerations.* RFC 6895. IETF.

[2] Prasenjeet Biswal and Omprakash Gnawali. 2016. Does QUIC Make the Web Faster?. In *Proceedings of IEEE Global Communications Conference (GLOBECOM '16).* IEEE Press, Piscataway, NJ, USA, 6 pages.

[3] Frank Cangialosi, Taejoong Chung, David Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. 2016. Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem. In *Proc. of ACM SIGSAC CCS* (Vienna, Austria) *(CCS '16).* ACM, New York, NY, USA, 628–640.

[4] Gaetano Carlucci, Luca De Cicco, and Saverio Mascolo. 2015. HTTP over UDP: An Experimental Investigation of QUIC. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (Salamanca, Spain) *(SAC '15).* ACM, New York, NY, USA, 609–614.

[5] Sarah Cook, Bertrand Mathieu, Patrick Truong, and Isabelle Hamchaoui. 2017. QUIC: Better for what and for whom?. In *Proceedings of IEEE International Conference on Communications (ICC '17).* IEEE Press, Piscataway, NJ, USA, 6 pages.

[6] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. 2008. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* RFC 5280. IETF.

[7] Yong Cui, Tianxiang Li, Cong Liu, Xingwei Wang, and Mirja Kühlewind. 2017. Innovating Transport with QUIC: Design Approaches and Research Challenges. *IEEE Internet Computing* 21, 2 (2017), 72–76.

[8] Piet De Vaere, Tobias Bühler, Mirja Kühlewind, and Brian Trammell. 2018. Three Bits Suffice: Explicit Support for Passive Measurement of Internet Latency in QUIC and TCP. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18).* ACM, New York, NY, USA, 22–28.

[9] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. 2013. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *22nd USENIX Security Symposium (USENIX Security '13).* USENIX Association, Washington, D.C., 605–620.

[10] Let's Encrypt. 2021. Let's Encrypt's Hierarchy as of August 2021. Website. `https://letsencrypt.org/certificates/` Last Access: Oct 2022.

[11] Eva Gagliardi and Olivier Levillain. 2020. Analysis of QUIC Session Establishment and Its Implementations. In *Information Security Theory and Practice.* LNCS, Vol. 12024. Springer Nature, Switzerland, 169–184.

[12] A. Ghedini and V. Vasiliev. 2020. *TLS Certificate Compression.* RFC 8879. IETF.

[13] Google. 2021. Quiche. Commit: Internal change. Open-source repository. `https://github.com/google/quiche/commit/36d9a1fbff6e0f8665a1c60c09e19aa38380ae85`

[14] HAW. 2022. Quiche fork with compression. Open-source repository. `https://github.com/josephnoir/quiche`

[15] Ralph Holz, Jens Hiller, Johanna Amann, Abbas Razaghpanah, Thomas Jost, Narseo Vallina-Rodriguez, and Oliver Hohlfeld. 2020. Tracking the Deployment of TLS 1.3 on the Web: A Story of Experimentation and Centralization. *SIGCOMM Computer Communication Review* 50, 3 (July 2020), 3–15.

[16] Facebook Incubator. 2019. mvfst – An implementation of the QUIC transport protocol. GitHub Repository. `https://github.com/facebookincubator/mvfst/` Last Access: Oct 2022.

[17] J. Iyengar and I. Swett. 2021. *QUIC Loss Detection and Congestion Control, Section 6.2.2.1, Before Address Validation.* RFC 9002. IETF.

[18] Janardhan Iyengar and Martin Thomson. 2017. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft – work in progress 01. IETF.

[19] Janardhan Iyengar and Martin Thomson. 2017. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft – work in progress 02. IETF.

[20] Jana Iyengar and Martin Thomson. 2018. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft – work in progress 09. IETF.

[21] Jana Iyengar and Martin Thomson. 2018. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft – work in progress 10. IETF.

[22] Jana Iyengar and Martin Thomson. 2018. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft – work in progress 15. IETF.

[23] J. Iyengar and M. Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport.* RFC 9000. IETF.

[24] Jana Iyengar and Martin Thomson. 2021. *QUIC: A UDP-Based Multiplexed and Secure Transport.* Internet-Draft – work in progress 33. IETF.

[25] Arash Molavi Kakhki, Samuel Jero, David Choffnes, Cristina Nita-Rotaru, and Alan Mislove. 2017. Taking a Long Look at QUIC: An Approach for Rigorous Evaluation of Rapidly Evolving Transport Protocols. In *Proceedings of the 2017 Internet Measurement Conference* (London, United Kingdom) *(IMC '17).* ACM, New York, NY, USA, 290–303.

[26] Mike Kosek, Trinh Viet Doan, Malte Granderath, and Vaibhav Bajpai. 2022. One to Rule Them All? A First Look at DNS over QUIC. In *Proceedings of PAM.* Springer International Publishing, Cham, 537–551.

[27] Mike Kosek, Luca Schumann, Robin Marx, Trinh Viet Doan, and Vaibhav Bajpai. 2022. DNS Privacy with Speed? Evaluating DNS over QUIC and its Impact on Web Performance. In *Proceedings of ACM IMC* (Nice, France) *(IMC '22).* ACM, New York, NY, USA. Pre-print.

[28] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17).* ACM, New York, NY, USA, 183–196.

[29] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS '19).* The Internet Society, San Diego, CA, USA.

[30] Sectigo Limited. 2015. Certificate Search, ID 3958242236. Website. `https://crt.sh/?id=3958242236` Last Access: Oct 2022.

[31] Sectigo Limited. 2015. Certificate Search, ID 9314791. Website. `https://crt.sh/?id=9314791` Last Access: Oct 2022.

[32] Robert Lychev, Samuel Jero, Alexandra Boldyreva, and Cristina Nita-Rotaru. 2015. How Secure and Quick is QUIC? Provable Security and Performance Analyses. In *Proceedings of IEEE Symposium on Security and Privacy.* IEEE Press, Piscataway, NJ, USA, 214–231.

[33] Diego Madariaga, Lucas Torrealba, Javier Madariaga, Javiera Bermúdez, and Javier Bustos-Jiménez. 2020. Analyzing the Adoption of QUIC From a Mobile Development Perspective. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC* (Virtual Event, USA) *(EPIQ '20).* ACM, New York, NY, USA, 35–41.

[34] John Mattsson. 2021. Background on the 3x anti-amplification limit. IETF Mail Archive. `https://mailarchive.ietf.org/arch/msg/quic/RdeQ_y4dHLzufgtXYccFPBiqrUQ/` Last Access: Oct 2022.

[35] Patrick McManus. 2020. Does the QUIC handshake require compression to be fast? Fastly Blog. `https://www.fastly.com/blog/quic-handshake-tls-compression-certificates-extension-study` Last Access May 2021.

[36] Microsoft. 2022. quicreach. Open-source repository. `https://github.com/microsoft/quicreach/`

[37] P.V. Mockapetris. 1987. *Domain names - implementation and specification.* RFC 1035. IETF.

[38] Jonas Mücke, Marcin Nawrocki, Raphael Hiesgen, Patrick Sattler, Johannes Zirngibl, Georg Carle, Thomas C. Schmidt, and Matthias Wählisch. 2022. *Waiting for QUIC: On the Opportunities of Passive Measurements to Understand QUIC Deployments.* Technical Report arXiv:2209.00965. Open Archive: arXiv.org. `https://arxiv.org/abs/2209.00965`

[39] Marcin Nawrocki, Raphael Hiesgen, Thomas C. Schmidt, and Matthias Wählisch. 2021. QUICsand: Quantifying QUIC Reconnaissance Scans and DoS Flooding Events. In *Proceedings of the 21st ACM Internet Measurement Conference* (Virtual Event) *(IMC '21).* ACM, New York, NY, USA, 283–291.

[40] E. Rescorla. 2018. *The Transport Layer Security (TLS) Protocol Version 1.3.* RFC 8446. IETF.

[41] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *Proceedings of NDSS.* Internet Society, 15 pages.

[42] Jan Rüth, Ingmar Poese, Christoph Dietzel, and Oliver Hohlfeld. 2018. A First Look at QUIC in the Wild. In *Passive and Active Measurement (PAM 2018).* Springer International Publishing, Cham, 255–268.

[43] Fabrice J. Ryba, Matthew Orlinski, Matthias Wählisch, Christian Rossow, and Thomas C. Schmidt. 2015. *Amplification and DRDoS Attack Defense – A Survey and New Perspectives.* Technical Report arXiv:1505.07892. Open Archive: arXiv.org. `http://arxiv.org/abs/1505.07892`

[44] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. 2018. A Long Way to the Top: Significance, Structure, and Stability of Internet Top Lists. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18).* ACM, New York, NY, USA, 478–493.

[45] Tanya Shreedhar, Rohit Panda, Sergey Podanev, and Vaibhav Bajpai. 2022. Evaluating QUIC Performance Over Web, Cloud Storage, and Video Workloads. *IEEE Transactions on Network and Service Management* 19, 2 (2022), 1366–1381.

[46] Jean-Pierre Smith, Prateek Mittal, and Adrian Perrig. 2021. Website Fingerprinting in the Age of QUIC. *Proceedings on Privacy Enhancing Technologies* 2021, 2 (2021), 48–69.

[47] Martin Thomson. 2018. Space for Packet Metadata. IETF Mail Archive. `https://mailarchive.ietf.org/arch/msg/quic/wzrNfxRCwwgfw499Dh0YsHYTSoI/` Last Access: Oct 2022.

[48] M. Thomson and S. Turner. 2021. *Using TLS to Secure QUIC.* RFC 9001. IETF.

[49] Martino Trevisan, Danilo Giordano, Idilio Drago, Maurizio Matteo Munafò, and Marco Mellia. 2020. Five Years at the Edge: Watching Internet From the ISP Network. *IEEE/ACM Transactions on Networking* 28, 2 (2020), 561–574.

[50] TUM. 2021. QScanner. Open-source repository. `https://github.com/tumi8/QScanner`

[51] Konrad Wolsing, Jan Rüth, Klaus Wehrle, and Oliver Hohlfeld. 2019. A Performance Perspective on Web Optimized Protocol Stacks: TCP+TLS+HTTP/2 vs. QUIC. In *Proceedings of the Applied Networking Research Workshop* (Montreal, QC, Canada) *(ANRW '19)*. ACM, New York, NY, USA, 1–7.

[52] Johannes Zirngibl, Philippe Buschmann, Patrick Sattler, Benedikt Jaeger, Juliane Aulbach, and Georg Carle. 2021. It's over 9000: Analyzing Early QUIC Deployments with the Standardization on the Horizon. In *Proceedings of the 21st ACM Internet Measurement Conference* (Virtual Event) *(IMC '21)*. ACM, New York, NY, USA, 261–275.

A ARTIFACT APPENDIX

A.1 Abstract

This section gives a brief overview of the artifacts of this paper. We contribute tools to conduct follow-up measurements as well as raw data and analysis scripts to reproduce the results and figures presented in this paper.

A.2 Artifact check-list (meta-information)

Data set: QUIC handshakes, TLS certificates.
Run-time environment: Python, Jupyter Notebooks.
Output: All paper figures.
How much disk space required? 500MB
Time needed to prepare workflow? 10 minutes
Time needed to complete experiments? 20 minutes
Publicly available? Yes
Archived? Yes: `https://doi.org/10.5281/zenodo.7157904`

A.3 Description

A.3.1 How to access. All artifacts are available via the following public repository:

`https://github.com/ilabrg/artifacts-conext22-quic-tls`

This public repository provides up-to-date instructions for installing, configuring, and running our artifacts. We also archive the camera-ready version of our software on Zenodo:

`https://doi.org/10.5281/zenodo.7157904`

A.3.2 Overview. We present an overview of our toolchain and related data flow in Figure 10.

A.3.3 Software dependencies. The minimal requirements to reproduce our figures are Python with Jupyter Notebook, Pandas, Matplotlib, Seaborn, and Zstd. For a full list of dependencies, see the `requirements.md` file in the repository.

A.3.4 Network dependencies. We add all data to reproduce our results. If you use the open-source tools to collect fresh data, your network has to allow fast connection setups on the default QUIC and HTTPS ports, *i.e.,* UDP/443 and TCP/443.

A.3.5 Data sets. We contribute two main data sets: (*i*) TLS certificate information of all 1M Tranco domains and related redirects and (*ii*) based on this list, QUIC handshakes to services with TLS certificates with varying `INITIAL` sizes. Supporting data sets are documented in the repository.

A.4 Installation

Clone the artifacts repository, follow the `README` instructions to install the requirements, and then run Jupyter Notebook. We provide different notebooks for different parts of the evaluation of the collected data. No further installation is required.

A.5 Evaluation and Expected Results

We now explain the structure of the repository and which Python scripts should be run in which order.

`code/` Contains all analysis notebooks to reproduce our figures. As a preparatory step, run `01-Prepare-Dataframes`, which parses and sanitizes the raw CSV data and prepares dataframes in compressed pickle format for fast reading. Thereafter, you can execute any notebook in arbitrary order.

`code/plots` Contains all figures in PDF and PNG format. These files will be overwritten when rerunning the analyses.

`data/csv` Raw, compressed CSV files from our active scans. Large files have been split into multiple chunks to avoid running into file limits at *e.g.,* Github.

`data/pkl/` Sanitized dataframes in compressed, binary format. One file per dataframe. These files will be created locally by running notebook `01-Prepare-Dataframes`.

`misc/` Collection of open-source tools that we used to scan the Internet and to create the raw CSV files. Some of these tools have been extended by us. We contributed the extensions upstream but our pull request is not merged into the main branch of the third-party tools, yet.

B ETHICAL CONCERNS

This work may raise the following ethical concerns.

Educating attackers. We discovered deployment behavior that conflicts with DDoS mitigation required by RFC 9000 and hence enables misuse. We follow a responsible disclosure policy and aim for fixing the bugs in collaboration with Cloudflare and Meta.

Responses from hypergiants. Meta significantly improved their QUIC deployment in October 2022. By rescanning all host addresses in `/24` on-net prefixes, we now observe homogeneously configured servers that limit the amount of QUIC retransmissions in case of unverified clients. However, with a mean amplification factor of 5×, the responses still exceed the anti-amplification limit specified in RFC 9000. We show the results in Figure 11, including 95% confidence intervals.

Cloudflare has responded, and explains the reason to exceed the limit is to help improve client performance, while respecting production constraints that are omitted from the QUIC specification. Specifically, in production environments the information needed to populate the `ServerHello` is contained in certificates that may be managed separately from connection termination, and unavailable at the moment of arrival of the client's `Initial`. The delay affects client estimates of RTT. Cloudflare mitigates the delay by immediately responding to client `Initials` with an `ACK` padded at the UDP layer. This occurs once, so the amplification factor is bound.

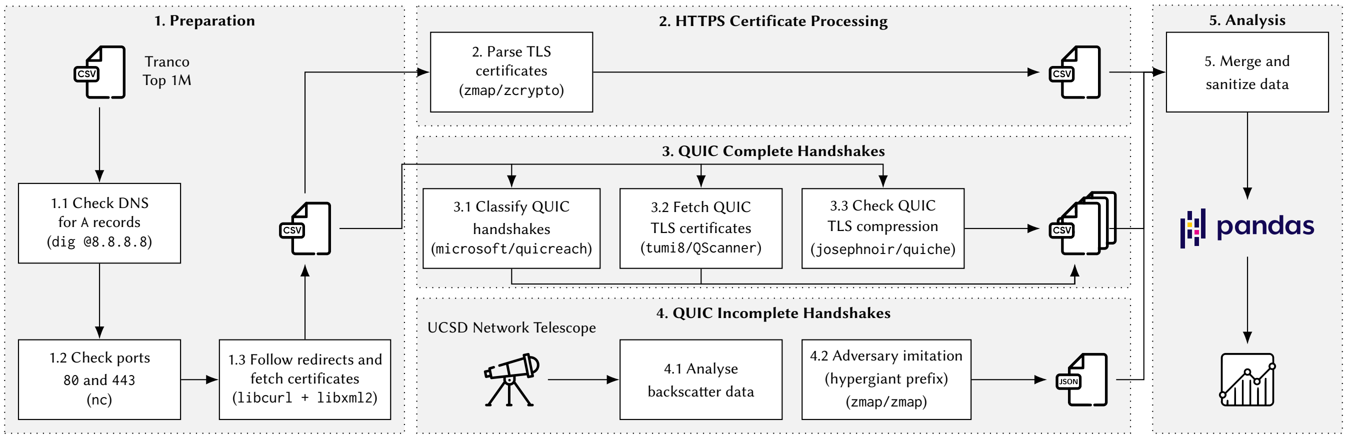
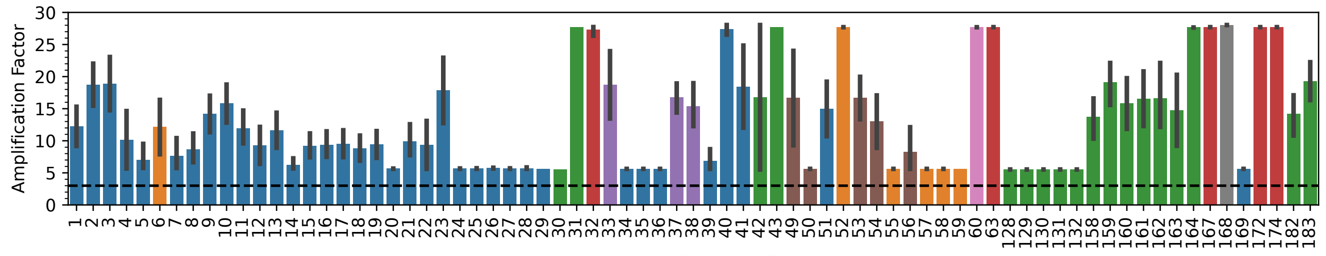
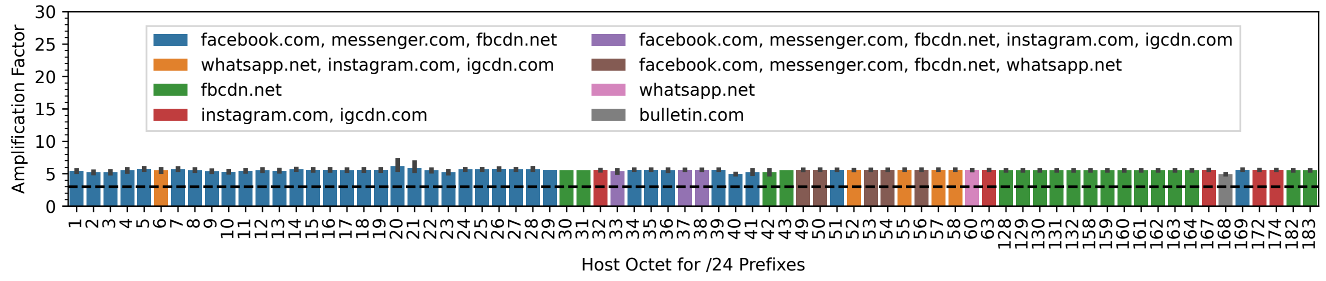

1. Preparation
Tranco Top 1M
1.1 Check DNS for A records (dig @8.8.8.8)
1.2 Check ports 80 and 443 (nc)
1.3 Follow redirects and fetch certificates (libcurl + libxml2)
2. HTTPS Certificate Processing
2. Parse TLS certificates (zmap/zcrypto)
3. QUIC Complete Handshakes
3.1 Classify QUIC handshakes (microsoft/quicreach)
3.2 Fetch QUIC TLS certificates (tumi8/QScanner)
3.3 Check QUIC TLS compression (josephnoir/quiche)
4. QUIC Incomplete Handshakes
UCSD Network Telescope
4.1 Analyse backscatter data
4.2 Adversary imitation (hypergiant prefix) (zmap/zmap)
5. Analysis
5. Merge and sanitize data
pandas

Figure 10: Overview of our HTTPS and QUIC analysis toolchain as well as related data flow.



(a) August 2022 (before disclosure)



(b) October 2022 (after disclosure)

Figure 11: Mean amplification factors for Meta services observed at all point-of-presences. We see a significant improvement after the responsible disclosure of our results. The anti-amplification limit is still slightly above the allowed threshold.

C QUIC ANTI-AMPLIFICATION LIMIT

In Table 3, we show the historical development of QUIC amplification mitigation as proposed in the different versions of the QUIC Internet Draft. Although amplification attacks have already been mentioned in Draft 01 [18], no limitations to reduce the attack potential have been specified for servers. Draft 02 [19], at least, specifies that clients must ensure that the first packet in a connection, *i.e.,* commonly an INITIAL, meets the requirement of minimum packet size. This requirement limits the overall amplification factor since any attacker needs to invest a minimum amount of data.

In Draft 09 [20], the first restriction for servers is introduced. A server may close a connection with an error code in the case of a too small client INITIAL. Otherwise, it must not respond or behave as if any part of the offending packet was processed as valid. In Draft 10 [21], a server is limited by the number of HANDSHAKE packets a server is allowed to send to unverified clients, even though this is not explicitly noted in the context of amplification mitigation. Since Draft 15 [22], the anti-amplification limit is specified relative to the client. Since Draft 33 [24], including the current RFC [23], this limit has been specified to three times of received *data.*

We find little discussion about the limit on the IETF mailing lists. In March 2018, 3600 (= 3 · 1200) bytes have been discussed as "decently large" [47] to carry TLS certificates. A recent question on the exact motivation behind the 3× remains unanswered [34].

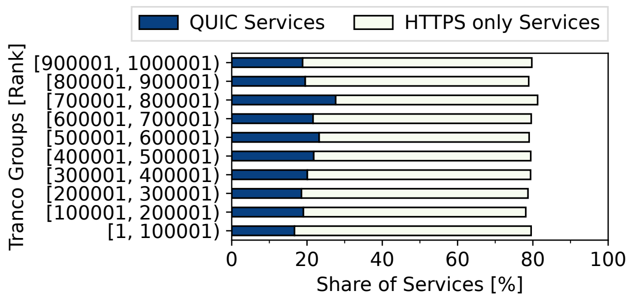
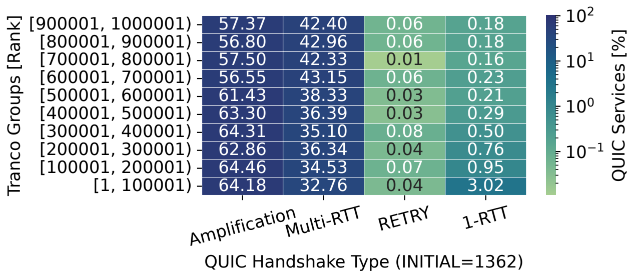


Figure 12: Service popularity across tranco rank groups. QUIC and HTTPS deployment rates are stable across rank groups.



Tranco Groups [Rank]	Amplification	Multi-RTT	RETRY	1-RTT
[900001, 1000001)	57.37	42.40	0.06	0.18
[800001, 900001)	56.80	42.96	0.06	0.18
[700001, 800001)	57.50	42.33	0.01	0.16
[600001, 700001)	56.55	43.15	0.06	0.23
[500001, 600001)	61.43	38.33	0.03	0.21
[400001, 500001)	63.30	36.39	0.03	0.29
[300001, 400001)	64.31	35.10	0.08	0.50
[200001, 300001)	62.86	36.34	0.04	0.76
[100001, 200001)	64.46	34.53	0.07	0.95
[1, 100001)	64.18	32.76	0.04	3.02

Figure 13: QUIC handshake classification per tranco rank group. Handshake types are mostly stable across rank groups.

Table 3: Descriptions of amplification mitgation in the different versions of the IETF QUIC Internet Draft, leading to the 3× anti-amplification limit. [Bold highlighting by us.]

IETF Spec	Date	Proposed Limit
Draft 09	01/2018	"A server MAY send a CONNECTION_CLOSE frame with error code PROTOCOL_VIOLATION in response to an Initial packet smaller than 1200 octets."
Draft 10 – 12	03/2018 – 05/2018	"Servers MUST NOT send more than three **Handshake** packets without receiving a packet from a verified source address."
Draft 13 – 14	06/2018 – 08/2018	"Servers MUST NOT send more than three **datagrams** including Initial and Handshake packets without receiving a packet from a verified source address."
Draft 15 – 32	10/2018 – 10/2020	"Servers MUST NOT send more than three times as many **bytes** as the number of bytes received prior to verifying the client's address."
Draft 33 – 34, RFC 9000	12/2020 – 01/2021, 05/2021	"[...] an endpoint MUST limit the amount of **data** it sends to the unvalidated address to three times the amount of data received from that address."

D INFLUENCE OF TOP LIST RANKS

We verify whether our results depend on some type of popularity of the QUIC-based Web service using the Tranco list [29]. To this end, we split the Tranco list in groups of 100k (ranked) names and initiate QUIC and HTTPS handshakes for each name.

Figure 12 exhibits the relative amount of servers that are reachable via QUIC or only via HTTPS. On average, 21% of domains per rank group are reachable via QUIC. On top of this, ≈ 59% of additional names own a TLS certificate and are reachable over HTTPS. The popularity of a server has no influence on the popularity of QUIC deployment, as we observe a small standard deviation of $\sigma = 3$ across rank groups.

We also check whether the QUIC handshake classification is stable across ranks by mapping responses to a QUIC handshake type (amplification, multi-RTT *etc.*) and counting the relative number of servers per type. Figure 13 visualizes the results. Again, we find no significant differences across rank groups. The only exceptions are 1-RTT handshakes, which appear more popular among the 100k most popular QUIC servers (3.02% vs. <0.95%).

Both analysis indicate that our results are independent of the specific Tranco rank.

E CRUISE-LINER TLS CERTIFICATES

Cruise-liner certificates [3] are certificates that are large in size due to many subject alternate names (SANs). We now check whether QUIC services are affected by cruise-liner certificates. To this end, we analyze all the leaf certificates received for all QUIC services. We inspect the total certificate size and the share of bytes required by all SANs. The results are visualized in Figure 14.

Overall, most SANs amount for less than 10% of bytes. Taking a closer look at the top 1% of certificates by SAN byte share, we find that they require at least 28.9% of bytes (horizontal threshold). Worryingly, 0.1% of certificates exhibit a high SAN byte share and exceed a common QUIC amplification limit (vertical threshold).

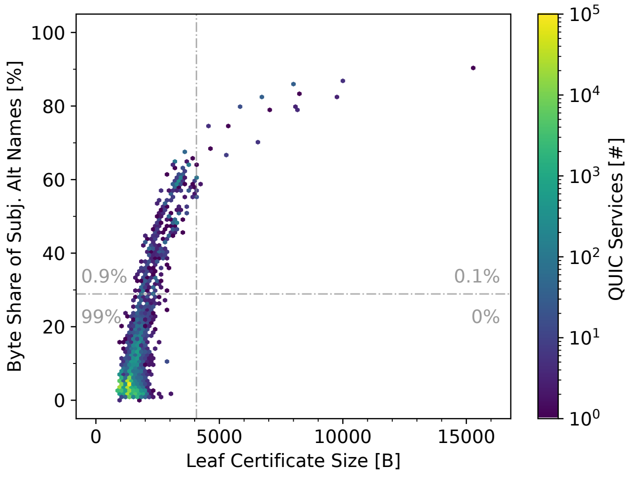


Figure 14: Relative size of subject alternative names (SANs). Cruise-liner certificates are rare for QUIC services.